

Gazebo
1.2.5

Generated by Doxygen 1.8.2

Thu Nov 8 2012 10:16:26

Contents

1	Gazebo API Reference	1
2	Todo List	3
3	Module Index	5
3.1	Modules	5
4	Namespace Index	7
4.1	Namespace List	7
5	Hierarchical Index	9
5.1	Class Hierarchy	9
6	Class Index	15
6.1	Class List	15
7	File Index	25
7.1	File List	25
8	Module Documentation	31
8.1	Common	31
8.1.1	Detailed Description	34
8.1.2	Macro Definition Documentation	34
8.1.2.1	DIAG_TIMER	34
8.1.2.2	gzclr_end	34
8.1.2.3	gzclr_start	34
8.1.2.4	gzdbg	34
8.1.2.5	gzerr	35
8.1.2.6	gzlog	35
8.1.2.7	gzmsg	35
8.1.2.8	gzthrow	35

8.1.2.9	gzwarn	35
8.1.3	Typedef Documentation	36
8.1.3.1	DiagnosticTimerPtr	36
8.1.4	Enumeration Type Documentation	36
8.1.4.1	PluginType	36
8.1.5	Function Documentation	36
8.1.5.1	add_search_path_suffix	36
8.1.5.2	DownloadDependencies	36
8.1.5.3	find_file	36
8.1.5.4	find_file_path	36
8.1.5.5	GetManifest	37
8.1.5.6	GetModelFile	37
8.1.5.7	GetModelName	37
8.1.5.8	GetModelPath	37
8.1.5.9	GetModels	38
8.1.5.10	GetURI	38
8.1.5.11	HasModel	38
8.2	Events	39
8.2.1	Detailed Description	41
8.2.2	Function Documentation	41
8.2.2.1	~EventT	41
8.2.2.2	Connect	42
8.2.2.3	ConnectAddEntity	42
8.2.2.4	ConnectCreateEntity	42
8.2.2.5	ConnectDeleteEntity	43
8.2.2.6	ConnectDiagTimerStart	43
8.2.2.7	ConnectDiagTimerStop	43
8.2.2.8	ConnectPause	44
8.2.2.9	ConnectPostRender	44
8.2.2.10	ConnectPreRender	44
8.2.2.11	ConnectRender	44
8.2.2.12	ConnectSetSelectedEntity	45
8.2.2.13	ConnectStep	45
8.2.2.14	ConnectStop	45
8.2.2.15	ConnectWorldCreated	46
8.2.2.16	ConnectWorldUpdateEnd	46
8.2.2.17	ConnectWorldUpdateStart	46

8.2.2.18	Disconnect	46
8.2.2.19	Disconnect	47
8.2.2.20	DisconnectAddEntity	47
8.2.2.21	DisconnectCreateEntity	47
8.2.2.22	DisconnectDeleteEntity	48
8.2.2.23	DisconnectDiagTimerStart	48
8.2.2.24	DisconnectDiagTimerStop	48
8.2.2.25	DisconnectPause	48
8.2.2.26	DisconnectPostRender	48
8.2.2.27	DisconnectPreRender	49
8.2.2.28	DisconnectRender	49
8.2.2.29	DisconnectSetSelectedEntity	49
8.2.2.30	DisconnectStep	49
8.2.2.31	DisconnectStop	49
8.2.2.32	DisconnectWorldCreated	50
8.2.2.33	DisconnectWorldUpdateEnd	50
8.2.2.34	DisconnectWorldUpdateStart	50
8.2.3	Variable Documentation	50
8.2.3.1	addEntity	50
8.2.3.2	deleteEntity	50
8.2.3.3	diagTimerStart	50
8.2.3.4	diagTimerStop	51
8.2.3.5	entityCreated	51
8.2.3.6	pause	51
8.2.3.7	postRender	51
8.2.3.8	preRender	51
8.2.3.9	render	51
8.2.3.10	setSelectedEntity	51
8.2.3.11	step	51
8.2.3.12	stop	52
8.2.3.13	worldCreated	52
8.2.3.14	worldUpdateEnd	52
8.2.3.15	worldUpdateStart	52
8.3	Math	53
8.3.1	Detailed Description	54
8.3.2	Function Documentation	55
8.3.2.1	clamp	55

8.3.2.2	equal	55
8.3.2.3	isnan	55
8.3.2.4	isnan	55
8.3.2.5	isPowerOfTwo	56
8.3.2.6	max	56
8.3.2.7	mean	56
8.3.2.8	min	56
8.3.2.9	parseFloat	57
8.3.2.10	parseInt	57
8.3.2.11	precision	57
8.3.2.12	variance	58
8.3.3	Variable Documentation	58
8.3.3.1	NAN_D	58
8.3.3.2	NAN_I	58
8.4	Messages	59
8.4.1	Detailed Description	60
8.4.2	Function Documentation	60
8.4.2.1	Convert	60
8.4.2.2	Convert	61
8.4.2.3	Convert	61
8.4.2.4	Convert	61
8.4.2.5	Convert	61
8.4.2.6	Convert	62
8.4.2.7	Convert	62
8.4.2.8	Convert	62
8.4.2.9	Convert	62
8.4.2.10	Convert	63
8.4.2.11	Convert	63
8.4.2.12	Convert	63
8.4.2.13	CreateRequest	63
8.4.2.14	FogFromSDF	64
8.4.2.15	GetHeader	64
8.4.2.16	GUIFromSDF	64
8.4.2.17	Init	64
8.4.2.18	LightFromSDF	65
8.4.2.19	SceneFromSDF	65
8.4.2.20	Set	65

8.4.2.21	Set	65
8.4.2.22	Set	65
8.4.2.23	Set	66
8.4.2.24	Set	66
8.4.2.25	Set	66
8.4.2.26	Set	66
8.4.2.27	Set	66
8.4.2.28	Set	67
8.4.2.29	Stamp	67
8.4.2.30	Stamp	67
8.4.2.31	TrackVisualFromSDF	67
8.4.2.32	VisualFromSDF	67
8.5	Classes for physics and dynamics	68
8.5.1	Detailed Description	71
8.5.2	Macro Definition Documentation	71
8.5.2.1	GZ_REGISTER_PHYSICS_ENGINE	71
8.5.3	Typedef Documentation	72
8.5.3.1	PhysicsFactoryFn	72
8.5.4	Function Documentation	72
8.5.4.1	create_world	72
8.5.4.2	fini	72
8.5.4.3	get_world	72
8.5.4.4	init_world	72
8.5.4.5	init_worlds	73
8.5.4.6	load	73
8.5.4.7	load_world	73
8.5.4.8	load_worlds	73
8.5.4.9	pause_world	73
8.5.4.10	pause_worlds	73
8.5.4.11	remove_worlds	73
8.5.4.12	run_world	74
8.5.4.13	run_worlds	74
8.5.4.14	stop_world	74
8.5.4.15	stop_worlds	74
8.5.5	Variable Documentation	74
8.5.5.1	EntityTypename	74
8.6	Bullet Physics	75

8.6.1	Detailed Description	76
8.7	ODE Physics	77
8.7.1	Detailed Description	78
8.7.2	Typedef Documentation	78
8.7.2.1	ODEPhysicsPtr	78
8.7.2.2	ODESurfaceParamsPtr	78
8.8	Rendering	79
8.8.1	Detailed Description	81
8.8.2	Function Documentation	81
8.8.2.1	create_scene	81
8.8.2.2	fini	81
8.8.2.3	get_scene	81
8.8.2.4	init	81
8.8.2.5	load	81
8.8.2.6	remove_scene	81
8.9	Gazebo_parser	82
8.9.1	Detailed Description	82
8.9.2	Typedef Documentation	82
8.9.2.1	CollisionPtr	82
8.9.2.2	VisualPtr	82
8.10	Sensors	83
8.10.1	Detailed Description	84
8.10.2	Macro Definition Documentation	85
8.10.2.1	GZ_REGISTER_STATIC_SENSOR	85
8.10.3	Function Documentation	85
8.10.3.1	create_sensor	85
8.10.3.2	fini	85
8.10.3.3	get_sensor	85
8.10.3.4	init	86
8.10.3.5	load	86
8.10.3.6	remove_sensor	86
8.10.3.7	remove_sensors	86
8.10.3.8	run	86
8.10.3.9	run_once	86
8.10.3.10	stop	87
8.11	Transport	88
8.11.1	Detailed Description	89

8.11.2	Typedef Documentation	89
8.11.2.1	CallbackHelperPtr	89
8.11.3	Function Documentation	89
8.11.3.1	clear_buffers	89
8.11.3.2	fini	90
8.11.3.3	get_master_uri	90
8.11.3.4	get_topic_namespaces	90
8.11.3.5	init	90
8.11.3.6	is_stopped	90
8.11.3.7	pause_incoming	90
8.11.3.8	request	90
8.11.3.9	run	91
8.11.3.10	stop	91
9	Namespace Documentation	93
9.1	boost Namespace Reference	93
9.2	gazebo Namespace Reference	93
9.2.1	Detailed Description	94
9.2.2	Typedef Documentation	95
9.2.2.1	GUIPluginPtr	95
9.2.2.2	ModelPluginPtr	95
9.2.2.3	SensorPluginPtr	95
9.2.2.4	SystemPluginPtr	95
9.2.2.5	VisualPluginPtr	95
9.2.2.6	WorldPluginPtr	95
9.2.3	Function Documentation	95
9.2.3.1	add_plugin	95
9.2.3.2	find_file	95
9.2.3.3	fini	95
9.2.3.4	init	95
9.2.3.5	load	95
9.2.3.6	print_version	95
9.2.3.7	run	95
9.2.3.8	stop	95
9.3	gazebo::common Namespace Reference	95
9.3.1	Detailed Description	97
9.3.2	Typedef Documentation	98

9.3.2.1	AnimationPtr	98
9.3.2.2	NodeMap	98
9.3.2.3	NodeMapIter	98
9.3.2.4	NumericAnimationPtr	98
9.3.2.5	Param_V	98
9.3.2.6	PoseAnimationPtr	98
9.3.2.7	RawNodeAnim	98
9.3.2.8	RawNodeWeights	98
9.3.2.9	RawSkeletonAnim	98
9.3.2.10	StrStr_M	98
9.4	gazebo::event Namespace Reference	98
9.4.1	Detailed Description	98
9.4.2	Typedef Documentation	99
9.4.2.1	Connection_V	99
9.4.2.2	ConnectionPtr	99
9.5	gazebo::math Namespace Reference	99
9.5.1	Detailed Description	101
9.5.2	Typedef Documentation	101
9.5.2.1	GeneratorType	101
9.5.2.2	NormalRealDist	101
9.5.2.3	NRealGen	101
9.5.2.4	UIntGen	101
9.5.2.5	UniformIntDist	101
9.5.2.6	UniformRealDist	101
9.5.2.7	URealGen	101
9.6	gazebo::msgs Namespace Reference	101
9.6.1	Detailed Description	103
9.7	gazebo::physics Namespace Reference	103
9.7.1	Detailed Description	108
9.7.2	Typedef Documentation	108
9.7.2.1	Actor_V	108
9.7.2.2	ActorPtr	108
9.7.2.3	Base_V	108
9.7.2.4	BasePtr	108
9.7.2.5	BoxShapePtr	108
9.7.2.6	BulletCollisionPtr	108
9.7.2.7	BulletLinkPtr	108

9.7.2.8	BulletPhysicsPtr	108
9.7.2.9	BulletRayShapePtr	108
9.7.2.10	Collision_V	108
9.7.2.11	CollisionPtr	108
9.7.2.12	ContactPtr	108
9.7.2.13	CylinderShapePtr	108
9.7.2.14	EntityPtr	109
9.7.2.15	HeightmapShapePtr	109
9.7.2.16	InertialPtr	109
9.7.2.17	Joint_V	109
9.7.2.18	JointPtr	109
9.7.2.19	Link_V	109
9.7.2.20	LinkPtr	109
9.7.2.21	MeshShapePtr	109
9.7.2.22	Model_V	109
9.7.2.23	ModelPtr	109
9.7.2.24	MultiRayShapePtr	109
9.7.2.25	ODECollisionPtr	109
9.7.2.26	ODELinkPtr	109
9.7.2.27	ODERayShapePtr	109
9.7.2.28	PhysicsEnginePtr	109
9.7.2.29	RayShapePtr	109
9.7.2.30	RoadPtr	109
9.7.2.31	ShapePtr	109
9.7.2.32	SphereShapePtr	109
9.7.2.33	SurfaceParamsPtr	109
9.7.2.34	WorldPtr	109
9.8	gazebo::rendering Namespace Reference	109
9.8.1	Detailed Description	112
9.8.2	Typedef Documentation	112
9.8.2.1	ArrowVisualPtr	112
9.8.2.2	AxisVisualPtr	112
9.8.2.3	CameraPtr	112
9.8.2.4	CameraVisualPtr	112
9.8.2.5	COMVisualPtr	112
9.8.2.6	ContactVisualPtr	112
9.8.2.7	DepthCameraPtr	112

9.8.2.8	DynamicLinesPtr	112
9.8.2.9	GpuLaserPtr	112
9.8.2.10	JointVisualPtr	112
9.8.2.11	LaserVisualPtr	112
9.8.2.12	LightPtr	112
9.8.2.13	RFIDTagVisualPtr	112
9.8.2.14	RFIDVisualPtr	112
9.8.2.15	ScenePtr	113
9.8.2.16	UserCameraPtr	113
9.8.2.17	VisualPtr	113
9.8.3	Enumeration Type Documentation	113
9.8.3.1	RenderOpType	113
9.9	gazebo::sensors Namespace Reference	113
9.9.1	Detailed Description	114
9.9.2	Typedef Documentation	115
9.9.2.1	CameraSensor_V	115
9.9.2.2	CameraSensorPtr	115
9.9.2.3	ContactSensor_V	115
9.9.2.4	ContactSensorPtr	115
9.9.2.5	DepthCameraSensor_V	115
9.9.2.6	DepthCameraSensorPtr	115
9.9.2.7	GpuRaySensor_V	115
9.9.2.8	GpuRaySensorPtr	115
9.9.2.9	RaySensor_V	115
9.9.2.10	RaySensorPtr	115
9.9.2.11	RFIDSensor_V	115
9.9.2.12	RFIDSensorPtr	115
9.9.2.13	RFIDTag_V	115
9.9.2.14	RFIDTagPtr	115
9.9.2.15	Sensor_V	115
9.9.2.16	SensorFactoryFn	115
9.9.2.17	SensorPtr	115
9.10	gazebo::transport Namespace Reference	115
9.10.1	Detailed Description	117
9.10.2	Typedef Documentation	117
9.10.2.1	ConnectionPtr	117
9.10.2.2	NodePtr	117

9.10.2.3	PublicationPtr	117
9.10.2.4	PublicationTransportPtr	117
9.10.2.5	PublisherPtr	117
9.10.2.6	SubscriberPtr	117
9.10.2.7	SubscriptionTransportPtr	117
9.11	google Namespace Reference	117
9.12	google::protobuf Namespace Reference	117
9.13	google::protobuf::compiler Namespace Reference	117
9.14	google::protobuf::compiler::cpp Namespace Reference	118
9.15	Ogre Namespace Reference	118
9.16	ogre Namespace Reference	118
9.17	sdf Namespace Reference	118
9.17.1	Detailed Description	119
9.17.2	Typedef Documentation	119
9.17.2.1	ElementPtr	119
9.17.2.2	ElementPtr_V	119
9.17.2.3	Param_V	119
9.17.2.4	ParamPtr	119
9.17.2.5	SDFPtr	119
9.17.3	Function Documentation	119
9.17.3.1	addNestedModel	119
9.17.3.2	copyChildren	119
9.17.3.3	init	119
9.17.3.4	initDoc	119
9.17.3.5	initDoc	119
9.17.3.6	initFile	119
9.17.3.7	initFile	119
9.17.3.8	initString	119
9.17.3.9	initXml	120
9.17.3.10	readDoc	120
9.17.3.11	readDoc	120
9.17.3.12	readFile	120
9.17.3.13	readString	120
9.17.3.14	readString	120
9.17.3.15	readXml	120
9.18	SkyX Namespace Reference	120
9.19	urdf2gazebo Namespace Reference	120

9.19.1 Detailed Description	120
10 Class Documentation	121
10.1 gazebo::physics::Actor Class Reference	121
10.1.1 Detailed Description	123
10.1.2 Constructor & Destructor Documentation	123
10.1.2.1 Actor	123
10.1.2.2 ~Actor	124
10.1.3 Member Function Documentation	124
10.1.3.1 Fini	124
10.1.3.2 GetSDF	124
10.1.3.3 Init	124
10.1.3.4 IsActive	124
10.1.3.5 Load	124
10.1.3.6 Play	124
10.1.3.7 Stop	124
10.1.3.8 Update	125
10.1.3.9 UpdateParameters	125
10.1.4 Member Data Documentation	125
10.1.4.1 active	125
10.1.4.2 autoStart	125
10.1.4.3 bonePosePub	125
10.1.4.4 interpolateX	125
10.1.4.5 lastPos	125
10.1.4.6 lastScriptTime	125
10.1.4.7 lastTraj	125
10.1.4.8 loop	126
10.1.4.9 mainLink	126
10.1.4.10 mesh	126
10.1.4.11 oldAction	126
10.1.4.12 pathLength	126
10.1.4.13 playStartTime	126
10.1.4.14 prevFrameTime	126
10.1.4.15 scriptLength	126
10.1.4.16 skelAnimation	126
10.1.4.17 skeleton	126
10.1.4.18 skelNodesMap	126

10.1.4.19	skinFile	127
10.1.4.20	skinScale	127
10.1.4.21	startDelay	127
10.1.4.22	trajectories	127
10.1.4.23	trajInfo	127
10.1.4.24	visualName	127
10.2	gazebo::math::Angle Class Reference	127
10.2.1	Detailed Description	129
10.2.2	Constructor & Destructor Documentation	129
10.2.2.1	Angle	129
10.2.2.2	Angle	129
10.2.2.3	Angle	129
10.2.2.4	~Angle	129
10.2.3	Member Function Documentation	129
10.2.3.1	Degree	129
10.2.3.2	GetAsDegree	129
10.2.3.3	GetAsRadian	130
10.2.3.4	Normalize	130
10.2.3.5	operator!=	130
10.2.3.6	operator*	130
10.2.3.7	operator*	130
10.2.3.8	operator*= operator+=	130
10.2.3.9	operator+	131
10.2.3.10	operator+=	131
10.2.3.11	operator-	131
10.2.3.12	operator-=	131
10.2.3.13	operator/	132
10.2.3.14	operator/=	132
10.2.3.15	operator<	132
10.2.3.16	operator<=	132
10.2.3.17	operator==	133
10.2.3.18	operator>	133
10.2.3.19	operator>=	133
10.2.3.20	Radian	133
10.2.3.21	SetFromDegree	134
10.2.3.22	SetFromRadian	134
10.2.4	Friends And Related Function Documentation	134

10.2.4.1	operator<<	134
10.2.4.2	operator>>	134
10.3	gazebo::common::Animation Class Reference	135
10.3.1	Detailed Description	136
10.3.2	Member Typedef Documentation	136
10.3.2.1	KeyFrame_V	136
10.3.3	Constructor & Destructor Documentation	136
10.3.3.1	Animation	136
10.3.3.2	~Animation	137
10.3.4	Member Function Documentation	137
10.3.4.1	AddTime	137
10.3.4.2	GetKeyFrame	137
10.3.4.3	GetKeyFrameCount	137
10.3.4.4	GetKeyFramesAtTime	137
10.3.4.5	GetLength	138
10.3.4.6	GetTime	138
10.3.4.7	SetLength	138
10.3.4.8	SetTime	138
10.3.5	Member Data Documentation	138
10.3.5.1	build	138
10.3.5.2	keyFrames	138
10.3.5.3	length	138
10.3.5.4	loop	139
10.3.5.5	name	139
10.3.5.6	timePos	139
10.4	gazebo::rendering::ArrowVisual Class Reference	139
10.4.1	Detailed Description	140
10.4.2	Constructor & Destructor Documentation	140
10.4.2.1	ArrowVisual	140
10.4.2.2	~ArrowVisual	140
10.4.3	Member Function Documentation	140
10.4.3.1	Load	140
10.4.3.2	ShowRotation	140
10.5	gazebo::rendering::AxisVisual Class Reference	140
10.5.1	Detailed Description	141
10.5.2	Constructor & Destructor Documentation	142
10.5.2.1	AxisVisual	142

10.5.2.2	~AxisVisual	142
10.5.3	Member Function Documentation	142
10.5.3.1	Load	142
10.5.3.2	ScaleXAxis	142
10.5.3.3	ScaleYAxis	142
10.5.3.4	ScaleZAxis	142
10.5.3.5	SetAxisMaterial	143
10.5.3.6	ShowRotation	143
10.6	gazebo::physics::BallJoint< T > Class Template Reference	143
10.6.1	Detailed Description	144
10.6.2	Constructor & Destructor Documentation	144
10.6.2.1	BallJoint	144
10.6.2.2	~BallJoint	144
10.6.3	Member Function Documentation	144
10.6.3.1	GetHighStop	144
10.6.3.2	GetLowStop	144
10.6.3.3	Load	144
10.6.3.4	SetAxis	144
10.6.3.5	SetHighStop	144
10.6.3.6	SetLowStop	145
10.7	gazebo::physics::Base Class Reference	145
10.7.1	Detailed Description	149
10.7.2	Member Enumeration Documentation	149
10.7.2.1	EntityType	149
10.7.3	Constructor & Destructor Documentation	150
10.7.3.1	Base	150
10.7.3.2	~Base	150
10.7.4	Member Function Documentation	150
10.7.4.1	AddChild	150
10.7.4.2	AddType	150
10.7.4.3	Fini	150
10.7.4.4	GetById	151
10.7.4.5	GetByName	151
10.7.4.6	GetChild	151
10.7.4.7	GetChild	151
10.7.4.8	GetChildCount	152
10.7.4.9	GetId	152

10.7.4.10	GetName	152
10.7.4.11	GetParent	152
10.7.4.12	GetParentId	152
10.7.4.13	GetSaveable	152
10.7.4.14	GetScopedName	153
10.7.4.15	GetSDF	153
10.7.4.16	GetType	153
10.7.4.17	GetWorld	153
10.7.4.18	HasType	153
10.7.4.19	Init	153
10.7.4.20	IsSelected	154
10.7.4.21	Load	154
10.7.4.22	operator==	154
10.7.4.23	Print	155
10.7.4.24	RemoveChild	155
10.7.4.25	RemoveChild	155
10.7.4.26	RemoveChildren	155
10.7.4.27	Reset	155
10.7.4.28	Reset	155
10.7.4.29	SetName	156
10.7.4.30	SetParent	156
10.7.4.31	SetSaveable	156
10.7.4.32	SetSelected	156
10.7.4.33	SetWorld	156
10.7.4.34	Update	157
10.7.4.35	UpdateParameters	157
10.7.5	Member Data Documentation	157
10.7.5.1	children	157
10.7.5.2	childrenEnd	157
10.7.5.3	parent	157
10.7.5.4	sdf	157
10.7.5.5	world	157
10.8	gazebo::math::Box Class Reference	158
10.8.1	Detailed Description	159
10.8.2	Constructor & Destructor Documentation	159
10.8.2.1	Box	159
10.8.2.2	Box	159

10.8.2.3	Box	159
10.8.2.4	~Box	159
10.8.3	Member Function Documentation	159
10.8.3.1	GetCenter	159
10.8.3.2	GetSize	160
10.8.3.3	GetXLength	160
10.8.3.4	GetYLength	160
10.8.3.5	GetZLength	160
10.8.3.6	Merge	160
10.8.3.7	operator+	160
10.8.3.8	operator+=	161
10.8.3.9	operator-	161
10.8.3.10	operator=	161
10.8.3.11	operator==	161
10.8.4	Friends And Related Function Documentation	162
10.8.4.1	operator<<	162
10.8.5	Member Data Documentation	162
10.8.5.1	max	162
10.8.5.2	min	162
10.9	gazebo::physics::BoxShape Class Reference	162
10.9.1	Detailed Description	164
10.9.2	Constructor & Destructor Documentation	164
10.9.2.1	BoxShape	164
10.9.2.2	~BoxShape	164
10.9.3	Member Function Documentation	164
10.9.3.1	FillMsg	164
10.9.3.2	FillShapeMsg	164
10.9.3.3	GetInertial	164
10.9.3.4	GetMass	165
10.9.3.5	GetSize	165
10.9.3.6	Init	165
10.9.3.7	ProcessMsg	165
10.9.3.8	SetSize	165
10.10	gazebo::physics::BulletBallJoint Class Reference	166
10.10.1	Detailed Description	167
10.10.2	Constructor & Destructor Documentation	167
10.10.2.1	BulletBallJoint	167

10.10.2.2	~BulletBallJoint	167
10.10.3	Member Function Documentation	168
10.10.3.1	Attach	168
10.10.3.2	GetAnchor	168
10.10.3.3	GetAngle	168
10.10.3.4	GetAngleImpl	168
10.10.3.5	GetAxis	168
10.10.3.6	GetGlobalAxis	168
10.10.3.7	GetMaxForce	168
10.10.3.8	GetVelocity	168
10.10.3.9	SetAnchor	168
10.10.3.10	SetDamping	169
10.10.3.11	SetHighStop	169
10.10.3.12	SetLowStop	169
10.10.3.13	SetMaxForce	169
10.10.3.14	SetVelocity	169
10.11	gazebo::physics::BulletBoxShape Class Reference	169
10.11.1	Detailed Description	170
10.11.2	Constructor & Destructor Documentation	170
10.11.2.1	BulletBoxShape	171
10.11.2.2	~BulletBoxShape	171
10.11.3	Member Function Documentation	171
10.11.3.1	SetSize	171
10.12	gazebo::physics::BulletCollision Class Reference	171
10.12.1	Detailed Description	173
10.12.2	Constructor & Destructor Documentation	173
10.12.2.1	BulletCollision	173
10.12.2.2	~BulletCollision	173
10.12.3	Member Function Documentation	173
10.12.3.1	GetBoundingBox	173
10.12.3.2	GetCollisionShape	173
10.12.3.3	Load	173
10.12.3.4	OnPoseChange	173
10.12.3.5	SetCategoryBits	174
10.12.3.6	SetCollideBits	174
10.12.3.7	SetCollisionShape	174
10.12.3.8	SetCompoundShapeIndex	174

10.12.4 Member Data Documentation	174
10.12.4.1 collisionShape	174
10.13 gazebo::physics::BulletCylinderShape Class Reference	174
10.13.1 Detailed Description	175
10.13.2 Constructor & Destructor Documentation	176
10.13.2.1 BulletCylinderShape	176
10.13.2.2 ~BulletCylinderShape	176
10.13.3 Member Function Documentation	176
10.13.3.1 SetSize	176
10.14 gazebo::physics::BulletHeightmapShape Class Reference	176
10.14.1 Detailed Description	177
10.14.2 Constructor & Destructor Documentation	178
10.14.2.1 BulletHeightmapShape	178
10.14.2.2 ~BulletHeightmapShape	178
10.14.3 Member Function Documentation	178
10.14.3.1 Init	178
10.15 gazebo::physics::BulletHinge2Joint Class Reference	178
10.15.1 Detailed Description	180
10.15.2 Constructor & Destructor Documentation	180
10.15.2.1 BulletHinge2Joint	180
10.15.2.2 ~BulletHinge2Joint	181
10.15.3 Member Function Documentation	181
10.15.3.1 Attach	181
10.15.3.2 GetAnchor	181
10.15.3.3 GetAngle	181
10.15.3.4 GetAngleImpl	181
10.15.3.5 GetAxis	181
10.15.3.6 GetGlobalAxis	181
10.15.3.7 GetHighStop	181
10.15.3.8 GetLowStop	181
10.15.3.9 GetMaxForce	182
10.15.3.10 GetVelocity	182
10.15.3.11 Load	182
10.15.3.12 SetAnchor	182
10.15.3.13 SetAxis	182
10.15.3.14 SetDamping	182
10.15.3.15 SetForce	182

10.15.3.16SetHighStop	182
10.15.3.17SetLowStop	183
10.15.3.18SetMaxForce	183
10.15.3.19SetVelocity	183
10.16gazebo::physics::BulletHingeJoint Class Reference	183
10.16.1 Detailed Description	185
10.16.2 Constructor & Destructor Documentation	185
10.16.2.1 BulletHingeJoint	185
10.16.2.2 ~BulletHingeJoint	186
10.16.3 Member Function Documentation	186
10.16.3.1 Attach	186
10.16.3.2 GetAnchor	186
10.16.3.3 GetAngle	186
10.16.3.4 GetAngleImpl	186
10.16.3.5 GetForce	186
10.16.3.6 GetGlobalAxis	186
10.16.3.7 GetHighStop	186
10.16.3.8 GetLowStop	186
10.16.3.9 GetMaxForce	187
10.16.3.10GetVelocity	187
10.16.3.11Load	187
10.16.3.12SetAnchor	187
10.16.3.13SetAxis	187
10.16.3.14SetDamping	187
10.16.3.15SetForce	187
10.16.3.16SetHighStop	187
10.16.3.17SetLowStop	188
10.16.3.18SetMaxForce	188
10.16.3.19SetVelocity	188
10.17gazebo::physics::BulletJoint Class Reference	188
10.17.1 Detailed Description	189
10.17.2 Constructor & Destructor Documentation	189
10.17.2.1 BulletJoint	189
10.17.2.2 ~BulletJoint	189
10.17.3 Member Function Documentation	189
10.17.3.1 AreConnected	189
10.17.3.2 Detach	190

10.17.3.3	GetAnchor	190
10.17.3.4	GetJointLink	190
10.17.3.5	GetLinkForce	190
10.17.3.6	GetLinkTorque	190
10.17.3.7	Load	190
10.17.3.8	Reset	191
10.17.3.9	SetAnchor	191
10.17.3.10	SetAttribute	191
10.17.3.11	SetDamping	191
10.17.4	Member Data Documentation	191
10.17.4.1	constraint	191
10.17.4.2	world	191
10.18	gazebo::physics::BulletLink Class Reference	192
10.18.1	Detailed Description	194
10.18.2	Constructor & Destructor Documentation	194
10.18.2.1	BulletLink	194
10.18.2.2	~BulletLink	194
10.18.3	Member Function Documentation	194
10.18.3.1	AddForce	194
10.18.3.2	AddForceAtRelativePosition	194
10.18.3.3	AddForceAtWorldPosition	194
10.18.3.4	AddRelativeForce	195
10.18.3.5	AddRelativeTorque	195
10.18.3.6	AddTorque	195
10.18.3.7	Fini	195
10.18.3.8	GetBulletLink	195
10.18.3.9	GetEnabled	195
10.18.3.10	GetGravityMode	195
10.18.3.11	GetWorldAngularVel	195
10.18.3.12	GetWorldForce	195
10.18.3.13	GetWorldLinearVel	196
10.18.3.14	GetWorldTorque	196
10.18.3.15	nit	196
10.18.3.16	Load	196
10.18.3.17	OnPoseChange	196
10.18.3.18	SetAngularDamping	196
10.18.3.19	SetAngularVel	196

10.18.3.20	SetAutoDisable	196
10.18.3.21	SetEnabled	197
10.18.3.22	SetForce	197
10.18.3.23	SetGravityMode	197
10.18.3.24	SetLinearDamping	197
10.18.3.25	SetLinearVel	197
10.18.3.26	SetSelfCollide	197
10.18.3.27	SetTorque	197
10.18.3.28	Update	197
10.18.3.29	UpdateCoM	197
10.18.4	Member Data Documentation	198
10.18.4.1	pose	198
10.19	gazebo::physics::BulletMotionState Class Reference	198
10.19.1	Detailed Description	199
10.19.2	Constructor & Destructor Documentation	199
10.19.2.1	BulletMotionState	199
10.19.2.2	~BulletMotionState	199
10.19.3	Member Function Documentation	199
10.19.3.1	GetWorldPose	199
10.19.3.2	getWorldTransform	199
10.19.3.3	SetCoG	199
10.19.3.4	SetWorldPose	199
10.19.3.5	SetWorldPosition	199
10.19.3.6	SetWorldRotation	200
10.19.3.7	setWorldTransform	200
10.20	gazebo::physics::BulletMultiRayShape Class Reference	200
10.20.1	Detailed Description	202
10.20.2	Constructor & Destructor Documentation	202
10.20.2.1	BulletMultiRayShape	202
10.20.2.2	~BulletMultiRayShape	202
10.20.3	Member Function Documentation	202
10.20.3.1	AddRay	202
10.20.3.2	UpdateRays	202
10.21	gazebo::physics::BulletPhysics Class Reference	202
10.21.1	Detailed Description	204
10.21.2	Constructor & Destructor Documentation	204
10.21.2.1	BulletPhysics	204

10.21.2.2 ~BulletPhysics	204
10.21.3 Member Function Documentation	204
10.21.3.1 ConvertMass	204
10.21.3.2 ConvertMass	204
10.21.3.3 ConvertPose	205
10.21.3.4 ConvertPose	205
10.21.3.5 CreateCollision	205
10.21.3.6 CreateJoint	205
10.21.3.7 CreateLink	205
10.21.3.8 CreateShape	205
10.21.3.9 DebugPrint	205
10.21.3.10Fini	205
10.21.3.11GetDynamicsWorld	206
10.21.3.12GetStepTime	206
10.21.3.13Init	206
10.21.3.14InitForThread	206
10.21.3.15Load	206
10.21.3.16SetGravity	206
10.21.3.17SetStepTime	206
10.21.3.18UpdateCollision	206
10.21.3.19UpdatePhysics	206
10.22gazebo::physics::BulletPlaneShape Class Reference	207
10.22.1 Detailed Description	208
10.22.2 Constructor & Destructor Documentation	208
10.22.2.1 BulletPlaneShape	208
10.22.2.2 ~BulletPlaneShape	208
10.22.3 Member Function Documentation	208
10.22.3.1 CreatePlane	208
10.22.3.2 SetAltitude	208
10.23gazebo::physics::BulletRaySensor Class Reference	208
10.23.1 Detailed Description	209
10.23.2 Constructor & Destructor Documentation	209
10.23.2.1 BulletRaySensor	209
10.23.2.2 ~BulletRaySensor	210
10.23.3 Member Function Documentation	210
10.23.3.1 AddRay	210
10.23.3.2 GetCount	210

10.23.3.3	GetFiducial	210
10.23.3.4	GetRange	210
10.23.3.5	GetRelativePoints	210
10.23.3.6	GetRetro	210
10.23.3.7	Update	210
10.24	gazebo::physics::BulletRayShape Class Reference	210
10.24.1	Detailed Description	212
10.24.2	Constructor & Destructor Documentation	212
10.24.2.1	BulletRayShape	212
10.24.2.2	BulletRayShape	212
10.24.2.3	~BulletRayShape	212
10.24.3	Member Function Documentation	212
10.24.3.1	GetIntersection	212
10.24.3.2	SetPoints	212
10.24.3.3	Update	212
10.25	gazebo::physics::BulletScrewJoint Class Reference	213
10.25.1	Detailed Description	214
10.25.2	Constructor & Destructor Documentation	214
10.25.2.1	BulletScrewJoint	214
10.25.2.2	~BulletScrewJoint	215
10.25.3	Member Function Documentation	215
10.25.3.1	Attach	215
10.25.3.2	GetAngle	215
10.25.3.3	GetAngleImpl	215
10.25.3.4	GetGlobalAxis	215
10.25.3.5	GetHighStop	215
10.25.3.6	GetLowStop	215
10.25.3.7	GetMaxForce	215
10.25.3.8	GetVelocity	215
10.25.3.9	Load	216
10.25.3.10	SetAxis	216
10.25.3.11	SetDamping	216
10.25.3.12	SetForce	216
10.25.3.13	SetHighStop	216
10.25.3.14	SetLowStop	216
10.25.3.15	SetMaxForce	216
10.25.3.16	SetVelocity	216

10.26gazebo::physics::BulletSliderJoint Class Reference	217
10.26.1 Detailed Description	218
10.26.2 Constructor & Destructor Documentation	218
10.26.2.1 BulletSliderJoint	218
10.26.2.2 ~BulletSliderJoint	219
10.26.3 Member Function Documentation	219
10.26.3.1 Attach	219
10.26.3.2 GetAngle	219
10.26.3.3 GetAngleImpl	219
10.26.3.4 GetGlobalAxis	219
10.26.3.5 GetHighStop	219
10.26.3.6 GetLowStop	219
10.26.3.7 GetMaxForce	219
10.26.3.8 GetVelocity	219
10.26.3.9 Load	220
10.26.3.10SetAxis	220
10.26.3.11SetDamping	220
10.26.3.12SetForce	220
10.26.3.13SetHighStop	220
10.26.3.14SetLowStop	220
10.26.3.15SetMaxForce	220
10.26.3.16SetVelocity	220
10.27gazebo::physics::BulletSphereShape Class Reference	221
10.27.1 Detailed Description	222
10.27.2 Constructor & Destructor Documentation	222
10.27.2.1 BulletSphereShape	222
10.27.2.2 ~BulletSphereShape	222
10.27.3 Member Function Documentation	222
10.27.3.1 SetRadius	222
10.28gazebo::physics::BulletTrimeshShape Class Reference	222
10.28.1 Detailed Description	224
10.28.2 Constructor & Destructor Documentation	224
10.28.2.1 BulletTrimeshShape	224
10.28.2.2 ~BulletTrimeshShape	224
10.28.3 Member Function Documentation	224
10.28.3.1 Init	224
10.28.3.2 Load	224

10.29gazebo::physics::BulletUniversalJoint Class Reference	224
10.29.1 Detailed Description	226
10.29.2 Constructor & Destructor Documentation	226
10.29.2.1 BulletUniversalJoint	226
10.29.2.2 ~BulletUniversalJoint	226
10.29.3 Member Function Documentation	227
10.29.3.1 Attach	227
10.29.3.2 GetAnchor	227
10.29.3.3 GetAngle	227
10.29.3.4 GetAngleImpl	227
10.29.3.5 GetAxis	227
10.29.3.6 GetGlobalAxis	227
10.29.3.7 GetHighStop	227
10.29.3.8 GetLowStop	227
10.29.3.9 GetMaxForce	227
10.29.3.10GetVelocity	228
10.29.3.11SetAnchor	228
10.29.3.12SetAxis	228
10.29.3.13SetDamping	228
10.29.3.14SetForce	228
10.29.3.15SetHighStop	228
10.29.3.16SetLowStop	228
10.29.3.17SetMaxForce	228
10.29.3.18SetVelocity	229
10.30gazebo::common::BVHLoader Class Reference	229
10.30.1 Detailed Description	229
10.30.2 Constructor & Destructor Documentation	229
10.30.2.1 BVHLoader	229
10.30.2.2 ~BVHLoader	229
10.30.3 Member Function Documentation	229
10.30.3.1 Load	229
10.31gazebo::transport::CallbackHelper Class Reference	230
10.31.1 Detailed Description	231
10.31.2 Constructor & Destructor Documentation	231
10.31.2.1 CallbackHelper	231
10.31.2.2 ~CallbackHelper	231
10.31.3 Member Function Documentation	231

10.31.3.1	GetLatching	231
10.31.3.2	GetMsgType	231
10.31.3.3	HandleData	231
10.31.3.4	IsLocal	231
10.31.4	Member Data Documentation	231
10.31.4.1	latching	231
10.32	gazebo::transport::CallbackHelperT< M > Class Template Reference	231
10.32.1	Detailed Description	232
10.32.2	Constructor & Destructor Documentation	232
10.32.2.1	CallbackHelperT	232
10.32.3	Member Function Documentation	232
10.32.3.1	GetMsgType	232
10.32.3.2	HandleData	233
10.32.3.3	IsLocal	233
10.33	gazebo::rendering::Camera Class Reference	233
10.33.1	Detailed Description	238
10.33.2	Constructor & Destructor Documentation	238
10.33.2.1	Camera	238
10.33.2.2	~Camera	238
10.33.3	Member Function Documentation	238
10.33.3.1	AttachToVisual	238
10.33.3.2	AttachToVisualImpl	239
10.33.3.3	AttachToVisualImpl	239
10.33.3.4	ConnectNewImageFrame	239
10.33.3.5	CreateRenderTexture	240
10.33.3.6	DisconnectNewImageFrame	240
10.33.3.7	EnableSaveFrame	240
10.33.3.8	Fini	240
10.33.3.9	GetAspectRatio	240
10.33.3.10	GetAvgFPS	240
10.33.3.11	GetCameraToViewportRay	241
10.33.3.12	GetDirection	241
10.33.3.13	GetFarClip	241
10.33.3.14	GetFrameFilename	241
10.33.3.15	GetHFOV	241
10.33.3.16	GetImageByteSize	242
10.33.3.17	GetImageByteSize	242

10.33.3.18	GetImageData	242
10.33.3.19	GetImageDepth	242
10.33.3.20	GetImageFormat	242
10.33.3.21	GetImageHeight	243
10.33.3.22	GetImageWidth	243
10.33.3.23	GetInitialized	243
10.33.3.24	GetLastRenderWallTime	243
10.33.3.25	GetName	243
10.33.3.26	GetNearClip	243
10.33.3.27	GetOgreCamera	244
10.33.3.28	GetRenderTexture	244
10.33.3.29	GetRight	244
10.33.3.30	GetScene	244
10.33.3.31	GetSceneNode	244
10.33.3.32	GetTextureHeight	244
10.33.3.33	GetTextureWidth	245
10.33.3.34	GetTriangleCount	245
10.33.3.35	GetUp	245
10.33.3.36	GetVFOV	245
10.33.3.37	GetViewport	245
10.33.3.38	GetViewportHeight	245
10.33.3.39	GetViewportWidth	246
10.33.3.40	GetWindowId	246
10.33.3.41	GetWorldPointOnPlane	246
10.33.3.42	GetWorldPose	246
10.33.3.43	GetWorldPosition	246
10.33.3.44	GetWorldRotation	247
10.33.3.45	GetZValue	247
10.33.3.46	Init	247
10.33.3.47	IsInitialized	247
10.33.3.48	IsVisible	247
10.33.3.49	IsVisible	247
10.33.3.50	Load	248
10.33.3.51	Load	248
10.33.3.52	MoveToPosition	248
10.33.3.53	MoveToPositions	248
10.33.3.54	PostRender	248

10.33.3.55Render	249
10.33.3.56RenderImpl	249
10.33.3.57RotatePitch	249
10.33.3.58RotateYaw	249
10.33.3.59SaveFrame	249
10.33.3.60SaveFrame	249
10.33.3.61SetAspectRatio	250
10.33.3.62SetCaptureData	250
10.33.3.63SetClipDist	250
10.33.3.64SetHFOV	250
10.33.3.65SetImageHeight	250
10.33.3.66SetImageSize	251
10.33.3.67SetImageWidth	251
10.33.3.68SetName	251
10.33.3.69SetRenderRate	251
10.33.3.70SetRenderTarget	251
10.33.3.71SetSaveFramePathname	252
10.33.3.72SetScene	252
10.33.3.73SetSceneNode	252
10.33.3.74SetWindowId	252
10.33.3.75SetWorldPose	252
10.33.3.76SetWorldPosition	252
10.33.3.77SetWorldRotation	252
10.33.3.78ShowWireframe	253
10.33.3.79ToggleShowWireframe	253
10.33.3.80TrackVisual	253
10.33.3.81TrackVisualImpl	253
10.33.3.82TrackVisualImpl	253
10.33.3.83Translate	254
10.33.3.84Update	254
10.33.4 Member Data Documentation	254
10.33.4.1 animState	254
10.33.4.2 bayerFrameBuffer	254
10.33.4.3 camera	254
10.33.4.4 captureData	254
10.33.4.5 connections	254
10.33.4.6 imageFormat	254

10.33.4.7	imageHeight	254
10.33.4.8	imageWidth	254
10.33.4.9	initialized	254
10.33.4.10	lastRenderWallTime	254
10.33.4.11	name	254
10.33.4.12	newData	254
10.33.4.13	newImageFrame	254
10.33.4.14	onAnimationComplete	254
10.33.4.15	pitchNode	254
10.33.4.16	prevAnimTime	255
10.33.4.17	renderTarget	255
10.33.4.18	renderTexture	255
10.33.4.19	requests	255
10.33.4.20	saveCount	255
10.33.4.21	saveFrameBuffer	255
10.33.4.22	scene	255
10.33.4.23	sceneNode	255
10.33.4.24	sdf	255
10.33.4.25	textureHeight	255
10.33.4.26	textureWidth	255
10.33.4.27	viewport	255
10.33.4.28	windowId	255
10.34	gazebo::sensors::CameraSensor Class Reference	255
10.34.1	Constructor & Destructor Documentation	257
10.34.1.1	CameraSensor	257
10.34.1.2	~CameraSensor	257
10.34.2	Member Function Documentation	257
10.34.2.1	Fini	257
10.34.2.2	GetCamera	257
10.34.2.3	GetImageData	257
10.34.2.4	GetImageHeight	257
10.34.2.5	GetImageWidth	258
10.34.2.6	GetTopic	258
10.34.2.7	Init	258
10.34.2.8	Load	258
10.34.2.9	Load	258
10.34.2.10	SaveFrame	258

10.34.2.11SetActive	259
10.34.2.12SetParent	259
10.34.2.13UpdateImpl	259
10.35gazebo::rendering::CameraVisual Class Reference	259
10.35.1 Detailed Description	260
10.35.2 Constructor & Destructor Documentation	260
10.35.2.1 CameraVisual	260
10.35.2.2 ~CameraVisual	261
10.35.3 Member Function Documentation	261
10.35.3.1 Load	261
10.36gazebo::common::ColladaLoader Class Reference	261
10.36.1 Detailed Description	262
10.36.2 Constructor & Destructor Documentation	262
10.36.2.1 ColladaLoader	262
10.36.2.2 ~ColladaLoader	262
10.36.3 Member Function Documentation	262
10.36.3.1 Load	262
10.37gazebo::physics::Collision Class Reference	262
10.37.1 Detailed Description	265
10.37.2 Constructor & Destructor Documentation	265
10.37.2.1 Collision	265
10.37.2.2 ~Collision	265
10.37.3 Member Function Documentation	265
10.37.3.1 AddContact	265
10.37.3.2 ConnectContact	266
10.37.3.3 DisconnectContact	266
10.37.3.4 FillCollisionMsg	266
10.37.3.5 FillMsg	266
10.37.3.6 Fini	266
10.37.3.7 GetBoundingBox	266
10.37.3.8 GetContactsEnabled	267
10.37.3.9 GetLaserRetro	267
10.37.3.10GetLink	267
10.37.3.11GetModel	267
10.37.3.12GetRelativeAngularAccel	267
10.37.3.13GetRelativeAngularVel	268
10.37.3.14GetRelativeLinearAccel	268

10.37.3.15	GetRelativeLinearVel	268
10.37.3.16	GetShape	268
10.37.3.17	GetShapeType	268
10.37.3.18	GetState	269
10.37.3.19	GetSurface	269
10.37.3.20	GetWorldAngularAccel	269
10.37.3.21	GetWorldAngularVel	269
10.37.3.22	GetWorldLinearAccel	269
10.37.3.23	GetWorldLinearVel	269
10.37.3.24	Init	270
10.37.3.25	IsPlaceable	270
10.37.3.26	Load	270
10.37.3.27	ProcessMsg	270
10.37.3.28	SetCategoryBits	270
10.37.3.29	SetCollideBits	270
10.37.3.30	SetCollision	271
10.37.3.31	SetContactsEnabled	271
10.37.3.32	SetLaserRetro	271
10.37.3.33	SetShape	271
10.37.3.34	SetState	271
10.37.3.35	UpdateParameters	272
10.37.4	Member Data Documentation	272
10.37.4.1	link	272
10.37.4.2	placeable	272
10.37.4.3	shape	272
10.38	gazebo::physics::CollisionState Class Reference	272
10.38.1	Detailed Description	273
10.38.2	Constructor & Destructor Documentation	273
10.38.2.1	CollisionState	273
10.38.2.2	CollisionState	273
10.38.2.3	~CollisionState	273
10.38.3	Member Function Documentation	273
10.38.3.1	GetPose	273
10.38.3.2	Load	274
10.39	gazebo::common::Color Class Reference	274
10.39.1	Detailed Description	276
10.39.2	Member Typedef Documentation	276

10.39.2.1 ABGR	276
10.39.2.2 ARGB	276
10.39.2.3 BGRA	276
10.39.2.4 RGBA	276
10.39.3 Constructor & Destructor Documentation	276
10.39.3.1 Color	277
10.39.3.2 Color	277
10.39.3.3 Color	277
10.39.3.4 ~Color	277
10.39.4 Member Function Documentation	277
10.39.4.1 GetAsABGR	277
10.39.4.2 GetAsARGB	277
10.39.4.3 GetAsBGRA	277
10.39.4.4 GetAsHSV	278
10.39.4.5 GetAsRGBA	278
10.39.4.6 GetAsYUV	278
10.39.4.7 operator!=	278
10.39.4.8 operator*	278
10.39.4.9 operator*	279
10.39.4.10operator*=	279
10.39.4.11operator+	279
10.39.4.12operator+	279
10.39.4.13operator+=	280
10.39.4.14operator-	280
10.39.4.15operator-	280
10.39.4.16operator-=	280
10.39.4.17operator/	281
10.39.4.18operator/	281
10.39.4.19operator/=	281
10.39.4.20operator=	281
10.39.4.21operator==	282
10.39.4.22operator[]	282
10.39.4.23Reset	282
10.39.4.24Set	282
10.39.4.25SetFromABGR	282
10.39.4.26SetFromARGB	283
10.39.4.27SetFromBGRA	283

10.39.4.28SetFromHSV	283
10.39.4.29SetFromRGBA	283
10.39.4.30SetFromYUV	283
10.39.5 Friends And Related Function Documentation	284
10.39.5.1 operator<<	284
10.39.5.2 operator>>	284
10.39.6 Member Data Documentation	284
10.39.6.1 a	284
10.39.6.2 b	284
10.39.6.3 Black	284
10.39.6.4 Blue	284
10.39.6.5 g	284
10.39.6.6 Green	284
10.39.6.7 Purple	284
10.39.6.8 r	285
10.39.6.9 Red	285
10.39.6.10White	285
10.39.6.11Yellow	285
10.40gazebo::rendering::COMVisual Class Reference	285
10.40.1 Detailed Description	286
10.40.2 Constructor & Destructor Documentation	286
10.40.2.1 COMVisual	286
10.40.2.2 ~COMVisual	286
10.40.3 Member Function Documentation	286
10.40.3.1 Load	286
10.40.3.2 Load	286
10.41gazebo::event::Connection Class Reference	287
10.41.1 Detailed Description	287
10.41.2 Constructor & Destructor Documentation	287
10.41.2.1 Connection	287
10.41.2.2 Connection	287
10.41.2.3 ~Connection	287
10.41.3 Member Function Documentation	288
10.41.3.1 GetId	288
10.42gazebo::transport::Connection Class Reference	288
10.42.1 Detailed Description	290
10.42.2 Member Typedef Documentation	290

10.42.2.1	AcceptCallback	290
10.42.2.2	ReadCallback	290
10.42.3	Constructor & Destructor Documentation	290
10.42.3.1	Connection	290
10.42.3.2	~Connection	290
10.42.4	Member Function Documentation	290
10.42.4.1	AsyncRead	290
10.42.4.2	Cancel	290
10.42.4.3	Connect	290
10.42.4.4	ConnectToShutdown	290
10.42.4.5	DisconnectShutdown	290
10.42.4.6	EnqueueMsg	290
10.42.4.7	GetLocalAddress	291
10.42.4.8	GetLocalHostname	291
10.42.4.9	GetLocalPort	291
10.42.4.10	GetLocalURI	291
10.42.4.11	GetRemoteAddress	291
10.42.4.12	GetRemoteHostname	291
10.42.4.13	GetRemotePort	291
10.42.4.14	GetRemoteURI	291
10.42.4.15	IsOpen	291
10.42.4.16	Listen	291
10.42.4.17	ProcessWriteQueue	291
10.42.4.18	Read	292
10.42.4.19	Shutdown	292
10.42.4.20	StartRead	292
10.42.4.21	StopRead	292
10.42.5	Member Data Documentation	292
10.42.5.1	id	292
10.42.5.2	writeCount	292
10.43	gazebo::transport::ConnectionManager Class Reference	292
10.43.1	Detailed Description	293
10.43.2	Member Function Documentation	293
10.43.2.1	Advertise	293
10.43.2.2	ConnectToRemoteHost	293
10.43.2.3	Fini	294
10.43.2.4	GetAllPublishers	294

10.43.2.5	GetTopicNamespaces	294
10.43.2.6	Init	294
10.43.2.7	IsRunning	294
10.43.2.8	RegisterTopicNamespace	294
10.43.2.9	RemoveConnection	294
10.43.2.10	Run	294
10.43.2.11	RunUpdate	294
10.43.2.12	Stop	294
10.43.2.13	Subscribe	294
10.43.2.14	Unadvertise	294
10.43.2.15	Unsubscribe	294
10.43.2.16	Unsubscribe	294
10.43.3	Member Data Documentation	294
10.43.3.1	eventConnections	295
10.44	gazebo::common::Console Class Reference	295
10.44.1	Detailed Description	295
10.44.2	Member Function Documentation	295
10.44.2.1	ColorErr	295
10.44.2.2	ColorMsg	296
10.44.2.3	Instance	296
10.44.2.4	Load	296
10.44.2.5	Log	296
10.44.2.6	SetQuiet	296
10.45	gazebo::physics::Contact Class Reference	296
10.45.1	Detailed Description	297
10.45.2	Constructor & Destructor Documentation	297
10.45.2.1	Contact	297
10.45.2.2	Contact	298
10.45.2.3	~Contact	298
10.45.3	Member Function Documentation	298
10.45.3.1	Clone	298
10.45.3.2	operator=	298
10.45.3.3	Reset	298
10.45.4	Member Data Documentation	298
10.45.4.1	collision1	298
10.45.4.2	collision2	298
10.45.4.3	count	299

10.45.4.4 depths	299
10.45.4.5 forces	299
10.45.4.6 normals	299
10.45.4.7 positions	299
10.45.4.8 time	299
10.46gazebo::physics::ContactFeedback Class Reference	299
10.46.1 Detailed Description	299
10.46.2 Member Data Documentation	300
10.46.2.1 contact	300
10.46.2.2 feedbackCount	300
10.46.2.3 feedbacks	300
10.47gazebo::sensors::ContactSensor Class Reference	300
10.47.1 Constructor & Destructor Documentation	301
10.47.1.1 ContactSensor	301
10.47.1.2 ~ContactSensor	301
10.47.2 Member Function Documentation	301
10.47.2.1 Fini	301
10.47.2.2 GetCollisionContact	302
10.47.2.3 GetCollisionContactCount	302
10.47.2.4 GetCollisionCount	302
10.47.2.5 GetCollisionName	302
10.47.2.6 GetContacts	302
10.47.2.7 GetUpdateMutex	303
10.47.2.8 Init	303
10.47.2.9 Load	303
10.47.2.10Load	303
10.47.2.11UpdateImpl	303
10.48gazebo::rendering::ContactVisual Class Reference	304
10.48.1 Detailed Description	305
10.48.2 Constructor & Destructor Documentation	305
10.48.2.1 ContactVisual	305
10.48.2.2 ~ContactVisual	305
10.49gazebo::rendering::Conversions Class Reference	305
10.49.1 Detailed Description	305
10.49.2 Member Function Documentation	306
10.49.2.1 Convert	306
10.49.2.2 Convert	306

10.49.2.3 Convert	306
10.49.2.4 Convert	306
10.49.2.5 Convert	307
10.50sdf::Converter Class Reference	307
10.50.1 Detailed Description	307
10.50.2 Member Function Documentation	307
10.50.2.1 Convert	307
10.51gazebo::physics::CylinderShape Class Reference	307
10.51.1 Detailed Description	309
10.51.2 Constructor & Destructor Documentation	309
10.51.2.1 CylinderShape	309
10.51.2.2 ~CylinderShape	309
10.51.3 Member Function Documentation	309
10.51.3.1 FillMsg	309
10.51.3.2 GetInertial	309
10.51.3.3 GetLength	310
10.51.3.4 GetMass	310
10.51.3.5 GetRadius	310
10.51.3.6 Init	310
10.51.3.7 ProcessMsg	310
10.51.3.8 SetLength	310
10.51.3.9 SetRadius	311
10.51.3.10SetSize	311
10.52gazebo::transport::DebugCallbackHelper Class Reference	311
10.52.1 Constructor & Destructor Documentation	312
10.52.1.1 DebugCallbackHelper	312
10.52.2 Member Function Documentation	312
10.52.2.1 GetMsgType	312
10.52.2.2 HandleData	312
10.52.2.3 IsLocal	312
10.53gazebo::rendering::DepthCamera Class Reference	312
10.53.1 Detailed Description	314
10.53.2 Constructor & Destructor Documentation	314
10.53.2.1 DepthCamera	314
10.53.2.2 ~DepthCamera	314
10.53.3 Member Function Documentation	314
10.53.3.1 ConnectNewDepthFrame	314

10.53.3.2 ConnectNewRGBPointCloud	315
10.53.3.3 CreateDepthTexture	315
10.53.3.4 DisconnectNewDepthFrame	315
10.53.3.5 DisconnectNewRGBPointCloud	315
10.53.3.6 Fini	316
10.53.3.7 GetDepthData	316
10.53.3.8 Init	316
10.53.3.9 Load	316
10.53.3.10 Load	316
10.53.3.11 PostRender	316
10.53.3.12 SetDepthTarget	316
10.53.4 Member Data Documentation	316
10.53.4.1 depthTarget	317
10.53.4.2 depthTexture	317
10.53.4.3 depthViewport	317
10.54 gazebo::sensors::DepthCameraSensor Class Reference	317
10.54.1 Constructor & Destructor Documentation	318
10.54.1.1 DepthCameraSensor	318
10.54.1.2 ~DepthCameraSensor	318
10.54.2 Member Function Documentation	318
10.54.2.1 Fini	318
10.54.2.2 GetDepthCamera	318
10.54.2.3 Init	319
10.54.2.4 Load	319
10.54.2.5 Load	319
10.54.2.6 SaveFrame	319
10.54.2.7 SetActive	319
10.54.2.8 SetParent	320
10.54.2.9 UpdateImpl	320
10.55 gazebo::common::DiagnosticManager Class Reference	320
10.55.1 Detailed Description	321
10.55.2 Member Function Documentation	321
10.55.2.1 CreateTimer	321
10.55.2.2 GetEnabled	321
10.55.2.3 GetLabel	321
10.55.2.4 GetTime	322
10.55.2.5 GetTime	322

10.55.2.6	GetTimerCount	322
10.55.2.7	SetEnabled	322
10.55.2.8	TimerStart	323
10.55.2.9	TimerStop	323
10.56	gazebo::common::DiagnosticTimer Class Reference	323
10.56.1	Detailed Description	324
10.56.2	Constructor & Destructor Documentation	324
10.56.2.1	DiagnosticTimer	324
10.56.2.2	~DiagnosticTimer	324
10.56.3	Member Function Documentation	324
10.56.3.1	GetName	324
10.57	gazebo::rendering::DynamicLines Class Reference	324
10.57.1	Detailed Description	326
10.57.2	Constructor & Destructor Documentation	326
10.57.2.1	DynamicLines	326
10.57.2.2	~DynamicLines	326
10.57.3	Member Function Documentation	326
10.57.3.1	AddPoint	326
10.57.3.2	Clear	326
10.57.3.3	CreateVertexDeclaration	326
10.57.3.4	FillHardwareBuffers	327
10.57.3.5	GetMovableType	327
10.57.3.6	getMovableType	327
10.57.3.7	GetPoint	327
10.57.3.8	GetPointCount	327
10.57.3.9	SetPoint	327
10.57.3.10	Update	328
10.58	gazebo::rendering::DynamicRenderable Class Reference	328
10.58.1	Detailed Description	329
10.58.2	Constructor & Destructor Documentation	329
10.58.2.1	DynamicRenderable	329
10.58.2.2	~DynamicRenderable	329
10.58.3	Member Function Documentation	329
10.58.3.1	CreateVertexDeclaration	329
10.58.3.2	FillHardwareBuffers	330
10.58.3.3	getBoundingRadius	330
10.58.3.4	GetOperationType	330

10.58.3.5	getSquaredViewDepth	330
10.58.3.6	Init	330
10.58.3.7	PrepareHardwareBuffers	331
10.58.3.8	SetOperationType	331
10.58.4	Member Data Documentation	331
10.58.4.1	indexBufferCapacity	331
10.58.4.2	vertexBufferCapacity	331
10.59sdf::Element	Class Reference	332
10.59.1	Detailed Description	334
10.59.2	Constructor & Destructor Documentation	334
10.59.2.1	Element	334
10.59.2.2	~Element	334
10.59.3	Member Function Documentation	334
10.59.3.1	AddAttribute	334
10.59.3.2	AddElement	334
10.59.3.3	AddElementDescription	334
10.59.3.4	AddValue	334
10.59.3.5	ClearElements	334
10.59.3.6	Clone	334
10.59.3.7	Copy	334
10.59.3.8	GetAttribute	335
10.59.3.9	GetAttribute	335
10.59.3.10	GetAttributeCount	335
10.59.3.11	GetAttributeSet	335
10.59.3.12	GetCopyChildren	335
10.59.3.13	GetDescription	335
10.59.3.14	GetElement	335
10.59.3.15	GetElement	335
10.59.3.16	GetElementDescription	335
10.59.3.17	GetElementDescription	335
10.59.3.18	GetElementDescriptionCount	335
10.59.3.19	GetElementImpl	336
10.59.3.20	GetFirstElement	336
10.59.3.21	GetInclude	336
10.59.3.22	GetName	336
10.59.3.23	GetNextElement	336
10.59.3.24	GetParent	336

10.59.3.25GetRequired	336
10.59.3.26GetValue	336
10.59.3.27GetValueBool	336
10.59.3.28GetValueChar	336
10.59.3.29GetValueColor	336
10.59.3.30GetValueDouble	336
10.59.3.31GetValueFloat	336
10.59.3.32GetValueInt	336
10.59.3.33GetValuePose	336
10.59.3.34GetValueQuaternion	336
10.59.3.35GetValueString	336
10.59.3.36GetValueTime	336
10.59.3.37GetValueUInt	336
10.59.3.38GetValueVector2d	336
10.59.3.39GetValueVector3	336
10.59.3.40HasAttribute	336
10.59.3.41HasElement	337
10.59.3.42HasElementDescription	337
10.59.3.43InsertElement	337
10.59.3.44PrintDescription	337
10.59.3.45PrintDocLeftPane	337
10.59.3.46PrintDocRightPane	337
10.59.3.47PrintValues	337
10.59.3.48PrintWiki	337
10.59.3.49Reset	337
10.59.3.50Set	337
10.59.3.51Set	337
10.59.3.52Set	337
10.59.3.53Set	337
10.59.3.54Set	337
10.59.3.55Set	338
10.59.3.56Set	338
10.59.3.57Set	338
10.59.3.58Set	338
10.59.3.59Set	338
10.59.3.60Set	338
10.59.3.61Set	338

10.59.3.62Set	338
10.59.3.63Set	338
10.59.3.64Set	338
10.59.3.65SetCopyChildren	338
10.59.3.66SetDescription	338
10.59.3.67SetInclude	338
10.59.3.68SetName	338
10.59.3.69SetParent	338
10.59.3.70SetRequired	338
10.59.3.71ToString	338
10.59.3.72Update	338
10.60gazebo::physics::Entity Class Reference	338
10.60.1 Detailed Description	341
10.60.2 Constructor & Destructor Documentation	341
10.60.2.1 Entity	341
10.60.2.2 ~Entity	341
10.60.3 Member Function Documentation	342
10.60.3.1 Fini	342
10.60.3.2 GetBoundingBox	342
10.60.3.3 GetChildCollision	342
10.60.3.4 GetChildLink	342
10.60.3.5 GetCollisionBoundingBox	342
10.60.3.6 GetDirtyPose	343
10.60.3.7 GetNearestEntityBelow	343
10.60.3.8 GetParentModel	343
10.60.3.9 GetRelativeAngularAccel	343
10.60.3.10GetRelativeAngularVel	343
10.60.3.11GetRelativeLinearAccel	344
10.60.3.12GetRelativeLinearVel	344
10.60.3.13GetRelativePose	344
10.60.3.14GetWorldAngularAccel	344
10.60.3.15GetWorldAngularVel	344
10.60.3.16GetWorldLinearAccel	345
10.60.3.17GetWorldLinearVel	345
10.60.3.18GetWorldPose	345
10.60.3.19sCanonicalLink	345
10.60.3.20sStatic	345

10.60.3.21	Load	346
10.60.3.22	OnPoseChange	346
10.60.3.23	PlaceOnEntity	346
10.60.3.24	PlaceOnNearestEntityBelow	346
10.60.3.25	Reset	346
10.60.3.26	SetAnimation	346
10.60.3.27	SetAnimation	347
10.60.3.28	SetCanonicalLink	347
10.60.3.29	SetInitialRelativePose	347
10.60.3.30	SetName	347
10.60.3.31	SetRelativePose	347
10.60.3.32	SetStatic	347
10.60.3.33	SetWorldPose	348
10.60.3.34	SetWorldTwist	348
10.60.3.35	StopAnimation	348
10.60.3.36	UpdateParameters	348
10.60.4	Member Data Documentation	348
10.60.4.1	animation	348
10.60.4.2	animationConnection	349
10.60.4.3	animationStartPose	349
10.60.4.4	connections	349
10.60.4.5	dirtyPose	349
10.60.4.6	node	349
10.60.4.7	parentEntity	349
10.60.4.8	poseMsg	349
10.60.4.9	prevAnimationTime	349
10.60.4.10	requestPub	349
10.60.4.11	visPub	349
10.60.4.12	visualMsg	349
10.61	gazebo::event::Event Class Reference	350
10.61.1	Detailed Description	351
10.61.2	Constructor & Destructor Documentation	351
10.61.2.1	~Event	351
10.61.3	Member Function Documentation	351
10.61.3.1	Disconnect	351
10.61.3.2	Disconnect	351
10.62	gazebo::event::Events Class Reference	352

10.62.1 Detailed Description	354
10.63gazebo::rendering::Events Class Reference	354
10.63.1 Detailed Description	355
10.63.2 Member Function Documentation	355
10.63.2.1 ConnectCreateScene	355
10.63.2.2 ConnectRemoveScene	355
10.63.2.3 DisconnectCreateScene	356
10.63.2.4 DisconnectRemoveScene	356
10.63.3 Member Data Documentation	356
10.63.3.1 createScene	356
10.63.3.2 removeScene	356
10.64gazebo::event::EventT< T > Class Template Reference	356
10.64.1 Detailed Description	359
10.64.2 Member Function Documentation	359
10.64.2.1 operator()	359
10.64.2.2 operator()	359
10.64.2.3 operator()	359
10.64.2.4 operator()	359
10.64.2.5 operator()	360
10.64.2.6 operator()	360
10.64.2.7 operator()	360
10.64.2.8 operator()	360
10.64.2.9 operator()	361
10.64.2.10operator()	361
10.64.2.11operator()	362
10.64.2.12Signal	362
10.64.2.13Signal	362
10.64.2.14Signal	362
10.64.2.15Signal	362
10.64.2.16Signal	363
10.64.2.17Signal	363
10.64.2.18Signal	363
10.64.2.19Signal	364
10.64.2.20Signal	364
10.64.2.21Signal	364
10.64.2.22Signal	365
10.65gazebo::common::Exception Class Reference	365

10.65.1 Detailed Description	366
10.65.2 Constructor & Destructor Documentation	366
10.65.2.1 Exception	366
10.65.2.2 Exception	366
10.65.2.3 ~Exception	366
10.65.3 Member Function Documentation	366
10.65.3.1 GetErrorFile	366
10.65.3.2 GetErrorStr	366
10.65.4 Friends And Related Function Documentation	366
10.65.4.1 operator<<	366
10.66 gazebo::rendering::FPSViewController Class Reference	367
10.66.1 Detailed Description	368
10.66.2 Constructor & Destructor Documentation	368
10.66.2.1 FPSViewController	368
10.66.2.2 ~FPSViewController	368
10.66.3 Member Function Documentation	368
10.66.3.1 GetTypeString	368
10.66.3.2 HandleKeyPressEvent	368
10.66.3.3 HandleKeyReleaseEvent	368
10.66.3.4 HandleMouseEvent	369
10.66.3.5 Init	369
10.66.3.6 Update	369
10.67 urdf2gazebo::GazeboExtension Class Reference	369
10.67.1 Constructor & Destructor Documentation	370
10.67.1.1 GazeboExtension	370
10.67.1.2 GazeboExtension	370
10.67.2 Member Data Documentation	370
10.67.2.1 blobs	370
10.67.2.2 damping_factor	371
10.67.2.3 fdir1	371
10.67.2.4 fudge_factor	371
10.67.2.5 gravity	371
10.67.2.6 initial_joint_position	371
10.67.2.7 is_damping_factor	371
10.67.2.8 is_fudge_factor	371
10.67.2.9 is_initial_joint_position	371
10.67.2.10 s_kd	371

10.67.2.11	is_kp	371
10.67.2.12	is_laser_retro	371
10.67.2.13	is_maxVel	372
10.67.2.14	is_minDepth	372
10.67.2.15	is_mu1	372
10.67.2.16	is_mu2	372
10.67.2.17	is_stop_cfm	372
10.67.2.18	is_stop_erp	372
10.67.2.19	kd	372
10.67.2.20	kp	372
10.67.2.21	laser_retro	372
10.67.2.22	material	372
10.67.2.23	maxVel	372
10.67.2.24	minDepth	373
10.67.2.25	mu1	373
10.67.2.26	mu2	373
10.67.2.27	old_link_name	373
10.67.2.28	provideFeedback	373
10.67.2.29	reduction_transform	373
10.67.2.30	self_collide	373
10.67.2.31	setStaticFlag	373
10.67.2.32	stop_cfm	373
10.67.2.33	stop_erp	373
10.68	google::protobuf::compiler::cpp::GazeboGenerator Class Reference	374
10.68.1	Detailed Description	374
10.68.2	Constructor & Destructor Documentation	374
10.68.2.1	GazeboGenerator	374
10.68.2.2	~GazeboGenerator	374
10.68.3	Member Function Documentation	374
10.68.3.1	Generate	374
10.69	gazebo::rendering::GpuLaser Class Reference	375
10.69.1	Detailed Description	376
10.69.2	Constructor & Destructor Documentation	376
10.69.2.1	GpuLaser	376
10.69.2.2	~GpuLaser	376
10.69.3	Member Function Documentation	376
10.69.3.1	ConnectNewLaserFrame	376

10.69.3.2 CreateLaserTexture	377
10.69.3.3 DisconnectNewLaserFrame	377
10.69.3.4 Fini	377
10.69.3.5 GetLaserData	377
10.69.3.6 Init	377
10.69.3.7 Load	377
10.69.3.8 Load	378
10.69.3.9 notifyRenderSingleObject	378
10.69.3.10PostRender	378
10.69.3.11SetParentSensor	378
10.69.3.12SetRangeCount	378
10.70gazebo::sensors::GpuRaySensor Class Reference	378
10.70.1 Constructor & Destructor Documentation	381
10.70.1.1 GpuRaySensor	381
10.70.1.2 ~GpuRaySensor	381
10.70.2 Member Function Documentation	381
10.70.2.1 ConnectNewLaserFrame	381
10.70.2.2 DisconnectNewLaserFrame	382
10.70.2.3 Fini	382
10.70.2.4 Get1stRatio	382
10.70.2.5 Get2ndRatio	382
10.70.2.6 GetAngleMax	382
10.70.2.7 GetAngleMin	382
10.70.2.8 GetAngleResolution	382
10.70.2.9 GetCameraCount	382
10.70.2.10GetCHFOV	383
10.70.2.11GetCVFOV	383
10.70.2.12GetFiducial	383
10.70.2.13GetHAngle	383
10.70.2.14GetHFOV	383
10.70.2.15GetLaserCamera	383
10.70.2.16GetRange	384
10.70.2.17GetRangeCount	384
10.70.2.18GetRangeMax	384
10.70.2.19GetRangeMin	384
10.70.2.20GetRangeResolution	384
10.70.2.21GetRanges	385

10.70.2.22	GetRayCount	385
10.70.2.23	GetRetro	385
10.70.2.24	GetVAngle	385
10.70.2.25	GetVerticalAngleMax	385
10.70.2.26	GetVerticalAngleMin	386
10.70.2.27	GetVerticalRangeCount	386
10.70.2.28	GetVerticalRayCount	386
10.70.2.29	GetVFOV	386
10.70.2.30	Init	386
10.70.2.31	IsHorizontal	386
10.70.2.32	Load	387
10.70.2.33	Load	387
10.70.2.34	SetAngleMax	387
10.70.2.35	SetAngleMin	387
10.70.2.36	SetVerticalAngleMax	387
10.70.2.37	SetVerticalAngleMin	387
10.70.2.38	UpdateImpl	388
10.70.3	Member Data Documentation	388
10.70.3.1	cameraCount	388
10.70.3.2	cameraElem	388
10.70.3.3	chfov	388
10.70.3.4	cvfov	388
10.70.3.5	far	388
10.70.3.6	hang	388
10.70.3.7	height_1st	388
10.70.3.8	height_2nd	388
10.70.3.9	hfov	388
10.70.3.10	horzElem	388
10.70.3.11	isHorizontal	388
10.70.3.12	near	388
10.70.3.13	offset	388
10.70.3.14	rangeElem	388
10.70.3.15	ratio_1st	388
10.70.3.16	ratio_2nd	388
10.70.3.17	rayElem	388
10.70.3.18	scanElem	388
10.70.3.19	vang	389

10.70.3.20	vertElem	389
10.70.3.21	vfov	389
10.70.3.22	width_1st	389
10.70.3.23	width_2nd	389
10.71	gazebo::rendering::Grid Class Reference	389
10.71.1	Detailed Description	390
10.71.2	Constructor & Destructor Documentation	390
10.71.2.1	Grid	390
10.71.2.2	~Grid	390
10.71.3	Member Function Documentation	390
10.71.3.1	Enable	390
10.71.3.2	GetCellCount	390
10.71.3.3	GetCellLength	391
10.71.3.4	GetColor	391
10.71.3.5	GetHeight	391
10.71.3.6	GetLineWidth	391
10.71.3.7	GetSceneNode	391
10.71.3.8	Init	391
10.71.3.9	SetCellCount	391
10.71.3.10	SetCellLength	392
10.71.3.11	SetColor	392
10.71.3.12	SetHeight	392
10.71.3.13	SetLineWidth	392
10.71.3.14	SetUserData	392
10.72	gazebo::physics::Gripper Class Reference	393
10.72.1	Detailed Description	393
10.72.2	Constructor & Destructor Documentation	393
10.72.2.1	Gripper	393
10.72.2.2	~Gripper	393
10.72.3	Member Function Documentation	393
10.72.3.1	Init	393
10.72.3.2	Load	393
10.73	gazebo::rendering::GUIOverlay Class Reference	394
10.73.1	Detailed Description	395
10.73.2	Constructor & Destructor Documentation	395
10.73.2.1	GUIOverlay	395
10.73.2.2	~GUIOverlay	395

10.73.3 Member Function Documentation	395
10.73.3.1 AttachCameraToImage	395
10.73.3.2 AttachCameraToImage	395
10.73.3.3 ButtonCallback	395
10.73.3.4 CreateWindow	396
10.73.3.5 HandleKeyPressEvent	396
10.73.3.6 HandleKeyReleaseEvent	396
10.73.3.7 HandleMouseEvent	396
10.73.3.8 Hide	397
10.73.3.9 Init	397
10.73.3.10 IsInitialized	397
10.73.3.11 LoadLayout	397
10.73.3.12 Resize	397
10.73.3.13 Show	397
10.73.3.14 Update	397
10.74 gazebo::rendering::Heightmap Class Reference	398
10.74.1 Detailed Description	398
10.74.2 Constructor & Destructor Documentation	398
10.74.2.1 Heightmap	398
10.74.2.2 ~Heightmap	398
10.74.3 Member Function Documentation	398
10.74.3.1 GetHeight	398
10.74.3.2 Load	399
10.74.3.3 LoadFromMsg	399
10.75 gazebo::physics::HeightmapShape Class Reference	399
10.75.1 Detailed Description	401
10.75.2 Constructor & Destructor Documentation	401
10.75.2.1 HeightmapShape	401
10.75.2.2 ~HeightmapShape	401
10.75.3 Member Function Documentation	401
10.75.3.1 FillMsg	401
10.75.3.2 GetHeight	402
10.75.3.3 GetMaxHeight	402
10.75.3.4 GetMinHeight	402
10.75.3.5 GetPos	402
10.75.3.6 GetSize	402
10.75.3.7 GetURI	403

10.75.3.8	GetVertexCount	403
10.75.3.9	Init	403
10.75.3.10	Load	403
10.75.3.11	ProcessMsg	403
10.75.4	Member Data Documentation	403
10.75.4.1	heights	403
10.75.4.2	img	404
10.75.4.3	scale	404
10.75.4.4	subSampling	404
10.75.4.5	vertSize	404
10.76	gazebo::physics::Hinge2Joint< T > Class Template Reference	404
10.76.1	Detailed Description	405
10.76.2	Constructor & Destructor Documentation	405
10.76.2.1	Hinge2Joint	405
10.76.2.2	~Hinge2Joint	405
10.76.3	Member Function Documentation	405
10.76.3.1	Load	405
10.77	gazebo::physics::HingeJoint< T > Class Template Reference	405
10.77.1	Detailed Description	406
10.77.2	Constructor & Destructor Documentation	406
10.77.2.1	HingeJoint	406
10.77.2.2	~HingeJoint	406
10.77.3	Member Function Documentation	407
10.77.3.1	Init	407
10.77.3.2	Load	407
10.78	gazebo::common::Image Class Reference	407
10.78.1	Detailed Description	408
10.78.2	Member Enumeration Documentation	408
10.78.2.1	PixelFormat	408
10.78.3	Constructor & Destructor Documentation	409
10.78.3.1	Image	409
10.78.3.2	~Image	409
10.78.4	Member Function Documentation	409
10.78.4.1	GetAvgColor	409
10.78.4.2	GetBPP	409
10.78.4.3	GetData	409
10.78.4.4	GetFilename	410

10.78.4.5	GetHeight	410
10.78.4.6	GetMaxColor	410
10.78.4.7	GetPitch	410
10.78.4.8	GetPixel	410
10.78.4.9	GetPixelFormat	410
10.78.4.10	GetRGBData	411
10.78.4.11	GetWidth	411
10.78.4.12	Load	411
10.78.4.13	Rescale	411
10.78.4.14	SavePNG	411
10.78.4.15	SetFromData	411
10.78.4.16	Valid	412
10.79	gazebo::sensors::ImuSensor Class Reference	412
10.79.1	Detailed Description	413
10.79.2	Constructor & Destructor Documentation	413
10.79.2.1	ImuSensor	413
10.79.2.2	~ImuSensor	413
10.79.3	Member Function Documentation	413
10.79.3.1	FiniChild	413
10.79.3.2	GetVelocity	414
10.79.3.3	InitChild	414
10.79.3.4	LoadChild	414
10.79.3.5	SaveChild	414
10.79.3.6	UpdateChild	414
10.80	gazebo::physics::Inertial Class Reference	414
10.80.1	Detailed Description	416
10.80.2	Constructor & Destructor Documentation	416
10.80.2.1	Inertial	416
10.80.2.2	Inertial	416
10.80.2.3	Inertial	417
10.80.2.4	~Inertial	417
10.80.3	Member Function Documentation	417
10.80.3.1	GetCoG	417
10.80.3.2	GetIXX	417
10.80.3.3	GetIXY	417
10.80.3.4	GetIXZ	417
10.80.3.5	GetIYY	418

10.80.3.6	GetIYZ	418
10.80.3.7	GetIZZ	418
10.80.3.8	GetMass	418
10.80.3.9	GetPose	418
10.80.3.10	GetPrincipalMoments	418
10.80.3.11	GetProductsofInertia	418
10.80.3.12	Load	419
10.80.3.13	operator+	419
10.80.3.14	operator+=	419
10.80.3.15	operator=	419
10.80.3.16	ProcessMsg	419
10.80.3.17	Reset	420
10.80.3.18	Rotate	420
10.80.3.19	SetCoG	420
10.80.3.20	SetCoG	420
10.80.3.21	SetInertiaMatrix	420
10.80.3.22	SetIXX	421
10.80.3.23	SetIXY	421
10.80.3.24	SetIXZ	421
10.80.3.25	SetIYY	421
10.80.3.26	SetIYZ	421
10.80.3.27	SetIZZ	421
10.80.3.28	SetMass	422
10.80.3.29	UpdateParameters	422
10.80.4	Friends And Related Function Documentation	422
10.80.4.1	operator<<	422
10.81	gazebo::transport::IOManager Class Reference	422
10.81.1	Detailed Description	422
10.81.2	Constructor & Destructor Documentation	423
10.81.2.1	IOManager	423
10.81.2.2	~IOManager	423
10.81.3	Member Function Documentation	423
10.81.3.1	DecCount	423
10.81.3.2	GetCount	423
10.81.3.3	GetIO	423
10.81.3.4	IncCount	423
10.81.3.5	Stop	423

10.82gazebo::physics::Joint Class Reference	423
10.82.1 Detailed Description	426
10.82.2 Member Enumeration Documentation	426
10.82.2.1 Attribute	426
10.82.3 Constructor & Destructor Documentation	427
10.82.3.1 Joint	427
10.82.3.2 ~Joint	427
10.82.4 Member Function Documentation	427
10.82.4.1 AreConnected	427
10.82.4.2 Attach	427
10.82.4.3 ConnectJointUpdate	427
10.82.4.4 Detach	428
10.82.4.5 DisconnectJointUpdate	428
10.82.4.6 FillJointMsg	428
10.82.4.7 FillMsg	428
10.82.4.8 GetAnchor	428
10.82.4.9 GetAngle	429
10.82.4.10GetAngleImpl	429
10.82.4.11GetChild	429
10.82.4.12GetForce	430
10.82.4.13GetGlobalAxis	430
10.82.4.14GetHighStop	430
10.82.4.15GetJointLink	431
10.82.4.16GetLinkForce	431
10.82.4.17GetLinkTorque	431
10.82.4.18GetLocalAxis	431
10.82.4.19GetLowStop	432
10.82.4.20GetMaxForce	432
10.82.4.21GetParent	432
10.82.4.22GetState	433
10.82.4.23GetVelocity	433
10.82.4.24Init	433
10.82.4.25Load	433
10.82.4.26Load	433
10.82.4.27Reset	434
10.82.4.28SetAnchor	434
10.82.4.29SetAngle	434

10.82.4.30	SetAttribute	435
10.82.4.31	SetAxis	435
10.82.4.32	SetDamping	435
10.82.4.33	SetForce	435
10.82.4.34	SetHighStop	436
10.82.4.35	SetLowStop	436
10.82.4.36	SetMaxForce	436
10.82.4.37	SetModel	437
10.82.4.38	SetState	437
10.82.4.39	SetVelocity	437
10.82.4.40	Update	437
10.82.4.41	UpdateParameters	437
10.82.5	Member Data Documentation	438
10.82.5.1	anchorLink	438
10.82.5.2	anchorPos	438
10.82.5.3	childLink	438
10.82.5.4	damping_coefficient	438
10.82.5.5	model	438
10.82.5.6	parentLink	438
10.83	gazebo::physics::JointController Class Reference	438
10.83.1	Detailed Description	439
10.83.2	Constructor & Destructor Documentation	439
10.83.2.1	JointController	439
10.83.3	Member Function Documentation	439
10.83.3.1	AddJoint	439
10.83.3.2	Reset	439
10.83.3.3	SetJointPosition	439
10.83.3.4	SetJointPosition	439
10.83.3.5	SetJointPositions	440
10.83.3.6	Update	440
10.84	gazebo::physics::JointFeedback Class Reference	440
10.84.1	Detailed Description	441
10.84.2	Member Function Documentation	441
10.84.2.1	operator=	441
10.84.3	Member Data Documentation	441
10.84.3.1	body1Force	441
10.84.3.2	body1Torque	441

10.84.3.3	body2Force	441
10.84.3.4	body2Torque	441
10.85	gazebo::physics::JointState Class Reference	442
10.85.1	Detailed Description	442
10.85.2	Constructor & Destructor Documentation	442
10.85.2.1	JointState	442
10.85.2.2	JointState	443
10.85.2.3	~JointState	443
10.85.3	Member Function Documentation	443
10.85.3.1	GetAngle	443
10.85.3.2	GetAngleCount	443
10.85.3.3	Load	443
10.86	gazebo::rendering::JointVisual Class Reference	444
10.86.1	Detailed Description	444
10.86.2	Constructor & Destructor Documentation	444
10.86.2.1	JointVisual	444
10.86.2.2	~JointVisual	445
10.86.3	Member Function Documentation	445
10.86.3.1	Load	445
10.87	gazebo::common::KeyFrame Class Reference	445
10.87.1	Detailed Description	446
10.87.2	Constructor & Destructor Documentation	446
10.87.2.1	KeyFrame	446
10.87.2.2	~KeyFrame	446
10.87.3	Member Function Documentation	446
10.87.3.1	GetTime	446
10.87.4	Member Data Documentation	446
10.87.4.1	time	446
10.88	gazebo::rendering::LaserVisual Class Reference	447
10.88.1	Detailed Description	447
10.88.2	Constructor & Destructor Documentation	447
10.88.2.1	LaserVisual	448
10.88.2.2	~LaserVisual	448
10.88.3	Member Function Documentation	448
10.88.3.1	SetEmissive	448
10.89	gazebo::rendering::Light Class Reference	448
10.89.1	Detailed Description	450

10.89.2 Constructor & Destructor Documentation	450
10.89.2.1 Light	450
10.89.2.2 ~Light	450
10.89.3 Member Function Documentation	450
10.89.3.1 FillMsg	450
10.89.3.2 GetDiffuseColor	450
10.89.3.3 GetDirection	450
10.89.3.4 GetName	450
10.89.3.5 GetPosition	451
10.89.3.6 GetSpecularColor	451
10.89.3.7 GetType	451
10.89.3.8 Load	451
10.89.3.9 Load	451
10.89.3.10 LoadFromMsg	451
10.89.3.11 OnPoseChange	452
10.89.3.12 SetAttenuation	452
10.89.3.13 SetCastShadows	452
10.89.3.14 SetDiffuseColor	452
10.89.3.15 SetDirection	452
10.89.3.16 SetLightType	452
10.89.3.17 SetName	453
10.89.3.18 SetPosition	453
10.89.3.19 SetRange	453
10.89.3.20 SetSelected	453
10.89.3.21 SetSpecularColor	453
10.89.3.22 SetSpotFalloff	453
10.89.3.23 SetSpotInnerAngle	454
10.89.3.24 SetSpotOuterAngle	454
10.89.3.25 ShowVisual	454
10.89.3.26 ToggleShowVisual	454
10.89.3.27 UpdateFromMsg	454
10.90 gazebo::physics::Link Class Reference	454
10.90.1 Detailed Description	459
10.90.2 Constructor & Destructor Documentation	459
10.90.2.1 Link	459
10.90.2.2 ~Link	459
10.90.3 Member Function Documentation	459

10.90.3.1 AddChildJoint	459
10.90.3.2 AddForce	459
10.90.3.3 AddForceAtRelativePosition	460
10.90.3.4 AddForceAtWorldPosition	460
10.90.3.5 AddParentJoint	460
10.90.3.6 AddRelativeForce	460
10.90.3.7 AddRelativeTorque	460
10.90.3.8 AddTorque	461
10.90.3.9 AttachStaticModel	461
10.90.3.10 ConnectEnabled	461
10.90.3.11 DetachAllStaticModels	461
10.90.3.12 DetachStaticModel	461
10.90.3.13 DisconnectEnabled	461
10.90.3.14 FillLinkMsg	462
10.90.3.15 FillMsg	462
10.90.3.16 Fini	462
10.90.3.17 GetAngularDamping	462
10.90.3.18 GetBoundingBox	462
10.90.3.19 GetChildJointsLinks	462
10.90.3.20 GetCollision	463
10.90.3.21 GetCollision	463
10.90.3.22 GetCollisionById	463
10.90.3.23 GetEnabled	463
10.90.3.24 GetGravityMode	463
10.90.3.25 GetInertial	464
10.90.3.26 GetKinematic	464
10.90.3.27 GetLinearDamping	464
10.90.3.28 GetModel	464
10.90.3.29 GetParentJointsLinks	464
10.90.3.30 GetRelativeAngularAccel	465
10.90.3.31 GetRelativeAngularVel	465
10.90.3.32 GetRelativeForce	465
10.90.3.33 GetRelativeLinearAccel	465
10.90.3.34 GetRelativeLinearVel	465
10.90.3.35 GetRelativeTorque	466
10.90.3.36 GetSelfCollide	466
10.90.3.37 GetSensorCount	466

10.90.3.38	GetSensorName	466
10.90.3.39	GetState	466
10.90.3.40	GetWorldAngularAccel	467
10.90.3.41	GetWorldForce	467
10.90.3.42	GetWorldLinearAccel	467
10.90.3.43	GetWorldTorque	467
10.90.3.44	Init	467
10.90.3.45	Load	468
10.90.3.46	OnPoseChange	468
10.90.3.47	ProcessMsg	468
10.90.3.48	RemoveChildJoint	468
10.90.3.49	RemoveParentJoint	468
10.90.3.50	Reset	468
10.90.3.51	SetAngularAccel	469
10.90.3.52	SetAngularDamping	469
10.90.3.53	SetAngularVel	469
10.90.3.54	SetAutoDisable	469
10.90.3.55	SetCollideMode	469
10.90.3.56	SetEnabled	470
10.90.3.57	SetForce	470
10.90.3.58	SetGravityMode	470
10.90.3.59	SetInertial	470
10.90.3.60	SetKinematic	470
10.90.3.61	SetLaserRetro	470
10.90.3.62	SetLinearAccel	471
10.90.3.63	SetLinearDamping	471
10.90.3.64	SetLinearVel	471
10.90.3.65	SetSelected	471
10.90.3.66	SetSelfCollide	471
10.90.3.67	SetState	472
10.90.3.68	SetTorque	472
10.90.3.69	Update	472
10.90.3.70	UpdateMass	472
10.90.3.71	UpdateParameters	472
10.90.3.72	UpdateSurface	472
10.90.4	Member Data Documentation	473
10.90.4.1	angularAccel	473

10.90.4.2 attachedModelsOffset	473
10.90.4.3 cgVisuals	473
10.90.4.4 inertial	473
10.90.4.5 linearAccel	473
10.90.4.6 visuals	473
10.91 gazebo::physics::LinkState Class Reference	473
10.91.1 Detailed Description	475
10.91.2 Constructor & Destructor Documentation	475
10.91.2.1 LinkState	475
10.91.2.2 LinkState	475
10.91.2.3 ~LinkState	475
10.91.3 Member Function Documentation	475
10.91.3.1 FillStateSDF	475
10.91.3.2 GetAcceleration	475
10.91.3.3 GetCollisionState	476
10.91.3.4 GetCollisionState	476
10.91.3.5 GetCollisionStateCount	476
10.91.3.6 GetForces	476
10.91.3.7 GetPose	476
10.91.3.8 GetVelocity	477
10.91.3.9 Load	477
10.91.3.10 UpdateLinkSDF	477
10.92 gazebo::physics::Logger Class Reference	477
10.92.1 Detailed Description	478
10.92.2 Member Function Documentation	478
10.92.2.1 Add	478
10.92.2.2 Remove	479
10.93 gazebo::physics::Logplay Class Reference	479
10.93.1 Detailed Description	480
10.93.2 Member Function Documentation	480
10.93.2.1 Begin	480
10.93.2.2 End	480
10.93.2.3 Open	481
10.93.2.4 Play	481
10.93.2.5 Play	481
10.93.2.6 Play	482
10.93.2.7 Play	482

10.93.2.8 Play	482
10.93.2.9 Play	483
10.94gazebo::physics::MapShape Class Reference	483
10.94.1 Detailed Description	485
10.94.2 Constructor & Destructor Documentation	485
10.94.2.1 MapShape	485
10.94.2.2 ~MapShape	485
10.94.3 Member Function Documentation	485
10.94.3.1 FillMsg	485
10.94.3.2 GetGranularity	485
10.94.3.3 GetHeight	486
10.94.3.4 GetScale	486
10.94.3.5 GetThreshold	486
10.94.3.6 GetURI	486
10.94.3.7 Init	486
10.94.3.8 Load	486
10.94.3.9 ProcessMsg	487
10.94.3.10Update	487
10.95gazebo::Master Class Reference	487
10.95.1 Detailed Description	488
10.95.2 Constructor & Destructor Documentation	488
10.95.2.1 Master	488
10.95.2.2 ~Master	488
10.95.3 Member Function Documentation	488
10.95.3.1 Fini	488
10.95.3.2 Init	488
10.95.3.3 Run	488
10.95.3.4 RunOnce	488
10.95.3.5 RunThread	488
10.95.3.6 Stop	488
10.96gazebo::common::Material Class Reference	489
10.96.1 Detailed Description	491
10.96.2 Member Enumeration Documentation	491
10.96.2.1 BlendMode	491
10.96.2.2 ShadeMode	491
10.96.3 Constructor & Destructor Documentation	492
10.96.3.1 Material	492

10.96.3.2 ~Material	492
10.96.3.3 Material	492
10.96.4 Member Function Documentation	492
10.96.4.1 GetAmbient	492
10.96.4.2 GetBlendFactors	492
10.96.4.3 GetBlendMode	492
10.96.4.4 GetDepthWrite	492
10.96.4.5 GetDiffuse	493
10.96.4.6 GetEmissive	493
10.96.4.7 GetLighting	493
10.96.4.8 GetName	493
10.96.4.9 GetPointSize	493
10.96.4.10GetShadeMode	493
10.96.4.11GetShininess	494
10.96.4.12GetSpecular	494
10.96.4.13GetTextureImage	494
10.96.4.14GetTransparency	494
10.96.4.15SetAmbient	494
10.96.4.16SetBlendFactors	494
10.96.4.17SetBlendMode	495
10.96.4.18SetDepthWrite	495
10.96.4.19SetDiffuse	495
10.96.4.20SetEmissive	495
10.96.4.21SetLighting	495
10.96.4.22SetPointSize	495
10.96.4.23SetShadeMode	496
10.96.4.24SetShininess	496
10.96.4.25SetSpecular	496
10.96.4.26SetTextureImage	496
10.96.4.27SetTextureImage	496
10.96.4.28SetTransparency	496
10.96.5 Friends And Related Function Documentation	496
10.96.5.1 operator<<	497
10.96.6 Member Data Documentation	497
10.96.6.1 ambient	497
10.96.6.2 blendMode	497
10.96.6.3 BlendModeStr	497

10.96.6.4 diffuse	497
10.96.6.5 emissive	497
10.96.6.6 name	497
10.96.6.7 pointSize	497
10.96.6.8 shadeMode	497
10.96.6.9 ShadeModeStr	497
10.96.6.10 shininess	497
10.96.6.11 specular	497
10.96.6.12 texImage	498
10.96.6.13 transparency	498
10.97 gazebo::math::Matrix3 Class Reference	498
10.97.1 Detailed Description	499
10.97.2 Constructor & Destructor Documentation	499
10.97.2.1 Matrix3	499
10.97.2.2 Matrix3	499
10.97.2.3 Matrix3	499
10.97.2.4 ~Matrix3	499
10.97.3 Member Function Documentation	499
10.97.3.1 operator==	500
10.97.3.2 operator[]	500
10.97.3.3 operator[]	500
10.97.3.4 SetCol	500
10.97.3.5 SetFromAxes	501
10.97.3.6 SetFromAxis	501
10.97.4 Friends And Related Function Documentation	501
10.97.4.1 operator<<	501
10.97.5 Member Data Documentation	501
10.97.5.1 m	501
10.98 gazebo::math::Matrix4 Class Reference	501
10.98.1 Detailed Description	503
10.98.2 Constructor & Destructor Documentation	503
10.98.2.1 Matrix4	503
10.98.2.2 Matrix4	503
10.98.2.3 Matrix4	503
10.98.2.4 ~Matrix4	504
10.98.3 Member Function Documentation	504
10.98.3.1 GetAsPose	504

10.98.3.2	GetEulerRotation	504
10.98.3.3	GetRotation	504
10.98.3.4	GetTranslation	504
10.98.3.5	Inverse	505
10.98.3.6	IsAffine	505
10.98.3.7	operator*	505
10.98.3.8	operator*	505
10.98.3.9	operator*	505
10.98.3.10	operator=	506
10.98.3.11	operator=	506
10.98.3.12	operator==	506
10.98.3.13	operator[]	506
10.98.3.14	operator[]	507
10.98.3.15	Set	507
10.98.3.16	SetScale	507
10.98.3.17	SetTranslate	507
10.98.3.18	TransformAffine	508
10.98.4	Friends And Related Function Documentation	508
10.98.4.1	operator<<	508
10.98.5	Member Data Documentation	508
10.98.5.1	IDENTITY	508
10.98.5.2	m	508
10.98.5.3	ZERO	508
10.99	gazebo::common::Mesh Class Reference	509
10.99.1	Detailed Description	510
10.99.2	Constructor & Destructor Documentation	510
10.99.2.1	Mesh	510
10.99.2.2	~Mesh	510
10.99.3	Member Function Documentation	510
10.99.3.1	AddMaterial	510
10.99.3.2	AddSubMesh	510
10.99.3.3	FillArrays	511
10.99.3.4	GenSphericalTexCoord	511
10.99.3.5	GetAABB	511
10.99.3.6	GetIndexCount	511
10.99.3.7	GetMaterial	511
10.99.3.8	GetMaterialCount	512

10.99.3.9	GetMax	512
10.99.3.10	GetMin	512
10.99.3.11	GetName	512
10.99.3.12	GetNormalCount	512
10.99.3.13	GetPath	512
10.99.3.14	GetSkeleton	513
10.99.3.15	GetSubMesh	513
10.99.3.16	GetSubMeshCount	513
10.99.3.17	GetTexCoordCount	513
10.99.3.18	GetVertexCount	513
10.99.3.19	HasSkeleton	513
10.99.3.20	RecalculateNormals	514
10.99.3.21	Scale	514
10.99.3.22	SetName	514
10.99.3.23	SetPath	514
10.99.3.24	SetSkeleton	514
10.100	gazebo::common::MeshLoader Class Reference	514
10.100.1	Detailed Description	515
10.100.2	Constructor & Destructor Documentation	515
10.100.2.1	MeshLoader	515
10.100.2.2	~MeshLoader	515
10.100.3	Member Function Documentation	515
10.100.3.1	Load	515
10.101	gazebo::common::MeshManager Class Reference	516
10.101.1	Detailed Description	517
10.101.2	Member Function Documentation	517
10.101.2.1	AddMesh	517
10.101.2.2	CreateBox	517
10.101.2.3	CreateCamera	518
10.101.2.4	CreateCone	518
10.101.2.5	CreateCylinder	518
10.101.2.6	CreatePlane	518
10.101.2.7	CreatePlane	519
10.101.2.8	CreateSphere	519
10.101.2.9	CreateTube	519
10.101.2.10	GenSphericalTexCoord	519
10.101.2.11	GetMesh	519

10.101.2.10	GetMeshAABB	520
10.101.2.11	HasMesh	520
10.101.2.12	IsValidFilename	520
10.101.2.13	Load	520
10.102	Gazebo::physics::Model Class Reference	521
10.102.1	Detailed Description	524
10.102.2	Constructor & Destructor Documentation	524
10.102.2.1	Model	524
10.102.2.2	~Model	524
10.102.3	Member Function Documentation	524
10.102.3.1	AttachStaticModel	524
10.102.3.2	DetachStaticModel	524
10.102.3.3	FillModelMsg	525
10.102.3.4	FillMsg	525
10.102.3.5	Finis	525
10.102.3.6	GetAllLinks	525
10.102.3.7	GetBoundingBox	525
10.102.3.8	GetJoint	525
10.102.3.9	GetJoint	526
10.102.3.10	GetJointCount	526
10.102.3.11	GetLink	526
10.102.3.12	GetLink	526
10.102.3.13	GetLinkById	527
10.102.3.14	GetRelativeAngularAccel	527
10.102.3.15	GetRelativeAngularVel	527
10.102.3.16	GetRelativeLinearAccel	527
10.102.3.17	GetRelativeLinearVel	527
10.102.3.18	GetSDF	528
10.102.3.19	GetState	528
10.102.3.20	GetWorldAngularAccel	528
10.102.3.21	GetWorldAngularVel	528
10.102.3.22	GetWorldLinearAccel	528
10.102.3.23	GetWorldLinearVel	529
10.102.3.24	Init	529
10.102.3.25	Load	529
10.102.3.26	LoadPlugins	529
10.102.3.27	OnPoseChange	529

10.102.3.28	ProcessMsg	529
10.102.3.29	RemoveChild	529
10.102.3.30	Reset	530
10.102.3.31	SetAngularAccel	530
10.102.3.32	SetAngularVel	530
10.102.3.33	SetAutoDisable	530
10.102.3.34	SetCollideMode	530
10.102.3.35	SetEnabled	530
10.102.3.36	SetGravityMode	531
10.102.3.37	SetJointAnimation	531
10.102.3.38	SetJointPosition	531
10.102.3.39	SetJointPositions	531
10.102.3.40	SetLaserRetro	532
10.102.3.41	SetLinearAccel	532
10.102.3.42	SetLinearVel	532
10.102.3.43	SetLinkWorldPose	532
10.102.3.44	SetLinkWorldPose	532
10.102.3.45	SetState	532
10.102.3.46	StopAnimation	533
10.102.3.47	Update	533
10.102.3.48	UpdateParameters	533
10.102.4	Member Data Documentation	533
10.102.4.1	attachedModels	533
10.102.4.2	attachedModelsOffset	533
10.103	gazebo::common::ModelDatabase Class Reference	533
10.103.1	Detailed Description	534
10.104	gazebo::ModelPlugin Class Reference	534
10.104.1	Detailed Description	535
10.104.2	Constructor & Destructor Documentation	535
10.104.2.1	ModelPlugin	535
10.104.2.2	~ModelPlugin	535
10.104.3	Member Function Documentation	535
10.104.3.1	Init	535
10.104.3.2	Load	535
10.104.3.3	Reset	536
10.105	gazebo::physics::ModelState Class Reference	536
10.105.1	Detailed Description	537

10.105.2	Constructor & Destructor Documentation	537
10.105.2.1	ModelState	537
10.105.2.2	ModelState	537
10.105.2.3	~ModelState	537
10.105.3	Member Function Documentation	537
10.105.3.1	FillStateSDF	537
10.105.3.2	GetJointState	538
10.105.3.3	GetJointState	538
10.105.3.4	GetJointStateCount	538
10.105.3.5	GetLinkState	538
10.105.3.6	GetLinkState	539
10.105.3.7	GetLinkStateCount	539
10.105.3.8	GetPose	539
10.105.3.9	Load	539
10.105.3.10	UpdateModelSDF	539
10.106	gazebo::common::MouseEvent Class Reference	540
10.106.1	Detailed Description	541
10.106.2	Member Enumeration Documentation	541
10.106.2.1	Buttons	541
10.106.2.2	EventType	541
10.106.3	Constructor & Destructor Documentation	541
10.106.3.1	MouseEvent	541
10.106.4	Member Data Documentation	542
10.106.4.1	alt	542
10.106.4.2	button	542
10.106.4.3	buttons	542
10.106.4.4	control	542
10.106.4.5	dragging	542
10.106.4.6	moveScale	542
10.106.4.7	pos	542
10.106.4.8	pressPos	542
10.106.4.9	prevPos	542
10.106.4.10	scroll	542
10.106.4.11	shift	542
10.106.4.12	type	543
10.107	gazebo::rendering::MovableText Class Reference	543
10.107.1	Detailed Description	544

10.107.2	Member Enumeration Documentation	545
10.107.2.1	HorizAlign	545
10.107.2.2	VertAlign	545
10.107.3	Constructor & Destructor Documentation	545
10.107.3.1	MovableText	545
10.107.3.2	~MovableText	545
10.107.4	Member Function Documentation	545
10.107.4.1	_setupGeometry	545
10.107.4.2	_updateColors	545
10.107.4.3	GetAABB	545
10.107.4.4	GetBaseline	545
10.107.4.5	getBoundingRadius	546
10.107.4.6	GetCharHeight	546
10.107.4.7	GetColor	546
10.107.4.8	GetFont	546
10.107.4.9	getLights	546
10.107.4.10	GetMaterial	546
10.107.4.11	getRenderOperation	546
10.107.4.12	GetShowOnTop	546
10.107.4.13	GetSpaceWidth	546
10.107.4.14	getSquaredViewDepth	546
10.107.4.15	GetText	546
10.107.4.16	getWorldTransforms	547
10.107.4.17	Load	547
10.107.4.18	SetBaseline	547
10.107.4.19	SetCharHeight	547
10.107.4.20	SetColor	547
10.107.4.21	SetFontName	547
10.107.4.22	SetShowOnTop	548
10.107.4.23	SetSpaceWidth	548
10.107.4.24	SetText	548
10.107.4.25	SetTextAlignment	548
10.107.4.26	Update	548
10.107.4.27	visitRenderables	548
10.108	Gazebo::physics::MultiRayShape Class Reference	549
10.108.1	Detailed Description	551
10.108.2	Constructor & Destructor Documentation	551

10.108.2.1MultiRayShape	551
10.108.2.2~MultiRayShape	551
10.108.3Member Function Documentation	551
10.108.3.1AddRay	551
10.108.3.2ConnectNewLaserScans	552
10.108.3.3DisconnectNewLaserScans	552
10.108.3.4FillMsg	552
10.108.3.5GetFiducial	552
10.108.3.6GetMaxAngle	553
10.108.3.7GetMaxRange	553
10.108.3.8GetMinAngle	553
10.108.3.9GetMinRange	553
10.108.3.10GetRange	553
10.108.3.11GetResRange	554
10.108.3.12GetRetro	554
10.108.3.13GetSampleCount	554
10.108.3.14GetScanResolution	554
10.108.3.15GetVerticalMaxAngle	554
10.108.3.16GetVerticalMinAngle	554
10.108.3.17GetVerticalSampleCount	555
10.108.3.18GetVerticalScanResolution	555
10.108.3.19Init	555
10.108.3.20ProcessMsg	555
10.108.3.21Update	555
10.108.3.22UpdateRays	555
10.108.4Member Data Documentation	555
10.108.4.1horzElem	556
10.108.4.2newLaserScans	556
10.108.4.3offset	556
10.108.4.4rangeElem	556
10.108.4.5rayElem	556
10.108.4.6rays	556
10.108.4.7scanElem	556
10.108.4.8vertElem	556
10.109gazebo::transport::Node Class Reference	556
10.109.1Detailed Description	558
10.109.2Constructor & Destructor Documentation	558

10.109.2.1	Node	558
10.109.2.2	~Node	558
10.109.3	Member Function Documentation	558
10.109.3.1	Advertise	558
10.109.3.2	DecodeTopicName	558
10.109.3.3	EncodeTopicName	558
10.109.3.4	Finis	558
10.109.3.5	GetId	558
10.109.3.6	GetMsgType	558
10.109.3.7	GetTopicNamespace	559
10.109.3.8	HandleData	559
10.109.3.9	Init	559
10.109.3.10	InsertLatchedMsg	559
10.109.3.11	ProcessIncoming	559
10.109.3.12	ProcessPublishers	559
10.109.3.13	Subscribe	559
10.109.3.14	Subscribe	559
10.110	Gazebo::common::NodeAnimation Class Reference	560
10.110.1	Detailed Description	561
10.110.2	Constructor & Destructor Documentation	561
10.110.2.1	NodeAnimation	561
10.110.2.2	~NodeAnimation	561
10.110.3	Member Function Documentation	561
10.110.3.1	AddKeyFrame	561
10.110.3.2	AddKeyFrame	561
10.110.3.3	GetFrameAt	561
10.110.3.4	GetFrameCount	562
10.110.3.5	GetKeyFrame	562
10.110.3.6	GetKeyFrame	562
10.110.3.7	GetLength	562
10.110.3.8	GetName	562
10.110.3.9	GetTimeAtX	563
10.110.3.10	Scale	563
10.110.3.11	SetName	563
10.110.4	Member Data Documentation	563
10.110.4.1	keyFrames	563
10.110.4.2	length	563

10.110.4.3name	563
10.110 gazebo::common::NodeAssignment Struct Reference	563
10.111.1Detailed Description	564
10.111.2Member Data Documentation	564
10.111.2.1nodeIndex	564
10.111.2.2vertexIndex	564
10.111.2.3weight	564
10.110 gazebo::common::NodeTransform Class Reference	564
10.112.1Detailed Description	566
10.112.2Member Enumeration Documentation	566
10.112.2.1TransformType	566
10.112.3Constructor & Destructor Documentation	566
10.112.3.1NodeTransform	566
10.112.3.2NodeTransform	566
10.112.3.3~NodeTransform	566
10.112.4Member Function Documentation	566
10.112.4.1Get	566
10.112.4.2GetSID	567
10.112.4.3GetType	567
10.112.4.4operator()	567
10.112.4.5operator*	567
10.112.4.6operator*	567
10.112.4.7PrintSource	568
10.112.4.8RecalculateMatrix	568
10.112.4.9Set	568
10.112.4.10SetComponent	568
10.112.4.11SetSID	568
10.112.4.12SetSourceValues	568
10.112.4.13SetSourceValues	568
10.112.4.14SetSourceValues	569
10.112.4.15SetType	569
10.112.5Member Data Documentation	569
10.112.5.1sid	569
10.112.5.2source	569
10.112.5.3transform	569
10.112.5.4type	569
10.110 gazebo::common::NumericAnimation Class Reference	569

10.113.1	Detailed Description	570
10.113.2	Constructor & Destructor Documentation	570
10.113.2.1	NumericAnimation	570
10.113.2.2	~NumericAnimation	570
10.113.3	Member Function Documentation	571
10.113.3.1	CreateKeyFrame	571
10.113.3.2	GetInterpolatedKeyFrame	571
10.114	gazebo::common::NumericKeyFrame Class Reference	571
10.114.1	Detailed Description	572
10.114.2	Constructor & Destructor Documentation	572
10.114.2.1	NumericKeyFrame	572
10.114.2.2	~NumericKeyFrame	572
10.114.3	Member Function Documentation	572
10.114.3.1	GetValue	572
10.114.3.2	SetValue	572
10.114.4	Member Data Documentation	573
10.114.4.1	value	573
10.115	gazebo::physics::ODEBallJoint Class Reference	573
10.115.1	Detailed Description	575
10.115.2	Constructor & Destructor Documentation	575
10.115.2.1	ODEBallJoint	575
10.115.2.2	~ODEBallJoint	575
10.115.3	Member Function Documentation	575
10.115.3.1	GetAnchor	575
10.115.3.2	GetAngleImpl	575
10.115.3.3	GetGlobalAxis	575
10.115.3.4	GetMaxForce	576
10.115.3.5	GetVelocity	576
10.115.3.6	SetAnchor	576
10.115.3.7	SetDamping	576
10.115.3.8	SetMaxForce	576
10.115.3.9	SetVelocity	576
10.116	gazebo::physics::ODEBoxShape Class Reference	576
10.116.1	Detailed Description	577
10.116.2	Constructor & Destructor Documentation	577
10.116.2.1	ODEBoxShape	577
10.116.2.2	~ODEBoxShape	578

10.116.3	Member Function Documentation	578
10.116.3.1	setSize	578
10.117	<code>gazebo::physics::ODECollision</code> Class Reference	578
10.117.1	Detailed Description	580
10.117.2	Constructor & Destructor Documentation	580
10.117.2.1	<code>ODECollision</code>	580
10.117.2.2	<code>~ODECollision</code>	580
10.117.3	Member Function Documentation	580
10.117.3.1	<code>Finis</code>	580
10.117.3.2	<code>GetBoundingBox</code>	580
10.117.3.3	<code>GetCollisionClass</code>	581
10.117.3.4	<code>GetCollisionId</code>	581
10.117.3.5	<code>GetSpaceld</code>	581
10.117.3.6	<code>Load</code>	581
10.117.3.7	<code>OnPoseChange</code>	581
10.117.3.8	<code>SetCategoryBits</code>	581
10.117.3.9	<code>SetCollideBits</code>	581
10.117.3.10	<code>SetCollision</code>	582
10.117.3.11	<code>SetSpaceld</code>	582
10.117.4	Member Data Documentation	582
10.117.4.1	<code>collisionId</code>	582
10.117.4.2	<code>spaceld</code>	582
10.118	<code>gazebo::physics::ODECylinderShape</code> Class Reference	582
10.118.1	Detailed Description	583
10.118.2	Constructor & Destructor Documentation	584
10.118.2.1	<code>ODECylinderShape</code>	584
10.118.2.2	<code>~ODECylinderShape</code>	584
10.118.3	Member Function Documentation	584
10.118.3.1	<code>setSize</code>	584
10.119	<code>gazebo::physics::ODEHeightmapShape</code> Class Reference	584
10.119.1	Detailed Description	585
10.119.2	Constructor & Destructor Documentation	586
10.119.2.1	<code>ODEHeightmapShape</code>	586
10.119.2.2	<code>~ODEHeightmapShape</code>	586
10.119.3	Member Function Documentation	586
10.119.3.1	<code>Init</code>	586
10.120	<code>gazebo::physics::ODEHinge2Joint</code> Class Reference	586

10.120.1	Detailed Description	588
10.120.2	Constructor & Destructor Documentation	588
10.120.2.1	ODEHinge2Joint	588
10.120.2.2	~ODEHinge2Joint	588
10.120.3	Member Function Documentation	589
10.120.3.1	GetAnchor	589
10.120.3.2	GetAngleImpl	589
10.120.3.3	GetGlobalAxis	589
10.120.3.4	GetMaxForce	589
10.120.3.5	GetParam	589
10.120.3.6	GetVelocity	589
10.120.3.7	Load	589
10.120.3.8	SetAnchor	589
10.120.3.9	SetAxis	590
10.120.3.10	SetDamping	590
10.120.3.11	SetForce	590
10.120.3.12	SetMaxForce	590
10.120.3.13	SetParam	590
10.120.3.14	SetVelocity	590
10.121	Gazebo::physics::ODEHingeJoint Class Reference	590
10.121.1	Detailed Description	592
10.121.2	Constructor & Destructor Documentation	592
10.121.2.1	ODEHingeJoint	592
10.121.2.2	~ODEHingeJoint	592
10.121.3	Member Function Documentation	593
10.121.3.1	ApplyDamping	593
10.121.3.2	GetAnchor	593
10.121.3.3	GetAngleImpl	593
10.121.3.4	GetGlobalAxis	593
10.121.3.5	GetMaxForce	593
10.121.3.6	GetParam	593
10.121.3.7	GetVelocity	593
10.121.3.8	Load	593
10.121.3.9	SetAnchor	593
10.121.3.10	SetAxis	594
10.121.3.11	SetDamping	594
10.121.3.12	SetForce	594

10.121.3.1	SetMaxForce	594
10.121.3.1	SetParam	594
10.121.3.1	SetVelocity	594
10.122	Gazebo::physics::ODEJoint Class Reference	594
10.122.1	Detailed Description	596
10.122.2	Constructor & Destructor Documentation	596
10.122.2.1	ODEJoint	596
10.122.2.2	~ODEJoint	596
10.122.3	Member Function Documentation	596
10.122.3.1	AreConnected	596
10.122.3.2	Attach	596
10.122.3.3	Detach	597
10.122.3.4	GetCFM	597
10.122.3.5	GetERP	597
10.122.3.6	GetFeedback	597
10.122.3.7	GetHighStop	597
10.122.3.8	GetJointLink	597
10.122.3.9	GetLinkForce	597
10.122.3.10	GetLinkTorque	597
10.122.3.10	GetLowStop	598
10.122.3.10	GetParam	598
10.122.3.10	Load	598
10.122.3.10	Reset	598
10.122.3.10	SetAttribute	598
10.122.3.10	SetCFM	598
10.122.3.10	SetERP	598
10.122.3.10	SetHighStop	598
10.122.3.10	SetLowStop	599
10.122.3.10	SetParam	599
10.122.4	Member Data Documentation	599
10.122.4.1	jointId	599
10.123	Gazebo::physics::ODELink Class Reference	599
10.123.1	Detailed Description	602
10.123.2	Constructor & Destructor Documentation	602
10.123.2.1	ODELink	602
10.123.2.2	~ODELink	602
10.123.3	Member Function Documentation	602

10.123.3.1AddForce	602
10.123.3.2AddForceAtRelativePosition	603
10.123.3.3AddForceAtWorldPosition	603
10.123.3.4AddRelativeForce	603
10.123.3.5AddRelativeTorque	603
10.123.3.6AddTorque	603
10.123.3.7DisabledCallback	603
10.123.3.8Fini	603
10.123.3.9GetEnabled	603
10.123.3.10GetGravityMode	603
10.123.3.11GetKinematic	604
10.123.3.12GetODEId	604
10.123.3.13GetSpaceId	604
10.123.3.14GetWorldAngularVel	604
10.123.3.15GetWorldForce	604
10.123.3.16GetWorldLinearVel	604
10.123.3.17GetWorldTorque	604
10.123.3.18Hit	604
10.123.3.19Load	605
10.123.3.20MoveCallback	605
10.123.3.21OnPoseChange	605
10.123.3.22SetAngularDamping	605
10.123.3.23SetAngularVel	605
10.123.3.24SetAutoDisable	605
10.123.3.25SetEnabled	605
10.123.3.26SetForce	605
10.123.3.27SetGravityMode	606
10.123.3.28SetKinematic	606
10.123.3.29SetLinearDamping	606
10.123.3.30SetLinearVel	606
10.123.3.31SetSelfCollide	606
10.123.3.32SetSpaceId	606
10.123.3.33SetTorque	606
10.123.3.34Update	606
10.123.3.35UpdateMass	607
10.123.3.36UpdateSurface	607
10.123.4Member Data Documentation	607

10.123.4.1	pose	607
10.124	gazebo::physics::ODEMultiRayShape Class Reference	607
10.124.1	Detailed Description	609
10.124.2	Constructor & Destructor Documentation	609
10.124.2.1	ODEMultiRayShape	609
10.124.2.2	~ODEMultiRayShape	609
10.124.3	Member Function Documentation	609
10.124.3.1	AddRay	609
10.124.3.2	UpdateRays	609
10.125	gazebo::physics::ODEPhysics Class Reference	609
10.125.1	Detailed Description	612
10.125.2	Constructor & Destructor Documentation	612
10.125.2.1	ODEPhysics	612
10.125.2.2	~ODEPhysics	612
10.125.3	Member Function Documentation	612
10.125.3.1	Collide	612
10.125.3.2	ConvertMass	613
10.125.3.3	ConvertMass	613
10.125.3.4	CreateCollision	613
10.125.3.5	CreateContact	613
10.125.3.6	CreateJoint	613
10.125.3.7	CreateLink	613
10.125.3.8	CreateShape	613
10.125.3.9	DebugPrint	613
10.125.3.10	Ini	613
10.125.3.11	GetContactMaxCorrectingVel	614
10.125.3.12	GetContactSurfaceLayer	614
10.125.3.13	GetGravity	614
10.125.3.14	GetMaxContacts	614
10.125.3.15	GetSORPGSIters	614
10.125.3.16	GetSORPGSPreconIters	614
10.125.3.17	GetSORPGSW	614
10.125.3.18	GetSpaceId	614
10.125.3.19	GetStepTime	614
10.125.3.20	GetStepType	615
10.125.3.21	GetWorldCFM	615
10.125.3.22	GetWorldERP	615

10.125.3.23	SetWorldId	615
10.125.3.24	Wait	615
10.125.3.25	WaitForThread	615
10.125.3.26	Load	615
10.125.3.27	OnPhysicsMsg	615
10.125.3.28	OnRequest	615
10.125.3.29	ProcessContactFeedback	616
10.125.3.30	Reset	616
10.125.3.31	SetContactMaxCorrectingVel	616
10.125.3.32	SetContactSurfaceLayer	616
10.125.3.33	SetGravity	616
10.125.3.34	SetMaxContacts	616
10.125.3.35	SetSORPGSIters	616
10.125.3.36	SetSORPGSPreconIters	616
10.125.3.37	SetSORPGSW	616
10.125.3.38	SetStepTime	617
10.125.3.39	SetStepType	617
10.125.3.40	SetWorldCFM	617
10.125.3.41	SetWorldERP	617
10.125.3.42	UpdateCollision	617
10.125.3.43	UpdatePhysics	617
10.126	Gazebo::physics::ODEPlaneShape Class Reference	617
10.126.1	Detailed Description	618
10.126.2	Constructor & Destructor Documentation	619
10.126.2.1	ODEPlaneShape	619
10.126.2.2	~ODEPlaneShape	619
10.126.3	Member Function Documentation	619
10.126.3.1	CreatePlane	619
10.126.3.2	SetAltitude	619
10.127	Gazebo::physics::ODERayShape Class Reference	619
10.127.1	Detailed Description	621
10.127.2	Constructor & Destructor Documentation	621
10.127.2.1	ODERayShape	621
10.127.2.2	~ODERayShape	621
10.127.2.3	~ODERayShape	621
10.127.3	Member Function Documentation	621
10.127.3.1	GetIntersection	621

10.127.3.2	SetPoints	621
10.127.3.3	Update	621
10.128	Gazebo::physics::ODEScrewJoint Class Reference	622
10.128.1	Detailed Description	623
10.128.2	Constructor & Destructor Documentation	623
10.128.2.1	ODEScrewJoint	623
10.128.2.2	~ODEScrewJoint	623
10.128.3	Member Function Documentation	624
10.128.3.1	ApplyDamping	624
10.128.3.2	GetAngleImpl	624
10.128.3.3	GetGlobalAxis	624
10.128.3.4	GetMaxForce	624
10.128.3.5	GetParam	624
10.128.3.6	GetVelocity	624
10.128.3.7	Load	624
10.128.3.8	SetAxis	624
10.128.3.9	SetDamping	624
10.128.3.10	SetForce	625
10.128.3.11	SetMaxForce	625
10.128.3.12	SetParam	625
10.128.3.13	SetThreadPitch	625
10.128.3.14	SetVelocity	625
10.129	Gazebo::physics::ODESliderJoint Class Reference	625
10.129.1	Detailed Description	627
10.129.2	Constructor & Destructor Documentation	627
10.129.2.1	ODESliderJoint	627
10.129.2.2	~ODESliderJoint	627
10.129.3	Member Function Documentation	627
10.129.3.1	ApplyDamping	627
10.129.3.2	GetAngleImpl	628
10.129.3.3	GetGlobalAxis	628
10.129.3.4	GetMaxForce	628
10.129.3.5	GetParam	628
10.129.3.6	GetVelocity	628
10.129.3.7	Load	628
10.129.3.8	SetAxis	628
10.129.3.9	SetDamping	628

10.129.3.1	SetForce	628
10.129.3.1	SetMaxForce	629
10.129.3.1	SetParam	629
10.129.3.1	SetVelocity	629
10.130	gazebo::physics::ODESphereShape Class Reference	629
10.130.1	Detailed Description	630
10.130.2	Constructor & Destructor Documentation	630
10.130.2.1	ODESphereShape	631
10.130.2.2	~ODESphereShape	631
10.130.3	Member Function Documentation	631
10.130.3.1	SetRadius	631
10.131	gazebo::physics::ODESurfaceParams Class Reference	631
10.131.1	Detailed Description	631
10.131.2	Constructor & Destructor Documentation	631
10.131.2.1	ODESurfaceParams	631
10.131.2.2	~ODESurfaceParams	632
10.131.3	Member Function Documentation	632
10.131.3.1	Load	632
10.132	gazebo::physics::ODETrimeshShape Class Reference	632
10.132.1	Detailed Description	634
10.132.2	Constructor & Destructor Documentation	634
10.132.2.1	ODETrimeshShape	634
10.132.2.2	~ODETrimeshShape	634
10.132.3	Member Function Documentation	634
10.132.3.1	Init	634
10.132.3.2	Load	634
10.132.3.3	Update	634
10.133	gazebo::physics::ODEUniversalJoint Class Reference	634
10.133.1	Detailed Description	636
10.133.2	Constructor & Destructor Documentation	636
10.133.2.1	ODEUniversalJoint	636
10.133.2.2	~ODEUniversalJoint	636
10.133.3	Member Function Documentation	636
10.133.3.1	GetAnchor	636
10.133.3.2	GetAngleImpl	636
10.133.3.3	GetGlobalAxis	637
10.133.3.4	GetMaxForce	637

10.133.3.5	GetVelocity	637
10.133.3.6	SetAnchor	637
10.133.3.7	SetAxis	637
10.133.3.8	SetDamping	637
10.133.3.9	SetForce	637
10.133.3.10	SetMaxForce	637
10.133.3.11	SetParam	637
10.133.3.12	SetVelocity	638
10.134	gazebo::rendering::OrbitViewController Class Reference	638
10.134.1	Detailed Description	639
10.134.2	Constructor & Destructor Documentation	639
10.134.2.1	OrbitViewController	639
10.134.2.2	~OrbitViewController	639
10.134.3	Member Function Documentation	639
10.134.3.1	GetFocalPoint	640
10.134.3.2	GetTypeString	640
10.134.3.3	HandleKeyPressEvent	640
10.134.3.4	HandleKeyReleaseEvent	640
10.134.3.5	HandleMouseEvent	640
10.134.3.6	Init	640
10.134.3.7	Init	641
10.134.3.8	SetDistance	641
10.134.3.9	SetDistanceRange	641
10.134.3.10	SetFocalPoint	641
10.134.3.11	SetPitch	641
10.134.3.12	SetYaw	641
10.134.3.13	Update	642
10.135	df::Param Class Reference	642
10.135.1	Detailed Description	644
10.135.2	Constructor & Destructor Documentation	644
10.135.2.1	Param	644
10.135.2.2	~Param	644
10.135.3	Member Function Documentation	644
10.135.3.1	Clone	644
10.135.3.2	Get	644
10.135.3.3	Get	644
10.135.3.4	Get	644

10.135.3.5Get	644
10.135.3.6Get	644
10.135.3.7Get	644
10.135.3.8Get	645
10.135.3.9Get	645
10.135.3.10Get	645
10.135.3.11Get	645
10.135.3.12Get	645
10.135.3.13Get	645
10.135.3.14Get	645
10.135.3.15Get	645
10.135.3.16GetAsString	645
10.135.3.17GetDefaultAsString	645
10.135.3.18GetDescription	645
10.135.3.19GetKey	645
10.135.3.20GetRequired	645
10.135.3.21GetSet	645
10.135.3.22GetTypeName	645
10.135.3.23Bool	645
10.135.3.24Char	645
10.135.3.25Color	646
10.135.3.26Double	646
10.135.3.27Float	646
10.135.3.28Int	646
10.135.3.29Pose	646
10.135.3.30Quaternion	646
10.135.3.31Str	646
10.135.3.32Time	646
10.135.3.33UInt	646
10.135.3.34Vector2d	646
10.135.3.35Vector2i	646
10.135.3.36Vector3	646
10.135.3.37reset	646
10.135.3.38Set	646
10.135.3.39Set	646
10.135.3.40Set	646
10.135.3.41Set	646

10.135.3.42	Set	646
10.135.3.43	Set	646
10.135.3.44	Set	646
10.135.3.45	Set	646
10.135.3.46	Set	646
10.135.3.47	Set	646
10.135.3.48	Set	647
10.135.3.49	Set	647
10.135.3.50	Set	647
10.135.3.51	Set	647
10.135.3.52	Set	647
10.135.3.53	Set	647
10.135.3.54	SetDescription	647
10.135.3.55	SetFromString	647
10.135.3.56	SetUpdateFunc	647
10.135.3.57	update	647
10.135.4	Member Data Documentation	647
10.135.4.1	description	647
10.135.4.2	key	647
10.135.4.3	required	647
10.135.4.4	set	647
10.135.4.5	typeName	648
10.135.4.6	updateFunc	648
10.137	ParamT< T > Class Template Reference	648
10.137	df::ParamT< T > Class Template Reference	648
10.137.1	Detailed Description	649
10.137.2	Constructor & Destructor Documentation	649
10.137.2.1	ParamT	649
10.137.2.2	~ParamT	649
10.137.3	Member Function Documentation	650
10.137.3.1	Clone	650
10.137.3.2	GetAsString	650
10.137.3.3	GetDefaultAsString	650
10.137.3.4	GetDefaultValue	650
10.137.3.5	GetValue	650
10.137.3.6	operator*	650
10.137.3.7	Reset	650
10.137.3.8	Set	650

10.137.3.9	SetFromString	650
10.137.3.10	SetValue	651
10.137.3.11	Update	651
10.137.4	Friends And Related Function Documentation	651
10.137.4.1	operator<<	651
10.137.5	Member Data Documentation	651
10.137.5.1	defaultValue	651
10.137.5.2	value	651
10.138	Gazebo::physics::PhysicsEngine Class Reference	651
10.138.1	Detailed Description	654
10.138.2	Constructor & Destructor Documentation	654
10.138.2.1	PhysicsEngine	654
10.138.2.2	~PhysicsEngine	654
10.138.3	Member Function Documentation	654
10.138.3.1	CreateCollision	654
10.138.3.2	CreateCollision	655
10.138.3.3	CreateJoint	655
10.138.3.4	CreateLink	655
10.138.3.5	CreateShape	655
10.138.3.6	DebugPrint	655
10.138.3.7	Finis	656
10.138.3.8	GetAutoDisableFlag	656
10.138.3.9	GetContactMaxCorrectingVel	656
10.138.3.10	GetContactSurfaceLayer	656
10.138.3.11	GetGravity	656
10.138.3.12	GetMaxContacts	656
10.138.3.13	GetPhysicsUpdateMutex	657
10.138.3.14	GetSORPGSIters	657
10.138.3.15	GetSORPGSPreconIters	657
10.138.3.16	GetSORPGSW	657
10.138.3.17	GetStepTime	658
10.138.3.18	GetUpdatePeriod	658
10.138.3.19	GetUpdateRate	658
10.138.3.20	GetWorldCFM	658
10.138.3.21	GetWorldERP	658
10.138.3.22	Init	659
10.138.3.23	InitForThread	659

10.138.3.24	load	659
10.138.3.25	onPhysicsMsg	659
10.138.3.26	onRequest	659
10.138.3.27	reset	659
10.138.3.28	setAutoDisableFlag	660
10.138.3.29	setContactMaxCorrectingVel	660
10.138.3.30	setContactSurfaceLayer	660
10.138.3.31	setGravity	660
10.138.3.32	setMaxContacts	660
10.138.3.33	setSORPGSIters	661
10.138.3.34	setSORPGSPreconIters	661
10.138.3.35	setSORPGSW	661
10.138.3.36	setStepTime	661
10.138.3.37	setUpdateRate	661
10.138.3.38	setWorldCFM	662
10.138.3.39	setWorldERP	662
10.138.3.40	updateCollision	662
10.138.3.41	updatePhysics	662
10.138.4	Member Data Documentation	662
10.138.4.1	contactPub	662
10.138.4.2	node	662
10.138.4.3	physicsSub	663
10.138.4.4	physicsUpdateMutex	663
10.138.4.5	requestSub	663
10.138.4.6	responsePub	663
10.138.4.7	sdf	663
10.138.4.8	world	663
10.139	gazebo::physics::PhysicsFactory Class Reference	663
10.139.1	Detailed Description	663
10.139.2	Member Function Documentation	664
10.139.2.1	NewPhysicsEngine	664
10.139.2.2	RegisterAll	664
10.139.2.3	RegisterPhysicsEngine	664
10.140	gazebo::common::PID Class Reference	664
10.140.1	Detailed Description	665
10.140.2	Constructor & Destructor Documentation	665
10.140.2.1	PID	665

10.140.2.2~PID	666
10.140.3 Member Function Documentation	666
10.140.3.1 GetCmd	666
10.140.3.2 GetErrors	666
10.140.3.3 Init	666
10.140.3.4 operator=	666
10.140.3.5 Reset	667
10.140.3.6 SetCmd	667
10.140.3.7 SetCmdMax	667
10.140.3.8 SetCmdMin	667
10.140.3.9 SetDGain	667
10.140.3.10 SetIGain	667
10.140.3.11 SetIMax	667
10.140.3.12 SetIMin	668
10.140.3.13 SetPGain	668
10.140.3.14 Update	668
10.141 gazebo::math::Plane Class Reference	668
10.141.1 Detailed Description	669
10.141.2 Constructor & Destructor Documentation	669
10.141.2.1 Plane	669
10.141.2.2 Plane	669
10.141.2.3 Plane	669
10.141.2.4 ~Plane	670
10.141.3 Member Function Documentation	670
10.141.3.1 Distance	670
10.141.3.2 operator=	670
10.141.3.3 Set	670
10.141.4 Member Data Documentation	670
10.141.4.1 Id	671
10.141.4.2 normal	671
10.141.4.3 size	671
10.142 gazebo::physics::PlaneShape Class Reference	671
10.142.1 Detailed Description	672
10.142.2 Constructor & Destructor Documentation	672
10.142.2.1 PlaneShape	672
10.142.2.2 ~PlaneShape	672
10.142.3 Member Function Documentation	672

10.142.3.1	CreatePlane	673
10.142.3.2	FillMsg	673
10.142.3.3	GetNormal	673
10.142.3.4	GetSize	673
10.142.3.5	Init	673
10.142.3.6	ProcessMsg	673
10.142.3.7	SetAltitude	673
10.142.3.8	SetNormal	673
10.142.3.9	SetSize	673
10.143	df::Plugin Class Reference	674
10.143.1	Constructor & Destructor Documentation	674
10.143.1.1	Plugin	674
10.143.2	Member Function Documentation	674
10.143.2.1	Clear	674
10.143.2.2	Print	674
10.143.3	Member Data Documentation	675
10.143.3.1	data	675
10.143.3.2	filename	675
10.143.3.3	name	675
10.144	gazebo::PluginT< T > Class Template Reference	675
10.144.1	Detailed Description	676
10.144.2	Member Typedef Documentation	676
10.144.2.1	TPtr	676
10.144.3	Member Function Documentation	676
10.144.3.1	Create	676
10.144.3.2	GetFilename	676
10.144.3.3	GetHandle	676
10.144.3.4	GetType	676
10.144.4	Member Data Documentation	677
10.144.4.1	filename	677
10.144.4.2	handle	677
10.144.4.3	type	677
10.145	gazebo::math::Pose Class Reference	677
10.145.1	Detailed Description	679
10.145.2	Constructor & Destructor Documentation	679
10.145.2.1	Pose	679
10.145.2.2	Pose	679

10.145.2.3	Pose	679
10.145.2.4	Pose	679
10.145.2.5	~Pose	679
10.145.3	Member Function Documentation	679
10.145.3.1	CoordPoseSolve	679
10.145.3.2	CoordPositionAdd	680
10.145.3.3	CoordPositionAdd	680
10.145.3.4	CoordPositionSub	680
10.145.3.5	CoordRotationAdd	680
10.145.3.6	CoordRotationSub	681
10.145.3.7	Correct	681
10.145.3.8	GetInverse	681
10.145.3.9	IsFinite	681
10.145.3.10	operator!=	681
10.145.3.11	operator*	682
10.145.3.12	operator+	682
10.145.3.13	operator+=	682
10.145.3.14	operator-	682
10.145.3.15	operator-=	683
10.145.3.16	operator==	683
10.145.3.17	Reset	683
10.145.3.18	RotatePositionAboutOrigin	683
10.145.3.19	Round	683
10.145.3.20	Set	684
10.145.3.21	Set	684
10.145.4	Friends And Related Function Documentation	684
10.145.4.1	operator<<	684
10.145.4.2	operator>>	684
10.145.5	Member Data Documentation	685
10.145.5.1	pos	685
10.145.5.2	rot	685
10.146	Gazebo::common::PoseAnimation Class Reference	685
10.146.1	Detailed Description	686
10.146.2	Constructor & Destructor Documentation	686
10.146.2.1	PoseAnimation	686
10.146.2.2	~PoseAnimation	686
10.146.3	Member Function Documentation	686

10.146.3.1	BuildInterpolationSplines	686
10.146.3.2	CreateKeyFrame	686
10.146.3.3	GetInterpolatedKeyFrame	687
10.146.3.4	GetInterpolatedKeyFrame	687
10.147	gazebo::common::PoseKeyFrame Class Reference	687
10.147.1	Detailed Description	688
10.147.2	Constructor & Destructor Documentation	688
10.147.2.1	PoseKeyFrame	688
10.147.2.2	~PoseKeyFrame	688
10.147.3	Member Function Documentation	688
10.147.3.1	GetRotation	688
10.147.3.2	GetTranslation	688
10.147.3.3	SetRotation	689
10.147.3.4	SetTranslation	689
10.147.4	Member Data Documentation	689
10.147.4.1	rotate	689
10.147.4.2	translate	689
10.148	gazebo::rendering::Projector Class Reference	689
10.148.1	Detailed Description	690
10.148.2	Constructor & Destructor Documentation	690
10.148.2.1	Projector	690
10.148.2.2	~Projector	690
10.148.3	Member Function Documentation	690
10.148.3.1	GetParent	690
10.148.3.2	Load	690
10.148.3.3	Load	691
10.148.3.4	Load	691
10.148.3.5	SetEnabled	691
10.148.3.6	SetTexture	691
10.148.3.7	Toggle	691
10.149	gazebo::transport::Publication Class Reference	692
10.149.1	Detailed Description	692
10.149.2	Constructor & Destructor Documentation	692
10.149.2.1	Publication	692
10.149.2.2	~Publication	693
10.149.3	Member Function Documentation	693
10.149.3.1	AddPublisher	693

10.149.3.2	AddSubscription	693
10.149.3.3	AddSubscription	693
10.149.3.4	AddTransport	693
10.149.3.5	GetCallbackCount	693
10.149.3.6	GetLocallyAdvertised	693
10.149.3.7	GetMsgType	693
10.149.3.8	GetNodeCount	693
10.149.3.9	GetRemoteSubscriptionCount	693
10.149.3.10	GetTransportCount	693
10.149.3.11	HasTransport	693
10.149.3.12	LocalPublish	693
10.149.3.13	Publish	693
10.149.3.14	RemoveSubscription	693
10.149.3.15	RemoveSubscription	694
10.149.3.16	RemoveTransport	694
10.149.3.17	SetLocallyAdvertised	694
10.150	gazebo::transport::PublicationTransport Class Reference	694
10.150.1	Detailed Description	694
10.150.2	Constructor & Destructor Documentation	694
10.150.2.1	PublicationTransport	694
10.150.2.2	~PublicationTransport	694
10.150.3	Member Function Documentation	694
10.150.3.1	AddCallback	694
10.150.3.2	Finis	694
10.150.3.3	GetConnection	695
10.150.3.4	GetMsgType	695
10.150.3.5	GetTopic	695
10.150.3.6	Init	695
10.151	gazebo::transport::Publisher Class Reference	695
10.151.1	Detailed Description	695
10.151.2	Constructor & Destructor Documentation	696
10.151.2.1	Publisher	696
10.151.2.2	~Publisher	696
10.151.3	Member Function Documentation	696
10.151.3.1	GetLatching	696
10.151.3.2	GetMsgType	696
10.151.3.3	GetOutgoingCount	696

10.151.3.4	GetPrevMsg	696
10.151.3.5	GetTopic	696
10.151.3.6	HasConnections	696
10.151.3.7	Publish	696
10.151.3.8	Publish	696
10.151.3.9	SendMessage	696
10.151.3.10	SetPublication	696
10.151.3.11	WaitForConnection	697
10.152	Gazebo::math::Quaternion Class Reference	697
10.152.1	Detailed Description	700
10.152.2	Constructor & Destructor Documentation	700
10.152.2.1	Quaternion	700
10.152.2.2	Quaternion	700
10.152.2.3	Quaternion	700
10.152.2.4	Quaternion	700
10.152.2.5	Quaternion	700
10.152.2.6	Quaternion	701
10.152.2.7	~Quaternion	701
10.152.3	Member Function Documentation	701
10.152.3.1	Correct	701
10.152.3.2	Dot	701
10.152.3.3	EulerToQuaternion	701
10.152.3.4	EulerToQuaternion	701
10.152.3.5	GetAsAxis	702
10.152.3.6	GetAsEuler	702
10.152.3.7	GetAsMatrix3	702
10.152.3.8	GetAsMatrix4	702
10.152.3.9	GetExp	702
10.152.3.10	GetInverse	702
10.152.3.11	GetLog	703
10.152.3.12	GetPitch	703
10.152.3.13	GetRoll	703
10.152.3.14	GetXAxis	703
10.152.3.15	GetYaw	703
10.152.3.16	GetYAxis	703
10.152.3.17	GetZAxis	704
10.152.3.18	Invert	704

10.152.3.18	Finite	704
10.152.3.20	Normalize	704
10.152.3.21	operator!=	704
10.152.3.22	operator*	704
10.152.3.23	operator*	705
10.152.3.24	operator*	705
10.152.3.25	operator*= operator+ operator+=	705
10.152.3.26	operator+	705
10.152.3.27	operator+=	705
10.152.3.28	operator-	706
10.152.3.29	operator-	706
10.152.3.30	operator-=	706
10.152.3.31	operator=	706
10.152.3.32	operator==	707
10.152.3.33	RotateVector	707
10.152.3.34	RotateVectorReverse	707
10.152.3.35	Round	707
10.152.3.36	Scale	707
10.152.3.37	Set	708
10.152.3.38	SetFromAxis	708
10.152.3.39	SetFromAxis	708
10.152.3.40	SetFromEuler	708
10.152.3.41	SetToIdentity	708
10.152.3.42	Slerp	709
10.152.3.43	Squad	709
10.152.4	Friends And Related Function Documentation	709
10.152.4.1	operator<<	709
10.152.4.2	operator>>	709
10.152.5	Member Data Documentation	710
10.152.5.1	w	710
10.152.5.2	x	710
10.152.5.3	y	710
10.152.5.4	z	710
10.152.6	Gazebo::math::Rand Class Reference	710
10.153.1	Detailed Description	711
10.153.2	Member Function Documentation	711
10.153.2.1	GetDbfNormal	711

10.153.2.2	GetDblUniform	711
10.153.2.3	GetIntNormal	711
10.153.2.4	GetIntUniform	711
10.154	gazebo::sensors::RaySensor Class Reference	711
10.154.1	Constructor & Destructor Documentation	713
10.154.1.1	RaySensor	713
10.154.1.2	~RaySensor	713
10.154.2	Member Function Documentation	713
10.154.2.1	Finis	713
10.154.2.2	GetAngleMax	714
10.154.2.3	GetAngleMin	714
10.154.2.4	GetAngleResolution	714
10.154.2.5	GetFiducial	714
10.154.2.6	GetLaserShape	714
10.154.2.7	GetRange	715
10.154.2.8	GetRangeCount	715
10.154.2.9	GetRangeMax	715
10.154.2.10	GetRangeMin	715
10.154.2.11	GetRangeResolution	715
10.154.2.12	GetRanges	716
10.154.2.13	GetRayCount	716
10.154.2.14	GetRetro	716
10.154.2.15	GetTopic	716
10.154.2.16	GetVerticalAngleMax	716
10.154.2.17	GetVerticalAngleMin	717
10.154.2.18	GetVerticalRangeCount	717
10.154.2.19	GetVerticalRayCount	717
10.154.2.20	hit	717
10.154.2.21	load	717
10.154.2.22	load	717
10.154.2.23	updateImpl	718
10.155	gazebo::physics::RayShape Class Reference	718
10.155.1	Detailed Description	719
10.155.2	Constructor & Destructor Documentation	720
10.155.2.1	RayShape	720
10.155.2.2	RayShape	720
10.155.2.3	~RayShape	720

10.155.3	Member Function Documentation	720
10.155.3.1	FillMsg	720
10.155.3.2	GetFiducial	720
10.155.3.3	GetGlobalPoints	720
10.155.3.4	GetIntersection	720
10.155.3.5	GetLength	721
10.155.3.6	GetRelativePoints	721
10.155.3.7	GetRetro	721
10.155.3.8	Init	721
10.155.3.9	ProcessMsg	721
10.155.3.10	SetFiducial	721
10.155.3.11	SetLength	721
10.155.3.12	SetPoints	721
10.155.3.13	SetRetro	722
10.155.3.14	Update	722
10.155.4	Member Data Documentation	722
10.155.4.1	contactFiducial	722
10.155.4.2	contactLen	722
10.155.4.3	contactRetro	722
10.155.4.4	globalEndPos	722
10.155.4.5	globalStartPos	722
10.155.4.6	relativeEndPos	722
10.155.4.7	relativeStartPos	722
10.156	Gazebo::rendering::RenderEngine Class Reference	722
10.156.1	Detailed Description	724
10.156.2	Member Enumeration Documentation	724
10.156.2.1	RenderPathType	724
10.156.3	Member Function Documentation	724
10.156.3.1	AddResourcePath	724
10.156.3.2	CreateScene	725
10.156.3.3	Fini	725
10.156.3.4	GetRenderPathType	725
10.156.3.5	GetScene	725
10.156.3.6	GetScene	725
10.156.3.7	GetSceneCount	726
10.156.3.8	Init	726
10.156.3.9	Load	726

10.156.3.1	RemoveScene	726
10.156.4	Member Data Documentation	726
10.156.4.1	dummyContext	726
10.156.4.2	dummyDisplay	726
10.156.4.3	dummyWindowId	726
10.156.4.4	root	726
10.157	gazebo::sensors::RFIDSensor Class Reference	727
10.157.1	Detailed Description	728
10.157.2	Constructor & Destructor Documentation	728
10.157.2.1	RFIDSensor	728
10.157.2.2	~RFIDSensor	728
10.157.3	Member Function Documentation	728
10.157.3.1	Fini	728
10.157.3.2	init	728
10.157.3.3	Load	728
10.157.3.4	Load	728
10.157.3.5	UpdateImpl	729
10.158	gazebo::sensors::RFIDTag Class Reference	729
10.158.1	Detailed Description	730
10.158.2	Constructor & Destructor Documentation	730
10.158.2.1	RFIDTag	730
10.158.2.2	~RFIDTag	730
10.158.3	Member Function Documentation	730
10.158.3.1	Fini	730
10.158.3.2	GetTagPose	731
10.158.3.3	init	731
10.158.3.4	Load	731
10.158.3.5	Load	731
10.158.3.6	UpdateImpl	731
10.159	gazebo::sensors::RFIDTagManager Class Reference	732
10.159.1	Detailed Description	732
10.159.2	Member Function Documentation	732
10.159.2.1	AddTaggedModel	732
10.159.2.2	GetTags	732
10.160	gazebo::rendering::RFIDTagVisual Class Reference	733
10.160.1	Detailed Description	733
10.160.2	Constructor & Destructor Documentation	734

10.160.2.1RFIDTagVisual	734
10.160.2.2~RFIDTagVisual	734
10.160gazebo::rendering::RFIDVisual Class Reference	734
10.161.1Detailed Description	735
10.161.2Constructor & Destructor Documentation	735
10.161.2.1RFIDVisual	735
10.161.2.2~RFIDVisual	736
10.162Road Class Reference	736
10.162.1Detailed Description	736
10.163gazebo::physics::Road Class Reference	736
10.163.1Detailed Description	737
10.163.2Constructor & Destructor Documentation	737
10.163.2.1Road	737
10.163.2.2~Road	737
10.163.3Member Function Documentation	737
10.163.3.1Init	737
10.163.3.2Load	737
10.164gazebo::rendering::Road2d Class Reference	737
10.164.1Constructor & Destructor Documentation	738
10.164.1.1Road2d	738
10.164.1.2~Road2d	738
10.164.2Member Function Documentation	738
10.164.2.1Load	738
10.165gazebo::math::RotationSpline Class Reference	738
10.165.1Detailed Description	739
10.165.2Constructor & Destructor Documentation	739
10.165.2.1RotationSpline	739
10.165.2.2~RotationSpline	739
10.165.3Member Function Documentation	739
10.165.3.1AddPoint	739
10.165.3.2Clear	740
10.165.3.3GetNumPoints	740
10.165.3.4GetPoint	740
10.165.3.5Interpolate	740
10.165.3.6Interpolate	741
10.165.3.7RecalcTangents	741
10.165.3.8SetAutoCalculate	741

10.165.3.9UpdatePoint	741
10.165.4Member Data Documentation	742
10.165.4.1autoCalc	742
10.165.4.2points	742
10.165.4.3tangents	742
10.166gazebo::rendering::RTShaderSystem Class Reference	742
10.166.1Detailed Description	743
10.166.2Member Enumeration Documentation	744
10.166.2.1LightingModel	744
10.166.3Member Function Documentation	744
10.166.3.1AddScene	744
10.166.3.2ApplyShadows	744
10.166.3.3AttachEntity	744
10.166.3.4AttachViewport	744
10.166.3.5Clear	745
10.166.3.6DetachEntity	745
10.166.3.7DetachViewport	745
10.166.3.8Fini	745
10.166.3.9GenerateShaders	745
10.166.3.10it	745
10.166.3.11RemoveScene	745
10.166.3.12RemoveShadows	746
10.166.3.13SetPerPixelLighting	746
10.166.3.14UpdateShaders	746
10.167gazebo::rendering::Scene Class Reference	746
10.167.1Detailed Description	749
10.167.2Constructor & Destructor Documentation	749
10.167.2.1Scene	749
10.167.2.2~Scene	749
10.167.3Member Function Documentation	749
10.167.3.1AddVisual	749
10.167.3.2Clear	750
10.167.3.3CloneVisual	750
10.167.3.4CreateCamera	750
10.167.3.5CreateDepthCamera	750
10.167.3.6CreateGpuLaser	751
10.167.3.7CreateGrid	751

10.167.3.8	CreateUserCamera	751
10.167.3.9	DrawLine	751
10.167.3.10	GetAmbientColor	752
10.167.3.10	GetBackgroundColor	752
10.167.3.10	GetCamera	752
10.167.3.10	GetCamera	752
10.167.3.10	GetCameraCount	752
10.167.3.10	GetFirstContact	753
10.167.3.10	GetGrid	753
10.167.3.10	GetGridCount	753
10.167.3.10	GetHeightmap	753
10.167.3.10	GetId	753
10.167.3.20	GetIdString	754
10.167.3.20	GetLight	754
10.167.3.20	GetLight	754
10.167.3.20	GetLightCount	754
10.167.3.20	GetManager	754
10.167.3.20	GetModelVisualAt	755
10.167.3.20	GetName	755
10.167.3.20	GetSelectedVisual	755
10.167.3.20	GetShadowsEnabled	755
10.167.3.20	GetUserCamera	755
10.167.3.30	GetUserCameraCount	756
10.167.3.30	GetVisual	756
10.167.3.30	GetVisualAt	756
10.167.3.30	GetVisualAt	756
10.167.3.30	GetVisualBelow	756
10.167.3.30	GetVisualsBelowPoint	757
10.167.3.30	GetWorldVisual	757
10.167.3.30	Init	757
10.167.3.30	Load	757
10.167.3.30	Load	757
10.167.3.40	PreRender	757
10.167.3.40	PrintSceneGraph	758
10.167.3.40	RemoveVisual	758
10.167.3.40	SelectVisual	758
10.167.3.40	SetAmbientColor	758

10.167.3.4	SetBackgroundColor	758
10.167.3.4	SetFog	758
10.167.3.4	SetGrid	759
10.167.3.4	SetShadowsEnabled	759
10.167.3.4	SetVisible	759
10.167.3.5	SnapVisualToNearestBelow	759
10.167.3.5	StripSceneName	759
10.167.4	Member Data Documentation	760
10.167.4.1	skyx	760
10.168	Gazebo::physics::ScrewJoint< T > Class Template Reference	760
10.168.1	Detailed Description	761
10.168.2	Constructor & Destructor Documentation	761
10.168.2.1	ScrewJoint	761
10.168.2.2	~ScrewJoint	761
10.168.3	Member Function Documentation	761
10.168.3.1	GetAnchor	761
10.168.3.2	Load	761
10.168.3.3	SetAnchor	761
10.168.3.4	SetThreadPitch	762
10.168.4	Member Data Documentation	762
10.168.4.1	fakeAnchor	762
10.168.4.2	threadPitch	762
10.169	df::SDF Class Reference	762
10.169.1	Detailed Description	763
10.169.2	Constructor & Destructor Documentation	763
10.169.2.1	SDF	763
10.169.3	Member Function Documentation	763
10.169.3.1	PrintDescription	763
10.169.3.2	PrintDoc	763
10.169.3.3	PrintValues	763
10.169.3.4	PrintWiki	763
10.169.3.5	SetFromString	763
10.169.3.6	ToString	763
10.169.3.7	Write	763
10.169.4	Member Data Documentation	763
10.169.4.1	root	763
10.169.4.2	version	763

10.170	gazebo::rendering::SelectionObj Class Reference	763
10.170.1	Detailed Description	764
10.170.2	Constructor & Destructor Documentation	764
10.170.2.1	SelectionObj	764
10.170.2.2	~SelectionObj	764
10.170.3	Member Function Documentation	764
10.170.3.1	Attach	764
10.170.3.2	Clear	765
10.170.3.3	GetVisualName	765
10.170.3.4	Init	765
10.170.3.5	IsActive	765
10.170.3.6	SetActive	765
10.170.3.7	SetHighlight	765
10.171	gazebo::sensors::Sensor Class Reference	765
10.171.1	Constructor & Destructor Documentation	767
10.171.1.1	Sensor	767
10.171.1.2	~Sensor	768
10.171.2	Member Function Documentation	768
10.171.2.1	FillMsg	768
10.171.2.2	Finis	768
10.171.2.3	GetLastUpdateTime	768
10.171.2.4	GetName	768
10.171.2.5	GetParentName	768
10.171.2.6	GetPose	769
10.171.2.7	GetScopedName	769
10.171.2.8	GetTopic	769
10.171.2.9	GetType	769
10.171.2.10	GetVisualize	769
10.171.2.11	GetWorldName	769
10.171.2.12	Init	770
10.171.2.13	IsActive	770
10.171.2.14	Load	770
10.171.2.15	Load	770
10.171.2.16	SetActive	770
10.171.2.17	SetParent	771
10.171.2.18	SetUpdateRate	771
10.171.2.19	Update	771

10.171.2.20updateImpl	771
10.171.3 Member Data Documentation	771
10.171.3.1active	771
10.171.3.2connections	771
10.171.3.3lastUpdateTime	771
10.171.3.4node	772
10.171.3.5parentName	772
10.171.3.6plugins	772
10.171.3.7pose	772
10.171.3.8poseSub	772
10.171.3.9sdf	772
10.171.3.10updatePeriod	772
10.171.3.11World	772
10.172 SensorFactor Class Reference	772
10.172.1 Detailed Description	772
10.173 gazebo::sensors::SensorFactory Class Reference	772
10.173.1 Member Function Documentation	773
10.173.1.1 GetSensorTypes	773
10.173.1.2 NewSensor	773
10.173.1.3 RegisterAll	773
10.173.1.4 RegisterSensor	773
10.174 gazebo::sensors::SensorManager Class Reference	774
10.174.1 Detailed Description	775
10.174.2 Member Function Documentation	775
10.174.2.1 CreateSensor	775
10.174.2.2 Fini	775
10.174.2.3 GetSensor	775
10.174.2.4 GetSensorTypes	775
10.174.2.5 Init	776
10.174.2.6 RemoveSensor	776
10.174.2.7 RemoveSensors	776
10.174.2.8 Run	776
10.174.2.9 SensorsInitialized	776
10.174.2.10 Stop	776
10.174.2.11 Update	776
10.175 gazebo::SensorPlugin Class Reference	777
10.175.1 Detailed Description	777

10.175.2	Constructor & Destructor Documentation	777
10.175.2.1	SensorPlugin	777
10.175.2.2	~SensorPlugin	778
10.175.3	Member Function Documentation	778
10.175.3.1	Init	778
10.175.3.2	Load	778
10.175.3.3	Reset	778
10.176	Gazebo::Server Class Reference	778
10.176.1	Constructor & Destructor Documentation	779
10.176.1.1	Server	779
10.176.1.2	~Server	779
10.176.2	Member Function Documentation	779
10.176.2.1	Fini	779
10.176.2.2	GetInitialized	779
10.176.2.3	Init	779
10.176.2.4	Load	779
10.176.2.5	ParseArgs	779
10.176.2.6	PrintUsage	779
10.176.2.7	Run	779
10.176.2.8	SetParams	779
10.176.2.9	Stop	779
10.176.3	Member Data Documentation	779
10.176.3.1	systemPluginsArgc	779
10.176.3.2	systemPluginsArgv	779
10.177	Gazebo::physics::Shape Class Reference	779
10.177.1	Detailed Description	781
10.177.2	Constructor & Destructor Documentation	781
10.177.2.1	Shape	781
10.177.2.2	~Shape	781
10.177.3	Member Function Documentation	781
10.177.3.1	FillMsg	781
10.177.3.2	FillShapeMsg	781
10.177.3.3	GetInertial	781
10.177.3.4	GetMass	781
10.177.3.5	Init	781
10.177.3.6	ProcessMsg	782
10.177.4	Member Data Documentation	782

10.177.4.1collisionParent	782
10.178 SingletonT< T > Class Template Reference	782
10.178.1 Detailed Description	784
10.178.2 Constructor & Destructor Documentation	784
10.178.2.1 SingletonT	784
10.178.2.2 ~SingletonT	784
10.178.3 Member Function Documentation	784
10.178.3.1 Instance	784
10.179 Gazebo::common::Skeleton Class Reference	784
10.179.1 Detailed Description	786
10.179.2 Constructor & Destructor Documentation	786
10.179.2.1 Skeleton	786
10.179.2.2 ~Skeleton	786
10.179.2.3 ~Skeleton	786
10.179.3 Member Function Documentation	786
10.179.3.1 AddAnimation	786
10.179.3.2 AddVertNodeWeight	787
10.179.3.3 BuildNodeMap	787
10.179.3.4 GetAnimation	787
10.179.3.5 GetBindShapeTransform	787
10.179.3.6 GetNodeByHandle	787
10.179.3.7 GetNodeById	787
10.179.3.8 GetNodeByName	788
10.179.3.9 GetNodes	788
10.179.3.10 GetNumAnimations	788
10.179.3.11 GetNumJoints	788
10.179.3.12 GetNumNodes	788
10.179.3.13 GetNumVertNodeWeights	789
10.179.3.14 GetRootNode	789
10.179.3.15 GetVertNodeWeight	789
10.179.3.16 PrintTransforms	789
10.179.3.17 Scale	789
10.179.3.18 SetBindShapeTransform	789
10.179.3.19 SetNumVertAttached	790
10.179.3.20 SetRootNode	790
10.179.4 Member Data Documentation	790
10.179.4.1 anims	790

10.179.4.2	bindShapeTransform	790
10.179.4.3	nodes	790
10.179.4.4	rawNW	790
10.179.4.5	root	790
10.180	gazebo::common::SkeletonAnimation Class Reference	791
10.180.1	Detailed Description	792
10.180.2	Constructor & Destructor Documentation	792
10.180.2.1	SkeletonAnimation	792
10.180.2.2	~SkeletonAnimation	792
10.180.3	Member Function Documentation	792
10.180.3.1	AddKeyFrame	792
10.180.3.2	AddKeyFrame	792
10.180.3.3	GetLength	792
10.180.3.4	GetName	793
10.180.3.5	GetNodeCount	793
10.180.3.6	GetNodePoseAt	793
10.180.3.7	GetPoseAt	793
10.180.3.8	GetPoseAtX	794
10.180.3.9	HasNode	794
10.180.3.10	Scale	794
10.180.3.11	SetName	794
10.180.4	Member Data Documentation	794
10.180.4.1	animations	794
10.180.4.2	length	795
10.180.4.3	name	795
10.181	gazebo::common::SkeletonNode Class Reference	795
10.181.1	Detailed Description	797
10.181.2	Member Enumeration Documentation	797
10.181.2.1	SkeletonNodeType	797
10.181.3	Constructor & Destructor Documentation	797
10.181.3.1	SkeletonNode	797
10.181.3.2	SkeletonNode	798
10.181.3.3	~SkeletonNode	798
10.181.4	Member Function Documentation	798
10.181.4.1	AddChild	798
10.181.4.2	AddRawTransform	798
10.181.4.3	GetChild	798

10.181.4.4	GetChildById	798
10.181.4.5	GetChildByName	799
10.181.4.6	GetChildCount	799
10.181.4.7	GetHandle	799
10.181.4.8	GetId	799
10.181.4.9	GetInverseBindTransform	799
10.181.4.10	GetModelTransform	800
10.181.4.11	GetName	800
10.181.4.12	GetNumRawTrans	800
10.181.4.13	GetParent	800
10.181.4.14	GetRawTransform	800
10.181.4.15	GetRawTransforms	800
10.181.4.16	GetTransform	801
10.181.4.17	GetTransforms	801
10.181.4.18	Joint	801
10.181.4.19	RootNode	801
10.181.4.20	Reset	801
10.181.4.21	SetHandle	801
10.181.4.22	SetId	802
10.181.4.23	SetInitialTransform	802
10.181.4.24	SetInverseBindTransform	802
10.181.4.25	SetModelTransform	802
10.181.4.26	SetName	802
10.181.4.27	SetParent	802
10.181.4.28	SetTransform	803
10.181.4.29	SetType	803
10.181.4.30	UpdateChildrenTransforms	803
10.181.5	Member Data Documentation	803
10.181.5.1	children	803
10.181.5.2	handle	803
10.181.5.3	d	803
10.181.5.4	initialTransform	803
10.181.5.5	invBindTransform	803
10.181.5.6	modelTransform	803
10.181.5.7	name	804
10.181.5.8	parent	804
10.181.5.9	rawTransforms	804

10.181.5.1	Transform	804
10.181.5.1	type	804
10.182	gazebo::physics::SliderJoint< T > Class Template Reference	804
10.182.1	Detailed Description	805
10.182.2	Constructor & Destructor Documentation	805
10.182.2.1	SliderJoint	805
10.182.2.2	~SliderJoint	805
10.182.3	Member Function Documentation	805
10.182.3.1	GetAnchor	805
10.182.3.2	Load	805
10.182.3.3	SetAnchor	806
10.182.4	Member Data Documentation	806
10.182.4.1	fakeAnchor	806
10.183	gazebo::physics::SphereShape Class Reference	806
10.183.1	Detailed Description	807
10.183.2	Constructor & Destructor Documentation	807
10.183.2.1	SphereShape	807
10.183.2.2	~SphereShape	807
10.183.3	Member Function Documentation	807
10.183.3.1	FillMsg	807
10.183.3.2	GetInertial	807
10.183.3.3	GetMass	808
10.183.3.4	GetRadius	808
10.183.3.5	Init	808
10.183.3.6	ProcessMsg	808
10.183.3.7	SetRadius	808
10.184	gazebo::math::Spline Class Reference	808
10.184.1	Detailed Description	809
10.184.2	Constructor & Destructor Documentation	809
10.184.2.1	Spline	809
10.184.2.2	~Spline	809
10.184.3	Member Function Documentation	810
10.184.3.1	AddPoint	810
10.184.3.2	Clear	810
10.184.3.3	GetPoint	810
10.184.3.4	GetPointCount	810
10.184.3.5	GetTangent	810

10.184.3.6	GetTension	810
10.184.3.7	Interpolate	811
10.184.3.8	Interpolate	811
10.184.3.9	RecalcTangents	811
10.184.3.10	SetAutoCalculate	811
10.184.3.11	SetTension	812
10.184.3.12	UpdatePoint	812
10.184.4	Member Data Documentation	812
10.184.4.1	autoCalc	812
10.184.4.2	coeffs	812
10.184.4.3	points	812
10.184.4.4	tangents	812
10.184.4.5	tension	812
10.185	gazebo::physics::State Class Reference	813
10.185.1	Detailed Description	814
10.185.2	Constructor & Destructor Documentation	814
10.185.2.1	State	814
10.185.2.2	State	814
10.185.2.3	~State	814
10.185.3	Member Function Documentation	814
10.185.3.1	GetName	814
10.185.3.2	GetRealTime	815
10.185.3.3	GetSimTime	815
10.185.3.4	GetWallTime	815
10.185.3.5	Load	815
10.185.4	Member Data Documentation	815
10.185.4.1	name	815
10.185.4.2	realTime	815
10.185.4.3	simTime	815
10.185.4.4	wallTime	816
10.186	gazebo::common::STLLoader Class Reference	816
10.186.1	Detailed Description	816
10.186.2	Constructor & Destructor Documentation	816
10.186.2.1	STLLoader	816
10.186.2.2	~STLLoader	817
10.186.3	Member Function Documentation	817
10.186.3.1	Load	817

10.187	7 gazebo::common::SubMesh Class Reference	817
10.187.1	Detailed Description	819
10.187.2	Member Enumeration Documentation	819
10.187.2.1	PrimitiveType	819
10.187.3	Constructor & Destructor Documentation	819
10.187.3.1	SubMesh	819
10.187.3.2	~SubMesh	820
10.187.4	Member Function Documentation	820
10.187.4.1	AddIndex	820
10.187.4.2	AddNodeAssignment	820
10.187.4.3	AddNormal	820
10.187.4.4	AddNormal	820
10.187.4.5	AddTexCoord	820
10.187.4.6	AddVertex	821
10.187.4.7	AddVertex	821
10.187.4.8	CopyNormals	821
10.187.4.9	CopyVertices	821
10.187.4.10	FillArrays	821
10.187.4.11	GenSphericalTexCoord	821
10.187.4.12	GetIndex	822
10.187.4.13	GetIndexCount	822
10.187.4.14	GetMaterialIndex	822
10.187.4.15	GetMax	822
10.187.4.16	GetMaxIndex	822
10.187.4.17	GetMin	822
10.187.4.18	GetNodeAssignment	822
10.187.4.19	GetNodeAssignmentsCount	823
10.187.4.20	GetNormal	823
10.187.4.21	GetNormalCount	823
10.187.4.22	GetPrimitiveType	823
10.187.4.23	GetTexCoord	823
10.187.4.24	GetTexCoordCount	823
10.187.4.25	GetVertex	823
10.187.4.26	GetVertexCount	824
10.187.4.27	GetVertexIndex	824
10.187.4.28	HasVertex	824
10.187.4.29	RecalculateNormals	824

10.187.4.30	Scale	824
10.187.4.31	SetIndexCount	824
10.187.4.32	SetMaterialIndex	825
10.187.4.33	SetNormal	825
10.187.4.34	SetNormalCount	825
10.187.4.35	SetPrimitiveType	825
10.187.4.36	SetSubMeshCenter	825
10.187.4.37	SetTexCoord	825
10.187.4.38	SetTexCoordCount	826
10.187.4.39	SetVertex	826
10.187.4.40	SetVertexCount	826
10.188	Gazebo::transport::SubscribeOptions Class Reference	826
10.188.1	Detailed Description	827
10.188.2	Constructor & Destructor Documentation	827
10.188.2.1	SubscribeOptions	827
10.188.3	Member Function Documentation	827
10.188.3.1	GetLatching	827
10.188.3.2	GetMsgType	827
10.188.3.3	GetNode	827
10.188.3.4	GetTopic	827
10.188.3.5	Init	827
10.189	Gazebo::transport::Subscriber Class Reference	827
10.189.1	Detailed Description	827
10.189.2	Constructor & Destructor Documentation	828
10.189.2.1	Subscriber	828
10.189.2.2	~Subscriber	828
10.189.3	Member Function Documentation	828
10.189.3.1	GetTopic	828
10.189.3.2	Unsubscribe	828
10.190	Gazebo::transport::SubscriptionTransport Class Reference	828
10.190.1	Detailed Description	829
10.190.2	Constructor & Destructor Documentation	829
10.190.2.1	SubscriptionTransport	829
10.190.2.2	~SubscriptionTransport	829
10.190.3	Member Function Documentation	829
10.190.3.1	GetConnection	829
10.190.3.2	HandleData	829

10.190.3.3	Init	829
10.190.3.4	IsLocal	830
10.191	gazebo::physics::SurfaceParams Class Reference	830
10.191.1	Detailed Description	831
10.191.2	Constructor & Destructor Documentation	831
10.191.2.1	SurfaceParams	831
10.191.2.2	~SurfaceParams	831
10.191.3	Member Function Documentation	831
10.191.3.1	FillSurfaceMsg	831
10.191.3.2	Load	831
10.191.3.3	ProcessMsg	831
10.191.4	Member Data Documentation	831
10.191.4.1	bounce	831
10.191.4.2	bounceThreshold	832
10.191.4.3	cfm	832
10.191.4.4	erp	832
10.191.4.5	dir1	832
10.191.4.6	d	832
10.191.4.7	kp	832
10.191.4.8	maxVel	833
10.191.4.9	minDepth	833
10.191.4.10	u1	833
10.191.4.11	u2	833
10.191.4.12	slip1	833
10.191.4.13	slip2	834
10.192	gazebo::common::SystemPaths Class Reference	834
10.192.1	Detailed Description	835
10.192.2	Member Function Documentation	835
10.192.2.1	AddGazeboPaths	835
10.192.2.2	AddOgrePaths	836
10.192.2.3	AddPluginPaths	836
10.192.2.4	AddSearchPathSuffix	836
10.192.2.5	ClearGazeboPaths	836
10.192.2.6	ClearOgrePaths	836
10.192.2.7	ClearPluginPaths	836
10.192.2.8	FindFile	836
10.192.2.9	FindFileURI	837

10.192.2.10	GetGazeboPaths	837
10.192.2.10	GetLogPath	837
10.192.2.10	GetModelPaths	837
10.192.2.10	GetOgrePaths	837
10.192.2.10	GetPluginPaths	837
10.192.2.10	GetWorldPathExtension	838
10.192.3	Member Data Documentation	838
10.192.3.1	gazeboPathsFromEnv	838
10.192.3.2	modelPathsFromEnv	838
10.192.3.3	ogrePathsFromEnv	838
10.192.3.4	pluginPathsFromEnv	838
10.193	gazebo::SystemPlugin Class Reference	838
10.193.1	Detailed Description	839
10.193.2	Constructor & Destructor Documentation	839
10.193.2.1	SystemPlugin	839
10.193.2.2	~SystemPlugin	839
10.193.3	Member Function Documentation	839
10.193.3.1	Init	839
10.193.3.2	Load	839
10.193.3.3	Reset	840
10.194	gazebo::common::Time Class Reference	840
10.194.1	Detailed Description	843
10.194.2	Constructor & Destructor Documentation	843
10.194.2.1	Time	843
10.194.2.2	Time	844
10.194.2.3	Time	844
10.194.2.4	Time	844
10.194.2.5	Time	844
10.194.2.6	Time	844
10.194.2.7	~Time	844
10.194.3	Member Function Documentation	844
10.194.3.1	Double	845
10.194.3.2	Float	845
10.194.3.3	GetWallTime	845
10.194.3.4	MicToNano	845
10.194.3.5	MilToNano	845
10.194.3.6	MSleep	846

10.194.3.7NSleep	846
10.194.3.8NSleep	846
10.194.3.9operator!=	846
10.194.3.10operator!=	846
10.194.3.11operator!=	847
10.194.3.12operator!=	847
10.194.3.13operator*	847
10.194.3.14operator*	847
10.194.3.15operator*	847
10.194.3.16operator*=	848
10.194.3.17operator*=	848
10.194.3.18operator*=	848
10.194.3.19operator+	848
10.194.3.20operator+	849
10.194.3.21operator+	849
10.194.3.22operator+=	849
10.194.3.23operator+=	849
10.194.3.24operator+=	850
10.194.3.25operator-	850
10.194.3.26operator-	850
10.194.3.27operator-	850
10.194.3.28operator-=	850
10.194.3.29operator-=	850
10.194.3.30operator-=	851
10.194.3.31operator/	851
10.194.3.32operator/	851
10.194.3.33operator/	851
10.194.3.34operator/=	851
10.194.3.35operator/=	852
10.194.3.36operator/=	852
10.194.3.37operator<	852
10.194.3.38operator<	852
10.194.3.39operator<	853
10.194.3.40operator<	853
10.194.3.41operator<=	853
10.194.3.42operator<=	853
10.194.3.43operator<=	854

10.194.3.44	operator<=	854
10.194.3.45	operator=	854
10.194.3.46	operator=	854
10.194.3.47	operator=	855
10.194.3.48	operator==	855
10.194.3.49	operator==	855
10.194.3.50	operator==	855
10.194.3.51	operator==	856
10.194.3.52	operator>	856
10.194.3.53	operator>	856
10.194.3.54	operator>	856
10.194.3.55	operator>	857
10.194.3.56	operator>=	857
10.194.3.57	operator>=	857
10.194.3.58	operator>=	857
10.194.3.59	operator>=	858
10.194.3.60	SecToNano	858
10.194.3.61	Set	858
10.194.3.62	Set	858
10.194.3.63	SetToWallTime	859
10.194.4	Friends And Related Function Documentation	859
10.194.4.1	operator<<	859
10.194.4.2	operator>>	859
10.194.5	Member Data Documentation	859
10.194.5.1	insec	859
10.194.5.2	sec	859
10.195	Gazebo::common::Timer Class Reference	859
10.195.1	Detailed Description	860
10.195.2	Constructor & Destructor Documentation	860
10.195.2.1	Timer	860
10.195.2.2	~Timer	860
10.195.3	Member Function Documentation	861
10.195.3.1	GetElapsed	861
10.195.3.2	Start	861
10.195.4	Friends And Related Function Documentation	861
10.195.4.1	operator<<	861
10.196	Gazebo::transport::TopicManager Class Reference	861

10.196.1	Detailed Description	862
10.196.2	Member Typedef Documentation	863
10.196.2.1	SubNodeMap	863
10.196.3	Member Function Documentation	863
10.196.3.1	AddNode	863
10.196.3.2	Advertise	863
10.196.3.3	ClearBuffers	863
10.196.3.4	ConnectPubToSub	863
10.196.3.5	ConnectSubscribers	863
10.196.3.6	ConnectSubToPub	863
10.196.3.7	DisconnectPubFromSub	863
10.196.3.8	DisconnectSubFromPub	863
10.196.3.9	FindPublication	863
10.196.3.10	Ini	864
10.196.3.11	GetTopicNamespaces	864
10.196.3.12	Wait	864
10.196.3.13	Advertised	864
10.196.3.14	PauseIncoming	864
10.196.3.15	ProcessNodes	864
10.196.3.16	Publish	864
10.196.3.17	RegisterTopicNamespace	864
10.196.3.18	RemoveNode	864
10.196.3.19	Subscribe	865
10.196.3.20	Unadvertise	865
10.196.3.21	Unsubscribe	865
10.196.3.22	UpdatePublications	865
10.197	Gazebo::physics::TrajectoryInfo Struct Reference	865
10.197.1	Member Data Documentation	865
10.197.1.1	duration	865
10.197.1.2	endTime	865
10.197.1.3	d	866
10.197.1.4	startTime	866
10.197.1.5	translated	866
10.197.1.6	type	866
10.198	Gazebo::physics::TrimeshShape Class Reference	866
10.198.1	Detailed Description	867
10.198.2	Constructor & Destructor Documentation	867

10.198.2.1	TrimeshShape	867
10.198.2.2	~TrimeshShape	867
10.198.3	Member Function Documentation	867
10.198.3.1	FillMsg	867
10.198.3.2	GetFilename	867
10.198.3.3	GetMass	868
10.198.3.4	GetSize	868
10.198.3.5	Init	868
10.198.3.6	ProcessMsg	868
10.198.3.7	SetFilename	868
10.198.3.8	SetScale	868
10.198.3.9	Update	868
10.198.4	Member Data Documentation	868
10.198.4.1	mesh	868
10.199	gazebo::physics::UniversalJoint< T > Class Template Reference	868
10.199.1	Detailed Description	869
10.199.2	Constructor & Destructor Documentation	869
10.199.2.1	UniversalJoint	869
10.199.2.2	~UniversalJoint	869
10.199.3	Member Function Documentation	870
10.199.3.1	Load	870
10.200	urdf2gazebo::URDF2Gazebo Class Reference	870
10.200.1	Constructor & Destructor Documentation	873
10.200.1.1	URDF2Gazebo	873
10.200.1.2	~URDF2Gazebo	873
10.200.2	Member Function Documentation	873
10.200.2.1	addKeyValue	873
10.200.2.2	addTransform	873
10.200.2.3	copyPose	873
10.200.2.4	copyPose	873
10.200.2.5	createCollision	873
10.200.2.6	createCollisions	873
10.200.2.7	createGeometry	873
10.200.2.8	createInertial	873
10.200.2.9	createJoint	873
10.200.2.10	createLink	874
10.200.2.11	createSDF	874

10.200.2.10	CreateVisual	874
10.200.2.10	CreateVisuals	874
10.200.2.10	GetGeometryBoundingBox	874
10.200.2.10	GetKeyValueAsString	874
10.200.2.10	InitModelDoc	874
10.200.2.10	InitModelFile	874
10.200.2.10	InitModelString	874
10.200.2.10	InitModelString	874
10.200.2.20	InsertGazeboExtensionCollision	874
10.200.2.21	InsertGazeboExtensionJoint	874
10.200.2.22	InsertGazeboExtensionLink	874
10.200.2.23	InsertGazeboExtensionRobot	874
10.200.2.24	InsertGazeboExtensionVisual	875
10.200.2.25	InverseTransformToParentFrame	875
10.200.2.26	IsGazeboExtensions	875
10.200.2.27	IsGazeboExtensions	875
10.200.2.28	ParseGazeboExtension	875
10.200.2.29	ParseVector3	875
10.200.2.30	PrintCollisionGroups	875
10.200.2.31	PrintMass	875
10.200.2.32	PrintMass	875
10.200.2.33	ReduceCollisionsToParent	875
10.200.2.34	ReduceCollisionToParent	876
10.200.2.35	ReduceFixedJoints	876
10.200.2.36	ReduceGazeboExtensionContactSensorFrameReplace	876
10.200.2.37	ReduceGazeboExtensionFrameReplace	876
10.200.2.38	ReduceGazeboExtensionGripperFrameReplace	876
10.200.2.39	ReduceGazeboExtensionJointFrameReplace	876
10.200.2.40	ReduceGazeboExtensionPluginFrameReplace	876
10.200.2.41	ReduceGazeboExtensionProjectorFrameReplace	876
10.200.2.42	ReduceGazeboExtensionProjectorTransformReduction	876
10.200.2.43	ReduceGazeboExtensionSensorTransformReduction	877
10.200.2.44	ReduceGazeboExtensionsTransformReduction	877
10.200.2.45	ReduceGazeboExtensionToParent	877
10.200.2.46	ReduceInertialToParent	877
10.200.2.47	ReduceJointsToParent	877
10.200.2.48	ReduceVisualsToParent	877

10.200.2.4	ReduceVisualToParent	877
10.200.2.50	TransformToParentFrame	877
10.200.2.51	TransformToParentFrame	877
10.200.2.52	TransformToParentFrame	877
10.200.2.53	Values2str	878
10.200.2.54	Vector32str	878
10.200.3	Member Data Documentation	878
10.200.3.1	gazebo_extensions_	878
10.201	gazebo::rendering::UserCamera Class Reference	878
10.201.1	Detailed Description	880
10.201.2	Constructor & Destructor Documentation	880
10.201.2.1	UserCamera	880
10.201.2.2	~UserCamera	880
10.201.3	Member Function Documentation	880
10.201.3.1	AttachToVisualImpl	880
10.201.3.2	EnableViewController	881
10.201.3.3	Fini	881
10.201.3.4	GetAvgFPS	881
10.201.3.5	GetGUIOverlay	881
10.201.3.6	GetTriangleCount	881
10.201.3.7	GetVisual	882
10.201.3.8	GetVisual	882
10.201.3.9	HandleKeyPressEvent	882
10.201.3.10	HandleKeyReleaseEvent	882
10.201.3.11	HandleMouseEvent	882
10.201.3.12	hit	883
10.201.3.13	load	883
10.201.3.14	load	883
10.201.3.15	MoveToPosition	883
10.201.3.16	MoveToVisual	883
10.201.3.17	MoveToVisual	883
10.201.3.18	PostRender	884
10.201.3.19	resize	884
10.201.3.20	SetFocalPoint	884
10.201.3.21	SetRenderTarget	884
10.201.3.22	SetViewController	884
10.201.3.23	SetViewController	884

10.201.3.28	SetViewportDimensions	885
10.201.3.29	SetWorldPose	885
10.201.3.28	BackVisualImpl	885
10.201.3.27	Update	885
10.202	Gazebo::math::Vector2d Class Reference	885
10.202.1	Detailed Description	887
10.202.2	Constructor & Destructor Documentation	887
10.202.2.1	Vector2d	887
10.202.2.2	Vector2d	887
10.202.2.3	Vector2d	887
10.202.2.4	~Vector2d	888
10.202.3	Member Function Documentation	888
10.202.3.1	Cross	888
10.202.3.2	Distance	888
10.202.3.3	IsFinite	888
10.202.3.4	Normalize	888
10.202.3.5	operator!=	888
10.202.3.6	operator*	889
10.202.3.7	operator*	889
10.202.3.8	operator*==	889
10.202.3.9	operator*==	889
10.202.3.10	operator+	890
10.202.3.11	operator+=	890
10.202.3.10	operator-	890
10.202.3.10	operator-=	890
10.202.3.10	operator/	891
10.202.3.10	operator/	891
10.202.3.10	operator/=	891
10.202.3.10	operator/=	891
10.202.3.10	operator=	892
10.202.3.10	operator=	892
10.202.3.20	operator==	892
10.202.3.21	operator[]	892
10.202.3.28	Set	893
10.202.4	Friends And Related Function Documentation	893
10.202.4.1	operator<<	893
10.202.4.2	operator>>	893

10.202.5	Member Data Documentation	893
10.202.5.1x	893
10.202.5.2y	894
10.203	Gazebo::math::Vector2i Class Reference	894
10.203.1	Detailed Description	895
10.203.2	Constructor & Destructor Documentation	895
10.203.2.1	Vector2i	895
10.203.2.2	Vector2i	896
10.203.2.3	Vector2i	896
10.203.2.4	~Vector2i	896
10.203.3	Member Function Documentation	896
10.203.3.1	Cross	896
10.203.3.2	Distance	896
10.203.3.3	IsFinite	897
10.203.3.4	Normalize	897
10.203.3.5	operator!=	897
10.203.3.6	operator*	897
10.203.3.7	operator*	897
10.203.3.8	operator*=	898
10.203.3.9	operator*=	898
10.203.3.10	operator+	898
10.203.3.11	operator+=	898
10.203.3.12	operator-	899
10.203.3.13	operator-=	899
10.203.3.14	operator/	899
10.203.3.15	operator/	900
10.203.3.16	operator/=	900
10.203.3.17	operator/=	900
10.203.3.18	operator=	901
10.203.3.19	operator=	901
10.203.3.20	operator==	901
10.203.3.21	operator[]	901
10.203.3.22	Set	901
10.203.4	Friends And Related Function Documentation	902
10.203.4.1	operator<<	902
10.203.4.2	operator>>	902
10.203.5	Member Data Documentation	902

10.203.5.1x	902
10.203.5.2y	902
10.204.1 gazebo::math::Vector3 Class Reference	902
10.204.1.1 Detailed Description	905
10.204.2 Constructor & Destructor Documentation	905
10.204.2.1 Vector3	905
10.204.2.2 Vector3	905
10.204.2.3 Vector3	905
10.204.2.4 ~Vector3	906
10.204.3 Member Function Documentation	906
10.204.3.1 Correct	906
10.204.3.2 Cross	906
10.204.3.3 Distance	906
10.204.3.4 Distance	906
10.204.3.5 Dot	907
10.204.3.6 Equal	907
10.204.3.7 GetAbs	907
10.204.3.8 GetDistToLine	907
10.204.3.9 GetLength	908
10.204.3.10 GetMax	908
10.204.3.11 GetMin	908
10.204.3.12 GetNormal	908
10.204.3.13 GetPerpendicular	908
10.204.3.14 GetRounded	908
10.204.3.15 GetSquaredLength	909
10.204.3.16 GetSum	909
10.204.3.17 IsFinite	909
10.204.3.18 Normalize	909
10.204.3.19 Operator!=	909
10.204.3.20 Operator*	909
10.204.3.21 Operator*	910
10.204.3.22 Operator*=	910
10.204.3.23 Operator*=	910
10.204.3.24 Operator+	910
10.204.3.25 Operator+=	911
10.204.3.26 Operator-	911
10.204.3.27 Operator-=	911

10.204.3.28	operator/	911
10.204.3.29	operator/	911
10.204.3.30	operator/=	912
10.204.3.31	operator/=	912
10.204.3.32	operator=	912
10.204.3.33	operator=	912
10.204.3.34	operator==	913
10.204.3.35	operator[]	913
10.204.3.36	round	913
10.204.3.37	round	913
10.204.3.38	Set	913
10.204.3.39	SetToMax	914
10.204.3.40	SetToMin	914
10.204.4	Friends And Related Function Documentation	914
10.204.4.1	operator<<	914
10.204.4.2	operator>>	914
10.204.5	Member Data Documentation	914
10.204.5.1x		914
10.204.5.2y		915
10.204.5.3z		915
10.205	Gazebo::math::Vector4 Class Reference	915
10.205.1	Detailed Description	917
10.205.2	Constructor & Destructor Documentation	917
10.205.2.1	Vector4	917
10.205.2.2	Vector4	917
10.205.2.3	Vector4	917
10.205.2.4	~Vector4	917
10.205.3	Member Function Documentation	917
10.205.3.1	Distance	917
10.205.3.2	GetLength	918
10.205.3.3	GetSquaredLength	918
10.205.3.4	IsFinite	918
10.205.3.5	Normalize	918
10.205.3.6	operator!=	918
10.205.3.7	operator*	918
10.205.3.8	operator*	919
10.205.3.9	operator*	919

10.205.3.10	operator*= operator*= operator+ operator+= operator- operator-= operator/ operator/ operator/=	919 920 920 920 920 921 921 921 922
10.205.3.11	operator/=	922
10.205.3.12	operator=	922
10.205.3.21	operator=	922
10.205.3.22	operator==	923
10.205.3.23	operator[]	923
10.205.3.29	set	923
10.205.4	Friends And Related Function Documentation	923
10.205.4.1	operator<<	923
10.205.4.2	operator>>	923
10.205.5	Member Data Documentation	924
10.205.5.1w		924
10.205.5.2x		924
10.205.5.3y		924
10.205.5.4z		924
10.206	gazebo::common::Video Class Reference	924
10.206.1	Detailed Description	925
10.206.2	Constructor & Destructor Documentation	925
10.206.2.1	Video	925
10.206.2.2	~Video	925
10.206.3	Member Function Documentation	925
10.206.3.1	GetHeight	925
10.206.3.2	GetNextFrame	925
10.206.3.3	GetWidth	925
10.206.3.4	Load	926
10.207	gazebo::rendering::VideoVisual Class Reference	926
10.207.1	Detailed Description	927
10.207.2	Constructor & Destructor Documentation	927
10.207.2.1	VideoVisual	927

10.207.2.2~VideoVisual	927
10.208 gazebo::rendering::ViewController Class Reference	927
10.208.1 Detailed Description	928
10.208.2 Constructor & Destructor Documentation	928
10.208.2.1 ViewController	928
10.208.2.2~ViewController	929
10.208.3 Member Function Documentation	929
10.208.3.1 GetTypeString	929
10.208.3.2 HandleKeyPressEvent	929
10.208.3.3 HandleKeyReleaseEvent	929
10.208.3.4 HandleMouseEvent	929
10.208.3.5 Init	930
10.208.3.6 Init	930
10.208.3.7 SetEnabled	930
10.208.3.8 Update	930
10.208.4 Member Data Documentation	930
10.208.4.1 camera	930
10.208.4.2 enabled	930
10.208.4.3 typeString	930
10.209 gazebo::rendering::Visual Class Reference	931
10.209.1 Detailed Description	935
10.209.2 Constructor & Destructor Documentation	935
10.209.2.1 Visual	935
10.209.2.2 Visual	936
10.209.2.3~Visual	936
10.209.3 Member Function Documentation	936
10.209.3.1 AttachAxes	936
10.209.3.2 AttachLineVertex	936
10.209.3.3 AttachMesh	936
10.209.3.4 AttachObject	936
10.209.3.5 AttachVisual	937
10.209.3.6 ClearParent	937
10.209.3.7 Clone	937
10.209.3.8 CreateDynamicLine	937
10.209.3.9 DeleteDynamicLine	937
10.209.3.10 DetachObjects	937
10.209.3.11 DetachVisual	938

10.209.3.12	DetachVisual	938
10.209.3.13	DisableTrackVisual	938
10.209.3.14	EnableTrackVisual	938
10.209.3.15	End	938
10.209.3.16	GetAttachedObjectCount	938
10.209.3.17	GetBoundingBox	938
10.209.3.18	GetChild	939
10.209.3.19	GetChildCount	939
10.209.3.20	GetMaterialName	939
10.209.3.21	GetName	939
10.209.3.22	GetNormalMap	939
10.209.3.23	GetParent	939
10.209.3.24	GetPose	940
10.209.3.25	GetPosition	940
10.209.3.26	GetRootVisual	940
10.209.3.27	GetRotation	940
10.209.3.28	GetScale	940
10.209.3.29	GetScene	940
10.209.3.30	GetSceneNode	941
10.209.3.31	GetShaderType	941
10.209.3.32	GetTransparency	941
10.209.3.33	GetVisibilityFlags	941
10.209.3.34	GetVisible	941
10.209.3.35	GetWorldPose	942
10.209.3.36	HasAttachedObject	942
10.209.3.37	Init	942
10.209.3.38	InsertMesh	942
10.209.3.39	InsertMesh	942
10.209.3.40	Plane	942
10.209.3.41	Static	942
10.209.3.42	Load	943
10.209.3.43	Load	943
10.209.3.44	LoadFromMsg	943
10.209.3.45	LoadPlugin	943
10.209.3.46	MakeStatic	943
10.209.3.47	MoveToPosition	943
10.209.3.48	MoveToPositions	944

10.209.3.49	RemovePlugin	944
10.209.3.50	SetAmbient	944
10.209.3.51	SetCastShadows	944
10.209.3.52	SetDiffuse	944
10.209.3.53	SetEmissive	944
10.209.3.54	SetHighlighted	945
10.209.3.55	SetMaterial	945
10.209.3.56	SetName	945
10.209.3.57	SetNormalMap	945
10.209.3.58	SetPose	945
10.209.3.59	SetPosition	945
10.209.3.60	SetRibbonTrail	946
10.209.3.61	SetRotation	946
10.209.3.62	SetScale	946
10.209.3.63	SetScene	946
10.209.3.64	SetShaderType	946
10.209.3.65	SetSkeletonPose	946
10.209.3.66	SetSpecular	947
10.209.3.67	SetTransparency	947
10.209.3.68	SetVisibilityFlags	947
10.209.3.69	SetVisible	947
10.209.3.70	SetWorldPose	947
10.209.3.71	SetWorldPosition	948
10.209.3.72	SetWorldRotation	948
10.209.3.73	ShowBoundingBox	948
10.209.3.74	ShowCollision	948
10.209.3.75	ShowCOM	948
10.209.3.76	ShowJoints	948
10.209.3.77	ShowSkeleton	949
10.209.3.78	ToggleVisible	949
10.209.3.79	Update	949
10.209.3.80	UpdateFromMsg	949
10.209.4	Member Data Documentation	949
10.209.4.1	parent	949
10.209.4.2	scene	949
10.209.4.3	sceneNode	949
10.210	Gazebo::VisualPlugin Class Reference	949

10.210.1	Detailed Description	950
10.210.2	Constructor & Destructor Documentation	950
10.210.2.1	VisualPlugin	950
10.210.3	Member Function Documentation	950
10.210.3.1	Init	950
10.210.3.2	Load	951
10.210.3.3	Reset	951
10.211	gazebo::rendering::WindowManager Class Reference	951
10.211.1	Detailed Description	952
10.211.2	Member Function Documentation	952
10.211.2.1	CreateWindow	952
10.211.2.2	Finis	952
10.211.2.3	GetAvgFPS	952
10.211.2.4	GetTriangleCount	953
10.211.2.5	GetWindow	953
10.211.2.6	Moved	953
10.211.2.7	Resize	953
10.211.2.8	SetCamera	953
10.212	gazebo::physics::World Class Reference	954
10.212.1	Detailed Description	956
10.212.2	Constructor & Destructor Documentation	957
10.212.2.1	World	957
10.212.2.2	~World	957
10.212.3	Member Function Documentation	957
10.212.3.1	Clear	957
10.212.3.2	DisableAllModels	957
10.212.3.3	EnableAllModels	957
10.212.3.4	EnablePhysicsEngine	957
10.212.3.5	Finis	957
10.212.3.6	GetByName	958
10.212.3.7	GetEnablePhysicsEngine	958
10.212.3.8	GetEntity	958
10.212.3.9	GetEntityBelowPoint	958
10.212.3.10	GetEntityByName	959
10.212.3.11	GetModel	959
10.212.3.12	GetModel	959
10.212.3.13	GetModelBelowPoint	959

10.212.3.10	GetModelByName	960
10.212.3.11	GetModelCount	960
10.212.3.12	GetModels	960
10.212.3.13	GetName	960
10.212.3.14	GetPauseTime	960
10.212.3.15	GetPhysicsEngine	960
10.212.3.16	GetRealTime	961
10.212.3.17	GetSelectedEntity	961
10.212.3.18	GetSetWorldPoseMutex	961
10.212.3.19	GetSimTime	961
10.212.3.20	GetStartTime	961
10.212.3.21	GetState	961
10.212.3.22	Get	962
10.212.3.23	InsertModelFile	962
10.212.3.24	InsertModelSDF	962
10.212.3.25	InsertModelString	962
10.212.3.26	IsPaused	962
10.212.3.27	Load	962
10.212.3.28	LoadPlugin	963
10.212.3.29	PrintEntityTree	963
10.212.3.30	RemovePlugin	963
10.212.3.31	Reset	963
10.212.3.32	ResetEntities	963
10.212.3.33	ResetTime	963
10.212.3.34	Run	964
10.212.3.35	Save	964
10.212.3.36	SetPaused	964
10.212.3.37	SetSimTime	964
10.212.3.38	SetState	964
10.212.3.39	StepWorld	964
10.212.3.40	Stop	965
10.212.3.41	StripWorldName	965
10.212.4	Member Data Documentation	965
10.212.4.1	dirtyPoses	965
10.213	Gazebo::WorldPlugin Class Reference	965
10.213.1	Detailed Description	966
10.213.2	Constructor & Destructor Documentation	966

10.213.2.1WorldPlugin	966
10.213.2.2~WorldPlugin	966
10.213.3Member Function Documentation	966
10.213.3.1Init	966
10.213.3.2Load	966
10.213.3.3Reset	966
10.214gazebo::physics::WorldState Class Reference	967
10.214.1Detailed Description	967
10.214.2Constructor & Destructor Documentation	968
10.214.2.1WorldState	968
10.214.2.2WorldState	968
10.214.2.3~WorldState	968
10.214.3Member Function Documentation	968
10.214.3.1GetModelState	968
10.214.3.2GetModelState	968
10.214.3.3GetModelStateCount	968
10.214.3.4GetSDF	969
10.214.3.5Load	969
11 File Documentation	971
11.1 Actor.hh File Reference	971
11.2 Angle.hh File Reference	972
11.2.1 Macro Definition Documentation	973
11.2.1.1 GZ_DTOR	973
11.2.1.2 GZ_NORMALIZE	973
11.2.1.3 GZ_RTOD	973
11.3 Animation.hh File Reference	974
11.4 ArrowVisual.hh File Reference	975
11.5 AxisVisual.hh File Reference	976
11.6 BallJoint.hh File Reference	976
11.7 Base.hh File Reference	977
11.8 Box.hh File Reference	978
11.9 BoxShape.hh File Reference	979
11.10bullet_inc.h File Reference	980
11.11BulletBallJoint.hh File Reference	981
11.12BulletBoxShape.hh File Reference	982
11.13BulletCollision.hh File Reference	982

11.14BulletCylinderShape.hh File Reference	983
11.15BulletHeightmapShape.hh File Reference	984
11.16BulletHinge2Joint.hh File Reference	984
11.17BulletHingeJoint.hh File Reference	985
11.18BulletJoint.hh File Reference	985
11.19BulletLink.hh File Reference	986
11.20BulletMotionState.hh File Reference	987
11.21BulletMultiRayShape.hh File Reference	988
11.22BulletPhysics.hh File Reference	988
11.23BulletPlaneShape.hh File Reference	989
11.24BulletRaySensor.hh File Reference	990
11.25BulletRayShape.hh File Reference	990
11.26BulletScrewJoint.hh File Reference	991
11.27BulletSliderJoint.hh File Reference	992
11.28BulletSphereShape.hh File Reference	992
11.29BulletTrimeshShape.hh File Reference	993
11.30BulletTypes.hh File Reference	994
11.30.1 Detailed Description	995
11.31BulletUniversalJoint.hh File Reference	995
11.32BVHLoader.hh File Reference	996
11.32.1 Macro Definition Documentation	997
11.32.1.1 X_POSITION	997
11.32.1.2 X_ROTATION	997
11.32.1.3 Y_POSITION	997
11.32.1.4 Y_ROTATION	997
11.32.1.5 Z_POSITION	997
11.32.1.6 Z_ROTATION	997
11.33CallbackHelper.hh File Reference	997
11.34Camera.hh File Reference	999
11.35CameraSensor.hh File Reference	1000
11.36CameraVisual.hh File Reference	1001
11.37cegui.h File Reference	1001
11.38ColladaLoader.hh File Reference	1002
11.39Collision.hh File Reference	1004
11.40CollisionState.hh File Reference	1005
11.41Color.hh File Reference	1005
11.42Common.hh File Reference	1007

11.43CommonTypes.hh File Reference	1008
11.43.1 Macro Definition Documentation	1009
11.43.1.1 GAZEBO_DEPRECATED	1009
11.43.1.2 GAZEBO_FORCEINLINE	1009
11.43.1.3 NULL	1009
11.44COMVisual.hh File Reference	1010
11.45Connection.hh File Reference	1010
11.45.1 Macro Definition Documentation	1012
11.45.1.1 HEADER_LENGTH	1012
11.46ConnectionManager.hh File Reference	1012
11.47Console.hh File Reference	1014
11.48Contact.hh File Reference	1015
11.48.1 Macro Definition Documentation	1016
11.48.1.1 MAX_COLLIDE_RETURNS	1016
11.48.1.2 MAX_CONTACT_JOINTS	1016
11.49ContactSensor.hh File Reference	1016
11.50ContactVisual.hh File Reference	1017
11.51Conversions.hh File Reference	1018
11.52Converter.hh File Reference	1018
11.53CylinderShape.hh File Reference	1019
11.54DepthCamera.hh File Reference	1020
11.55DepthCameraSensor.hh File Reference	1021
11.56Diagnostics.hh File Reference	1022
11.57DynamicLines.hh File Reference	1024
11.58DynamicRenderable.hh File Reference	1024
11.59Entity.hh File Reference	1025
11.60Event.hh File Reference	1026
11.61Events.hh File Reference	1027
11.62Exception.hh File Reference	1027
11.63FPSViewController.hh File Reference	1029
11.64gazebo.hh File Reference	1029
11.65gazebo_core.hh File Reference	1031
11.66GazeboGenerator.hh File Reference	1031
11.67GpuLaser.hh File Reference	1032
11.68GpuRaySensor.hh File Reference	1033
11.69Grid.hh File Reference	1034
11.70Gripper.hh File Reference	1034

11.71	GUIOverlay.hh File Reference	1035
11.72	Heightmap.hh File Reference	1036
11.73	HeightmapShape.hh File Reference	1037
11.74	Helpers.hh File Reference	1038
11.74.1	Macro Definition Documentation	1040
11.74.1.1	GZ_DBL_MAX	1040
11.74.1.2	GZ_DBL_MIN	1040
11.74.1.3	GZ_FLT_MAX	1040
11.74.1.4	GZ_FLT_MIN	1040
11.75	Hinge2Joint.hh File Reference	1040
11.76	HingeJoint.hh File Reference	1041
11.77	Image.hh File Reference	1042
11.78	ImuSensor.hh File Reference	1043
11.79	Inertial.hh File Reference	1044
11.80	IOManager.hh File Reference	1045
11.81	Joint.hh File Reference	1046
11.82	JointController.hh File Reference	1047
11.83	JointFeedback.hh File Reference	1048
11.84	JointState.hh File Reference	1049
11.85	JointVisual.hh File Reference	1050
11.86	KeyFrame.hh File Reference	1051
11.87	LaserVisual.hh File Reference	1052
11.88	Light.hh File Reference	1053
11.89	Link.hh File Reference	1054
11.90	LinkState.hh File Reference	1055
11.91	Logger.hh File Reference	1057
11.92	LogPlay.hh File Reference	1058
11.93	mainpage.html File Reference	1058
11.94	MapShape.hh File Reference	1058
11.95	Master.hh File Reference	1059
11.96	Material.hh File Reference	1060
11.97	Material.hh File Reference	1062
11.98	MathTypes.hh File Reference	1062
11.98.1	Detailed Description	1063
11.99	Matrix3.hh File Reference	1063
11.100	Matrix4.hh File Reference	1064
11.101	Mesh.hh File Reference	1065

11.102	MeshLoader.hh File Reference	1066
11.103	MeshManager.hh File Reference	1068
11.104	Model.hh File Reference	1069
11.105	ModelDatabase.hh File Reference	1070
11.106	ModelState.hh File Reference	1071
11.107	MouseEvent.hh File Reference	1073
11.108	MovableText.hh File Reference	1074
11.109	msgs.hh File Reference	1075
11.110	MultiRayShape.hh File Reference	1077
11.111	Node.hh File Reference	1078
11.112	ode_inc.h File Reference	1080
11.113	ODEBallJoint.hh File Reference	1080
11.114	ODEBoxShape.hh File Reference	1081
11.115	ODECollision.hh File Reference	1082
11.116	ODECylinderShape.hh File Reference	1083
11.117	ODEHeightmapShape.hh File Reference	1083
11.118	ODEHinge2Joint.hh File Reference	1084
11.119	ODEHingeJoint.hh File Reference	1085
11.120	ODEJoint.hh File Reference	1085
11.121	ODELink.hh File Reference	1086
11.122	ODEMultiRayShape.hh File Reference	1087
11.123	ODEPhysics.hh File Reference	1087
11.124	ODEPlaneShape.hh File Reference	1088
11.125	ODERayShape.hh File Reference	1089
11.126	ODEScrewJoint.hh File Reference	1090
11.127	ODESliderJoint.hh File Reference	1090
11.128	ODESphereShape.hh File Reference	1091
11.129	ODESurfaceParams.hh File Reference	1092
11.130	ODETrimeshShape.hh File Reference	1092
11.131	ODETypes.hh File Reference	1093
11.131.1	Detailed Description	1094
11.132	ODEUniversalJoint.hh File Reference	1094
11.133	gre_gazebo.h File Reference	1095
11.134	OrbitViewController.hh File Reference	1096
11.135	Param.hh File Reference	1096
11.136	parser.hh File Reference	1098
11.137	parser_urdf.hh File Reference	1099

11.138	Physics.hh File Reference	1100
11.139	PhysicsEngine.hh File Reference	1101
11.140	PhysicsFactory.hh File Reference	1102
11.141	PhysicsTypes.hh File Reference	1103
11.141.1	Detailed Description	1105
11.141.2	Macro Definition Documentation	1105
11.141.2.1	GZ_ALL_COLLIDE	1105
11.141.2.2	GZ_FIXED_COLLIDE	1105
11.141.2.3	GZ_GHOST_COLLIDE	1105
11.141.2.4	GZ_NONE_COLLIDE	1105
11.141.2.5	GZ_SENSOR_COLLIDE	1105
11.142	ID.hh File Reference	1106
11.143	Plane.hh File Reference	1107
11.144	PlaneShape.hh File Reference	1108
11.145	Plugin.hh File Reference	1109
11.145.1	Macro Definition Documentation	1111
11.145.1.1	GZ_REGISTER_MODEL_PLUGIN	1111
11.145.1.2	GZ_REGISTER_SENSOR_PLUGIN	1111
11.145.1.3	GZ_REGISTER_SYSTEM_PLUGIN	1112
11.145.1.4	GZ_REGISTER_VISUAL_PLUGIN	1112
11.145.1.5	GZ_REGISTER_WORLD_PLUGIN	1112
11.146	Plugin.hh File Reference	1113
11.147	Pose.hh File Reference	1113
11.148	Projector.hh File Reference	1114
11.149	Publication.hh File Reference	1115
11.150	PublicationTransport.hh File Reference	1117
11.151	Publisher.hh File Reference	1119
11.152	Quaternion.hh File Reference	1121
11.153	Band.hh File Reference	1122
11.154	RaySensor.hh File Reference	1123
11.155	RayShape.hh File Reference	1124
11.156	RenderEngine.hh File Reference	1125
11.157	RenderEvents.hh File Reference	1126
11.158	Rendering.h File Reference	1126
11.159	Rendering.hh File Reference	1127
11.160	RenderTypes.hh File Reference	1128
11.160.1	Macro Definition Documentation	1129

11.160.1.1GZ_VISIBILITY_ALL	1129
11.160.1.2GZ_VISIBILITY_GUI	1129
11.160.1.3GZ_VISIBILITY_NOT_SELECTABLE	1130
11.16RFIDSensor.hh File Reference	1130
11.16RFIDTag.hh File Reference	1131
11.16RFIDTagManager.hh File Reference	1132
11.16RFIDTagVisual.hh File Reference	1133
11.16RFIDVisual.hh File Reference	1133
11.16Road.hh File Reference	1134
11.16Road2d.hh File Reference	1135
11.16RotationSpline.hh File Reference	1135
11.16RTShaderSystem.hh File Reference	1137
11.17Scene.hh File Reference	1137
11.17ScrewJoint.hh File Reference	1138
11.17Sdf.hh File Reference	1139
11.17SDF.hh File Reference	1140
11.173. Macro Definition Documentation	1141
11.173.1.SDF_VERSION	1141
11.17SelectionObj.hh File Reference	1141
11.17Sensor.hh File Reference	1142
11.17SensorFactory.hh File Reference	1143
11.17SensorManager.hh File Reference	1144
11.17Sensors.hh File Reference	1145
11.17SensorTypes.hh File Reference	1147
11.179. Detailed Description	1148
11.18Server.hh File Reference	1148
11.18Shape.hh File Reference	1149
11.18SingletonT.hh File Reference	1150
11.18Skeleton.hh File Reference	1151
11.18SkeletonAnimation.hh File Reference	1153
11.18SliderJoint.hh File Reference	1154
11.18SphereShape.hh File Reference	1155
11.18Spline.hh File Reference	1156
11.18State.hh File Reference	1158
11.18STLLoader.hh File Reference	1159
11.189. Macro Definition Documentation	1160
11.189.1.COR3_MAX	1160

11.189.1.2	FACE_MAX	1161
11.189.1.3	LINE_MAX_LEN	1161
11.189.1.4	ORDER_MAX	1161
11.196	SubscribeOptions.hh File Reference	1161
11.196	Subscriber.hh File Reference	1163
11.198	SubscriptionTransport.hh File Reference	1165
11.198	SurfaceParams.hh File Reference	1166
11.198	SystemPaths.hh File Reference	1167
11.194.1	Macro Definition Documentation	1168
11.194.1.1	GetCurrentDir	1168
11.194.1.2	LINUX	1168
11.195	Time.hh File Reference	1168
11.195	Timer.hh File Reference	1169
11.197	TopicManager.hh File Reference	1170
11.198	Transport.hh File Reference	1172
11.199	TransportTypes.hh File Reference	1174
11.199.1	Detailed Description	1175
11.200	TrimeshShape.hh File Reference	1175
11.200	UniversalJoint.hh File Reference	1176
11.202	UserCamera.hh File Reference	1177
11.203	Vector2d.hh File Reference	1178
11.204	Vector2i.hh File Reference	1179
11.205	Vector3.hh File Reference	1180
11.206	Vector4.hh File Reference	1181
11.207	Video.hh File Reference	1183
11.208	VideoVisual.hh File Reference	1184
11.209	ViewController.hh File Reference	1185
11.210	Visual.hh File Reference	1186
11.211	WindowManager.hh File Reference	1187
11.212	World.hh File Reference	1188
11.213	WorldState.hh File Reference	1190

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. 430) (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. 258) () const

to be implemented

Member gazebo::sensors::GpuRaySensor::Get2ndRatio (p. 382) ()

Document me

Member gazebo::sensors::GpuRaySensor::GetCHFOV (p. 383) ()

Document me

Member gazebo::sensors::GpuRaySensor::GetCVFOV (p. 383) ()

Document me

Member gazebo::sensors::GpuRaySensor::GetHAngle (p. 383) ()

Document me

Member gazebo::sensors::GpuRaySensor::GetHFOV (p. 383) ()

Document me

Member gazebo::sensors::GpuRaySensor::GetVAngle (p. 385) ()

Document me

Member gazebo::sensors::GpuRaySensor::GetVFOV (p. 386) ()

Document me

Member gazebo::sensors::ImuSensor::GetVelocity (p. 414) ()

storing x,y,z components in a quaternion seems like a bad idea

Class gazebo::SystemPlugin (p. 838)

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

Member urdf2gazebo::URDF2Gazebo::parseGazeboExtension (p. 875) (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	31
Events	39
Math	53
Messages	59
Classes for physics and dynamics	68
Bullet Physics	75
ODE Physics	77
Rendering	79
Gazebo_parser	82
Sensors	83
Transport	88

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost	93
gazebo	
Forward declarations for the common classes	93
gazebo::common	
Common namespace	95
gazebo::event	
Event (p. 350) namespace	98
gazebo::math	
Math namespace	99
gazebo::msgs	
Messages namespace	101
gazebo::physics	
Namespace for physics	103
gazebo::rendering	
Rendering namespace	109
gazebo::sensors	
Sensors namespace	113
gazebo::transport	
Transport namespace	115
google	117
google::protobuf	117
google::protobuf::compiler	117
google::protobuf::compiler::cpp	118
Ogre	118
ogre	118
sdf	
Namespace for Simulation Description Format parser	118
SkyX	120
urdf2gazebo	
Namespace for URDF to SDF parser	120

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

gazebo::math::Angle	127
gazebo::common::Animation	135
gazebo::common::NumericAnimation	569
gazebo::common::PoseAnimation	685
gazebo::math::Box	158
btMotionState	
gazebo::physics::BulletMotionState	198
gazebo::common::BVHLoader	229
gazebo::transport::CallbackHelper	230
gazebo::transport::CallbackHelperT< M >	231
gazebo::transport::DebugCallbackHelper	311
gazebo::transport::SubscriptionTransport	828
CodeGenerator	
google::protobuf::compiler::cpp::GazeboGenerator	374
gazebo::common::Color	274
gazebo::event::Connection	287
gazebo::common::Console	295
gazebo::physics::Contact	296
gazebo::physics::ContactFeedback	299
gazebo::rendering::Conversions	305
sdf::Converter	307
enable_shared_from_this	
gazebo::physics::Base	145
gazebo::physics::Entity	338
gazebo::physics::Collision	262
gazebo::physics::BulletCollision	171
gazebo::physics::ODECollision	578
gazebo::physics::Link	454
gazebo::physics::BulletLink	192
gazebo::physics::ODELink	599
gazebo::physics::Model	521
gazebo::physics::Actor	121
gazebo::physics::Joint	423

gazebo::physics::BulletJoint	188
gazebo::physics::BallJoint< BulletJoint >	143
gazebo::physics::BulletBallJoint	166
gazebo::physics::Hinge2Joint< BulletJoint >	404
gazebo::physics::BulletHinge2Joint	178
gazebo::physics::HingeJoint< BulletJoint >	405
gazebo::physics::BulletHingeJoint	183
gazebo::physics::ScrewJoint< BulletJoint >	760
gazebo::physics::BulletScrewJoint	213
gazebo::physics::SliderJoint< BulletJoint >	804
gazebo::physics::BulletSliderJoint	217
gazebo::physics::UniversalJoint< BulletJoint >	868
gazebo::physics::BulletUniversalJoint	224
gazebo::physics::ODEJoint	594
gazebo::physics::BallJoint< ODEJoint >	143
gazebo::physics::ODEBallJoint	573
gazebo::physics::Hinge2Joint< ODEJoint >	404
gazebo::physics::ODEHinge2Joint	586
gazebo::physics::HingeJoint< ODEJoint >	405
gazebo::physics::ODEHingeJoint	590
gazebo::physics::ScrewJoint< ODEJoint >	760
gazebo::physics::ODEScrewJoint	622
gazebo::physics::SliderJoint< ODEJoint >	804
gazebo::physics::ODESliderJoint	625
gazebo::physics::UniversalJoint< ODEJoint >	868
gazebo::physics::ODEUniversalJoint	634
gazebo::physics::Road	736
gazebo::physics::Shape	779
gazebo::physics::BoxShape	162
gazebo::physics::BulletBoxShape	169
gazebo::physics::ODEBoxShape	576
gazebo::physics::CylinderShape	307
gazebo::physics::BulletCylinderShape	174
gazebo::physics::ODECylinderShape	582
gazebo::physics::HeightmapShape	399
gazebo::physics::BulletHeightmapShape	176
gazebo::physics::ODEHeightmapShape	584
gazebo::physics::MapShape	483
gazebo::physics::MultiRayShape	549
gazebo::physics::BulletMultiRayShape	200
gazebo::physics::ODEMultiRayShape	607
gazebo::physics::PlaneShape	671
gazebo::physics::BulletPlaneShape	207
gazebo::physics::ODEPlaneShape	617
gazebo::physics::RayShape	718
gazebo::physics::BulletRayShape	210
gazebo::physics::ODERayShape	619
gazebo::physics::SphereShape	806
gazebo::physics::BulletSphereShape	221
gazebo::physics::ODESphereShape	629
gazebo::physics::TrimeshShape	866

gazebo::physics::BulletTrimeshShape	222
gazebo::physics::ODETrimeshShape	632
gazebo::physics::World	954
gazebo::rendering::Camera	233
gazebo::rendering::DepthCamera	312
gazebo::rendering::GpuLaser	375
gazebo::rendering::UserCamera	878
gazebo::rendering::Scene	746
gazebo::rendering::Visual	931
gazebo::rendering::ArrowVisual	139
gazebo::rendering::AxisVisual	140
gazebo::rendering::CameraVisual	259
gazebo::rendering::COMVisual	285
gazebo::rendering::ContactVisual	304
gazebo::rendering::JointVisual	444
gazebo::rendering::LaserVisual	447
gazebo::rendering::RFIDTagVisual	733
gazebo::rendering::RFIDVisual	734
gazebo::rendering::VideoVisual	926
gazebo::sensors::Sensor	765
gazebo::sensors::CameraSensor	255
gazebo::sensors::ContactSensor	300
gazebo::sensors::DepthCameraSensor	317
gazebo::sensors::GpuRaySensor	378
gazebo::sensors::ImuSensor	412
gazebo::sensors::RaySensor	711
gazebo::sensors::RFIDSensor	727
gazebo::sensors::RFIDTag	729
gazebo::transport::Connection	288
gazebo::transport::Node	556
sdf::Element	332
gazebo::event::Event	350
gazebo::event::EventT< void()>	356
gazebo::event::EventT< void(bool)>	356
gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>	356
gazebo::event::EventT< void(const std::string &)>	356
gazebo::event::EventT< void(const std::string &, const Contact &)>	356
gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>	356
gazebo::event::EventT< void(std::string)>	356
gazebo::event::EventT< void(std::string, std::string)>	356
gazebo::event::EventT< T >	356
gazebo::event::Events	352
gazebo::rendering::Events	354
gazebo::common::Exception	365
urdf2gazebo::GazeboExtension	369
gazebo::rendering::Grid	389
gazebo::physics::Gripper	393
gazebo::rendering::GUIOverlay	394
gazebo::rendering::Heightmap	398
gazebo::common::Image	407
gazebo::physics::Inertial	414
gazebo::transport::IOManager	422
gazebo::physics::JointController	438

gazebo::physics::JointFeedback	440
gazebo::common::KeyFrame	445
gazebo::common::NumericKeyFrame	571
gazebo::common::PoseKeyFrame	687
gazebo::rendering::Light	448
gazebo::physics::Logplay	479
gazebo::Master	487
gazebo::common::Material	489
gazebo::math::Matrix3	498
gazebo::math::Matrix4	501
gazebo::common::Mesh	509
gazebo::common::MeshLoader	514
gazebo::common::ColladaLoader	261
gazebo::common::STLLoader	816
gazebo::common::ModelDatabase	533
gazebo::common::MouseEvent	540
MovableObject	
gazebo::rendering::MovableText	543
gazebo::common::NodeAnimation	560
gazebo::common::NodeAssignment	563
gazebo::common::NodeTransform	564
gazebo::physics::ODESurfaceParams	631
sdf::Param	642
sdf::ParamT< std::string >	648
sdf::ParamT< T >	648
ParamT< T >	648
gazebo::physics::PhysicsEngine	651
gazebo::physics::BulletPhysics	202
gazebo::physics::ODEPhysics	609
gazebo::physics::PhysicsFactory	663
PhysicsRaySensor	
gazebo::physics::BulletRaySensor	208
gazebo::common::PID	664
gazebo::math::Plane	668
gazebo::PluginT< T >	675
gazebo::PluginT< ModelPlugin >	675
gazebo::ModelPlugin	534
gazebo::PluginT< SensorPlugin >	675
gazebo::SensorPlugin	777
gazebo::PluginT< SystemPlugin >	675
gazebo::SystemPlugin	838
gazebo::PluginT< VisualPlugin >	675
gazebo::VisualPlugin	949
gazebo::PluginT< WorldPlugin >	675
gazebo::WorldPlugin	965
gazebo::math::Pose	677
gazebo::rendering::Projector	689
gazebo::transport::Publication	692
gazebo::transport::PublicationTransport	694
gazebo::transport::Publisher	695
gazebo::math::Quaternion	697

gazebo::math::Rand	710
Renderable	
gazebo::rendering::MovableText	543
RenderObjectListener	
gazebo::rendering::GpuLaser	375
Road	736
gazebo::rendering::Road2d	737
gazebo::math::RotationSpline	738
sdf::SDF	762
SDFBase	
sdf::Plugin	674
gazebo::rendering::SelectionObj	763
SensorFactor	772
gazebo::sensors::SensorFactory	772
gazebo::Server	778
SimpleRenderable	
gazebo::rendering::DynamicRenderable	328
gazebo::rendering::DynamicLines	324
SingletonT< T >	782
gazebo::common::DiagnosticManager	320
gazebo::common::MeshManager	516
gazebo::common::SystemPaths	834
gazebo::physics::Logger	477
gazebo::rendering::RenderEngine	722
gazebo::rendering::RTShaderSystem	742
gazebo::rendering::WindowManager	951
gazebo::sensors::RFIDTagManager	732
gazebo::sensors::SensorManager	774
gazebo::transport::ConnectionManager	292
gazebo::transport::TopicManager	861
SingletonT< ConnectionManager >	782
SingletonT< DiagnosticManager >	782
SingletonT< Logger >	782
SingletonT< MeshManager >	782
SingletonT< RenderEngine >	782
SingletonT< RFIDTagManager >	782
SingletonT< RTShaderSystem >	782
SingletonT< SensorManager >	782
SingletonT< SystemPaths >	782
SingletonT< TopicManager >	782
SingletonT< WindowManager >	782
gazebo::common::Skeleton	784
gazebo::common::SkeletonAnimation	791
gazebo::common::SkeletonNode	795
gazebo::math::Spline	808
gazebo::physics::State	813
gazebo::physics::CollisionState	272
gazebo::physics::JointState	442
gazebo::physics::LinkState	473
gazebo::physics::ModelState	536
gazebo::physics::WorldState	967
gazebo::common::SubMesh	817
gazebo::transport::SubscribeOptions	826
gazebo::transport::Subscriber	827

gazebo::physics::SurfaceParams	830
gazebo::common::Time	840
gazebo::common::Timer	859
gazebo::common::DiagnosticTimer	323
gazebo::physics::TrajectoryInfo	865
urdf2gazebo::URDF2Gazebo	870
gazebo::math::Vector2d	885
gazebo::math::Vector2i	894
gazebo::math::Vector3	902
gazebo::math::Vector4	915
gazebo::common::Video	924
gazebo::rendering::ViewController	927
gazebo::rendering::FPSViewController	367
gazebo::rendering::OrbitViewController	638
T	
gazebo::physics::BallJoint< T >	143
gazebo::physics::Hinge2Joint< T >	404
gazebo::physics::HingeJoint< T >	405
gazebo::physics::ScrewJoint< T >	760
gazebo::physics::SliderJoint< T >	804
gazebo::physics::UniversalJoint< T >	868

Chapter 6

Class Index

6.1 Class List

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

gazebo::physics::Actor	
Actor (p. 121) class enables GPU based mesh model / skeleton scriptable animation	121
gazebo::math::Angle	
An angle and related functions	127
gazebo::common::Animation	
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes	135
gazebo::rendering::ArrowVisual	
Basic arrow visualization	139
gazebo::rendering::AxisVisual	
Basic axis visualization	140
gazebo::physics::BallJoint< T >	
Base (p. 145) class for a ball joint	143
gazebo::physics::Base	
Base (p. 145) class for most physics classes	145
gazebo::math::Box	
Mathematical representation of a box and related functions	158
gazebo::physics::BoxShape	
Box geometry primitive	162
gazebo::physics::BulletBallJoint	
BulletBallJoint (p. 166) class models a ball joint in Bullet	166
gazebo::physics::BulletBoxShape	
Bullet box collision	169
gazebo::physics::BulletCollision	
Bullet collisions	171
gazebo::physics::BulletCylinderShape	
Cylinder collision	174
gazebo::physics::BulletHeightmapShape	
Height map collision	176
gazebo::physics::BulletHinge2Joint	
A two axis hinge joint	178
gazebo::physics::BulletHingeJoint	
A single axis hinge joint	183

gazebo::physics::BulletJoint	
Base (p. 145) class for all joints	188
gazebo::physics::BulletLink	
Bullet Link (p. 454) class	192
gazebo::physics::BulletMotionState	
Bullet btMotionState encapsulation	198
gazebo::physics::BulletMultiRayShape	
Bullet specific version of MultiRayShape (p. 549)	200
gazebo::physics::BulletPhysics	
Bullet physics engine	202
gazebo::physics::BulletPlaneShape	
Bullet collision for an infinite plane	207
gazebo::physics::BulletRaySensor	
An Bullet Ray sensor	208
gazebo::physics::BulletRayShape	
Ray shape for bullet	210
gazebo::physics::BulletScrewJoint	
A screw joint	213
gazebo::physics::BulletSliderJoint	
A slider joint	217
gazebo::physics::BulletSphereShape	
Bullet sphere collision	221
gazebo::physics::BulletTrimeshShape	
Triangle mesh collision	222
gazebo::physics::BulletUniversalJoint	
A bullet universal joint class	224
gazebo::common::BVHLoader	
Handles loading BVH animation files	229
gazebo::transport::CallbackHelper	
A helper class to handle callbacks when messages arrive	230
gazebo::transport::CallbackHelperT< M >	
Callback helper Template	231
gazebo::rendering::Camera	
Basic camera sensor	233
gazebo::sensors::CameraSensor	255
gazebo::rendering::CameraVisual	
Basic camera visualization	259
gazebo::common::ColladaLoader	
Class used to load Collada mesh files	261
gazebo::physics::Collision	
Base (p. 145) class for all collision entities	262
gazebo::physics::CollisionState	
Store state information of a physics::Collision (p. 262) object	272
gazebo::common::Color	
Defines a color	274
gazebo::rendering::COMVisual	
Basic Center of Mass visualization	285
gazebo::event::Connection	
A class that encapsulates a connection	287
gazebo::transport::Connection	
TCP/IP Connection (p. 288)	288
gazebo::transport::ConnectionManager	
Manager of connections	292

gazebo::common::Console	
Message, error, warning, and logging functionality	295
gazebo::physics::Contact	
A contact between two collisions	296
gazebo::physics::ContactFeedback	
Data structure for contact feedbacks	299
gazebo::sensors::ContactSensor	300
gazebo::rendering::ContactVisual	
Contact visualization	304
gazebo::rendering::Conversions	
Conversions (p. 305) Conversions.hh (p. 1018) rendering/Conversions.hh (p. 1018)	305
sdf::Converter	
Convert from one version of SDF (p. 762) to another	307
gazebo::physics::CylinderShape	
Cylinder collision	307
gazebo::transport::DebugCallbackHelper	311
gazebo::rendering::DepthCamera	
Depth camera used to render depth data into an image buffer	312
gazebo::sensors::DepthCameraSensor	317
gazebo::common::DiagnosticManager	
A diagnostic manager class	320
gazebo::common::DiagnosticTimer	
A timer designed for diagnostics	323
gazebo::rendering::DynamicLines	
Class for drawing lines that can change	324
gazebo::rendering::DynamicRenderable	
Abstract base class providing mechanisms for dynamically growing hardware buffers	328
sdf::Element	
SDF (p. 762) Element (p. 332) class	332
gazebo::physics::Entity	
Base (p. 145) class for all physics objects in Gazebo	338
gazebo::event::Event	
Base class for all events	350
gazebo::event::Events	
An Event (p. 350) class to get notifications for simulator events	352
gazebo::rendering::Events	
Base class for rendering events	354
gazebo::event::EventT < T >	
A class for event processing	356
gazebo::common::Exception	
Class for generating exceptions	365
gazebo::rendering::FPSViewController	
First Person Shooter style view controller	367
urdf2gazebo::GazeboExtension	369
google::protobuf::compiler::cpp::GazeboGenerator	
Google protobuf message generator for gazebo::msgs (p. 101)	374
gazebo::rendering::GpuLaser	
GPU based laser distance sensor	375
gazebo::sensors::GpuRaySensor	378
gazebo::rendering::Grid	
Displays a grid of cells, drawn with lines	389
gazebo::physics::Gripper	
A gripper abstraction	393

gazebo::rendering::GUIOverlay	A class that creates a CEGUI overlay on a render window	394
gazebo::rendering::Heightmap	Rendering a terrain using heightmap information	398
gazebo::physics::HeightmapShape	HeightmapShape (p. 399) collision shape builds a heightmap from an image	399
gazebo::physics::Hinge2Joint< T >	A two axis hinge joint	404
gazebo::physics::HingeJoint< T >	A single axis hinge joint	405
gazebo::common::Image	Encapsulates an image	407
gazebo::sensors::ImuSensor	An IMU sensor	412
gazebo::physics::Inertial	A class for inertial information about a link	414
gazebo::transport::IOManager	Managers boost::asio IO	422
gazebo::physics::Joint	Base (p. 145) class for all joints	423
gazebo::physics::JointController	A class for manipulating physics::Joint (p. 423)	438
gazebo::physics::JointFeedback	Feedback information from a joint	440
gazebo::physics::JointState	Keeps track of state of a physics::Joint (p. 423)	442
gazebo::rendering::JointVisual	Visualization for joints	444
gazebo::common::KeyFrame	A key frame in an animation	445
gazebo::rendering::LaserVisual	Visualization for laser data	447
gazebo::rendering::Light	A light source	448
gazebo::physics::Link	Link (p. 454) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body	454
gazebo::physics::LinkState	Store state information of a physics::Link (p. 454) object	473
gazebo::physics::Logger	Handles logging of data to disk	477
gazebo::physics::Logplay	Open and playback log files that were recorded using Logger (p. 477)	479
gazebo::physics::MapShape	Creates box extrusions based on an image	483
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	487
gazebo::common::Material	Encapsulates description of a material	489
gazebo::math::Matrix3	A 3x3 matrix class	498
gazebo::math::Matrix4	A 3x3 matrix class	501

gazebo::common::Mesh	
A 3D mesh	509
gazebo::common::MeshLoader	
Base class for loading meshes	514
gazebo::common::MeshManager	
Maintains and manages all meshes	516
gazebo::physics::Model	
A model is a collection of links, joints, and plugins	521
gazebo::common::ModelDatabase	
Connects to model database, and has utility functions to find models	533
gazebo::ModelPlugin	
A plugin with access to physics::Model (p. 521)	534
gazebo::physics::ModelState	
Store state information of a physics::Model (p. 521) object	536
gazebo::common::MouseEvent	
Generic description of a mouse event	540
gazebo::rendering::MovableText	
Movable text	543
gazebo::physics::MultiRayShape	
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner	549
gazebo::transport::Node	
A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics	556
gazebo::common::NodeAnimation	
Node animation	560
gazebo::common::NodeAssignment	
Vertex to node weighted assignment for skeleton animation visualization	563
gazebo::common::NodeTransform	
A transformation node	564
gazebo::common::NumericAnimation	
A numeric animation	569
gazebo::common::NumericKeyFrame	
A keyframe for a NumericAnimation (p. 569)	571
gazebo::physics::ODEBallJoint	
An ODEBallJoint (p. 573)	573
gazebo::physics::ODEBoxShape	
ODE Box shape	576
gazebo::physics::ODECollision	
Base (p. 145) class for all ODE collisions	578
gazebo::physics::ODECylinderShape	
ODE cylinder shape	582
gazebo::physics::ODEHeightmapShape	
ODE Height map collision	584
gazebo::physics::ODEHinge2Joint	
A two axis hinge joint	586
gazebo::physics::ODEHingeJoint	
A single axis hinge joint	590
gazebo::physics::ODEJoint	
ODE joint interface	594
gazebo::physics::ODELink	
ODE Link (p. 454) class	599
gazebo::physics::ODEMultiRayShape	
ODE specific version of MultiRayShape (p. 549)	607

gazebo::physics::ODEPhysics	
ODE physics engine	609
gazebo::physics::ODEPlaneShape	
An ODE Plane shape	617
gazebo::physics::ODERayShape	
Ray collision	619
gazebo::physics::ODEScrewJoint	
A screw joint	622
gazebo::physics::ODESliderJoint	
A slider joint	625
gazebo::physics::ODESphereShape	
A ODE sphere shape	629
gazebo::physics::ODESurfaceParams	
Surface params	631
gazebo::physics::ODETrimeshShape	
Triangle mesh collision	632
gazebo::physics::ODEUniversalJoint	
A universal joint	634
gazebo::rendering::OrbitViewController	
Orbit view controller	638
sdf::Param	
A parameter class	642
ParamT< T >	648
sdf::ParamT< T >	
Templatized parameter class	648
gazebo::physics::PhysicsEngine	
Base (p. 145) class for a physics engine	651
gazebo::physics::PhysicsFactory	
The physics factory instantiates different physics engines	663
gazebo::common::PID	
Generic PID (p. 664) controller class	664
gazebo::math::Plane	
A plane and related functions	668
gazebo::physics::PlaneShape	
Collision (p. 262) for an infinite plane	671
sdf::Plugin	674
gazebo::PluginT< T >	
A class which all plugins must inherit from	675
gazebo::math::Pose	
Encapsulates a position and rotation in three space	677
gazebo::common::PoseAnimation	
A pose animation	685
gazebo::common::PoseKeyFrame	
A keyframe for a PoseAnimation (p. 685)	687
gazebo::rendering::Projector	
Projects a material onto surface, light a light projector	689
gazebo::transport::Publication	
A publication for a topic	692
gazebo::transport::PublicationTransport	
Reads data from a remote advertiser, and passes the data along to local subscribers	694
gazebo::transport::Publisher	
A publisher of messages on a topic	695
gazebo::math::Quaternion	
A quaternion class	697

gazebo::math::Rand	
Random number generator class	710
gazebo::sensors::RaySensor	711
gazebo::physics::RayShape	
Base (p. 145) class for Ray collision geometry	718
gazebo::rendering::RenderEngine	
Adaptor to Ogre3d	722
gazebo::sensors::RFIDSensor	
Sensor (p. 765) class for RFID type of sensor	727
gazebo::sensors::RFIDTag	
RFIDTag (p. 729) to interact with RFIDTagSensors Nate check	729
gazebo::sensors::RFIDTagManager	
Nate fill in	732
gazebo::rendering::RFIDTagVisual	
Visualization for RFID tags sensor	733
gazebo::rendering::RFIDVisual	
Visualization for RFID sensor	734
Road	
Used to render a strip of road	736
gazebo::physics::Road	
For building a Road (p. 736) from SDF	736
gazebo::rendering::Road2d	737
gazebo::math::RotationSpline	
Spline (p. 808) for rotations	738
gazebo::rendering::RTShaderSystem	
Implements Ogre (p. 118)'s Run-Time Shader system	742
gazebo::rendering::Scene	
Representation of an entire scene graph	746
gazebo::physics::ScrewJoint< T >	
A screw joint	760
sdf::SDF	
Base SDF (p. 762) class	762
gazebo::rendering::SelectionObj	
A graphical selection object	763
gazebo::sensors::Sensor	765
SensorFactor	
The sensor factory; the class is just for namespacing purposes	772
gazebo::sensors::SensorFactory	772
gazebo::sensors::SensorManager	
Class to manage and update all sensors	774
gazebo::SensorPlugin	
A plugin with access to physics::Sensor	777
gazebo::Server	778
gazebo::physics::Shape	
Base (p. 145) class for all shapes	779
SingletonT< T >	
Singleton template class	782
gazebo::common::Skeleton	
A skeleton	784
gazebo::common::SkeletonAnimation	
Skeleton (p. 784) animation	791
gazebo::common::SkeletonNode	
A skeleton node	795

gazebo::physics::SliderJoint< T >	
A slider joint	804
gazebo::physics::SphereShape	
Sphere collision	806
gazebo::math::Spline	
Splines	808
gazebo::physics::State	
State (p. 813) of an entity	813
gazebo::common::STLLoader	
Class used to load STL mesh files	816
gazebo::common::SubMesh	
A child mesh	817
gazebo::transport::SubscribeOptions	
Options for a subscription	826
gazebo::transport::Subscriber	
A subscriber to a topic	827
gazebo::transport::SubscriptionTransport	
Handles sending data over the wire to remote subscribers	828
gazebo::physics::SurfaceParams	
SurfaceParams (p. 830) defines various Surface contact parameters	830
gazebo::common::SystemPaths	
Functions to handle getting system paths, keeps track of:	834
gazebo::SystemPlugin	
A plugin loaded within the gzserver on startup	838
gazebo::common::Time	
A Time (p. 840) class, can be used to hold wall- or sim-time	840
gazebo::common::Timer	
A timer class, used to time things in real world walltime	859
gazebo::transport::TopicManager	
Manages topics and their subscriptions	861
gazebo::physics::TrajectoryInfo	865
gazebo::physics::TrimeshShape	
Triangle mesh collision shape	866
gazebo::physics::UniversalJoint< T >	
A universal joint	868
urdf2gazebo::URDF2Gazebo	870
gazebo::rendering::UserCamera	
A camera used for user visualization of a scene	878
gazebo::math::Vector2d	
Generic double x, y vector	885
gazebo::math::Vector2i	
Generic integer x, y vector	894
gazebo::math::Vector3	
Generic vector containing 3 elements	902
gazebo::math::Vector4	
Double Generic x, y, z, w vector	915
gazebo::common::Video	
Handle video encoding and decoding using libavcodec	924
gazebo::rendering::VideoVisual	
A visual element that displays a video as a texture	926
gazebo::rendering::ViewController	
Base class for view controllers	927
gazebo::rendering::Visual	
A renderable object	931

gazebo::VisualPlugin	
A plugin loaded within the gzserver on startup	949
gazebo::rendering::WindowManager	
Class to manage render windows	951
gazebo::physics::World	
The world provides access to all other object within a simulated environment	954
gazebo::WorldPlugin	
A plugin with access to physics::World (p. 954)	965
gazebo::physics::WorldState	
Store state information of a physics::World (p. 954) object	967

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Actor.hh	971
Angle.hh	972
Animation.hh	974
ArrowVisual.hh	975
AxisVisual.hh	976
BallJoint.hh	976
Base.hh	977
Box.hh	978
BoxShape.hh	979
bullet_inc.h	980
BulletBallJoint.hh	981
BulletBoxShape.hh	982
BulletCollision.hh	982
BulletCylinderShape.hh	983
BulletHeightmapShape.hh	984
BulletHinge2Joint.hh	984
BulletHingeJoint.hh	985
BulletJoint.hh	985
BulletLink.hh	986
BulletMotionState.hh	987
BulletMultiRayShape.hh	988
BulletPhysics.hh	988
BulletPlaneShape.hh	989
BulletRaySensor.hh	990
BulletRayShape.hh	990
BulletScrewJoint.hh	991
BulletSliderJoint.hh	992
BulletSphereShape.hh	992
BulletTrimeshShape.hh	993
BulletTypes.hh	
Bullet wrapper forward declarations and typedefs	994
BulletUniversalJoint.hh	995
BVHLoader.hh	996
CallbackHelper.hh	997

Camera.hh	999
CameraSensor.hh	1000
CameraVisual.hh	1001
cegui.h	1001
ColladaLoader.hh	1002
Collision.hh	1004
CollisionState.hh	1005
Color.hh	1005
Common.hh	1007
CommonTypes.hh	1008
COMVisual.hh	1010
Connection.hh	1010
ConnectionManager.hh	1012
Console.hh	1014
Contact.hh	1015
ContactSensor.hh	1016
ContactVisual.hh	1017
Conversions.hh	1018
Converter.hh	1018
CylinderShape.hh	1019
DepthCamera.hh	1020
DepthCameraSensor.hh	1021
Diagnostics.hh	1022
DynamicLines.hh	1024
DynamicRenderable.hh	1024
Entity.hh	1025
Event.hh	1026
Events.hh	1027
Exception.hh	1027
FPSViewController.hh	1029
gazebo.hh	1029
gazebo_core.hh	1031
GazeboGenerator.hh	1031
GpuLaser.hh	1032
GpuRaySensor.hh	1033
Grid.hh	1034
Gripper.hh	1034
GUIOverlay.hh	1035
Heightmap.hh	1036
HeightmapShape.hh	1037
Helpers.hh	1038
Hinge2Joint.hh	1040
HingeJoint.hh	1041
Image.hh	1042
ImuSensor.hh	1043
Inertial.hh	1044
IOManager.hh	1045
Joint.hh	1046
JointController.hh	1047
JointFeedback.hh	1048
JointState.hh	1049
JointVisual.hh	1050
KeyFrame.hh	1051
LaserVisual.hh	1052

Light.hh	1053
Link.hh	1054
LinkState.hh	1055
Logger.hh	1057
LogPlay.hh	1058
mainpage.html	1058
MapShape.hh	1058
Master.hh	1059
common/Material.hh	1060
rendering/Material.hh	1062
MathTypes.hh	
Forward declarations for the math classes	1062
Matrix3.hh	1063
Matrix4.hh	1064
Mesh.hh	1065
MeshLoader.hh	1066
MeshManager.hh	1068
Model.hh	1069
ModelDatabase.hh	1070
ModelState.hh	1071
MouseEvent.hh	1073
MovableText.hh	1074
msgs.hh	1075
MultiRayShape.hh	1077
Node.hh	1078
ode_inc.h	1080
ODEBallJoint.hh	1080
ODEBoxShape.hh	1081
ODECollision.hh	1082
ODECylinderShape.hh	1083
ODEHeightmapShape.hh	1083
ODEHinge2Joint.hh	1084
ODEHingeJoint.hh	1085
ODEJoint.hh	1085
ODELink.hh	1086
ODEMultiRayShape.hh	1087
ODEPhysics.hh	1087
ODEPlaneShape.hh	1088
ODERayShape.hh	1089
ODEScrewJoint.hh	1090
ODESliderJoint.hh	1090
ODESphereShape.hh	1091
ODESurfaceParams.hh	1092
ODETrimeshShape.hh	1092
ODETypes.hh	
ODE wrapper forward declarations and typedefs	1093
ODEUniversalJoint.hh	1094
ogre_gazebo.h	1095
OrbitViewController.hh	1096
Param.hh	1096
parser.hh	1098
parser_urdf.hh	1099
Physics.hh	1100
PhysicsEngine.hh	1101

PhysicsFactory.hh	1102
PhysicsTypes.hh	
Default namespace for gazebo	1103
PID.hh	1106
Plane.hh	1107
PlaneShape.hh	1108
common/Plugin.hh	1109
sdf/interface/Plugin.hh	1113
Pose.hh	1113
Projector.hh	1114
Publication.hh	1115
PublicationTransport.hh	1117
Publisher.hh	1119
Quaternion.hh	1121
Rand.hh	1122
RaySensor.hh	1123
RayShape.hh	1124
RenderEngine.hh	1125
RenderEvents.hh	1126
rendering.h	1126
Rendering.hh	1127
RenderTypes.hh	1128
RFIDSensor.hh	1130
RFIDTag.hh	1131
RFIDTagManager.hh	1132
RFIDTagVisual.hh	1133
RFIDVisual.hh	1133
Road.hh	1134
Road2d.hh	1135
RotationSpline.hh	1135
RTShaderSystem.hh	1137
Scene.hh	1137
ScrewJoint.hh	1138
sdf.hh	1139
SDF.hh	1140
SelectionObj.hh	1141
Sensor.hh	1142
SensorFactory.hh	1143
SensorManager.hh	1144
Sensors.hh	1145
SensorTypes.hh	
Forward declarations and typedefs for sensors	1147
Server.hh	1148
Shape.hh	1149
SingletonT.hh	1150
Skeleton.hh	1151
SkeletonAnimation.hh	1153
SliderJoint.hh	1154
SphereShape.hh	1155
Spline.hh	1156
State.hh	1158
STLLoader.hh	1159
SubscribeOptions.hh	1161
Subscriber.hh	1163

SubscriptionTransport.hh	1165
SurfaceParams.hh	1166
SystemPaths.hh	1167
Time.hh	1168
Timer.hh	1169
TopicManager.hh	1170
Transport.hh	1172
TransportTypes.hh	
Forward declarations for transport	1174
TrimeshShape.hh	1175
UniversalJoint.hh	1176
UserCamera.hh	1177
Vector2d.hh	1178
Vector2i.hh	1179
Vector3.hh	1180
Vector4.hh	1181
Video.hh	1183
VideoVisual.hh	1184
ViewController.hh	1185
Visual.hh	1186
WindowManager.hh	1187
World.hh	1188
WorldState.hh	1190

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, and logging 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. 521).*
- 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**
A transformation node.
- class **gazebo::common::NumericAnimation**
A numeric animation.
- class **gazebo::common::NumericKeyFrame**
*A keyframe for a **NumericAnimation** (p. 569).*
- class **gazebo::common::PID**
*Generic **PID** (p. 664) 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. 685).*
- 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. 784) 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. 840) 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. 954).*

Macros

- #define **DIAG_TIMER**(name) DiagnosticManager::Instance()->CreateTimer(name);
Create an instance of common::DiagnosticManager.
- #define **gzclr_end** "\033[0m"
end marker
- #define **gzclr_start**(clr) "\033[1;33m"
start marker
- #define **gzdbg** (gazebo::common::Console::Instance()->ColorMsg("Dbg", 36))
Output a debug message.
- #define **gzerr**
Output an error message.
- #define **gzlog**
Log a 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

- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)
*add path prefix to **common::SystemPaths** (p. 834)*
- static 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. 834)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 834)*
- static std::string **gazebo::common::ModelDatabase::GetManifest** (const std::string &_uri)
Return the manifest.xml file as a string.
- static std::string **gazebo::common::ModelDatabase::GetModelFile** (const std::string &_uri)
Get a model's SDF file based on a URI.
- static std::string **gazebo::common::ModelDatabase::GetModelName** (const std::string &_uri)
Get the name of a model based on a URI.
- static std::string **gazebo::common::ModelDatabase::GetModelPath** (const std::string &_uri)
Get the local path to a model.
- static std::map< std::string,
std::string > **gazebo::common::ModelDatabase::GetModels** ()
Returns the dictionary of all the model names.
- static std::string **gazebo::common::ModelDatabase::GetURI** ()
Returns the the global model database URI.
- static bool **gazebo::common::ModelDatabase::HasModel** (const std::string &_modelName)
Returns true if the model exists on the database.

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::BulletJoint::GetAnchor(), gazebo::physics::BulletJoint::GetLinkForce(), gazebo::physics::BulletJoint::GetLinkTorque(), gazebo::physics::ScrewJoint< BulletJoint >::Load(), sdf::ParamT< std::string >::Set(), gazebo::physics::BulletJoint::SetAnchor(), gazebo::physics::BulletJoint::SetAttribute(), and gazebo::physics::BulletJoint::SetDamping().

8.1.2.6 #define gzlog

Value:

```
(gazebo::common::Console::Instance()->Log() << "[" <<\
    __FILE__ << ":" << __LINE__ << "]" ")
```

Log a message.

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

Output a message.

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

8.1.2.8 #define gzthrow(msg)

Value:

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

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

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

8.1.2.9 #define gzwarn

Value:

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

Output a warning message.

8.1.3 Typedef Documentation

8.1.3.1 typedef DiagnosticTimer* gazebo::common::DiagnosticTimerPtr

8.1.4 Enumeration Type Documentation

8.1.4.1 enum gazebo::PluginType

Used to specify the type of plugin.

Enumerator:

WORLD_PLUGIN A 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 void gazebo::common::add_search_path_suffix (const std::string & *_suffix*)

add path prefix to **common::SystemPaths** (p. 834)

8.1.5.2 static void gazebo::common::ModelDatabase::DownloadDependencies (const std::string & *_path*) [static]

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

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

search for file in **common::SystemPaths** (p. 834)

Parameters

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

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

search for a file in **common::SystemPaths** (p. 834)

Parameters

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

Returns

The path containing the file

8.1.5.5 `static std::string gazebo::common::ModelDatabase::GetManifest (const std::string & _uri) [static]`

Return the manifest.xml file as a string.

Returns

the manifest file from the model database.

8.1.5.6 `static std::string gazebo::common::ModelDatabase::GetModelFile (const std::string & _uri) [static]`

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

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

Parameters

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

Returns

The full path and filename to the SDF file

8.1.5.7 `static std::string gazebo::common::ModelDatabase::GetModelName (const std::string & _uri) [static]`

Get the name of a model based on a URI.

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

Parameters

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

Returns

the model's name.

8.1.5.8 `static std::string gazebo::common::ModelDatabase::GetModelPath (const std::string & _uri) [static]`

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

8.1.5.9 `static std::map<std::string, std::string> gazebo::common::ModelDatabase::GetModels () [static]`

Returns the dictionary of all the model names.

Returns

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

8.1.5.10 `static std::string gazebo::common::ModelDatabase::GetURI () [static]`

Returns the the global model database URI.

Returns

the URI.

8.1.5.11 `static bool gazebo::common::ModelDatabase::HasModel (const std::string & _modelName) [static]`

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.

8.2 Events

Namespaces

- namespace **gazebo::event**
Event (p. 350) namespace.

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.

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.
- template<typename T >
static ConnectionPtr gazebo::event::Events::ConnectAddEntity (T _subscriber)
Connect a boost::slot the the add entity signal.
- template<typename T >
static ConnectionPtr gazebo::event::Events::ConnectCreateEntity (T _subscriber)
Connect a boost::slot the the add entity signal.
- template<typename T >
static ConnectionPtr gazebo::event::Events::ConnectDeleteEntity (T _subscriber)
Connect a boost::slot the delete entity.
- template<typename T >
static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStart (T _subscriber)
Connect a boost::slot the diagnostic timer start signal.
- template<typename T >
static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStop (T _subscriber)
Connect a boost::slot the diagnostic timer stop signal.
- template<typename T >
static ConnectionPtr gazebo::event::Events::ConnectPause (T _subscriber)
Connect a boost::slot the the pause signal.
- template<typename T >
static ConnectionPtr gazebo::event::Events::ConnectPostRender (T _subscriber)
Connect a boost::slot the post render update signal.
- template<typename T >
static ConnectionPtr gazebo::event::Events::ConnectPreRender (T _subscriber)
Render start signal.
- template<typename T >
static ConnectionPtr gazebo::event::Events::ConnectRender (T _subscriber)

Connect a boost::slot the render update signal.

- `template<typename T >`
`static ConnectionPtr gazebo::event::Events::ConnectSetSelectedEntity (T _subscriber)`
Connect a boost::slot the set selected entity.
- `template<typename T >`
`static ConnectionPtr gazebo::event::Events::ConnectStep (T _subscriber)`
Connect a boost::slot the the step signal.
- `template<typename T >`
`static ConnectionPtr gazebo::event::Events::ConnectStop (T _subscriber)`
Connect a boost::slot the the stop signal.
- `template<typename T >`
`static ConnectionPtr gazebo::event::Events::ConnectWorldCreated (T _subscriber)`
Connect a boost::slot the the world created signal.
- `template<typename T >`
`static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateEnd (T _subscriber)`
Connect a boost::slot the the world update end signal.
- `template<typename T >`
`static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateStart (T _subscriber)`
Connect a boost::slot the the world update start signal.
- `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.
- `static void gazebo::event::Events::DisconnectAddEntity (ConnectionPtr _subscriber)`
Disconnect a boost::slot the the add entity signal.
- `static void gazebo::event::Events::DisconnectCreateEntity (ConnectionPtr _subscriber)`
Disconnect a boost::slot the the add entity signal.
- `static void gazebo::event::Events::DisconnectDeleteEntity (ConnectionPtr _subscriber)`
Disconnect a boost::slot the delete entity.
- `static void gazebo::event::Events::DisconnectDiagTimerStart (ConnectionPtr _subscriber)`
Disconnect a boost::slot the diagnostic timer start signal.
- `static void gazebo::event::Events::DisconnectDiagTimerStop (ConnectionPtr _subscriber)`
Disconnect a boost::slot the diagnostic timer stop signal.
- `static void gazebo::event::Events::DisconnectPause (ConnectionPtr _subscriber)`
Disconnect a boost::slot the the pause signal.
- `static void gazebo::event::Events::DisconnectPostRender (ConnectionPtr _subscriber)`
Disconnect a boost::slot the post render update signal.
- `static void gazebo::event::Events::DisconnectPreRender (ConnectionPtr _subscriber)`
Disconnect a render start signal.
- `static void gazebo::event::Events::DisconnectRender (ConnectionPtr _subscriber)`
Disconnect a boost::slot the render update signal.
- `static void gazebo::event::Events::DisconnectSetSelectedEntity (ConnectionPtr _subscriber)`
Disconnect a boost::slot the set selected entity.
- `static void gazebo::event::Events::DisconnectStep (ConnectionPtr _subscriber)`
Disconnect a boost::slot the the step signal.
- `static void gazebo::event::Events::DisconnectStop (ConnectionPtr _subscriber)`
Disconnect a boost::slot the the stop signal.
- `static void gazebo::event::Events::DisconnectWorldCreated (ConnectionPtr _subscriber)`

Disconnect a boost::slot the the world created signal.

- static void **gazebo::event::Events::DisconnectWorldUpdateEnd** (ConnectionPtr _subscriber)

Disconnect a boost::slot the the world update end signal.

- static void **gazebo::event::Events::DisconnectWorldUpdateStart** (ConnectionPtr _subscriber)

Disconnect a boost::slot the the world update start signal.

Variables

- static EventT< void(std::string)> **gazebo::event::Events::addEntity**
An entity has been added.
- static EventT< void(std::string)> **gazebo::event::Events::deleteEntity**
An entity has been deleted.
- static EventT< void(std::string)> **gazebo::event::Events::diagTimerStart**
Diagnostic timer start.
- static EventT< void(std::string)> **gazebo::event::Events::diagTimerStop**
Diagnostic timer stop.
- static EventT< void(std::string)> **gazebo::event::Events::entityCreated**
An entity has been created.
- static EventT< void(bool)> **gazebo::event::Events::pause**
Pause signal.
- static EventT< void()> **gazebo::event::Events::postRender**
Post-Render.
- static EventT< void()> **gazebo::event::Events::preRender**
Pre-render.
- static EventT< void()> **gazebo::event::Events::render**
Render.
- static EventT< void(std::string, std::string)> **gazebo::event::Events::setSelectedEntity**
An entity has been selected.
- static EventT< void()> **gazebo::event::Events::step**
Step the simulation once signal.
- static EventT< void()> **gazebo::event::Events::stop**
Simulation stop signal.
- static EventT< void(std::string)> **gazebo::event::Events::worldCreated**
A world has been created.
- static EventT< void()> **gazebo::event::Events::worldUpdateEnd**
World update has ended.
- static EventT< void()> **gazebo::event::Events::worldUpdateStart**
World update has started.

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. 287) 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::event::Events::ConnectWorldCreated()`, `gazebo::event::Events::ConnectWorldUpdateEnd()`, and `gazebo::event::Events::ConnectWorldUpdateStart()`.

8.2.2.3 `template<typename T> static ConnectionPtr gazebo::event::Events::ConnectAddEntity (T _subscriber) [inline], [static]`

Connect a boost::slot the the add entity signal.

Parameters

in	_subscriber	the subscriber to this event
----	-------------	------------------------------

Returns

a connection

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

8.2.2.4 `template<typename T> static ConnectionPtr gazebo::event::Events::ConnectCreateEntity (T _subscriber) [inline], [static]`

Connect a boost::slot the the add entity signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

8.2.2.5 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDeleteEntity (T _subscriber) [inline], [static]`

Connect a boost::slot the delete entity.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

8.2.2.6 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStart (T _subscriber) [inline], [static]`

Connect a boost::slot the diagnostic timer start signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

8.2.2.7 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectDiagTimerStop (T _subscriber) [inline], [static]`

Connect a boost::slot the diagnostic timer stop signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

8.2.2.8 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPause (T _subscriber) [inline], [static]`

Connect a boost::slot the the pause signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

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

8.2.2.9 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPostRender (T _subscriber) [inline], [static]`

Connect a boost::slot the post render update signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

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

8.2.2.10 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectPreRender (T _subscriber) [inline], [static]`

Render start signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

Returns

a connection

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

8.2.2.11 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectRender (T _subscriber) [inline], [static]`

Connect a boost::slot the render update signal.

Parameters

in	<i>_subscriber</i>	the subscriber to this event
----	--------------------	------------------------------

Returns

a connection

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

8.2.2.12 `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 gazebo::event::Events::setSelectedEntity.

8.2.2.13 `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 gazebo::event::Events::step.

8.2.2.14 `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 gazebo::event::Events::stop.

8.2.2.15 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldCreated (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the the world created signal.

Parameters

<code>in</code>	<code>_subscriber</code>	the subscriber to this event
-----------------	--------------------------	------------------------------

Returns

a connection

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

8.2.2.16 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateEnd (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the the world update end signal.

Parameters

<code>in</code>	<code>_subscriber</code>	the subscriber to this event
-----------------	--------------------------	------------------------------

Returns

a connection

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

8.2.2.17 `template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateStart (T _subscriber)`
`[inline],[static]`

Connect a boost::slot the the world update start signal.

Parameters

<code>in</code>	<code>_subscriber</code>	the subscriber to this event
-----------------	--------------------------	------------------------------

Returns

a connection

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

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

Disconnect a callback to this event.

Removes a connection.

Parameters

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

Implements **gazebo::event::Event** (p. 351).

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

Referenced by gazebo::event::Events::DisconnectAddEntity(), gazebo::physics::Collision::DisconnectContact(), gazebo::event::Events::DisconnectCreateEntity(), gazebo::rendering::Events::DisconnectCreateScene(), gazebo::event::Events::DisconnectDeleteEntity(), gazebo::event::Events::DisconnectDiagTimerStart(), gazebo::event::Events::DisconnectDiagTimerStop(), gazebo::physics::Link::DisconnectEnabled(), gazebo::physics::Joint::DisconnectJointUpdate(), gazebo::rendering::DepthCamera::DisconnectNewDepthFrame(), gazebo::rendering::Camera::DisconnectNewImageFrame(), gazebo::rendering::GpuLaser::DisconnectNewLaserFrame(), gazebo::physics::MultiRayShape::DisconnectNewLaserScans(), gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud(), gazebo::event::Events::DisconnectPause(), gazebo::event::Events::DisconnectPostRender(), gazebo::event::Events::DisconnectPreRender(), gazebo::rendering::Events::DisconnectRemoveScene(), gazebo::event::Events::DisconnectRender(), gazebo::event::Events::DisconnectSetSelectedEntity(), gazebo::transport::Connection::DisconnectShutdown(), gazebo::event::Events::DisconnectStep(), gazebo::event::Events::DisconnectStop(), gazebo::event::Events::DisconnectWorldCreated(), gazebo::event::Events::DisconnectWorldUpdateEnd(), and gazebo::event::Events::DisconnectWorldUpdateStart().

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

Disconnect a callback to this event.

Removes a connection.

Parameters

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

Implements **gazebo::event::Event** (p. 351).

8.2.2.20 `static void gazebo::event::Events::DisconnectAddEntity (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the add entity signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

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

8.2.2.21 `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 gazebo::event::Events::entityCreated.

8.2.2.22 `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 gazebo::event::Events::deleteEntity, and gazebo::event::EventT< T >::Disconnect().

8.2.2.23 `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 gazebo::event::Events::diagTimerStart, and gazebo::event::EventT< T >::Disconnect().

8.2.2.24 `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 gazebo::event::Events::diagTimerStop, and gazebo::event::EventT< T >::Disconnect().

8.2.2.25 `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 gazebo::event::Events::pause.

8.2.2.26 `static void gazebo::event::Events::DisconnectPostRender (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the post render update signal.

Parameters

in	<code>_subscriber</code>	the subscriber to this event
----	--------------------------	------------------------------

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

8.2.2.27 `static void gazebo::event::Events::DisconnectPreRender (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a render start signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

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

8.2.2.28 `static void gazebo::event::Events::DisconnectRender (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the render update signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

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

8.2.2.29 `static void gazebo::event::Events::DisconnectSetSelectedEntity (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the set selected entity.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

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

8.2.2.30 `static void gazebo::event::Events::DisconnectStep (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the step signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

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

8.2.2.31 `static void gazebo::event::Events::DisconnectStop (ConnectionPtr _subscriber) [inline],[static]`

Disconnect a boost::slot the the stop signal.

Parameters

<code>in</code>	<code><i>_subscriber</i></code>	the subscriber to this event
-----------------	---------------------------------	------------------------------

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

8.2.2.32 `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 gazebo::event::Events::worldCreated.

8.2.2.33 `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 gazebo::event::Events::worldUpdateEnd.

8.2.2.34 `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 gazebo::event::Events::worldUpdateStart.

8.2.3 Variable Documentation

8.2.3.1 `EventT<void (std::string)> gazebo::event::Events::addEntity [static]`

An entity has been added.

Referenced by gazebo::event::Events::ConnectAddEntity(), and gazebo::event::Events::DisconnectAddEntity().

8.2.3.2 `EventT<void (std::string)> gazebo::event::Events::deleteEntity [static]`

An entity has been deleted.

Referenced by gazebo::event::Events::ConnectDeleteEntity(), and gazebo::event::Events::DisconnectDeleteEntity().

8.2.3.3 `EventT<void (std::string)> gazebo::event::Events::diagTimerStart [static]`

Diagnostic timer start.

Referenced by gazebo::event::Events::ConnectDiagTimerStart(), and gazebo::event::Events::DisconnectDiagTimerStart().

8.2.3.4 `EventT<void (std::string)> gazebo::event::Events::diagTimerStop` [static]

Diagnostic timer stop.

Referenced by `gazebo::event::Events::ConnectDiagTimerStop()`, and `gazebo::event::Events::DisconnectDiagTimerStop()`.

8.2.3.5 `EventT<void (std::string)> gazebo::event::Events::entityCreated` [static]

An entity has been created.

Referenced by `gazebo::event::Events::ConnectCreateEntity()`, and `gazebo::event::Events::DisconnectCreateEntity()`.

8.2.3.6 `EventT<void (bool)> gazebo::event::Events::pause` [static]

Pause signal.

Referenced by `gazebo::event::Events::ConnectPause()`, and `gazebo::event::Events::DisconnectPause()`.

8.2.3.7 `EventT<void ()> gazebo::event::Events::postRender` [static]

Post-Render.

Referenced by `gazebo::event::Events::ConnectPostRender()`, and `gazebo::event::Events::DisconnectPostRender()`.

8.2.3.8 `EventT<void ()> gazebo::event::Events::preRender` [static]

Pre-render.

Referenced by `gazebo::event::Events::ConnectPreRender()`, and `gazebo::event::Events::DisconnectPreRender()`.

8.2.3.9 `EventT<void ()> gazebo::event::Events::render` [static]

Render.

Referenced by `gazebo::event::Events::ConnectRender()`, and `gazebo::event::Events::DisconnectRender()`.

8.2.3.10 `EventT<void (std::string, std::string)> gazebo::event::Events::setSelectedEntity` [static]

An entity has been selected.

Referenced by `gazebo::event::Events::ConnectSetSelectedEntity()`, and `gazebo::event::Events::DisconnectSetSelectedEntity()`.

8.2.3.11 `EventT<void ()> gazebo::event::Events::step` [static]

Step the simulation once signal.

Referenced by `gazebo::event::Events::ConnectStep()`, and `gazebo::event::Events::DisconnectStep()`.

8.2.3.12 `EventT<void ()> gazebo::event::Events::stop` `[static]`

Simulation stop signal.

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

8.2.3.13 `EventT<void (std::string)> gazebo::event::Events::worldCreated` `[static]`

A world has been created.

Referenced by `gazebo::event::Events::ConnectWorldCreated()`, and `gazebo::event::Events::DisconnectWorldCreated()`.

8.2.3.14 `EventT<void ()> gazebo::event::Events::worldUpdateEnd` `[static]`

World update has ended.

Referenced by `gazebo::event::Events::ConnectWorldUpdateEnd()`, and `gazebo::event::Events::DisconnectWorldUpdateEnd()`.

8.2.3.15 `EventT<void ()> gazebo::event::Events::worldUpdateStart` `[static]`

World update has started.

Referenced by `gazebo::event::Events::ConnectWorldUpdateStart()`, and `gazebo::event::Events::DisconnectWorldUpdateStart()`.

8.3 Math

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

Files

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

Namespaces

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

Classes

- class **gazebo::math::Angle**
An angle and related functions.
- class **gazebo::math::Box**
Mathematical representation of a box and related functions.
- class **gazebo::math::Matrix3**
A 3x3 matrix class.
- class **gazebo::math::Matrix4**
A 3x3 matrix class.
- class **gazebo::math::Plane**
A plane and related functions.
- class **gazebo::math::Pose**
Encapsulates a position and rotation in three space.
- class **gazebo::math::Quaternion**
A quaternion class.
- class **gazebo::math::Rand**
Random number generator class.
- class **gazebo::math::RotationSpline**
***Spline** (p. 808) 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. 902) 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()
Not a number.
- static const double **gazebo::math::NaN_I** = std::numeric_limits<int>::quiet_NaN()
TODO Nate: type int has no quiet_NaN ... what does this 0 mean?

8.3.1 Detailed Description

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

8.3.2 Function Documentation

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

simple clamping function

Parameters

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

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

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

check if two values are equal, within a tolerance

Parameters

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

Referenced by `gazebo::math::Quaternion::Correct()`, and `gazebo::math::Quaternion::GetInverse()`.

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

check if a float is NaN

Parameters

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

Returns

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

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

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

check if a double is NaN

Parameters

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

Returns

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

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

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

is this a power of 2?

Parameters

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

Returns

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

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

get the maximum value of vector of values

Parameters

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

Returns

maximum

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

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

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

get mean of vector of values

Parameters

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

Returns

the mean

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

get the minimum value of vector of values

Parameters

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

Returns

minimum

References gazebo::math::max().

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

8.3.2.9 double gazebo::math::parseFloat (const std::string & *_input*) [inline]

parse string into float

Parameters

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

Returns

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

References gazebo::math::NAN_D.

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

parse string into an integer

Parameters

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

Returns

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

References gazebo::math::NAN_I.

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

get value at a specified precision

Parameters

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

Returns

the value for the specified precision

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

get variance of vector of values

Parameters

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

Returns

the squared deviation

8.3.3 Variable Documentation

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

Not a number.

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

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

TODO Nate: type int has no quiet_NaN ... what does this 0 mean?

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

8.4 Messages

All messages and helper functions.

Namespaces

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

Classes

- class **google::protobuf::compiler::cpp::GazeboGenerator**
*Google protobuf message generator for **gazebo::msgs** (p. 101).*

Functions

- `msgs::Vector3d gazebo::msgs::Convert` (const `math::Vector3` &_v)
*Convert a **math::Vector3** (p. 902) to a `msgs::Vector3d`.*
- `msgs::Quaternion gazebo::msgs::Convert` (const `math::Quaternion` &_q)
*Convert a **math::Quaternion** (p. 697) to a `msgs::Quaternion`.*
- `msgs::Pose gazebo::msgs::Convert` (const `math::Pose` &_p)
*Convert a **math::Pose** (p. 677) to a `msgs::Pose`.*
- `msgs::Color gazebo::msgs::Convert` (const `common::Color` &_c)
*Convert a **common::Color** (p. 274) to a `msgs::Color`.*
- `msgs::Time gazebo::msgs::Convert` (const `common::Time` &_t)
*Convert a **common::Time** (p. 840) to a `msgs::Time`.*
- `msgs::PlaneGeom gazebo::msgs::Convert` (const `math::Plane` &_p)
*Convert a **math::Plane** (p. 668) 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. 697).*
- `math::Pose gazebo::msgs::Convert` (const `msgs::Pose` &_p)
*Convert a `msgs::Pose` to a **math::Pose** (p. 677).*
- `common::Color gazebo::msgs::Convert` (const `msgs::Color` &_c)
*Convert a `msgs::Color` to a **common::Color** (p. 274).*
- `common::Time gazebo::msgs::Convert` (const `msgs::Time` &_t)
*Convert a `msgs::Time` to a **common::Time** (p. 840).*
- `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. 407).*
- `void gazebo::msgs::Set (msgs::Image *_msg, const common::Image &_i)`
*Set a msgs::Image from a **common::Image** (p. 407).*
- `void gazebo::msgs::Set (msgs::Vector3d *_pt, const math::Vector3 &_v)`
*Set a msgs::Vector3d from a **math::Vector3** (p. 902).*
- `void gazebo::msgs::Set (msgs::Vector2d *_pt, const math::Vector2d &_v)`
*Set a msgs::Vector2d from a **math::Vector3** (p. 902).*
- `void gazebo::msgs::Set (msgs::Quaternion *_q, const math::Quaternion &_v)`
*Set a msgs::Quaternion from a **math::Quaternion** (p. 697).*
- `void gazebo::msgs::Set (msgs::Pose *_p, const math::Pose &_v)`
*Set a msgs::Pose from a **math::Pose** (p. 677).*
- `void gazebo::msgs::Set (msgs::Color *_c, const common::Color &_v)`
*Set a msgs::Color from a **common::Color** (p. 274).*
- `void gazebo::msgs::Set (msgs::Time *_t, const common::Time &_v)`
*Set a msgs::Time from a **common::Time** (p. 840).*
- `void gazebo::msgs::Set (msgs::PlaneGeom *_p, const math::Plane &_v)`
*Set a msgs::Plane from a **math::Plane** (p. 668).*
- `void gazebo::msgs::Stamp (msgs::Header *_header)`
Time stamp a header.
- `void gazebo::msgs::Stamp (msgs::Time *_time)`
Set the time in a time message.
- `msgs::TrackVisual gazebo::msgs::TrackVisualFromSDF (sdf::ElementPtr _sdf)`
Create a msgs::TrackVisual from a track visual SDF element.
- `msgs::Visual gazebo::msgs::VisualFromSDF (sdf::ElementPtr _sdf)`
Create a msgs::Visual from a visual SDF element.

8.4.1 Detailed Description

All messages and helper functions.

8.4.2 Function Documentation

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

Convert a **math::Vector3** (p. 902) to a `msgs::Vector3d`.

Parameters

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

Returns

A `msgs::Vector3d` object

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

Convert a **`math::Quaternion`** (p. 697) to a `msgs::Quaternion`.

Parameters

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

Returns

A `msgs::Quaternion` object

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

Convert a **`math::Pose`** (p. 677) to a `msgs::Pose`.

Parameters

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

Returns

A `msgs::Pose` object

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

Convert a **`common::Color`** (p. 274) to a `msgs::Color`.

Parameters

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

Returns

A `msgs::Color` object

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

Convert a **`common::Time`** (p. 840) to a `msgs::Time`.

Parameters

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

Returns

A `msgs::Time` object

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

Convert a **`math::Plane`** (p. 668) to a `msgs::PlaneGeom`.

Parameters

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

Returns

A `msgs::PlaneGeom` object

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

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

Parameters

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

Returns

A **`math::Vector3`** (p. 902) object

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

Convert a `msgs::Quaternion` to a **`math::Quaternion`** (p. 697).

Parameters

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

Returns

A **`math::Quaternion`** (p. 697) object

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

Convert a `msgs::Pose` to a **`math::Pose`** (p. 677).

Parameters

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

Returns

A **math::Pose** (p. 677) object

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

Convert a `msgs::Color` to a **common::Color** (p. 274).

Parameters

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

Returns

A **common::Color** (p. 274) object

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

Convert a `msgs::Time` to a **common::Time** (p. 840).

Parameters

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

Returns

A **common::Time** (p. 840) object

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

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

Parameters

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

Returns

A `common::Plane` object

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

Create a request message.

Parameters

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

Returns

A Request message

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

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

Parameters

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

Returns

The new `msgs::Fog` object

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

Get the header from a protobuf message.

Parameters

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

Returns

A pointer to the message's header

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

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

Parameters

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

Returns

The new `msgs::GUI` object

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

Initialize a message.

Parameters

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

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

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

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

Parameters

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

Returns

The new msgs::Light object

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

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

Parameters

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

Returns

The new msgs::Scene object

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

Convert a msgs::Image to a **common::Image** (p. 407).

Parameters

<code>_img</code>	The common::Image (p. 407) container
<code>_msg</code>	The Image message to convert

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

Set a msgs::Image from a **common::Image** (p. 407).

Parameters

<code>_msg</code>	A msgs::Image pointer
<code>_i</code>	A common::Image (p. 407) reference

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

Set a msgs::Vector3d from a **math::Vector3** (p. 902).

Parameters

<code>_pt</code>	A <code>msgs::Vector3d</code> pointer
<code>_v</code>	A <code>math::Vector3</code> (p. 902) reference

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

Set a `msgs::Vector2d` from a `math::Vector3` (p. 902).

Parameters

<code>_pt</code>	A <code>msgs::Vector2d</code> pointer
<code>_v</code>	A <code>math::Vector2d</code> (p. 885) reference

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

Set a `msgs::Quaternion` from a `math::Quaternion` (p. 697).

Parameters

<code>_q</code>	A <code>msgs::Quaternion</code> pointer
<code>_v</code>	A <code>math::Quaternion</code> (p. 697) reference

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

Set a `msgs::Pose` from a `math::Pose` (p. 677).

Parameters

<code>_p</code>	A <code>msgs::Pose</code> pointer
<code>_v</code>	A <code>math::Pose</code> (p. 677) reference

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

Set a `msgs::Color` from a `common::Color` (p. 274).

Parameters

<code>_p</code>	A <code>msgs::Color</code> pointer
<code>_v</code>	A <code>common::Color</code> (p. 274) reference

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

Set a `msgs::Time` from a `common::Time` (p. 840).

Parameters

<code>_p</code>	A <code>msgs::Time</code> pointer
<code>_v</code>	A <code>common::Time</code> (p. 840) reference

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

Set a `msgs::Plane` from a `math::Plane` (p. 668).

Parameters

<code><i>_p</i></code>	A <code>msgs::Plane</code> pointer
<code><i>_v</i></code>	A <code>math::Plane</code> (p. 668) reference

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

Time stamp a header.

Parameters

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

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

Set the time in a time message.

Parameters

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

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

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

Parameters

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

Returns

The new `msgs::TrackVisual` object

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

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

Parameters

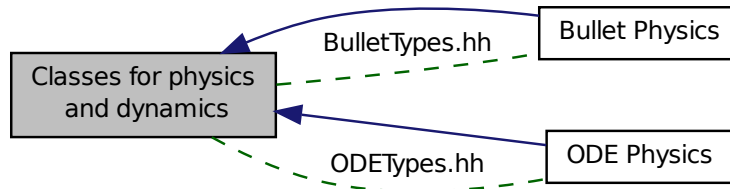
<code><i>_sdf</i></code>	The sdf element
--------------------------	-----------------

Returns

The new `msgs::Visual` object

8.5 Classes for physics and dynamics

Collaboration diagram for Classes for physics and dynamics:



Modules

- **Bullet Physics**
bullet physics engine wrapper
- **ODE Physics**
ODE physics engine wrapper.

Files

- file **BulletTypes.hh**
Bullet wrapper forward declarations and typedefs.
- file **ODETypes.hh**
ODE wrapper forward declarations and typedefs.
- file **PhysicsTypes.hh**
default namespace for gazebo

Namespaces

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

Classes

- class **gazebo::physics::Actor**
Actor (p. 121) class enables GPU based mesh model / skeleton scriptable animation.
- class **gazebo::physics::BallJoint< T >**
Base (p. 145) class for a ball joint.
- class **gazebo::physics::Base**
Base (p. 145) class for most physics classes.
- class **gazebo::physics::BoxShape**

- Box geometry primitive.*

 - class **gazebo::physics::Collision**
 - Base** (p. 145) class for all collision entities.
 - class **gazebo::physics::CollisionState**
 - Store state information of a **physics::Collision** (p. 262) object.
 - class **gazebo::physics::Contact**
 - A contact between two collisions.
 - class **gazebo::physics::CylinderShape**
 - Cylinder collision.
 - class **gazebo::physics::Entity**
 - Base** (p. 145) class for all physics objects in Gazebo.
 - class **gazebo::physics::Gripper**
 - A gripper abstraction.
 - class **gazebo::physics::HeightmapShape**
 - HeightmapShape** (p. 399) 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. 145) class for all joints.
 - class **gazebo::physics::JointController**
 - A class for manipulating **physics::Joint** (p. 423).
 - class **gazebo::physics::JointFeedback**
 - Feedback information from a joint.
 - class **gazebo::physics::JointState**
 - keeps track of state of a **physics::Joint** (p. 423)
 - class **gazebo::physics::Link**
 - Link** (p. 454) 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. 454) object.
 - class **gazebo::physics::Logger**
 - Handles logging of data to disk.
 - class **gazebo::physics::Logplay**
 - Open and playback log files that were recorded using **Logger** (p. 477).
 - class **gazebo::physics::MapShape**
 - Creates box extrusions based on an image.
 - 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. 521) 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. 145) class for a physics engine.
- class **gazebo::physics::PhysicsFactory**
 - The physics factory instantiates different physics engines.*
- class **gazebo::physics::PlaneShape**
 - Collision* (p. 262) for an infinite plane.
- class **gazebo::physics::RayShape**
 - Base* (p. 145) class for Ray collision geometry.
- class **gazebo::physics::Road**
 - for building a Road* (p. 736) from SDF
- class **gazebo::physics::ScrewJoint**< T >
 - A screw joint.*
- class **gazebo::physics::Shape**
 - Base* (p. 145) class for all shapes.
- class **gazebo::physics::SliderJoint**< T >
 - A slider joint.*
- class **gazebo::physics::SphereShape**
 - Sphere collision.*
- class **gazebo::physics::State**
 - State* (p. 813) of an entity.
- class **gazebo::physics::SurfaceParams**
 - SurfaceParams* (p. 830) 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. 954) 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. 90).*
- 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. 838)'s and call gazebo::transport::init (p. 90).
- void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
Load world from sdf::Element (p. 332) pointer.
- void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)
load multiple worlds from single sdf::Element (p. 332) pointer
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)
Pause world by calling World::SetPaused (p. 964).
- 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. 964) 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. 965) 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.

8.5.1 Detailed Description

8.5.2 Macro Definition Documentation

8.5.2.1 #define GZ_REGISTER_PHYSICS_ENGINE(name, classname)

Value:

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

Static physics registration macro.

Use this macro to register physics engine with the server.

Parameters

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.

8.5.3 Typedef Documentation

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

8.5.4 Function Documentation

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

Create a world given a name.

Parameters

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

Returns

Pointer to the new world.

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

Finalize transport by calling `gazebo::transport::fini` (p. 90).

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

Returns a pointer to a world by name.

Parameters

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

Returns

Pointer to the world.

8.5.4.4 `void gazebo::physics::init_world (WorldPtr _world)`

Init world given a pointer to it.

Parameters

in	<i>_world</i>	World (p. 954) to initialize.
----	---------------	--------------------------------------

8.5.4.5 void gazebo::physics::init_worlds ()

initialize multiple worlds stored in static variable gazebo::g_worlds

8.5.4.6 bool gazebo::physics::load ()

Setup **gazebo::SystemPlugin** (p. 838)'s and call **gazebo::transport::init** (p. 90).

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

Load world from **sdf::Element** (p. 332) pointer.

Parameters

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

8.5.4.8 void gazebo::physics::load_worlds (sdf::ElementPtr _sdf)

load multiple worlds from single **sdf::Element** (p. 332) pointer

Parameters

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

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

Pause world by calling **World::SetPaused** (p. 964).

Parameters

in	<i>_world</i>	World (p. 954) to pause or unpause.
in	<i>_pause</i>	True to pause, False to unpause.

8.5.4.10 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.
----	---------------	----------------------------------

8.5.4.11 void gazebo::physics::remove_worlds ()

remove multiple worlds stored in static variable gazebo::g_worlds

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

Run world by calling **World::Run()** (p. 964) given a pointer to it.

Parameters

in	<i>_world</i>	World (p. 954) to run.
----	---------------	-------------------------------

8.5.4.13 void gazebo::physics::run_worlds ()

run multiple worlds stored in static variable gazebo::g_worlds

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

Stop world by calling **World::Stop()** (p. 965) given a pointer to it.

Parameters

in	<i>_world</i>	World (p. 954) to stop.
----	---------------	--------------------------------

8.5.4.15 void gazebo::physics::stop_worlds ()

stop multiple worlds stored in static variable gazebo::g_worlds

8.5.5 Variable Documentation

8.5.5.1 std::string gazebo::physics::EntityTypename[] [static]

Initial value:

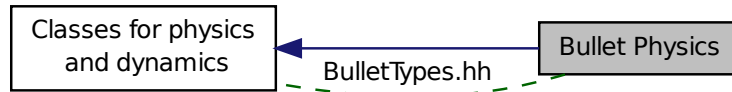
```
= {
    "common",
    "entity",
    "model",
    "actor",
    "link",
    "collision",
    "light",
    "visual",
    "joint",
    "ball",
    "hinge2",
    "hinge",
    "slider",
    "universal",
    "shape",
    "box",
    "cylinder",
    "heightmap",
    "map",
    "multiray",
    "ray",
    "plane",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

String names for the different entity types.

8.6 Bullet Physics

bullet physics engine wrapper

Collaboration diagram for Bullet Physics:



Files

- file **BulletTypes.hh**
Bullet wrapper forward declarations and typedefs.

Classes

- class **gazebo::physics::BulletBallJoint**
***BulletBallJoint** (p. 166) class models a ball joint in Bullet.*
- class **gazebo::physics::BulletBoxShape**
Bullet box collision.
- class **gazebo::physics::BulletCollision**
Bullet collisions.
- class **gazebo::physics::BulletCylinderShape**
Cylinder collision.
- class **gazebo::physics::BulletHeightmapShape**
Height map collision.
- class **gazebo::physics::BulletHinge2Joint**
A two axis hinge joint.
- class **gazebo::physics::BulletHingeJoint**
A single axis hinge joint.
- class **gazebo::physics::BulletJoint**
***Base** (p. 145) class for all joints.*
- class **gazebo::physics::BulletLink**
*Bullet **Link** (p. 454) class.*
- class **gazebo::physics::BulletMotionState**
*Bullet **btMotionState** encapsulation.*
- class **gazebo::physics::BulletMultiRayShape**
*Bullet specific version of **MultiRayShape** (p. 549).*
- class **gazebo::physics::BulletPhysics**
Bullet physics engine.
- class **gazebo::physics::BulletPlaneShape**

Bullet collision for an infinite plane.

- class **gazebo::physics::BulletRaySensor**
An Bullet Ray sensor.
- class **gazebo::physics::BulletRayShape**
Ray shape for bullet.
- class **gazebo::physics::BulletScrewJoint**
A screw joint.
- class **gazebo::physics::BulletSliderJoint**
A slider joint.
- class **gazebo::physics::BulletSphereShape**
Bullet sphere collision.
- class **gazebo::physics::BulletTrimeshShape**
Triangle mesh collision.
- class **gazebo::physics::BulletUniversalJoint**
A bullet universal joint class.

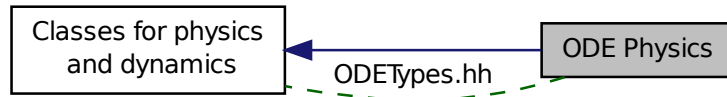
8.6.1 Detailed Description

bullet physics engine wrapper

8.7 ODE Physics

ODE physics engine wrapper.

Collaboration diagram for ODE Physics:



Files

- file **ODETypes.hh**
ODE wrapper forward declarations and typedefs.

Classes

- class **gazebo::physics::ContactFeedback**
data structure for contact feedbacks
- class **gazebo::physics::ODEBallJoint**
*An **ODEBallJoint** (p. 573).*
- class **gazebo::physics::ODEBoxShape**
ODE Box shape.
- class **gazebo::physics::ODECollision**
***Base** (p. 145) class for all ODE collisions.*
- class **gazebo::physics::ODECylinderShape**
ODE cylinder shape.
- class **gazebo::physics::ODEHeightmapShape**
ODE Height map collision.
- class **gazebo::physics::ODEHinge2Joint**
A two axis hinge joint.
- class **gazebo::physics::ODEHingeJoint**
A single axis hinge joint.
- class **gazebo::physics::ODEJoint**
ODE joint interface.
- class **gazebo::physics::ODELink**
*ODE **Link** (p. 454) class.*
- class **gazebo::physics::ODEMultiRayShape**
*ODE specific version of **MultiRayShape** (p. 549).*
- class **gazebo::physics::ODEPhysics**
ODE physics engine.

- class **gazebo::physics::ODEPlaneShape**
An ODE Plane shape.
- class **gazebo::physics::ODERayShape**
Ray collision.
- class **gazebo::physics::ODEScrewJoint**
A screw joint.
- class **gazebo::physics::ODESliderJoint**
A slider joint.
- class **gazebo::physics::ODESphereShape**
A ODE sphere shape.
- class **gazebo::physics::ODESurfaceParams**
Surface params.
- class **gazebo::physics::ODETrimeshShape**
Triangle mesh collision.
- class **gazebo::physics::ODEUniversalJoint**
A universal joint.

Typedefs

- typedef ODEPhysics * **gazebo::physics::ODEPhysicsPtr**
- typedef ODESurfaceParams * **gazebo::physics::ODESurfaceParamsPtr**

8.7.1 Detailed Description

ODE physics engine wrapper.

8.7.2 Typedef Documentation

8.7.2.1 typedef ODEPhysics * **gazebo::physics::ODEPhysicsPtr**

8.7.2.2 typedef ODESurfaceParams* **gazebo::physics::ODESurfaceParamsPtr**

8.8 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. 305) Conversions.hh (p. 1018) rendering/Conversions.hh (p. 1018).
- 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. 118)'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.

Functions

- rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations)
 - create **rendering::Scene** (p. 746) 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. 746) 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. 746) by name*

8.8.1 Detailed Description

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

8.8.2 Function Documentation

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

create **rendering::Scene** (p. 746) by name.

Parameters

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

8.8.2.2 `bool gazebo::rendering::fini ()`

teardown rendering engine.

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

get pointer to **rendering::Scene** (p. 746) by name.

Parameters

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

8.8.2.4 `bool gazebo::rendering::init ()`

init rendering engine.

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

load rendering engine.

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

remove a **rendering::Scene** (p. 746) by name

Parameters

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

8.9 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. 762) Element (p. 332) class.
- class **urdf2gazebo::GazeboExtension**
- class **sdf::SDF**
Base SDF (p. 762) class.
- class **urdf2gazebo::URDF2Gazebo**

Typedefs

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

8.9.1 Detailed Description

8.9.2 Typedef Documentation

8.9.2.1 typedef urdf::Collision* urdf2gazebo::CollisionPtr

8.9.2.2 typedef urdf::Visual* urdf2gazebo::VisualPtr

8.10 Sensors

A set of sensor classes, functions, and definitions.

Files

- file **SensorTypes.hh**

Forward declarations and typedefs for sensors.

Namespaces

- namespace **gazebo::sensors**

Sensors namespace.

Classes

- class **gazebo::sensors::CameraSensor**
- class **gazebo::sensors::ContactSensor**
- class **gazebo::sensors::DepthCameraSensor**
- class **gazebo::sensors::GpuRaySensor**
- class **gazebo::sensors::ImuSensor**
An IMU sensor.
- class **gazebo::sensors::RaySensor**
- class **gazebo::sensors::RFIDSensor**
Sensor (p. 765) class for RFID type of sensor.
- class **gazebo::sensors::RFIDTag**
RFIDTag (p. 729) to interact with RFIDTagSensors Nate check.
- class **gazebo::sensors::RFIDTagManager**
Nate fill in
- class **gazebo::sensors::Sensor**
- 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 ()`
- `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.

8.10.1 Detailed Description

A set of sensor classes, functions, and definitions. Load the sensor library.

class **Sensor** (p. 765) **Sensor.hh** (p. 1142) sensors/sensors.hh

Sensor (p. 765) with one or more rays.

This class inherits from **Sensor** (p. 765), but looks like it specifically doesn't override any methods, is this intentional? i.e.

GPU based laser sensor.

Contact sensor.

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

This sensor detects and reports contacts between objects

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).

LoadChild instead of Load, InitChild instead of Init

Base class for sensors

Returns

True if successfully loaded, false if not Nate check

8.10.2 Macro Definition Documentation

8.10.2.1 `#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.

8.10.3 Function Documentation

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

Create a sensor using SDF.

Parameters

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

Returns

The name of the new sensor

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

shutdown the sensor generation loop.

Returns

True if successfully finalized, false if not

8.10.3.3 `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

in	<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.

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

initialize the sensor generation loop.

Returns

True if successfully initialized, false if not

8.10.3.5 `bool gazebo::sensors::load ()`8.10.3.6 `void gazebo::sensors::remove_sensor (const std::string & _sensorName)`

Remove a sensor by name.

Parameters

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

8.10.3.7 `bool gazebo::sensors::remove_sensors ()`

Remove all sensors.

Returns

True if all successfully removed, false if not

8.10.3.8 `void gazebo::sensors::run ()`

Run sensor generation continuously. This is a blocking call.

8.10.3.9 `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.
-----------------------	--

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

Stop the sensor generation loop.

8.11 Transport

Handles transportation of messages.

Files

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

Namespaces

- namespace **gazebo::transport**
Transport namespace.

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**
*TCP/IP **Connection** (p. 288).*
- class **gazebo::transport::ConnectionManager**
Manager of connections.
- class **gazebo::transport::DebugCallbackHelper**
- class **gazebo::transport::IOManager**
Managers 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**
Reads data from a remote advertiser, and passes the data along to local subscribers.
- 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**
Handles sending data over the wire to remote subscribers.
- class **gazebo::transport::TopicManager**
Manages topics and their subscriptions.

Typedefs

- typedef CallbackHelper * **gazebo::transport::CallbackHelperPtr**
boost shared pointer to **transport::CallbackHelper** (p. 230)

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** ()
Return true if the transport system is stopped.
- void **gazebo::transport::pause_incoming** (bool _pause)
Set to true to pause 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.

8.11.1 Detailed Description

Handles transportation of messages.

8.11.2 Typedef Documentation

8.11.2.1 typedef CallbackHelper* gazebo::transport::CallbackHelperPtr

boost shared pointer to **transport::CallbackHelper** (p. 230)

8.11.3 Function Documentation

8.11.3.1 void gazebo::transport::clear_buffers ()

clear any remaining communication buffers

8.11.3.2 void gazebo::transport::fini ()

Cleanup the transport component.

8.11.3.3 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

<i>master_host</i>	The hostname of the master is set to this param
<i>master_port</i>	The port of the master is set to this param

Returns

False if the GAZEBO_MASTER_URI was not found

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

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

8.11.3.5 bool gazebo::transport::init (const std::string & *master_host* = " ", unsigned int *master_port* = 0)

Initialize the transport system.

Parameters

<i>master_host</i>	The hostname or IP of the master. Leave empty to use pull address from the GAZEBO_MASTER_URI env var.
<i>master_port</i>	The port of the master. Leave empty to use pull address from the GAZEBO_MASTER_URI env var.

8.11.3.6 bool gazebo::transport::is_stopped ()

Return true if the transport system is stopped.

8.11.3.7 void gazebo::transport::pause_incoming (bool *_pause*)

Set to true to pause incoming messages.

They are still queued for later delivery

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

Send a request, and receive a response.

8.11.3.9 void gazebo::transport::run ()

Run the transport component.

This starts message passing. This is a blocking call

8.11.3.10 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. 350) namespace.*
- namespace **math**
Math namespace.
- namespace **msgs**
Messages namespace.
- namespace **physics**
namespace for physics
- namespace **rendering**
Rendering namespace.
- namespace **sensors**
Sensors namespace.
- namespace **transport**
Transport namespace.

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. 521).*

- 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. 954).

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. Nate check this base class and I can propogate changes to subclasses

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, and logging 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 **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**
A transformation node.
- class **NumericAnimation**
A numeric animation.
- class **NumericKeyFrame**
*A keyframe for a **NumericAnimation** (p. 569).*
- class **PID**
*Generic **PID** (p. 664) controller class.*
- class **PoseAnimation**
A pose animation.
- class **PoseKeyFrame**
*A keyframe for a **PoseAnimation** (p. 685).*
- class **Skeleton**
A skeleton.
- class **SkeletonAnimation**
***Skeleton** (p. 784) 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. 840) 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. 834)*
- std::string **find_file** (const std::string &_file, bool _searchLocalPath=true)
*search for file in **common::SystemPaths** (p. 834)*
- std::string **find_file_path** (const std::string &_file)
*search for a file in **common::SystemPaths** (p. 834)*

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. 350) namespace.

Classes

- class **Connection**
A class that encapsulates a connection.
- class **Event**
Base class for all events.
- class **Events**
*An **Event** (p. 350) 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. 350) 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. 808) for rotations.
- class **Spline**
Splines.
- class **Vector2d**
Generic double x, y vector.
- class **Vector2i**
Generic integer x, y vector.
- class **Vector3**
*The **Vector3** (p. 902) 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()
Not a number.
- static const double **NAN_I** = std::numeric_limits<int>::quiet_NaN()
TODO Nate: type int has no quiet_NaN ... what does this 0 mean?

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. 902) to a msgs::Vector3d.*
- msgs::Quaternion **Convert** (const math::Quaternion &_q)
*Convert a **math::Quaternion** (p. 697) to a msgs::Quaternion.*
- msgs::Pose **Convert** (const math::Pose &_p)
*Convert a **math::Pose** (p. 677) to a msgs::Pose.*
- msgs::Color **Convert** (const common::Color &_c)
*Convert a **common::Color** (p. 274) to a msgs::Color.*
- msgs::Time **Convert** (const common::Time &_t)
*Convert a **common::Time** (p. 840) to a msgs::Time.*
- msgs::PlaneGeom **Convert** (const math::Plane &_p)
*Convert a **math::Plane** (p. 668) 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. 697).*
- **math::Pose Convert** (const msgs::Pose &_p)
*Convert a msgs::Pose to a **math::Pose** (p. 677).*
- **common::Color Convert** (const msgs::Color &_c)
*Convert a msgs::Color to a **common::Color** (p. 274).*
- **common::Time Convert** (const msgs::Time &_t)
*Convert a msgs::Time to a **common::Time** (p. 840).*
- **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. 407).*
- void **Set** (msgs::Image *_msg, const **common::Image** &_i)
*Set a msgs::Image from a **common::Image** (p. 407).*
- void **Set** (msgs::Vector3d *_pt, const **math::Vector3** &_v)
*Set a msgs::Vector3d from a **math::Vector3** (p. 902).*
- void **Set** (msgs::Vector2d *_pt, const **math::Vector2d** &_v)
*Set a msgs::Vector2d from a **math::Vector3** (p. 902).*
- void **Set** (msgs::Quaternion *_q, const **math::Quaternion** &_v)
*Set a msgs::Quaternion from a **math::Quaternion** (p. 697).*
- void **Set** (msgs::Pose *_p, const **math::Pose** &_v)
*Set a msgs::Pose from a **math::Pose** (p. 677).*
- void **Set** (msgs::Color *_c, const **common::Color** &_v)
*Set a msgs::Color from a **common::Color** (p. 274).*
- void **Set** (msgs::Time *_t, const **common::Time** &_v)
*Set a msgs::Time from a **common::Time** (p. 840).*
- void **Set** (msgs::PlaneGeom *_p, const **math::Plane** &_v)
*Set a msgs::Plane from a **math::Plane** (p. 668).*
- 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. 121) class enables GPU based mesh model / skeleton scriptable animation.
- class **BallJoint**
Base (p. 145) class for a ball joint.
- class **Base**
Base (p. 145) class for most physics classes.
- class **BoxShape**
Box geometry primitive.
- class **BulletBallJoint**
BulletBallJoint (p. 166) class models a ball joint in Bullet.
- class **BulletBoxShape**
Bullet box collision.
- class **BulletCollision**
Bullet collisions.
- class **BulletCylinderShape**
Cylinder collision.
- class **BulletHeightmapShape**
Height map collision.
- class **BulletHinge2Joint**
A two axis hinge joint.
- class **BulletHingeJoint**
A single axis hinge joint.
- class **BulletJoint**
Base (p. 145) class for all joints.
- class **BulletLink**
Bullet Link (p. 454) class.
- class **BulletMotionState**
Bullet btMotionState encapsulation.
- class **BulletMultiRayShape**
Bullet specific version of MultiRayShape (p. 549).
- class **BulletPhysics**
Bullet physics engine.
- class **BulletPlaneShape**
Bullet collision for an infinite plane.
- class **BulletRaySensor**
An Bullet Ray sensor.

- class **BulletRayShape**
Ray shape for bullet.
- class **BulletScrewJoint**
A screw joint.
- class **BulletSliderJoint**
A slider joint.
- class **BulletSphereShape**
Bullet sphere collision.
- class **BulletTrimeshShape**
Triangle mesh collision.
- class **BulletUniversalJoint**
A bullet universal joint class.
- class **Collision**
Base (p. 145) class for all collision entities.
- class **CollisionState**
Store state information of a `physics::Collision` (p. 262) object.
- class **Contact**
A contact between two collisions.
- class **ContactFeedback**
data structure for contact feedbacks
- class **CylinderShape**
Cylinder collision.
- class **Entity**
Base (p. 145) class for all physics objects in Gazebo.
- class **Gripper**
A gripper abstraction.
- class **HeightmapShape**
HeightmapShape (p. 399) 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. 145) class for all joints.
- class **JointController**
A class for manipulating `physics::Joint` (p. 423).
- class **JointFeedback**
Feedback information from a joint.
- class **JointState**
keeps track of state of a `physics::Joint` (p. 423)
- class **Link**
Link (p. 454) 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. 454) object.

- class **Logger**
Handles logging of data to disk.
- class **Logplay**
*Open and playback log files that were recorded using **Logger** (p. 477).*
- class **MapShape**
Creates box extrusions based on an image.
- class **Model**
A model is a collection of links, joints, and plugins.
- class **ModelState**
*Store state information of a **physics::Model** (p. 521) object.*
- class **MultiRayShape**
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.
- class **ODEBallJoint**
*An **ODEBallJoint** (p. 573).*
- class **ODEBoxShape**
ODE Box shape.
- class **ODECollision**
***Base** (p. 145) class for all ODE collisions.*
- class **ODECylinderShape**
ODE cylinder shape.
- class **ODEHeightmapShape**
ODE Height map collision.
- class **ODEHinge2Joint**
A two axis hinge joint.
- class **ODEHingeJoint**
A single axis hinge joint.
- class **ODEJoint**
ODE joint interface.
- class **ODELink**
*ODE **Link** (p. 454) class.*
- class **ODEMultiRayShape**
*ODE specific version of **MultiRayShape** (p. 549).*
- class **ODEPhysics**
ODE physics engine.
- class **ODEPlaneShape**
An ODE Plane shape.
- class **ODERayShape**
Ray collision.
- class **ODEScrewJoint**
A screw joint.
- class **ODESliderJoint**
A slider joint.
- class **ODESphereShape**
A ODE sphere shape.
- class **ODESurfaceParams**
Surface params.
- class **ODETrimeshShape**

- Triangle mesh collision.*
- class **ODEUniversalJoint**
 - A universal joint.*
- class **PhysicsEngine**
 - Base** (p. 145) *class for a physics engine.*
- class **PhysicsFactory**
 - The physics factory instantiates different physics engines.*
- class **PlaneShape**
 - Collision** (p. 262) *for an infinite plane.*
- class **RayShape**
 - Base** (p. 145) *class for Ray collision geometry.*
- class **Road**
 - for building a **Road** (p. 736) from SDF*
- class **ScrewJoint**
 - A screw joint.*
- class **Shape**
 - Base** (p. 145) *class for all shapes.*
- class **SliderJoint**
 - A slider joint.*
- class **SphereShape**
 - Sphere collision.*
- class **State**
 - State** (p. 813) *of an entity.*
- class **SurfaceParams**
 - SurfaceParams** (p. 830) *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. 954) object.*

Typedefs

- typedef std::vector< **ActorPtr** > **Actor_V**
- typedef **Actor** * **ActorPtr**
- typedef std::vector< **BasePtr** > **Base_V**
- typedef **Base** * **BasePtr**
- typedef **BoxShape** * **BoxShapePtr**
- typedef **BulletCollision** * **BulletCollisionPtr**
- typedef **BulletLink** * **BulletLinkPtr**
- typedef **BulletPhysics** * **BulletPhysicsPtr**
- typedef **BulletRayShape** * **BulletRayShapePtr**
- 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 **ODECollision** * **ODECollisionPtr**
- typedef **ODELink** * **ODELinkPtr**
- typedef **ODEPhysics** * **ODEPhysicsPtr**
- typedef **ODERayShape** * **ODERayShapePtr**
- typedef **ODESurfaceParams** * **ODESurfaceParamsPtr**
- 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. 90).
- **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. 838)'s and call `gazebo::transport::init` (p. 90).
- void **load_world** (**WorldPtr** _world, sdf::ElementPtr _sdf)
Load world from `sdf::Element` (p. 332) pointer.
- void **load_worlds** (sdf::ElementPtr _sdf)
load multiple worlds from single `sdf::Element` (p. 332) pointer
- void **pause_world** (**WorldPtr** _world, bool _pause)
Pause world by calling `World::SetPaused` (p. 964).
- 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. 964) 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. 965) 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 BulletCollision* gazebo::physics::BulletCollisionPtr

9.7.2.7 typedef BulletLink* gazebo::physics::BulletLinkPtr

9.7.2.8 typedef BulletPhysics* gazebo::physics::BulletPhysicsPtr

9.7.2.9 typedef BulletRayShape* gazebo::physics::BulletRayShapePtr

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

9.7.2.11 typedef Collision* gazebo::physics::CollisionPtr

9.7.2.12 typedef Contact* gazebo::physics::ContactPtr

9.7.2.13 typedef CylinderShape* gazebo::physics::CylinderShapePtr

- 9.7.2.14 `typedef Entity* gazebo::physics::EntityPtr`
- 9.7.2.15 `typedef HeightmapShape* gazebo::physics::HeightmapShapePtr`
- 9.7.2.16 `typedef Inertial* gazebo::physics::InertialPtr`
- 9.7.2.17 `typedef std::vector<JointPtr> gazebo::physics::Joint_V`
- 9.7.2.18 `typedef Joint* gazebo::physics::JointPtr`
- 9.7.2.19 `typedef std::vector<LinkPtr> gazebo::physics::Link_V`
- 9.7.2.20 `typedef Link* gazebo::physics::LinkPtr`
- 9.7.2.21 `typedef MeshShape* gazebo::physics::MeshShapePtr`
- 9.7.2.22 `typedef std::vector<ModelPtr> gazebo::physics::Model_V`
- 9.7.2.23 `typedef Model* gazebo::physics::ModelPtr`
- 9.7.2.24 `typedef MultiRayShape* gazebo::physics::MultiRayShapePtr`
- 9.7.2.25 `typedef ODECollision* gazebo::physics::ODECollisionPtr`
- 9.7.2.26 `typedef ODELink* gazebo::physics::ODELinkPtr`
- 9.7.2.27 `typedef ODERayShape* gazebo::physics::ODERayShapePtr`
- 9.7.2.28 `typedef PhysicsEngine* gazebo::physics::PhysicsEnginePtr`
- 9.7.2.29 `typedef RayShape* gazebo::physics::RayShapePtr`
- 9.7.2.30 `typedef Road* gazebo::physics::RoadPtr`
- 9.7.2.31 `typedef Shape* gazebo::physics::ShapePtr`
- 9.7.2.32 `typedef SphereShape* gazebo::physics::SphereShapePtr`
- 9.7.2.33 `typedef SurfaceParams* gazebo::physics::SurfaceParamsPtr`
- 9.7.2.34 `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. 305) **Conversions.hh** (p. 1018) **rendering/Conversions.hh** (p. 1018).
 - 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. 118)'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.

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. 746) by name.*
- bool **fini** ()
teardown rendering engine.
- **rendering::ScenePtr get_scene** (const std::string &_name)
*get pointer to **rendering::Scene** (p. 746) by name.*
- bool **init** ()
init rendering engine.
- bool **load** ()
load rendering engine.
- void **remove_scene** (const std::string &_name)
*remove a **rendering::Scene** (p. 746) 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**
- class **ContactSensor**
- class **DepthCameraSensor**
- class **GpuRaySensor**
- class **ImuSensor**
 - An IMU sensor.*
- class **RaySensor**
- class **RFIDSensor**
 - Sensor (p. 765) class for RFID type of sensor.*
- class **RFIDTag**
 - RFIDTag (p. 729) to interact with RFIDTagSensors Nate check.*
- class **RFIDTagManager**
 - Nate fill in*
- class **Sensor**
- 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** ()
- 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

Transport namespace.

Classes

- class **CallbackHelper**
A helper class to handle callbacks when messages arrive.
- class **CallbackHelperT**
Callback helper Template.
- class **Connection**
TCP/IP Connection (p. 288).
- class **ConnectionManager**
Manager of connections.

- class **DebugCallbackHelper**
- class **IOManager**
Managers 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**
Reads data from a remote advertiser, and passes the data along to local subscribers.
- class **Publisher**
A publisher of messages on a topic.
- class **SubscribeOptions**
Options for a subscription.
- class **Subscriber**
A subscriber to a topic.
- class **SubscriptionTransport**
Handles sending data over the wire to remote subscribers.
- class **TopicManager**
Manages topics and their subscriptions.

Typedefs

- typedef **CallbackHelper * CallbackHelperPtr**
boost shared pointer to `transport::CallbackHelper` (p. 230)
- 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 ()**
Return true if the transport system is stopped.
- void **pause_incoming** (bool _pause)

Set to true to pause 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 Detailed Description

Transport namespace.

9.10.2 Typedef Documentation

9.10.2.1 typedef Connection* gazebo::transport::ConnectionPtr

9.10.2.2 typedef Node* gazebo::transport::NodePtr

9.10.2.3 typedef Publication* gazebo::transport::PublicationPtr

9.10.2.4 typedef PublicationTransport* gazebo::transport::PublicationTransportPtr

9.10.2.5 typedef Publisher* gazebo::transport::PublisherPtr

9.10.2.6 typedef Subscriber* gazebo::transport::SubscriberPtr

9.10.2.7 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. 101).

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. 762) to another.*
- class **Element**
***SDF** (p. 762) **Element** (p. 332) class.*
- class **Param**
A parameter class.
- class **ParamT**
Templatized parameter class.
- class **Plugin**
- class **SDF**
*Base **SDF** (p. 762) 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. 762) 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. 762) values from a file.*
- bool **readString** (const std::string &_xmlString, **SDFPtr** _sdf)
 - Populate the **SDF** (p. 762) 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. 762) 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. 762) values from a file.

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

Populate the **SDF** (p. 762) 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

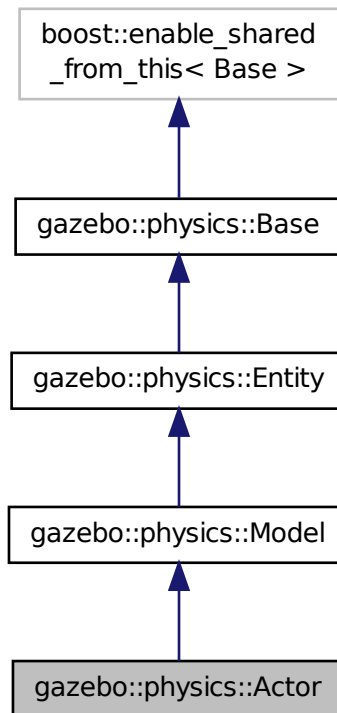
Class Documentation

10.1 gazebo::physics::Actor Class Reference

Actor (p. 121) 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. 145) 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. 121) 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. 525).

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. 528).

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

Initialize the actor.

Reimplemented from **gazebo::physics::Model** (p. 529).

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. 346).

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. 157).

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. 533).

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. 145) 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 <Angle.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. 127) 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. 127) 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. 127) 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. 127) 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. 127) 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. 127) 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. 127) 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. 127) 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. 127) 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. 127) 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. 127) 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

in	<i>_angle</i>	Angle (p. 127) 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

in	<i>_angle</i>	Angle (p. 127) 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

in	<i>_angle</i>	Angle (p. 127) 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

in	<i>_angle</i>	Angle (p. 127) 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:

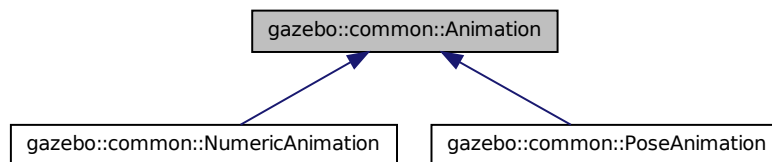
- **Angle.hh**

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 <Animation.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:

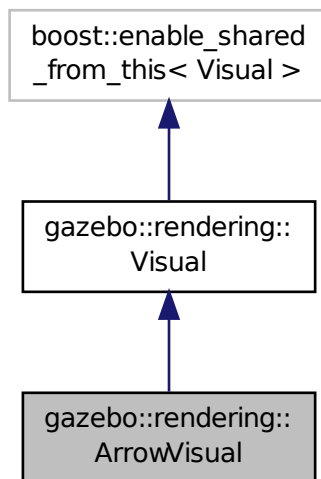
- **Animation.hh**

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

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

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

Destructor.

10.4.3 Member Function Documentation

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

Load the visual with default parameters.

Reimplemented from `gazebo::rendering::Visual` (p. 943).

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:

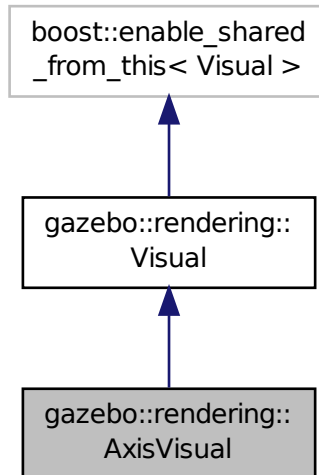
- **ArrowVisual.hh**

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. 140)
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. 943).

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:

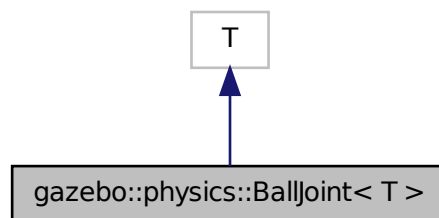
- **AxisVisual.hh**

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

Base (p. 145) 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 **math::Angle** **GetHighStop** (int)
- virtual **math::Angle** **GetLowStop** (int)
- virtual void **SetAxis** (int, const **math::Vector3** &)
- virtual void **SetHighStop** (int, **math::Angle**)
- virtual void **SetLowStop** (int, **math::Angle**)

Protected Member Functions

- void **Load** (**sdf::ElementPtr** _sdf)
*Template to ::Load the **BallJoint** (p. 143).*

10.6.1 Detailed Description

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

Base (p. 145) 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]

Constructor.

Parameters

in	_parent	Pointer to the parent link.
----	----------------	-----------------------------

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

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

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

Template to ::Load the **BallJoint** (p. 143).

Parameters

in	_sdf	SDF to load the joint from.
----	-------------	-----------------------------

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

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

```
10.6.3.6 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:

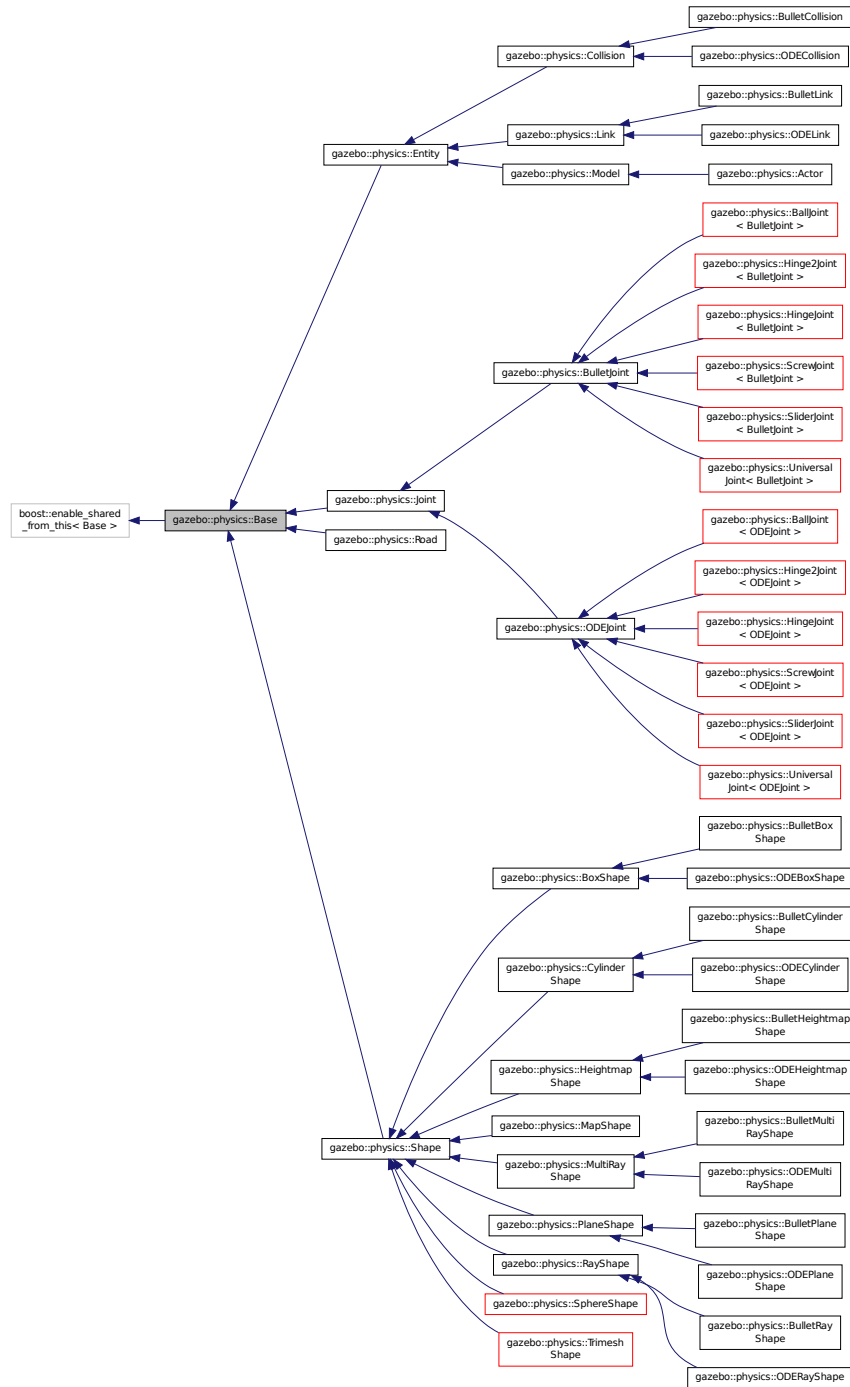
- [BallJoint.hh](#)

10.7 gazebo::physics::Base Class Reference

Base (p. 145) 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. 954) 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. 149)'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. 145) 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. 145) type.
- ENTITY** **Entity** (p. 338) type.
- MODEL** **Model** (p. 521) type.
- LINK** **Link** (p. 454) type.
- COLLISION** **Collision** (p. 262) type.
- ACTOR** **Actor** (p. 121) type.
- LIGHT** **Light** type.
- VISUAL** **Visual** type.
- JOINT** **Joint** (p. 423) type.
- BALL_JOINT** **BallJoint** (p. 143) type.
- HINGE2_JOINT** **Hing2Joint** type.
- HINGE_JOINT** **HingeJoint** (p. 405) type.
- SLIDER_JOINT** **SliderJoint** (p. 804) type.
- SCREW_JOINT** **ScrewJoint** (p. 760) type.
- UNIVERSAL_JOINT** **UniversalJoint** (p. 868) type.
- SHAPE** **Shape** (p. 779) type.
- BOX_SHAPE** **BoxShape** (p. 162) type.
- CYLINDER_SHAPE** **CylinderShape** (p. 307) type.
- HEIGHTMAP_SHAPE** **HeightmapShape** (p. 399) type.
- MAP_SHAPE** **MapShape** (p. 483) type.

MULTIRAY_SHAPE MultiRayShape (p. 549) type.

RAY_SHAPE RayShape (p. 718) type.

PLANE_SHAPE PlaneShape (p. 671) type.

SPHERE_SHAPE SphereShape (p. 806) type.

TRIMESH_SHAPE TrimeshShape (p. 866) type.

10.7.3 Constructor & Destructor Documentation

10.7.3.1 gazebo::physics::Base::Base (BasePtr parent)

Constructor.

Parameters

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

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

Add a type specifier.

Parameters

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

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

Finalize the object.

Reimplemented in **gazebo::physics::Actor** (p. 124), **gazebo::physics::Model** (p. 525), **gazebo::physics::Entity** (p. 342), **gazebo::physics::Link** (p. 462), **gazebo::physics::ODECollision** (p. 580), **gazebo::physics::BulletLink** (p. 195), **gazebo::physics::ODELink** (p. 603), and **gazebo::physics::Collision** (p. 266).

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

Get a child or self by id.

Parameters

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

Returns

A 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

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

Returns

A pointer to the object, NULL if not found

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

Get a child by index.

Parameters

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

Returns

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

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

Get a child by name.

Parameters

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

Returns

A pointer to the object, NULL if not found

10.7.4.8 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. 124), and `gazebo::physics::Model` (p. 528).

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. 954) this object is in.

Returns

The **World** (p. 954) 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

<code>in</code>	<code>_t</code>	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::Joint` (p. 433), `gazebo::physics::RayShape` (p. 721), `gazebo::physics::Actor` (p. 124), `gazebo::physics::Model` (p. 529), `gazebo::physics::Link` (p. 467), `gazebo::physics::MapShape` (p. 486), `gazebo::physics::Collision` (p. 270), `gazebo::physics::HeightmapShape` (p. 403), `gazebo::physics::BulletLink` (p. 196), `gazebo::physics::HingeJoint< ODEJoint >` (p. 407), `gazebo::physics::HingeJoint< BulletJoint >` (p. 407), `gazebo::physics::ODELink` (p. 604), `gazebo::physics::TrimeshShape` (p. 868), `gazebo::physics::BulletHeightmapShape` (p. 178), `gazebo::physics::MultiRayShape` (p. 555), `gazebo::physics::ODETrimeshShape` (p. 634), `gazebo::physics::ODEHeightmapShape` (p. 586), `gazebo::physics::PlaneShape` (p. 673), `gazebo::physics::BulletTrimeshShape` (p. 224), `gazebo::physics::BoxShape` (p. 165), `gazebo::physics::CylinderShape` (p. 310), `gazebo::physics::Shape` (p. 781), `gazebo::physics::Road` (p. 737), and `gazebo::physics::SphereShape` (p. 808).

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. 433), `gazebo::physics::Actor` (p. 124), `gazebo::physics::Entity` (p. 346), `gazebo::physics::Model` (p. 529), `gazebo::physics::Link` (p. 468), `gazebo::physics::BulletCollision` (p. 173), `gazebo::physics::MapShape` (p. 486), `gazebo::physics::Collision` (p. 270), `gazebo::physics::BallJoint< ODEJoint >` (p. 144), `gazebo::physics::BallJoint< BulletJoint >` (p. 144), `gazebo::physics::HeightmapShape` (p. 403), `gazebo::physics::BulletLink` (p. 196), `gazebo::physics::BulletScrewJoint` (p. 216), `gazebo::physics::ScrewJoint< ODEJoint >` (p. 761), `gazebo::physics::ScrewJoint< BulletJoint >` (p. 761), `gazebo::physics::BulletHinge2Joint` (p. 182), `gazebo::physics::BulletHingeJoint` (p. 187), `gazebo::physics::BulletSliderJoint` (p. 220), `gazebo::physics::ODECollision` (p. 581), `gazebo::physics::ODEScrewJoint` (p. 624), `gazebo::physics::ODEHinge2Joint` (p. 589), `gazebo::physics::ODEHingeJoint` (p. 593), `gazebo::physics::SliderJoint< ODEJoint >` (p. 805), `gazebo::physics::SliderJoint< BulletJoint >` (p. 805), `gazebo::physics::ODELink` (p. 605), `gazebo::physics::ODETrimeshShape` (p. 634), `gazebo::physics::Hinge2Joint< ODEJoint >` (p. 405), `gazebo::physics::Hinge2Joint< BulletJoint >` (p. 405), `gazebo::physics::HingeJoint< ODEJoint >` (p. 407), `gazebo::physics::HingeJoint< BulletJoint >` (p. 407), `gazebo::physics::UniversalJoint< ODEJoint >` (p. 870), `gazebo::physics::UniversalJoint< BulletJoint >` (p. 870), `gazebo::physics::BulletJoint` (p. 190), `gazebo::physics::ODEJoint` (p. 598), `gazebo::physics::ODESliderJoint` (p. 628), `gazebo::physics::BulletTrimeshShape` (p. 224), and `gazebo::physics::Road` (p. 737).

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

in	_ent	Base (p. 145) 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

in	_prefix	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

in	_id	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

in	_name	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. 434), **gazebo::physics::Model** (p. 530), **gazebo::physics::Entity** (p. 346), **gazebo::physics::Link** (p. 468), **gazebo::physics::BulletJoint** (p. 191), and **gazebo::physics::ODEJoint** (p. 598).

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

Calls recursive Reset on one of the **Base::EntityType** (p. 149)'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. 347).

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. 471).

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

in	<code>_newWorld</code>	The new World (p. 954) 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. 555), `gazebo::physics::Joint` (p. 437), `gazebo::physics::Actor` (p. 125), `gazebo::physics::RayShape` (p. 722), `gazebo::physics::Link` (p. 472), `gazebo::physics::Model` (p. 533), `gazebo::physics::BulletLink` (p. 197), `gazebo::physics::ODELink` (p. 606), `gazebo::physics::MapShape` (p. 487), `gazebo::physics::ODERayShape` (p. 621), `gazebo::physics::BulletRayShape` (p. 212), `gazebo::physics::TrimeshShape` (p. 868), and `gazebo::physics::ODETrimeshShape` (p. 634).

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

Update the parameters using new sdf values.

Parameters

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

Reimplemented in `gazebo::physics::Joint` (p. 437), `gazebo::physics::Actor` (p. 125), `gazebo::physics::Model` (p. 533), `gazebo::physics::Entity` (p. 348), `gazebo::physics::Link` (p. 472), and `gazebo::physics::Collision` (p. 272).

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 <Box.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. 158) 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. 158) to add to this box
-----------------	-------------------	--

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

Addition operator.

result = this + `_b`

Parameters

<code>in</code>	<code>_b</code>	Box (p. 158) 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

<i>in</i>	<i>_b</i>	Box (p. 158) 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

<i>_v</i>	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

<i>in</i>	<i>_b</i>	Box (p. 158) to copy
-----------	-----------	-----------------------------

Returns

The new box.

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

Equality test operator.

Parameters

<i>in</i>	<i>_b</i>	Box (p. 158) 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

<i>in</i>	<i>_out</i>	Output stream
<i>in</i>	<i>_b</i>	Box (p. 158) 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:

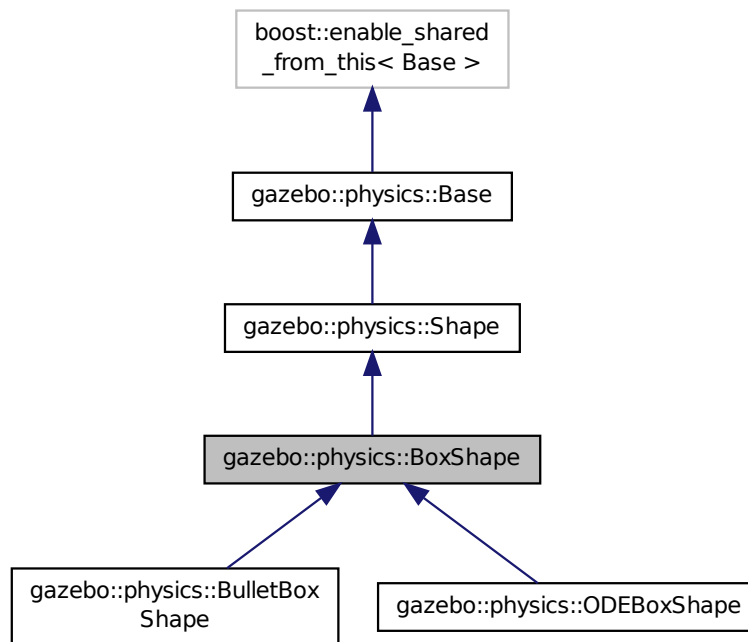
- **Box.hh**

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
Get inertial for a shape.
- virtual double **GetMass** (double _density) const
Get the mass of the box give a density value.
- **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. 262).
----	----------------	-----------------------------------

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. 781).

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

Deprecated.

Reimplemented from **gazebo::physics::Shape** (p. 781).

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

Get inertial for a shape.

Parameters

in	<i>_mass</i>	Mass of the box.
out	<i>_inertial</i>	Inertial (p. 414) element to populate with the result.

Reimplemented from **gazebo::physics::Shape** (p. 781).

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

Get the mass of the box give a density value.

Parameters

<code>in</code>	<code><i>_density</i></code>	Density to compute a mass from.
-----------------	------------------------------	---------------------------------

Reimplemented from `gazebo::physics::Shape` (p. 781).

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. 781).

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

Process a geometry message.

Parameters

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

Implements `gazebo::physics::Shape` (p. 782).

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

Reimplemented in `gazebo::physics::BulletBoxShape` (p. 171), and `gazebo::physics::ODEBoxShape` (p. 578).

Referenced by `gazebo::physics::ODEBoxShape::SetSize()`, and `gazebo::physics::BulletBoxShape::SetSize()`.

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

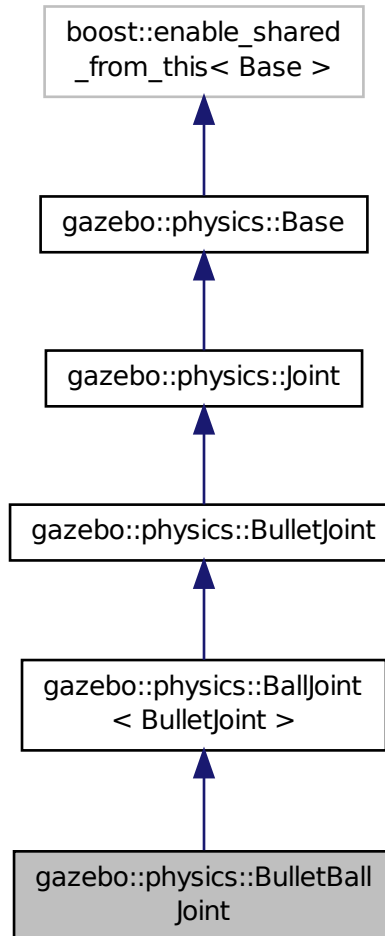
- **BoxShape.hh**

10.10 gazebo::physics::BulletBallJoint Class Reference

BulletBallJoint (p. 166) class models a ball joint in Bullet.

```
#include <BulletBallJoint.hh>
```

Inheritance diagram for gazebo::physics::BulletBallJoint:



Public Member Functions

- **BulletBallJoint** (btDynamicsWorld *_world, BasePtr _parent)
Bullet Ball Joint (p. 423) *Constructor.*
- virtual ~**BulletBallJoint** ()
Destructor.
- void **Attach** (LinkPtr _one, LinkPtr _two)

Attach the two bodies with this joint.

- **math::Vector3 GetAnchor** (int *_index*) const
Get joint's anchor point.
- virtual **math::Angle GetAngle** (int *_index*) const
Get the angle of rotation of an axis(index)
- virtual **math::Angle GetAngleImpl** (int *_index*) const
Get the angle of rotation.
- virtual **math::Vector3 GetAxis** (int) const
Get the axis of rotation.
- virtual **math::Vector3 GetGlobalAxis** (int *_index*) const
Get the axis of rotation.
- virtual double **GetMaxForce** (int *_index*)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int *_index*) const
Get the rotation rate of an axis(index)
- void **SetAnchor** (int *_index*, const **math::Vector3** &*_anchor*)
Set joint's anchor point.
- virtual void **SetDamping** (int *_index*, double *_damping*)
Set joint damping, not yet implemented.
- virtual void **SetHighStop** (int *_index*, const **math::Angle** &*_angle*)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int *_index*, const **math::Angle** &*_angle*)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int *_index*, double *_t*)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int *_index*, double *_angle*)
Set the velocity of an axis(index).

Additional Inherited Members

10.10.1 Detailed Description

BulletBallJoint (p. 166) class models a ball joint in Bullet.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 gazebo::physics::BulletBallJoint::BulletBallJoint (**btDynamicsWorld** * *_world*, **BasePtr** *_parent*)

Bullet Ball **Joint** (p. 423) Constructor.

10.10.2.2 virtual gazebo::physics::BulletBallJoint::~~BulletBallJoint () [virtual]

Destructor.

10.10.3 Member Function Documentation

10.10.3.1 `void gazebo::physics::BulletBallJoint::Attach (LinkPtr _one, LinkPtr _two) [virtual]`

Attach the two bodies with this joint.

Reimplemented from `gazebo::physics::Joint` (p. 427).

10.10.3.2 `math::Vector3 gazebo::physics::BulletBallJoint::GetAnchor (int _index) const [virtual]`

Get joint's anchor point.

Reimplemented from `gazebo::physics::BulletJoint` (p. 190).

10.10.3.3 `virtual math::Angle gazebo::physics::BulletBallJoint::GetAngle (int _index) const [virtual]`

Get the angle of rotation of an axis(index)

10.10.3.4 `virtual math::Angle gazebo::physics::BulletBallJoint::GetAngleImpl (int _index) const [virtual]`

Get the angle of rotation.

Implements `gazebo::physics::Joint` (p. 429).

10.10.3.5 `virtual math::Vector3 gazebo::physics::BulletBallJoint::GetAxis (int) const [inline],[virtual]`

Get the axis of rotation.

10.10.3.6 `virtual math::Vector3 gazebo::physics::BulletBallJoint::GetGlobalAxis (int _index) const [virtual]`

Get the axis of rotation.

Implements `gazebo::physics::Joint` (p. 430).

10.10.3.7 `virtual double gazebo::physics::BulletBallJoint::GetMaxForce (int _index) [virtual]`

Get the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 432).

10.10.3.8 `virtual double gazebo::physics::BulletBallJoint::GetVelocity (int _index) const [virtual]`

Get the rotation rate of an axis(index)

Implements `gazebo::physics::Joint` (p. 433).

10.10.3.9 `void gazebo::physics::BulletBallJoint::SetAnchor (int _index, const math::Vector3 & _anchor) [virtual]`

Set joint's anchor point.

Reimplemented from `gazebo::physics::BulletJoint` (p. 191).

10.10.3.10 virtual void gazebo::physics::BulletBallJoint::SetDamping (int *_index*, double *_damping*) [virtual]

Set joint damping, not yet implemented.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.10.3.11 virtual void gazebo::physics::BulletBallJoint::SetHighStop (int *_index*, const math::Angle & *_angle*) [virtual]

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.10.3.12 virtual void gazebo::physics::BulletBallJoint::SetLowStop (int *_index*, const math::Angle & *_angle*) [virtual]

Set the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.10.3.13 virtual void gazebo::physics::BulletBallJoint::SetMaxForce (int *_index*, double *_t*) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.10.3.14 virtual void gazebo::physics::BulletBallJoint::SetVelocity (int *_index*, double *_angle*) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

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

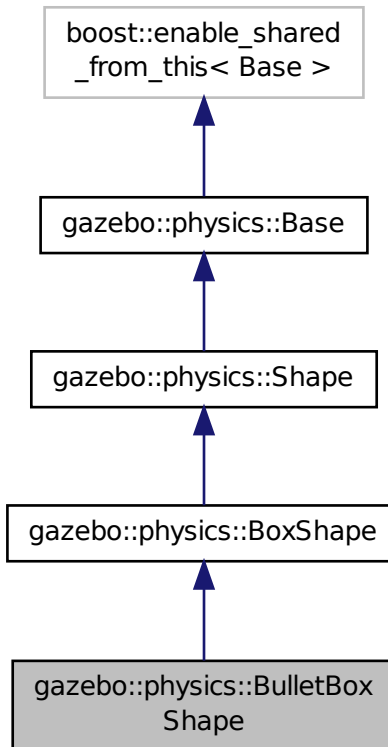
- **BulletBallJoint.hh**

10.11 gazebo::physics::BulletBoxShape Class Reference

Bullet box collision.

```
#include <BulletBoxShape.hh>
```

Inheritance diagram for gazebo::physics::BulletBoxShape:



Public Member Functions

- **BulletBoxShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~BulletBoxShape** ()
Destructor.
- void **SetSize** (const **math::Vector3** &_size)
Set the size of the box.

Additional Inherited Members

10.11.1 Detailed Description

Bullet box collision.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 `gazebo::physics::BulletBoxShape::BulletBoxShape (CollisionPtr _parent) [inline]`

Constructor.

10.11.2.2 `virtual gazebo::physics::BulletBoxShape::~~BulletBoxShape () [inline],[virtual]`

Destructor.

10.11.3 Member Function Documentation

10.11.3.1 `void gazebo::physics::BulletBoxShape::SetSize (const math::Vector3 & _size) [inline],[virtual]`

Set the size of the box.

Bullet requires the half-extents of the box

Reimplemented from `gazebo::physics::BoxShape` (p. 165).

References `gazebo::physics::Shape::collisionParent`, `gazebo::physics::BulletCollision::SetCollisionShape()`, `gazebo::physics::BoxShape::SetSize()`, `gazebo::math::Vector3::x`, `gazebo::math::Vector3::y`, and `gazebo::math::Vector3::z`.

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

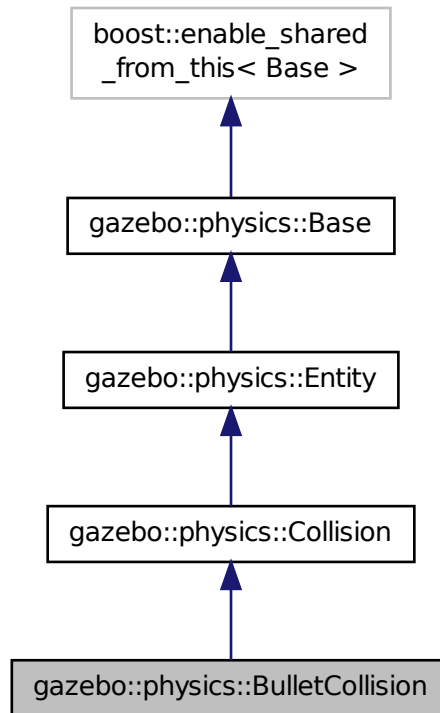
- `BulletBoxShape.hh`

10.12 gazebo::physics::BulletCollision Class Reference

Bullet collisions.

```
#include <BulletCollision.hh>
```

Inheritance diagram for gazebo::physics::BulletCollision:



Public Member Functions

- **BulletCollision** (**LinkPtr** _parent)
Constructor.
- virtual **~BulletCollision** ()
Destructor.
- virtual **math::Box** **GetBoundingBox** () const
Get the bounding box, defined by the physics engine.
- **btCollisionShape** * **GetCollisionShape** () const
Get the bullet collision shape.
- virtual void **Load** (**sdf::ElementPtr** _ptr)
Load the collision.
- virtual void **OnPoseChange** ()
On pose change.
- virtual void **SetCategoryBits** (unsigned int bits)
Set the category bits, used during collision detection.
- virtual void **SetCollideBits** (unsigned int bits)
Set the collide bits, used during collision detection.

- void **SetCollisionShape** (btCollisionShape ***shape**)
Set the collision shape.
- void **SetCompoundShapeIndex** (int index)
Set the index of the compound shape.

Protected Attributes

- btCollisionShape * **collisionShape**

Additional Inherited Members

10.12.1 Detailed Description

Bullet collisions.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 gazebo::physics::BulletCollision::BulletCollision (LinkPtr *parent*)

Constructor.

10.12.2.2 virtual gazebo::physics::BulletCollision::~~BulletCollision () [virtual]

Destructor.

10.12.3 Member Function Documentation

10.12.3.1 virtual math::Box gazebo::physics::BulletCollision::GetBoundingBox () const [virtual]

Get the bounding box, defined by the physics engine.

Implements **gazebo::physics::Collision** (p. 266).

10.12.3.2 btCollisionShape* gazebo::physics::BulletCollision::GetCollisionShape () const

Get the bullet collision shape.

10.12.3.3 virtual void gazebo::physics::BulletCollision::Load (sdf::ElementPtr *ptr*) [virtual]

Load the collision.

Reimplemented from **gazebo::physics::Collision** (p. 270).

10.12.3.4 virtual void gazebo::physics::BulletCollision::OnPoseChange () [virtual]

On pose change.

Implements **gazebo::physics::Entity** (p. 346).

10.12.3.5 virtual void gazebo::physics::BulletCollision::SetCategoryBits (unsigned int *bits*) [virtual]

Set the category bits, used during collision detection.

Parameters

<i>bits</i>	The bits
-------------	----------

Implements gazebo::physics::Collision (p. 270).

10.12.3.6 virtual void gazebo::physics::BulletCollision::SetCollideBits (unsigned int *bits*) [virtual]

Set the collide bits, used during collision detection.

Parameters

<i>bits</i>	The bits
-------------	----------

Implements gazebo::physics::Collision (p. 270).

10.12.3.7 void gazebo::physics::BulletCollision::SetCollisionShape (btCollisionShape * *shape*)

Set the collision shape.

Referenced by gazebo::physics::BulletPlaneShape::CreatePlane(), gazebo::physics::BulletSphereShape::SetRadius(), gazebo::physics::BulletBoxShape::SetSize(), and gazebo::physics::BulletCylinderShape::SetSize().

10.12.3.8 void gazebo::physics::BulletCollision::SetCompoundShapeIndex (int *index*)

Set the index of the compound shape.

10.12.4 Member Data Documentation

10.12.4.1 btCollisionShape* gazebo::physics::BulletCollision::collisionShape [protected]

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

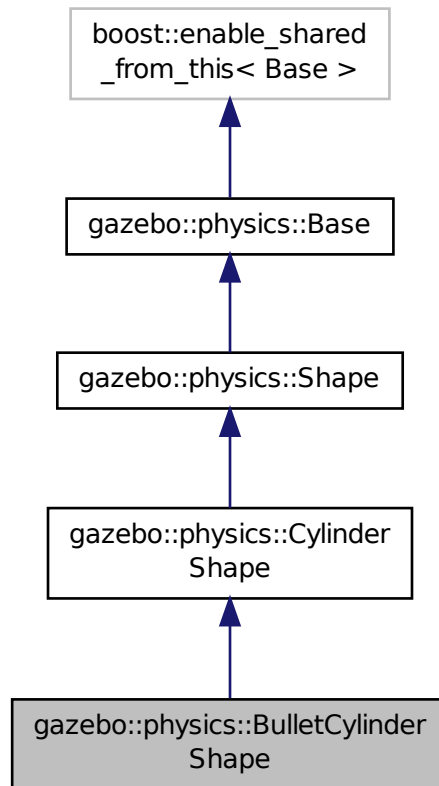
- **BulletCollision.hh**

10.13 gazebo::physics::BulletCylinderShape Class Reference

Cylinder collision.

```
#include <BulletCylinderShape.hh>
```

Inheritance diagram for gazebo::physics::BulletCylinderShape:



Public Member Functions

- **BulletCylinderShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~BulletCylinderShape** ()
Destructor.
- void **SetSize** (const double &_radius, const double &_length)
Set the size of the cylinder.

Additional Inherited Members

10.13.1 Detailed Description

Cylinder collision.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 `gazebo::physics::BulletCylinderShape::BulletCylinderShape (CollisionPtr _parent) [inline]`

Constructor.

10.13.2.2 `virtual gazebo::physics::BulletCylinderShape::~~BulletCylinderShape () [inline],[virtual]`

Destructor.

10.13.3 Member Function Documentation

10.13.3.1 `void gazebo::physics::BulletCylinderShape::SetSize (const double & _radius, const double & _length) [inline]`

Set the size of the cylinder.

References `gazebo::physics::Shape::collisionParent`, `gazebo::physics::BulletCollision::SetCollisionShape()`, and `gazebo::physics::CylinderShape::SetSize()`.

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

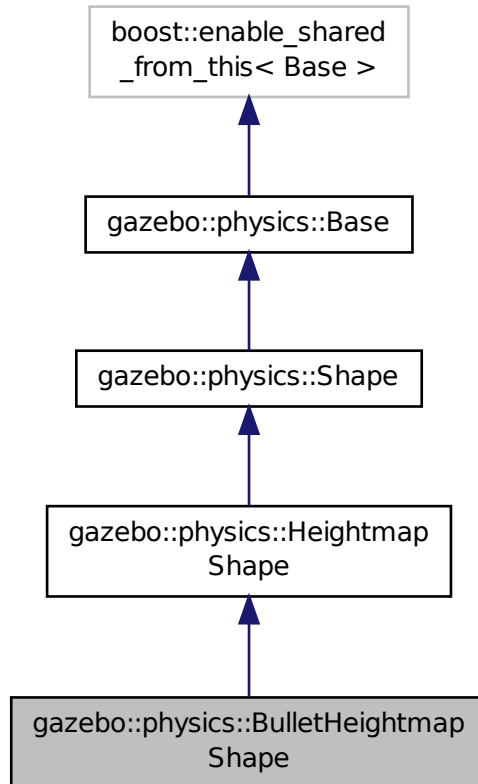
- `BulletCylinderShape.hh`

10.14 `gazebo::physics::BulletHeightmapShape` Class Reference

Height map collision.

```
#include <BulletHeightmapShape.hh>
```

Inheritance diagram for gazebo::physics::BulletHeightmapShape:



Public Member Functions

- **BulletHeightmapShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~BulletHeightmapShape** ()
Destructor.
- virtual void **Init** ()
Load the heightmap.

Additional Inherited Members

10.14.1 Detailed Description

Height map collision.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 gazebo::physics::BulletHeightmapShape::BulletHeightmapShape (CollisionPtr *_parent*)

Constructor.

10.14.2.2 virtual gazebo::physics::BulletHeightmapShape::~~BulletHeightmapShape () [virtual]

Destructor.

10.14.3 Member Function Documentation

10.14.3.1 virtual void gazebo::physics::BulletHeightmapShape::Init () [virtual]

Load the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 403).

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

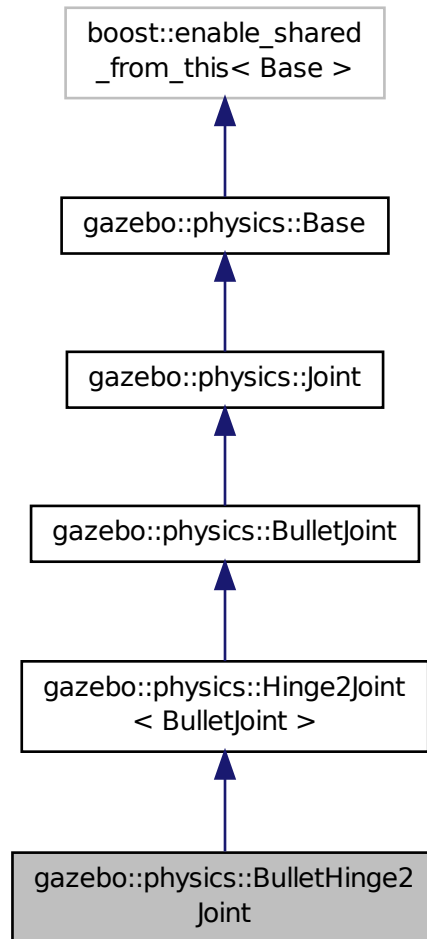
- **BulletHeightmapShape.hh**

10.15 gazebo::physics::BulletHinge2Joint Class Reference

A two axis hinge joint.

```
#include <BulletHinge2Joint.hh>
```

Inheritance diagram for gazebo::physics::BulletHinge2Joint:



Public Member Functions

- **BulletHinge2Joint** (btDynamicsWorld *world, BasePtr _parent)
Constructor.
- virtual ~**BulletHinge2Joint** ()
Destructor.
- virtual void **Attach** (LinkPtr _one, LinkPtr _two)
Attach the two bodies with this joint.
- virtual **math::Vector3 GetAnchor** (int _index) const
Get anchor point.
- **math::Angle GetAngle** (int _index) const
Get angle of rotation about first axis.

- virtual **math::Angle GetAngleImpl** (int _index) const
Get the angle of rotation.
- virtual **math::Vector3 GetAxis** (int _index) const
Get first axis of rotation.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const
Get the axis of rotation.
- virtual **math::Angle GetHighStop** (int _index)
Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)
Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)
Get the max allowed force of an axis(index).
- double **GetVelocity** (int _index) const
Get rate of rotation about first axis.
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)
Set the anchor point.
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)
Set the first axis of rotation.
- virtual void **SetDamping** (int _index, double _damping)
Set joint damping, not yet implemented.
- void **SetForce** (int _index, double _torque)
Set the torque.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load the **BulletHinge2Joint** (p. 178).*

Additional Inherited Members

10.15.1 Detailed Description

A two axis hinge joint.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 gazebo::physics::BulletHinge2Joint::BulletHinge2Joint (**btDynamicsWorld** * *world*, **BasePtr** *_parent*)

Constructor.

10.15.2.2 virtual gazebo::physics::BulletHinge2Joint::~~BulletHinge2Joint () [virtual]

Destructor.

10.15.3 Member Function Documentation

10.15.3.1 virtual void gazebo::physics::BulletHinge2Joint::Attach (LinkPtr *_one*, LinkPtr *_two*) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.15.3.2 virtual math::Vector3 gazebo::physics::BulletHinge2Joint::GetAnchor (int *_index*) const [virtual]

Get anchor point.

Reimplemented from **gazebo::physics::BulletJoint** (p. 190).

10.15.3.3 math::Angle gazebo::physics::BulletHinge2Joint::GetAngle (int *_index*) const

Get angle of rotation about first axis.

10.15.3.4 virtual math::Angle gazebo::physics::BulletHinge2Joint::GetAngleImpl (int *_index*) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.15.3.5 virtual math::Vector3 gazebo::physics::BulletHinge2Joint::GetAxis (int *_index*) const [virtual]

Get first axis of rotation.

10.15.3.6 virtual math::Vector3 gazebo::physics::BulletHinge2Joint::GetGlobalAxis (int *_index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.15.3.7 virtual math::Angle gazebo::physics::BulletHinge2Joint::GetHighStop (int *_index*) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

10.15.3.8 virtual math::Angle gazebo::physics::BulletHinge2Joint::GetLowStop (int *_index*) [virtual]

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.15.3.9 `virtual double gazebo::physics::BulletHinge2Joint::GetMaxForce (int _index) [virtual]`

Get the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 432).

10.15.3.10 `double gazebo::physics::BulletHinge2Joint::GetVelocity (int _index) const [virtual]`

Get rate of rotation about first axis.

Implements `gazebo::physics::Joint` (p. 433).

10.15.3.11 `virtual void gazebo::physics::BulletHinge2Joint::Load (sdf::ElementPtr _sdf) [protected],[virtual]`

Load the `BulletHinge2Joint` (p. 178).

Reimplemented from `gazebo::physics::Hinge2Joint< BulletJoint >` (p. 405).

10.15.3.12 `virtual void gazebo::physics::BulletHinge2Joint::SetAnchor (int _index, const math::Vector3 & _anchor) [virtual]`

Set the anchor point.

Reimplemented from `gazebo::physics::BulletJoint` (p. 191).

10.15.3.13 `virtual void gazebo::physics::BulletHinge2Joint::SetAxis (int _index, const math::Vector3 & _axis) [virtual]`

Set the first axis of rotation.

Implements `gazebo::physics::Joint` (p. 435).

10.15.3.14 `virtual void gazebo::physics::BulletHinge2Joint::SetDamping (int _index, double _damping) [virtual]`

Set joint damping, not yet implemented.

Reimplemented from `gazebo::physics::BulletJoint` (p. 191).

10.15.3.15 `void gazebo::physics::BulletHinge2Joint::SetForce (int _index, double _torque) [virtual]`

Set the torque.

Reimplemented from `gazebo::physics::Joint` (p. 435).

10.15.3.16 `virtual void gazebo::physics::BulletHinge2Joint::SetHighStop (int _index, const math::Angle & _angle) [virtual]`

Set the high stop of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.15.3.17 virtual void gazebo::physics::BulletHinge2Joint::SetLowStop (int *_index*, const math::Angle & *_angle*)
[virtual]

Set the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.15.3.18 virtual void gazebo::physics::BulletHinge2Joint::SetMaxForce (int *_index*, double *_f*) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.15.3.19 virtual void gazebo::physics::BulletHinge2Joint::SetVelocity (int *_index*, double *_angle*) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

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

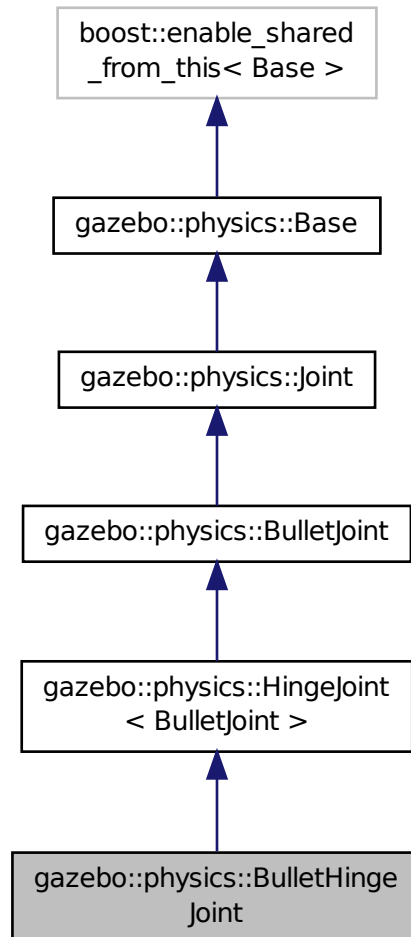
- **BulletHinge2Joint.hh**

10.16 gazebo::physics::BulletHingeJoint Class Reference

A single axis hinge joint.

```
#include <BulletHingeJoint.hh>
```

Inheritance diagram for gazebo::physics::BulletHingeJoint:



Public Member Functions

- **BulletHingeJoint** (btDynamicsWorld *world, BasePtr _parent)
Constructor.
- virtual ~**BulletHingeJoint** ()
Destructor.
- virtual void **Attach** (LinkPtr _one, LinkPtr _two)
Attach the two bodies with this joint.
- virtual **math::Vector3** **GetAnchor** (int _index) const
Get the anchor point.
- virtual **math::Angle** **GetAngle** (int _index) const
Get the angle of rotation.

- virtual **math::Angle GetAngleImpl** (int _index) const
Get the angle of rotation.
- virtual double **GetForce** (int _index)
Get the torque of a joint.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const
Get the axis of rotation.
- virtual **math::Angle GetHighStop** (int _index)
Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)
Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const
Get the rotation rate.
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)
Set the anchor point.
- void **SetAxis** (int _index, const **math::Vector3** &_axis)
Set the axis of rotation.
- virtual void **SetDamping** (int _index, double _damping)
Set joint damping, not yet implemented.
- void **SetForce** (int _index, double _torque)
Set the torque of a joint.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load the **BulletHingeJoint** (p. 183).*

Additional Inherited Members

10.16.1 Detailed Description

A single axis hinge joint.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 gazebo::physics::BulletHingeJoint::BulletHingeJoint (**btDynamicsWorld** * *world*, **BasePtr** _parent)

Constructor.

10.16.2.2 `virtual gazebo::physics::BulletHingeJoint::~~BulletHingeJoint () [virtual]`

Destructor.

10.16.3 Member Function Documentation

10.16.3.1 `virtual void gazebo::physics::BulletHingeJoint::Attach (LinkPtr _one, LinkPtr _two) [virtual]`

Attach the two bodies with this joint.

Reimplemented from `gazebo::physics::Joint` (p. 427).

10.16.3.2 `virtual math::Vector3 gazebo::physics::BulletHingeJoint::GetAnchor (int _index) const [virtual]`

Get the anchor point.

Reimplemented from `gazebo::physics::BulletJoint` (p. 190).

10.16.3.3 `virtual math::Angle gazebo::physics::BulletHingeJoint::GetAngle (int _index) const [virtual]`

Get the angle of rotation.

10.16.3.4 `virtual math::Angle gazebo::physics::BulletHingeJoint::GetAngleImpl (int _index) const [virtual]`

Get the angle of rotation.

Implements `gazebo::physics::Joint` (p. 429).

10.16.3.5 `virtual double gazebo::physics::BulletHingeJoint::GetForce (int _index) [virtual]`

Get the torque of a joint.

Reimplemented from `gazebo::physics::Joint` (p. 430).

10.16.3.6 `virtual math::Vector3 gazebo::physics::BulletHingeJoint::GetGlobalAxis (int _index) const [virtual]`

Get the axis of rotation.

Implements `gazebo::physics::Joint` (p. 430).

10.16.3.7 `virtual math::Angle gazebo::physics::BulletHingeJoint::GetHighStop (int _index) [virtual]`

Get the high stop of an axis(index).

Implements `gazebo::physics::Joint` (p. 430).

10.16.3.8 `virtual math::Angle gazebo::physics::BulletHingeJoint::GetLowStop (int _index) [virtual]`

Get the low stop of an axis(index).

Implements `gazebo::physics::Joint` (p. 432).

10.16.3.9 virtual double gazebo::physics::BulletHingeJoint::GetMaxForce (int *_index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.16.3.10 virtual double gazebo::physics::BulletHingeJoint::GetVelocity (int *_index*) const [virtual]

Get the rotation rate.

Implements **gazebo::physics::Joint** (p. 433).

10.16.3.11 virtual void gazebo::physics::BulletHingeJoint::Load (sdf::ElementPtr *_sdf*) [protected],[virtual]

Load the **BulletHingeJoint** (p. 183).

Reimplemented from **gazebo::physics::HingeJoint< BulletJoint >** (p. 407).

10.16.3.12 virtual void gazebo::physics::BulletHingeJoint::SetAnchor (int *_index*, const math::Vector3 & *_anchor*) [virtual]

Set the anchor point.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.16.3.13 void gazebo::physics::BulletHingeJoint::SetAxis (int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation.

Implements **gazebo::physics::Joint** (p. 435).

10.16.3.14 virtual void gazebo::physics::BulletHingeJoint::SetDamping (int *_index*, double *_damping*) [virtual]

Set joint damping, not yet implemented.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.16.3.15 void gazebo::physics::BulletHingeJoint::SetForce (int *_index*, double *_torque*) [virtual]

Set the torque of a joint.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.16.3.16 virtual void gazebo::physics::BulletHingeJoint::SetHighStop (int *_index*, const math::Angle & *_angle*) [virtual]

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.16.3.17 `virtual void gazebo::physics::BulletHingeJoint::SetLowStop (int _index, const math::Angle & _angle)` [virtual]

Set the low stop of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.16.3.18 `virtual void gazebo::physics::BulletHingeJoint::SetMaxForce (int _index, double _f)` [virtual]

Set the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.16.3.19 `virtual void gazebo::physics::BulletHingeJoint::SetVelocity (int _index, double _angle)` [virtual]

Set the velocity of an axis(index).

Implements `gazebo::physics::Joint` (p. 437).

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

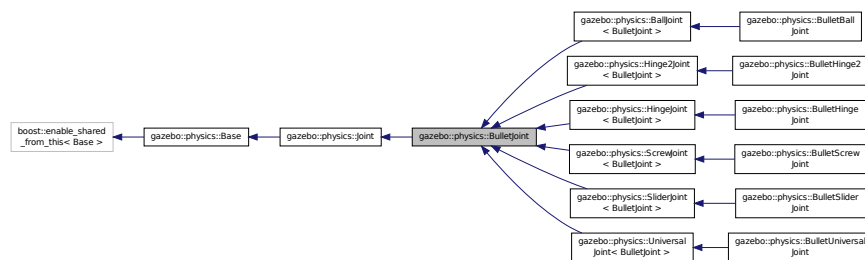
- `BulletHingeJoint.hh`

10.17 gazebo::physics::BulletJoint Class Reference

Base (p. 145) class for all joints.

```
#include <BulletJoint.hh>
```

Inheritance diagram for `gazebo::physics::BulletJoint`:



Public Member Functions

- **BulletJoint** (`BasePtr` *_parent*)
Constructor.
- `virtual ~BulletJoint` ()
Destructor.
- `bool AreConnected` (`LinkPtr` *_one*, `LinkPtr` *_two*) `const`
Determines of the two bodies are connected by a joint.
- `virtual void Detach` ()

Detach this joint from all bodies.

- virtual **math::Vector3 GetAnchor** (int) const
Get the anchor point.
- **LinkPtr GetJointLink** (int _index) const
Get the body to which the joint is attached according the _index.
- virtual **math::Vector3 GetLinkForce** (unsigned int) const
Get the force the joint applies to the first body.
- virtual **math::Vector3 GetLinkTorque** (unsigned int) const
Get the torque the joint applies to the first body.
- void **Load** (sdf::ElementPtr _sdf)
*Load a **BulletJoint** (p. 188).*
- virtual void **Reset** ()
Reset the joint.
- virtual void **SetAnchor** (int, const gazebo::math::Vector3 &)
Set the anchor point.
- virtual void **SetAttribute** (Attribute, int, double)
Set a parameter for the joint.
- virtual void **SetDamping** (int, const double)
Set the joint damping.

Protected Attributes

- btTypedConstraint * **constraint**
- btDynamicsWorld * **world**

Additional Inherited Members

10.17.1 Detailed Description

Base (p. 145) class for all joints.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 gazebo::physics::BulletJoint::BulletJoint (BasePtr _parent)

Constructor.

10.17.2.2 virtual gazebo::physics::BulletJoint::~~BulletJoint () [virtual]

Destructor.

10.17.3 Member Function Documentation

10.17.3.1 bool gazebo::physics::BulletJoint::AreConnected (LinkPtr _one, LinkPtr _two) const [virtual]

Determines if the two bodies are connected by a joint.

Implements **gazebo::physics::Joint** (p. 427).

10.17.3.2 `virtual void gazebo::physics::BulletJoint::Detach () [virtual]`

Detach this joint from all bodies.

Reimplemented from `gazebo::physics::Joint` (p. 428).

10.17.3.3 `virtual math::Vector3 gazebo::physics::BulletJoint::GetAnchor (int) const [inline],[virtual]`

Get the anchor point.

Implements `gazebo::physics::Joint` (p. 428).

Reimplemented in `gazebo::physics::ScrewJoint< BulletJoint >` (p. 761), `gazebo::physics::BulletHinge2Joint` (p. 181), `gazebo::physics::SliderJoint< BulletJoint >` (p. 805), `gazebo::physics::BulletHingeJoint` (p. 186), `gazebo::physics::BulletUniversalJoint` (p. 227), and `gazebo::physics::BulletBallJoint` (p. 168).

References gzerr.

10.17.3.4 `LinkPtr gazebo::physics::BulletJoint::GetJointLink (int _index) const [virtual]`

Get the body to which the joint is attached according the `_index`.

Implements `gazebo::physics::Joint` (p. 431).

10.17.3.5 `virtual math::Vector3 gazebo::physics::BulletJoint::GetLinkForce (unsigned int) const [inline],[virtual]`

Get the force the joint applies to the first body.

Parameters

<i>index</i>	The index of the body(0 or 1)
--------------	-------------------------------

Implements `gazebo::physics::Joint` (p. 431).

References gzerr.

10.17.3.6 `virtual math::Vector3 gazebo::physics::BulletJoint::GetLinkTorque (unsigned int) const [inline],[virtual]`

Get the torque the joint applies to the first body.

Parameters

<i>index</i>	The index of the body(0 or 1)
--------------	-------------------------------

Implements `gazebo::physics::Joint` (p. 431).

References gzerr.

10.17.3.7 `void gazebo::physics::BulletJoint::Load (sdf::ElementPtr _sdf) [virtual]`

Load a `BulletJoint` (p. 188).

Reimplemented from `gazebo::physics::Joint` (p. 433).

Reimplemented in **gazebo::physics::BallJoint**< **BulletJoint** > (p.144), **gazebo::physics::BulletScrewJoint** (p.216), **gazebo::physics::ScrewJoint**< **BulletJoint** > (p.761), **gazebo::physics::BulletSliderJoint** (p.220), **gazebo::physics::SliderJoint**< **BulletJoint** > (p.805), **gazebo::physics::Hinge2Joint**< **BulletJoint** > (p.405), **gazebo::physics::HingeJoint**< **BulletJoint** > (p.407), and **gazebo::physics::UniversalJoint**< **BulletJoint** > (p.870).

10.17.3.8 `virtual void gazebo::physics::BulletJoint::Reset () [virtual]`

Reset the joint.

Reimplemented from **gazebo::physics::Joint** (p.434).

10.17.3.9 `virtual void gazebo::physics::BulletJoint::SetAnchor (int , const gazebo::math::Vector3 &) [inline], [virtual]`

Set the anchor point.

Implements **gazebo::physics::Joint** (p.434).

Reimplemented in **gazebo::physics::ScrewJoint**< **BulletJoint** > (p.761), **gazebo::physics::BulletHingeJoint** (p.187), **gazebo::physics::BulletHinge2Joint** (p.182), **gazebo::physics::BulletUniversalJoint** (p.228), **gazebo::physics::SliderJoint**< **BulletJoint** > (p.806), and **gazebo::physics::BulletBallJoint** (p.168).

References gzerr.

10.17.3.10 `virtual void gazebo::physics::BulletJoint::SetAttribute (Attribute , int , double) [inline], [virtual]`

Set a parameter for the joint.

Implements **gazebo::physics::Joint** (p.435).

References gzerr.

10.17.3.11 `virtual void gazebo::physics::BulletJoint::SetDamping (int , const double) [inline], [virtual]`

Set the joint damping.

Implements **gazebo::physics::Joint** (p.435).

Reimplemented in **gazebo::physics::BulletHinge2Joint** (p.182), **gazebo::physics::BulletHingeJoint** (p.187), **gazebo::physics::BulletScrewJoint** (p.216), **gazebo::physics::BulletUniversalJoint** (p.228), **gazebo::physics::BulletSliderJoint** (p.220), and **gazebo::physics::BulletBallJoint** (p.169).

References gzerr.

10.17.4 Member Data Documentation

10.17.4.1 `btTypedConstraint* gazebo::physics::BulletJoint::constraint [protected]`

10.17.4.2 `btDynamicsWorld* gazebo::physics::BulletJoint::world [protected]`

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

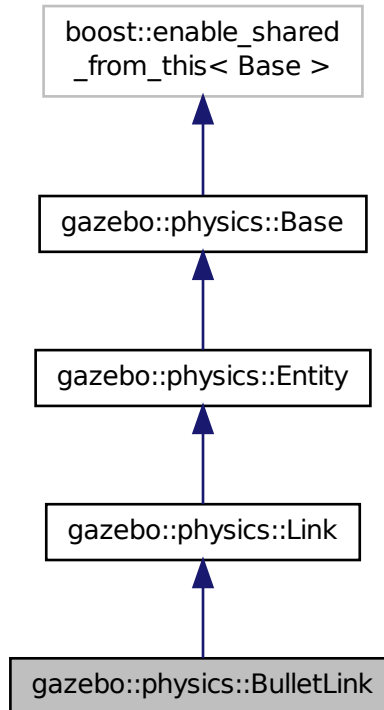
- **BulletJoint.hh**

10.18 gazebo::physics::BulletLink Class Reference

Bullet **Link** (p. 454) class.

```
#include <BulletLink.hh>
```

Inheritance diagram for gazebo::physics::BulletLink:



Public Member Functions

- **BulletLink** (**EntityPtr** _parent)
Constructor.
- virtual \sim **BulletLink** ()
Destructor.
- virtual void **AddForce** (const **math::Vector3** &_force)
Set the relative pose of a child collision.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)
Add a force to the body at position expressed to the body's own frame of reference.
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)
Add a force to the body using a global position.
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)

- Add a force to the body, components are relative to the body's own frame of reference.*

 - virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)
- Add a torque to the body, components are relative to the body's own frame of reference.*

 - virtual void **AddTorque** (const **math::Vector3** &_torque)
- Add a torque to the body.*

 - virtual void **Fini** ()
- Finalize the body.*

 - btRigidBody * **GetBulletLink** () const
- Get the bullet rigid body.*

 - virtual bool **GetEnabled** () const
- Get whether this body is enabled in the physics engine.*

 - virtual bool **GetGravityMode** ()
- Get the gravity mode.*

 - virtual **math::Vector3** **GetWorldAngularVel** () const
- Get the angular velocity of the body in the world frame.*

 - virtual **math::Vector3** **GetWorldForce** () const
- Get the force applied to the body in the world frame.*

 - virtual **math::Vector3** **GetWorldLinearVel** () const
- Get the linear velocity of the body in the world frame.*

 - virtual **math::Vector3** **GetWorldTorque** () const
- Get the torque applied to the body in the world frame.*

 - virtual void **Init** ()
- Initialize the body.*

 - virtual void **Load** (**sdf::ElementPtr** _ptr)
- Load the body based on an common::XMLConfig node.*

 - virtual void **OnPoseChange** ()
- Called when the pose of the entity (or one of its parents) has changed.*

 - virtual void **SetAngularDamping** (double damping)
- Set the angular damping factor.*

 - virtual void **SetAngularVel** (const **math::Vector3** &vel)
- Set the angular velocity of the body.*

 - virtual void **SetAutoDisable** (bool _disable)
- Allow the link to auto disable.*

 - virtual void **SetEnabled** (bool enable) const
- Set whether this body is enabled.*

 - virtual void **SetForce** (const **math::Vector3** &force)
- Set the force applied to the body.*

 - virtual void **SetGravityMode** (bool mode)
- Set whether gravity affects this body.*

 - virtual void **SetLinearDamping** (double damping)
- Set the linear damping factor.*

 - virtual void **SetLinearVel** (const **math::Vector3** &vel)
- Set the linear velocity of the body.*

 - void **SetSelfCollide** (bool collide)
- Set whether this body will collide with others in the model.*

 - virtual void **SetTorque** (const **math::Vector3** &force)
- Set the torque applied to the body.*

- virtual void **Update** ()
Update the body.
- virtual void **UpdateCoM** ()
Update the center of mass.

Protected Attributes

- **math::Pose** pose

Additional Inherited Members

10.18.1 Detailed Description

Bullet **Link** (p. 454) class.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 gazebo::physics::BulletLink::BulletLink (EntityPtr _parent)

Constructor.

10.18.2.2 virtual gazebo::physics::BulletLink::~~BulletLink () [virtual]

Destructor.

10.18.3 Member Function Documentation

10.18.3.1 virtual void gazebo::physics::BulletLink::AddForce (const math::Vector3 & _force) [virtual]

Set the relative pose of a child collision.

Add a force to the body

Implements **gazebo::physics::Link** (p. 459).

10.18.3.2 virtual void gazebo::physics::BulletLink::AddForceAtRelativePosition (const math::Vector3 & _force, const math::Vector3 & _relpos) [virtual]

Add a force to the body at position expressed to the body's own frame of reference.

Implements **gazebo::physics::Link** (p. 460).

10.18.3.3 virtual void gazebo::physics::BulletLink::AddForceAtWorldPosition (const math::Vector3 & _force, const math::Vector3 & _pos) [virtual]

Add a force to the body using a global position.

Implements **gazebo::physics::Link** (p. 460).

10.18.3.4 virtual void gazebo::physics::BulletLink::AddRelativeForce (const math::Vector3 & *_force*) [virtual]

Add a force to the body, components are relative to the body's own frame of reference.

Implements **gazebo::physics::Link** (p. 460).

10.18.3.5 virtual void gazebo::physics::BulletLink::AddRelativeTorque (const math::Vector3 & *_torque*) [virtual]

Add a torque to the body, components are relative to the body's own frame of reference.

Implements **gazebo::physics::Link** (p. 460).

10.18.3.6 virtual void gazebo::physics::BulletLink::AddTorque (const math::Vector3 & *_torque*) [virtual]

Add a torque to the body.

Implements **gazebo::physics::Link** (p. 461).

10.18.3.7 virtual void gazebo::physics::BulletLink::Fini () [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Entity** (p. 342).

10.18.3.8 btRigidBody* gazebo::physics::BulletLink::GetBulletLink () const

Get the bullet rigid body.

10.18.3.9 virtual bool gazebo::physics::BulletLink::GetEnabled () const [inline],[virtual]

Get whether this body is enabled in the physics engine.

Implements **gazebo::physics::Link** (p. 463).

10.18.3.10 virtual bool gazebo::physics::BulletLink::GetGravityMode () [virtual]

Get the gravity mode.

Implements **gazebo::physics::Link** (p. 463).

10.18.3.11 virtual math::Vector3 gazebo::physics::BulletLink::GetWorldAngularVel () const [virtual]

Get the angular velocity of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.18.3.12 virtual math::Vector3 gazebo::physics::BulletLink::GetWorldForce () const [virtual]

Get the force applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 467).

10.18.3.13 virtual `math::Vector3 gazebo::physics::BulletLink::GetWorldLinearVel () const` [virtual]

Get the linear velocity of the body in the world frame.

Reimplemented from `gazebo::physics::Entity` (p. 345).

10.18.3.14 virtual `math::Vector3 gazebo::physics::BulletLink::GetWorldTorque () const` [virtual]

Get the torque applied to the body in the world frame.

Implements `gazebo::physics::Link` (p. 467).

10.18.3.15 virtual void `gazebo::physics::BulletLink::Init ()` [virtual]

Initialize the body.

Reimplemented from `gazebo::physics::Link` (p. 467).

10.18.3.16 virtual void `gazebo::physics::BulletLink::Load (sdf::ElementPtr .ptr)` [virtual]

Load the body based on an `common::XMLConfig` node.

Reimplemented from `gazebo::physics::Link` (p. 468).

10.18.3.17 virtual void `gazebo::physics::BulletLink::OnPoseChange ()` [virtual]

Called when the pose of the entity (or one of its parents) has changed.

Reimplemented from `gazebo::physics::Link` (p. 468).

10.18.3.18 virtual void `gazebo::physics::BulletLink::SetAngularDamping (double damping)` [virtual]

Set the angular damping factor.

Implements `gazebo::physics::Link` (p. 469).

10.18.3.19 virtual void `gazebo::physics::BulletLink::SetAngularVel (const math::Vector3 & vel)` [virtual]

Set the angular velocity of the body.

Implements `gazebo::physics::Link` (p. 469).

10.18.3.20 virtual void `gazebo::physics::BulletLink::SetAutoDisable (bool _disable)` [virtual]

Allow the link to auto disable.

Parameters

<code>in</code>	<code>_disable</code>	If true, the link is allowed to auto disable.
-----------------	-----------------------	---

Implements `gazebo::physics::Link` (p. 469).

10.18.3.21 virtual void gazebo::physics::BulletLink::SetEnabled (bool *enable*) const [virtual]

Set whether this body is enabled.

Implements **gazebo::physics::Link** (p. 470).

10.18.3.22 virtual void gazebo::physics::BulletLink::SetForce (const math::Vector3 & *force*) [virtual]

Set the force applied to the body.

Implements **gazebo::physics::Link** (p. 470).

10.18.3.23 virtual void gazebo::physics::BulletLink::SetGravityMode (bool *mode*) [virtual]

Set whether gravity affects this body.

Implements **gazebo::physics::Link** (p. 470).

10.18.3.24 virtual void gazebo::physics::BulletLink::SetLinearDamping (double *damping*) [virtual]

Set the linear damping factor.

Implements **gazebo::physics::Link** (p. 471).

10.18.3.25 virtual void gazebo::physics::BulletLink::SetLinearVel (const math::Vector3 & *vel*) [virtual]

Set the linear velocity of the body.

Implements **gazebo::physics::Link** (p. 471).

10.18.3.26 void gazebo::physics::BulletLink::SetSelfCollide (bool *collide*) [virtual]

Set whether this body will collide with others in the model.

Implements **gazebo::physics::Link** (p. 471).

10.18.3.27 virtual void gazebo::physics::BulletLink::SetTorque (const math::Vector3 & *force*) [virtual]

Set the torque applied to the body.

Implements **gazebo::physics::Link** (p. 472).

10.18.3.28 virtual void gazebo::physics::BulletLink::Update () [virtual]

Update the body.

Reimplemented from **gazebo::physics::Link** (p. 472).

10.18.3.29 virtual void gazebo::physics::BulletLink::UpdateCoM () [virtual]

Update the center of mass.

10.18.4 Member Data Documentation

10.18.4.1 `math::Pose` `gazebo::physics::BulletLink::pose` [protected]

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

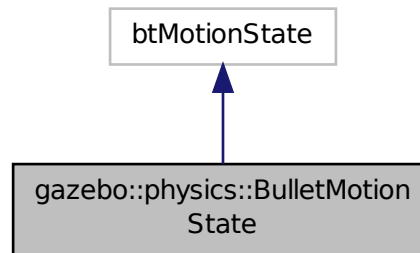
- **BulletLink.hh**

10.19 `gazebo::physics::BulletMotionState` Class Reference

Bullet `btMotionState` encapsulation.

```
#include <BulletMotionState.hh>
```

Inheritance diagram for `gazebo::physics::BulletMotionState`:



Public Member Functions

- **BulletMotionState** (`Link *_link`)
Constructor.
- virtual `~BulletMotionState` ()
Constructor.
- `math::Pose` **GetWorldPose** () const
Get the pose.
- virtual void **getWorldTransform** (`btTransform &_worldTrans`) const
Get the world transform.
- void **SetCoG** (const `math::Vector3` &_cog)
Set the center of mass offset.
- void **SetWorldPose** (const `math::Pose` &_pose)
Set the pose.
- virtual void **SetWorldPosition** (const `math::Vector3` &_pos)
Set the position of the body.
- virtual void **SetWorldRotation** (const `math::Quaternion` &_rot)
Set the rotation of the body.

- virtual void **setWorldTransform** (const btTransform &_worldTrans)
Set the world transform.

10.19.1 Detailed Description

Bullet btMotionState encapsulation.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 gazebo::physics::BulletMotionState::BulletMotionState (Link * *link*)

Constructor.

10.19.2.2 virtual gazebo::physics::BulletMotionState::~~BulletMotionState () [virtual]

Constructor.

Destructor

10.19.3 Member Function Documentation

10.19.3.1 math::Pose gazebo::physics::BulletMotionState::GetWorldPose () const

Get the pose.

10.19.3.2 virtual void gazebo::physics::BulletMotionState::getWorldTransform (btTransform & *_worldTrans*) const [virtual]

Get the world transform.

10.19.3.3 void gazebo::physics::BulletMotionState::SetCoG (const math::Vector3 & *_cog*)

Set the center of mass offset.

10.19.3.4 void gazebo::physics::BulletMotionState::SetWorldPose (const math::Pose & *_pose*)

Set the pose.

10.19.3.5 virtual void gazebo::physics::BulletMotionState::SetWorldPosition (const math::Vector3 & *_pos*) [virtual]

Set the position of the body.

Parameters

<i>pos</i>	math::Vector position
------------	-----------------------

10.19.3.6 virtual void gazebo::physics::BulletMotionState::SetWorldRotation (const math::Quaternion & _rot) [virtual]

Set the rotation of the body.

Parameters

<i>rot</i>	Quaternion rotation
------------	---------------------

10.19.3.7 virtual void gazebo::physics::BulletMotionState::setWorldTransform (const btTransform & _worldTrans) [virtual]

Set the world transform.

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

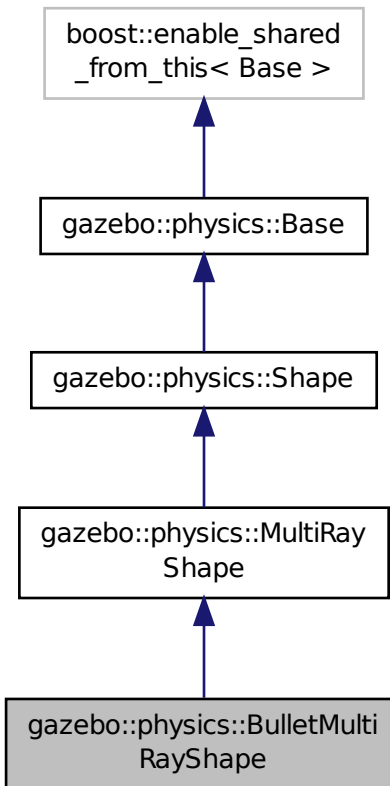
- **BulletMotionState.hh**

10.20 gazebo::physics::BulletMultiRayShape Class Reference

Bullet specific version of **MultiRayShape** (p. 549).

```
#include <BulletMultiRayShape.hh>
```

Inheritance diagram for gazebo::physics::BulletMultiRayShape:



Public Member Functions

- **BulletMultiRayShape** (CollisionPtr parent)

Constructor.

- virtual ~**BulletMultiRayShape** ()

Destructor.

- virtual void **UpdateRays** ()

Update the rays.

Protected Member Functions

- void **AddRay** (const math::Vector3 &start, const math::Vector3 &end)

Add a ray to the collision.

Additional Inherited Members

10.20.1 Detailed Description

Bullet specific version of **MultiRayShape** (p. 549).

10.20.2 Constructor & Destructor Documentation

10.20.2.1 gazebo::physics::BulletMultiRayShape::BulletMultiRayShape (CollisionPtr parent)

Constructor.

10.20.2.2 virtual gazebo::physics::BulletMultiRayShape::~~BulletMultiRayShape () [virtual]

Destructor.

10.20.3 Member Function Documentation

10.20.3.1 void gazebo::physics::BulletMultiRayShape::AddRay (const math::Vector3 & start, const math::Vector3 & end) [protected], [virtual]

Add a ray to the collision.

Reimplemented from **gazebo::physics::MultiRayShape** (p. 551).

10.20.3.2 virtual void gazebo::physics::BulletMultiRayShape::UpdateRays () [virtual]

Update the rays.

Implements **gazebo::physics::MultiRayShape** (p. 555).

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

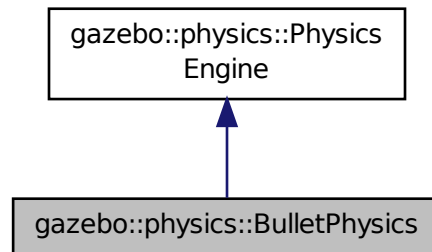
- **BulletMultiRayShape.hh**

10.21 gazebo::physics::BulletPhysics Class Reference

Bullet physics engine.

```
#include <BulletPhysics.hh>
```

Inheritance diagram for gazebo::physics::BulletPhysics:



Public Member Functions

- **BulletPhysics** (**WorldPtr** _world)
Constructor.
- virtual **~BulletPhysics** ()
Destructor.
- virtual void **ConvertMass** (**InertialPtr** _inertial, void *_engineMass)
Create a physics based ray sensor.
- virtual void **ConvertMass** (void *_engineMass, **InertialPtr** _inertial)
Convert an gazebo Mass to a bullet Mass.
- virtual **CollisionPtr** **CreateCollision** (const std::string &_type, **LinkPtr** _body)
Create a new collision.
- virtual **JointPtr** **CreateJoint** (const std::string &_type, **ModelPtr** _parent)
Create a new joint.
- virtual **LinkPtr** **CreateLink** (**ModelPtr** _parent)
Create a new body.
- virtual **ShapePtr** **CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)
*Create a **physics::Shape** (p. 779) object.*
- virtual void **DebugPrint** () const
Debug print out of the physic engine state.
- virtual void **Fini** ()
Finilize the Bullet engine.
- **btDynamicsWorld** * **GetDynamicsWorld** () const
Register a joint with the dynamics world.
- virtual double **GetStepTime** ()
Get the simulation step time.
- virtual void **Init** ()
Initialize the Bullet engine.
- virtual void **InitForThread** ()
Init the engine for threads.

- virtual void **Load** (`sdf::ElementPtr _sdf`)
Load the Bullet engine.
- virtual void **SetGravity** (const `gazebo::math::Vector3 &gravity`)
Set the gravity vector.
- virtual void **SetStepTime** (double `_value`)
Set the simulation step time.
- virtual void **UpdateCollision** ()
Update the Bullet collision.
- virtual void **UpdatePhysics** ()
Update the Bullet engine.

Static Public Member Functions

- static `math::Pose` **ConvertPose** (const `btTransform &_bt`)
Convert a bullet transform to a gazebo pose.
- static `btTransform` **ConvertPose** (const `math::Pose &_pose`)
Convert a gazebo pose to a bullet transform.

Additional Inherited Members

10.21.1 Detailed Description

Bullet physics engine.

10.21.2 Constructor & Destructor Documentation

10.21.2.1 `gazebo::physics::BulletPhysics::BulletPhysics (WorldPtr _world)`

Constructor.

10.21.2.2 `virtual gazebo::physics::BulletPhysics::~~BulletPhysics ()` `[virtual]`

Destructor.

10.21.3 Member Function Documentation

10.21.3.1 `virtual void gazebo::physics::BulletPhysics::ConvertMass (InertialPtr _inertial, void * _engineMass)` `[virtual]`

Create a physics based ray sensor.

Convert an bullet mass to a gazebo Mass

10.21.3.2 `virtual void gazebo::physics::BulletPhysics::ConvertMass (void * _engineMass, InertialPtr _inertial)` `[virtual]`

Convert an gazebo Mass to a bullet Mass.

10.21.3.3 `static math::Pose gazebo::physics::BulletPhysics::ConvertPose (const btTransform & _bt) [static]`

Convert a bullet transform to a gazebo pose.

10.21.3.4 `static btTransform gazebo::physics::BulletPhysics::ConvertPose (const math::Pose & _pose) [static]`

Convert a gazebo pose to a bullet transform.

10.21.3.5 `virtual CollisionPtr gazebo::physics::BulletPhysics::CreateCollision (const std::string & _type, LinkPtr _body) [virtual]`

Create a new collision.

Implements **gazebo::physics::PhysicsEngine** (p. 654).

10.21.3.6 `virtual JointPtr gazebo::physics::BulletPhysics::CreateJoint (const std::string & _type, ModelPtr _parent) [virtual]`

Create a new joint.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.21.3.7 `virtual LinkPtr gazebo::physics::BulletPhysics::CreateLink (ModelPtr _parent) [virtual]`

Create a new body.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.21.3.8 `virtual ShapePtr gazebo::physics::BulletPhysics::CreateShape (const std::string & _shapeType, CollisionPtr _collision) [virtual]`

Create a **physics::Shape** (p. 779) object.

Parameters

in	<i>_shapeType</i>	Type of shape to create.
in	<i>_collision</i>	Collision (p. 262) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.21.3.9 `virtual void gazebo::physics::BulletPhysics::DebugPrint () const [virtual]`

Debug print out of the physic engine state.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.21.3.10 `virtual void gazebo::physics::BulletPhysics::Fini () [virtual]`

Finilize the Bullet engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 656).

10.21.3.11 `btDynamicsWorld*` `gazebo::physics::BulletPhysics::GetDynamicsWorld () const` `[inline]`

Register a joint with the dynamics world.

10.21.3.12 `virtual double` `gazebo::physics::BulletPhysics::GetStepTime ()` `[virtual]`

Get the simulation step time.

Implements `gazebo::physics::PhysicsEngine` (p. 658).

10.21.3.13 `virtual void` `gazebo::physics::BulletPhysics::Init ()` `[virtual]`

Initialize the Bullet engine.

Implements `gazebo::physics::PhysicsEngine` (p. 659).

10.21.3.14 `virtual void` `gazebo::physics::BulletPhysics::InitForThread ()` `[virtual]`

Init the engine for threads.

Implements `gazebo::physics::PhysicsEngine` (p. 659).

10.21.3.15 `virtual void` `gazebo::physics::BulletPhysics::Load (sdf::ElementPtr _sdf)` `[virtual]`

Load the Bullet engine.

Reimplemented from `gazebo::physics::PhysicsEngine` (p. 659).

10.21.3.16 `virtual void` `gazebo::physics::BulletPhysics::SetGravity (const gazebo::math::Vector3 & gravity)` `[virtual]`

Set the gravity vector.

Implements `gazebo::physics::PhysicsEngine` (p. 660).

10.21.3.17 `virtual void` `gazebo::physics::BulletPhysics::SetStepTime (double _value)` `[virtual]`

Set the simulation step time.

Implements `gazebo::physics::PhysicsEngine` (p. 661).

10.21.3.18 `virtual void` `gazebo::physics::BulletPhysics::UpdateCollision ()` `[virtual]`

Update the Bullet collision.

Implements `gazebo::physics::PhysicsEngine` (p. 662).

10.21.3.19 `virtual void` `gazebo::physics::BulletPhysics::UpdatePhysics ()` `[virtual]`

Update the Bullet engine.

Reimplemented from `gazebo::physics::PhysicsEngine` (p. 662).

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

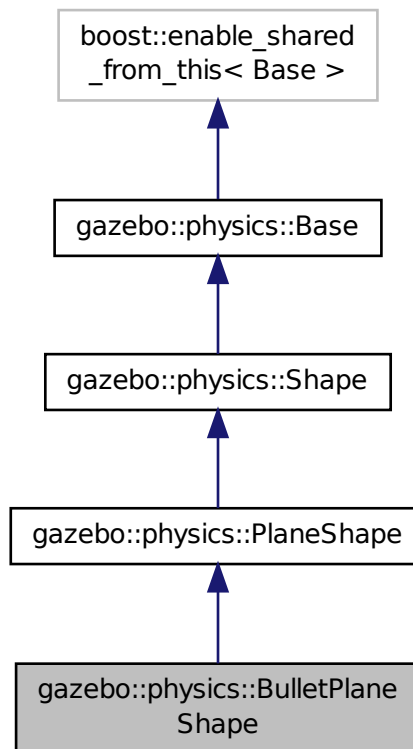
- **BulletPhysics.hh**

10.22 gazebo::physics::BulletPlaneShape Class Reference

Bullet collision for an infinite plane.

```
#include <BulletPlaneShape.hh>
```

Inheritance diagram for gazebo::physics::BulletPlaneShape:



Public Member Functions

- **BulletPlaneShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~BulletPlaneShape** ()
Destructor.
- void **CreatePlane** ()
Create the plane.
- void **SetAltitude** (const **math::Vector3** &pos)
Set the altitude of the plane.

Additional Inherited Members

10.22.1 Detailed Description

Bullet collision for an infinite plane.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 `gazebo::physics::BulletPlaneShape::BulletPlaneShape (CollisionPtr _parent) [inline]`

Constructor.

10.22.2.2 `virtual gazebo::physics::BulletPlaneShape::~~BulletPlaneShape () [inline],[virtual]`

Destructor.

10.22.3 Member Function Documentation

10.22.3.1 `void gazebo::physics::BulletPlaneShape::CreatePlane () [inline],[virtual]`

Create the plane.

Reimplemented from `gazebo::physics::PlaneShape` (p. 673).

References `gazebo::physics::Shape::collisionParent`, `gazebo::physics::PlaneShape::CreatePlane()`, `gazebo::physics::PlaneShape::GetNormal()`, and `gazebo::physics::BulletCollision::SetCollisionShape()`.

10.22.3.2 `void gazebo::physics::BulletPlaneShape::SetAltitude (const math::Vector3 & pos) [inline],[virtual]`

Set the altitude of the plane.

Reimplemented from `gazebo::physics::PlaneShape` (p. 673).

References `gazebo::physics::PlaneShape::SetAltitude()`.

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

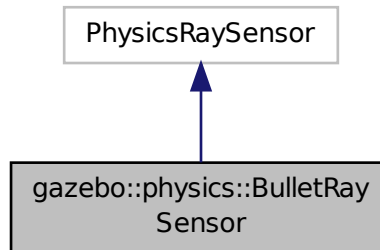
- `BulletPlaneShape.hh`

10.23 gazebo::physics::BulletRaySensor Class Reference

An Bullet Ray sensor.

```
#include <BulletRaySensor.hh>
```

Inheritance diagram for gazebo::physics::BulletRaySensor:



Public Member Functions

- **BulletRaySensor** (**Link** *body)
Constructor.
- virtual \sim **BulletRaySensor** ()
Destructor.
- void **AddRay** (**math::Vector3** start, **math::Vector3** end, double minRange, double maxRange, bool display)
Add a ray to the sensor.
- int **GetCount** () const
Get the number of rays.
- double **GetFiducial** (int index) const
Get the fiducial value of a ray.
- double **GetRange** (int index) const
Get the range of a ray.
- void **GetRelativePoints** (int index, **math::Vector3** &a, **math::Vector3** &b)
Get the relative starting and ending points of a ray.
- double **GetRetro** (int index) const
Get the retro reflectance value of a ray.
- virtual void **Update** ()
Update the ray sensor.

10.23.1 Detailed Description

An Bullet Ray sensor.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 gazebo::physics::BulletRaySensor::BulletRaySensor (**Link** * *body*)

Constructor.

10.23.2.2 `virtual gazebo::physics::BulletRaySensor::~~BulletRaySensor () [virtual]`

Destructor.

10.23.3 Member Function Documentation

10.23.3.1 `void gazebo::physics::BulletRaySensor::AddRay (math::Vector3 start, math::Vector3 end, double minRange, double maxRange, bool display)`

Add a ray to the sensor.

10.23.3.2 `int gazebo::physics::BulletRaySensor::GetCount () const`

Get the number of rays.

10.23.3.3 `double gazebo::physics::BulletRaySensor::GetFiducial (int index) const`

Get the fiducial value of a ray.

10.23.3.4 `double gazebo::physics::BulletRaySensor::GetRange (int index) const`

Get the range of a ray.

10.23.3.5 `void gazebo::physics::BulletRaySensor::GetRelativePoints (int index, math::Vector3 & a, math::Vector3 & b)`

Get the relative starting and ending points of a ray.

10.23.3.6 `double gazebo::physics::BulletRaySensor::GetRetro (int index) const`

Get the retro reflectance value of a ray.

10.23.3.7 `virtual void gazebo::physics::BulletRaySensor::Update () [virtual]`

Update the ray sensor.

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

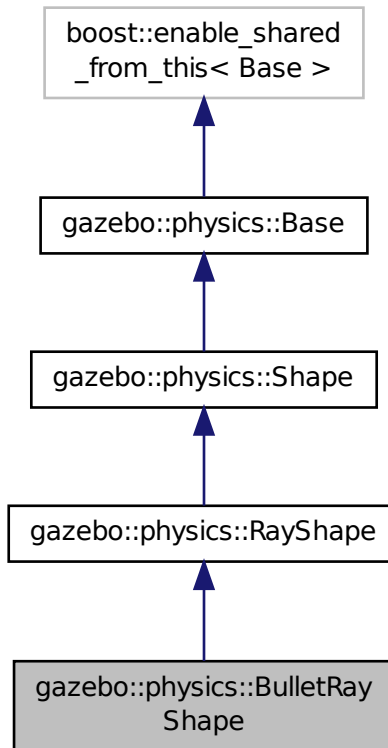
- **BulletRaySensor.hh**

10.24 gazebo::physics::BulletRayShape Class Reference

Ray shape for bullet.

```
#include <BulletRayShape.hh>
```

Inheritance diagram for gazebo::physics::BulletRayShape:



Public Member Functions

- **BulletRayShape** (**PhysicsEnginePtr** _physicsEngine)
- **BulletRayShape** (**CollisionPtr** _collision)
 - Constructor.*
- virtual **~BulletRayShape** ()
 - Destructor.*
- virtual void **GetIntersection** (double &_dist, std::string &_entity)
 - Get the nearest intersection.*
- void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)
 - Set the ray based on starting and ending points relative to the body.*
- virtual void **Update** ()
 - Update the ray collision.*

Additional Inherited Members

10.24.1 Detailed Description

Ray shape for bullet.

10.24.2 Constructor & Destructor Documentation

10.24.2.1 `gazebo::physics::BulletRayShape::BulletRayShape (PhysicsEnginePtr _physicsEngine)`

10.24.2.2 `gazebo::physics::BulletRayShape::BulletRayShape (CollisionPtr _collision)`

Constructor.

Parameters

<i>body</i>	Link (p. 454) the ray is attached to
-------------	---

10.24.2.3 `virtual gazebo::physics::BulletRayShape::~~BulletRayShape () [virtual]`

Destructor.

10.24.3 Member Function Documentation

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

Get the nearest intersection.

Implements **gazebo::physics::RayShape** (p. 720).

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

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

Parameters

<i>posStart</i>	Start position, relative the body
<i>posEnd</i>	End position, relative to the body

Reimplemented from **gazebo::physics::RayShape** (p. 721).

10.24.3.3 `virtual void gazebo::physics::BulletRayShape::Update () [virtual]`

Update the ray collision.

Implements **gazebo::physics::RayShape** (p. 722).

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

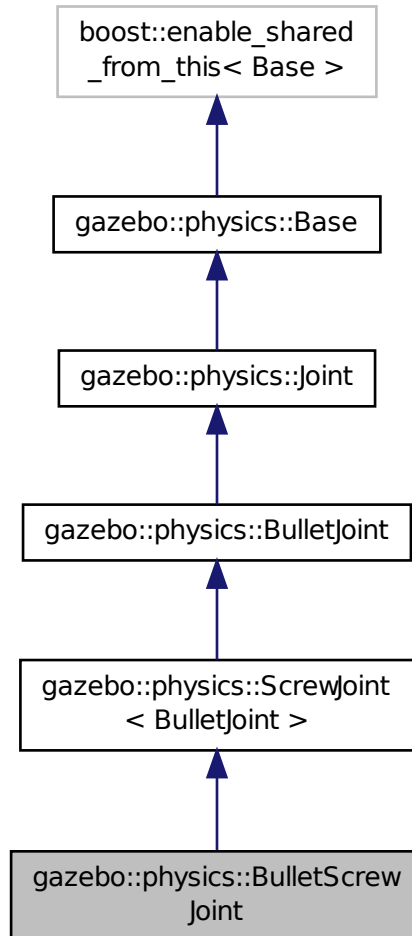
- **BulletRayShape.hh**

10.25 gazebo::physics::BulletScrewJoint Class Reference

A screw joint.

```
#include <BulletScrewJoint.hh>
```

Inheritance diagram for gazebo::physics::BulletScrewJoint:



Public Member Functions

- **BulletScrewJoint** (btDynamicsWorld *world, BasePtr _parent)
Constructor.
- virtual ~**BulletScrewJoint** ()
Destructor.
- void **Attach** (LinkPtr _one, LinkPtr _two)

Attach the two bodies with this joint.

- virtual **math::Angle** **GetAngle** (int _index) const
Get the position of the joint.
- virtual **math::Angle** **GetAngleImpl** (int _index) const
Get the angle of rotation.
- virtual **math::Vector3** **GetGlobalAxis** (int _index) const
Get the axis of rotation.
- virtual **math::Angle** **GetHighStop** (int _index)
Get the high stop of an axis(index).
- virtual **math::Angle** **GetLowStop** (int _index)
Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const
Get the rate of change.
- void **SetAxis** (int _index, const **math::Vector3** &_axis)
Set the axis of motion.
- virtual void **SetDamping** (int _index, double _damping)
Set joint damping, not yet implemented.
- virtual void **SetForce** (int _index, double _force)
Set the screw force.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load the **BulletScrewJoint** (p. 213).*

Additional Inherited Members

10.25.1 Detailed Description

A screw joint.

10.25.2 Constructor & Destructor Documentation

10.25.2.1 gazebo::physics::BulletScrewJoint::BulletScrewJoint (**btDynamicsWorld** * *world*, **BasePtr** *_parent*)

Constructor.

10.25.2.2 virtual gazebo::physics::BulletScrewJoint::~~BulletScrewJoint () [virtual]

Destructor.

10.25.3 Member Function Documentation

10.25.3.1 void gazebo::physics::BulletScrewJoint::Attach (LinkPtr *_one*, LinkPtr *_two*) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.25.3.2 virtual math::Angle gazebo::physics::BulletScrewJoint::GetAngle (int *_index*) const [virtual]

Get the position of the joint.

10.25.3.3 virtual math::Angle gazebo::physics::BulletScrewJoint::GetAngleImpl (int *_index*) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.25.3.4 virtual math::Vector3 gazebo::physics::BulletScrewJoint::GetGlobalAxis (int *_index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.25.3.5 virtual math::Angle gazebo::physics::BulletScrewJoint::GetHighStop (int *_index*) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

10.25.3.6 virtual math::Angle gazebo::physics::BulletScrewJoint::GetLowStop (int *_index*) [virtual]

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.25.3.7 virtual double gazebo::physics::BulletScrewJoint::GetMaxForce (int *_index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.25.3.8 virtual double gazebo::physics::BulletScrewJoint::GetVelocity (int *_index*) const [virtual]

Get the rate of change.

Implements **gazebo::physics::Joint** (p. 433).

10.25.3.9 `virtual void gazebo::physics::BulletScrewJoint::Load (sdf::ElementPtr _sdf)` [protected],[virtual]

Load the **BulletScrewJoint** (p. 213).

Reimplemented from **gazebo::physics::ScrewJoint< BulletJoint >** (p. 761).

10.25.3.10 `void gazebo::physics::BulletScrewJoint::SetAxis (int _index, const math::Vector3 & _axis)` [virtual]

Set the axis of motion.

Implements **gazebo::physics::Joint** (p. 435).

10.25.3.11 `virtual void gazebo::physics::BulletScrewJoint::SetDamping (int _index, double _damping)` [virtual]

Set joint damping, not yet implemented.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.25.3.12 `virtual void gazebo::physics::BulletScrewJoint::SetForce (int _index, double _force)` [virtual]

Set the screw force.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.25.3.13 `virtual void gazebo::physics::BulletScrewJoint::SetHighStop (int _index, const math::Angle & _angle)`
[virtual]

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.25.3.14 `virtual void gazebo::physics::BulletScrewJoint::SetLowStop (int _index, const math::Angle & _angle)`
[virtual]

Set the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.25.3.15 `virtual void gazebo::physics::BulletScrewJoint::SetMaxForce (int _index, double _f)` [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.25.3.16 `virtual void gazebo::physics::BulletScrewJoint::SetVelocity (int _index, double _angle)` [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

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

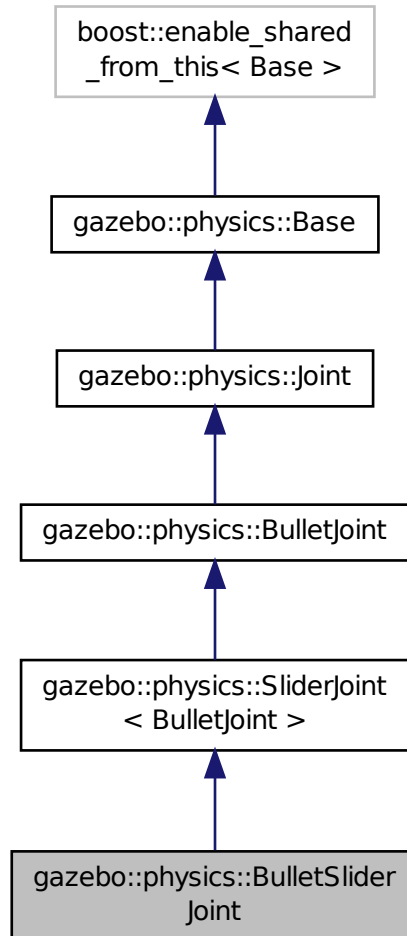
- **BulletScrewJoint.hh**

10.26 gazebo::physics::BulletSliderJoint Class Reference

A slider joint.

```
#include <BulletSliderJoint.hh>
```

Inheritance diagram for gazebo::physics::BulletSliderJoint:



Public Member Functions

- **BulletSliderJoint** (btDynamicsWorld *world, BasePtr _parent)
Constructor.
- virtual ~**BulletSliderJoint** ()
Destructor.
- void **Attach** (LinkPtr _one, LinkPtr _two)

Attach the two bodies with this joint.

- virtual **math::Angle GetAngle** (int _index) const
Get the position of the joint.
- virtual **math::Angle GetAngleImpl** (int _index) const
Get the angle of rotation.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const
Get the axis of rotation.
- virtual **math::Angle GetHighStop** (int _index)
Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)
Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const
Get the rate of change.
- void **SetAxis** (int _index, const **math::Vector3** &_axis)
Set the axis of motion.
- virtual void **SetDamping** (int _index, const double _damping)
Set joint damping, not yet implemented.
- virtual void **SetForce** (int _index, double _force)
Set the slider force.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load the **BulletSliderJoint** (p. 217).*

Additional Inherited Members

10.26.1 Detailed Description

A slider joint.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 gazebo::physics::BulletSliderJoint::BulletSliderJoint (**btDynamicsWorld** * *world*, **BasePtr** _parent)

Constructor.

10.26.2.2 virtual gazebo::physics::BulletSliderJoint::~~BulletSliderJoint () [virtual]

Destructor.

10.26.3 Member Function Documentation

10.26.3.1 void gazebo::physics::BulletSliderJoint::Attach (LinkPtr *_one*, LinkPtr *_two*) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.26.3.2 virtual math::Angle gazebo::physics::BulletSliderJoint::GetAngle (int *_index*) const [virtual]

Get the position of the joint.

10.26.3.3 virtual math::Angle gazebo::physics::BulletSliderJoint::GetAngleImpl (int *_index*) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.26.3.4 virtual math::Vector3 gazebo::physics::BulletSliderJoint::GetGlobalAxis (int *_index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.26.3.5 virtual math::Angle gazebo::physics::BulletSliderJoint::GetHighStop (int *_index*) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

10.26.3.6 virtual math::Angle gazebo::physics::BulletSliderJoint::GetLowStop (int *_index*) [virtual]

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.26.3.7 virtual double gazebo::physics::BulletSliderJoint::GetMaxForce (int *_index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.26.3.8 virtual double gazebo::physics::BulletSliderJoint::GetVelocity (int *_index*) const [virtual]

Get the rate of change.

Implements **gazebo::physics::Joint** (p. 433).

10.26.3.9 virtual void gazebo::physics::BulletSliderJoint::Load (sdf::ElementPtr _sdf) [protected],[virtual]

Load the **BulletSliderJoint** (p.217).

Reimplemented from **gazebo::physics::SliderJoint**< **BulletJoint** > (p.805).

10.26.3.10 void gazebo::physics::BulletSliderJoint::SetAxis (int _index, const math::Vector3 & _axis) [virtual]

Set the axis of motion.

Implements **gazebo::physics::Joint** (p.435).

10.26.3.11 virtual void gazebo::physics::BulletSliderJoint::SetDamping (int _index, const double _damping) [virtual]

Set joint damping, not yet implemented.

Reimplemented from **gazebo::physics::BulletJoint** (p.191).

10.26.3.12 virtual void gazebo::physics::BulletSliderJoint::SetForce (int _index, double _force) [virtual]

Set the slider force.

Reimplemented from **gazebo::physics::Joint** (p.435).

10.26.3.13 virtual void gazebo::physics::BulletSliderJoint::SetHighStop (int _index, const math::Angle & _angle)
[virtual]

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p.436).

10.26.3.14 virtual void gazebo::physics::BulletSliderJoint::SetLowStop (int _index, const math::Angle & _angle)
[virtual]

Set the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p.436).

10.26.3.15 virtual void gazebo::physics::BulletSliderJoint::SetMaxForce (int _index, double _f) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p.436).

10.26.3.16 virtual void gazebo::physics::BulletSliderJoint::SetVelocity (int _index, double _angle) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p.437).

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

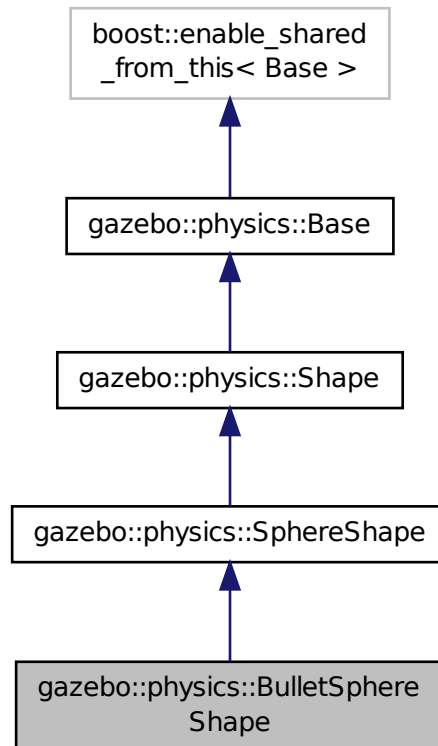
- **BulletSliderJoint.hh**

10.27 gazebo::physics::BulletSphereShape Class Reference

Bullet sphere collision.

```
#include <BulletSphereShape.hh>
```

Inheritance diagram for gazebo::physics::BulletSphereShape:



Public Member Functions

- **BulletSphereShape** (`CollisionPtr _parent`)
Constructor.
- virtual **~BulletSphereShape** ()
Destructor.
- void **SetRadius** (`const double &_radius`)
Set the radius.

Additional Inherited Members

10.27.1 Detailed Description

Bullet sphere collision.

10.27.2 Constructor & Destructor Documentation

10.27.2.1 `gazebo::physics::BulletSphereShape::BulletSphereShape (CollisionPtr _parent)` `[inline]`

Constructor.

10.27.2.2 `virtual gazebo::physics::BulletSphereShape::~~BulletSphereShape ()` `[inline],[virtual]`

Destructor.

10.27.3 Member Function Documentation

10.27.3.1 `void gazebo::physics::BulletSphereShape::SetRadius (const double & _radius)` `[inline]`

Set the radius.

References `gazebo::physics::Shape::collisionParent`, `gazebo::physics::BulletCollision::SetCollisionShape()`, and `gazebo::physics::SphereShape::SetRadius()`.

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

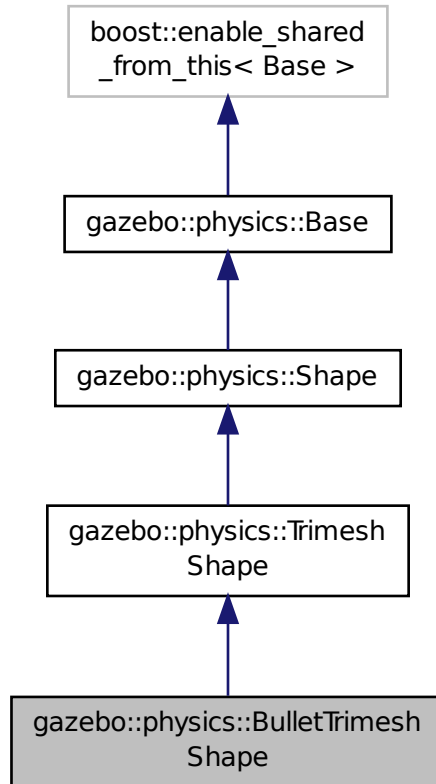
- **BulletSphereShape.hh**

10.28 gazebo::physics::BulletTrimeshShape Class Reference

Triangle mesh collision.

```
#include <BulletTrimeshShape.hh>
```

Inheritance diagram for gazebo::physics::BulletTrimeshShape:



Public Member Functions

- **BulletTrimeshShape** (`CollisionPtr` _parent)
Constructor.
- virtual `~BulletTrimeshShape` ()
Destructor.
- virtual void **Load** (`sdf::ElementPtr` _sdf)
Load the trimesh.

Protected Member Functions

- virtual void **Init** ()
Init the trimesh shape.

Additional Inherited Members

10.28.1 Detailed Description

Triangle mesh collision.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 gazebo::physics::BulletTrimeshShape::BulletTrimeshShape (CollisionPtr *_parent*)

Constructor.

10.28.2.2 virtual gazebo::physics::BulletTrimeshShape::~~BulletTrimeshShape () [virtual]

Destructor.

10.28.3 Member Function Documentation

10.28.3.1 virtual void gazebo::physics::BulletTrimeshShape::Init () [protected], [virtual]

Init the trimesh shape.

Reimplemented from **gazebo::physics::TrimeshShape** (p. 868).

10.28.3.2 virtual void gazebo::physics::BulletTrimeshShape::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the trimesh.

Reimplemented from **gazebo::physics::Base** (p. 154).

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

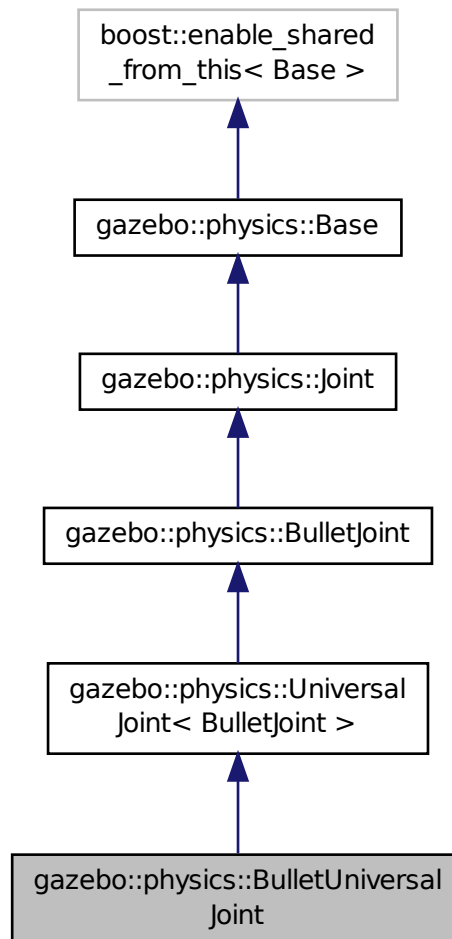
- **BulletTrimeshShape.hh**

10.29 gazebo::physics::BulletUniversalJoint Class Reference

A bullet universal joint class.

```
#include <BulletUniversalJoint.hh>
```

Inheritance diagram for gazebo::physics::BulletUniversalJoint:



Public Member Functions

- **BulletUniversalJoint** (btDynamicsWorld *world, BasePtr _parent)
Constructor.
- virtual ~**BulletUniversalJoint** ()
Destructor.
- void **Attach** (LinkPtr _one, LinkPtr _two)
Attach the two bodies with this joint.
- virtual **math::Vector3** **GetAnchor** (int _index) const
Get the anchor point.
- virtual **math::Angle** **GetAngle** (int _index) const
Get the angle of axis 1.

- virtual **math::Angle GetAngleImpl** (int _index) const
Get the angle of rotation.
- virtual **math::Vector3 GetAxis** (int _index) const
Get the first axis of rotation.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const
Get the axis of rotation.
- virtual **math::Angle GetHighStop** (int _index)
Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)
Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const
Get the angular rate of axis 1.
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)
Set the anchor point.
- void **SetAxis** (int _index, const **math::Vector3** &_axis)
Set the first axis of rotation.
- virtual void **SetDamping** (int _index, const double _damping)
Set joint damping, not yet implemented.
- virtual void **SetForce** (int _index, double _torque)
Set the torque of a joint.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)
Set the velocity of an axis(index).

Additional Inherited Members

10.29.1 Detailed Description

A bullet universal joint class.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 gazebo::physics::BulletUniversalJoint::BulletUniversalJoint (**btDynamicsWorld** * *world*, **BasePtr** *_parent*)

Constructor.

10.29.2.2 virtual gazebo::physics::BulletUniversalJoint::~~BulletUniversalJoint () [virtual]

Destuctor.

10.29.3 Member Function Documentation

10.29.3.1 void gazebo::physics::BulletUniversalJoint::Attach (LinkPtr *_one*, LinkPtr *_two*) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.29.3.2 virtual math::Vector3 gazebo::physics::BulletUniversalJoint::GetAnchor (int *_index*) const [virtual]

Get the anchor point.

Reimplemented from **gazebo::physics::BulletJoint** (p. 190).

10.29.3.3 virtual math::Angle gazebo::physics::BulletUniversalJoint::GetAngle (int *_index*) const [virtual]

Get the angle of axis 1.

10.29.3.4 virtual math::Angle gazebo::physics::BulletUniversalJoint::GetAngleImpl (int *_index*) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.29.3.5 virtual math::Vector3 gazebo::physics::BulletUniversalJoint::GetAxis (int *_index*) const [virtual]

Get the first axis of rotation.

10.29.3.6 virtual math::Vector3 gazebo::physics::BulletUniversalJoint::GetGlobalAxis (int *_index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.29.3.7 virtual math::Angle gazebo::physics::BulletUniversalJoint::GetHighStop (int *_index*) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

10.29.3.8 virtual math::Angle gazebo::physics::BulletUniversalJoint::GetLowStop (int *_index*) [virtual]

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.29.3.9 virtual double gazebo::physics::BulletUniversalJoint::GetMaxForce (int *_index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.29.3.10 `virtual double gazebo::physics::BulletUniversalJoint::GetVelocity (int _index) const` [virtual]

Get the angular rate of axis 1.

Implements `gazebo::physics::Joint` (p. 433).

10.29.3.11 `virtual void gazebo::physics::BulletUniversalJoint::SetAnchor (int _index, const math::Vector3 & _anchor)`
[virtual]

Set the anchor point.

Reimplemented from `gazebo::physics::BulletJoint` (p. 191).

10.29.3.12 `void gazebo::physics::BulletUniversalJoint::SetAxis (int _index, const math::Vector3 & _axis)` [virtual]

Set the first axis of rotation.

Implements `gazebo::physics::Joint` (p. 435).

10.29.3.13 `virtual void gazebo::physics::BulletUniversalJoint::SetDamping (int _index, const double _damping)` [virtual]

Set joint damping, not yet implemented.

Reimplemented from `gazebo::physics::BulletJoint` (p. 191).

10.29.3.14 `virtual void gazebo::physics::BulletUniversalJoint::SetForce (int _index, double _torque)` [virtual]

Set the torque of a joint.

Reimplemented from `gazebo::physics::Joint` (p. 435).

10.29.3.15 `virtual void gazebo::physics::BulletUniversalJoint::SetHighStop (int _index, const math::Angle & _angle)`
[virtual]

Set the high stop of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.29.3.16 `virtual void gazebo::physics::BulletUniversalJoint::SetLowStop (int _index, const math::Angle & _angle)`
[virtual]

Set the low stop of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.29.3.17 `virtual void gazebo::physics::BulletUniversalJoint::SetMaxForce (int _index, double _f)` [virtual]

Set the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.29.3.18 virtual void gazebo::physics::BulletUniversalJoint::SetVelocity (int *_index*, double *_angle*) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

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

- **BulletUniversalJoint.hh**

10.30 gazebo::common::BVHLoader Class Reference

Handles loading BVH animation files.

```
#include <BVHLoader.hh>
```

Public Member Functions

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

10.30.1 Detailed Description

Handles loading BVH animation files.

10.30.2 Constructor & Destructor Documentation

10.30.2.1 gazebo::common::BVHLoader::BVHLoader ()

Constructor.

10.30.2.2 gazebo::common::BVHLoader::~~BVHLoader ()

Desutrctor.

10.30.3 Member Function Documentation

10.30.3.1 **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. 784)

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

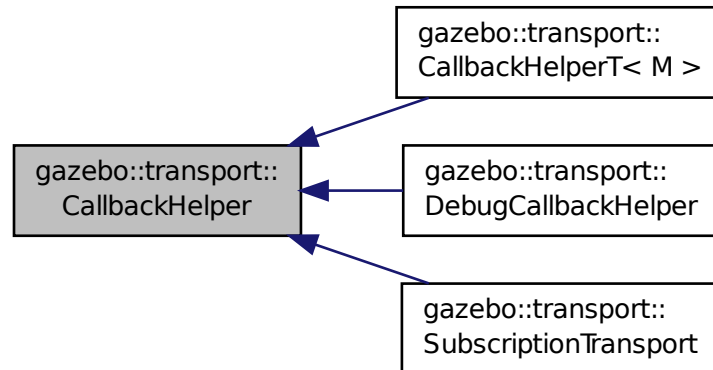
- **BVHLoader.hh**

10.31 gazebo::transport::CallbackHelper Class Reference

A helper class to handle callbacks when messages arrive.

```
#include <CallbackHelper.hh>
```

Inheritance diagram for gazebo::transport::CallbackHelper:

**Public Member Functions**

- **CallbackHelper** ()
- virtual **~CallbackHelper** ()
- bool **GetLatching** () const
- virtual std::string **GetMsgType** () const
Get the typename of the message that is handled.
- virtual bool **HandleData** (const std::string &newdata)=0
- virtual bool **IsLocal** () const =0

Return true if the callback is local, false if the callback is tied to a remote connection.

Protected Attributes

- bool **latching**

10.31.1 Detailed Description

A helper class to handle callbacks when messages arrive.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 `gazebo::transport::CallbackHelper::CallbackHelper ()` `[inline]`

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

10.31.3 Member Function Documentation

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

References latching.

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

Get the typename of the message that is handled.

Reimplemented in `gazebo::transport::DebugCallbackHelper` (p. 312), and `gazebo::transport::CallbackHelperT< M >` (p. 232).

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

Implemented in `gazebo::transport::DebugCallbackHelper` (p. 312), `gazebo::transport::CallbackHelperT< M >` (p. 233), and `gazebo::transport::SubscriptionTransport` (p. 829).

10.31.3.4 `virtual bool gazebo::transport::CallbackHelper::IsLocal () const` `[pure virtual]`

Return true if the callback is local, false if the callback is tied to a remote connection.

Implemented in `gazebo::transport::DebugCallbackHelper` (p. 312), `gazebo::transport::CallbackHelperT< M >` (p. 233), and `gazebo::transport::SubscriptionTransport` (p. 830).

10.31.4 Member Data Documentation

10.31.4.1 `bool gazebo::transport::CallbackHelper::latching` `[protected]`

Referenced by `GetLatching()`.

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

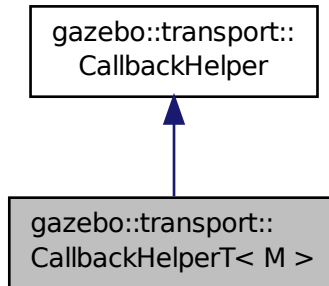
- `CallbackHelper.hh`

10.32 gazebo::transport::CallbackHelperT< M > Class Template Reference

Callback helper Template.

```
#include <CallbackHelper.hh>
```

Inheritance diagram for gazebo::transport::CallbackHelperT< M >:



Public Member Functions

- **CallbackHelperT** (const boost::function< void(const M const > &)*&cb)
- std::string **GetMsgType** () const
Get the typename of the message that is handled.
- virtual bool **HandleData** (const std::string &newdata)
- virtual bool **IsLocal** () const
Return true if the callback is local, false if the callback is tied to a remote connection.

Additional Inherited Members

10.32.1 Detailed Description

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

Callback helper Template.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 template<class M > gazebo::transport::CallbackHelperT< M >::CallbackHelperT (const boost::function< void(const M const > &) [inline]

10.32.3 Member Function Documentation

10.32.3.1 template<class M > std::string gazebo::transport::CallbackHelperT< M >::GetMsgType () const [inline],[virtual]

Get the typename of the message that is handled.

Reimplemented from **gazebo::transport::CallbackHelper** (p. 231).

References gzthrow, and NULL.

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

Implements `gazebo::transport::CallbackHelper` (p. 231).

10.32.3.3 `template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::IsLocal () const [inline],[virtual]`

Return true if the callback is local, false if the callback is tied to a remote connection.

Implements `gazebo::transport::CallbackHelper` (p. 231).

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

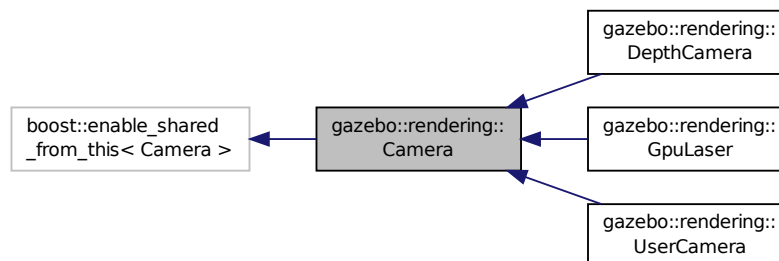
- `CallbackHelper.hh`

10.33 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, **Scene** *_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.*
 - 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.*
 - **Ogre::Texture** * **GetRenderTexture** () const
 - Get the render texture.*
 - **math::Vector3** **GetRight** ()
 - Get the viewport right vector.*
 - **Scene** * **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.
- `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.
- `void Load (sdf::ElementPtr _sdf)`
Load the camera with a set of parameters.
- `void Load ()`
Load the camera with default parameters.
- `virtual bool MoveToPosition (const math::Pose &_pose, double _time)`
Move the camera to a position.
- `bool MoveToPositions (const std::vector< math::Pose > &_pts, double _time, boost::function< void()> _on-Complete=NULL)`
Move the camera to a series of positions.

- 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** (**Scene** *_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 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. 253) call.
- virtual bool **TrackVisualImpl** (VisualPtr _visual)
Set the camera to track a scene node.

Protected Attributes

- Ogre::AnimationState * **animState**
- unsigned char * **bayerFrameBuffer**
- Ogre::Camera * **camera**
- bool **captureData**
- std::vector< event::ConnectionPtr > **connections**
- int **imageFormat**
- int **imageHeight**
- int **imageWidth**
- bool **initialized**
- common::Time **lastRenderWaitTime**
- std::string **name**
- bool **newData**
- event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)> **newImageFrame**

- boost::function< void()> **onAnimationComplete**
- Ogre::SceneNode * **pitchNode**
- **common::Time** **prevAnimTime**
- Ogre::RenderTarget * **renderTarget**
- Ogre::Texture * **renderTexture**
- std::list< msgs::Request > **requests**
- unsigned int **saveCount**
- unsigned char * **saveFrameBuffer**
- **Scene** * **scene**
- Ogre::SceneNode * **sceneNode**
- **sdf::ElementPtr** **sdf**
- unsigned int **textureHeight**
- unsigned int **textureWidth**
- Ogre::Viewport * **viewport**
- unsigned int **windowId**

10.33.1 Detailed Description

Basic camera sensor.

This is the base class for all cameras.

10.33.2 Constructor & Destructor Documentation

10.33.2.1 gazebo::rendering::Camera::Camera (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. 746) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

10.33.2.2 virtual gazebo::rendering::Camera::~Camera () [virtual]

Destructor.

10.33.3 Member Function Documentation

10.33.3.1 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

10.33.3.2 `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

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

Attach the camera to a visual.

Parameters

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

Returns

True on success

Reimplemented in `gazebo::rendering::UserCamera` (p. 880).

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

Connect a to the new image signal.

Parameters

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

Returns

A pointer to the connection. This must be kept in scope.

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

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

Set the render target.

Parameters

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

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

Disconnect from an image frame.

Parameters

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

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

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

Enable or disable saving.

Parameters

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

10.33.3.8 void gazebo::rendering::Camera::Fini ()

Finalize the camera.

This function is called before the camera is destructed

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

Get the aspect ratio.

Returns

The aspect ratio (width / height) in pixels

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

Get the average FPS.

Returns

The average frames per second

10.33.3.11 `void gazebo::rendering::Camera::GetCameraToViewportRay (int _screenx, int _screeny, math::Vector3 & _origin, math::Vector3 & _dir)`

Get a world space ray as cast from the camera through the viewport.

Parameters

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

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

Get the camera's direction vector.

Returns

Direction the camera is facing

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

Get the far clip distance.

Returns

Far clip distance

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

Get the next frame filename based on SDF parameters.

Returns

The frame's filename

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

Get the camera FOV (horizontal)

Returns

The horizontal field of view

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

Get the image size in bytes.

Returns

Size in bytes

10.33.3.17 `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

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

Get a pointer to the image data.

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

Parameters

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

Returns

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

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

Get the depth of the image.

Returns

Depth of the image

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

Get the height of the image.

Returns

String representation of the image format.

10.33.3.21 unsigned int gazebo::rendering::Camera::GetImageHeight () const

Get the height of the image.

Returns

Image height

10.33.3.22 unsigned int gazebo::rendering::Camera::GetImageWidth () const

Get the width of the image.

Returns

Image width

10.33.3.23 bool gazebo::rendering::Camera::GetInitialized () const

Returns true if initialized.

Returns

Ture if the camera is initialized

10.33.3.24 common::Time gazebo::rendering::Camera::GetLastRenderWallTime ()

Get the last time the camera was rendered.

Returns

Time the camera was last rendered

10.33.3.25 std::string gazebo::rendering::Camera::GetName () const

Get the camera's name.

Returns

The name of the camera

10.33.3.26 double gazebo::rendering::Camera::GetNearClip ()

Get the near clip distance.

Returns

Near clip distance

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

Get a pointer to the ogre camera.

Returns

Pointer to the OGRE camera

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

Get the render texture.

Returns

Pointer to the render texture

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

Get the viewport right vector.

Returns

The viewport right vector

10.33.3.30 `Scene*` gazebo::rendering::Camera::GetScene () const

Get the scene this camera is in.

Returns

Pointer to scene containing this camera

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

Get the camera's scene node.

Returns

The scene node the camera is attached to

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

Get the height of the off-screen render texture.

Returns

Render texture height

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

Get the width of the off-screen render texture.

Returns

Render texture width

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

Get the triangle count.

Returns

The current triangle count

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

Get the viewport up vector.

Returns

The viewport up vector

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

Get the camera FOV (vertical)

Returns

The vertical field of view

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

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

Returns

Pointer to the `Ogre::Viewport`

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

Get the viewport height in pixels.

Returns

The viewport height

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

Get the viewport width in pixels.

Returns

The viewport width

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

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

Returns

The ID of the window.

10.33.3.41 `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

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

Get the global pose of the camera.

Returns

Pose of the camera in the world coordinate frame

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

Get the camera position in the world.

Returns

The world position of the camera

10.33.3.44 `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. 697)

10.33.3.45 `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.

10.33.3.46 `void gazebo::rendering::Camera::Init ()`

Initialize the camera.

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

Return true if the camera has been initialized.

Returns

True if initialized was successful

References initialized.

10.33.3.48 `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

10.33.3.49 `bool gazebo::rendering::Camera::IsVisible (const std::string & _visualName)`

Return true if the visual is within the camera's view frustum.

Parameters

in	<i>_visualName</i>	Name of the visual to check for visibility
----	--------------------	--

Returns

True if the *_visual* is in the camera's frustum

10.33.3.50 `void gazebo::rendering::Camera::Load (sdf::ElementPtr _sdf)`

Load the camera with a set of parameters.

Parameters

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

10.33.3.51 `void gazebo::rendering::Camera::Load ()`

Load the camera with default parameters.

10.33.3.52 `virtual bool gazebo::rendering::Camera::MoveToPosition (const math::Pose & _pose, double _time) [virtual]`

Move the camera to a position.

This is an animated motion

Parameters

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

Reimplemented in `gazebo::rendering::UserCamera` (p. 883).

10.33.3.53 `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 positions.

Parameters

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

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

Post render.

Called after the render signal.

Reimplemented in `gazebo::rendering::GpuLaser` (p. 378), `gazebo::rendering::DepthCamera` (p. 316), and `gazebo-`

`::rendering::UserCamera` (p. 884).

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

Render the camera.

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

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

Implementation of the render call.

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

Rotate the camera around the pitch axis.

Parameters

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

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

Rotate the camera around the yaw axis.

Parameters

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

10.33.3.59 `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

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

Save a frame using an image buffer.

Parameters

<code>in</code>	<code>_image</code>	The raw image buffer
<code>in</code>	<code>_width</code>	Width of the image

in	<code>_height</code>	Height of the image
in	<code>_depth</code>	Depth of the image data
in	<code>_format</code>	Format the image data is in
in	<code>_filename</code>	Name of the file in which to write the frame

Returns

True if saving was successful

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

Set the aspect ratio.

Parameters

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

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

Set whether to capture data.

Parameters

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

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

Set the clip distances.

Parameters

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

10.33.3.64 `void gazebo::rendering::Camera::SetHFOV (math::Angle _angle)`

Set the camera FOV (horizontal)

Parameters

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

10.33.3.65 `void gazebo::rendering::Camera::SetImageHeight (unsigned int _h)`

Set the image height.

Parameters

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

10.33.3.66 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

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

Set the image height.

Parameters

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

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

Set the camera's name.

Parameters

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

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

Set the render Hz rate.

Parameters

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

10.33.3.70 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. 884).

10.33.3.71 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
----	------------------	--

10.33.3.72 void gazebo::rendering::Camera::SetScene (Scene * *_scene*)

Set the scene this camera is viewing.

Parameters

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

10.33.3.73 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
----	--------------	---

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

10.33.3.75 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. 677) of the camera
----	--------------	--

Reimplemented in **gazebo::rendering::UserCamera** (p. 885).

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

Set the world position.

Parameters

in	<i>_pos</i>	The new position of the camera
----	-------------	--------------------------------

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

Set the world orientation.

Parameters

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

10.33.3.78 void gazebo::rendering::Camera::ShowWireframe (bool *s*)

Set whether to view the world in wireframe.

Parameters

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

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

Toggle whether to view the world in wireframe.

10.33.3.80 void gazebo::rendering::Camera::TrackVisual (const std::string & *visualName*)

Set the camera to track a scene node.

Parameters

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

10.33.3.81 bool gazebo::rendering::Camera::TrackVisualImpl (const std::string & *visualName*) [protected]

Implementation of the **Camera::TrackVisual** (p. 253) call.

Parameters

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

Returns

True if able to track the visual

10.33.3.82 virtual bool gazebo::rendering::Camera::TrackVisualImpl (**VisualPtr** *visual*) [protected], [virtual]

Set the camera to track a scene node.

Parameters

in	_visual	The visual to track
----	---------	---------------------

Returns

True if able to track the visual

Reimplemented in **gazebo::rendering::UserCamera** (p. 885).

10.33.3.83 void gazebo::rendering::Camera::Translate (const math::Vector3 & *direction*)

Translate the camera.

Parameters

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

10.33.3.84 virtual void gazebo::rendering::Camera::Update () [virtual]

Reimplemented in **gazebo::rendering::UserCamera** (p. 885).

10.33.4 Member Data Documentation

10.33.4.1 Ogre::AnimationState* gazebo::rendering::Camera::animState [protected]

10.33.4.2 unsigned char* gazebo::rendering::Camera::bayerFrameBuffer [protected]

10.33.4.3 Ogre::Camera* gazebo::rendering::Camera::camera [protected]

10.33.4.4 bool gazebo::rendering::Camera::captureData [protected]

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

10.33.4.6 int gazebo::rendering::Camera::imageFormat [protected]

10.33.4.7 int gazebo::rendering::Camera::imageHeight [protected]

10.33.4.8 int gazebo::rendering::Camera::imageWidth [protected]

10.33.4.9 bool gazebo::rendering::Camera::initialized [protected]

Referenced by IsInitialized().

10.33.4.10 common::Time gazebo::rendering::Camera::lastRenderWallTime [protected]

10.33.4.11 std::string gazebo::rendering::Camera::name [protected]

10.33.4.12 bool gazebo::rendering::Camera::newData [protected]

10.33.4.13 event::EventT<void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>
gazebo::rendering::Camera::newImageFrame [protected]

Referenced by ConnectNewImageFrame(), and DisconnectNewImageFrame().

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

10.33.4.15 Ogre::SceneNode* gazebo::rendering::Camera::pitchNode [protected]

- 10.33.4.16 `common::Time` `gazebo::rendering::Camera::prevAnimTime` [protected]
- 10.33.4.17 `Ogre::RenderTarget*` `gazebo::rendering::Camera::renderTarget` [protected]
- 10.33.4.18 `Ogre::Texture*` `gazebo::rendering::Camera::renderTexture` [protected]
- 10.33.4.19 `std::list<msgs::Request>` `gazebo::rendering::Camera::requests` [protected]
- 10.33.4.20 `unsigned int` `gazebo::rendering::Camera::saveCount` [protected]
- 10.33.4.21 `unsigned char*` `gazebo::rendering::Camera::saveFrameBuffer` [protected]
- 10.33.4.22 `Scene*` `gazebo::rendering::Camera::scene` [protected]
- 10.33.4.23 `Ogre::SceneNode*` `gazebo::rendering::Camera::sceneNode` [protected]
- 10.33.4.24 `sdf::ElementPtr` `gazebo::rendering::Camera::sdf` [protected]
- 10.33.4.25 `unsigned int` `gazebo::rendering::Camera::textureHeight` [protected]
- 10.33.4.26 `unsigned int` `gazebo::rendering::Camera::textureWidth` [protected]
- 10.33.4.27 `Ogre::Viewport*` `gazebo::rendering::Camera::viewport` [protected]
- 10.33.4.28 `unsigned int` `gazebo::rendering::Camera::windowId` [protected]

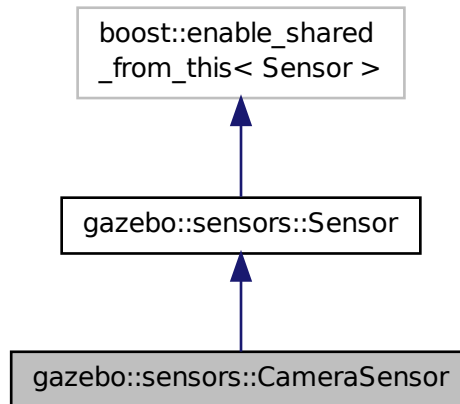
The documentation for this class was generated from the following file:

- [Camera.hh](#)

10.34 gazebo::sensors::CameraSensor Class Reference

```
#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. 233).*
- 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 **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 **UpdateImpl** (bool _force)
Update the sensor information.

Additional Inherited Members

10.34.1 Constructor & Destructor Documentation

10.34.1.1 gazebo::sensors::CameraSensor::CameraSensor ()

Constructor.

10.34.1.2 virtual gazebo::sensors::CameraSensor::~~CameraSensor () [virtual]

Destructor.

10.34.2 Member Function Documentation

10.34.2.1 virtual void gazebo::sensors::CameraSensor::Fini () [protected],[virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

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

Returns a pointer to the **rendering::Camera** (p. 233).

Returns

The Pointer to the camera sensor : nate check

10.34.2.3 const unsigned char* gazebo::sensors::CameraSensor::GetImageData ()

Gets the raw image data from the sensor.

Returns

The pointer to the data array : nate check

10.34.2.4 unsigned int gazebo::sensors::CameraSensor::GetImageHeight () const

Gets the height of the image in pixels.

Returns

The height in pixels of the image : nate check

10.34.2.5 `unsigned int gazebo::sensors::CameraSensor::GetImageWidth () const`

Gets the width of the image in pixels.

Returns

The width in pixels of the image : nate check

10.34.2.6 `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. 769).

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

Initialize the camera.

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

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

Load the sensor with SDF parameters.

Parameters

<code>in</code>	<code>_sdf</code>	SDF <code>Sensor</code> (p. 765) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

10.34.2.9 `virtual void gazebo::sensors::CameraSensor::Load (const std::string & _worldName)` [virtual]

Load the sensor with default parameters.

Parameters

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

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

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

Saves the image to the disk.

Parameters

<code>in</code>	<code>&_filename</code>	The name of the file to be saved
-----------------	-----------------------------	----------------------------------

Returns

True if successful, false if unsuccessful : nate check

10.34.2.11 `virtual void gazebo::sensors::CameraSensor::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 from `gazebo::sensors::Sensor` (p. 770).

10.34.2.12 `virtual void gazebo::sensors::CameraSensor::SetParent (const std::string & _name) [virtual]`

Set the parent of the sensor.

Parameters

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

Reimplemented from `gazebo::sensors::Sensor` (p. 771).

10.34.2.13 `virtual void gazebo::sensors::CameraSensor::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. 771).

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

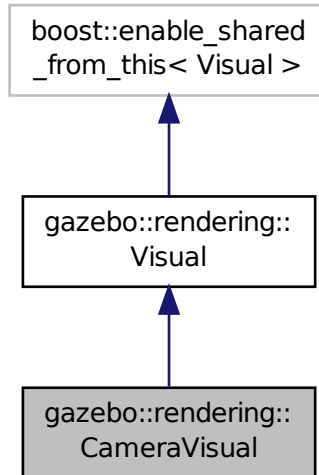
- `CameraSensor.hh`

10.35 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. 931).*

Additional Inherited Members

10.35.1 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.

10.35.2 Constructor & Destructor Documentation

10.35.2.1 gazebo::rendering::CameraVisual::CameraVisual (const std::string & _name, **VisualPtr** _vis)

Constructor.

Parameters

in	<code>_name</code>	Name of the Visual (p. 931)
in	<code>_vis</code>	Pointer to the parent Visual (p. 931)

10.35.2.2 `virtual gazebo::rendering::CameraVisual::~~CameraVisual () [virtual]`

Destructor.

10.35.3 Member Function Documentation

10.35.3.1 `void gazebo::rendering::CameraVisual::Load (unsigned int _width, unsigned int _height)`

Load the **Visual** (p. 931).

Parameters

in	<code>_width</code>	Width of the Camera (p. 233) image
in	<code>_height</code>	Height of the Camera (p. 233) image

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

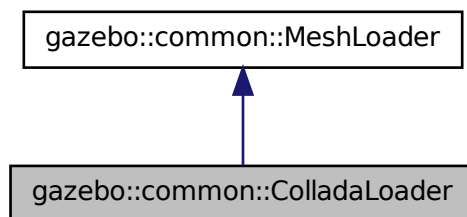
- **CameraVisual.hh**

10.36 gazebo::common::ColladaLoader Class Reference

Class used to load Collada mesh files.

```
#include <ColladaLoader.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.

10.36.1 Detailed Description

Class used to load Collada mesh files.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 gazebo::common::ColladaLoader::ColladaLoader ()

Constructor.

10.36.2.2 virtual gazebo::common::ColladaLoader::~~ColladaLoader () [virtual]

Destructor.

10.36.3 Member Function Documentation

10.36.3.1 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. 509)

Implements `gazebo::common::MeshLoader` (p. 515).

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

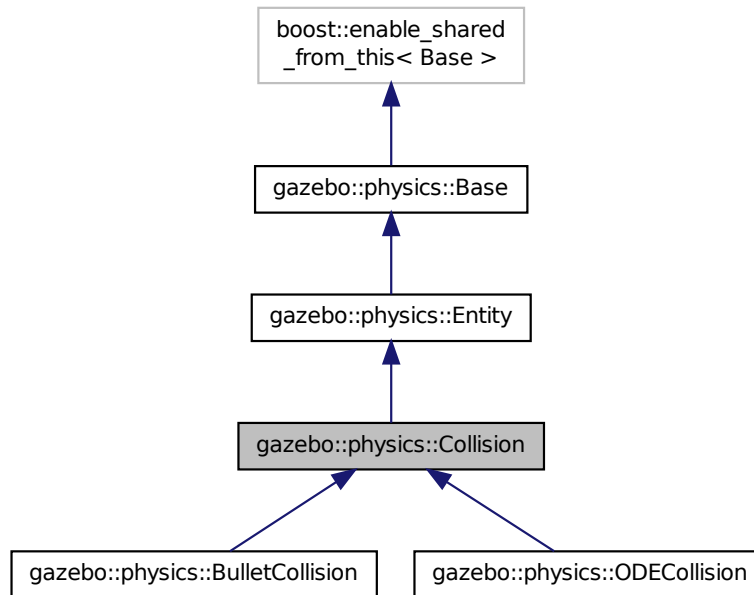
- **ColladaLoader.hh**

10.37 gazebo::physics::Collision Class Reference

Base (p. 145) 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)
Setup callback for contact event.
- void **DisconnectContact** (**event::ConnectionPtr** &_conn)
Disconnect callback for contact event.
- void **FillCollisionMsg** (msgs::Collision &_msg) **GAZEBO_DEPRECATED**
DEPRECATED.
- void **FillMsg** (msgs::Collision &_msg)
Fill a collision message.
- 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. 779).*

Additional Inherited Members

10.37.1 Detailed Description

Base (p. 145) class for all collision entities.

10.37.2 Constructor & Destructor Documentation

10.37.2.1 `gazebo::physics::Collision::Collision (LinkPtr .link)` `[explicit]`

Constructor.

Parameters

<code>in</code>	<code>_link</code>	Link (p. 454) that contains this collision object.
-----------------	--------------------	---

10.37.2.2 `virtual gazebo::physics::Collision::~~Collision ()` `[virtual]`

Destructor.

10.37.3 Member Function Documentation

10.37.3.1 `void gazebo::physics::Collision::AddContact (const Contact & .contact)`

Add an occurrence of a contact to this collision.

Parameters

<code>in</code>	<code>_contact</code>	The contact which was detected by a collision engine.
-----------------	-----------------------	---

10.37.3.2 `template<typename T > event::ConnectionPtr gazebo::physics::Collision::ConnectContact (T _subscriber)`
`[inline]`

Setup callback for contact event.

Parameters

<code>in</code>	<code>_subscriber</code>	The callback.
-----------------	--------------------------	---------------

Returns

Pointer to the new connection, which must be kept in scope.

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

10.37.3.3 `void gazebo::physics::Collision::DisconnectContact (event::ConnectionPtr & _conn)` `[inline]`

Disconnect callback for contact event.

Parameters

<code>in</code>	<code>_conn</code>	The connection to disconnect.
-----------------	--------------------	-------------------------------

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

10.37.3.4 `void gazebo::physics::Collision::FillCollisionMsg (msgs::Collision & _msg)`

DEPRECATED.

10.37.3.5 `void gazebo::physics::Collision::FillMsg (msgs::Collision & _msg)`

Fill a collision message.

Parameters

<code>out</code>	<code>_msg</code>	The message to fill with this collision's data.
------------------	-------------------	---

10.37.3.6 `void gazebo::physics::Collision::Fini ()` `[virtual]`

Finalize the collision.

Reimplemented from `gazebo::physics::Entity` (p. 342).

Reimplemented in `gazebo::physics::ODECollision` (p. 580).

10.37.3.7 `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. 342).

Implemented in **gazebo::physics::ODECollision** (p. 580), and **gazebo::physics::BulletCollision** (p. 173).

10.37.3.8 `bool gazebo::physics::Collision::GetContactsEnabled () const`

Return true of contacts are on.

Returns

True of contact are on.

10.37.3.9 `float gazebo::physics::Collision::GetLaserRetro () const`

Get the laser retro reflectiveness.

Returns

The laser retro value.

10.37.3.10 `LinkPtr gazebo::physics::Collision::GetLink () const`

Get the link this collision belongs to.

Returns

The parent **Link** (p. 454).

10.37.3.11 `ModelPtr gazebo::physics::Collision::GetModel () const`

Get the model this collision belongs to.

Returns

The parent model.

10.37.3.12 `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. 343).

10.37.3.13 `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. 343).

10.37.3.14 `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. 344).

10.37.3.15 `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. 344).

10.37.3.16 `ShapePtr gazebo::physics::Collision::GetShape () const`

Get the collision shape.

Returns

The collision shape.

10.37.3.17 `unsigned int gazebo::physics::Collision::GetShapeType ()`

Get the shape type.

Returns

The shape type.

See Also

`EntityType` (p. 149)

10.37.3.18 CollisionState gazebo::physics::Collision::GetState ()

Get the collision state.

Returns

The collision state.

10.37.3.19 SurfaceParamsPtr gazebo::physics::Collision::GetSurface () const [inline]

Get the surface parameters.

Returns

The surface parameters.

10.37.3.20 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. 344).

10.37.3.21 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. 344).

10.37.3.22 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. 345).

10.37.3.23 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. 345).

10.37.3.24 `virtual void gazebo::physics::Collision::Init () [virtual]`

Initialize the collision.

Reimplemented from **`gazebo::physics::Base`** (p. 153).

10.37.3.25 `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.

10.37.3.26 `virtual void gazebo::physics::Collision::Load (sdf::ElementPtr _sdf) [virtual]`

Load the collision.

Parameters

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

Reimplemented from **`gazebo::physics::Entity`** (p. 346).

Reimplemented in **`gazebo::physics::BulletCollision`** (p. 173), and **`gazebo::physics::ODECollision`** (p. 581).

10.37.3.27 `void gazebo::physics::Collision::ProcessMsg (const msgs::Collision & _msg)`

Update parameters from a message.

Parameters

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

10.37.3.28 `virtual void gazebo::physics::Collision::SetCategoryBits (unsigned int _bits) [pure virtual]`

Set the category bits, used during collision detection.

Parameters

<code>in</code>	<code>_bits</code>	The bits to set.
-----------------	--------------------	------------------

Implemented in **`gazebo::physics::ODECollision`** (p. 581), and **`gazebo::physics::BulletCollision`** (p. 174).

10.37.3.29 `virtual void gazebo::physics::Collision::SetCollideBits (unsigned int _bits) [pure virtual]`

Set the collide bits, used during collision detection.

Parameters

in	<i>_bits</i>	The bits to set.
----	--------------	------------------

Implemented in **gazebo::physics::ODECollision** (p. 581), and **gazebo::physics::BulletCollision** (p. 174).

10.37.3.30 void gazebo::physics::Collision::SetCollision (bool *_placeable*)

Set the encapsulated collision object.

Parameters

in	<i>_placeable</i>	True to make the object m.
----	-------------------	----------------------------

10.37.3.31 void gazebo::physics::Collision::SetContactsEnabled (bool *_enable*)

Turn contact recording on or off.

Parameters

in	<i>_enable</i>	True to enable collision contacts.
----	----------------	------------------------------------

10.37.3.32 void gazebo::physics::Collision::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	<i>_retro</i>	The laser retro value.
----	---------------	------------------------

10.37.3.33 void gazebo::physics::Collision::SetShape (ShapePtr *_shape*)

Set the shape for this collision.

Parameters

in	<i>_shape</i>	The shape for this collision object.
----	---------------	--------------------------------------

10.37.3.34 void gazebo::physics::Collision::SetState (const CollisionState & *_state*)

Set the current collision state.

Parameters

in	<i>The</i>	collision state.
----	------------	------------------

10.37.3.35 `virtual void gazebo::physics::Collision::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. 348).

10.37.4 Member Data Documentation

10.37.4.1 `LinkPtr gazebo::physics::Collision::link [protected]`

The link this collision belongs to.

10.37.4.2 `bool gazebo::physics::Collision::placeable [protected]`

Flag for placeable.

10.37.4.3 `ShapePtr gazebo::physics::Collision::shape [protected]`

Pointer to `physics::Shape` (p. 779).

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

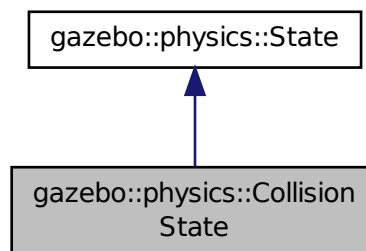
- `Collision.hh`

10.38 gazebo::physics::CollisionState Class Reference

Store state information of a `physics::Collision` (p. 262) object.

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

Inheritance diagram for `gazebo::physics::CollisionState`:



Public Member Functions

- **CollisionState** ()
Default constructor.
- **CollisionState** (const **CollisionPtr** _collision)
Constructor.
- virtual **~CollisionState** ()
Destructor.
- **math::Pose GetPose** () const
*Get the **Collision** (p. 262) pose.*
- virtual void **Load** (**sdf::ElementPtr** _elem)
Load state from SDF element.

Additional Inherited Members

10.38.1 Detailed Description

Store state information of a **physics::Collision** (p. 262) object.

This class captures the entire state of a **Collision** (p. 262) at one specific time during a simulation run.

State (p. 813) of a **Collision** (p. 262) is its Pose.

10.38.2 Constructor & Destructor Documentation

10.38.2.1 gazebo::physics::CollisionState::CollisionState ()

Default constructor.

10.38.2.2 gazebo::physics::CollisionState::CollisionState (const **CollisionPtr** _collision)

Constructor.

Build a **CollisionState** (p. 272) from an existing **Collision** (p. 262).

Parameters

in	_model	Pointer to the Link (p. 454) from which to gather state info.
----	--------	--

10.38.2.3 virtual gazebo::physics::CollisionState::~~CollisionState () [virtual]

Destructor.

10.38.3 Member Function Documentation

10.38.3.1 **math::Pose** gazebo::physics::CollisionState::GetPose () const

Get the **Collision** (p. 262) pose.

10.38.3.2 virtual void gazebo::physics::CollisionState::Load (sdf::ElementPtr _elem) [virtual]

Load state from SDF element.

Load **CollisionState** (p. 272) information from stored data in and SDF::Element

Parameters

in	_elem	Pointer to the SDF::Element containing state info.
----	-------	--

Implements **gazebo::physics::State** (p. 815).

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

- **CollisionState.hh**

10.39 gazebo::common::Color Class Reference

Defines a color.

```
#include <Color.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.

10.39.1 Detailed Description

Defines a color.

10.39.2 Member Typedef Documentation

10.39.2.1 typedef unsigned int gazebo::common::Color::ABGR

10.39.2.2 typedef unsigned int gazebo::common::Color::ARGB

10.39.2.3 typedef unsigned int gazebo::common::Color::BGRA

10.39.2.4 typedef unsigned int gazebo::common::Color::RGBA

10.39.3 Constructor & Destructor Documentation

10.39.3.1 gazebo::common::Color::Color ()

Constructor.

10.39.3.2 gazebo::common::Color::Color (float *_r*, float *_g*, float *_b*, float *_a* = 1.0)

Constructor.

Parameters

in	<i>_r</i>	Red value (range 0 to 1)
in	<i>_g</i>	Green value (range 0 to 1)
in	<i>_b</i>	Blue value (range 0 to 1)
in	<i>_a</i>	Alpha value (0=transparent, 1=opaque)

10.39.3.3 gazebo::common::Color::Color (const Color & *_clr*)

Copy Constructor.

Parameters

in	<i>_clr</i>	Color (p. 274) to copy
----	-------------	-------------------------------

10.39.3.4 virtual gazebo::common::Color::~~Color () [virtual]

Destructor.

10.39.4 Member Function Documentation

10.39.4.1 ABGR gazebo::common::Color::GetAsABGR () const

Get as uint32 ABGR packed value.

Returns

the color

10.39.4.2 ARGB gazebo::common::Color::GetAsARGB () const

Get as uint32 ARGB packed value.

Returns

the color

10.39.4.3 BGRA gazebo::common::Color::GetAsBGRA () const

Get as uint32 BGRA packed value.

Returns

the color

10.39.4.4 math::Vector3 gazebo::common::Color::GetAsHSV () const

Get the color in HSV colorspace.

Returns

HSV values in a **math::Vector3** (p. 902) format

10.39.4.5 RGBA gazebo::common::Color::GetAsRGBA () const

Get as uint32 RGBA packed value.

Returns

the color

10.39.4.6 math::Vector3 gazebo::common::Color::GetAsYUV () const

Get the color in YUV colorspace.

Returns

the YUV color

10.39.4.7 bool gazebo::common::Color::operator!=(const Color & _pt) const

Inequality operator.

Parameters

<i>in</i>	<i>_pt</i>	The color to check for inequality
-----------	------------	-----------------------------------

Returns

True if the this color does not equal *_pt*

10.39.4.8 const Color gazebo::common::Color::operator*(const Color & _pt) const

Multiplication operator.

Parameters

<i>in</i>	<i>_pt</i>	The color to multiply by
-----------	------------	--------------------------

Returns

The resulting color

10.39.4.9 `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

10.39.4.10 `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

10.39.4.11 `Color gazebo::common::Color::operator+(const Color & _pt) const`

Addition operator (this + `_pt`)

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 274) to add
-----------------	------------------	------------------------------

Returns

The resulting color

10.39.4.12 `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

10.39.4.13 `const Color& gazebo::common::Color::operator+=(const Color & _pt)`

Addition equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 274) to add
-----------------	------------------	------------------------------

Returns

The resulting color

10.39.4.14 `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

10.39.4.15 `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

10.39.4.16 `const Color& gazebo::common::Color::operator-= (const Color & _pt)`

Subtraction equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 274) to subtract
-----------------	------------------	-----------------------------------

Returns

The resulting color

10.39.4.17 `const Color gazebo::common::Color::operator/ (const Color & _pt) const`

Division operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 274) to divide by
-----------------	------------------	------------------------------------

Returns

The resulting color

10.39.4.18 `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

10.39.4.19 `const Color& gazebo::common::Color::operator/= (const Color & _pt)`

Division equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 274) to divide by
-----------------	------------------	------------------------------------

Returns

The resulting color

10.39.4.20 `Color& gazebo::common::Color::operator= (const Color & _pt)`

Equal operator.

Parameters

<code>in</code>	<code>_pt</code>	Color (p. 274) to copy
-----------------	------------------	-------------------------------

Returns

Reference to this color

10.39.4.21 `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`

10.39.4.22 `float gazebo::common::Color::operator[](unsigned int _index)`

Array index operator.

Parameters

<code>in</code>	<code>_index</code>	Color (p. 274) component index(0=red, 1=green, 2=blue)
-----------------	---------------------	---

Returns

r, g, b, or a when `_index` is 0, 1, 2 or 3

10.39.4.23 `void gazebo::common::Color::Reset ()`

Reset the color to default values.

10.39.4.24 `void gazebo::common::Color::Set (float _r = 1, float _g = 1, float _b = 1, float _a = 1)`

Set the contents of the vector.

Parameters

<code>in</code>	<code>_r</code>	Red value (range 0 to 1)
<code>in</code>	<code>_g</code>	Green value (range 0 to 1)
<code>in</code>	<code>_b</code>	Blue value (range 0 to 1)
<code>in</code>	<code>_a</code>	Alpha value (0=transparent, 1=opaque)

10.39.4.25 `void gazebo::common::Color::SetFromABGR (const ABGR _v)`

Set from uint32 ABGR packed value.

Parameters

in	_v	the new color
----	----	---------------

10.39.4.26 void gazebo::common::Color::SetFromARGB (const ARGB _v)

Set from uint32 ARGB packed value.

Parameters

in	_v	the new color
----	----	---------------

10.39.4.27 void gazebo::common::Color::SetFromBGRA (const BGRA _v)

Set from uint32 BGRA packed value.

Parameters

in	_v	the new color
----	----	---------------

10.39.4.28 void gazebo::common::Color::SetFromHSV (float _h, float _s, float _v)

Set a color based on HSV values.

Parameters

in	_h	Hue(0..360)
in	_s	Saturation(0..1)
in	_v	Value(0..1)

10.39.4.29 void gazebo::common::Color::SetFromRGBA (const RGBA _v)

Set from uint32 RGBA packed value.

Parameters

in	_v	the new color
----	----	---------------

10.39.4.30 void gazebo::common::Color::SetFromYUV (float _y, float _u, float _v)

Set from yuv.

Parameters

in	_y	value
in	_u	value
in	_v	value

10.39.5 Friends And Related Function Documentation

10.39.5.1 `std::ostream& operator<< (std::ostream & _out, const Color & _pt)` [*friend*]

Stream insertion operator.

Parameters

<code>in</code>	<code><i>_out</i></code>	the output stream
<code>in</code>	<code><i>_pt</i></code>	the color

Returns

the output stream

10.39.5.2 `std::istream& operator>> (std::istream & _in, Color & _pt)` [*friend*]

Stream insertion operator.

Parameters

<code>in</code>	<code><i>_in</i></code>	the input stream
<code>in</code>	<code><i>_pt</i></code>	

10.39.6 Member Data Documentation

10.39.6.1 `float gazebo::common::Color::a`

10.39.6.2 `float gazebo::common::Color::b`

10.39.6.3 `const Color gazebo::common::Color::Black` [*static*]

(0, 0, 0)

10.39.6.4 `const Color gazebo::common::Color::Blue` [*static*]

(0, 0, 1)

10.39.6.5 `float gazebo::common::Color::g`

10.39.6.6 `const Color gazebo::common::Color::Green` [*static*]

(0, 1, 0)

10.39.6.7 `const Color gazebo::common::Color::Purple` [*static*]

(1, 0, 1)

10.39.6.8 float gazebo::common::Color::r

10.39.6.9 const Color gazebo::common::Color::Red [static]

(1, 0, 0)

10.39.6.10 const Color gazebo::common::Color::White [static]

(1, 1, 1)

10.39.6.11 const Color gazebo::common::Color::Yellow [static]

(1, 1, 0)

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

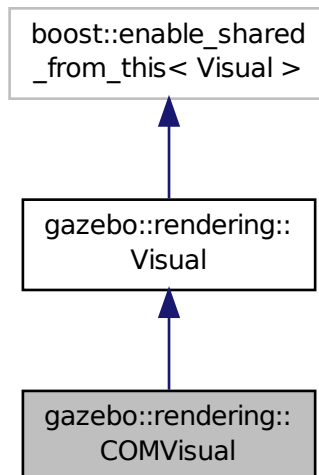
- **Color.hh**

10.40 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. 931) from an SDF pointer.*
- virtual void **Load** (ConstLinkPtr &_msg)
Load from a message.

Additional Inherited Members

10.40.1 Detailed Description

Basic Center of Mass visualization.

10.40.2 Constructor & Destructor Documentation

10.40.2.1 gazebo::rendering::COMVisual::COMVisual (const std::string & _name, VisualPtr _vis)

Constructor.

Parameters

in	<code>_name</code>	Name of the Visual (p. 931)
in	<code>_vis</code>	Parent Visual (p. 931)

10.40.2.2 virtual gazebo::rendering::COMVisual::~COMVisual () [virtual]

Destructor.

10.40.3 Member Function Documentation

10.40.3.1 virtual void gazebo::rendering::COMVisual::Load (sdf::ElementPtr _elem) [virtual]

Load the **Visual** (p. 931) from an SDF pointer.

Parameters

in	<code>_elem</code>	SDF Element pointer
----	--------------------	---------------------

10.40.3.2 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**

10.41 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.

10.41.1 Detailed Description

A class that encapsulates a connection.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 gazebo::event::Connection::Connection () [inline]

Constructor.

10.41.2.2 gazebo::event::Connection::Connection (**Event** * _e, int _i)

Constructor.

Parameters

in	<code>_e</code>	Event (p. 350) pointer to connect with
in	<code>_i</code>	Unique id

10.41.2.3 gazebo::event::Connection::~~Connection ()

Destructor.

10.41.3 Member Function Documentation

10.41.3.1 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:

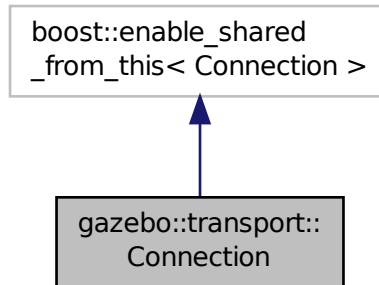
- **Event.hh**

10.42 gazebo::transport::Connection Class Reference

TCP/IP **Connection** (p. 288).

```
#include <Connection.hh>
```

Inheritance diagram for gazebo::transport::Connection:



Public Types

- typedef boost::function< void(const **ConnectionPtr** &)> **AcceptCallback**
- typedef boost::function< void(const std::string &data)> **ReadCallback**

Public Member Functions

- **Connection ()**
Constructor.

- virtual `~Connection ()`
Destructor.
- template<typename Handler >
void **AsyncRead** (Handler handler)
Perform and asynchronous read.
- 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_)
- void **DisconnectShutdown** (event::ConnectionPtr subscriber_)
- void **EnqueueMsg** (const std::string &_buffer, bool _force=false)
Write data to the socket.
- std::string **GetLocalAddress** () const
Get the 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
Return true if the connection is open.
- void **Listen** (unsigned int port, const **AcceptCallback** &accept_cb)
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.

Public Attributes

- unsigned int **id**
- unsigned int **writeCount**

10.42.1 Detailed Description

TCP/IP **Connection** (p. 288).

10.42.2 Member Typedef Documentation

10.42.2.1 typedef boost::function<void(const **ConnectionPtr**&)> gazebo::transport::Connection::AcceptCallback

10.42.2.2 typedef boost::function<void(const std::string &data)> gazebo::transport::Connection::ReadCallback

10.42.3 Constructor & Destructor Documentation

10.42.3.1 gazebo::transport::Connection::Connection ()

Constructor.

10.42.3.2 virtual gazebo::transport::Connection::~~Connection () [virtual]

Destructor.

10.42.4 Member Function Documentation

10.42.4.1 template<typename Handler > void gazebo::transport::Connection::AsyncRead (Handler *handler*) [inline]

Perform and asynchronous read.

References gzerr, HEADER_LENGTH, and IsOpen().

10.42.4.2 void gazebo::transport::Connection::Cancel ()

Cancel all async operations on an open socket.

10.42.4.3 bool gazebo::transport::Connection::Connect (const std::string & *host*, unsigned int *port*)

Connect to a remote host.

10.42.4.4 event::ConnectionPtr gazebo::transport::Connection::ConnectToShutdown (boost::function< void()> *subscriber_*) [inline]

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

10.42.4.5 void gazebo::transport::Connection::DisconnectShutdown (event::ConnectionPtr *subscriber_*) [inline]

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

10.42.4.6 void gazebo::transport::Connection::EnqueueMsg (const std::string & *_buffer*, bool *_force* = false)

Write data to the socket.

10.42.4.7 `std::string gazebo::transport::Connection::GetLocalAddress () const`

Get the address of this connection.

10.42.4.8 `std::string gazebo::transport::Connection::GetLocalHostname () const`

Get the local hostname.

10.42.4.9 `unsigned int gazebo::transport::Connection::GetLocalPort () const`

Get the port of this connection.

10.42.4.10 `std::string gazebo::transport::Connection::GetLocalURI () const`

Get the local URI.

10.42.4.11 `std::string gazebo::transport::Connection::GetRemoteAddress () const`

Get the remote address.

10.42.4.12 `std::string gazebo::transport::Connection::GetRemoteHostname () const`

Get the remote hostname.

10.42.4.13 `unsigned int gazebo::transport::Connection::GetRemotePort () const`

Get the remote port number.

10.42.4.14 `std::string gazebo::transport::Connection::GetRemoteURI () const`

Get the remote URI.

10.42.4.15 `bool gazebo::transport::Connection::IsOpen () const`

Return true if the connection is open.

Referenced by `AsyncRead()`.

10.42.4.16 `void gazebo::transport::Connection::Listen (unsigned int port, const AcceptCallback & accept_cb)`

Start a server that listens on a port.

10.42.4.17 `void gazebo::transport::Connection::ProcessWriteQueue ()`

Handle on write callbacks.

10.42.4.18 `bool gazebo::transport::Connection::Read (std::string & data)`

Read data from the socket.

10.42.4.19 `void gazebo::transport::Connection::Shutdown ()`

Shutdown the socket.

10.42.4.20 `void gazebo::transport::Connection::StartRead (const ReadCallback & cb)`

Start a thread that reads from the connection, and passes new message to the ReadCallback.

10.42.4.21 `void gazebo::transport::Connection::StopRead ()`

Stop the read loop.

10.42.5 Member Data Documentation

10.42.5.1 `unsigned int gazebo::transport::Connection::id`

10.42.5.2 `unsigned int gazebo::transport::Connection::writeCount`

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

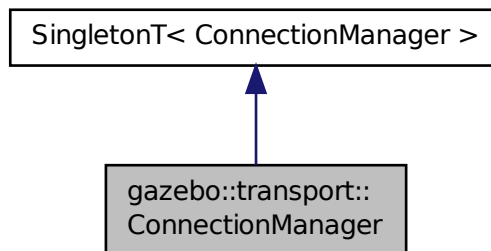
- **Connection.hh**

10.43 gazebo::transport::ConnectionManager Class Reference

Manager of connections.

```
#include <ConnectionManager.hh>
```

Inheritance diagram for gazebo::transport::ConnectionManager:



Public Member Functions

- void **Advertise** (const std::string &topic, const std::string &msgType)
- **ConnectionPtr ConnectToRemoteHost** (const std::string &host, unsigned int port)
 - Connect to a remote server.*
- void **Fini** ()
 - Finalize the connecton 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 &master_host, unsigned int master_port)
- bool **IsRunning** () const
 - Return true if running (not stopped)*
- void **RegisterTopicNamespace** (const std::string &_name)
 - Register a new topic namespace.*
- void **RemoveConnection** (**ConnectionPtr** &conn)
 - Remove a connection.*
- void **Run** ()
 - Run the connection manager loop.*
- void **RunUpdate** ()
- void **Stop** ()
 - Stop the connecton manager.*
- void **Subscribe** (const std::string &_topic, const std::string &_msgType, bool _latching)
- void **Unadvertise** (const std::string &topic)
- void **Unsubscribe** (const msgs::Subscribe &_sub)
- void **Unsubscribe** (const std::string &_topic, const std::string &_msgType)

Protected Attributes

- std::vector< **event::ConnectionPtr** > **eventConnections**

Additional Inherited Members

10.43.1 Detailed Description

Manager of connections.

10.43.2 Member Function Documentation

10.43.2.1 void gazebo::transport::ConnectionManager::Advertise (const std::string & *topic*, const std::string & *msgType*)

10.43.2.2 **ConnectionPtr** gazebo::transport::ConnectionManager::ConnectToRemoteHost (const std::string & *host*, unsigned int *port*)

Connect to a remote server.

10.43.2.3 void gazebo::transport::ConnectionManager::Fini ()

Finalize the connecton manager.

10.43.2.4 void gazebo::transport::ConnectionManager::GetAllPublishers (std::list< msgs::Publish > & *publishers*)

Explicitly update the publisher list.

10.43.2.5 void gazebo::transport::ConnectionManager::GetTopicNamespaces (std::list< std::string > & *_namespaces*)

Get all the topic namespaces.

10.43.2.6 bool gazebo::transport::ConnectionManager::Init (const std::string & *master_host*, unsigned int *master_port*)

10.43.2.7 bool gazebo::transport::ConnectionManager::IsRunning () const

Return true if running (not stopped)

10.43.2.8 void gazebo::transport::ConnectionManager::RegisterTopicNamespace (const std::string & *_name*)

Register a new topic namespace.

10.43.2.9 void gazebo::transport::ConnectionManager::RemoveConnection (ConnectionPtr & *conn*)

Remove a connection.

10.43.2.10 void gazebo::transport::ConnectionManager::Run ()

Run the connection manager loop.

10.43.2.11 void gazebo::transport::ConnectionManager::RunUpdate ()

10.43.2.12 void gazebo::transport::ConnectionManager::Stop ()

Stop the connecton manager.

10.43.2.13 void gazebo::transport::ConnectionManager::Subscribe (const std::string & *_topic*, const std::string & *_msgType*, bool *_latching*)

10.43.2.14 void gazebo::transport::ConnectionManager::Unadvertise (const std::string & *topic*)

10.43.2.15 void gazebo::transport::ConnectionManager::Unsubscribe (const msgs::Subscribe & *_sub*)

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

10.43.3 Member Data Documentation

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

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

- **ConnectionManager.hh**

10.44 gazebo::common::Console Class Reference

Message, error, warning, and logging functionality.

```
#include <Console.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.
- `std::ofstream & Log ()`
Use this to output a message to a log file.
- `void SetQuiet (bool _q)`
Set quiet output.

Static Public Member Functions

- `static Console * Instance ()`
Return an instance to this class.

10.44.1 Detailed Description

Message, error, warning, and logging functionality.

10.44.2 Member Function Documentation

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

Use this to output an error to the terminal.

Parameters

<code>in</code>	<code>_lbl</code>	Text label
<code>in</code>	<code>_file</code>	File containing the error
<code>in</code>	<code>_line</code>	Line containing the error
<code>in</code>	<code>_color</code>	Color (p. 274) to make the label

Returns

Reference to an output stream

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

Use this to output a colored message to the terminal.

Parameters

<code>in</code>	<code>_lbl</code>	Text label
<code>in</code>	<code>_color</code>	Color (p. 274) to make the label

Returns

Reference to an output stream

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

Return an instance to this class.

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

Load the message parameters.

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

Use this to output a message to a log file.

Returns

Reference to output stream

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

Set quiet output.

Parameters

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

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

- **Console.hh**

10.45 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
Clone the contact.
- **Contact & operator=** (const **Contact** &_contact)
Operator =.
- void **Reset** ()
Reset to default values.

Public Attributes

- **Collision * collision1**
Pointer to the first collision in the contact.
- **Collision * collision2**
Pointer to the second collision in the contact.
- int **count**
Length of all the arrays.
- double **depths** [32]
Array of contact depths.
- **JointFeedback forces** [32]
Array of forces for the contact.
- **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.

10.45.1 Detailed Description

A contact between two collisions.

Each contact can consist of a number of contact points

10.45.2 Constructor & Destructor Documentation

10.45.2.1 gazebo::physics::Contact::Contact ()

Constructor.

10.45.2.2 gazebo::physics::Contact::Contact (const Contact & _contact)

Copy constructor.

Parameters

in	_contact	Contact (p. 296) to copy.
----	----------	----------------------------------

10.45.2.3 virtual gazebo::physics::Contact::~~Contact () [virtual]

Destructor.

10.45.3 Member Function Documentation

10.45.3.1 Contact gazebo::physics::Contact::Clone () const

Clone the contact.

Returns

A cope of this contact.

10.45.3.2 Contact& gazebo::physics::Contact::operator= (const Contact & _contact)

Operator =.

Parameters

in	_contact	Contact (p. 296) to copy.
----	----------	----------------------------------

Returns

Reference to this contact

10.45.3.3 void gazebo::physics::Contact::Reset ()

Reset to default values.

10.45.4 Member Data Documentation

10.45.4.1 Collision* gazebo::physics::Contact::collision1

Pointer to the first collision in the contact.

10.45.4.2 Collision* gazebo::physics::Contact::collision2

Pointer to the second collision in the contact.

10.45.4.3 `int gazebo::physics::Contact::count`

Length of all the arrays.

10.45.4.4 `double gazebo::physics::Contact::depths[32]`

Array of contact depths.

10.45.4.5 **JointFeedback** `gazebo::physics::Contact::forces[32]`

Array of forces for the contact.

10.45.4.6 `math::Vector3 gazebo::physics::Contact::normals[32]`

Array of force normals.

10.45.4.7 `math::Vector3 gazebo::physics::Contact::positions[32]`

Array of force positions.

10.45.4.8 `common::Time gazebo::physics::Contact::time`

Time at which the contact occurred.

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

- **Contact.hh**

10.46 gazebo::physics::ContactFeedback Class Reference

data structure for contact feedbacks

```
#include <ODEPhysics.hh>
```

Public Attributes

- **Contact** `contact`
- `int` **feedbackCount**
- `dJointFeedback` **feedbacks** [**MAX_CONTACT_JOINTS**]

10.46.1 Detailed Description

data structure for contact feedbacks

10.46.2 Member Data Documentation

10.46.2.1 `Contact` gazebo::physics::ContactFeedback::contact

10.46.2.2 `int` gazebo::physics::ContactFeedback::feedbackCount

10.46.2.3 `dJointFeedback` gazebo::physics::ContactFeedback::feedbacks[MAX_CONTACT_JOINTS]

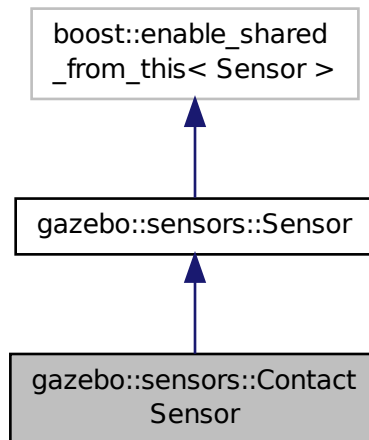
The documentation for this class was generated from the following file:

- **ODEPhysics.hh**

10.47 gazebo::sensors::ContactSensor Class Reference

```
#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
Get a contact for a collision by index.
- 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.
- boost::mutex * **GetUpdateMutex** () const
returns a pointer to the mutex for locking while reading internally kept map of map of collision names and contacts
- 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 **Fini** ()
Finalize the sensor.
- virtual void **UpdateImpl** (bool _force)
Update the sensor information.

Additional Inherited Members

10.47.1 Constructor & Destructor Documentation

10.47.1.1 gazebo::sensors::ContactSensor::ContactSensor ()

Constructor.

Parameters

<i>body</i>	The underlying collision test uses an ODE collision, so ray sensors must be attached to a body.
-------------	---

10.47.1.2 virtual gazebo::sensors::ContactSensor::~~ContactSensor () [virtual]

Destructor.

10.47.2 Member Function Documentation

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

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

10.47.2.2 `physics::Contact gazebo::sensors::ContactSensor::GetCollisionContact (const std::string & _collisionName, unsigned int _index) const`

Get a contact for a collision by index.

Parameters

<code>in</code>	<code><i>_collisionName</i></code>	Name of collision contact
<code>in</code>	<code><i>_index</i></code>	Index of collision to get

Returns

Contact object : nate check

10.47.2.3 `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><i>_collisionName</i></code>	The name of the observed collision
-----------------	------------------------------------	------------------------------------

Returns

The collision contact count

10.47.2.4 `unsigned int gazebo::sensors::ContactSensor::GetCollisionCount () const`

Get the number of collisions that the sensor is observing.

Returns

Number of collisions

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

Get a collision name at index `_index`.

Parameters

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

Returns

name of collision

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

Gets contacts of a collision.

Parameters

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

Returns

Container of contacts : nate check

10.47.2.7 `boost::mutex* gazebo::sensors::ContactSensor::GetUpdateMutex () const [inline]`

returns a pointer to the mutex for locking while reading internally kept map of map of collision names and contacts

Returns

The mutex for the sensor

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

Initialize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

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

Load the sensor with SDF parameters.

Parameters

in	<code>_sdf</code>	SDF Sensor (p. 765) parameters
in	<code>_worldName</code>	Name of world to load from

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

10.47.2.10 `virtual void gazebo::sensors::ContactSensor::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

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

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

10.47.2.11 `virtual void gazebo::sensors::ContactSensor::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. 771).

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

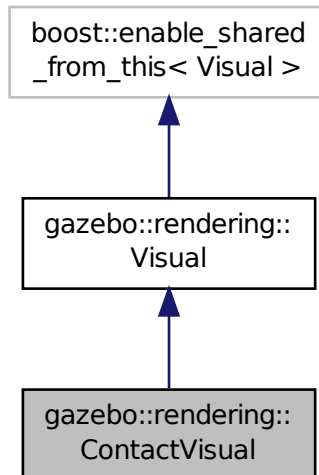
- `ContactSensor.hh`

10.48 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.

Additional Inherited Members

10.48.1 Detailed Description

Contact visualization.

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

10.48.2 Constructor & Destructor Documentation

10.48.2.1 `gazebo::rendering::ContactVisual::ContactVisual (const std::string & _name, VisualPtr _vis, const std::string & _topicName)`

Constructor.

Parameters

in	<i>_name</i>	Name of the ContactVisual (p. 304)
in	<i>_vis</i>	Pointer the parent Visual (p. 931)
in	<i>_topicName</i>	Name of the topic which publishes the contact information.

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

Destructor.

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

- **ContactVisual.hh**

10.49 gazebo::rendering::Conversions Class Reference

Conversions (p. 305) **Conversions.hh** (p. 1018) **rendering/Conversions.hh** (p. 1018).

```
#include <Conversions.hh>
```

Static Public Member Functions

- static `Ogre::ColourValue Convert (const common::Color &_clr)`
Return the equivalent ogre color.
- static `Ogre::Vector3 Convert (const math::Vector3 &_v)`
*return **Ogre** (p. 118) 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. 118) quaternion.*
- static `math::Quaternion Convert (const Ogre::Quaternion &_v)`
***Ogre** (p. 118) quaternion to Gazebo quaternion.*

10.49.1 Detailed Description

Conversions (p. 305) **Conversions.hh** (p. 1018) **rendering/Conversions.hh** (p. 1018).

A set of utility function to convert between Gazebo and **Ogre** (p. 118) data types

10.49.2 Member Function Documentation

10.49.2.1 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. 118) color value

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

return **Ogre** (p. 118) Vector from Gazebo Vector3

Parameters

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

Returns

Ogre (p. 118) vector

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

return gazebo Vector from ogre Vector3

Parameters

in	_v	Ogre (p. 118) vector
----	----	-----------------------------

Returns

Gazebo vector

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

Gazebo quaternion to **Ogre** (p. 118) quaternion.

Parameters

in	_v	Gazebo quaternion
----	----	-------------------

Returns

Ogre (p. 118) quaternion

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

Ogre (p. 118) quaternion to Gazebo quaternion.

Parameters

<code>in</code>	<code>_v</code>	Ogre (p. 118) quaternion return Gazebo quaternion
-----------------	-----------------	--

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

- **Conversions.hh**

10.50 sdf::Converter Class Reference

Convert from one version of **SDF** (p. 762) to another.

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

Static Public Member Functions

- static bool **Convert** (TiXmlElement **_elem*, const std::string &*_toVersion*, bool *_quiet*=false)

10.50.1 Detailed Description

Convert from one version of **SDF** (p. 762) to another.

10.50.2 Member Function Documentation

10.50.2.1 `static bool sdf::Converter::Convert (TiXmlElement * _elem, const std::string & _toVersion, bool quiet = false) [static]`

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

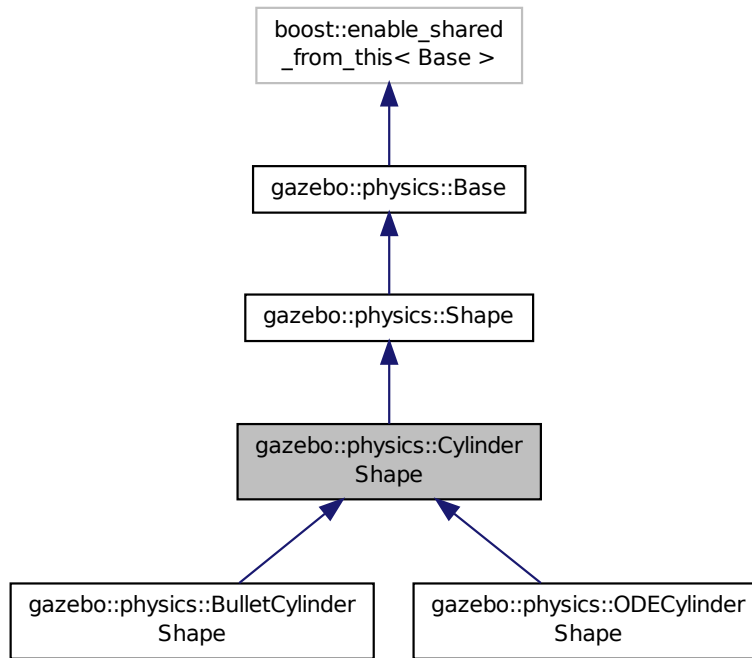
- **Converter.hh**

10.51 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
Get inertial for a shape.
- double **GetLength** () const
Get length.
- virtual double **GetMass** (double _density) const
Get the mass of the shape based on a density.
- 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

10.51.1 Detailed Description

Cylinder collision.

10.51.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

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

Destructor.

10.51.3 Member Function Documentation

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

Fill in the values for a geometry message.

Parameters

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

Implements gazebo::physics::Shape (p. 781).

10.51.3.2 virtual void gazebo::physics::CylinderShape::GetInertial (double *_mass*, InertialPtr *_inertial*) const [virtual]

Get inertial for a shape.

Parameters

in	<i>_mass</i>	Mass of the cylinder.
out	<i>_inertial</i>	Inertial (p. 414) element to populate with the result.

Reimplemented from gazebo::physics::Shape (p. 781).

10.51.3.3 `double gazebo::physics::CylinderShape::GetLength () const`

Get length.

Returns

The cylinder length.

10.51.3.4 `virtual double gazebo::physics::CylinderShape::GetMass (double _density) const` `[virtual]`

Get the mass of the shape based on a density.

Parameters

<code>in</code>	<code><i>_density</i></code>	Density of the cylinder.
-----------------	------------------------------	--------------------------

Reimplemented from `gazebo::physics::Shape` (p. 781).

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

Get radius.

Returns

The cylinder radius.

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

Initialize the cylinder.

Implements `gazebo::physics::Shape` (p. 781).

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

Update values based on a message.

Parameters

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

Implements `gazebo::physics::Shape` (p. 782).

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

Set length.

Parameters

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

10.51.3.9 void gazebo::physics::CylinderShape::SetRadius (double *_radius*)

Set radius.

Parameters

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

10.51.3.10 virtual void gazebo::physics::CylinderShape::SetSize (double *_radius*, double *_length*) [virtual]

Set the size of the cylinder.

Parameters

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

Reimplemented in **gazebo::physics::ODECylinderShape** (p. 584).

Referenced by gazebo::physics::ODECylinderShape::SetSize(), and gazebo::physics::BulletCylinderShape::SetSize().

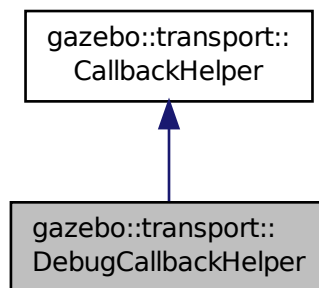
The documentation for this class was generated from the following file:

- **CylinderShape.hh**

10.52 gazebo::transport::DebugCallbackHelper Class Reference

```
#include <CallbackHelper.hh>
```

Inheritance diagram for gazebo::transport::DebugCallbackHelper:



Public Member Functions

- **DebugCallbackHelper** (const boost::function< void(ConstGzStringPtr &)> &cb)
- std::string **GetMsgType** () const

Get the typename of the message that is handled.

- virtual bool **HandleData** (const std::string &newdata)
- virtual bool **IsLocal** () const

Return true if the callback is local, false if the callback is tied to a remote connection.

Additional Inherited Members

10.52.1 Constructor & Destructor Documentation

10.52.1.1 gazebo::transport::DebugCallbackHelper::DebugCallbackHelper (const boost::function< void(ConstGzStringPtr &)> & *cb*) [inline]

10.52.2 Member Function Documentation

10.52.2.1 std::string gazebo::transport::DebugCallbackHelper::GetMsgType () const [inline],[virtual]

Get the typename of the message that is handled.

Reimplemented from **gazebo::transport::CallbackHelper** (p. 231).

10.52.2.2 virtual bool gazebo::transport::DebugCallbackHelper::HandleData (const std::string & *newdata*) [inline],[virtual]

Implements **gazebo::transport::CallbackHelper** (p. 231).

10.52.2.3 virtual bool gazebo::transport::DebugCallbackHelper::IsLocal () const [inline],[virtual]

Return true if the callback is local, false if the callback is tied to a remote connection.

Implements **gazebo::transport::CallbackHelper** (p. 231).

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

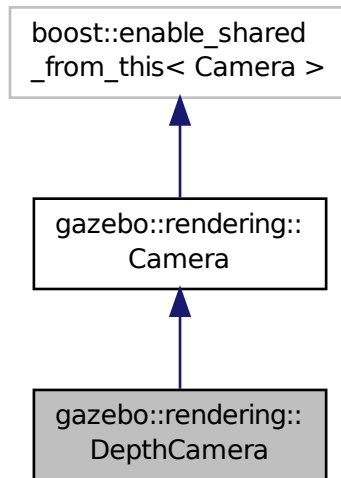
- **CallbackHelper.hh**

10.53 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, **Scene** *_scene, bool _autoRender=true)
Constructor.
- virtual **~DepthCamera** ()
Destructor.
- template<typename T >
event::ConnectionPtr ConnectNewDepthFrame (T _subscriber)
Connect a to the new depth image signal.
- template<typename T >
event::ConnectionPtr ConnectNewRGBPointCloud (T subscriber)
Connect a to the new rgb point cloud signal.
- void **CreateDepthTexture** (const std::string &_textureName)
Create a texture which will hold the depth data.
- void **DisconnectNewDepthFrame** (event::ConnectionPtr &_c)
Disconnect from an depth image singal.
- void **DisconnectNewRGBPointCloud** (event::ConnectionPtr &c)
Disconnect from an rgb point cloud singal.
- void **Fini** ()
Finalize the camera.
- virtual const float * **GetDepthData** ()
All things needed to get back z buffer for depth data.
- void **Init** ()
Initialize the camera.
- void **Load** (sdf::ElementPtr &_sdf)

Load the camera with a set of parameters.

- void **Load** ()

Load the camera with default parameters.

- virtual void **PostRender** ()

Render the camera.

- virtual void **SetDepthTarget** (Ogre::RenderTarget *_target)

Set the render target, which renders the depth data.

Protected Attributes

- Ogre::RenderTarget * **depthTarget**

Pointer to the depth target.

- Ogre::Texture * **depthTexture**

Pointer to the depth texture.

- Ogre::Viewport * **depthViewport**

Pointer to the depth viewport.

Additional Inherited Members

10.53.1 Detailed Description

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

10.53.2 Constructor & Destructor Documentation

10.53.2.1 gazebo::rendering::DepthCamera::DepthCamera (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. 746) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

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

Destructor.

10.53.3 Member Function Documentation

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

Connect a to the new depth image signal.

Parameters

in	<i>_subscriber</i>	Subscriber callback function
----	--------------------	------------------------------

Returns

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

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

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

Connect a to the new rgb point cloud signal.

Parameters

in	<i>_subscriber</i>	Subscriber callback function
----	--------------------	------------------------------

Returns

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

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

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

Create a texture which will hold the depth data.

Parameters

in	<i>_textureName</i>	Name of the texture to create
----	---------------------	-------------------------------

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

Disconnect from an depth image singal.

Parameters

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

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

10.53.3.5 `void gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud (event::ConnectionPtr & _c) [inline]`

Disconnect from an rgb point cloud singal.

Parameters

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

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

10.53.3.6 void gazebo::rendering::DepthCamera::Fini ()

Finalize the camera.

10.53.3.7 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

10.53.3.8 void gazebo::rendering::DepthCamera::Init ()

Initialize the camera.

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

Load the camera with a set of parameters.

Parameters

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

10.53.3.10 void gazebo::rendering::DepthCamera::Load ()

Load the camera with default parameters.

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

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 248).

10.53.3.12 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
----	----------------	------------------------------

10.53.4 Member Data Documentation

10.53.4.1 `Ogre::RenderTarget*` `gazebo::rendering::DepthCamera::depthTarget` [protected]

Pointer to the depth target.

10.53.4.2 `Ogre::Texture*` `gazebo::rendering::DepthCamera::depthTexture` [protected]

Pointer to the depth texture.

10.53.4.3 `Ogre::Viewport*` `gazebo::rendering::DepthCamera::depthViewport` [protected]

Pointer to the depth viewport.

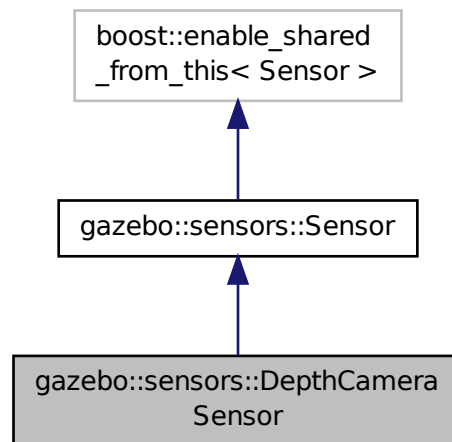
The documentation for this class was generated from the following file:

- `DepthCamera.hh`

10.54 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. 312).*
- 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

10.54.1 Constructor & Destructor Documentation

10.54.1.1 gazebo::sensors::DepthCameraSensor::DepthCameraSensor ()

Constructor.

10.54.1.2 virtual gazebo::sensors::DepthCameraSensor::~~DepthCameraSensor () [virtual]

Destructor.

10.54.2 Member Function Documentation

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

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

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

Returns a pointer to the **rendering::DepthCamera** (p. 312).

Returns

Depth Camera pointer

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

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.54.2.4 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. 765) parameters
in	<i>_worldName</i>	Name of world to load from

10.54.2.5 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. 770).

10.54.2.6 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

10.54.2.7 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. 770).

10.54.2.8 `virtual void gazebo::sensors::DepthCameraSensor::SetParent (const std::string & _name) [virtual]`

Set the parent of the sensor.

Parameters

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

Reimplemented from `gazebo::sensors::Sensor` (p. 771).

10.54.2.9 `virtual void gazebo::sensors::DepthCameraSensor::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. 771).

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

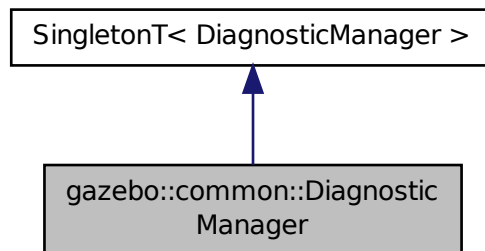
- `DepthCameraSensor.hh`

10.55 gazebo::common::DiagnosticManager Class Reference

A diagnostic manager class.

```
#include <Diagnostics.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

10.55.1 Detailed Description

A diagnostic manager class.

10.55.2 Member Function Documentation

10.55.2.1 DiagnosticTimerPtr gazebo::common::DiagnosticManager::CreateTimer (const std::string & _name)

Create a new timer instance.

Parameters

in	_name	Name of the timer.
----	-------	--------------------

Returns

A pointer to the new diagnostic timer

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

Get whether the timers are enabled.

Returns

TRue if the timers are enabled

10.55.2.3 std::string gazebo::common::DiagnosticManager::GetLabel (int _index) const

Get a label for a timer.

Parameters

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

Returns

Label of the specified timer

10.55.2.4 `Time gazebo::common::DiagnosticManager::GetTime (int _index) const`

Get the time of a timer instance.

Parameters

<code>in</code>	<code>_index</code>	The index of a timer instance
-----------------	---------------------	-------------------------------

Returns

Time (p. 840) of the specified timer

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

Get a time based on a label.

Parameters

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

Returns

Time (p. 840) of the specified timer

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

Get the number of timers.

Returns

The number of timers

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

Set whether timers are enabled.

Parameters

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

10.55.2.8 void gazebo::common::DiagnosticManager::TimerStart (DiagnosticTimer * *_timer*)

A diagnostic timer has started.

Parameters

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

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

10.55.2.9 void gazebo::common::DiagnosticManager::TimerStop (DiagnosticTimer * *_timer*)

A diagnostic timer has stopped.

Parameters

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

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

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

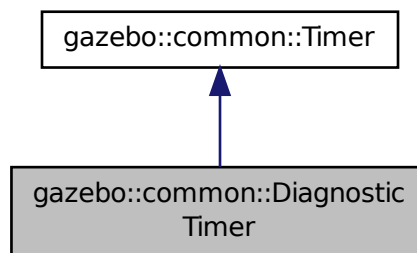
- **Diagnostics.hh**

10.56 gazebo::common::DiagnosticTimer Class Reference

A timer designed for diagnostics.

```
#include <Diagnostics.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.

10.56.1 Detailed Description

A timer designed for diagnostics.

10.56.2 Constructor & Destructor Documentation

10.56.2.1 `gazebo::common::DiagnosticTimer::DiagnosticTimer (const std::string & _name) [inline]`

Constructor.

Parameters

in	_name	Name of the timer
----	-------	-------------------

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

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

Destructor.

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

10.56.3 Member Function Documentation

10.56.3.1 `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:

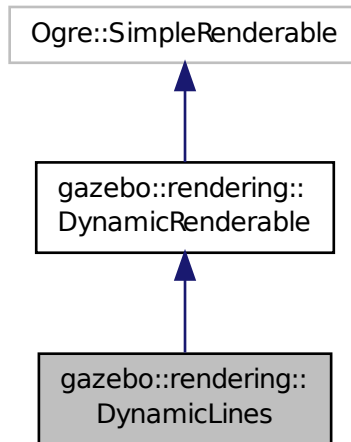
- **Diagnostics.hh**

10.57 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 **Clear** ()
Remove all points from the point list.
- virtual const **Ogre::String** & **getMovableType** () const
*Overridden function from **Ogre** (p. 118)'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. 328), creates a simple vertex-only decl.
- virtual void **FillHardwareBuffers** ()
Implementation *DynamicRenderable* (p. 328), pushes point list out to hardware memory.

Additional Inherited Members

10.57.1 Detailed Description

Class for drawing lines that can change.

10.57.2 Constructor & Destructor Documentation

10.57.2.1 gazebo::rendering::DynamicLines::DynamicLines (RenderOpType *_opType* = RENDERING_LINE_STRIP)

Constructor.

Parameters

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

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

Destructor.

10.57.3 Member Function Documentation

10.57.3.1 void gazebo::rendering::DynamicLines::AddPoint (const math::Vector3 & *_pt*)

Add a point to the point list.

Parameters

in	<i>pt</i>	math::Vector3 (p. 902) point
----	-----------	-------------------------------------

10.57.3.2 void gazebo::rendering::DynamicLines::Clear ()

Remove all points from the point list.

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

Implementation *DynamicRenderable* (p. 328), creates a simple vertex-only decl.

Implements **gazebo::rendering::DynamicRenderable** (p. 329).

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

Implementation **DynamicRenderable** (p. 328), pushes point list out to hardware memory.

Implements **gazebo::rendering::DynamicRenderable** (p. 330).

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

Get type of movable.

Returns

This returns "gazebo::dynamiclines"

10.57.3.6 virtual const Ogre::String& gazebo::rendering::DynamicLines::getMovableType () const [virtual]

Overridden function from **Ogre** (p. 118)'s base class.

Returns

Returns "gazebo::ogredynamicslines"

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

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

Parameters

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

Returns

math::Vector3 (p. 902) value of the point

10.57.3.8 unsigned int gazebo::rendering::DynamicLines::GetPointCount () const

Return the total number of points in the point list.

Returns

Number of points

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

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

Parameters

<i>in</i>	<i>_index</i>	Index of the point to set
<i>in</i>	<i>_value</i>	math::Vector3 (p. 902) value to set the point to

10.57.3.10 void gazebo::rendering::DynamicLines::Update ()

Call this to update the hardware buffer after making changes.

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

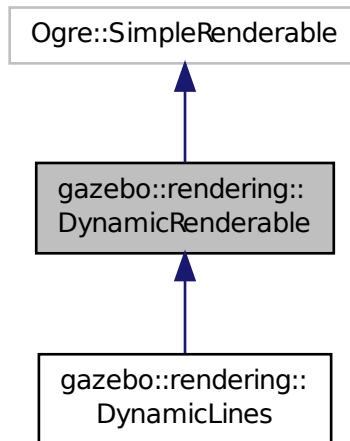
- **DynamicLines.hh**

10.58 gazebo::rendering::DynamicRenderable Class Reference

Abstract base class providing mechanisms for dynamically growing hardware buffers.

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

Inheritance diagram for gazebo::rendering::DynamicRenderable:



Public Member Functions

- **DynamicRenderable** ()
Constructor.
- virtual **~DynamicRenderable** ()
Virtual destructor.
- virtual Ogre::Real **getBoundingRadius** () const
Implementation of Ogre::SimpleRenderable.
- **RenderOpType GetOperationType** () const
Get the render operation type.
- virtual Ogre::Real **getSquaredViewDepth** (const Ogre::Camera *_cam) const
Implementation of Ogre::SimpleRenderable.
- void **Init** (**RenderOpType** _opType, bool _useIndices=false)

Initializes the dynamic renderable.

- void **SetOperationType** (**RenderOpType** _opType)
Set the render operation type.

Protected Member Functions

- virtual void **CreateVertexDeclaration** ()=0
Creates the vertex declaration.
- virtual void **FillHardwareBuffers** ()=0
Fills the hardware vertex and index buffers with data.
- void **PrepareHardwareBuffers** (size_t _vertexCount, size_t _indexCount)
Prepares the hardware buffers for the requested vertex and index counts.

Protected Attributes

- size_t **indexBufferCapacity**
Maximum capacity of the currently allocated index buffer.
- size_t **vertexBufferCapacity**
Maximum capacity of the currently allocated vertex buffer.

10.58.1 Detailed Description

Abstract base class providing mechanisms for dynamically growing hardware buffers.

10.58.2 Constructor & Destructor Documentation

10.58.2.1 gazebo::rendering::DynamicRenderable::DynamicRenderable ()

Constructor.

10.58.2.2 virtual gazebo::rendering::DynamicRenderable::~~DynamicRenderable () [virtual]

Virtual destructor.

10.58.3 Member Function Documentation

10.58.3.1 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. 326).

10.58.3.2 `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 the 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. 327).

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

Implementation of `Ogre::SimpleRenderable`.

Returns

The bounding radius

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

Get the render operation type.

Returns

The render operation type.

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

Implementation of `Ogre::SimpleRenderable`.

Parameters

in	_cam	Pointer to the Ogre (p. 118) camera that views the renderable.
----	------	---

Returns

The squared depth in the **Camera** (p. 233)'s view

10.58.3.6 `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. 329).

Parameters

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

10.58.3.7 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.

10.58.3.8 void gazebo::rendering::DynamicRenderable::SetOperationType (RenderOpType *opType*)

Set the render operation type.

Parameters

in	<code>_opType</code>	The type of render operation to perform.
----	----------------------	--

10.58.4 Member Data Documentation

10.58.4.1 size_t gazebo::rendering::DynamicRenderable::indexBufferCapacity [protected]

Maximum capacity of the currently allocated index buffer.

10.58.4.2 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:

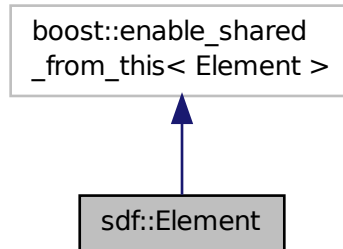
- **DynamicRenderable.hh**

10.59 sdf::Element Class Reference

SDF (p. 762) **Element** (p. 332) 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. 332).*
- **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 description 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. 763).
- void **PrintDocRightPane** (std::string &_html, int _spacing)
Helper function for SDF::PrintDoc (p. 763).
- 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** ()

10.59.1 Detailed Description

SDF (p. 762) **Element** (p. 332) class.

10.59.2 Constructor & Destructor Documentation

10.59.2.1 sdf::Element::Element ()

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

10.59.3 Member Function Documentation

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

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

10.59.3.3 void sdf::Element::AddElementDescription (ElementPtr _elem)

Add a new element description.

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

10.59.3.5 void sdf::Element::ClearElements ()

10.59.3.6 Element* sdf::Element::Clone () const

10.59.3.7 void sdf::Element::Copy (const ElementPtr _elem)

Copy values from an **Element** (p. 332).

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

Get the param of an attribute.

Parameters

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

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

Get an attribute using an index.

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

Get the number of attributes.

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

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

10.59.3.12 bool sdf::Element::GetCopyChildren () const**10.59.3.13 std::string sdf::Element::GetDescription () const**

Get a text description of the element.

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

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

10.59.3.15 ElementPtr sdf::Element::GetElement (const std::string & *_name*)**10.59.3.16 ElementPtr sdf::Element::GetElementDescription (unsigned int *_index*) const**

Get an element description using an index.

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

Get an element descriptio using a key.

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

Get the number of element descriptions.

10.59.3.19 `ElementPtr sdf::Element::GetElementImpl (const std::string & _name) const`

10.59.3.20 `ElementPtr sdf::Element::GetFirstElement () const`

10.59.3.21 `std::string sdf::Element::GetInclude () const`

10.59.3.22 `const std::string& sdf::Element::GetName () const`

10.59.3.23 `ElementPtr sdf::Element::GetNextElement (const std::string & _name = " ") const`

10.59.3.24 `ElementPtr sdf::Element::GetParent () const`

10.59.3.25 `const std::string& sdf::Element::GetRequired () const`

10.59.3.26 `ParamPtr sdf::Element::GetValue ()`

Get the param of the elements value.

10.59.3.27 `bool sdf::Element::GetValueBool (const std::string & _key = " ")`

10.59.3.28 `char sdf::Element::GetValueChar (const std::string & _key = " ")`

10.59.3.29 `gazebo::common::Color sdf::Element::GetValueColor (const std::string & _key = " ")`

10.59.3.30 `double sdf::Element::GetValueDouble (const std::string & _key = " ")`

Referenced by `gazebo::physics::ScrewJoint< BulletJoint >::Load()`.

10.59.3.31 `float sdf::Element::GetValueFloat (const std::string & _key = " ")`

10.59.3.32 `int sdf::Element::GetValueInt (const std::string & _key = " ")`

10.59.3.33 `gazebo::math::Pose sdf::Element::GetValuePose (const std::string & _key = " ")`

10.59.3.34 `gazebo::math::Quaternion sdf::Element::GetValueQuaternion (const std::string & _key = " ")`

10.59.3.35 `std::string sdf::Element::GetValueString (const std::string & _key = " ")`

10.59.3.36 `gazebo::common::Time sdf::Element::GetValueTime (const std::string & _key = " ")`

10.59.3.37 `unsigned int sdf::Element::GetValueUInt (const std::string & _key = " ")`

10.59.3.38 `gazebo::math::Vector2d sdf::Element::GetValueVector2d (const std::string & _key = " ")`

10.59.3.39 `gazebo::math::Vector3 sdf::Element::GetValueVector3 (const std::string & _key = " ")`

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

10.59.3.40 `bool sdf::Element::HasAttribute (const std::string & _key)`

10.59.3.41 `bool sdf::Element::HasElement (const std::string & _name) const`

Referenced by `gazebo::physics::ScrewJoint< BulletJoint >::Load()`.

10.59.3.42 `bool sdf::Element::HasElementDescription (const std::string & _name)`

Return true if an element description exists.

10.59.3.43 `void sdf::Element::InsertElement (ElementPtr _elem)`

10.59.3.44 `void sdf::Element::PrintDescription (std::string _prefix)`

10.59.3.45 `void sdf::Element::PrintDocLeftPane (std::string & _html, int _spacing, int & _index)`

Helper function for **SDF::PrintDoc** (p. 763).

This generates the **SDF** (p. 762) html documentation.

Parameters

out	<i>_html</i>	Accumulated HTML for output.
in	<i>_spacing</i>	Amount of spacing for this element.
in	<i>_index</i>	Unique index for this element.

10.59.3.46 `void sdf::Element::PrintDocRightPane (std::string & _html, int _spacing)`

Helper function for **SDF::PrintDoc** (p. 763).

This generates the **SDF** (p. 762) html documentation.

Parameters

out	<i>_html</i>	Accumulated HTML for output
in	<i>_spacing</i>	Amount of spacing for this element.

10.59.3.47 `void sdf::Element::PrintValues (std::string _prefix)`

10.59.3.48 `void sdf::Element::PrintWiki (std::string _prefix)`

10.59.3.49 `void sdf::Element::Reset ()`

10.59.3.50 `bool sdf::Element::Set (const bool & _value)`

10.59.3.51 `bool sdf::Element::Set (const int & _value)`

10.59.3.52 `bool sdf::Element::Set (const unsigned int & _value)`

10.59.3.53 `bool sdf::Element::Set (const float & _value)`

10.59.3.54 `bool sdf::Element::Set (const double & _value)`

- 10.59.3.55 `bool sdf::Element::Set (const char & _value)`
- 10.59.3.56 `bool sdf::Element::Set (const std::string & _value)`
- 10.59.3.57 `bool sdf::Element::Set (const char * _value)`
- 10.59.3.58 `bool sdf::Element::Set (const gazebo::math::Vector3 & _value)`
- 10.59.3.59 `bool sdf::Element::Set (const gazebo::math::Vector2i & _value)`
- 10.59.3.60 `bool sdf::Element::Set (const gazebo::math::Vector2d & _value)`
- 10.59.3.61 `bool sdf::Element::Set (const gazebo::math::Quaternion & _value)`
- 10.59.3.62 `bool sdf::Element::Set (const gazebo::math::Pose & _value)`
- 10.59.3.63 `bool sdf::Element::Set (const gazebo::common::Color & _value)`
- 10.59.3.64 `bool sdf::Element::Set (const gazebo::common::Time & _value)`
- 10.59.3.65 `void sdf::Element::SetCopyChildren (bool _value)`
- 10.59.3.66 `void sdf::Element::SetDescription (const std::string & _desc)`

Set a text description for the element.

- 10.59.3.67 `void sdf::Element::SetInclude (const std::string & _filename)`
- 10.59.3.68 `void sdf::Element::SetName (const std::string & _name)`
- 10.59.3.69 `void sdf::Element::SetParent (const ElementPtr _parent)`
- 10.59.3.70 `void sdf::Element::SetRequired (const std::string & _req)`
- 10.59.3.71 `std::string sdf::Element::ToString (const std::string & _prefix) const`
- 10.59.3.72 `void sdf::Element::Update ()`

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

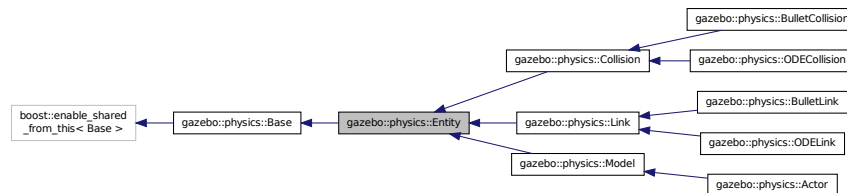
- **SDF.hh**

10.60 gazebo::physics::Entity Class Reference

Base (p. 145) 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. 349).*
- void **GetNearestEntityBelow** (double &_distBelow, std::string &_entityName)
Get the distance to the nearest entity below (along the Z-axis) this entity.
- **ModelPtr** **GetParentModel** ()
Get the parent model, if one exists.
- virtual **math::Vector3** **GetRelativeAngularAccel** () const
Get the angular acceleration of the entity.
- virtual **math::Vector3** **GetRelativeAngularVel** () const
Get the angular velocity of the entity.
- virtual **math::Vector3** **GetRelativeLinearAccel** () const
Get the linear acceleration of the entity.
- virtual **math::Vector3** **GetRelativeLinearVel** () const
Get the linear velocity of the entity.
- **math::Pose** **GetRelativePose** () const
Get the pose of the entity relative to its parent.
- virtual **math::Vector3** **GetWorldAngularAccel** () const
Get the angular acceleration of the entity in the world frame.
- virtual **math::Vector3** **GetWorldAngularVel** () const
Get the angular velocity of the entity in the world frame.

- virtual **math::Vector3 GetWorldLinearAccel** () const
Get the linear acceleration of the entity in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** () const
Get the linear velocity of the entity in the world frame.
- const **math::Pose & GetWorldPose** () const
Get the absolute pose of the entity.
- bool **IsCanonicalLink** () const
A helper function that checks if this is a canonical body.
- bool **IsStatic** () const
Return whether this entity is static.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the entity.
- void **PlaceOnEntity** (const std::string &_entityName)
Move this entity to be ontop of another entity by name.
- void **PlaceOnNearestEntityBelow** ()
Move this entity to be ontop of the nearest entity below.
- virtual void **Reset** ()
Reset the entity.
- void **SetAnimation** (const common::PoseAnimationPtr &_anim, boost::function< void()> _onComplete)
Set an animation for this entity.
- void **SetAnimation** (common::PoseAnimationPtr _anim)
Set an animation for this entity.
- void **SetCanonicalLink** (bool _value)
Set to true if this entity is a canonical link for a model.
- void **SetInitialRelativePose** (const math::Pose &_pose)
Set the initial pose.
- virtual void **SetName** (const std::string &_name)
Set the name of the entity.
- void **SetRelativePose** (const math::Pose &_pose, bool _notify=true, bool _publish=true)
Set the pose of the entity relative to its parent.
- void **SetStatic** (const bool &_static)
Set whether this entity is static: immovable.
- void **SetWorldPose** (const math::Pose &_pose, bool _notify=true, bool _publish=true)
Set the world pose of the entity.
- void **SetWorldTwist** (const math::Vector3 &_linear, const math::Vector3 &_angular, bool _updateChildren=true)
*Set angular and linear rates of an **physics::Entity** (p. 338).*
- 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

10.60.1 Detailed Description

Base (p. 145) class for all physics objects in Gazebo.

10.60.2 Constructor & Destructor Documentation

10.60.2.1 `gazebo::physics::Entity::Entity(BasePtr _parent) [explicit]`

Constructor.

Parameters

in	<code>_parent</code>	Parent of the entity.
----	----------------------	-----------------------

10.60.2.2 `virtual gazebo::physics::Entity::~~Entity() [virtual]`

Destructor.

10.60.3 Member Function Documentation

10.60.3.1 virtual void gazebo::physics::Entity::Fini () [virtual]

Finalize the entity.

Reimplemented from **gazebo::physics::Base** (p. 150).

Reimplemented in **gazebo::physics::Actor** (p. 124), **gazebo::physics::Model** (p. 525), **gazebo::physics::Link** (p. 462), **gazebo::physics::ODECollision** (p. 580), **gazebo::physics::BulletLink** (p. 195), **gazebo::physics::ODELink** (p. 603), and **gazebo::physics::Collision** (p. 266).

10.60.3.2 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. 462), **gazebo::physics::Model** (p. 525), **gazebo::physics::Collision** (p. 266), **gazebo::physics::ODECollision** (p. 580), and **gazebo::physics::BulletCollision** (p. 173).

10.60.3.3 CollisionPtr gazebo::physics::Entity::GetChildCollision (const std::string & _name)

Get a child collision entity, if one exists.

Parameters

in	_name	Name of the child collision object.
----	-------	-------------------------------------

Returns

Pointer to the **Collision** (p. 262) object, or NULL if not found.

10.60.3.4 LinkPtr gazebo::physics::Entity::GetChildLink (const std::string & _name)

Get a child linke entity, if one exists.

Parameters

in	_name	Name of the child Link (p. 454) object.
----	-------	--

Returns

Pointer to the **Link** (p. 454) object, or NULL if not found.

10.60.3.5 math::Box gazebo::physics::Entity::GetCollisionBoundingBox () const

Returns collision bounding box.

Returns

Collision bounding box.

10.60.3.6 `const math::Pose& gazebo::physics::Entity::GetDirtyPose () const`

Returns **Entity::dirtyPose** (p. 349).

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.

10.60.3.7 `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.

10.60.3.8 `ModelPtr gazebo::physics::Entity::GetParentModel ()`

Get the parent model, if one exists.

Returns

Pointer to a model, or NULL if no parent model exists.

10.60.3.9 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularAccel () const [inline],[virtual]`

Get the angular acceleration of the entity.

Returns

A **math::Vector3** (p. 902) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 465), **gazebo::physics::Collision** (p. 267), and **gazebo::physics::Model** (p. 527).

10.60.3.10 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularVel () const [inline],[virtual]`

Get the angular velocity of the entity.

Returns

A **math::Vector3** (p. 902) for the velocity.

Reimplemented in **gazebo::physics::Link** (p. 465), **gazebo::physics::Collision** (p. 268), and **gazebo::physics::Model** (p. 527).

10.60.3.11 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearAccel () const [inline],[virtual]`

Get the linear acceleration of the entity.

Returns

A `math::Vector3` (p. 902) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 465), `gazebo::physics::Collision` (p. 268), and `gazebo::physics::Model` (p. 527).

10.60.3.12 `virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearVel () const [inline],[virtual]`

Get the linear velocity of the entity.

Returns

A `math::Vector3` (p. 902) for the linear velocity.

Reimplemented in `gazebo::physics::Link` (p. 465), `gazebo::physics::Collision` (p. 268), and `gazebo::physics::Model` (p. 527).

10.60.3.13 `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.

10.60.3.14 `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. 902) for the acceleration.

Reimplemented in `gazebo::physics::Link` (p. 467), `gazebo::physics::Collision` (p. 269), and `gazebo::physics::Model` (p. 528).

10.60.3.15 `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. 902) for the velocity.

Reimplemented in `gazebo::physics::Collision` (p. 269), `gazebo::physics::Model` (p. 528), `gazebo::physics::ODE-Link` (p. 604), and `gazebo::physics::BulletLink` (p. 195).

10.60.3.16 `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. 902) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 467), **gazebo::physics::Collision** (p. 269), and **gazebo::physics::Model** (p. 528).

10.60.3.17 `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. 902) for the linear velocity.

Reimplemented in **gazebo::physics::Collision** (p. 269), **gazebo::physics::Model** (p. 529), **gazebo::physics::ODE-Link** (p. 604), and **gazebo::physics::BulletLink** (p. 196).

10.60.3.18 `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()`.

10.60.3.19 `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.

10.60.3.20 `bool gazebo::physics::Entity::IsStatic () const`

Return whether this entity is static.

Returns

True if static.

10.60.3.21 `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. 154).

Reimplemented in `gazebo::physics::Actor` (p. 124), `gazebo::physics::Model` (p. 529), `gazebo::physics::Link` (p. 468), `gazebo::physics::BulletCollision` (p. 173), `gazebo::physics::Collision` (p. 270), `gazebo::physics::BulletLink` (p. 196), `gazebo::physics::ODECollision` (p. 581), and `gazebo::physics::ODELink` (p. 605).

10.60.3.22 `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. 468), `gazebo::physics::Model` (p. 529), `gazebo::physics::ODECollision` (p. 581), `gazebo::physics::BulletLink` (p. 196), `gazebo::physics::BulletCollision` (p. 173), and `gazebo::physics::ODELink` (p. 605).

10.60.3.23 `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. 338) this Entity (p. 338) should be ontop of.
-----------------	--------------------------	--

10.60.3.24 `void gazebo::physics::Entity::PlaceOnNearestEntityBelow ()`

Move this entity to be ontop of the nearest entity below.

10.60.3.25 `virtual void gazebo::physics::Entity::Reset () [virtual]`

Reset the entity.

Reimplemented from `gazebo::physics::Base` (p. 155).

Reimplemented in `gazebo::physics::Model` (p. 530), and `gazebo::physics::Link` (p. 468).

10.60.3.26 `void gazebo::physics::Entity::SetAnimation (const common::PoseAnimationPtr & _anim, boost::function< void()> _onComplete)`

Set an animation for this entity.

Parameters

<code>in</code>	<code>_anim</code>	Pose animation.
<code>in</code>	<code>_onComplete</code>	Callback for when the animation completes.

10.60.3.27 void gazebo::physics::Entity::SetAnimation (common::PoseAnimationPtr *_anim*)

Set an animation for this entity.

Parameters

in	<i>_anim</i>	Pose animation.
----	--------------	-----------------

10.60.3.28 void gazebo::physics::Entity::SetCanonicalLink (bool *_value*)

Set to true if this entity is a canonical link for a model.

Parameters

in	<i>_value</i>	True if the link is canonical.
----	---------------	--------------------------------

10.60.3.29 void gazebo::physics::Entity::SetInitialRelativePose (const math::Pose & *_pose*)

Set the initial pose.

Parameters

in	<i>_pose</i>	The initial pose.
----	--------------	-------------------

10.60.3.30 virtual void gazebo::physics::Entity::SetName (const std::string & *_name*) [virtual]

Set the name of the entity.

Parameters

in	<i>_name</i>	The new name.
----	--------------	---------------

Reimplemented from **gazebo::physics::Base** (p. 156).

10.60.3.31 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.

10.60.3.32 void gazebo::physics::Entity::SetStatic (const bool & *_static*)

Set whether this entity is static: immovable.

Parameters

<code>in</code>	<code>_static</code>	True = static.
-----------------	----------------------	----------------

10.60.3.33 `void gazebo::physics::Entity::SetWorldPose (const math::Pose & _pose, bool _notify = true, bool _publish = true)`

Set the world pose of the entity.

Parameters

<code>in</code>	<code>_pose</code>	The new world pose.
<code>in</code>	<code>_notify</code>	True = tell children of the pose change.
<code>in</code>	<code>_publish</code>	True to publish the pose.

10.60.3.34 `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. 338).

Parameters

<code>in</code>	<code>_linear</code>	Linear twist.
<code>in</code>	<code>_angular</code>	Angular twist.
<code>in</code>	<code>_updateChildren</code>	True to pass this update to child entities.

10.60.3.35 `virtual void gazebo::physics::Entity::StopAnimation () [virtual]`

Stop the current animation, if any.

Reimplemented in **gazebo::physics::Model** (p. 533).

10.60.3.36 `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. 157).

Reimplemented in **gazebo::physics::Actor** (p. 125), **gazebo::physics::Model** (p. 533), **gazebo::physics::Link** (p. 472), and **gazebo::physics::Collision** (p. 272).

10.60.4 Member Data Documentation

10.60.4.1 `common::PoseAnimationPtr gazebo::physics::Entity::animation [protected]`

Current pose animation.

10.60.4.2 `event::ConnectionPtr gazebo::physics::Entity::animationConnection` [protected]

Connection used to update an animation.

10.60.4.3 `math::Pose gazebo::physics::Entity::animationStartPose` [protected]

Start pose of an animation.

10.60.4.4 `std::vector<event::ConnectionPtr> gazebo::physics::Entity::connections` [protected]

All our event connections.

10.60.4.5 `math::Pose gazebo::physics::Entity::dirtyPose` [protected]

The pose set by a physics engine.

10.60.4.6 `transport::NodePtr gazebo::physics::Entity::node` [protected]

Communication node.

10.60.4.7 `EntityPtr gazebo::physics::Entity::parentEntity` [protected]

A helper that prevents numerous `dynamic_casts`.

10.60.4.8 `msgs::Pose* gazebo::physics::Entity::poseMsg` [protected]

Pose message containr.

10.60.4.9 `common::Time gazebo::physics::Entity::prevAnimationTime` [protected]

Previous time an animation was updated.

10.60.4.10 `transport::PublisherPtr gazebo::physics::Entity::requestPub` [protected]

Request publisher.

10.60.4.11 `transport::PublisherPtr gazebo::physics::Entity::visPub` [protected]

Visual publisher.

10.60.4.12 `msgs::Visual* gazebo::physics::Entity::visualMsg` [protected]

Visual message container.

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

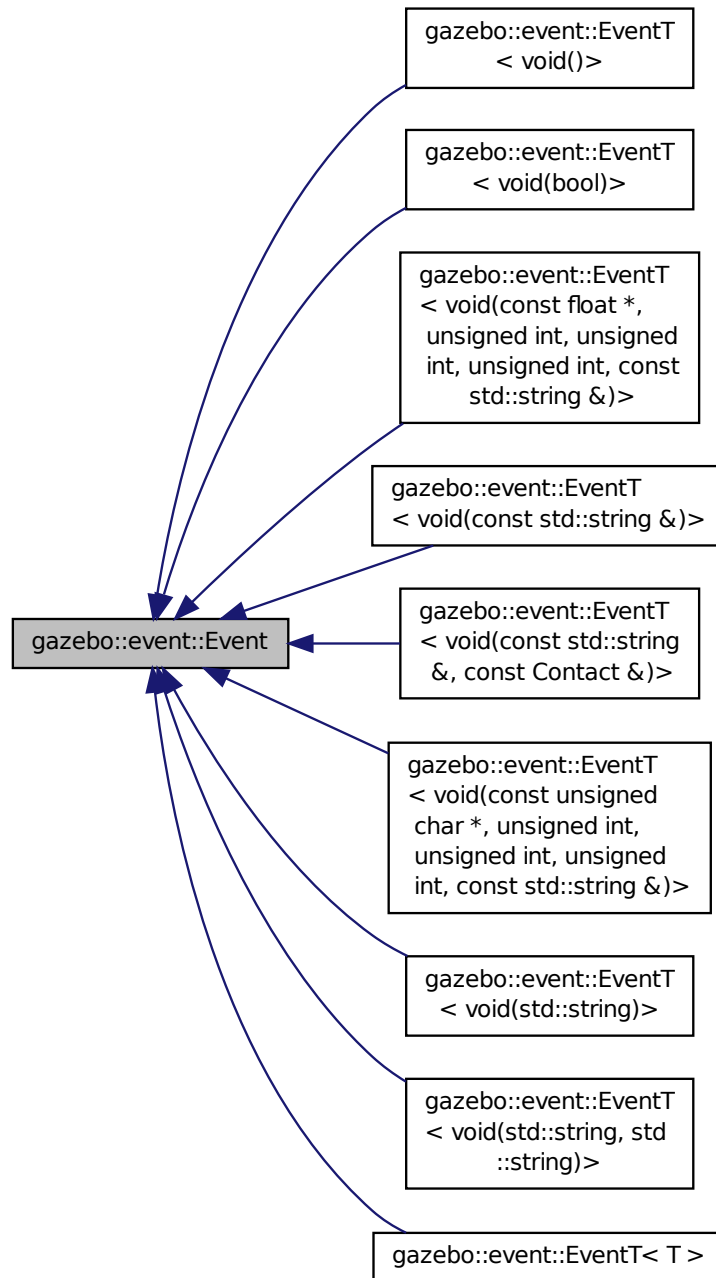
- **Entity.hh**

10.61 gazebo::event::Event Class Reference

Base class for all events.

```
#include <Event.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.

10.61.1 Detailed Description

Base class for all events.

10.61.2 Constructor & Destructor Documentation

10.61.2.1 virtual gazebo::event::Event::~~Event () [inline],[virtual]

Constructor.

10.61.3 Member Function Documentation

10.61.3.1 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. 46), `gazebo::event::EventT< void(std::string)>` (p. 46), `gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 46), `gazebo::event::EventT< void(const std::string &)>` (p. 46), `gazebo::event::EventT< void()>` (p. 46), `gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 46), `gazebo::event::EventT< void(const std::string &, const Contact &)>` (p. 46), `gazebo::event::EventT< void(std::string, std::string)>` (p. 46), and `gazebo::event::EventT< void(bool)>` (p. 46).

10.61.3.2 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. 47), `gazebo::event::EventT< void(std::string)>` (p. 47), `gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 47), `gazebo::event::EventT< void(const std::string &)>` (p. 47), `gazebo::event::EventT< void()>` (p. 47), `gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>` (p. 47), `gazebo::event::EventT< void(const std::string &, const Contact &)>` (p. 47), `gazebo::event::EventT< void(std::string, std::string)>` (p. 47), and `gazebo::event::EventT< void(bool)>` (p. 47).

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

- **Event.hh**

10.62 gazebo::event::Events Class Reference

An **Event** (p. 350) class to get notifications for simulator events.

```
#include <Events.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.*

10.62.1 Detailed Description

An **Event** (p. 350) class to get notifications for simulator events.

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

- **Events.hh**

10.63 gazebo::rendering::Events Class Reference

Base class for rendering events.

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

Static Public Member Functions

- template<typename T >
static **event::ConnectionPtr ConnectCreateScene** (T _subscriber)
- *Connect to a scene created event.*
- template<typename T >
static **event::ConnectionPtr ConnectRemoveScene** (T _subscriber)

Connect to a scene removed event.

- static void **DisconnectCreateScene** (event::ConnectionPtr _connection)

Disconnect from a scene created event.

- static void **DisconnectRemoveScene** (event::ConnectionPtr _connection)

Disconnect from a scene removed event.

Static Public Attributes

- static event::EventT < void(const std::string &) > **createScene**
The event used to trigger a create scene event.
- static event::EventT < void(const std::string &) > **removeScene**
The event used to trigger a remove scene event.

10.63.1 Detailed Description

Base class for rendering events.

10.63.2 Member Function Documentation

10.63.2.1 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectCreateScene (T _subscriber) [inline],[static]`

Connect to a scene created event.

Parameters

in	<code>_subscriber</code>	Callback to trigger when event occurs.
----	--------------------------	--

Returns

Pointer the connection. This must stay in scope.

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

10.63.2.2 `template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectRemoveScene (T _subscriber) [inline],[static]`

Connect to a scene removed event.

Parameters

in	<code>_subscriber</code>	Callback to trigger when event occurs.
----	--------------------------	--

Returns

Pointer the connection. This must stay in scope.

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

10.63.2.3 `static void gazebo::rendering::Events::DisconnectCreateScene (event::ConnectionPtr _connection) [inline], [static]`

Disconnect from a scene created event.

Parameters

in	<code>_connection</code>	The connection to disconnect.
----	--------------------------	-------------------------------

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

10.63.2.4 `static void gazebo::rendering::Events::DisconnectRemoveScene (event::ConnectionPtr _connection) [inline], [static]`

Disconnect from a scene removed event.

Parameters

in	<code>_connection</code>	The connection to disconnect.
----	--------------------------	-------------------------------

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

10.63.3 Member Data Documentation

10.63.3.1 `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()`.

10.63.3.2 `event::EventT<void (const std::string &)> gazebo::rendering::Events::removeScene [static]`

The event used to trigger a remove scene event.

Referenced by `ConnectRemoveScene()`, and `DisconnectRemoveScene()`.

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

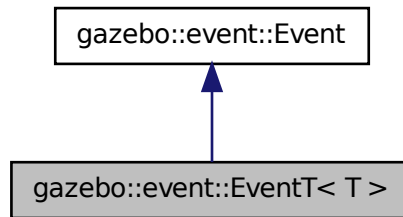
- **RenderEvents.hh**

10.64 gazebo::event::EventT< T > Class Template Reference

A class for event processing.

```
#include <Event.hh>
```


Inheritance diagram for gazebo::event::EventT< T >:



Public Member Functions

- virtual \sim **EventT** ()
Destructor.
- **ConnectionPtr Connect** (const boost::function< T > &_subscriber)
Connect a callback to this event.
- 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.

10.64.1 Detailed Description

```
template<typename T>class gazebo::event::EventT< T >
```

A class for event processing.

10.64.2 Member Function Documentation

10.64.2.1 `template<typename T> void gazebo::event::EventT< T >::operator()() [inline]`

Access the signal.

10.64.2.2 `template<typename T> template<typename P > void gazebo::event::EventT< T >::operator()(const P & _p) [inline]`

Signal the event with one parameter.

Parameters

<code>in</code>	<code>_p</code>	the parameter
-----------------	-----------------	---------------

10.64.2.3 `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

<code>in</code>	<code>_p1</code>	the first parameter
<code>in</code>	<code>_p2</code>	the second parameter

10.64.2.4 `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

<code>in</code>	<code>_p1</code>	the first parameter
<code>in</code>	<code>_p2</code>	the second parameter
<code>in</code>	<code>_p3</code>	the second parameter

10.64.2.5 `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

10.64.2.6 `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

10.64.2.7 `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

10.64.2.8 `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

10.64.2.9 `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

10.64.2.10 `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

10.64.2.11 `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

10.64.2.12 `template<typename T> void gazebo::event::EventT< T >::Signal () [inline]`

Signal the event for all subscribers.

Referenced by `gazebo::event::EventT< void(bool)>::operator()()`.

10.64.2.13 `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
----	-----------------	-----------

10.64.2.14 `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

10.64.2.15 `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

10.64.2.16 `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

10.64.2.17 `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

10.64.2.18 `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

10.64.2.19 `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

10.64.2.20 `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

10.64.2.21 `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

10.64.2.22 `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

<code>in</code>	<code>_p1</code>	the first parameter
<code>in</code>	<code>_p2</code>	the second parameter
<code>in</code>	<code>_p3</code>	the second parameter
<code>in</code>	<code>_p4</code>	the first parameter
<code>in</code>	<code>_p5</code>	the fifth parameter
<code>in</code>	<code>_p6</code>	the sixth parameter
<code>in</code>	<code>_p7</code>	the seventh parameter
<code>in</code>	<code>_p8</code>	the eighth parameter
<code>in</code>	<code>_p9</code>	the ninth parameter
<code>in</code>	<code>_p10</code>	the tenth parameter

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

- **Event.hh**

10.65 gazebo::common::Exception Class Reference

Class for generating exceptions.

```
#include <Exception.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.

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **gazebo::common::Exception** &_err)
stream insertion operator for Gazebo Error

10.65.1 Detailed Description

Class for generating exceptions.

10.65.2 Constructor & Destructor Documentation

10.65.2.1 gazebo::common::Exception::Exception ()

Constructor.

10.65.2.2 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

10.65.2.3 virtual gazebo::common::Exception::~~Exception () [virtual]

Destructor.

10.65.3 Member Function Documentation

10.65.3.1 std::string gazebo::common::Exception::GetErrorFile () const

Return the error function.

Returns

The error function name

10.65.3.2 std::string gazebo::common::Exception::GetErrorStr () const

Return the error string.

Returns

The error string

10.65.4 Friends And Related Function Documentation

10.65.4.1 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:

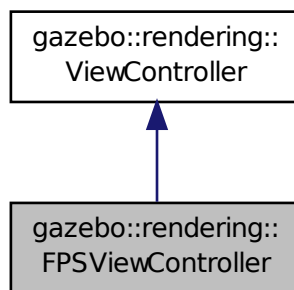
- **Exception.hh**

10.66 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

10.66.1 Detailed Description

First Person Shooter style view controller.

10.66.2 Constructor & Destructor Documentation

10.66.2.1 gazebo::rendering::FPSViewController::FPSViewController (**UserCameraPtr** *_camera*)

Constructor.

Parameters

in	Camera (p. 233)	to controll
----	------------------------	-------------

10.66.2.2 virtual gazebo::rendering::FPSViewController::~~FPSViewController () [virtual]

Destructor.

10.66.3 Member Function Documentation

10.66.3.1 static std::string gazebo::rendering::FPSViewController::GetTypeString () [static]

Get the type name of this view controller.

Returns

The name of the controller type: "fps"

10.66.3.2 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. 929).

10.66.3.3 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. 929).

10.66.3.4 `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. 929).

10.66.3.5 `virtual void gazebo::rendering::FPSViewController::Init ()` [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 930).

10.66.3.6 `virtual void gazebo::rendering::FPSViewController::Update ()` [virtual]

Update the camera position.

Implements **gazebo::rendering::ViewController** (p. 930).

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

- **FPSViewController.hh**

10.67 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**

10.67.1 Constructor & Destructor Documentation

10.67.1.1 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.

10.67.1.2 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.

10.67.2 Member Data Documentation

10.67.2.1 std::vector<TiXmlElement*> urdf2gazebo::GazeboExtension::blobs

Referenced by GazeboExtension().

10.67.2.2 double urdf2gazebo::GazeboExtension::damping_factor

Referenced by GazeboExtension().

10.67.2.3 std::string urdf2gazebo::GazeboExtension::fdir1

Referenced by GazeboExtension().

10.67.2.4 double urdf2gazebo::GazeboExtension::fudge_factor

Referenced by GazeboExtension().

10.67.2.5 bool urdf2gazebo::GazeboExtension::gravity

Referenced by GazeboExtension().

10.67.2.6 double urdf2gazebo::GazeboExtension::initial_joint_position

Referenced by GazeboExtension().

10.67.2.7 bool urdf2gazebo::GazeboExtension::is_damping_factor

Referenced by GazeboExtension().

10.67.2.8 bool urdf2gazebo::GazeboExtension::is_fudge_factor

Referenced by GazeboExtension().

10.67.2.9 bool urdf2gazebo::GazeboExtension::is_initial_joint_position

Referenced by GazeboExtension().

10.67.2.10 bool urdf2gazebo::GazeboExtension::is_kd

Referenced by GazeboExtension().

10.67.2.11 bool urdf2gazebo::GazeboExtension::is_kp

Referenced by GazeboExtension().

10.67.2.12 bool urdf2gazebo::GazeboExtension::is_laser_retro

Referenced by GazeboExtension().

10.67.2.13 `bool urdf2gazebo::GazeboExtension::is_maxVel`

Referenced by `GazeboExtension()`.

10.67.2.14 `bool urdf2gazebo::GazeboExtension::is_minDepth`

Referenced by `GazeboExtension()`.

10.67.2.15 `bool urdf2gazebo::GazeboExtension::is_mu1`

Referenced by `GazeboExtension()`.

10.67.2.16 `bool urdf2gazebo::GazeboExtension::is_mu2`

Referenced by `GazeboExtension()`.

10.67.2.17 `bool urdf2gazebo::GazeboExtension::is_stop_cfm`

Referenced by `GazeboExtension()`.

10.67.2.18 `bool urdf2gazebo::GazeboExtension::is_stop_erp`

Referenced by `GazeboExtension()`.

10.67.2.19 `double urdf2gazebo::GazeboExtension::kd`

Referenced by `GazeboExtension()`.

10.67.2.20 `double urdf2gazebo::GazeboExtension::kp`

Referenced by `GazeboExtension()`.

10.67.2.21 `double urdf2gazebo::GazeboExtension::laser_retro`

Referenced by `GazeboExtension()`.

10.67.2.22 `std::string urdf2gazebo::GazeboExtension::material`

Referenced by `GazeboExtension()`.

10.67.2.23 `double urdf2gazebo::GazeboExtension::maxVel`

Referenced by `GazeboExtension()`.

10.67.2.24 `double urdf2gazebo::GazeboExtension::minDepth`

Referenced by `GazeboExtension()`.

10.67.2.25 `double urdf2gazebo::GazeboExtension::mu1`

Referenced by `GazeboExtension()`.

10.67.2.26 `double urdf2gazebo::GazeboExtension::mu2`

Referenced by `GazeboExtension()`.

10.67.2.27 `std::string urdf2gazebo::GazeboExtension::old_link_name`

Referenced by `GazeboExtension()`.

10.67.2.28 `bool urdf2gazebo::GazeboExtension::provideFeedback`

Referenced by `GazeboExtension()`.

10.67.2.29 `gazebo::math::Pose urdf2gazebo::GazeboExtension::reduction_transform`

Referenced by `GazeboExtension()`.

10.67.2.30 `bool urdf2gazebo::GazeboExtension::self_collide`

Referenced by `GazeboExtension()`.

10.67.2.31 `bool urdf2gazebo::GazeboExtension::setStaticFlag`

Referenced by `GazeboExtension()`.

10.67.2.32 `double urdf2gazebo::GazeboExtension::stop_cfm`

Referenced by `GazeboExtension()`.

10.67.2.33 `double urdf2gazebo::GazeboExtension::stop_erp`

Referenced by `GazeboExtension()`.

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

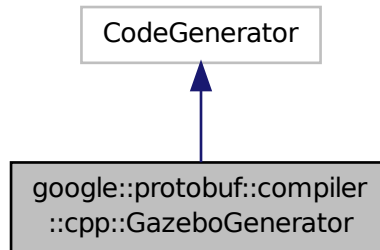
- **parser_urdf.hh**

10.68 google::protobuf::compiler::cpp::GazeboGenerator Class Reference

Google protobuf message generator for **gazebo::msgs** (p. 101).

```
#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

10.68.1 Detailed Description

Google protobuf message generator for **gazebo::msgs** (p. 101).

10.68.2 Constructor & Destructor Documentation

10.68.2.1 google::protobuf::compiler::cpp::GazeboGenerator::GazeboGenerator (const std::string & *_name*)

10.68.2.2 virtual google::protobuf::compiler::cpp::GazeboGenerator::~~GazeboGenerator () [virtual]

10.68.3 Member Function Documentation

10.68.3.1 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:

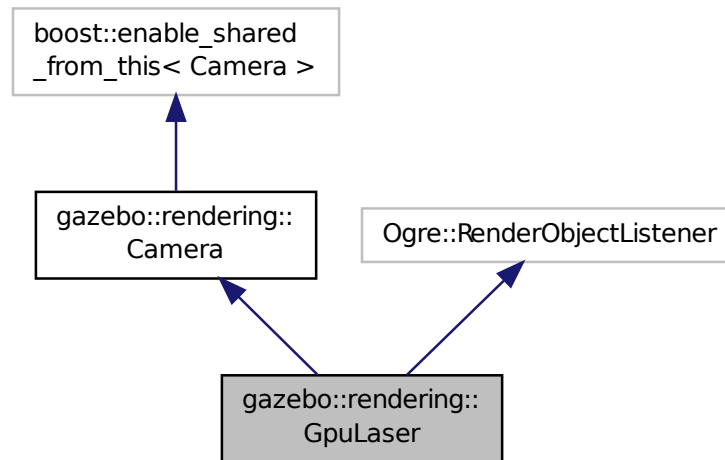
- **GazeboGenerator.hh**

10.69 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.
- void **Fini** ()
Finalize the camera.
- virtual const float * **GetLaserData** ()
All things needed to get back z buffer for laser data.
- void **Init** ()
Initialize the camera.
- void **Load** (sdf::ElementPtr &_sdf)
Load the camera with a set of parameters.

- void **Load** ()
Load the camera with default parameters.
- virtual void **notifyRenderSingleObject** (Ogre::Renderable *_rend, const Ogre::Pass *_p, const Ogre::AutoParamDataSource *_s, const Ogre::LightList *_ll, bool _supp)
- virtual void **PostRender** ()
Render the camera.
- 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

10.69.1 Detailed Description

GPU based laser distance sensor.

This is the base class for all cameras.

10.69.2 Constructor & Destructor Documentation

10.69.2.1 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. 746) that will contain the camera
in	<i>_autoRender</i>	Almost everyone should leave this as true.

10.69.2.2 virtual gazebo::rendering::GpuLaser::~GpuLaser () [virtual]

Destructor.

10.69.3 Member Function Documentation

10.69.3.1 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().

10.69.3.2 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
----	---------------------	-------------------------

10.69.3.3 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().

10.69.3.4 void gazebo::rendering::GpuLaser::Fini ()

Finalize the camera.

10.69.3.5 virtual const float* gazebo::rendering::GpuLaser::GetLaserData () [virtual]

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

Returns

Array of laser data

10.69.3.6 void gazebo::rendering::GpuLaser::Init ()

Initialize the camera.

10.69.3.7 void gazebo::rendering::GpuLaser::Load (sdf::ElementPtr & *_sdf*)

Load the camera with a set of parameters.

Parameters

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

10.69.3.8 void gazebo::rendering::GpuLaser::Load ()

Load the camera with default parameters.

10.69.3.9 virtual void gazebo::rendering::GpuLaser::notifyRenderSingleObject (Ogre::Renderable * *_rend*, const Ogre::Pass * *_p*, const Ogre::AutoParamDataSource * *_s*, const Ogre::LightList * *_ll*, bool *_supp*) [virtual]

10.69.3.10 virtual void gazebo::rendering::GpuLaser::PostRender () [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 248).

10.69.3.11 void gazebo::rendering::GpuLaser::SetParentSensor (sensors::GpuRaySensor * *_parent*)

Set the parent sensor.

Parameters

in	<i>_parent</i>	Pointer to a sensors::GpuRaySensor (p. 378)
----	----------------	--

10.69.3.12 void gazebo::rendering::GpuLaser::SetRangeCount (unsigned int *_w*, unsigned int *_h = 1*)

Set the number of laser samples in the width and height.

Parameters

in	<i>_w</i>	Number of samples in the horizontal sweep
in	<i>_h</i>	Number of samples in the vertical sweep

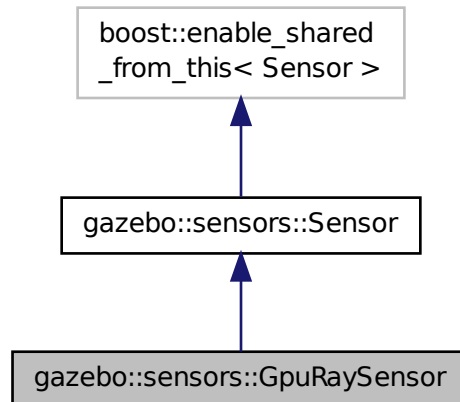
The documentation for this class was generated from the following file:

- **GpuLaser.hh**

10.70 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 a to the add entity signal Nate do these parameters need to be specified here?
- void **DisconnectNewLaserFrame** (event::ConnectionPtr &c)
Disconnect Laser Frame.
- double **Get1stRatio** ()
Nate fill in.
- double **Get2ndRatio** ()
- **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** ()
Gets the camera count.
- double **GetCHFOV** ()
- double **GetCVFOV** ()
- int **GetFiducial** (int _index)
Get detected fiducial value for a ray.
- double **GetHAngle** ()

- double **GetHFOV** ()
- **rendering::GpuLaserPtr GetLaserCamera** () const
*Returns a pointer to the internally kept **rendering::GpuLaser** (p. 375).*
- 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 If RangeResolution is 1, the number of simulated rays is equal to the number of returned range readings.
- void **GetRanges** (std::vector< double > &_ranges)
Get all the ranges.
- int **GetRayCount** () const
Get the ray count.
- double **GetRetro** (int _index)
Get detected retro (intensity) value for a ray.
- double **GetVAngle** ()
- **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** ()
- virtual void **Init** ()
Initialize the ray.
- bool **IsHorizontal** ()
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**
- **sdf::ElementPtr** **cameraElem**
- double **chfov**
- double **cvfov**
- double **far**
- double **hang**
- unsigned int **height_1st**
- unsigned int **height_2nd**
- double **hfov**
- **sdf::ElementPtr** **horzElem**
- bool **isHorizontal**
- double **near**
- **math::Vector3** **offset**
- **sdf::ElementPtr** **rangeElem**
- double **ratio_1st**
- double **ratio_2nd**
- **sdf::ElementPtr** **rayElem**
- **sdf::ElementPtr** **scanElem**
- double **vang**
- **sdf::ElementPtr** **vertElem**
- double **vfov**
- unsigned int **width_1st**
- unsigned int **width_2nd**

10.70.1 Constructor & Destructor Documentation

10.70.1.1 gazebo::sensors::GpuRaySensor::GpuRaySensor ()

Constructor.

10.70.1.2 virtual gazebo::sensors::GpuRaySensor::~~GpuRaySensor () [virtual]

Destructor.

10.70.2 Member Function Documentation

10.70.2.1 event::ConnectionPtr gazebo::sensors::GpuRaySensor::ConnectNewLaserFrame (boost::function< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)> *subscriber*)

Connect a to the add entity signal Nate do these parameters need to be specified here?

10.70.2.2 `void gazebo::sensors::GpuRaySensor::DisconnectNewLaserFrame (event::ConnectionPtr & c)`

Disconnect Laser Frame.

Parameters

<i>Connection</i>	pointer to disconnect
-------------------	-----------------------

10.70.2.3 `virtual void gazebo::sensors::GpuRaySensor::Fini ()` [protected],[virtual]

Finalize the ray.

Reimplemented from `gazebo::sensors::Sensor` (p. 768).

10.70.2.4 `double gazebo::sensors::GpuRaySensor::Get1stRatio ()`

Nate fill in.

I'm not sure what these what these sensor parameters refer to

10.70.2.5 `double gazebo::sensors::GpuRaySensor::Get2ndRatio ()`

Todo Document me

10.70.2.6 `math::Angle gazebo::sensors::GpuRaySensor::GetAngleMax () const`

Get the maximum angle.

Returns

the maximum angle

10.70.2.7 `math::Angle gazebo::sensors::GpuRaySensor::GetAngleMin () const`

Get the minimum angle.

Returns

The minimum angle

10.70.2.8 `double gazebo::sensors::GpuRaySensor::GetAngleResolution () const`

Get radians between each range.

10.70.2.9 `unsigned int gazebo::sensors::GpuRaySensor::GetCameraCount ()`

Gets the camera count.

Returns

Number of cameras

10.70.2.10 `double gazebo::sensors::GpuRaySensor::GetCHFOV ()`

Todo Document me

10.70.2.11 `double gazebo::sensors::GpuRaySensor::GetCVFOV ()`

Todo Document me

10.70.2.12 `int gazebo::sensors::GpuRaySensor::GetFiducial (int _index)`

Get detected fiducial value for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of specific ray
-----------------	----------------------------	-----------------------

Returns

Fiducial value of ray

10.70.2.13 `double gazebo::sensors::GpuRaySensor::GetHAngle ()`

Todo Document me

10.70.2.14 `double gazebo::sensors::GpuRaySensor::GetHFOV ()`

Todo Document me

10.70.2.15 `rendering::GpuLaserPtr gazebo::sensors::GpuRaySensor::GetLaserCamera () const` `[inline]`

Returns a pointer to the internally kept `rendering::GpuLaser` (p. 375).

Returns

Pointer to GpuLaser

10.70.2.16 `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><i>_index</i></code>	Index of specific ray
-----------------	----------------------------	-----------------------

Returns

Returns `DBL_MAX` for no detection.

10.70.2.17 `int gazebo::sensors::GpuRaySensor::GetRangeCount () const`

Get the range count.

Returns

The number of ranges

10.70.2.18 `double gazebo::sensors::GpuRaySensor::GetRangeMax () const`

Get the maximum range.

Returns

The maximum range

10.70.2.19 `double gazebo::sensors::GpuRaySensor::GetRangeMin () const`

Get the minimum range.

Returns

The minimum range

10.70.2.20 `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

10.70.2.21 `void gazebo::sensors::GpuRaySensor::GetRanges (std::vector< double > & _ranges)`

Get all the ranges.

Parameters

<code>_range</code>	A vector that will contain all the range data
---------------------	---

10.70.2.22 `int gazebo::sensors::GpuRaySensor::GetRayCount () const`

Get the ray count.

Returns

The number of rays

10.70.2.23 `double gazebo::sensors::GpuRaySensor::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

Intensity value of ray

10.70.2.24 `double gazebo::sensors::GpuRaySensor::GetVAngle ()`

Todo Document me

10.70.2.25 `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMax () const`

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.70.2.26 `math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMin () const`

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.70.2.27 `int gazebo::sensors::GpuRaySensor::GetVerticalRangeCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.70.2.28 `int gazebo::sensors::GpuRaySensor::GetVerticalRayCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.70.2.29 `double gazebo::sensors::GpuRaySensor::GetVFOV ()`

Todo Document me

10.70.2.30 `virtual void gazebo::sensors::GpuRaySensor::Init () [virtual]`

Initialize the ray.

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

10.70.2.31 `bool gazebo::sensors::GpuRaySensor::IsHorizontal ()`

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

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

Load the sensor with SDF parameters.

Parameters

in	<i>_sdf</i>	SDF Sensor (p. 765) parameters
in	<i>_worldName</i>	Name of world to load from

10.70.2.33 virtual void gazebo::sensors::GpuRaySensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

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

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.70.2.34 void gazebo::sensors::GpuRaySensor::SetAngleMax (double *_angle*)

Set the scan maximum angle.

Parameters

in	<i>_angle</i>	The maximum angle
----	---------------	-------------------

10.70.2.35 void gazebo::sensors::GpuRaySensor::SetAngleMin (double *_angle*)

Set the scan minimum angle.

Parameters

in	<i>_angle</i>	The minimum angle
----	---------------	-------------------

10.70.2.36 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMax (double *_angle*)

Set the vertical scan line top angle.

Parameters

in	<i>_angle</i>	The Maximum angle of the scan block
----	---------------	-------------------------------------

10.70.2.37 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMin (double *_angle*)

Set the vertical scan bottom angle.

Parameters

<code>in</code>	<code>_angle</code>	The minimum angle of the scan block
-----------------	---------------------	-------------------------------------

10.70.2.38 `virtual void gazebo::sensors::GpuRaySensor::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. 771).

10.70.3 Member Data Documentation

10.70.3.1 `unsigned int gazebo::sensors::GpuRaySensor::cameraCount` [protected]

10.70.3.2 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::cameraElem` [protected]

10.70.3.3 `double gazebo::sensors::GpuRaySensor::chfov` [protected]

10.70.3.4 `double gazebo::sensors::GpuRaySensor::cvfov` [protected]

10.70.3.5 `double gazebo::sensors::GpuRaySensor::far` [protected]

10.70.3.6 `double gazebo::sensors::GpuRaySensor::hang` [protected]

10.70.3.7 `unsigned int gazebo::sensors::GpuRaySensor::height_1st` [protected]

10.70.3.8 `unsigned int gazebo::sensors::GpuRaySensor::height_2nd` [protected]

10.70.3.9 `double gazebo::sensors::GpuRaySensor::hfov` [protected]

10.70.3.10 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::horzElem` [protected]

10.70.3.11 `bool gazebo::sensors::GpuRaySensor::isHorizontal` [protected]

10.70.3.12 `double gazebo::sensors::GpuRaySensor::near` [protected]

10.70.3.13 `math::Vector3 gazebo::sensors::GpuRaySensor::offset` [protected]

10.70.3.14 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::rangeElem` [protected]

10.70.3.15 `double gazebo::sensors::GpuRaySensor::ratio_1st` [protected]

10.70.3.16 `double gazebo::sensors::GpuRaySensor::ratio_2nd` [protected]

10.70.3.17 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::rayElem` [protected]

10.70.3.18 `sdf::ElementPtr gazebo::sensors::GpuRaySensor::scanElem` [protected]

- 10.70.3.19 double gazebo::sensors::GpuRaySensor::vang [protected]
- 10.70.3.20 sdf::ElementPtr gazebo::sensors::GpuRaySensor::vertElem [protected]
- 10.70.3.21 double gazebo::sensors::GpuRaySensor::vfov [protected]
- 10.70.3.22 unsigned int gazebo::sensors::GpuRaySensor::width_1st [protected]
- 10.70.3.23 unsigned int gazebo::sensors::GpuRaySensor::width_2nd [protected]

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

- **GpuRaySensor.hh**

10.71 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 numb.
- 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. 118) 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.

10.71.1 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.

10.71.2 Constructor & Destructor Documentation

10.71.2.1 gazebo::rendering::Grid::Grid (Scene * _scene, uint32_t _cellCount, float _cellLength, float _lineWidth, const common::Color & _color)

Constructor.

Parameters

in	Scene (p. 746)	The scene this object is part of
in	<i>cell_count</i>	The number of cells to draw
in	<i>cell_length</i>	The size of each cell
in	<i>r</i>	Red color component, in the range [0, 1]
in	<i>g</i>	Green color component, in the range [0, 1]
in	<i>b</i>	Blue color component, in the range [0, 1]

10.71.2.2 gazebo::rendering::Grid::~~Grid ()

Destructor.

10.71.3 Member Function Documentation

10.71.3.1 void gazebo::rendering::Grid::Enable (bool _enable)

Enable or disable the grid.

Parameters

in	<i>_enable</i>	Set to true to view the grid, false to make invisible.
----	----------------	--

10.71.3.2 uint32_t gazebo::rendering::Grid::GetCellCount () const `[inline]`

Get the numb.

10.71.3.3 `float gazebo::rendering::Grid::GetCellLength () const` `[inline]`

Get the cell length.

Returns

The cell length

10.71.3.4 `common::Color gazebo::rendering::Grid::GetColor () const` `[inline]`

Return the grid color.

Returns

The grid color

10.71.3.5 `uint32_t gazebo::rendering::Grid::GetHeight () const` `[inline]`

Get the height of the grid.

Returns

The height

10.71.3.6 `float gazebo::rendering::Grid::GetLineWidth () const` `[inline]`

Get the width of the grid line.

Returns

The line width

10.71.3.7 `Ogre::SceneNode* gazebo::rendering::Grid::GetSceneNode ()` `[inline]`

Get the **Ogre** (p. 118) scene node associated with this grid.

Returns

The **Ogre** (p. 118) scene node associated with this grid

10.71.3.8 `void gazebo::rendering::Grid::Init ()`

Initialize the grid.

10.71.3.9 `void gazebo::rendering::Grid::SetCellCount (uint32_t _count)`

Set the number of cells.

Parameters

in	<i>The</i>	number of cells
----	------------	-----------------

10.71.3.10 void gazebo::rendering::Grid::SetCellLength (float *_len*)

Set the cell length.

Parameters

in	<i>The</i>	cell length
----	------------	-------------

10.71.3.11 void gazebo::rendering::Grid::SetColor (const common::Color & *_color*)

Sets the color of the grid.

Parameters

in	<i>_color</i>	The grid color
----	---------------	----------------

10.71.3.12 void gazebo::rendering::Grid::SetHeight (uint32_t *_count*)

Set the height of the grid.

Parameters

in	<i>_count</i>	Grid (p. 389) height
----	---------------	-----------------------------

10.71.3.13 void gazebo::rendering::Grid::SetLineWidth (float *_width*)

Set the line width.

Parameters

in	<i>_width</i>	The width of the grid
----	---------------	-----------------------

10.71.3.14 void gazebo::rendering::Grid::SetUserData (const Ogre::Any & *_data*)

Sets user data on all ogre objects we own.

Parameters

in	<i>_data</i>	The user data
----	--------------	---------------

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

- **Grid.hh**

10.72 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.

10.72.1 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.

10.72.2 Constructor & Destructor Documentation

10.72.2.1 gazebo::physics::Gripper::Gripper (**ModelPtr** _model) [explicit]

Constructor.

Parameters

in	_model	The model which contains the Gripper (p. 393).
----	--------	---

10.72.2.2 virtual gazebo::physics::Gripper::~~Gripper () [virtual]

Destructor.

10.72.3 Member Function Documentation

10.72.3.1 virtual void gazebo::physics::Gripper::Init () [virtual]

Initialize.

10.72.3.2 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**

10.73 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. 233) to and overlay window.*
- bool **AttachCameraToImage** (**DepthCameraPtr** &_camera, const std::string &_windowName)
*Use this function to draw the output from a **rendering::DepthCamera** (p. 312) 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.

10.73.1 Detailed Description

A class that creates a CEGUI overlay on a render window.

10.73.2 Constructor & Destructor Documentation

10.73.2.1 gazebo::rendering::GUIOverlay::GUIOverlay ()

Constructor.

10.73.2.2 virtual gazebo::rendering::GUIOverlay::~~GUIOverlay () [virtual]

Destructor.

10.73.3 Member Function Documentation

10.73.3.1 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (CameraPtr & _camera, const std::string & _windowName)

Use this function to draw the output from a **rendering::Camera** (p. 233) to and overlay window.

Parameters

in	<i>_camera</i>	Pointer to the camera.
in	<i>_windowName</i>	Name of the window to receive the camera image

Returns

True if successful

10.73.3.2 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (DepthCameraPtr & _camera, const std::string & _windowName)

Use this function to draw the output from a **rendering::DepthCamera** (p. 312) to and overlay window.

Parameters

in	<i>_camera</i>	Pointer to the camera.
in	<i>_windowName</i>	Name of the window to receive the camera image

Returns

True if successful

10.73.3.3 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> .

10.73.3.4 `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.

10.73.3.5 `bool gazebo::rendering::GUIOverlay::HandleKeyPressEvent (const std::string & _key)`

Handle a key press event.

Parameters

in	<code>_key</code>	The key pressed.
----	-------------------	------------------

Returns

True if the key press event was handled.

10.73.3.6 `bool gazebo::rendering::GUIOverlay::HandleKeyReleaseEvent (const std::string & _key)`

Handle a key release event.

Parameters

in	<code>_key</code>	The key released.
----	-------------------	-------------------

Returns

True if the key release event was handled.

10.73.3.7 `bool gazebo::rendering::GUIOverlay::HandleMouseEvent (const common::MouseEvent & _evt)`

Handle a mouse event.

Parameters

<i>in</i>	<i>_evt</i>	The mouse event.
-----------	-------------	------------------

Returns

True if the mouse event was handled.

10.73.3.8 void gazebo::rendering::GUIOverlay::Hide ()

Make the overlay invisible.

10.73.3.9 void gazebo::rendering::GUIOverlay::Init (Ogre::RenderTarget * *_renderTarget*)

Initialize the overlay.

Parameters

<i>in</i>	<i>_renderTarget</i>	The render target which will have the overlay.
-----------	----------------------	--

10.73.3.10 bool gazebo::rendering::GUIOverlay::IsInitialized ()

Return true if the overlay has been initialized.

Returns

True if initialized

10.73.3.11 void gazebo::rendering::GUIOverlay::LoadLayout (const std::string & *_filename*)

Load a CEGUI layout file.

Parameters

<i>in</i>	<i>_filename</i>	Name of the layout file.
-----------	------------------	--------------------------

10.73.3.12 void gazebo::rendering::GUIOverlay::Resize (unsigned int *_width*, unsigned int *_height*)

Resize the window.

10.73.3.13 void gazebo::rendering::GUIOverlay::Show ()

Make the overlay visible.

10.73.3.14 void gazebo::rendering::GUIOverlay::Update ()

Update the overlay's objects.

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

- **GUIOverlay.hh**

10.74 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)
Get the height at a location.
- void **Load** ()
Load the heightmap.
- void **LoadFromMsg** (ConstVisualPtr &_msg)
Load the heightmap from a visual message.

10.74.1 Detailed Description

Rendering a terrain using heightmap information.

10.74.2 Constructor & Destructor Documentation

10.74.2.1 gazebo::rendering::Heightmap::Heightmap (ScenePtr _scene)

Constructor.

Parameters

in	_scene	Pointer to the scene that will contain the heightmap
----	--------	--

10.74.2.2 virtual gazebo::rendering::Heightmap::~~Heightmap () [virtual]

Destructor.

10.74.3 Member Function Documentation

10.74.3.1 double gazebo::rendering::Heightmap::GetHeight (double _x, double _y)

Get the height at a location.

Parameters

in	_x	X location
in	_y	Y location

Returns

The height at the specified location

10.74.3.2 void gazebo::rendering::Heightmap::Load ()

Load the heightmap.

10.74.3.3 void gazebo::rendering::Heightmap::LoadFromMsg (ConstVisualPtr & _msg)

Load the heightmap from a visual message.

Parameters

in	_msg	The visual message containing heightmap info
----	------	--

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

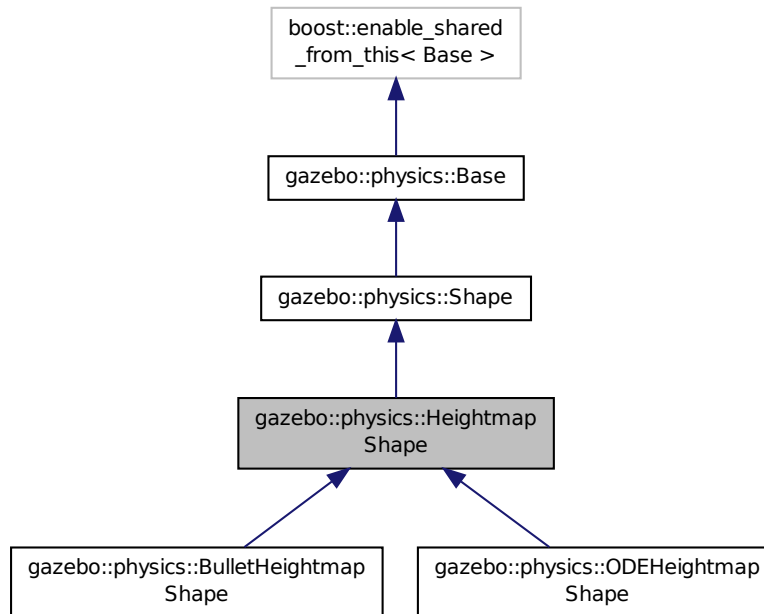
- **Heightmap.hh**

10.75 gazebo::physics::HeightmapShape Class Reference

HeightmapShape (p. 399) 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.
- 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

10.75.1 Detailed Description

HeightmapShape (p. 399) 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.

10.75.2 Constructor & Destructor Documentation

10.75.2.1 gazebo::physics::HeightmapShape::HeightmapShape (CollisionPtr _parent) [explicit]

Constructor.

Parameters

in	_parent	Parent Collision (p. 262) object.
----	---------	--

10.75.2.2 virtual gazebo::physics::HeightmapShape::~~HeightmapShape () [virtual]

Destructor.

10.75.3 Member Function Documentation

10.75.3.1 void gazebo::physics::HeightmapShape::FillMsg (msgs::Geometry & _msg) [virtual]

Fill a geometry message with this shape's data.

Parameters

<code>in</code>	<code>_msg</code>	Message to fill.
-----------------	-------------------	------------------

Implements **gazebo::physics::Shape** (p. 781).

10.75.3.2 `float gazebo::physics::HeightmapShape::GetHeight (int _x, int _y)`

Get a height at a position.

Parameters

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

Returns

The height at a the specified location.

10.75.3.3 `float gazebo::physics::HeightmapShape::GetMaxHeight () const`

Get the maximum height.

Returns

The maximum height.

10.75.3.4 `float gazebo::physics::HeightmapShape::GetMinHeight () const`

Get the minimum height.

Returns

The minimum height.

10.75.3.5 `math::Vector3 gazebo::physics::HeightmapShape::GetPos () const`

Get the origin in world coordinate frame.

Returns

The origin in world coordinate frame.

10.75.3.6 `math::Vector3 gazebo::physics::HeightmapShape::GetSize () const`

Get the size in meters.

Returns

The size in meters.

10.75.3.7 `std::string gazebo::physics::HeightmapShape::GetURI () const`

Get the URI of the heightmap image.

Returns

The heightmap image URI.

10.75.3.8 `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. 894), result.x = width, result.y = length/height.

10.75.3.9 `virtual void gazebo::physics::HeightmapShape::Init () [virtual]`

Initialize the heightmap.

Implements **gazebo::physics::Shape** (p. 781).

Reimplemented in **gazebo::physics::BulletHeightmapShape** (p.178), and **gazebo::physics::ODEHeightmapShape** (p. 586).

10.75.3.10 `virtual void gazebo::physics::HeightmapShape::Load (sdf::ElementPtr _sdf) [virtual]`

Load the heightmap.

Parameters

in	_sdf	SDF value to load from.
----	------	-------------------------

Reimplemented from **gazebo::physics::Base** (p. 154).

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

Update the heightmap from a message.

Parameters

in	_msg	Message to update from.
----	------	-------------------------

Implements **gazebo::physics::Shape** (p. 782).

10.75.4 Member Data Documentation

10.75.4.1 `std::vector<float> gazebo::physics::HeightmapShape::heights [protected]`

Lookup table of heights.

10.75.4.2 `common::Image gazebo::physics::HeightmapShape::img` [protected]

Image used to generate the heights.

10.75.4.3 `math::Vector3 gazebo::physics::HeightmapShape::scale` [protected]

Scaling factor.

10.75.4.4 `int gazebo::physics::HeightmapShape::subSampling` [protected]

Level of subsampling.

10.75.4.5 `unsigned int gazebo::physics::HeightmapShape::vertSize` [protected]

Size of the height lookup table.

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

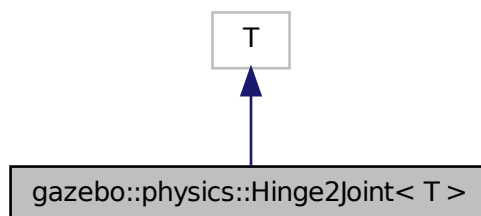
- **HeightmapShape.hh**

10.76 gazebo::physics::Hinge2Joint< T > Class Template Reference

A two axis hinge joint.

```
#include <Hinge2Joint.hh>
```

Inheritance diagram for gazebo::physics::Hinge2Joint< T >:



Public Member Functions

- **Hinge2Joint (BasePtr _parent)**
Constructor.
- **virtual ~Hinge2Joint ()**
Destructor.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

Load the joint.

10.76.1 Detailed Description

template<class T>class gazebo::physics::Hinge2Joint< T >

A two axis hinge joint.

10.76.2 Constructor & Destructor Documentation

10.76.2.1 template<class T> gazebo::physics::Hinge2Joint< T >::Hinge2Joint (BasePtr _parent) [inline]

Constructor.

10.76.2.2 template<class T> virtual gazebo::physics::Hinge2Joint< T >::~~Hinge2Joint () [inline],
[virtual]

Destructor.

10.76.3 Member Function Documentation

10.76.3.1 template<class T> virtual void gazebo::physics::Hinge2Joint< T >::Load (sdf::ElementPtr _sdf)
[inline], [protected], [virtual]

Load the joint.

Reimplemented in **gazebo::physics::BulletHinge2Joint** (p. 182), and **gazebo::physics::ODEHinge2Joint** (p. 589).

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

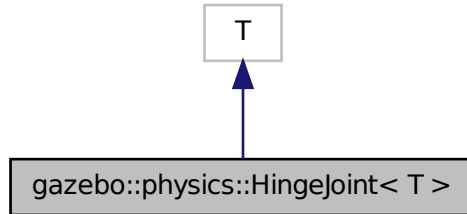
- **Hinge2Joint.hh**

10.77 gazebo::physics::HingeJoint< T > Class Template Reference

A single axis hinge joint.

```
#include <HingeJoint.hh>
```

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



Public Member Functions

- **HingeJoint** (**BasePtr** _parent)
Constructor.
- virtual **~HingeJoint** ()
Destructor.

Protected Member Functions

- virtual void **Init** ()
- virtual void **Load** (**sdf::ElementPtr** _sdf)
Load joint.

10.77.1 Detailed Description

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

A single axis hinge joint.

10.77.2 Constructor & Destructor Documentation

10.77.2.1 `template<class T> gazebo::physics::HingeJoint< T >::HingeJoint (BasePtr _parent)` `[inline]`

Constructor.

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

Destructor.

10.77.3 Member Function Documentation

10.77.3.1 `template<class T> virtual void gazebo::physics::HingeJoint< T >::Init () [inline],[protected],[virtual]`

10.77.3.2 `template<class T> virtual void gazebo::physics::HingeJoint< T >::Load (sdf::ElementPtr _sdf) [inline],[protected],[virtual]`

Load joint.

Reimplemented in `gazebo::physics::BulletHingeJoint` (p. 187), and `gazebo::physics::ODEHingeJoint` (p. 593).

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

- `HingeJoint.hh`

10.78 gazebo::common::Image Class Reference

Encapsulates an image.

```
#include <Image.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.

10.78.1 Detailed Description

Encapsulates an image.

10.78.2 Member Enumeration Documentation

10.78.2.1 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_GRBG8***

10.78.3 Constructor & Destructor Documentation

10.78.3.1 gazebo::common::Image (const std::string & *_filename* = " ")

Constructor.

Parameters

in	<i>_filename</i>	the path to the image
----	------------------	-----------------------

10.78.3.2 virtual gazebo::common::Image::~~Image () [virtual]

Destructor.

10.78.4 Member Function Documentation

10.78.4.1 Color gazebo::common::Image::GetAvgColor ()

Get the average color.

Returns

The average color

10.78.4.2 unsigned int gazebo::common::Image::GetBPP () const

Get the size of one pixel in bits.

Returns

The BPP of the image

10.78.4.3 void gazebo::common::Image::GetData (unsigned char ** *_data*, unsigned int & *_count*) const

Get the image as a data array.

Parameters

out	<i>_data</i>	Pointer to a NULL array of char.
out	<i>_count</i>	The resulting data array size

10.78.4.4 `std::string gazebo::common::Image::GetFilename () const`

Get the full filename of the image.

Returns

The filename used to load the image

10.78.4.5 `unsigned int gazebo::common::Image::GetHeight () const`

Get the height.

Returns

The image height

10.78.4.6 `Color gazebo::common::Image::GetMaxColor ()`

Get the max color.

Returns

The max color

10.78.4.7 `int gazebo::common::Image::GetPitch () const`

Returns

The pitch of the image

10.78.4.8 `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

10.78.4.9 `PixelFormat gazebo::common::Image::GetPixelFormat () const`

Get the pixel format.

Returns

PixelFormat

10.78.4.10 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

10.78.4.11 unsigned int gazebo::common::Image::GetWidth () const

Get the width.

Returns

The image width

10.78.4.12 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
----	------------------	----------------------------

10.78.4.13 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

10.78.4.14 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
----	------------------	-----------------------------

10.78.4.15 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

10.78.4.16 `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:

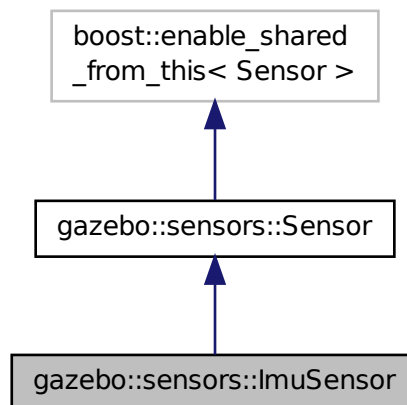
- **Image.hh**

10.79 gazebo::sensors::ImuSensor Class Reference

An IMU sensor.

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

Inheritance diagram for gazebo::sensors::ImuSensor:



Public Member Functions

- **ImuSensor** (Body *_body)

Constructor.

- virtual \sim **ImuSensor** ()

Destructor.

- Pose **GetVelocity** ()

*Returns velocity as a **math::Pose** (p. 677) FIXME storing x,y,z components in a quaternion seems like a bad idea.*

Protected Member Functions

- virtual void **FiniChild** ()

Finalize the ray.

- virtual void **InitChild** ()

Initialize the ray.

- virtual void **LoadChild** (XMLConfigNode *_node)

*Load the **ImuSensor** (p. 412) from XMLConfigNode.*

- virtual void **SaveChild** (std::string &_prefix, std::ostream &_stream)

Save the sensor info in XML format.

- virtual void **UpdateChild** ()

Update sensed values.

Additional Inherited Members

10.79.1 Detailed Description

An IMU sensor.

10.79.2 Constructor & Destructor Documentation

10.79.2.1 gazebo::sensors::ImuSensor::ImuSensor (Body * _body)

Constructor.

Parameters

<code>_body</code>	The IMU sensor must be attached to a body.
--------------------	--

10.79.2.2 virtual gazebo::sensors::ImuSensor::~ImuSensor () [virtual]

Destructor.

10.79.3 Member Function Documentation

10.79.3.1 virtual void gazebo::sensors::ImuSensor::FiniChild () [protected], [virtual]

Finalize the ray.

10.79.3.2 Pose gazebo::sensors::ImuSensor::GetVelocity ()

Returns velocity as a **math::Pose** (p. 677) FIXME storing x,y,z components in a quaternion seems like a bad idea.

Todo storing x,y,z components in a quaternion seems like a bad idea

Returns

velocity data stored in Pose Nate check

10.79.3.3 virtual void gazebo::sensors::ImuSensor::InitChild () [protected],[virtual]

Initialize the ray.

10.79.3.4 virtual void gazebo::sensors::ImuSensor::LoadChild (XMLConfigNode * _node) [protected],[virtual]

Load the **ImuSensor** (p. 412) from XMLConfigNode.

Parameters

<code>_node</code>	The XMLConfig node
--------------------	--------------------

10.79.3.5 virtual void gazebo::sensors::ImuSensor::SaveChild (std::string & *_prefix*, std::ostream & *_stream*) [protected],[virtual]

Save the sensor info in XML format.

Parameters

<code>_prefix</code>	
<code>_stream</code>	Nate fill in

10.79.3.6 virtual void gazebo::sensors::ImuSensor::UpdateChild () [protected],[virtual]

Update sensed values.

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

- **ImuSensor.hh**

10.80 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 \sim **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. 454) frame.*
- **math::Vector3** **GetPrincipalMoments** () const
Get the principal moments of inertia (Ixx, Iyy, Izz).
- **math::Vector3** **GetProductsofInertia** () const
Get the products of inertia (Ixy, Ixy, 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.*

10.80.1 Detailed Description

A class for inertial information about a link.

10.80.2 Constructor & Destructor Documentation

10.80.2.1 gazebo::physics::Inertial::Inertial ()

Default Constructor.

10.80.2.2 gazebo::physics::Inertial::Inertial (double _mass)

Constructor.

Parameters

<code>in</code>	<code>_mass</code>	Mass value in kg if using metric.
-----------------	--------------------	-----------------------------------

10.80.2.3 gazebo::physics::Inertial::Inertial (const Inertial & *_inertial*)

Copy constructor.

Parameters

<i>in</i>	<i>_inertial</i>	Inertial (p. 414) element to copy
-----------	------------------	--

10.80.2.4 virtual gazebo::physics::Inertial::~~Inertial () [virtual]

Destructor.

10.80.3 Member Function Documentation

10.80.3.1 const math::Vector3& gazebo::physics::Inertial::GetCoG () const [inline]

Get the center of gravity.

Returns

The center of gravity.

10.80.3.2 double gazebo::physics::Inertial::GetIXX () const

Get IXX.

Returns

IXX value

10.80.3.3 double gazebo::physics::Inertial::GetIXY () const

Get IXY.

Returns

IXY value

10.80.3.4 double gazebo::physics::Inertial::GetIXZ () const

Get IXZ.

Returns

IXZ value

10.80.3.5 `double gazebo::physics::Inertial::GetYZ () const`

Get IYZ.

Returns

IYZ value

10.80.3.6 `double gazebo::physics::Inertial::GetYX () const`

Get IXY.

Returns

IXY value

10.80.3.7 `double gazebo::physics::Inertial::GetIZZ () const`

Get IZZ.

Returns

IZZ value

10.80.3.8 `double gazebo::physics::Inertial::GetMass () const`

Get the mass.

10.80.3.9 `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. 454) frame.

Returns

The inertial pose.

10.80.3.10 `math::Vector3 gazebo::physics::Inertial::GetPrincipalMoments () const`

Get the principal moments of inertia (Ixx, Iyy, Izz).

Returns

The principal moments.

10.80.3.11 `math::Vector3 gazebo::physics::Inertial::GetProductsofInertia () const`

Get the products of inertia (Ixy, Ixy, Iyz).

Returns

The products of inertia.

10.80.3.12 void gazebo::physics::Inertial::Load (sdf::ElementPtr *_sdf*)

Load from SDF values.

Parameters

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

10.80.3.13 Inertial gazebo::physics::Inertial::operator+ (const Inertial & *_inertial*) const

Addition operator.

Parameters

<i>in</i>	<i>_inertial</i>	Inertial (p. 414) to add.
-----------	------------------	----------------------------------

Returns

The result of the addition.

10.80.3.14 const Inertial& gazebo::physics::Inertial::operator+= (const Inertial & *_inertial*)

Addition equal operator.

Parameters

<i>in</i>	<i>_inertial</i>	Inertial (p. 414) to add.
-----------	------------------	----------------------------------

Returns

Reference to this object.

10.80.3.15 Inertial& gazebo::physics::Inertial::operator= (const Inertial & *_inertial*)

Equal operator.

Parameters

<i>in</i>	<i>_inertial</i>	Inertial (p. 414) to copy.
-----------	------------------	-----------------------------------

Returns

Reference to this object.

10.80.3.16 void gazebo::physics::Inertial::ProcessMsg (const msgs::Inertial & *_msg*)

Update parameters from a message.

Parameters

in	<code>_msg</code>	Message to read
----	-------------------	-----------------

10.80.3.17 `void gazebo::physics::Inertial::Reset ()`

Reset all the mass properties.

10.80.3.18 `void gazebo::physics::Inertial::Rotate (const math::Quaternion & rot)`

Rotate this mass.

Parameters

in	<code>_rot</code>	Rotation amount.
----	-------------------	------------------

10.80.3.19 `void gazebo::physics::Inertial::SetCoG (double _cx, double _cy, double _cz)`

Set the center of gravity.

Parameters

in	<code>_cx</code>	X position.
in	<code>_cy</code>	Y position.
in	<code>_cz</code>	Z position.

10.80.3.20 `void gazebo::physics::Inertial::SetCoG (const math::Vector3 & _center)`

Set the center of gravity.

Parameters

in	<code>_center</code>	Center of the gravity.
----	----------------------	------------------------

10.80.3.21 `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.

10.80.3.22 void gazebo::physics::Inertial::SetIXX (double _v)

Set IXX.

Parameters

in	_v	IXX value
----	----	-----------

10.80.3.23 void gazebo::physics::Inertial::SetIXY (double _v)

Set IXY.

Parameters

in	_v	IXY value
----	----	-----------

10.80.3.24 void gazebo::physics::Inertial::SetIXZ (double _v)

Set IXZ.

Parameters

in	_v	IXZ value
----	----	-----------

10.80.3.25 void gazebo::physics::Inertial::SetIYY (double _v)

Set IYY.

Parameters

in	_v	IYY value
----	----	-----------

10.80.3.26 void gazebo::physics::Inertial::SetIYZ (double _v)

Set IYZ.

Parameters

in	_v	IXX value
----	----	-----------

10.80.3.27 void gazebo::physics::Inertial::SetIZZ (double _v)

Set IZZ.

Parameters

in	_v	IZZ value
----	----	-----------

10.80.3.28 void gazebo::physics::Inertial::SetMass (double *m*)

Set the mass.

10.80.3.29 void gazebo::physics::Inertial::UpdateParameters (sdf::ElementPtr *_sdf*)

update the parameters using new sdf values.

Parameters

in	<i>_sdf</i>	Update values from.
----	-------------	---------------------

10.80.4 Friends And Related Function Documentation

10.80.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::physics::Inertial & *_inertial*) [friend]

Output operator.

Parameters

in	<i>_out</i>	Output stream.
in	<i>_inertial</i>	Inertial (p. 414) object to output.

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

- **Inertial.hh**

10.81 gazebo::transport::IOManager Class Reference

Managers boost::asio IO.

```
#include <IOManager.hh>
```

Public Member Functions

- **IOManager** ()
- **~IOManager** ()
- void **DecCount** ()
- unsigned int **GetCount** () const
- boost::asio::io_service & **GetIO** ()
- void **IncCount** ()
- void **Stop** ()

10.81.1 Detailed Description

Managers boost::asio IO.

10.81.2 Constructor & Destructor Documentation

10.81.2.1 gazebo::transport::IOManager::IOManager ()

10.81.2.2 gazebo::transport::IOManager::~~IOManager ()

10.81.3 Member Function Documentation

10.81.3.1 void gazebo::transport::IOManager::DecCount ()

10.81.3.2 unsigned int gazebo::transport::IOManager::GetCount () const

10.81.3.3 boost::asio::io_service& gazebo::transport::IOManager::GetIO ()

10.81.3.4 void gazebo::transport::IOManager::IncCount ()

10.81.3.5 void gazebo::transport::IOManager::Stop ()

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

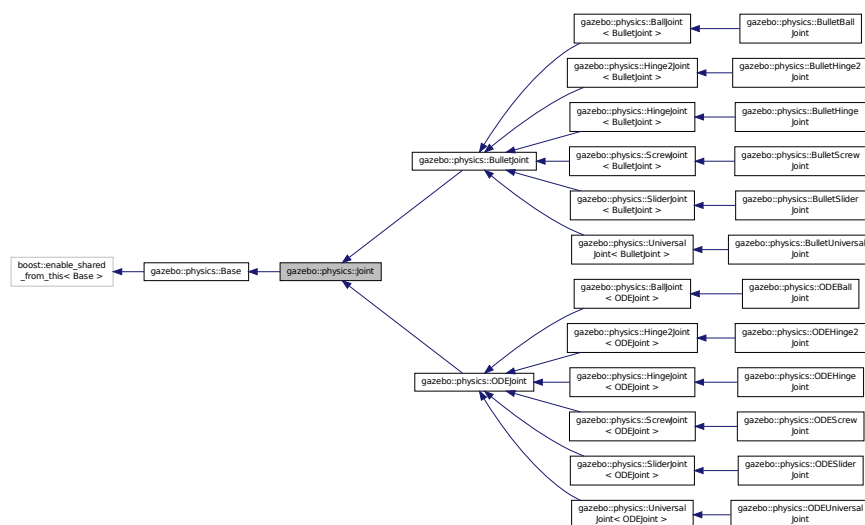
- [IOManager.hh](#)

10.82 gazebo::physics::Joint Class Reference

Base (p. 145) 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. 423) 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)
- **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. 454) due to the existence of this **Joint** (p. 423).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const =0
*Get the torque applied to the center of mass of a **physics::Link** (p. 454) due to the existence of this **Joint** (p. 423).*
- **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.
- **JointState GetState** ()
Get the joint state.
- 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. 423).*
- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load **physics::Joint** (p. 423) from a SDF **sdf::Element** (p. 332).*
- 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. 423) is static, Gazebo stores the state of this **Joint** (p. 423) as a scalar inside the **Joint** (p. 423) class, so this call will NOT move the joint dynamically for a static **Model** (p. 521).*
- virtual void **SetAttribute** (**Attribute** _attr, int _index, double _value)=0
Set a 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. 423).*
- 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.

10.82.1 Detailed Description

Base (p. 145) class for all joints.

10.82.2 Member Enumeration Documentation

10.82.2.1 enum gazebo::physics::Joint::Attribute

Joint (p. 423) 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.

10.82.3 Constructor & Destructor Documentation

10.82.3.1 gazebo::physics::Joint::Joint (BasePtr _parent) [explicit]

Constructor.

Parameters

in	Joint (p. 423)	parent
----	-----------------------	--------

10.82.3.2 virtual gazebo::physics::Joint::~~Joint () [virtual]

Destructor.

10.82.4 Member Function Documentation

10.82.4.1 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	<i>_one</i>	First link.
in	<i>_two</i>	Second link.

Returns

True if the two links are connected by a joint.

Implemented in **gazebo::physics::BulletJoint** (p. 189), and **gazebo::physics::ODEJoint** (p. 596).

10.82.4.2 virtual void gazebo::physics::Joint::Attach (LinkPtr _parent, LinkPtr _child) [virtual]

Attach the two bodies with this joint.

Parameters

in	<i>_parent</i>	Parent link.
in	<i>_child</i>	Child link.

Reimplemented in **gazebo::physics::ODEJoint** (p. 596), **gazebo::physics::BulletBallJoint** (p. 168), **gazebo::physics::BulletScrewJoint** (p. 215), **gazebo::physics::BulletHinge2Joint** (p. 181), **gazebo::physics::BulletHingeJoint** (p. 186), **gazebo::physics::BulletSliderJoint** (p. 219), and **gazebo::physics::BulletUniversalJoint** (p. 227).

10.82.4.3 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().

10.82.4.4 virtual void gazebo::physics::Joint::Detach () [virtual]

Detach this joint from all links.

Reimplemented in **gazebo::physics::ODEJoint** (p. 597), and **gazebo::physics::BulletJoint** (p. 190).

10.82.4.5 void gazebo::physics::Joint::DisconnectJointUpdate (event::ConnectionPtr & _conn) [inline]

Disconnect a boost::slot the the joint update signal.

Parameters

in	<code>_conn</code>	Connection to disconnect.
----	--------------------	---------------------------

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

10.82.4.6 void gazebo::physics::Joint::FillJointMsg (msgs::Joint & _msg)

DEPRECATED.

See Also

Joint::FillMsg (p. 428)

10.82.4.7 void gazebo::physics::Joint::FillMsg (msgs::Joint & _msg)

Fill a joint message.

Parameters

out	<code>_msg</code>	Message to fill with this joint's properties.
-----	-------------------	---

10.82.4.8 virtual math::Vector3 gazebo::physics::Joint::GetAnchor (int _index) const [pure virtual]

Get the anchor point.

Parameters

in	<code>_index</code>	Index of the axis.
----	---------------------	--------------------

Returns

Anchor value for the axis.

Implemented in **gazebo::physics::ScrewJoint**< **ODEJoint** > (p. 761), **gazebo::physics::ScrewJoint**< **BulletJoint** > (p. 761), **gazebo::physics::BulletJoint** (p. 190), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::BulletHinge2Joint** (p. 181), **gazebo::physics::SliderJoint**< **ODEJoint** > (p. 805), **gazebo::physics::SliderJoint**< **BulletJoint** > (p. 805), **gazebo::physics::BulletHingeJoint** (p. 186), **gazebo::physics::BulletUniversalJoint** (p. 227), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::BulletBallJoint** (p. 168), **gazebo::physics::ODEBallJoint** (p. 575), and **gazebo::physics::ODEUniversalJoint** (p. 636).

10.82.4.9 `math::Angle gazebo::physics::Joint::GetAngle (int _index) const`

Get the angle of rotation of an axis(index)

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Angle of the axis.

10.82.4.10 `virtual math::Angle gazebo::physics::Joint::GetAngleImpl (int _index) const` `[protected]`, `[pure virtual]`

Get the angle of an axis helper function.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

Angle of the axis.

Implemented in **gazebo::physics::BulletHinge2Joint** (p. 181), **gazebo::physics::BulletHingeJoint** (p. 186), **gazebo::physics::BulletUniversalJoint** (p. 227), **gazebo::physics::BulletScrewJoint** (p. 215), **gazebo::physics::BulletSliderJoint** (p. 219), **gazebo::physics::BulletBallJoint** (p. 168), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEBallJoint** (p. 575), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::ODESliderJoint** (p. 628), and **gazebo::physics::ODEUniversalJoint** (p. 636).

10.82.4.11 `LinkPtr gazebo::physics::Joint::GetChild () const`

Get the child link.

Returns

Pointer to the child link.

10.82.4.12 `virtual double gazebo::physics::Joint::GetForce (int _index) [virtual]`

Todo : not yet implemented. Get the internal forces at a this **Joint** (p.423). Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

The force applied to an axis.

Reimplemented in **`gazebo::physics::BulletHingeJoint`** (p.186).

10.82.4.13 `virtual math::Vector3 gazebo::physics::Joint::GetGlobalAxis (int _index) const [pure virtual]`

Get the axis of rotation in global coordinate frame.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis to get.
-----------------	----------------------------	---------------------------

Returns

Axis value for the provided index.

Implemented in **`gazebo::physics::BulletHinge2Joint`** (p.181), **`gazebo::physics::BulletHingeJoint`** (p.186), **`gazebo::physics::BulletUniversalJoint`** (p.227), **`gazebo::physics::BulletScrewJoint`** (p.215), **`gazebo::physics::BulletSliderJoint`** (p.219), **`gazebo::physics::BulletBallJoint`** (p.168), **`gazebo::physics::ODEHinge2Joint`** (p.589), **`gazebo::physics::ODEHingeJoint`** (p.593), **`gazebo::physics::ODEUniversalJoint`** (p.637), **`gazebo::physics::ODEBallJoint`** (p.575), **`gazebo::physics::ODEScrewJoint`** (p.624), and **`gazebo::physics::ODESliderJoint`** (p.628).

10.82.4.14 `virtual math::Angle gazebo::physics::Joint::GetHighStop (int _index) [pure virtual]`

Get the high stop of an axis(index).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
-----------------	----------------------------	--------------------

Returns

Angle of the high stop value.

Implemented in **`gazebo::physics::BulletHinge2Joint`** (p.181), **`gazebo::physics::BulletHingeJoint`** (p.186), **`gazebo::physics::ODEJoint`** (p.597), **`gazebo::physics::BulletUniversalJoint`** (p.227), **`gazebo::physics::BallJoint < ODEJoint >`** (p.144), **`gazebo::physics::BallJoint < BulletJoint >`** (p.144), **`gazebo::physics::BulletScrewJoint`** (p.215), and **`gazebo::physics::BulletSliderJoint`** (p.219).

10.82.4.15 `virtual LinkPtr gazebo::physics::Joint::GetJointLink (int _index) const` [pure virtual]

Get the link to which the joint is attached according the `_index`.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the link to retrieve.
-----------------	----------------------------	--------------------------------

Returns

Pointer to the request link. NULL if the index was invalid.

Implemented in `gazebo::physics::BulletJoint` (p. 190), and `gazebo::physics::ODEJoint` (p. 597).

10.82.4.16 `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. 454) due to the existence of this `Joint` (p. 423).

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

Parameters

<code>in</code>	<code><i>index</i></code>	The index of the link(0 or 1).
-----------------	---------------------------	--------------------------------

Returns

Force applied to the link.

Implemented in `gazebo::physics::ODEJoint` (p. 597), and `gazebo::physics::BulletJoint` (p. 190).

10.82.4.17 `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. 454) due to the existence of this `Joint` (p. 423).

Note that the unit of torque should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons-Meters. If using imperial units (sorry), then unit of force is lb-force-inches not (lb-mass-inches), etc.

Parameters

<code>in</code>	<code><i>index</i></code>	The index of the link(0 or 1)
-----------------	---------------------------	-------------------------------

Returns

Torque applied to the link.

Implemented in `gazebo::physics::ODEJoint` (p. 597), and `gazebo::physics::BulletJoint` (p. 190).

10.82.4.18 `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.

10.82.4.19 `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.

Implemented in `gazebo::physics::BulletHinge2Joint` (p. 181), `gazebo::physics::BulletHingeJoint` (p. 186), `gazebo::physics::ODEJoint` (p. 598), `gazebo::physics::BulletUniversalJoint` (p. 227), `gazebo::physics::BallJoint < ODEJoint >` (p. 144), `gazebo::physics::BallJoint < BulletJoint >` (p. 144), `gazebo::physics::BulletScrewJoint` (p. 215), and `gazebo::physics::BulletSliderJoint` (p. 219).

10.82.4.20 `virtual double gazebo::physics::Joint::GetMaxForce (int _index) [pure virtual]`

Get the max allowed force of an axis(index).

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

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
-----------------	---------------------	--------------------

Returns

The maximum force.

Implemented in `gazebo::physics::BulletScrewJoint` (p. 215), `gazebo::physics::BulletSliderJoint` (p. 219), `gazebo::physics::BulletHinge2Joint` (p. 182), `gazebo::physics::ODEScrewJoint` (p. 624), `gazebo::physics::BulletUniversalJoint` (p. 227), `gazebo::physics::ODEHingeJoint` (p. 593), `gazebo::physics::BulletHingeJoint` (p. 187), `gazebo::physics::ODEHinge2Joint` (p. 589), `gazebo::physics::ODESliderJoint` (p. 628), `gazebo::physics::ODEUniversalJoint` (p. 637), `gazebo::physics::BulletBallJoint` (p. 168), and `gazebo::physics::ODEBallJoint` (p. 576).

10.82.4.21 `LinkPtr gazebo::physics::Joint::GetParent () const`

Get the parent link.

Returns

Pointer to the parent link.

10.82.4.22 JointState gazebo::physics::Joint::GetState ()

Get the joint state.

Returns

The current joint state.

10.82.4.23 virtual double gazebo::physics::Joint::GetVelocity (int *_index*) const [pure virtual]

Get the rotation rate of an axis(index)

Parameters

in	<i>_index</i>	Index of the axis.
----	---------------	--------------------

Returns

The rotaional velocity of the joint axis.

Implemented in **gazebo::physics::BulletScrewJoint** (p. 215), **gazebo::physics::BulletSliderJoint** (p. 219), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::BulletBallJoint** (p. 168), **gazebo::physics::ODESliderJoint** (p. 628), **gazebo::physics::ODEUniversalJoint** (p. 637), and **gazebo::physics::ODEBallJoint** (p. 576).

10.82.4.24 virtual void gazebo::physics::Joint::Init () [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Base** (p. 153).

Reimplemented in **gazebo::physics::HingeJoint** < **ODEJoint** > (p. 407), and **gazebo::physics::HingeJoint** < **BulletJoint** > (p. 407).

10.82.4.25 void gazebo::physics::Joint::Load (LinkPtr *_parent*, LinkPtr *_child*, const math::Pose & *_pose*)

Set pose, parent and child links of a **physics::Joint** (p. 423).

Parameters

in	<i>_parent</i>	Parent link.
in	<i>_child</i>	Child link.
in	<i>_pose</i>	Pose of the link.

10.82.4.26 virtual void gazebo::physics::Joint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load **physics::Joint** (p. 423) from a SDF **sdf::Element** (p. 332).

Parameters

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

Reimplemented from `gazebo::physics::Base` (p. 154).

Reimplemented in `gazebo::physics::BallJoint< ODEJoint >` (p. 144), `gazebo::physics::BallJoint< BulletJoint >` (p. 144), `gazebo::physics::BulletScrewJoint` (p. 216), `gazebo::physics::ScrewJoint< ODEJoint >` (p. 761), `gazebo::physics::ScrewJoint< BulletJoint >` (p. 761), `gazebo::physics::BulletHinge2Joint` (p. 182), `gazebo::physics::BulletHingeJoint` (p. 187), `gazebo::physics::BulletSliderJoint` (p. 220), `gazebo::physics::ODEScrewJoint` (p. 624), `gazebo::physics::ODEHinge2Joint` (p. 589), `gazebo::physics::ODEHingeJoint` (p. 593), `gazebo::physics::SliderJoint< ODEJoint >` (p. 805), `gazebo::physics::SliderJoint< BulletJoint >` (p. 805), `gazebo::physics::Hinge2Joint< ODEJoint >` (p. 405), `gazebo::physics::Hinge2Joint< BulletJoint >` (p. 405), `gazebo::physics::HingeJoint< ODEJoint >` (p. 407), `gazebo::physics::HingeJoint< BulletJoint >` (p. 407), `gazebo::physics::UniversalJoint< ODEJoint >` (p. 870), `gazebo::physics::UniversalJoint< BulletJoint >` (p. 870), `gazebo::physics::BulletJoint` (p. 190), `gazebo::physics::ODEJoint` (p. 598), and `gazebo::physics::ODESliderJoint` (p. 628).

10.82.4.27 `virtual void gazebo::physics::Joint::Reset () [virtual]`

Reset the joint.

Reimplemented from `gazebo::physics::Base` (p. 155).

Reimplemented in `gazebo::physics::BulletJoint` (p. 191), and `gazebo::physics::ODEJoint` (p. 598).

10.82.4.28 `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.

Implemented in `gazebo::physics::ScrewJoint< ODEJoint >` (p. 761), `gazebo::physics::ScrewJoint< BulletJoint >` (p. 761), `gazebo::physics::BulletJoint` (p. 191), `gazebo::physics::BulletHingeJoint` (p. 187), `gazebo::physics::BulletHinge2Joint` (p. 182), `gazebo::physics::BulletUniversalJoint` (p. 228), `gazebo::physics::ODEHingeJoint` (p. 593), `gazebo::physics::SliderJoint< ODEJoint >` (p. 806), `gazebo::physics::SliderJoint< BulletJoint >` (p. 806), `gazebo::physics::BulletBallJoint` (p. 168), `gazebo::physics::ODEHinge2Joint` (p. 589), `gazebo::physics::ODEBallJoint` (p. 576), and `gazebo::physics::ODEUniversalJoint` (p. 637).

10.82.4.29 `void gazebo::physics::Joint::SetAngle (int _index, math::Angle _angle)`

If the **Joint** (p. 423) is static, Gazebo stores the state of this **Joint** (p. 423) as a scalar inside the **Joint** (p. 423) class, so this call will NOT move the joint dynamically for a static **Model** (p. 521).

But if this **Model** (p. 521) is not static, then it is updated dynamically, all the connected children **Link** (p. 454)'s are moved as a result of the **Joint** (p. 423) angle setting. Dynamic **Joint** (p. 423) angle update is accomplished by calling `JointController::SetJointPosition` (p. 439).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_angle</code>	Angle to set the joint to.

10.82.4.30 `virtual void gazebo::physics::Joint::SetAttribute (Attribute _attr, int _index, double _value) [pure virtual]`

Set a parameter for the joint.

Parameters

<code>in</code>	<code>_attr</code>	Attribute to set.
<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_value</code>	Value of the attribute.

Implemented in `gazebo::physics::ODEJoint` (p. 598), and `gazebo::physics::BulletJoint` (p. 191).

10.82.4.31 `virtual void gazebo::physics::Joint::SetAxis (int _index, const math::Vector3 & _axis) [pure virtual]`

Set the axis of rotation.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set.
<code>in</code>	<code>_axis</code>	Axis value.

Implemented in `gazebo::physics::BulletHinge2Joint` (p. 182), `gazebo::physics::BulletHingeJoint` (p. 187), `gazebo::physics::ODEHingeJoint` (p. 594), `gazebo::physics::BallJoint< ODEJoint >` (p. 144), `gazebo::physics::BallJoint< BulletJoint >` (p. 144), `gazebo::physics::BulletScrewJoint` (p. 216), `gazebo::physics::BulletUniversalJoint` (p. 228), `gazebo::physics::ODEUniversalJoint` (p. 637), `gazebo::physics::BulletSliderJoint` (p. 220), `gazebo::physics::ODEScrewJoint` (p. 624), `gazebo::physics::ODEHinge2Joint` (p. 590), and `gazebo::physics::ODESliderJoint` (p. 628).

10.82.4.32 `virtual void gazebo::physics::Joint::SetDamping (int _index, double _damping) [pure virtual]`

Set the joint damping.

Parameters

<code>in</code>	<code>_index</code>	Index of the axis to set.
<code>in</code>	<code>_damping</code>	Damping value for the axis.

Implemented in `gazebo::physics::BulletJoint` (p. 191), `gazebo::physics::BulletHinge2Joint` (p. 182), `gazebo::physics::BulletHingeJoint` (p. 187), `gazebo::physics::ODEHingeJoint` (p. 594), `gazebo::physics::BulletScrewJoint` (p. 216), `gazebo::physics::BulletUniversalJoint` (p. 228), `gazebo::physics::BulletSliderJoint` (p. 220), `gazebo::physics::ODEScrewJoint` (p. 624), `gazebo::physics::ODEHinge2Joint` (p. 590), `gazebo::physics::BulletBallJoint` (p. 169), `gazebo::physics::ODEBallJoint` (p. 576), `gazebo::physics::ODESliderJoint` (p. 628), and `gazebo::physics::ODEUniversalJoint` (p. 637).

10.82.4.33 `virtual void gazebo::physics::Joint::SetForce (int _index, double _force) [virtual]`

Set the force applied to this `physics::Joint` (p. 423).

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.

Reimplemented in `gazebo::physics::BulletScrewJoint` (p.216), `gazebo::physics::ODEHingeJoint` (p.594), `gazebo::physics::BulletHingeJoint` (p.187), `gazebo::physics::BulletSliderJoint` (p.220), `gazebo::physics::ODEHinge2Joint` (p.590), `gazebo::physics::BulletHinge2Joint` (p.182), `gazebo::physics::ODEScrewJoint` (p.625), `gazebo::physics::BulletUniversalJoint` (p.228), `gazebo::physics::ODESliderJoint` (p.628), and `gazebo::physics::ODEUniversalJoint` (p.637).

10.82.4.34 `virtual void gazebo::physics::Joint::SetHighStop (int _index, const math::Angle & _angle) [pure virtual]`

Set the high stop of an axis(index).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_angle</code>	High stop angle.

Implemented in `gazebo::physics::BulletHinge2Joint` (p.182), `gazebo::physics::BulletHingeJoint` (p.187), `gazebo::physics::ODEJoint` (p.598), `gazebo::physics::BulletUniversalJoint` (p.228), `gazebo::physics::BulletBallJoint` (p.169), `gazebo::physics::BulletScrewJoint` (p.216), and `gazebo::physics::BulletSliderJoint` (p.220).

10.82.4.35 `virtual void gazebo::physics::Joint::SetLowStop (int _index, const math::Angle & _angle) [pure virtual]`

Set the low stop of an axis(index).

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_angle</code>	Low stop angle.

Implemented in `gazebo::physics::BulletHinge2Joint` (p.183), `gazebo::physics::BulletHingeJoint` (p.188), `gazebo::physics::ODEJoint` (p.599), `gazebo::physics::BulletUniversalJoint` (p.228), `gazebo::physics::BulletBallJoint` (p.169), `gazebo::physics::BulletScrewJoint` (p.216), and `gazebo::physics::BulletSliderJoint` (p.220).

10.82.4.36 `virtual void gazebo::physics::Joint::SetMaxForce (int _index, double _force) [pure virtual]`

Set the max allowed force of an axis(index).

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

Parameters

in	<code>_index</code>	Index of the axis.
in	<code>_force</code>	Maximum force that can be applied to the axis.

Implemented in `gazebo::physics::BulletScrewJoint` (p.216), `gazebo::physics::BulletSliderJoint` (p.220), `gazebo::physics::BulletHinge2Joint` (p.183), `gazebo::physics::ODEScrewJoint` (p.625), `gazebo::physics::BulletUniversalJoint` (p.228), `gazebo::physics::ODEHingeJoint` (p.594), `gazebo::physics::BulletHingeJoint` (p.188), `gazebo::physics::ODEHinge2Joint` (p.590), `gazebo::physics::BulletBallJoint` (p.169), `gazebo::physics::ODE-`

SliderJoint (p. 629), **gazebo::physics::ODEUniversalJoint** (p. 637), and **gazebo::physics::ODEBallJoint** (p. 576).

10.82.4.37 `void gazebo::physics::Joint::SetModel (ModelPtr _model)`

Set the model this joint belongs too.

Parameters

<code>in</code>	<code>_model</code>	Pointer to a model.
-----------------	---------------------	---------------------

10.82.4.38 `void gazebo::physics::Joint::SetState (const JointState & _state)`

Set the joint state.

Parameters

<code>in</code>	<code>_state</code>	Joint (p. 423) state
-----------------	---------------------	-----------------------------

10.82.4.39 `virtual void gazebo::physics::Joint::SetVelocity (int _index, double _vel)` [pure virtual]

Set the velocity of an axis(index).

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_vel</code>	Velocity.

Implemented in **gazebo::physics::BulletScrewJoint** (p. 216), **gazebo::physics::BulletSliderJoint** (p. 220), **gazebo::physics::BulletHinge2Joint** (p. 183), **gazebo::physics::ODEScrewJoint** (p. 625), **gazebo::physics::ODEHinge2Joint** (p. 590), **gazebo::physics::ODEHingeJoint** (p. 594), **gazebo::physics::BulletHingeJoint** (p. 188), **gazebo::physics::BulletUniversalJoint** (p. 229), **gazebo::physics::ODESliderJoint** (p. 629), **gazebo::physics::ODEUniversalJoint** (p. 638), **gazebo::physics::BulletBallJoint** (p. 169), and **gazebo::physics::ODEBallJoint** (p. 576).

10.82.4.40 `void gazebo::physics::Joint::Update ()` [virtual]

Update the joint.

Reimplemented from **gazebo::physics::Base** (p. 157).

10.82.4.41 `virtual void gazebo::physics::Joint::UpdateParameters (sdf::ElementPtr _sdf)` [virtual]

Update the parameters using new sdf values.

Parameters

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

Reimplemented from **gazebo::physics::Base** (p. 157).

10.82.5 Member Data Documentation

10.82.5.1 `LinkPtr gazebo::physics::Joint::anchorLink` [protected]

Anchor link.

10.82.5.2 `math::Vector3 gazebo::physics::Joint::anchorPos` [protected]

Anchor pose.

10.82.5.3 `LinkPtr gazebo::physics::Joint::childLink` [protected]

The first link this joint connects to.

10.82.5.4 `double gazebo::physics::Joint::damping_coefficient` [protected]

joint damping_coefficient

10.82.5.5 `ModelPtr gazebo::physics::Joint::model` [protected]

Pointer to the parent model.

10.82.5.6 `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**

10.83 gazebo::physics::JointController Class Reference

A class for manipulating `physics::Joint` (p. 423).

```
#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. 423) by name.*
- void **SetJointPosition** (**JointPtr** _joint, double _position)

Set the positions of a **Joint** (p. 423) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 804) and radians for **HingeJoint** (p. 405), etc.

- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)

Set the positions of a set of **Joint** (p. 423)'s.

- void **Update** ()

Update the joint control.

10.83.1 Detailed Description

A class for manipulating **physics::Joint** (p. 423).

10.83.2 Constructor & Destructor Documentation

10.83.2.1 gazebo::physics::JointController::JointController (ModelPtr _model) [explicit]

Constructor.

Parameters

in	_model	Model (p. 521) that uses this joint controller.
----	--------	--

10.83.3 Member Function Documentation

10.83.3.1 void gazebo::physics::JointController::AddJoint (JointPtr _joint)

Add a joint to control.

Parameters

in	_joint	Joint (p. 423) to control.
----	--------	-----------------------------------

10.83.3.2 void gazebo::physics::JointController::Reset ()

Reset all commands.

10.83.3.3 void gazebo::physics::JointController::SetJointPosition (const std::string & _name, double _position)

Set the positions of a **Joint** (p. 423) by name.

See Also

JointController::SetJointPosition(JointPtr, double) (p. 439)

10.83.3.4 void gazebo::physics::JointController::SetJointPosition (JointPtr _joint, double _position)

Set the positions of a **Joint** (p. 423) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 804) and radians for **HingeJoint** (p. 405), etc.

Implementation: In order to change the position of a **Joint** (p. 423) inside a **Model** (p. 521), this call must recursively crawl through all the connected children **Link** (p. 454)'s in this **Model** (p. 521), and update each **Link** (p. 454) Pose affected by this **Joint** (p. 423) 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

<code>in</code>	<code>_joint</code>	Joint (p. 423) to set.
<code>in</code>	<code>_position</code>	Position of the joint.

10.83.3.5 `void gazebo::physics::JointController::SetJointPositions (const std::map< std::string, double > & _jointPositions)`

Set the positions of a set of **Joint** (p. 423)'s.

See Also

JointController::SetJointPosition(JointPtr, double) (p. 439)

10.83.3.6 `void gazebo::physics::JointController::Update ()`

Update the joint control.

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

- **JointController.hh**

10.84 gazebo::physics::JointFeedback Class Reference

Feedback information from a joint.

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

Public Member Functions

- **JointFeedback & operator=** (const **JointFeedback** &_feedback)
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.

10.84.1 Detailed Description

Feedback information from a joint.

These are forces and torques on parent and child **Link** (p. 454)'s

10.84.2 Member Function Documentation

10.84.2.1 JointFeedback& gazebo::physics::JointFeedback::operator= (const JointFeedback & *feedback*) [inline]

Operator =.

Parameters

<code>in</code>	<code>_feedback</code>	Joint (p. 423) feedback to set from.
-----------------	------------------------	---

Returns

*this

References body1Force, body1Torque, body2Force, and body2Torque.

10.84.3 Member Data Documentation

10.84.3.1 math::Vector3 gazebo::physics::JointFeedback::body1Force

Force on the first link.

Referenced by operator=().

10.84.3.2 math::Vector3 gazebo::physics::JointFeedback::body1Torque

Torque on the first link.

Referenced by operator=().

10.84.3.3 math::Vector3 gazebo::physics::JointFeedback::body2Force

Force on the second link.

Referenced by operator=().

10.84.3.4 math::Vector3 gazebo::physics::JointFeedback::body2Torque

Torque on the second link.

Referenced by operator=().

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

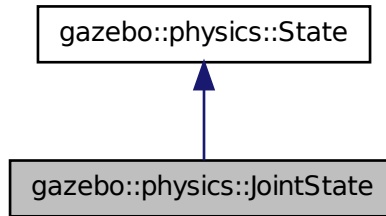
- **JointFeedback.hh**

10.85 gazebo::physics::JointState Class Reference

keeps track of state of a **physics::Joint** (p. 423)

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

Inheritance diagram for gazebo::physics::JointState:



Public Member Functions

- **JointState** ()
Default constructor.
- **JointState** (**JointPtr** _joint)
Constructor.
- virtual **~JointState** ()
Destructor.
- **math::Angle GetAngle** (unsigned int _axis) const
Get the joint angle.
- unsigned int **GetAngleCount** () const
Get the number of angles.
- virtual void **Load** (**sdf::ElementPtr** _elem)
Load state from SDF element.

Additional Inherited Members

10.85.1 Detailed Description

keeps track of state of a **physics::Joint** (p. 423)

10.85.2 Constructor & Destructor Documentation

10.85.2.1 gazebo::physics::JointState::JointState ()

Default constructor.

10.85.2.2 gazebo::physics::JointState::JointState (JointPtr *joint*) [explicit]

Constructor.

Parameters

in	<i>_joint</i>	Joint (p. 423) to get the state of.
----	---------------	-------------------------------------

10.85.2.3 virtual gazebo::physics::JointState::~~JointState () [virtual]

Destructor.

10.85.3 Member Function Documentation

10.85.3.1 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.

10.85.3.2 unsigned int gazebo::physics::JointState::GetAngleCount () const

Get the number of angles.

Returns

The number of angles.

10.85.3.3 virtual void gazebo::physics::JointState::Load (sdf::ElementPtr *elem*) [virtual]

Load state from SDF element.

Parameters

in	<i>_elem</i>	SDF values to load from.
----	--------------	--------------------------

Implements gazebo::physics::State (p. 815).

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

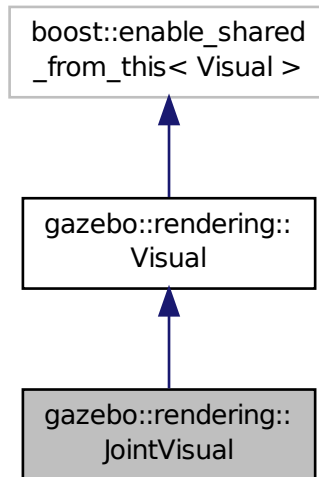
- JointState.hh

10.86 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

10.86.1 Detailed Description

Visualization for joints.

10.86.2 Constructor & Destructor Documentation

10.86.2.1 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

10.86.2.2 `virtual gazebo::rendering::JointVisual::~~JointVisual() [virtual]`

Destructor.

10.86.3 Member Function Documentation

10.86.3.1 `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:

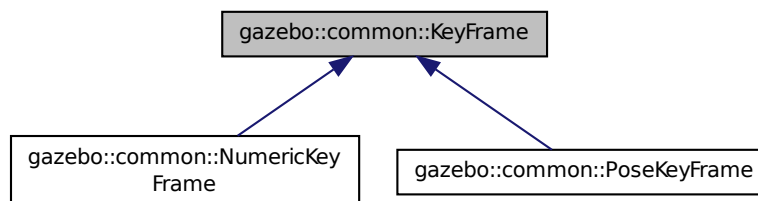
- **JointVisual.hh**

10.87 gazebo::common::KeyFrame Class Reference

A key frame in an animation.

```
#include <KeyFrame.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

10.87.1 Detailed Description

A key frame in an animation.

10.87.2 Constructor & Destructor Documentation

10.87.2.1 gazebo::common::KeyFrame::KeyFrame (double *time*)

Constructor.

Parameters

in	<i>_time</i>	Time (p. 840) of the keyframe in seconds
----	--------------	---

10.87.2.2 virtual gazebo::common::KeyFrame::~~KeyFrame () [virtual]

Destructor.

10.87.3 Member Function Documentation

10.87.3.1 double gazebo::common::KeyFrame::GetTime () const

Get the time of the keyframe.

Returns

the time

10.87.4 Member Data Documentation

10.87.4.1 double gazebo::common::KeyFrame::time [protected]

time of key frame

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

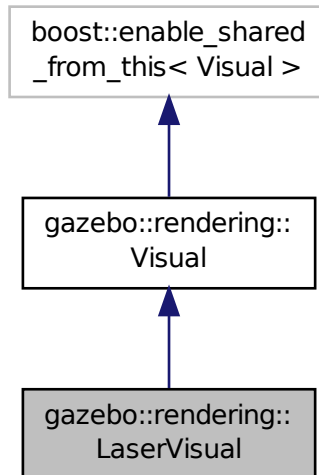
- **KeyFrame.hh**

10.88 gazebo::rendering::LaserVisual Class Reference

Visualization for laser data.

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

Inheritance diagram for gazebo::rendering::LaserVisual:



Public Member Functions

- **LaserVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)
Constructor.
- virtual \sim **LaserVisual** ()
Destructor.
- virtual void **SetEmissive** (const **common::Color** &_color)
Documentation inherited from parent.

Additional Inherited Members

10.88.1 Detailed Description

Visualization for laser data.

10.88.2 Constructor & Destructor Documentation

10.88.2.1 `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. 931).
in	<code>_topicName</code>	Name of the topic that has laser data.

10.88.2.2 `virtual gazebo::rendering::LaserVisual::~~LaserVisual () [virtual]`

Destructor.

10.88.3 Member Function Documentation

10.88.3.1 `virtual void gazebo::rendering::LaserVisual::SetEmissive (const common::Color & _color) [virtual]`

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 944).

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

- **LaserVisual.hh**

10.89 gazebo::rendering::Light Class Reference

A light source.

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

Public Member Functions

- **Light (Scene *_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.

10.89.1 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.

10.89.2 Constructor & Destructor Documentation

10.89.2.1 gazebo::rendering::Light::Light (Scene * _scene)

Constructor.

Parameters

in	_scene	Pointer to the scene that contains the Light (p. 448).
----	--------	---

10.89.2.2 virtual gazebo::rendering::Light::~~Light () [virtual]

Destructor.

10.89.3 Member Function Documentation

10.89.3.1 void gazebo::rendering::Light::FillMsg (msgs::Light & _msg) const

Fill the contents of a light message.

Parameters

out	_msg	Message to fill.
-----	------	------------------

10.89.3.2 common::Color gazebo::rendering::Light::GetDiffuseColor () const

Get the diffuse color.

Returns

The light's diffuse color.

10.89.3.3 math::Vector3 gazebo::rendering::Light::GetDirection () const

Get the direction.

Returns

The light's direction.

10.89.3.4 std::string gazebo::rendering::Light::GetName () const

Get the name of the visual.

Returns

The light's name.

10.89.3.5 math::Vector3 gazebo::rendering::Light::GetPosition () const

Get the position of the light.

Returns

The position of the light

10.89.3.6 common::Color gazebo::rendering::Light::GetSpecularColor () const

Get the specular color.

Returns

The specular color

10.89.3.7 std::string gazebo::rendering::Light::GetType () const

Get the type of the light.

Returns

The light type: "point", "spot", "directional".

10.89.3.8 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. 448) description.
----	------	--

10.89.3.9 void gazebo::rendering::Light::Load ()

Load the light using default parameters.

10.89.3.10 void gazebo::rendering::Light::LoadFromMsg (ConstLightPtr & _msg)

Load from a light message.

Parameters

in	_msg	Containing the light information.
----	------	-----------------------------------

10.89.3.11 `virtual void gazebo::rendering::Light::OnPoseChange () [inline],[protected],[virtual]`

On pose change callback.

10.89.3.12 `void gazebo::rendering::Light::SetAttenuation (double _constant, double _linear, double _quadratic)`

Set the attenuation.

Parameters

<code>in</code>	<code><i>_constant</i></code>	Constant attenuation
<code>in</code>	<code><i>_linear</i></code>	Linear attenuation
<code>in</code>	<code><i>_quadratic</i></code>	Quadratic attenuation

10.89.3.13 `void gazebo::rendering::Light::SetCastShadows (const bool & _cast)`

Set cast shadows.

Parameters

<code>in</code>	<code><i>_cast</i></code>	Set to true to cast shadows.
-----------------	---------------------------	------------------------------

10.89.3.14 `void gazebo::rendering::Light::SetDiffuseColor (const common::Color & _color)`

Set the diffuse color.

Parameters

<code>in</code>	<code><i>_color</i></code>	Light (p. 448) diffuse color.
-----------------	----------------------------	--------------------------------------

10.89.3.15 `void gazebo::rendering::Light::SetDirection (const math::Vector3 & _dir)`

Set the direction.

Parameters

<code>in</code>	<code><i>_dir</i></code>	Set the light's direction. Only applicable to spot and directional lights.
-----------------	--------------------------	--

10.89.3.16 `void gazebo::rendering::Light::SetLightType (const std::string & _type)`

Set the light type.

Parameters

<code>in</code>	<code><i>_type</i></code>	The light type: "point", "spot", "directional"
-----------------	---------------------------	--

10.89.3.17 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.
----	--------------	---------------------------

10.89.3.18 void gazebo::rendering::Light::SetPosition (const math::Vector3 & *_p*)

Set the position of the light.

Parameters

in	<i>_p</i>	New position for the light
----	-----------	----------------------------

10.89.3.19 void gazebo::rendering::Light::SetRange (const double & *_range*)

Set the range.

Parameters

in	<i>_range</i>	Range of the light in meters.
----	---------------	-------------------------------

10.89.3.20 virtual bool gazebo::rendering::Light::SetSelected (bool *_s*) [virtual]

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

Parameters

in	<i>_s</i>	Set to True when the light is selected by the user.
----	-----------	---

10.89.3.21 void gazebo::rendering::Light::SetSpecularColor (const common::Color & *_color*)

Set the specular color.

Parameters

in	<i>_color</i>	The specular color
----	---------------	--------------------

10.89.3.22 void gazebo::rendering::Light::SetSpotFalloff (const double & *_value*)

Set the spot light falloff.

Parameters

in	<i>_value</i>	Falloff value
----	---------------	---------------

10.89.3.23 void gazebo::rendering::Light::SetSpotInnerAngle (const double & *_angle*)

Set the spot light inner angle.

Parameters

in	<i>_angle</i>	Inner angle in radians
----	---------------	------------------------

10.89.3.24 void gazebo::rendering::Light::SetSpotOuterAngle (const double & *_angle*)

Set the spot light outer angle.

Parameters

in	<i>_angle</i>	Outer angle in radians
----	---------------	------------------------

10.89.3.25 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.
----	-----------	--

10.89.3.26 void gazebo::rendering::Light::ToggleShowVisual ()

10.89.3.27 void gazebo::rendering::Light::UpdateFromMsg (ConstLightPtr & *_msg*)

Update a light source from a message.

Parameters

in	<i>_msg</i>	Light (p. 448) message to update from
----	-------------	--

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

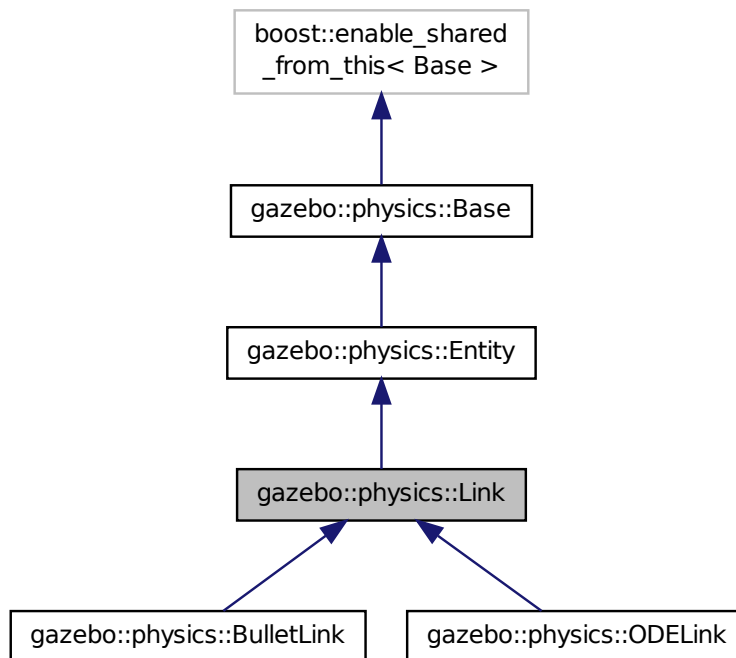
- **Light.hh**

10.90 gazebo::physics::Link Class Reference

Link (p. 454) 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. 454) as a parent **Link** (p. 454).*
- 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. 454) as a child **Link** (p. 454).*
- 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.
- 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.
- **LinkState GetState** ()
Get the link state.
- **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. 454) as a parent **Link** (p. 454).*
- **void RemoveParentJoint** (JointPtr _joint)
*Remove Joints that have this **Link** (p. 454) as a child **Link** (p. 454).*
- **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. 414) properties.

- **math::Vector3 linearAccel**

Linear acceleration.

- `std::vector< std::string >` **visuals**

Link (p. 454) visual elements.

Additional Inherited Members

10.90.1 Detailed Description

Link (p. 454) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

10.90.2 Constructor & Destructor Documentation

10.90.2.1 `gazebo::physics::Link::Link (EntityPtr _parent) [explicit]`

Constructor.

Parameters

in	_parent	Parent of this link.
----	---------	----------------------

10.90.2.2 `virtual gazebo::physics::Link::~~Link () [virtual]`

Destructor.

10.90.3 Member Function Documentation

10.90.3.1 `void gazebo::physics::Link::AddChildJoint (JointPtr _joint)`

Joints that have this **Link** (p. 454) as a parent **Link** (p. 454).

Parameters

in	_joint	Joint (p. 423) that is a child of this link.
----	--------	---

10.90.3.2 `virtual void gazebo::physics::Link::AddForce (const math::Vector3 & _force) [pure virtual]`

Add a force to the body.

Parameters

in	_force	Force to add.
----	--------	---------------

Implemented in **gazebo::physics::BulletLink** (p. 194), and **gazebo::physics::ODELink** (p. 602).

10.90.3.3 `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

<code>in</code>	<code><i>_force</i></code>	Force to add.
<code>in</code>	<code><i>_relPos</i></code>	Position on the link to add the force.

Implemented in `gazebo::physics::BulletLink` (p. 194), and `gazebo::physics::ODELink` (p. 603).

10.90.3.4 `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

<code>in</code>	<code><i>_force</i></code>	Force to add.
<code>in</code>	<code><i>_pos</i></code>	Position in global coord frame to add the force.

Implemented in `gazebo::physics::BulletLink` (p. 194), and `gazebo::physics::ODELink` (p. 603).

10.90.3.5 `void gazebo::physics::Link::AddParentJoint (JointPtr _joint)`

Joints that have this `Link` (p. 454) as a child `Link` (p. 454).

Parameters

<code>in</code>	<code><i>_joint</i></code>	<code>Joint</code> (p. 423) that is a parent of this link.
-----------------	----------------------------	--

10.90.3.6 `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

<code>in</code>	<code><i>_force</i></code>	Force to add.
-----------------	----------------------------	---------------

Implemented in `gazebo::physics::BulletLink` (p. 195), and `gazebo::physics::ODELink` (p. 603).

10.90.3.7 `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

<code>in</code>	<code><i>_torque</i></code>	Torque value to add.
-----------------	-----------------------------	----------------------

Implemented in `gazebo::physics::BulletLink` (p. 195), and `gazebo::physics::ODELink` (p. 603).

10.90.3.8 `virtual void gazebo::physics::Link::AddTorque (const math::Vector3 & _torque) [pure virtual]`

Add a torque to the body.

Parameters

<code>in</code>	<code>_torque</code>	Torque value to add to the link.
-----------------	----------------------	----------------------------------

Implemented in `gazebo::physics::BulletLink` (p. 195), and `gazebo::physics::ODELink` (p. 603).

10.90.3.9 `void gazebo::physics::Link::AttachStaticModel (ModelIPtr & _model, const math::Pose & _offset)`

Attach a static model to this link.

Parameters

<code>in</code>	<code>_model</code>	Pointer to a static model.
<code>in</code>	<code>_offset</code>	Pose relative to this link to place the model.

10.90.3.10 `template<typename T > event::ConnectionPtr gazebo::physics::Link::ConnectEnabled (T _subscriber) [inline]`

Connect to the add entity signal.

Parameters

<code>in</code>	<code>_subscriber</code>	Subscriber callback function.
-----------------	--------------------------	-------------------------------

Returns

Pointer to the connection, which must be kept in scope.

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

10.90.3.11 `void gazebo::physics::Link::DetachAllStaticModels ()`

Detach all static models from this link.

10.90.3.12 `void gazebo::physics::Link::DetachStaticModel (const std::string & _modelName)`

Detach a static model from this link.

Parameters

<code>in</code>	<code>_modelName</code>	Name of an attached model to detach.
-----------------	-------------------------	--------------------------------------

10.90.3.13 `void gazebo::physics::Link::DisconnectEnabled (event::ConnectionPtr & _conn) [inline]`

Disconnect to the add entity signal.

Parameters

in	_conn	Connection pointer to disconnect.
----	-------	-----------------------------------

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

10.90.3.14 void gazebo::physics::Link::FillLinkMsg (msgs::Link & .msg)

DEPRECATED.

10.90.3.15 void gazebo::physics::Link::FillMsg (msgs::Link & .msg)

Fill a link message.

Parameters

out	_msg	Message to fill
-----	------	-----------------

10.90.3.16 void gazebo::physics::Link::Fini () [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Entity** (p. 342).

Reimplemented in **gazebo::physics::ODELink** (p. 603).

10.90.3.17 double gazebo::physics::Link::GetAngularDamping () const

Get the angular damping factor.

Returns

Angular damping.

10.90.3.18 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. 342).

10.90.3.19 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.

10.90.3.20 CollisionPtr gazebo::physics::Link::GetCollision (const std::string & *_name*)

Get a child collision by name.

Parameters

<i>in</i>	<i>_name</i>	Name of the collision object.
-----------	--------------	-------------------------------

Returns

Pointer to the collision, NULL if the name was not found.

10.90.3.21 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.

10.90.3.22 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.

10.90.3.23 virtual bool gazebo::physics::Link::GetEnabled () const [pure virtual]

Get whether this body is enabled in the physics engine.

Returns

True if the link is enabled.

Implemented in **gazebo::physics::BulletLink** (p. 195), and **gazebo::physics::ODELink** (p. 603).

10.90.3.24 virtual bool gazebo::physics::Link::GetGravityMode () [pure virtual]

Get the gravity mode.

Returns

True if gravity is enabled.

Implemented in **gazebo::physics::ODELink** (p. 603), and **gazebo::physics::BulletLink** (p. 195).

10.90.3.25 `InertialPtr gazebo::physics::Link::GetInertial () const [inline]`

Get the inertia of the link.

Returns

Inertia of the link.

References inertial.

10.90.3.26 `virtual bool gazebo::physics::Link::GetKinematic () const [inline],[virtual]`

Implement this function.

Get whether this body is in the kinematic state.

Returns

True if the link is kinematic only.

Reimplemented in **gazebo::physics::ODELink** (p. 604).

10.90.3.27 `double gazebo::physics::Link::GetLinearDamping () const`

Get the linear damping factor.

Returns

Linear damping.

10.90.3.28 `ModelPtr gazebo::physics::Link::GetModel () const`

Get the model that this body belongs to.

Returns

Model (p. 521) that this body belongs to.

10.90.3.29 `Link_V gazebo::physics::Link::GetParentJointsLinks () const`

Returns a vector of parent Links connected by joints.

Returns

Vector of parent Links connected by joints.

10.90.3.30 `math::Vector3 gazebo::physics::Link::GetRelativeAngularAccel () const [virtual]`

Get the angular acceleration of the body.

Returns

Angular acceleration of the body.

Reimplemented from `gazebo::physics::Entity` (p. 343).

10.90.3.31 `math::Vector3 gazebo::physics::Link::GetRelativeAngularVel () const [virtual]`

Get the angular velocity of the body.

Returns

Angular velocity of the body.

Reimplemented from `gazebo::physics::Entity` (p. 343).

10.90.3.32 `math::Vector3 gazebo::physics::Link::GetRelativeForce () const`

Get the force applied to the body.

Returns

Force applied to the body.

10.90.3.33 `math::Vector3 gazebo::physics::Link::GetRelativeLinearAccel () const [virtual]`

Get the linear acceleration of the body.

Returns

Linear acceleration of the body.

Reimplemented from `gazebo::physics::Entity` (p. 344).

10.90.3.34 `math::Vector3 gazebo::physics::Link::GetRelativeLinearVel () const [virtual]`

Get the linear velocity of the body.

Returns

Linear velocity of the body.

Reimplemented from `gazebo::physics::Entity` (p. 344).

10.90.3.35 `math::Vector3 gazebo::physics::Link::GetRelativeTorque () const`

Get the torque applied to the body.

Returns

Torque applied to the body.

10.90.3.36 `bool gazebo::physics::Link::GetSelfCollide ()`

Get Self-Collision Flag, if this is true, this body will collide with other bodies even if they share the same parent.

Returns

True if self collision is enabled.

10.90.3.37 `unsigned int gazebo::physics::Link::GetSensorCount () const`

Get sensor count.

This will return the number of sensors created by the link when it was loaded. This function is commonly used with **Link::GetSensorName** (p. 466).

Returns

The number of sensors created by the link.

10.90.3.38 `std::string gazebo::physics::Link::GetSensorName (unsigned int _index) const`

Get sensor name.

Get the name of a sensor based on an index. The index should be in the range of 0...**Link::GetSensorCount()** (p. 466).

Note

A **Link** (p. 454) does not manage or maintain a pointer to a **sensors::Sensor** (p. 765). Access to a Sensor object is accomplished through the **sensors::SensorManager** (p. 774). This was done to separate the physics engine from the sensor engine.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the sensor name.
-----------------	----------------------------	---------------------------

Returns

The name of the sensor, or empty string if the index is out of bounds.

10.90.3.39 `LinkState gazebo::physics::Link::GetState ()`

Get the link state.

Returns

The state of this link.

10.90.3.40 `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. 344).

10.90.3.41 `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.

Implemented in **gazebo::physics::ODELink** (p. 604), and **gazebo::physics::BulletLink** (p. 195).

10.90.3.42 `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. 345).

10.90.3.43 `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.

Implemented in **gazebo::physics::ODELink** (p. 604), and **gazebo::physics::BulletLink** (p. 196).

10.90.3.44 `virtual void gazebo::physics::Link::Init () [virtual]`

Initialize the body.

Reimplemented from **gazebo::physics::Base** (p. 153).

Reimplemented in **gazebo::physics::BulletLink** (p. 196), and **gazebo::physics::ODELink** (p. 604).

10.90.3.45 `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. 346).

Reimplemented in `gazebo::physics::BulletLink` (p. 196), and `gazebo::physics::ODELink` (p. 605).

10.90.3.46 `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. 346).

Reimplemented in `gazebo::physics::BulletLink` (p. 196), and `gazebo::physics::ODELink` (p. 605).

10.90.3.47 `void gazebo::physics::Link::ProcessMsg (const msgs::Link & _msg)`

Update parameters from a message.

Parameters

in	_msg	Message to read.
----	------	------------------

10.90.3.48 `void gazebo::physics::Link::RemoveChildJoint (JointPtr _joint)`

Remove Joints that have this `Link` (p. 454) as a parent `Link` (p. 454).

Parameters

in	_joint	<code>Joint</code> (p. 423) that is a child of this link.
----	--------	---

10.90.3.49 `void gazebo::physics::Link::RemoveParentJoint (JointPtr _joint)`

Remove Joints that have this `Link` (p. 454) as a child `Link` (p. 454).

Parameters

in	_joint	<code>Joint</code> (p. 423) that is a parent of this link.
----	--------	--

10.90.3.50 `void gazebo::physics::Link::Reset () [virtual]`

Reset the link.

Reimplemented from `gazebo::physics::Entity` (p. 346).

10.90.3.51 void gazebo::physics::Link::SetAngularAccel (const math::Vector3 & *_accel*)

Set the angular acceleration of the body.

Parameters

in	<i>_accel</i>	Angular acceleration.
----	---------------	-----------------------

10.90.3.52 virtual void gazebo::physics::Link::SetAngularDamping (double *_damping*) [pure virtual]

Set the angular damping factor.

Parameters

in	<i>_damping</i>	Angular damping factor.
----	-----------------	-------------------------

Implemented in **gazebo::physics::ODELink** (p. 605), and **gazebo::physics::BulletLink** (p. 196).

10.90.3.53 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.
----	-------------	-------------------

Implemented in **gazebo::physics::ODELink** (p. 605), and **gazebo::physics::BulletLink** (p. 196).

10.90.3.54 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.
----	-----------------	---

Implemented in **gazebo::physics::ODELink** (p. 605), and **gazebo::physics::BulletLink** (p. 196).

10.90.3.55 void gazebo::physics::Link::SetCollideMode (const std::string & *_mode*)

Set the collide mode of the body.

Parameters

in	<i>_mode</i>	Collision (p. 262) 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
----	--------------	--

10.90.3.56 `virtual void gazebo::physics::Link::SetEnabled (bool _enable) const [pure virtual]`

Set whether this body is enabled.

Parameters

<code>in</code>	<code>_enable</code>	True to enable the link in the physics engine.
-----------------	----------------------	--

Implemented in `gazebo::physics::BulletLink` (p. 197), and `gazebo::physics::ODELink` (p. 605).

10.90.3.57 `virtual void gazebo::physics::Link::SetForce (const math::Vector3 & _force) [pure virtual]`

Set the force applied to the body.

Parameters

<code>in</code>	<code>_force</code>	Force value.
-----------------	---------------------	--------------

Implemented in `gazebo::physics::ODELink` (p. 605), and `gazebo::physics::BulletLink` (p. 197).

10.90.3.58 `virtual void gazebo::physics::Link::SetGravityMode (bool _mode) [pure virtual]`

Set whether gravity affects this body.

Parameters

<code>in</code>	<code>_mode</code>	True to enable gravity.
-----------------	--------------------	-------------------------

Implemented in `gazebo::physics::ODELink` (p. 606), and `gazebo::physics::BulletLink` (p. 197).

10.90.3.59 `void gazebo::physics::Link::SetInertial (const InertialPtr & _inertial)`

Set the mass of the link.

[in] `_inertial` **Inertial** (p. 414) value for the link.

10.90.3.60 `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>_kinematic</code>	True to make the link kinematic only.
-----------------	-------------------------	---------------------------------------

Reimplemented in `gazebo::physics::ODELink` (p. 606).

10.90.3.61 `void gazebo::physics::Link::SetLaserRetro (float _retro)`

Set the laser retro reflectiveness.

Parameters

in	<code>_retro</code>	Retro value for all child collisions.
----	---------------------	---------------------------------------

10.90.3.62 `void gazebo::physics::Link::SetLinearAccel (const math::Vector3 & _accel)`

Set the linear acceleration of the body.

Parameters

in	<code>_accel</code>	Linear acceleration.
----	---------------------	----------------------

10.90.3.63 `virtual void gazebo::physics::Link::SetLinearDamping (double _damping) [pure virtual]`

Set the linear damping factor.

Parameters

in	<code>_damping</code>	Linear damping factor.
----	-----------------------	------------------------

Implemented in `gazebo::physics::ODELink` (p. 606), and `gazebo::physics::BulletLink` (p. 197).

10.90.3.64 `virtual void gazebo::physics::Link::SetLinearVel (const math::Vector3 & _vel) [pure virtual]`

Set the linear velocity of the body.

Parameters

in	<code>_vel</code>	Linear velocity.
----	-------------------	------------------

Implemented in `gazebo::physics::ODELink` (p. 606), and `gazebo::physics::BulletLink` (p. 197).

10.90.3.65 `virtual bool gazebo::physics::Link::SetSelected (bool _set) [virtual]`

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

Parameters

in	<code>_set</code>	True to set the link as selected.
----	-------------------	-----------------------------------

Reimplemented from `gazebo::physics::Base` (p. 156).

10.90.3.66 `virtual void gazebo::physics::Link::SetSelfCollide (bool _collide) [pure virtual]`

Set whether this body will collide with others in the model.

Parameters

in	<code>_collid</code>	True to enable collisions.
----	----------------------	----------------------------

Implemented in **gazebo::physics::ODELink** (p. 606), and **gazebo::physics::BulletLink** (p. 197).

10.90.3.67 `void gazebo::physics::Link::SetState (const LinkState & _state)`

Set the current link state.

Parameters

in	_state	The state to set the link to.
----	--------	-------------------------------

10.90.3.68 `virtual void gazebo::physics::Link::SetTorque (const math::Vector3 & _torque)` [pure virtual]

Set the torque applied to the body.

Parameters

in	_torque	Torque value.
----	---------	---------------

Implemented in **gazebo::physics::ODELink** (p. 606), and **gazebo::physics::BulletLink** (p. 197).

10.90.3.69 `virtual void gazebo::physics::Link::Update ()` [virtual]

Update the body.

Reimplemented from **gazebo::physics::Base** (p. 157).

Reimplemented in **gazebo::physics::BulletLink** (p. 197), and **gazebo::physics::ODELink** (p. 606).

10.90.3.70 `virtual void gazebo::physics::Link::UpdateMass ()` [inline],[virtual]

Update the mass matrix.

Reimplemented in **gazebo::physics::ODELink** (p. 607).

10.90.3.71 `virtual void gazebo::physics::Link::UpdateParameters (sdf::ElementPtr _sdf)` [virtual]

Update the parameters using new sdf values.

Parameters

in	_sdf	SDF values to load from.
----	------	--------------------------

Reimplemented from **gazebo::physics::Entity** (p. 348).

10.90.3.72 `virtual void gazebo::physics::Link::UpdateSurface ()` [inline],[virtual]

Update surface parameters.

Reimplemented in **gazebo::physics::ODELink** (p. 607).

10.90.4 Member Data Documentation

10.90.4.1 `math::Vector3 gazebo::physics::Link::angularAccel` [protected]

Angular acceleration.

10.90.4.2 `std::vector<math::Pose> gazebo::physics::Link::attachedModelsOffset` [protected]

Offsets for the attached models.

10.90.4.3 `std::vector<std::string> gazebo::physics::Link::cgVisuals` [protected]

Center of gravity visual elements.

10.90.4.4 `InertialPtr gazebo::physics::Link::inertial` [protected]

Inertial (p. 414) properties.

Referenced by `GetInertial()`.

10.90.4.5 `math::Vector3 gazebo::physics::Link::linearAccel` [protected]

Linear acceleration.

10.90.4.6 `std::vector<std::string> gazebo::physics::Link::visuals` [protected]

Link (p. 454) visual elements.

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

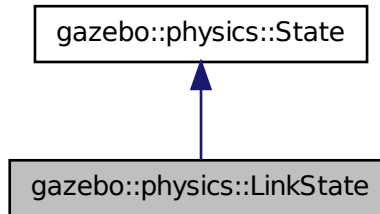
- **Link.hh**

10.91 gazebo::physics::LinkState Class Reference

Store state information of a **physics::Link** (p. 454) object.

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

Inheritance diagram for gazebo::physics::LinkState:



Public Member Functions

- **LinkState** ()
Default constructor.
- **LinkState** (const **LinkPtr** _link)
Constructor.
- virtual \sim **LinkState** ()
Destructor.
- void **FillStateSDF** (**sdf::ElementPtr** _elem)
*Fill a **State** (p. 813) SDF element with state info.*
- **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.
- std::list< **math::Pose** > **GetForces** () const
*Get the forces applied to the **Link** (p. 454).*
- **math::Pose GetPose** () const
Get the link pose.
- **math::Pose GetVelocity** () const
Get the link velocity.
- virtual void **Load** (**sdf::ElementPtr** _elem)
Load state from SDF element.
- void **UpdateLinkSDF** (**sdf::ElementPtr** _elem)
*Update a **Link** (p. 454) SDF element with this state info.*

Additional Inherited Members

10.91.1 Detailed Description

Store state information of a **physics::Link** (p. 454) object.

This class captures the entire state of a **Link** (p. 454) at one specific time during a simulation run.

State (p. 813) of a **Link** (p. 454) includes the state of itself all its child **Collision** (p. 262) entities.

10.91.2 Constructor & Destructor Documentation

10.91.2.1 gazebo::physics::LinkState::LinkState ()

Default constructor.

10.91.2.2 gazebo::physics::LinkState::LinkState (const LinkPtr *_link*) [explicit]

Constructor.

Build a **LinkState** (p. 473) from an existing **Link** (p. 454).

Parameters

in	<i>_model</i>	Pointer to the Link (p. 454) from which to gather state info.
----	---------------	--

10.91.2.3 virtual gazebo::physics::LinkState::~~LinkState () [virtual]

Destructor.

10.91.3 Member Function Documentation

10.91.3.1 void gazebo::physics::LinkState::FillStateSDF (sdf::ElementPtr *_elem*)

Fill a **State** (p. 813) SDF element with state info.

Stored state information into an SDF::Element pointer.

Parameters

in	<i>_elem</i>	Pointer to the SDF::Element which receives the data.
----	--------------	--

10.91.3.2 math::Pose gazebo::physics::LinkState::GetAcceleration () const

Get the link acceleration.

Returns

The acceleration represented as a **math::Pose** (p. 677)

10.91.3.3 CollisionState gazebo::physics::LinkState::GetCollisionState (unsigned int *_index*) const

Get a collision state.

Get a **Collision** (p. 262) **State** (p. 813) based on an index, where index is in the range of 0...**LinkState::GetCollisionStateCount** (p. 476)

Parameters

<i>in</i>	<i>_index</i>	Index of the CollisionState (p. 272)
-----------	---------------	---

Returns

State (p. 813) of the **Collision** (p. 262)

10.91.3.4 CollisionState gazebo::physics::LinkState::GetCollisionState (const std::string & *_collisionName*) const

Get a link state by link name.

Searches through all CollisionStates. Returns the **CollisionState** (p. 272) with the matching name, if any.

Parameters

<i>in</i>	<i>_collisionName</i>	Name of the CollisionState (p. 272)
-----------	-----------------------	--

Returns

State (p. 813) of the **Collision** (p. 262).

10.91.3.5 unsigned int gazebo::physics::LinkState::GetCollisionStateCount () const

Get the number of link states.

This returns the number of Collisions recorded.

Returns

Number of **CollisionState** (p. 272) recorded

10.91.3.6 std::list<math::Pose> gazebo::physics::LinkState::GetForces () const

Get the forces applied to the **Link** (p. 454).

Returns

The list of forces represented as a **math::Pose** (p. 677)

10.91.3.7 math::Pose gazebo::physics::LinkState::GetPose () const

Get the link pose.

Returns

The **math::Pose** (p. 677) of the **Link** (p. 454)

10.91.3.8 **math::Pose** gazebo::physics::LinkState::GetVelocity () const

Get the link velocity.

Returns

The velocity represented as a **math::Pose** (p. 677)

10.91.3.9 virtual void gazebo::physics::LinkState::Load (sdf::ElementPtr _elem) [virtual]

Load state from SDF element.

Load **LinkState** (p. 473) information from stored data in and SDF::Element

Parameters

in	_elem	Pointer to the SDF::Element containing state info.
----	-------	--

Implements **gazebo::physics::State** (p. 815).

10.91.3.10 void gazebo::physics::LinkState::UpdateLinkSDF (sdf::ElementPtr _elem)

Update a **Link** (p. 454) SDF element with this state info.

Set the values in a Links's SDF::Element with the information stored in this instance.

Parameters

in	_elem	Pointer to a Links's SDF::Element
----	-------	-----------------------------------

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

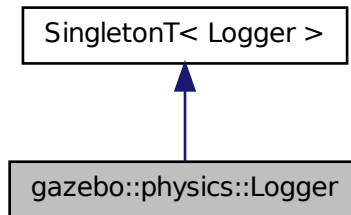
- **LinkState.hh**

10.92 gazebo::physics::Logger Class Reference

Handles logging of data to disk.

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

Inheritance diagram for gazebo::physics::Logger:



Public Member Functions

- bool **Add** (const std::string &_entityName, const std::string &_filename)
Add an entity to a log file.
- bool **Remove** (const std::string &_entity)
Remove an entity from a log.

Additional Inherited Members

10.92.1 Detailed Description

Handles logging of data to disk.

The **Logger** (p. 477) class is a Singleton that manages data logging of any entity within a running simulation. An entity may be a **World** (p. 954), **Model** (p. 521), or any of their child entities. This class only writes log files, see LogPlay for playback functionality.

State (p. 813) information for an entity may be logged through the **Logger::Add** (p. 478) function, and stopped through the **Logger::Remove** (p. 479) function. Data may be logged into a single file, or split into many separate files by specifying different filenames for the **Logger::Add** (p. 478) function.

The **Logger** (p. 477) is updated at the start of each simulation step. This guarantees that all data is stored.

See Also

Logplay (p. 479), **State** (p. 813)

10.92.2 Member Function Documentation

10.92.2.1 bool gazebo::physics::Logger::Add (const std::string & _entityName, const std::string & _filename)

Add an entity to a log file.

Add a new entity to a log. An entity can be any valid named object in the world, including the world itself. Duplicate additions are ignored. Entites can be added to the same file by passing specifying the same `_filename`.

Parameters

<code>_entityName</code>	Name of the entity to log.
<code>_filename</code>	Filename of the log file.

Returns

True if the **Entity** (p. 338) was added to a log. False if the entity is already being logged.

10.92.2.2 bool gazebo::physics::Logger::Remove (const std::string & _entity)

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>_entity</code>	Name of the entity to stop logging
----------------------	------------------------------------

Returns

True if the entity existed and was removed. False if the entity was not registered with the logger.

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

- **Logger.hh**

10.93 gazebo::physics::Logplay Class Reference

Open and playback log files that were recorded using **Logger** (p. 477).

```
#include <LogPlay.hh>
```

Public Member Functions

- Iterator **Begin** ()
Get an iterator to the beginning of the log file.
- Iterator **End** ()
Get an iterator to the end of the log file.
- bool **Open** (const std::string & _logFile)
Open a log file for reading.
- bool **Play** (**WorldPtr** _world)
*Play a log file in a given **World** (p. 954).*
- bool **Play** (**WorldPtr** _world, const std::string & _entityName)
*Play a log file in a given **World** (p. 954) using only data for a given entity.*
- bool **Play** (**WorldPtr** _world, const std::list< std::string > & _entityNames)
*Play a log file in a given **World** (p. 954) using only data for a list of entities.*
- bool **Play** (**WorldPtr** _world, **common::Time** _start, **common::Time** _stop)

*Play a segment of a log file in a given **World** (p. 954).*

- bool **Play** (**WorldPtr** _world, const std::string &_entityName, **common::Time** _start, **common::Time** _stop)

*Play a segment of a log file in a given **World** (p. 954) using only data for a given entity.*

- bool **Play** (**WorldPtr** _world, const std::list< std::string > &_entityNames, **common::Time** _start, **common::Time** _stop)

*Play a segment of a log file in a given **World** (p. 954) using only data for a list of entities.*

10.93.1 Detailed Description

Open and playback log files that were recorded using **Logger** (p. 477).

Use **Logplay** (p. 479) to open a log file (**Logplay::Open** (p. 481)), 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** (p. 954) using the **Play** functions. Replay involves reading and applying state information to a **World** (p. 954).

See Also

Logger (p. 477), **State** (p. 813)

10.93.2 Member Function Documentation

10.93.2.1 Iterator gazebo::physics::Logplay::Begin ()

Get an iterator to the beginning of the log file.

Requires a log file to be opened first. Use the iterator to step through a log file.

See Also

Logplay::Open (p. 481).

Returns

Iterator the beginning of the opened log file

10.93.2.2 Iterator gazebo::physics::Logplay::End ()

Get an iterator to the end of the log file.

Requires a log file to be opened first Use the iterator to step through a log file.

See Also

Logplay::Open (p. 481).

Returns

Iterator the end of the opened log file

10.93.2.3 bool gazebo::physics::Logplay::Open (const std::string & *_logFile*)

Open a log file for reading.

Open a log file that was previously recorded.

Parameters

<i>_logFile</i>	The file to load
-----------------	------------------

Returns

True if the log file was successfully loaded

10.93.2.4 bool gazebo::physics::Logplay::Play (WorldPtr *_world*)

Play a log file in a given **World** (p. 954).

Replay a complete log file in a **World** (p. 954). Requires a log file to be opened first,

See Also

Logplay::Open (p. 481).

Parameters

<i>_world</i>	Pointer to the World (p. 954)
---------------	--------------------------------------

Returns

True if replay was successful

10.93.2.5 bool gazebo::physics::Logplay::Play (WorldPtr *_world*, const std::string & *_entityName*)

Play a log file in a given **World** (p. 954) using only data for a given entity.

Replay a log file in a **World** (p. 954), but only apply data for the given entity. Requires a log file to be opened first

See Also

Logplay::Open (p. 481).

Parameters

<i>_world</i>	Pointer to the World (p. 954)
<i>_entityName</i>	Name of the entity to search the log for

Returns

True if replay was successful

10.93.2.6 `bool gazebo::physics::Logplay::Play (WorldPtr _world, const std::list< std::string > & _entityNames)`

Play a log file in a given **World** (p. 954) using only data for a list of entities.

Replay a log file in a **World** (p. 954), but only apply data for the given list of entities. Requires a log file to be opened first,

See Also

Logplay::Open (p. 481).

Parameters

<code>_world</code>	Pointer to the World (p. 954)
<code>_entityNames</code>	Names of the entities to search the log for

Returns

True if replay was successful

10.93.2.7 `bool gazebo::physics::Logplay::Play (WorldPtr _world, common::Time _start, common::Time _stop)`

Play a segment of a log file in a given **World** (p. 954).

Replay a segment of log file in a **World** (p. 954). The segment is defined by a start and end time, where time is simulation time. Requires a log file to be opened first,

See Also

Logplay::Open (p. 481).

Parameters

<code>_world</code>	Pointer to the World (p. 954).
<code>_start</code>	Start time, in simulation time.
<code>_stop</code>	Stop time, in simulation time.

Returns

True if replay was successful.

10.93.2.8 `bool gazebo::physics::Logplay::Play (WorldPtr _world, const std::string & _entityName, common::Time _start, common::Time _stop)`

Play a segment of a log file in a given **World** (p. 954) using only data for a given entity.

Replay a segment of log file in a **World** (p. 954), but only apply data for the given entity. The segment is defined by a start and end time, where time is simulation time. Requires a log file to be opened first,

See Also

Logplay::Open (p. 481).

Parameters

<code>_world</code>	Pointer to the World (p. 954).
<code>_entityName</code>	Name of the entity to search the log for
<code>_start</code>	Start time, in simulation time.
<code>_stop</code>	Stop time, in simulation time.

Returns

True if replay was successful.

10.93.2.9 `bool gazebo::physics::Logplay::Play (WorldPtr _world, const std::list< std::string > & _entityNames, common::Time _start, common::Time _stop)`

Play a segment of a log file in a given **World** (p. 954) using only data for a list of entities.

Replay a segment of a log file in a **World** (p. 954), but only apply data for the given list of entities. Requires a log file to be opened first,

See Also

Logplay::Open (p. 481).

Parameters

<code>_world</code>	Pointer to the World (p. 954)
<code>_entityNames</code>	Names of the entities to search the log for
<code>_start</code>	Start time, in simulation time.
<code>_stop</code>	Stop time, in simulation time.

Returns

True if replay was successful

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

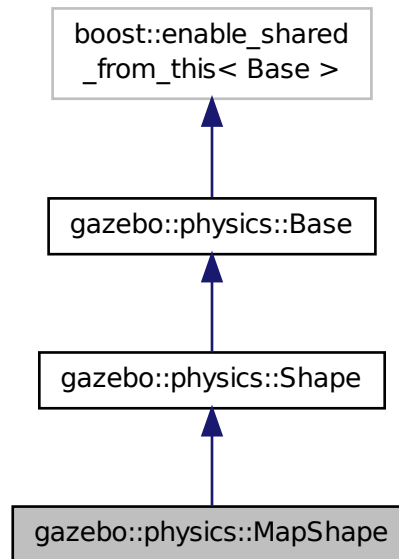
- **LogPlay.hh**

10.94 gazebo::physics::MapShape Class Reference

Creates box extrusions based on an image.

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

Inheritance diagram for gazebo::physics::MapShape:



Public Member Functions

- **MapShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~MapShape** ()
Destructor.
- void **FillMsg** (msgs::Geometry &_msg)
Fills out a msgs::Geometry message containing information about this map geometry object.
- int **GetGranularity** () const
Returns granularity of this geometry.
- double **GetHeight** () const
Returns height of this geometry.
- double **GetScale** () const
Returns scaling factor for this geometry.
- int **GetThreshold** () const
Returns image threshold for this geometry.
- std::string **GetURI** () const
Returns the image URI for this geometry.
- virtual void **Init** ()
Init the map.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the map.

- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
: *Implement this function.*
- void **Update** ()
Update function.

Additional Inherited Members

10.94.1 Detailed Description

Creates box extrusions based on an image.

This function is not yet complete, to be implemented.

10.94.2 Constructor & Destructor Documentation

10.94.2.1 gazebo::physics::MapShape::MapShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

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

10.94.2.2 virtual gazebo::physics::MapShape::~~MapShape () [virtual]

Destructor.

10.94.3 Member Function Documentation

10.94.3.1 void gazebo::physics::MapShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Fills out a msgs::Geometry message containing information about this map geometry object.

Parameters

in	<i>_msg</i>	Message to fill with this object's data.
----	-------------	--

Implements **gazebo::physics::Shape** (p. 781).

10.94.3.2 int gazebo::physics::MapShape::GetGranularity () const

Returns granularity of this geometry.

Returns

Granularity (amount of error between the image pixels and the 3D shapes created).

10.94.3.3 `double gazebo::physics::MapShape::GetHeight () const`

Returns height of this geometry.

All regions in image with value larger than `MapShape::scale` will be replaced by boxes with `MapShape::height`.

Returns

Height of the map shapes.

10.94.3.4 `double gazebo::physics::MapShape::GetScale () const`

Returns scaling factor for this geometry.

Returns

Scaling factor.

10.94.3.5 `int gazebo::physics::MapShape::GetThreshold () const`

Returns image threshold for this geometry.

All regions in image with value larger than `MapShape::scale` will be replaced by boxes with `MapShape::height`.

Returns

Image threshold value.

10.94.3.6 `std::string gazebo::physics::MapShape::GetURI () const`

Returns the image URI for this geometry.

Returns

The image URI that was used to load the map.

10.94.3.7 `virtual void gazebo::physics::MapShape::Init () [virtual]`

Init the map.

Implements **`gazebo::physics::Shape`** (p. 781).

10.94.3.8 `virtual void gazebo::physics::MapShape::Load (sdf::ElementPtr _sdf) [virtual]`

Load the map.

Parameters

in	_sdf	Load the map from SDF values.
----	------	-------------------------------

Reimplemented from **gazebo::physics::Base** (p. 154).

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

: Implement this function.

Parameters

<code>in</code>	<code>_msg</code>	Message to process, which will alter the map.
-----------------	-------------------	---

Implements **gazebo::physics::Shape** (p. 782).

10.94.3.10 `void gazebo::physics::MapShape::Update () [virtual]`

Update function.

Reimplemented from **gazebo::physics::Base** (p. 157).

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

- **MapShape.hh**

10.95 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.

10.95.1 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.487), runs World update and sensor generation loops.

10.95.2 Constructor & Destructor Documentation

10.95.2.1 gazebo::Master::Master ()

Constructor.

10.95.2.2 virtual gazebo::Master::~~Master () [virtual]

Destructor.

10.95.3 Member Function Documentation

10.95.3.1 void gazebo::Master::Fini ()

Finalize the master.

10.95.3.2 void gazebo::Master::Init (uint16_t _port)

Initialize.

Parameters

in	<code>_port</code>	The master's port
----	--------------------	-------------------

10.95.3.3 void gazebo::Master::Run ()

Run the master.

10.95.3.4 void gazebo::Master::RunOnce ()

Run the master one iteration.

10.95.3.5 void gazebo::Master::RunThread ()

Run the master in a new thread.

10.95.3.6 void gazebo::Master::Stop ()

Stop the master.

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

- **Master.hh**

10.96 gazebo::common::Material Class Reference

Encapsulates description of a material.

```
#include <Material.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.

10.96.1 Detailed Description

Encapsulates description of a material.

10.96.2 Member Enumeration Documentation

10.96.2.1 enum gazebo::common::Material::BlendMode

Enumerator:

ADD

MODULATE

REPLACE

BLEND_COUNT

10.96.2.2 enum gazebo::common::Material::ShadeMode

Enumerator:

FLAT

GOURAUD

PHONG

BLINN

SHADE_COUNT

10.96.3 Constructor & Destructor Documentation

10.96.3.1 gazebo::common::Material::Material ()

Constructor.

10.96.3.2 virtual gazebo::common::Material::~~Material () [virtual]

Destructor.

10.96.3.3 gazebo::common::Material::Material (const Color & _clr)

Create a material with a default color.

Parameters

in	<i>_clr</i>	Color (p. 274) of the material
----	-------------	---------------------------------------

10.96.4 Member Function Documentation

10.96.4.1 Color gazebo::common::Material::GetAmbient () const

Get the ambient color.

Returns

The ambient color

10.96.4.2 void gazebo::common::Material::GetBlendFactors (double & _srcFactor, double & _dstFactor)

Get the blend factors.

Parameters

in	<i>_srcFactor</i>	Source factor is returned in this variable
in	<i>_dstFactor</i>	Destination factor is returned in this variable

10.96.4.3 BlendMode gazebo::common::Material::GetBlendMode () const

Get the blending mode.

Returns

the blend mode

10.96.4.4 bool gazebo::common::Material::GetDepthWrite () const

Get depth write.

Returns

the depth write enabled state

10.96.4.5 Color gazebo::common::Material::GetDiffuse () const

Get the diffuse color.

Returns

The diffuse color

10.96.4.6 Color gazebo::common::Material::GetEmissive () const

Get the emissive color.

Returns

The emissive color

10.96.4.7 bool gazebo::common::Material::GetLighting () const

Get lighting enabled.

Returns

the lighting enabled state

10.96.4.8 std::string gazebo::common::Material::GetName () const

Get the name of the material.

Returns

The name of the material

10.96.4.9 double gazebo::common::Material::GetPointSize () const

Get the point size.

Returns

the point size

10.96.4.10 ShadeMode gazebo::common::Material::GetShadeMode () const

Get the shading mode.

Returns

the shading mode

10.96.4.11 `double gazebo::common::Material::GetShininess () const`

Get the shininess.

Returns

The shininess value

10.96.4.12 `Color gazebo::common::Material::GetSpecular () const`

Get the specular color.

Returns

The specular color

10.96.4.13 `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

10.96.4.14 `double gazebo::common::Material::GetTransparency () const`

Get the transparency percentage (0..1)

Returns

The transparency percentage

10.96.4.15 `void gazebo::common::Material::SetAmbient (const Color & _clr)`

Set the ambient color.

Parameters

<code>in</code>	<code>_clr</code>	The ambient color
-----------------	-------------------	-------------------

10.96.4.16 `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

<code>in</code>	<code>_srcFactor</code>	The source factor
<code>in</code>	<code>_dstFactor</code>	The destination factor

10.96.4.17 void gazebo::common::Material::SetBlendMode (BlendMode *_b*)

Set the blending mode.

Parameters

in	<i>_b</i>	the blend mode
----	-----------	----------------

10.96.4.18 void gazebo::common::Material::SetDepthWrite (bool *_value*)

Set depth write.

Parameters

in	<i>_value</i>	the depth write enabled state
----	---------------	-------------------------------

10.96.4.19 void gazebo::common::Material::SetDiffuse (const Color & *_clr*)

Set the diffuse color.

Parameters

in	<i>_clr</i>	The diffuse color
----	-------------	-------------------

10.96.4.20 void gazebo::common::Material::SetEmissive (const Color & *_clr*)

Set the emissive color.

Parameters

in	<i>_clr</i>	The emissive color
----	-------------	--------------------

10.96.4.21 void gazebo::common::Material::SetLighting (bool *_value*)

Set lighting enabled.

Parameters

in	<i>_value</i>	the lighting enabled state
----	---------------	----------------------------

10.96.4.22 void gazebo::common::Material::SetPointSize (double *_size*)

Set the point size.

Parameters

in	<i>_size</i>	the size
----	--------------	----------

10.96.4.23 void gazebo::common::Material::SetShadeMode (ShadeMode *_b*)

Set the shading mode param[in] the shading mode.

10.96.4.24 void gazebo::common::Material::SetShininess (double *_t*)

Set the shininess.

Parameters

<i>in</i>	<i>_t</i>	The shininess value
-----------	-----------	---------------------

10.96.4.25 void gazebo::common::Material::SetSpecular (const Color & *_clr*)

Set the specular color.

Parameters

<i>in</i>	<i>_clr</i>	The specular color
-----------	-------------	--------------------

10.96.4.26 void gazebo::common::Material::SetTextureImage (const std::string & *_tex*)

Set a texture image.

Parameters

<i>ij</i>	<i>_tex</i>	The name of the texture, which must be in Gazebo's resource path
-----------	-------------	--

10.96.4.27 void gazebo::common::Material::SetTextureImage (const std::string & *_tex*, const std::string & *_resourcePath*)

Set a texture image.

Parameters

<i>ij</i>	<i>_tex</i>	The name of the texture
<i>_resourcePath</i>	<i>_resourcePath</i>	Path which contains <i>_tex</i>

10.96.4.28 void gazebo::common::Material::SetTransparency (double *_t*)

Set the transparency percentage (0..1)

Parameters

<i>in</i>	<i>_t</i>	The amount of transparency (0..1)
-----------	-----------	-----------------------------------

10.96.5 Friends And Related Function Documentation

10.96.5.1 `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.

10.96.6 Member Data Documentation

10.96.6.1 **Color** `gazebo::common::Material::ambient` [protected]

the ambient light color

10.96.6.2 **BlendMode** `gazebo::common::Material::blendMode` [protected]

blend mode

10.96.6.3 `std::string gazebo::common::Material::BlendModeStr[BLEND_COUNT]` [static]

10.96.6.4 **Color** `gazebo::common::Material::diffuse` [protected]

the diffuse lighth color

10.96.6.5 **Color** `gazebo::common::Material::emissive` [protected]

the emissive light color

10.96.6.6 `std::string gazebo::common::Material::name` [protected]

the name of the material

10.96.6.7 `double gazebo::common::Material::pointSize` [protected]

point size

10.96.6.8 **ShadeMode** `gazebo::common::Material::shadeMode` [protected]

the shade mode

10.96.6.9 `std::string gazebo::common::Material::ShadeModeStr[SHADE_COUNT]` [static]

10.96.6.10 `double gazebo::common::Material::shininess` [protected]

shininess value (0 to 1)

10.96.6.11 **Color** `gazebo::common::Material::specular` [protected]

the specular light color

10.96.6.12 `std::string gazebo::common::Material::texImage` [protected]

the texture image file name

10.96.6.13 `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**

10.97 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.

10.97.1 Detailed Description

A 3x3 matrix class.

10.97.2 Constructor & Destructor Documentation

10.97.2.1 gazebo::math::Matrix3::Matrix3 ()

Constructor.

10.97.2.2 gazebo::math::Matrix3::Matrix3 (const Matrix3 & _m)

Copy constructor.

Parameters

<code>_m</code>	Matrix to copy
-----------------	----------------

10.97.2.3 gazebo::math::Matrix3::Matrix3 (double _v00, double _v01, double _v02, double _v10, double _v11, double _v12, double _v20, double _v21, double _v22)

Constructor.

Parameters

<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>_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>_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

10.97.2.4 virtual gazebo::math::Matrix3::~~Matrix3 () [virtual]

Destructor.

10.97.3 Member Function Documentation

10.97.3.1 `bool gazebo::math::Matrix3::operator==(const Matrix3 & _m) const`

Equality test operator.

Parameters

<code>in</code>	<code>_m</code>	Matrix3 (p. 498) to test
-----------------	-----------------	---------------------------------

Returns

True if equal (using the default tolerance of 1e-6)

10.97.3.2 `const double* gazebo::math::Matrix3::operator[](size_t _row) const` `[inline]`

Array subscript operator.

Parameters

<code>in</code>	<code>_row</code>	row index
-----------------	-------------------	-----------

Returns

a pointer to the row

References m.

10.97.3.3 `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.

10.97.3.4 `void gazebo::math::Matrix3::SetCol(unsigned int _c, const Vector3 & _v)`

Set a column.

Parameters

<code>in</code>	<code>_c</code>	The column index (0, 1, 2)
<code>in</code>	<code>_v</code>	The value to set in each row of the column

10.97.3.5 void gazebo::math::Matrix3::SetFromAxes (const Vector3 & *_xAxis*, const Vector3 & *_yAxis*, const Vector3 & *_zAxis*)

Set the matrix from three axis (1 per column)

Parameters

in	<i>_xAxis</i>	The x axis
in	<i>_yAxis</i>	The y axis
in	<i>_zAxis</i>	The z axis

10.97.3.6 void gazebo::math::Matrix3::SetFromAxis (const Vector3 & *_axis*, double *_angle*)

Set the matrix from an axis and angle.

Parameters

in	<i>_axis</i>	the axis
in	<i>_angle</i>	ccw rotation around the axis in radians

10.97.4 Friends And Related Function Documentation

10.97.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Matrix3 & *_m*) [friend]

Stream insertion operator.

Parameters

in	<i>_out</i>	Output stream
in	<i>_m</i>	Matrix to output

Returns

the stream

10.97.5 Member Data Documentation

10.97.5.1 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**

10.98 gazebo::math::Matrix4 Class Reference

A 3x3 matrix class.

```
#include <Matrix4.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. 677).*
- **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. 902).*
- **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.

10.98.1 Detailed Description

A 3x3 matrix class.

10.98.2 Constructor & Destructor Documentation

10.98.2.1 gazebo::math::Matrix4::Matrix4 ()

Constructor.

10.98.2.2 gazebo::math::Matrix4::Matrix4 (const Matrix4 & _m)

Copy constructor.

Parameters

<code>_m</code>	Matrix to copy
-----------------	----------------

10.98.2.3 gazebo::math::Matrix4::Matrix4 (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

Constructor.

Parameters

<code>in</code>	<code>_v00</code>	Row 0, Col 0 value
<code>in</code>	<code>_v01</code>	Row 0, Col 1 value
<code>in</code>	<code>_v02</code>	Row 0, Col 2 value
<code>in</code>	<code>_v03</code>	Row 0, Col 3 value
<code>in</code>	<code>_v10</code>	Row 1, Col 0 value

in	<code>_v11</code>	Row 1, Col 1 value
in	<code>_v12</code>	Row 1, Col 2 value
in	<code>_v13</code>	Row 1, Col 3 value
in	<code>_v20</code>	Row 2, Col 0 value
in	<code>_v21</code>	Row 2, Col 1 value
in	<code>_v22</code>	Row 2, Col 2 value
in	<code>_v23</code>	Row 2, Col 3 value
in	<code>_v30</code>	Row 3, Col 0 value
in	<code>_v31</code>	Row 3, Col 1 value
in	<code>_v32</code>	Row 3, Col 2 value
in	<code>_v33</code>	Row 3, Col 3 value

10.98.2.4 virtual gazebo::math::Matrix4::~Matrix4 () [virtual]

Destructor.

10.98.3 Member Function Documentation

10.98.3.1 **math::Pose** gazebo::math::Matrix4::GetAsPose () const

Get the transformation as **math::Pose** (p. 677).

Returns

the pose

10.98.3.2 **Vector3** gazebo::math::Matrix4::GetEulerRotation (unsigned int *solution_number* = 1) const

Get the rotation as a Euler angles.

Returns

the rotation

10.98.3.3 **Quaternion** gazebo::math::Matrix4::GetRotation () const

Get the rotation as a quaternion.

Returns

the rotation

10.98.3.4 **Vector3** gazebo::math::Matrix4::GetTranslation () const

Get the translational values as a **Vector3** (p. 902).

Returns

x,y,z

10.98.3.5 **Matrix4** gazebo::math::Matrix4::Inverse () const

Return the inverse matrix.

10.98.3.6 **bool** gazebo::math::Matrix4::IsAffine () const

Return true if the matrix is affine.

Returns

true if the matrix is affine, false otherwise

10.98.3.7 **Matrix4** gazebo::math::Matrix4::operator* (const **Matrix4** & *_mat*) const

Multiplication operator.

Parameters

<i>_mat</i>	Incoming matrix
-------------	-----------------

Returns

This matrix * *_mat*

10.98.3.8 **Matrix4** gazebo::math::Matrix4::operator* (const **Matrix3** & *_mat*) const

Multiplication operator.

Parameters

<i>_mat</i>	Incoming matrix
-------------	-----------------

Returns

This matrix * *_mat*

10.98.3.9 **Vector3** gazebo::math::Matrix4::operator* (const **Vector3** & *_vec*) const

Multiplication operator.

Parameters

<i>_vec</i>	Vector3 (p. 902)
-------------	-------------------------

Returns

Resulting vector from multiplication

10.98.3.10 `Matrix4& gazebo::math::Matrix4::operator=(const Matrix4 & _mat)`

Equal operator.

this = _mat

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

itself

10.98.3.11 `const Matrix4& gazebo::math::Matrix4::operator=(const Matrix3 & _mat)`

Equal operator for 3x3 matrix.

Parameters

<code>_mat</code>	Incoming matrix
-------------------	-----------------

Returns

itself

10.98.3.12 `bool gazebo::math::Matrix4::operator==(const Matrix4 & _m) const`

Equality operator.

Parameters

<code>in</code>	<code>_m</code>	Matrix3 (p. 498) to test
-----------------	-----------------	---------------------------------

Returns

true if the 2 matrices are equal (using the tolerance 1e-6), false otherwise

10.98.3.13 `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.

10.98.3.14 `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.

10.98.3.15 `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

<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

10.98.3.16 `void gazebo::math::Matrix4::SetScale (const Vector3 & _s)`

Set the scale.

Parameters

<code>in</code>	<code>_s</code>	scale
-----------------	-----------------	-------

10.98.3.17 `void gazebo::math::Matrix4::SetTranslate (const Vector3 & _t)`

Set the translational values [(0, 3) (1, 3) (2, 3)].

Parameters

<code>in</code>	<code>_t</code>	Values to set
-----------------	-----------------	---------------

10.98.3.18 `Vector3 gazebo::math::Matrix4::TransformAffine (const Vector3 & _v) const`

Perform an affine transformation.

Parameters

<code>_v</code>	Vector3 (p. 902) value for the transformation
-----------------	--

Returns

The result of the transformation

10.98.4 Friends And Related Function Documentation

10.98.4.1 `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

10.98.5 Member Data Documentation

10.98.5.1 `const Matrix4 gazebo::math::Matrix4::IDENTITY` [static]

Identity matrix.

10.98.5.2 `double gazebo::math::Matrix4::m[4][4]` [protected]

The 4x4 matrix.

Referenced by `operator[]()`.

10.98.5.3 `const Matrix4 gazebo::math::Matrix4::ZERO` [static]

Zero matrix.

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

- **Matrix4.hh**

10.99 gazebo::common::Mesh Class Reference

A 3D mesh.

```
#include <Mesh.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.

10.99.1 Detailed Description

A 3D mesh.

10.99.2 Constructor & Destructor Documentation

10.99.2.1 gazebo::common::Mesh::Mesh ()

Constructor.

10.99.2.2 virtual gazebo::common::Mesh::~~Mesh () [virtual]

Destructor.

10.99.3 Member Function Documentation

10.99.3.1 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

10.99.3.2 void gazebo::common::Mesh::AddSubMesh (**SubMesh** * _child)

Add a submesh mesh.

The **Mesh** (p. 509) object takes ownership of the submesh.

Parameters

in	<i>_child</i>	the submesh
----	---------------	-------------

10.99.3.3 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

10.99.3.4 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
----	----------------	------------------------------

10.99.3.5 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

10.99.3.6 unsigned int gazebo::common::Mesh::GetIndexCount () const

Return the number of indices.

Returns

the count

10.99.3.7 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

10.99.3.8 unsigned int gazebo::common::Mesh::GetMaterialCount () const

Get the number of materials.

Returns

the count

10.99.3.9 math::Vector3 gazebo::common::Mesh::GetMax () const

Get the maximum X, Y, Z values.

Returns

the upper bounds of the bounding box

10.99.3.10 math::Vector3 gazebo::common::Mesh::GetMin () const

Get the minimum X, Y, Z values.

Returns

the lower bounds of the bounding box

10.99.3.11 std::string gazebo::common::Mesh::GetName () const

Get the name of this mesh.

Returns

the name

10.99.3.12 unsigned int gazebo::common::Mesh::GetNormalCount () const

Return the number of normals.

Returns

the count

10.99.3.13 std::string gazebo::common::Mesh::GetPath () const

Get the path which contains the mesh resource.

Returns

the path to the mesh resource

10.99.3.14 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.

10.99.3.15 const SubMesh* gazebo::common::Mesh::GetSubMesh (unsigned int *_i*) const

Get a child mesh.

Parameters

<i>in</i>	<i>_i</i>	the index
-----------	-----------	-----------

Returns

the submesh. An exception is thrown if the index is out of bounds

10.99.3.16 unsigned int gazebo::common::Mesh::GetSubMeshCount () const

Get the number of children.

Returns

the count

10.99.3.17 unsigned int gazebo::common::Mesh::GetTexCoordCount () const

Return the number of texture coordinates.

Returns

the count

10.99.3.18 unsigned int gazebo::common::Mesh::GetVertexCount () const

Return the number of vertices.

Returns

the count

10.99.3.19 bool gazebo::common::Mesh::HasSkeleton () const

Return true if mesh is attached to a skeleton.

10.99.3.20 void gazebo::common::Mesh::RecalculateNormals ()

Recalculate all the normals of each face defined by three indices.

10.99.3.21 void gazebo::common::Mesh::Scale (double *_factor*)

Scale all vertices by *_factor*.

Parameters

<i>_factor</i>	Scaling factor
----------------	----------------

10.99.3.22 void gazebo::common::Mesh::SetName (const std::string & *_n*)

Set the name of this mesh.

Parameters

<i>in</i>	<i>_n</i>	the name to set
-----------	-----------	-----------------

10.99.3.23 void gazebo::common::Mesh::SetPath (const std::string & *_path*)

Set the path which contains the mesh resource.

Parameters

<i>in</i>	<i>_path</i>	the file path
-----------	--------------	---------------

10.99.3.24 void gazebo::common::Mesh::SetSkeleton (Skeleton * *_skel*)

Set the mesh skeleton.

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

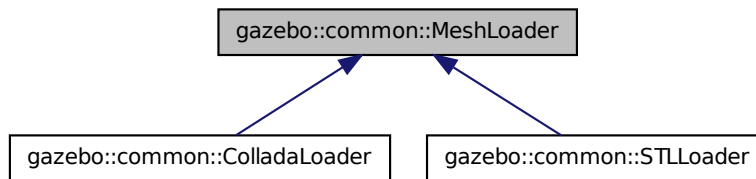
- **Mesh.hh**

10.100 gazebo::common::MeshLoader Class Reference

Base class for loading meshes.

```
#include <MeshLoader.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.

10.100.1 Detailed Description

Base class for loading meshes.

10.100.2 Constructor & Destructor Documentation

10.100.2.1 gazebo::common::MeshLoader::MeshLoader ()

Constructor.

10.100.2.2 virtual gazebo::common::MeshLoader::~~MeshLoader () [virtual]

Destructor.

10.100.3 Member Function Documentation

10.100.3.1 virtual Mesh* gazebo::common::MeshLoader::Load (const std::string & filename) [pure virtual]

Load a 3D mesh.

Parameters

<i>in</i>	<i>the</i>	path to the mesh
-----------	------------	------------------

Returns

a pointer to the created mesh

Implemented in `gazebo::common::ColladaLoader` (p. 262), and `gazebo::common::STLLoader` (p. 817).

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

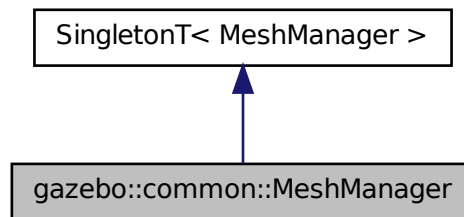
- `MeshLoader.hh`

10.101 gazebo::common::MeshManager Class Reference

Maintains and manages all meshes.

```
#include <MeshManager.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

10.101.1 Detailed Description

Maintains and manages all meshes.

10.101.2 Member Function Documentation

10.101.2.1 void gazebo::common::MeshManager::AddMesh (**Mesh** * _mesh)

Add a mesh to the manager.

This **MeshManager** (p. 516) takes ownership of the mesh and will destroy it. See `~MeshManager`.

Parameters

in	<i>the</i>	mesh to add.
----	------------	--------------

10.101.2.2 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 dimentionions of eah side in meter
in	<i>_uvCoords</i>	the texture coordinates

10.101.2.3 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

10.101.2.4 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

10.101.2.5 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

10.101.2.6 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

10.101.2.7 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

10.101.2.8 void gazebo::common::MeshManager::CreateSphere (const std::string & *_name*, float *_radius*, int *_rings*, int *_segments*)

Create a sphere mesh.

Parameters

in	<i>_name</i>	the name of the mesh
in	<i>_radius</i>	radius of the sphere in meter
in	<i>_rings</i>	number of circles on th y axis
in	<i>_segments</i>	number of segment per circle

10.101.2.9 void gazebo::common::MeshManager::CreateTube (const std::string & *_name*, float *_innerRadius*, float *_outterRadius*, float *_height*, int *_rings*, int *_segments*)

Create a tube mesh.

Generates rings inside and outside the cylinder Needs at least two rings and 3 segments

Parameters

in	<i>_name</i>	the name of the new mesh
in	<i>_innerRadius</i>	the inner radius of the tube in the x y plane
in	<i>_outterRadius</i>	the outer radius of the tube in the x y plane
in	<i>_height</i>	the height along z
in	<i>_rings</i>	the number of circles along the height
in	<i>_segments</i>	the number of segment per circle

10.101.2.10 void gazebo::common::MeshManager::GenSphericalTexCoord (const Mesh * *_mesh*, math::Vector3 *_center*)

generate spherical texture coordinates

10.101.2.11 const Mesh* gazebo::common::MeshManager::GetMesh (const std::string & *_name*) const

Get a mesh by name.

Parameters

in	<i>_name</i>	the name of the mesh to look for
----	--------------	----------------------------------

Returns

the mesh or NULL if not found

10.101.2.12 `void gazebo::common::MeshManager::GetMeshAABB (const Mesh * _mesh, math::Vector3 & _center, math::Vector3 & _min_xyz, math::Vector3 & _max_xyz)`

Get mesh aabb and center.

Parameters

in	<i>_mesh</i>	the mesh
out	<i>_center</i>	the AAB center position
out	<i>_min_xyz</i>	the bounding box minimum
out	<i>_max_xyz</i>	the bounding box maximum

10.101.2.13 `bool gazebo::common::MeshManager::HasMesh (const std::string & _name) const`

Return true if the mesh exists.

Parameters

in	<i>_name</i>	the name of the mesh
----	--------------	----------------------

10.101.2.14 `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

10.101.2.15 `const Mesh* gazebo::common::MeshManager::Load (const std::string & _filename)`

Load a mesh from a file.

Parameters

in	<i>_filename</i>	the path to the mesh
----	------------------	----------------------

Returns

a pointer to the created mesh

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

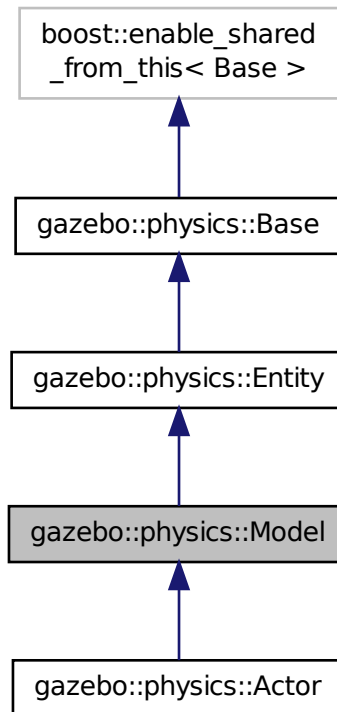
- **MeshManager.hh**

10.102 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

*Construct and return a vector of **Link** (p. 454)'s in this model.*

 - virtual **math::Box GetBoundingBox** () const

Get the size of the bounding box.

 - **JointPtr GetJoint** (unsigned int _index) const

Get a joint by index.

 - **JointPtr GetJoint** (const std::string &_name)

Get a joint.

 - unsigned int **GetJointCount** () const

Get the number of joints.

 - **LinkPtr GetLink** (const std::string &_name="canonical") const

Get a link by name.

 - **LinkPtr GetLink** (unsigned int _index) const

Get a child link by index.

 - **LinkPtr GetLinkById** (unsigned int _id) const

Get a link by id.

 - 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.

 - **ModelState GetState** ()

Get the current model state.

 - 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. 454), 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. 423) Animation.*
- void **SetJointPosition** (const std::string &_jointName, double _position)
 - Set the positions of a **Joint** (p. 423) 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. 521) by specifying desired Pose of a **Link** (p. 454) within the **Model** (p. 521).*
- void **SetLinkWorldPose** (const **math::Pose** &_pose, const **LinkPtr** &_link)
 - Set the Pose of the entire **Model** (p. 521) by specifying desired Pose of a **Link** (p. 454) within the **Model** (p. 521).*
- 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. 524).
- `std::vector< math::Pose > attachedModelsOffset`
used by `Model::AttachStaticModel` (p. 524).

Additional Inherited Members

10.102.1 Detailed Description

A model is a collection of links, joints, and plugins.

10.102.2 Constructor & Destructor Documentation

10.102.2.1 `gazebo::physics::Model::Model (BasePtr _parent) [explicit]`

Constructor.

Parameters

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

10.102.2.2 `virtual gazebo::physics::Model::~~Model () [virtual]`

Destructor.

10.102.3 Member Function Documentation

10.102.3.1 `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. 521), which is a **Model** (p. 521) that has been marked as static (no physics simulation), and attaches it to this **Model** (p. 521) with a given offset.

This function is useful when you want to simulate a grasp of a static object, or move a static object around using a dynamic model.

If you are in doubt, do not use this function.

Parameters

<code>in</code>	<code>_model</code>	Pointer to the static model.
<code>in</code>	<code>_offset</code>	Offset, relative to this Model (p. 521), to place <code>_model</code> .

10.102.3.2 `void gazebo::physics::Model::DetachStaticModel (const std::string & _model)`

Detach a static model from this model.

Parameters

in	_model	Name of an attached static model to remove.
----	--------	---

See Also

Model::AttachStaticModel (p. 524).

10.102.3.3 void gazebo::physics::Model::FillModelMsg (msgs::Model & _msg)

DEPRECATED.

10.102.3.4 void gazebo::physics::Model::FillMsg (msgs::Model & _msg)

Fill a model message.

Parameters

in	_msg	Message to fill using this model's data.
----	------	--

10.102.3.5 virtual void gazebo::physics::Model::Fini () [virtual]

Finalize the model.

Reimplemented from **gazebo::physics::Entity** (p. 342).

Reimplemented in **gazebo::physics::Actor** (p. 124).

10.102.3.6 Link_V gazebo::physics::Model::GetAllLinks () const

Construct and return a vector of **Link** (p. 454)'s in this model.

Note this constructs the vector of **Link** (p. 454)'s on the fly, could be costly.

Returns

A vector of **Link** (p. 454)'s in this model.

10.102.3.7 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. 342).

10.102.3.8 JointPtr gazebo::physics::Model::GetJoint (unsigned int _index) const

Get a joint by index.

Parameters

<code>in</code>	<code>_index</code>	Index of the joint.
-----------------	---------------------	---------------------

Returns

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

10.102.3.9 JointPtr gazebo::physics::Model::GetJoint (const std::string & *_name*)

Get a joint.

Parameters

<code>in</code>	<code>_name</code>	The name of the joint, specified in the world file.
-----------------	--------------------	---

Returns

Pointer to the joint, NULL if the name is invalid.

10.102.3.10 unsigned int gazebo::physics::Model::GetJointCount () const

Get the number of joints.

Returns

Get the number of joints.

10.102.3.11 LinkPtr gazebo::physics::Model::GetLink (const std::string & *_name* = "canonical") const

Get a link by name.

Parameters

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

Returns

Pointer to the link, NULL if the name is invalid.

10.102.3.12 LinkPtr gazebo::physics::Model::GetLink (unsigned int *_index*) const

Get a child link by index.

Parameters

<code>in</code>	<code>_index</code>	Index of the link.
-----------------	---------------------	--------------------

Returns

Point to the link, NULL if the index is invalid.

10.102.3.13 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.

10.102.3.14 virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularAccel () const [virtual]

Get the angular acceleration of the entity.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 343).

10.102.3.15 virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularVel () const [virtual]

Get the angular velocity of the entity.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 343).

10.102.3.16 virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearAccel () const [virtual]

Get the linear acceleration of the entity.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.102.3.17 virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearVel () const [virtual]

Get the linear velocity of the entity.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.102.3.18 `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. 153).

Reimplemented in `gazebo::physics::Actor` (p. 124).

10.102.3.19 `ModelState gazebo::physics::Model::GetState ()`

Get the current model state.

Returns

The current model state.

10.102.3.20 `virtual math::Vector3 gazebo::physics::Model::GetWorldAngularAccel () const [virtual]`

Get the angular acceleration of the entity in the world frame.

Returns

`math::Vector3` (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from `gazebo::physics::Entity` (p. 344).

10.102.3.21 `virtual math::Vector3 gazebo::physics::Model::GetWorldAngularVel () const [virtual]`

Get the angular velocity of the entity in the world frame.

Returns

`math::Vector3` (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from `gazebo::physics::Entity` (p. 344).

10.102.3.22 `virtual math::Vector3 gazebo::physics::Model::GetWorldLinearAccel () const [virtual]`

Get the linear acceleration of the entity in the world frame.

Returns

`math::Vector3` (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from `gazebo::physics::Entity` (p. 345).

10.102.3.23 `virtual math::Vector3 gazebo::physics::Model::GetWorldLinearVel () const [virtual]`

Get the linear velocity of the entity in the world frame.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 345).

10.102.3.24 `virtual void gazebo::physics::Model::Init () [virtual]`

Initialize the model.

Reimplemented from **gazebo::physics::Base** (p. 153).

Reimplemented in **gazebo::physics::Actor** (p. 124).

10.102.3.25 `void gazebo::physics::Model::Load (sdf::ElementPtr _sdf) [virtual]`

Load the model.

Parameters

in	_sdf	SDF parameters to load from.
----	------	------------------------------

Reimplemented from **gazebo::physics::Entity** (p. 346).

10.102.3.26 `void gazebo::physics::Model::LoadPlugins ()`

Load all plugins.

Load all plugins specified in the SDF for the model.

10.102.3.27 `virtual void gazebo::physics::Model::OnPoseChange () [protected],[virtual]`

Callback when the pose of the model has been changed.

Implements **gazebo::physics::Entity** (p. 346).

10.102.3.28 `void gazebo::physics::Model::ProcessMsg (const msgs::Model & _msg)`

Update parameters from a model message.

Parameters

in	_msg	Message to process.
----	------	---------------------

10.102.3.29 `virtual void gazebo::physics::Model::RemoveChild (EntityPtr _child) [virtual]`

Remove a child.

Parameters

in	<code>_child</code>	Remove a child entity.
----	---------------------	------------------------

10.102.3.30 `void gazebo::physics::Model::Reset () [virtual]`

Reset the model.

Reimplemented from `gazebo::physics::Entity` (p. 346).

10.102.3.31 `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
----	-------------------	------------------------------

10.102.3.32 `void gazebo::physics::Model::SetAngularVel (const math::Vector3 & _vel)`

Set the angular velocity of the model, and all its links.

Parameters

in	<code>_vel</code>	The new angular velocity.
----	-------------------	---------------------------

10.102.3.33 `void gazebo::physics::Model::SetAutoDisable (bool _disable)`

Allow the model the auto disable.

This is ignored if the model has joints.

Parameters

in	<code>_disable</code>	If true, the model is allowed to auto disable.
----	-----------------------	--

10.102.3.34 `void gazebo::physics::Model::SetCollideMode (const std::string & _mode)`

This is not implemented in `Link` (p. 454), which means this function doesn't do anything.

Set the collide mode of the model.

Parameters

in	<code>_mode</code>	The collision mode
----	--------------------	--------------------

10.102.3.35 `void gazebo::physics::Model::SetEnabled (bool _enabled)`

Enable all the links in all the models.

Parameters

in	<code>_enabled</code>	True to enable all the links.
----	-----------------------	-------------------------------

10.102.3.36 `void gazebo::physics::Model::SetGravityMode (const bool & _value)`

Set the gravity mode of the model.

Parameters

in	<code>_value</code>	False to turn gravity on for the model.
----	---------------------	---

10.102.3.37 `void gazebo::physics::Model::SetJointAnimation (const std::map< std::string, common::NumericAnimationPtr > _anim, boost::function< void()> _onComplete = NULL)`

Joint (p. 423) 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.

10.102.3.38 `void gazebo::physics::Model::SetJointPosition (const std::string & _jointName, double _position)`

Set the positions of a **Joint** (p. 423) by name.

See Also

JointController::SetJointPosition (p. 439)

Parameters

in	<code>_jointName</code>	Name of the joint to set.
in	<code>_position</code>	Position to set the joint to.

10.102.3.39 `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. 440).

Parameters

in	<code>_jointPositions</code>	Map of joint names to their positions.
----	------------------------------	--

10.102.3.40 `void gazebo::physics::Model::SetLaserRetro (const float _retro)`

Set the laser retro reflectiveness of the model.

Parameters

in	_retro	Retro reflectance value.
----	--------	--------------------------

10.102.3.41 `void gazebo::physics::Model::SetLinearAccel (const math::Vector3 & _vel)`

Set the linear acceleration of the model, and all its links.

Parameters

in	_vel	The new linear acceleration.
----	------	------------------------------

10.102.3.42 `void gazebo::physics::Model::SetLinearVel (const math::Vector3 & _vel)`

Set the linear velocity of the model, and all its links.

Parameters

in	_vel	The new linear velocity.
----	------	--------------------------

10.102.3.43 `void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & _pose, std::string _linkName)`

Set the Pose of the entire **Model** (p. 521) by specifying desired Pose of a **Link** (p. 454) within the **Model** (p. 521). Doing so, keeps the configuration of the **Model** (p. 521) unchanged, i.e. all **Joint** (p. 423) angles are unchanged.

Parameters

in	_pose	Pose to set the link to.
in	_linkName	Name of the link to set.

10.102.3.44 `void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & _pose, const LinkPtr & _link)`

Set the Pose of the entire **Model** (p. 521) by specifying desired Pose of a **Link** (p. 454) within the **Model** (p. 521). Doing so, keeps the configuration of the **Model** (p. 521) unchanged, i.e. all **Joint** (p. 423) angles are unchanged.

Parameters

in	_pose	Pose to set the link to.
in	_link	Pointer to the link to set.

10.102.3.45 `void gazebo::physics::Model::SetState (const ModelState & _state)`

Set the current model state.

Parameters

in	_state	State (p. 813) to set the model to.
----	--------	-------------------------------------

10.102.3.46 virtual void gazebo::physics::Model::StopAnimation () [virtual]

Stop the current animations.

Reimplemented from **gazebo::physics::Entity** (p. 348).

10.102.3.47 void gazebo::physics::Model::Update () [virtual]

Update the model.

Reimplemented from **gazebo::physics::Base** (p. 157).

10.102.3.48 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. 348).

Reimplemented in **gazebo::physics::Actor** (p. 125).

10.102.4 Member Data Documentation

10.102.4.1 std::vector<ModelPtr> gazebo::physics::Model::attachedModels [protected]

Used by **Model::AttachStaticModel** (p. 524).

10.102.4.2 std::vector<math::Pose> gazebo::physics::Model::attachedModelsOffset [protected]

used by **Model::AttachStaticModel** (p. 524).

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

- **Model.hh**

10.103 gazebo::common::ModelDatabase Class Reference

Connects to model database, and has utility functions to find models.

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

Static Public Member Functions

- static void **DownloadDependencies** (const std::string &_path)

Download all dependencies for a give model path.

- static std::string **GetManifest** (const std::string &_uri)
Return the manifest.xml file as a string.
- static std::string **GetModelFile** (const std::string &_uri)
Get a model's SDF file based on a URI.
- static std::string **GetModelName** (const std::string &_uri)
Get the name of a model based on a URI.
- static std::string **GetModelPath** (const std::string &_uri)
Get the local path to a model.
- static std::map< std::string, std::string > **GetModels** ()
Returns the dictionary of all the model names.
- static std::string **GetURI** ()
Returns the the global model database URI.
- static bool **HasModel** (const std::string &_modelName)
Returns true if the model exists on the database.

10.103.1 Detailed Description

Connects to model database, and has utility functions to find models.

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

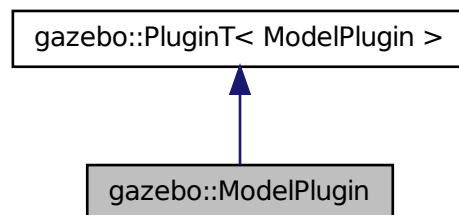
- **ModelDatabase.hh**

10.104 gazebo::ModelPlugin Class Reference

A plugin with access to **physics::Model** (p. 521).

```
#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

10.104.1 Detailed Description

A plugin with access to **physics::Model** (p. 521).

See [reference](#).

10.104.2 Constructor & Destructor Documentation

10.104.2.1 gazebo::ModelPlugin::ModelPlugin () [inline]

Constructor.

References [gazebo::MODEL_PLUGIN](#), and [gazebo::PluginT< ModelPlugin >::type](#).

10.104.2.2 virtual gazebo::ModelPlugin::~~ModelPlugin () [inline],[virtual]

Destructor.

10.104.3 Member Function Documentation

10.104.3.1 virtual void gazebo::ModelPlugin::Init () [inline],[virtual]

Override this method for custom plugin initialization behavior.

10.104.3.2 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

<code>_model</code>	Pointer the Model
<code>_sdf</code>	Pointer the the SDF element of the plugin.

10.104.3.3 `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:

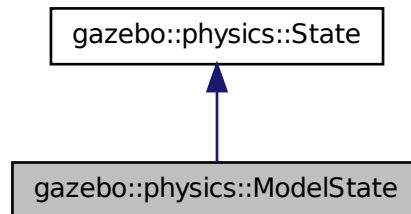
- `common/Plugin.hh`

10.105 gazebo::physics::ModelState Class Reference

Store state information of a `physics::Model` (p. 521) object.

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

Inheritance diagram for `gazebo::physics::ModelState`:



Public Member Functions

- **ModelState** ()
Default constructor.
- **ModelState** (**ModelPtr** _model)
Constructor.
- virtual **~ModelState** ()
Destructor.
- void **FillStateSDF** (**sdf::ElementPtr** _elem)
*Fill a **State** (p. 813) SDF element with state info.*
- **JointState** **GetJointState** (unsigned int _index) const
*Get a **Joint** (p. 423) state.*
- **JointState** **GetJointState** (const std::string &_jointName) const
*Get a **Joint** (p. 423) state by **Joint** (p. 423) name.*
- unsigned int **GetJointStateCount** () const
Get the number of 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. 454) name.

- unsigned int **GetLinkStateCount** () const

Get the number of link states.

- **math::Pose GetPose** () const

Get the stored model pose.

- virtual void **Load** (sdf::ElementPtr _elem)

Load state from SDF element.

- void **UpdateModelSDF** (sdf::ElementPtr _elem)

Update a **Model** (p. 521) SDF element with this state info.

Additional Inherited Members

10.105.1 Detailed Description

Store state information of a **physics::Model** (p. 521) object.

This class captures the entire state of a **Model** (p. 521) at one specific time during a simulation run.

State (p. 813) of a **Model** (p. 521) includes the state of all its child Links and Joints.

10.105.2 Constructor & Destructor Documentation

10.105.2.1 gazebo::physics::ModelState::ModelState ()

Default constructor.

10.105.2.2 gazebo::physics::ModelState::ModelState (ModelPtr _model) [explicit]

Constructor.

Build a **ModelState** (p. 536) from an existing **Model** (p. 521).

Parameters

in	_model	Pointer to the model from which to gather state info.
----	--------	---

10.105.2.3 virtual gazebo::physics::ModelState::~~ModelState () [virtual]

Destructor.

10.105.3 Member Function Documentation

10.105.3.1 void gazebo::physics::ModelState::FillStateSDF (sdf::ElementPtr _elem)

Fill a **State** (p. 813) SDF element with state info.

Stored state information into an SDF::Element pointer.

Parameters

in	_elem	Pointer to the SDF::Element which receives the data.
----	-------	--

10.105.3.2 JointState gazebo::physics::ModelState::GetJointState (unsigned int _index) const

Get a **Joint** (p. 423) state.

Return a **JointState** (p. 442) based on a index, where index is between 0...**ModelState::GetJointStateCount()** (p. 538).

Parameters

in	_index	Index of a JointState (p. 442).
----	--------	--

Returns

State (p. 813) of a **Joint** (p. 423).

10.105.3.3 JointState gazebo::physics::ModelState::GetJointState (const std::string & _jointName) const

Get a **Joint** (p. 423) state by **Joint** (p. 423) name.

Searches through all JointStates. Returns the **JointState** (p. 442) with the matching name, if any.

Parameters

in	_jointName	Name of the JointState (p. 442).
----	------------	---

Returns

State (p. 813) of the **Joint** (p. 423).

10.105.3.4 unsigned int gazebo::physics::ModelState::GetJointStateCount () const

Get the number of joint states.

Returns the number of JointStates recorded.

Returns

Number of JointStates.

10.105.3.5 LinkState gazebo::physics::ModelState::GetLinkState (unsigned int _index) const

Get a link state.

Get a **Link** (p. 454) **State** (p. 813) based on an index, where index is in the range of 0...**ModelState::GetLinkStateCount** (p. 539)

Parameters

in	_index	Index of the LinkState (p. 473)
----	--------	--

Returns

State (p. 813) of the **Link** (p. 454)

10.105.3.6 `LinkState gazebo::physics::ModelState::GetLinkState (const std::string & _linkName) const`

Get a link state by **Link** (p. 454) name.

Searches through all LinkStates. Returns the **LinkState** (p. 473) with the matching name, if any.

Parameters

<code>in</code>	<code>_linkName</code>	Name of the LinkState (p. 473)
-----------------	------------------------	---------------------------------------

Returns

State (p. 813) of the **Link** (p. 454).

10.105.3.7 `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. 473) recorded

10.105.3.8 `math::Pose gazebo::physics::ModelState::GetPose () const`

Get the stored model pose.

Returns

The **math::Pose** (p. 677) of the **Model** (p. 521)

10.105.3.9 `virtual void gazebo::physics::ModelState::Load (sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Load **ModelState** (p. 536) information from stored data in and SDF::Element

Parameters

<code>in</code>	<code>_elem</code>	Pointer to the SDF::Element containing state info.
-----------------	--------------------	--

Implements **gazebo::physics::State** (p. 815).

10.105.3.10 `void gazebo::physics::ModelState::UpdateModelSDF (sdf::ElementPtr _elem)`

Update a **Model** (p. 521) SDF element with this state info.

Set the values in a **Model** (p. 521)'s SDF::Element with the information. stored in this instance.

Parameters

in	_elem	Pointer to a Models's SDF::Element.
----	-------	-------------------------------------

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

- **ModelState.hh**

10.106 gazebo::common::MouseEvent Class Reference

Generic description of a mouse event.

```
#include <MouseEvent.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.

10.106.1 Detailed Description

Generic description of a mouse event.

10.106.2 Member Enumeration Documentation

10.106.2.1 enum gazebo::common::MouseEvent::Buttons

Standard mouse buttons enumeration.

Enumerator:

NO_BUTTON

LEFT

MIDDLE

RIGHT

10.106.2.2 enum gazebo::common::MouseEvent::EventType

Mouse event types enumeration.

Enumerator:

NO_EVENT

MOVE

PRESS

RELEASE

SCROLL

10.106.3 Constructor & Destructor Documentation

10.106.3.1 gazebo::common::MouseEvent::MouseEvent() [inline]

Constructor.

10.106.4 Member Data Documentation

10.106.4.1 `bool gazebo::common::MouseEvent::alt`

Alt key press flag.

10.106.4.2 `unsigned int gazebo::common::MouseEvent::button`

The button which caused the event.

10.106.4.3 `unsigned int gazebo::common::MouseEvent::buttons`

State of the buttons when the event was generated.

10.106.4.4 `bool gazebo::common::MouseEvent::control`

Control key press flag.

10.106.4.5 `bool gazebo::common::MouseEvent::dragging`

Flag for mouse drag motion.

10.106.4.6 `float gazebo::common::MouseEvent::moveScale`

Scaling factor.

10.106.4.7 `math::Vector2i gazebo::common::MouseEvent::pos`

Mouse pointer position on the screen.

10.106.4.8 `math::Vector2i gazebo::common::MouseEvent::pressPos`

Position of button press.

10.106.4.9 `math::Vector2i gazebo::common::MouseEvent::prevPos`

Previous position.

10.106.4.10 `math::Vector2i gazebo::common::MouseEvent::scroll`

Scroll position.

10.106.4.11 `bool gazebo::common::MouseEvent::shift`

Shift key press flag.

10.106.4.12 EventType gazebo::common::MouseEvent::type

Event type.

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

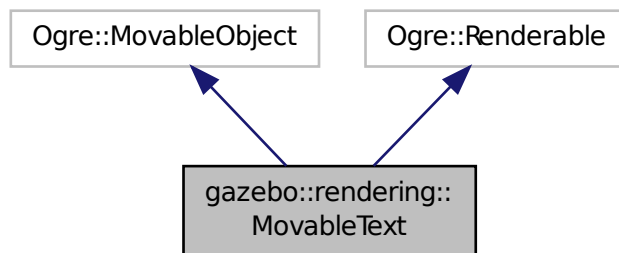
- **MouseEvent.hh**

10.107 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

10.107.1 Detailed Description

Movable text.

10.107.2 Member Enumeration Documentation

10.107.2.1 enum gazebo::rendering::MovableText::HorizAlign

Horizontal alignment.

Enumerator:

H_LEFT Left alignment.

H_CENTER Center alignment.

10.107.2.2 enum gazebo::rendering::MovableText::VertAlign

vertical alignment

Enumerator:

V_BELOW Align below.

V_ABOVE Align above.

10.107.3 Constructor & Destructor Documentation

10.107.3.1 gazebo::rendering::MovableText::MovableText ()

Constructor.

10.107.3.2 virtual gazebo::rendering::MovableText::~~MovableText () [virtual]

Destructor.

10.107.4 Member Function Documentation

10.107.4.1 void gazebo::rendering::MovableText::_setupGeometry () [protected]

10.107.4.2 void gazebo::rendering::MovableText::_updateColors () [protected]

10.107.4.3 math::Box gazebo::rendering::MovableText::GetAABB ()

Get the axis aligned bounding box of the text.

Returns

The axis aligned bounding box.

10.107.4.4 float gazebo::rendering::MovableText::GetBaseline () const

Get the baseline height.

Returns

Baseline height

10.107.4.5 `float gazebo::rendering::MovableText::getBoundingRadius () const` [protected]

10.107.4.6 `float gazebo::rendering::MovableText::GetCharHeight () const`

Set the height of a characters return Height of the characters.

10.107.4.7 `const common::Color& gazebo::rendering::MovableText::GetColor () const`

Get the text color.

Returns

Textuer color.

10.107.4.8 `const std::string& gazebo::rendering::MovableText::GetFont () const`

Get the font.

Returns

The font name

10.107.4.9 `const Ogre::LightList& gazebo::rendering::MovableText::getLights (void) const` [protected]

10.107.4.10 `const Ogre::MaterialPtr& gazebo::rendering::MovableText::getMaterial (void) const` [protected]

10.107.4.11 `void gazebo::rendering::MovableText::getRenderOperation (Ogre::RenderOperation & op)` [protected]

10.107.4.12 `bool gazebo::rendering::MovableText::GetShowOnTop () const`

True = text is displayed on top.

Returns

True if `MovableText::SetShownOnTop(true)` was called.

10.107.4.13 `float gazebo::rendering::MovableText::GetSpaceWidth () const`

Get the width of a space.

Returns

Space width

10.107.4.14 `float gazebo::rendering::MovableText::getSquaredViewDepth (const Ogre::Camera * cam) const` [protected]

10.107.4.15 `const std::string& gazebo::rendering::MovableText::GetText () const`

Get the displayed text.

Returns

The displayed text.

10.107.4.16 `void gazebo::rendering::MovableText::getWorldTransforms (Ogre::Matrix4 * xform) const` [protected]

10.107.4.17 `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	<code><i>_name</i></code>	Name of the text object
in	<code><i>_text</i></code>	Text to render
in	<code><i>_fontName</i></code>	Font to use
in	<code><i>_charHeight</i></code>	Height of the characters
in	<code><i>_color</i></code>	Text color

10.107.4.18 `void gazebo::rendering::MovableText::SetBaseline (float _height)`

Set the baseline height of the text.

Parameters

in	<code><i>_height</i></code>	Baseline height
----	-----------------------------	-----------------

10.107.4.19 `void gazebo::rendering::MovableText::SetCharHeight (float _height)`

Set the height of a character.

Parameters

in	<code><i>_height</i></code>	Height of the characters.
----	-----------------------------	---------------------------

10.107.4.20 `void gazebo::rendering::MovableText::SetColor (const common::Color & _color)`

Set the text color.

Parameters

in	<code><i>_color</i></code>	Text color.
----	----------------------------	-------------

10.107.4.21 `void gazebo::rendering::MovableText::SetFontName (const std::string & _font)`

Set the font.

Parameters

in	<i>_font</i>	Name of the font
----	--------------	------------------

10.107.4.22 `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.
----	--------------	---

10.107.4.23 `void gazebo::rendering::MovableText::SetSpaceWidth (float _width)`

Set the width of a space.

Parameters

in	<i>_width</i>	space width
----	---------------	-------------

10.107.4.24 `void gazebo::rendering::MovableText::SetText (const std::string & _text)`

Set the text to display.

Parameters

in	<i>The</i>	text to display.
----	------------	------------------

10.107.4.25 `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

10.107.4.26 `void gazebo::rendering::MovableText::Update ()`

Update the text.

10.107.4.27 `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:

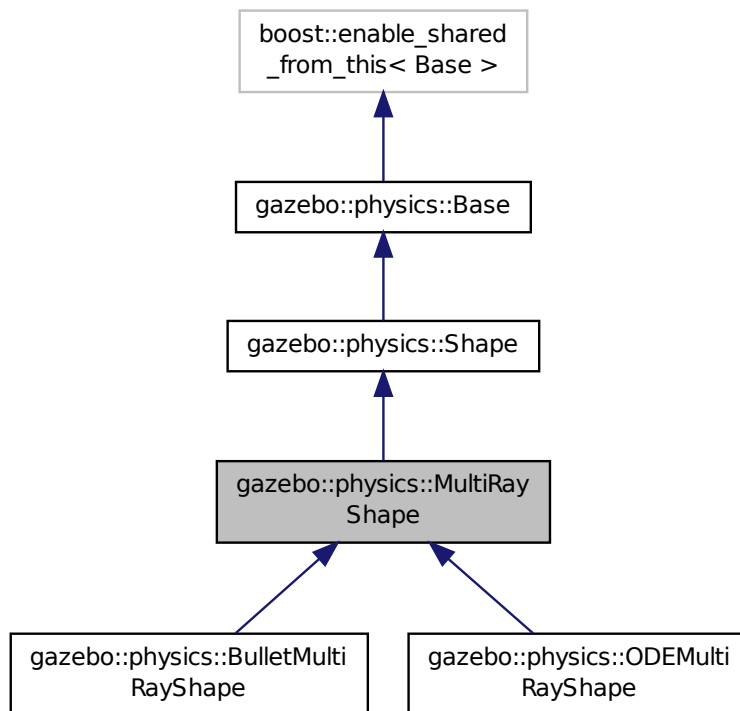
- **MovableText.hh**

10.108 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

10.108.1 Detailed Description

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

10.108.2 Constructor & Destructor Documentation

10.108.2.1 `gazebo::physics::MultiRayShape::MultiRayShape (CollisionPtr _parent) [explicit]`

Constructor.

Parameters

in	<i>_parent</i>	Parent collision shape.
----	----------------	-------------------------

10.108.2.2 `virtual gazebo::physics::MultiRayShape::~~MultiRayShape () [virtual]`

Destructor.

10.108.3 Member Function Documentation

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

Add a ray to the collision.

Parameters

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

Reimplemented in `gazebo::physics::ODEMultiRayShape` (p. 609), and `gazebo::physics::BulletMultiRayShape` (p. 202).

10.108.3.2 `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`.

10.108.3.3 `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`.

10.108.3.4 `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. 781).

10.108.3.5 `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.

10.108.3.6 math::Angle gazebo::physics::MultiRayShape::GetMaxAngle () const

Get the maximum angle.

Returns

Maximum angle of ray scan.

10.108.3.7 double gazebo::physics::MultiRayShape::GetMaxRange () const

Get the maximum range.

Returns

Maximum range of all the rays.

10.108.3.8 math::Angle gazebo::physics::MultiRayShape::GetMinAngle () const

Get the minimum angle.

Returns

Minimum angle of ray scan.

10.108.3.9 double gazebo::physics::MultiRayShape::GetMinRange () const

Get the minimum range.

Returns

Minimum range of all the rays.

10.108.3.10 double gazebo::physics::MultiRayShape::GetRange (int *_index*)

Get detected range for a ray.

Parameters

<i>in</i>	<i>_index</i>	Index of the ray.
-----------	---------------	-------------------

Returns

Returns DBL_MAX for no detection.

10.108.3.11 `double gazebo::physics::MultiRayShape::GetResRange () const`

Get the range resolution.

Returns

Range resolution of all the rays.

10.108.3.12 `double gazebo::physics::MultiRayShape::GetRetro (int _index)`

Get detected retro (intensity) value for a ray.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the ray.
-----------------	----------------------------	-------------------

Returns

Retro value for the ray.

10.108.3.13 `int gazebo::physics::MultiRayShape::GetSampleCount () const`

Get the horizontal sample count.

Returns

Horizontal sample count.

10.108.3.14 `double gazebo::physics::MultiRayShape::GetScanResolution () const`

Get the horizontal resolution.

Returns

Horizontal resolution.

10.108.3.15 `math::Angle gazebo::physics::MultiRayShape::GetVerticalMaxAngle () const`

Get the vertical max angle.

Returns

Vertical max angle.

10.108.3.16 `math::Angle gazebo::physics::MultiRayShape::GetVerticalMinAngle () const`

Get the vertical min angle.

Returns

Vertical min angle.

10.108.3.17 `int gazebo::physics::MultiRayShape::GetVerticalSampleCount () const`

Get the vertical sample count.

Returns

Verical sample count.

10.108.3.18 `double gazebo::physics::MultiRayShape::GetVerticalScanResolution () const`

Get the vertical range resolution.

Returns

Vertical range resolution.

10.108.3.19 `virtual void gazebo::physics::MultiRayShape::Init () [virtual]`

Init the shape.

Implements **gazebo::physics::Shape** (p. 781).

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

This function is not implemented.

Update the ray based on a message.

Parameters

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

Implements **gazebo::physics::Shape** (p. 782).

10.108.3.21 `void gazebo::physics::MultiRayShape::Update () [virtual]`

Update the ray collisions.

Reimplemented from **gazebo::physics::Base** (p. 157).

10.108.3.22 `virtual void gazebo::physics::MultiRayShape::UpdateRays () [protected],[pure virtual]`

Physics engine specific method for updating the rays.

Implemented in **gazebo::physics::BulletMultiRayShape** (p.202), and **gazebo::physics::ODEMultiRayShape** (p. 609).

10.108.4 Member Data Documentation

10.108.4.1 `sdf::ElementPtr gazebo::physics::MultiRayShape::horzElem` [protected]

Horizontal SDF element pointer.

10.108.4.2 `event::EventT<void()> gazebo::physics::MultiRayShape::newLaserScans` [protected]

New laser scans event.

Referenced by `ConnectNewLaserScans()`, and `DisconnectNewLaserScans()`.

10.108.4.3 `math::Pose gazebo::physics::MultiRayShape::offset` [protected]

Pose offset of all the rays.

10.108.4.4 `sdf::ElementPtr gazebo::physics::MultiRayShape::rangeElem` [protected]

Range SDF element pointer.

10.108.4.5 `sdf::ElementPtr gazebo::physics::MultiRayShape::rayElem` [protected]

Ray SDF element pointer.

10.108.4.6 `std::vector<RayShapePtr> gazebo::physics::MultiRayShape::rays` [protected]

Ray data.

10.108.4.7 `sdf::ElementPtr gazebo::physics::MultiRayShape::scanElem` [protected]

Scan SDF element pointer.

10.108.4.8 `sdf::ElementPtr gazebo::physics::MultiRayShape::vertElem` [protected]

Vertical SDF element pointer.

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

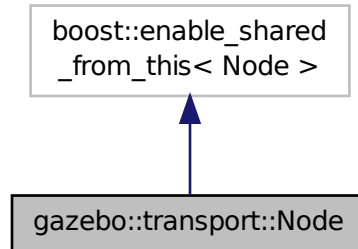
- **MultiRayShape.hh**

10.109 gazebo::transport::Node Class Reference

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

```
#include <Node.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** ()
- 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)
- 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** ()
- 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, and return data on the callback.

- `template<typename M >`

SubscriberPtr Subscribe (const std::string &topic, void(*fp)(const M const *&), bool _latching=false)

Subscribe to a topic, and return data on the callback.

10.109.1 Detailed Description

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

10.109.2 Constructor & Destructor Documentation

10.109.2.1 gazebo::transport::Node::Node ()

Constructor.

10.109.2.2 virtual gazebo::transport::Node::~~Node () [virtual]

Destructor.

10.109.3 Member Function Documentation

10.109.3.1 template<typename M > transport::PublisherPtr gazebo::transport::Node::Advertise (const std::string & topic, unsigned int _queueLimit = 1000, bool _latch = false) [inline]

Advertise a topic.

References DecodeTopicName(), and SingletonT< T >::Instance().

10.109.3.2 std::string gazebo::transport::Node::DecodeTopicName (const std::string & topic)

Decode a topic name.

Referenced by Advertise(), and Subscribe().

10.109.3.3 std::string gazebo::transport::Node::EncodeTopicName (const std::string & topic)

Encode a topic name.

10.109.3.4 void gazebo::transport::Node::Fini ()

10.109.3.5 unsigned int gazebo::transport::Node::GetId () const

Get the unique ID of the node.

10.109.3.6 std::string gazebo::transport::Node::GetMsgType (const std::string & _topic) const

Get the message type for a topic.

10.109.3.7 `std::string gazebo::transport::Node::GetTopicNamespace () const`

Get the topic namespace for this node.

Returns

The namespace

10.109.3.8 `bool gazebo::transport::Node::HandleData (const std::string & _topic, const std::string & _msg)`

10.109.3.9 `void gazebo::transport::Node::Init (const std::string & _space = " ")`

Init the node.

Parameters

<i>space</i>	Set the global namespace of all topics. If left blank, the topic will initialize to the first namespace on the Master (p. 487)
--------------	---

10.109.3.10 `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

<i>in</i>	<i>_topic</i>	Name of the topic to publish data on.
<i>in</i>	<i>_msg</i>	The message to publish.

10.109.3.11 `void gazebo::transport::Node::ProcessIncoming ()`

10.109.3.12 `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

10.109.3.13 `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, and return data on the callback.

References DecodeTopicName(), and SingletonT< T >::Instance().

10.109.3.14 `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, and return data on the callback.

References DecodeTopicName(), and SingletonT< T >::Instance().

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

- **Node.hh**

10.110 gazebo::common::NodeAnimation Class Reference

Node animation.

```
#include <SkeletonAnimation.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

10.110.1 Detailed Description

Node animation.

10.110.2 Constructor & Destructor Documentation

10.110.2.1 gazebo::common::NodeAnimation::NodeAnimation (const std::string & *_name*)

constructor

Parameters

in	<i>_name</i>	the name of the node
----	--------------	----------------------

10.110.2.2 gazebo::common::NodeAnimation::~~NodeAnimation ()

Destructor. It empties the key frames list.

10.110.3 Member Function Documentation

10.110.3.1 void gazebo::common::NodeAnimation::AddKeyFrame (const double *_time*, const math::Matrix4 *_trans*)

Adds a key frame at a specific time.

Parameters

in	<i>_time</i>	the time of the key frame
in	<i>_trans</i>	the transformation

10.110.3.2 void gazebo::common::NodeAnimation::AddKeyFrame (const double *_time*, const math::Pose *_pose*)

Adds a key fram at a specific time.

Parameters

in	<i>_time</i>	the tiem of the key frame
in	<i>_pose</i>	the pose

10.110.3.3 math::Matrix4 gazebo::common::NodeAnimation::GetFrameAt (double *_time*, bool *_loop* = true) const

Returns a frame transformation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.

Parameters

in	<i>_time</i>	the time
in	<i>_loop</i>	when true, the time is divided by the duration (see GetLength)

10.110.3.4 `unsigned int gazebo::common::NodeAnimation::GetFrameCount () const`

Returns the number of key frames.

Returns

the count

10.110.3.5 `void gazebo::common::NodeAnimation::GetKeyFrame (const unsigned int _i, double & _time, math::Matrix4 & _trans) const`

Finds a key frame using the index.

Note the index of a key frame can change as frames are added.

Parameters

in	<i>_i</i>	the index
out	<i>_time</i>	the time of the frame, or -1 if the index id is out of bounds
out	<i>_trans</i>	the transformation for this key frame

10.110.3.6 `std::pair<double, math::Matrix4> gazebo::common::NodeAnimation::GetKeyFrame (const unsigned int _i) const`

Returns a key frame using the index.

Note the index of a key frame can change as frames are added.

Parameters

in	<i>_i</i>	the index
----	-----------	-----------

Returns

a pair that contains the time and transformation. **Time** (p. 840) is -1 if the index is out of bounds

10.110.3.7 `double gazebo::common::NodeAnimation::GetLength () const`

Returns the duration of the animations.

Returns

the time of the last animation

10.110.3.8 `std::string gazebo::common::NodeAnimation::GetName () const`

Returns the name.

Returns

the name

10.110.3.9 `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.
-----------------	-----------------	--

10.110.3.10 `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
-----------------	---------------------	--------------------

10.110.3.11 `void gazebo::common::NodeAnimation::SetName (const std::string & _name)`

Changes the name of the animation.

Parameters

<code>in</code>	<code>the</code>	new name
-----------------	------------------	----------

10.110.4 Member Data Documentation

10.110.4.1 `std::map<double, math::Matrix4> gazebo::common::NodeAnimation::keyFrames` `[protected]`

the dictionary of key frames, indexed by time

10.110.4.2 `double gazebo::common::NodeAnimation::length` `[protected]`

the duration of the animations (time of last key frame)

10.110.4.3 `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**

10.111 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)

10.111.1 Detailed Description

Vertex to node weighted assignement for skeleton animation visualization.

10.111.2 Member Data Documentation

10.111.2.1 unsigned int gazebo::common::NodeAssignment::nodeIndex

node (or bone) index

10.111.2.2 unsigned int gazebo::common::NodeAssignment::vertexIndex

index of the vertex

10.111.2.3 float gazebo::common::NodeAssignment::weight

the weight (between 0 and 1)

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

- **Mesh.hh**

10.112 gazebo::common::NodeTransform Class Reference

A transformation node.

```
#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

10.112.1 Detailed Description

A transformation node.

10.112.2 Member Enumeration Documentation

10.112.2.1 enum gazebo::common::NodeTransform::TransformType

Enumeration of the transform types.

Enumerator:

TRANSLATE

ROTATE

SCALE

MATRIX

10.112.3 Constructor & Destructor Documentation

10.112.3.1 gazebo::common::NodeTransform::NodeTransform (TransformType _type = MATRIX)

Constructor.

Parameters

in	<i>_type</i>	the type of transform
----	--------------	-----------------------

10.112.3.2 gazebo::common::NodeTransform::NodeTransform (math::Matrix4 _mat, std::string _sid = "_default_", TransformType _type = MATRIX)

Constructor.

Parameters

in	<i>_mat</i>	the matrix
in	<i>_sid</i>	identifier
in	<i>_type</i>	the type of transform

10.112.3.3 gazebo::common::NodeTransform::~~NodeTransform ()

Destructor. It does nothing.

10.112.4 Member Function Documentation

10.112.4.1 math::Matrix4 gazebo::common::NodeTransform::Get ()

Returns the transformation matrix.

Returns

the matrix

10.112.4.2 `std::string gazebo::common::NodeTransform::GetSID ()`

Returns the SID.

Returns

the SID

10.112.4.3 `TransformType gazebo::common::NodeTransform::GetType ()`

Returns the transformation type.

Returns

the type

10.112.4.4 `math::Matrix4 gazebo::common::NodeTransform::operator() ()`

Matrix cast operator.

Returns

the transform

10.112.4.5 `math::Matrix4 gazebo::common::NodeTransform::operator* (NodeTransform _t)`

Node transform multiplication operator.

Parameters

<code>in</code>	<code>_t</code>	a transform
-----------------	-----------------	-------------

Returns

transform matrix multiplied by `_t`'s transform

10.112.4.6 `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`

10.112.4.7 `void gazebo::common::NodeTransform::PrintSource ()`

Prints the transform matrix to `std::err` stream.

10.112.4.8 `void gazebo::common::NodeTransform::RecalculateMatrix ()`

Sets the transform matrix from the source according to the type.

10.112.4.9 `void gazebo::common::NodeTransform::Set (math::Matrix4 _mat)`

Assign a transformation.

Parameters

<code>in</code>	<code>_mat</code>	the transform
-----------------	-------------------	---------------

10.112.4.10 `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

10.112.4.11 `void gazebo::common::NodeTransform::SetSID (std::string _sid)`

Set the SID.

Parameters

<code>in</code>	<code>_sid</code>	the sid
-----------------	-------------------	---------

10.112.4.12 `void gazebo::common::NodeTransform::SetSourceValues (math::Matrix4 _mat)`

Set source data values `_param[in]` `_mat` the values.

10.112.4.13 `void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 _vec)`

Set source data values.

10.112.4.14 `void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 _axis, double _angle)`

Sets source matrix values from roation.

Parameters

<code>in</code>	<code><i>_axis</i></code>	of rotation
<code>in</code>	<code><i>_angle</i></code>	of rotation

10.112.4.15 `void gazebo::common::NodeTransform::SetType (TransformType _type)`

Set transform type.

Parameters

<code>in</code>	<code><i>_type</i></code>	the type
-----------------	---------------------------	----------

10.112.5 Member Data Documentation

10.112.5.1 `std::string gazebo::common::NodeTransform::sid` `[protected]`

the sid

10.112.5.2 `std::vector<double> gazebo::common::NodeTransform::source` `[protected]`

source data values (can be a matrix, a position or rotation)

10.112.5.3 `math::Matrix4 gazebo::common::NodeTransform::transform` `[protected]`

transform

10.112.5.4 `TransformType gazebo::common::NodeTransform::type` `[protected]`

transform type

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

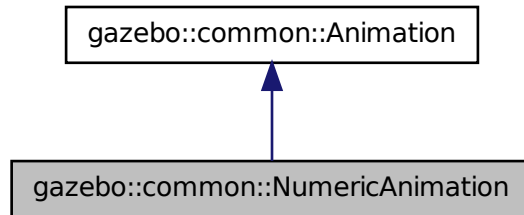
- **Skeleton.hh**

10.113 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

10.113.1 Detailed Description

A numeric animation.

10.113.2 Constructor & Destructor Documentation

10.113.2.1 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

10.113.2.2 virtual gazebo::common::NumericAnimation::~~NumericAnimation () [virtual]

Destructor.

10.113.3 Member Function Documentation

10.113.3.1 NumericKeyFrame* gazebo::common::NumericAnimation::CreateKeyFrame (double *_time*)

Create a numeric keyframe at the given time.

Parameters

in	<i>_time</i>	Time (p. 840) at which to create the keyframe
----	--------------	--

Returns

Pointer to the new keyframe

10.113.3.2 void gazebo::common::NumericAnimation::GetInterpolatedKeyFrame (NumericKeyFrame & *_kf*) const

Get a keyframe using the animation's current time.

Parameters

out	<i>_kf</i>	NumericKeyFrame (p. 571) reference to hold the interpolated result
-----	------------	---

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

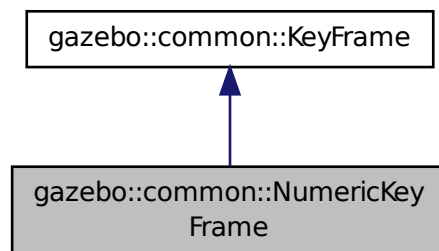
- **Animation.hh**

10.114 gazebo::common::NumericKeyFrame Class Reference

A keyframe for a **NumericAnimation** (p. 569).

```
#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

10.114.1 Detailed Description

A keyframe for a **NumericAnimation** (p. 569).

10.114.2 Constructor & Destructor Documentation

10.114.2.1 gazebo::common::NumericKeyFrame::NumericKeyFrame (double *_time*)

Constructor.

Parameters

<i>in</i>	<i>Time</i> (p. 840)	of the keyframe
-----------	-----------------------------	-----------------

10.114.2.2 virtual gazebo::common::NumericKeyFrame::~~NumericKeyFrame () [virtual]

Destructor.

10.114.3 Member Function Documentation

10.114.3.1 const double& gazebo::common::NumericKeyFrame::GetValue () const

Get the value of the keyframe.

Returns

the value of the keyframe

10.114.3.2 void gazebo::common::NumericKeyFrame::SetValue (const double & *_value*)

Set the value of the keyframe.

Parameters

in	<i>_value</i>	The new value
----	---------------	---------------

10.114.4 Member Data Documentation

10.114.4.1 double gazebo::common::NumericKeyFrame::value [protected]

numeric value

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

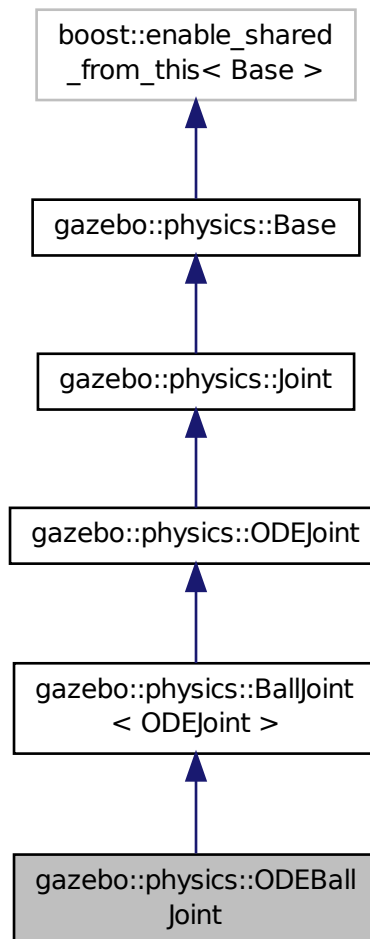
- [KeyFrame.hh](#)

10.115 gazebo::physics::ODEBallJoint Class Reference

An **ODEBallJoint** (p. 573).

```
#include <ODEBallJoint.hh>
```

Inheritance diagram for gazebo::physics::ODEBallJoint:



Public Member Functions

- **ODEBallJoint** (dWorldID worldId, **BasePtr** _parent)
Constructor.
- virtual **~ODEBallJoint** ()
Destructor.
- virtual **math::Vector3 GetAnchor** (int index) const
Get joint's anchor point.
- virtual **math::Angle GetAngleImpl** (int) const
Get the angle of rotation of an axis(index)
- virtual **math::Vector3 GetGlobalAxis** (int) const
Get the axis of rotation.

- virtual double **GetMaxForce** (int)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int) const
Get the rotation rate of an axis(index)
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)
Set joint's anchor point.
- virtual void **SetDamping** (int _index, double _damping)
Set joint damping, not yet implemented.
- virtual void **SetMaxForce** (int, double)
Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int, double)
Set the velocity of an axis(index).

Additional Inherited Members

10.115.1 Detailed Description

An **ODEBallJoint** (p. 573).

10.115.2 Constructor & Destructor Documentation

10.115.2.1 gazebo::physics::ODEBallJoint::ODEBallJoint (dWorldID worldId, BasePtr _parent)

Constructor.

10.115.2.2 virtual gazebo::physics::ODEBallJoint::~~ODEBallJoint () [virtual]

Destructor.

10.115.3 Member Function Documentation

10.115.3.1 virtual **math::Vector3** gazebo::physics::ODEBallJoint::GetAnchor (int index) const [virtual]

Get joint's anchor point.

Implements **gazebo::physics::Joint** (p. 428).

10.115.3.2 virtual **math::Angle** gazebo::physics::ODEBallJoint::GetAngleImpl (int) const [inline],[virtual]

Get the angle of rotation of an axis(index)

Implements **gazebo::physics::Joint** (p. 429).

10.115.3.3 virtual **math::Vector3** gazebo::physics::ODEBallJoint::GetGlobalAxis (int) const [inline],[virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.115.3.4 `virtual double gazebo::physics::ODEBallJoint::GetMaxForce (int) [inline],[virtual]`

Get the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 432).

10.115.3.5 `virtual double gazebo::physics::ODEBallJoint::GetVelocity (int) const [inline],[virtual]`

Get the rotation rate of an axis(index)

Implements `gazebo::physics::Joint` (p. 433).

10.115.3.6 `virtual void gazebo::physics::ODEBallJoint::SetAnchor (int index, const math::Vector3 & anchor) [virtual]`

Set joint's anchor point.

Implements `gazebo::physics::Joint` (p. 434).

10.115.3.7 `virtual void gazebo::physics::ODEBallJoint::SetDamping (int index, double damping) [virtual]`

Set joint damping, not yet implemented.

Implements `gazebo::physics::Joint` (p. 435).

10.115.3.8 `virtual void gazebo::physics::ODEBallJoint::SetMaxForce (int, double) [inline],[virtual]`

Set the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.115.3.9 `virtual void gazebo::physics::ODEBallJoint::SetVelocity (int, double) [inline],[virtual]`

Set the velocity of an axis(index).

Implements `gazebo::physics::Joint` (p. 437).

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

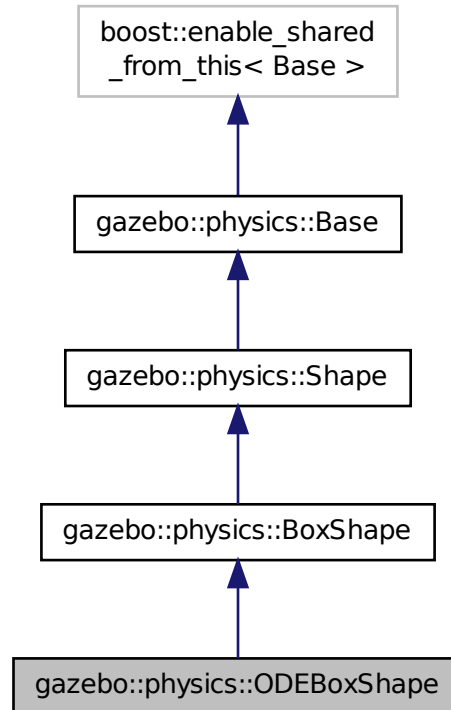
- `ODEBallJoint.hh`

10.116 `gazebo::physics::ODEBoxShape` Class Reference

ODE Box shape.

```
#include <ODEBoxShape.hh>
```

Inheritance diagram for gazebo::physics::ODEBoxShape:



Public Member Functions

- **ODEBoxShape** (`ODECollisionPtr` *_parent*)
- virtual `~ODEBoxShape` ()
- virtual void **SetSize** (`const math::Vector3` &*size*)

Set the size of the box.

Additional Inherited Members

10.116.1 Detailed Description

ODE Box shape.

10.116.2 Constructor & Destructor Documentation

10.116.2.1 `gazebo::physics::ODEBoxShape::ODEBoxShape (ODECollisionPtr _parent)` `[inline]`

10.116.2.2 virtual gazebo::physics::ODEBoxShape::~~ODEBoxShape () [inline],[virtual]

10.116.3 Member Function Documentation

10.116.3.1 virtual void gazebo::physics::ODEBoxShape::SetSize (const math::Vector3 & *_size*) [inline],[virtual]

Set the size of the box.

Parameters

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

Reimplemented from **gazebo::physics::BoxShape** (p. 165).

References gazebo::physics::Shape::collisionParent, gazebo::physics::ODECollision::GetCollisionId(), NULL, gazebo::physics::ODECollision::SetCollision(), gazebo::physics::BoxShape::SetSize(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

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

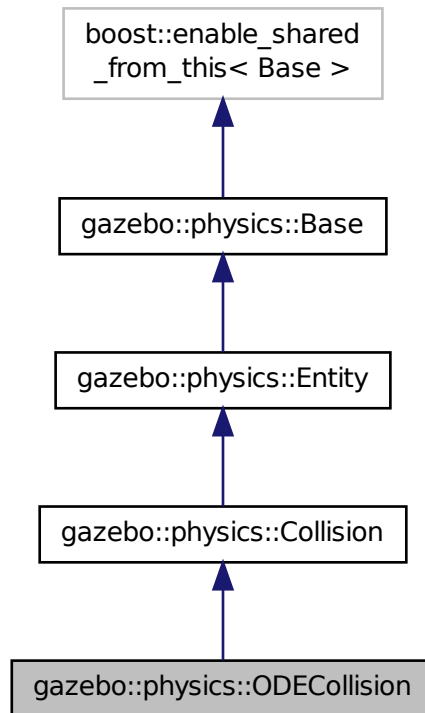
- **ODEBoxShape.hh**

10.117 gazebo::physics::ODECollision Class Reference

Base (p. 145) class for all ODE collisions.

```
#include <ODECollision.hh>
```

Inheritance diagram for gazebo::physics::ODECollision:



Public Member Functions

- **ODECollision** (LinkPtr link)
Constructor.
- virtual **~ODECollision** ()
Destructor.
- void **Fini** ()
Finalize the collision.
- virtual **math::Box GetBoundingBox** () const
Get the bounding box, defined by the physics engine.
- int **GetCollisionClass** () const
Get the ODE collision class.
- dGeomID **GetCollisionId** () const
Return the collision id.
- dSpaceID **GetSpaceId** () const
Get the collision's space ID.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the collision.

- virtual void **OnPoseChange** ()
This function is called when the entity's (or one of its parents) pose of the parent has changed.
- virtual void **SetCategoryBits** (unsigned int bits)
Set the category bits, used during collision detection.
- virtual void **SetCollideBits** (unsigned int bits)
Set the collide bits, used during collision detection.
- void **SetCollision** (dGeomID **collisionId**, bool **placeable**)
Set the encapsulated geometry object.
- void **SetSpaceId** (dSpaceID spaceid)
Set the collision's space ID.

Protected Attributes

- dGeomID **collisionId**
ID for the sub-collision.
- dSpaceID **spaceId**

Additional Inherited Members

10.117.1 Detailed Description

Base (p. 145) class for all ODE collisions.

10.117.2 Constructor & Destructor Documentation

10.117.2.1 gazebo::physics::ODECollision::ODECollision (LinkPtr link)

Constructor.

10.117.2.2 virtual gazebo::physics::ODECollision::~~ODECollision () [virtual]

Destructor.

10.117.3 Member Function Documentation

10.117.3.1 void gazebo::physics::ODECollision::Fini () [virtual]

Finalize the collision.

Reimplemented from **gazebo::physics::Collision** (p. 266).

10.117.3.2 virtual math::Box gazebo::physics::ODECollision::GetBoundingBox () const [virtual]

Get the bounding box, defined by the physics engine.

Implements **gazebo::physics::Collision** (p. 266).

10.117.3.3 `int gazebo::physics::ODECollision::GetCollisionClass () const`

Get the ODE collision class.

10.117.3.4 `dGeomID gazebo::physics::ODECollision::GetCollisionId () const`

Return the collision id.

Returns

The collision id

Referenced by `gazebo::physics::ODEPlaneShape::CreatePlane()`, `gazebo::physics::ODEPlaneShape::SetAltitude()`, `gazebo::physics::ODESphereShape::SetRadius()`, `gazebo::physics::ODECylinderShape::SetSize()`, and `gazebo::physics::ODEBoxShape::SetSize()`.

10.117.3.5 `dSpaceID gazebo::physics::ODECollision::GetSpaceId () const`

Get the collision's space ID.

Referenced by `gazebo::physics::ODEPlaneShape::CreatePlane()`.

10.117.3.6 `virtual void gazebo::physics::ODECollision::Load (sdf::ElementPtr _sdf) [virtual]`

Load the collision.

Reimplemented from `gazebo::physics::Collision` (p. 270).

10.117.3.7 `virtual void gazebo::physics::ODECollision::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. 346).

10.117.3.8 `virtual void gazebo::physics::ODECollision::SetCategoryBits (unsigned int bits) [virtual]`

Set the category bits, used during collision detection.

Parameters

<i>bits</i>	The bits
-------------	----------

Implements `gazebo::physics::Collision` (p. 270).

10.117.3.9 `virtual void gazebo::physics::ODECollision::SetCollideBits (unsigned int bits) [virtual]`

Set the collide bits, used during collision detection.

Parameters

<i>bits</i>	The bits
-------------	----------

Implements `gazebo::physics::Collision` (p.270).

10.117.3.10 `void gazebo::physics::ODECollision::SetCollision (dGeomID collisionId, bool placeable)`

Set the encapsulated geometry object.

Referenced by `gazebo::physics::ODEPlaneShape::CreatePlane()`, `gazebo::physics::ODESphereShape::SetRadius()`, `gazebo::physics::ODECylinderShape::SetSize()`, and `gazebo::physics::ODEBoxShape::SetSize()`.

10.117.3.11 `void gazebo::physics::ODECollision::SetSpaceId (dSpaceID spaceId)`

Set the collision's space ID.

10.117.4 Member Data Documentation

10.117.4.1 `dGeomID gazebo::physics::ODECollision::collisionId` `[protected]`

ID for the sub-collision.

10.117.4.2 `dSpaceID gazebo::physics::ODECollision::spaceId` `[protected]`

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

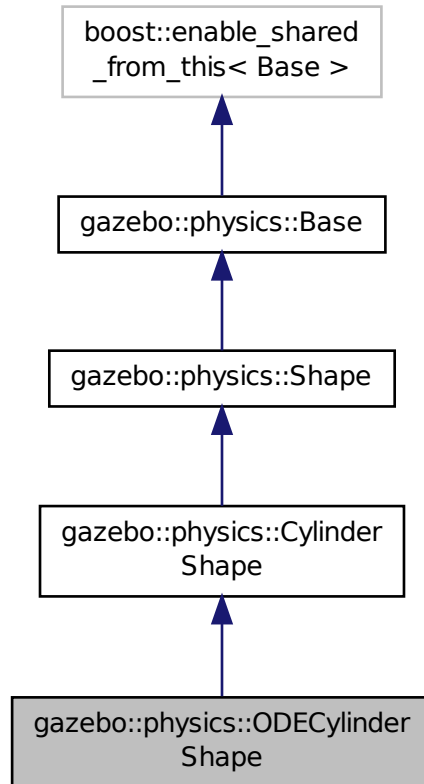
- `ODECollision.hh`

10.118 `gazebo::physics::ODECylinderShape` Class Reference

ODE cylinder shape.

```
#include <ODECylinderShape.hh>
```

Inheritance diagram for gazebo::physics::ODECylinderShape:



Public Member Functions

- **ODECylinderShape** (`CollisionPtr _parent`)
constructor
- virtual `~ODECylinderShape` ()
destructor
- void **SetSize** (`double _radius`, `double _length`)
Set radius and length of a cylinder.

Additional Inherited Members

10.118.1 Detailed Description

ODE cylinder shape.

10.118.2 Constructor & Destructor Documentation

10.118.2.1 `gazebo::physics::ODECylinderShape::ODECylinderShape (CollisionPtr _parent)` `[inline]`

constructor

10.118.2.2 `virtual gazebo::physics::ODECylinderShape::~~ODECylinderShape ()` `[inline],[virtual]`

destructor

10.118.3 Member Function Documentation

10.118.3.1 `void gazebo::physics::ODECylinderShape::SetSize (double _radius, double _length)` `[inline],[virtual]`

Set radius and length of a cylinder.

Reimplemented from `gazebo::physics::CylinderShape` (p. 311).

References `gazebo::physics::Shape::collisionParent`, `gazebo::physics::ODECollision::GetCollisionId()`, `NULL`, `gazebo::physics::ODECollision::SetCollision()`, and `gazebo::physics::CylinderShape::SetSize()`.

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

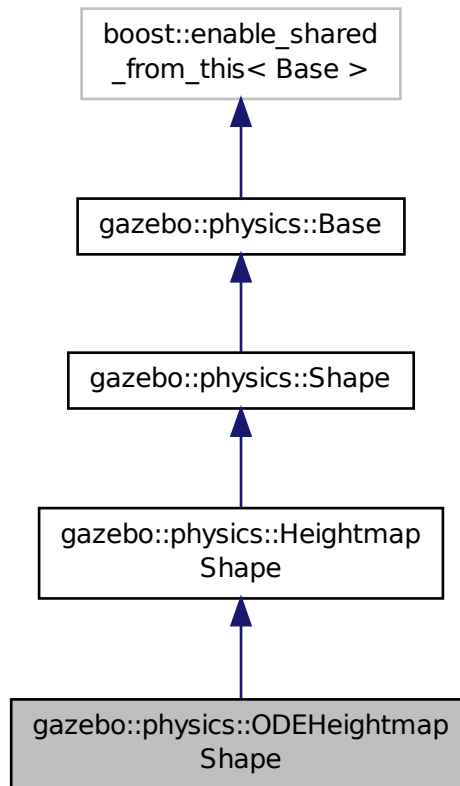
- `ODECylinderShape.hh`

10.119 gazebo::physics::ODEHeightmapShape Class Reference

ODE Height map collision.

```
#include <ODEHeightmapShape.hh>
```

Inheritance diagram for gazebo::physics::ODEHeightmapShape:



Public Member Functions

- **ODEHeightmapShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~ODEHeightmapShape** ()
Destructor.
- virtual void **Init** ()
Load the heightmap.

Additional Inherited Members

10.119.1 Detailed Description

ODE Height map collision.

10.119.2 Constructor & Destructor Documentation

10.119.2.1 gazebo::physics::ODEHeightmapShape::ODEHeightmapShape (CollisionPtr *_parent*)

Constructor.

10.119.2.2 virtual gazebo::physics::ODEHeightmapShape::~~ODEHeightmapShape () [virtual]

Destructor.

10.119.3 Member Function Documentation

10.119.3.1 virtual void gazebo::physics::ODEHeightmapShape::Init () [virtual]

Load the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 403).

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

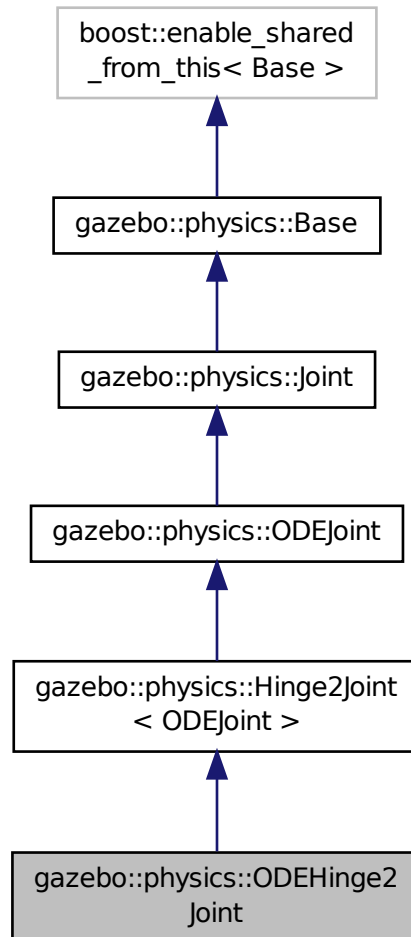
- **ODEHeightmapShape.hh**

10.120 gazebo::physics::ODEHinge2Joint Class Reference

A two axis hinge joint.

```
#include <ODEHinge2Joint.hh>
```

Inheritance diagram for gazebo::physics::ODEHinge2Joint:



Public Member Functions

- **ODEHinge2Joint** (dWorldID worldId, **BasePtr** _parent)
Constructor.
- virtual **~ODEHinge2Joint** ()
Destructor.
- virtual **math::Vector3 GetAnchor** (int index) const
Get anchor point.
- virtual **math::Angle GetAngleImpl** (int index) const
Get angle of rotation about first axis.
- virtual **math::Vector3 GetGlobalAxis** (int index) const
Get first axis of rotation.

- virtual double **GetMaxForce** (int index)
Get the max allowed force of an axis(index).
- virtual double **GetParam** (int parameter) const
Get the specified parameter.
- virtual double **GetVelocity** (int index) const
Get rate of rotation about first axis.
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)
Set the anchor point.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)
Set the first axis of rotation.
- virtual void **SetDamping** (int _index, double _damping)
Set joint damping, not yet implemented.
- virtual void **SetForce** (int index, double torque)
Set the torque.
- virtual void **SetMaxForce** (int index, double t)
Set the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)
Set _parameter with _value.
- virtual void **SetVelocity** (int index, double angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load the **ODEHinge2Joint** (p. 586).*

Additional Inherited Members

10.120.1 Detailed Description

A two axis hinge joint.

10.120.2 Constructor & Destructor Documentation

10.120.2.1 **gazebo::physics::ODEHinge2Joint::ODEHinge2Joint** (**dWorldID** *worldId*, **BasePtr** *_parent*)

Constructor.

10.120.2.2 **virtual gazebo::physics::ODEHinge2Joint::~~ODEHinge2Joint** () [virtual]

Destructor.

10.120.3 Member Function Documentation

10.120.3.1 virtual `math::Vector3` gazebo::physics::ODEHinge2Joint::GetAnchor (int *index*) const [virtual]

Get anchor point.

Implements `gazebo::physics::Joint` (p. 428).

10.120.3.2 virtual `math::Angle` gazebo::physics::ODEHinge2Joint::GetAngleImpl (int *index*) const [virtual]

Get angle of rotation about first axis.

Implements `gazebo::physics::Joint` (p. 429).

10.120.3.3 virtual `math::Vector3` gazebo::physics::ODEHinge2Joint::GetGlobalAxis (int *index*) const [virtual]

Get first axis of rotation.

Implements `gazebo::physics::Joint` (p. 430).

10.120.3.4 virtual double gazebo::physics::ODEHinge2Joint::GetMaxForce (int *index*) [virtual]

Get the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 432).

10.120.3.5 virtual double gazebo::physics::ODEHinge2Joint::GetParam (int *parameter*) const [virtual]

Get the specified parameter.

Reimplemented from `gazebo::physics::ODEJoint` (p. 598).

10.120.3.6 virtual double gazebo::physics::ODEHinge2Joint::GetVelocity (int *index*) const [virtual]

Get rate of rotation about first axis.

Implements `gazebo::physics::Joint` (p. 433).

10.120.3.7 virtual void gazebo::physics::ODEHinge2Joint::Load (sdf::ElementPtr *sdf*) [protected],[virtual]

Load the `ODEHinge2Joint` (p. 586).

Reimplemented from `gazebo::physics::Hinge2Joint< ODEJoint >` (p. 405).

10.120.3.8 virtual void gazebo::physics::ODEHinge2Joint::SetAnchor (int *index*, const `math::Vector3` & *anchor*) [virtual]

Set the anchor point.

Implements `gazebo::physics::Joint` (p. 434).

10.120.3.9 `virtual void gazebo::physics::ODEHinge2Joint::SetAxis (int index, const math::Vector3 & axis) [virtual]`

Set the first axis of rotation.

Implements `gazebo::physics::Joint` (p. 435).

10.120.3.10 `virtual void gazebo::physics::ODEHinge2Joint::SetDamping (int _index, double _damping) [virtual]`

Set joint damping, not yet implemented.

Implements `gazebo::physics::Joint` (p. 435).

10.120.3.11 `virtual void gazebo::physics::ODEHinge2Joint::SetForce (int index, double torque) [virtual]`

Set the torque.

Reimplemented from `gazebo::physics::Joint` (p. 435).

10.120.3.12 `virtual void gazebo::physics::ODEHinge2Joint::SetMaxForce (int index, double t) [virtual]`

Set the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.120.3.13 `virtual void gazebo::physics::ODEHinge2Joint::SetParam (int parameter, double value) [virtual]`

Set `_parameter` with `_value`.

Reimplemented from `gazebo::physics::ODEJoint` (p. 599).

10.120.3.14 `virtual void gazebo::physics::ODEHinge2Joint::SetVelocity (int index, double angle) [virtual]`

Set the velocity of an axis(index).

Implements `gazebo::physics::Joint` (p. 437).

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

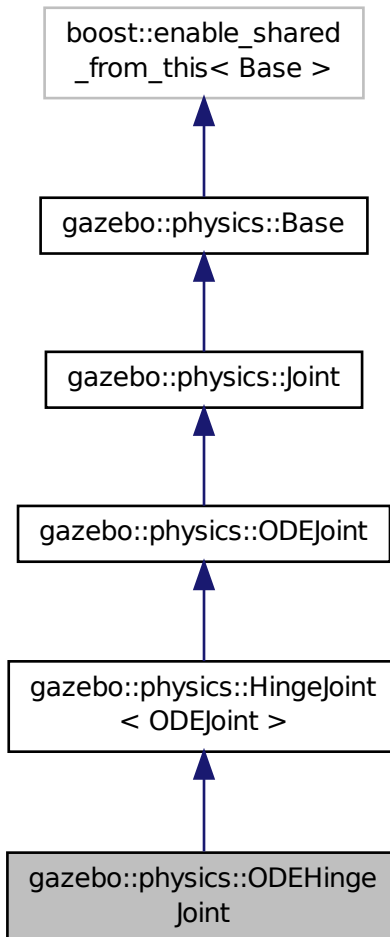
- `ODEHinge2Joint.hh`

10.121 gazebo::physics::ODEHingeJoint Class Reference

A single axis hinge joint.

```
#include <ODEHingeJoint.hh>
```

Inheritance diagram for gazebo::physics::ODEHingeJoint:



Public Member Functions

- **ODEHingeJoint** (dWorldID worldId, **BasePtr** _parent)
Constructor.
- virtual **~ODEHingeJoint** ()
Destructor.
- void **ApplyDamping** ()
callback to apply damping force to joint
- virtual **math::Vector3 GetAnchor** (int index) const
Get the anchor point.
- virtual **math::Angle GetAngleImpl** (int index) const
Get the angle of rotation.

- virtual **math::Vector3 GetGlobalAxis** (int index) const
Get the axis of rotation.
- virtual double **GetMaxForce** (int index)
Get the max allowed force of an axis(index).
- virtual double **GetParam** (int parameter) const
Get the specified parameter.
- virtual double **GetVelocity** (int index) const
Get the rotation rate of an axis(index)
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)
Set the anchor point.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)
Set the axis of rotation.
- virtual void **SetDamping** (int _index, double _damping)
Set the joint damping.
- virtual void **SetForce** (int index, double torque)
Set the torque of a joint.
- virtual void **SetMaxForce** (int index, double t)
Set the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)
Set the parameter to value.
- virtual void **SetVelocity** (int index, double angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)
Load joint.

Additional Inherited Members

10.121.1 Detailed Description

A single axis hinge joint.

10.121.2 Constructor & Destructor Documentation

10.121.2.1 gazebo::physics::ODEHingeJoint::ODEHingeJoint (dWorldID worldId, BasePtr _parent)

Constructor.

10.121.2.2 virtual gazebo::physics::ODEHingeJoint::~~ODEHingeJoint () [virtual]

Destructor.

10.121.3 Member Function Documentation

10.121.3.1 void gazebo::physics::ODEHingeJoint::ApplyDamping ()

callback to apply damping force to joint

10.121.3.2 virtual math::Vector3 gazebo::physics::ODEHingeJoint::GetAnchor (int *index*) const [virtual]

Get the anchor point.

Implements **gazebo::physics::Joint** (p. 428).

10.121.3.3 virtual math::Angle gazebo::physics::ODEHingeJoint::GetAngleImpl (int *index*) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.121.3.4 virtual math::Vector3 gazebo::physics::ODEHingeJoint::GetGlobalAxis (int *index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.121.3.5 virtual double gazebo::physics::ODEHingeJoint::GetMaxForce (int *index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.121.3.6 virtual double gazebo::physics::ODEHingeJoint::GetParam (int *parameter*) const [virtual]

Get the specified parameter.

Reimplemented from **gazebo::physics::ODEJoint** (p. 598).

10.121.3.7 virtual double gazebo::physics::ODEHingeJoint::GetVelocity (int *index*) const [virtual]

Get the rotation rate of an axis(index)

Implements **gazebo::physics::Joint** (p. 433).

10.121.3.8 virtual void gazebo::physics::ODEHingeJoint::Load (sdf::ElementPtr *_sdf*) [protected],[virtual]

Load joint.

Reimplemented from **gazebo::physics::HingeJoint< ODEJoint >** (p. 407).

10.121.3.9 virtual void gazebo::physics::ODEHingeJoint::SetAnchor (int *index*, const math::Vector3 & *anchor*) [virtual]

Set the anchor point.

Implements **gazebo::physics::Joint** (p. 434).

10.121.3.10 `virtual void gazebo::physics::ODEHingeJoint::SetAxis (int index, const math::Vector3 & axis) [virtual]`

Set the axis of rotation.

Implements **gazebo::physics::Joint** (p. 435).

10.121.3.11 `virtual void gazebo::physics::ODEHingeJoint::SetDamping (int _index, double _damping) [virtual]`

Set the joint damping.

Implements **gazebo::physics::Joint** (p. 435).

10.121.3.12 `virtual void gazebo::physics::ODEHingeJoint::SetForce (int index, double torque) [virtual]`

Set the torque of a joint.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.121.3.13 `virtual void gazebo::physics::ODEHingeJoint::SetMaxForce (int index, double t) [virtual]`

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.121.3.14 `virtual void gazebo::physics::ODEHingeJoint::SetParam (int parameter, double value) [virtual]`

Set the parameter to value.

Reimplemented from **gazebo::physics::ODEJoint** (p. 599).

10.121.3.15 `virtual void gazebo::physics::ODEHingeJoint::SetVelocity (int index, double angle) [virtual]`

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

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

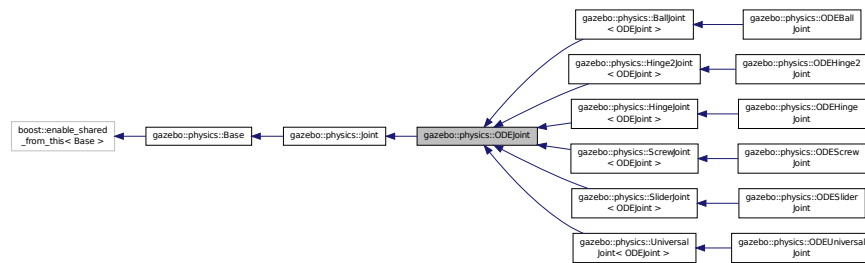
- **ODEHingeJoint.hh**

10.122 gazebo::physics::ODEJoint Class Reference

ODE joint interface.

```
#include <ODEJoint.hh>
```

Inheritance diagram for gazebo::physics::ODEJoint:



Public Member Functions

- **ODEJoint** (**BasePtr** _parent)
Constructor.
- virtual **~ODEJoint** ()
Destructor.
- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const
Determines if the two bodies are connected by a joint.
- virtual void **Attach** (**LinkPtr** parent, **LinkPtr** child)
Attach the two bodies with this joint.
- virtual void **Detach** ()
Detach this joint from all bodies.
- double **GetCFM** ()
Get the ERP of this joint.
- double **GetERP** ()
Get the ERP of this joint.
- **dJointFeedback** * **GetFeedback** ()
Get the feedback data structure for this joint, if set.
- virtual **math::Angle** **GetHighStop** (int index)
Get the high stop of an axis(index).
- virtual **LinkPtr** **GetJointLink** (int _index) const
Get the link to which the joint is attached according the _index.
- virtual **math::Vector3** **GetLinkForce** (unsigned int index) const
Get the force the joint applies to the first link.
- virtual **math::Vector3** **GetLinkTorque** (unsigned int index) const
Get the torque the joint applies to the first link.
- virtual **math::Angle** **GetLowStop** (int index)
Get the low stop of an axis(index).
- virtual double **GetParam** (int _parameter) const
The default function does nothing.
- virtual void **Load** (**sdf::ElementPtr** _sdf)
Load ODEJoint (p. 594).
- virtual void **Reset** ()
Reset the joint.

- virtual void **SetAttribute** (**Attribute**, int index, double value)
Set a parameter for the joint.
- void **SetCFM** (double newCFM)
Set the CFM of this joint.
- void **SetERP** (double newERP)
Set the ERP of this joint.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)
Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)
Set the low stop of an axis(index).
- virtual void **SetParam** (int _parameter, double _value)
By default this does nothing.

Protected Attributes

- dJointID **jointId**
This is our id.

Additional Inherited Members

10.122.1 Detailed Description

ODE joint interface.

10.122.2 Constructor & Destructor Documentation

10.122.2.1 gazebo::physics::ODEJoint::ODEJoint (BasePtr _parent)

Constructor.

10.122.2.2 virtual gazebo::physics::ODEJoint::~~ODEJoint () [virtual]

Destructor.

10.122.3 Member Function Documentation

10.122.3.1 virtual bool gazebo::physics::ODEJoint::AreConnected (LinkPtr _one, LinkPtr _two) const [virtual]

Determines if the two bodies are connected by a joint.

Implements **gazebo::physics::Joint** (p. 427).

10.122.3.2 virtual void gazebo::physics::ODEJoint::Attach (LinkPtr parent, LinkPtr child) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.122.3.3 virtual void gazebo::physics::ODEJoint::Detach () [virtual]

Detach this joint from all bodies.

Reimplemented from **gazebo::physics::Joint** (p. 428).

10.122.3.4 double gazebo::physics::ODEJoint::GetCFM ()

Get the ERP of this joint.

10.122.3.5 double gazebo::physics::ODEJoint::GetERP ()

Get the ERP of this joint.

10.122.3.6 dJointFeedback* gazebo::physics::ODEJoint::GetFeedback ()

Get the feedback data structure for this joint, if set.

10.122.3.7 virtual math::Angle gazebo::physics::ODEJoint::GetHighStop (int *index*) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

Reimplemented in **gazebo::physics::BallJoint< ODEJoint >** (p. 144).

10.122.3.8 virtual LinkPtr gazebo::physics::ODEJoint::GetJointLink (int *_index*) const [virtual]

Get the link to which the joint is attached according the *_index*.

Implements **gazebo::physics::Joint** (p. 431).

10.122.3.9 virtual math::Vector3 gazebo::physics::ODEJoint::GetLinkForce (unsigned int *index*) const [virtual]

Get the force the joint applies to the first link.

Parameters

<i>index</i>	The index of the link(0 or 1)
--------------	-------------------------------

Implements **gazebo::physics::Joint** (p. 431).

10.122.3.10 virtual math::Vector3 gazebo::physics::ODEJoint::GetLinkTorque (unsigned int *index*) const [virtual]

Get the torque the joint applies to the first link.

Parameters

<i>index</i>	The index of the link(0 or 1)
--------------	-------------------------------

Implements **gazebo::physics::Joint** (p. 431).

10.122.3.11 `virtual math::Angle gazebo::physics::ODEJoint::GetLowStop (int index) [virtual]`

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

Reimplemented in **gazebo::physics::BallJoint< ODEJoint >** (p. 144).

10.122.3.12 `virtual double gazebo::physics::ODEJoint::GetParam (int _parameter) const [virtual]`

The default function does nothing.

This should be overridden in the child classes where appropriate

Reimplemented in **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEHinge2Joint** (p. 589), and **gazebo::physics::ODESliderJoint** (p. 628).

10.122.3.13 `virtual void gazebo::physics::ODEJoint::Load (sdf::ElementPtr _sdf) [virtual]`

Load **ODEJoint** (p. 594).

Reimplemented from **gazebo::physics::Joint** (p. 433).

Reimplemented in **gazebo::physics::BallJoint< ODEJoint >** (p. 144), **gazebo::physics::ScrewJoint< ODEJoint >** (p. 761), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::SliderJoint< ODEJoint >** (p. 805), **gazebo::physics::Hinge2Joint< ODEJoint >** (p. 405), **gazebo::physics::HingeJoint< ODEJoint >** (p. 407), **gazebo::physics::UniversalJoint< ODEJoint >** (p. 870), and **gazebo::physics::ODESliderJoint** (p. 628).

10.122.3.14 `virtual void gazebo::physics::ODEJoint::Reset () [virtual]`

Reset the joint.

Reimplemented from **gazebo::physics::Joint** (p. 434).

10.122.3.15 `virtual void gazebo::physics::ODEJoint::SetAttribute (Attribute , int index, double value) [virtual]`

Set a parameter for the joint.

Implements **gazebo::physics::Joint** (p. 435).

10.122.3.16 `void gazebo::physics::ODEJoint::SetCFM (double newCFM)`

Set the CFM of this joint.

10.122.3.17 `void gazebo::physics::ODEJoint::SetERP (double newERP)`

Set the ERP of this joint.

10.122.3.18 `virtual void gazebo::physics::ODEJoint::SetHighStop (int _index, const math::Angle & _angle) [virtual]`

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.122.3.19 `virtual void gazebo::physics::ODEJoint::SetLowStop (int _index, const math::Angle & _angle) [virtual]`

Set the low stop of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.122.3.20 `virtual void gazebo::physics::ODEJoint::SetParam (int _parameter, double _value) [virtual]`

By default this does nothing.

It should be overridden in child classes where appropriate

Reimplemented in `gazebo::physics::ODEHingeJoint` (p. 594), `gazebo::physics::ODEScrewJoint` (p. 625), `gazebo::physics::ODEHinge2Joint` (p. 590), `gazebo::physics::ODESliderJoint` (p. 629), and `gazebo::physics::ODEUniversalJoint` (p. 637).

10.122.4 Member Data Documentation

10.122.4.1 `dJointID gazebo::physics::ODEJoint::jointId [protected]`

This is our id.

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

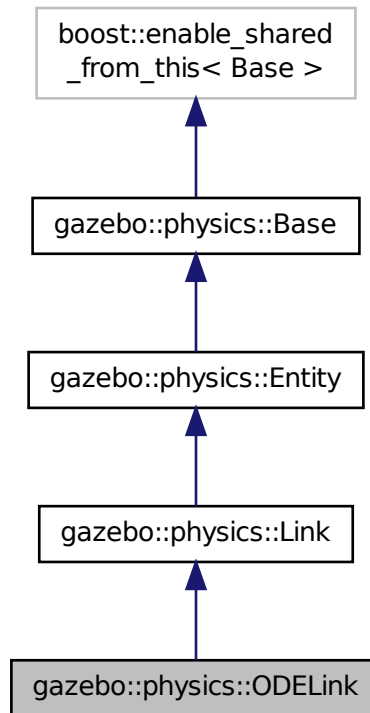
- `ODEJoint.hh`

10.123 gazebo::physics::ODELink Class Reference

ODE Link (p. 454) class.

```
#include <ODELink.hh>
```

Inheritance diagram for gazebo::physics::ODELink:



Public Member Functions

- **ODELink** (**EntityPtr** _parent)
Constructor.
- virtual **~ODELink** ()
Destructor.
- virtual void **AddForce** (const **math::Vector3** &_force)
Add a force to the body.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)
Set the force applied to the body (add by Stefano)
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)
Add a force to the body using a global position.
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)
Add a force to the body.
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)
Add a torque to the body relative frame.
- virtual void **AddTorque** (const **math::Vector3** &_torque)
Add a torque to the body.

- virtual void **Fini** ()
Finalize the link.
- virtual bool **GetEnabled** () const
Get whether this link is enabled in the physics engine.
- virtual bool **GetGravityMode** ()
Get the gravity mode.
- virtual bool **GetKinematic** () const
Get whether this link is in the kinematic state.
- dBodyID **GetODEId** () const
Return the ID of this link.
- dSpaceID **GetSpaceId** () const
Get the link's space ID.
- virtual **math::Vector3** **GetWorldAngularVel** () const
Get the angular velocity of the link in the world frame.
- virtual **math::Vector3** **GetWorldForce** () const
Get the force applied to the link in the world frame.
- virtual **math::Vector3** **GetWorldLinearVel** () const
Get the linear velocity of the link in the world frame.
- virtual **math::Vector3** **GetWorldTorque** () const
Get the torque applied to the link in the world frame.
- virtual void **Init** ()
Initialize the link.
- virtual void **Load** (**sdf::ElementPtr** _sdf)
Load the link based on SDF parameters.
- virtual void **OnPoseChange** ()
Called when the pose of the entity (or one of its parents) has changed.
- virtual void **SetAngularDamping** (double damping)
Set the angular damping factor.
- virtual void **SetAngularVel** (const **math::Vector3** &vel)
Set the angular velocity of the link.
- virtual void **SetAutoDisable** (bool _disable)
Allow the link to auto disable.
- virtual void **SetEnabled** (bool enable) const
Set whether this link is enabled.
- virtual void **SetForce** (const **math::Vector3** &_force)
Set the force applied to the link.
- virtual void **SetGravityMode** (bool mode)
Set whether gravity affects this link.
- virtual void **SetKinematic** (const bool &state)
Set whether this link is in the kinematic state.
- virtual void **SetLinearDamping** (double damping)
Set the linear damping factor.
- virtual void **SetLinearVel** (const **math::Vector3** &vel)
Set the linear velocity of the link.
- void **SetSelfCollide** (bool collide)
Set whether this link will collide with others in the model.
- void **SetSpaceId** (dSpaceID spaceid)

Set the link's space ID.

- virtual void **SetTorque** (const **math::Vector3** &_torque)

Set the torque applied to the link.

- virtual void **Update** ()

Update the link.

- virtual void **UpdateMass** ()

Update the mass matrix.

- virtual void **UpdateSurface** ()

Update other parameters for ODE.

Static Public Member Functions

- static void **DisabledCallback** (dBodyID _id)

callback when ODE determines a body is disabled

- static void **MoveCallback** (dBodyID id)

when ODE updates dynamics bodies, this callback propagates the changes in pose back to Gazebo

Protected Attributes

- **math::Pose** pose

Additional Inherited Members

10.123.1 Detailed Description

ODE **Link** (p. 454) class.

10.123.2 Constructor & Destructor Documentation

10.123.2.1 gazebo::physics::ODELink::ODELink (EntityPtr _parent)

Constructor.

10.123.2.2 virtual gazebo::physics::ODELink::~~ODELink () [virtual]

Destructor.

10.123.3 Member Function Documentation

10.123.3.1 virtual void gazebo::physics::ODELink::AddForce (const **math::Vector3** & _force) [virtual]

Add a force to the body.

Implements **gazebo::physics::Link** (p. 459).

10.123.3.2 `virtual void gazebo::physics::ODELink::AddForceAtRelativePosition (const math::Vector3 & _force, const math::Vector3 & _relpos) [virtual]`

Set the force applied to the body (add by Stefano)

Implements **gazebo::physics::Link** (p. 460).

10.123.3.3 `virtual void gazebo::physics::ODELink::AddForceAtWorldPosition (const math::Vector3 & _force, const math::Vector3 & _pos) [virtual]`

Add a force to the body using a global position.

Implements **gazebo::physics::Link** (p. 460).

10.123.3.4 `virtual void gazebo::physics::ODELink::AddRelativeForce (const math::Vector3 & _force) [virtual]`

Add a force to the body.

Implements **gazebo::physics::Link** (p. 460).

10.123.3.5 `virtual void gazebo::physics::ODELink::AddRelativeTorque (const math::Vector3 & _torque) [virtual]`

Add a torque to the body relative frame.

Implements **gazebo::physics::Link** (p. 460).

10.123.3.6 `virtual void gazebo::physics::ODELink::AddTorque (const math::Vector3 & _torque) [virtual]`

Add a torque to the body.

Implements **gazebo::physics::Link** (p. 461).

10.123.3.7 `static void gazebo::physics::ODELink::DisabledCallback (dBodyID _id) [static]`

callback when ODE determines a body is disabled

10.123.3.8 `virtual void gazebo::physics::ODELink::Fini () [virtual]`

Finalize the link.

Reimplemented from **gazebo::physics::Link** (p. 462).

10.123.3.9 `virtual bool gazebo::physics::ODELink::GetEnabled () const [virtual]`

Get whether this link is enabled in the physics engine.

Implements **gazebo::physics::Link** (p. 463).

10.123.3.10 `virtual bool gazebo::physics::ODELink::GetGravityMode () [virtual]`

Get the gravity mode.

Implements **gazebo::physics::Link** (p. 463).

10.123.3.11 `virtual bool gazebo::physics::ODELink::GetKinematic () const [virtual]`

Get whether this link is in the kinematic state.

Reimplemented from **gazebo::physics::Link** (p. 464).

10.123.3.12 `dBodyID gazebo::physics::ODELink::GetODEId () const`

Return the ID of this link.

Returns

ODE link id

10.123.3.13 `dSpaceID gazebo::physics::ODELink::GetSpaceId () const`

Get the link's space ID.

10.123.3.14 `virtual math::Vector3 gazebo::physics::ODELink::GetWorldAngularVel () const [virtual]`

Get the angular velocity of the link in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.123.3.15 `virtual math::Vector3 gazebo::physics::ODELink::GetWorldForce () const [virtual]`

Get the force applied to the link in the world frame.

Implements **gazebo::physics::Link** (p. 467).

10.123.3.16 `virtual math::Vector3 gazebo::physics::ODELink::GetWorldLinearVel () const [virtual]`

Get the linear velocity of the link in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 345).

10.123.3.17 `virtual math::Vector3 gazebo::physics::ODELink::GetWorldTorque () const [virtual]`

Get the torque applied to the link in the world frame.

Implements **gazebo::physics::Link** (p. 467).

10.123.3.18 `virtual void gazebo::physics::ODELink::Init () [virtual]`

Initialize the link.

Reimplemented from **gazebo::physics::Link** (p. 467).

10.123.3.19 `virtual void gazebo::physics::ODELink::Load (sdf::ElementPtr _sdf) [virtual]`

Load the link based on SDF parameters.

Parameters

<code>_sdf</code>	the sdf parameters
-------------------	--------------------

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

10.123.3.20 `static void gazebo::physics::ODELink::MoveCallback (dBodyID id) [static]`

when ODE updates dynamics bodies, this callback propagates the changes in pose back to Gazebo

10.123.3.21 `virtual void gazebo::physics::ODELink::OnPoseChange () [virtual]`

Called when the pose of the entity (or one of its parents) has changed.

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

10.123.3.22 `virtual void gazebo::physics::ODELink::SetAngularDamping (double damping) [virtual]`

Set the angular damping factor.

Implements **gazebo::physics::Link** (p. 469).

10.123.3.23 `virtual void gazebo::physics::ODELink::SetAngularVel (const math::Vector3 & vel) [virtual]`

Set the angular velocity of the link.

Implements **gazebo::physics::Link** (p. 469).

10.123.3.24 `virtual void gazebo::physics::ODELink::SetAutoDisable (bool _disable) [virtual]`

Allow the link to auto disable.

Parameters

<code>_disable</code>	If true, the link is allowed to auto disable.
-----------------------	---

Implements **gazebo::physics::Link** (p. 469).

10.123.3.25 `virtual void gazebo::physics::ODELink::SetEnabled (bool enable) const [virtual]`

Set whether this link is enabled.

Implements **gazebo::physics::Link** (p. 470).

10.123.3.26 `virtual void gazebo::physics::ODELink::SetForce (const math::Vector3 & .force) [virtual]`

Set the force applied to the link.

Implements **gazebo::physics::Link** (p. 470).

10.123.3.27 `virtual void gazebo::physics::ODELink::SetGravityMode (bool mode) [virtual]`

Set whether gravity affects this link.

Implements **gazebo::physics::Link** (p. 470).

10.123.3.28 `virtual void gazebo::physics::ODELink::SetKinematic (const bool & state) [virtual]`

Set whether this link is in the kinematic state.

Reimplemented from **gazebo::physics::Link** (p. 470).

10.123.3.29 `virtual void gazebo::physics::ODELink::SetLinearDamping (double damping) [virtual]`

Set the linear damping factor.

Implements **gazebo::physics::Link** (p. 471).

10.123.3.30 `virtual void gazebo::physics::ODELink::SetLinearVel (const math::Vector3 & vel) [virtual]`

Set the linear velocity of the link.

Implements **gazebo::physics::Link** (p. 471).

10.123.3.31 `void gazebo::physics::ODELink::SetSelfCollide (bool collide) [virtual]`

Set whether this link will collide with others in the model.

Implements **gazebo::physics::Link** (p. 471).

10.123.3.32 `void gazebo::physics::ODELink::SetSpaceId (dSpaceID spaceid)`

Set the link's space ID.

10.123.3.33 `virtual void gazebo::physics::ODELink::SetTorque (const math::Vector3 & torque) [virtual]`

Set the torque applied to the link.

Implements **gazebo::physics::Link** (p. 472).

10.123.3.34 `virtual void gazebo::physics::ODELink::Update () [virtual]`

Update the link.

Reimplemented from **gazebo::physics::Link** (p. 472).

10.123.3.35 `virtual void gazebo::physics::ODELink::UpdateMass () [virtual]`

Update the mass matrix.

Reimplemented from **gazebo::physics::Link** (p. 472).

10.123.3.36 `virtual void gazebo::physics::ODELink::UpdateSurface () [virtual]`

Update other parameters for ODE.

Reimplemented from **gazebo::physics::Link** (p. 472).

10.123.4 Member Data Documentation

10.123.4.1 `math::Pose gazebo::physics::ODELink::pose [protected]`

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

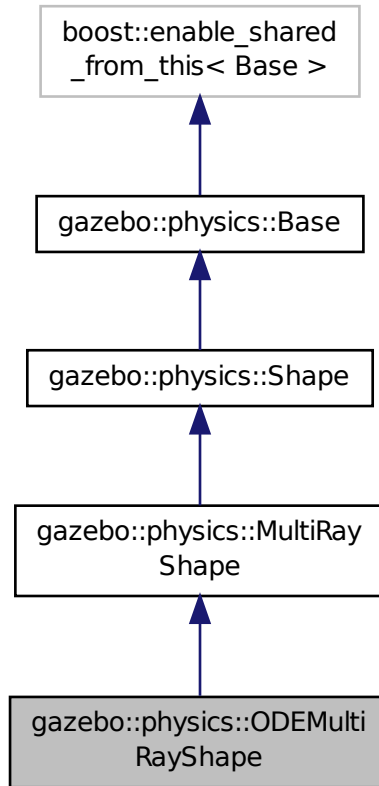
- **ODELink.hh**

10.124 gazebo::physics::ODEMultiRayShape Class Reference

ODE specific version of **MultiRayShape** (p. 549).

```
#include <ODEMultiRayShape.hh>
```

Inheritance diagram for gazebo::physics::ODEMultiRayShape:



Public Member Functions

- **ODEMultiRayShape** (*CollisionPtr* parent)

Constructor.

- virtual **~ODEMultiRayShape** ()

Destructor.

- virtual void **UpdateRays** ()

Update the rays.

Protected Member Functions

- void **AddRay** (const **math::Vector3** &start, const **math::Vector3** &end)

Add a ray to the collision.

Additional Inherited Members

10.124.1 Detailed Description

ODE specific version of **MultiRayShape** (p. 549).

10.124.2 Constructor & Destructor Documentation

10.124.2.1 gazebo::physics::ODEMultiRayShape::ODEMultiRayShape (CollisionPtr parent)

Constructor.

10.124.2.2 virtual gazebo::physics::ODEMultiRayShape::~~ODEMultiRayShape () [virtual]

Destructor.

10.124.3 Member Function Documentation

10.124.3.1 void gazebo::physics::ODEMultiRayShape::AddRay (const math::Vector3 & start, const math::Vector3 & end) [protected], [virtual]

Add a ray to the collision.

Reimplemented from **gazebo::physics::MultiRayShape** (p. 551).

10.124.3.2 virtual void gazebo::physics::ODEMultiRayShape::UpdateRays () [virtual]

Update the rays.

Implements **gazebo::physics::MultiRayShape** (p. 555).

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

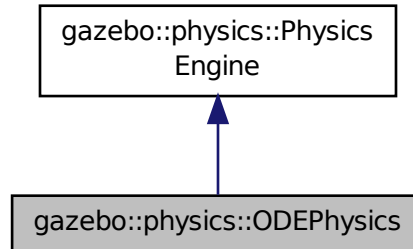
- **ODEMultiRayShape.hh**

10.125 gazebo::physics::ODEPhysics Class Reference

ODE physics engine.

```
#include <ODEPhysics.hh>
```

Inheritance diagram for gazebo::physics::ODEPhysics:



Public Member Functions

- **ODEPhysics (WorldPtr world)**
Constructor.
- virtual **~ODEPhysics ()**
Destructor.
- void **Collide (ODECollision *collision1, ODECollision *collision2, dContactGeom *contactCollisions)**
Collide two collisions.
- virtual **CollisionPtr CreateCollision (const std::string &_shapeType, LinkPtr _parent)**
Create a collision.
- void **CreateContact (ODECollision *collision1, ODECollision *collision2)**
Not yet implemented.
- virtual **JointPtr CreateJoint (const std::string &type, ModelPtr _parent)**
Create a new joint.
- virtual **LinkPtr CreateLink (ModelPtr _parent)**
Create a new body.
- virtual **ShapePtr CreateShape (const std::string &_shapeType, CollisionPtr _collision)**
*Create a collision **physics::Shape** (p. 779) object given its type.*
- virtual void **DebugPrint () const**
Debug print out of the physic engine state.
- virtual void **Fini ()**
Finilize the ODE engine.
- double **GetContactMaxCorrectingVel ()**
access functions to set ODE parameters
- double **GetContactSurfaceLayer ()**
access functions to set ODE parameters
- **math::Vector3 GetGravity () const**
Get gravity vector.
- int **GetMaxContacts ()**
access functions to set ODE parameters

- int **GetSORPGSIters** ()
access functions to set ODE parameters
- int **GetSORPGSPreconIters** ()
access functions to set ODE parameters
- double **GetSORPGSW** ()
access functions to set ODE parameters
- dSpaceID **GetSpaceId** () const
Return the space id.
- virtual double **GetStepTime** ()
Get the simulation step time.
- virtual std::string **GetStepType** () const
Get the step type.
- double **GetWorldCFM** ()
access functions to set ODE parameters
- double **GetWorldERP** ()
access functions to set ODE parameters
- dWorldID **GetWorldId** ()
Get the world id.
- virtual void **Init** ()
Initialize the ODE engine.
- virtual void **InitForThread** ()
Init the engine for threads.
- virtual void **Load** (sdf::ElementPtr _sdf)
Load the ODE engine.
- void **ProcessContactFeedback** (ContactFeedback *feedback, msgs::Contact *_msg)
*process contact feedbacks into **physics::ContactFeedback** (p. 299)*
- void **Reset** ()
Empties dynamically created contact joints.
- void **SetContactMaxCorrectingVel** (double vel)
access functions to set ODE parameters
- void **SetContactSurfaceLayer** (double layer_depth)
access functions to set ODE parameters
- virtual void **SetGravity** (const gazebo::math::Vector3 &gravity)
Set the gavity vector.
- void **SetMaxContacts** (unsigned int max_contacts)
access functions to set ODE parameters
- void **SetSORPGSIters** (unsigned int iters)
access functions to set ODE parameters
- void **SetSORPGSPreconIters** (unsigned int iters)
access functions to set ODE parameters
- void **SetSORPGSW** (double w)
access functions to set ODE parameters
- void **SetStepTime** (double _value)
Set the step time.
- virtual void **SetStepType** (const std::string &_type)
Set the step type.
- void **SetWorldCFM** (double cfm)

access functions to set ODE parameters

- void **SetWorldERP** (double erp)

access functions to set ODE parameters

- virtual void **UpdateCollision** ()

Update the ODE collision.

- virtual void **UpdatePhysics** ()

Update the ODE engine.

Static Public Member Functions

- static void **ConvertMass** (**InertialPtr** _inertial, void *odeMass)

Convert an odeMass to Mass.

- static void **ConvertMass** (void *odeMass, **InertialPtr** _inertial)

Convert an odeMass to Mass.

Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &)

act on a msgs::Physics message request

- virtual void **OnRequest** (ConstRequestPtr &)

act on a msgs::Request for physics

Additional Inherited Members

10.125.1 Detailed Description

ODE physics engine.

10.125.2 Constructor & Destructor Documentation

10.125.2.1 gazebo::physics::ODEPhysics::ODEPhysics (**WorldPtr** world)

Constructor.

10.125.2.2 virtual gazebo::physics::ODEPhysics::~~ODEPhysics () [virtual]

Destructor.

10.125.3 Member Function Documentation

10.125.3.1 void gazebo::physics::ODEPhysics::Collide (**ODECollision** * collision1, **ODECollision** * collision2, dContactGeom * contactCollisions)

Collide two collisions.

10.125.3.2 static void gazebo::physics::ODEPhysics::ConvertMass (InertialPtr *_inertial*, void * *odeMass*) [static]

Convert an odeMass to Mass.

10.125.3.3 static void gazebo::physics::ODEPhysics::ConvertMass (void * *odeMass*, InertialPtr *_inertial*) [static]

Convert an odeMass to Mass.

10.125.3.4 virtual CollisionPtr gazebo::physics::ODEPhysics::CreateCollision (const std::string & *_shapeType*, LinkPtr *_parent*) [virtual]

Create a collision.

Implements gazebo::physics::PhysicsEngine (p. 654).

10.125.3.5 void gazebo::physics::ODEPhysics::CreateContact (ODECollision * *collision1*, ODECollision * *collision2*)

Not yet implemented.

10.125.3.6 virtual JointPtr gazebo::physics::ODEPhysics::CreateJoint (const std::string & *type*, ModelPtr *_parent*) [virtual]

Create a new joint.

Implements gazebo::physics::PhysicsEngine (p. 655).

10.125.3.7 virtual LinkPtr gazebo::physics::ODEPhysics::CreateLink (ModelPtr *_parent*) [virtual]

Create a new body.

Implements gazebo::physics::PhysicsEngine (p. 655).

10.125.3.8 virtual ShapePtr gazebo::physics::ODEPhysics::CreateShape (const std::string & *_shapeType*, CollisionPtr *_collision*) [virtual]

Create a collision physics::Shape (p. 779) object given its type.

Implements gazebo::physics::PhysicsEngine (p. 655).

10.125.3.9 virtual void gazebo::physics::ODEPhysics::DebugPrint () const [virtual]

Debug print out of the physic engine state.

Implements gazebo::physics::PhysicsEngine (p. 655).

10.125.3.10 virtual void gazebo::physics::ODEPhysics::Fini () [virtual]

Finilize the ODE engine.

Reimplemented from gazebo::physics::PhysicsEngine (p. 656).

10.125.3.11 `double gazebo::physics::ODEPhysics::GetContactMaxCorrectingVel () [virtual]`

access functions to set ODE parameters

Reimplemented from **`gazebo::physics::PhysicsEngine`** (p. 656).

10.125.3.12 `double gazebo::physics::ODEPhysics::GetContactSurfaceLayer () [virtual]`

access functions to set ODE parameters

Reimplemented from **`gazebo::physics::PhysicsEngine`** (p. 656).

10.125.3.13 `math::Vector3 gazebo::physics::ODEPhysics::GetGravity () const`

Get gravity vector.

10.125.3.14 `int gazebo::physics::ODEPhysics::GetMaxContacts () [virtual]`

access functions to set ODE parameters

Reimplemented from **`gazebo::physics::PhysicsEngine`** (p. 656).

10.125.3.15 `int gazebo::physics::ODEPhysics::GetSORPGSIters () [virtual]`

access functions to set ODE parameters

Reimplemented from **`gazebo::physics::PhysicsEngine`** (p. 657).

10.125.3.16 `int gazebo::physics::ODEPhysics::GetSORPGSPreconlters () [virtual]`

access functions to set ODE parameters

Reimplemented from **`gazebo::physics::PhysicsEngine`** (p. 657).

10.125.3.17 `double gazebo::physics::ODEPhysics::GetSORPGSW () [virtual]`

access functions to set ODE parameters

Reimplemented from **`gazebo::physics::PhysicsEngine`** (p. 657).

10.125.3.18 `dSpaceID gazebo::physics::ODEPhysics::GetSpaceId () const`

Return the space id.

10.125.3.19 `virtual double gazebo::physics::ODEPhysics::GetStepTime () [virtual]`

Get the simulation step time.

Implements **`gazebo::physics::PhysicsEngine`** (p. 658).

10.125.3.20 `virtual std::string gazebo::physics::ODEPhysics::GetStepType () const [virtual]`

Get the step type.

10.125.3.21 `double gazebo::physics::ODEPhysics::GetWorldCFM () [virtual]`

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 658).

10.125.3.22 `double gazebo::physics::ODEPhysics::GetWorldERP () [virtual]`

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 658).

10.125.3.23 `dWorldID gazebo::physics::ODEPhysics::GetWorldId ()`

Get the world id.

10.125.3.24 `virtual void gazebo::physics::ODEPhysics::Init () [virtual]`

Initialize the ODE engine.

Implements **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.25 `virtual void gazebo::physics::ODEPhysics::InitForThread () [virtual]`

Init the engine for threads.

Implements **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.26 `virtual void gazebo::physics::ODEPhysics::Load (sdf::ElementPtr _sdf) [virtual]`

Load the ODE engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.27 `virtual void gazebo::physics::ODEPhysics::OnPhysicsMsg (ConstPhysicsPtr &) [protected],[virtual]`

act on a msgs::Physics message request

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.28 `virtual void gazebo::physics::ODEPhysics::OnRequest (ConstRequestPtr &) [protected],[virtual]`

act on a msgs::Request for physics

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.29 `void gazebo::physics::ODEPhysics::ProcessContactFeedback (ContactFeedback * feedback, msgs::Contact * _msg)`

process contact feedbacks into **physics::ContactFeedback** (p. 299)

10.125.3.30 `void gazebo::physics::ODEPhysics::Reset () [virtual]`

Empties dynamically created contact joints.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.31 `void gazebo::physics::ODEPhysics::SetContactMaxCorrectingVel (double vel) [virtual]`

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 660).

10.125.3.32 `void gazebo::physics::ODEPhysics::SetContactSurfaceLayer (double layer_depth) [virtual]`

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 660).

10.125.3.33 `virtual void gazebo::physics::ODEPhysics::SetGravity (const gazebo::math::Vector3 & gravity) [virtual]`

Set the gravity vector.

Implements **gazebo::physics::PhysicsEngine** (p. 660).

10.125.3.34 `void gazebo::physics::ODEPhysics::SetMaxContacts (unsigned int max_contacts)`

access functions to set ODE parameters

10.125.3.35 `void gazebo::physics::ODEPhysics::SetSORPGSIters (unsigned int iters) [virtual]`

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 661).

10.125.3.36 `void gazebo::physics::ODEPhysics::SetSORPGSPreconIters (unsigned int iters) [virtual]`

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 661).

10.125.3.37 `void gazebo::physics::ODEPhysics::SetSORPGSW (double w) [virtual]`

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 661).

10.125.3.38 `void gazebo::physics::ODEPhysics::SetStepTime (double _value) [virtual]`

Set the step time.

Implements **gazebo::physics::PhysicsEngine** (p. 661).

10.125.3.39 `virtual void gazebo::physics::ODEPhysics::SetStepType (const std::string & _type) [virtual]`

Set the step type.

10.125.3.40 `void gazebo::physics::ODEPhysics::SetWorldCFM (double cfm) [virtual]`

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 662).

10.125.3.41 `void gazebo::physics::ODEPhysics::SetWorldERP (double erp) [virtual]`

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 662).

10.125.3.42 `virtual void gazebo::physics::ODEPhysics::UpdateCollision () [virtual]`

Update the ODE collision.

Implements **gazebo::physics::PhysicsEngine** (p. 662).

10.125.3.43 `virtual void gazebo::physics::ODEPhysics::UpdatePhysics () [virtual]`

Update the ODE engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 662).

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

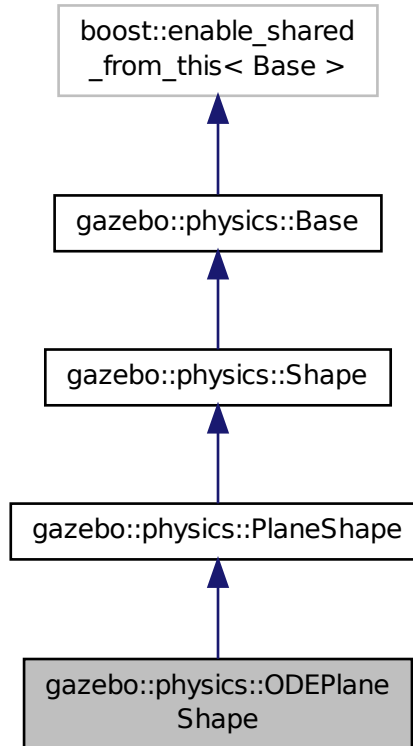
- **ODEPhysics.hh**

10.126 gazebo::physics::ODEPlaneShape Class Reference

An ODE Plane shape.

```
#include <ODEPlaneShape.hh>
```

Inheritance diagram for gazebo::physics::ODEPlaneShape:



Public Member Functions

- **ODEPlaneShape** (*CollisionPtr* _parent)
Constructor.
- virtual **~ODEPlaneShape** ()
Destructor.
- void **CreatePlane** ()
Create the plane.
- void **SetAltitude** (const **math::Vector3** &pos)
Set the altitude of the plane.

Additional Inherited Members

10.126.1 Detailed Description

An ODE Plane shape.

10.126.2 Constructor & Destructor Documentation

10.126.2.1 gazebo::physics::ODEPlaneShape::ODEPlaneShape (CollisionPtr *_parent*) [inline]

Constructor.

10.126.2.2 virtual gazebo::physics::ODEPlaneShape::~~ODEPlaneShape () [inline],[virtual]

Destructor.

10.126.3 Member Function Documentation

10.126.3.1 void gazebo::physics::ODEPlaneShape::CreatePlane () [inline],[virtual]

Create the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 673).

References gazebo::physics::Shape::collisionParent, gazebo::physics::PlaneShape::CreatePlane(), gazebo::physics::ODECollision::GetCollisionId(), gazebo::physics::PlaneShape::GetNormal(), gazebo::physics::ODECollision::GetSpaceId(), NULL, gazebo::physics::ODECollision::SetCollision(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

10.126.3.2 void gazebo::physics::ODEPlaneShape::SetAltitude (const math::Vector3 & *pos*) [inline],[virtual]

Set the altitude of the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 673).

References gazebo::physics::Shape::collisionParent, gazebo::physics::ODECollision::GetCollisionId(), gazebo::physics::PlaneShape::SetAltitude(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

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

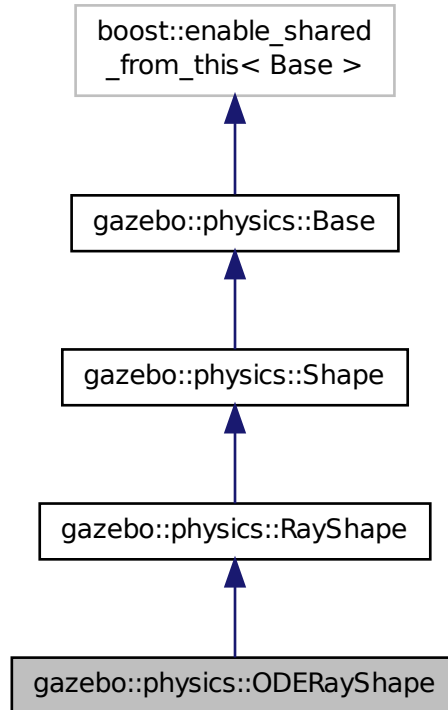
- **ODEPlaneShape.hh**

10.127 gazebo::physics::ODERayShape Class Reference

Ray collision.

```
#include <ODERayShape.hh>
```

Inheritance diagram for gazebo::physics::ODERayShape:



Public Member Functions

- **ODERayShape (PhysicsEnginePtr _physicsEngine)**
Constructor for a global ray.
- **ODERayShape (CollisionPtr collision)**
Constructor.
- virtual **~ODERayShape ()**
Destructor.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)
Get the nearest intersection.
- virtual void **SetPoints** (const **math::Vector3** &posStart, const **math::Vector3** &posEnd)
Set the ray based on starting and ending points relative to the body.
- virtual void **Update ()**
Update the ray collision.

Additional Inherited Members

10.127.1 Detailed Description

Ray collision.

10.127.2 Constructor & Destructor Documentation

10.127.2.1 gazebo::physics::ODERayShape::ODERayShape (PhysicsEnginePtr *physicsEngine*)

Constructor for a global ray.

10.127.2.2 gazebo::physics::ODERayShape::ODERayShape (CollisionPtr *collision*)

Constructor.

Parameters

<i>body</i>	Link (p. 454) the ray is attached to
-------------	---

10.127.2.3 virtual gazebo::physics::ODERayShape::~~ODERayShape () [virtual]

Destructor.

10.127.3 Member Function Documentation

10.127.3.1 virtual void gazebo::physics::ODERayShape::GetIntersection (double & *dist*, std::string & *entity*) [virtual]

Get the nearest intersection.

Implements **gazebo::physics::RayShape** (p. 720).

10.127.3.2 virtual void gazebo::physics::ODERayShape::SetPoints (const math::Vector3 & *posStart*, const math::Vector3 & *posEnd*) [virtual]

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

Parameters

<i>posStart</i>	Start position, relative the body
<i>posEnd</i>	End position, relative to the body

Reimplemented from **gazebo::physics::RayShape** (p. 721).

10.127.3.3 virtual void gazebo::physics::ODERayShape::Update () [virtual]

Update the ray collision.

Implements **gazebo::physics::RayShape** (p. 722).

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

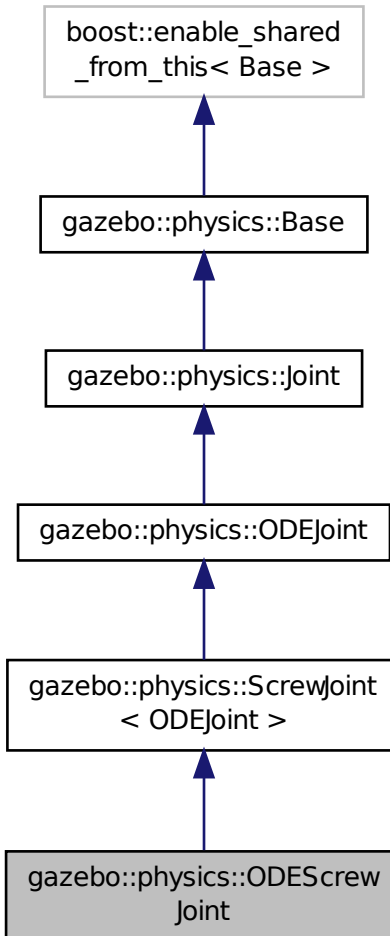
- **ODERayShape.hh**

10.128 gazebo::physics::ODEScrewJoint Class Reference

A screw joint.

```
#include <ODEScrewJoint.hh>
```

Inheritance diagram for gazebo::physics::ODEScrewJoint:



Public Member Functions

- **ODEScrewJoint** (dWorldID worldId, **BasePtr** _parent)
Constructor.
- virtual **~ODEScrewJoint** ()
Destructor.
- void **ApplyDamping** ()

callback to apply damping force to joint

- virtual **math::Angle GetAngleImpl** (int index) const
Get the position of the joint.
- virtual **math::Vector3 GetGlobalAxis** (int index) const
*Get the axis of rotation of an **ODEScrewJoint** (p. 622).*
- virtual double **GetMaxForce** (int index)
Get the max allowed force of an axis(index).
- virtual double **GetParam** (int parameter) const
Get the _parameter.
- virtual double **GetVelocity** (int index) const
Get the rate of change.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)
Set the axis of motion.
- virtual void **SetDamping** (int _index, double _damping)

- virtual void **SetForce** (int index, double force)
Set the screw force.
- virtual void **SetMaxForce** (int index, double t)
Set the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)
Set the _parameter.
- virtual void **SetThreadPitch** (int _index, double _threadPitch)
Set screw joint thread pitch.
- virtual void **SetVelocity** (int index, double angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load the **ODEScrewJoint** (p. 622) from ::Element.*

Additional Inherited Members

10.128.1 Detailed Description

A screw joint.

10.128.2 Constructor & Destructor Documentation

10.128.2.1 gazebo::physics::ODEScrewJoint::ODEScrewJoint (dWorldID worldId, BasePtr _parent)

Constructor.

10.128.2.2 virtual gazebo::physics::ODEScrewJoint::~~ODEScrewJoint () [virtual]

Destructor.

10.128.3 Member Function Documentation

10.128.3.1 void gazebo::physics::ODEScrewJoint::ApplyDamping ()

callback to apply damping force to joint

10.128.3.2 virtual math::Angle gazebo::physics::ODEScrewJoint::GetAngleImpl (int *index*) const [virtual]

Get the position of the joint.

Implements **gazebo::physics::Joint** (p. 429).

10.128.3.3 virtual math::Vector3 gazebo::physics::ODEScrewJoint::GetGlobalAxis (int *index*) const [virtual]

Get the axis of rotation of an **ODEScrewJoint** (p. 622).

Implements **gazebo::physics::Joint** (p. 430).

10.128.3.4 virtual double gazebo::physics::ODEScrewJoint::GetMaxForce (int *index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.128.3.5 virtual double gazebo::physics::ODEScrewJoint::GetParam (int *parameter*) const [virtual]

Get the `_parameter`.

Reimplemented from **gazebo::physics::ODEJoint** (p. 598).

10.128.3.6 virtual double gazebo::physics::ODEScrewJoint::GetVelocity (int *index*) const [virtual]

Get the rate of change.

Implements **gazebo::physics::Joint** (p. 433).

10.128.3.7 virtual void gazebo::physics::ODEScrewJoint::Load (sdf::ElementPtr *sdf*) [protected],[virtual]

Load the **ODEScrewJoint** (p. 622) from `::Element`.

Reimplemented from **gazebo::physics::ScrewJoint< ODEJoint >** (p. 761).

10.128.3.8 virtual void gazebo::physics::ODEScrewJoint::SetAxis (int *index*, const math::Vector3 & *axis*) [virtual]

Set the axis of motion.

Implements **gazebo::physics::Joint** (p. 435).

10.128.3.9 virtual void gazebo::physics::ODEScrewJoint::SetDamping (int *index*, double *damping*) [virtual]

Implements **gazebo::physics::Joint** (p. 435).

10.128.3.10 `virtual void gazebo::physics::ODEScrewJoint::SetForce (int index, double force) [virtual]`

Set the screw force.

Reimplemented from `gazebo::physics::Joint` (p. 435).

10.128.3.11 `virtual void gazebo::physics::ODEScrewJoint::SetMaxForce (int index, double t) [virtual]`

Set the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.128.3.12 `virtual void gazebo::physics::ODEScrewJoint::SetParam (int parameter, double value) [virtual]`

Set the `_parameter`.

Reimplemented from `gazebo::physics::ODEJoint` (p. 599).

10.128.3.13 `virtual void gazebo::physics::ODEScrewJoint::SetThreadPitch (int _index, double _threadPitch) [virtual]`

Set screw joint thread pitch.

This must be implemented in a child class

Parameters

<code>in</code>	<code>_index</code>	Index of the axis.
<code>in</code>	<code>_threadPitch</code>	Thread pitch value.

Implements `gazebo::physics::ScrewJoint< ODEJoint >` (p. 762).

10.128.3.14 `virtual void gazebo::physics::ODEScrewJoint::SetVelocity (int index, double angle) [virtual]`

Set the velocity of an axis(index).

Implements `gazebo::physics::Joint` (p. 437).

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

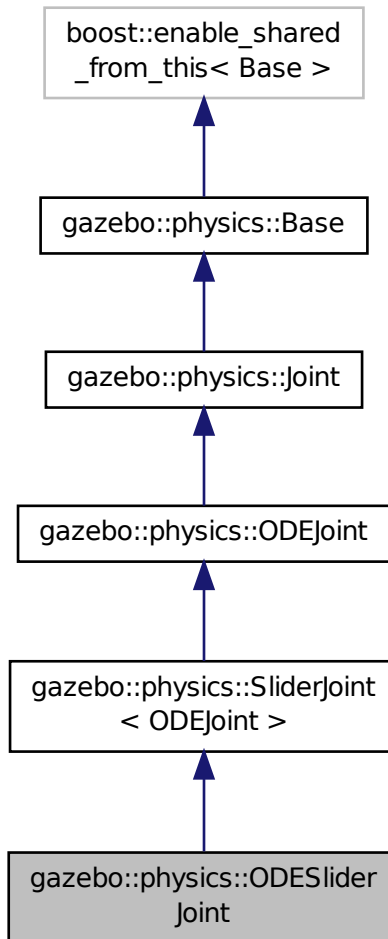
- `ODEScrewJoint.hh`

10.129 gazebo::physics::ODESliderJoint Class Reference

A slider joint.

```
#include <ODESliderJoint.hh>
```

Inheritance diagram for gazebo::physics::ODESliderJoint:



Public Member Functions

- **ODESliderJoint** (dWorldID worldId, **BasePtr** _parent)
Constructor.
- virtual **~ODESliderJoint** ()
Destructor.
- void **ApplyDamping** ()
callback to apply damping force to joint
- virtual **math::Angle GetAngleImpl** (int index) const
Get the position of the joint.
- virtual **math::Vector3 GetGlobalAxis** (int index) const
Get the axis of rotation.

- virtual double **GetMaxForce** (int index)
Get the max allowed force of an axis(index).
- virtual double **GetParam** (int parameter) const
Get the _parameter.
- virtual double **GetVelocity** (int index) const
Get the rate of change.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)
Set the axis of motion.
- virtual void **SetDamping** (int _index, double _damping)
Set joint damping, not yet implemented.
- virtual void **SetForce** (int index, double force)
Set the slider force.
- virtual void **SetMaxForce** (int index, double t)
Set the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)
Set the _parameter.
- virtual void **SetVelocity** (int index, double angle)
Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load the **ODESliderJoint** (p. 625).*

Additional Inherited Members

10.129.1 Detailed Description

A slider joint.

10.129.2 Constructor & Destructor Documentation

10.129.2.1 gazebo::physics::ODESliderJoint::ODESliderJoint (dWorldID worldId, BasePtr _parent)

Constructor.

10.129.2.2 virtual gazebo::physics::ODESliderJoint::~~ODESliderJoint () [virtual]

Destructor.

10.129.3 Member Function Documentation

10.129.3.1 void gazebo::physics::ODESliderJoint::ApplyDamping ()

callback to apply damping force to joint

10.129.3.2 `virtual math::Angle gazebo::physics::ODESliderJoint::GetAngleImpl (int index) const [virtual]`

Get the position of the joint.

Implements `gazebo::physics::Joint` (p. 429).

10.129.3.3 `virtual math::Vector3 gazebo::physics::ODESliderJoint::GetGlobalAxis (int index) const [virtual]`

Get the axis of rotation.

Implements `gazebo::physics::Joint` (p. 430).

10.129.3.4 `virtual double gazebo::physics::ODESliderJoint::GetMaxForce (int index) [virtual]`

Get the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 432).

10.129.3.5 `virtual double gazebo::physics::ODESliderJoint::GetParam (int parameter) const [virtual]`

Get the `_parameter`.

Reimplemented from `gazebo::physics::ODEJoint` (p. 598).

10.129.3.6 `virtual double gazebo::physics::ODESliderJoint::GetVelocity (int index) const [virtual]`

Get the rate of change.

Implements `gazebo::physics::Joint` (p. 433).

10.129.3.7 `virtual void gazebo::physics::ODESliderJoint::Load (sdf::ElementPtr sdf) [protected],[virtual]`

Load the `ODESliderJoint` (p. 625).

Reimplemented from `gazebo::physics::SliderJoint< ODEJoint >` (p. 805).

10.129.3.8 `virtual void gazebo::physics::ODESliderJoint::SetAxis (int index, const math::Vector3 & axis) [virtual]`

Set the axis of motion.

Implements `gazebo::physics::Joint` (p. 435).

10.129.3.9 `virtual void gazebo::physics::ODESliderJoint::SetDamping (int index, double damping) [virtual]`

Set joint damping, not yet implemented.

Implements `gazebo::physics::Joint` (p. 435).

10.129.3.10 `virtual void gazebo::physics::ODESliderJoint::SetForce (int index, double force) [virtual]`

Set the slider force.

Reimplemented from `gazebo::physics::Joint` (p. 435).

10.129.3.11 `virtual void gazebo::physics::ODESliderJoint::SetMaxForce (int index, double t) [virtual]`

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.129.3.12 `virtual void gazebo::physics::ODESliderJoint::SetParam (int parameter, double value) [virtual]`

Set the `_parameter`.

Reimplemented from **gazebo::physics::ODEJoint** (p. 599).

10.129.3.13 `virtual void gazebo::physics::ODESliderJoint::SetVelocity (int index, double angle) [virtual]`

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

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

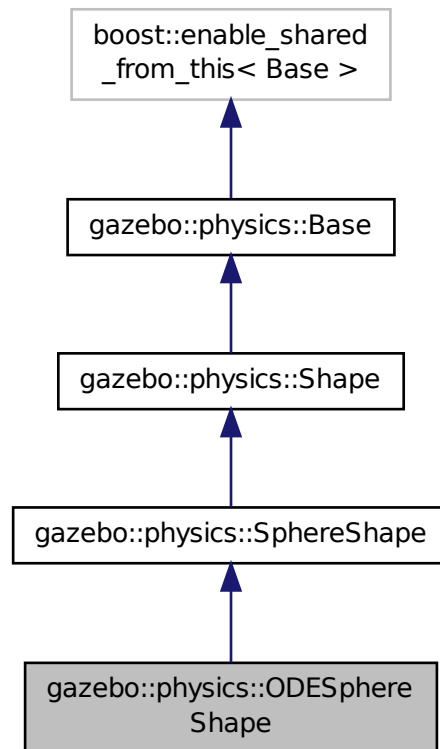
- **ODESliderJoint.hh**

10.130 gazebo::physics::ODESphereShape Class Reference

A ODE sphere shape.

```
#include <ODESphereShape.hh>
```

Inheritance diagram for gazebo::physics::ODESphereShape:



Public Member Functions

- **ODESphereShape** (`ODECollisionPtr _parent`)
Constructor.
- virtual **~ODESphereShape** ()
Destructor.
- void **SetRadius** (double `_radius`)
Set the radius.

Additional Inherited Members

10.130.1 Detailed Description

A ODE sphere shape.

10.130.2 Constructor & Destructor Documentation

10.130.2.1 gazebo::physics::ODESphereShape::ODESphereShape (ODECollisionPtr *_parent*) [inline]

Constructor.

10.130.2.2 virtual gazebo::physics::ODESphereShape::~~ODESphereShape () [inline],[virtual]

Destructor.

10.130.3 Member Function Documentation

10.130.3.1 void gazebo::physics::ODESphereShape::SetRadius (double *_radius*) [inline],[virtual]

Set the radius.

Reimplemented from **gazebo::physics::SphereShape** (p. 808).

References gazebo::physics::Shape::collisionParent, gazebo::physics::ODECollision::GetCollisionId(), NULL, gazebo::physics::ODECollision::SetCollision(), and gazebo::physics::SphereShape::SetRadius().

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

- **ODESphereShape.hh**

10.131 gazebo::physics::ODESurfaceParams Class Reference

Surface params.

```
#include <ODESurfaceParams.hh>
```

Public Member Functions

- **ODESurfaceParams** ()
Constructor.
- virtual **~ODESurfaceParams** ()
Destructor.
- virtual void **Load** (sdf::ElementPtr *_sdf*)
Load the contact params.

10.131.1 Detailed Description

Surface params.

10.131.2 Constructor & Destructor Documentation

10.131.2.1 gazebo::physics::ODESurfaceParams::ODESurfaceParams ()

Constructor.

10.131.2.2 virtual gazebo::physics::ODESurfaceParams::~~ODESurfaceParams () [virtual]

Destructor.

10.131.3 Member Function Documentation

10.131.3.1 virtual void gazebo::physics::ODESurfaceParams::Load (sdf::ElementPtr _sdf) [virtual]

Load the contact params.

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

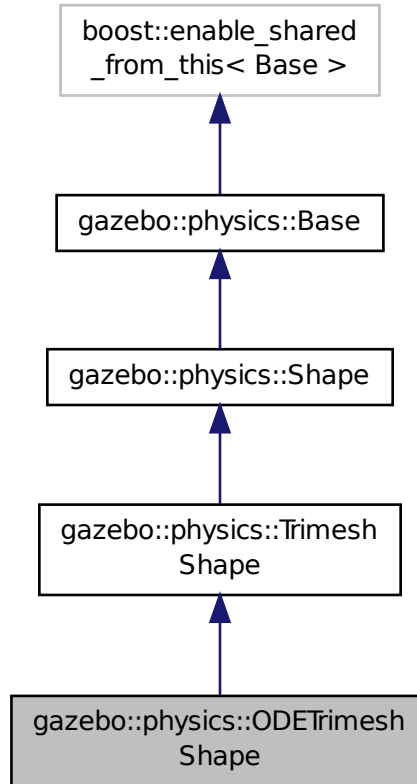
- **ODESurfaceParams.hh**

10.132 gazebo::physics::ODETrimeshShape Class Reference

Triangle mesh collision.

```
#include <ODETrimeshShape.hh>
```

Inheritance diagram for gazebo::physics::ODETrimeshShape:



Public Member Functions

- **ODETrimeshShape** (**CollisionPtr** _parent)
Constructor.
- virtual **~ODETrimeshShape** ()
Destructor.
- void **Update** ()
Update function.

Protected Member Functions

- virtual void **Init** ()
Init the trimesh shape.
- virtual void **Load** (**sdf::ElementPtr** _sdf)
Load the trimesh.

Additional Inherited Members

10.132.1 Detailed Description

Triangle mesh collision.

10.132.2 Constructor & Destructor Documentation

10.132.2.1 gazebo::physics::ODETrimeshShape::ODETrimeshShape (CollisionPtr *_parent*)

Constructor.

10.132.2.2 virtual gazebo::physics::ODETrimeshShape::~~ODETrimeshShape () [virtual]

Destructor.

10.132.3 Member Function Documentation

10.132.3.1 virtual void gazebo::physics::ODETrimeshShape::Init () [protected],[virtual]

Init the trimesh shape.

Reimplemented from **gazebo::physics::TrimeshShape** (p. 868).

10.132.3.2 virtual void gazebo::physics::ODETrimeshShape::Load (sdf::ElementPtr *_sdf*) [protected],[virtual]

Load the trimesh.

Reimplemented from **gazebo::physics::Base** (p. 154).

10.132.3.3 void gazebo::physics::ODETrimeshShape::Update () [virtual]

Update function.

Reimplemented from **gazebo::physics::TrimeshShape** (p. 868).

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

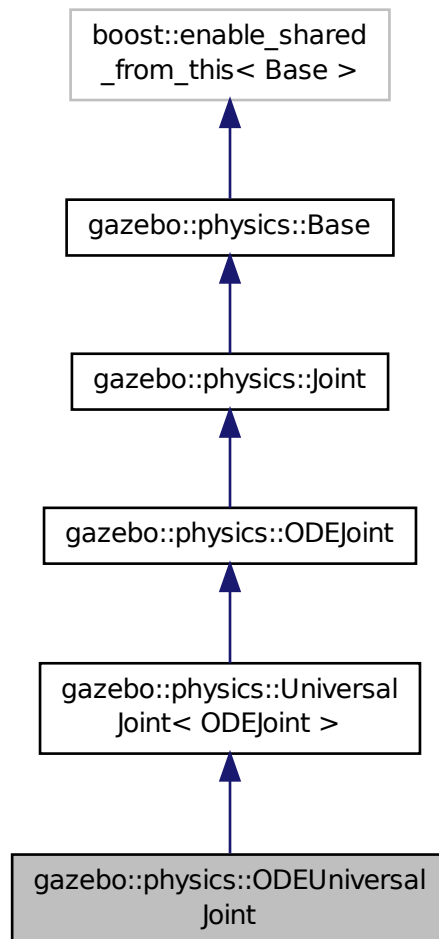
- **ODETrimeshShape.hh**

10.133 gazebo::physics::ODEUniversalJoint Class Reference

A universal joint.

```
#include <ODEUniversalJoint.hh>
```

Inheritance diagram for gazebo::physics::ODEUniversalJoint:



Public Member Functions

- **ODEUniversalJoint** (dWorldID worldId, **BasePtr** _parent)
Constructor.
- virtual **~ODEUniversalJoint** ()
Destructor.
- virtual **math::Vector3 GetAnchor** (int index) const
Get the anchor point.
- virtual **math::Angle GetAngleImpl** (int index) const
Get the angle of axis.
- virtual **math::Vector3 GetGlobalAxis** (int index) const
Get the first axis of rotation.

- virtual double **GetMaxForce** (int index)
Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int index) const
Get the angular rate of an axis.
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)
Set the anchor point.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)
Set the first axis of rotation.
- virtual void **SetDamping** (int index, double damping)
Set joint damping, not yet implemented.
- virtual void **SetForce** (int index, double torque)
Set the torque of a joint.
- virtual void **SetMaxForce** (int index, double t)
Set the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)
Set the parameter to value.
- virtual void **SetVelocity** (int index, double angle)
Set the velocity of an axis(index).

Additional Inherited Members

10.133.1 Detailed Description

A universal joint.

10.133.2 Constructor & Destructor Documentation

10.133.2.1 `gazebo::physics::ODEUniversalJoint::ODEUniversalJoint (dWorldID worldId, BasePtr _parent)`

Constructor.

10.133.2.2 `virtual gazebo::physics::ODEUniversalJoint::~~ODEUniversalJoint () [virtual]`

Destuctor.

10.133.3 Member Function Documentation

10.133.3.1 `virtual math::Vector3 gazebo::physics::ODEUniversalJoint::GetAnchor (int index) const [virtual]`

Get the anchor point.

Implements **gazebo::physics::Joint** (p. 428).

10.133.3.2 `virtual math::Angle gazebo::physics::ODEUniversalJoint::GetAngleImpl (int index) const [virtual]`

Get the angle of axis.

Implements **gazebo::physics::Joint** (p. 429).

10.133.3.3 virtual `math::Vector3` gazebo::physics::ODEUniversalJoint::GetGlobalAxis (int *index*) const [virtual]

Get the first axis of rotation.

Implements `gazebo::physics::Joint` (p. 430).

10.133.3.4 virtual double gazebo::physics::ODEUniversalJoint::GetMaxForce (int *index*) [virtual]

Get the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 432).

10.133.3.5 virtual double gazebo::physics::ODEUniversalJoint::GetVelocity (int *index*) const [virtual]

Get the angular rate of an axis.

Implements `gazebo::physics::Joint` (p. 433).

10.133.3.6 virtual void gazebo::physics::ODEUniversalJoint::SetAnchor (int *index*, const `math::Vector3` & *anchor*) [virtual]

Set the anchor point.

Implements `gazebo::physics::Joint` (p. 434).

10.133.3.7 virtual void gazebo::physics::ODEUniversalJoint::SetAxis (int *index*, const `math::Vector3` & *axis*) [virtual]

Set the first axis of rotation.

Implements `gazebo::physics::Joint` (p. 435).

10.133.3.8 virtual void gazebo::physics::ODEUniversalJoint::SetDamping (int *index*, double *damping*) [virtual]

Set joint damping, not yet implemented.

Implements `gazebo::physics::Joint` (p. 435).

10.133.3.9 virtual void gazebo::physics::ODEUniversalJoint::SetForce (int *index*, double *torque*) [virtual]

Set the torque of a joint.

Reimplemented from `gazebo::physics::Joint` (p. 435).

10.133.3.10 virtual void gazebo::physics::ODEUniversalJoint::SetMaxForce (int *index*, double *t*) [virtual]

Set the max allowed force of an axis(index).

Implements `gazebo::physics::Joint` (p. 436).

10.133.3.11 virtual void gazebo::physics::ODEUniversalJoint::SetParam (int *parameter*, double *value*) [virtual]

Set the parameter to value.

Reimplemented from `gazebo::physics::ODEJoint` (p. 599).

10.133.3.12 `virtual void gazebo::physics::ODEUniversalJoint::SetVelocity (int index, double angle) [virtual]`

Set the velocity of an axis(index).

Implements `gazebo::physics::Joint` (p. 437).

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

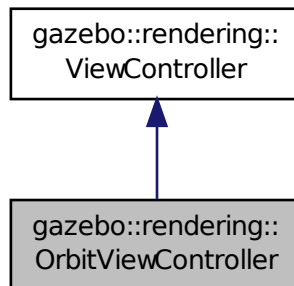
- `ODEUniversalJoint.hh`

10.134 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

10.134.1 Detailed Description

Orbit view controller.

10.134.2 Constructor & Destructor Documentation

10.134.2.1 gazebo::rendering::OrbitViewController::OrbitViewController (**UserCameraPtr** _camera)

Constructor.

Parameters

in	_camera	Pointer to the camera to control.
----	---------	-----------------------------------

10.134.2.2 virtual gazebo::rendering::OrbitViewController::~~OrbitViewController () [virtual]

Destructor.

10.134.3 Member Function Documentation

10.134.3.1 `math::Vector3 gazebo::rendering::OrbitViewController::GetFocalPoint () const`

Get the focal point.

Returns

The focal point

10.134.3.2 `static std::string gazebo::rendering::OrbitViewController::GetTypeString () [static]`

Get the type name of this view controller.

Returns

The view controller name: "orbit".

10.134.3.3 `virtual void gazebo::rendering::OrbitViewController::HandleKeyPressEvent (const std::string & _key) [virtual]`

Handle a key press event.

Parameters

in	_key	The key that was pressed.
----	------	---------------------------

Implements `gazebo::rendering::ViewController` (p. 929).

10.134.3.4 `void gazebo::rendering::OrbitViewController::HandleKeyReleaseEvent (const std::string & _key) [virtual]`

Handle a key release event.

Parameters

in	_key	The key that was released.
----	------	----------------------------

Implements `gazebo::rendering::ViewController` (p. 929).

10.134.3.5 `virtual void gazebo::rendering::OrbitViewController::HandleMouseEvent (const common::MouseEvent & _event) [virtual]`

Handle a mouse event.

Parameters

in	_event	The mouse event.
----	--------	------------------

Implements `gazebo::rendering::ViewController` (p. 929).

10.134.3.6 `virtual void gazebo::rendering::OrbitViewController::Init () [virtual]`

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 930).

10.134.3.7 `virtual void gazebo::rendering::OrbitViewController::Init (const math::Vector3 & _focalPoint) [virtual]`

Initialize the controller with a focal point.

Parameters

<code>in</code>	<code>_focalPoint</code>	Point to look at.
-----------------	--------------------------	-------------------

Reimplemented from **gazebo::rendering::ViewController** (p. 930).

10.134.3.8 `void gazebo::rendering::OrbitViewController::SetDistance (float _d)`

Set the distance to the focal point.

Parameters

<code>in</code>	<code>_d</code>	The distance from the focal point.
-----------------	-----------------	------------------------------------

10.134.3.9 `void gazebo::rendering::OrbitViewController::SetDistanceRange (double _minDist, double _maxDist)`

Set the min and max distance from the focal point.

Parameters

<code>in</code>	<code>_minDist</code>	Min distance to the focal point.
<code>in</code>	<code>_maxDist</code>	Max distance from the focal point.

10.134.3.10 `void gazebo::rendering::OrbitViewController::SetFocalPoint (const math::Vector3 & _fp)`

Set the focal point.

Parameters

<code>in</code>	<code>_fp</code>	The focal point
-----------------	------------------	-----------------

10.134.3.11 `void gazebo::rendering::OrbitViewController::SetPitch (double _pitch)`

Set the pitch angle of the camera.

Parameters

<code>in</code>	<code>Pitch</code>	angle in radians.
-----------------	--------------------	-------------------

10.134.3.12 `void gazebo::rendering::OrbitViewController::SetYaw (double _yaw)`

Set the yaw angle of the camera.

Parameters

in	Yaw	angle in radians
----	-----	------------------

10.134.3.13 virtual void gazebo::rendering::OrbitViewController::Update () [virtual]

Update.

Implements **gazebo::rendering::ViewController** (p. 930).

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

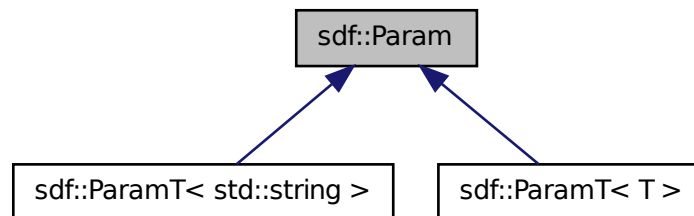
- **OrbitViewController.hh**

10.135 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**

10.135.1 Detailed Description

A parameter class.

10.135.2 Constructor & Destructor Documentation

10.135.2.1 sdf::Param::Param (Param * _newParam)

Constructor.

10.135.2.2 virtual sdf::Param::~~Param () [virtual]

Destructor.

10.135.3 Member Function Documentation

10.135.3.1 virtual Param* sdf::Param::Clone () const [pure virtual]

Implemented in **sdf::ParamT< T >** (p. 650), and **sdf::ParamT< std::string >** (p. 650).

10.135.3.2 bool sdf::Param::Get (bool & _value)

10.135.3.3 bool sdf::Param::Get (int & _value)

10.135.3.4 bool sdf::Param::Get (unsigned int & _value)

10.135.3.5 bool sdf::Param::Get (float & _value)

10.135.3.6 bool sdf::Param::Get (double & _value)

10.135.3.7 bool sdf::Param::Get (char & _value)

10.135.3.8 `bool sdf::Param::Get (std::string & _value)`

10.135.3.9 `bool sdf::Param::Get (gazebo::math::Vector3 & _value)`

10.135.3.10 `bool sdf::Param::Get (gazebo::math::Vector2i & _value)`

10.135.3.11 `bool sdf::Param::Get (gazebo::math::Vector2d & _value)`

10.135.3.12 `bool sdf::Param::Get (gazebo::math::Quaternion & _value)`

10.135.3.13 `bool sdf::Param::Get (gazebo::math::Pose & _value)`

10.135.3.14 `bool sdf::Param::Get (gazebo::common::Color & _value)`

10.135.3.15 `bool sdf::Param::Get (gazebo::common::Time & _value)`

10.135.3.16 `virtual std::string sdf::Param::GetAsString () const [inline],[virtual]`

Get the type.

Reimplemented in `sdf::ParamT< T >` (p. 650), and `sdf::ParamT< std::string >` (p. 650).

10.135.3.17 `virtual std::string sdf::Param::GetDefaultAsString () const [inline],[virtual]`

Reimplemented in `sdf::ParamT< T >` (p. 650), and `sdf::ParamT< std::string >` (p. 650).

10.135.3.18 `std::string sdf::Param::GetDescription () const`

Get the description of the parameter.

10.135.3.19 `const std::string& sdf::Param::GetKey () const [inline]`

References key.

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

10.135.3.20 `bool sdf::Param::GetRequired () const [inline]`

References required.

10.135.3.21 `bool sdf::Param::GetSet () const [inline]`

Return true if the parameter has been set.

10.135.3.22 `std::string sdf::Param::GetTypeNames () const`

10.135.3.23 `bool sdf::Param::IsBool () const`

10.135.3.24 `bool sdf::Param::IsChar () const`

- 10.135.3.25 `bool sdf::Param::IsColor () const`
- 10.135.3.26 `bool sdf::Param::IsDouble () const`
- 10.135.3.27 `bool sdf::Param::IsFloat () const`
- 10.135.3.28 `bool sdf::Param::IsInt () const`
- 10.135.3.29 `bool sdf::Param::IsPose () const`
- 10.135.3.30 `bool sdf::Param::IsQuaternion () const`
- 10.135.3.31 `bool sdf::Param::IsStr () const`
- 10.135.3.32 `bool sdf::Param::IsTime () const`
- 10.135.3.33 `bool sdf::Param::IsUInt () const`
- 10.135.3.34 `bool sdf::Param::IsVector2d () const`
- 10.135.3.35 `bool sdf::Param::IsVector2i () const`
- 10.135.3.36 `bool sdf::Param::IsVector3 () const`
- 10.135.3.37 `virtual void sdf::Param::Reset () [pure virtual]`

Reset the parameter.

Implemented in `sdf::ParamT< T >` (p. 650), and `sdf::ParamT< std::string >` (p. 650).

- 10.135.3.38 `bool sdf::Param::Set (const bool & _value)`

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

- 10.135.3.39 `bool sdf::Param::Set (const int & _value)`
- 10.135.3.40 `bool sdf::Param::Set (const unsigned int & _value)`
- 10.135.3.41 `bool sdf::Param::Set (const float & _value)`
- 10.135.3.42 `bool sdf::Param::Set (const double & _value)`
- 10.135.3.43 `bool sdf::Param::Set (const char & _value)`
- 10.135.3.44 `bool sdf::Param::Set (const std::string & _value)`
- 10.135.3.45 `bool sdf::Param::Set (const char * _value)`
- 10.135.3.46 `bool sdf::Param::Set (const gazebo::math::Vector3 & _value)`
- 10.135.3.47 `bool sdf::Param::Set (const gazebo::math::Vector2i & _value)`

10.135.3.48 `bool sdf::Param::Set (const gazebo::math::Vector2d & _value)`

10.135.3.49 `bool sdf::Param::Set (const gazebo::math::Quaternion & _value)`

10.135.3.50 `bool sdf::Param::Set (const gazebo::math::Pose & _value)`

10.135.3.51 `bool sdf::Param::Set (const gazebo::common::Color & _value)`

10.135.3.52 `bool sdf::Param::Set (const gazebo::common::Time & _value)`

10.135.3.53 `void sdf::Param::SetDescription (const std::string & _desc)`

Set the description of the parameter.

10.135.3.54 `virtual bool sdf::Param::SetFromString (const std::string &) [inline], [virtual]`

Set the parameter value from a string.

Reimplemented in `sdf::ParamT< T >` (p. 650), and `sdf::ParamT< std::string >` (p. 650).

10.135.3.55 `template<typename T> void sdf::Param::SetUpdateFunc (T _updateFunc) [inline]`

Update function.

References updateFunc.

10.135.3.56 `virtual void sdf::Param::Update () [pure virtual]`

Implemented in `sdf::ParamT< T >` (p. 651), and `sdf::ParamT< std::string >` (p. 651).

10.135.4 Member Data Documentation

10.135.4.1 `std::string sdf::Param::description [protected]`

Referenced by `sdf::ParamT< std::string >::Clone()`, and `sdf::ParamT< std::string >::ParamT()`.

10.135.4.2 `std::string sdf::Param::key [protected]`

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

10.135.4.3 `bool sdf::Param::required [protected]`

Referenced by `sdf::ParamT< std::string >::Clone()`, `GetRequired()`, `sdf::ParamT< std::string >::ParamT()`, and `sdf::ParamT< std::string >::Set()`.

10.135.4.4 `bool sdf::Param::set [protected]`

10.135.4.5 `std::string sdf::Param::typeName` [protected]

Referenced by `sdf::ParamT< std::string >::Clone()`, and `sdf::ParamT< std::string >::ParamT()`.

10.135.4.6 `boost::function<boost::any ()> sdf::Param::updateFunc` [protected]

Referenced by `SetUpUpdateFunc()`, and `sdf::ParamT< std::string >::Update()`.

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

- **Param.hh**

10.136 `ParamT< T >` Class Template Reference

```
#include <CommonTypes.hh>
```

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

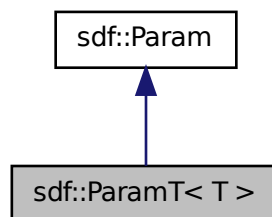
- **CommonTypes.hh**

10.137 `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** ()

Protected Attributes

- T **defaultValue**
- T **value**

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **ParamT**< T > &_p)

10.137.1 Detailed Description

template<typename T>class sdf::ParamT< T >

Templatized parameter class.

10.137.2 Constructor & Destructor Documentation

10.137.2.1 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.

10.137.2.2 template<typename T> virtual sdf::ParamT< T >::~~ParamT () [inline],[virtual]

Destructor.

10.137.3 Member Function Documentation

10.137.3.1 `template<typename T> virtual Param* sdf::ParamT< T >::Clone () const [inline],[virtual]`

Implements **sdf::Param** (p. 644).

10.137.3.2 `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. 645).

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

10.137.3.3 `template<typename T> virtual std::string sdf::ParamT< T >::GetDefaultAsString () const [inline],[virtual]`

Reimplemented from **sdf::Param** (p. 645).

10.137.3.4 `template<typename T> T sdf::ParamT< T >::GetDefaultValue () const [inline]`

Get the value.

10.137.3.5 `template<typename T> T sdf::ParamT< T >::GetValue () const [inline]`

Get the value.

10.137.3.6 `template<typename T> T sdf::ParamT< T >::operator*() const [inline]`

10.137.3.7 `template<typename T> virtual void sdf::ParamT< T >::Reset () [inline],[virtual]`

Reset to default value.

Implements **sdf::Param** (p. 646).

10.137.3.8 `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()`.

10.137.3.9 `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. 647).

10.137.3.10 `template<typename T> void sdf::ParamT< T >::SetValue (const T & _value) [inline]`

Set the value of the parameter.

10.137.3.11 `template<typename T> virtual void sdf::ParamT< T >::Update () [inline],[virtual]`

Implements `sdf::Param` (p. 647).

10.137.4 Friends And Related Function Documentation

10.137.4.1 `template<typename T> std::ostream& operator<< (std::ostream & _out, const ParamT< T > & _p) [friend]`

10.137.5 Member Data Documentation

10.137.5.1 `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()`.

10.137.5.2 `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:

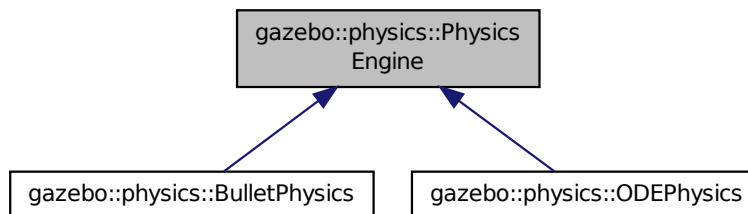
- `Param.hh`

10.138 gazebo::physics::PhysicsEngine Class Reference

Base (p. 145) class for a physics engine.

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

Inheritance diagram for `gazebo::physics::PhysicsEngine`:



Public Member Functions

- **PhysicsEngine** (**WorldPtr** _world)
Default constructor.
- virtual **~PhysicsEngine** ()
Destructor.
- virtual **CollisionPtr CreateCollision** (const std::string &_shapeType, **LinkPtr** _link)=0
Create a collision.
- **CollisionPtr CreateCollision** (const std::string &_shapeType, const std::string &_linkName)
Create a collision.
- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent)=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. 779) 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
- 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.
- **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. 663).*
- virtual int **GetSORPGSIters** ()
: Remove this function, and replace it with a more generic property map
- virtual int **GetSORPGSPreconlters** ()
: Remove this function, and replace it with a more generic property map
- virtual double **GetSORPGSW** ()
: Remove this function, and replace it with a more generic property map.
- virtual double **GetStepTime** ()=0
Get the simulation step time.
- double **GetUpdatePeriod** ()
Get the simulation update period.
- double **GetUpdateRate** ()
Get the simulation update rate.
- virtual double **GetWorldCFM** ()
: Remove this function, and replace it with a more generic property map
- virtual double **GetWorldERP** ()

- : Remove this function, and replace it with a more generic property map*
- virtual void **Init** ()=0
 - Initialize the physics engine.*
- virtual void **InitForThread** ()=0
 - Init the engine for threads.*
- virtual void **Load** (sdf::ElementPtr _sdf)
 - Load the physics engine.*
- virtual void **Reset** ()
 - Rest the physics engine.*
- virtual void **SetAutoDisableFlag** (bool _autoDisable)
 - : Remove this function, and replace it with a more generic property map*
- virtual void **SetContactMaxCorrectingVel** (double _vel)
 - : Remove this function, and replace it with a more generic property map*
- virtual void **SetContactSurfaceLayer** (double _layerDepth)
 - : Remove this function, and replace it with a more generic property map*
- virtual void **SetGravity** (const gazebo::math::Vector3 &_gravity)=0
 - Set the gavity vector.*
- virtual void **SetMaxContacts** (double _maxContacts)
 - : Remove this function, and replace it with a more generic property map*
- virtual void **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

- **transport::PublisherPtr contactPub**
Contact (p. 296) publisher.
- **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.

10.138.1 Detailed Description

Base (p. 145) class for a physics engine.

10.138.2 Constructor & Destructor Documentation

10.138.2.1 gazebo::physics::PhysicsEngine::PhysicsEngine (WorldPtr *world*) [explicit]

Default constructor.

Parameters

in	<i>_world</i>	Pointer to the world.
----	---------------	-----------------------

10.138.2.2 virtual gazebo::physics::PhysicsEngine::~PhysicsEngine () [virtual]

Destructor.

10.138.3 Member Function Documentation

10.138.3.1 virtual CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & *_shapeType*, LinkPtr *link*) [pure virtual]

Create a collision.

Parameters

in	<i>_shapeType</i>	Type of collision to create.
in	<i>_link</i>	Parent link.

Implemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.2 **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.

10.138.3.3 **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. 521) parent.

Implemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.4 **virtual LinkPtr** gazebo::physics::PhysicsEngine::CreateLink (ModelPtr *_parent*) [pure virtual]

Create a new body.

Parameters

in	<i>_parent</i>	Parent model for the link.
----	----------------	----------------------------

Implemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.5 **virtual ShapePtr** gazebo::physics::PhysicsEngine::CreateShape (const std::string & *_shapeType*, CollisionPtr *_collision*) [pure virtual]

Create a **physics::Shape** (p. 779) object.

Parameters

in	<i>_shapeType</i>	Type of shape to create.
in	<i>_collision</i>	Collision (p. 262) parent.

Implemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.6 **virtual void** gazebo::physics::PhysicsEngine::DebugPrint () const [pure virtual]

Debug print out of the physics engine state.

Implemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.7 virtual void gazebo::physics::PhysicsEngine::Fini () [virtual]

Finalize the physics engine.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.8 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.

10.138.3.9 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.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.10 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. 296) surface layer depth.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.11 math::Vector3 gazebo::physics::PhysicsEngine::GetGravity () const

Return the gravity vector.

Returns

The gravity vector.

10.138.3.12 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.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.13 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::GetPhysicsUpdateMutex () const [inline]`

returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 663).

Returns

Pointer to the physics mutex.

References physicsUpdateMutex.

10.138.3.14 `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.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.15 `virtual int gazebo::physics::PhysicsEngine::GetSORPGSPreconIters () [inline],[virtual]`

: Remove this function, and replace it with a more generic property map access functions to set ODE parameters.

Returns

SORPGS precondition iterations.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.16 `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.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.17 `virtual double gazebo::physics::PhysicsEngine::GetStepTime ()` [pure virtual]

Get the simulation step time.

Returns

Simulation step time.

Implemented in `gazebo::physics::ODEPhysics` (p. 614), and `gazebo::physics::BulletPhysics` (p. 206).

10.138.3.18 `double gazebo::physics::PhysicsEngine::GetUpdatePeriod ()`

Get the simulation update period.

Returns

Simulation update period.

10.138.3.19 `double gazebo::physics::PhysicsEngine::GetUpdateRate ()`

Get the simulation update rate.

Returns

Update rate.

10.138.3.20 `virtual double gazebo::physics::PhysicsEngine::GetWorldCFM ()` [inline],[virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 954) CFM.

Returns

World (p. 954) CFM.

Reimplemented in `gazebo::physics::ODEPhysics` (p. 615).

10.138.3.21 `virtual double gazebo::physics::PhysicsEngine::GetWorldERP ()` [inline],[virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 954) ERP.

Returns

World (p. 954) ERP.

Reimplemented in `gazebo::physics::ODEPhysics` (p. 615).

10.138.3.22 `virtual void gazebo::physics::PhysicsEngine::Init () [pure virtual]`

Initialize the physics engine.

Implemented in **gazebo::physics::ODEPhysics** (p. 615), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.23 `virtual void gazebo::physics::PhysicsEngine::InitForThread () [pure virtual]`

Init the engine for threads.

Implemented in **gazebo::physics::ODEPhysics** (p. 615), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.24 `virtual void gazebo::physics::PhysicsEngine::Load (sdf::ElementPtr _sdf) [virtual]`

Load the physics engine.

Parameters

in	_sdf	Pointer to the SDF parameters.
----	------	--------------------------------

Reimplemented in **gazebo::physics::ODEPhysics** (p. 615), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.25 `virtual void gazebo::physics::PhysicsEngine::OnPhysicsMsg (ConstPhysicsPtr & _msg) [protected], [virtual]`

virtual callback for gztopic "~/physics".

Parameters

in	_msg	Physics message.
----	------	------------------

Reimplemented in **gazebo::physics::ODEPhysics** (p. 615).

10.138.3.26 `virtual void gazebo::physics::PhysicsEngine::OnRequest (ConstRequestPtr & _msg) [protected], [virtual]`

virtual callback for gztopic "~/request".

Parameters

in	_msg	Request message.
----	------	------------------

Reimplemented in **gazebo::physics::ODEPhysics** (p. 615).

10.138.3.27 `virtual void gazebo::physics::PhysicsEngine::Reset () [inline], [virtual]`

Rest the physics engine.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 616).

10.138.3.28 `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	<i>_autoDisable</i>	True to enable auto disabling of bodies.
----	---------------------	--

10.138.3.29 `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.
----	-------------	--------------------------

Reimplemented in **`gazebo::physics::ODEPhysics`** (p. 616).

10.138.3.30 `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
----	--------------------	---------------------

Reimplemented in **`gazebo::physics::ODEPhysics`** (p. 616).

10.138.3.31 `virtual void gazebo::physics::PhysicsEngine::SetGravity (const gazebo::math::Vector3 & _gravity) [pure virtual]`

Set the gavity vector.

Parameters

in	<i>_gravity</i>	New gravity vector.
----	-----------------	---------------------

Implemented in **`gazebo::physics::ODEPhysics`** (p. 616), and **`gazebo::physics::BulletPhysics`** (p. 206).

10.138.3.32 `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	_maxContacts	Maximum number of contacts.
----	--------------	-----------------------------

10.138.3.33 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSlters (unsigned int _iters) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	_iter	Number of iterations.
----	-------	-----------------------

Reimplemented in **gazebo::physics::ODEPhysics** (p. 616).

10.138.3.34 `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

in	_iter	Number of iterations.
----	-------	-----------------------

Reimplemented in **gazebo::physics::ODEPhysics** (p. 616).

10.138.3.35 `virtual void gazebo::physics::PhysicsEngine::SetSORPGSW (double _w) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	_w	SORPGSW value.
----	----	----------------

Reimplemented in **gazebo::physics::ODEPhysics** (p. 616).

10.138.3.36 `virtual void gazebo::physics::PhysicsEngine::SetStepTime (double _value) [pure virtual]`

Set the simulation step time.

Parameters

in	_value	Value of the step time.
----	--------	-------------------------

Implemented in **gazebo::physics::ODEPhysics** (p. 617), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.37 `void gazebo::physics::PhysicsEngine::SetUpdateRate (double _value)`

Set the simulation update rate.

Parameters

in	_value	Value of the update rate.
----	--------	---------------------------

10.138.3.38 `virtual void gazebo::physics::PhysicsEngine::SetWorldCFM (double _cfm) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	_cfm	Constraint force mixing.
----	------	--------------------------

Reimplemented in `gazebo::physics::ODEPhysics` (p. 617).

10.138.3.39 `virtual void gazebo::physics::PhysicsEngine::SetWorldERP (double _erp) [virtual]`

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	_erp	Error reduction parameter.
----	------	----------------------------

Reimplemented in `gazebo::physics::ODEPhysics` (p. 617).

10.138.3.40 `virtual void gazebo::physics::PhysicsEngine::UpdateCollision () [pure virtual]`

Update the physics engine collision.

Implemented in `gazebo::physics::ODEPhysics` (p. 617), and `gazebo::physics::BulletPhysics` (p. 206).

10.138.3.41 `virtual void gazebo::physics::PhysicsEngine::UpdatePhysics () [inline],[virtual]`

Update the physics engine.

Reimplemented in `gazebo::physics::ODEPhysics` (p. 617), and `gazebo::physics::BulletPhysics` (p. 206).

10.138.4 Member Data Documentation

10.138.4.1 `transport::PublisherPtr gazebo::physics::PhysicsEngine::contactPub [protected]`

Contact (p. 296) publisher.

10.138.4.2 `transport::NodePtr gazebo::physics::PhysicsEngine::node [protected]`

Node for communication.

10.138.4.3 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::physicsSub` [protected]

Subscribe to the physics topic.

10.138.4.4 `boost::recursive_mutex* gazebo::physics::PhysicsEngine::physicsUpdateMutex` [protected]

Mutex to protect the update cycle.

Referenced by `GetPhysicsUpdateMutex()`.

10.138.4.5 `transport::SubscriberPtr gazebo::physics::PhysicsEngine::requestSub` [protected]

Subscribe to the request topic.

10.138.4.6 `transport::PublisherPtr gazebo::physics::PhysicsEngine::responsePub` [protected]

Response publisher.

10.138.4.7 `sdf::ElementPtr gazebo::physics::PhysicsEngine::sdf` [protected]

Our SDF values.

10.138.4.8 `WorldPtr gazebo::physics::PhysicsEngine::world` [protected]

Pointer to the world.

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

- **PhysicsEngine.hh**

10.139 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.

10.139.1 Detailed Description

The physics factory instantiates different physics engines.

10.139.2 Member Function Documentation

10.139.2.1 `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. 954) to pass to the created physics engine.

10.139.2.2 `static void gazebo::physics::PhysicsFactory::RegisterAll () [static]`

Register everything.

10.139.2.3 `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**

10.140 gazebo::common::PID Class Reference

Generic **PID** (p. 664) controller class.

```
#include <PID.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. 664) controller.*
- void **GetErrors** (double &_pe, double &_ie, double &_de)
*Return **PID** (p. 664) 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]-[1:12].

- **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. 664) 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.

10.140.1 Detailed Description

Generic **PID** (p. 664) 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.

10.140.2 Constructor & Destructor Documentation

10.140.2.1 **gazebo::common::PID::PID** (double *p* = 0.0, double *i* = 0.0, double *d* = 0.0, double *imax* = 0.0, double *imin* = 0.0, double *cmdMax* = 0.0, double *cmdMin* = 0.0)

Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[1:12].

Parameters

in	<i>_p</i>	The proportional gain.
in	<i>_i</i>	The integral gain.
in	<i>_d</i>	The derivative gain.
in	<i>_imax</i>	The integral upper limit.
in	<i>_imin</i>	The integral lower limit.

10.140.2.2 `virtual gazebo::common::PID::~~PID () [virtual]`

Destructor.

10.140.3 Member Function Documentation

10.140.3.1 `double gazebo::common::PID::GetCmd ()`

Return current command for this **PID** (p. 664) controller.

Returns

the command value

10.140.3.2 `void gazebo::common::PID::GetErrors (double & _pe, double & _ie, double & _de)`

Return **PID** (p. 664) 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.

10.140.3.3 `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.

10.140.3.4 `PID& gazebo::common::PID::operator= (const PID & _p) [inline]`

Assignment operator.

Parameters

in	<code>_p</code>	a reference to a PID (p. 664) to assign values from
----	-----------------	--

Returns

reference to this instance

References Reset().

10.140.3.5 void gazebo::common::PID::Reset ()

Reset the errors and command.

Referenced by operator=().

10.140.3.6 void gazebo::common::PID::SetCmd (double *_cmd*)

Set current target command for this **PID** (p. 664) controller.

Parameters

<i>in</i>	<i>_cmd</i>	New command
-----------	-------------	-------------

10.140.3.7 void gazebo::common::PID::SetCmdMax (double *_c*)

Set the maximum value for the command.

Parameters

<i>in</i>	<i>_c</i>	The maximum value
-----------	-----------	-------------------

10.140.3.8 void gazebo::common::PID::SetCmdMin (double *_c*)

Set the maximum value for the command.

Parameters

<i>in</i>	<i>_c</i>	The maximum value
-----------	-----------	-------------------

10.140.3.9 void gazebo::common::PID::SetDGain (double *_d*)

Set the derivative Gain.

Parameters

<i>in</i>	<i>_p</i>	derivative gain value
-----------	-----------	-----------------------

10.140.3.10 void gazebo::common::PID::SetIGain (double *_i*)

Set the integral Gain.

Parameters

<i>in</i>	<i>_p</i>	integral gain value
-----------	-----------	---------------------

10.140.3.11 void gazebo::common::PID::SetIMax (double *_i*)

Set the integral upper limit.

Parameters

<code>in</code>	<code>_p</code>	integral upper limit value
-----------------	-----------------	----------------------------

10.140.3.12 `void gazebo::common::PID::SetIMin (double i)`

Set the integral lower limit.

Parameters

<code>in</code>	<code>_p</code>	integral lower limit value
-----------------	-----------------	----------------------------

10.140.3.13 `void gazebo::common::PID::SetPGain (double p)`

Set the proportional Gain.

Parameters

<code>in</code>	<code>_p</code>	proportional gain value
-----------------	-----------------	-------------------------

10.140.3.14 `double gazebo::common::PID::Update (double error, common::Time dt)`

Update the Pid loop with nonuniform time step size.

Parameters

<code>in]</code>	<code>_error</code>	Error since last call (<code>p_state - p_target</code>).
<code>in]</code>	<code>_dt</code>	Change in time since last update call. Normally, this is called at every time step, The return value is an updated command to be passed to the object being controlled.

Returns

the command value

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

- **PID.hh**

10.141 gazebo::math::Plane Class Reference

A plane and related functions.

```
#include <Plane.hh>
```

Public Member Functions

- **Plane** ()
Constructor.
- **Plane** (const **Vector3** &`_normal`, double `_offset=0.0`)

Constructor from a normal and a distance.

- **Plane** (const **Vector3** &_normal, const **Vector2d** &_size, double _offset)

Constructor.

- virtual ~**Plane** ()

Destructor.

- double **Distance** (const **Vector3** &_origin, const **Vector3** &_dir) const

Get distance to the plane give an origin and direction.

- **Plane** & **operator=** (const **Plane** &_p)

Equal operator.

- void **Set** (const **Vector3** &_normal, const **Vector2d** &_size, double offset)

Set the plane.

Public Attributes

- double **d**

Plane (p. 668) offset.

- **Vector3** **normal**

Plane (p. 668) normal.

- **Vector2d** **size**

Plane (p. 668) size.

10.141.1 Detailed Description

A plane and related functions.

10.141.2 Constructor & Destructor Documentation

10.141.2.1 gazebo::math::Plane::Plane ()

Constructor.

10.141.2.2 gazebo::math::Plane::Plane (const **Vector3** &_normal, double _offset = 0.0)

Constructor from a normal and a distance.

Parameters

in	<code>_normal</code>	The plane normal
in	<code>_offset</code>	Offset along the normal

10.141.2.3 gazebo::math::Plane::Plane (const **Vector3** &_normal, const **Vector2d** &_size, double _offset)

Constructor.

Parameters

in	<code>_normal</code>	The plane normal
in	<code>_size</code>	Size of the plane

in	<code>_offset</code>	Offset along the normal
----	----------------------	-------------------------

10.141.2.4 `virtual gazebo::math::Plane::~~Plane () [virtual]`

Destructor.

10.141.3 Member Function Documentation

10.141.3.1 `double gazebo::math::Plane::Distance (const Vector3 & _origin, const Vector3 & _dir) const`

Get distance to the plane give an origin and direction.

Parameters

in	<code>_origin</code>	the origin
in	<code>_dir</code>	a direction

Returns

the shortest distance

10.141.3.2 `Plane& gazebo::math::Plane::operator= (const Plane & _p)`

Equal operator.

Parameters

	<code>_p</code>	another plane
--	-----------------	---------------

Returns

itself

10.141.3.3 `void gazebo::math::Plane::Set (const Vector3 & _normal, const Vector2d & _size, double offset)`

Set the plane.

Parameters

in	<code>_normal</code>	The plane normal
in	<code>_size</code>	Size of the plane
in	<code>_offset</code>	Offset along the normal

10.141.4 Member Data Documentation

10.141.4.1 double gazebo::math::Plane::d

Plane (p. 668) offset.

10.141.4.2 Vector3 gazebo::math::Plane::normal

Plane (p. 668) normal.

10.141.4.3 Vector2d gazebo::math::Plane::size

Plane (p. 668) size.

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

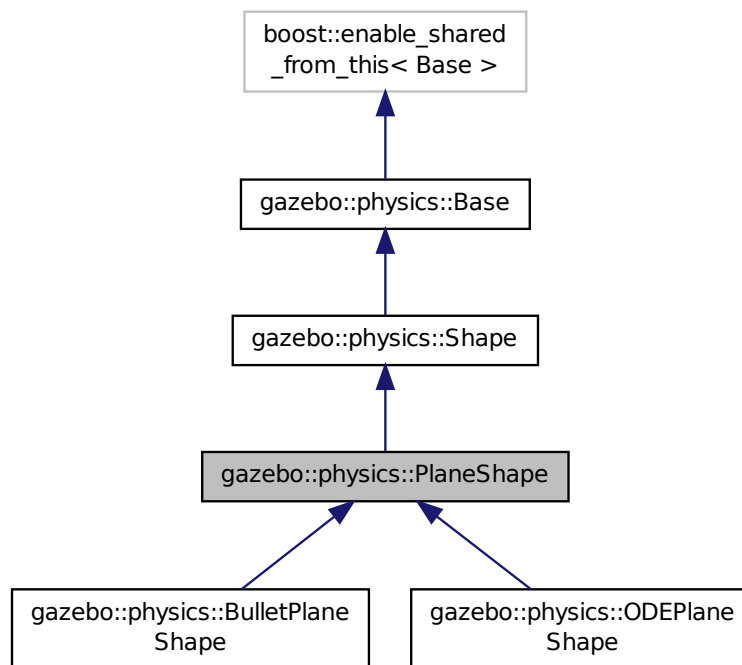
- **Plane.hh**

10.142 gazebo::physics::PlaneShape Class Reference

Collision (p. 262) for an infinite plane.

```
#include <PlaneShape.hh>
```

Inheritance diagram for gazebo::physics::PlaneShape:



Public Member Functions

- **PlaneShape** (**CollisionPtr** parent)
Constructor.
- virtual \sim **PlaneShape** ()
Destructor.
- virtual void **CreatePlane** ()
Create the plane.
- void **FillMsg** (msgs::Geometry &_msg)
- **math::Vector3** **GetNormal** () const
- **math::Vector2d** **GetSize** () const
Get the size.
- virtual void **Init** ()
Initialize the plane.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
- 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

10.142.1 Detailed Description

Collision (p. 262) 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.

10.142.2 Constructor & Destructor Documentation

10.142.2.1 gazebo::physics::PlaneShape::PlaneShape (**CollisionPtr** parent)

Constructor.

Parameters

<i>body</i>	Link (p. 454) to which we are attached.
-------------	--

10.142.2.2 virtual gazebo::physics::PlaneShape::~~PlaneShape () [virtual]

Destructor.

10.142.3 Member Function Documentation

10.142.3.1 `virtual void gazebo::physics::PlaneShape::CreatePlane () [virtual]`

Create the plane.

Reimplemented in **`gazebo::physics::BulletPlaneShape`** (p. 208), and **`gazebo::physics::ODEPlaneShape`** (p. 619).

Referenced by `gazebo::physics::ODEPlaneShape::CreatePlane()`, and `gazebo::physics::BulletPlaneShape::CreatePlane()`.

10.142.3.2 `void gazebo::physics::PlaneShape::FillMsg (msgs::Geometry & _msg) [virtual]`

Implements **`gazebo::physics::Shape`** (p. 781).

10.142.3.3 `math::Vector3 gazebo::physics::PlaneShape::GetNormal () const`

Referenced by `gazebo::physics::ODEPlaneShape::CreatePlane()`, and `gazebo::physics::BulletPlaneShape::CreatePlane()`.

10.142.3.4 `math::Vector2d gazebo::physics::PlaneShape::GetSize () const`

Get the size.

10.142.3.5 `virtual void gazebo::physics::PlaneShape::Init () [virtual]`

Initialize the plane.

Implements **`gazebo::physics::Shape`** (p. 781).

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

Implements **`gazebo::physics::Shape`** (p. 782).

10.142.3.7 `virtual void gazebo::physics::PlaneShape::SetAltitude (const math::Vector3 & _pos) [virtual]`

Set the altitude of the plane.

Reimplemented in **`gazebo::physics::ODEPlaneShape`** (p. 619), and **`gazebo::physics::BulletPlaneShape`** (p. 208).

Referenced by `gazebo::physics::BulletPlaneShape::SetAltitude()`, and `gazebo::physics::ODEPlaneShape::SetAltitude()`.

10.142.3.8 `void gazebo::physics::PlaneShape::SetNormal (const math::Vector3 & _norm)`

Set the normal.

10.142.3.9 `void gazebo::physics::PlaneShape::SetSize (const math::Vector2d & _size)`

Set the size.

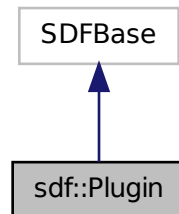
The documentation for this class was generated from the following file:

- **`PlaneShape.hh`**

10.143 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**

10.143.1 Constructor & Destructor Documentation

10.143.1.1 `sdf::Plugin::Plugin ()` [inline]

10.143.2 Member Function Documentation

10.143.2.1 `void sdf::Plugin::Clear ()` [inline]

References data.

10.143.2.2 `void sdf::Plugin::Print (const std::string & prefix)` [inline]

References filename, and name.

10.143.3 Member Data Documentation

10.143.3.1 `std::vector<ParamT<std::string>>` `sdf::Plugin::data`

Referenced by `Clear()`.

10.143.3.2 `ParamT<std::string>` `sdf::Plugin::filename`

Referenced by `Print()`.

10.143.3.3 `ParamT<std::string>` `sdf::Plugin::name`

Referenced by `Print()`.

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

- `sdf/interface/Plugin.hh`

10.144 gazebo::PluginT< T > Class Template Reference

A class which all plugins must inherit from.

```
#include <Plugin.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.

10.144.1 Detailed Description

```
template<class T>class gazebo::PluginT< T >
```

A class which all plugins must inherit from.

10.144.2 Member Typedef Documentation

10.144.2.1 `template<class T> typedef T* gazebo::PluginT< T >::TPtr`

plugin pointer type definition

10.144.3 Member Function Documentation

10.144.3.1 `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

10.144.3.2 `template<class T> std::string gazebo::PluginT< T >::GetFilename () const [inline]`

Get the name of the handler.

10.144.3.3 `template<class T> std::string gazebo::PluginT< T >::GetHandle () const [inline]`

Get the short name of the handler.

10.144.3.4 `template<class T> PluginType gazebo::PluginT< T >::GetType () const [inline]`

Returns the type of the plugin.

10.144.4 Member Data Documentation

10.144.4.1 `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()`.

10.144.4.2 `template<class T> std::string gazebo::PluginT< T >::handle` [protected]

Short name.

Referenced by `gazebo::PluginT< ModelPlugin >::Create()`, and `gazebo::PluginT< ModelPlugin >::GetHandle()`.

10.144.4.3 `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`

10.145 gazebo::math::Pose Class Reference

Encapsulates a position and rotation in three space.

```
#include <Pose.hh>
```

Public Member Functions

- **Pose** ()
Default constructors.
- **Pose** (const **Vector3** &_pos, const **Quaternion** &_rot)
Constructor.
- **Pose** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)
Constructor.
- **Pose** (const **Pose** &_pose)
Copy constructor.
- virtual **~Pose** ()
Destructor.
- **Pose CoordPoseSolve** (const **Pose** &_b) const
Find the inverse of a pose; i.e., if $b = this + a$, given b and $this$, find a .
- **Vector3 CoordPositionAdd** (const **Vector3** &_pos) const
Add one point to a vector: result = this + pos.
- **Vector3 CoordPositionAdd** (const **Pose** &_pose) const
Add one point to another: result = this + pose.
- **Vector3 CoordPositionSub** (const **Pose** &_pose) const
Subtract one position from another: result = this - pose.

- **Quaternion CoordRotationAdd** (const **Quaternion** &_rot) const
Add one rotation to another: result = this->rot + rot.
- **Quaternion CoordRotationSub** (const **Quaternion** &_rot) const
Subtract one rotation from another: result = this->rot - rot.
- void **Correct** ()
Fix any nan values.
- **Pose GetInverse** () const
Get the inverse of this pose.
- bool **IsFinite** () const
See if a pose is finite (e.g., not nan)
- bool **operator!=** (const **Pose** &_pose) const
Inequality operator.
- **Pose operator*** (const **Pose** &_pose)
Multiplication operator.
- **Pose operator+** (const **Pose** &_pose) const
Addition operator.
- const **Pose** & **operator+=** (const **Pose** &_pose)
Add-Equals operator.
- **Pose operator-** (const **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. 902) and a **Quaternion** (p. 697).*
- 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.

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.

10.145.1 Detailed Description

Encapsulates a position and rotation in three space.

10.145.2 Constructor & Destructor Documentation

10.145.2.1 gazebo::math::Pose::Pose ()

Default constructors.

Referenced by operator-().

10.145.2.2 gazebo::math::Pose::Pose (const Vector3 & _pos, const Quaternion & _rot)

Constructor.

Parameters

<i>pos</i>	A position
<i>rot</i>	A rotation

10.145.2.3 gazebo::math::Pose::Pose (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

Constructor.

10.145.2.4 gazebo::math::Pose::Pose (const Pose & _pose)

Copy constructor.

Parameters

<i>pose</i>	Pose (p. 677) to copy
-------------	------------------------------

10.145.2.5 virtual gazebo::math::Pose::~~Pose () [virtual]

Destructor.

10.145.3 Member Function Documentation

10.145.3.1 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

<i>in</i>	<i>_b</i>	the other pose
-----------	-----------	----------------

10.145.3.2 Vector3 gazebo::math::Pose::CoordPositionAdd (const Vector3 & *_pos*) const

Add one point to a vector: result = this + pos.

Parameters

<i>in</i>	<i>_pos</i>	Position to add to this pose
-----------	-------------	------------------------------

Returns

the resulting position

10.145.3.3 Vector3 gazebo::math::Pose::CoordPositionAdd (const Pose & *_pose*) const

Add one point to another: result = this + pose.

Parameters

<i>in</i>	<i>_pose</i>	The Pose (p. 677) to add
-----------	--------------	---------------------------------

Returns

The resulting position

10.145.3.4 Vector3 gazebo::math::Pose::CoordPositionSub (const Pose & *_pose*) const [inline]

Subtract one position from another: result = this - pose.

Parameters

<i>in</i>	<i>_pose</i>	Pose (p. 677) 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-().

10.145.3.5 Quaternion gazebo::math::Pose::CoordRotationAdd (const Quaternion & *_rot*) const

Add one rotation to another: result = this->rot + rot.

Parameters

<i>in</i>	<i>_rot</i>	Rotation to add
-----------	-------------	-----------------

Returns

The resulting rotation

10.145.3.6 Quaternion gazebo::math::Pose::CoordRotationSub (const Quaternion & *_rot*) const [inline]

Subtract one rotation from another: result = this->rot - rot.

Parameters

<i>in</i>	<i>_rot</i>	The rotation to subtract
-----------	-------------	--------------------------

Returns

The resulting rotation

References gazebo::math::Quaternion::GetInverse(), gazebo::math::Quaternion::Normalize(), and rot.

Referenced by operator-().

10.145.3.7 void gazebo::math::Pose::Correct () [inline]

Fix any nan values.

References gazebo::math::Vector3::Correct(), gazebo::math::Quaternion::Correct(), pos, and rot.

10.145.3.8 Pose gazebo::math::Pose::GetInverse () const

Get the inverse of this pose.

Returns

the inverse pose

10.145.3.9 bool gazebo::math::Pose::IsFinite () const

See if a pose is finite (e.g., not nan)

10.145.3.10 bool gazebo::math::Pose::operator!=(const Pose & *_pose*) const

Inequality operator.

Parameters

<i>in</i>	<i>_pose</i>	Pose (p. 677) for comparison
-----------	--------------	-------------------------------------

Returns

True if not equal

10.145.3.11 Pose gazebo::math::Pose::operator* (const Pose & *_pose*)

Multiplication operator.

Parameters

<i>in</i>	<i>_pose</i>	the other pose
-----------	--------------	----------------

Returns

itself

10.145.3.12 Pose gazebo::math::Pose::operator+ (const Pose & *_pose*) const

Addition operator.

Parameters

<i>in</i>	<i>pose</i>	Pose (p. 677) to add to this pose
-----------	-------------	--

Returns

The resulting pose

10.145.3.13 const Pose& gazebo::math::Pose::operator+= (const Pose & *_pose*)

Add-Equals operator.

Parameters

<i>in</i>	<i>pose</i>	Pose (p. 677) to add to this pose
-----------	-------------	--

Returns

The resulting pose

10.145.3.14 Pose gazebo::math::Pose::operator- (const Pose & *_pose*) const [inline]

Subtraction operator.

Parameters

<i>in</i>	<i>pose</i>	Pose (p. 677) to subtract from this one
-----------	-------------	--

Returns

The resulting pose

References CoordPositionSub(), CoordRotationSub(), Pose(), and rot.

10.145.3.15 `const Pose& gazebo::math::Pose::operator-= (const Pose & _pose)`

Subtraction operator.

Parameters

<code>in</code>	<code><i>_pose</i></code>	Pose (p. 677) to subtract from this one
-----------------	---------------------------	--

Returns

The resulting pose

10.145.3.16 `bool gazebo::math::Pose::operator==(const Pose & _pose) const`

Equality operator.

Parameters

<code>in</code>	<code><i>_pose</i></code>	Pose (p. 677) for comparison
-----------------	---------------------------	-------------------------------------

Returns

True if equal

10.145.3.17 `void gazebo::math::Pose::Reset ()`

Reset the pose.

10.145.3.18 `Pose gazebo::math::Pose::RotatePositionAboutOrigin (const Quaternion & _rot) const`

Rotate vector part of a pose about the origin.

Parameters

<code>in</code>	<code><i>_rot</i></code>	rotation
-----------------	--------------------------	----------

Returns

the rotated pose

10.145.3.19 `void gazebo::math::Pose::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

<code>in</code>	<code><i>_precision</i></code>	
-----------------	--------------------------------	--

10.145.3.20 `void gazebo::math::Pose::Set (const Vector3 & _pos, const Quaternion & _rot)`

Set the pose from a **Vector3** (p. 902) and a **Quaternion** (p. 697).

Parameters

<code>in</code>	<code>_pos</code>	The position.
<code>in</code>	<code>_rot</code>	The rotation.

10.145.3.21 `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

<code>in</code>	<code>_x</code>	x position in meters.
<code>in</code>	<code>_y</code>	y position in meters.
<code>in</code>	<code>_z</code>	z position in meters.
<code>in</code>	<code>_roll</code>	Roll (rotation about X-axis) in radians.
<code>in</code>	<code>_pitch</code>	Pitch (rotation about y-axis) in radians.
<code>in</code>	<code>_yaw</code>	Pitch (rotation about z-axis) in radians.

10.145.4 Friends And Related Function Documentation

10.145.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Pose & _pose)` [`friend`]

Stream insertion operator.

Parameters

<code>out</code>	output stream
<code>pose</code>	pose to output

Returns

the stream

10.145.4.2 `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

10.145.5 Member Data Documentation

10.145.5.1 Vector3 gazebo::math::Pose::pos

The position.

Referenced by CoordPositionSub(), and Correct().

10.145.5.2 Quaternion gazebo::math::Pose::rot

The rotation.

Referenced by CoordPositionSub(), CoordRotationSub(), Correct(), and operator-().

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

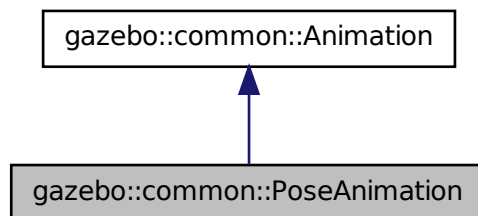
- **Pose.hh**

10.146 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

10.146.1 Detailed Description

A pose animation.

10.146.2 Constructor & Destructor Documentation

10.146.2.1 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

10.146.2.2 virtual gazebo::common::PoseAnimation::~~PoseAnimation () [virtual]

Destructor.

10.146.3 Member Function Documentation

10.146.3.1 void gazebo::common::PoseAnimation::BuildInterpolationSplines () const [protected]

Update the pose splines.

10.146.3.2 **PoseKeyFrame*** gazebo::common::PoseAnimation::CreateKeyFrame (double _time)

Create a pose keyframe at the given time.

Parameters

in	<code>_time</code>	Time (p. 840) at which to create the keyframe
----	--------------------	--

Returns

Pointer to the new keyframe

10.146.3.3 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (PoseKeyFrame & _kf) const

Get a keyframe using the animation's current time.

Parameters

out	_kf	PoseKeyFrame (p. 687) reference to hold the interpolated result
-----	-----	--

10.146.3.4 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (double _time, PoseKeyFrame & _kf) const
[protected]

Get a keyframe using a passed in time.

Parameters

in	_time	Time (p. 840) in seconds
out	_kf	PoseKeyFrame (p. 687) reference to hold the interpolated result

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

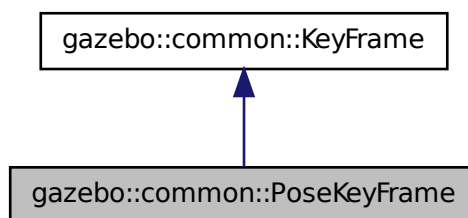
- **Animation.hh**

10.147 gazebo::common::PoseKeyFrame Class Reference

A keyframe for a **PoseAnimation** (p. 685).

```
#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

10.147.1 Detailed Description

A keyframe for a **PoseAnimation** (p. 685).

10.147.2 Constructor & Destructor Documentation

10.147.2.1 gazebo::common::PoseKeyFrame::PoseKeyFrame (double *_time*)

Constructor.

Parameters

<i>in</i>	<i>_time</i>	of the keyframe
-----------	--------------	-----------------

10.147.2.2 virtual gazebo::common::PoseKeyFrame::~~PoseKeyFrame () [virtual]

Destructor.

10.147.3 Member Function Documentation

10.147.3.1 const **math::Quaternion**& gazebo::common::PoseKeyFrame::GetRotation () const

Get the rotation of the keyframe.

Returns

The rotation amount

10.147.3.2 const **math::Vector3**& gazebo::common::PoseKeyFrame::GetTranslation () const

Get the translation of the keyframe.

Returns

The translation amount

10.147.3.3 `void gazebo::common::PoseKeyFrame::SetRotation (const math::Quaternion & _rot)`

Set the rotation for the keyframe.

Parameters

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

10.147.3.4 `void gazebo::common::PoseKeyFrame::SetTranslation (const math::Vector3 & _trans)`

Set the translation for the keyframe.

Parameters

<code>in</code>	<code>_trans</code>	Translation amount
-----------------	---------------------	--------------------

10.147.4 Member Data Documentation

10.147.4.1 `math::Quaternion gazebo::common::PoseKeyFrame::rotate` `[protected]`

the rotation quaternion

10.147.4.2 `math::Vector3 gazebo::common::PoseKeyFrame::translate` `[protected]`

the translation vector

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

- **KeyFrame.hh**

10.148 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.

10.148.1 Detailed Description

Projects a material onto surface, light a light projector.

10.148.2 Constructor & Destructor Documentation

10.148.2.1 gazebo::rendering::Projector::Projector (**VisualPtr** _parent)

Constructor.

Parameters

in	_parent	Name of the parent visual.
----	---------	----------------------------

10.148.2.2 virtual gazebo::rendering::Projector::~~Projector () [virtual]

Destructor.

10.148.3 Member Function Documentation

10.148.3.1 **VisualPtr** gazebo::rendering::Projector::GetParent ()

Get the parent visual.

Returns

Pointer of the parent visual.

10.148.3.2 void gazebo::rendering::Projector::Load (**sdf::ElementPtr** _sdf)

Load from an sdf pointer.

Parameters

in	<code>_sdf</code>	Pointer to the SDF element.
----	-------------------	-----------------------------

10.148.3.3 `void gazebo::rendering::Projector::Load (const msgs::Projector & _msg)`

Load from a message.

Parameters

in	<code>_msg</code>	Load from a message.
----	-------------------	----------------------

10.148.3.4 `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.

10.148.3.5 `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.
----	-----------------------	-------------------------------

10.148.3.6 `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.
----	---------------------------	---------------------------------

10.148.3.7 `void gazebo::rendering::Projector::Toggle ()`

Toggle the activation of the projector.

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

- **Projector.hh**

10.149 gazebo::transport::Publication Class Reference

A publication for a topic.

```
#include <Publication.hh>
```

Public Member Functions

- **Publication** (const std::string &topic, const std::string &msgType)
Constructor.
- virtual **~Publication** ()
Destructor.
- void **AddPublisher** (**PublisherPtr** _pub)
- void **AddSubscription** (const **CallbackHelperPtr** &callback)
- void **AddSubscription** (const **NodePtr** &_node)
- void **AddTransport** (const **PublicationTransportPtr** &publink)
- unsigned int **GetCallbackCount** () const
- bool **GetLocallyAdvertised** () const
Return true if the topic has been advertised from this process.
- std::string **GetMsgType** () const
Get the type of message.
- unsigned int **GetNodeCount** () const
- unsigned int **GetRemoteSubscriptionCount** ()
- unsigned int **GetTransportCount** () const
- bool **HasTransport** (const std::string &_host, unsigned int _port)
- void **LocalPublish** (const std::string &data)
Publish data.
- void **Publish** (const google::protobuf::Message &msg, const boost::function< void()> &cb=NULL)
- void **RemoveSubscription** (const **NodePtr** &_node)
Remove a subscription.
- void **RemoveSubscription** (const std::string &host, unsigned int port)
Remove a subscription.
- void **RemoveTransport** (const std::string &host, unsigned int port)
- void **SetLocallyAdvertised** (bool _value)
Set whether this topic has been advertised from this process.

10.149.1 Detailed Description

A publication for a topic.

This facilitates transport of messages

10.149.2 Constructor & Destructor Documentation

10.149.2.1 gazebo::transport::Publication::Publication (const std::string & topic, const std::string & msgType)

Constructor.

10.149.2.2 virtual gazebo::transport::Publication::~~Publication () [virtual]

Destructor.

10.149.3 Member Function Documentation

10.149.3.1 void gazebo::transport::Publication::AddPublisher (PublisherPtr *_pub*)

Referenced by gazebo::transport::TopicManager::Advertise().

10.149.3.2 void gazebo::transport::Publication::AddSubscription (const CallbackHelperPtr & *callback*)

Referenced by gazebo::transport::TopicManager::Advertise().

10.149.3.3 void gazebo::transport::Publication::AddSubscription (const NodePtr & *_node*)

10.149.3.4 void gazebo::transport::Publication::AddTransport (const PublicationTransportPtr & *publink*)

10.149.3.5 unsigned int gazebo::transport::Publication::GetCallbackCount () const

10.149.3.6 bool gazebo::transport::Publication::GetLocallyAdvertised () const

Return true if the topic has been advertised from this process.

Referenced by gazebo::transport::TopicManager::Advertise().

10.149.3.7 std::string gazebo::transport::Publication::GetMsgType () const

Get the type of message.

10.149.3.8 unsigned int gazebo::transport::Publication::GetNodeCount () const

10.149.3.9 unsigned int gazebo::transport::Publication::GetRemoteSubscriptionCount ()

10.149.3.10 unsigned int gazebo::transport::Publication::GetTransportCount () const

10.149.3.11 bool gazebo::transport::Publication::HasTransport (const std::string & *_host*, unsigned int *_port*)

10.149.3.12 void gazebo::transport::Publication::LocalPublish (const std::string & *data*)

Publish data.

10.149.3.13 void gazebo::transport::Publication::Publish (const google::protobuf::Message & *msg*, const boost::function< void()> & *cb* = NULL)

10.149.3.14 void gazebo::transport::Publication::RemoveSubscription (const NodePtr & *_node*)

Remove a subscription.

10.149.3.15 void gazebo::transport::Publication::RemoveSubscription (const std::string & *host*, unsigned int *port*)

Remove a subscription.

10.149.3.16 void gazebo::transport::Publication::RemoveTransport (const std::string & *host*, unsigned int *port*)

10.149.3.17 void gazebo::transport::Publication::SetLocallyAdvertised (bool *_value*)

Set whether this topic has been advertised from this process.

Referenced by gazebo::transport::TopicManager::Advertise().

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

- **Publication.hh**

10.150 gazebo::transport::PublicationTransport Class Reference

Reads data from a remote advertiser, and passes the data along to local subscribers.

```
#include <PublicationTransport.hh>
```

Public Member Functions

- **PublicationTransport** (const std::string &topic, const std::string &msgType)
- virtual ~**PublicationTransport** ()
- void **AddCallback** (const boost::function< void(const std::string &)> &cb)
- void **Fini** ()
- const **ConnectionPtr GetConnection** () const
- std::string **GetMsgType** () const
- std::string **GetTopic** () const
- void **Init** (const **ConnectionPtr** &conn)

10.150.1 Detailed Description

Reads data from a remote advertiser, and passes the data along to local subscribers.

10.150.2 Constructor & Destructor Documentation

10.150.2.1 gazebo::transport::PublicationTransport::PublicationTransport (const std::string & *topic*, const std::string & *msgType*)

10.150.2.2 virtual gazebo::transport::PublicationTransport::~PublicationTransport () [virtual]

10.150.3 Member Function Documentation

10.150.3.1 void gazebo::transport::PublicationTransport::AddCallback (const boost::function< void(const std::string &)> & *cb*)

10.150.3.2 void gazebo::transport::PublicationTransport::Fini ()

10.150.3.3 `const ConnectionPtr gazebo::transport::PublicationTransport::GetConnection () const`

10.150.3.4 `std::string gazebo::transport::PublicationTransport::GetMsgType () const`

10.150.3.5 `std::string gazebo::transport::PublicationTransport::GetTopic () const`

10.150.3.6 `void gazebo::transport::PublicationTransport::Init (const ConnectionPtr & conn)`

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

- **PublicationTransport.hh**

10.151 gazebo::transport::Publisher Class Reference

A publisher of messages on a topic.

```
#include <Publisher.hh>
```

Public Member Functions

- **Publisher** (const std::string &topic, const std::string &msg_type, unsigned int _limit, bool _latch)
Use this constructor.
- virtual **~Publisher** ()
Destructor.
- bool **GetLatching** () const
- std::string **GetMsgType** () const
Get the message type.
- unsigned int **GetOutgoingCount** () const
- std::string **GetPrevMsg** () const
- std::string **GetTopic** () const
Get the topic name.
- bool **HasConnections** () const
- void **Publish** (const google::protobuf::Message &_message, bool _block=false)
Publish a message on the topic.
- template<typename M >
void **Publish** (M _message, bool _block=false)
- void **SendMessage** ()
Send latest message over the wire. For internal use only.
- void **SetPublication** (PublicationPtr &_publication, int _i)
- void **WaitForConnection** () const
Block until a connection has been established with this publisher.

10.151.1 Detailed Description

A publisher of messages on a topic.

10.151.2 Constructor & Destructor Documentation

10.151.2.1 `gazebo::transport::Publisher::Publisher (const std::string & topic, const std::string & msg_type, unsigned int limit, bool latch)`

Use this constructor.

Parameters

<i>topic</i>	Name of topic
<i>msg_type</i>	Type of the message which is to be published

10.151.2.2 `virtual gazebo::transport::Publisher::~~Publisher () [virtual]`

Destructor.

10.151.3 Member Function Documentation

10.151.3.1 `bool gazebo::transport::Publisher::GetLatching () const`

10.151.3.2 `std::string gazebo::transport::Publisher::GetMsgType () const`

Get the message type.

10.151.3.3 `unsigned int gazebo::transport::Publisher::GetOutgoingCount () const`

10.151.3.4 `std::string gazebo::transport::Publisher::GetPrevMsg () const`

10.151.3.5 `std::string gazebo::transport::Publisher::GetTopic () const`

Get the topic name.

10.151.3.6 `bool gazebo::transport::Publisher::HasConnections () const`

10.151.3.7 `void gazebo::transport::Publisher::Publish (const google::protobuf::Message & _message, bool _block = false) [inline]`

Publish a message on the topic.

10.151.3.8 `template<typename M > void gazebo::transport::Publisher::Publish (M _message, bool _block = false) [inline]`

10.151.3.9 `void gazebo::transport::Publisher::SendMessage ()`

Send latest message over the wire. For internal use only.

10.151.3.10 `void gazebo::transport::Publisher::SetPublication (PublicationPtr & _publication, int _i)`

10.151.3.11 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**

10.152 gazebo::math::Quaternion Class Reference

A quaternion class.

```
#include <Quaternion.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.
- **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. 902) multiplication operator.*
- **Quaternion operator*=** (const **Quaternion** &qt)
Multiplication operator.
- **Quaternion operator+** (const **Quaternion** &_qt) const
Addition operator.
- **Quaternion operator+=** (const **Quaternion** &_qt)
Addition operator.
- **Quaternion operator-** (const **Quaternion** &_qt) const
Substraction operator.
- **Quaternion operator-** () const
Unary minus operator.
- **Quaternion operator-=** (const **Quaternion** &_qt)
Substraction operator.
- **Quaternion & operator=** (const **Quaternion** &qt)
Equal operator.
- bool **operator==** (const **Quaternion** &_qt) const
Equal to operator.
- **Vector3 RotateVector** (const **Vector3** &_vec) const
Rotate a vector using the quaternion.
- **Vector3 RotateVectorReverse** (**Vector3** _vec) const
Do the reverse rotation of a vector by this quaternion.
- void **Round** (int _precision)

Round all values to `_precision` decimal places.

- void **Scale** (double `_scale`)
Scale a Quaternionion.
- void **Set** (double `_u`, double `_x`, double `_y`, double `_z`)
Set this quaternion from 4 floating numbers.
- void **SetFromAxis** (double `_x`, double `_y`, double `_z`, double `_a`)
Set the quaternion from an axis and angle.
- void **SetFromAxis** (const **Vector3** &`_axis`, double `_a`)
Set the quaternion from an axis and angle.
- void **SetFromEuler** (const **Vector3** &`_vec`)
Set the quaternion from Euler angles.
- void **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.

10.152.1 Detailed Description

A quaternion class.

10.152.2 Constructor & Destructor Documentation

10.152.2.1 gazebo::math::Quaternion::Quaternion ()

Default Constructor.

Referenced by operator*().

10.152.2.2 gazebo::math::Quaternion::Quaternion (const double & *_w*, const double & *_x*, const double & *_y*, const double & *_z*)

Constructor.

Parameters

in	<i>_w</i>	W param
in	<i>_x</i>	X param
in	<i>_y</i>	Y param
in	<i>_z</i>	Z param

10.152.2.3 gazebo::math::Quaternion::Quaternion (const double & *_roll*, const double & *_pitch*, const double & *_yaw*)

Constructor from Euler angles.

Parameters

in	<i>_roll</i>	roll
in	<i>_pitch</i>	pitch
in	<i>_yaw</i>	yaw

10.152.2.4 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

10.152.2.5 gazebo::math::Quaternion::Quaternion (const Vector3 & *_rpy*)

Constructor.

Parameters

in	<i>_rpy</i>	euler angles
----	-------------	--------------

10.152.2.6 gazebo::math::Quaternion::Quaternion (const Quaternion & *qt*)

Copy constructor.

Parameters

<i>qt</i>	Quaternion (p. 697) to copy
-----------	-----------------------------

10.152.2.7 gazebo::math::Quaternion::~~Quaternion ()

Destructor.

10.152.3 Member Function Documentation

10.152.3.1 void gazebo::math::Quaternion::Correct () [inline]

Correct any nan.

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.152.3.2 double gazebo::math::Quaternion::Dot (const Quaternion & *q*) const

Dot product.

Parameters

<i>in</i>	<i>the</i>	other quaternion
-----------	------------	------------------

Returns

the product

10.152.3.3 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (const Vector3 & *_vec*) [static]

Convert euler angles to quatern.

Parameters

<i>in</i>		
-----------	--	--

10.152.3.4 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (double *_x*, double *_y*, double *_z*) [static]

Convert euler angles to quatern.

Parameters

<i>_x</i>	rotation along x
<i>_y</i>	rotation along y
<i>_z</i>	rotation along z

10.152.3.5 `void gazebo::math::Quaternion::GetAsAxis (Vector3 & _axis, double & _angle) const`

Return rotation as axis and angle.

Parameters

<code>in</code>	<code><i>_axis</i></code>	rotation axis
<code>in</code>	<code><i>_angle</i></code>	ccw angle in radians

10.152.3.6 `Vector3 gazebo::math::Quaternion::GetAsEuler () const`

Return the rotation in Euler angles.

Returns

This quaternion as an Euler vector

10.152.3.7 `Matrix3 gazebo::math::Quaternion::GetAsMatrix3 () const`

Get the quaternion as a 3x3 matrix.

10.152.3.8 `Matrix4 gazebo::math::Quaternion::GetAsMatrix4 () const`

Get the quaternion as a 4x4 matrix.

Returns

a 4x4 matrix

10.152.3.9 `Quaternion gazebo::math::Quaternion::GetExp () const`

Return the exponent.

Returns

the exp

10.152.3.10 `Quaternion gazebo::math::Quaternion::GetInverse () const` `[inline]`

Get the inverse of this quaternion.

Returns

Inverse quarenion

References `gazebo::math::equal()`, `w`, `x`, `y`, and `z`.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `gazebo::math::Pose::CoordRotationSub()`, and `RotateVector()`.

10.152.3.11 `Quaternion gazebo::math::Quaternion::GetLog () const`

Return the logarithm.

Returns

the log

10.152.3.12 `double gazebo::math::Quaternion::GetPitch ()`

Get the Euler pitch angle in radians.

Returns

the pitch

10.152.3.13 `double gazebo::math::Quaternion::GetRoll ()`

Get the Euler roll angle in radians.

Returns

the roll

10.152.3.14 `Vector3 gazebo::math::Quaternion::GetXAxis () const`

Return the X axis.

Returns

the vector

10.152.3.15 `double gazebo::math::Quaternion::GetYaw ()`

Get the Euler yaw angle in radians.

Returns

the yaw

10.152.3.16 `Vector3 gazebo::math::Quaternion::GetYAxis () const`

Return the Y axis.

Returns

the vector

10.152.3.17 **Vector3** gazebo::math::Quaternion::GetZAxis () const

Return the Z axis.

Returns

the vector

10.152.3.18 **void** gazebo::math::Quaternion::Invert ()

Invert the quaternion.

10.152.3.19 **bool** gazebo::math::Quaternion::IsFinite () const

See if a quatern is finite (e.g., not nan)

Returns

True if quatern is finite

10.152.3.20 **void** gazebo::math::Quaternion::Normalize ()

Normalize the quaternion.

Referenced by gazebo::math::Pose::CoordRotationSub().

10.152.3.21 **bool** gazebo::math::Quaternion::operator!=(const Quaternion & _qt) const

Not equal to operator.

Parameters

<code>in</code>	<code>_qt</code>	Quaternion (p. 697) for comparison
-----------------	------------------	---

Returns

True if not equal

10.152.3.22 **Quaternion** gazebo::math::Quaternion::operator*(const Quaternion & _q) const `[inline]`

Multiplication operator.

Parameters

<code>in</code>	<code>_qt</code>	Quaternion (p. 697) for multiplication
-----------------	------------------	---

Returns

This quaternion multiplied by the parameter

References Quaternion(), w, x, y, and z.

10.152.3.23 Quaternion gazebo::math::Quaternion::operator* (const double & _f) const

Multiplication operator.

Parameters

<code>in</code>	<code>_f</code>	factor
-----------------	-----------------	--------

Returns

quaternion multiplied by `_f`

10.152.3.24 Vector3 gazebo::math::Quaternion::operator* (const Vector3 & v) const

Vector3 (p. 902) multiplication operator.

10.152.3.25 Quaternion gazebo::math::Quaternion::operator*= (const Quaternion & qt)

Multiplication operator.

Parameters

<code>in</code>	<code>_qt</code>	Quaternion (p. 697) for multiplication
-----------------	------------------	---

Returns

This quatern multiplied by the parameter

10.152.3.26 Quaternion gazebo::math::Quaternion::operator+ (const Quaternion & _qt) const

Addition operator.

Parameters

<code>_qt[in]</code>	quaternion for addition
----------------------	-------------------------

Returns

this quaternion + `_qt`

10.152.3.27 Quaternion gazebo::math::Quaternion::operator+= (const Quaternion & _qt)

Addition operator.

Parameters

<i>in</i>	<i>_qt</i>	quaternion for addition
-----------	------------	-------------------------

Returns

this quaternion + qt

10.152.3.28 **Quaternion** gazebo::math::Quaternion::operator- (const Quaternion & *_qt*) const

Substraction operator.

Parameters

<i>in</i>	<i>_qt</i>	quaternion to subtract
-----------	------------	------------------------

Returns

this quaternion - *_qt*

10.152.3.29 **Quaternion** gazebo::math::Quaternion::operator- () const

Unary minus operator.

Returns

negates each component of the quaternion

10.152.3.30 **Quaternion** gazebo::math::Quaternion::operator-= (const Quaternion & *_qt*)

Substraction operator.

Parameters

<i>in</i>	<i>_qt</i>	Quaternion (p. 697) for subtraction
-----------	------------	--

Returns

This quatern - qt

10.152.3.31 **Quaternion&** gazebo::math::Quaternion::operator= (const Quaternion & *qt*)

Equal operator.

Parameters

<i>in</i>	<i>qt</i>	Quaternion (p. 697) to copy
-----------	-----------	------------------------------------

10.152.3.32 `bool gazebo::math::Quaternion::operator==(const Quaternion & _qt) const`

Equal to operator.

Parameters

<code>in</code>	<code>_qt</code>	Quaternion (p. 697) for comparison
-----------------	------------------	---

Returns

True if equal

10.152.3.33 `Vector3 gazebo::math::Quaternion::RotateVector (const Vector3 & _vec) const [inline]`

Rotate a vector using the quaternion.

Parameters

<code>in</code>	<code>_vec</code>	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.

10.152.3.34 `Vector3 gazebo::math::Quaternion::RotateVectorReverse (Vector3 _vec) const`

Do the reverse rotation of a vector by this quaternion.

Parameters

<code>in</code>	<code>_vec</code>	the vector
-----------------	-------------------	------------

Returns

the

10.152.3.35 `void gazebo::math::Quaternion::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

<code>in</code>	<code>_precision</code>	the precision
-----------------	-------------------------	---------------

10.152.3.36 `void gazebo::math::Quaternion::Scale (double _scale)`

Scale a Quaternionion.

Parameters

in	<code>_scale</code>	Amount to scale this rotation
----	---------------------	-------------------------------

10.152.3.37 `void gazebo::math::Quaternion::Set (double _u, double _x, double _y, double _z)`

Set this quaternion from 4 floating numbers.

Parameters

in	<code>_u</code>	u
in	<code>_x</code>	x
in	<code>_y</code>	y
in	<code>_z</code>	z

10.152.3.38 `void gazebo::math::Quaternion::SetFromAxis (double _x, double _y, double _z, double _a)`

Set the quaternion from an axis and angle.

Parameters

in	<code>_x</code>	X axis
in	<code>_y</code>	Y axis
in	<code>_z</code>	Z axis
in	<code>_a</code>	Angle (p. 127) in radians

10.152.3.39 `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. 127) in radians

10.152.3.40 `void gazebo::math::Quaternion::SetFromEuler (const Vector3 & _vec)`

Set the quaternion from Euler angles.

Parameters

in	<code>vec</code>	Euler angle
----	------------------	-------------

10.152.3.41 `void gazebo::math::Quaternion::SetToIdentity ()`

Set the quatern to the identity.

10.152.3.42 `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><i>_ft</i></code>	the interpolation parameter
in	<code><i>_rkP</i></code>	the beginning quaternion
in	<code><i>_rkQ</i></code>	the end quaternion
in	<code><i>_shortestPath</i></code>	when true, the rotation may be inverted to get to minimize rotation

10.152.3.43 `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><i>_ft</i></code>	the interpolation parameter
in	<code><i>_rkP</i></code>	the beginning quaternion
in	<code><i>_rkA</i></code>	first intermediate quaternion
in	<code><i>_rkB</i></code>	second intermediate quaternion
in	<code><i>_rkQ</i></code>	the end quaternion
in	<code><i>_shortestPath</i></code>	when true, the rotation may be inverted to get to minimize rotation

10.152.4 Friends And Related Function Documentation

10.152.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Quaternion & _q) [friend]`

Stream insertion operator.

Parameters

in	<code><i>_out</i></code>	output stream
in	<code><i>_q</i></code>	quaternion to output

Returns

the stream

10.152.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Quaternion & _q) [friend]`

Stream extraction operator.

Parameters

in	<code><i>_in</i></code>	input stream
in	<code><i>_q</i></code>	Quaternion (p. 697) to read values into

Returns

The istream

10.152.5 Member Data Documentation**10.152.5.1 double gazebo::math::Quaternion::w**

Attributes of the quaternion.

Referenced by `Correct()`, `GetInverse()`, and `operator*()`.

10.152.5.2 double gazebo::math::Quaternion::x

Attributes of the quaternion.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `GetInverse()`, `operator*()`, and `RotateVector()`.

10.152.5.3 double gazebo::math::Quaternion::y

Attributes of the quaternion.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `GetInverse()`, `operator*()`, and `RotateVector()`.

10.152.5.4 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**

10.153 gazebo::math::Rand Class Reference

Random number generator class.

```
#include <Rand.hh>
```

Static Public Member Functions

- static double **GetDbfNormal** (double _mean=0, double _sigma=1)
Get a double from a normal distribution.
- static double **GetDbfUniform** (double _min=0, double _max=1)
Get a double from a uniform distribution.
- static int **GetIntNormal** (int _mean, int _sigma)
Get a double from a normal distribution.
- static int **GetIntUniform** (int _min, int _max)
Get a integer from a uniform distribution.

10.153.1 Detailed Description

Random number generator class.

10.153.2 Member Function Documentation

10.153.2.1 `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

10.153.2.2 `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

10.153.2.3 `static int gazebo::math::Rand::GetIntNormal (int _mean, int _sigma) [static]`

Get a double from a normal distribution.

Parameters

in	<i>_mean</i>	Mean value for the distribution
in	<i>_sigma</i>	Sigma value for the distribution

10.153.2.4 `static int gazebo::math::Rand::GetIntUniform (int _min, int _max) [static]`

Get a integer from a uniform distribution.

Parameters

in	<i>_min</i>	Minimum bound for the random number
in	<i>_max</i>	Maximum bound for the random number

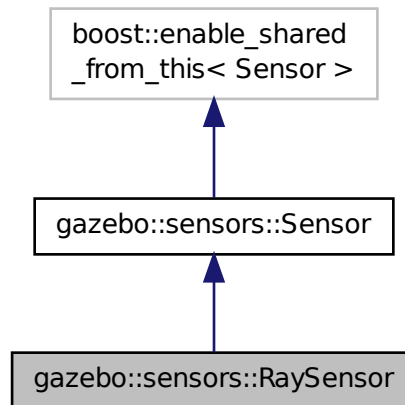
The documentation for this class was generated from the following file:

- **Rand.hh**

10.154 gazebo::sensors::RaySensor Class Reference

```
#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 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. 549).*
- 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
Gets the topic name of the sensor.
- **math::Angle GetVerticalAngleMax** () const
Get the vertical scan line top angle.
- **math::Angle GetVerticalAngleMin** () const
Get the vertical scan bottom angle.
- int **GetVerticalRangeCount** () const
Get the vertical scan line count.
- int **GetVerticalRayCount** () const
Get the vertical scan line count.
- virtual void **Init** ()
Initialize the ray.
- virtual 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 ray.
- virtual void **UpdateImpl** (bool _force)
Update the sensor information.

Additional Inherited Members

10.154.1 Constructor & Destructor Documentation

10.154.1.1 gazebo::sensors::RaySensor::RaySensor ()

Constructor.

10.154.1.2 virtual gazebo::sensors::RaySensor::~~RaySensor () [virtual]

Destructor.

10.154.2 Member Function Documentation

10.154.2.1 virtual void gazebo::sensors::RaySensor::Fini () [protected], [virtual]

Finalize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

10.154.2.2 `math::Angle gazebo::sensors::RaySensor::GetAngleMax () const`

Get the maximum angle.

Returns

the maximum angle

10.154.2.3 `math::Angle gazebo::sensors::RaySensor::GetAngleMin () const`

Get the minimum angle.

Returns

The minimum angle

10.154.2.4 `double gazebo::sensors::RaySensor::GetAngleResolution () const`

Get radians between each range.

Returns

Resolution of the angle

10.154.2.5 `int gazebo::sensors::RaySensor::GetFiducial (int _index)`

Get detected fiducial value for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index value of specific ray
-----------------	----------------------------	-----------------------------

Returns

Fiducial value

10.154.2.6 `physics::MultiRayShapePtr gazebo::sensors::RaySensor::GetLaserShape () const` `[inline]`

Returns a pointer to the internal `physics::MultiRayShape` (p. 549).

Returns

Pointer to ray shape

10.154.2.7 `double gazebo::sensors::RaySensor::GetRange (int _index)`

Get detected range for a ray.

Warning: If you are accessing all the ray data in a loop it's possible that the Ray will update in the middle of your access loop. This means some data will come from one scan, and some from another scan. You can solve this problem by using `SetActive(false)` <your accessor loop> `SetActive(true)`.

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of specific ray
-----------------	----------------------------	-----------------------

Returns

Returns `DBL_MAX` for no detection.

10.154.2.8 `int gazebo::sensors::RaySensor::GetRangeCount () const`

Get the range count.

Returns

The number of ranges

10.154.2.9 `double gazebo::sensors::RaySensor::GetRangeMax () const`

Get the maximum range.

Returns

The maximum range

10.154.2.10 `double gazebo::sensors::RaySensor::GetRangeMin () const`

Get the minimum range.

Returns

The minimum range

10.154.2.11 `double gazebo::sensors::RaySensor::GetRangeResolution () const`

Get the range resolution.

Returns

Resolution of the range

10.154.2.12 `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
----------------------	---

10.154.2.13 `int gazebo::sensors::RaySensor::GetRayCount () const`

Get the ray count.

Returns

The number of rays

10.154.2.14 `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

10.154.2.15 `virtual std::string gazebo::sensors::RaySensor::GetTopic () const` [virtual]

Gets the topic name of the sensor.

Returns

Topic name

Reimplemented from `gazebo::sensors::Sensor` (p. 769).

10.154.2.16 `math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMax () const`

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.154.2.17 `math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMin () const`

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.154.2.18 `int gazebo::sensors::RaySensor::GetVerticalRangeCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.154.2.19 `int gazebo::sensors::RaySensor::GetVerticalRayCount () const`

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.154.2.20 `virtual void gazebo::sensors::RaySensor::Init () [virtual]`

Initialize the ray.

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

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

Load the sensor with SDF parameters.

Parameters

in	<code>_sdf</code>	SDF <code>Sensor</code> (p. 765) parameters
in	<code>_worldName</code>	Name of world to load from

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

10.154.2.22 `virtual void gazebo::sensors::RaySensor::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

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

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

10.154.2.23 `virtual void gazebo::sensors::RaySensor::UpdateImpl (bool _force)` [protected],[virtual]

Update the sensor information.

Reimplemented from `gazebo::sensors::Sensor` (p. 771).

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

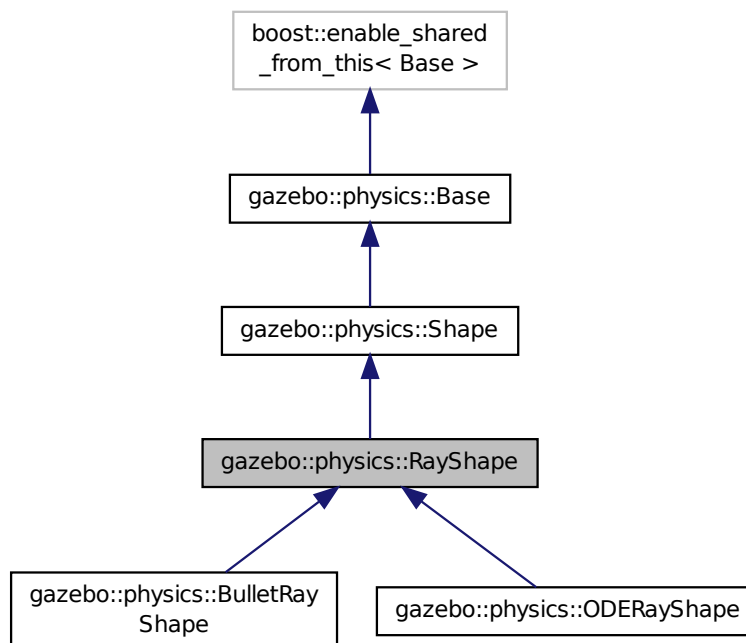
- `RaySensor.hh`

10.155 gazebo::physics::RayShape Class Reference

Base (p. 145) class for Ray collision geometry.

```
#include <RayShape.hh>
```

Inheritance diagram for `gazebo::physics::RayShape`:



Public Member Functions

- **RayShape** (`PhysicsEnginePtr _physicsEngine`)
Constructor for a global ray.
- **RayShape** (`CollisionPtr parent`)

Constructor.

- virtual \sim **RayShape** ()

Destructor.

- void **FillMsg** (msgs::Geometry &)
- 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 &)
- 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**
- double **contactLen**

Contact (p. 296) information; this is filled out during collision detection.

- double **contactRetro**
- math::Vector3 **globalEndPos**
- math::Vector3 **globalStartPos**

Start and end positions of the ray in global cs.

- math::Vector3 **relativeEndPos**
- math::Vector3 **relativeStartPos**

Start and end positions of the ray, relative to the body.

Additional Inherited Members

10.155.1 Detailed Description

Base (p. 145) class for Ray collision geometry.

10.155.2 Constructor & Destructor Documentation

10.155.2.1 gazebo::physics::RayShape::RayShape (PhysicsEnginePtr *_physicsEngine*)

Constructor for a global ray.

10.155.2.2 gazebo::physics::RayShape::RayShape (CollisionPtr *parent*)

Constructor.

Parameters

<i>body</i>	Link (p. 454) the ray is attached to
<i>displayRays</i>	Indicates if the rays should be displayed when rendering images

10.155.2.3 virtual gazebo::physics::RayShape::~~RayShape () [virtual]

Destructor.

10.155.3 Member Function Documentation

10.155.3.1 void gazebo::physics::RayShape::FillMsg (msgs::Geometry &) [inline],[virtual]

Implements **gazebo::physics::Shape** (p. 781).

10.155.3.2 int gazebo::physics::RayShape::GetFiducial () const

Get the fiducial id detected by this ray.

10.155.3.3 virtual void gazebo::physics::RayShape::GetGlobalPoints (math::Vector3 & *posA*, math::Vector3 & *posB*) [virtual]

Get the global starting and ending points.

Parameters

<i>posA</i>	Returns the starting point
<i>posB</i>	Returns the ending point

10.155.3.4 virtual void gazebo::physics::RayShape::GetIntersection (double & *_dist*, std::string & *_entity*) [pure virtual]

Get the nearest intersection.

Implemented in **gazebo::physics::ODERayShape** (p. 621), and **gazebo::physics::BulletRayShape** (p. 212).

10.155.3.5 `double gazebo::physics::RayShape::GetLength () const`

Get the length of the ray.

10.155.3.6 `virtual void gazebo::physics::RayShape::GetRelativePoints (math::Vector3 & posA, math::Vector3 & posB) [virtual]`

Get the relative starting and ending points.

Parameters

<i>posA</i>	Returns the starting point
<i>posB</i>	Returns the ending point

10.155.3.7 `float gazebo::physics::RayShape::GetRetro () const`

Get the retro-reflectivness detected by this ray.

10.155.3.8 `virtual void gazebo::physics::RayShape::Init () [virtual]`

In the ray.

Implements **gazebo::physics::Shape** (p. 781).

10.155.3.9 `virtual void gazebo::physics::RayShape::ProcessMsg (const msgs::Geometry &) [inline],[virtual]`

Implements **gazebo::physics::Shape** (p. 782).

10.155.3.10 `void gazebo::physics::RayShape::SetFiducial (int fid)`

Set the fiducial id detected by this ray.

10.155.3.11 `virtual void gazebo::physics::RayShape::SetLength (double len) [virtual]`

Set the length of the ray.

Parameters

<i>len</i>	Length of the array
------------	---------------------

10.155.3.12 `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

<i>posStart</i>	Start position, relative the body
<i>posEnd</i>	End position, relative to the body

Reimplemented in `gazebo::physics::ODERayShape` (p. 621), and `gazebo::physics::BulletRayShape` (p. 212).

10.155.3.13 `void gazebo::physics::RayShape::SetRetro (float retro)`

Set the retro-reflectivness detected by this ray.

10.155.3.14 `virtual void gazebo::physics::RayShape::Update ()` [pure virtual]

Update the ray collision.

Reimplemented from `gazebo::physics::Base` (p. 157).

Implemented in `gazebo::physics::ODERayShape` (p. 621), and `gazebo::physics::BulletRayShape` (p. 212).

10.155.4 Member Data Documentation

10.155.4.1 `int gazebo::physics::RayShape::contactFiducial` [protected]

10.155.4.2 `double gazebo::physics::RayShape::contactLen` [protected]

Contact (p. 296) information; this is filled out during collision detection.

10.155.4.3 `double gazebo::physics::RayShape::contactRetro` [protected]

10.155.4.4 `math::Vector3 gazebo::physics::RayShape::globalEndPos` [protected]

10.155.4.5 `math::Vector3 gazebo::physics::RayShape::globalStartPos` [protected]

Start and end positions of the ray in global cs.

10.155.4.6 `math::Vector3 gazebo::physics::RayShape::relativeEndPos` [protected]

10.155.4.7 `math::Vector3 gazebo::physics::RayShape::relativeStartPos` [protected]

Start and end positions of the ray, relative to the body.

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

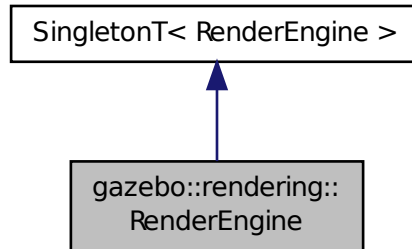
- `RayShape.hh`

10.156 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. 118) 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. 118). Load must happen before Init.*
- void **Load** ()
*Load the parameters for **Ogre** (p. 118). 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

10.156.1 Detailed Description

Adaptor to Ogre3d.

Provides the interface to load, initialize the rendering engine.

10.156.2 Member Enumeration Documentation

10.156.2.1 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.

10.156.3 Member Function Documentation

10.156.3.1 void gazebo::rendering::RenderEngine::AddResourcePath (const std::string & _uri)

Add a new path for **Ogre** (p. 118) 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>
----	------	--

10.156.3.2 `ScenePtr gazebo::rendering::RenderEngine::CreateScene (const std::string & _name, bool _enableVisualizations)`

Create a scene.

Parameters

in	_name	The name of the scene.
in	_enable- Visualizations	True enables visualization elements such as laser lines.

10.156.3.3 `void gazebo::rendering::RenderEngine::Fini ()`

Tears down the rendering engine.

10.156.3.4 `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

10.156.3.5 `ScenePtr gazebo::rendering::RenderEngine::GetScene (const std::string & _name)`

Get a scene by name.

Parameters

in	_name	Name of the scene to retrieve.
----	-------	--------------------------------

Returns

A pointer to the **Scene** (p. 746), or NULL if the scene doesn't exist.

10.156.3.6 `ScenePtr gazebo::rendering::RenderEngine::GetScene (unsigned int _index)`

Get a scene by index.

The index should be between 0 and **GetSceneCount()** (p. 726).

Parameters

in	_index	The index of the scene.
----	--------	-------------------------

Returns

A pointer to a **Scene** (p. 746), or NULL if the index was invalid.

10.156.3.7 `unsigned int gazebo::rendering::RenderEngine::GetSceneCount () const`

Get the number of scenes.

Returns

The number of scenes created by the **RenderEngine** (p. 722).

10.156.3.8 `void gazebo::rendering::RenderEngine::Init ()`

Initialize **Ogre** (p. 118). Load must happen before Init.

10.156.3.9 `void gazebo::rendering::RenderEngine::Load ()`

Load the parameters for **Ogre** (p. 118). Load must happen before Init.

10.156.3.10 `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.
-----------------	--------------------	----------------------------------

10.156.4 Member Data Documentation

10.156.4.1 `void* gazebo::rendering::RenderEngine::dummyContext` [protected]

GLX context used to render the scenes.Used for gui-less operation.

10.156.4.2 `void* gazebo::rendering::RenderEngine::dummyDisplay` [protected]

Pointer to the dummy display.Used for gui-less operation.

10.156.4.3 `uint64_t gazebo::rendering::RenderEngine::dummyWindowId` [protected]

ID for a dummy window. Used for gui-less operation.

10.156.4.4 `Ogre::Root* gazebo::rendering::RenderEngine::root`

Pointer to the root scene node.

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

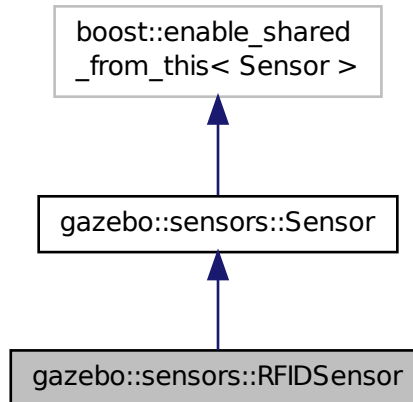
- **RenderEngine.hh**

10.157 gazebo::sensors::RFIDSensor Class Reference

Sensor (p. 765) class for RFID type of sensor.

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

Inheritance diagram for gazebo::sensors::RFIDSensor:



Public Member Functions

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

10.157.1 Detailed Description

Sensor (p. 765) class for RFID type of sensor.

10.157.2 Constructor & Destructor Documentation

10.157.2.1 gazebo::sensors::RFIDSensor::RFIDSensor ()

Constructor.

10.157.2.2 virtual gazebo::sensors::RFIDSensor::~~RFIDSensor () [virtual]

Destructor.

10.157.3 Member Function Documentation

10.157.3.1 virtual void gazebo::sensors::RFIDSensor::Fini () [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

10.157.3.2 virtual void gazebo::sensors::RFIDSensor::Init () [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.157.3.3 virtual void gazebo::sensors::RFIDSensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf) [virtual]

Load the sensor with SDF parameters.

Parameters

in	<code>_sdf</code>	SDF Sensor (p. 765) parameters
in	<code>_worldName</code>	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.157.3.4 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. 770).

10.157.3.5 `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. 771).

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

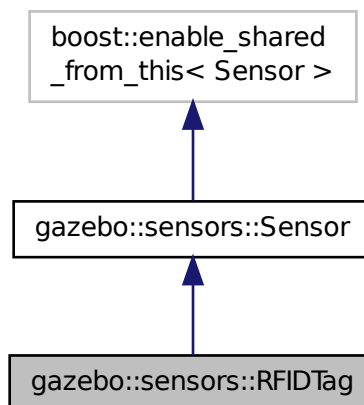
- `RFIDSensor.hh`

10.158 gazebo::sensors::RFIDTag Class Reference

RFIDTag (p. 729) to interact with RFIDTagSensors Nate check.

```
#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 ()`

returns pose of tag in world coordinate

- virtual void `Init ()`

Initialize the sensor.

- virtual void `Load (const std::string &_worldName, sdf::ElementPtr &_sdf)`

Load the sensor with SDF parameters.

- virtual void `Load (const std::string &_worldName)`

Load the sensor with default parameters.

Protected Member Functions

- virtual void `UpdateImpl (bool _force)`

Update the sensor information.

Additional Inherited Members

10.158.1 Detailed Description

RFIDTag (p. 729) to interact with RFIDTagSensors Nate check.

10.158.2 Constructor & Destructor Documentation

10.158.2.1 gazebo::sensors::RFIDTag::RFIDTag ()

Constructor.

10.158.2.2 virtual gazebo::sensors::RFIDTag::~~RFIDTag () [virtual]

Destructor.

10.158.3 Member Function Documentation

10.158.3.1 virtual void gazebo::sensors::RFIDTag::Fini () [virtual]

Finalize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 768).

10.158.3.2 `math::Pose gazebo::sensors::RFIDTag::GetTagPose () [inline]`

returns pose of tag in world coordinate

Returns

Pose of object

References `gazebo::physics::Entity::GetWorldPose()`.

10.158.3.3 `virtual void gazebo::sensors::RFIDTag::Init () [virtual]`

Initialize the sensor.

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

10.158.3.4 `virtual void gazebo::sensors::RFIDTag::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. 765) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

10.158.3.5 `virtual void gazebo::sensors::RFIDTag::Load (const std::string & _worldName) [virtual]`

Load the sensor with default parameters.

Parameters

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

Reimplemented from `gazebo::sensors::Sensor` (p. 770).

10.158.3.6 `virtual void gazebo::sensors::RFIDTag::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. 771).

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

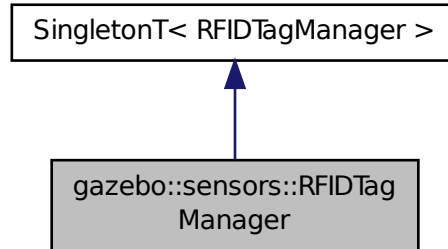
- **RFIDTag.hh**

10.159 gazebo::sensors::RFIDTagManager Class Reference

Nate fill in

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

Inheritance diagram for gazebo::sensors::RFIDTagManager:



Public Member Functions

- void **AddTaggedModel** (RFIDTag *_model)
Adds tag model.
- std::vector< RFIDTag * > **GetTags** ()
Gets vector of tagged models.

Additional Inherited Members

10.159.1 Detailed Description

Nate fill in

10.159.2 Member Function Documentation

10.159.2.1 void gazebo::sensors::RFIDTagManager::AddTaggedModel (RFIDTag * _model)

Adds tag model.

Parameters

<code>_model</code>	Tagged model to add Nate check
---------------------	--------------------------------

10.159.2.2 std::vector<RFIDTag*> gazebo::sensors::RFIDTagManager::GetTags () [inline]

Gets vector of tagged models.

Returns

vector of **RFIDTag** (p. 729), the tagged models Nate check

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

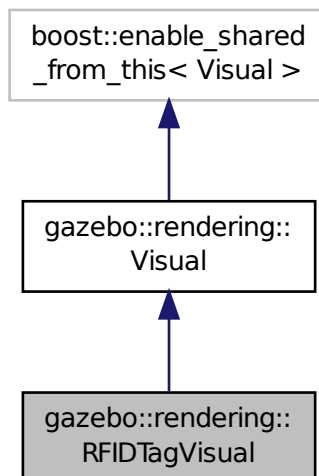
- **RFIDTagManager.hh**

10.160 gazebo::rendering::RFIDTagVisual Class Reference

Visualization for RFID tags sensor.

```
#include <RFIDTagVisual.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**10.160.1 Detailed Description**

Visualization for RFID tags sensor.

10.160.2 Constructor & Destructor Documentation

10.160.2.1 `gazebo::rendering::RFIDTagVisual::RFIDTagVisual (const std::string & _name, VisualIPtr _vis, const std::string & _topicName)`

Constructor.

Parameters

<code>in</code>	<code>_name</code>	Name of the visual.
<code>in</code>	<code>_vis</code>	Parent visual.
<code>in</code>	<code>_topicName</code>	Name of the topic that publishes RFID data.

See Also

`sensors::RFIDSensor` (p. 727)

10.160.2.2 `virtual gazebo::rendering::RFIDTagVisual::~~RFIDTagVisual () [virtual]`

Destructor.

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

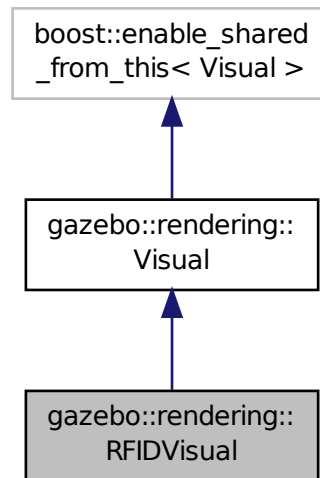
- **`RFIDTagVisual.hh`**

10.161 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

10.161.1 Detailed Description

Visualization for RFID sensor.

10.161.2 Constructor & Destructor Documentation

10.161.2.1 gazebo::rendering::RFIDVisual::RFIDVisual (const std::string & *_name*, **VisualPtr** *_vis*, const std::string & *_topicName*)

Constructor.

Parameters

in	<i>_name</i>	Name of the Visual (p. 931).
in	<i>_vis</i>	Parent Visual (p. 931).
in	<i>_topicName</i>	Name of the topic which publishes RFID data.

10.161.2.2 virtual gazebo::rendering::RFIDVisual::~~RFIDVisual () [virtual]

Destructor.

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

- **RFIDVisual.hh**

10.162 Road Class Reference

Used to render a strip of road.

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

10.162.1 Detailed Description

Used to render a strip of road.

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

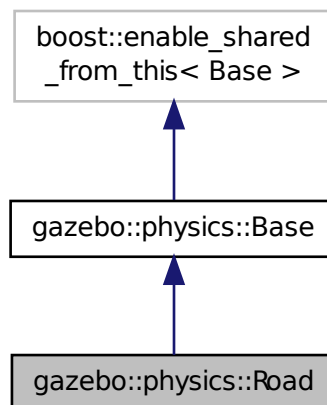
- **Road2d.hh**

10.163 gazebo::physics::Road Class Reference

for building a **Road** (p. 736) from SDF

```
#include <Road.hh>
```

Inheritance diagram for gazebo::physics::Road:



Public Member Functions

- **Road** (**BasePtr** _parent)
Constructor.
- virtual **~Road** ()
Destructor.
- virtual void **Init** ()
Initialize the object.
- void **Load** (**sdf::ElementPtr** _elem)
Load the road from SDF.

Additional Inherited Members

10.163.1 Detailed Description

for building a **Road** (p. 736) from SDF

10.163.2 Constructor & Destructor Documentation

10.163.2.1 gazebo::physics::Road::Road (**BasePtr** _parent)

Constructor.

10.163.2.2 virtual gazebo::physics::Road::~~Road () [virtual]

Destructor.

10.163.3 Member Function Documentation

10.163.3.1 virtual void gazebo::physics::Road::Init () [virtual]

Initialize the object.

Reimplemented from **gazebo::physics::Base** (p. 153).

10.163.3.2 void gazebo::physics::Road::Load (**sdf::ElementPtr** _elem) [virtual]

Load the road from SDF.

Reimplemented from **gazebo::physics::Base** (p. 154).

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

- **Road.hh**

10.164 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.

10.164.1 Constructor & Destructor Documentation

10.164.1.1 gazebo::rendering::Road2d::Road2d ()

Constructor.

10.164.1.2 virtual gazebo::rendering::Road2d::~~Road2d () [virtual]

Destructor.

10.164.2 Member Function Documentation

10.164.2.1 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**

10.165 gazebo::math::RotationSpline Class Reference

Spline (p. 808) for rotations.

```
#include <RotationSpline.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 tangeants when control points are updated.
- std::vector< **Quaternion** > **points**
the control points
- std::vector< **Quaternion** > **tangents**
the tangents

10.165.1 Detailed Description

Spline (p. 808) for rotations.

10.165.2 Constructor & Destructor Documentation

10.165.2.1 gazebo::math::RotationSpline::RotationSpline ()

Constructor. Sets the autoCalc to true.

10.165.2.2 gazebo::math::RotationSpline::~~RotationSpline ()

Destructor. Nothing is done.

10.165.3 Member Function Documentation

10.165.3.1 void gazebo::math::RotationSpline::AddPoint (const **Quaternion** & _p)

Adds a control point to the end of the spline.

Parameters

in	_p	control point
----	----	---------------

10.165.3.2 `void gazebo::math::RotationSpline::Clear ()`

Clears all the points in the spline.

10.165.3.3 `unsigned int gazebo::math::RotationSpline::GetNumPoints () const`

Gets the number of control points in the spline.

Returns

the count

10.165.3.4 `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)

10.165.3.5 `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

10.165.3.6 **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

10.165.3.7 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.

10.165.3.8 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.
----	------------------	--

10.165.3.9 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

10.165.4 Member Data Documentation

10.165.4.1 `bool gazebo::math::RotationSpline::autoCalc` [protected]

Automatic recalculation of tangents when control points are updated.

10.165.4.2 `std::vector<Quaternion> gazebo::math::RotationSpline::points` [protected]

the control points

10.165.4.3 `std::vector<Quaternion> gazebo::math::RotationSpline::tangents` [protected]

the tangents

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

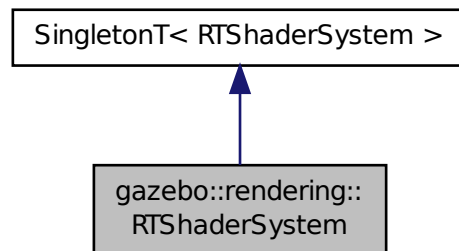
- **RotationSpline.hh**

10.166 gazebo::rendering::RTShaderSystem Class Reference

Implements **Ogre** (p. 118)'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** (**Scene** *_scene)
Add a scene manager.
- void **ApplyShadows** (**Scene** *_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** (**Scene** *_scene)
Remove a scene.
- void **RemoveShadows** (**Scene** *_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, **Scene** *_scene)
Set a viewport to use shaders.
- static void **DetachViewport** (Ogre::Viewport *_viewport, **Scene** *_scene)
Set a viewport to not use shaders.

Additional Inherited Members

10.166.1 Detailed Description

Implements **Ogre** (p. 118)'s Run-Time Shader system.

This class allows Gazebo to generate per-pixel shaders for every material at run-time.

10.166.2 Member Enumeration Documentation

10.166.2.1 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.

10.166.3 Member Function Documentation

10.166.3.1 void gazebo::rendering::RTShaderSystem::AddScene (Scene * _scene)

Add a scene manager.

Parameters

in	<code>_scene</code>	The scene to process
----	---------------------	----------------------

10.166.3.2 void gazebo::rendering::RTShaderSystem::ApplyShadows (Scene * _scene)

Apply shadows to a scene.

Parameters

in	<code>_scene</code>	The scene to receive shadows.
----	---------------------	-------------------------------

10.166.3.3 void gazebo::rendering::RTShaderSystem::AttachEntity (Visual * vis)

Set an Ogre::Entity to use RT shaders.

Parameters

in	<code>_vis</code>	Visual (p. 931) that will use the RTShaderSystem (p. 742).
----	-------------------	--

10.166.3.4 static void gazebo::rendering::RTShaderSystem::AttachViewport (Ogre::Viewport * _viewport, Scene * _scene) [static]

Set a viewport to use shaders.

Parameters

in	<code>_viewport</code>	The viewport to add.
in	<code>_scene</code>	The scene that the viewport uses.

10.166.3.5 void gazebo::rendering::RTShaderSystem::Clear ()

Clear the shader system.

10.166.3.6 void gazebo::rendering::RTShaderSystem::DetachEntity (Visual * *_vis*)

Remove and entity.

Parameters

in	<i>_vis</i>	Remove this visual.
----	-------------	---------------------

10.166.3.7 static void gazebo::rendering::RTShaderSystem::DetachViewport (Ogre::Viewport * *_viewport*, Scene * *_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.

10.166.3.8 void gazebo::rendering::RTShaderSystem::Fini ()

Finalize the shader system.

10.166.3.9 void gazebo::rendering::RTShaderSystem::GenerateShaders (Visual * *_vis*)

Generate shaders for an entity.

Parameters

in	<i>_vis</i>	The visual to generate shaders for.
----	-------------	-------------------------------------

10.166.3.10 void gazebo::rendering::RTShaderSystem::Init ()

Init the run time shader system.

10.166.3.11 void gazebo::rendering::RTShaderSystem::RemoveScene (Scene * *_scene*)

Remove a scene.

Parameters

in	<i>The</i>	scene to remove
----	------------	-----------------

10.166.3.12 `void gazebo::rendering::RTShaderSystem::RemoveShadows (Scene * _scene)`

Remove shadows from a scene.

Parameters

<code>in</code>	<code>_scene</code>	The scene to remove shadows from.
-----------------	---------------------	-----------------------------------

10.166.3.13 `void gazebo::rendering::RTShaderSystem::SetPerPixelLighting (bool _set)`

Set the lighting model to per pixel or per vertex.

Parameters

<code>in</code>	<code>_set</code>	True means to use per-pixel shaders.
-----------------	-------------------	--------------------------------------

10.166.3.14 `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:

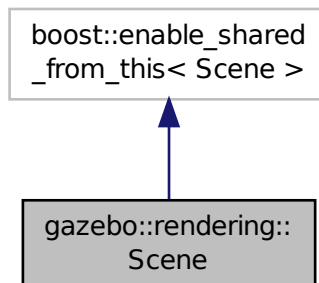
- **RTShaderSystem.hh**

10.167 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. 746).*
- **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.
- **GpuLaserPtr CreateGpuLaser** (const std::string &_name, bool _autoRender=true)
Create laser that generates data from rendering.
- void **CreateGrid** (uint32_t _cellCount, float _cellLength, float _lineWidth, const **common::Color** &_color)
Create a square grid of cells.
- **UserCameraPtr CreateUserCamera** (const std::string &_name)
Create a user camera.
- void **DrawLine** (const **math::Vector3** &_start, const **math::Vector3** &_end, const std::string &_name)
Draw a named line.
- **common::Color GetAmbientColor** () const
Get the ambient color.
- **common::Color GetBackgroundColor** () const
Get the background color.
- **CameraPtr GetCamera** (uint32_t _index) const
Get a camera based on an index.
- **CameraPtr GetCamera** (const std::string &_name) const
Get a camera by name.
- uint32_t **GetCameraCount** () const
Get the number of cameras in this scene.
- **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.
- **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. 746).*

 - 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.

Public Attributes

- SkyX::SkyX * **skyx**
Pointer to the sky.

10.167.1 Detailed Description

Representation of an entire scene graph.

Maintains all the Visuals, Lights, and Cameras for a World.

10.167.2 Constructor & Destructor Documentation

10.167.2.1 gazebo::rendering::Scene::Scene (const std::string & _name, bool _enableVisualizations = false)

Constructor.

Parameters

in	<i>_name</i>	Name of the scene.
in	<i>_enable-Visualizations</i>	True to enable visualizations, this should be set to true for user interfaces, and false for sensor generation.

10.167.2.2 virtual gazebo::rendering::Scene::~Scene () [virtual]

Destructor.

10.167.3 Member Function Documentation

10.167.3.1 void gazebo::rendering::Scene::AddVisual (VisualPtr _vis)

Add a visual to the scene.

Parameters

in	<i>_vis</i>	Visual (p. 931) to add.
----	-------------	--------------------------------

10.167.3.2 `void gazebo::rendering::Scene::Clear ()`

Clear **rendering::Scene** (p. 746).

10.167.3.3 `VisualIPtr gazebo::rendering::Scene::CloneVisual (const std::string & _visualName, const std::string & _newName)`

Clone a visual.

Parameters

in	<i>_visualName</i>	Name of the visual to clone.
in	<i>_newName</i>	New name of the visual.

Returns

Pointer to the cloned visual.

10.167.3.4 `CameraPtr gazebo::rendering::Scene::CreateCamera (const std::string & _name, bool _autoRender = true)`

Create a camera.

Parameters

in	<i>_name</i>	Name of the new camera.
in	<i>_autoRender</i>	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.167.3.5 `DepthCameraPtr gazebo::rendering::Scene::CreateDepthCamera (const std::string & _name, bool _autoRender = true)`

Create depth camera.

Parameters

in	<i>_name</i>	Name of the new camera.
in	<i>_autoRender</i>	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.167.3.6 GpuLaserPtr gazebo::rendering::Scene::CreateGpuLaser (const std::string & *_name*, bool *_autoRender* = true)

Create laser that generates data from rendering.

Parameters

in	<i>_name</i>	Name of the new laser.
in	<i>_autoRender</i>	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new laser.

10.167.3.7 void gazebo::rendering::Scene::CreateGrid (uint32_t *_cellCount*, float *_cellLength*, float *_lineWidth*, const common::Color & *_color*)

Create a square grid of cells.

Parameters

in	<i>_cellCount</i>	Number of grid cells in one direction.
in	<i>_cellLength</i>	Length of one grid cell.
in	<i>_lineWidth</i>	Width of the grid lines.
in	<i>_color</i>	Color of the grid lines.

10.167.3.8 UserCameraPtr gazebo::rendering::Scene::CreateUserCamera (const std::string & *_name*)

Create a user camera.

A user camera is one design for use with a GUI.

Parameters

in	<i>_name</i>	Name of the UserCamera (p. 878).
----	--------------	---

Returns

A pointer to the new **UserCamera** (p. 878).

10.167.3.9 void gazebo::rendering::Scene::DrawLine (const math::Vector3 & *_start*, const math::Vector3 & *_end*, const std::string & *_name*)

Draw a named line.

Parameters

in	<i>_start</i>	Start position of the line.
in	<i>_end</i>	End position of the line.
in	<i>_name</i>	Name of the line.

10.167.3.10 `common::Color gazebo::rendering::Scene::GetAmbientColor () const`

Get the ambient color.

Returns

The scene's ambient color.

10.167.3.11 `common::Color gazebo::rendering::Scene::GetBackgroundColor () const`

Get the background color.

Returns

The background color.

10.167.3.12 `CameraPtr gazebo::rendering::Scene::GetCamera (uint32_t _index) const`

Get a camera based on an index.

Index must be between 0 and `Scene::GetCameraCount` (p. 752).

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.

10.167.3.13 `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.

10.167.3.14 `uint32_t gazebo::rendering::Scene::GetCameraCount () const`

Get the number of cameras in this scene.

Returns

Number of lasers.

10.167.3.15 `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><i>_camera</i></code>	Pointer to the camera.
<code>in</code>	<code><i>_mousePos</i></code>	2D position of the mouse in pixels.

Returns

3D position of the first contact point.

10.167.3.16 `Grid* gazebo::rendering::Scene::GetGrid (uint32_t _index) const`

Get a grid based on an index.

Index must be between 0 and `Scene::GetGridCount` (p. 753).

Parameters

<code>in</code>	<code><i>_index</i></code>	Index of the grid.
-----------------	----------------------------	--------------------

10.167.3.17 `uint32_t gazebo::rendering::Scene::GetGridCount () const`

Get the number of grids.

Returns

The number of grids.

10.167.3.18 `Heightmap* gazebo::rendering::Scene::GetHeightmap () const`

Get a pointer to the heightmap.

Returns

Pointer to the heightmap, NULL if no heightmap.

10.167.3.19 `uint32_t gazebo::rendering::Scene::GetId () const`

Get the scene ID.

Returns

The ID of the scene.

10.167.3.20 `std::string gazebo::rendering::Scene::GetIdString () const`

Get the scene Id as a string.

Returns

The ID as a string.

10.167.3.21 `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.

10.167.3.22 `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. 754).

Parameters

<code>in</code>	<code>_index</code>	Index of the light.
-----------------	---------------------	---------------------

Returns

Pointer to the **Light** (p. 448) or NULL if index was invalid.

10.167.3.23 `uint32_t gazebo::rendering::Scene::GetLightCount () const`

Get the count of the lights.

Returns

The number of lights.

10.167.3.24 `Ogre::SceneManager* gazebo::rendering::Scene::GetManager () const`

Get the OGRE scene manager.

Returns

Pointer to the **Ogre** (p. 118) SceneManager.

10.167.3.25 `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.

10.167.3.26 `std::string gazebo::rendering::Scene::GetName () const`

Get the name of the scene.

Returns

Name of the scene.

10.167.3.27 `VisualPtr gazebo::rendering::Scene::GetSelectedVisual () const`

Get the currently selected visual.

Returns

Pointer to the currently selected visual, or NULL if nothing is selected.

10.167.3.28 `bool gazebo::rendering::Scene::GetShadowsEnabled () const`

Get whether shadows are on or off.

Returns

True if shadows are enabled.

10.167.3.29 `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. 756).

Parameters

<code>in</code>	<code>_index</code>	Index of the <code>UserCamera</code> (p. 878) to get.
-----------------	---------------------	---

Returns

Pointer to the **UserCamera** (p. 878), or NULL if the index was invalid.

10.167.3.30 `uint32_t gazebo::rendering::Scene::GetUserCameraCount () const`

Get the number of user cameras in this scene.

Returns

The number of user cameras.

10.167.3.31 `VisualPtr gazebo::rendering::Scene::GetVisual (const std::string & _name) const`

Get a visual by name.

10.167.3.32 `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	<i>camera</i>	The ogre camera, used to do mouse picking
in	<i>mousePos</i>	The position of the mouse in screen coordinates
out	<i>_mod</i>	Used for object manipulation

Returns

The selected entity, or NULL

10.167.3.33 `VisualPtr gazebo::rendering::Scene::GetVisualAt (CameraPtr _camera, const math::Vector2i & _mousePos)`

Get a visual at a mouse position.

Parameters

in	<i>_camera</i>	Pointer to the camera used to project the mouse position.
in	<i>_mousePos</i>	The 2d position of the mouse in pixels.

Returns

Pointer to the visual, NULL if none found.

10.167.3.34 `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.

10.167.3.35 `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.

10.167.3.36 `VisualPtr gazebo::rendering::Scene::GetWorldVisual () const`

Get the top level world visual.

Returns

Pointer to the world visual.

10.167.3.37 `void gazebo::rendering::Scene::Init ()`

Init **rendering::Scene** (p. 746).

10.167.3.38 `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.
----	---------------	----------------------------

10.167.3.39 `void gazebo::rendering::Scene::Load ()`

Load the scene with default parameters.

10.167.3.40 `void gazebo::rendering::Scene::PreRender ()`

Process all received messages.

10.167.3.41 void gazebo::rendering::Scene::PrintSceneGraph ()

Print the scene graph to std_out.

10.167.3.42 void gazebo::rendering::Scene::RemoveVisual (VisualPtr _vis)

Remove a visual from the scene.

Parameters

in	<i>_vis</i>	Visual (p. 931) to remove.
----	-------------	-----------------------------------

10.167.3.43 void gazebo::rendering::Scene::SelectVisual (const std::string & _name, const std::string & _mode)

Select a visual by name.

Parameters

in	<i>_name</i>	Name of the visual to select.
in	<i>_mode</i>	Selection mode (normal, or move).

10.167.3.44 void gazebo::rendering::Scene::SetAmbientColor (const common::Color & _color)

Set the ambient color.

Parameters

in	<i>_color</i>	The ambient color to use.
----	---------------	---------------------------

10.167.3.45 void gazebo::rendering::Scene::SetBackgroundColor (const common::Color & _color)

Set the background color.

Parameters

in	<i>_color</i>	The background color.
----	---------------	-----------------------

10.167.3.46 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.

10.167.3.47 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
----	-----------------	---------------------------------

10.167.3.48 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
----	---------------	--

10.167.3.49 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.

10.167.3.50 void gazebo::rendering::Scene::SnapVisualToNearestBelow (const std::string & *_visualName*)

Move the visual to be ontop of the nearest visual below it.

Parameters

in	<i>_visualName</i>	Name of the visual to move.
----	--------------------	-----------------------------

10.167.3.51 std::string gazebo::rendering::Scene::StripSceneName (const std::string & *_name*) const

Remove the name of scene from a string.

Parameters

in	<i>_name</i>	Name to string the scene name from.
----	--------------	-------------------------------------

Returns

The stripped name.

10.167.4 Member Data Documentation**10.167.4.1 SkyX::SkyX* gazebo::rendering::Scene::skyx**

Pointer to the sky.

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

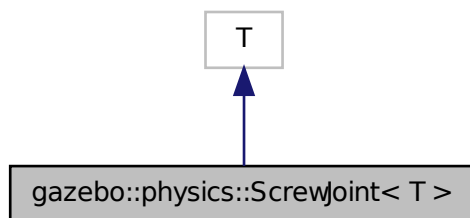
- **Scene.hh**

10.168 gazebo::physics::ScrewJoint< T > Class Template Reference

A screw joint.

```
#include <ScrewJoint.hh>
```

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

**Public Member Functions**

- **ScrewJoint** (**BasePtr** _parent)
Constructor.
- virtual **~ScrewJoint** ()
Destructor.
- virtual **math::Vector3 GetAnchor** (int) const
Get the anchor.
- virtual void **SetAnchor** (int, const **math::Vector3** &anchor)
Set the anchor.
- virtual void **SetThreadPitch** (int _index, double _threadPitch)=0
Set screw joint thread pitch.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)
Load a ScreJoint.

Protected Attributes

- math::Vector3 **fakeAnchor**
- double **threadPitch**

10.168.1 Detailed Description

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

A screw joint.

10.168.2 Constructor & Destructor Documentation

10.168.2.1 `template<class T> gazebo::physics::ScrewJoint< T >::ScrewJoint (BasePtr _parent) [inline]`

Constructor.

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

Destructor.

10.168.3 Member Function Documentation

10.168.3.1 `template<class T> virtual math::Vector3 gazebo::physics::ScrewJoint< T >::GetAnchor (int) const [inline], [virtual]`

Get the anchor.

10.168.3.2 `template<class T> virtual void gazebo::physics::ScrewJoint< T >::Load (sdf::ElementPtr _sdf) [inline], [protected], [virtual]`

Load a ScreJoint.

Reimplemented in `gazebo::physics::BulletScrewJoint` (p. 216), and `gazebo::physics::ODEScrewJoint` (p. 624).

10.168.3.3 `template<class T> virtual void gazebo::physics::ScrewJoint< T >::SetAnchor (int , const math::Vector3 & anchor) [inline], [virtual]`

Set the anchor.

10.168.3.4 `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

<code>in</code>	<code><i>_index</i></code>	Index of the axis.
<code>in</code>	<code><i>_threadPitch</i></code>	Thread pitch value.

Implemented in `gazebo::physics::ODEScrewJoint` (p. 625).

10.168.4 Member Data Documentation

10.168.4.1 `template<class T> math::Vector3 gazebo::physics::ScrewJoint< T >::fakeAnchor [protected]`

Referenced by `gazebo::physics::ScrewJoint< BulletJoint >::GetAnchor()`, and `gazebo::physics::ScrewJoint< BulletJoint >::SetAnchor()`.

10.168.4.2 `template<class T> double gazebo::physics::ScrewJoint< T >::threadPitch [protected]`

Referenced by `gazebo::physics::ScrewJoint< BulletJoint >::Load()`.

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

- `ScrewJoint.hh`

10.169 sdf::SDF Class Reference

Base **SDF** (p. 762) 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. 762) 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**

10.169.1 Detailed Description

Base **SDF** (p. 762) class.

10.169.2 Constructor & Destructor Documentation

10.169.2.1 sdf::SDF::SDF ()

10.169.3 Member Function Documentation

10.169.3.1 void sdf::SDF::PrintDescription ()

10.169.3.2 void sdf::SDF::PrintDoc ()

10.169.3.3 void sdf::SDF::PrintValues ()

10.169.3.4 void sdf::SDF::PrintWiki ()

10.169.3.5 void sdf::SDF::SetFromString (const std::string & *_sdfData*)

Set **SDF** (p. 762) values from a string.

10.169.3.6 std::string sdf::SDF::ToString () const

10.169.3.7 void sdf::SDF::Write (const std::string & *_filename*)

10.169.4 Member Data Documentation

10.169.4.1 ElementPtr sdf::SDF::root

10.169.4.2 std::string sdf::SDF::version [static]

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

- **SDF.hh**

10.170 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. 763) object.

- `std::string GetVisualName ()` const

Get the name of the visual the selection obj is attached to.

- void **Init** ()

Initialize the `rendering::SelectionObj` (p. 763) 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.

10.170.1 Detailed Description

A graphical selection object.

Used to draw a visual around a selected object.

10.170.2 Constructor & Destructor Documentation

10.170.2.1 `gazebo::rendering::SelectionObj::SelectionObj (Scene * _scene)`

Constructor.

Parameters

<code>in</code>	<code>_scene</code>	Scene (p. 746) to use.
-----------------	---------------------	-------------------------------

10.170.2.2 `virtual gazebo::rendering::SelectionObj::~SelectionObj ()` [virtual]

Destructor.

10.170.3 Member Function Documentation

10.170.3.1 `void gazebo::rendering::SelectionObj::Attach (VisualPtr _visual)`

Set the position of the node.

Parameters

<code>in</code>	<i>This</i>	draws the selection object around the passed in visual.
-----------------	-------------	---

10.170.3.2 void gazebo::rendering::SelectionObj::Clear ()

Clear the **rendering::SelectionObj** (p. 763) object.

10.170.3.3 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.

10.170.3.4 void gazebo::rendering::SelectionObj::Init ()

Initialize the **rendering::SelectionObj** (p. 763) object.

10.170.3.5 bool gazebo::rendering::SelectionObj::IsActive () const

Return true if the user is move the selection obj.

Returns

True if something is selected.

10.170.3.6 void gazebo::rendering::SelectionObj::SetActive (bool *_active*)

Set true if the user is moving the selection obj.

Parameters

<i>in</i>	<i>_active</i>	True if the user is interacting with the selection object.
-----------	----------------	--

10.170.3.7 void gazebo::rendering::SelectionObj::SetHighlight (const std::string & *_mod*)

Highlight the selection object based on a modifier.

Parameters

<i>in</i>	<i>_mod</i>	Modifier used when highlighting the selection object.
-----------	-------------	---

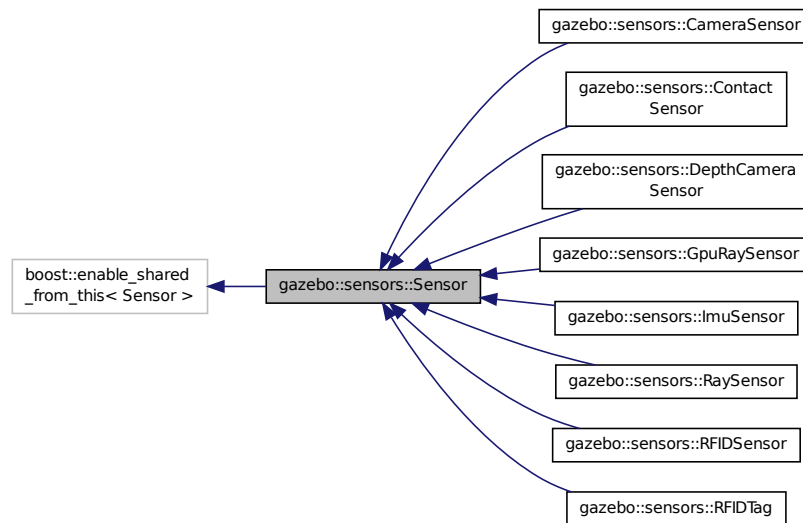
The documentation for this class was generated from the following file:

- **SelectionObj.hh**

10.171 gazebo::sensors::Sensor Class Reference

```
#include <Sensor.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 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.
- 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 active.
- std::vector< event::ConnectionPtr > **connections**
- common::Time **lastUpdateTime**
- transport::NodePtr **node**
- std::string **parentName**
- std::vector< SensorPluginPtr > **plugins**
- math::Pose **pose**
- transport::SubscriberPtr **poseSub**
- sdf::ElementPtr **sdf**
- common::Time **updatePeriod**
- gazebo::physics::WorldPtr **world**

10.171.1 Constructor & Destructor Documentation

10.171.1.1 gazebo::sensors::Sensor::Sensor ()

Constructor.

10.171.1.2 `virtual gazebo::sensors::Sensor::~~Sensor() [virtual]`

Destructor.

10.171.2 Member Function Documentation

10.171.2.1 `void gazebo::sensors::Sensor::FillMsg(msgs::Sensor & _msg)`

fills a `msgs::Sensor` message.

Parameters

<code>_msg</code>	Message to fill
-------------------	-----------------

10.171.2.2 `virtual void gazebo::sensors::Sensor::Fini() [virtual]`

Finalize the sensor.

Reimplemented in `gazebo::sensors::CameraSensor` (p.257), `gazebo::sensors::RaySensor` (p.713), `gazebo::sensors::ContactSensor` (p.301), `gazebo::sensors::GpuRaySensor` (p.382), `gazebo::sensors::DepthCameraSensor` (p.318), `gazebo::sensors::RFIDSensor` (p.728), and `gazebo::sensors::RFIDTag` (p.730).

10.171.2.3 `common::Time gazebo::sensors::Sensor::GetLastUpdateTime()`

return last update time

Returns

Time of last update

10.171.2.4 `std::string gazebo::sensors::Sensor::GetName() const`

Get name.

Returns

name of sensor

10.171.2.5 `std::string gazebo::sensors::Sensor::GetParentName() const`

Returns the name of the sensor parent.

The parent name is set by `Sensor::SetParent` (p.771).

Returns

Name of Parent

10.171.2.6 `virtual math::Pose gazebo::sensors::Sensor::GetPose () const [virtual]`

Get the current pose.

Returns

Current pose of the sensor

10.171.2.7 `std::string gazebo::sensors::Sensor::GetScopedName () const`

Get fully scoped name of the sensor.

Returns

world_name::parent_name::sensor_name

10.171.2.8 `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. 716), and **`gazebo::sensors::CameraSensor`** (p. 258).

10.171.2.9 `std::string gazebo::sensors::Sensor::GetType () const`

Get sensor type.

Returns

Type of sensor

10.171.2.10 `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

10.171.2.11 `std::string gazebo::sensors::Sensor::GetWorldName () const`

Returns the name of the world the sensor is in.

Returns

Name of the world

10.171.2.12 `virtual void gazebo::sensors::Sensor::Init () [virtual]`

Initialize the sensor.

Reimplemented in `gazebo::sensors::RaySensor` (p.717), `gazebo::sensors::ContactSensor` (p.303), `gazebo::sensors::CameraSensor` (p.258), `gazebo::sensors::GpuRaySensor` (p.386), `gazebo::sensors::DepthCameraSensor` (p.319), `gazebo::sensors::RFIDSensor` (p.728), and `gazebo::sensors::RFIDTag` (p.731).

10.171.2.13 `bool gazebo::sensors::Sensor::IsActive ()`

Returns true if sensor generation is active.

Returns

True if active, false if not

10.171.2.14 `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.765) parameters
<code>in</code>	<code>_worldName</code>	Name of world to load from

Reimplemented in `gazebo::sensors::RaySensor` (p.717), `gazebo::sensors::ContactSensor` (p.303), `gazebo::sensors::CameraSensor` (p.258), and `gazebo::sensors::RFIDSensor` (p.728).

10.171.2.15 `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::RaySensor` (p.717), `gazebo::sensors::ContactSensor` (p.303), `gazebo::sensors::CameraSensor` (p.258), `gazebo::sensors::GpuRaySensor` (p.387), `gazebo::sensors::DepthCameraSensor` (p.319), `gazebo::sensors::RFIDSensor` (p.728), and `gazebo::sensors::RFIDTag` (p.731).

10.171.2.16 `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::CameraSensor` (p.259), and `gazebo::sensors::DepthCameraSensor` (p.319).

10.171.2.17 `virtual void gazebo::sensors::Sensor::SetParent (const std::string & _name) [virtual]`

Set the parent of the sensor.

Reimplemented in `gazebo::sensors::CameraSensor` (p. 259), and `gazebo::sensors::DepthCameraSensor` (p. 320).

10.171.2.18 `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
-----------------	------------------	-----------------------

10.171.2.19 `void gazebo::sensors::Sensor::Update (bool _force)`

Update the sensor.

Parameters

<code>in</code>	<code>_force</code>	True to force update, false otherwise
-----------------	---------------------	---------------------------------------

10.171.2.20 `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

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

Reimplemented in `gazebo::sensors::CameraSensor` (p. 259), `gazebo::sensors::RaySensor` (p. 718), `gazebo::sensors::ContactSensor` (p. 303), `gazebo::sensors::GpuRaySensor` (p. 388), `gazebo::sensors::DepthCameraSensor` (p. 320), `gazebo::sensors::RFIDSensor` (p. 729), and `gazebo::sensors::RFIDTag` (p. 731).

10.171.3 Member Data Documentation

10.171.3.1 `bool gazebo::sensors::Sensor::active [protected]`

True if active.

10.171.3.2 `std::vector<event::ConnectionPtr> gazebo::sensors::Sensor::connections [protected]`

10.171.3.3 `common::Time gazebo::sensors::Sensor::lastUpdateTime [protected]`

- 10.171.3.4 `transport::NodePtr gazebo::sensors::Sensor::node` [protected]
- 10.171.3.5 `std::string gazebo::sensors::Sensor::parentName` [protected]
- 10.171.3.6 `std::vector<SensorPluginPtr> gazebo::sensors::Sensor::plugins` [protected]
- 10.171.3.7 `math::Pose gazebo::sensors::Sensor::pose` [protected]
- 10.171.3.8 `transport::SubscriberPtr gazebo::sensors::Sensor::poseSub` [protected]
- 10.171.3.9 `sdf::ElementPtr gazebo::sensors::Sensor::sdf` [protected]
- 10.171.3.10 `common::Time gazebo::sensors::Sensor::updatePeriod` [protected]
- 10.171.3.11 `gazebo::physics::WorldPtr gazebo::sensors::Sensor::world` [protected]

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

- **Sensor.hh**

10.172 SensorFactor Class Reference

The sensor factory; the class is just for namespacing purposes.

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

10.172.1 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**

10.173 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).

10.173.1 Member Function Documentation

10.173.1.1 static void gazebo::sensors::SensorFactory::GetSensorTypes (std::vector< std::string > & *_types*) [static]

Get all the sensor types.

Parameters

<i>_types</i>	Vector of strings of the sensor types, populated by function
---------------	--

10.173.1.2 static **SensorPtr** gazebo::sensors::SensorFactory::NewSensor (const std::string & *_classname*) [static]

Create a new instance of a sensor.

Used by the world when reading the world file.

Parameters

in	<i>Name</i>	of sensor class
----	-------------	-----------------

Returns

Pointer to **Sensor** (p. 765)

10.173.1.3 static void gazebo::sensors::SensorFactory::RegisterAll () [static]

Register all known sensors.

- **sensors::CameraSensor** (p. 255)
- **sensors::DepthCameraSensor** (p. 317)
- **sensors::GpuRaySensor** (p. 378)
- **sensors::RaySensor** (p. 711)
- **sensors::ContactSensor** (p. 300)
- **sensors::RFIDSensor** (p. 727)
- **sensors::RFIDTag** (p. 729)

10.173.1.4 static void gazebo::sensors::SensorFactory::RegisterSensor (const std::string & *_classname*, **SensorFactoryFn** *_factoryfn*) [static]

Register a sensor class (called by sensor registration function).

Parameters

in	<i>_classname</i>	Name of class of sensor to register
	<i>_factoryfn</i>	Function handle for registration Nate check

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

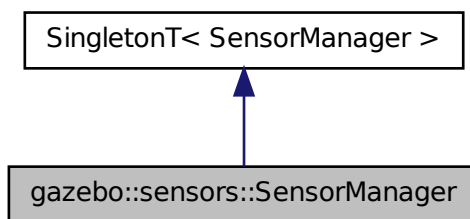
- **SensorFactory.hh**

10.174 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.
- 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

10.174.1 Detailed Description

Class to manage and update all sensors.

10.174.2 Member Function Documentation

10.174.2.1 `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

<i>in</i>	<i>_elem</i>	The SDF element that describes the sensor
<i>in</i>	<i>_worldName</i>	Name of the world in which to create the sensor
<i>in</i>	<i>_parentName</i>	The name of the parent link which the sensor is attached to.

Returns

The name of the sensor

10.174.2.2 `void gazebo::sensors::SensorManager::Fini ()`

Finalize all the sensors.

10.174.2.3 `SensorPtr gazebo::sensors::SensorManager::GetSensor (const std::string & _name)`

Get a sensor.

Parameters

<i>in</i>	<i>_name</i>	The name of a sensor to find.
-----------	--------------	-------------------------------

Returns

A pointer to the sensor. NULL if not found.

10.174.2.4 `void gazebo::sensors::SensorManager::GetSensorTypes (std::vector< std::string > & _types) const`

Get all the sensor types.

Parameters

out	All	the sensor types.
-----	-----	-------------------

10.174.2.5 void gazebo::sensors::SensorManager::Init ()

Init all the sensors.

10.174.2.6 void gazebo::sensors::SensorManager::RemoveSensor (const std::string & *_name*)

Remove a sensor.

Parameters

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

10.174.2.7 void gazebo::sensors::SensorManager::RemoveSensors ()

Remove all sensors.

10.174.2.8 void gazebo::sensors::SensorManager::Run ()

Run the sensor manager update in a new thread.

10.174.2.9 bool gazebo::sensors::SensorManager::SensorsInitialized ()

True if SensorManager::initSensors queue is empty i.e.

all sensors managed by **SensorManager** (p. 774) have been initialized

10.174.2.10 void gazebo::sensors::SensorManager::Stop ()

Stop the run thread.

10.174.2.11 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:

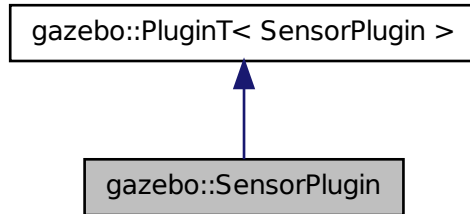
- **SensorManager.hh**

10.175 gazebo::SensorPlugin Class Reference

A plugin with access to physics::Sensor.

```
#include <Plugin.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

10.175.1 Detailed Description

A plugin with access to physics::Sensor.

See [reference](#).

10.175.2 Constructor & Destructor Documentation

10.175.2.1 gazebo::SensorPlugin::SensorPlugin () [inline]

Constructor.

References `gazebo::SENSOR_PLUGIN`, and `gazebo::PluginT< SensorPlugin >::type`.

10.175.2.2 `virtual gazebo::SensorPlugin::~~SensorPlugin () [inline],[virtual]`

Destructor.

10.175.3 Member Function Documentation

10.175.3.1 `virtual void gazebo::SensorPlugin::Init () [inline],[virtual]`

Override this method for custom plugin initialization behavior.

10.175.3.2 `virtual void gazebo::SensorPlugin::Load (sensors::SensorPtr _sensor, sdf::ElementPtr _sdf) [pure virtual]`

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

<code>in</code>	<code>_sensor</code>	Pointer the Sensor.
<code>in</code>	<code>_sdf</code>	Pointer the the SDF element of the plugin.

10.175.3.3 `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`

10.176 gazebo::Server Class Reference

```
#include <Server.hh>
```

Public Member Functions

- `Server ()`
- `virtual ~Server ()`
- `void Fini ()`
- `bool GetInitialized () const`
- `void Init ()`
- `bool Load (const std::string &_filename="worlds/empty.world")`
- `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**

10.176.1 Constructor & Destructor Documentation

10.176.1.1 gazebo::Server::Server ()

10.176.1.2 virtual gazebo::Server::~Server () [virtual]

10.176.2 Member Function Documentation

10.176.2.1 void gazebo::Server::Fini ()

10.176.2.2 bool gazebo::Server::GetInitialized () const

10.176.2.3 void gazebo::Server::Init ()

10.176.2.4 bool gazebo::Server::Load (const std::string & *filename* = "worlds/empty.world")

10.176.2.5 bool gazebo::Server::ParseArgs (int *argc*, char ** *argv*)

10.176.2.6 void gazebo::Server::PrintUsage ()

10.176.2.7 void gazebo::Server::Run ()

10.176.2.8 void gazebo::Server::SetParams (const common::StrStr_M & *params*)

10.176.2.9 void gazebo::Server::Stop ()

10.176.3 Member Data Documentation

10.176.3.1 int gazebo::Server::systemPluginsArgc

10.176.3.2 char** gazebo::Server::systemPluginsArgv

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

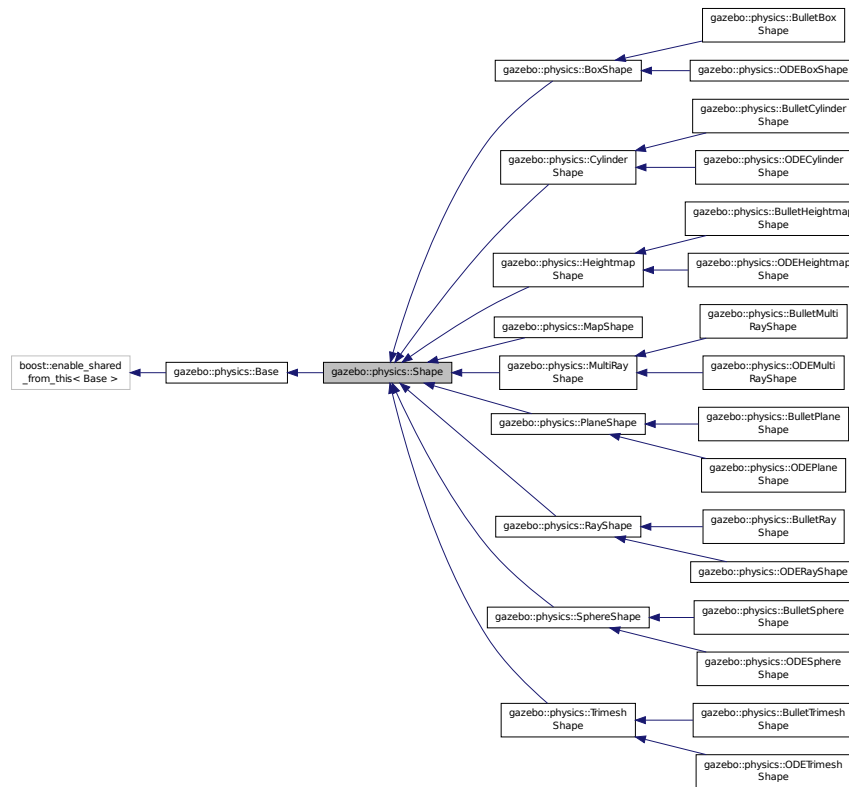
- **Server.hh**

10.177 gazebo::physics::Shape Class Reference

Base (p. 145) class for all shapes.

```
#include <Shape.hh>
```

Inheritance diagram for gazebo::physics::Shape:



Public Member Functions

- **Shape** (*CollisionPtr* p)
Constructor.
- virtual **~Shape** ()
Destructor.
- virtual void **FillMsg** (msgs::Geometry &_msg)=0
- virtual void **FillShapeMsg** (msgs::Geometry &_msg) **GAZEBO_DEPRECATED**
- virtual void **GetInertial** (double _mass, **InertialPtr** _inertial) const
Get inertial for a shape.
- virtual double **GetMass** (double _density) const
Get the mass of a shape.
- virtual void **Init** ()=0
Initialize the shape.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)=0

Public Attributes

- **CollisionPtr** collisionParent

Additional Inherited Members

10.177.1 Detailed Description

Base (p. 145) class for all shapes.

10.177.2 Constructor & Destructor Documentation

10.177.2.1 gazebo::physics::Shape::Shape (CollisionPtr p)

Constructor.

10.177.2.2 virtual gazebo::physics::Shape::~~Shape () [virtual]

Destructor.

10.177.3 Member Function Documentation

10.177.3.1 virtual void gazebo::physics::Shape::FillMsg (msgs::Geometry & _msg) [pure virtual]

Implemented in **gazebo::physics::MultiRayShape** (p. 552), **gazebo::physics::RayShape** (p. 720), **gazebo::physics::HeightmapShape** (p. 401), **gazebo::physics::CylinderShape** (p. 309), **gazebo::physics::MapShape** (p. 485), **gazebo::physics::PlaneShape** (p. 673), **gazebo::physics::TrimeshShape** (p. 867), **gazebo::physics::BoxShape** (p. 164), and **gazebo::physics::SphereShape** (p. 807).

10.177.3.2 virtual void gazebo::physics::Shape::FillShapeMsg (msgs::Geometry & _msg) [virtual]

Reimplemented in **gazebo::physics::BoxShape** (p. 164).

10.177.3.3 virtual void gazebo::physics::Shape::GetInertial (double _mass, InertialPtr _inertial) const [virtual]

Get inertial for a shape.

Reimplemented in **gazebo::physics::CylinderShape** (p. 309), **gazebo::physics::BoxShape** (p. 164), and **gazebo::physics::SphereShape** (p. 807).

10.177.3.4 virtual double gazebo::physics::Shape::GetMass (double _density) const [inline],[virtual]

Get the mass of a shape.

Reimplemented in **gazebo::physics::CylinderShape** (p. 310), **gazebo::physics::BoxShape** (p. 165), **gazebo::physics::TrimeshShape** (p. 868), and **gazebo::physics::SphereShape** (p. 808).

10.177.3.5 virtual void gazebo::physics::Shape::Init () [pure virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::Base** (p. 153).

Implemented in [gazebo::physics::RayShape](#) (p. 721), [gazebo::physics::MapShape](#) (p. 486), [gazebo::physics::HeightmapShape](#) (p. 403), [gazebo::physics::TrimeshShape](#) (p. 868), [gazebo::physics::BulletHeightmapShape](#) (p. 178), [gazebo::physics::MultiRayShape](#) (p. 555), [gazebo::physics::ODETrimeshShape](#) (p. 634), [gazebo::physics::ODEHeightmapShape](#) (p. 586), [gazebo::physics::PlaneShape](#) (p. 673), [gazebo::physics::BulletTrimeshShape](#) (p. 224), [gazebo::physics::BoxShape](#) (p. 165), [gazebo::physics::CylinderShape](#) (p. 310), and [gazebo::physics::SphereShape](#) (p. 808).

10.177.3.6 virtual void gazebo::physics::Shape::ProcessMsg (const msgs::Geometry & _msg) [pure virtual]

Implemented in [gazebo::physics::MultiRayShape](#) (p. 555), [gazebo::physics::RayShape](#) (p. 721), [gazebo::physics::HeightmapShape](#) (p. 403), [gazebo::physics::CylinderShape](#) (p. 310), [gazebo::physics::MapShape](#) (p. 487), [gazebo::physics::PlaneShape](#) (p. 673), [gazebo::physics::BoxShape](#) (p. 165), [gazebo::physics::TrimeshShape](#) (p. 868), and [gazebo::physics::SphereShape](#) (p. 808).

10.177.4 Member Data Documentation

10.177.4.1 CollisionPtr gazebo::physics::Shape::collisionParent

Referenced by [gazebo::physics::ODEPlaneShape::CreatePlane\(\)](#), [gazebo::physics::BulletPlaneShape::CreatePlane\(\)](#), [gazebo::physics::ODEPlaneShape::SetAltitude\(\)](#), [gazebo::physics::ODESphereShape::SetRadius\(\)](#), [gazebo::physics::BulletSphereShape::SetRadius\(\)](#), [gazebo::physics::ODECylinderShape::SetSize\(\)](#), [gazebo::physics::ODEBoxShape::SetSize\(\)](#), [gazebo::physics::BulletBoxShape::SetSize\(\)](#), and [gazebo::physics::BulletCylinderShape::SetSize\(\)](#).

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

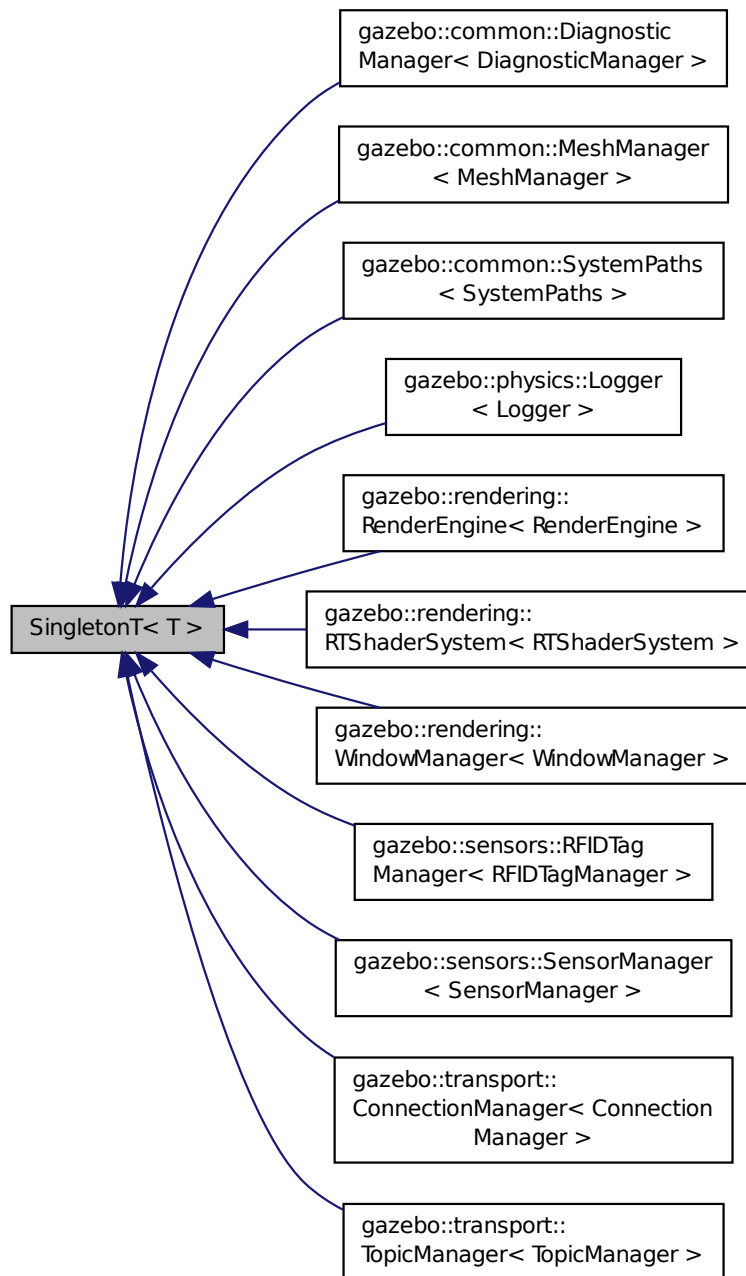
- [Shape.hh](#)

10.178 SingletonT< T > Class Template Reference

Singleton template class.

```
#include <SingletonT.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.

10.178.1 Detailed Description

```
template<class T>class SingletonT< T >
```

Singleton template class.

10.178.2 Constructor & Destructor Documentation

10.178.2.1 `template<class T> SingletonT< T >::SingletonT () [inline],[protected]`

Constructor.

10.178.2.2 `template<class T> virtual SingletonT< T >::~~SingletonT () [inline],[protected],[virtual]`

Destructor.

10.178.3 Member Function Documentation

10.178.3.1 `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**

10.179 gazebo::common::Skeleton Class Reference

A skeleton.

```
#include <Skeleton.hh>
```

Public Member Functions

- **Skeleton** ()
Constructor.
- **Skeleton** (SkeletonNode *_root)
Constructor.

- virtual `~Skeleton ()`
Destructor.
- void **AddAnimation** (**SkeletonAnimation** *_anim)
Add an animation.
- void **AddVertNodeWeight** (unsigned int _vertex, std::string _node, double _weight)
Add a new weight to a node (bone)
- **SkeletonAnimation** * **GetAnimation** (const unsigned int _i)
Find animation.
- **math::Matrix4** **GetBindShapeTransform** ()
Return bind pose skeletal transform.
- **SkeletonNode** * **GetNodeByHandle** (unsigned int _handle)
Find or create node with handle.
- **SkeletonNode** * **GetNodeById** (std::string _id)
Find node by index.
- **SkeletonNode** * **GetNodeByName** (std::string _name)
Find a node.
- **NodeMap** **GetNodes** ()
Get a copy or the node dictionary.
- unsigned int **GetNumAnimations** ()
Returns the number of animations.
- unsigned int **GetNumJoints** ()
Returns the number of joints.
- unsigned int **GetNumNodes** ()
Returns the node count.
- unsigned int **GetNumVertNodeWeights** (unsigned int _vertex)
Returns the number of bone weights for a vertex.
- **SkeletonNode** * **GetRootNode** ()
Return the root.
- std::pair< std::string, double > **GetVertNodeWeight** (unsigned int _v, unsigned int _i)
Weight of a bone for a vertex.
- void **PrintTransforms** ()
Outputs the transforms to std::err stream.
- void **Scale** (double _scale)
Scale all nodes, transforms and animation data.
- void **SetBindShapeTransform** (**math::Matrix4** _trans)
Set the bind pose skeletal transform.
- void **SetNumVertAttached** (unsigned int _vertices)
Resizes the raw node weight array.
- void **SetRootNode** (**SkeletonNode** *_node)
Change the root node.

Protected Member Functions

- void **BuildNodeMap** ()
Initializes the handle numbers for each node in the map using breath 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

10.179.1 Detailed Description

A skeleton.

10.179.2 Constructor & Destructor Documentation

10.179.2.1 gazebo::common::Skeleton::Skeleton ()

Constructor.

10.179.2.2 gazebo::common::Skeleton::Skeleton (**SkeletonNode** * *_root*)

Constructor.

Parameters

<i>in</i>	<i>_root</i>	node
-----------	--------------	------

10.179.2.3 virtual gazebo::common::Skeleton::~~Skeleton () [virtual]

Destructor.

10.179.3 Member Function Documentation

10.179.3.1 void gazebo::common::Skeleton::AddAnimation (**SkeletonAnimation** * *_anim*)

Add an animation.

The skeleton does not take ownership of the animation

Parameters

<i>_anim</i>	the animation
--------------	---------------

10.179.3.2 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)

10.179.3.3 void gazebo::common::Skeleton::BuildNodeMap () [protected]

Initializes the handle numbers for each node in the map using breath first traversal.

10.179.3.4 SkeletonAnimation* gazebo::common::Skeleton::GetAnimation (const unsigned int *_i*)

Find animation.

Parameters

	<i>_i</i>	the animation index
--	-----------	---------------------

Returns

the animation, or NULL if *_i* is out of bounds

10.179.3.5 math::Matrix4 gazebo::common::Skeleton::GetBindShapeTransform ()

Return bind pose skeletal transform.

Returns

a matrix

10.179.3.6 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

10.179.3.7 SkeletonNode* gazebo::common::Skeleton::GetNodeById (std::string *_id*)

Find node by index.

Parameters

<code>in</code>	<code>_id</code>	the index
-----------------	------------------	-----------

Returns

the node, or NULL if not found

10.179.3.8 `SkeletonNode*` `gazebo::common::Skeleton::GetNodeByName (std::string _name)`

Find a node.

Parameters

<code>in</code>	<code>_name</code>	the name of the node to look for
-----------------	--------------------	----------------------------------

Returns

the node, or NULL if not found

10.179.3.9 `NodeMap` `gazebo::common::Skeleton::GetNodes ()`

Get a copy of the node dictionary.

10.179.3.10 `unsigned int` `gazebo::common::Skeleton::GetNumAnimations ()`

Returns the number of animations.

Returns

the count

10.179.3.11 `unsigned int` `gazebo::common::Skeleton::GetNumJoints ()`

Returns the number of joints.

Returns

the count

10.179.3.12 `unsigned int` `gazebo::common::Skeleton::GetNumNodes ()`

Returns the node count.

Returns

the count

10.179.3.13 `unsigned int gazebo::common::Skeleton::GetNumVertNodeWeights (unsigned int _vertex)`

Returns the number of bone weights for a vertex.

Parameters

<code>in</code>	<i>the</i>	index of the vertex
-----------------	------------	---------------------

Returns

the count

10.179.3.14 `SkeletonNode* gazebo::common::Skeleton::GetRootNode ()`

Return the root.

Returns

the root

10.179.3.15 `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>_v</code>	the index of the vertex
<code>in</code>	<code>_i</code>	the index of the weight for that vertex

Returns

a pair containing the name of the node and the weight

10.179.3.16 `void gazebo::common::Skeleton::PrintTransforms ()`

Outputs the transforms to `std::err` stream.

10.179.3.17 `void gazebo::common::Skeleton::Scale (double _scale)`

Scale all nodes, transforms and animation data.

Parameters

<code>in</code>	<i>the</i>	scaling factor
-----------------	------------	----------------

10.179.3.18 `void gazebo::common::Skeleton::SetBindShapeTransform (math::Matrix4 _trans)`

Set the bind pose skeletal transform.

Parameters

in	<i>_trans</i>	the transform
----	---------------	---------------

10.179.3.19 void gazebo::common::Skeleton::SetNumVertAttached (unsigned int *_vertices*)

Resizes the raw node weight array.

Parameters

in	<i>_vertices</i>	the new size
----	------------------	--------------

10.179.3.20 void gazebo::common::Skeleton::SetRootNode (SkeletonNode * *_node*)

Change the root node.

Parameters

in	<i>_node</i>	the new node
----	--------------	--------------

10.179.4 Member Data Documentation

10.179.4.1 std::vector<SkeletonAnimation*> gazebo::common::Skeleton::anim [protected]

the array of animations

10.179.4.2 math::Matrix4 gazebo::common::Skeleton::bindShapeTransform [protected]

the bind pose skeletal transform

10.179.4.3 NodeMap gazebo::common::Skeleton::nodes [protected]

The dictionary of nodes, indexed by name.

10.179.4.4 RawNodeWeights gazebo::common::Skeleton::rawNW [protected]

the node weight table

10.179.4.5 SkeletonNode* gazebo::common::Skeleton::root [protected]

the root node

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

- **Skeleton.hh**

10.180 gazebo::common::SkeletonAnimation Class Reference

Skeleton (p. 784) 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

10.180.1 Detailed Description

Skeleton (p. 784) animation.

10.180.2 Constructor & Destructor Documentation

10.180.2.1 gazebo::common::SkeletonAnimation::SkeletonAnimation (const std::string & *_name*)

The Constructor.

Parameters

in	<i>_name</i>	the name of the animation
----	--------------	---------------------------

10.180.2.2 gazebo::common::SkeletonAnimation::~~SkeletonAnimation ()

The destructor.

Clears the list without destroying the animations

10.180.3 Member Function Documentation

10.180.3.1 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

10.180.3.2 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. 677)

10.180.3.3 double gazebo::common::SkeletonAnimation::GetLength () const

Returns the duration of the animations.

Returns

the duration in seconds

10.180.3.4 `std::string gazebo::common::SkeletonAnimation::GetName () const`

Returns the name.

Returns

the name

10.180.3.5 `unsigned int gazebo::common::SkeletonAnimation::GetNodeCount () const`

Returns the number of animation nodes.

Returns

the count

10.180.3.6 `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

<i>in</i>	<i>_node</i>	the name of the animation node
<i>in</i>	<i>_time</i>	the time
<i>in</i>	<i>_loop</i>	when true, the time is divided by the duration (see GetLength)

Returns

the transformation

10.180.3.7 `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

<i>in</i>	<i>_time</i>	the time
<i>in</i>	<i>_loop</i>	when true, the time is divided by the duration (see GetLength)

Returns

the transformation for every node

10.180.3.8 `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 <code>GetLength</code>)

10.180.3.9 `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

10.180.3.10 `void gazebo::common::SkeletonAnimation::Scale (const double _scale)`

Scales every animation in the animations list.

Parameters

in	<i>_scale</i>	the scaling factor
----	---------------	--------------------

10.180.3.11 `void gazebo::common::SkeletonAnimation::SetName (const std::string & _name)`

Changes the name.

Parameters

in	<i>_name</i>	the new name
----	--------------	--------------

10.180.4 Member Data Documentation

10.180.4.1 `std::map<std::string, NodeAnimation*> gazebo::common::SkeletonAnimation::animations` `[protected]`

a dictionary of node animations

10.180.4.2 double gazebo::common::SkeletonAnimation::length [protected]

the duration of the longest animation

10.180.4.3 std::string gazebo::common::SkeletonAnimation::name [protected]

the node name

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

- **SkeletonAnimation.hh**

10.181 gazebo::common::SkeletonNode Class Reference

A skeleton node.

```
#include <Skeleton.hh>
```

Public Types

- enum **SkeletonNodeType** { **NODE**, **JOINT** }
enumeration of node types

Public Member Functions

- **SkeletonNode** (**SkeletonNode** *_parent)
Constructor.
- **SkeletonNode** (**SkeletonNode** *_parent, std::string _name, std::string _id, **SkeletonNodeType** _type=**JOINT**)
Constructor.
- virtual ~**SkeletonNode** ()
Destructor.
- void **AddChild** (**SkeletonNode** *_child)
Add a new child.
- void **AddRawTransform** (**NodeTransform** _t)
Add a raw transform.
- **SkeletonNode** * **GetChild** (unsigned int _index)
Find a child by index.
- **SkeletonNode** * **GetChildById** (std::string _id)
Get child by string id.
- **SkeletonNode** * **GetChildByName** (std::string _name)
Get child by name.
- unsigned int **GetChildCount** ()
Returns the children count.
- unsigned int **GetHandle** ()
Get the handle index.
- std::string **GetId** ()
Returns the index.

- **math::Matrix4 GetInverseBindTransform ()**
Retrieve the inverse of the bind pose skeletal transform.
- **math::Matrix4 GetModelTransform ()**
Retrieve the model transform.
- **std::string GetName ()**
Returns the name.
- **unsigned int GetNumRawTrans ()**
Return the raw transformations count.
- **SkeletonNode * GetParent ()**
Returns the parent node.
- **NodeTransform GetRawTransform (unsigned int _i)**
Find a raw transformation.
- **std::vector< NodeTransform > GetRawTransforms ()**
Retrieve the raw transformations.
- **math::Matrix4 GetTransform ()**
Get transform relative to parent.
- **std::vector< NodeTransform > GetTransforms ()**
Returns a copy of the array of transformations.
- **bool IsJoint ()**
Is a joint query.
- **bool IsRootNode ()**
Queries wether a node has no parent parent.
- **void Reset (bool _resetChildren)**
Reset the transformation to the initial transformation.
- **void SetHandle (unsigned int _h)**
Assign a handle number.
- **void SetId (std::string _id)**
Change the id string.
- **void SetInitialTransform (math::Matrix4 _tras)**
Sets the initial transformation.
- **void SetInverseBindTransform (math::Matrix4 _invBM)**
Assign the inverse of the bind pose skeletal transform.
- **void SetModelTransform (math::Matrix4 _trans, bool _updateChildren=true)**
Set the model transformation.
- **void SetName (std::string _name)**
Change the name.
- **void SetParent (SkeletonNode *_parent)**
Set the parent node.
- **void SetTransform (math::Matrix4 _trans, bool _updateChildren=true)**
Set a transformation.
- **void SetType (SkeletonNodeType _type)**
Change the skeleton node type.
- **void UpdateChildrenTransforms ()**
Apply model transformations in order for each node in the tree.

Protected Attributes

- `std::vector< SkeletonNode * >` **children**
the children nodes
- `unsigned int` **handle**
handle index number
- `std::string` **id**
a string identifier
- `math::Matrix4` **initialTransform**
the initial transformation
- `math::Matrix4` **invBindTransform**
the inverse of the bind pose skeletal transform
- `math::Matrix4` **modelTransform**
the model transformation
- `std::string` **name**
the name of the skeletal node
- `SkeletonNode *` **parent**
the parent node
- `std::vector< NodeTransform >` **rawTransforms**
the raw transformation
- `math::Matrix4` **transform**
the transform
- `SkeletonNodeType` **type**
the type fo node

10.181.1 Detailed Description

A skeleton node.

10.181.2 Member Enumeration Documentation

10.181.2.1 enum gazebo::common::SkeletonNode::SkeletonNodeType

enumeration of node types

Enumerator:

NODE

JOINT

10.181.3 Constructor & Destructor Documentation

10.181.3.1 gazebo::common::SkeletonNode::SkeletonNode (`SkeletonNode * _parent`)

Constructor.

10.181.3.2 `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

10.181.3.3 `virtual gazebo::common::SkeletonNode::~~SkeletonNode () [virtual]`

Destructor.

10.181.4 Member Function Documentation

10.181.4.1 `void gazebo::common::SkeletonNode::AddChild (SkeletonNode * _child)`

Add a new child.

Parameters

in	<code>_child</code>	a child
----	---------------------	---------

10.181.4.2 `void gazebo::common::SkeletonNode::AddRawTransform (NodeTransform _t)`

Add a raw transform.

Parameters

in	<code>_t</code>	the transform
----	-----------------	---------------

10.181.4.3 `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

10.181.4.4 `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

10.181.4.5 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

10.181.4.6 unsigned int gazebo::common::SkeletonNode::GetChildCount ()

Returns the children count.

Returns

the count

10.181.4.7 unsigned int gazebo::common::SkeletonNode::GetHandle ()

Get the handle index.

Returns

the handle index

10.181.4.8 std::string gazebo::common::SkeletonNode::GetId ()

Returns the index.

Returns

the id string

10.181.4.9 math::Matrix4 gazebo::common::SkeletonNode::GetInverseBindTransform ()

Retrieve the inverse of the bind pose skeletal transform.

Returns

the transform

10.181.4.10 `math::Matrix4 gazebo::common::SkeletonNode::GetModelTransform ()`

Retrieve the model transform.

Returns

the transform

10.181.4.11 `std::string gazebo::common::SkeletonNode::GetName ()`

Returns the name.

Returns

the name

10.181.4.12 `unsigned int gazebo::common::SkeletonNode::GetNumRawTrans ()`

Return the raw transformations count.

Returns

the count

10.181.4.13 `SkeletonNode* gazebo::common::SkeletonNode::GetParent ()`

Returns the parent node.

Returns

the parent

10.181.4.14 `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

10.181.4.15 `std::vector<NodeTransform> gazebo::common::SkeletonNode::GetRawTransforms ()`

Retrieve the raw transformations.

Returns

an array of transformations

10.181.4.16 `math::Matrix4 gazebo::common::SkeletonNode::GetTransform ()`

Get transform relative to parent.

10.181.4.17 `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)

10.181.4.18 `bool gazebo::common::SkeletonNode::IsJoint ()`

Is a joint query.

Returns

true if the skeleton type is a joint, false otherwise

10.181.4.19 `bool gazebo::common::SkeletonNode::IsRootNode ()`

Queries whether a node has no parent parent.

Returns

true if the node has no parent, false otherwise

10.181.4.20 `void gazebo::common::SkeletonNode::Reset (bool _resetChildren)`

Reset the transformation to the initial transformation.

Parameters

<code>in</code>	<code>_resetChildren</code>	when true, performs the operation for every node in the tree
-----------------	-----------------------------	--

10.181.4.21 `void gazebo::common::SkeletonNode::SetHandle (unsigned int _h)`

Assign a handle number.

Parameters

<code>in</code>	<code>_h</code>	the handle
-----------------	-----------------	------------

10.181.4.22 `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
-----------------	-------------------------	-------------------

10.181.4.23 `void gazebo::common::SkeletonNode::SetInitialTransform (math::Matrix4 _tras)`

Sets the initial transformation.

Parameters

<code>in</code>	<code><i>_tras</i></code>	the transformation matrix
-----------------	---------------------------	---------------------------

10.181.4.24 `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
-----------------	----------------------------	---------------

10.181.4.25 `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

10.181.4.26 `void gazebo::common::SkeletonNode::SetName (std::string _name)`

Change the name.

Parameters

<code>in</code>	<code><i>_name</i></code>	the new name
-----------------	---------------------------	--------------

10.181.4.27 `void gazebo::common::SkeletonNode::SetParent (SkeletonNode * _parent)`

Set the parent node.

Parameters

<code>in</code>	<code><i>_parent</i></code>	the new parent
-----------------	-----------------------------	----------------

10.181.4.28 void gazebo::common::SkeletonNode::SetTransform (math::Matrix4 *_trans*, bool *_updateChildren = true*)

Set a transformation.

Parameters

in	<i>_trans</i>	the transformation
in	<i>_updateChildren</i>	when true the UpdateChildrenTransforms operation is performed

10.181.4.29 void gazebo::common::SkeletonNode::SetType (SkeletonNodeType *_type*)

Change the skeleton node type.

Parameters

in	<i>_type</i>	the new type
----	--------------	--------------

10.181.4.30 void gazebo::common::SkeletonNode::UpdateChildrenTransforms ()

Apply model transformations in order for each node in the tree.

10.181.5 Member Data Documentation

10.181.5.1 std::vector<SkeletonNode*> gazebo::common::SkeletonNode::children [protected]

the children nodes

10.181.5.2 unsigned int gazebo::common::SkeletonNode::handle [protected]

handle index number

10.181.5.3 std::string gazebo::common::SkeletonNode::id [protected]

a string identifier

10.181.5.4 math::Matrix4 gazebo::common::SkeletonNode::initialTransform [protected]

the initial transformation

10.181.5.5 math::Matrix4 gazebo::common::SkeletonNode::invBindTransform [protected]

the inverse of the bind pose skeletal transform

10.181.5.6 math::Matrix4 gazebo::common::SkeletonNode::modelTransform [protected]

the model transformation

10.181.5.7 `std::string gazebo::common::SkeletonNode::name` [protected]

the name of the skeletal node

10.181.5.8 `SkeletonNode* gazebo::common::SkeletonNode::parent` [protected]

the parent node

10.181.5.9 `std::vector<NodeTransform> gazebo::common::SkeletonNode::rawTransforms` [protected]

the raw transformation

10.181.5.10 `math::Matrix4 gazebo::common::SkeletonNode::transform` [protected]

the transform

10.181.5.11 `SkeletonNodeType gazebo::common::SkeletonNode::type` [protected]

the type fo node

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

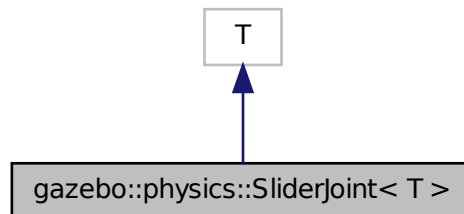
- [Skeleton.hh](#)

10.182 `gazebo::physics::SliderJoint< T >` Class Template Reference

A slider joint.

```
#include <SliderJoint.hh>
```

Inheritance diagram for `gazebo::physics::SliderJoint< T >`:



Public Member Functions

- `SliderJoint (BasePtr _parent)`

Constructor.

- virtual `~SliderJoint ()`

Destructor.

- virtual `math::Vector3 GetAnchor (int) const`

Get the anchor.

- virtual `void SetAnchor (int, const math::Vector3 &_anchor)`

Set the anchor.

Protected Member Functions

- virtual `void Load (sdf::ElementPtr _sdf)`

*Load a **SliderJoint** (p. 804).*

Protected Attributes

- `math::Vector3 fakeAnchor`

10.182.1 Detailed Description

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

A slider joint.

10.182.2 Constructor & Destructor Documentation

10.182.2.1 `template<class T> gazebo::physics::SliderJoint< T >::SliderJoint (BasePtr _parent) [inline]`

Constructor.

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

Destructor.

10.182.3 Member Function Documentation

10.182.3.1 `template<class T> virtual math::Vector3 gazebo::physics::SliderJoint< T >::GetAnchor (int) const [inline], [virtual]`

Get the anchor.

10.182.3.2 `template<class T> virtual void gazebo::physics::SliderJoint< T >::Load (sdf::ElementPtr _sdf) [inline], [protected], [virtual]`

Load a **SliderJoint** (p. 804).

Reimplemented in `gazebo::physics::BulletSliderJoint` (p. 220), and `gazebo::physics::ODESliderJoint` (p. 628).

10.182.3.3 `template<class T> virtual void gazebo::physics::SliderJoint< T >::SetAnchor (int , const math::Vector3 & .anchor) [inline],[virtual]`

Set the anchor.

10.182.4 Member Data Documentation

10.182.4.1 `template<class T> math::Vector3 gazebo::physics::SliderJoint< T >::fakeAnchor [protected]`

Referenced by `gazebo::physics::SliderJoint< BulletJoint >::GetAnchor()`, and `gazebo::physics::SliderJoint< BulletJoint >::SetAnchor()`.

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

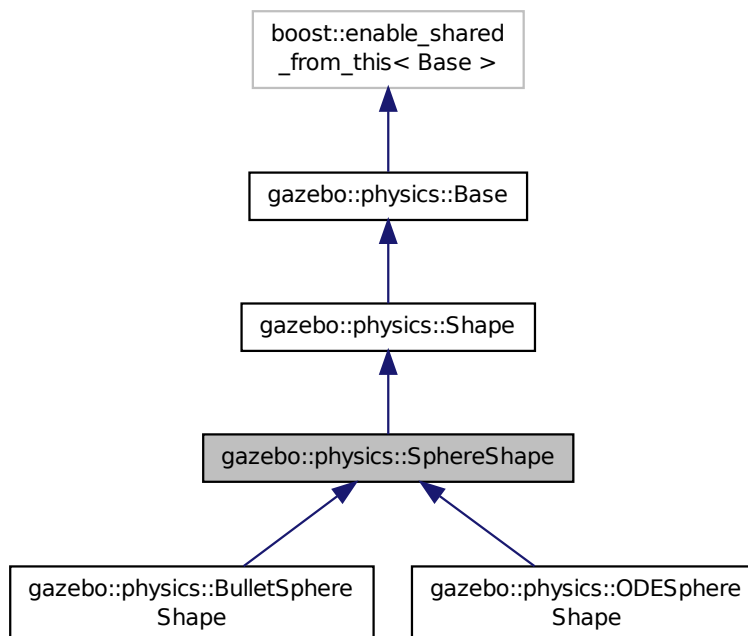
- **SliderJoint.hh**

10.183 gazebo::physics::SphereShape Class Reference

Sphere collision.

```
#include <SphereShape.hh>
```

Inheritance diagram for `gazebo::physics::SphereShape`:



Public Member Functions

- **SphereShape** (**CollisionPtr** parent)

Constructor.

- virtual **~SphereShape** ()

Destructor.

- virtual void **FillMsg** (msgs::Geometry &_msg)

- virtual void **GetInertial** (double _mass, **InertialPtr** _inertial) const

Get inertial for a shape.

- virtual double **GetMass** (double _density) const

Get the mass of a shape.

- double **GetRadius** () const

- virtual void **Init** ()

Initialize the sphere.

- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

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

Set the size.

Additional Inherited Members

10.183.1 Detailed Description

Sphere collision.

10.183.2 Constructor & Destructor Documentation

10.183.2.1 gazebo::physics::SphereShape::SphereShape (**CollisionPtr** parent)

Constructor.

10.183.2.2 virtual gazebo::physics::SphereShape::~~SphereShape () [virtual]

Destructor.

10.183.3 Member Function Documentation

10.183.3.1 virtual void gazebo::physics::SphereShape::FillMsg (msgs::Geometry & .msg) [virtual]

Implements **gazebo::physics::Shape** (p. 781).

10.183.3.2 virtual void gazebo::physics::SphereShape::GetInertial (double .mass, **InertialPtr** .inertial) const [virtual]

Get inertial for a shape.

Reimplemented from **gazebo::physics::Shape** (p. 781).

10.183.3.3 `virtual double gazebo::physics::SphereShape::GetMass (double _density) const` [virtual]

Get the mass of a shape.

Reimplemented from `gazebo::physics::Shape` (p. 781).

10.183.3.4 `double gazebo::physics::SphereShape::GetRadius () const`

10.183.3.5 `virtual void gazebo::physics::SphereShape::Init ()` [virtual]

Initialize the sphere.

Implements `gazebo::physics::Shape` (p. 781).

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

Implements `gazebo::physics::Shape` (p. 782).

10.183.3.7 `virtual void gazebo::physics::SphereShape::SetRadius (double _radius)` [virtual]

Set the size.

Reimplemented in `gazebo::physics::ODESphereShape` (p. 631).

Referenced by `gazebo::physics::ODESphereShape::SetRadius()`, and `gazebo::physics::BulletSphereShape::SetRadius()`.

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

- `SphereShape.hh`

10.184 gazebo::math::Spline Class Reference

Splines.

```
#include <Spline.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.

10.184.1 Detailed Description

Splines.

10.184.2 Constructor & Destructor Documentation

10.184.2.1 gazebo::math::Spline::Spline ()

constructor

10.184.2.2 gazebo::math::Spline::~~Spline ()

destructor

10.184.3 Member Function Documentation

10.184.3.1 void gazebo::math::Spline::AddPoint (const Vector3 & _pt)

Adds a control point to the end of the spline.

Parameters

in	_pt	point to add
----	-----	--------------

10.184.3.2 void gazebo::math::Spline::Clear ()

Clears all the points in the spline.

10.184.3.3 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

10.184.3.4 unsigned int gazebo::math::Spline::GetPointCount () const

Gets the number of control points in the spline.

Returns

the count

10.184.3.5 Vector3 gazebo::math::Spline::GetTangent (unsigned int _index) const

Get the tangent value for a point.

Parameters

in	_index	the control point index
----	--------	-------------------------

10.184.3.6 double gazebo::math::Spline::GetTension () const

Get the tension value.

Returns

The value of the tension, which is between 0.0 and 1.0

10.184.3.7 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)
-----------	-----------	--------------------------

10.184.3.8 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

10.184.3.9 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.

10.184.3.10 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.
-----------	------------------	--

10.184.3.11 `void gazebo::math::Spline::SetTension (double _t)`

Set the tension parameter.

A value of 0 = Catmull-Rom spline.

Parameters

in	_t	Tension value between 0.0 and 1.0
----	----	-----------------------------------

10.184.3.12 `void gazebo::math::Spline::UpdatePoint (unsigned int _index, const Vector3 & _value)`

Updates a single point in the spline.

Remarks

an error to the error stream is printed when the index is out of bounds

Parameters

in	_index	the control point index
in	_value	the new position

10.184.4 Member Data Documentation

10.184.4.1 `bool gazebo::math::Spline::autoCalc` [protected]

when true, the tangents are recalculated when the control point change

10.184.4.2 `Matrix4 gazebo::math::Spline::coeffs` [protected]

Matrix of coefficients.

10.184.4.3 `std::vector<Vector3> gazebo::math::Spline::points` [protected]

control points

10.184.4.4 `std::vector<Vector3> gazebo::math::Spline::tangents` [protected]

tangents

10.184.4.5 `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:

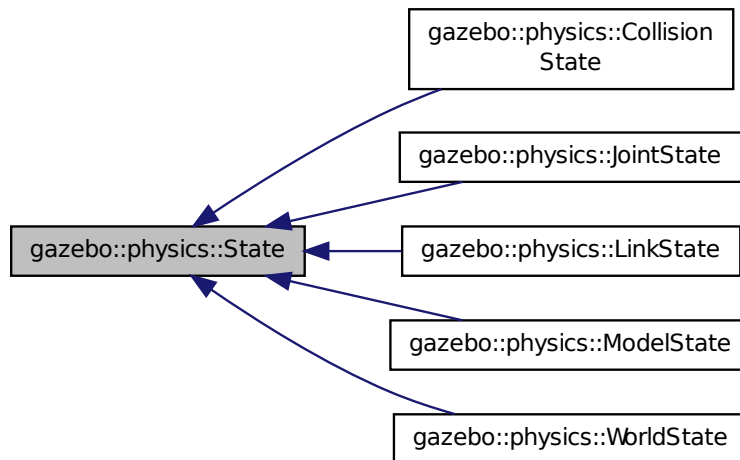
- **Spline.hh**

10.185 gazebo::physics::State Class Reference

State (p. 813) of an entity.

```
#include <physics/State.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. 813).*
- **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** (sdf::ElementPtr _elem)=0
Load state from SDF element.

Protected Attributes

- `std::string name`
Name associated with this **State** (p. 813).
- `common::Time realTime`
- `common::Time simTime`
- `common::Time wallTime`
Times for the state data.

10.185.1 Detailed Description

State (p. 813) of an entity.

This is the base class for all **State** (p. 813) information.

10.185.2 Constructor & Destructor Documentation

10.185.2.1 `gazebo::physics::State::State ()`

Default constructor.

10.185.2.2 `gazebo::physics::State::State (const std::string & _name, const common::Time & _realTime, const common::Time & _simTime)`

Constructor.

Construct a **State** (p. 813) object using some basic information.

Parameters

<code>_name</code>	Name associated with the State (p. 813) information. This is typically the name of an Entity (p. 338). <code>_realTime</code> Clock time since simulation started.
<code>_simTime</code>	Simulation time associated with this State (p. 813) info.

10.185.2.3 `virtual gazebo::physics::State::~~State () [virtual]`

Destructor.

10.185.3 Member Function Documentation

10.185.3.1 `std::string gazebo::physics::State::GetName () const`

Get the name associated with this **State** (p. 813).

Returns

Name associated with this state information. Typically a name of an **Entity** (p. 338).

10.185.3.2 `common::Time gazebo::physics::State::GetRealTime () const`

Get the real time when this state was generated.

Returns

Clock time since simulation was stated.

10.185.3.3 `common::Time gazebo::physics::State::GetSimTime () const`

Get the sim time when this state was generated.

Returns

Simulation time when the data was recorded.

10.185.3.4 `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. 813) data was recorded.

10.185.3.5 `virtual void gazebo::physics::State::Load (sdf::ElementPtr _elem) [pure virtual]`

Load state from SDF element.

Populates the **State** (p. 813) information from data stored in an SDF::Element

Parameters

<code>_elem</code>	Pointer to the SDF::Element
--------------------	-----------------------------

Implemented in **gazebo::physics::LinkState** (p. 477), **gazebo::physics::ModelState** (p. 539), **gazebo::physics::WorldState** (p. 969), **gazebo::physics::CollisionState** (p. 274), and **gazebo::physics::JointState** (p. 443).

10.185.4 Member Data Documentation

10.185.4.1 `std::string gazebo::physics::State::name [protected]`

Name associated with this **State** (p. 813).

10.185.4.2 `common::Time gazebo::physics::State::realTime [protected]`

10.185.4.3 `common::Time gazebo::physics::State::simTime [protected]`

10.185.4.4 `common::Time gazebo::physics::State::wallTime` [protected]

Times for the state data.

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

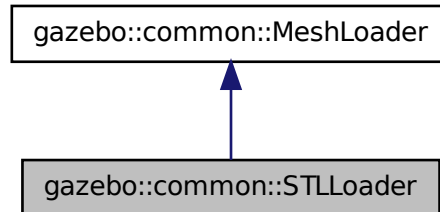
- **State.hh**

10.186 gazebo::common::STLLoader Class Reference

Class used to load STL mesh files.

```
#include <STLLoader.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.

10.186.1 Detailed Description

Class used to load STL mesh files.

10.186.2 Constructor & Destructor Documentation

10.186.2.1 `gazebo::common::STLLoader::STLLoader ()`

Constructor.

10.186.2.2 virtual gazebo::common::STLLoader::~~STLLoader () [virtual]

Destructor.

10.186.3 Member Function Documentation

10.186.3.1 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. 515).

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

- STLLoader.hh

10.187 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 maximum X, Y, Z values.
- unsigned int **GetMaxIndex** () const
Get the highest index value.
- **math::Vector3** **GetMin** () const
Get the minimum X, Y, Z values.
- **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.

10.187.1 Detailed Description

A child mesh.

10.187.2 Member Enumeration Documentation

10.187.2.1 enum gazebo::common::SubMesh::PrimitiveType

An enumeration of the geometric mesh primitives.

Enumerator:

POINTS
LINES
LINESTRIPS
TRIANGLES
TRIFANS
TRISTRIPS

10.187.3 Constructor & Destructor Documentation

10.187.3.1 gazebo::common::SubMesh::SubMesh ()

Constructor.

10.187.3.2 `virtual gazebo::common::SubMesh::~~SubMesh () [virtual]`

Destructor.

10.187.4 Member Function Documentation

10.187.4.1 `void gazebo::common::SubMesh::AddIndex (unsigned int _i)`

Add an index to the mesh.

Parameters

<code>in</code>	<code><i>_i</i></code>	the new vertex index
-----------------	------------------------	----------------------

10.187.4.2 `void gazebo::common::SubMesh::AddNodeAssignment (unsigned int _vertex, unsigned int _node, float _weight)`

Add a vertex - skeleton node assignment.

Parameters

<code>in</code>	<code><i>_vertex</i></code>	the vertex index
<code>in</code>	<code><i>_node</i></code>	the node index
<code>in</code>	<code><i>_weight</i></code>	the weight (between 0 and 1)

10.187.4.3 `void gazebo::common::SubMesh::AddNormal (const math::Vector3 & _n)`

Add a normal to the mesh.

Parameters

<code>in</code>	<code><i>_n</i></code>	the normal
-----------------	------------------------	------------

10.187.4.4 `void gazebo::common::SubMesh::AddNormal (double _x, double _y, double _z)`

Add a normal to the mesh.

Parameters

<code>in</code>	<code><i>_x</i></code>	position along x
<code>in</code>	<code><i>_y</i></code>	position along y
<code>in</code>	<code><i>_z</i></code>	position along z

10.187.4.5 `void gazebo::common::SubMesh::AddTexCoord (double _u, double _v)`

Add a texture coord to the mesh.

Parameters

<code>in</code>	<code><i>_u</i></code>	position along u
<code>in</code>	<code><i>_v</i></code>	position along v

10.187.4.6 void gazebo::common::SubMesh::AddVertex (const math::Vector3 & _v)

Add a vertex to the mesh.

Parameters

in	_v	the new position
----	----	------------------

10.187.4.7 void gazebo::common::SubMesh::AddVertex (double _x, double _y, double _z)

Add a vertex to the mesh.

Parameters

in	_x	position along x
in	_y	position along y
in	_z	position along z

10.187.4.8 void gazebo::common::SubMesh::CopyNormals (const std::vector< math::Vector3 > & _norms)

Copy normals from a vector.

Parameters

in	_norms	to copy from
----	--------	--------------

10.187.4.9 void gazebo::common::SubMesh::CopyVertices (const std::vector< math::Vector3 > & _verts)

Copy vertices from a vector.

Parameters

in	_verts	the vertices to copy from
----	--------	---------------------------

10.187.4.10 void gazebo::common::SubMesh::FillArrays (float ** _vertArr, int ** _indArr) const

Put all the data into flat arrays.

Parameters

in	_vertArr	
in	_indArr	

10.187.4.11 void gazebo::common::SubMesh::GenSphericalTexCoord (const math::Vector3 & _center)

Generate texture coordinates using spherical projection from center.

Parameters

in	<i>_center</i>	
----	----------------	--

10.187.4.12 unsigned int gazebo::common::SubMesh::GetIndex (unsigned int *i*) const

Get an index.

Parameters

in	<i>_i</i>	
----	-----------	--

10.187.4.13 unsigned int gazebo::common::SubMesh::GetIndexCount () const

Return the number of indicies.

10.187.4.14 unsigned int gazebo::common::SubMesh::GetMaterialIndex () const

Get the material index.

10.187.4.15 math::Vector3 gazebo::common::SubMesh::GetMax () const

Get the maximun X, Y, Z values.

Returns

10.187.4.16 unsigned int gazebo::common::SubMesh::GetMaxIndex () const

Get the highest index value.

10.187.4.17 math::Vector3 gazebo::common::SubMesh::GetMin () const

Get the minimum X, Y, Z values.

Returns

10.187.4.18 **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
----	-----------	-----------------------------

10.187.4.19 `unsigned int gazebo::common::SubMesh::GetNodeAssignmentsCount () const`

Return the number of vertex - skeleton node assignments.

10.187.4.20 `math::Vector3 gazebo::common::SubMesh::GetNormal (unsigned int i) const`

Get a normal.

Parameters

<code><i>i</i></code>	<code><i>i</i></code>	the normal index
-----------------------	-----------------------	------------------

Returns

the orientation of the normal, or throws an exception

10.187.4.21 `unsigned int gazebo::common::SubMesh::GetNormalCount () const`

Return the number of normals.

10.187.4.22 `PrimitiveType gazebo::common::SubMesh::GetPrimitiveType () const`

Get the primitive type.

Returns

the primitive type

10.187.4.23 `math::Vector2d gazebo::common::SubMesh::GetTexCoord (unsigned int i) const`

Get a tex coord.

Parameters

<code><i>i</i></code>	<code><i>i</i></code>	the texture index
-----------------------	-----------------------	-------------------

Returns

the texture coordinates

10.187.4.24 `unsigned int gazebo::common::SubMesh::GetTexCoordCount () const`

Return the number of texture coordinates.

10.187.4.25 `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

10.187.4.26 unsigned int gazebo::common::SubMesh::GetVertexCount () const

Return the number of vertices.

10.187.4.27 unsigned int gazebo::common::SubMesh::GetVertexIndex (const math::Vector3 & *_v*) const

Get the index of the vertex.

Parameters

in	<i>_v</i>	
----	-----------	--

10.187.4.28 bool gazebo::common::SubMesh::HasVertex (const math::Vector3 & *_v*) const

Return true if this submesh has the vertex.

Parameters

in	<i>_v</i>	
----	-----------	--

10.187.4.29 void gazebo::common::SubMesh::RecalculateNormals ()

Recalculate all the normals.

10.187.4.30 void gazebo::common::SubMesh::Scale (double *_factor*)

Scale all vertices by *_factor*.

Parameters

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

10.187.4.31 void gazebo::common::SubMesh::SetIndexCount (unsigned int *_count*)

Resize the index array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

10.187.4.32 void gazebo::common::SubMesh::SetMaterialIndex (unsigned int *_index*)

Set the material index.

Relates to the parent mesh material list

Parameters

in	<i>_index</i>	
----	---------------	--

10.187.4.33 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

10.187.4.34 void gazebo::common::SubMesh::SetNormalCount (unsigned int *_count*)

Resize the normal array.

Parameters

in	<i>_count</i>	the new size of the array
----	---------------	---------------------------

10.187.4.35 void gazebo::common::SubMesh::SetPrimitiveType (PrimitiveType *_type*)

Set the primitive type.

Parameters

in	<i>_type</i>	the type
----	--------------	----------

10.187.4.36 void gazebo::common::SubMesh::SetSubMeshCenter (math::Vector3 *_center*)

Reset mesh center to geometric center.

Parameters

in	<i>_center</i>	
----	----------------	--

10.187.4.37 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>	

10.187.4.38 void gazebo::common::SubMesh::SetTexCoordCount (unsigned int *_count*)

Resize the texture coordinate array.

Parameters

in	<i>_count</i>	
----	---------------	--

10.187.4.39 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

10.187.4.40 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**

10.188 gazebo::transport::SubscribeOptions Class Reference

Options for a subscription.

```
#include <SubscribeOptions.hh>
```

Public Member Functions

- **SubscribeOptions** ()
- bool **GetLatching** () const
- std::string **GetMsgType** () const
- **NodePtr** **GetNode** () const
- std::string **GetTopic** () const
- template<class M >
void **Init** (const std::string & *_topic*, **NodePtr** *_node*, bool *_latching*)

10.188.1 Detailed Description

Options for a subscription.

10.188.2 Constructor & Destructor Documentation

10.188.2.1 `gazebo::transport::SubscribeOptions::SubscribeOptions ()` `[inline]`

10.188.3 Member Function Documentation

10.188.3.1 `bool gazebo::transport::SubscribeOptions::GetLatching () const` `[inline]`

10.188.3.2 `std::string gazebo::transport::SubscribeOptions::GetMsgType () const` `[inline]`

10.188.3.3 `NodePtr gazebo::transport::SubscribeOptions::GetNode () const` `[inline]`

10.188.3.4 `std::string gazebo::transport::SubscribeOptions::GetTopic () const` `[inline]`

10.188.3.5 `template<class M > void gazebo::transport::SubscribeOptions::Init (const std::string & _topic, NodePtr _node, bool _latching)` `[inline]`

References `gzthrow`, and `NULL`.

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

- **SubscribeOptions.hh**

10.189 gazebo::transport::Subscriber Class Reference

A subscriber to a topic.

```
#include <Subscriber.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.

10.189.1 Detailed Description

A subscriber to a topic.

10.189.2 Constructor & Destructor Documentation

10.189.2.1 `gazebo::transport::Subscriber::Subscriber (const std::string & topic, NodePtr _node)`

Constructor.

10.189.2.2 `virtual gazebo::transport::Subscriber::~Subscriber () [virtual]`

Destructor.

10.189.3 Member Function Documentation

10.189.3.1 `std::string gazebo::transport::Subscriber::GetTopic () const`

Get the topic name.

10.189.3.2 `void gazebo::transport::Subscriber::Unsubscribe () const`

Unsubscribe from the topic.

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

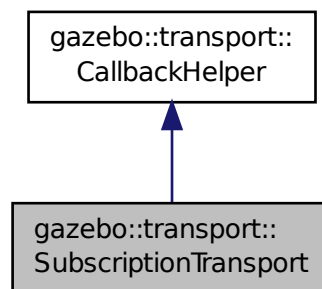
- **Subscriber.hh**

10.190 gazebo::transport::SubscriptionTransport Class Reference

Handles sending data over the wire to remote subscribers.

```
#include <SubscriptionTransport.hh>
```

Inheritance diagram for `gazebo::transport::SubscriptionTransport`:



Public Member Functions

- **SubscriptionTransport** ()
Constructor.
- virtual **~SubscriptionTransport** ()
Destructor.
- const **ConnectionPtr** & **GetConnection** () const
Get the connection.
- 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
Return true if the callback is local, false if the callback is tied to a remote connection.

Additional Inherited Members

10.190.1 Detailed Description

Handles sending data over the wire to remote subscribers.

10.190.2 Constructor & Destructor Documentation

10.190.2.1 gazebo::transport::SubscriptionTransport::SubscriptionTransport ()

Constructor.

10.190.2.2 virtual gazebo::transport::SubscriptionTransport::~~SubscriptionTransport () [virtual]

Destructor.

10.190.3 Member Function Documentation

10.190.3.1 const ConnectionPtr& gazebo::transport::SubscriptionTransport::GetConnection () const

Get the connection.

10.190.3.2 virtual bool gazebo::transport::SubscriptionTransport::HandleData (const std::string & newdata) [virtual]

Output a message to a connection.

Implements **gazebo::transport::CallbackHelper** (p. 231).

10.190.3.3 void gazebo::transport::SubscriptionTransport::Init (const ConnectionPtr & conn, bool _latching)

Initialize the publication link.

10.190.3.4 `virtual bool gazebo::transport::SubscriptionTransport::IsLocal() const [virtual]`

Return true if the callback is local, false if the callback is tied to a remote connection.

Implements `gazebo::transport::CallbackHelper` (p. 231).

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

- `SubscriptionTransport.hh`

10.191 gazebo::physics::SurfaceParams Class Reference

`SurfaceParams` (p. 830) defines various Surface contact parameters.

```
#include <SurfaceParams.hh>
```

Public Member Functions

- `SurfaceParams()`
Constructor.
- `virtual ~SurfaceParams()`
Constructor.
- `void FillSurfaceMsg(msgs::Surface &_msg)`
Fill in a surface message.
- `virtual void Load(sdf::ElementPtr _sdf)`
Load the contact params.
- `virtual void ProcessMsg(const msgs::Surface &_msg)`

Public Attributes

- `double bounce`
bounce restitution coefficient [0,1], with 0 being inelastic, and 1 being perfectly elastic.
- `double bounceThreshold`
minimum contact velocity for bounce to take effect, otherwise the collision is treated as an inelastic collision.
- `double cfm`
Constraint Force Mixing parameter.
- `double erp`
Error Reduction Parameter.
- `math::Vector3 fdir1`
Primary friction direction for dry friction coefficient (`SurfaceParams::mu1` (p. 833)) of the friction pyramid.
- `double kd`
spring damping constant equivalents of a contact as a function of `SurfaceParams::cfm` (p. 832) and `SurfaceParams::erp` (p. 832).
- `double kp`
spring constant equivalents of a contact as a function of `SurfaceParams::cfm` (p. 832) and `SurfaceParams::erp` (p. 832).
- `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.

10.191.1 Detailed Description

SurfaceParams (p. 830) defines various Surface contact parameters.

These parameters defines the properties of a **physics::Contact** (p. 296) constraint.

10.191.2 Constructor & Destructor Documentation

10.191.2.1 gazebo::physics::SurfaceParams::SurfaceParams ()

Constructor.

10.191.2.2 virtual gazebo::physics::SurfaceParams::~~SurfaceParams () [virtual]

Constructor.

10.191.3 Member Function Documentation

10.191.3.1 void gazebo::physics::SurfaceParams::FillSurfaceMsg (msgs::Surface & _msg)

Fill in a surface message.

10.191.3.2 virtual void gazebo::physics::SurfaceParams::Load (sdf::ElementPtr _sdf) [virtual]

Load the contact params.

10.191.3.3 virtual void gazebo::physics::SurfaceParams::ProcessMsg (const msgs::Surface & _msg) [virtual]

10.191.4 Member Data Documentation

10.191.4.1 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

10.191.4.2 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

10.191.4.3 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.

10.191.4.4 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.

10.191.4.5 math::Vector3 gazebo::physics::SurfaceParams::fdir1

Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 833)) 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

10.191.4.6 double gazebo::physics::SurfaceParams::kd

spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 832) and **SurfaceParams::erp** (p. 832).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.191.4.7 double gazebo::physics::SurfaceParams::kp

spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 832) and **SurfaceParams::erp** (p. 832).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.191.4.8 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)

10.191.4.9 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)

10.191.4.10 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

10.191.4.11 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

10.191.4.12 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

10.191.4.13 double gazebo::physics::SurfaceParams::slip2

Artificial contact slip in the secondary friction dirction.

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:

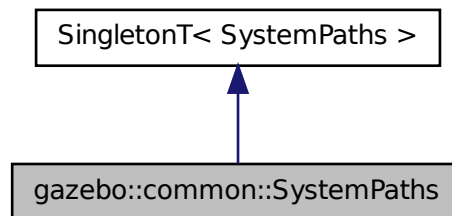
- **SurfaceParams.hh**

10.192 gazebo::common::SystemPaths Class Reference

Functions to handle getting system paths, keeps track of:

```
#include <SystemPaths.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. 837)*
- bool **modelPathsFromEnv**
 - if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 837)*
- bool **ogrePathsFromEnv**
 - if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 837)*
- bool **pluginPathsFromEnv**
 - if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 837)*

Additional Inherited Members

10.192.1 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. 118) RenderSystem_GL.so et. al.
- SystemPaths::pluginPaths - plugin library paths for common::WorldPlugin

10.192.2 Member Function Documentation

10.192.2.1 void gazebo::common::SystemPaths::AddGazeboPaths (const std::string & _path)

Add colon delimited paths to Gazebo install.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.192.2.2 void gazebo::common::SystemPaths::AddOgrePaths (const std::string & *_path*)

Add colon delimited paths to ogre install.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.192.2.3 void gazebo::common::SystemPaths::AddPluginPaths (const std::string & *_path*)

Add colon delimited paths to plugins.

Parameters

in	<i>_path</i>	the directory to add
----	--------------	----------------------

10.192.2.4 void gazebo::common::SystemPaths::AddSearchPathSuffix (const std::string & *_suffix*)

add *_suffix* to the list of path search suffixes

10.192.2.5 void gazebo::common::SystemPaths::ClearGazeboPaths ()

clear out SystemPaths::gazeboPaths

10.192.2.6 void gazebo::common::SystemPaths::ClearOgrePaths ()

clear out SystemPaths::ogrePaths

10.192.2.7 void gazebo::common::SystemPaths::ClearPluginPaths ()

clear out SystemPaths::pluginPaths

10.192.2.8 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.

10.192.2.9 `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
-----------------	-------------------	---------------------------------

10.192.2.10 `const std::list<std::string>& gazebo::common::SystemPaths::GetGazeboPaths ()`

Get the gazebo install paths.

Returns

a list of paths

10.192.2.11 `std::string gazebo::common::SystemPaths::GetLogPath () const`

Get the log path.

Returns

the path

10.192.2.12 `const std::list<std::string>& gazebo::common::SystemPaths::GetModelPaths ()`

Get the model paths.

Returns

a list of paths

10.192.2.13 `const std::list<std::string>& gazebo::common::SystemPaths::GetOgrePaths ()`

Get the ogre install paths.

Returns

a list of paths

10.192.2.14 `const std::list<std::string>& gazebo::common::SystemPaths::GetPluginPaths ()`

Get the plugin paths.

Returns

a list of paths

10.192.2.15 `std::string gazebo::common::SystemPaths::GetWorldPathExtension ()`

Returns the world path extension.

Returns

Right now, it just returns "/worlds"

10.192.3 Member Data Documentation

10.192.3.1 `bool gazebo::common::SystemPaths::gazeboPathsFromEnv`

if true, call `UpdateGazeboPaths()` within `GetGazeboPaths()` (p. 837)

10.192.3.2 `bool gazebo::common::SystemPaths::modelPathsFromEnv`

if true, call `UpdateGazeboPaths()` within `GetGazeboPaths()` (p. 837)

10.192.3.3 `bool gazebo::common::SystemPaths::ogrePathsFromEnv`

if true, call `UpdateOgrePaths()` within `GetOgrePaths()` (p. 837)

10.192.3.4 `bool gazebo::common::SystemPaths::pluginPathsFromEnv`

if true, call `UpdatePluginPaths()` within `GetPluginPaths()` (p. 837)

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

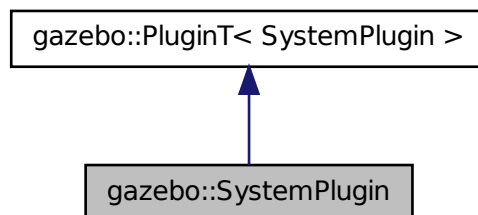
- `SystemPaths.hh`

10.193 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

10.193.1 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?

10.193.2 Constructor & Destructor Documentation

10.193.2.1 gazebo::SystemPlugin::SystemPlugin () [inline]

Constructor.

References `gazebo::SYSTEM_PLUGIN`, and `gazebo::PluginT< SystemPlugin >::type`.

10.193.2.2 virtual gazebo::SystemPlugin::~~SystemPlugin () [inline],[virtual]

Destructor.

10.193.3 Member Function Documentation

10.193.3.1 virtual void gazebo::SystemPlugin::Init () [inline],[virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.193.3.2 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.

10.193.3.3 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**

10.194 gazebo::common::Time Class Reference

A **Time** (p. 840) class, can be used to hold wall- or sim-time.

```
#include <Time.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.

10.194.1 Detailed Description

A **Time** (p. 840) class, can be used to hold wall- or sim-time.

stored as sec and nano-sec.

10.194.2 Constructor & Destructor Documentation

10.194.2.1 gazebo::common::Time::Time ()

Constructors.

10.194.2.2 gazebo::common::Time::Time (const Time & *time*)

Copy constructor.

Parameters

<i>in</i>	<i>time</i>	Time (p. 840) to copy
-----------	-------------	------------------------------

10.194.2.3 gazebo::common::Time::Time (const struct timeval & *tv*)

Constructor.

Parameters

<i>in</i>	<i>tv</i>	Time (p. 840) to initialize to
-----------	-----------	---------------------------------------

10.194.2.4 gazebo::common::Time::Time (const struct timespec & *tv*)

Constructor.

Parameters

<i>in</i>	<i>tv</i>	Time (p. 840) to initialize to
-----------	-----------	---------------------------------------

10.194.2.5 gazebo::common::Time::Time (int32_t *_sec*, int32_t *_nsec*)

Constructor.

Parameters

<i>in</i>	<i>sec</i>	Seconds
<i>in</i>	<i>nsec</i>	Microseconds

10.194.2.6 gazebo::common::Time::Time (double *time*)

Constructor.

Parameters

<i>in</i>	<i>time</i>	Time (p. 840) in double format sec.nsec
-----------	-------------	--

10.194.2.7 virtual gazebo::common::Time::~~Time () [virtual]

Destructor.

10.194.3 Member Function Documentation

10.194.3.1 `double gazebo::common::Time::Double () const`

Get the time as a double.

Returns

Time (p. 840) as a double in seconds

10.194.3.2 `float gazebo::common::Time::Float () const`

Get the time as a float.

Returns

Time (p. 840) as a float in seconds

10.194.3.3 `static const Time& gazebo::common::Time::GetWallTime () [static]`

Get the wall time.

Returns

the current time

10.194.3.4 `static double gazebo::common::Time::MicToNano (double _ms) [inline],[static]`

Convert microseconds to nanoseconds.

Parameters

<code><i>_ms</i></code>	microseconds
-------------------------	--------------

Returns

nanoseconds

10.194.3.5 `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

10.194.3.6 `static Time gazebo::common::Time::MSleep (unsigned int _ms) [static]`

Millisecond sleep.

Parameters

<code>in</code>	<code><i>_ms</i></code>	milliseconds
-----------------	-------------------------	--------------

10.194.3.7 `static Time gazebo::common::Time::NSleep (unsigned int _ns) [static]`

Nano sleep.

Parameters

<code>in</code>	<code><i>_ns</i></code>	nanoseconds
-----------------	-------------------------	-------------

10.194.3.8 `static Time gazebo::common::Time::NSleep (Time _time) [static]`

Nano sleep.

Parameters

<code>in</code>	<code><i>_time</i></code>	is a Time (p. 840)
-----------------	---------------------------	---------------------------

10.194.3.9 `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

10.194.3.10 `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

10.194.3.11 `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

10.194.3.12 `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

10.194.3.13 `Time gazebo::common::Time::operator* (const struct timeval & tv) const`

Multiplication operator.

Returns

Time (p. 840) instance

10.194.3.14 `Time gazebo::common::Time::operator* (const struct timespec & tv) const`

Multiplication operator.

Returns

Time (p. 840) instance

10.194.3.15 `Time gazebo::common::Time::operator* (const Time & time) const`

Multiplication operators.

Parameters

<code>in</code>	<code><i>time</i></code>	the scaling factor
-----------------	--------------------------	--------------------

Returns

a scaled **Time** (p. 840) instance

10.194.3.16 `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

10.194.3.17 `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

10.194.3.18 `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. 840) instance

10.194.3.19 `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. 840) instance

10.194.3.20 **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. 840) instance

10.194.3.21 **Time** gazebo::common::Time::operator+ (const Time & *time*) const

Addition operators.

Returns

a **Time** (p. 840) instance

10.194.3.22 **const Time&** gazebo::common::Time::operator+= (const struct timeval & *tv*)

Addition assignment operator.

Parameters

<i>in</i>	<i>tv</i>	
-----------	-----------	--

Returns

a reference to this instance

10.194.3.23 **const Time&** gazebo::common::Time::operator+= (const struct timespec & *tv*)

Addition assignment operator.

Parameters

<i>in</i>	<i>tv</i>	
-----------	-----------	--

Returns

a reference to this instance

10.194.3.24 `const Time& gazebo::common::Time::operator+=(const Time & time)`

Addition assignment operator.

Returns

a **Time** (p. 840) instance

10.194.3.25 `Time gazebo::common::Time::operator- (const struct timeval & tv) const`

Subtraction operator.

Returns

a **Time** (p. 840) instance

10.194.3.26 `Time gazebo::common::Time::operator- (const struct timespec & tv) const`

Subtraction operator.

Returns

a **Time** (p. 840) instance

10.194.3.27 `Time gazebo::common::Time::operator- (const Time & time) const`

Subtraction operator.

Returns

a **Time** (p. 840) instance

10.194.3.28 `const Time& gazebo::common::Time::operator-= (const struct timeval & tv)`

Subtraction assignment operator.

Returns

a **Time** (p. 840) instance

10.194.3.29 `const Time& gazebo::common::Time::operator-= (const struct timespec & tv)`

Subtraction assignment operator.

Returns

a **Time** (p. 840) instance

10.194.3.30 `const Time& gazebo::common::Time::operator-= (const Time & time)`

Subtraction assignment operator.

Returns

a reference to this instance

10.194.3.31 `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. 840) instance

10.194.3.32 `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. 840) instance

10.194.3.33 `Time gazebo::common::Time::operator/ (const Time & time) const`

Division operator.

Parameters

<i>in</i>	<i>time</i>	the divisor
-----------	-------------	-------------

Returns

a **Time** (p. 840) instance

10.194.3.34 `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. 840) instance

10.194.3.35 `const Time& gazebo::common::Time::operator/= (const struct timespec & tv)`

Division assignment operator.

Parameters

<i>in</i>	<i>tv</i>	a divisor
-----------	-----------	-----------

Returns

a **Time** (p. 840) instance

10.194.3.36 `const Time& gazebo::common::Time::operator/= (const Time & time)`

Division assignment operator.

Parameters

<i>in</i>	<i>time</i>	the divisor
-----------	-------------	-------------

Returns

a **Time** (p. 840) instance

10.194.3.37 `bool gazebo::common::Time::operator< (const struct timeval & tv) const`

Less than operator.

Parameters

<i>in</i>	<i>tv</i>	the time to compare with
-----------	-----------	--------------------------

Returns

true if tv is shorter than this, false otherwise

10.194.3.38 `bool gazebo::common::Time::operator< (const struct timespec & tv) const`

Less than operator.

Parameters

<i>in</i>	<i>tv</i>	the time to compare with
-----------	-----------	--------------------------

Returns

true if *tv* is shorter than this, false otherwise

10.194.3.39 `bool gazebo::common::Time::operator< (const Time & time) const`

Less than operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare with
-----------	-------------	--------------------------

Returns

true if *time* is shorter than this, false otherwise

10.194.3.40 `bool gazebo::common::Time::operator< (double time) const`

Less than operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare with
-----------	-------------	--------------------------

Returns

true if *time* is shorter than this, false otherwise

10.194.3.41 `bool gazebo::common::Time::operator<= (const struct timeval & tv) const`

Less than or equal to operator.

Parameters

<i>in</i>	<i>tv</i>	the time to compare with
-----------	-----------	--------------------------

Returns

true if *tv* is shorter than or equal to this, false otherwise

10.194.3.42 `bool gazebo::common::Time::operator<= (const struct timespec & tv) const`

Less than or equal to operator.

Parameters

<i>in</i>	<i>tv</i>	the time to compare with
-----------	-----------	--------------------------

Returns

true if *tv* is shorter than or equal to this, false otherwise

10.194.3.43 `bool gazebo::common::Time::operator<= (const Time & time) const`

Less than or equal to operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare with
-----------	-------------	--------------------------

Returns

true if *time* is shorter than or equal to this, false otherwise

10.194.3.44 `bool gazebo::common::Time::operator<= (double time) const`

Less than or equal to operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare with
-----------	-------------	--------------------------

Returns

true if *time* is shorter than or equal to this, false otherwise

10.194.3.45 `Time& gazebo::common::Time::operator= (const struct timeval & tv)`

Assignment operator.

Parameters

<i>in</i>	<i>tv</i>	the new time
-----------	-----------	--------------

Returns

a reference to this instance

10.194.3.46 `Time& gazebo::common::Time::operator= (const struct timespec & tv)`

Assignment operator.

Parameters

<i>in</i>	<i>tv</i>	the new time
-----------	-----------	--------------

Returns

a reference to this instance

10.194.3.47 `Time& gazebo::common::Time::operator=(const Time & time)`

Assignment operator.

Parameters

<i>in</i>	<i>time</i>	the new time
-----------	-------------	--------------

Returns

a reference to this instance

10.194.3.48 `bool gazebo::common::Time::operator==(const struct timeval & tv) const`

Equal to operator.

Parameters

<i>in</i>	<i>tv</i>	the time to compare to
-----------	-----------	------------------------

Returns

true if values are the same, false otherwise

10.194.3.49 `bool gazebo::common::Time::operator==(const struct timespec & tv) const`

Equal to operator.

Parameters

<i>in</i>	<i>tv</i>	the time to compare to
-----------	-----------	------------------------

Returns

true if values are the same, false otherwise

10.194.3.50 `bool gazebo::common::Time::operator==(const Time & time) const`

Equal to operator.

Parameters

<i>in</i>	<i>tv</i>	the time to compare to
-----------	-----------	------------------------

Returns

true if values are the same, false otherwise

10.194.3.51 `bool gazebo::common::Time::operator==(double time) const`

Equal to operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare to
-----------	-------------	------------------------

Returns

true if values are the same, false otherwise

10.194.3.52 `bool gazebo::common::Time::operator>(const struct timeval & tv) const`

Greater than operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare with
-----------	-------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.194.3.53 `bool gazebo::common::Time::operator>(const struct timespec & tv) const`

Greater than operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare with
-----------	-------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.194.3.54 `bool gazebo::common::Time::operator>(const Time & time) const`

Greater than operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare with
-----------	-------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.194.3.55 `bool gazebo::common::Time::operator> (double time) const`

Greater than operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare with
-----------	-------------	--------------------------

Returns

true if time is greater than this, false otherwise

10.194.3.56 `bool gazebo::common::Time::operator>= (const struct timeval & tv) const`

Greater than or equal operator.

Parameters

<i>in</i>	<i>tv</i>	the time to compare with
-----------	-----------	--------------------------

Returns

true if tv is greater than or equal to this, false otherwise

10.194.3.57 `bool gazebo::common::Time::operator>= (const struct timespec & tv) const`

Greater than or equal operator.

Parameters

<i>in</i>	<i>tv</i>	the time to compare with
-----------	-----------	--------------------------

Returns

true if tv is greater than or equal to this, false otherwise

10.194.3.58 `bool gazebo::common::Time::operator>= (const Time & time) const`

Greater than or equal operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare with
-----------	-------------	--------------------------

Returns

true if time is greater than or equal to this, false otherwise

10.194.3.59 `bool gazebo::common::Time::operator>= (double time) const`

Greater than or equal operator.

Parameters

<i>in</i>	<i>time</i>	the time to compare with
-----------	-------------	--------------------------

Returns

true if time is greater than or equal to this, false otherwise

10.194.3.60 `static double gazebo::common::Time::SecToNano (double _sec) [inline],[static]`

Convert seconds to nanoseconds.

Parameters

<i>in</i>	<i>_sec</i>	duration in seconds
-----------	-------------	---------------------

Returns

nanoseconds

10.194.3.61 `void gazebo::common::Time::Set (int32.t _sec, int32.t _nsec)`

Set to sec and nsec.

Parameters

<i>in</i>	<i>sec</i>	Seconds
<i>in</i>	<i>nsec</i>	micro seconds

10.194.3.62 `void gazebo::common::Time::Set (double _seconds)`

Set to seconds.

Parameters

<i>in</i>	<i>seconds</i>	Number of seconds
-----------	----------------	-------------------

10.194.3.63 void gazebo::common::Time::SetToWallTime ()

Set the time to the wall time.

10.194.4 Friends And Related Function Documentation

10.194.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::common::Time & *_time*) [friend]

Stream insertion operator.

Parameters

<i>in</i>	<i>_out</i>	the output stream
<i>in</i>	<i>_time</i>	time to write to the stream

Returns

the output stream

10.194.4.2 std::istream& operator>> (std::istream & *_in*, gazebo::common::Time & *_time*) [friend]

Stream extraction operator.

Parameters

<i>in</i>	<i>_in</i>	the input stream
<i>in</i>	<i>_time</i>	time to read from to the stream

Returns

the input stream

10.194.5 Member Data Documentation

10.194.5.1 int32_t gazebo::common::Time::nsec

Microseconds.

10.194.5.2 int32_t gazebo::common::Time::sec

Seconds.

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

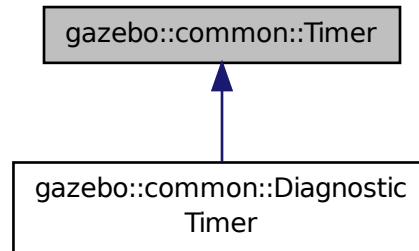
- **Time.hh**

10.195 gazebo::common::Timer Class Reference

A timer class, used to time things in real world walltime.

```
#include <Timer.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

10.195.1 Detailed Description

A timer class, used to time things in real world walltime.

10.195.2 Constructor & Destructor Documentation

10.195.2.1 gazebo::common::Timer::Timer ()

Constructor.

10.195.2.2 virtual gazebo::common::Timer::~~Timer () [virtual]

Destructor.

10.195.3 Member Function Documentation

10.195.3.1 Time gazebo::common::Timer::GetElapsed () const

Get the elapsed time.

10.195.3.2 void gazebo::common::Timer::Start ()

Start the timer.

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

10.195.4 Friends And Related Function Documentation

10.195.4.1 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:

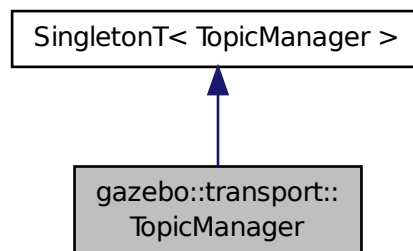
- **Timer.hh**

10.196 gazebo::transport::TopicManager Class Reference

Manages topics and their subscriptions.

```
#include <TopicManager.hh>
```

Inheritance diagram for gazebo::transport::TopicManager:



Public Types

- typedef std::map< std::string, std::list< **NodePtr** > > **SubNodeMap**

Public Member Functions

- void **AddNode** (**NodePtr** _node)
- template<typename M >
PublisherPtr Advertise (const std::string &_topic, unsigned int _queueLimit, bool _latch)
Advertise on a topic.
- void **ClearBuffers** ()
- void **ConnectPubToSub** (const std::string &topic, const **SubscriptionTransportPtr** &sublink)
Connection (p. 288) a local **Publisher** (p. 695) to a remote **Subscriber** (p. 827).
- 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. 827) to a remote **Publisher** (p. 695).*
- 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)
Disconnection all local subscribers from a remote publisher.
- **PublicationPtr FindPublication** (const std::string &topic)
- void **Fini** ()
- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)
Get all the topic namespaces.
- void **Init** ()
- bool **IsAdvertised** (const std::string &_topic)
Returns true if the topic has been advertised.
- void **PauseIncoming** (bool _pause)
- void **ProcessNodes** (bool _onlyOut=false)
- 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)
- **SubscriberPtr Subscribe** (const **SubscribeOptions** &options)
Subscribe to a topic.
- void **Unadvertise** (const std::string &topic)
Stop advertising on 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

10.196.1 Detailed Description

Manages topics and their subscriptions.

10.196.2 Member Typedef Documentation

10.196.2.1 `typedef std::map<std::string, std::list<NodePtr> > gazebo::transport::TopicManager::SubNodeMap`

10.196.3 Member Function Documentation

10.196.3.1 `void gazebo::transport::TopicManager::AddNode (NodePtr _node)`

10.196.3.2 `template<typename M > PublisherPtr gazebo::transport::TopicManager::Advertise (const std::string & _topic, unsigned int _queueLimit, bool _latch) [inline]`

Advertise on a topic.

Parameters

<i>topic</i>	The name of the topic
--------------	-----------------------

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()`.

10.196.3.3 `void gazebo::transport::TopicManager::ClearBuffers ()`

10.196.3.4 `void gazebo::transport::TopicManager::ConnectPubToSub (const std::string & topic, const SubscriptionTransportPtr & sublink)`

Connection (p. 288) a local **Publisher** (p. 695) to a remote **Subscriber** (p. 827).

10.196.3.5 `void gazebo::transport::TopicManager::ConnectSubscribers (const std::string & topic)`

Connect all subscribers on a topic to known publishers.

10.196.3.6 `void gazebo::transport::TopicManager::ConnectSubToPub (const msgs::Publish & _pub)`

Connect a local **Subscriber** (p. 827) to a remote **Publisher** (p. 695).

10.196.3.7 `void gazebo::transport::TopicManager::DisconnectPubFromSub (const std::string & topic, const std::string & host, unsigned int port)`

Disconnect a local publisher from a remote subscriber.

10.196.3.8 `void gazebo::transport::TopicManager::DisconnectSubFromPub (const std::string & topic, const std::string & host, unsigned int port)`

Disconnection all local subscribers from a remote publisher.

10.196.3.9 `PublicationPtr gazebo::transport::TopicManager::FindPublication (const std::string & topic)`

Referenced by `Advertise()`.

10.196.3.10 void gazebo::transport::TopicManager::Fini ()

10.196.3.11 void gazebo::transport::TopicManager::GetTopicNamespaces (std::list< std::string > & _namespaces)

Get all the topic namespaces.

10.196.3.12 void gazebo::transport::TopicManager::Init ()

10.196.3.13 bool gazebo::transport::TopicManager::IsAdvertised (const std::string & _topic)

Returns true if the topic has been advertised.

Parameters

<i>_topic</i>	The name of the topic to check
---------------	--------------------------------

Returns

True if the topic has been advertised

10.196.3.14 void gazebo::transport::TopicManager::PauseIncoming (bool _pause)

10.196.3.15 void gazebo::transport::TopicManager::ProcessNodes (bool _onlyOut = false)

Parameters

<i>_onlyOut</i>	True means only outbound messages on nodes will be sent. False means nodes process both outbound and inbound messages
-----------------	---

10.196.3.16 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. 695) instead of calling this function directly.

Parameters

<i>topic</i>	Name of the topic
<i>message</i>	The message to send.
<i>cb</i>	Callback, used when the publish is completed.

10.196.3.17 void gazebo::transport::TopicManager::RegisterTopicNamespace (const std::string & _name)

Register a new topic namespace.

10.196.3.18 void gazebo::transport::TopicManager::RemoveNode (unsigned int _id)

10.196.3.19 **SubscriberPtr** gazebo::transport::TopicManager::Subscribe (const **SubscribeOptions** & *options*)

Subscribe to a topic.

10.196.3.20 void gazebo::transport::TopicManager::Unadvertise (const std::string & *topic*)

Stop advertising on a topic.

10.196.3.21 void gazebo::transport::TopicManager::Unsubscribe (const std::string & *topic*, const **NodePtr** & *sub*)

Unsubscribe from a topic.

Use a **Subscriber** (p. 827) rather than calling this function directly

10.196.3.22 **PublicationPtr** gazebo::transport::TopicManager::UpdatePublications (const std::string & *topic*, const std::string & *msgType*)

Update our list of advertised topics.

Returns

True if the provided params define a new publisher.

Referenced by Advertise().

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

- **TopicManager.hh**

10.197 gazebo::physics::TrajectoryInfo Struct Reference

```
#include <Actor.hh>
```

Public Attributes

- double **duration**
- double **endTime**
- unsigned int **id**
- double **startTime**
- bool **translated**
- std::string **type**

10.197.1 Member Data Documentation

10.197.1.1 double gazebo::physics::TrajectoryInfo::duration

10.197.1.2 double gazebo::physics::TrajectoryInfo::endTime

10.197.1.3 unsigned int gazebo::physics::TrajectoryInfo::id

10.197.1.4 double gazebo::physics::TrajectoryInfo::startTime

10.197.1.5 bool gazebo::physics::TrajectoryInfo::translated

10.197.1.6 std::string gazebo::physics::TrajectoryInfo::type

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

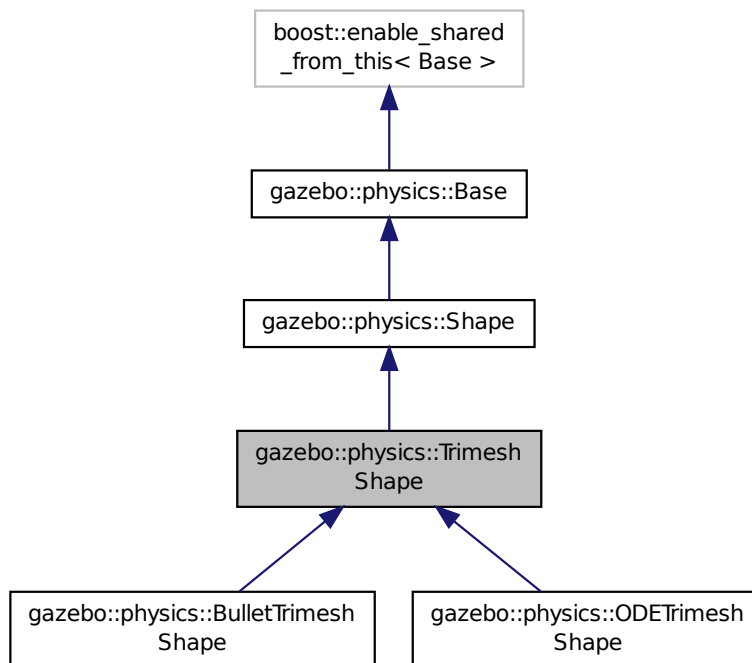
- **Actor.hh**

10.198 gazebo::physics::TrimeshShape Class Reference

Triangle mesh collision shape.

```
#include <TrimeshShape.hh>
```

Inheritance diagram for gazebo::physics::TrimeshShape:



Public Member Functions

- **TrimeshShape (CollisionPtr parent)**

Constructor.

- virtual `~TrimeshShape ()`
Destructor.
- void `FillMsg (msgs::Geometry &_msg)`
- `std::string GetFilename () const`
Get the filename of the mesh data.
- virtual double `GetMass (double _density) const`
Get the mass of a shape.
- virtual `math::Vector3 GetSize () const`
- virtual void `Init ()`
Init the trimesh shape.
- virtual void `ProcessMsg (const msgs::Geometry &_msg)`
- void `SetFilename (const std::string &_filename)`
- void `SetScale (const math::Vector3 &_scale)`
- virtual void `Update ()`
Update the object.

Protected Attributes

- const `common::Mesh * mesh`

Additional Inherited Members

10.198.1 Detailed Description

Triangle mesh collision shape.

10.198.2 Constructor & Destructor Documentation

10.198.2.1 `gazebo::physics::TrimeshShape::TrimeshShape (CollisionPtr parent)`

Constructor.

10.198.2.2 `virtual gazebo::physics::TrimeshShape::~~TrimeshShape () [virtual]`

Destructor.

10.198.3 Member Function Documentation

10.198.3.1 `void gazebo::physics::TrimeshShape::FillMsg (msgs::Geometry &_msg) [virtual]`

Implements `gazebo::physics::Shape` (p. 781).

10.198.3.2 `std::string gazebo::physics::TrimeshShape::GetFilename () const`

Get the filename of the mesh data.

10.198.3.3 `virtual double gazebo::physics::TrimeshShape::GetMass (double _density) const [virtual]`

Get the mass of a shape.

Reimplemented from `gazebo::physics::Shape` (p. 781).

10.198.3.4 `virtual math::Vector3 gazebo::physics::TrimeshShape::GetSize () const [virtual]`

10.198.3.5 `virtual void gazebo::physics::TrimeshShape::Init () [virtual]`

Init the trimesh shape.

Implements `gazebo::physics::Shape` (p. 781).

Reimplemented in `gazebo::physics::ODETrimeshShape` (p. 634), and `gazebo::physics::BulletTrimeshShape` (p. 224).

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

Implements `gazebo::physics::Shape` (p. 782).

10.198.3.7 `void gazebo::physics::TrimeshShape::SetFilename (const std::string & _filename)`

10.198.3.8 `void gazebo::physics::TrimeshShape::SetScale (const math::Vector3 & _scale)`

10.198.3.9 `virtual void gazebo::physics::TrimeshShape::Update () [inline],[virtual]`

Update the object.

Reimplemented from `gazebo::physics::Base` (p. 157).

Reimplemented in `gazebo::physics::ODETrimeshShape` (p. 634).

10.198.4 Member Data Documentation

10.198.4.1 `const common::Mesh* gazebo::physics::TrimeshShape::mesh [protected]`

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

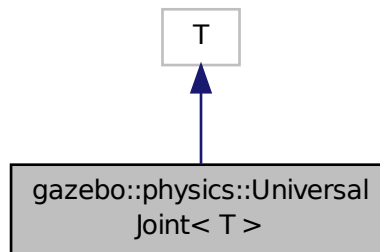
- `TrimeshShape.hh`

10.199 gazebo::physics::UniversalJoint< T > Class Template Reference

A universal joint.

```
#include <UniversalJoint.hh>
```


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



Public Member Functions

- **UniversalJoint** (**BasePtr** _parent)
Constructor.
- virtual **~UniversalJoint** ()
Destuctor.

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)
*Load a **UniversalJoint** (p. 868).*

10.199.1 Detailed Description

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

A universal joint.

10.199.2 Constructor & Destructor Documentation

10.199.2.1 `template<class T> gazebo::physics::UniversalJoint< T >::UniversalJoint (BasePtr _parent) [inline]`

Constructor.

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

Destuctor.

10.199.3 Member Function Documentation

10.199.3.1 `template<class T> virtual void gazebo::physics::UniversalJoint< T >::Load (sdf::ElementPtr _sdf)`
`[inline], [protected], [virtual]`

Load a **UniversalJoint** (p. 868).

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

- **UniversalJoint.hh**

10.200 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_**

10.200.1 Constructor & Destructor Documentation

10.200.1.1 `urdf2gazebo::URDF2Gazebo::URDF2Gazebo ()`

10.200.1.2 `urdf2gazebo::URDF2Gazebo::~~URDF2Gazebo ()`

10.200.2 Member Function Documentation

10.200.2.1 `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

10.200.2.2 `void urdf2gazebo::URDF2Gazebo::addTransform (TiXmlElement * elem, const ::gazebo::math::Pose & transform)`

append transform (pose) to the end of the xml element

10.200.2.3 `gazebo::math::Pose urdf2gazebo::URDF2Gazebo::copyPose (urdf::Pose pose)`

reduced fixed joints: utility to copy between urdf::Pose and math::Pose

10.200.2.4 `urdf::Pose urdf2gazebo::URDF2Gazebo::copyPose (gazebo::math::Pose pose)`

reduced fixed joints: utility to copy between urdf::Pose and math::Pose

10.200.2.5 `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

10.200.2.6 `void urdf2gazebo::URDF2Gazebo::createCollisions (TiXmlElement * elem, const urdf::Link * link)`

create collision blocks from urdf collisions

10.200.2.7 `void urdf2gazebo::URDF2Gazebo::createGeometry (TiXmlElement * elem, urdf::Geometry * geometry)`

create SDF geometry block based on URDF

10.200.2.8 `void urdf2gazebo::URDF2Gazebo::createInertial (TiXmlElement * elem, const urdf::Link * link)`

create SDF Inertial block based on URDF

10.200.2.9 `void urdf2gazebo::URDF2Gazebo::createJoint (TiXmlElement * root, const urdf::Link * link, gazebo::math::Pose & currentTransform)`

create SDF Joint block based on URDF

10.200.2.10 `void urdf2gazebo::URDF2Gazebo::createLink (TiXmlElement * root, const urdf::Link * link, gazebo::math::Pose & currentTransform)`

create SDF Link block based on URDF

10.200.2.11 `void urdf2gazebo::URDF2Gazebo::createSDF (TiXmlElement * root, const urdf::Link * link, const gazebo::math::Pose & transform)`

create SDF from URDF link

10.200.2.12 `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

10.200.2.13 `void urdf2gazebo::URDF2Gazebo::createVisuals (TiXmlElement * elem, const urdf::Link * link)`

create visual blocks from urdf visuals

10.200.2.14 `std::string urdf2gazebo::URDF2Gazebo::getGeometryBoundingBox (urdf::Geometry * geometry, double * sizeVals)`

10.200.2.15 `std::string urdf2gazebo::URDF2Gazebo::getKeyValueAsString (TiXmlElement * elem)`

get value from <key value="..."> pair and return it as string

10.200.2.16 `TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelDoc (TiXmlDocument * _xmlDoc)`

10.200.2.17 `TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelFile (std::string filename)`

10.200.2.18 `TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelString (std::string urdf_str)`

10.200.2.19 `TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelString (std::string urdf_str, bool _enforce_limits)`

10.200.2.20 `void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionCollision (TiXmlElement * elem, std::string link_name)`

insert extensions into collision geoms

10.200.2.21 `void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionJoint (TiXmlElement * elem, std::string joint_name)`

insert extensions into joints

10.200.2.22 `void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionLink (TiXmlElement * elem, std::string link_name)`

insert extensions into links

10.200.2.23 `void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionRobot (TiXmlElement * elem)`

insert extensions into model

10.200.2.24 void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionVisual (TiXmlElement * *elem*, std::string *link_name*)

insert extensions into visuals

10.200.2.25 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

10.200.2.26 void urdf2gazebo::URDF2Gazebo::listGazeboExtensions ()

list extensions for debugging

10.200.2.27 void urdf2gazebo::URDF2Gazebo::listGazeboExtensions (std::string *reference*)

list extensions for debugging

10.200.2.28 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

10.200.2.29 urdf::Vector3 urdf2gazebo::URDF2Gazebo::parseVector3 (TiXmlNode * *key*, double *scale* = 1.0)

parser xml for vector 3

10.200.2.30 void urdf2gazebo::URDF2Gazebo::printCollisionGroups (urdf::Link * *link*)

print collision groups for debugging purposes

10.200.2.31 void urdf2gazebo::URDF2Gazebo::printMass (urdf::Link * *link*)

print mass for link for debugging

10.200.2.32 void urdf2gazebo::URDF2Gazebo::printMass (std::string *link_name*, dMass *mass*)

print mass for link for debugging

10.200.2.33 void urdf2gazebo::URDF2Gazebo::reduceCollisionsToParent (urdf::Link * *link*)

reduce fixed joints: lump collisions to parent link

10.200.2.34 `void urdf2gazebo::URDF2Gazebo::reduceCollisionToParent (urdf::Link * link, std::string group_name, urdf::Collision * collision)`

reduce fixed joints: lump collision when reducing fixed joints

10.200.2.35 `void urdf2gazebo::URDF2Gazebo::reduceFixedJoints (TiXmlElement * root, urdf::Link * link)`

reduce fixed joints by lumping inertial, visual and

10.200.2.36 `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

10.200.2.37 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionFrameReplace (GazeboExtension * ge, urdf::Link * link)`

reduced fixed joints: apply appropriate frame updates in urdf extensions when doing fixed joint reduction

10.200.2.38 `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

10.200.2.39 `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

10.200.2.40 `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

10.200.2.41 `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

10.200.2.42 `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

10.200.2.43 `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

10.200.2.44 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionsTransformReduction (GazeboExtension * ge)`

reduced fixed joints: apply transform reduction to extensions when doing fixed joint reduction

10.200.2.45 `void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionToParent (urdf::Link * link)`

reduced fixed joints: apply appropriate updates to urdf extensions when doing fixed joint reduction

10.200.2.46 `void urdf2gazebo::URDF2Gazebo::reduceInertialToParent (urdf::Link * link)`

reduce fixed joints: lump inertial to parent link

10.200.2.47 `void urdf2gazebo::URDF2Gazebo::reduceJointsToParent (urdf::Link * link)`

reduce fixed joints: lump joints to parent link

10.200.2.48 `void urdf2gazebo::URDF2Gazebo::reduceVisualsToParent (urdf::Link * link)`

reduce fixed joints: lump visuals to parent link

10.200.2.49 `void urdf2gazebo::URDF2Gazebo::reduceVisualToParent (urdf::Link * link, std::string group_name, urdf::Visual * visual)`

reduce fixed joints: lump visuals when reducing fixed joints

10.200.2.50 `urdf::Pose urdf2gazebo::URDF2Gazebo::transformToParentFrame (urdf::Pose transform_in_link_frame, urdf::Pose parent_to_link_transform)`

reduced fixed joints: transform to parent frame

10.200.2.51 `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

10.200.2.52 `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

10.200.2.53 `std::string urdf2gazebo::URDF2Gazebo::values2str (unsigned int count, const double * values)`

convert values to string

10.200.2.54 `std::string urdf2gazebo::URDF2Gazebo::vector32str (const urdf::Vector3 vector)`

convert Vector3 to string

10.200.3 Member Data Documentation

10.200.3.1 `std::map<std::string, std::vector<GazeboExtension*> > urdf2gazebo::URDF2Gazebo::gazebo_extensions_`

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

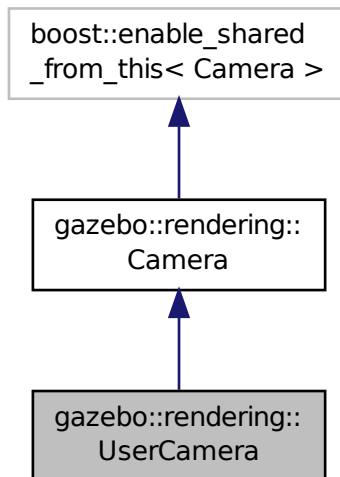
- `parser_urdf.hh`

10.201 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, **Scene** *_scene)

Constructor.

- virtual \sim **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.

- 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 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

10.201.1 Detailed Description

A camera used for user visualization of a scene.

10.201.2 Constructor & Destructor Documentation

10.201.2.1 gazebo::rendering::UserCamera::UserCamera (const std::string & *_name*, Scene * *_scene*)

Constructor.

Parameters

<i>in</i>	<i>_name</i>	Name of the camera.
<i>in</i>	<i>_scene</i>	Scene (p. 746) to put the camera in.

10.201.2.2 virtual gazebo::rendering::UserCamera::~UserCamera () [virtual]

Destructor.

10.201.3 Member Function Documentation

10.201.3.1 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

<i>in</i>	<i>_visual</i>	The visual to attach to.
<i>in</i>	<i>_inherit-Orientation</i>	True if the camera should also rotate when the visual rotates.
<i>in</i>	<i>_minDist</i>	Minimum distance the camera can get to the visual.
<i>in</i>	<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. 239).

10.201.3.2 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

<i>in</i>	<i>_value</i>	True to enable viewcontroller, False to disable.
-----------	---------------	--

10.201.3.3 void gazebo::rendering::UserCamera::Fini ()

Finalize.

10.201.3.4 float gazebo::rendering::UserCamera::GetAvgFPS () const

Get the average frames per second.

Returns

The average rendering frames per second

10.201.3.5 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. 394).

10.201.3.6 float gazebo::rendering::UserCamera::GetTriangleCount () const

Get the triangle count.

Returns

The number of triangles currently being rendered.

10.201.3.7 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

in	<i>_mousePos</i>	The position of the mouse in screen coordinates
out	<i>_mod</i>	Used for object manipulation

Returns

The selected entity, or NULL

10.201.3.8 VisualPtr gazebo::rendering::UserCamera::GetVisual (const math::Vector2i & _mousePos) const

Get a visual at a mouse position.

Parameters

in	<i>_mousePos</i>	2D position of the mouse in pixels.
----	------------------	-------------------------------------

10.201.3.9 void gazebo::rendering::UserCamera::HandleKeyPressEvent (const std::string & _key)

Handle a key press.

Parameters

in	<i>_key</i>	The key pressed.
----	-------------	------------------

10.201.3.10 void gazebo::rendering::UserCamera::HandleKeyReleaseEvent (const std::string & _key)

Handle a key release.

Parameters

in	<i>_key</i>	The key released.
----	-------------	-------------------

10.201.3.11 void gazebo::rendering::UserCamera::HandleMouseEvent (const common::MouseEvent & _evt)

Handle a mouse event.

Parameters

in	<code>_evt</code>	The mouse event.
----	-------------------	------------------

10.201.3.12 `void gazebo::rendering::UserCamera::Init ()`

Initialize.

10.201.3.13 `void gazebo::rendering::UserCamera::Load (sdf::ElementPtr _sdf)`

Load the user camera.

Parameters

in	<code>_sdf</code>	Parameters for the camera.
----	-------------------	----------------------------

10.201.3.14 `void gazebo::rendering::UserCamera::Load ()`

Generic load function.

10.201.3.15 `virtual bool gazebo::rendering::UserCamera::MoveToPosition (const math::Pose & _pose, double _time)`
[virtual]

Move the camera to a position.

This is an animated motion

Parameters

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

Reimplemented from `gazebo::rendering::Camera` (p. 248).

10.201.3.16 `void gazebo::rendering::UserCamera::MoveToVisual (VisualPtr _visual)`

Move the camera to focus on a visual.

Parameters

in	<code>_visual</code>	Visual (p. 931) to move the camera to.
----	----------------------	---

10.201.3.17 `void gazebo::rendering::UserCamera::MoveToVisual (const std::string & _visualName)`

Move the camera to focus on a visual.

Parameters

in	<code>_visualName</code>	Name of the visual to move the camera to.
----	--------------------------	---

10.201.3.18 `virtual void gazebo::rendering::UserCamera::PostRender () [virtual]`

Post render.

Reimplemented from `gazebo::rendering::Camera` (p. 248).

10.201.3.19 `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.

10.201.3.20 `void gazebo::rendering::UserCamera::SetFocalPoint (const math::Vector3 & _pt)`

Set the point the camera should orbit around.

Parameters

<code>in</code>	<code>_pt</code>	The focal point
-----------------	------------------	-----------------

10.201.3.21 `virtual void gazebo::rendering::UserCamera::SetRenderTarget (Ogre::RenderTarget * _target) [virtual]`

Set to true to enable rendering.

Use this only if you really know what you're doing.

Parameters

<code>in</code>	<code>_target</code>	The new rendering target.
-----------------	----------------------	---------------------------

Reimplemented from `gazebo::rendering::Camera` (p. 251).

10.201.3.22 `void gazebo::rendering::UserCamera::SetViewController (const std::string & _type)`

Set view controller.

Parameters

<code>in</code>	<code>_type</code>	The type of view controller: "orbit", "fps"
-----------------	--------------------	---

10.201.3.23 `void gazebo::rendering::UserCamera::SetViewController (const std::string & _type, const math::Vector3 & _pos)`

Set view controller.

Parameters

<code>in</code>	<code>_type</code>	The type of view controller: "orbit", "fps"
<code>in</code>	<code>_pos</code>	The initial pose of the camera.

10.201.3.24 `void gazebo::rendering::UserCamera::SetViewportDimensions (float _x, float _y, float _w, float _h)`

Set the dimensions of the viewport.

Parameters

<code>in</code>	<code>_x</code>	X position of the viewport.
<code>in</code>	<code>_y</code>	Y position of the viewport.
<code>in</code>	<code>_w</code>	Width of the viewport.
<code>in</code>	<code>_h</code>	Height of the viewport.

10.201.3.25 `virtual void gazebo::rendering::UserCamera::SetWorldPose (const math::Pose & _pose) [virtual]`

Set the pose in the world coordinate frame.

Parameters

<code>in</code>	<code>_pose</code>	New pose of the camera.
-----------------	--------------------	-------------------------

Reimplemented from **gazebo::rendering::Camera** (p. 252).

10.201.3.26 `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

<code>in</code>	<code>_visual</code>	Visual (p. 931) to track.
-----------------	----------------------	----------------------------------

Returns

True if the camera is now tracking the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 253).

10.201.3.27 `virtual void gazebo::rendering::UserCamera::Update () [virtual]`

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 254).

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

- **UserCamera.hh**

10.202 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 \sim **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.

10.202.1 Detailed Description

Generic double x, y vector.

10.202.2 Constructor & Destructor Documentation

10.202.2.1 gazebo::math::Vector2d::Vector2d ()

Constructor.

10.202.2.2 gazebo::math::Vector2d::Vector2d (const double & _x, const double & _y)

Constructor.

Parameters

<i>in</i>	<i>_x</i>	value along x
<i>in</i>	<i>_y</i>	value along y

10.202.2.3 gazebo::math::Vector2d::Vector2d (const Vector2d & _v)

Copy constructor.

Parameters

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

10.202.2.4 virtual gazebo::math::Vector2d::~~Vector2d () [virtual]

Destructor.

10.202.3 Member Function Documentation

10.202.3.1 Vector2d gazebo::math::Vector2d::Cross (const Vector2d & _v) const

Return the cross product of this vector and _v.

Parameters

in	_v	the vector
----	----	------------

Returns

the cross product

10.202.3.2 double gazebo::math::Vector2d::Distance (const Vector2d & _pt) const

Calc distance to the given point.

Returns

the distance

10.202.3.3 bool gazebo::math::Vector2d::IsFinite () const

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.202.3.4 void gazebo::math::Vector2d::Normalize ()

Normalize the vector length.

10.202.3.5 bool gazebo::math::Vector2d::operator!= (const Vector2d & _v) const

Not equal to operator.

Returns

true if elements are of different values (tolerance 1e-6)

10.202.3.6 `const Vector2d gazebo::math::Vector2d::operator* (const Vector2d & _v) const`

Multiplication operators.

Parameters

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

Returns

the result

10.202.3.7 `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

10.202.3.8 `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

10.202.3.9 `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

10.202.3.10 `Vector2d gazebo::math::Vector2d::operator+ (const Vector2d & _v) const`

Addition operator.

Parameters

<code>in</code>	<code>_v</code>	vector to add
-----------------	-----------------	---------------

Returns

sum vector

10.202.3.11 `const Vector2d& gazebo::math::Vector2d::operator+= (const Vector2d & _v)`

Addition assignment operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to add
-----------------	-----------------	-------------------

10.202.3.12 `Vector2d gazebo::math::Vector2d::operator- (const Vector2d & _v) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

the subtracted vector

10.202.3.13 `const Vector2d& gazebo::math::Vector2d::operator-= (const Vector2d & _v)`

Subtraction assignment operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

this

10.202.3.14 `const Vector2d gazebo::math::Vector2d::operator/ (const Vector2d & _v) const`

Division operator.

Remarks

this is an element wise division

Parameters

<code>in</code>	<code>_v</code>	a vector
-----------------	-----------------	----------

Returns

a result

10.202.3.15 `const Vector2d gazebo::math::Vector2d::operator/ (double _v) const`

Division operator.

Parameters

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

Returns

a vector

10.202.3.16 `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

10.202.3.17 `const Vector2d& gazebo::math::Vector2d::operator/= (double _v)`

Division operator.

Parameters

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

Returns

a vector

10.202.3.18 `Vector2d& gazebo::math::Vector2d::operator= (const Vector2d & _v)`

Assignment operator.

Parameters

<code>in</code>	<code>_v</code>	a value for x and y element
-----------------	-----------------	-----------------------------

Returns

this

10.202.3.19 `const Vector2d& gazebo::math::Vector2d::operator= (double _v)`

Assignment operator.

Parameters

<code>in</code>	<code>_v</code>	the value for x and y element
-----------------	-----------------	-------------------------------

Returns

this

10.202.3.20 `bool gazebo::math::Vector2d::operator==(const Vector2d & _v) const`

Equal to operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to compare to
-----------------	-----------------	--------------------------

Returns

true if the elements of the 2 vectors are equal within a tolerance (1e-6)

10.202.3.21 `double gazebo::math::Vector2d::operator[] (unsigned int _index) const`

Array subscript operator.

Parameters

in	<code>_index</code>	the index
----	---------------------	-----------

Returns

the value, or 0 if `_index` is out of bounds

10.202.3.22 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

10.202.4 Friends And Related Function Documentation

10.202.4.1 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. 885) to output

Returns

The stream

10.202.4.2 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. 902) to read values into

Returns

The stream

10.202.5 Member Data Documentation

10.202.5.1 double gazebo::math::Vector2d::x

x data

10.202.5.2 double gazebo::math::Vector2d::y

y data

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

- **Vector2d.hh**

10.203 gazebo::math::Vector2i Class Reference

Generic integer x, y vector.

```
#include <Vector2i.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.

10.203.1 Detailed Description

Generic integer x, y vector.

10.203.2 Constructor & Destructor Documentation

10.203.2.1 gazebo::math::Vector2i::Vector2i ()

Constructor.

10.203.2.2 gazebo::math::Vector2i::Vector2i (const int & *_x*, const int & *_y*)

Constructor.

Parameters

in	<i>_x</i>	value along x
in	<i>_y</i>	value along y

10.203.2.3 gazebo::math::Vector2i::Vector2i (const Vector2i & *_pt*)

Copy onstructor.

Parameters

in	<i>_pt</i>	a point
----	------------	---------

10.203.2.4 virtual gazebo::math::Vector2i::~~Vector2i () [virtual]

Destructor.

10.203.3 Member Function Documentation

10.203.3.1 Vector2i gazebo::math::Vector2i::Cross (const Vector2i & *_pt*) const

Return the cross product of this vector and *pt*.

Parameters

in	<i>the</i>	other vector
----	------------	--------------

Returns

the product

10.203.3.2 int gazebo::math::Vector2i::Distance (const Vector2i & *_pt*) const

Calc distance to the given point.

Parameters

in	<i>_pt</i>	a point
----	------------	---------

Returns

the distance

10.203.3.3 `bool gazebo::math::Vector2i::IsFinite () const`

See if a point is finite (e.g., not nan)

Returns

the result

10.203.3.4 `void gazebo::math::Vector2i::Normalize ()`

Normalize the vector length.

10.203.3.5 `bool gazebo::math::Vector2i::operator!= (const Vector2i & _v) const`

Equality operators.

Parameters

<code>_v</code>	the vector to compare with
-----------------	----------------------------

Returns

true if component have different values, false otherwise

10.203.3.6 `const Vector2i gazebo::math::Vector2i::operator* (const Vector2i & _v) const`

Multiplication operator.

Remarks

this is an element wise multiplication

Parameters

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

Returns

the result

10.203.3.7 `const Vector2i gazebo::math::Vector2i::operator* (int _v) const`

Multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	the scaling factor
-----------------	-----------------	--------------------

Returns

the result

10.203.3.8 `const Vector2i& gazebo::math::Vector2i::operator*=(const Vector2i & _v)`

Multiplication operators.

Remarks

this is an element wise multiplication

Parameters

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

Returns

this

10.203.3.9 `const Vector2i& gazebo::math::Vector2i::operator*=(int _v)`

Multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

this

10.203.3.10 `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

10.203.3.11 `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

10.203.3.12 `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

10.203.3.13 `const Vector2i& gazebo::math::Vector2i::operator-= (const Vector2i & _v)`

Subtraction operators.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

this

10.203.3.14 `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

10.203.3.15 `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

10.203.3.16 `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

10.203.3.17 `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

10.203.3.18 `Vector2i& gazebo::math::Vector2i::operator= (const Vector2i & _v)`

Assignment operator.

Parameters

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

Returns

this

10.203.3.19 `const Vector2i& gazebo::math::Vector2i::operator= (int _value)`

Assignment operator.

Parameters

<code>in</code>	<code>_value</code>	the value for x and y
-----------------	---------------------	-----------------------

Returns

this

10.203.3.20 `bool gazebo::math::Vector2i::operator==(const Vector2i & _v) const`

Equality operator.

Parameters

<code>_v</code>	the vector to compare with
-----------------	----------------------------

Returns

true if component have the same values, false otherwise

10.203.3.21 `int gazebo::math::Vector2i::operator[] (unsigned int _index) const`

Array subscript operator.

Parameters

<code>in</code>	<code>_index</code>	the array index
-----------------	---------------------	-----------------

10.203.3.22 `void gazebo::math::Vector2i::Set (int _x, int _y)`

Set the contents of the vector.

Parameters

in	<code>_x</code>	value along x
in	<code>_y</code>	value along y

10.203.4 Friends And Related Function Documentation

10.203.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector2i & _pt)` [friend]

Stream insertion operator.

Parameters

in	<code>_out</code>	output stream
in	<code>pt</code>	Vector2i (p. 894) to output

Returns

the stream

10.203.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector2i & _pt)` [friend]

Stream extraction operator.

Parameters

in	<code>_in</code>	input stream
in	<code>pt</code>	Vector3 (p. 902) to read values into

Returns

The stream

10.203.5 Member Data Documentation

10.203.5.1 `int gazebo::math::Vector2i::x`

x data

10.203.5.2 `int gazebo::math::Vector2i::y`

y data

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

- **Vector2i.hh**

10.204 gazebo::math::Vector3 Class Reference

The **Vector3** (p. 902) class represents the generic vector containing 3 elements.

```
#include <Vector3.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.

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.

10.204.1 Detailed Description

The **Vector3** (p. 902) 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.

10.204.2 Constructor & Destructor Documentation

10.204.2.1 gazebo::math::Vector3::Vector3 ()

Constructor.

Referenced by operator-().

10.204.2.2 gazebo::math::Vector3::Vector3 (const double & _x, const double & _y, const double & _z)

Constructor.

Parameters

in	_x	value along x
in	_y	value along y
in	_z	value along z

10.204.2.3 gazebo::math::Vector3::Vector3 (const Vector3 & _v)

Copy constructor.

Parameters

in	_v	a vector
----	----	----------

10.204.2.4 `virtual gazebo::math::Vector3::~~Vector3 () [virtual]`

Destructor.

10.204.3 Member Function Documentation

10.204.3.1 `void gazebo::math::Vector3::Correct () [inline]`

Corrects any nan values.

References x, y, and z.

Referenced by `gazebo::math::Pose::Correct()`.

10.204.3.2 `Vector3 gazebo::math::Vector3::Cross (const Vector3 & _pt) const`

Return the cross product of this vector and pt.

Returns

the product

10.204.3.3 `double gazebo::math::Vector3::Distance (const Vector3 & _pt) const`

Calc distance to the given point.

Parameters

<code>in</code>	<code>_pt</code>	the point
-----------------	------------------	-----------

Returns

the distance

10.204.3.4 `double gazebo::math::Vector3::Distance (double _x, double _y, double _z) const`

Calc distance to the given point.

Parameters

<code>in</code>	<code>_x</code>	value along x
<code>in</code>	<code>_y</code>	value along y
<code>in</code>	<code>_z</code>	value along z

Returns

the distance

10.204.3.5 `double gazebo::math::Vector3::Dot (const Vector3 & _pt) const`

Return the dot product of this vector and pt.

Returns

the product

10.204.3.6 `bool gazebo::math::Vector3::Equal (const Vector3 & _v) const`

Equality test.

Remarks

This is equivalent to the == operator

Parameters

in	_v	the other vector
----	----	------------------

Returns

true if the 2 vectors have the same values, false otherwise

10.204.3.7 `Vector3 gazebo::math::Vector3::GetAbs () const`

Get the absolute value of the vector.

Returns

a vector with positive elements

10.204.3.8 `double gazebo::math::Vector3::GetDistToLine (const Vector3 & _pt1, const Vector3 & _pt2)`

Get distance to a line.

Parameters

in	_pt1	first point on the line
in	_pt2	second point on the line

Returns

the minimum distance from this point to the line

10.204.3.9 `double gazebo::math::Vector3::GetLength () const`

Returns the length (magnitude) of the vector \ return the length.

10.204.3.10 `double gazebo::math::Vector3::GetMax () const`

Get the maximum value in the vector.

Returns

the maximum element

10.204.3.11 `double gazebo::math::Vector3::GetMin () const`

Get the minimum value in the vector.

Returns

the minimum element

10.204.3.12 `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

10.204.3.13 `Vector3 gazebo::math::Vector3::GetPerpendicular () const`

Return a vector that is perpendicular to this one.

Returns

an orthogonal vector

10.204.3.14 `Vector3 gazebo::math::Vector3::GetRounded () const`

Get a rounded version of this vector.

Returns

a rounded vector

10.204.3.15 `double gazebo::math::Vector3::GetSquaredLength () const`

Return the square of the length (magnitude) of the vector.

Returns

the length

10.204.3.16 `double gazebo::math::Vector3::GetSum () const`

Return the sum of the values.

Returns

the sum

10.204.3.17 `bool gazebo::math::Vector3::IsFinite () const`

See if a point is finite (e.g., not nan)

10.204.3.18 `Vector3 gazebo::math::Vector3::Normalize ()`

Normalize the vector length.

Returns

unit length vector

10.204.3.19 `bool gazebo::math::Vector3::operator!=(const Vector3 & _v) const`

Not equal to operator.

Parameters

<code>in</code>	<code>_v</code>	a vector
-----------------	-----------------	----------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.204.3.20 `Vector3 gazebo::math::Vector3::operator*(const Vector3 & _p) const`

Multiplication operator.

Remarks

this is an element wise multiplication, not a cross product

Parameters

<code>in</code>	<code>_v</code>	
-----------------	-----------------	--

10.204.3.21 `Vector3 gazebo::math::Vector3::operator*(double _v) const`

Multiplication operators.

Parameters

<code>in</code>	<code>_v</code>	the scaling factor
-----------------	-----------------	--------------------

Returns

a scaled vector

10.204.3.22 `const Vector3& gazebo::math::Vector3::operator*=(const Vector3 & _v)`

Multiplication operators.

Remarks

this is an element wise multiplication, not a cross product

Parameters

<code>in</code>	<code>_v</code>	a vector
-----------------	-----------------	----------

Returns

this

10.204.3.23 `const Vector3& gazebo::math::Vector3::operator*=(double _v)`

Multiplication operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

this

10.204.3.24 `Vector3 gazebo::math::Vector3::operator+(const Vector3 & _v) const`

Addition operator.

Parameters

<code>in</code>	<code>_v</code>	vector to add
-----------------	-----------------	---------------

Returns

the sum vector

10.204.3.25 `const Vector3& gazebo::math::Vector3::operator+=(const Vector3 & _v)`

Addition assignment operator.

Parameters

<code>in</code>	<code>_v</code>	vector to add
-----------------	-----------------	---------------

10.204.3.26 `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.

10.204.3.27 `const Vector3& gazebo::math::Vector3::operator-=(const Vector3 & _pt)`

Subtraction operators.

10.204.3.28 `const Vector3 gazebo::math::Vector3::operator/(const Vector3 & _pt) const`

Division operator.

Remarks

this is an element wise division

Returns

a vector

10.204.3.29 `const Vector3 gazebo::math::Vector3::operator/(double _v) const`

Division operator.

Remarks

this is an element wise division

Returns

a vector

10.204.3.30 `const Vector3& gazebo::math::Vector3::operator/= (const Vector3 & _pt)`

Division assignment operator.

Remarks

this is an element wise division

Returns

a vector

10.204.3.31 `const Vector3& gazebo::math::Vector3::operator/= (double _v)`

Division operator.

Remarks

this is an element wise division

Returns

this

10.204.3.32 `Vector3& gazebo::math::Vector3::operator= (const Vector3 & _v)`

Assignment operator.

Parameters

<code>in</code>	<code>_v</code>	a new value
-----------------	-----------------	-------------

Returns

this

10.204.3.33 `Vector3& gazebo::math::Vector3::operator= (double _value)`

Assignment operator.

Parameters

<code>in</code>	<code>_value</code>	assigned to all elements
-----------------	---------------------	--------------------------

Returns

this

10.204.3.34 `bool gazebo::math::Vector3::operator==(const Vector3 & _pt) const`

Equal to operator.

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.204.3.35 `double gazebo::math::Vector3::operator[](unsigned int index) const`

[] operator

10.204.3.36 `Vector3 gazebo::math::Vector3::Round ()`

Round to near whole number, return the result.

Returns

the result

10.204.3.37 `void gazebo::math::Vector3::Round (int _precision)`

Round all values to `_precision` decimal places.

Parameters

<code>in</code>	<code>_precision</code>	the decimal places
-----------------	-------------------------	--------------------

10.204.3.38 `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.

10.204.3.39 `void gazebo::math::Vector3::SetToMax (const Vector3 & _v)`

Set this vector's components to the maximum of itself and the passed in vector.

Parameters

<code>in</code>	<code>_v</code>	the maximum clamping vector
-----------------	-----------------	-----------------------------

10.204.3.40 `void gazebo::math::Vector3::SetToMin (const Vector3 & _v)`

Set this vector's components to the minimum of itself and the passed in vector.

Parameters

<code>in</code>	<code>the</code>	minimum clamping vector
-----------------	------------------	-------------------------

10.204.4 Friends And Related Function Documentation

10.204.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector3 & _pt)` [*friend*]

Stream insertion operator.

Parameters

<code>_out</code>	output stream
<code>_pt</code>	Vector3 (p. 902) to output

Returns

the stream

10.204.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector3 & _pt)` [*friend*]

Stream extraction operator.

Parameters

<code>_in</code>	input stream
<code>_pt</code>	vector3 to read values into

Returns

the stream

10.204.5 Member Data Documentation

10.204.5.1 `double gazebo::math::Vector3::x`

X location.

Referenced by `gazebo::math::Pose::CoordPositionSub()`, `Correct()`, `gazebo::physics::ODEPlaneShape::CreatePlane()`,

operator-(), gazebo::math::Quaternion::RotateVector(), Set(), gazebo::physics::ODEPlaneShape::SetAltitude(), gazebo::physics::ODEBoxShape::SetSize(), and gazebo::physics::BulletBoxShape::SetSize().

10.204.5.2 double gazebo::math::Vector3::y

Y location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), gazebo::physics::ODEPlaneShape::CreatePlane(), operator-(), gazebo::math::Quaternion::RotateVector(), Set(), gazebo::physics::ODEPlaneShape::SetAltitude(), gazebo::physics::ODEBoxShape::SetSize(), and gazebo::physics::BulletBoxShape::SetSize().

10.204.5.3 double gazebo::math::Vector3::z

Z location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), gazebo::physics::ODEPlaneShape::CreatePlane(), operator-(), gazebo::math::Quaternion::RotateVector(), Set(), gazebo::physics::ODEPlaneShape::SetAltitude(), gazebo::physics::ODEBoxShape::SetSize(), and gazebo::physics::BulletBoxShape::SetSize().

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

- **Vector3.hh**

10.205 gazebo::math::Vector4 Class Reference

double Generic x, y, z, w vector

```
#include <Vector4.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.

10.205.1 Detailed Description

double Generic x, y, z, w vector

10.205.2 Constructor & Destructor Documentation

10.205.2.1 gazebo::math::Vector4::Vector4 ()

Constructor.

10.205.2.2 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

10.205.2.3 gazebo::math::Vector4::Vector4 (const Vector4 &_v)

Copy constructor.

Parameters

<code>in</code>	<code>_v</code>	vector
-----------------	-----------------	--------

10.205.2.4 virtual gazebo::math::Vector4::~~Vector4 () [virtual]

Destructor.

10.205.3 Member Function Documentation

10.205.3.1 double gazebo::math::Vector4::Distance (const Vector4 &_pt) const

Calc distance to the given point.

Parameters

in	<code>_pt</code>	the point
----	------------------	-----------

Returns

the distance

10.205.3.2 `double gazebo::math::Vector4::GetLength () const`

Returns the length (magnitude) of the vector.

10.205.3.3 `double gazebo::math::Vector4::GetSquaredLength () const`

Return the square of the length (magnitude) of the vector.

Returns

the length

10.205.3.4 `bool gazebo::math::Vector4::IsFinite () const`

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.205.3.5 `void gazebo::math::Vector4::Normalize ()`

Normalize the vector length.

10.205.3.6 `bool gazebo::math::Vector4::operator!= (const Vector4 & _pt) const`

Not 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

10.205.3.7 `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

10.205.3.8 `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`

10.205.3.9 `const Vector4 gazebo::math::Vector4::operator* (double _v) const`

Multiplication operators.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

a scaled vector

10.205.3.10 `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

10.205.3.11 `const Vector4& gazebo::math::Vector4::operator*=(double _v)`

Multiplication assignment operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

this

10.205.3.12 `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

10.205.3.13 `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

10.205.3.14 `Vector4 gazebo::math::Vector4::operator-(const Vector4 & _v) const`

Subtraction operator.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

a vector

10.205.3.15 `const Vector4& gazebo::math::Vector4::operator-= (const Vector4 & _v)`

Subtraction assignment operators.

Parameters

<code>in</code>	<code>_v</code>	the vector to subtract
-----------------	-----------------	------------------------

Returns

this vector

10.205.3.16 `const Vector4 gazebo::math::Vector4::operator/ (const Vector4 & _v) const`

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

<code>in</code>	<code>_v</code>	the vector to perform element wise division with
-----------------	-----------------	--

Returns

a result vector

10.205.3.17 `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

10.205.3.18 `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

10.205.3.19 `const Vector4& gazebo::math::Vector4::operator/= (double _v)`

Division operator.

Parameters

<code>in</code>	<code>_v</code>	scaling factor
-----------------	-----------------	----------------

Returns

a vector

10.205.3.20 `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

10.205.3.21 `Vector4& gazebo::math::Vector4::operator= (double _value)`

Assignment operator.

Parameters

<code>in</code>	<code>_value</code>	
-----------------	---------------------	--

10.205.3.22 `bool gazebo::math::Vector4::operator==(const Vector4 & _pt) const`

Equal to operator.

Parameters

<code>in</code>	<code>_pt</code>	the other vector
-----------------	------------------	------------------

Returns

true if each component is equal withing a default tolerance (1e-6), false otherwise

10.205.3.23 `double gazebo::math::Vector4::operator[] (unsigned int _index) const`

Array subscript operator.

Parameters

<code>in</code>	<code>_index</code>	
-----------------	---------------------	--

10.205.3.24 `void gazebo::math::Vector4::Set (double _x = 0, double _y = 0, double _z = 0, double _w = 0)`

Set the contents of the vector.

Parameters

<code>in</code>	<code>_x</code>	value along x axis
<code>in</code>	<code>_y</code>	value along y axis
<code>in</code>	<code>_z</code>	value along z axis
<code>in</code>	<code>_w</code>	value along w axis

10.205.4 Friends And Related Function Documentation

10.205.4.1 `std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector4 & _pt) [friend]`

Stream insertion operator.

Parameters

<code>in</code>	<code>_out</code>	output stream
<code>in</code>	<code>_pt</code>	Vector4 (p. 915) to output

Returns

The stream

10.205.4.2 `std::istream& operator>> (std::istream & _in, gazebo::math::Vector4 & _pt) [friend]`

Stream extraction operator.

Parameters

<code>in</code>	<code>_in</code>	input stream
<code>in</code>	<code>_pt</code>	Vector4 (p. 915) to read values into

Returns

the stream

10.205.5 Member Data Documentation

10.205.5.1 double gazebo::math::Vector4::w

W value.

10.205.5.2 double gazebo::math::Vector4::x

X value.

10.205.5.3 double gazebo::math::Vector4::y

Y value.

10.205.5.4 double gazebo::math::Vector4::z

Z value.

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

- **Vector4.hh**

10.206 gazebo::common::Video Class Reference

Handle video encoding and decoding using libavcodec.

```
#include <Video.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.

10.206.1 Detailed Description

Handle video encoding and decoding using libavcodec.

10.206.2 Constructor & Destructor Documentation

10.206.2.1 gazebo::common::Video::Video ()

Constructor.

10.206.2.2 virtual gazebo::common::Video::~~Video () [virtual]

Destructor.

10.206.3 Member Function Documentation

10.206.3.1 int gazebo::common::Video::GetHeight () const

Get the height of the video in pixels.

Returns

the height

10.206.3.2 bool gazebo::common::Video::GetNextFrame (unsigned char ** _buffer)

Get the next frame of the video.

Parameters

out	<i>_img</i>	Image (p. 407) in which the frame is stored
-----	-------------	--

Returns

false if HAVE_FFmpeg is not defined, true otherwise

10.206.3.3 int gazebo::common::Video::GetWidth () const

Get the width of the video in pixels.

Returns

the width

10.206.3.4 `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:

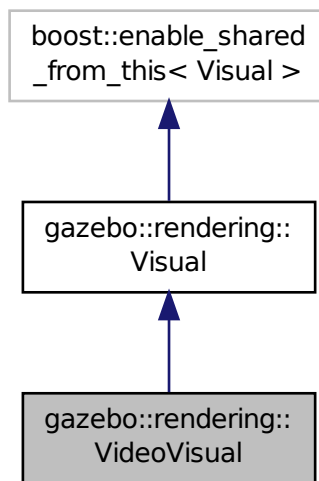
- **Video.hh**

10.207 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

10.207.1 Detailed Description

A visual element that displays a video as a texture.

10.207.2 Constructor & Destructor Documentation

10.207.2.1 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.

10.207.2.2 virtual gazebo::rendering::VideoVisual::~~VideoVisual () [virtual]

Destructor.

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

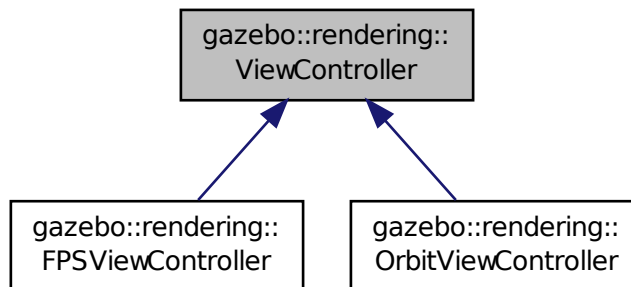
- **VideoVisual.hh**

10.208 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. 233).*

Protected Attributes

- **UserCameraPtr** camera
Pointer to the camera to control.
- bool **enabled**
True if enabled.
- std::string **typeString**
Type of view controller.

10.208.1 Detailed Description

Base class for view controllers.

10.208.2 Constructor & Destructor Documentation

10.208.2.1 gazebo::rendering::ViewController::ViewController (**UserCameraPtr** _camera)

Constructor.

Parameters

in	_camera	The user camera to controll.
----	---------	------------------------------

10.208.2.2 virtual gazebo::rendering::ViewController::~~ViewController () [virtual]

Destructor.

10.208.3 Member Function Documentation

10.208.3.1 std::string gazebo::rendering::ViewController::GetTypeString () const

Get the type of view controller.

Returns

The view controller type string.

10.208.3.2 virtual void gazebo::rendering::ViewController::HandleKeyPressEvent (const std::string & *_key*) [pure virtual]

Handle a key press event.

Parameters

in	<i>_key</i>	The key that was pressed.
----	-------------	---------------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 640), and **gazebo::rendering::FPSViewController** (p. 368).

10.208.3.3 virtual void gazebo::rendering::ViewController::HandleKeyReleaseEvent (const std::string & *_key*) [pure virtual]

Handle a key release event.

Parameters

in	<i>_key</i>	The key that was released.
----	-------------	----------------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 640), and **gazebo::rendering::FPSViewController** (p. 368).

10.208.3.4 virtual void gazebo::rendering::ViewController::HandleMouseEvent (const common::MouseEvent & *_event*) [pure virtual]

Handle a mouse event.

Parameters

in	<i>_event</i>	The mouse position.
----	---------------	---------------------

Implemented in **gazebo::rendering::OrbitViewController** (p. 640), and **gazebo::rendering::FPSViewController** (p. 369).

10.208.3.5 `virtual void gazebo::rendering::ViewController::Init ()` [pure virtual]

Initialize the view controller.

Implemented in `gazebo::rendering::OrbitViewController` (p. 640), and `gazebo::rendering::FPSViewController` (p. 369).

10.208.3.6 `virtual void gazebo::rendering::ViewController::Init (const math::Vector3 & _focalPoint)` [virtual]

Initialize with a focus point.

Parameters

<code>in</code>	<code>_focalPoint</code>	The point to look at.
-----------------	--------------------------	-----------------------

Reimplemented in `gazebo::rendering::OrbitViewController` (p. 641).

10.208.3.7 `void gazebo::rendering::ViewController::SetEnabled (bool _value)`

Set whether the controller is enabled.

Parameters

<code>in</code>	<code>_value</code>	True if the controller is enabled.
-----------------	---------------------	------------------------------------

10.208.3.8 `virtual void gazebo::rendering::ViewController::Update ()` [pure virtual]

Update the controller, which should update the position of the **Camera** (p. 233).

Implemented in `gazebo::rendering::OrbitViewController` (p. 642), and `gazebo::rendering::FPSViewController` (p. 369).

10.208.4 Member Data Documentation

10.208.4.1 `UserCameraPtr gazebo::rendering::ViewController::camera` [protected]

Pointer to the camera to control.

10.208.4.2 `bool gazebo::rendering::ViewController::enabled` [protected]

True if enabled.

10.208.4.3 `std::string gazebo::rendering::ViewController::typeString` [protected]

Type of view controller.

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

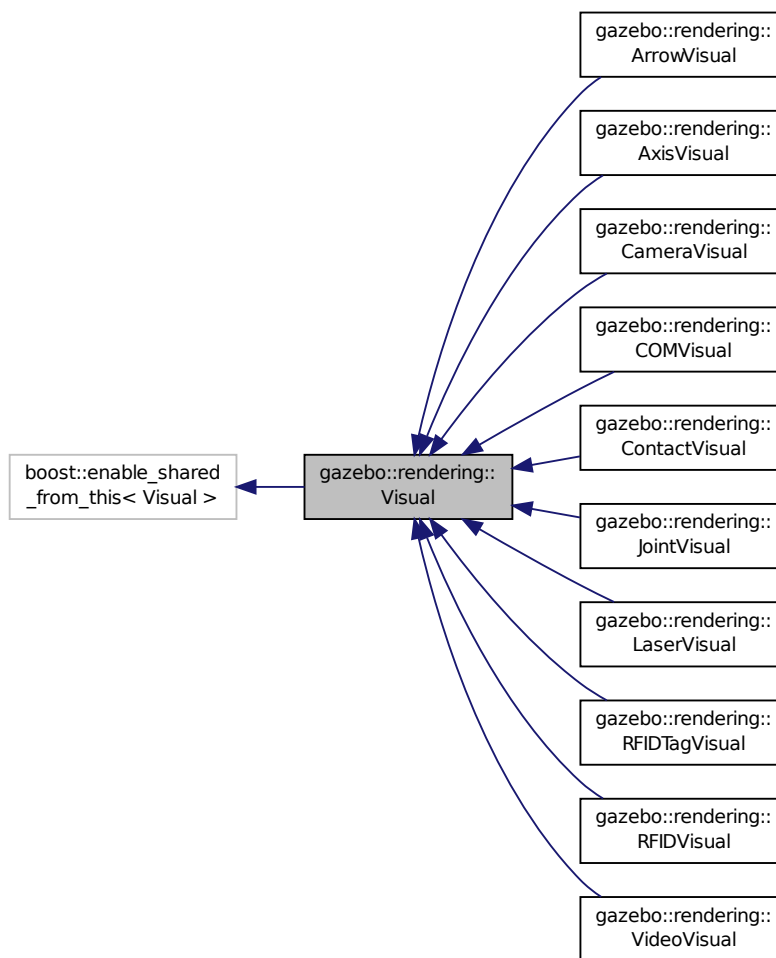
- `ViewController.hh`

10.209 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 reusable object to the visual.
- void **AttachVisual** (**VisualPtr** _vis)

Attach a visual to this visual.
- void **ClearParent** ()

Clear parents.
- **VisualPtr Clone** (const std::string &_name, **VisualPtr** _newParent)

Clone the visual with a new name.
- **DynamicLines** * **CreateDynamicLine** (**RenderOpType** _type=RENDERING_LINE_STRIP)

Add a line to the visual.
- void **DeleteDynamicLine** (**DynamicLines** *_line)

Delete a dynamic line.
- void **DetachObjects** ()

Detach all objects.
- void **DetachVisual** (**VisualPtr** _vis)

Detach a visual.
- void **DetachVisual** (const std::string &_name)

Detach a visual.
- void **DisableTrackVisual** ()

Disable tracking of a visual.
- void **EnableTrackVisual** (**VisualPtr** _vis)

Set one visual to track/follow another.
- void **Fini** ()

Helper for the destructor.
- unsigned int **GetAttachedObjectCount** () const

Return the number of attached movable objects.
- **math::Box GetBoundingBox** () const

Get the bounding box for the visual.
- **VisualPtr GetChild** (unsigned int _index)

Get an attached visual based on an index.
- unsigned int **GetChildCount** ()

Get the number of attached visuals.
- 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. 118).*
- **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)
- void **SetMaterial** (const std::string &_materialName, bool _unique=true)
Set the material.
- void **SetName** (const std::string &_name)
Set the name of the visual.
- void **SetNormalMap** (const std::string &nmap)
Set the normal map.
- void **SetPose** (const **math::Pose** &_pose)
Set the pose of the visual.
- void **SetPosition** (const **math::Vector3** &_pos)
Set the position of the visual.
- void **SetRibbonTrail** (bool _value, const **common::Color** &_initialColor, const **common::Color** &_change-Color)
True on or off a ribbon trail.
- void **SetRotation** (const **math::Quaternion** &_rot)
Set the rotation of the visual.
- void **SetScale** (const **math::Vector3** &_scale)
Set the scale.
- void **SetScene** (**ScenePtr** _scene)
Set current scene.
- void **SetShaderType** (const std::string &_type)
Set the shader type for the visual's material.
- void **SetSkeletonPose** (const msgs::PoseAnimation &_pose)
Set animation skeleton pose.
- void **SetSpecular** (const **common::Color** &_color)
Set the specular color of the visual.
- void **SetTransparency** (float _trans)
Set the transparency.
- void **SetVisibilityFlags** (uint32_t _flags)
Set visibility flags for this visual and all children.
- void **SetVisible** (bool _visible, bool _cascade=true)
Set whether the visual is visible.
- void **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. 118).*

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. 118).*

10.209.1 Detailed Description

A renderable object.

10.209.2 Constructor & Destructor Documentation

10.209.2.1 gazebo::rendering::Visual::Visual (const std::string & _name, VisualPtr _parent, bool _useRTShader = true)

Constructor.

Parameters

in	<code>_name</code>	Name of the visual.
in	<code>_parent</code>	Parent of the visual.
in	<code>_useRTShader</code>	True if the visual should use the real-time shader system (RTShader).

10.209.2.2 `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. 746) containing the visual.
in	<code>_useRTShader</code>	True if the visual should use the real-time shader system (RTShader).

10.209.2.3 `virtual gazebo::rendering::Visual::~Visual () [virtual]`

Destructor.

10.209.3 Member Function Documentation

10.209.3.1 `void gazebo::rendering::Visual::AttachAxes ()`

Attach visualization axes.

10.209.3.2 `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.

10.209.3.3 `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.

10.209.3.4 `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.
----	-------------------	---

10.209.3.5 void gazebo::rendering::Visual::AttachVisual (VisualPtr _vis)

Attach a visual to this visual.

Parameters

in	_vis	Visual (p. 931) to attach.
----	------	----------------------------

10.209.3.6 void gazebo::rendering::Visual::ClearParent ()

Clear parents.

10.209.3.7 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. 931).
in	_newParent	Parent of the cloned Visual (p. 931).

Returns

The visual.

10.209.3.8 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.

10.209.3.9 void gazebo::rendering::Visual::DeleteDynamicLine (DynamicLines * _line)

Delete a dynamic line.

Parameters

in	_line	Pointer to the line to delete.
----	-------	--------------------------------

10.209.3.10 void gazebo::rendering::Visual::DetachObjects ()

Detach all objects.

10.209.3.11 `void gazebo::rendering::Visual::DetachVisual (VisualPtr _vis)`

Detach a visual.

Parameters

<code>in</code>	<code>_vis</code>	Visual (p. 931) to detach.
-----------------	-------------------	-----------------------------------

10.209.3.12 `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.
-----------------	--------------------	-------------------------------

10.209.3.13 `void gazebo::rendering::Visual::DisableTrackVisual ()`

Disable tracking of a visual.

10.209.3.14 `void gazebo::rendering::Visual::EnableTrackVisual (VisualPtr _vis)`

Set one visual to track/follow another.

Parameters

<code>in</code>	<code>_vis</code>	Visual (p. 931) to track.
-----------------	-------------------	----------------------------------

10.209.3.15 `void gazebo::rendering::Visual::Fini ()`

Helper for the destructor.

10.209.3.16 `unsigned int gazebo::rendering::Visual::GetAttachedObjectCount () const`

Return the number of attached movable objects.

Returns

The number of attached movable objects.

10.209.3.17 `math::Box gazebo::rendering::Visual::GetBoundingBox () const`

Get the bounding box for the visual.

Returns

The bounding box in world coordinates.

10.209.3.18 `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. 939).

Parameters

<code>in</code>	<code>_index</code>	Index of the child to retrieve.
-----------------	---------------------	---------------------------------

Returns

Pointer to the child visual, NULL if index is invalid.

10.209.3.19 `unsigned int gazebo::rendering::Visual::GetChildCount ()`

Get the number of attached visuals.

Returns

The number of children.

10.209.3.20 `std::string gazebo::rendering::Visual::GetMaterialName () const`

Get the name of the material.

Returns

The name of the visual applied to this visual.

10.209.3.21 `std::string gazebo::rendering::Visual::GetName () const`

Get the name of the visual.

Returns

The name of the visual.

10.209.3.22 `std::string gazebo::rendering::Visual::GetNormalMap () const`

Get the normal map.

Returns

The name of the normal map material.

10.209.3.23 `VisualPtr gazebo::rendering::Visual::GetParent () const`

Get the parent visual, if one exists.

Returns

Pointer to the parent visual, NULL if no parent.

10.209.3.24 `math::Pose gazebo::rendering::Visual::GetPose () const`

Get the pose of the visual.

Returns

The **Visual** (p. 931)'s pose.

10.209.3.25 `math::Vector3 gazebo::rendering::Visual::GetPosition () const`

Get the position of the visual.

Returns

The visual's position.

10.209.3.26 `VisualPtr gazebo::rendering::Visual::GetRootVisual ()`

Get the root visual.

Returns

The root visual, which is one level below the world visual.

10.209.3.27 `math::Quaternion gazebo::rendering::Visual::GetRotation () const`

Get the rotation of the visual.

Returns

The visual's rotation.

10.209.3.28 `math::Vector3 gazebo::rendering::Visual::GetScale ()`

Get the scale.

Returns

The scaling factor.

10.209.3.29 `ScenePtr gazebo::rendering::Visual::GetScene () const`

Get current.

Returns

Pointer to the scene.

10.209.3.30 `Ogre::SceneNode* gazebo::rendering::Visual::GetSceneNode () const`

Return the scene Node of this visual entity.

Returns

The **Ogre** (p. 118) scene node.

10.209.3.31 `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".

10.209.3.32 `float gazebo::rendering::Visual::GetTransparency ()`

Get the transparency.

Returns

The transparency.

10.209.3.33 `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. 1129)

GZ_VISIBILITY_GUI (p. 1129)

GZ_VISIBILITY_NOT_SELECTABLE (p. 1130)

10.209.3.34 `bool gazebo::rendering::Visual::GetVisible () const`

Get whether the visual is visible.

Returns

True if the visual is visible.

10.209.3.35 `math::Pose gazebo::rendering::Visual::GetWorldPose () const`

Get the global pose of the node.

Returns

The pose in the world coordinate frame.

10.209.3.36 `bool gazebo::rendering::Visual::HasAttachedObject (const std::string & _name)`

Returns true if an object with `_name` is attached.

Parameters

<code>in</code>	<code>_name</code>	Name of an object to find.
-----------------	--------------------	----------------------------

10.209.3.37 `void gazebo::rendering::Visual::Init ()`

Helper for the constructor.

10.209.3.38 `void gazebo::rendering::Visual::InsertMesh (const std::string & _meshName)`

Insert a mesh into **Ogre** (p. 118).

Parameters

<code>in</code>	<code>_meshName</code>	Name of the mesh to insert.
-----------------	------------------------	-----------------------------

10.209.3.39 `static void gazebo::rendering::Visual::InsertMesh (const common::Mesh * _mesh) [static]`

Insert a mesh into **Ogre** (p. 118).

Parameters

<code>in</code>	<code>_mesh</code>	Pointer to the mesh to insert.
-----------------	--------------------	--------------------------------

10.209.3.40 `bool gazebo::rendering::Visual::IsPlane () const`

Return true if the visual is a plane.

Returns

True if a plane.

10.209.3.41 `bool gazebo::rendering::Visual::IsStatic () const`

Return true if the visual is a static geometry.

Returns

True if the visual is static.

10.209.3.42 void gazebo::rendering::Visual::Load (sdf::ElementPtr *_sdf*)

Load the visual with a set of parameters.

Parameters

in	<i>_sdf</i>	Load from an SDF element.
----	-------------	---------------------------

10.209.3.43 virtual void gazebo::rendering::Visual::Load () [virtual]

Load the visual with default parameters.

Reimplemented in **gazebo::rendering::ArrowVisual** (p. 140), and **gazebo::rendering::AxisVisual** (p. 142).

10.209.3.44 void gazebo::rendering::Visual::LoadFromMsg (ConstVisualPtr & *_msg*)

Load from a message.

Parameters

in	<i>_msg</i>	A visual message.
----	-------------	-------------------

10.209.3.45 void gazebo::rendering::Visual::LoadPlugin (const std::string & *_filename*, const std::string & *_name*, sdf::ElementPtr *_sdf*)

Load a plugin.

Parameters

<i>_filename</i>	The filename of the plugin
<i>_name</i>	A unique name for the plugin
<i>_sdf</i>	The SDF to pass into the plugin.

10.209.3.46 void gazebo::rendering::Visual::MakeStatic ()

Make the visual objects static renderables.

10.209.3.47 void gazebo::rendering::Visual::MoveToPosition (const math::Pose & *_pose*, double *_time*)

Move to a pose and over a given time.

Parameters

in	<i>_pose</i>	Pose the visual will end at.
in	<i>_time</i>	Time it takes the visual to move to the pose.

10.209.3.48 `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

<code>in</code>	<code>_poses</code>	Series of poses the visual will move to.
<code>in</code>	<code>_time</code>	Time it takes the visual to move to the pose.
<code>in</code>	<code>_onComplete</code>	Callback used when the move is complete.

10.209.3.49 `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
--------------------	---

10.209.3.50 `void gazebo::rendering::Visual::SetAmbient (const common::Color & _color)`

Set the ambient color of the visual.

Parameters

<code>in</code>	<code>_color</code>	The ambient color.
-----------------	---------------------	--------------------

10.209.3.51 `void gazebo::rendering::Visual::SetCastShadows (bool _shadows)`

Set whether the visual should cast shadows.

Parameters

<code>in</code>	<code>_shadows</code>	True to enable shadows.
-----------------	-----------------------	-------------------------

10.209.3.52 `void gazebo::rendering::Visual::SetDiffuse (const common::Color & _color)`

Set the diffuse color of the visual.

Parameters

<code>in</code>	<code>_color</code>	Set the diffuse color.
-----------------	---------------------	------------------------

10.209.3.53 `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. 448).

10.209.3.54 void gazebo::rendering::Visual::SetHighlighted (bool *_highlighted*)

10.209.3.55 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.

10.209.3.56 void gazebo::rendering::Visual::SetName (const std::string & *_name*)

Set the name of the visual.

Parameters

in	<code>_name</code>	Name of the visual
----	--------------------	--------------------

10.209.3.57 void gazebo::rendering::Visual::SetNormalMap (const std::string & *nmap*)

Set the normal map.

Parameters

in	<code>_nmap</code>	Name of the normal map material.
----	--------------------	----------------------------------

10.209.3.58 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.
----	--------------------	-----------------------------

10.209.3.59 void gazebo::rendering::Visual::SetPosition (const math::Vector3 & *_pos*)

Set the position of the visual.

Parameters

in	<code>_pos</code>	The position to set the visual to.
----	-------------------	------------------------------------

10.209.3.60 `void gazebo::rendering::Visual::SetRibbonTrail (bool _value, const common::Color & _initialColor, const common::Color & _changeColor)`

True on or off a ribbon trail.

Parameters

<code>in</code>	<code><i>_value</i></code>	True to enable ribbon trail.
<code>in</code>	<code><i>_initialColor</i></code>	The initial color of the ribbon trail.
<code>in</code>	<code><i>_changeColor</i></code>	Color to change too as the trail grows.

10.209.3.61 `void gazebo::rendering::Visual::SetRotation (const math::Quaternion & _rot)`

Set the rotation of the visual.

Parameters

<code>in</code>	<code><i>_rot</i></code>	The rotation of the visual.
-----------------	--------------------------	-----------------------------

10.209.3.62 `void gazebo::rendering::Visual::SetScale (const math::Vector3 & _scale)`

Set the scale.

Parameters

<code>in</code>	<code><i>_scale</i></code>	The scaling factor for the visual.
-----------------	----------------------------	------------------------------------

10.209.3.63 `void gazebo::rendering::Visual::SetScene (ScenePtr _scene)`

Set current scene.

Parameters

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

10.209.3.64 `void gazebo::rendering::Visual::SetShaderType (const std::string & _type)`

Set the shader type for the visual's material.

Parameters

<code>in</code>	<code><i>_type</i></code>	Shader type string: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".
-----------------	---------------------------	---

10.209.3.65 `void gazebo::rendering::Visual::SetSkeletonPose (const msgs::PoseAnimation & _pose)`

Set animation skeleton pose.

Parameters

<i>in</i>	<i>_pose</i>	Skelton message
-----------	--------------	-----------------

10.209.3.66 `void gazebo::rendering::Visual::SetSpecular (const common::Color & _color)`

Set the specular color of the visual.

Parameters

<i>in</i>	<i>_color</i>	Specular color.
-----------	---------------	-----------------

10.209.3.67 `void gazebo::rendering::Visual::SetTransparency (float _trans)`

Set the transparency.

Parameters

<i>in</i>	<i>_trans</i>	The transparency, between 0 and 1 where 0 is no transparency.
-----------	---------------	---

10.209.3.68 `void gazebo::rendering::Visual::SetVisibilityFlags (uint32_t _flags)`

Set visibility flags for this visual and all children.

Parameters

<i>in</i>	<i>_flags</i>	The visibility flags.
-----------	---------------	-----------------------

See Also

GZ_VISIBILITY_ALL (p. 1129)

GZ_VISIBILITY_GUI (p. 1129)

GZ_VISIBILITY_NOT_SELECTABLE (p. 1130)

10.209.3.69 `void gazebo::rendering::Visual::SetVisible (bool _visible, bool _cascade = true)`

Set whether the visual is visible.

Parameters

<i>in</i>	<i>_visible</i>	set this node visible.
<i>in</i>	<i>_cascade</i>	setting this parameter in children too.

10.209.3.70 `void gazebo::rendering::Visual::SetWorldPose (const math::Pose _pose)`

Set the world pose of the visual.

Parameters

<code>in</code>	<code>_pose</code>	Pose of the visual in the world coordinate frame.
-----------------	--------------------	---

10.209.3.71 `void gazebo::rendering::Visual::SetWorldPosition (const math::Vector3 & _pos)`

Set the world linear position of the visual.

Parameters

<code>in</code>	<code>_pose</code>	Position in the world coordinate frame.
-----------------	--------------------	---

10.209.3.72 `void gazebo::rendering::Visual::SetWorldRotation (const math::Quaternion & _rot)`

Set the world orientation of the visual.

Parameters

<code>in</code>	<code>_rot</code>	Rotation in the world coordinate frame.
-----------------	-------------------	---

10.209.3.73 `void gazebo::rendering::Visual::ShowBoundingBox ()`

Display the bounding box visual.

10.209.3.74 `void gazebo::rendering::Visual::ShowCollision (bool _show)`

Display the collision visuals.

Parameters

<code>in</code>	<code>_show</code>	True to show visuals labeled as collision objects.
-----------------	--------------------	--

10.209.3.75 `void gazebo::rendering::Visual::ShowCOM (bool _show)`

Display Center of Mass visuals.

Parameters

<code>in</code>	<code>_show</code>	True to show center of mass visualizations.
-----------------	--------------------	---

10.209.3.76 `void gazebo::rendering::Visual::ShowJoints (bool _show)`

Display joint visuals.

Parameters

<code>in</code>	<code>_show</code>	True to show joint visualizations.
-----------------	--------------------	------------------------------------

10.209.3.77 void gazebo::rendering::Visual::ShowSkeleton (bool *_show*)

Display the skeleton visuals.

Parameters

in	<i>_show</i>	True to show skeleton visuals.
----	--------------	--------------------------------

10.209.3.78 void gazebo::rendering::Visual::ToggleVisible ()

Toggle whether this visual is visible.

10.209.3.79 void gazebo::rendering::Visual::Update ()

Update the visual.

10.209.3.80 void gazebo::rendering::Visual::UpdateFromMsg (ConstVisualPtr & *_msg*)

Update a visual based on a message.

Parameters

in	<i>_msg</i>	The visual message.
----	-------------	---------------------

10.209.4 Member Data Documentation

10.209.4.1 VisualPtr gazebo::rendering::Visual::parent [protected]

Parent visual.

10.209.4.2 ScenePtr gazebo::rendering::Visual::scene [protected]

Pointer to the visual's scene.

10.209.4.3 Ogre::SceneNode* gazebo::rendering::Visual::sceneNode [protected]

Pointer to the visual's scene node in **Ogre** (p. 118).

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

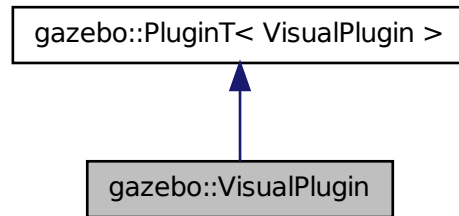
- **Visual.hh**

10.210 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

10.210.1 Detailed Description

A plugin loaded within the gzserver on startup.

See [reference](#).

10.210.2 Constructor & Destructor Documentation

10.210.2.1 gazebo::VisualPlugin::VisualPlugin () [inline]

References [gazebo::PluginT< VisualPlugin >::type](#), and [gazebo::VISUAL_PLUGIN](#).

10.210.3 Member Function Documentation

10.210.3.1 virtual void gazebo::VisualPlugin::Init () [inline], [virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.210.3.2 `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.

10.210.3.3 `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:

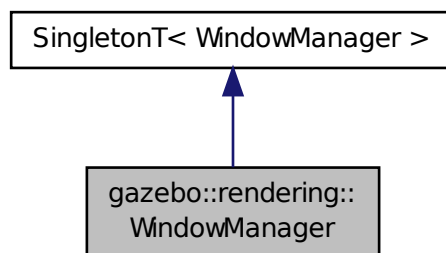
- `common/Plugin.hh`

10.211 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. 118) 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

10.211.1 Detailed Description

Class to manage render windows.

10.211.2 Member Function Documentation

10.211.2.1 int gazebo::rendering::WindowManager::CreateWindow (const std::string & _ogreHandle, uint32_t _width, uint32_t _height)

Create a window.

Parameters

in	<code>_ogreHandle</code>	String representing the ogre window handle.
in	<code>_width</code>	Width of the window in pixels.
in	<code>_height</code>	Height of the window in pixels.

10.211.2.2 void gazebo::rendering::WindowManager::Fini ()

Shutdown all the windows.

10.211.2.3 float gazebo::rendering::WindowManager::GetAvgFPS (uint32_t _id)

Get the average FPS.

Parameters

in	<code>_id</code>	ID of the window.
----	------------------	-------------------

Returns

The frames per second.

10.211.2.4 `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.

10.211.2.5 `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.

10.211.2.6 `void gazebo::rendering::WindowManager::Moved (uint32_t _id)`

Tells **Ogre** (p. 118) the window has moved, and needs updating.

Parameters

<code>in</code>	<code>_id</code>	ID of the window.
-----------------	------------------	-------------------

10.211.2.7 `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.

10.211.2.8 `void gazebo::rendering::WindowManager::SetCamera (int _windowId, CameraPtr _camera)`

Attach a camera to a window.

Parameters

<code>in</code>	<code>_windowId</code>	Id of the window to add the camera to.
<code>in</code>	<code>_camera</code>	Pointer to the camera to attach.

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

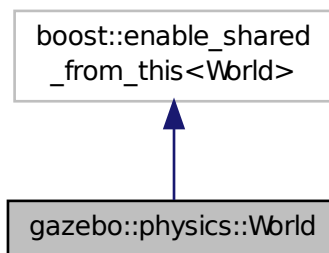
- **WindowManager.hh**

10.212 gazebo::physics::World Class Reference

The world provides access to all other object within a simulated environment.

```
#include <physics/World.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** ()
Finilize 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. 338) 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**
Get a pointer to a entity based on a name.
 - **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**
Get a model by name **DEPRECATED**.
 - unsigned int **GetModelCount** () const
Get the number of models.
 - std::list< **ModelPtr** > **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. 338).
 - 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 **World::GetRealTime** (p. 961).
 - **common::Time GetStartTime** () const
Get the wall time simulation was started.
 - **WorldState GetState** ()
Get the current world state.
 - 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. 338) 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** (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.

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. 348) on the **physics::Link** (p. 454) in **World::Update**.*

10.212.1 Detailed Description

The world provides access to all other object within a simulated environment.

The **World** (p. 954) is the container for all models and their components (links, joints, sensors, plugins, etc), and **World-Plugin** (p. 965) instances. Many core function are also handled in the **World** (p. 954), including physics update, model updates, and message processing.

10.212.2 Constructor & Destructor Documentation

10.212.2.1 gazebo::physics::World::World (const std::string & *_name* = " ")

Constructor.

Constructor for the **World** (p. 954). Must specify a unique name.

Parameters

<i>_name</i>	Name of the world
--------------	-------------------

10.212.2.2 gazebo::physics::World::~~World ()

Destructor.

10.212.3 Member Function Documentation

10.212.3.1 void gazebo::physics::World::Clear ()

Remove all entities from the world.

10.212.3.2 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.

10.212.3.3 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.

10.212.3.4 void gazebo::physics::World::EnablePhysicsEngine (bool *_enable*) [inline]

enable/disable physics engine during World::Update

Parameters

<i>in</i>	<i>_enable</i>	True to enable the physics engine.
-----------	----------------	------------------------------------

10.212.3.5 void gazebo::physics::World::Fini ()

Finalize the world.

Call this function to tear-down the world.

10.212.3.6 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

<i>in</i>	<i>_name</i>	The name of the Model (p. 521) to find.
-----------	--------------	--

Returns

A pointer to the entity, or NULL if no entity was found.

10.212.3.7 bool gazebo::physics::World::GetEnablePhysicsEngine () [inline]

check if physics engine is enabled/disabled.

Parameters

<i>True</i>	if the physics engine is enabled.
-------------	-----------------------------------

10.212.3.8 EntityPtr gazebo::physics::World::GetEntity (const std::string & *_name*)

Get a pointer to an **Entity** (p. 338) based on a name.

This function is the same as GetByName, but limits the search to only Entities.

Parameters

<i>in</i>	<i>_name</i>	The name of the Entity (p. 338) to find.
-----------	--------------	---

Returns

A pointer to the **Entity** (p. 338), or NULL if no **Entity** (p. 338) was found.

10.212.3.9 EntityPtr gazebo::physics::World::GetEntityBelowPoint (const math::Vector3 & *_pt*)

Get the nearest entity below a point.

Projects a Ray down (-Z axis) starting at the given point. The first entity hit by the Ray is returned.

Parameters

<i>in</i>	<i>_pt</i>	The 3D point to search below
-----------	------------	------------------------------

Returns

A pointer to nearest **Entity** (p. 338), NULL if none is found.

10.212.3.10 EntityPtr gazebo::physics::World::GetEntityByName (const std::string & *_name*)

Get a pointer to a entity based on a name.

Parameters

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

10.212.3.11 ModelPtr gazebo::physics::World::GetModel (unsigned int *_index*) const

Get a model based on an index.

Get a **Model** (p. 521) using an index, where index must be greater than zero and less than **World::GetModelCount()** (p. 960)

Parameters

<i>in</i>	<i>_index</i>	The index of the model [0..GetModelCount)
-----------	---------------	---

Returns

A pointer to the **Model** (p. 521). NULL if *_index* is invalid.

10.212.3.12 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

<i>in</i>	<i>_name</i>	The name of the Model (p. 521) to find.
-----------	--------------	--

Returns

A pointer to the **Model** (p. 521), or NULL if no model was found.

10.212.3.13 ModelPtr gazebo::physics::World::GetModelBelowPoint (const math::Vector3 & *_pt*)

Get the nearest model below a point.

This function makes use of **World::GetEntityBelowPoint** (p. 958).

Parameters

<i>in</i>	<i>_pt</i>	The 3D point to search below
-----------	------------	------------------------------

Returns

A pointer to nearest **Model** (p. 521), NULL if none is found.

10.212.3.14 **ModelPtr** gazebo::physics::World::GetModelByName (const std::string & name)

Get a model by name DEPRECATED.

10.212.3.15 unsigned int gazebo::physics::World::GetModelCount () const

Get the number of models.

Returns

The number of models in the **World** (p. 954).

10.212.3.16 std::list<ModelPtr> gazebo::physics::World::GetModels () const

Get a list of all the models.

Returns

A list of all the Models in the world

10.212.3.17 std::string gazebo::physics::World::GetName () const

Get the name of the world.

Returns

The name of the world.

10.212.3.18 common::Time gazebo::physics::World::GetPauseTime () const

Get the amount of time simulation has been paused.

Returns

The pause time

10.212.3.19 **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

10.212.3.20 `common::Time gazebo::physics::World::GetRealTime () const`

Get the real time (elapsed time)

Returns

The real time

10.212.3.21 `EntityPtr gazebo::physics::World::GetSelectedEntity () const`

Get the selected **Entity** (p. 338).

The selected entity is set via the GUI.

Returns

A point to the **Entity** (p. 338), NULL if nothing is selected

10.212.3.22 `boost::mutex* gazebo::physics::World::GetSetWorldPoseMutex () const [inline]`

Get the set world pose mutex.

Returns

Pointer to the mutex.

10.212.3.23 `common::Time gazebo::physics::World::GetSimTime () const`

Get the world simulation time, note if you want the PC wall clock call **World::GetRealTime** (p. 961).

Returns

The current simulation time

10.212.3.24 `common::Time gazebo::physics::World::GetStartTime () const`

Get the wall time simulation was started.

Returns

The start time

10.212.3.25 `WorldState gazebo::physics::World::GetState ()`

Get the current world state.

Returns

A object that contains the entire state of the **World** (p. 954)

10.212.3.26 `void gazebo::physics::World::Init ()`

Initialize the world.

This is called after Load.

10.212.3.27 `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).
-----------------	---------------------------	--

10.212.3.28 `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
-----------------	-------------------	------------------------------

10.212.3.29 `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.
-----------------	-------------------------	---------------------------------------

10.212.3.30 `bool gazebo::physics::World::IsPaused () const`

Returns the state of the simulation true if paused.

Returns

True if paused.

10.212.3.31 `void gazebo::physics::World::Load (sdf::ElementPtr _sdf)`

Load the world using SDF parameters.

Load a world from and SDF pointer

Parameters

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

10.212.3.32 `void gazebo::physics::World::LoadPlugin (const std::string & _filename, const std::string & _name, sdf::ElementPtr _sdf)`

Load a plugin.

Parameters

<i>in</i>	<i>_filename</i>	The filename of the plugin
<i>in</i>	<i>_name</i>	A unique name for the plugin
<i>in</i>	<i>_sdf</i>	The SDF to pass into the plugin.

10.212.3.33 `void gazebo::physics::World::PrintEntityTree ()`

Print **Entity** (p. 338) tree.

Prints all the entities to stdout

10.212.3.34 `void gazebo::physics::World::RemovePlugin (const std::string & _name)`

Remove a running plugin.

Parameters

<i>in</i>	<i>_name</i>	The unique name of the plugin to remove
-----------	--------------	---

10.212.3.35 `void gazebo::physics::World::Reset ()`

Reset time and model poses, configurations in simulation.

10.212.3.36 `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. 149).

Parameters

<i>in</i>	<i>_type</i>	The type of reset.
-----------	--------------	--------------------

10.212.3.37 `void gazebo::physics::World::ResetTime ()`

Reset simulation time back to zero.

10.212.3.38 void gazebo::physics::World::Run ()

Run the world in a thread.

Run the update loop.

10.212.3.39 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

<i>_filename</i>	Name of the file to save into
------------------	-------------------------------

10.212.3.40 void gazebo::physics::World::SetPaused (bool *_p*)

Set whether the simulation is paused.

Parameters

<i>in</i>	<i>_p</i>	True pauses the simulation. False runs the simulation.
-----------	-----------	--

10.212.3.41 void gazebo::physics::World::SetSimTime (common::Time *_t*)

Set the sim time.

Parameters

<i>in</i>	<i>_t</i>	The new simulation time
-----------	-----------	-------------------------

10.212.3.42 void gazebo::physics::World::SetState (const WorldState & *_state*)

Set the current world state.

Parameters

<i>_state</i>	The state to set the World (p. 954) to.
---------------	--

10.212.3.43 void gazebo::physics::World::StepWorld (int *_steps*)

Step callback.

Parameters

<i>in</i>	<i>_steps</i>	The number of steps the World (p. 954) should take
-----------	---------------	---

10.212.3.44 `void gazebo::physics::World::Stop ()`

Stop the world.

Stop the update loop.

10.212.3.45 `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

10.212.4 Member Data Documentation

10.212.4.1 `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. 348) on the **physics::Link** (p. 454) in `World::Update`.

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

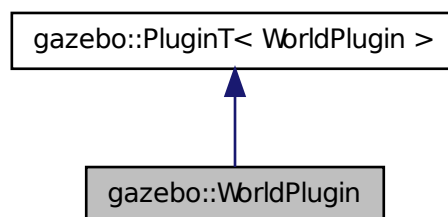
- **World.hh**

10.213 gazebo::WorldPlugin Class Reference

A plugin with access to **physics::World** (p. 954).

```
#include <Plugin.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

10.213.1 Detailed Description

A plugin with access to **physics::World** (p. 954).

See [reference](#).

10.213.2 Constructor & Destructor Documentation

10.213.2.1 gazebo::WorldPlugin::WorldPlugin () [inline]

Constructor.

References [gazebo::PluginT< WorldPlugin >::type](#), and [gazebo::WORLD_PLUGIN](#).

10.213.2.2 virtual gazebo::WorldPlugin::~~WorldPlugin () [inline],[virtual]

Destructor.

10.213.3 Member Function Documentation

10.213.3.1 virtual void gazebo::WorldPlugin::Init () [inline],[virtual]

10.213.3.2 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.

10.213.3.3 virtual void gazebo::WorldPlugin::Reset () [inline],[virtual]

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

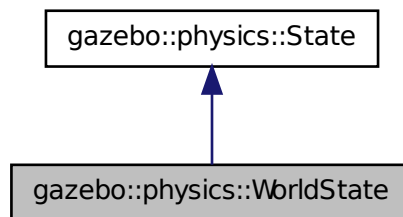
- `common/Plugin.hh`

10.214 gazebo::physics::WorldState Class Reference

Store state information of a `physics::World` (p. 954) object.

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

Inheritance diagram for `gazebo::physics::WorldState`:



Public Member Functions

- **WorldState** ()
Default constructor.
- **WorldState** (`WorldPtr` _world)
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 `sdf::ElementPtr` & **GetSDF** () const
Get the state in SDF format.
- virtual void **Load** (`sdf::ElementPtr` _elem)
Load state from SDF element.

Additional Inherited Members

10.214.1 Detailed Description

Store state information of a `physics::World` (p. 954) object.

Instances of this class contain the state of a **World** (p. 954) at a specific time. **World** (p. 954) state includes the state of all models, and their children.

10.214.2 Constructor & Destructor Documentation

10.214.2.1 gazebo::physics::WorldState::WorldState ()

Default constructor.

10.214.2.2 gazebo::physics::WorldState::WorldState (WorldPtr *_world*)

Constructor.

Generate a **WorldState** (p. 967) from an instance of a **World** (p. 954).

Parameters

<i>_world</i>	Pointer to a world
---------------	--------------------

10.214.2.3 virtual gazebo::physics::WorldState::~~WorldState () [virtual]

Destructor.

10.214.3 Member Function Documentation

10.214.3.1 ModelState gazebo::physics::WorldState::GetModelState (unsigned int *_index*) const

Get a model state.

Get the state of a **Model** (p. 521) based on an index. The min index is and the max is **WorldState::GetModelStateCount()** (p. 968)

Parameters

<i>_index</i>	Index of the model
---------------	--------------------

Returns

State (p. 813) of the requested **Model** (p. 521)

10.214.3.2 ModelState gazebo::physics::WorldState::GetModelState (const std::string & *_modelName*) const

Get a model state by model name.

10.214.3.3 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

10.214.3.4 `const sdf::ElementPtr& gazebo::physics::WorldState::GetSDF () const`

Get the state in SDF format.

Returns a pointer to the SDF representation of the **WorldState** (p. 967)

Returns

Pointer to the SDF representation of the **WorldState** (p. 967)

10.214.3.5 `virtual void gazebo::physics::WorldState::Load (sdf::ElementPtr _elem) [virtual]`

Load state from SDF element.

Set a **WorldState** (p. 967) from an SDF element containing **WorldState** (p. 967) info.

Parameters

<code>_elem</code>	Pointer to the WorldState (p. 967) SDF element.
--------------------	--

Implements **gazebo::physics::State** (p. 815).

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

- **WorldState.hh**

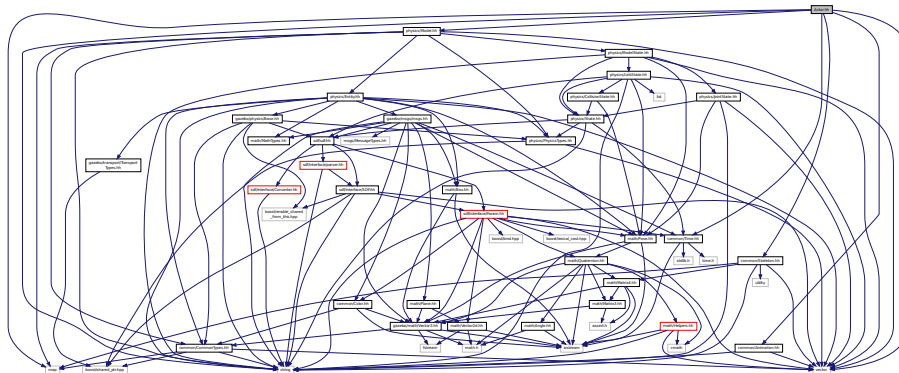
Chapter 11

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:



Classes

- class **gazebo::physics::Actor**
Actor (p. 121) class enables GPU based mesh model / skeleton scriptable animation.
- struct **gazebo::physics::TrajectoryInfo**

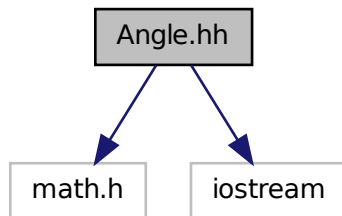
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.

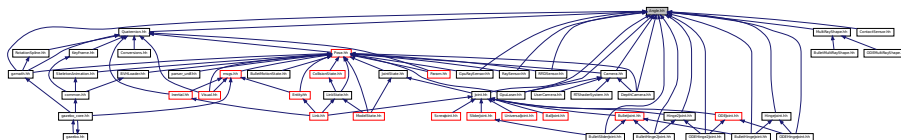
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::physics**
namespace for physics

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

in	<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

in	<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

in	<i>radians</i>	
----	----------------	--

Returns

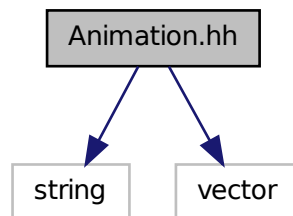
degrees

11.3 Animation.hh File Reference

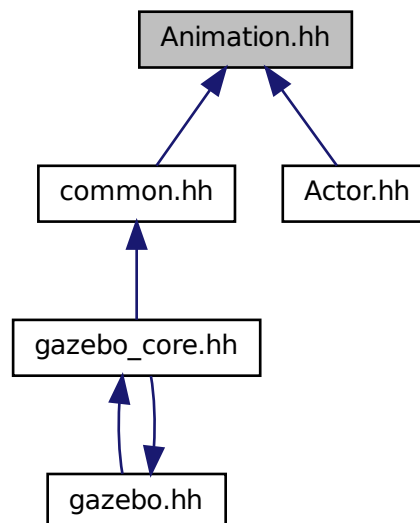
```
#include <string>
```

```
#include <vector>
```

Include dependency graph for Animation.hh:



This graph shows which files directly or indirectly include this file:



Classes

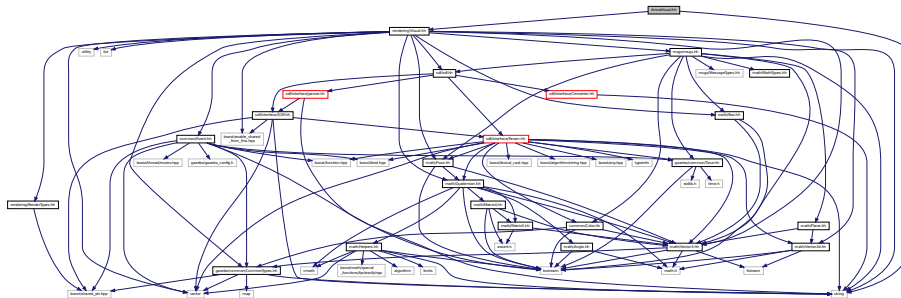
- class **gazebo::common::Animation**
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.
- class **gazebo::common::NumericAnimation**
A numeric animation.
- class **gazebo::common::PoseAnimation**
A pose animation.

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

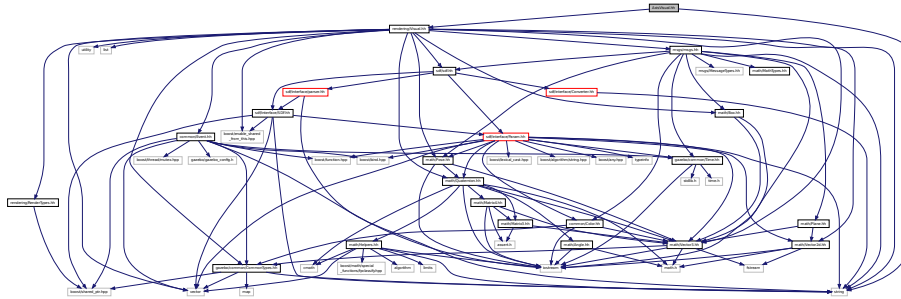
- class **gazebo::rendering::ArrowVisual**
Basic arrow visualization.

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

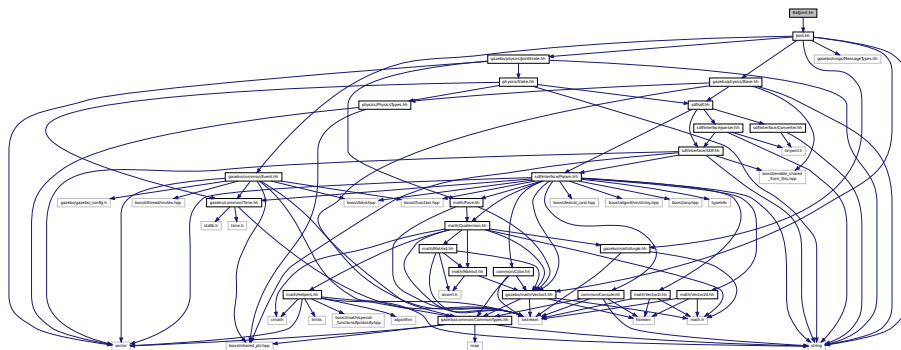
- class **gazebo::rendering::AxisVisual**
Basic axis visualization.

Namespaces

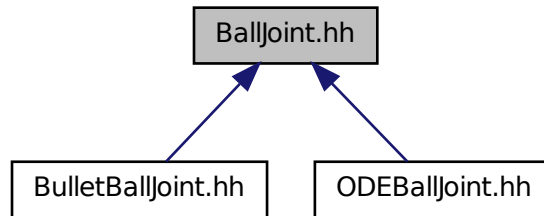
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

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. 145) class for a ball joint.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

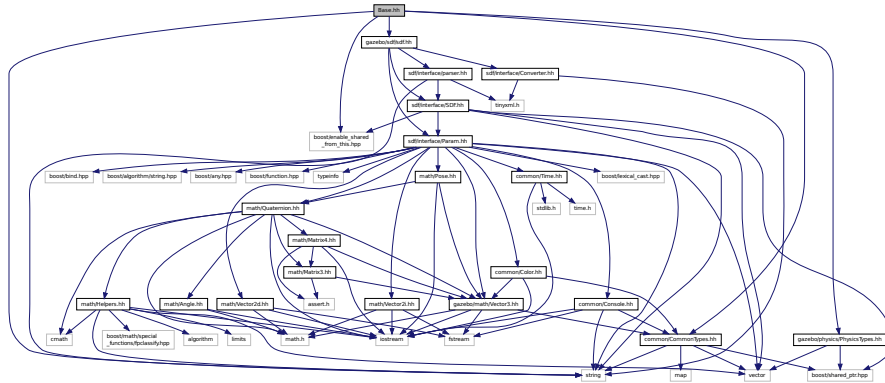
- namespace **gazebo::physics**

namespace for physics

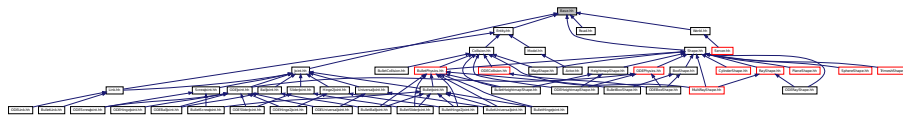
11.7 Base.hh File Reference

```
#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 Base.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Base**
Base (p. 145) class for most physics classes.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

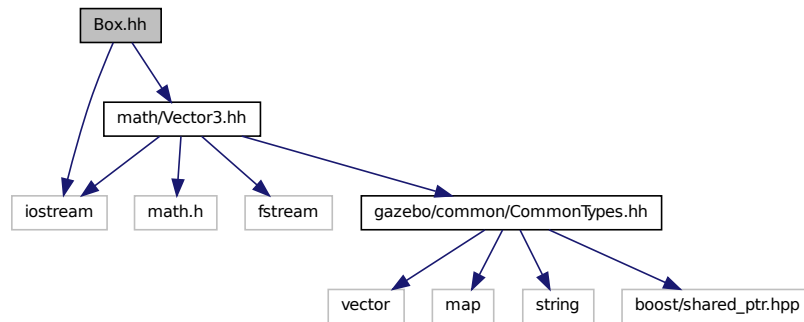
Variables

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

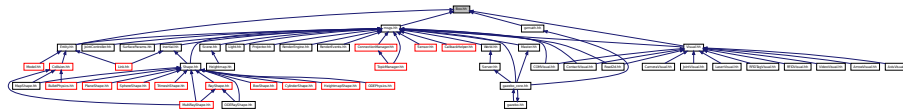
11.8 Box.hh File Reference

```
#include <iostream>
#include "math/Vector3.hh"
```

Include dependency graph for Box.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Box**

Mathematical representation of a box and related functions.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

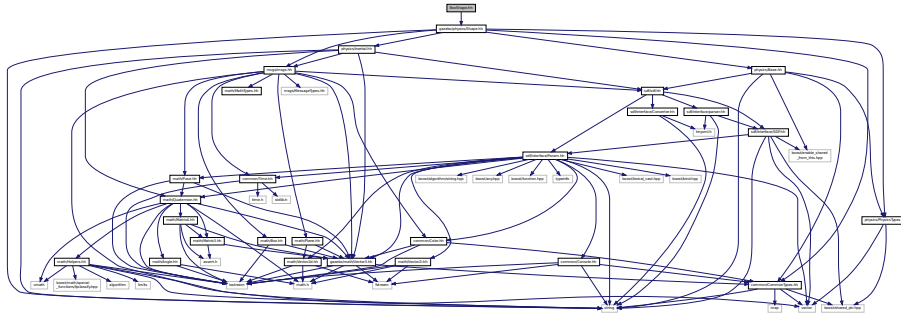
- namespace **gazebo::math**

Math namespace.

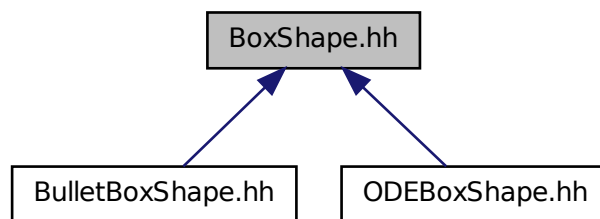
11.9 BoxShape.hh File Reference

```
#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

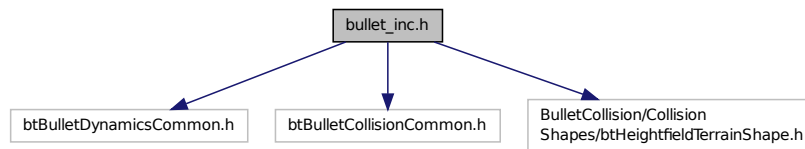
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.10 bullet_inc.h File Reference

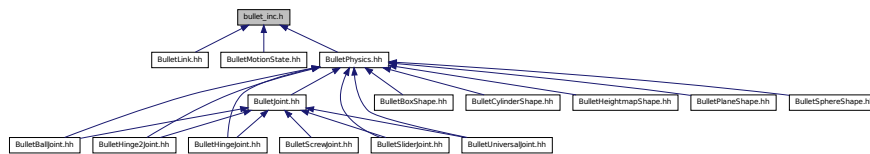
```

#include <btBulletDynamicsCommon.h>
#include <btBulletCollisionCommon.h>
#include <BulletCollision/CollisionShapes/btHeightfieldTerrainShape.h>
  
```


Include dependency graph for bullet_inc.h:



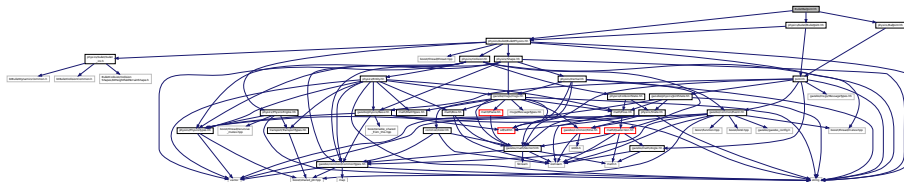
This graph shows which files directly or indirectly include this file:



11.11 BulletBallJoint.hh File Reference

```
#include "physics/BallJoint.hh"
#include "physics/bullet/BulletJoint.hh"
#include "physics/bullet/BulletPhysics.hh"
```

Include dependency graph for BulletBallJoint.hh:



Classes

- class **gazebo::physics::BulletBallJoint**
BulletBallJoint (p. 166) class models a ball joint in Bullet.

Namespaces

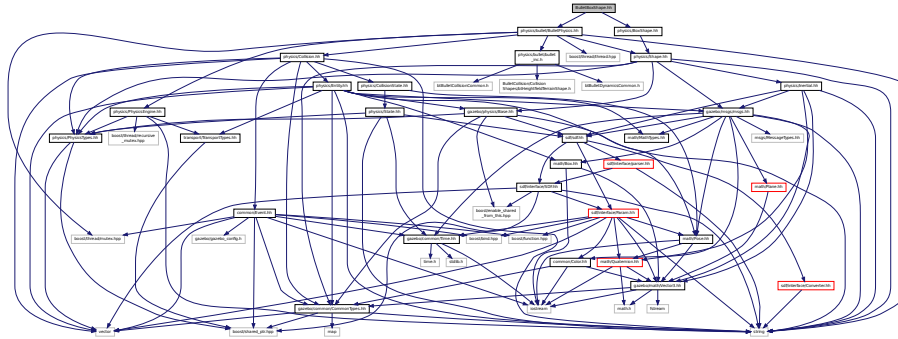
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.12 BulletBoxShape.hh File Reference

```
#include "physics/bullet/BulletPhysics.hh"
```

```
#include "physics/BoxShape.hh"
```

Include dependency graph for BulletBoxShape.hh:



Classes

- class **gazebo::physics::BulletBoxShape**

Bullet box collision.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

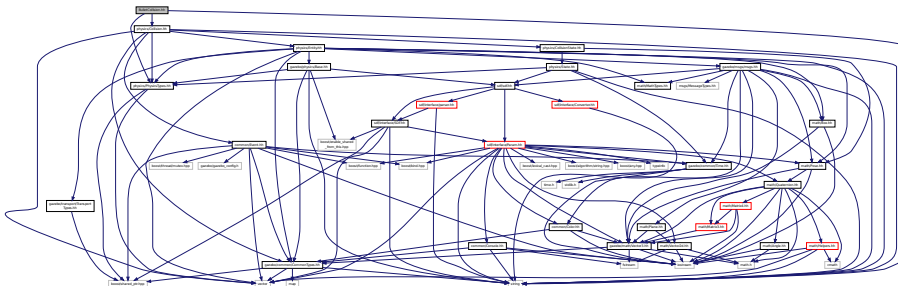
11.13 BulletCollision.hh File Reference

```
#include <string>
```

```
#include "physics/PhysicsTypes.hh"
```

```
#include "physics/Collision.hh"
```

Include dependency graph for BulletCollision.hh:



Classes

- class **gazebo::physics::BulletCollision**

Bullet collisions.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

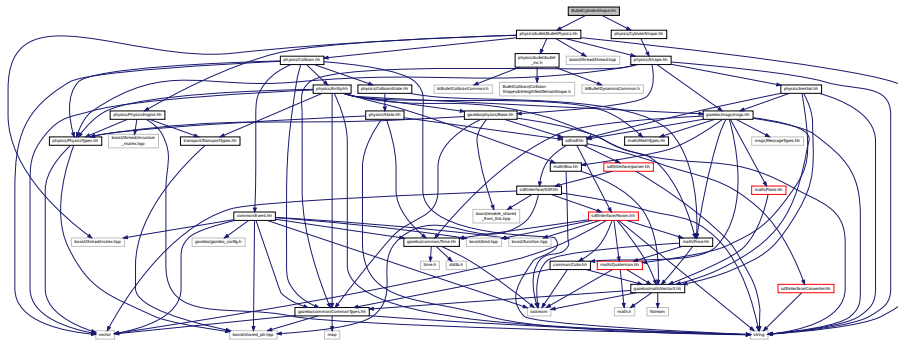
namespace for physics

11.14 BulletCylinderShape.hh File Reference

```
#include "physics/bullet/BulletPhysics.hh"
```

```
#include "physics/CylinderShape.hh"
```

Include dependency graph for BulletCylinderShape.hh:



Classes

- class **gazebo::physics::BulletCylinderShape**

Cylinder collision.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

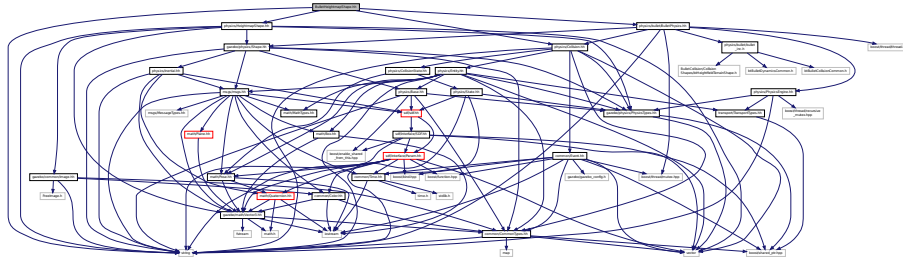
- namespace **gazebo::physics**

namespace for physics

11.15 BulletHeightmapShape.hh File Reference

```
#include <string>
#include "physics/HeightmapShape.hh"
#include "physics/bullet/BulletPhysics.hh"
#include "physics/Collision.hh"
```

Include dependency graph for BulletHeightmapShape.hh:



Classes

- class **gazebo::physics::BulletHeightmapShape**
Height map collision.

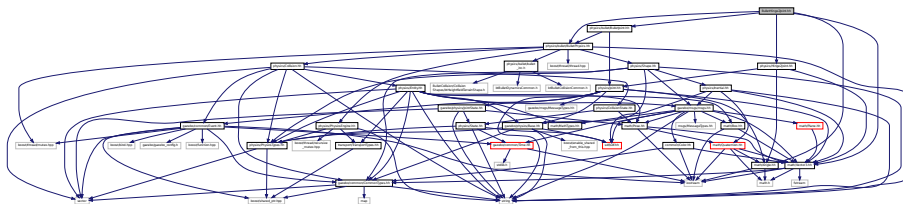
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.16 BulletHinge2Joint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/Hinge2Joint.hh"
#include "physics/bullet/BulletJoint.hh"
#include "physics/bullet/BulletPhysics.hh"
```

Include dependency graph for BulletHinge2Joint.hh:



Classes

- class **gazebo::physics::BulletHinge2Joint**
A two axis hinge joint.

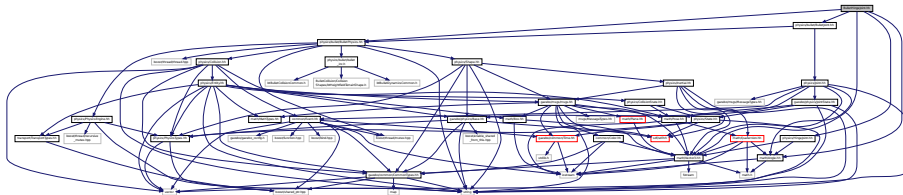
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.17 BulletHingeJoint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/HingeJoint.hh"
#include "physics/bullet/BulletJoint.hh"
#include "physics/bullet/BulletPhysics.hh"
```

Include dependency graph for BulletHingeJoint.hh:



Classes

- class **gazebo::physics::BulletHingeJoint**
A single axis hinge joint.

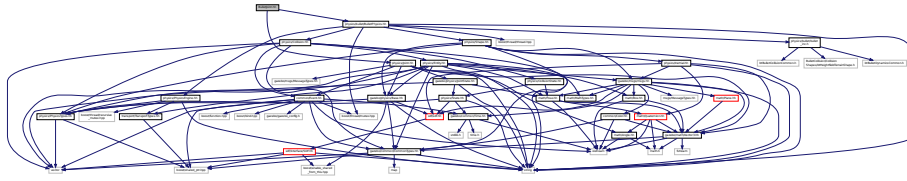
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

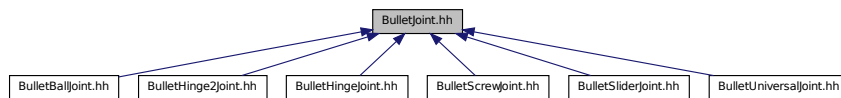
11.18 BulletJoint.hh File Reference

```
#include "physics/bullet/BulletPhysics.hh"
#include "physics/Joint.hh"
```

Include dependency graph for BulletJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::BulletJoint**

Base (p. 145) class for all joints.

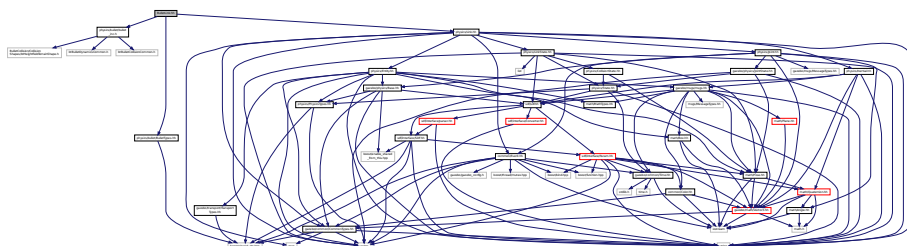
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.19 BulletLink.hh File Reference

```
#include "physics/bullet/bullet_inc.h"
#include "physics/bullet/BulletTypes.hh"
#include "physics/Link.hh"
```

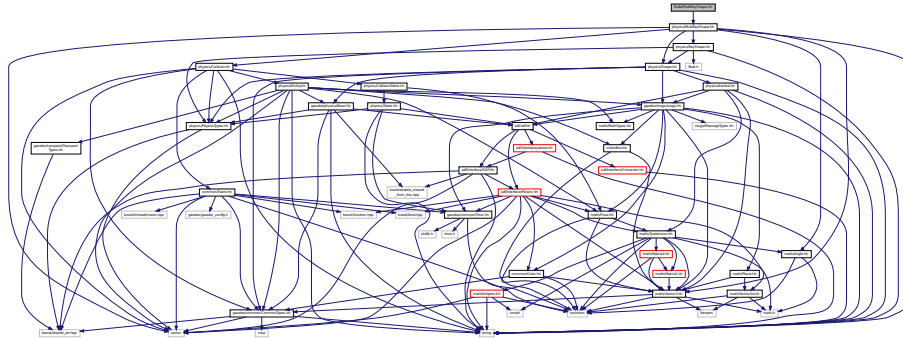
Include dependency graph for BulletLink.hh:



11.21 BulletMultiRayShape.hh File Reference

```
#include "physics/MultiRayShape.hh"
```

Include dependency graph for BulletMultiRayShape.hh:



Classes

- class **gazebo::physics::BulletMultiRayShape**
*Bullet specific version of **MultiRayShape** (p. 549).*

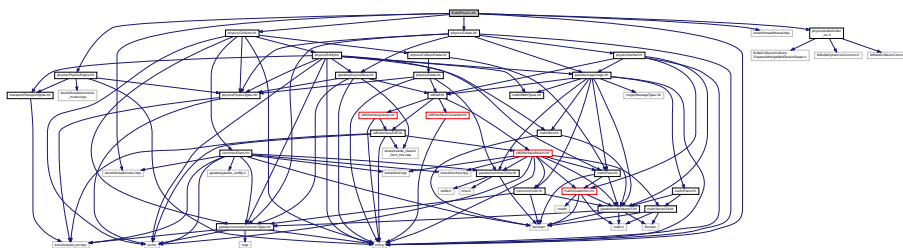
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

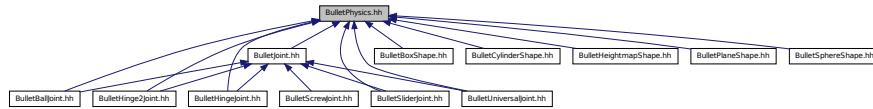
11.22 BulletPhysics.hh File Reference

```
#include <string>
#include <boost/thread/thread.hpp>
#include <boost/thread/mutex.hpp>
#include "physics/bullet/bullet_inc.h"
#include "physics/PhysicsEngine.hh"
#include "physics/Collision.hh"
#include "physics/Shape.hh"
```

Include dependency graph for BulletPhysics.hh:



This graph shows which files directly or indirectly include this file:



Classes

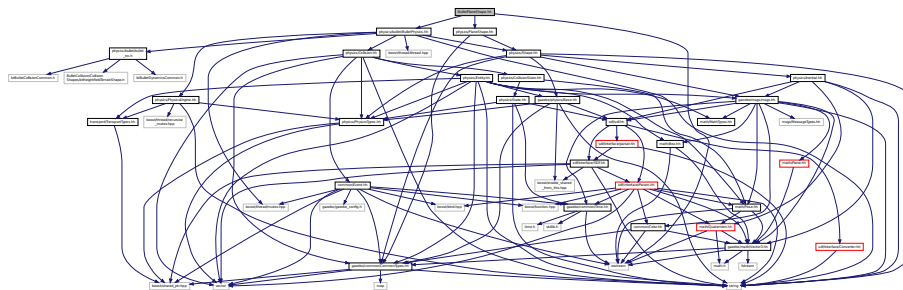
- class **gazebo::physics::BulletPhysics**
Bullet physics engine.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.23 BulletPlaneShape.hh File Reference

```
#include <iostream>
#include "physics/bullet/BulletPhysics.hh"
#include "physics/PlaneShape.hh"
Include dependency graph for BulletPlaneShape.hh:
```



Classes

- class **gazebo::physics::BulletPlaneShape**
Bullet collision for an infinite plane.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

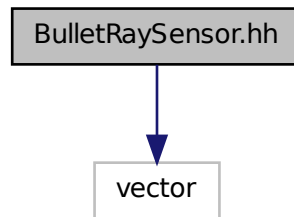
- namespace **gazebo::physics**

namespace for physics

11.24 BulletRaySensor.hh File Reference

```
#include <vector>
```

Include dependency graph for BulletRaySensor.hh:



Classes

- class **gazebo::physics::BulletRaySensor**

An Bullet Ray sensor.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

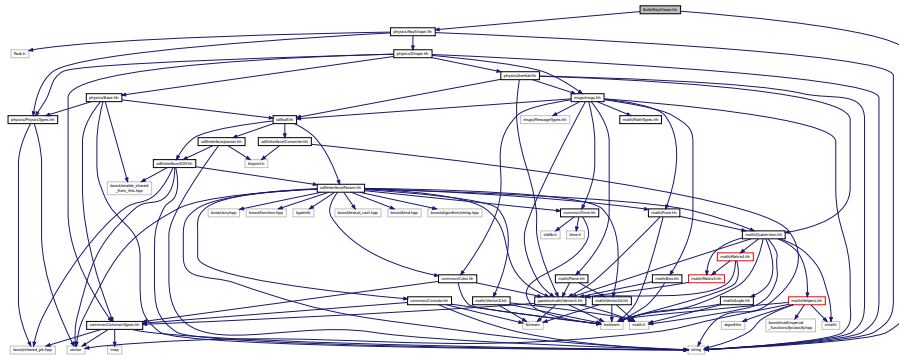
namespace for physics

11.25 BulletRayShape.hh File Reference

```
#include <string>
```

```
#include "physics/RayShape.hh"
```

Include dependency graph for BulletRayShape.hh:



Classes

- class **gazebo::physics::BulletRayShape**
Ray shape for bullet.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.26 BulletScrewJoint.hh File Reference

```
#include "physics/bullet/BulletJoint.hh"
#include "physics/ScrewJoint.hh"
```

Include dependency graph for BulletScrewJoint.hh:



Classes

- class **gazebo::physics::BulletScrewJoint**
A screw joint.

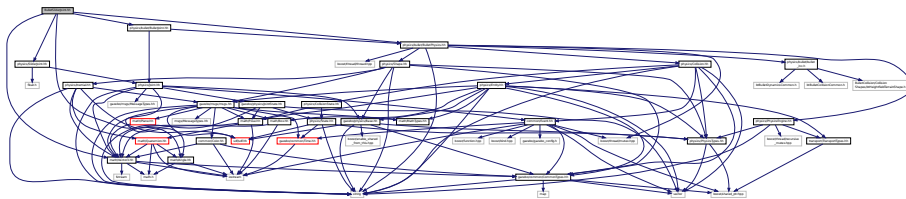
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.27 BulletSliderJoint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/bullet/BulletJoint.hh"
#include "physics/SliderJoint.hh"
#include "physics/bullet/BulletPhysics.hh"
```

Include dependency graph for BulletSliderJoint.hh:



Classes

- class **gazebo::physics::BulletSliderJoint**
A slider joint.

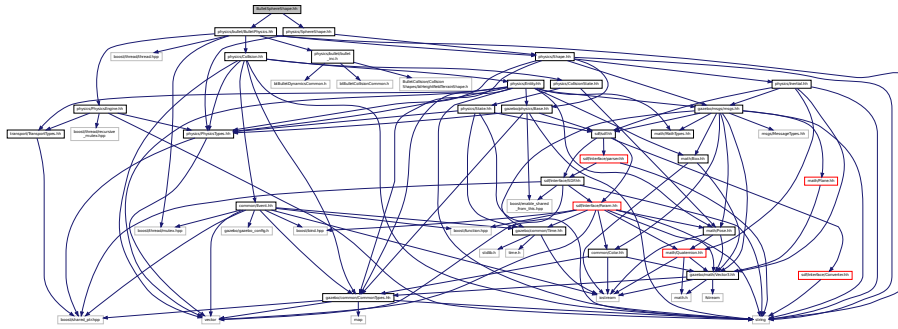
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.28 BulletSphereShape.hh File Reference

```
#include "physics/bullet/BulletPhysics.hh"
#include "physics/SphereShape.hh"
```

Include dependency graph for BulletSphereShape.hh:



Classes

- class **gazebo::physics::BulletSphereShape**
Bullet sphere collision.

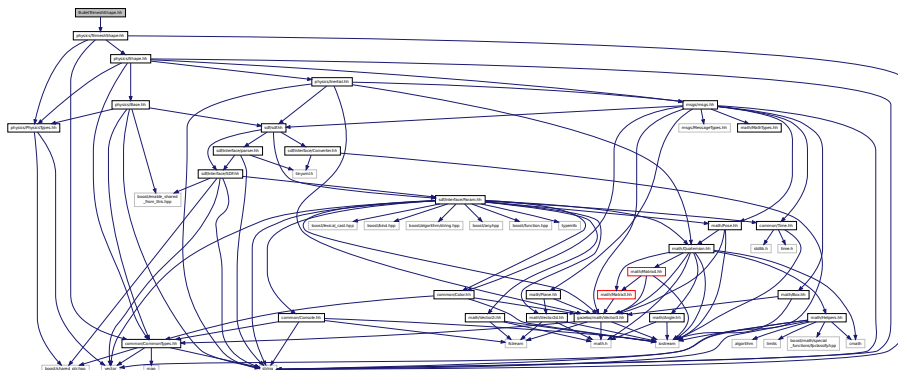
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.29 BulletTrimeshShape.hh File Reference

```
#include "physics/TrimeshShape.hh"
```

Include dependency graph for BulletTrimeshShape.hh:



Classes

- class **gazebo::physics::BulletTrimeshShape**

Triangle mesh collision.

Namespaces

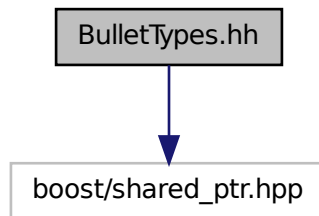
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.30 BulletTypes.hh File Reference

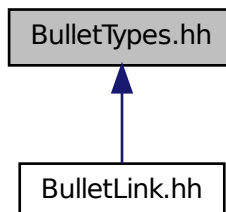
Bullet wrapper forward declarations and typedefs.

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for BulletTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Typedefs

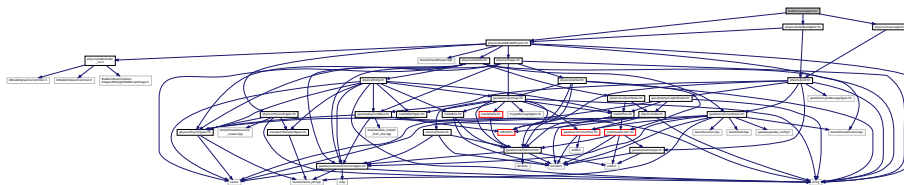
- typedef BulletCollision * **gazebo::physics::BulletCollisionPtr**
- typedef BulletLink * **gazebo::physics::BulletLinkPtr**
- typedef BulletPhysics * **gazebo::physics::BulletPhysicsPtr**
- typedef BulletRayShape * **gazebo::physics::BulletRayShapePtr**

11.30.1 Detailed Description

Bullet wrapper forward declarations and typedefs.

11.31 BulletUniversalJoint.hh File Reference

```
#include "physics/UniversalJoint.hh"
#include "physics/bullet/BulletJoint.hh"
#include "physics/bullet/BulletPhysics.hh"
Include dependency graph for BulletUniversalJoint.hh:
```



Classes

- class **gazebo::physics::BulletUniversalJoint**
A bullet universal joint class.

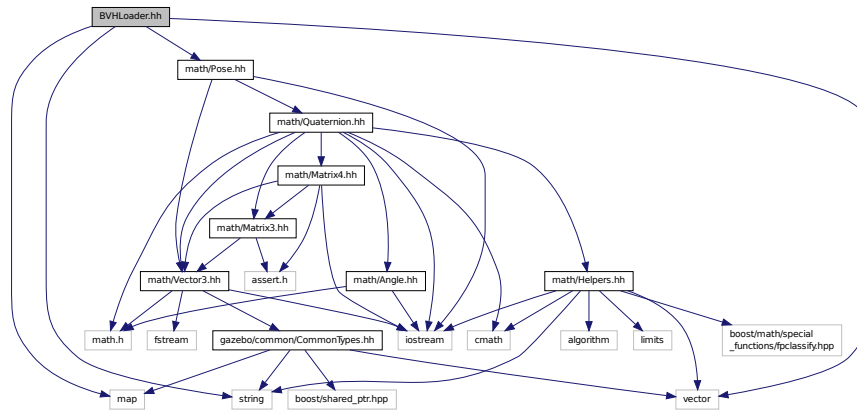
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

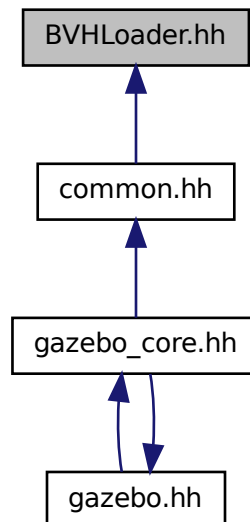
11.32 BVHLoader.hh File Reference

```
#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`

11.32.1 Macro Definition Documentation

11.32.1.1 `#define X_POSITION 0`

11.32.1.2 `#define X_ROTATION 3`

11.32.1.3 `#define Y_POSITION 1`

11.32.1.4 `#define Y_ROTATION 4`

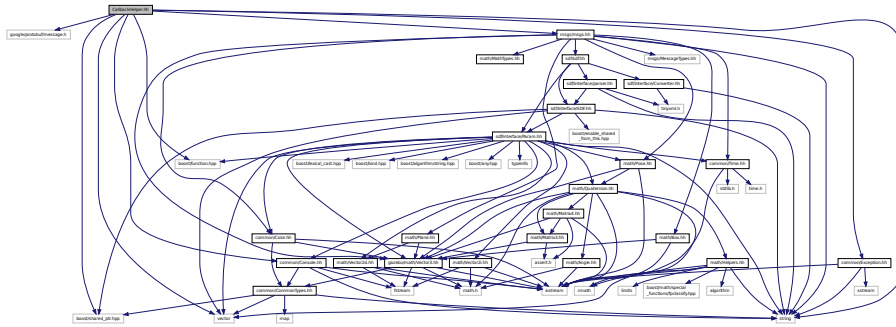
11.32.1.5 `#define Z_POSITION 2`

11.32.1.6 `#define Z_ROTATION 5`

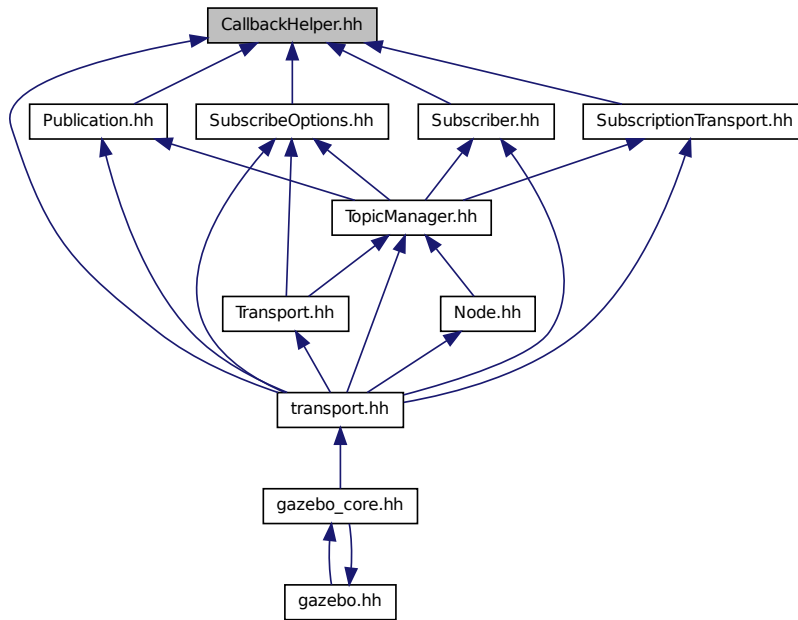
11.33 CallbackHelper.hh File Reference

```
#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**

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::transport**

Transport namespace.

Typedefs

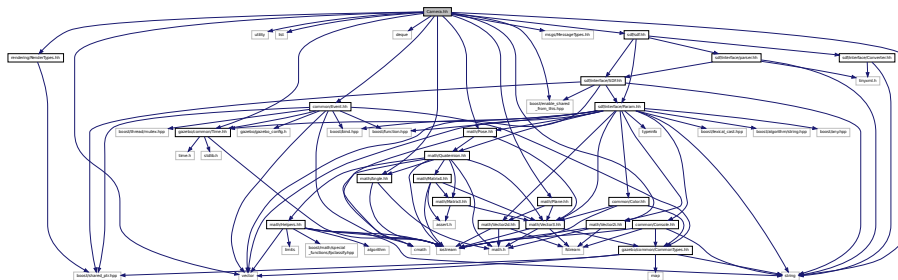
- typedef CallbackHelper * **gazebo::transport::CallbackHelperPtr**

*boost shared pointer to **transport::CallbackHelper** (p. 230)*

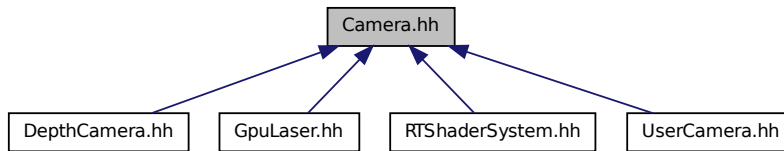
11.34 Camera.hh File Reference

```
#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**

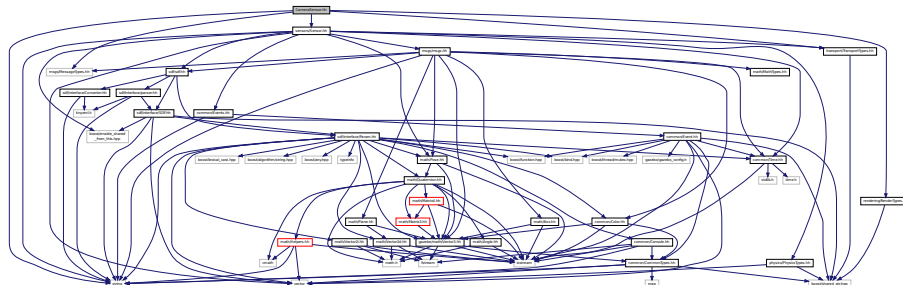
11.35 CameraSensor.hh File Reference

```

#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

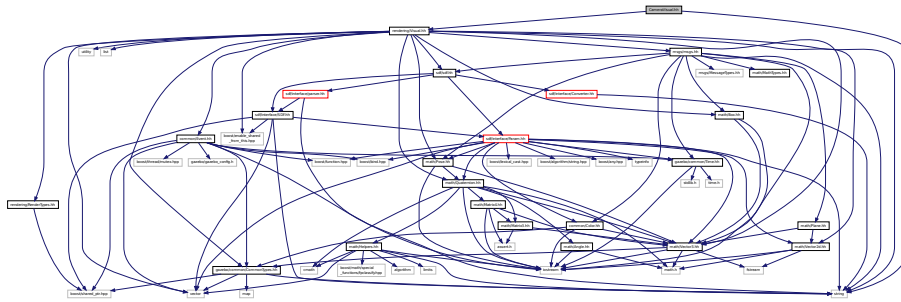
- class **gazebo::sensors::CameraSensor**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.36 CameraVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
Include dependency graph for CameraVisual.hh:
```



Classes

- class **gazebo::rendering::CameraVisual**
Basic camera visualization.

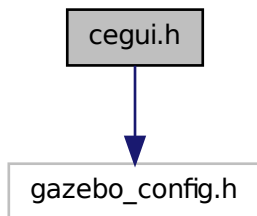
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

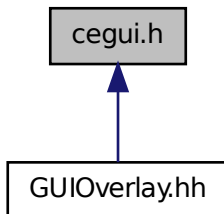
11.37 cegui.h File Reference

```
#include "gazebo_config.h"
```

Include dependency graph for cegui.h:



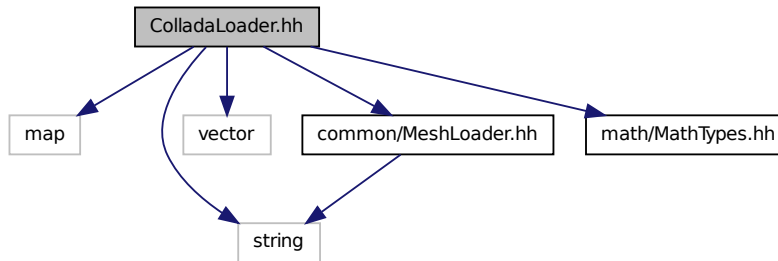
This graph shows which files directly or indirectly include this file:



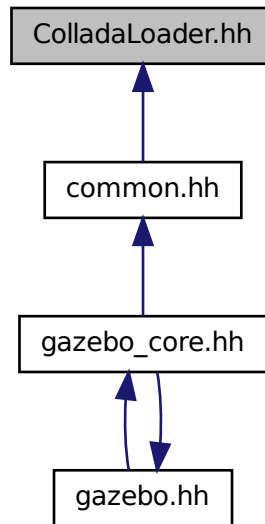
11.38 ColladaLoader.hh File Reference

```
#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.

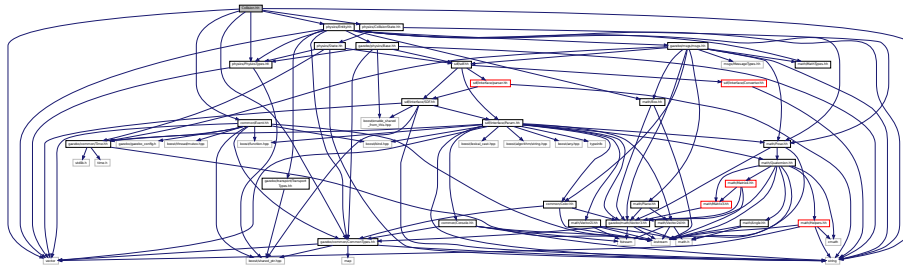
- namespace **gazebo::common**

Common namespace.

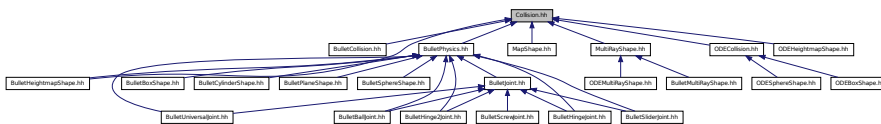
11.39 Collision.hh File Reference

```
#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. 145) class for all collision entities.

Namespaces

- namespace **gazebo**

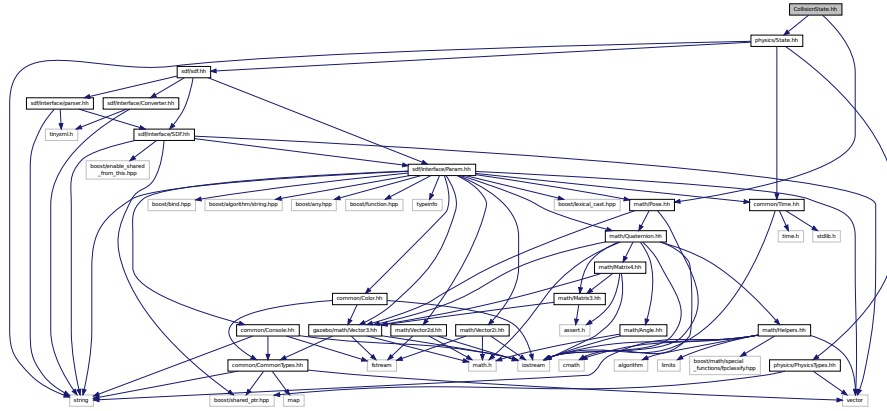
Forward declarations for the common classes.

- namespace **gazebo::physics**

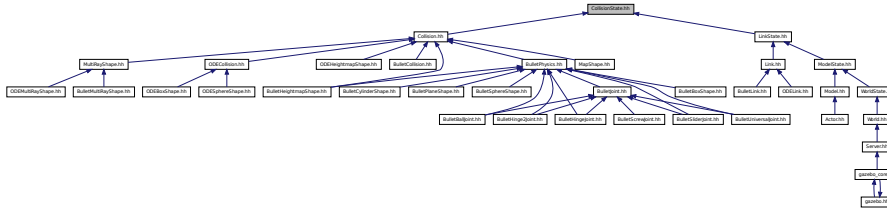
namespace for physics

11.40 CollisionState.hh File Reference

```
#include "physics/State.hh"
#include "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. 262) object.

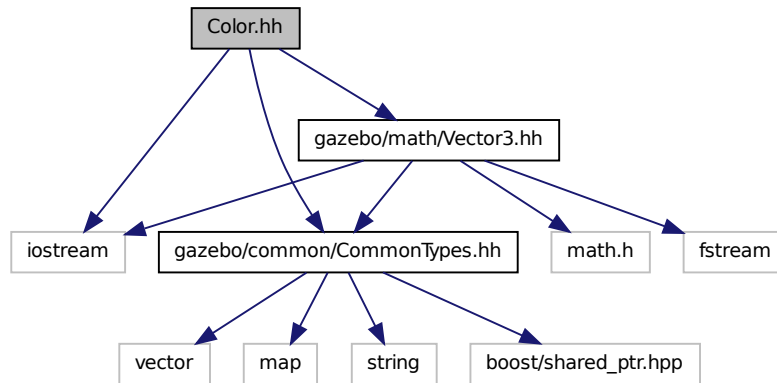
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

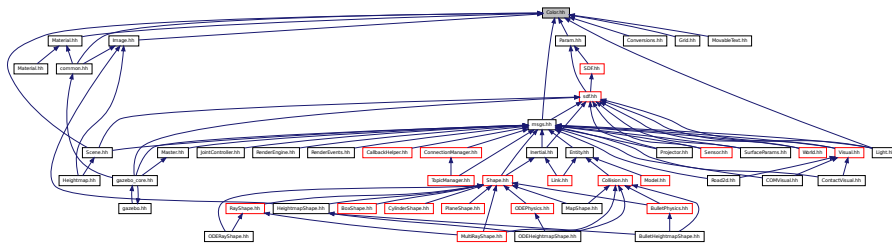
11.41 Color.hh File Reference

```
#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.

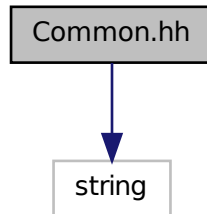
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

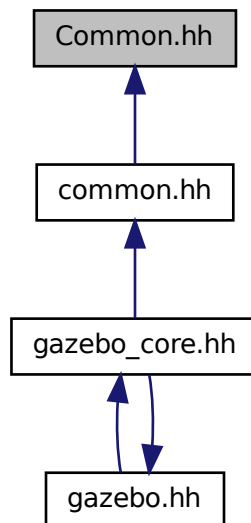
11.42 Common.hh File Reference

```
#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**

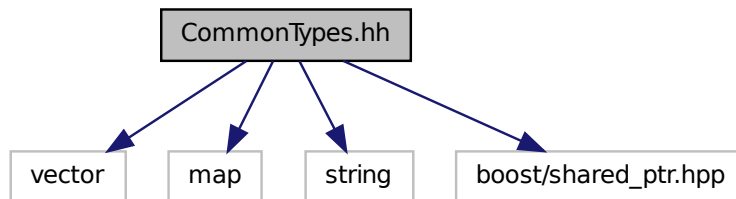
Common namespace.

Functions

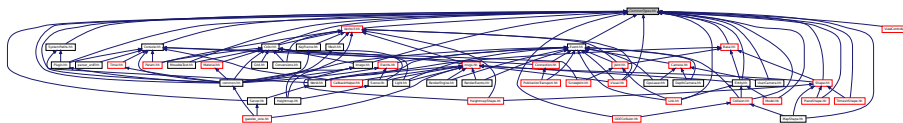
- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)
add path prefix to **common::SystemPaths** (p. 834)
- std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath=true)
search for file in **common::SystemPaths** (p. 834)
- std::string **gazebo::common::find_file_path** (const std::string &_file)
search for a file in **common::SystemPaths** (p. 834)

11.43 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. 350) 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**

11.43.1 Macro Definition Documentation

11.43.1.1 #define **GAZEBO_DEPRECATED**

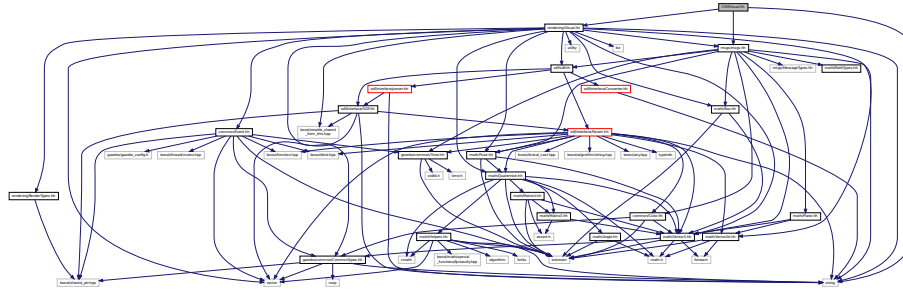
11.43.1.2 #define **GAZEBO_FORCEINLINE**

11.43.1.3 #define **NULL** 0

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::physics::ODEPlaneShape::CreatePlane(), gazebo::event::EventT< T >::Disconnect(), gazebo::transport::CallbackHelperT< M >::GetMsgType(), gazebo::transport::SubscribeOptions::Init(), gazebo::physics::ODESphereShape::SetRadius(), gazebo::physics::ODECylinderShape::SetSize(), and gazebo::physics::ODEBoxShape::SetSize().

11.44 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**

11.45 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"
```


Typedefs

- typedef Connection * **gazebo::transport::ConnectionPtr**

Functions

- bool **gazebo::transport::is_stopped** ()

Return true if the transport system is stopped.

11.45.1 Macro Definition Documentation

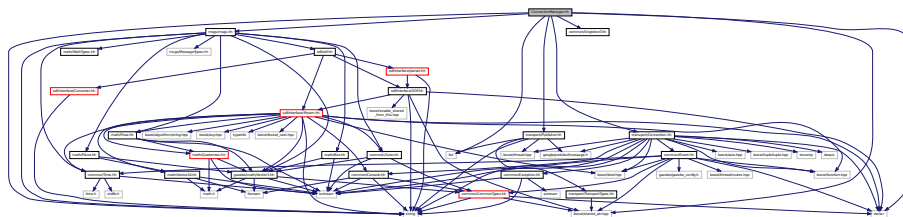
11.45.1.1 #define HEADER_LENGTH 8

Referenced by gazebo::transport::Connection::AsyncRead().

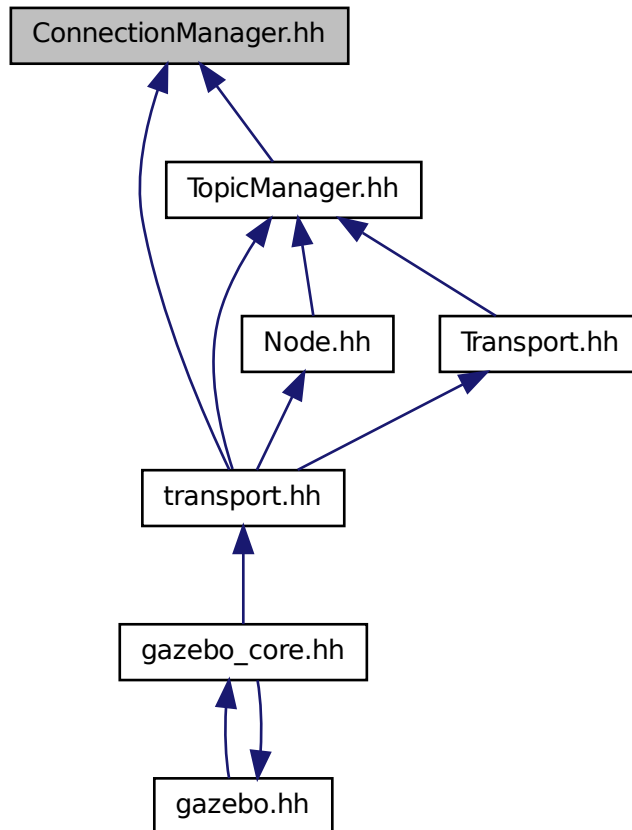
11.46 ConnectionManager.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include <list>
#include <vector>
#include "msgs/msgs.hh"
#include "common/SingletonT.hh"
#include "transport/Publisher.hh"
#include "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.

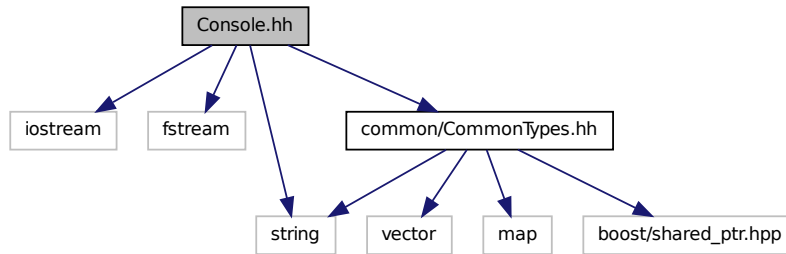
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**
Transport namespace.

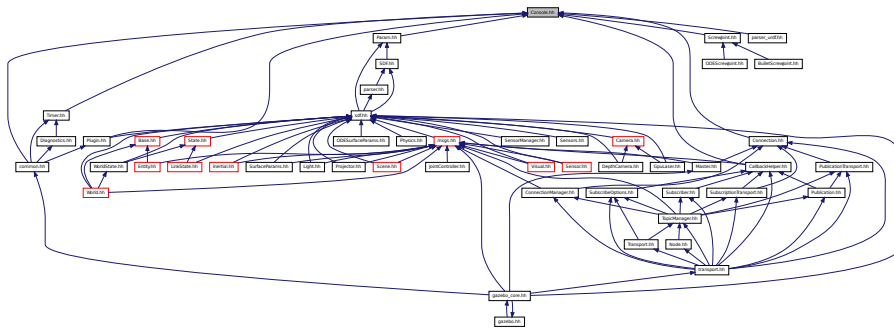
11.47 Console.hh File Reference

```
#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, and logging functionality.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

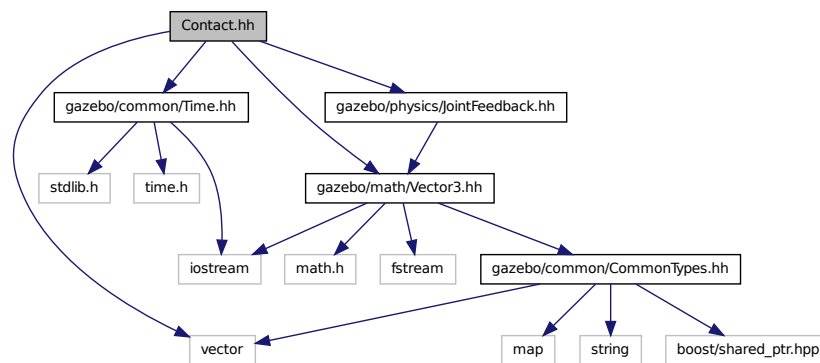
Macros

- `#define gzclr_end "\033[0m"`
end marker
- `#define gzclr_start clr "\033[1;33m"`
start marker
- `#define gzdbg (gazebo::common::Console::Instance()->ColorMsg("Dbg", 36))`
Output a debug message.
- `#define gzerr`
Output an error message.
- `#define gzlog`
Log a message.
- `#define gzmsg (gazebo::common::Console::Instance()->ColorMsg("Msg", 32))`
Output a message.
- `#define gzwarn`
Output a warning message.

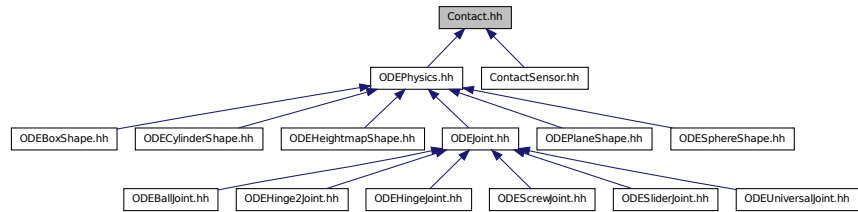
11.48 Contact.hh File Reference

```
#include <vector>
#include "gazebo/common/Time.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/JointFeedback.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

11.48.1 Macro Definition Documentation

11.48.1.1 #define MAX_COLLIDE_RETURNS 250

11.48.1.2 #define MAX_CONTACT_JOINTS 32

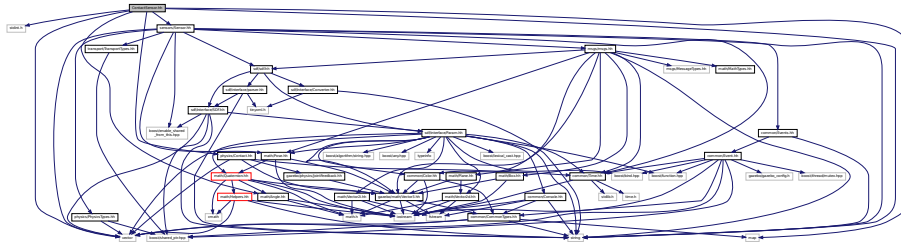
11.49 ContactSensor.hh File Reference

```

#include <stdint.h>
#include <vector>
#include <map>
#include <string>
#include "math/Angle.hh"
#include "sensors/Sensor.hh"
#include "physics/Contact.hh"

```

Include dependency graph for ContactSensor.hh:



Classes

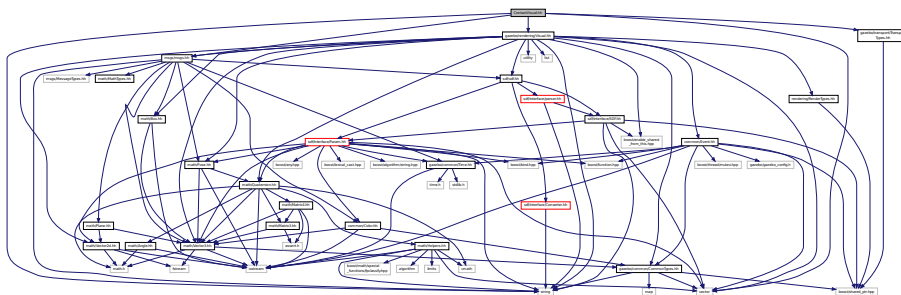
- class **gazebo::sensors::ContactSensor**

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.50 ContactVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
Include dependency graph for ContactVisual.hh:
```



Classes

- class **gazebo::rendering::ContactVisual**
Contact visualization.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

Rendering namespace.

- namespace **Ogre**

11.51 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. 305) *Conversions.hh* (p. 1018) *rendering/Conversions.hh* (p. 1018).

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

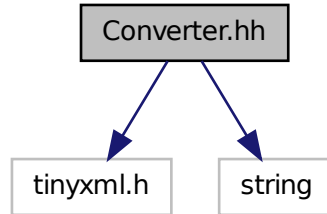
- namespace **gazebo::rendering**

Rendering namespace.

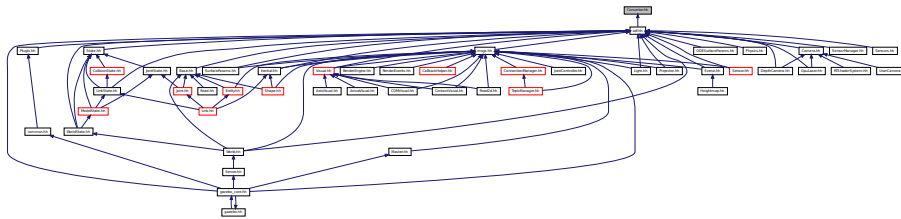
11.52 Converter.hh File Reference

```
#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. 762) to another.*

Namespaces

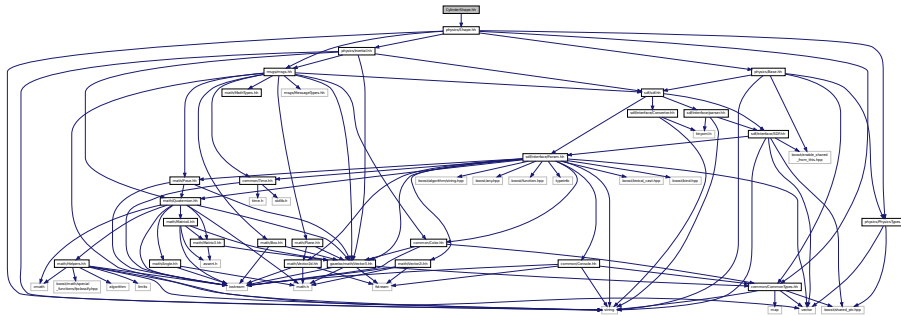
- namespace **sdf**

namespace for Simulation Description Format parser

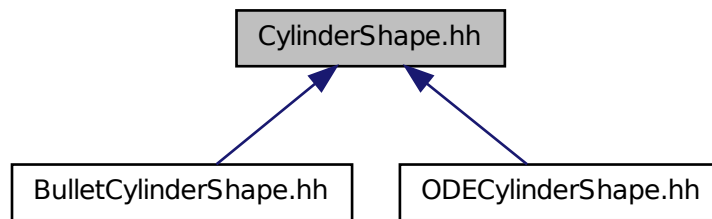
11.53 CylinderShape.hh File Reference

```
#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

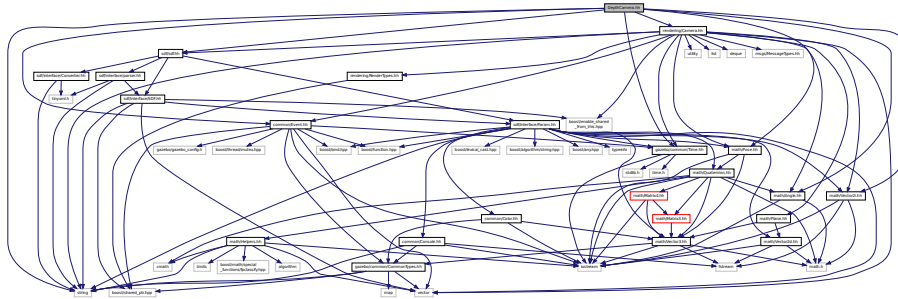
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.54 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**

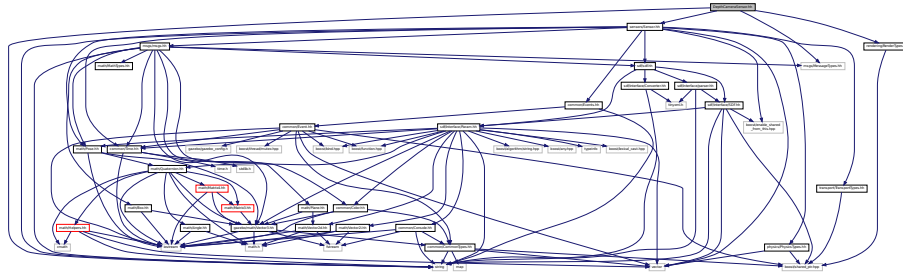
Rendering namespace.

- namespace **Ogre**

11.55 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**

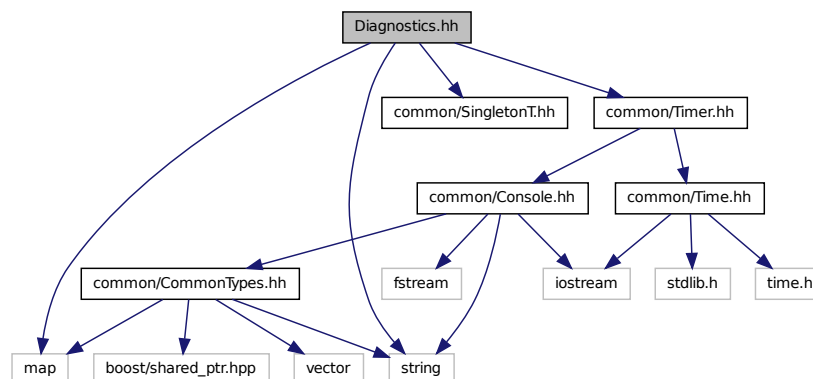
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

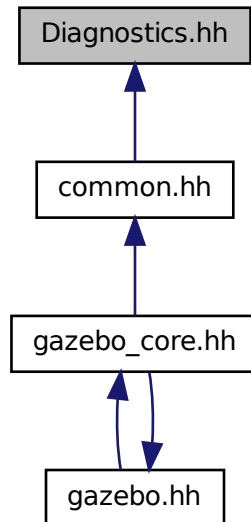
11.56 Diagnostics.hh File Reference

```
#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**

11.57 DynamicLines.hh File Reference

```
#include <vector>
#include <string>
#include "math/Vector3.hh"
#include "rendering/DynamicRenderable.hh"
```

Include dependency graph for DynamicLines.hh:



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.

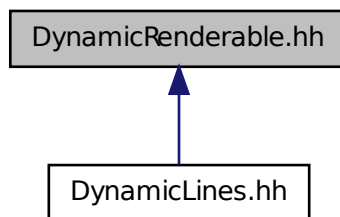
11.58 DynamicRenderable.hh File Reference

```
#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.

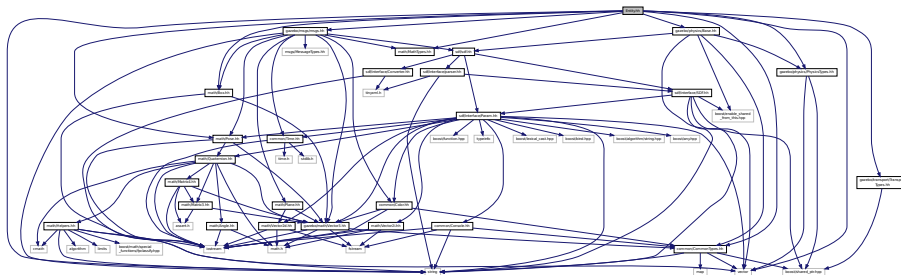
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

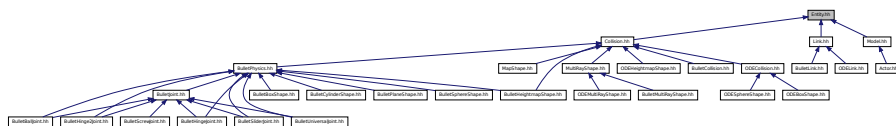
11.59 Entity.hh File Reference

```
#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. 145) class for all physics objects in Gazebo.

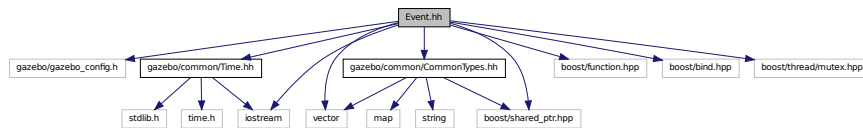
Namespaces

- namespace **boost**
- namespace **gazebo**
 - Forward declarations for the common classes.*
- namespace **gazebo::physics**
 - namespace for physics*

11.60 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.*

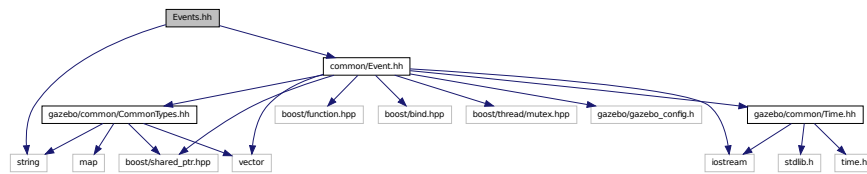
Namespaces

- namespace **gazebo**
 - Forward declarations for the common classes.*

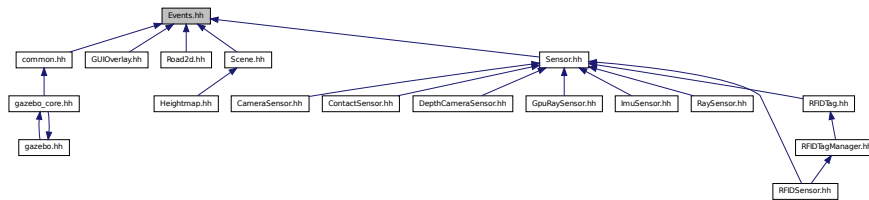
- namespace **gazebo::event**
Event (p. 350) namespace.

11.61 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. 350) class to get notifications for simulator events.

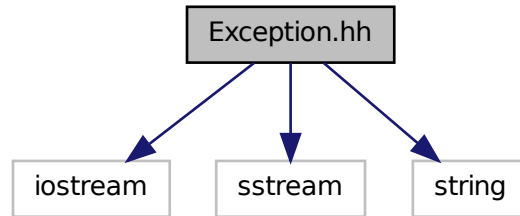
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::event**
Event (p. 350) namespace.

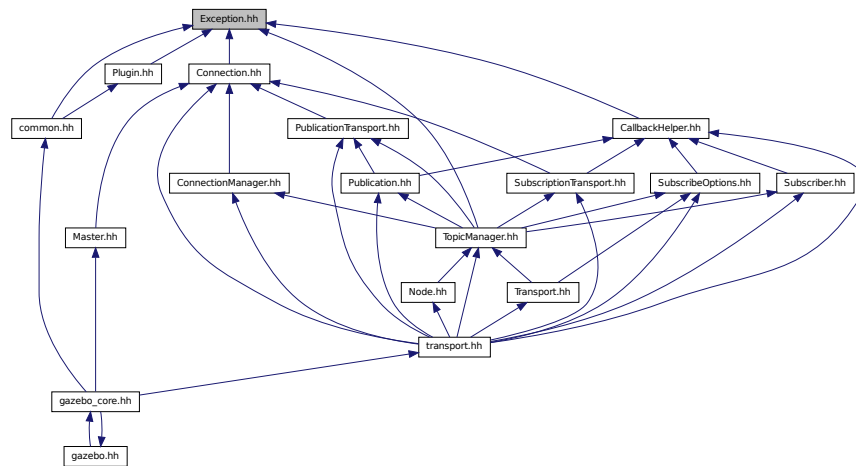
11.62 Exception.hh File Reference

```
#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.

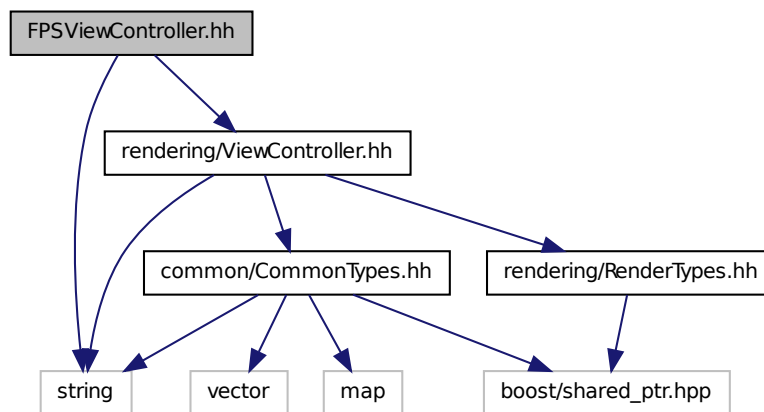
Macros

- `#define gzthrow(msg)`

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

11.63 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.

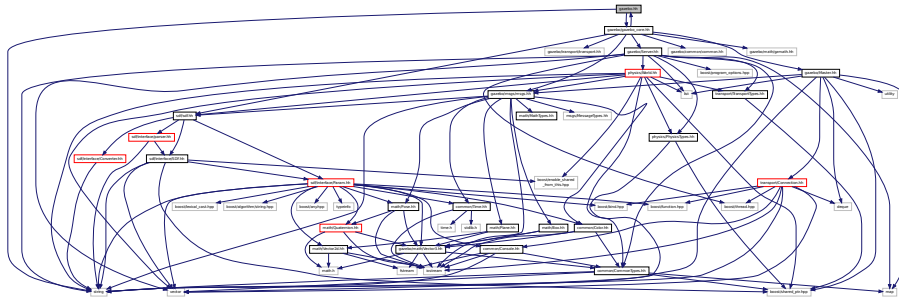
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

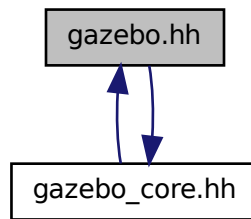
11.64 gazebo.hh File Reference

```
#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)

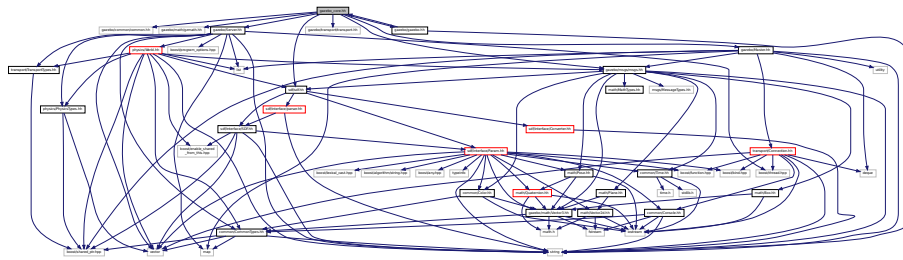
Find a file in the gazebo search paths.

- 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** ()

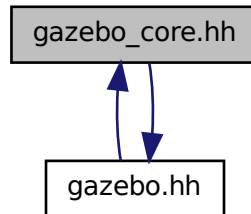
11.65 gazebo_core.hh File Reference

```
#include <gazebo/common/common.hh>
#include <gazebo/math/gzmath.hh>
#include <gazebo/msgs/msgs.hh>
#include <gazebo/sdf/sdf.hh>
#include <gazebo/transport/transport.hh>
#include <gazebo/Server.hh>
#include <gazebo/Master.hh>
#include <gazebo/gazebo.hh>
```

Include dependency graph for gazebo_core.hh:



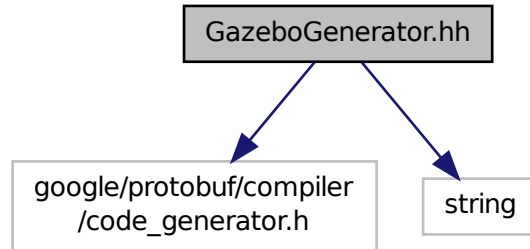
This graph shows which files directly or indirectly include this file:



11.66 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. 101).

Namespaces

- namespace **google**
- namespace **google::protobuf**
- namespace **google::protobuf::compiler**
- namespace **google::protobuf::compiler::cpp**

11.67 GpuLaser.hh File Reference

```

#include <string>
#include <vector>
#include "rendering/ogre_gazebo.h"
#include "rendering/Camera.hh"
#include "sensors/SensorTypes.hh"
#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 dependency graph for GpuLaser.hh:



Classes

- class **gazebo::rendering::GpuLaser**
GPU based laser distance sensor.

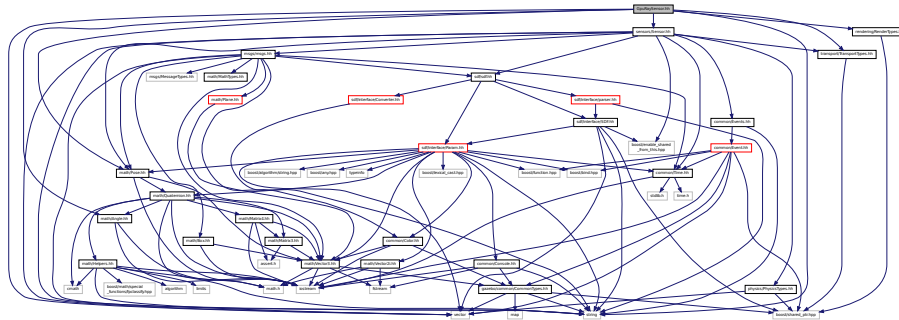
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.68 GpuRaySensor.hh File Reference

```
#include <vector>
#include <string>
#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.

11.69 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**

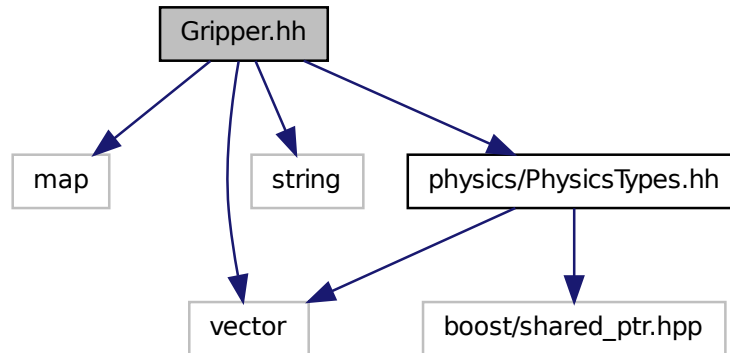
Rendering namespace.

- namespace **Ogre**

11.70 Gripper.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#include "physics/PhysicsTypes.hh"
```

Include dependency graph for Gripper.hh:



Classes

- class **gazebo::physics::Gripper**

A gripper abstraction.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

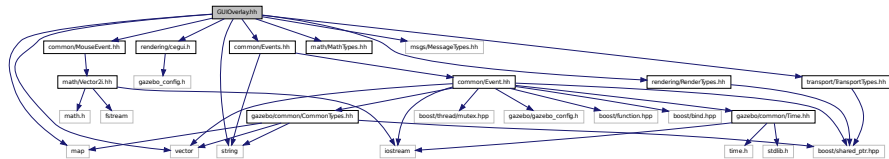
namespace for physics

11.71 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.

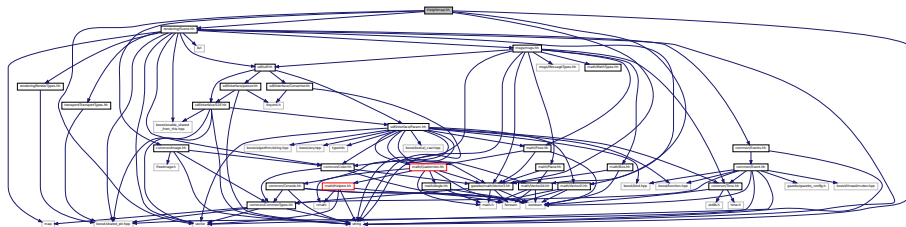
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.72 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.

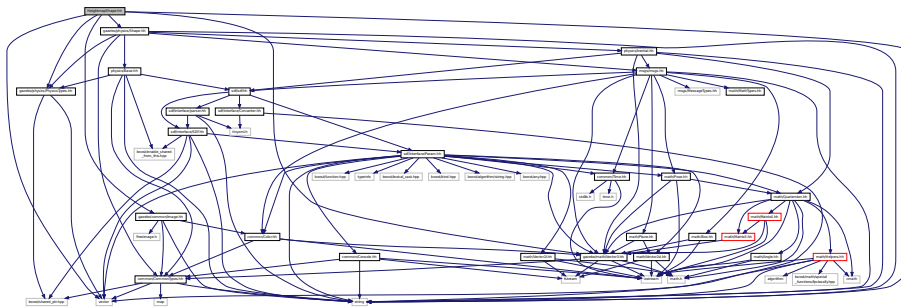
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

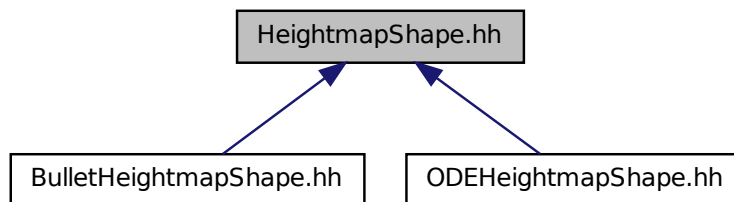
11.73 HeightmapShape.hh File Reference

```
#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. 399) collision shape builds a heightmap from an image.

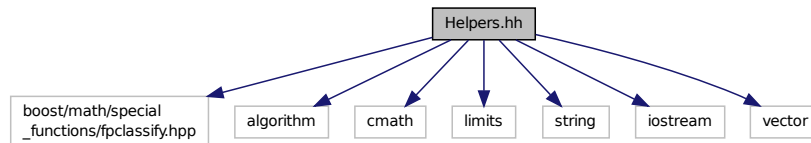
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

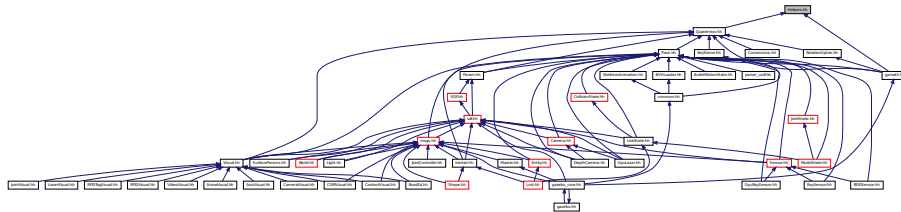
11.74 Helpers.hh File Reference

```
#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()
Not a number.
- static const double **gazebo::math::NAN_I** = std::numeric_limits<int>::quiet_NaN()
TODO Nate: type int has no quiet_NaN ... what does this 0 mean?

11.74.1 Macro Definition Documentation

11.74.1.1 `#define GZ_DBL_MAX std::numeric_limits<double>::max()`

11.74.1.2 `#define GZ_DBL_MIN std::numeric_limits<double>::min()`

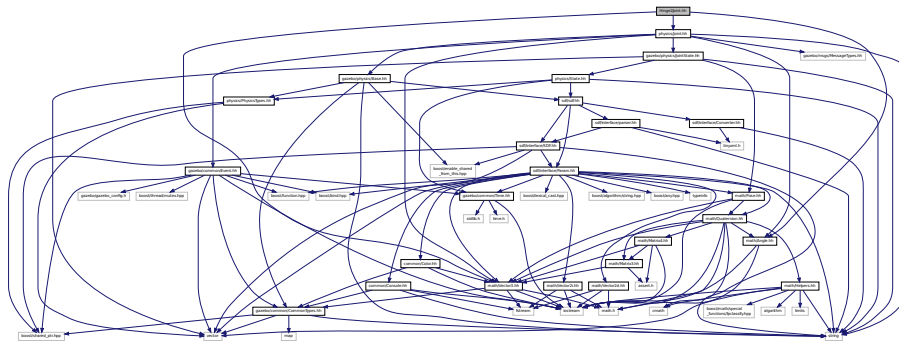
11.74.1.3 `#define GZ_FLT_MAX std::numeric_limits<float>::max()`

11.74.1.4 `#define GZ_FLT_MIN std::numeric_limits<float>::min()`

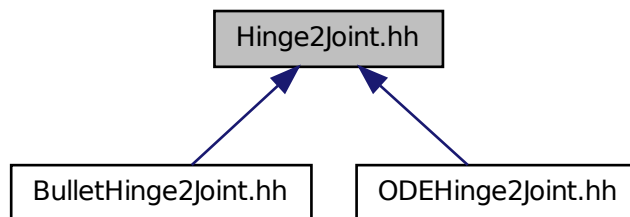
11.75 Hinge2Joint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "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

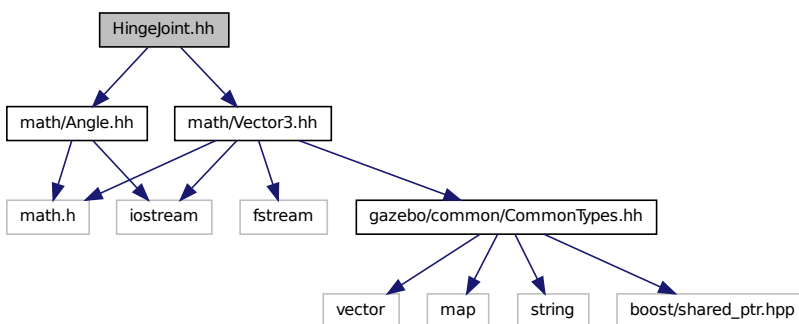
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.76 HingeJoint.hh File Reference

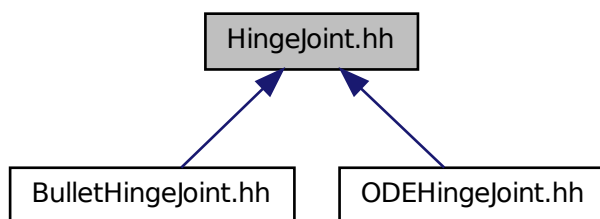
```
#include "math/Angle.hh"
```

```
#include "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.

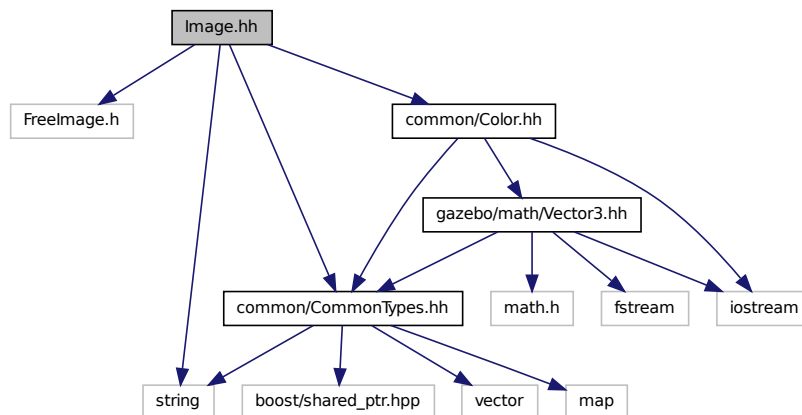
- namespace **gazebo::physics**

namespace for physics

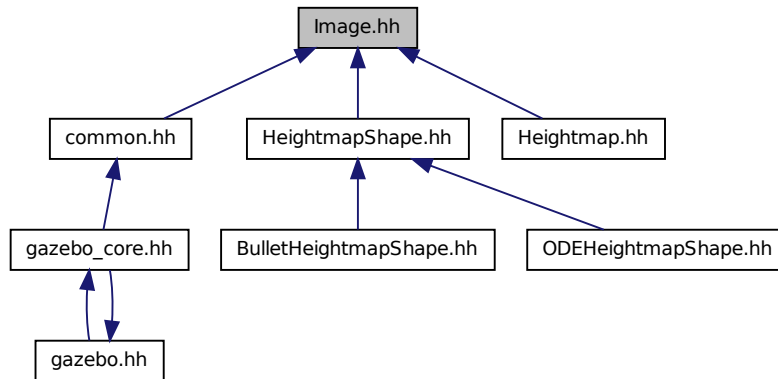
11.77 Image.hh File Reference

```
#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.

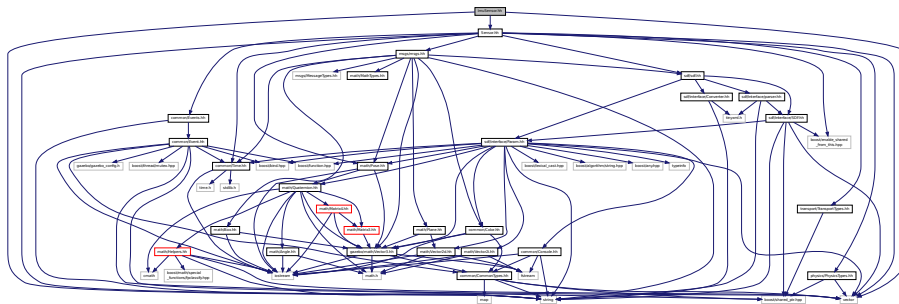
Namespaces

- namespace **`gazebo`**
Forward declarations for the common classes.
- namespace **`gazebo::common`**
Common namespace.

11.78 ImuSensor.hh File Reference

```
#include <vector>
#include <string>
#include "Sensor.hh"
```

Include dependency graph for `ImuSensor.hh`:



Classes

- class **gazebo::sensors::ImuSensor**

An IMU sensor.

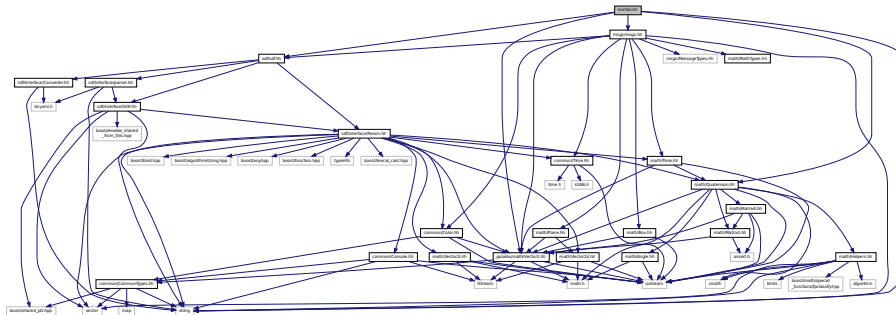
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

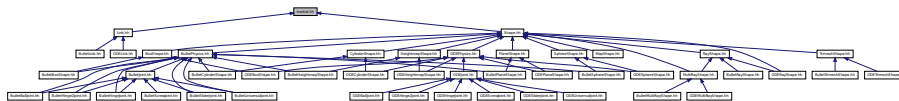
11.79 Inertial.hh File Reference

```
#include <string>
#include "msgs/msgs.hh"
#include "sdf/sdf.hh"
#include "math/Quaternion.hh"
#include "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**

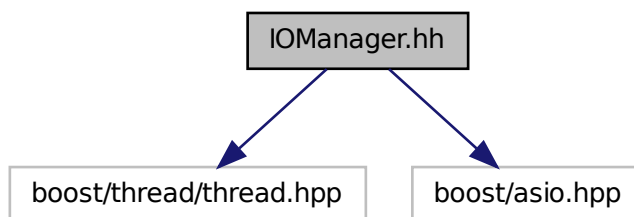
namespace for physics

11.80 IOManager.hh File Reference

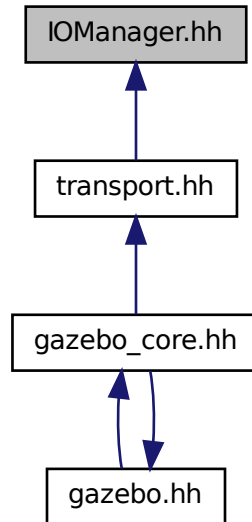
```
#include <boost/thread/thread.hpp>
```

```
#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**
Managers boost::asio IO.

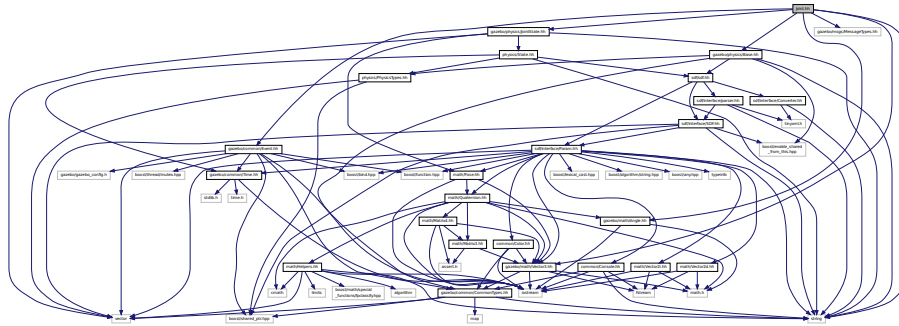
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**
Transport namespace.

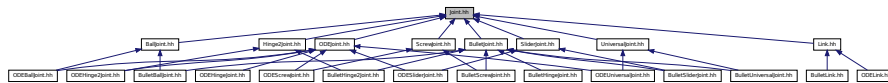
11.81 Joint.hh File Reference

```
#include <string>
#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. 145) class for all joints.

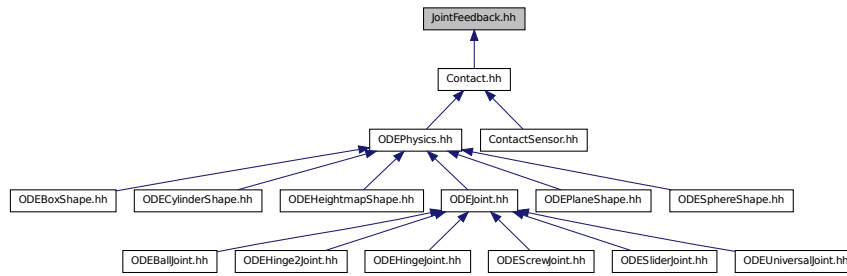
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.82 JointController.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include "gazebo/common/PID.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/messages/msgs.hh"
```


This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointFeedback**
Feedback information from a joint.

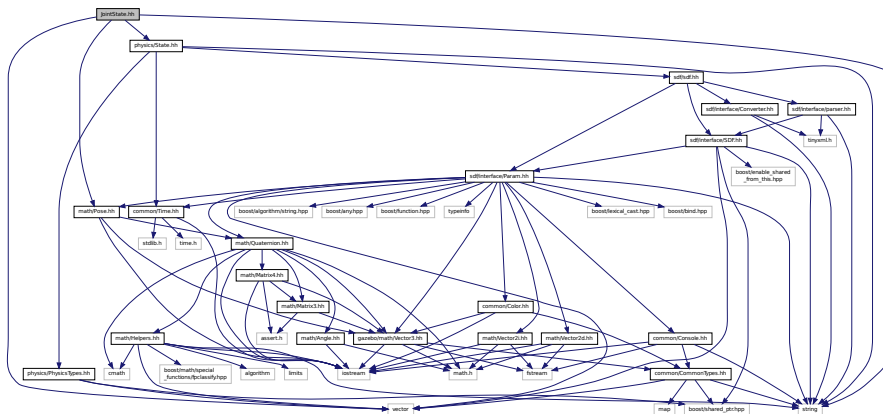
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.84 JointState.hh File Reference

```
#include <vector>
#include <string>
#include "physics/State.hh"
#include "math/Pose.hh"
```

Include dependency graph for JointState.hh:



Namespaces

- namespace **gazebo**

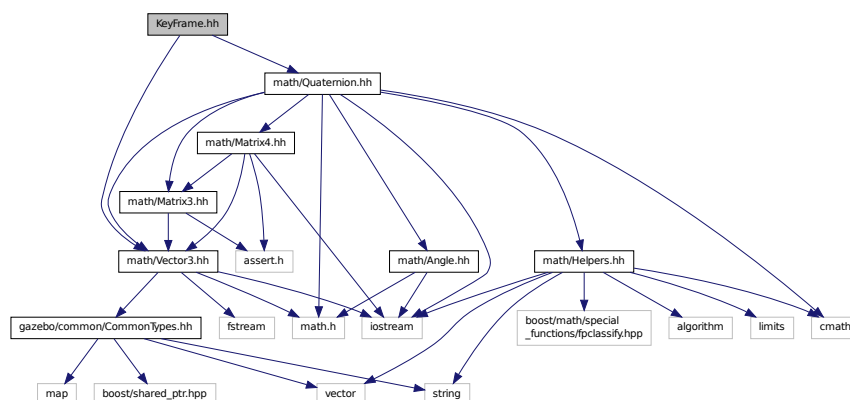
Forward declarations for the common classes.

- namespace **gazebo::rendering**

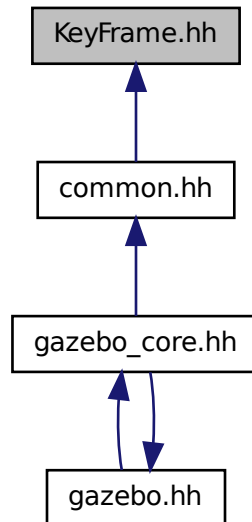
Rendering namespace.

11.86 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. 569).*
- class **gazebo::common::PoseKeyFrame**
*A keyframe for a **PoseAnimation** (p. 685).*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

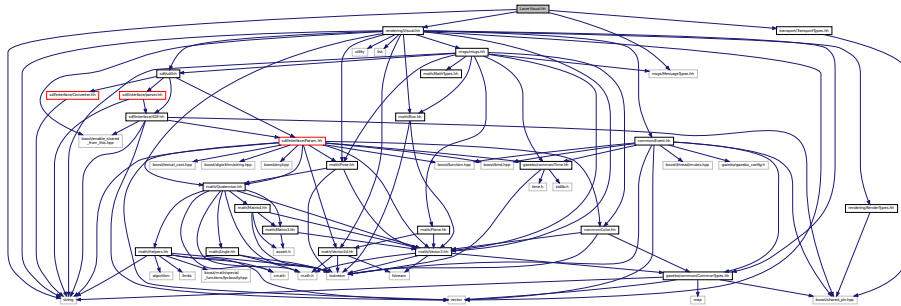
11.87 LaserVisual.hh File Reference

```

#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.

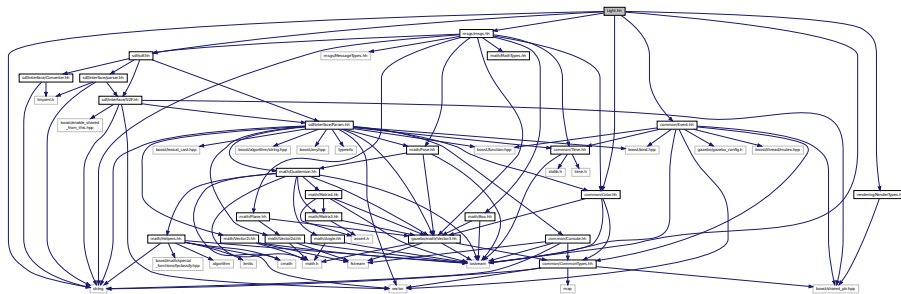
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.88 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**

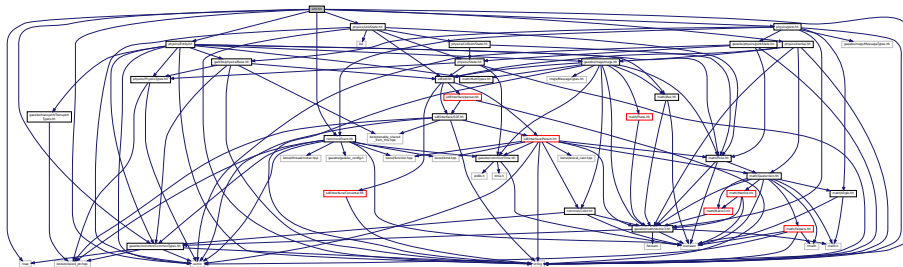
Rendering namespace.

- namespace **Ogre**

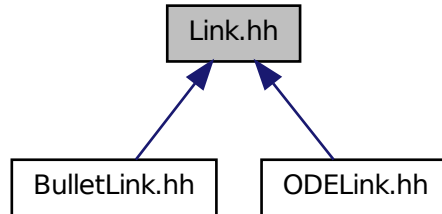
11.89 Link.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#include "common/Event.hh"
#include "common/CommonTypes.hh"
#include "physics/LinkState.hh"
#include "physics/Entity.hh"
#include "physics/Inertial.hh"
#include "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. 454) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

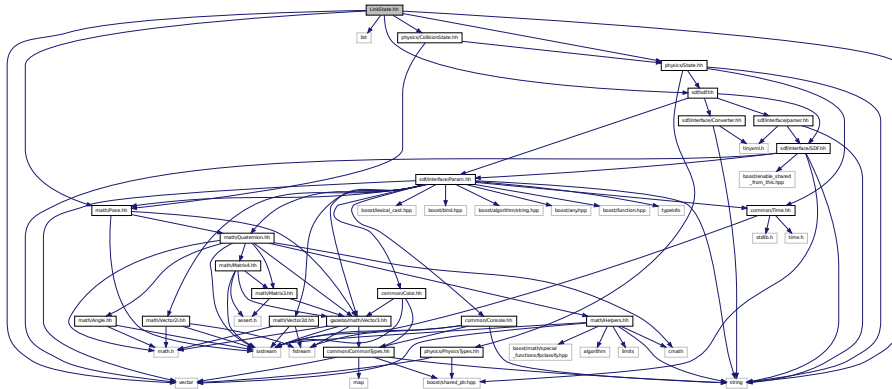
- namespace **gazebo::physics**

namespace for physics

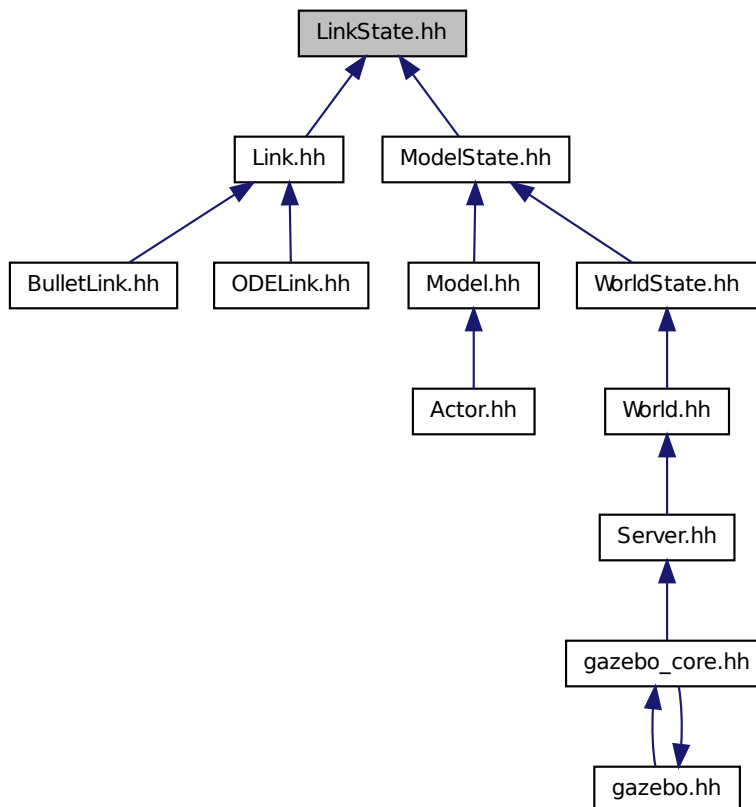
11.90 LinkState.hh File Reference

```
#include <vector>
#include <string>
#include <list>
#include "sdf/sdf.hh"
#include "physics/State.hh"
#include "physics/CollisionState.hh"
#include "math/Pose.hh"
```

Include dependency graph for LinkState.hh:



This graph shows which files directly or indirectly include this file:



Classes

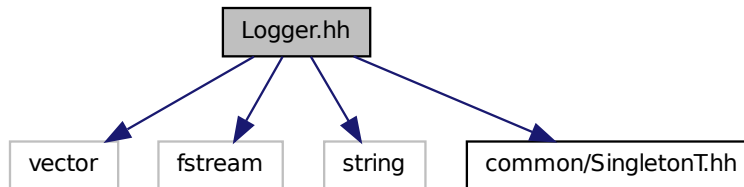
- class **gazebo::physics::LinkState**
*Store state information of a **physics::Link** (p. 454) object.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.91 Logger.hh File Reference

```
#include <vector>
#include <fstream>
#include <string>
#include "common/SingletonT.hh"
Include dependency graph for Logger.hh:
```



Classes

- class **gazebo::physics::Logger**
Handles logging of data to disk.

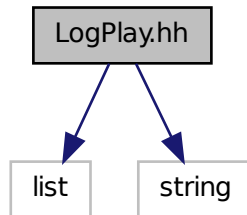
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.92 LogPlay.hh File Reference

```
#include <list>
#include <string>
```

Include dependency graph for LogPlay.hh:



Classes

- class **gazebo::physics::Logplay**

*Open and playback log files that were recorded using **Logger** (p. 477).*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

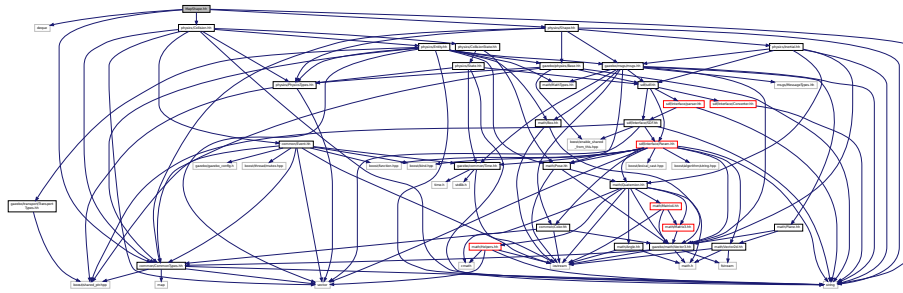
namespace for physics

11.93 mainpage.html File Reference

11.94 MapShape.hh File Reference

```
#include <deque>
#include <string>
#include "common/CommonTypes.hh"
#include "physics/Collision.hh"
#include "physics/Shape.hh"
```

Include dependency graph for MapShape.hh:



Classes

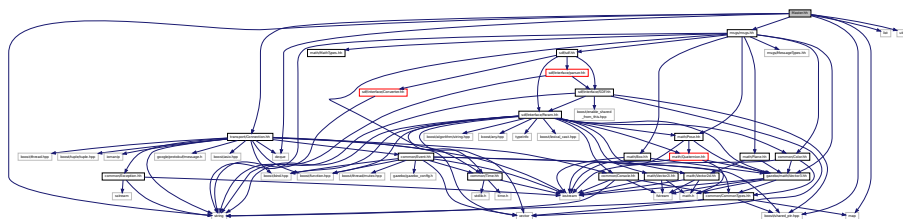
- class **gazebo::physics::MapShape**
Creates box extrusions based on an image.

Namespaces

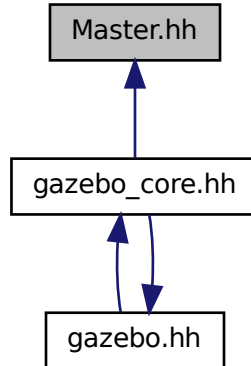
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.95 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

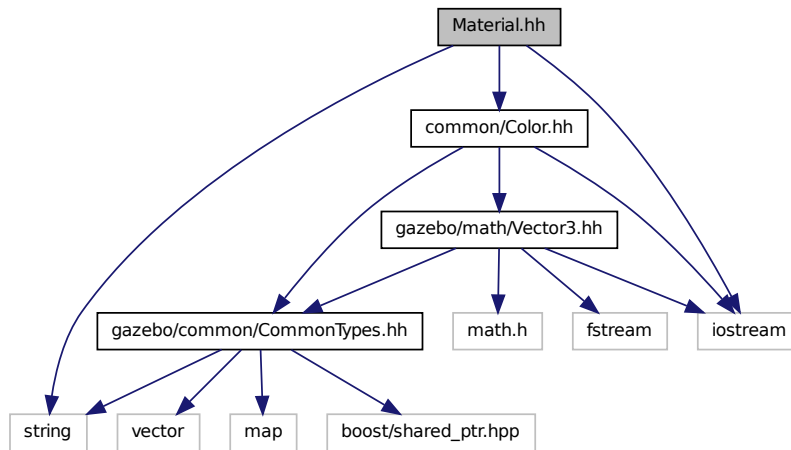
- namespace **gazebo**

Forward declarations for the common classes.

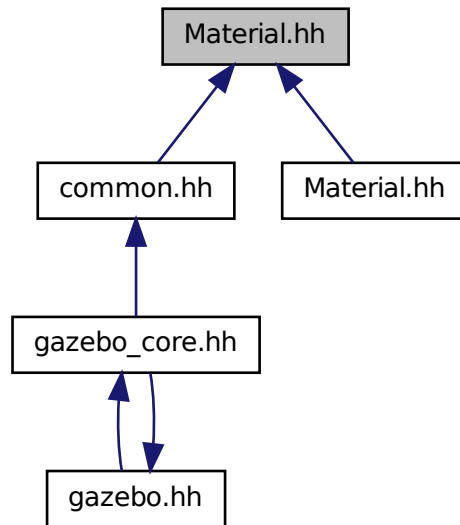
11.96 Material.hh File Reference

```
#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.

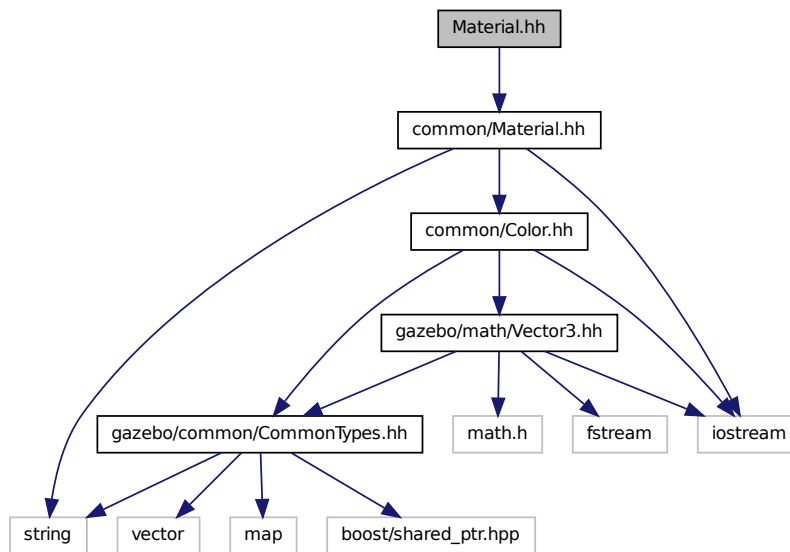
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.97 Material.hh File Reference

```
#include "common/Material.hh"
```

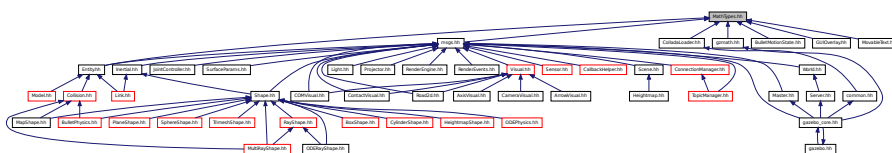
Include dependency graph for rendering/Material.hh:



11.98 MathTypes.hh File Reference

Forward declarations for the math classes.

This graph shows which files directly or indirectly include this file:



Namespaces

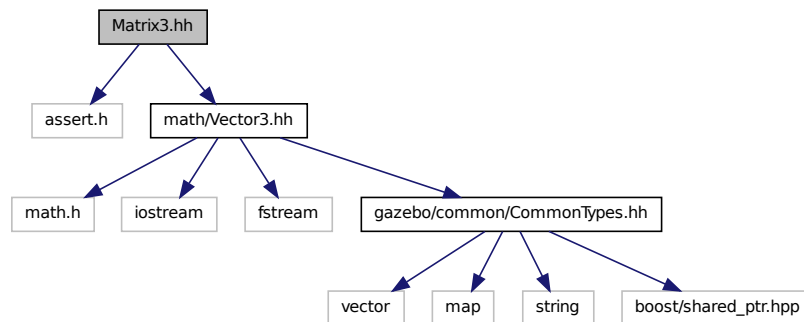
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.98.1 Detailed Description

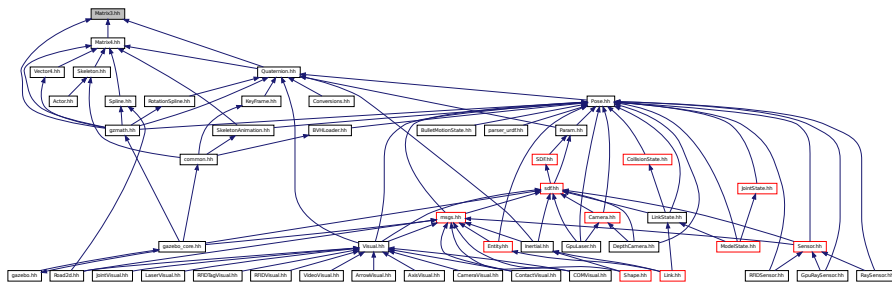
Forward declarations for the math classes.

11.99 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

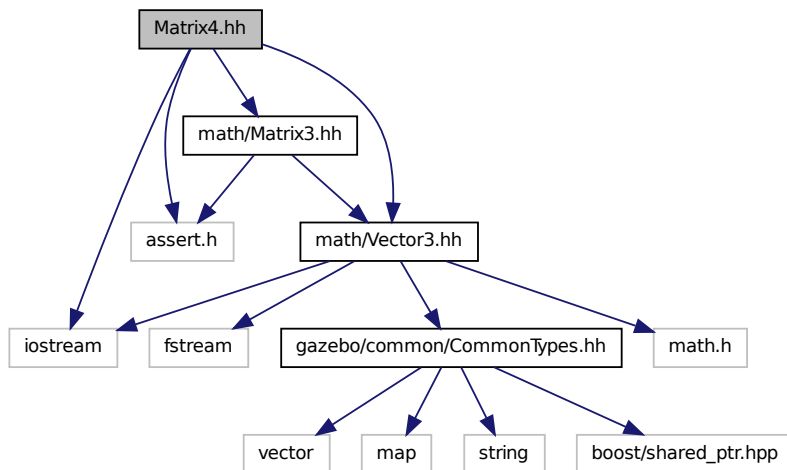
- class **gazebo::math::Matrix3**
A 3x3 matrix class.

Namespaces

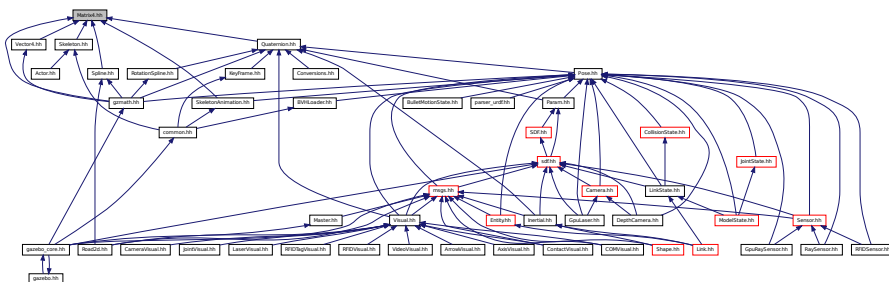
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.100 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`

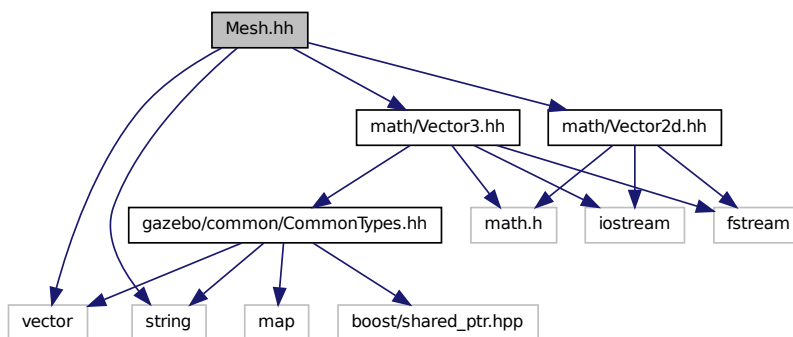
Forward declarations for the common classes.

- namespace `gazebo::math`

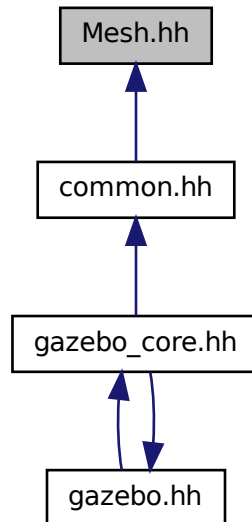
Math namespace.

11.101 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.

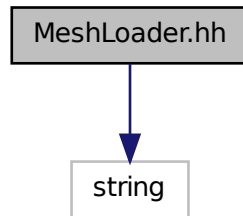
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

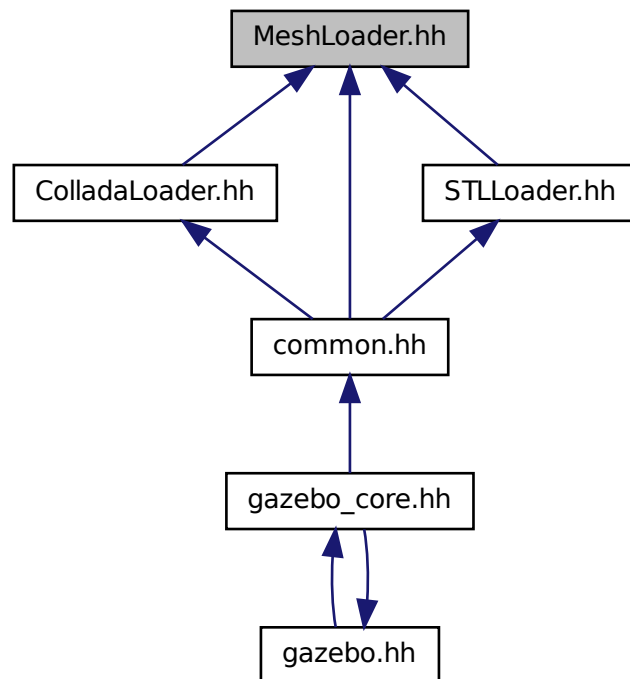
11.102 MeshLoader.hh File Reference

```
#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**

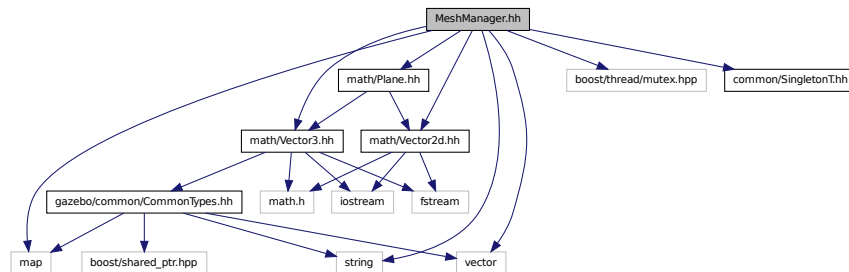
Forward declarations for the common classes.

- namespace **gazebo::common**

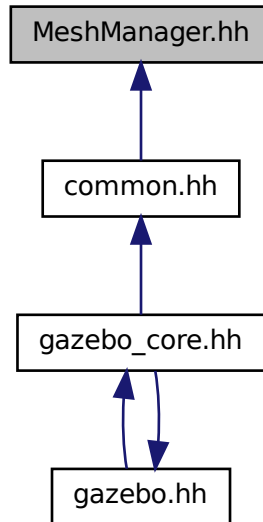
Common namespace.

11.103 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.

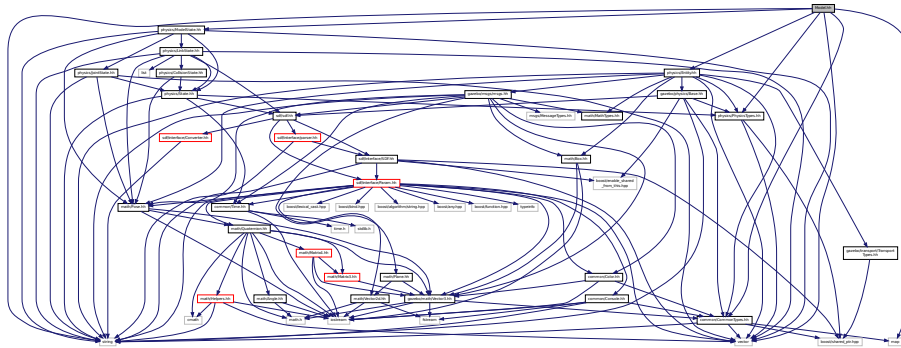
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

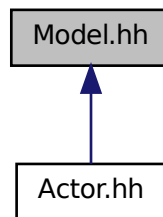
11.104 Model.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/ModelState.hh"
#include "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

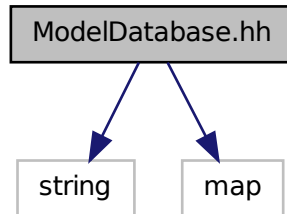
- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.105 ModelDatabase.hh File Reference

```
#include <string>
```

```
#include <map>
```

Include dependency graph for ModelStateDatabase.hh:



Classes

- class **gazebo::common::ModelStateDatabase**

Connects to model database, and has utility functions to find models.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

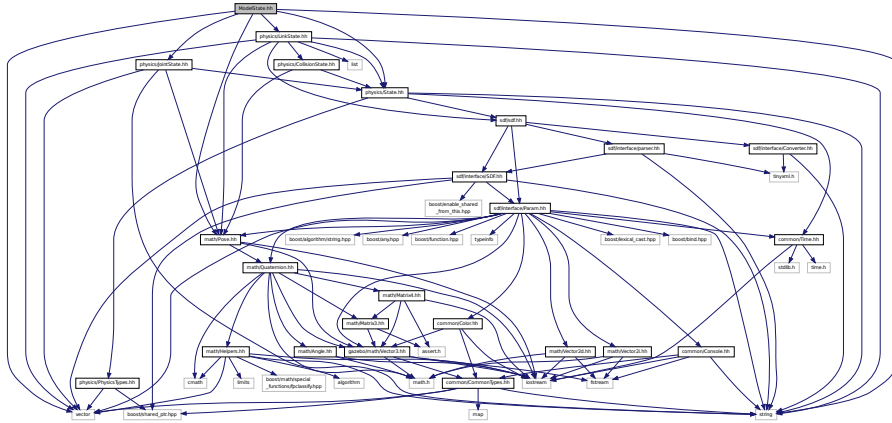
- namespace **gazebo::common**

Common namespace.

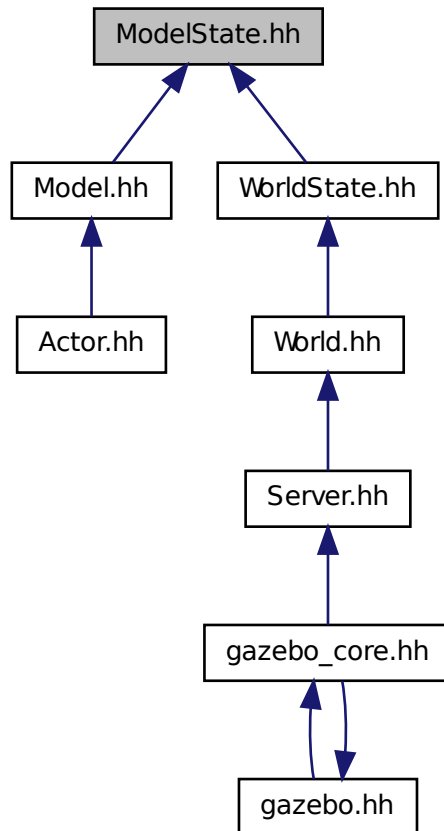
11.106 ModelState.hh File Reference

```
#include <vector>
#include <string>
#include "physics/State.hh"
#include "physics/LinkState.hh"
#include "physics/JointState.hh"
#include "math/Pose.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. 521) object.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

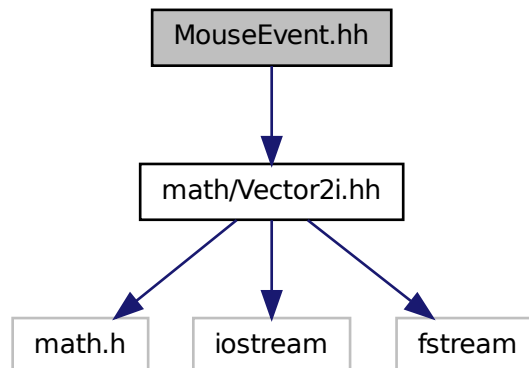
- namespace **gazebo::physics**

namespace for physics

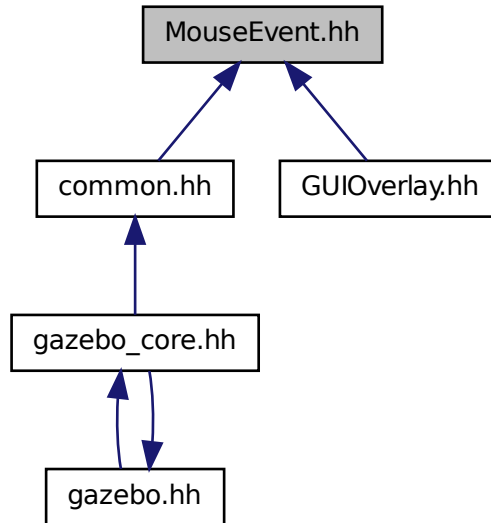
11.107 MouseEvent.hh File Reference

```
#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.

11.108 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.

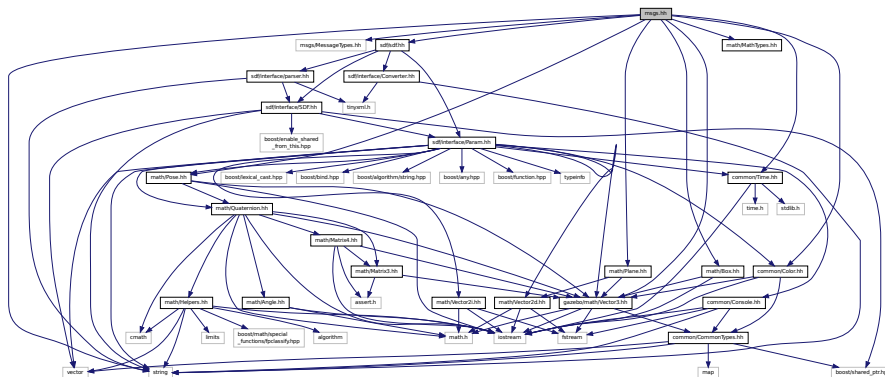
Namespaces

- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.109 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

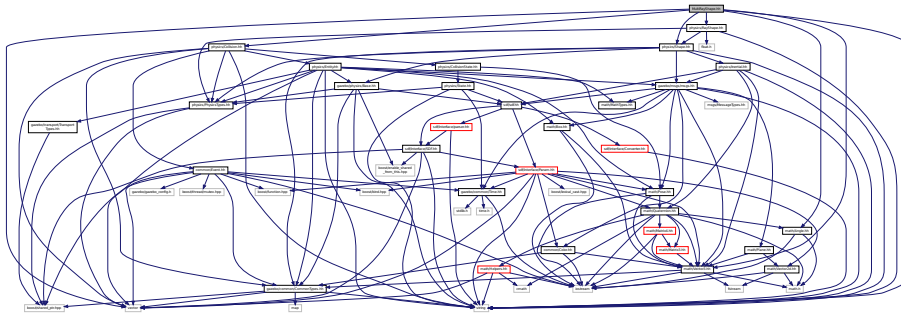
- msgs::Vector3d **gazebo::msgs::Convert** (const math::Vector3 &_v)
*Convert a **math::Vector3** (p. 902) to a msgs::Vector3d.*
- msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)
*Convert a **math::Quaternion** (p. 697) to a msgs::Quaternion.*
- msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)
*Convert a **math::Pose** (p. 677) to a msgs::Pose.*
- msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)
*Convert a **common::Color** (p. 274) to a msgs::Color.*
- msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)
*Convert a **common::Time** (p. 840) to a msgs::Time.*
- msgs::PlaneGeom **gazebo::msgs::Convert** (const math::Plane &_p)
*Convert a **math::Plane** (p. 668) 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. 697).*
- math::Pose **gazebo::msgs::Convert** (const msgs::Pose &_p)
*Convert a msgs::Pose to a **math::Pose** (p. 677).*
- common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)
*Convert a msgs::Color to a **common::Color** (p. 274).*
- common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)
*Convert a msgs::Time to a **common::Time** (p. 840).*
- 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. 407).*
- `void gazebo::msgs::Set (msgs::Image *_msg, const common::Image &_i)`
*Set a msgs::Image from a **common::Image** (p. 407).*
- `void gazebo::msgs::Set (msgs::Vector3d *_pt, const math::Vector3 &_v)`
*Set a msgs::Vector3d from a **math::Vector3** (p. 902).*
- `void gazebo::msgs::Set (msgs::Vector2d *_pt, const math::Vector2d &_v)`
*Set a msgs::Vector2d from a **math::Vector3** (p. 902).*
- `void gazebo::msgs::Set (msgs::Quaternion *_q, const math::Quaternion &_v)`
*Set a msgs::Quaternion from a **math::Quaternion** (p. 697).*
- `void gazebo::msgs::Set (msgs::Pose *_p, const math::Pose &_v)`
*Set a msgs::Pose from a **math::Pose** (p. 677).*
- `void gazebo::msgs::Set (msgs::Color *_c, const common::Color &_v)`
*Set a msgs::Color from a **common::Color** (p. 274).*
- `void gazebo::msgs::Set (msgs::Time *_t, const common::Time &_v)`
*Set a msgs::Time from a **common::Time** (p. 840).*
- `void gazebo::msgs::Set (msgs::PlaneGeom *_p, const math::Plane &_v)`
*Set a msgs::Plane from a **math::Plane** (p. 668).*
- `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.

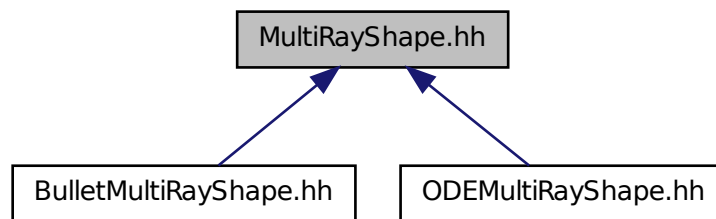
11.110 MultiRayShape.hh File Reference

```
#include <vector>
#include <string>
#include "math/Vector3.hh"
#include "math/Angle.hh"
#include "physics/Collision.hh"
#include "physics/Shape.hh"
#include "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**

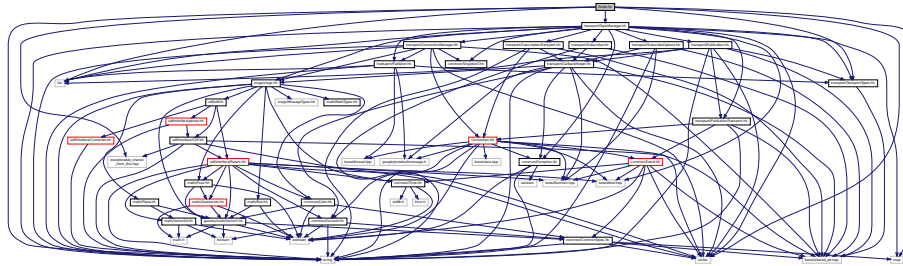
namespace for physics

11.111 Node.hh File Reference

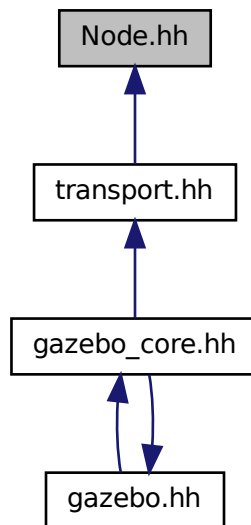
```
#include <boost/enable_shared_from_this.hpp>
```

```
#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.

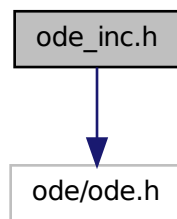
- namespace **gazebo::transport**

Transport namespace.

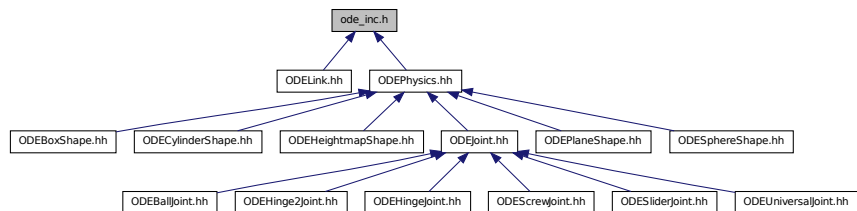
11.112 ode_inc.h File Reference

```
#include <ode/ode.h>
```

Include dependency graph for ode_inc.h:



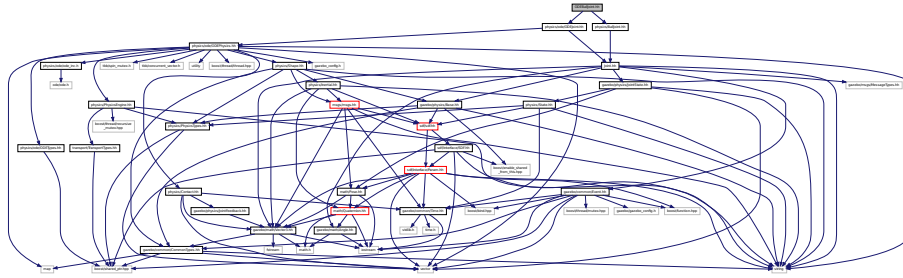
This graph shows which files directly or indirectly include this file:



11.113 ODEBallJoint.hh File Reference

```
#include "physics/BallJoint.hh"
#include "physics/ode/ODEJoint.hh"
```

Include dependency graph for ODEBallJoint.hh:



Classes

- class **gazebo::physics::ODEBallJoint**

An *ODEBallJoint* (p. 573).

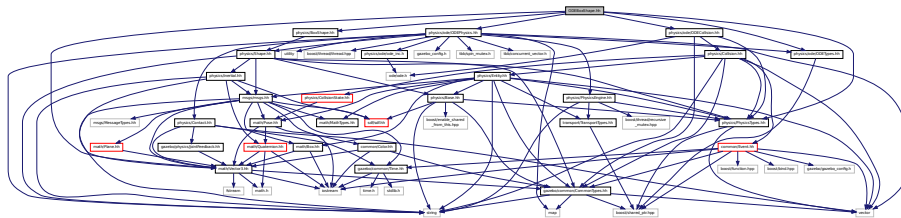
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.114 ODEBoxShape.hh File Reference

```
#include "math/Vector3.hh"
#include "physics/ode/ODEPhysics.hh"
#include "physics/ode/ODETypes.hh"
#include "physics/ode/ODECollision.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/BoxShape.hh"
```

Include dependency graph for ODEBoxShape.hh:



Classes

- class **gazebo::physics::ODEBoxShape**

ODE Box shape.

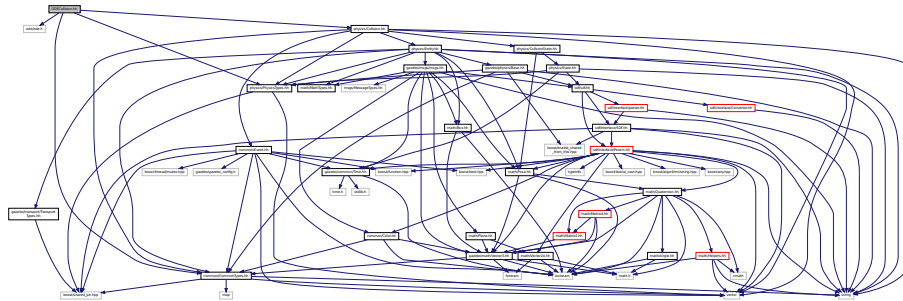
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

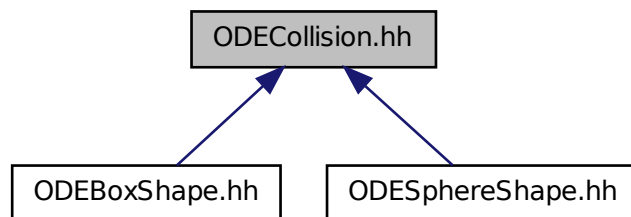
11.115 ODECollision.hh File Reference

```
#include "ode/ode.h"
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/Collision.hh"
```

Include dependency graph for ODECollision.hh:



This graph shows which files directly or indirectly include this file:



Classes

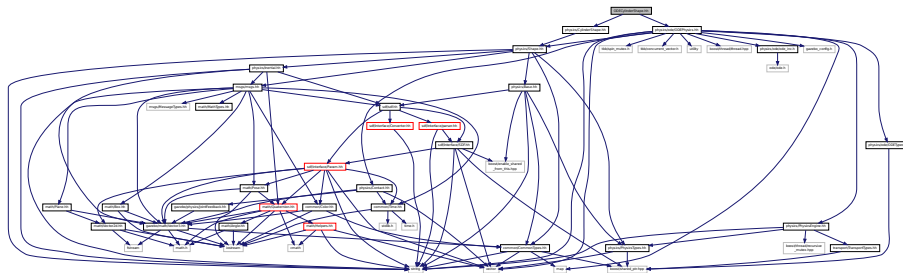
- class **gazebo::physics::ODECollision**
Base (p. 145) class for all ODE collisions.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.116 ODECylinderShape.hh File Reference

```
#include "physics/CylinderShape.hh"
#include "physics/ode/ODEPhysics.hh"
Include dependency graph for ODECylinderShape.hh:
```



Classes

- class **gazebo::physics::ODECylinderShape**
ODE cylinder shape.

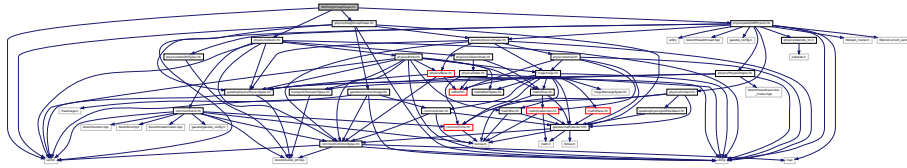
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.117 ODEHeightmapShape.hh File Reference

```
#include <vector>
#include "physics/HeightmapShape.hh"
#include "physics/ode/ODEPhysics.hh"
#include "physics/Collision.hh"
```

Include dependency graph for ODEHeightmapShape.hh:



Classes

- class **gazebo::physics::ODEHeightmapShape**
ODE Height map collision.

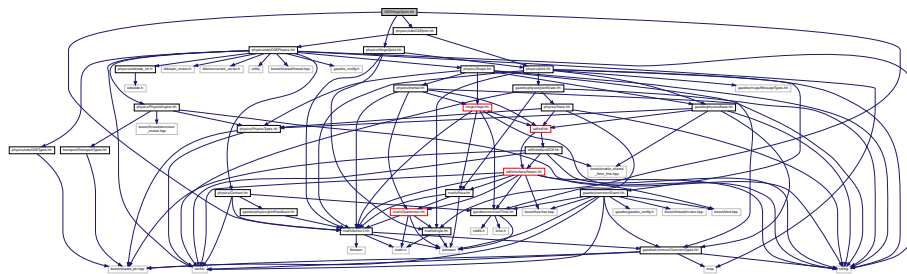
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.118 ODEHinge2Joint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/Hinge2Joint.hh"
#include "physics/ode/ODEJoint.hh"
```

Include dependency graph for ODEHinge2Joint.hh:



Classes

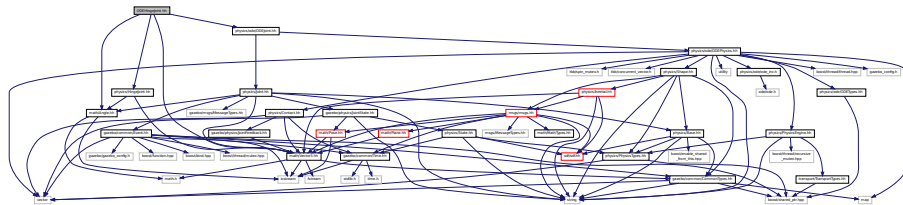
- class **gazebo::physics::ODEHinge2Joint**
A two axis hinge joint.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.119 ODEHingeJoint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/HingeJoint.hh"
#include "physics/ode/ODEJoint.hh"
Include dependency graph for ODEHingeJoint.hh:
```



Classes

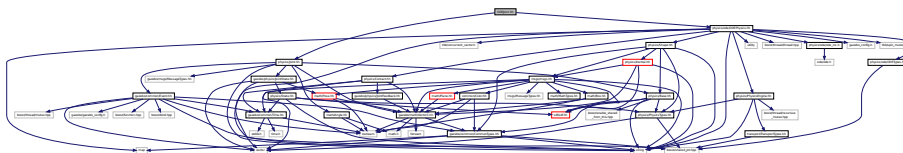
- class **gazebo::physics::ODEHingeJoint**
A single axis hinge joint.

Namespaces

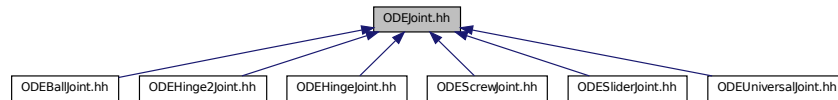
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.120 ODEJoint.hh File Reference

```
#include "physics/ode/ODEPhysics.hh"
#include "physics/Joint.hh"
Include dependency graph for ODEJoint.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ODEJoint**
ODE joint interface.

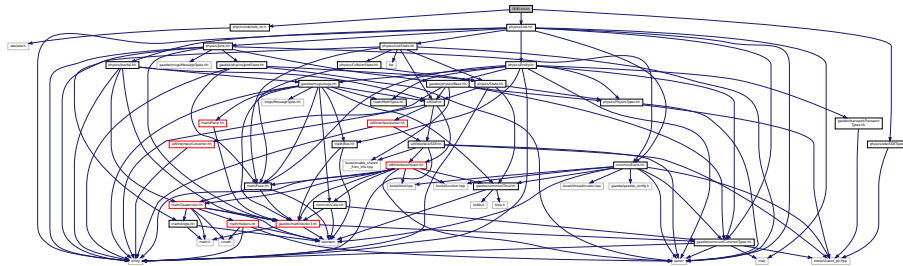
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.121 ODELink.hh File Reference

```
#include "physics/ode/ode_inc.h"
#include "physics/ode/ODETypes.hh"
#include "physics/Link.hh"
```

Include dependency graph for ODELink.hh:



Classes

- class **gazebo::physics::ODELink**
ODE Link (p. 454) class.

Namespaces

- namespace **gazebo**

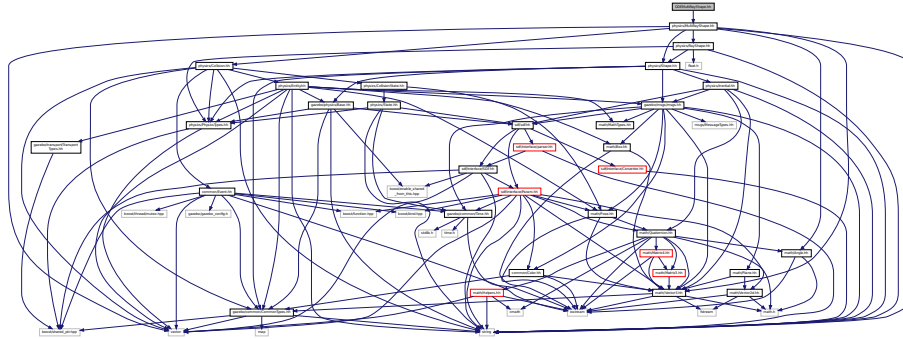
Forward declarations for the common classes.

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

11.122 ODEMultiRayShape.hh File Reference

```
#include "physics/MultiRayShape.hh"
```

Include dependency graph for ODEMultiRayShape.hh:



Classes

- class **gazebo::physics::ODEMultiRayShape**
*ODE specific version of **MultiRayShape** (p. 549).*

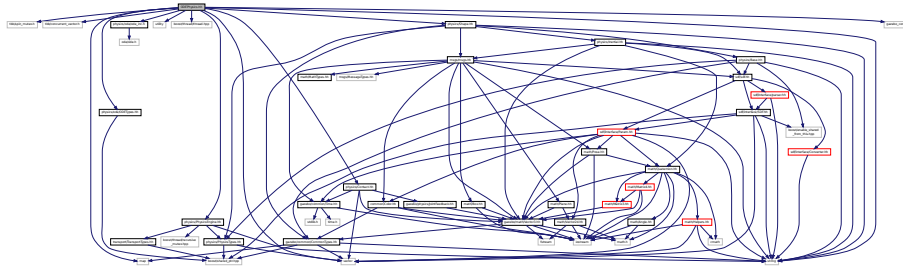
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

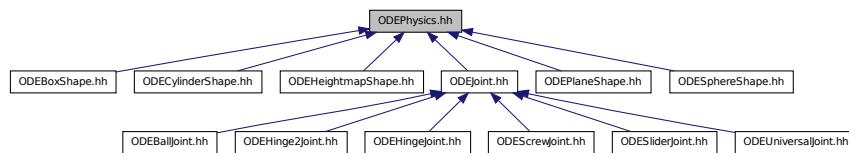
11.123 ODEPhysics.hh File Reference

```
#include <ttb/spin_mutex.h>
#include <ttb/concurrent_vector.h>
#include <map>
#include <string>
#include <vector>
#include <utility>
#include <boost/thread/thread.hpp>
#include "physics/ode/ode_inc.h"
#include "physics/ode/ODETypes.hh"
#include "physics/PhysicsEngine.hh"
#include "physics/Contact.hh"
#include "physics/Shape.hh"
#include "gazebo_config.h"
```

Include dependency graph for ODEPhysics.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ContactFeedback**
data structure for contact feedbacks
- class **gazebo::physics::ODEPhysics**
ODE physics engine.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

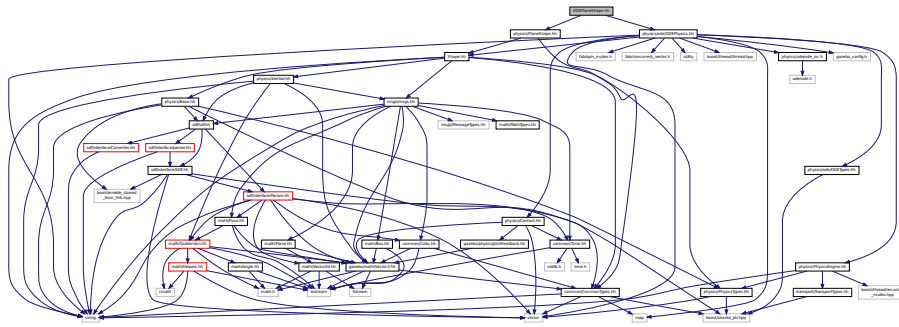
Typedefs

- typedef ODEPhysics * **gazebo::physics::ODEPhysicsPtr**

11.124 ODEPlaneShape.hh File Reference

```
#include "physics/PlaneShape.hh"
#include "physics/ode/ODEPhysics.hh"
```

Include dependency graph for ODEPlaneShape.hh:



Classes

- class **gazebo::physics::ODEPlaneShape**
An ODE Plane shape.

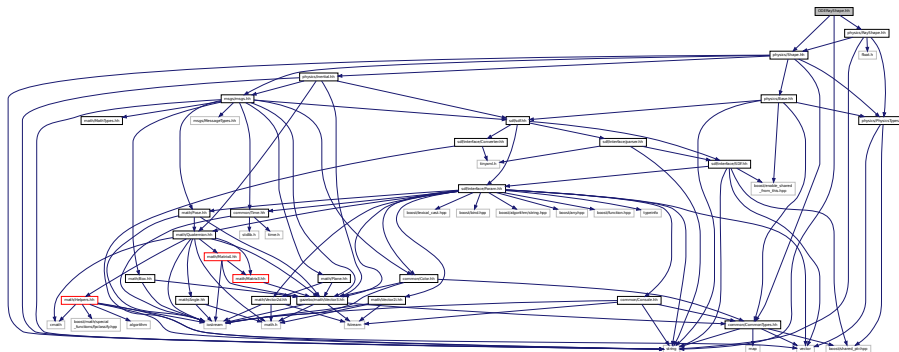
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.125 ODERayShape.hh File Reference

```
#include <string>
#include "physics/RayShape.hh"
#include "physics/Shape.hh"
```

Include dependency graph for ODERayShape.hh:



Classes

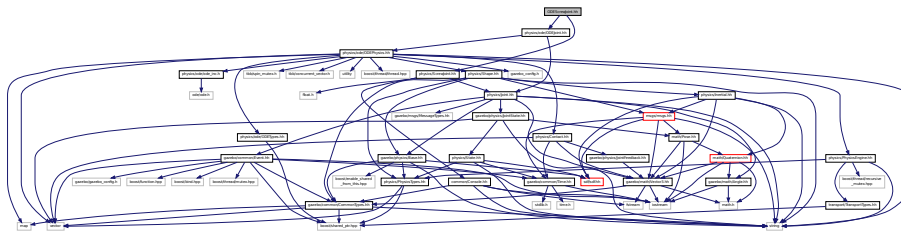
- class **gazebo::physics::ODERayShape**
Ray collision.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.126 ODEScrewJoint.hh File Reference

```
#include "physics/ScrewJoint.hh"
#include "physics/ode/ODEJoint.hh"
Include dependency graph for ODEScrewJoint.hh:
```



Classes

- class **gazebo::physics::ODEScrewJoint**
A screw joint.

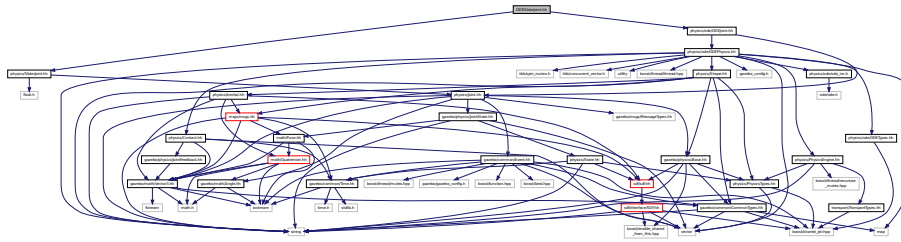
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.127 ODESliderJoint.hh File Reference

```
#include "physics/SliderJoint.hh"
#include "physics/ode/ODEJoint.hh"
```

Include dependency graph for ODESliderJoint.hh:



Classes

- class **gazebo::physics::ODESliderJoint**
A slider joint.

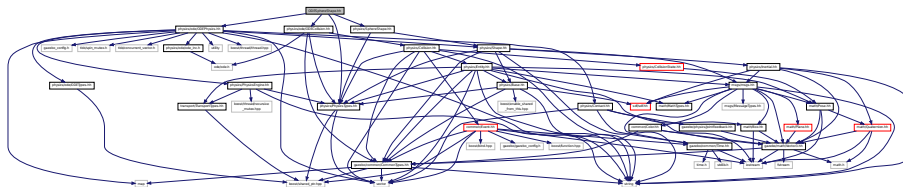
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.128 ODESphereShape.hh File Reference

```
#include "physics/ode/ODEPhysics.hh"
#include "physics/ode/ODECollision.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/SphereShape.hh"
```

Include dependency graph for ODESphereShape.hh:



Classes

- class **gazebo::physics::ODESphereShape**
A ODE sphere shape.

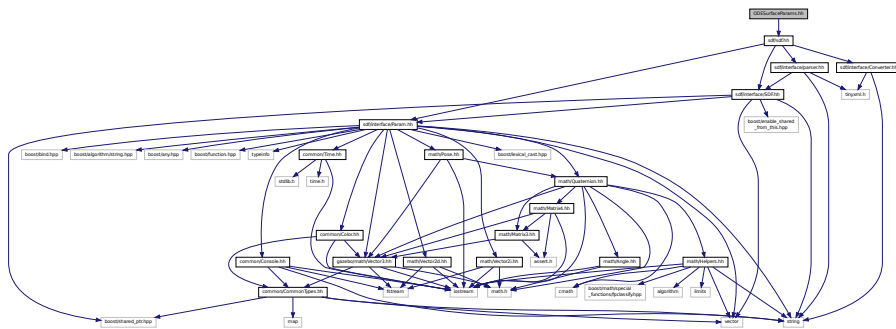
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.129 ODESurfaceParams.hh File Reference

```
#include "sdf/sdf.hh"
```

Include dependency graph for ODESurfaceParams.hh:



Classes

- class **gazebo::physics::ODESurfaceParams**
Surface params.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

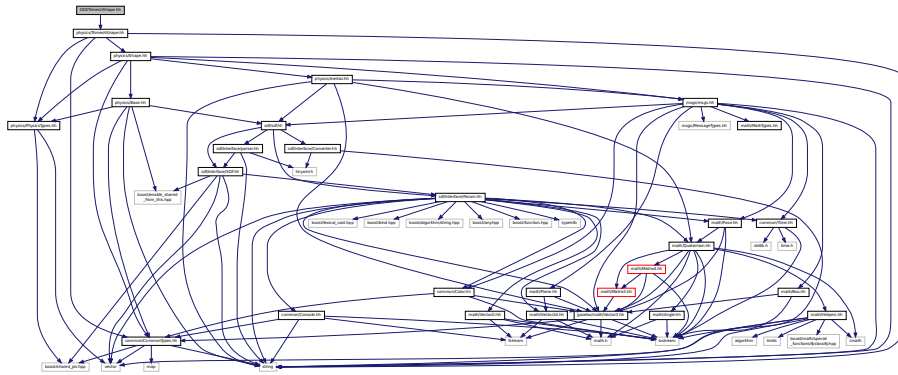
Typedefs

- typedef ODESurfaceParams * **gazebo::physics::ODESurfaceParamsPtr**

11.130 ODETrimeshShape.hh File Reference

```
#include "physics/TrimeshShape.hh"
```


Include dependency graph for ODETrimeshShape.hh:



Classes

- class **gazebo::physics::ODETrimeshShape**
Triangle mesh collision.

Namespaces

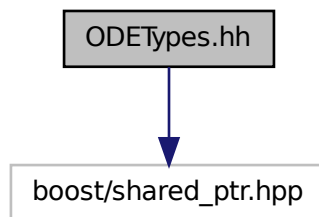
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.131 ODETypes.hh File Reference

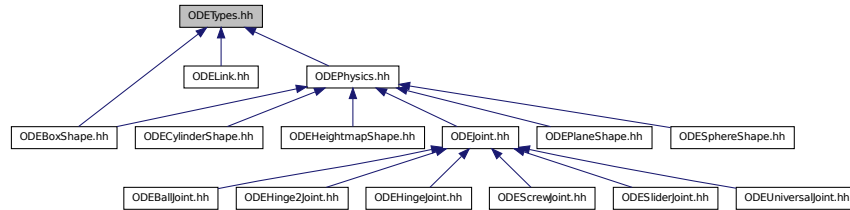
ODE wrapper forward declarations and typedefs.

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for ODETypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Typedefs

- typedef ODECollision * **gazebo::physics::ODECollisionPtr**
- typedef ODELink * **gazebo::physics::ODELinkPtr**
- typedef ODERayShape * **gazebo::physics::ODERayShapePtr**

11.131.1 Detailed Description

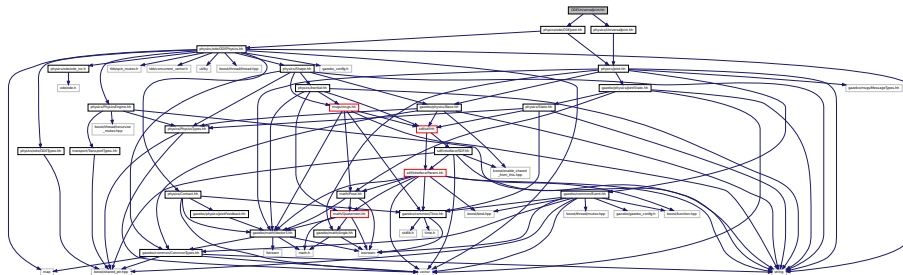
ODE wrapper forward declarations and typedefs.

11.132 ODEUniversalJoint.hh File Reference

```
#include "physics/UniversalJoint.hh"
```

```
#include "physics/ode/ODEJoint.hh"
```

Include dependency graph for ODEUniversalJoint.hh:



Classes

- class **gazebo::physics::ODEUniversalJoint**

A universal joint.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

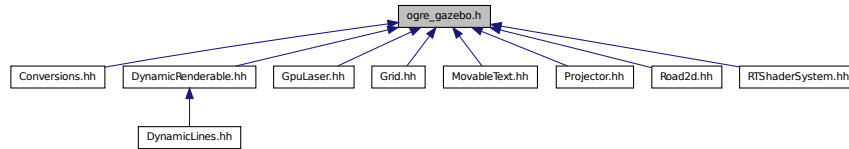
namespace for physics

11.133 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:

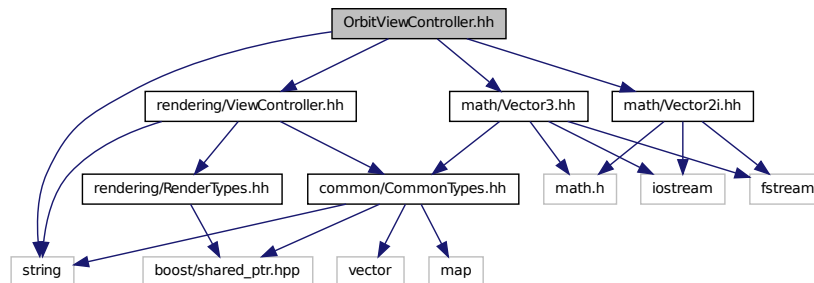


11.134 OrbitViewController.hh File Reference

```

#include <string>
#include "rendering/ViewController.hh"
#include "math/Vector3.hh"
#include "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.

11.135 Param.hh File Reference

```

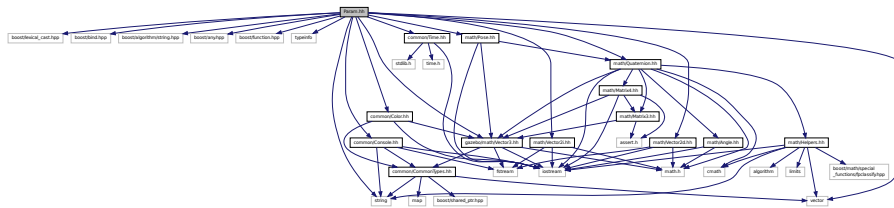
#include <boost/lexical_cast.hpp>
  
```

```

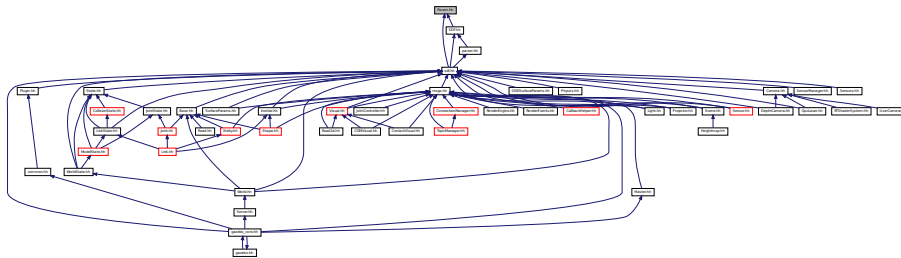
#include <boost/bind.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/any.hpp>
#include <boost/function.hpp>
#include <typeinfo>
#include <string>
#include <vector>
#include "common/Console.hh"
#include "common/Color.hh"
#include "common/Time.hh"
#include "math/Vector3.hh"
#include "math/Vector2i.hh"
#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. 762) 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)

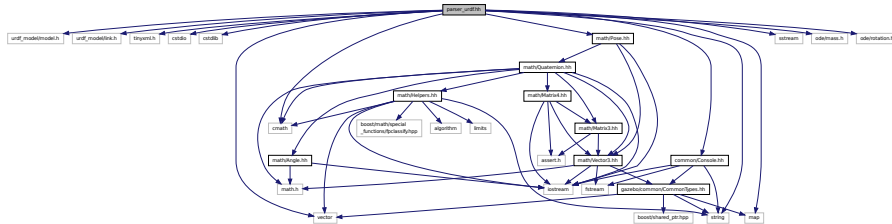
*Populate the **SDF** (p. 762) values from a file.*
- bool **sdf::readString** (const std::string &_xmlString, SDFPtr _sdf)

*Populate the **SDF** (p. 762) values from a string.*
- bool **sdf::readString** (const std::string &_xmlString, ElementPtr _sdf)
- bool **sdf::readXml** (TiXmlElement *_xml, ElementPtr _sdf)

11.137 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**

Namespaces

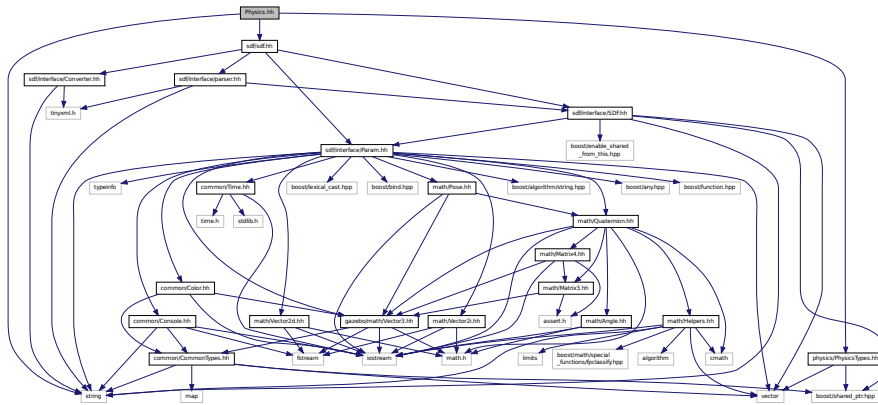
- namespace **urdf2gazebo**
namespace for URDF to SDF parser

Typedefs

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

11.138 Physics.hh File Reference

```
#include <string>
#include "physics/PhysicsTypes.hh"
#include "sdf/sdf.hh"
Include dependency graph for Physics.hh:
```



Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Functions

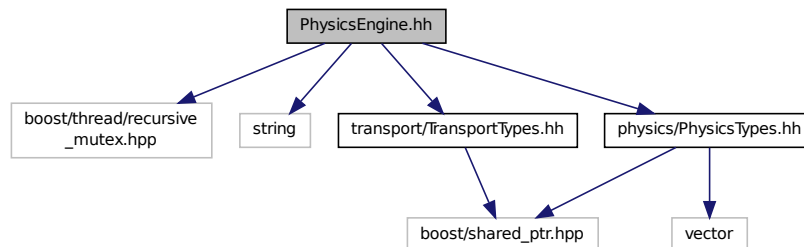
- WorldPtr **gazebo::physics::create_world** (const std::string &_name="")
Create a world given a name.
- bool **gazebo::physics::fini** ()
Finalize transport by calling `gazebo::transport::fini` (p. 90).
- WorldPtr **gazebo::physics::get_world** (const std::string &_name="")
Returns a pointer to a world 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. 838)'s and call **gazebo::transport::init** (p. 90).*
- void **gazebo::physics::load_world** (WorldPtr _world, sdf::ElementPtr _sdf)
*Load world from **sdf::Element** (p. 332) pointer.*
- void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)
*load multiple worlds from single **sdf::Element** (p. 332) pointer*
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)
*Pause world by calling **World::SetPaused** (p. 964).*
- 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. 964) 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. 965) given a pointer to it.*
- void **gazebo::physics::stop_worlds** ()
stop multiple worlds stored in static variable gazebo::g_worlds

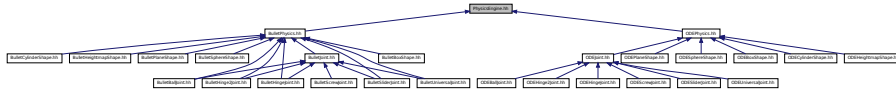
11.139 PhysicsEngine.hh File Reference

```
#include <boost/thread/recursive_mutex.hpp>
#include <string>
#include "transport/TransportTypes.hh"
#include "physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsEngine.hh:



This graph shows which files directly or indirectly include this file:



Classes

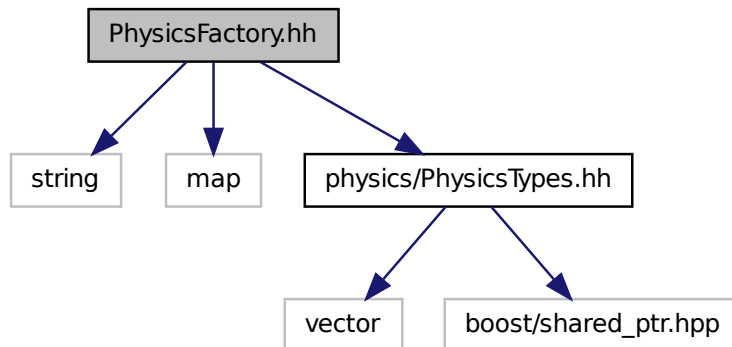
- class **gazebo::physics::PhysicsEngine**
Base (p. 145) class for a physics engine.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.140 PhysicsFactory.hh File Reference

```
#include <string>
#include <map>
#include "physics/PhysicsTypes.hh"
Include dependency graph for PhysicsFactory.hh:
```



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.

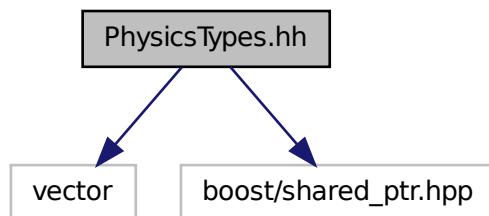
Typedefs

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

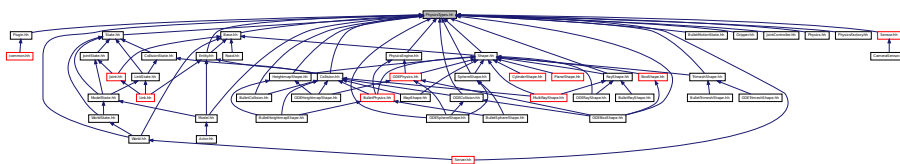
11.141 PhysicsTypes.hh File Reference

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**
Collides with everything.
- **#define GZ_FIXED_COLLIDE 0x00000001**
Collides with everything else but other fixed.
- **#define GZ_GHOST_COLLIDE 0x10000000**
Collides with everything else but other ghost.
- **#define GZ_NONE_COLLIDE 0x00000000**
Collides with nothing.
- **#define GZ_SENSOR_COLLIDE 0x00000003**
Collides with everything else but other 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**

11.141.1 Detailed Description

default namespace for gazebo

11.141.2 Macro Definition Documentation

11.141.2.1 `#define GZ_ALL_COLLIDE 0xFFFFFFFF`

Collides with everything.

11.141.2.2 `#define GZ_FIXED_COLLIDE 0x00000001`

Collides with everything else but other fixed.

11.141.2.3 `#define GZ_GHOST_COLLIDE 0x10000000`

Collides with everything else but other ghost.

11.141.2.4 `#define GZ_NONE_COLLIDE 0x00000000`

Collides with nothing.

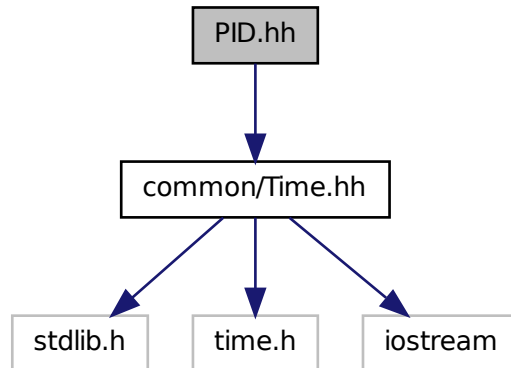
11.141.2.5 `#define GZ_SENSOR_COLLIDE 0x00000003`

Collides with everything else but other sensors.

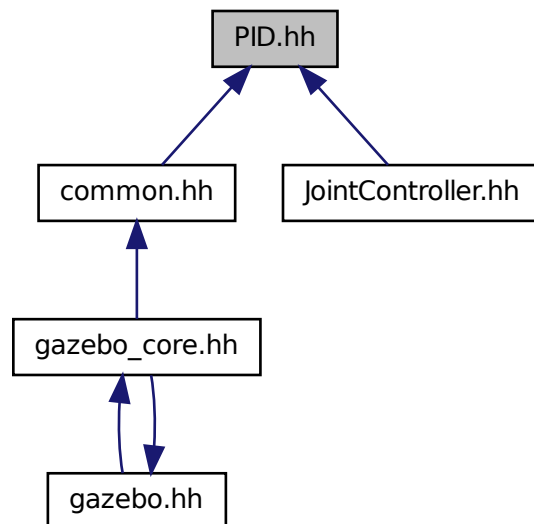
11.142 PID.hh File Reference

```
#include "common/Time.hh"
```

Include dependency graph for PID.hh:



This graph shows which files directly or indirectly include this file:



Classes

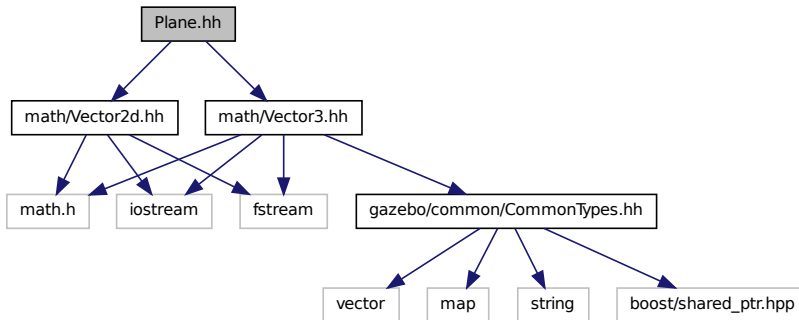
- class **gazebo::common::PID**
Generic *PID* (p. 664) controller class.

Namespaces

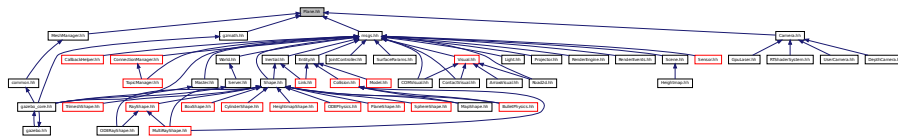
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

11.143 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

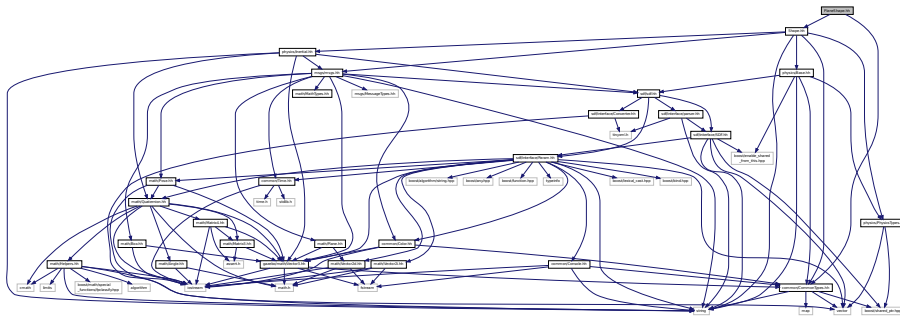
- class **gazebo::math::Plane**
A plane and related functions.

Namespaces

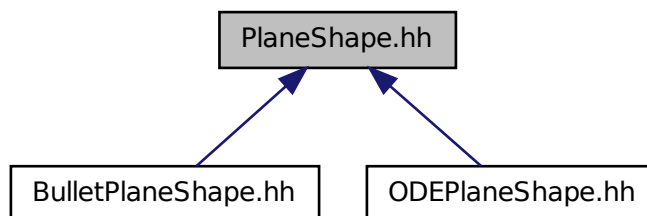
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.144 PlaneShape.hh File Reference

```
#include "common/CommonTypes.hh"
#include "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. 262) for an infinite plane.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

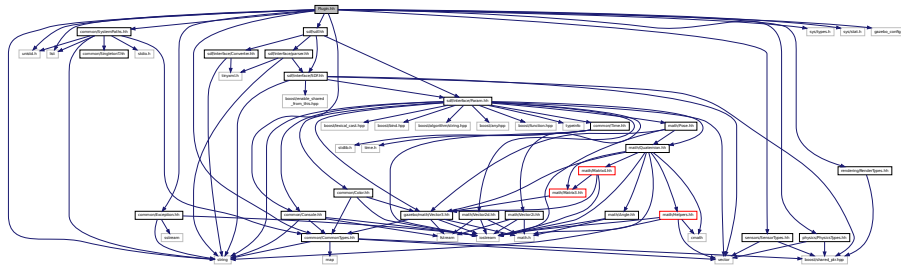
- namespace **gazebo::physics**

namespace for physics

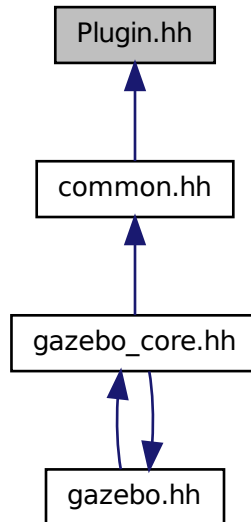
11.145 Plugin.hh File Reference

```
#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. 521).*
- 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. 954).*

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.

11.145.1 Macro Definition Documentation

11.145.1.1 #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

11.145.1.2 #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

11.145.1.3 #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

11.145.1.4 #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

11.145.1.5 #define GZ_REGISTER_WORLD_PLUGIN(*classname*)**Value:**

```
extern "C" gazebo::WorldPlugin *RegisterPlugin();   gazebo::WorldPlugin *RegisterPlugin() \
{\
    return new classname();\
}
```

Plugin registration function for world plugin.

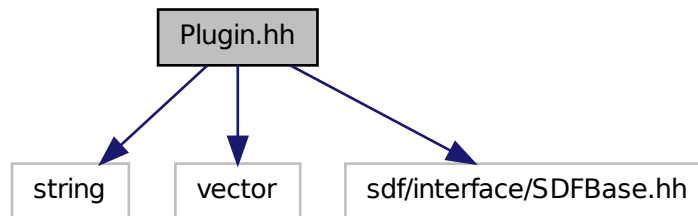
Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.146 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

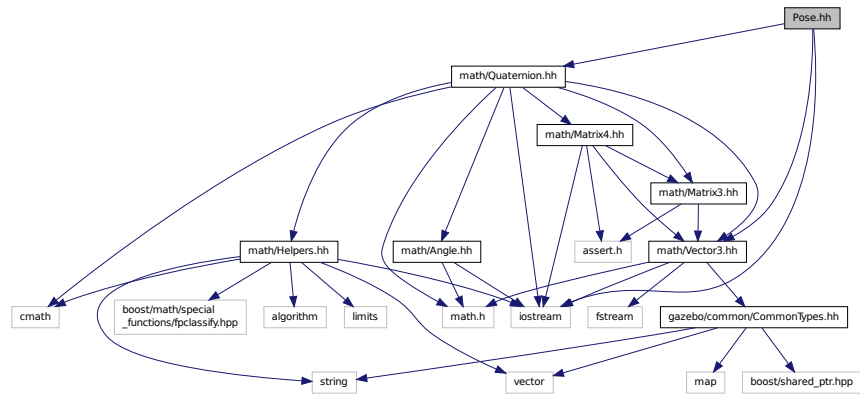
- namespace **sdf**

namespace for Simulation Description Format parser

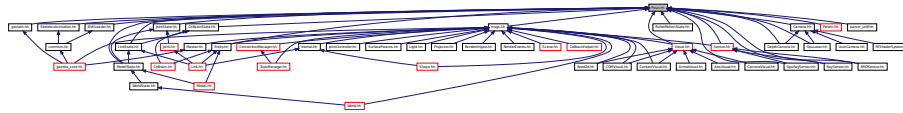
11.147 Pose.hh File Reference

```
#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.

11.148 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.

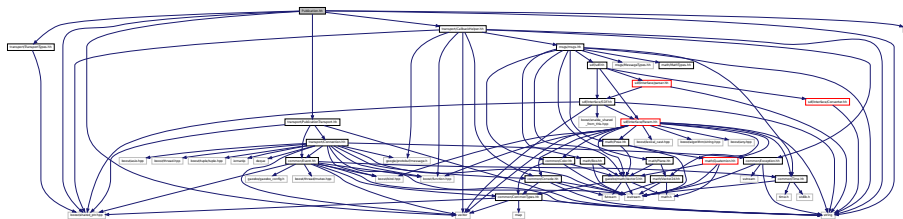
- namespace **gazebo::rendering**

Rendering namespace.

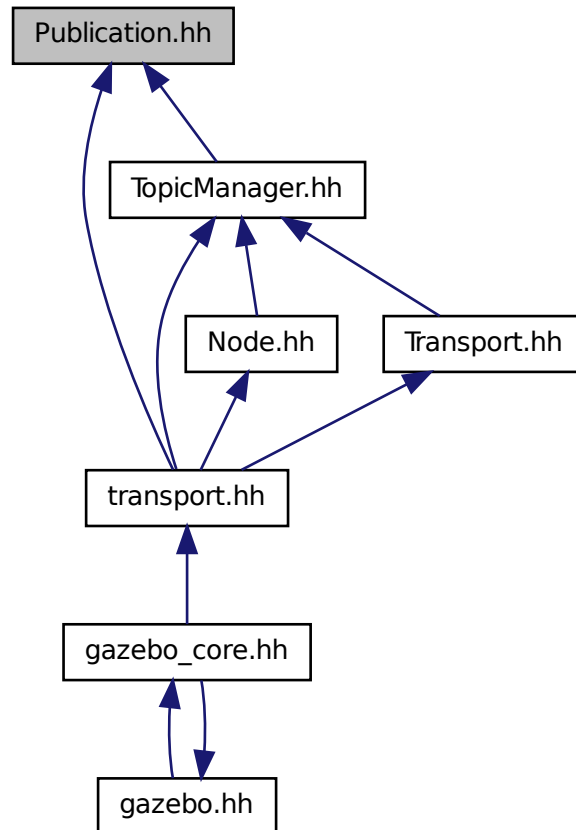
11.149 Publication.hh File Reference

```
#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:



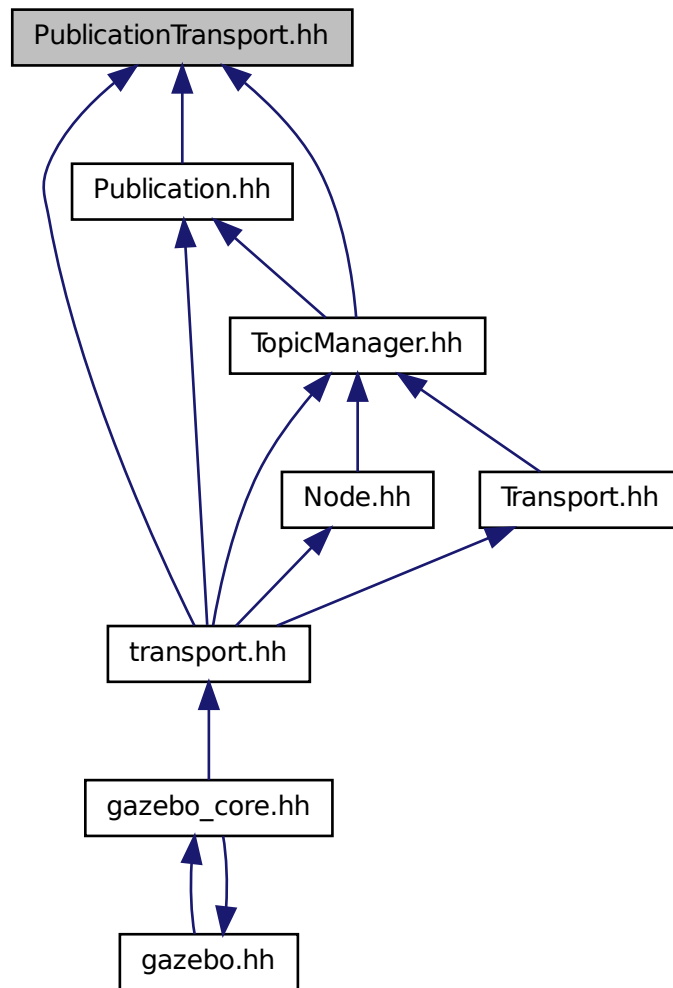
Classes

- class **gazebo::transport::Publication**
A publication for a topic.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**
Transport namespace.

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::PublicationTransport**
Reads data from a remote advertiser, and passes the data along to local subscribers.

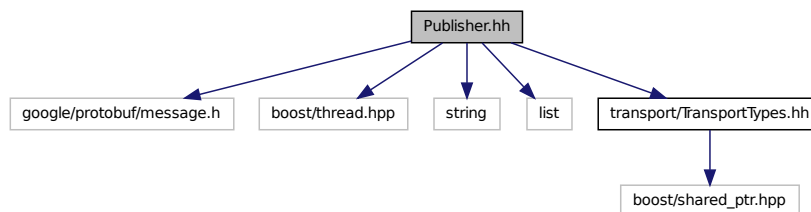
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**
Transport namespace.

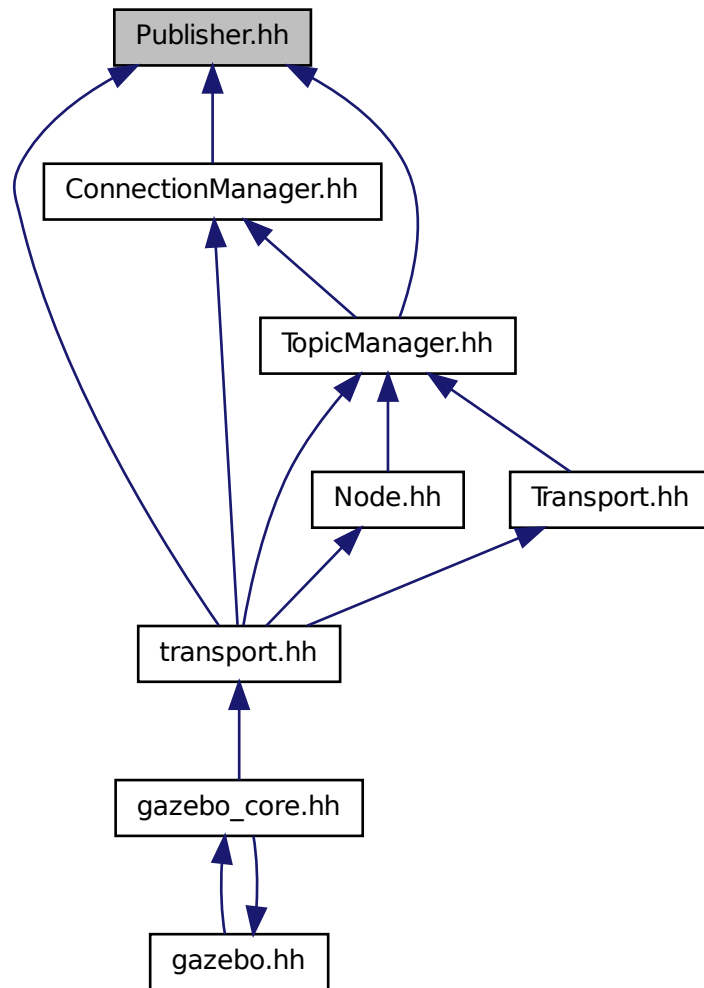
11.151 Publisher.hh File Reference

```
#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.

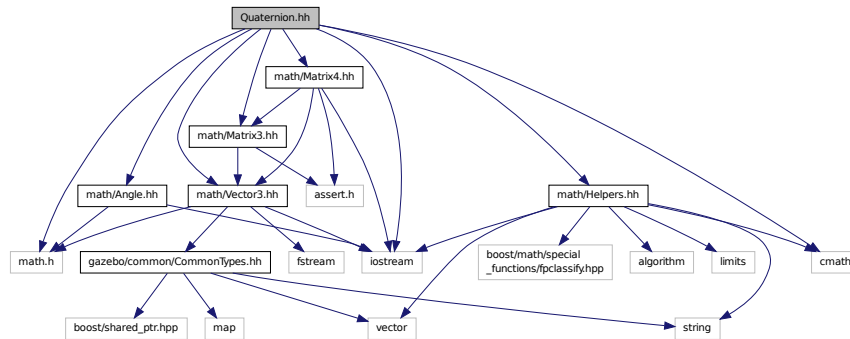
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**
Transport namespace.

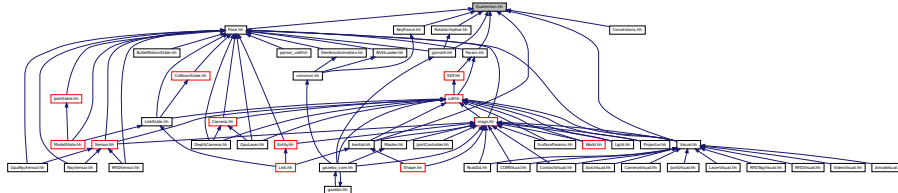
11.152 Quaternion.hh File Reference

```
#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 "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.

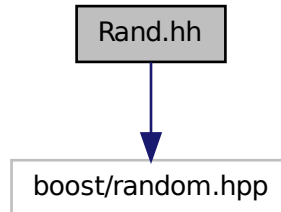
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

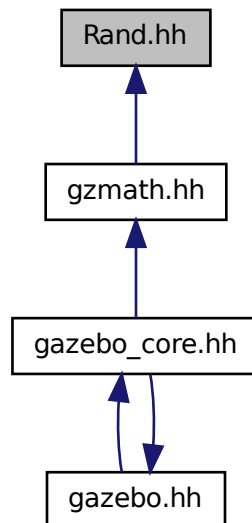
11.153 Rand.hh File Reference

```
#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.

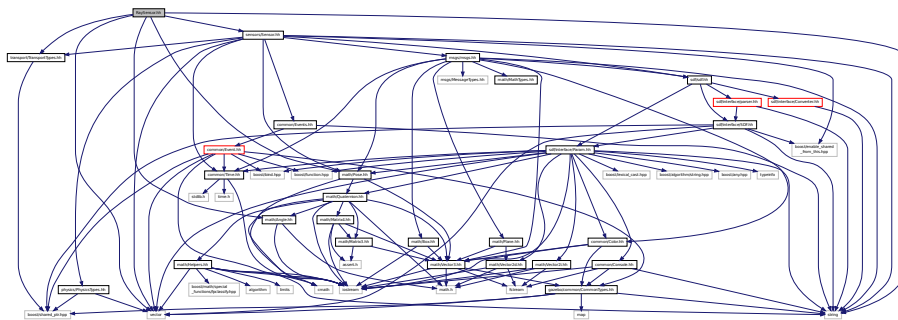
Typedefs

- 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**

11.154 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**

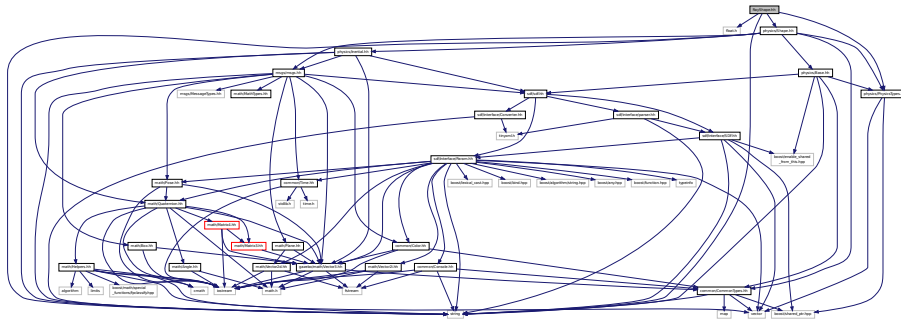
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

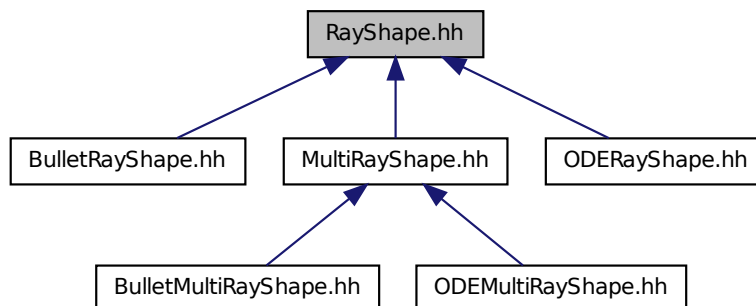
11.155 RayShape.hh File Reference

```
#include <float.h>
#include <string>
#include "physics/PhysicsTypes.hh"
#include "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. 145) class for Ray collision geometry.

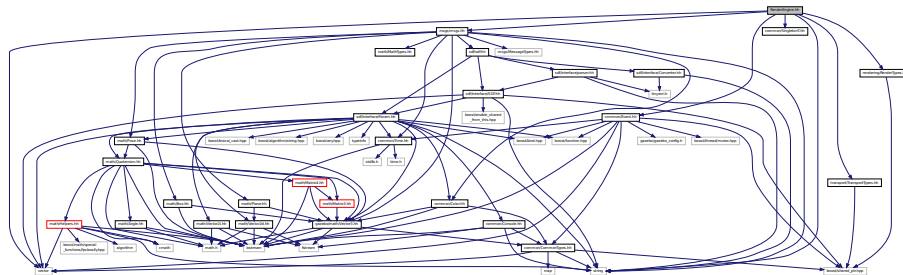
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.156 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

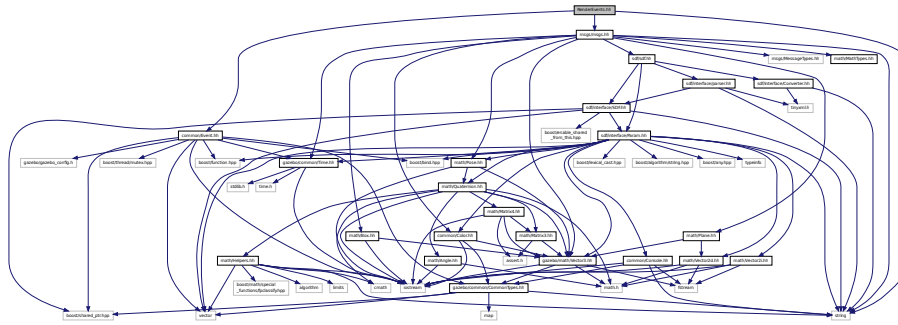
- class **gazebo::rendering::RenderEngine**
Adaptor to Ogre3d.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**

11.157 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.

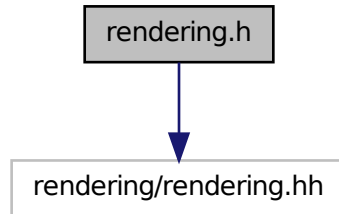
- namespace **gazebo::rendering**

Rendering namespace.

11.158 rendering.h File Reference

```
#include "rendering/rendering.hh"
```

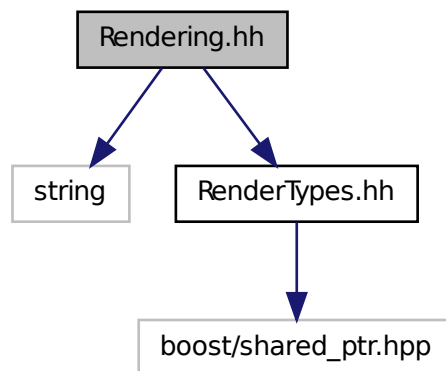
Include dependency graph for rendering.h:



11.159 Rendering.hh File Reference

```
#include <string>
#include "RenderTypes.hh"
```

Include dependency graph for Rendering.hh:



Namespaces

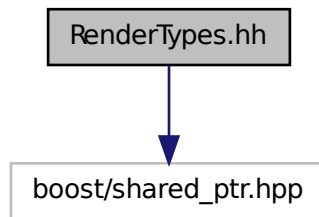
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

Functions

- rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations)
 - create **rendering::Scene** (p. 746) 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. 746) 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. 746) by name*

11.160 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 0xFFFFFFFF`
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.

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.

11.160.1 Macro Definition Documentation

11.160.1.1 `#define GZ_VISIBILITY_ALL 0xFFFFFFFF`

Render everything visibility mask.

11.160.1.2 `#define GZ_VISIBILITY_GUI 0x00000001`

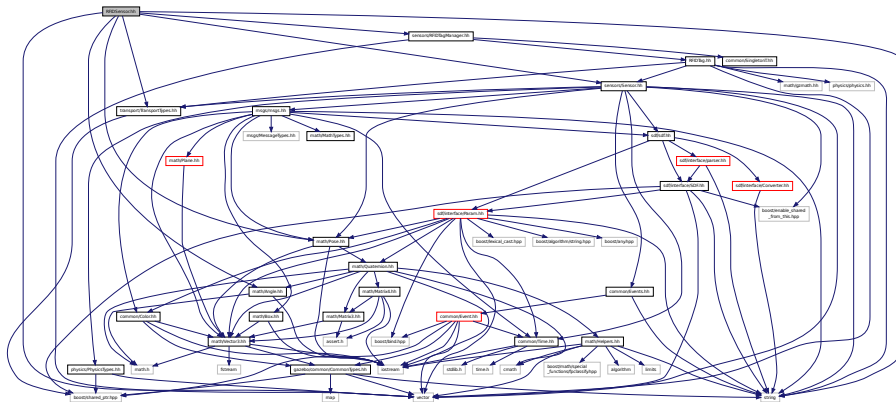
Render GUI visuals mask.

11.160.1.3 `#define GZ_VISIBILITY_NOT_SELECTABLE 0x00000002`

Render visuals that are not selectable mask.

11.161 RFIDSensor.hh File Reference

```
#include <vector>
#include <string>
#include "transport/TransportTypes.hh"
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "sensors/Sensor.hh"
#include "sensors/RFIDTagManager.hh"
Include dependency graph for RFIDSensor.hh:
```



Classes

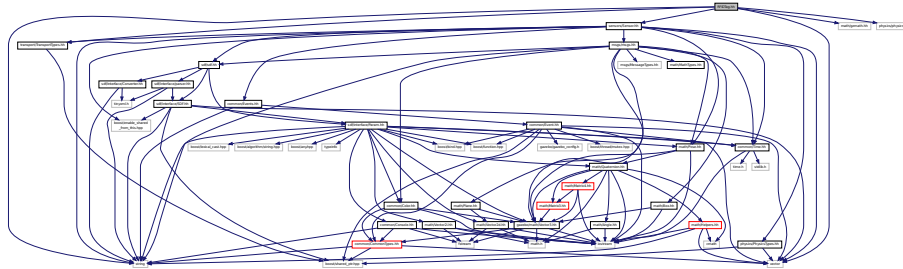
- class **gazebo::sensors::RFIDSensor**
Sensor (p. 765) class for RFID type of sensor.

Namespaces

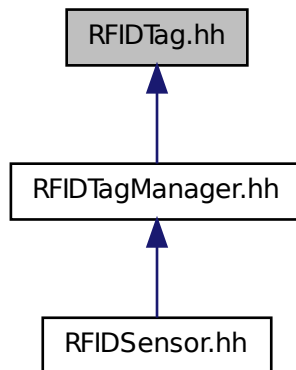
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.162 RFIDTag.hh File Reference

```
#include <vector>
#include <string>
#include "transport/TransportTypes.hh"
#include "sensors/Sensor.hh"
#include "math/gzmath.hh"
#include "physics/physics.hh"
Include dependency graph for RFIDTag.hh:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::RFIDTag**
RFIDTag (p. 729) to interact with *RFIDTagSensors* Nate check.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

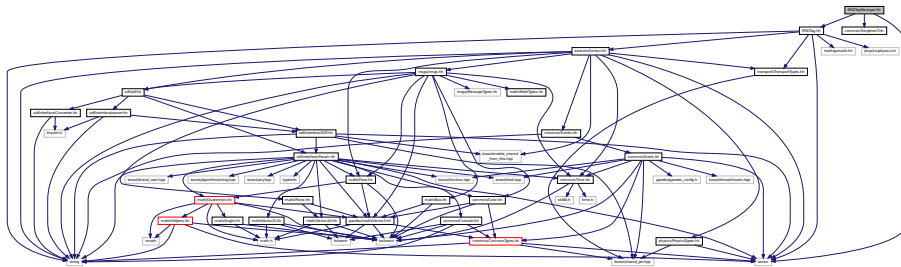
- namespace **gazebo::sensors**

Sensors namespace.

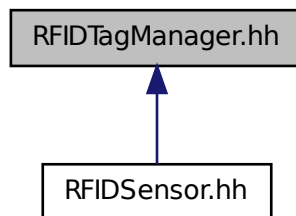
11.163 RFIDTagManager.hh File Reference

```
#include <vector>
#include "common/SingletonT.hh"
#include "RFIDTag.hh"
```

Include dependency graph for RFIDTagManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::RFIDTagManager**

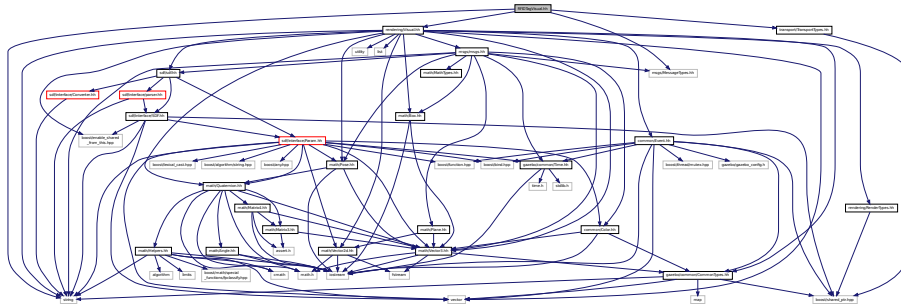
Nate fill in

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::sensors**
Sensors namespace.

11.164 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.

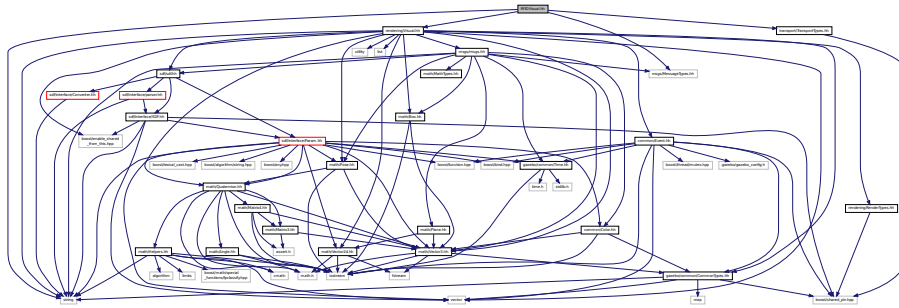
- namespace **gazebo::rendering**

Rendering namespace.

11.165 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.

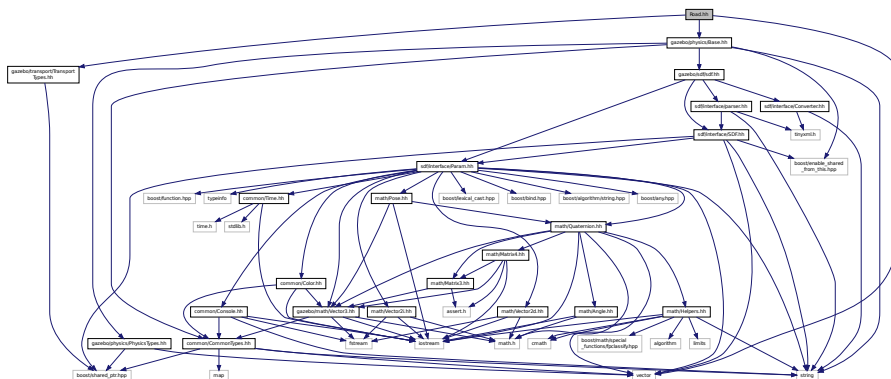
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

11.166 Road.hh File Reference

```
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/Base.hh"
```

Include dependency graph for Road.hh:



Classes

- class **gazebo::physics::Road**
*for building a **Road** (p. 736) from SDF*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.167 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**

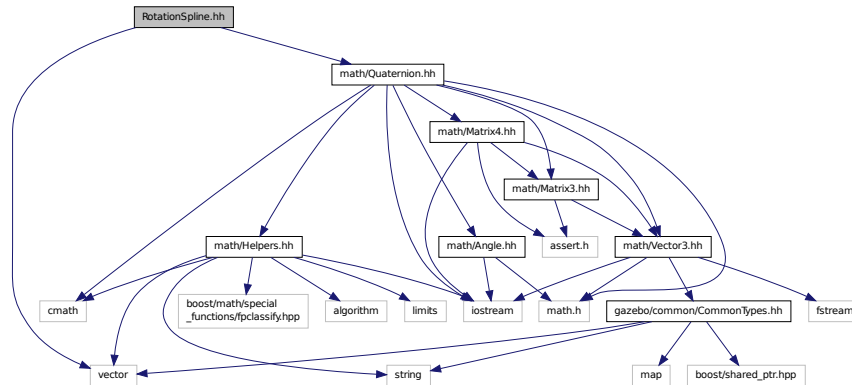
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

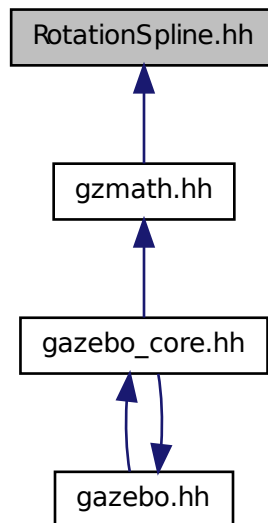
11.168 RotationSpline.hh File Reference

```
#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. 808) for rotations.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

11.169 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. 118)'s Run-Time Shader system.*

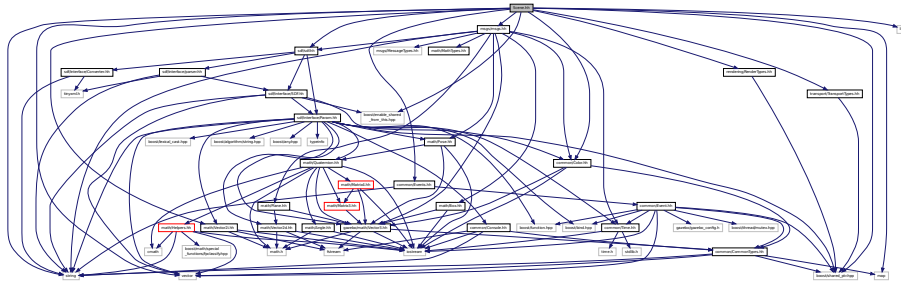
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.

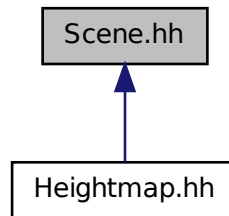
11.170 Scene.hh File Reference

```
#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.

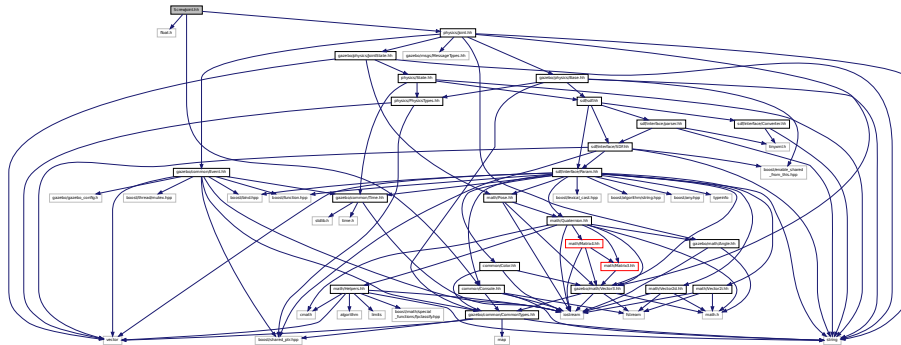
Namespaces

- namespace **boost**
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::rendering**
Rendering namespace.
- namespace **Ogre**
- namespace **SkyX**

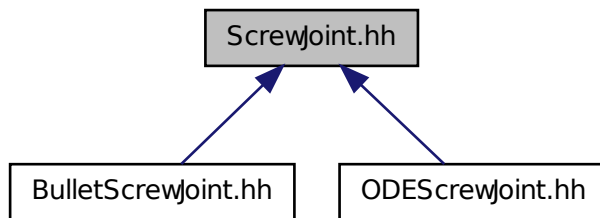
11.171 ScrewJoint.hh File Reference

```
#include <float.h>
```

```
#include "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.

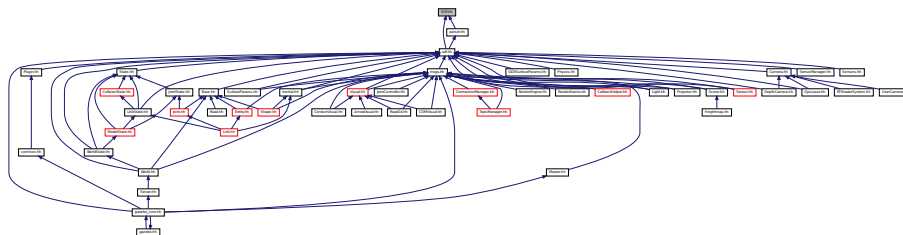
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.172 sdf.hh File Reference

```
#include "sdf/interface/SDF.hh"
```


This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Element**
SDF (p. 762) *Element* (p. 332) class.
- class **sdf::SDF**
Base SDF (p. 762) class.

Namespaces

- namespace **sdf**
namespace for Simulation Description Format parser

Macros

- #define **SDF_VERSION** "1.2"

Typedefs

- typedef Element * **sdf::ElementPtr**
- typedef std::vector< ElementPtr > **sdf::ElementPtr_V**
- typedef SDF * **sdf::SDFPtr**

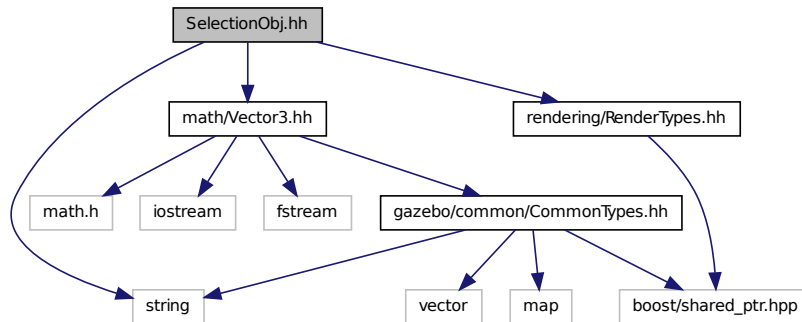
11.173.1 Macro Definition Documentation

11.173.1.1 #define SDF_VERSION "1.2"

11.174 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.

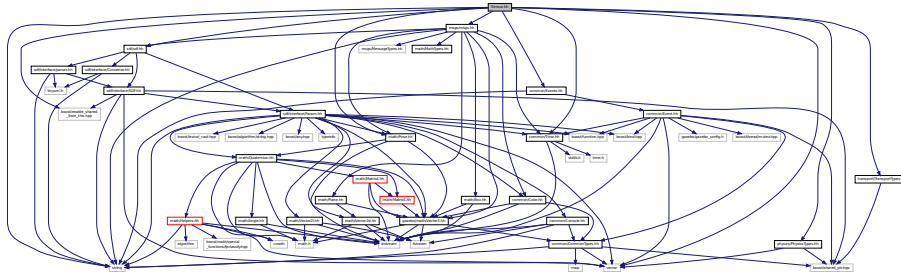
11.175 Sensor.hh File Reference

```

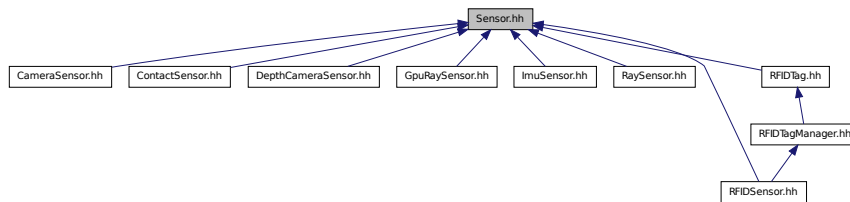
#include <boost/enable_shared_from_this.hpp>
#include <vector>
#include <string>
#include "sdf/sdf.hh"
#include "physics/PhysicsTypes.hh"
#include "msgs/msgs.hh"
#include "common/Events.hh"
#include "common/Time.hh"
#include "math/Pose.hh"
#include "transport/TransportTypes.hh"

```

Include dependency graph for Sensor.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class `gazebo::sensors::Sensor`

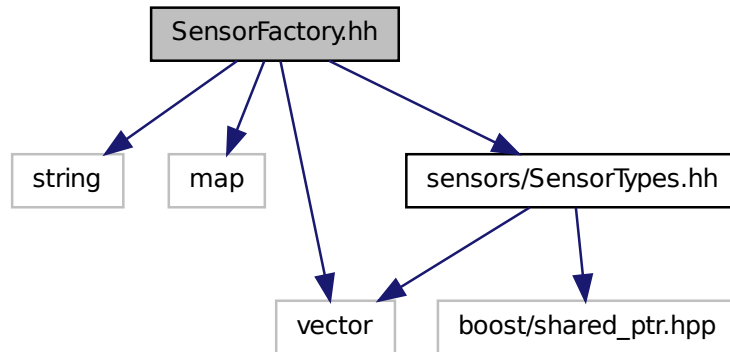
Namespaces

- namespace `gazebo`
Forward declarations for the common classes.
- namespace `gazebo::sensors`
Sensors namespace.

11.176 SensorFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "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

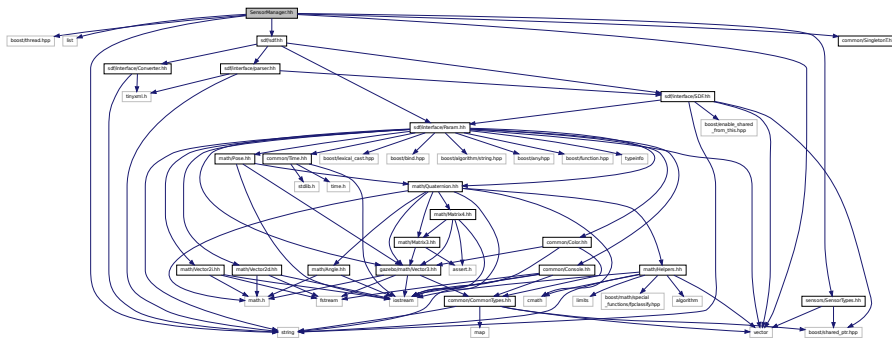
- typedef Sensor ***(* gazebo::sensors::SensorFactoryFn)()**

11.177 SensorManager.hh File Reference

```
#include <boost/thread.hpp>
```

```
#include <list>
#include <string>
#include <vector>
#include "common/SingletonT.hh"
#include "sensors/SensorTypes.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for SensorManager.hh:



Classes

- class **gazebo::sensors::SensorManager**

Class to manage and update all sensors.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::sensors**

Sensors namespace.

11.178 Sensors.hh File Reference

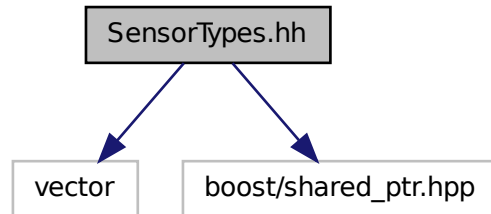
```
#include <string>
#include "sdf/sdf.hh"
#include "sensors/SensorTypes.hh"
```


11.179 SensorTypes.hh File Reference

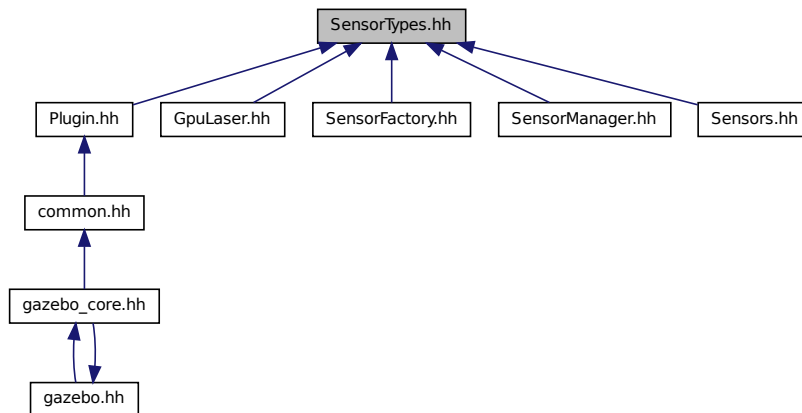
Forward declarations and typedefs for sensors.

```
#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**

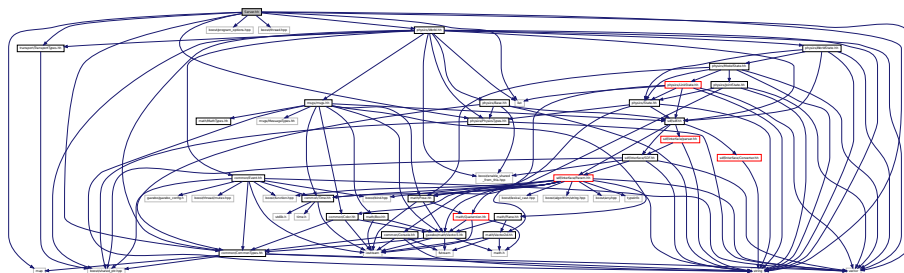
11.179.1 Detailed Description

Forward declarations and typedefs for sensors.

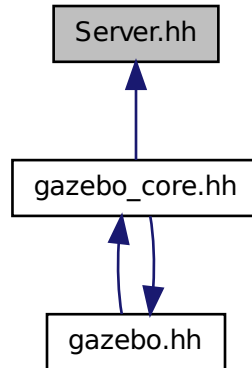
11.180 Server.hh File Reference

```
#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

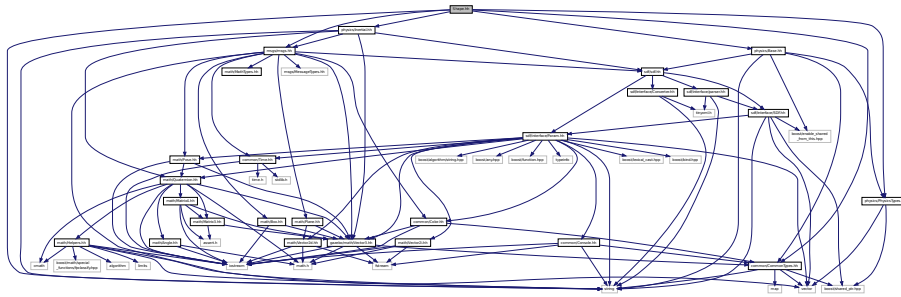
- namespace **boost**
- namespace **gazebo**

Forward declarations for the common classes.

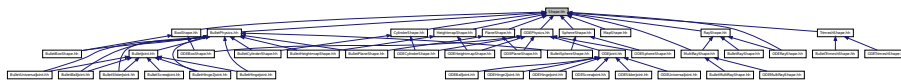
11.181 Shape.hh File Reference

```
#include <string>
#include "msgs/msgs.hh"
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/Inertial.hh"
#include "physics/Base.hh"
```

Include dependency graph for Shape.hh:



This graph shows which files directly or indirectly include this file:



Classes

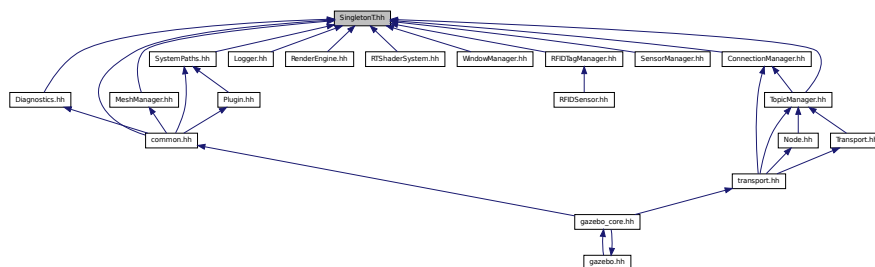
- class **gazebo::physics::Shape**
Base (p. 145) class for all shapes.

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.182 SingletonT.hh File Reference

This graph shows which files directly or indirectly include this file:



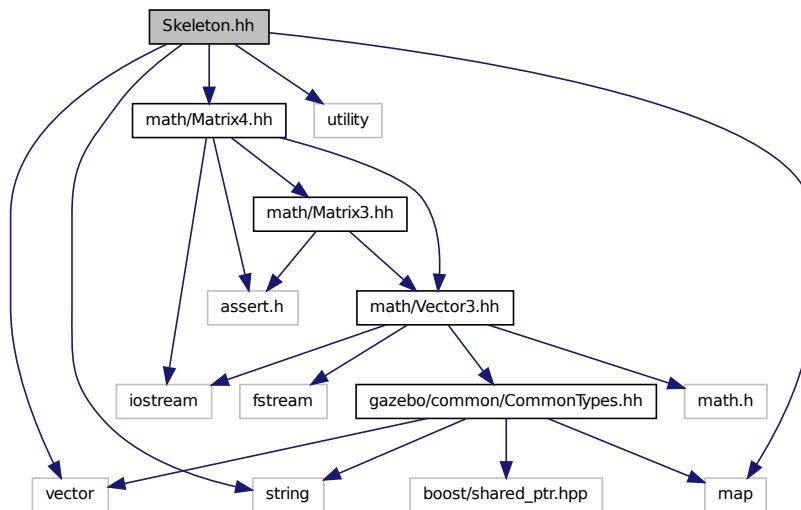
Classes

- class **SingletonT**< T >

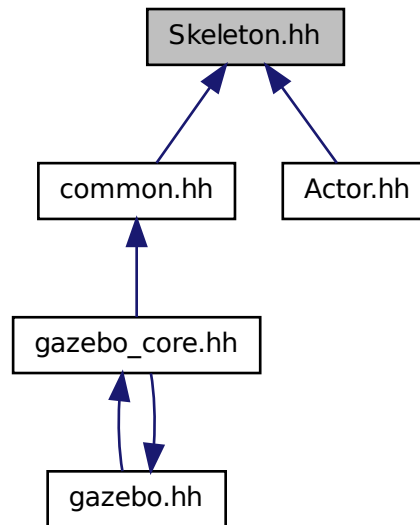
Singleton template class.

11.183 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**
A transformation node.
- 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

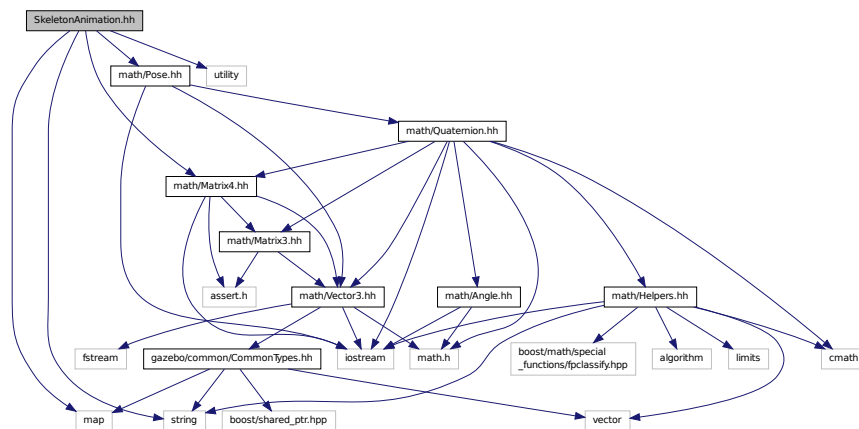
- typedef `std::map< unsigned int, SkeletonNode * >` **gazebo::common::NodeMap**
- typedef `std::map< unsigned int, SkeletonNode * >::iterator` **gazebo::common::NodeMapIter**

- 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**

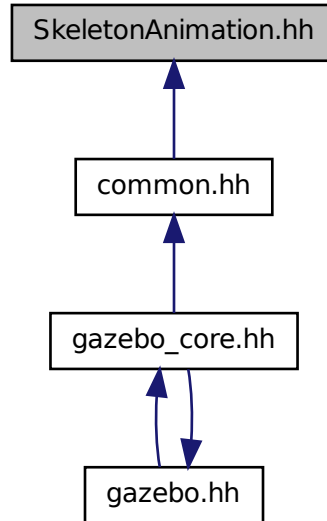
11.184 SkeletonAnimation.hh File Reference

```
#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. 784) animation.*

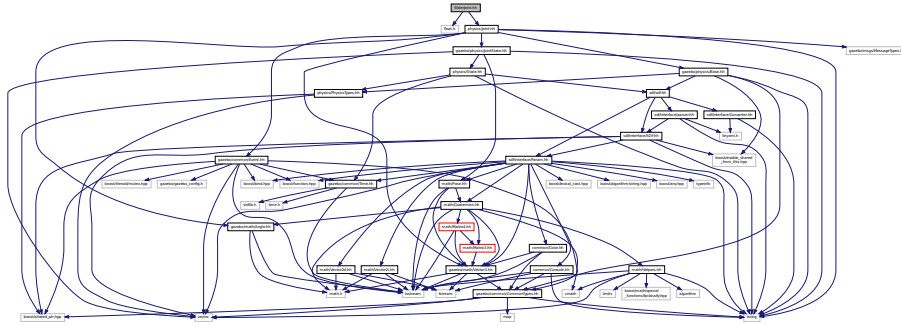
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

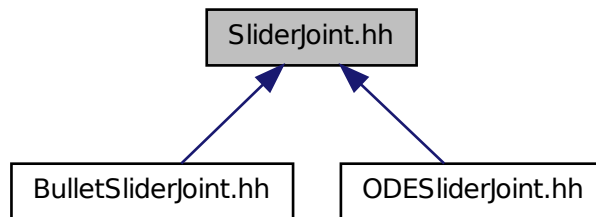
11.185 SliderJoint.hh File Reference

```
#include <float.h>
#include "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.

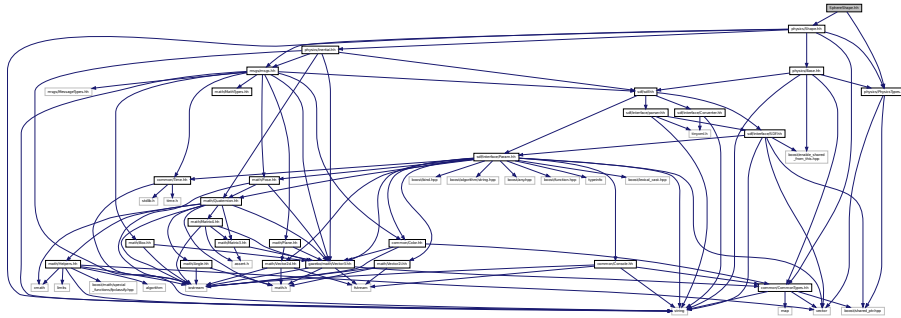
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

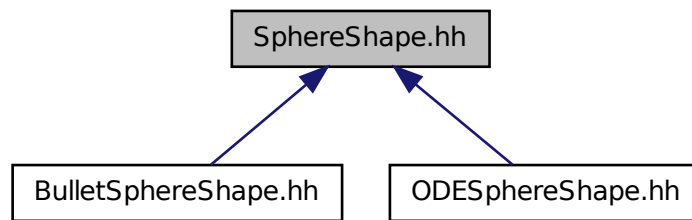
11.186 SphereShape.hh File Reference

```
#include "physics/Shape.hh"
#include "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.

Namespaces

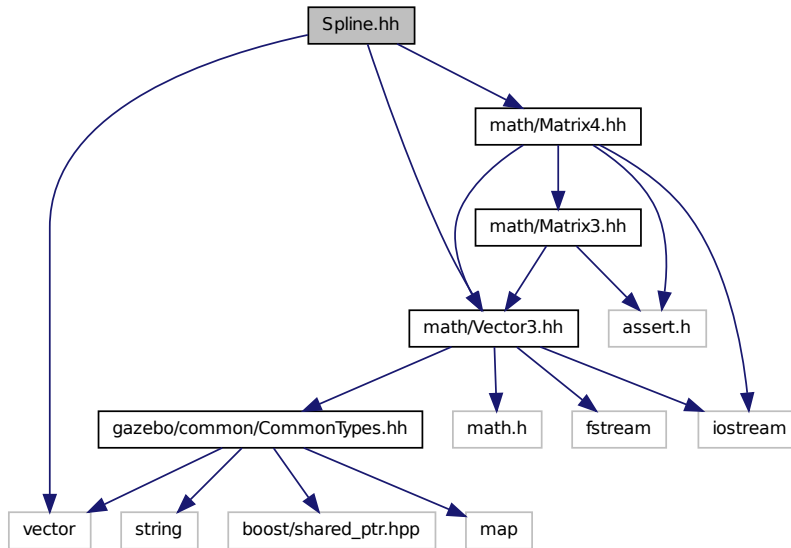
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.187 Spline.hh File Reference

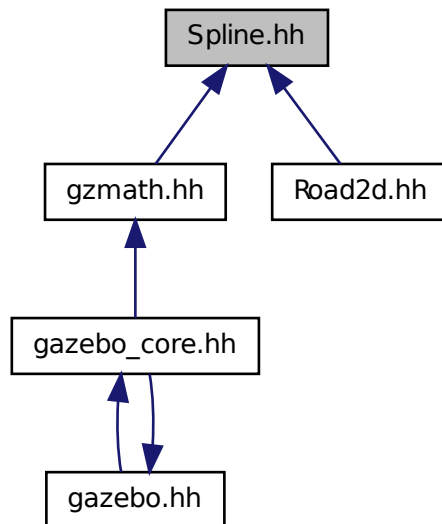
```

#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.

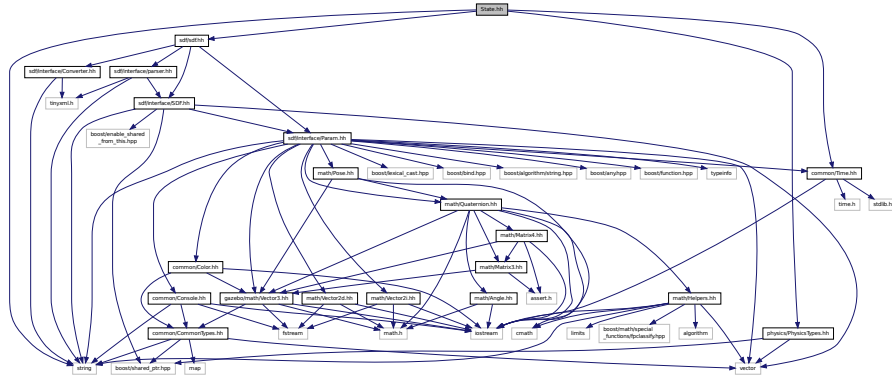
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::math**
Math namespace.

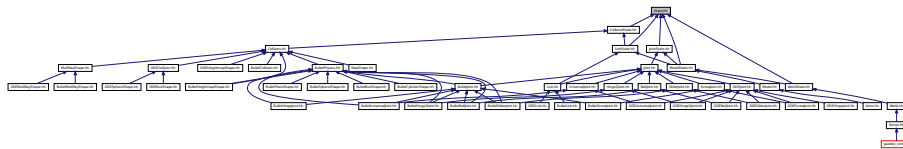
11.188 State.hh File Reference

```
#include <string>
#include "sdf/sdf.hh"
#include "physics/PhysicsTypes.hh"
#include "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. 813) of an entity.

Namespaces

- namespace **gazebo**

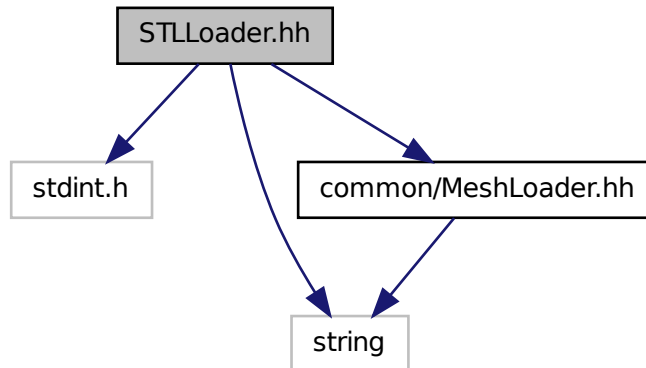
Forward declarations for the common classes.

- namespace **gazebo::physics**

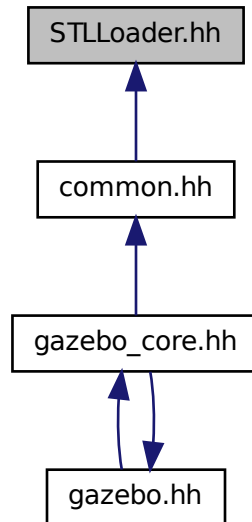
namespace for physics

11.189 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

11.189.1 Macro Definition Documentation

11.189.1.1 #define COR3_MAX 200000

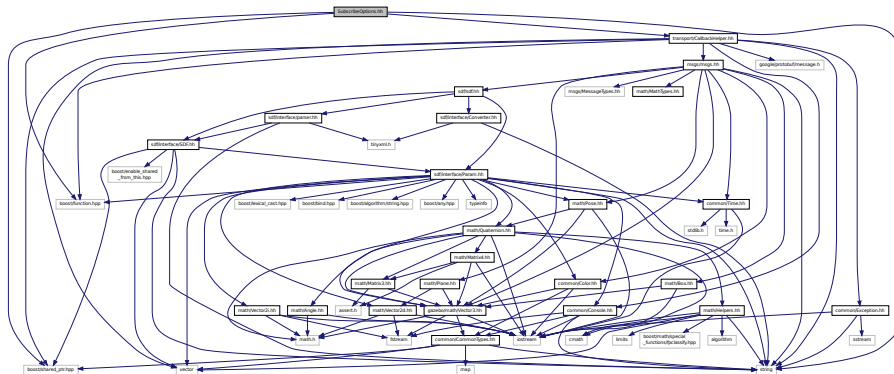
11.189.1.2 `#define FACE_MAX 200000`

11.189.1.3 `#define LINE_MAX_LEN 256`

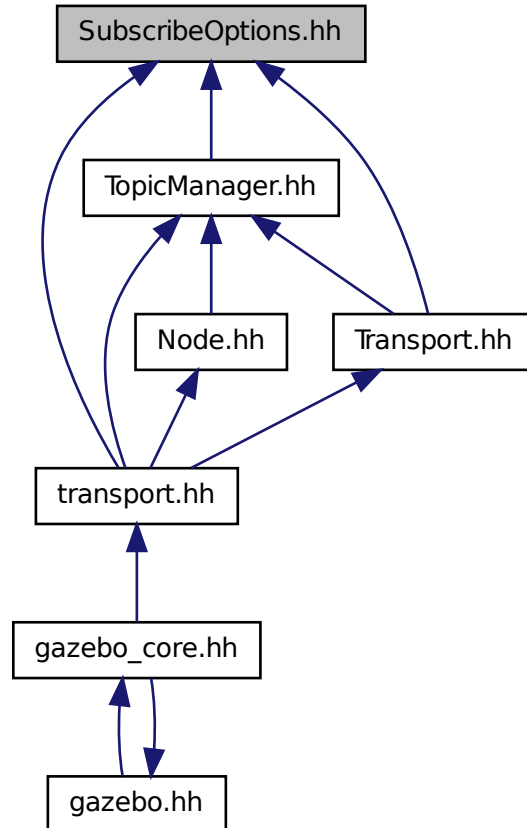
11.189.1.4 `#define ORDER_MAX 10`

11.190 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

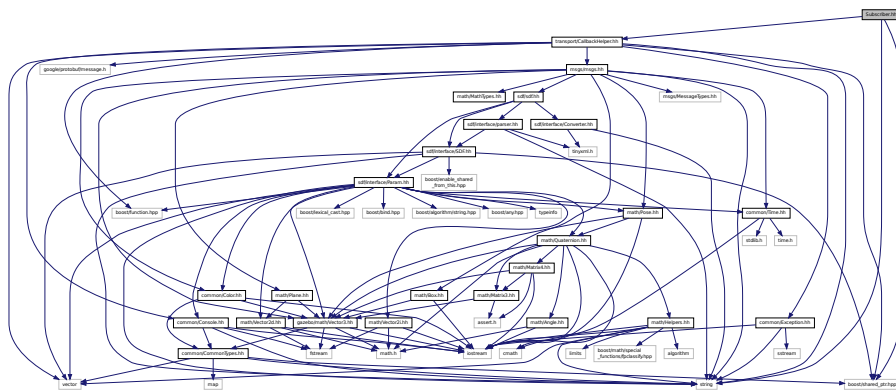
- class **gazebo::transport::SubscribeOptions**
Options for a subscription.

Namespaces

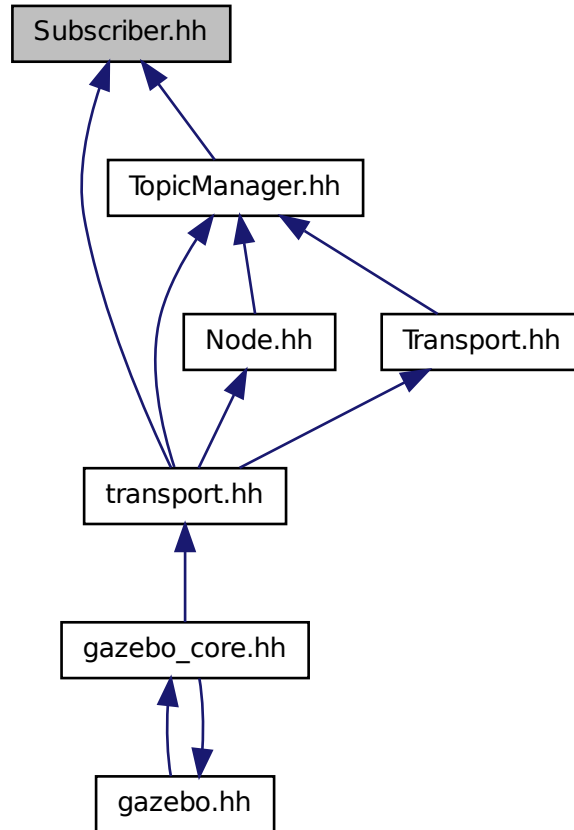
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**
Transport namespace.

11.191 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

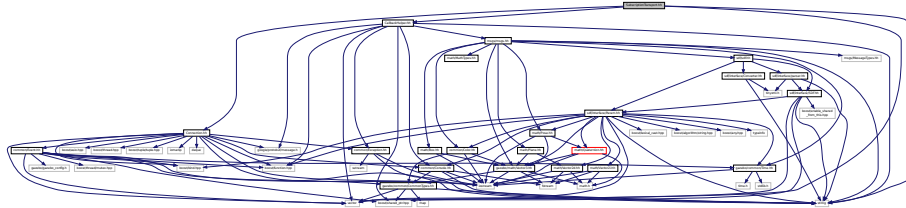
- class **gazebo::transport::Subscriber**
A subscriber to a topic.

Namespaces

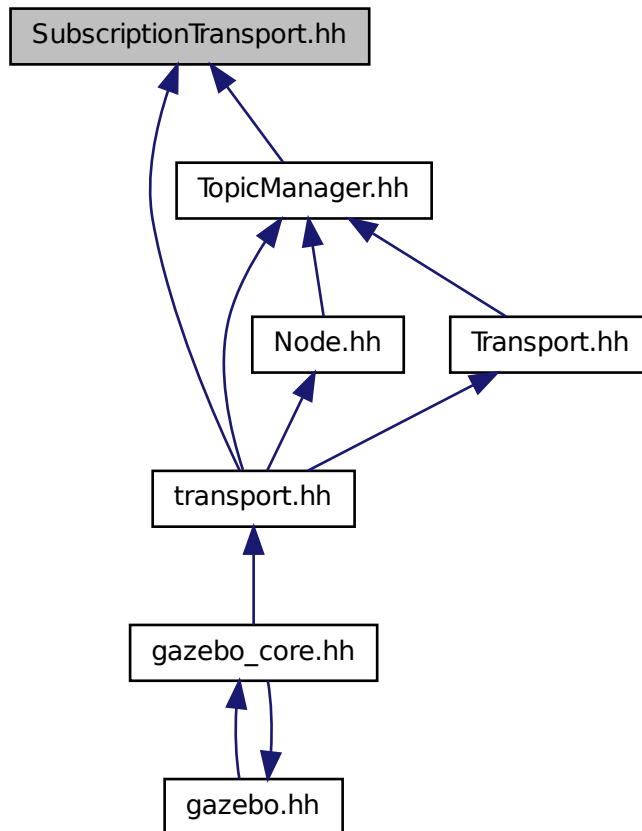
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**
Transport namespace.

11.192 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**
Handles sending data over the wire to remote subscribers.

Namespaces

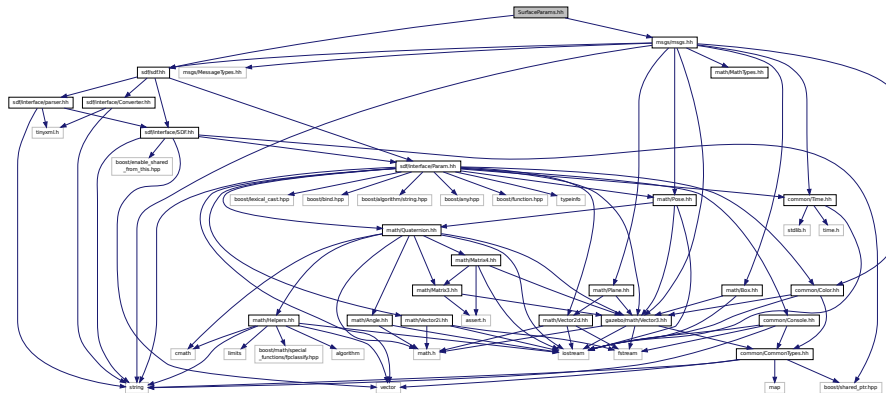
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::transport**
Transport namespace.

11.193 SurfaceParams.hh File Reference

```
#include "msgs/msgs.hh"
```

```
#include "sdf/sdf.hh"
```

Include dependency graph for SurfaceParams.hh:



Classes

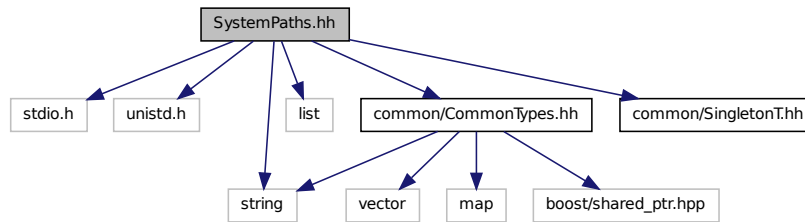
- class **gazebo::physics::SurfaceParams**
SurfaceParams (p. 830) defines various Surface contact parameters.

Namespaces

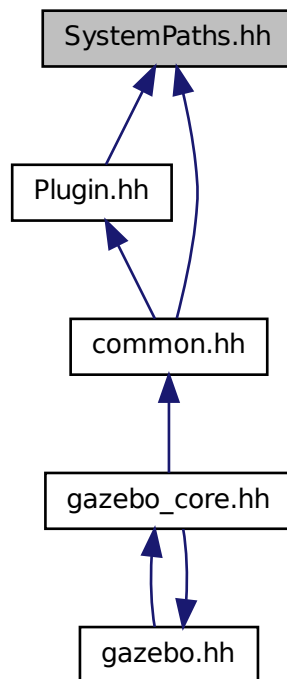
- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.194 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**

11.194.1 Macro Definition Documentation

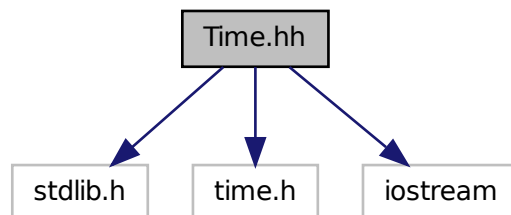
11.194.1.1 #define **GetCurrentDir** getcwd

11.194.1.2 #define **LINUX**

11.195 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. 840) class, can be used to hold wall- or sim-time.

Namespaces

- namespace `gazebo`

Forward declarations for the common classes.

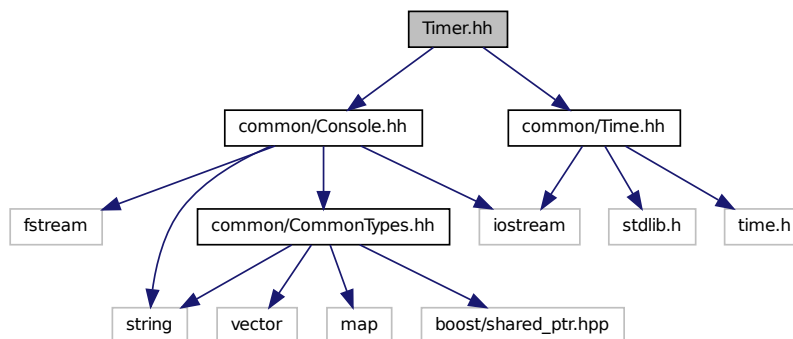
- namespace `gazebo::common`

Common namespace.

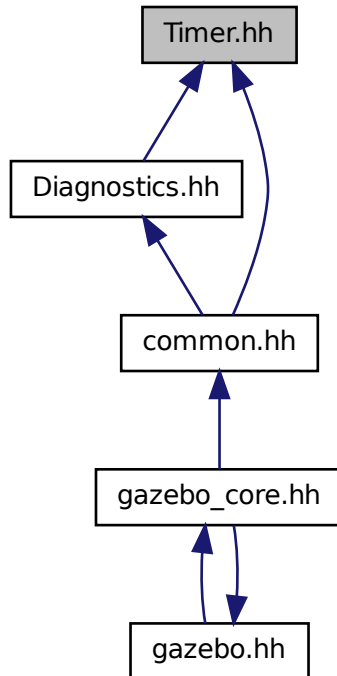
11.196 Timer.hh File Reference

```
#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.

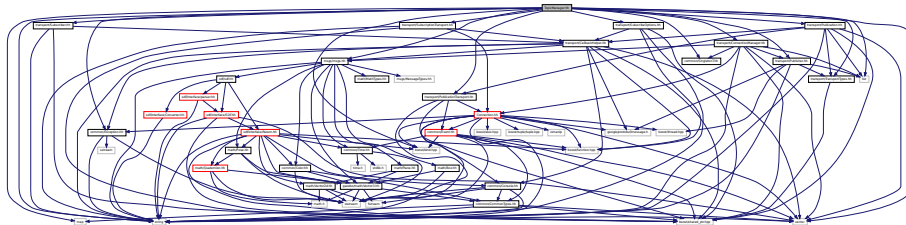
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.

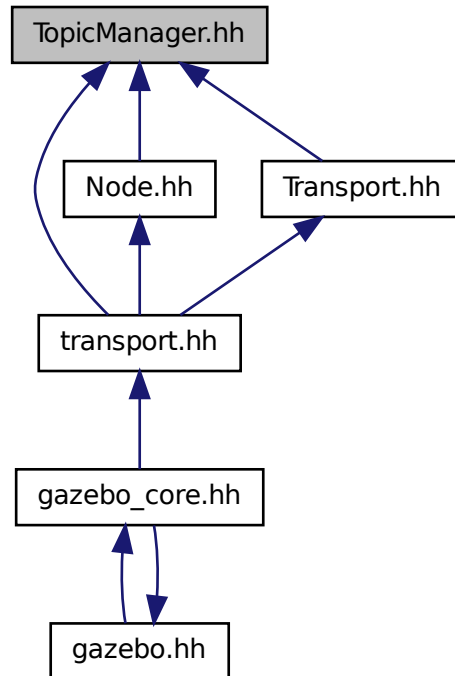
11.197 TopicManager.hh File Reference

```
#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**
Transport namespace.

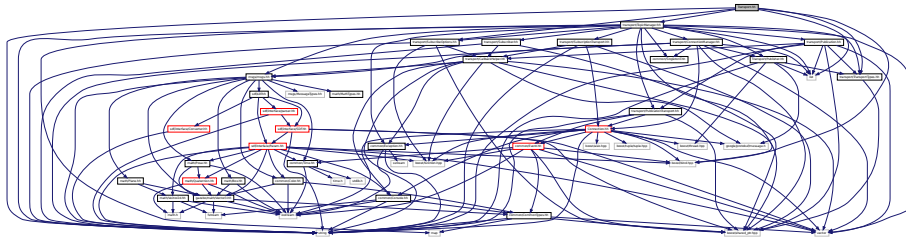
11.198 Transport.hh File Reference

```
#include <boost/bind.hpp>
```

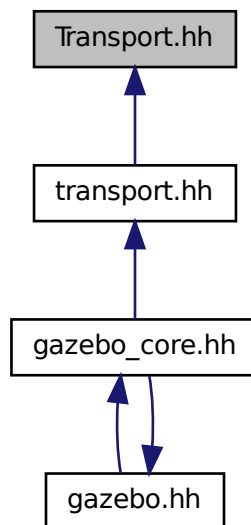


```
#include <string>
#include <list>
#include "transport/TransportTypes.hh"
#include "transport/SubscribeOptions.hh"
#include "transport/TopicManager.hh"
```

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**
Transport namespace.

Functions

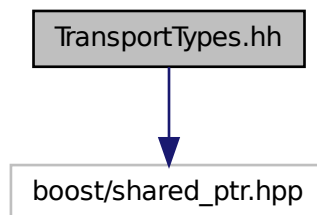
- void **gazebo::transport::clear_buffers** ()
clear any remaining communication buffers
- void **gazebo::transport::fini** ()
Cleanup the transport component.
- bool **gazebo::transport::get_master_uri** (std::string &master_host, unsigned int &master_port)
Get the hostname and port of the master from 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** ()
Return true if the transport system is stopped.
- void **gazebo::transport::pause_incoming** (bool _pause)
Set to true to pause 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.

11.199 TransportTypes.hh File Reference

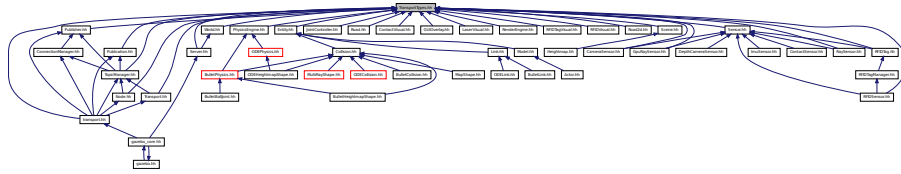
Forward declarations for transport.

```
#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**

Transport namespace.

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**

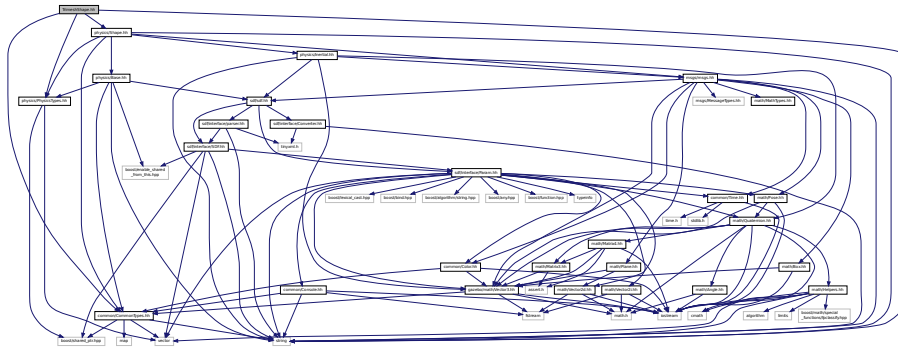
11.199.1 Detailed Description

Forward declarations for transport.

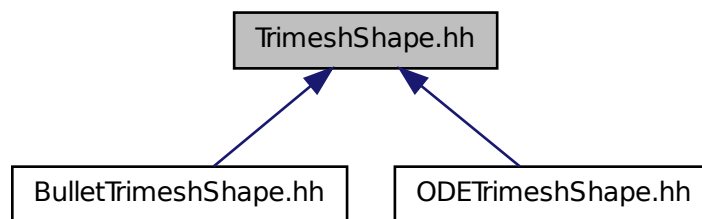
11.200 TrimeshShape.hh File Reference

```
#include <string>
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "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.

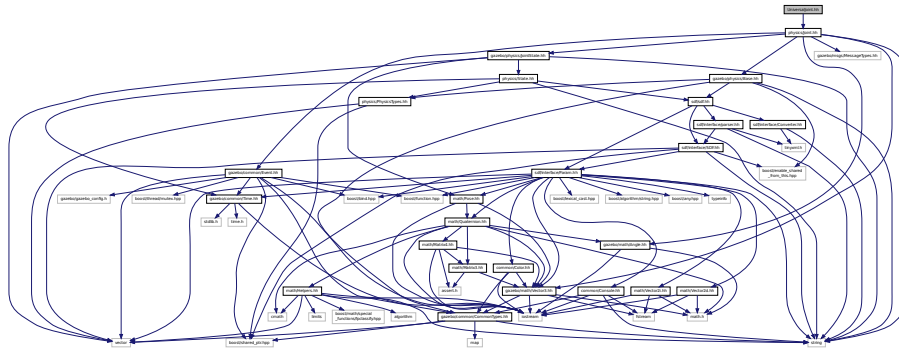
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

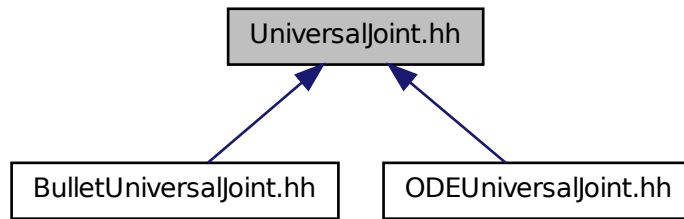
11.201 UniversalJoint.hh File Reference

```
#include "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.

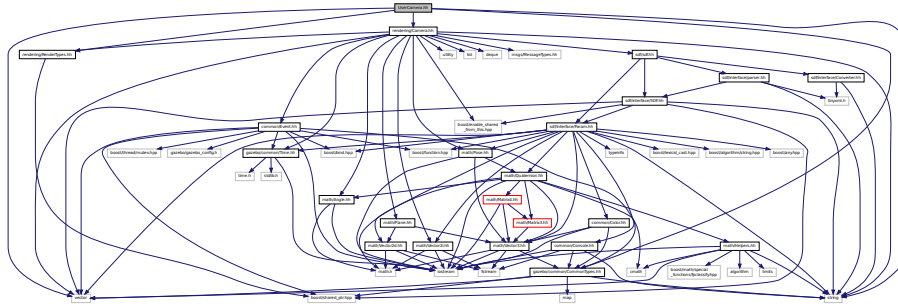
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

11.202 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.

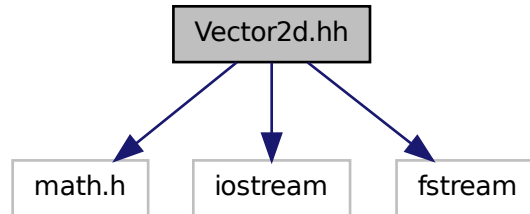
- namespace **gazebo::rendering**

Rendering namespace.

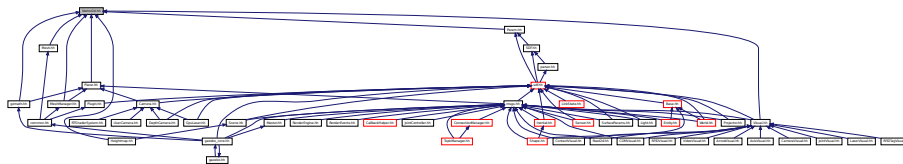
11.203 Vector2d.hh File Reference

```
#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.

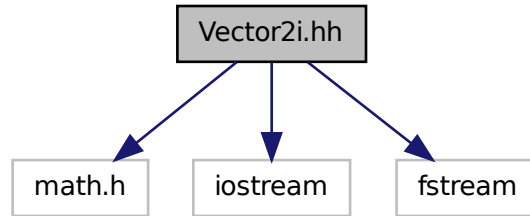
- namespace **gazebo::math**

Math namespace.

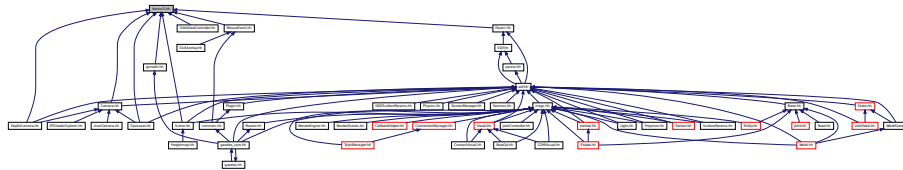
11.204 Vector2i.hh File Reference

```
#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**

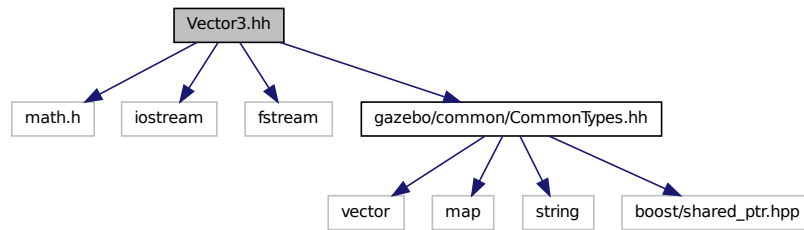
Math namespace.

11.205 Vector3.hh File Reference

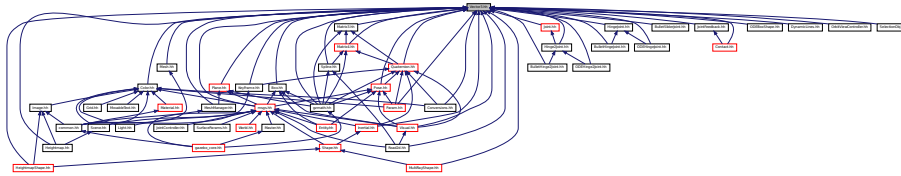
```

#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. 902) class represents the generic vector containing 3 elements.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::math**

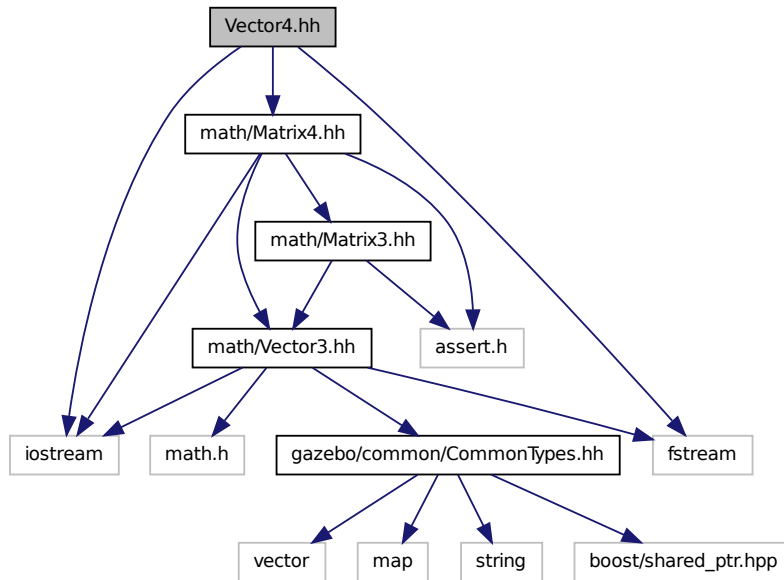
Math namespace.

11.206 Vector4.hh File Reference

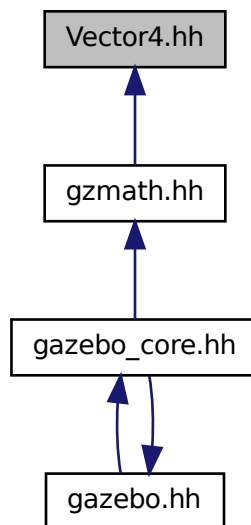
```

#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.

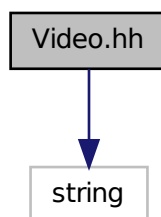
- namespace **gazebo::math**

Math namespace.

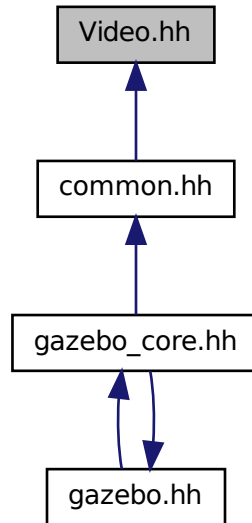
11.207 Video.hh File Reference

```
#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.

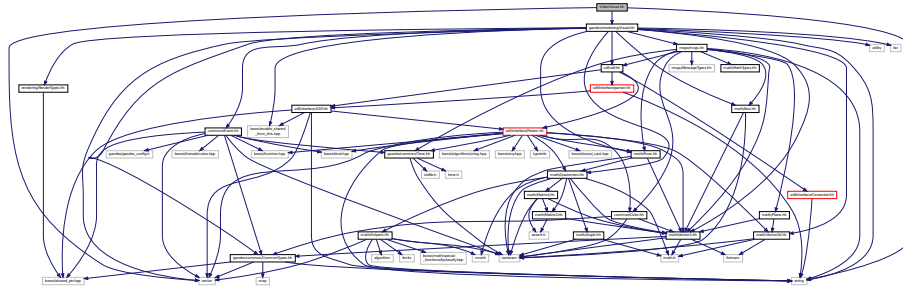
- namespace **gazebo::common**

Common namespace.

11.208 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.

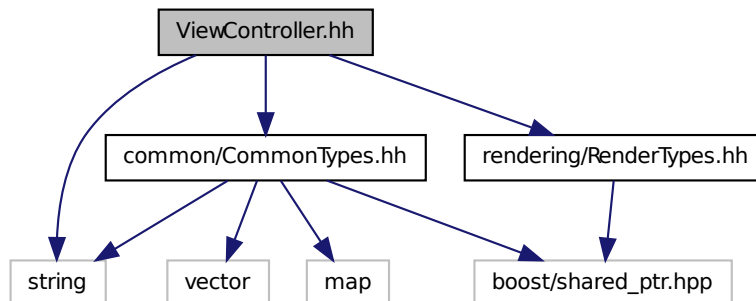
Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::common**
Common namespace.
- namespace **gazebo::rendering**
Rendering namespace.

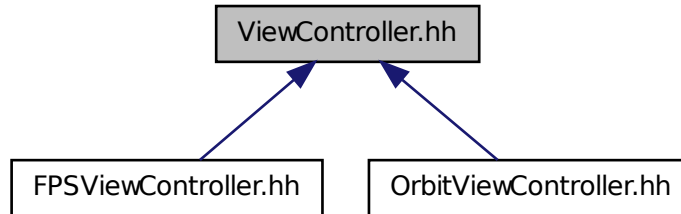
11.209 ViewController.hh File Reference

```
#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.

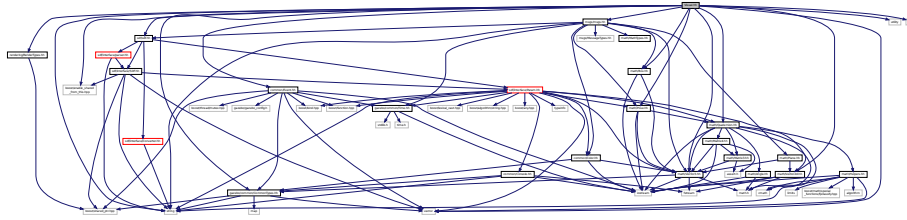
- namespace **gazebo::rendering**

Rendering namespace.

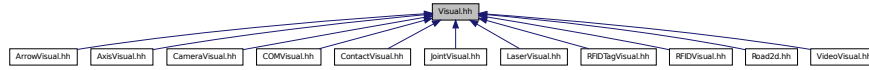
11.210 Visual.hh File Reference

```
#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**

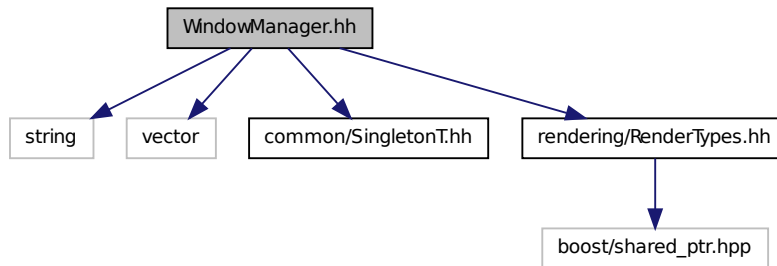
Rendering namespace.

- namespace **Ogre**

11.211 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**

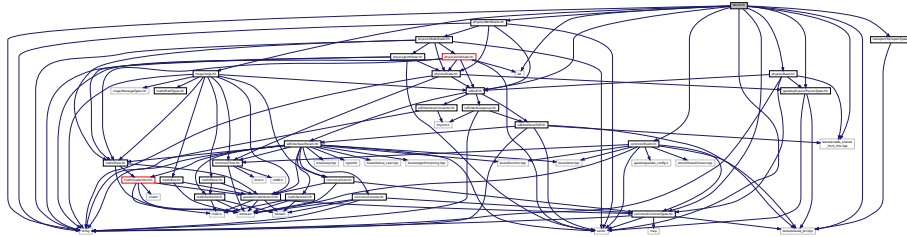
11.212 World.hh File Reference

```

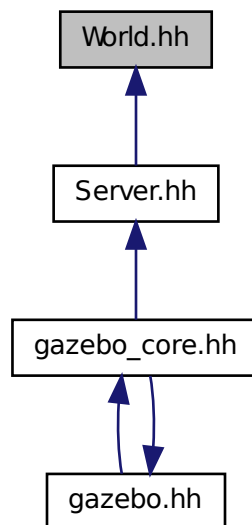
#include <vector>
#include <list>
#include <string>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include "transport/TransportTypes.hh"
#include "msgs/msgs.hh"
#include "common/CommonTypes.hh"
#include "common/Event.hh"
#include "physics/Base.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/WorldState.hh"
#include "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**

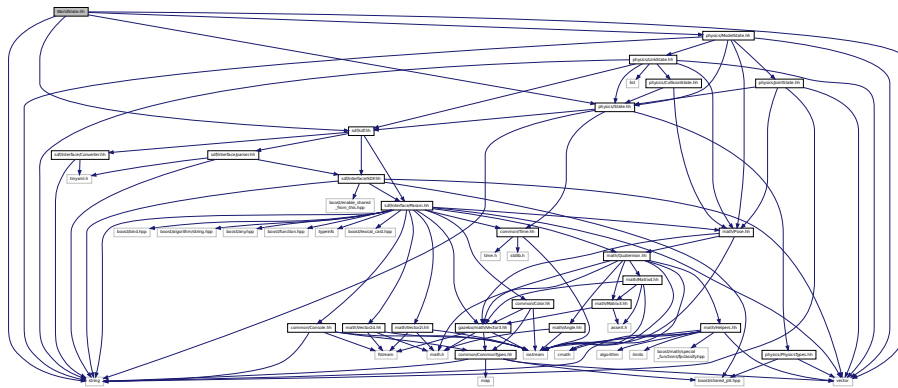
Forward declarations for the common classes.

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

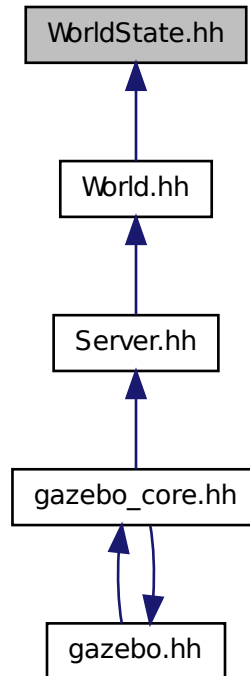
11.213 WorldState.hh File Reference

```
#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. 954) object.*

Namespaces

- namespace **gazebo**
Forward declarations for the common classes.
- namespace **gazebo::physics**
namespace for physics

Index

- ~Actor
 - gazebo::physics::Actor, 124
- ~Angle
 - gazebo::math::Angle, 129
- ~Animation
 - gazebo::common::Animation, 136
- ~ArrowVisual
 - gazebo::rendering::ArrowVisual, 140
- ~AxisVisual
 - gazebo::rendering::AxisVisual, 142
- ~BVHLoader
 - gazebo::common::BVHLoader, 229
- ~BallJoint
 - gazebo::physics::BallJoint, 144
- ~Base
 - gazebo::physics::Base, 150
- ~Box
 - gazebo::math::Box, 159
- ~BoxShape
 - gazebo::physics::BoxShape, 164
- ~BulletBallJoint
 - gazebo::physics::BulletBallJoint, 167
- ~BulletBoxShape
 - gazebo::physics::BulletBoxShape, 171
- ~BulletCollision
 - gazebo::physics::BulletCollision, 173
- ~BulletCylinderShape
 - gazebo::physics::BulletCylinderShape, 176
- ~BulletHeightmapShape
 - gazebo::physics::BulletHeightmapShape, 178
- ~BulletHinge2Joint
 - gazebo::physics::BulletHinge2Joint, 180
- ~BulletHingeJoint
 - gazebo::physics::BulletHingeJoint, 185
- ~BulletJoint
 - gazebo::physics::BulletJoint, 189
- ~BulletLink
 - gazebo::physics::BulletLink, 194
- ~BulletMotionState
 - gazebo::physics::BulletMotionState, 199
- ~BulletMultiRayShape
 - gazebo::physics::BulletMultiRayShape, 202
- ~BulletPhysics
 - gazebo::physics::BulletPhysics, 204
- ~BulletPlaneShape
 - gazebo::physics::BulletPlaneShape, 208
- ~BulletRaySensor
 - gazebo::physics::BulletRaySensor, 209
- ~BulletRayShape
 - gazebo::physics::BulletRayShape, 212
- ~BulletScrewJoint
 - gazebo::physics::BulletScrewJoint, 214
- ~BulletSliderJoint
 - gazebo::physics::BulletSliderJoint, 218
- ~BulletSphereShape
 - gazebo::physics::BulletSphereShape, 222
- ~BulletTrimeshShape
 - gazebo::physics::BulletTrimeshShape, 224
- ~BulletUniversalJoint
 - gazebo::physics::BulletUniversalJoint, 226
- ~COMVisual
 - gazebo::rendering::COMVisual, 286
- ~CallbackHelper
 - gazebo::transport::CallbackHelper, 231
- ~Camera
 - gazebo::rendering::Camera, 238
- ~CameraSensor
 - gazebo::sensors::CameraSensor, 257
- ~CameraVisual
 - gazebo::rendering::CameraVisual, 261
- ~ColladaLoader
 - gazebo::common::ColladaLoader, 262
- ~Collision
 - gazebo::physics::Collision, 265
- ~CollisionState
 - gazebo::physics::CollisionState, 273
- ~Color
 - gazebo::common::Color, 277
- ~Connection
 - gazebo::event::Connection, 287
 - gazebo::transport::Connection, 290
- ~Contact
 - gazebo::physics::Contact, 298
- ~ContactSensor
 - gazebo::sensors::ContactSensor, 301
- ~ContactVisual
 - gazebo::rendering::ContactVisual, 305
- ~CylinderShape
 - gazebo::physics::CylinderShape, 309
- ~DepthCamera

- gazebo::rendering::DepthCamera, 314
- ~DepthCameraSensor
 - gazebo::sensors::DepthCameraSensor, 318
- ~DiagnosticTimer
 - gazebo::common::DiagnosticTimer, 324
- ~DynamicLines
 - gazebo::rendering::DynamicLines, 326
- ~DynamicRenderable
 - gazebo::rendering::DynamicRenderable, 329
- ~Element
 - sdf::Element, 334
- ~Entity
 - gazebo::physics::Entity, 341
- ~Event
 - gazebo::event::Event, 351
- ~EventT
 - Events, 41
- ~Exception
 - gazebo::common::Exception, 366
- ~FPSViewController
 - gazebo::rendering::FPSViewController, 368
- ~GUIOverlay
 - gazebo::rendering::GUIOverlay, 395
- ~GazeboGenerator
 - google::protobuf::compiler::cpp::GazeboGenerator, 374
- ~GpuLaser
 - gazebo::rendering::GpuLaser, 376
- ~GpuRaySensor
 - gazebo::sensors::GpuRaySensor, 381
- ~Grid
 - gazebo::rendering::Grid, 390
- ~Gripper
 - gazebo::physics::Gripper, 393
- ~Heightmap
 - gazebo::rendering::Heightmap, 398
- ~HeightmapShape
 - gazebo::physics::HeightmapShape, 401
- ~Hinge2Joint
 - gazebo::physics::Hinge2Joint, 405
- ~HingeJoint
 - gazebo::physics::HingeJoint, 406
- ~IOManager
 - gazebo::transport::IOManager, 423
- ~Image
 - gazebo::common::Image, 409
- ~ImuSensor
 - gazebo::sensors::ImuSensor, 413
- ~Inertial
 - gazebo::physics::Inertial, 417
- ~Joint
 - gazebo::physics::Joint, 427
- ~JointState
 - gazebo::physics::JointState, 443
- ~JointVisual
 - gazebo::rendering::JointVisual, 445
- ~KeyFrame
 - gazebo::common::KeyFrame, 446
- ~LaserVisual
 - gazebo::rendering::LaserVisual, 448
- ~Light
 - gazebo::rendering::Light, 450
- ~Link
 - gazebo::physics::Link, 459
- ~LinkState
 - gazebo::physics::LinkState, 475
- ~MapShape
 - gazebo::physics::MapShape, 485
- ~Master
 - gazebo::Master, 488
- ~Material
 - gazebo::common::Material, 492
- ~Matrix3
 - gazebo::math::Matrix3, 499
- ~Matrix4
 - gazebo::math::Matrix4, 504
- ~Mesh
 - gazebo::common::Mesh, 510
- ~MeshLoader
 - gazebo::common::MeshLoader, 515
- ~Model
 - gazebo::physics::Model, 524
- ~ModelPlugin
 - gazebo::ModelPlugin, 535
- ~ModelState
 - gazebo::physics::ModelState, 537
- ~MovableText
 - gazebo::rendering::MovableText, 545
- ~MultiRayShape
 - gazebo::physics::MultiRayShape, 551
- ~Node
 - gazebo::transport::Node, 558
- ~NodeAnimation
 - gazebo::common::NodeAnimation, 561
- ~NodeTransform
 - gazebo::common::NodeTransform, 566
- ~NumericAnimation
 - gazebo::common::NumericAnimation, 570
- ~NumericKeyFrame
 - gazebo::common::NumericKeyFrame, 572
- ~ODEBallJoint
 - gazebo::physics::ODEBallJoint, 575
- ~ODEBoxShape
 - gazebo::physics::ODEBoxShape, 577
- ~ODECollision
 - gazebo::physics::ODECollision, 580
- ~ODECylinderShape
 - gazebo::physics::ODECylinderShape, 584

- ~ODEHeightmapShape
 - gazebo::physics::ODEHeightmapShape, 586
- ~ODEHinge2Joint
 - gazebo::physics::ODEHinge2Joint, 588
- ~ODEHingeJoint
 - gazebo::physics::ODEHingeJoint, 592
- ~ODEJoint
 - gazebo::physics::ODEJoint, 596
- ~ODELink
 - gazebo::physics::ODELink, 602
- ~ODEMultiRayShape
 - gazebo::physics::ODEMultiRayShape, 609
- ~ODEPhysics
 - gazebo::physics::ODEPhysics, 612
- ~ODEPlaneShape
 - gazebo::physics::ODEPlaneShape, 619
- ~ODERayShape
 - gazebo::physics::ODERayShape, 621
- ~ODEScrewJoint
 - gazebo::physics::ODEScrewJoint, 623
- ~ODESliderJoint
 - gazebo::physics::ODESliderJoint, 627
- ~ODESphereShape
 - gazebo::physics::ODESphereShape, 631
- ~ODESurfaceParams
 - gazebo::physics::ODESurfaceParams, 631
- ~ODETrimeshShape
 - gazebo::physics::ODETrimeshShape, 634
- ~ODEUniversalJoint
 - gazebo::physics::ODEUniversalJoint, 636
- ~OrbitViewController
 - gazebo::rendering::OrbitViewController, 639
- ~PID
 - gazebo::common::PID, 665
- ~Param
 - sdf::Param, 644
- ~ParamT
 - sdf::ParamT, 649
- ~PhysicsEngine
 - gazebo::physics::PhysicsEngine, 654
- ~Plane
 - gazebo::math::Plane, 670
- ~PlaneShape
 - gazebo::physics::PlaneShape, 672
- ~Pose
 - gazebo::math::Pose, 679
- ~PoseAnimation
 - gazebo::common::PoseAnimation, 686
- ~PoseKeyFrame
 - gazebo::common::PoseKeyFrame, 688
- ~Projector
 - gazebo::rendering::Projector, 690
- ~Publication
 - gazebo::transport::Publication, 692
- ~PublicationTransport
 - gazebo::transport::PublicationTransport, 694
- ~Publisher
 - gazebo::transport::Publisher, 696
- ~Quaternion
 - gazebo::math::Quaternion, 701
- ~RFIDSensor
 - gazebo::sensors::RFIDSensor, 728
- ~RFIDTag
 - gazebo::sensors::RFIDTag, 730
- ~RFIDTagVisual
 - gazebo::rendering::RFIDTagVisual, 734
- ~RFIDVisual
 - gazebo::rendering::RFIDVisual, 735
- ~RaySensor
 - gazebo::sensors::RaySensor, 713
- ~RayShape
 - gazebo::physics::RayShape, 720
- ~Road
 - gazebo::physics::Road, 737
- ~Road2d
 - gazebo::rendering::Road2d, 738
- ~RotationSpline
 - gazebo::math::RotationSpline, 739
- ~STLLoader
 - gazebo::common::STLLoader, 816
- ~Scene
 - gazebo::rendering::Scene, 749
- ~ScrewJoint
 - gazebo::physics::ScrewJoint, 761
- ~SelectionObj
 - gazebo::rendering::SelectionObj, 764
- ~Sensor
 - gazebo::sensors::Sensor, 767
- ~SensorPlugin
 - gazebo::SensorPlugin, 777
- ~Server
 - gazebo::Server, 779
- ~Shape
 - gazebo::physics::Shape, 781
- ~SingletonT
 - SingletonT, 784
- ~Skeleton
 - gazebo::common::Skeleton, 786
- ~SkeletonAnimation
 - gazebo::common::SkeletonAnimation, 792
- ~SkeletonNode
 - gazebo::common::SkeletonNode, 798
- ~SliderJoint
 - gazebo::physics::SliderJoint, 805
- ~SphereShape
 - gazebo::physics::SphereShape, 807
- ~Spline
 - gazebo::math::Spline, 809

- ~State
 - gazebo::physics::State, 814
- ~SubMesh
 - gazebo::common::SubMesh, 819
- ~Subscriber
 - gazebo::transport::Subscriber, 828
- ~SubscriptionTransport
 - gazebo::transport::SubscriptionTransport, 829
- ~SurfaceParams
 - gazebo::physics::SurfaceParams, 831
- ~SystemPlugin
 - gazebo::SystemPlugin, 839
- ~Time
 - gazebo::common::Time, 844
- ~Timer
 - gazebo::common::Timer, 860
- ~TrimeshShape
 - gazebo::physics::TrimeshShape, 867
- ~URDF2Gazebo
 - urdf2gazebo::URDF2Gazebo, 873
- ~UniversalJoint
 - gazebo::physics::UniversalJoint, 869
- ~UserCamera
 - gazebo::rendering::UserCamera, 880
- ~Vector2d
 - gazebo::math::Vector2d, 888
- ~Vector2i
 - gazebo::math::Vector2i, 896
- ~Vector3
 - gazebo::math::Vector3, 905
- ~Vector4
 - gazebo::math::Vector4, 917
- ~Video
 - gazebo::common::Video, 925
- ~VideoVisual
 - gazebo::rendering::VideoVisual, 927
- ~ViewController
 - gazebo::rendering::ViewController, 928
- ~Visual
 - gazebo::rendering::Visual, 936
- ~World
 - gazebo::physics::World, 957
- ~WorldPlugin
 - gazebo::WorldPlugin, 966
- ~WorldState
 - gazebo::physics::WorldState, 968
- _setupGeometry
 - gazebo::rendering::MovableText, 545
- _updateColors
 - gazebo::rendering::MovableText, 545
- a
 - gazebo::common::Color, 284
- ABGR
 - gazebo::common::Color, 276
- ACTOR
 - gazebo::physics::Base, 149
- ADD
 - gazebo::common::Material, 491
- ARGB
 - gazebo::common::Color, 276
- AcceptCallback
 - gazebo::transport::Connection, 290
- active
 - gazebo::physics::Actor, 125
 - gazebo::sensors::Sensor, 771
- Actor
 - gazebo::physics::Actor, 123
- Actor.hh, 971
- Actor_V
 - gazebo::physics, 108
- ActorPtr
 - gazebo::physics, 108
- Add
 - gazebo::physics::Logger, 478
- add_plugin
 - gazebo, 95
- add_search_path_suffix
 - Common, 36
- AddAnimation
 - gazebo::common::Skeleton, 786
- AddAttribute
 - sdf::Element, 334
- AddCallback
 - gazebo::transport::PublicationTransport, 694
- AddChild
 - gazebo::common::SkeletonNode, 798
 - gazebo::physics::Base, 150
- AddChildJoint
 - gazebo::physics::Link, 459
- AddContact
 - gazebo::physics::Collision, 265
- AddElement
 - sdf::Element, 334
- AddElementDescription
 - sdf::Element, 334
- addEntity
 - Events, 50
- AddForce
 - gazebo::physics::BulletLink, 194
 - gazebo::physics::Link, 459
 - gazebo::physics::ODELink, 602
- AddForceAtRelativePosition
 - gazebo::physics::BulletLink, 194
 - gazebo::physics::Link, 459
 - gazebo::physics::ODELink, 602
- AddForceAtWorldPosition
 - gazebo::physics::BulletLink, 194

- gazebo::physics::Link, 460
- gazebo::physics::ODELink, 603
- AddGazeboPaths
 - gazebo::common::SystemPaths, 835
- AddIndex
 - gazebo::common::SubMesh, 820
- AddJoint
 - gazebo::physics::JointController, 439
- AddKeyFrame
 - gazebo::common::NodeAnimation, 561
 - gazebo::common::SkeletonAnimation, 792
- addKeyValue
 - urdf2gazebo::URDF2Gazebo, 873
- AddMaterial
 - gazebo::common::Mesh, 510
- AddMesh
 - gazebo::common::MeshManager, 517
- addNestedModel
 - sdf, 119
- AddNode
 - gazebo::transport::TopicManager, 863
- AddNodeAssignment
 - gazebo::common::SubMesh, 820
- AddNormal
 - gazebo::common::SubMesh, 820
- AddOgrePaths
 - gazebo::common::SystemPaths, 836
- AddParentJoint
 - gazebo::physics::Link, 460
- AddPluginPaths
 - gazebo::common::SystemPaths, 836
- AddPoint
 - gazebo::math::RotationSpline, 739
 - gazebo::math::Spline, 810
 - gazebo::rendering::DynamicLines, 326
- AddPublisher
 - gazebo::transport::Publication, 693
- AddRawTransform
 - gazebo::common::SkeletonNode, 798
- AddRay
 - gazebo::physics::BulletMultiRayShape, 202
 - gazebo::physics::BulletRaySensor, 210
 - gazebo::physics::MultiRayShape, 551
 - gazebo::physics::ODEMultiRayShape, 609
- AddRelativeForce
 - gazebo::physics::BulletLink, 194
 - gazebo::physics::Link, 460
 - gazebo::physics::ODELink, 603
- AddRelativeTorque
 - gazebo::physics::BulletLink, 195
 - gazebo::physics::Link, 460
 - gazebo::physics::ODELink, 603
- AddResourcePath
 - gazebo::rendering::RenderEngine, 724
- AddScene
 - gazebo::rendering::RTShaderSystem, 744
- AddSearchPathSuffix
 - gazebo::common::SystemPaths, 836
- AddSubMesh
 - gazebo::common::Mesh, 510
- AddSubscription
 - gazebo::transport::Publication, 693
- AddTaggedModel
 - gazebo::sensors::RFIDTagManager, 732
- AddTexCoord
 - gazebo::common::SubMesh, 820
- AddTime
 - gazebo::common::Animation, 137
- AddTorque
 - gazebo::physics::BulletLink, 195
 - gazebo::physics::Link, 460
 - gazebo::physics::ODELink, 603
- addTransform
 - urdf2gazebo::URDF2Gazebo, 873
- AddTransport
 - gazebo::transport::Publication, 693
- AddType
 - gazebo::physics::Base, 150
- AddValue
 - sdf::Element, 334
- AddVertNodeWeight
 - gazebo::common::Skeleton, 786
- AddVertex
 - gazebo::common::SubMesh, 821
- AddVisual
 - gazebo::rendering::Scene, 749
- Advertise
 - gazebo::transport::ConnectionManager, 293
 - gazebo::transport::Node, 558
 - gazebo::transport::TopicManager, 863
- alt
 - gazebo::common::MouseEvent, 542
- ambient
 - gazebo::common::Material, 497
- anchorLink
 - gazebo::physics::Joint, 438
- anchorPos
 - gazebo::physics::Joint, 438
- Angle
 - gazebo::math::Angle, 129
- Angle.hh, 972
 - GZ_DTOR, 973
 - GZ_NORMALIZE, 973
 - GZ_RTOD, 973
- angularAccel
 - gazebo::physics::Link, 473
- animState
 - gazebo::rendering::Camera, 254

- Animation
 - gazebo::common::Animation, 136
- animation
 - gazebo::physics::Entity, 348
- Animation.hh, 974
- animationConnection
 - gazebo::physics::Entity, 348
- AnimationPtr
 - gazebo::common, 98
- animationStartPose
 - gazebo::physics::Entity, 349
- animations
 - gazebo::common::SkeletonAnimation, 794
- anims
 - gazebo::common::Skeleton, 790
- ApplyDamping
 - gazebo::physics::ODEHingeJoint, 593
 - gazebo::physics::ODEScrewJoint, 624
 - gazebo::physics::ODESliderJoint, 627
- ApplyShadows
 - gazebo::rendering::RTShaderSystem, 744
- AreConnected
 - gazebo::physics::BulletJoint, 189
 - gazebo::physics::Joint, 427
 - gazebo::physics::ODEJoint, 596
- ArrowVisual
 - gazebo::rendering::ArrowVisual, 140
- ArrowVisual.hh, 975
- ArrowVisualPtr
 - gazebo::rendering, 112
- AsyncRead
 - gazebo::transport::Connection, 290
- Attach
 - gazebo::physics::BulletBallJoint, 168
 - gazebo::physics::BulletHinge2Joint, 181
 - gazebo::physics::BulletHingeJoint, 186
 - gazebo::physics::BulletScrewJoint, 215
 - gazebo::physics::BulletSliderJoint, 219
 - gazebo::physics::BulletUniversalJoint, 227
 - gazebo::physics::Joint, 427
 - gazebo::physics::ODEJoint, 596
 - gazebo::rendering::SelectionObj, 764
- AttachAxes
 - gazebo::rendering::Visual, 936
- AttachCameraToImage
 - gazebo::rendering::GUIOverlay, 395
- AttachEntity
 - gazebo::rendering::RTShaderSystem, 744
- AttachLineVertex
 - gazebo::rendering::Visual, 936
- AttachMesh
 - gazebo::rendering::Visual, 936
- AttachObject
 - gazebo::rendering::Visual, 936
- AttachStaticModel
 - gazebo::physics::Link, 461
 - gazebo::physics::Model, 524
- AttachToVisual
 - gazebo::rendering::Camera, 238
- AttachToVisualImpl
 - gazebo::rendering::Camera, 239
 - gazebo::rendering::UserCamera, 880
- AttachViewport
 - gazebo::rendering::RTShaderSystem, 744
- AttachVisual
 - gazebo::rendering::Visual, 936
- attachedModels
 - gazebo::physics::Model, 533
- attachedModelsOffset
 - gazebo::physics::Link, 473
 - gazebo::physics::Model, 533
- Attribute
 - gazebo::physics::Joint, 426
- autoCalc
 - gazebo::math::RotationSpline, 742
 - gazebo::math::Spline, 812
- autoStart
 - gazebo::physics::Actor, 125
- AxisVisual
 - gazebo::rendering::AxisVisual, 142
- AxisVisual.hh, 976
- AxisVisualPtr
 - gazebo::rendering, 112
- b
 - gazebo::common::Color, 284
- BALL_JOINT
 - gazebo::physics::Base, 149
- BASE
 - gazebo::physics::Base, 149
- BAYER_GBRG8
 - gazebo::common::Image, 409
- BAYER_GRBG8
 - gazebo::common::Image, 409
- BAYER_RRGB8
 - gazebo::common::Image, 409
- BAYER_RGGR8
 - gazebo::common::Image, 409
- BGR_INT16
 - gazebo::common::Image, 408
- BGR_INT32
 - gazebo::common::Image, 408
- BGR_INT8
 - gazebo::common::Image, 408
- BGRA
 - gazebo::common::Color, 276
- BGRA_INT8
 - gazebo::common::Image, 408

- BLEND_COUNT
 - gazebo::common::Material, 491
- BLINN
 - gazebo::common::Material, 491
- BOX_SHAPE
 - gazebo::physics::Base, 149
- BVHLoader
 - gazebo::common::BVHLoader, 229
- BVHLoader.hh, 996
 - X_POSITION, 997
 - X_ROTATION, 997
 - Y_POSITION, 997
 - Y_ROTATION, 997
 - Z_POSITION, 997
 - Z_ROTATION, 997
- BallJoint
 - gazebo::physics::BallJoint, 144
- BallJoint.hh, 976
- Base
 - gazebo::physics::Base, 150
- Base.hh, 977
- Base_V
 - gazebo::physics, 108
- BasePtr
 - gazebo::physics, 108
- bayerFrameBuffer
 - gazebo::rendering::Camera, 254
- Begin
 - gazebo::physics::Logplay, 480
- bindShapeTransform
 - gazebo::common::Skeleton, 790
- Black
 - gazebo::common::Color, 284
- BlendMode
 - gazebo::common::Material, 491
- blendMode
 - gazebo::common::Material, 497
- BlendModeStr
 - gazebo::common::Material, 497
- blobs
 - urdf2gazebo::GazeboExtension, 370
- Blue
 - gazebo::common::Color, 284
- body1Force
 - gazebo::physics::JointFeedback, 441
- body1Torque
 - gazebo::physics::JointFeedback, 441
- body2Force
 - gazebo::physics::JointFeedback, 441
- body2Torque
 - gazebo::physics::JointFeedback, 441
- bonePosePub
 - gazebo::physics::Actor, 125
- boost, 93
- bounce
 - gazebo::physics::SurfaceParams, 831
- bounceThreshold
 - gazebo::physics::SurfaceParams, 831
- Box
 - gazebo::math::Box, 159
- Box.hh, 978
- BoxShape
 - gazebo::physics::BoxShape, 164
- BoxShape.hh, 979
- BoxShapePtr
 - gazebo::physics, 108
- build
 - gazebo::common::Animation, 138
- BuildInterpolationSplines
 - gazebo::common::PoseAnimation, 686
- BuildNodeMap
 - gazebo::common::Skeleton, 787
- Bullet Physics, 75
- bullet_inc.h, 980
- BulletBallJoint
 - gazebo::physics::BulletBallJoint, 167
- BulletBallJoint.hh, 981
- BulletBoxShape
 - gazebo::physics::BulletBoxShape, 170
- BulletBoxShape.hh, 982
- BulletCollision
 - gazebo::physics::BulletCollision, 173
- BulletCollision.hh, 982
- BulletCollisionPtr
 - gazebo::physics, 108
- BulletCylinderShape
 - gazebo::physics::BulletCylinderShape, 176
- BulletCylinderShape.hh, 983
- BulletHeightmapShape
 - gazebo::physics::BulletHeightmapShape, 178
- BulletHeightmapShape.hh, 984
- BulletHinge2Joint
 - gazebo::physics::BulletHinge2Joint, 180
- BulletHinge2Joint.hh, 984
- BulletHingeJoint
 - gazebo::physics::BulletHingeJoint, 185
- BulletHingeJoint.hh, 985
- BulletJoint
 - gazebo::physics::BulletJoint, 189
- BulletJoint.hh, 985
- BulletLink
 - gazebo::physics::BulletLink, 194
- BulletLink.hh, 986
- BulletLinkPtr
 - gazebo::physics, 108
- BulletMotionState
 - gazebo::physics::BulletMotionState, 199
- BulletMotionState.hh, 987

- BulletMultiRayShape
 - gazebo::physics::BulletMultiRayShape, 202
- BulletMultiRayShape.hh, 988
- BulletPhysics
 - gazebo::physics::BulletPhysics, 204
- BulletPhysics.hh, 988
- BulletPhysicsPtr
 - gazebo::physics, 108
- BulletPlaneShape
 - gazebo::physics::BulletPlaneShape, 208
- BulletPlaneShape.hh, 989
- BulletRaySensor
 - gazebo::physics::BulletRaySensor, 209
- BulletRaySensor.hh, 990
- BulletRayShape
 - gazebo::physics::BulletRayShape, 212
- BulletRayShape.hh, 990
- BulletRayShapePtr
 - gazebo::physics, 108
- BulletScrewJoint
 - gazebo::physics::BulletScrewJoint, 214
- BulletScrewJoint.hh, 991
- BulletSliderJoint
 - gazebo::physics::BulletSliderJoint, 218
- BulletSliderJoint.hh, 992
- BulletSphereShape
 - gazebo::physics::BulletSphereShape, 222
- BulletSphereShape.hh, 992
- BulletTrimeshShape
 - gazebo::physics::BulletTrimeshShape, 224
- BulletTrimeshShape.hh, 993
- BulletTypes.hh, 994
- BulletUniversalJoint
 - gazebo::physics::BulletUniversalJoint, 226
- BulletUniversalJoint.hh, 995
- button
 - gazebo::common::MouseEvent, 542
- ButtonCallback
 - gazebo::rendering::GUIOverlay, 395
- Buttons
 - gazebo::common::MouseEvent, 541
- buttons
 - gazebo::common::MouseEvent, 542
- CFM
 - gazebo::physics::Joint, 426
- COLLISION
 - gazebo::physics::Base, 149
- COMVisual
 - gazebo::rendering::COMVisual, 286
- COMVisual.hh, 1010
- COMVisualPtr
 - gazebo::rendering, 112
- COR3_MAX
 - STLLoader.hh, 1160
- CYLINDER_SHAPE
 - gazebo::physics::Base, 149
- CallbackHelper
 - gazebo::transport::CallbackHelper, 231
- CallbackHelper.hh, 997
- CallbackHelperPtr
 - Transport, 89
- CallbackHelperT
 - gazebo::transport::CallbackHelperT, 232
- Camera
 - gazebo::rendering::Camera, 238
- camera
 - gazebo::rendering::Camera, 254
 - gazebo::rendering::ViewController, 930
- Camera.hh, 999
- cameraCount
 - gazebo::sensors::GpuRaySensor, 388
- cameraElem
 - gazebo::sensors::GpuRaySensor, 388
- CameraPtr
 - gazebo::rendering, 112
- CameraSensor
 - gazebo::sensors::CameraSensor, 257
- CameraSensor.hh, 1000
- CameraSensor_V
 - gazebo::sensors, 115
- CameraSensorPtr
 - gazebo::sensors, 115
- CameraVisual
 - gazebo::rendering::CameraVisual, 260
- CameraVisual.hh, 1001
- CameraVisualPtr
 - gazebo::rendering, 112
- Cancel
 - gazebo::transport::Connection, 290
- captureData
 - gazebo::rendering::Camera, 254
- cegui.h, 1001
- cfm
 - gazebo::physics::SurfaceParams, 832
- cgVisuals
 - gazebo::physics::Link, 473
- chfov
 - gazebo::sensors::GpuRaySensor, 388
- childLink
 - gazebo::physics::Joint, 438
- children
 - gazebo::common::SkeletonNode, 803
 - gazebo::physics::Base, 157
- childrenEnd
 - gazebo::physics::Base, 157
- clamp
 - Math, 55

- Classes for physics and dynamics, 68
 - create_world, 72
 - EntityTypename, 74
 - fini, 72
 - GZ_REGISTER_PHYSICS_ENGINE, 71
 - get_world, 72
 - init_world, 72
 - init_worlds, 72
 - load, 73
 - load_world, 73
 - load_worlds, 73
 - pause_world, 73
 - pause_worlds, 73
 - PhysicsFactoryFn, 72
 - remove_worlds, 73
 - run_world, 73
 - run_worlds, 74
 - stop_world, 74
 - stop_worlds, 74
- Clear
 - gazebo::math::RotationSpline, 740
 - gazebo::math::Spline, 810
 - gazebo::physics::World, 957
 - gazebo::rendering::DynamicLines, 326
 - gazebo::rendering::RTShaderSystem, 745
 - gazebo::rendering::Scene, 750
 - gazebo::rendering::SelectionObj, 764
 - sdf::Plugin, 674
- clear_buffers
 - Transport, 89
- ClearBuffers
 - gazebo::transport::TopicManager, 863
- ClearElements
 - sdf::Element, 334
- ClearGazeboPaths
 - gazebo::common::SystemPaths, 836
- ClearOgrePaths
 - gazebo::common::SystemPaths, 836
- ClearParent
 - gazebo::rendering::Visual, 937
- ClearPluginPaths
 - gazebo::common::SystemPaths, 836
- Clone
 - gazebo::physics::Contact, 298
 - gazebo::rendering::Visual, 937
 - sdf::Element, 334
 - sdf::Param, 644
 - sdf::ParamT, 650
- CloneVisual
 - gazebo::rendering::Scene, 750
- coeffs
 - gazebo::math::Spline, 812
- ColladaLoader
 - gazebo::common::ColladaLoader, 262
- ColladaLoader.hh, 1002
- Collide
 - gazebo::physics::ODEPhysics, 612
- Collision
 - gazebo::physics::Collision, 265
- Collision.hh, 1004
- collision1
 - gazebo::physics::Contact, 298
- collision2
 - gazebo::physics::Contact, 298
- Collision_V
 - gazebo::physics, 108
- collisionId
 - gazebo::physics::ODECollision, 582
- collisionParent
 - gazebo::physics::Shape, 782
- CollisionPtr
 - gazebo::physics, 108
 - Gazebo_parser, 82
- collisionShape
 - gazebo::physics::BulletCollision, 174
- CollisionState
 - gazebo::physics::CollisionState, 273
- CollisionState.hh, 1005
- Color
 - gazebo::common::Color, 276, 277
- Color.hh, 1005
- ColorErr
 - gazebo::common::Console, 295
- ColorMsg
 - gazebo::common::Console, 296
- Common, 31
 - add_search_path_suffix, 36
 - DIAG_TIMER, 34
 - DiagnosticTimerPtr, 36
 - DownloadDependencies, 36
 - find_file, 36
 - find_file_path, 36
 - GetManifest, 37
 - GetModelFile, 37
 - GetModelName, 37
 - GetModelPath, 37
 - GetModels, 38
 - GetURI, 38
 - gzclr_end, 34
 - gzclr_start, 34
 - gzdbg, 34
 - gzerr, 34
 - gzlog, 35
 - gzmsg, 35
 - gzthrow, 35
 - gzwarn, 35
 - HasModel, 38
 - MODEL_PLUGIN, 36

- PluginType, 36
- SENSOR_PLUGIN, 36
- SYSTEM_PLUGIN, 36
- VISUAL_PLUGIN, 36
- WORLD_PLUGIN, 36
- Common.hh, 1007
- common/Plugin.hh
 - GZ_REGISTER_MODEL_PLUGIN, 1111
 - GZ_REGISTER_SENSOR_PLUGIN, 1111
 - GZ_REGISTER_SYSTEM_PLUGIN, 1112
 - GZ_REGISTER_VISUAL_PLUGIN, 1112
 - GZ_REGISTER_WORLD_PLUGIN, 1112
- CommonTypes.hh, 1008
 - GAZEBO_DEPRECATED, 1009
 - GAZEBO_FORCEINLINE, 1009
 - NULL, 1009
- Connect
 - Events, 41
 - gazebo::transport::Connection, 290
- ConnectAddEntity
 - Events, 42
- ConnectContact
 - gazebo::physics::Collision, 265
- ConnectCreateEntity
 - Events, 42
- ConnectCreateScene
 - gazebo::rendering::Events, 355
- ConnectDeleteEntity
 - Events, 43
- ConnectDiagTimerStart
 - Events, 43
- ConnectDiagTimerStop
 - Events, 43
- ConnectEnabled
 - gazebo::physics::Link, 461
- ConnectJointUpdate
 - gazebo::physics::Joint, 427
- ConnectNewDepthFrame
 - gazebo::rendering::DepthCamera, 314
- ConnectNewImageFrame
 - gazebo::rendering::Camera, 239
- ConnectNewLaserFrame
 - gazebo::rendering::GpuLaser, 376
 - gazebo::sensors::GpuRaySensor, 381
- ConnectNewLaserScans
 - gazebo::physics::MultiRayShape, 552
- ConnectNewRGBPointCloud
 - gazebo::rendering::DepthCamera, 315
- ConnectPause
 - Events, 44
- ConnectPostRender
 - Events, 44
- ConnectPreRender
 - Events, 44
- ConnectPubToSub
 - gazebo::transport::TopicManager, 863
- ConnectRemoveScene
 - gazebo::rendering::Events, 355
- ConnectRender
 - Events, 44
- ConnectSetSelectedEntity
 - Events, 45
- ConnectStep
 - Events, 45
- ConnectStop
 - Events, 45
- ConnectSubToPub
 - gazebo::transport::TopicManager, 863
- ConnectSubscribers
 - gazebo::transport::TopicManager, 863
- ConnectToRemoteHost
 - gazebo::transport::ConnectionManager, 293
- ConnectToShutdown
 - gazebo::transport::Connection, 290
- ConnectWorldCreated
 - Events, 46
- ConnectWorldUpdateEnd
 - Events, 46
- ConnectWorldUpdateStart
 - Events, 46
- Connection
 - gazebo::event::Connection, 287
 - gazebo::transport::Connection, 290
- Connection.hh, 1010
 - HEADER_LENGTH, 1012
- Connection_V
 - gazebo::event, 99
- ConnectionManager.hh, 1012
- ConnectionPtr
 - gazebo::event, 99
 - gazebo::transport, 117
- connections
 - gazebo::physics::Entity, 349
 - gazebo::rendering::Camera, 254
 - gazebo::sensors::Sensor, 771
- Console.hh, 1014
- constraint
 - gazebo::physics::BulletJoint, 191
- Contact
 - gazebo::physics::Contact, 297
- contact
 - gazebo::physics::ContactFeedback, 300
- Contact.hh, 1015
 - MAX_COLLIDE_RETURNS, 1016
 - MAX_CONTACT_JOINTS, 1016
- contactFiducial
 - gazebo::physics::RayShape, 722
- contactLen

- gazebo::physics::RayShape, 722
- ContactPtr
 - gazebo::physics, 108
- contactPub
 - gazebo::physics::PhysicsEngine, 662
- contactRetro
 - gazebo::physics::RayShape, 722
- ContactSensor
 - gazebo::sensors::ContactSensor, 301
- ContactSensor.hh, 1016
- ContactSensor_V
 - gazebo::sensors, 115
- ContactSensorPtr
 - gazebo::sensors, 115
- ContactVisual
 - gazebo::rendering::ContactVisual, 305
- ContactVisual.hh, 1017
- ContactVisualPtr
 - gazebo::rendering, 112
- control
 - gazebo::common::MouseEvent, 542
- Conversions.hh, 1018
- Convert
 - gazebo::rendering::Conversions, 306
 - Messages, 60–63
 - sdf::Converter, 307
- ConvertMass
 - gazebo::physics::BulletPhysics, 204
 - gazebo::physics::ODEPhysics, 612, 613
- ConvertPose
 - gazebo::physics::BulletPhysics, 204, 205
- Converter.hh, 1018
- CoordPoseSolve
 - gazebo::math::Pose, 679
- CoordPositionAdd
 - gazebo::math::Pose, 679, 680
- CoordPositionSub
 - gazebo::math::Pose, 680
- CoordRotationAdd
 - gazebo::math::Pose, 680
- CoordRotationSub
 - gazebo::math::Pose, 681
- Copy
 - sdf::Element, 334
- copyChildren
 - sdf, 119
- CopyNormals
 - gazebo::common::SubMesh, 821
- copyPose
 - urdf2gazebo::URDF2Gazebo, 873
- CopyVertices
 - gazebo::common::SubMesh, 821
- Correct
 - gazebo::math::Pose, 681
- gazebo::math::Quaternion, 701
- gazebo::math::Vector3, 906
- count
 - gazebo::physics::Contact, 298
- Create
 - gazebo::PluginT, 676
- create_scene
 - Rendering, 81
- create_sensor
 - Sensors, 85
- create_world
 - Classes for physics and dynamics, 72
- CreateBox
 - gazebo::common::MeshManager, 517
- CreateCamera
 - gazebo::common::MeshManager, 517
 - gazebo::rendering::Scene, 750
- CreateCollision
 - gazebo::physics::BulletPhysics, 205
 - gazebo::physics::ODEPhysics, 613
 - gazebo::physics::PhysicsEngine, 654, 655
- createCollision
 - urdf2gazebo::URDF2Gazebo, 873
- createCollisions
 - urdf2gazebo::URDF2Gazebo, 873
- CreateCone
 - gazebo::common::MeshManager, 518
- CreateContact
 - gazebo::physics::ODEPhysics, 613
- CreateCylinder
 - gazebo::common::MeshManager, 518
- CreateDepthCamera
 - gazebo::rendering::Scene, 750
- CreateDepthTexture
 - gazebo::rendering::DepthCamera, 315
- CreateDynamicLine
 - gazebo::rendering::Visual, 937
- createGeometry
 - urdf2gazebo::URDF2Gazebo, 873
- CreateGpuLaser
 - gazebo::rendering::Scene, 750
- CreateGrid
 - gazebo::rendering::Scene, 751
- createInertial
 - urdf2gazebo::URDF2Gazebo, 873
- CreateJoint
 - gazebo::physics::BulletPhysics, 205
 - gazebo::physics::ODEPhysics, 613
 - gazebo::physics::PhysicsEngine, 655
- createJoint
 - urdf2gazebo::URDF2Gazebo, 873
- CreateKeyFrame
 - gazebo::common::NumericAnimation, 571
 - gazebo::common::PoseAnimation, 686

- CreateLaserTexture
 - gazebo::rendering::GpuLaser, 377
- CreateLink
 - gazebo::physics::BulletPhysics, 205
 - gazebo::physics::ODEPhysics, 613
 - gazebo::physics::PhysicsEngine, 655
- createLink
 - urdf2gazebo::URDF2Gazebo, 873
- CreatePlane
 - gazebo::common::MeshManager, 518
 - gazebo::physics::BulletPlaneShape, 208
 - gazebo::physics::ODEPlaneShape, 619
 - gazebo::physics::PlaneShape, 672
- CreateRenderTexture
 - gazebo::rendering::Camera, 240
- CreateRequest
 - Messages, 63
- createSDF
 - urdf2gazebo::URDF2Gazebo, 874
- CreateScene
 - gazebo::rendering::RenderEngine, 724
- createScene
 - gazebo::rendering::Events, 356
- CreateSensor
 - gazebo::sensors::SensorManager, 775
- CreateShape
 - gazebo::physics::BulletPhysics, 205
 - gazebo::physics::ODEPhysics, 613
 - gazebo::physics::PhysicsEngine, 655
- CreateSphere
 - gazebo::common::MeshManager, 519
- CreateTimer
 - gazebo::common::DiagnosticManager, 321
- CreateTube
 - gazebo::common::MeshManager, 519
- CreateUserCamera
 - gazebo::rendering::Scene, 751
- CreateVertexDeclaration
 - gazebo::rendering::DynamicLines, 326
 - gazebo::rendering::DynamicRenderable, 329
- createVisual
 - urdf2gazebo::URDF2Gazebo, 874
- createVisuals
 - urdf2gazebo::URDF2Gazebo, 874
- CreateWindow
 - gazebo::rendering::GUIOverlay, 396
 - gazebo::rendering::WindowManager, 952
- Cross
 - gazebo::math::Vector2d, 888
 - gazebo::math::Vector2i, 896
 - gazebo::math::Vector3, 906
- cvfov
 - gazebo::sensors::GpuRaySensor, 388
- CylinderShape
 - gazebo::physics::CylinderShape, 309
- CylinderShape.hh, 1019
- CylinderShapePtr
 - gazebo::physics, 108
- d
 - gazebo::math::Plane, 670
- DEFERRED
 - gazebo::rendering::RenderEngine, 724
- DIAG_TIMER
 - Common, 34
- damping_coefficient
 - gazebo::physics::Joint, 438
- damping_factor
 - urdf2gazebo::GazeboExtension, 370
- data
 - sdf::Plugin, 675
- DebugCallbackHelper
 - gazebo::transport::DebugCallbackHelper, 312
- DebugPrint
 - gazebo::physics::BulletPhysics, 205
 - gazebo::physics::ODEPhysics, 613
 - gazebo::physics::PhysicsEngine, 655
- DecCount
 - gazebo::transport::IOManager, 423
- DecodeTopicName
 - gazebo::transport::Node, 558
- defaultValue
 - sdf::ParamT, 651
- Degree
 - gazebo::math::Angle, 129
- DeleteDynamicLine
 - gazebo::rendering::Visual, 937
- deleteEntity
 - Events, 50
- DepthCamera
 - gazebo::rendering::DepthCamera, 314
- DepthCamera.hh, 1020
- DepthCameraPtr
 - gazebo::rendering, 112
- DepthCameraSensor
 - gazebo::sensors::DepthCameraSensor, 318
- DepthCameraSensor.hh, 1021
- DepthCameraSensor_V
 - gazebo::sensors, 115
- DepthCameraSensorPtr
 - gazebo::sensors, 115
- depthTarget
 - gazebo::rendering::DepthCamera, 316
- depthTexture
 - gazebo::rendering::DepthCamera, 317
- depthViewport
 - gazebo::rendering::DepthCamera, 317
- depths

- gazebo::physics::Contact, 299
- description
 - sdf::Param, 647
- Detach
 - gazebo::physics::BulletJoint, 189
 - gazebo::physics::Joint, 428
 - gazebo::physics::ODEJoint, 596
- DetachAllStaticModels
 - gazebo::physics::Link, 461
- DetachEntity
 - gazebo::rendering::RTShaderSystem, 745
- DetachObjects
 - gazebo::rendering::Visual, 937
- DetachStaticModel
 - gazebo::physics::Link, 461
 - gazebo::physics::Model, 524
- DetachViewport
 - gazebo::rendering::RTShaderSystem, 745
- DetachVisual
 - gazebo::rendering::Visual, 937, 938
- diagTimerStart
 - Events, 50
- diagTimerStop
 - Events, 50
- DiagnosticTimer
 - gazebo::common::DiagnosticTimer, 324
- DiagnosticTimerPtr
 - Common, 36
- Diagnostics.hh, 1022
- diffuse
 - gazebo::common::Material, 497
- dirtyPose
 - gazebo::physics::Entity, 349
- dirtyPoses
 - gazebo::physics::World, 965
- DisableAllModels
 - gazebo::physics::World, 957
- DisableTrackVisual
 - gazebo::rendering::Visual, 938
- DisabledCallback
 - gazebo::physics::ODELink, 603
- Disconnect
 - Events, 46, 47
 - gazebo::event::Event, 351
- DisconnectAddEntity
 - Events, 47
- DisconnectContact
 - gazebo::physics::Collision, 266
- DisconnectCreateEntity
 - Events, 47
- DisconnectCreateScene
 - gazebo::rendering::Events, 355
- DisconnectDeleteEntity
 - Events, 47
- DisconnectDiagTimerStart
 - Events, 48
- DisconnectDiagTimerStop
 - Events, 48
- DisconnectEnabled
 - gazebo::physics::Link, 461
- DisconnectJointUpdate
 - gazebo::physics::Joint, 428
- DisconnectNewDepthFrame
 - gazebo::rendering::DepthCamera, 315
- DisconnectNewImageFrame
 - gazebo::rendering::Camera, 240
- DisconnectNewLaserFrame
 - gazebo::rendering::GpuLaser, 377
 - gazebo::sensors::GpuRaySensor, 381
- DisconnectNewLaserScans
 - gazebo::physics::MultiRayShape, 552
- DisconnectNewRGBPointCloud
 - gazebo::rendering::DepthCamera, 315
- DisconnectPause
 - Events, 48
- DisconnectPostRender
 - Events, 48
- DisconnectPreRender
 - Events, 48
- DisconnectPubFromSub
 - gazebo::transport::TopicManager, 863
- DisconnectRemoveScene
 - gazebo::rendering::Events, 356
- DisconnectRender
 - Events, 49
- DisconnectSetSelectedEntity
 - Events, 49
- DisconnectShutdown
 - gazebo::transport::Connection, 290
- DisconnectStep
 - Events, 49
- DisconnectStop
 - Events, 49
- DisconnectSubFromPub
 - gazebo::transport::TopicManager, 863
- DisconnectWorldCreated
 - Events, 49
- DisconnectWorldUpdateEnd
 - Events, 50
- DisconnectWorldUpdateStart
 - Events, 50
- Distance
 - gazebo::math::Plane, 670
 - gazebo::math::Vector2d, 888
 - gazebo::math::Vector2i, 896
 - gazebo::math::Vector3, 906
 - gazebo::math::Vector4, 917
- Dot

- gazebo::math::Quaternion, 701
- gazebo::math::Vector3, 907
- Double
 - gazebo::common::Time, 844
- DownloadDependencies
 - Common, 36
- dragging
 - gazebo::common::MouseEvent, 542
- DrawLine
 - gazebo::rendering::Scene, 751
- dummyContext
 - gazebo::rendering::RenderEngine, 726
- dummyDisplay
 - gazebo::rendering::RenderEngine, 726
- dummyWindowId
 - gazebo::rendering::RenderEngine, 726
- duration
 - gazebo::physics::TrajectoryInfo, 865
- DynamicLines
 - gazebo::rendering::DynamicLines, 326
- DynamicLines.hh, 1024
- DynamicLinesPtr
 - gazebo::rendering, 112
- DynamicRenderable
 - gazebo::rendering::DynamicRenderable, 329
- DynamicRenderable.hh, 1024
- ENTITY
 - gazebo::physics::Base, 149
- ERP
 - gazebo::physics::Joint, 426
- Element
 - sdf::Element, 334
- ElementPtr
 - sdf, 119
- ElementPtr_V
 - sdf, 119
- emissive
 - gazebo::common::Material, 497
- Enable
 - gazebo::rendering::Grid, 390
- EnableAllModels
 - gazebo::physics::World, 957
- EnablePhysicsEngine
 - gazebo::physics::World, 957
- EnableSaveFrame
 - gazebo::rendering::Camera, 240
- EnableTrackVisual
 - gazebo::rendering::Visual, 938
- EnableViewController
 - gazebo::rendering::UserCamera, 881
- enabled
 - gazebo::rendering::ViewController, 930
- EncodeTopicName
 - gazebo::transport::Node, 558
- End
 - gazebo::physics::Logplay, 480
- endTime
 - gazebo::physics::TrajectoryInfo, 865
- EnqueueMsg
 - gazebo::transport::Connection, 290
- Entity
 - gazebo::physics::Entity, 341
- Entity.hh, 1025
- entityCreated
 - Events, 51
- EntityPtr
 - gazebo::physics, 108
- EntityType
 - gazebo::physics::Base, 149
- EntityTypeName
 - Classes for physics and dynamics, 74
- Equal
 - gazebo::math::Vector3, 907
- equal
 - Math, 55
- erp
 - gazebo::physics::SurfaceParams, 832
- EulerToQuaternion
 - gazebo::math::Quaternion, 701
- Event.hh, 1026
- eventConnections
 - gazebo::transport::ConnectionManager, 294
- EventType
 - gazebo::common::MouseEvent, 541
- Events, 39
 - ~EventT, 41
 - addEntity, 50
 - Connect, 41
 - ConnectAddEntity, 42
 - ConnectCreateEntity, 42
 - ConnectDeleteEntity, 43
 - ConnectDiagTimerStart, 43
 - ConnectDiagTimerStop, 43
 - ConnectPause, 44
 - ConnectPostRender, 44
 - ConnectPreRender, 44
 - ConnectRender, 44
 - ConnectSetSelectedEntity, 45
 - ConnectStep, 45
 - ConnectStop, 45
 - ConnectWorldCreated, 46
 - ConnectWorldUpdateEnd, 46
 - ConnectWorldUpdateStart, 46
 - deleteEntity, 50
 - diagTimerStart, 50
 - diagTimerStop, 50
 - Disconnect, 46, 47

- DisconnectAddEntity, 47
- DisconnectCreateEntity, 47
- DisconnectDeleteEntity, 47
- DisconnectDiagTimerStart, 48
- DisconnectDiagTimerStop, 48
- DisconnectPause, 48
- DisconnectPostRender, 48
- DisconnectPreRender, 48
- DisconnectRender, 49
- DisconnectSetSelectedEntity, 49
- DisconnectStep, 49
- DisconnectStop, 49
- DisconnectWorldCreated, 49
- DisconnectWorldUpdateEnd, 50
- DisconnectWorldUpdateStart, 50
- entityCreated, 51
- pause, 51
- postRender, 51
- preRender, 51
- render, 51
- setSelectedEntity, 51
- step, 51
- stop, 51
- worldCreated, 52
- worldUpdateEnd, 52
- worldUpdateStart, 52
- Events.hh, 1027
- Exception
 - gazebo::common::Exception, 366
- Exception.hh, 1027
- FACE_MAX
 - STLLoader.hh, 1160
- FLAT
 - gazebo::common::Material, 491
- FMAX
 - gazebo::physics::Joint, 426
- FORWARD
 - gazebo::rendering::RenderEngine, 724
- FPSViewController
 - gazebo::rendering::FPSViewController, 368
- FPSViewController.hh, 1029
- FUDGE_FACTOR
 - gazebo::physics::Joint, 426
- fakeAnchor
 - gazebo::physics::ScrewJoint, 762
 - gazebo::physics::SliderJoint, 806
- far
 - gazebo::sensors::GpuRaySensor, 388
- fdir1
 - gazebo::physics::SurfaceParams, 832
 - urdf2gazebo::GazeboExtension, 371
- feedbackCount
 - gazebo::physics::ContactFeedback, 300
- feedbacks
 - gazebo::physics::ContactFeedback, 300
- filename
 - gazebo::PluginT, 677
 - sdf::Plugin, 675
- FillArrays
 - gazebo::common::Mesh, 511
 - gazebo::common::SubMesh, 821
- FillCollisionMsg
 - gazebo::physics::Collision, 266
- FillHardwareBuffers
 - gazebo::rendering::DynamicLines, 326
 - gazebo::rendering::DynamicRenderable, 329
- FillJointMsg
 - gazebo::physics::Joint, 428
- FillLinkMsg
 - gazebo::physics::Link, 462
- FillModelMsg
 - gazebo::physics::Model, 525
- FillMsg
 - gazebo::physics::BoxShape, 164
 - gazebo::physics::Collision, 266
 - gazebo::physics::CylinderShape, 309
 - gazebo::physics::HeightmapShape, 401
 - gazebo::physics::Joint, 428
 - gazebo::physics::Link, 462
 - gazebo::physics::MapShape, 485
 - gazebo::physics::Model, 525
 - gazebo::physics::MultiRayShape, 552
 - gazebo::physics::PlaneShape, 673
 - gazebo::physics::RayShape, 720
 - gazebo::physics::Shape, 781
 - gazebo::physics::SphereShape, 807
 - gazebo::physics::TrimeshShape, 867
 - gazebo::rendering::Light, 450
 - gazebo::sensors::Sensor, 768
- FillShapeMsg
 - gazebo::physics::BoxShape, 164
 - gazebo::physics::Shape, 781
- FillStateSDF
 - gazebo::physics::LinkState, 475
 - gazebo::physics::ModelState, 537
- FillSurfaceMsg
 - gazebo::physics::SurfaceParams, 831
- find_file
 - Common, 36
 - gazebo, 95
- find_file_path
 - Common, 36
- FindFile
 - gazebo::common::SystemPaths, 836
- FindFileURI
 - gazebo::common::SystemPaths, 836
- FindPublication

- gazebo::transport::TopicManager, 863
- Fini
 - gazebo::Master, 488
 - gazebo::physics::Actor, 124
 - gazebo::physics::Base, 150
 - gazebo::physics::BulletLink, 195
 - gazebo::physics::BulletPhysics, 205
 - gazebo::physics::Collision, 266
 - gazebo::physics::Entity, 342
 - gazebo::physics::Link, 462
 - gazebo::physics::Model, 525
 - gazebo::physics::ODECollision, 580
 - gazebo::physics::ODELink, 603
 - gazebo::physics::ODEPhysics, 613
 - gazebo::physics::PhysicsEngine, 655
 - gazebo::physics::World, 957
 - gazebo::rendering::Camera, 240
 - gazebo::rendering::DepthCamera, 316
 - gazebo::rendering::GpuLaser, 377
 - gazebo::rendering::RenderEngine, 725
 - gazebo::rendering::RTShaderSystem, 745
 - gazebo::rendering::UserCamera, 881
 - gazebo::rendering::Visual, 938
 - gazebo::rendering::WindowManager, 952
 - gazebo::sensors::CameraSensor, 257
 - gazebo::sensors::ContactSensor, 301
 - gazebo::sensors::DepthCameraSensor, 318
 - gazebo::sensors::GpuRaySensor, 382
 - gazebo::sensors::RaySensor, 713
 - gazebo::sensors::RFIDSensor, 728
 - gazebo::sensors::RFIDTag, 730
 - gazebo::sensors::Sensor, 768
 - gazebo::sensors::SensorManager, 775
 - gazebo::Server, 779
 - gazebo::transport::ConnectionManager, 293
 - gazebo::transport::Node, 558
 - gazebo::transport::PublicationTransport, 694
 - gazebo::transport::TopicManager, 863
- fini
 - Classes for physics and dynamics, 72
 - gazebo, 95
 - Rendering, 81
 - Sensors, 85
 - Transport, 89
- FiniChild
 - gazebo::sensors::ImuSensor, 413
- Float
 - gazebo::common::Time, 845
- FogFromSDF
 - Messages, 64
- forces
 - gazebo::physics::Contact, 299
- fudge_factor
 - urdf2gazebo::GazeboExtension, 371
- g
 - gazebo::common::Color, 284
 - GAZEBO_DEPRECATED
 - CommonTypes.hh, 1009
 - GAZEBO_FORCEINLINE
 - CommonTypes.hh, 1009
 - GOURAUD
 - gazebo::common::Material, 491
 - GUIFromSDF
 - Messages, 64
 - GUIOverlay
 - gazebo::rendering::GUIOverlay, 395
 - GUIOverlay.hh, 1035
 - GUIPluginPtr
 - gazebo, 95
 - GZ_ALL_COLLIDE
 - PhysicsTypes.hh, 1105
 - GZ_DBL_MAX
 - Helpers.hh, 1040
 - GZ_DBL_MIN
 - Helpers.hh, 1040
 - GZ_DTOR
 - Angle.hh, 973
 - GZ_FIXED_COLLIDE
 - PhysicsTypes.hh, 1105
 - GZ_FLT_MAX
 - Helpers.hh, 1040
 - GZ_FLT_MIN
 - Helpers.hh, 1040
 - GZ_GHOST_COLLIDE
 - PhysicsTypes.hh, 1105
 - GZ_NONE_COLLIDE
 - PhysicsTypes.hh, 1105
 - GZ_NORMALIZE
 - Angle.hh, 973
 - GZ_REGISTER_MODEL_PLUGIN
 - common/Plugin.hh, 1111
 - GZ_REGISTER_PHYSICS_ENGINE
 - Classes for physics and dynamics, 71
 - GZ_REGISTER_SENSOR_PLUGIN
 - common/Plugin.hh, 1111
 - GZ_REGISTER_STATIC_SENSOR
 - Sensors, 85
 - GZ_REGISTER_SYSTEM_PLUGIN
 - common/Plugin.hh, 1112
 - GZ_REGISTER_VISUAL_PLUGIN
 - common/Plugin.hh, 1112
 - GZ_REGISTER_WORLD_PLUGIN
 - common/Plugin.hh, 1112
 - GZ_RTOD
 - Angle.hh, 973
 - GZ_SENSOR_COLLIDE
 - PhysicsTypes.hh, 1105
 - GZ_VISIBILITY_ALL

- RenderTypes.hh, 1129
- GZ_VISIBILITY_GUI
 - RenderTypes.hh, 1129
- GZ_VISIBILITY_NOT_SELECTABLE
 - RenderTypes.hh, 1129
- gazebo, 93
 - add_plugin, 95
 - find_file, 95
 - fini, 95
 - GUIPluginPtr, 95
 - init, 95
 - load, 95
 - ModelPluginPtr, 95
 - print_version, 95
 - run, 95
 - SensorPluginPtr, 95
 - stop, 95
 - SystemPluginPtr, 95
 - VisualPluginPtr, 95
 - WorldPluginPtr, 95
- gazebo.hh, 1029
- gazebo::Master, 487
 - ~Master, 488
 - Fini, 488
 - Init, 488
 - Master, 488
 - Run, 488
 - RunOnce, 488
 - RunThread, 488
 - Stop, 488
- gazebo::ModelPlugin, 534
 - ~ModelPlugin, 535
 - Init, 535
 - Load, 535
 - ModelPlugin, 535
 - Reset, 536
- gazebo::PluginT
 - Create, 676
 - filename, 677
 - GetFilename, 676
 - GetHandle, 676
 - GetType, 676
 - handle, 677
 - TPtr, 676
 - type, 677
- gazebo::PluginT< T >, 675
- gazebo::SensorPlugin, 777
 - ~SensorPlugin, 777
 - Init, 778
 - Load, 778
 - Reset, 778
 - SensorPlugin, 777
- gazebo::Server, 778
 - ~Server, 779
- Fin, 779
- GetInitialized, 779
- Init, 779
- Load, 779
- ParseArgs, 779
- PrintUsage, 779
- Run, 779
- Server, 779
- SetParams, 779
- Stop, 779
- systemPluginsArgc, 779
- systemPluginsArgv, 779
- gazebo::SystemPlugin, 838
 - ~SystemPlugin, 839
 - Init, 839
 - Load, 839
 - Reset, 840
 - SystemPlugin, 839
- gazebo::VisualPlugin, 949
 - Init, 950
 - Load, 950
 - Reset, 951
 - VisualPlugin, 950
- gazebo::WorldPlugin, 965
 - ~WorldPlugin, 966
 - Init, 966
 - Load, 966
 - Reset, 966
 - WorldPlugin, 966
- gazebo::common, 95
 - AnimationPtr, 98
 - NodeMap, 98
 - NodeMapItr, 98
 - NumericAnimationPtr, 98
 - Param_V, 98
 - PoseAnimationPtr, 98
 - RawNodeAnim, 98
 - RawNodeWeights, 98
 - RawSkeletonAnim, 98
 - StrStr_M, 98
- gazebo::common::Animation, 135
 - ~Animation, 136
 - AddTime, 137
 - Animation, 136
 - build, 138
 - GetKeyFrame, 137
 - GetKeyFrameCount, 137
 - GetKeyFramesAtTime, 137
 - GetLength, 137
 - GetTime, 138
 - KeyFrame_V, 136
 - keyFrames, 138
 - length, 138
 - loop, 138

- name, 139
- SetLength, 138
- SetTime, 138
- timePos, 139
- gazebo::common::BVHLoader, 229
 - ~BVHLoader, 229
 - BVHLoader, 229
 - Load, 229
- gazebo::common::ColladaLoader, 261
 - ~ColladaLoader, 262
 - ColladaLoader, 262
 - Load, 262
- gazebo::common::Color, 274
 - ~Color, 277
 - a, 284
 - ABGR, 276
 - ARGB, 276
 - b, 284
 - BGRA, 276
 - Black, 284
 - Blue, 284
 - Color, 276, 277
 - g, 284
 - GetAsABGR, 277
 - GetAsARGB, 277
 - GetAsBGRA, 277
 - GetAsHSV, 278
 - GetAsRGBA, 278
 - GetAsYUV, 278
 - Green, 284
 - operator<<, 284
 - operator>>, 284
 - operator*, 278, 279
 - operator*=:, 279
 - operator+, 279
 - operator+=, 280
 - operator-, 280
 - operator-=, 280
 - operator/, 281
 - operator/=, 281
 - operator=, 281
 - operator==, 282
 - operator[], 282
 - Purple, 284
 - r, 284
 - RGBA, 276
 - Red, 285
 - Reset, 282
 - Set, 282
 - SetFromABGR, 282
 - SetFromARGB, 283
 - SetFromBGRA, 283
 - SetFromHSV, 283
 - SetFromRGBA, 283
 - SetFromYUV, 283
 - White, 285
 - Yellow, 285
- gazebo::common::Console, 295
 - ColorErr, 295
 - ColorMsg, 296
 - Instance, 296
 - Load, 296
 - Log, 296
 - SetQuiet, 296
- gazebo::common::DiagnosticManager, 320
 - CreateTimer, 321
 - GetEnabled, 321
 - GetLabel, 321
 - GetTime, 322
 - GetTimerCount, 322
 - SetEnabled, 322
 - TimerStart, 322
 - TimerStop, 323
- gazebo::common::DiagnosticTimer, 323
 - ~DiagnosticTimer, 324
 - DiagnosticTimer, 324
 - GetName, 324
- gazebo::common::Exception, 365
 - ~Exception, 366
 - Exception, 366
 - GetErrorFile, 366
 - GetErrorStr, 366
 - operator<<, 366
- gazebo::common::Image, 407
 - ~Image, 409
 - BAYER_GBRG8, 409
 - BAYER_GRBG8, 409
 - BAYER_RGGB8, 409
 - BAYER_RGGR8, 409
 - BGR_INT16, 408
 - BGR_INT32, 408
 - BGR_INT8, 408
 - BGRA_INT8, 408
 - GetAvgColor, 409
 - GetBPP, 409
 - GetData, 409
 - GetFilename, 409
 - GetHeight, 410
 - GetMaxColor, 410
 - GetPitch, 410
 - GetPixel, 410
 - GetPixelFormat, 410
 - GetRGBData, 410
 - GetWidth, 411
 - Image, 409
 - L_INT16, 408
 - L_INT8, 408
 - Load, 411

- PixelFormat, 408
- R_FLOAT16, 408
- R_FLOAT32, 408
- RGB_FLOAT16, 408
- RGB_FLOAT32, 409
- RGB_INT16, 408
- RGB_INT32, 408
- RGB_INT8, 408
- RGBA_INT8, 408
- Rescale, 411
- SavePNG, 411
- SetFromData, 411
- UNKNOWN, 408
- Valid, 412
- gazebo::common::KeyFrame, 445
 - ~KeyFrame, 446
 - GetTime, 446
 - KeyFrame, 446
 - time, 446
- gazebo::common::Material, 489
 - ~Material, 492
 - ADD, 491
 - ambient, 497
 - BLEND_COUNT, 491
 - BLINN, 491
 - BlendMode, 491
 - blendMode, 497
 - BlendModeStr, 497
 - diffuse, 497
 - emissive, 497
 - FLAT, 491
 - GOURAUD, 491
 - GetAmbient, 492
 - GetBlendFactors, 492
 - GetBlendMode, 492
 - GetDepthWrite, 492
 - GetDiffuse, 493
 - GetEmissive, 493
 - GetLighting, 493
 - GetName, 493
 - GetPointSize, 493
 - GetShadeMode, 493
 - GetShininess, 493
 - GetSpecular, 494
 - GetTextureImage, 494
 - GetTransparency, 494
 - MODULATE, 491
 - Material, 492
 - name, 497
 - operator<<, 496
 - PHONG, 491
 - pointSize, 497
 - REPLACE, 491
 - SHADE_COUNT, 491
 - SetAmbient, 494
 - SetBlendFactors, 494
 - SetBlendMode, 495
 - SetDepthWrite, 495
 - SetDiffuse, 495
 - SetEmissive, 495
 - SetLighting, 495
 - SetPointSize, 495
 - SetShadeMode, 495
 - SetShininess, 496
 - SetSpecular, 496
 - SetTextureImage, 496
 - SetTransparency, 496
 - ShadeMode, 491
 - shadeMode, 497
 - ShadeModeStr, 497
 - shininess, 497
 - specular, 497
 - texImage, 497
 - transparency, 498
- gazebo::common::Mesh, 509
 - ~Mesh, 510
 - AddMaterial, 510
 - AddSubMesh, 510
 - FillArrays, 511
 - GenSphericalTexCoord, 511
 - GetAABB, 511
 - GetIndexCount, 511
 - GetMaterial, 511
 - GetMaterialCount, 512
 - GetMax, 512
 - GetMin, 512
 - GetName, 512
 - GetNormalCount, 512
 - GetPath, 512
 - GetSkeleton, 512
 - GetSubMesh, 513
 - GetSubMeshCount, 513
 - GetTexCoordCount, 513
 - GetVertexCount, 513
 - HasSkeleton, 513
 - Mesh, 510
 - RecalculateNormals, 513
 - Scale, 514
 - SetName, 514
 - SetPath, 514
 - SetSkeleton, 514
- gazebo::common::MeshLoader, 514
 - ~MeshLoader, 515
 - Load, 515
 - MeshLoader, 515
- gazebo::common::MeshManager, 516
 - AddMesh, 517
 - CreateBox, 517

- CreateCamera, 517
- CreateCone, 518
- CreateCylinder, 518
- CreatePlane, 518
- CreateSphere, 519
- CreateTube, 519
- GenSphericalTexCoord, 519
- GetMesh, 519
- GetMeshAABB, 520
- HasMesh, 520
- IsValidFilename, 520
- Load, 520
- gazebo::common::ModelDatabase, 533
- gazebo::common::MouseEvent, 540
 - alt, 542
 - button, 542
 - Buttons, 541
 - buttons, 542
 - control, 542
 - dragging, 542
 - EventType, 541
 - LEFT, 541
 - MIDDLE, 541
 - MOVE, 541
 - MouseEvent, 541
 - moveScale, 542
 - NO_BUTTON, 541
 - NO_EVENT, 541
 - PRESS, 541
 - pos, 542
 - pressPos, 542
 - prevPos, 542
 - RELEASE, 541
 - RIGHT, 541
 - SCROLL, 541
 - scroll, 542
 - shift, 542
 - type, 542
- gazebo::common::NodeAnimation, 560
 - ~NodeAnimation, 561
 - AddKeyFrame, 561
 - GetFrameAt, 561
 - GetFrameCount, 561
 - GetKeyFrame, 562
 - GetLength, 562
 - GetName, 562
 - GetTimeAtX, 562
 - keyFrames, 563
 - length, 563
 - name, 563
 - NodeAnimation, 561
 - Scale, 563
 - SetName, 563
- gazebo::common::NodeAssignment, 563
 - nodeIndex, 564
 - vertexIndex, 564
 - weight, 564
- gazebo::common::NodeTransform, 564
 - ~NodeTransform, 566
 - Get, 566
 - GetSID, 567
 - GetType, 567
 - MATRIX, 566
 - NodeTransform, 566
 - operator*, 567
 - operator(), 567
 - PrintSource, 568
 - ROTATE, 566
 - RecalculateMatrix, 568
 - SCALE, 566
 - Set, 568
 - SetComponent, 568
 - SetSID, 568
 - SetSourceValues, 568
 - SetType, 569
 - sid, 569
 - source, 569
 - TRANSLATE, 566
 - transform, 569
 - TransformType, 566
 - type, 569
- gazebo::common::NumericAnimation, 569
 - ~NumericAnimation, 570
 - CreateKeyFrame, 571
 - GetInterpolatedKeyFrame, 571
 - NumericAnimation, 570
- gazebo::common::NumericKeyFrame, 571
 - ~NumericKeyFrame, 572
 - GetValue, 572
 - NumericKeyFrame, 572
 - SetValue, 572
 - value, 573
- gazebo::common::PID, 664
 - ~PID, 665
 - GetCmd, 666
 - GetErrors, 666
 - Init, 666
 - operator=, 666
 - PID, 665
 - Reset, 666
 - SetCmd, 667
 - SetCmdMax, 667
 - SetCmdMin, 667
 - SetDGain, 667
 - SetIGain, 667
 - SetIMax, 667
 - SetIMin, 668
 - SetPGain, 668

- Update, 668
- gazebo::common::PoseAnimation, 685
 - ~PoseAnimation, 686
 - BuildInterpolationSplines, 686
 - CreateKeyFrame, 686
 - GetInterpolatedKeyFrame, 686, 687
 - PoseAnimation, 686
- gazebo::common::PoseKeyFrame, 687
 - ~PoseKeyFrame, 688
 - GetRotation, 688
 - GetTranslation, 688
 - PoseKeyFrame, 688
 - rotate, 689
 - SetRotation, 689
 - SetTranslation, 689
 - translate, 689
- gazebo::common::STLLoader, 816
 - ~STLLoader, 816
 - Load, 817
 - STLLoader, 816
- gazebo::common::Skeleton, 784
 - ~Skeleton, 786
 - AddAnimation, 786
 - AddVertNodeWeight, 786
 - anims, 790
 - bindShapeTransform, 790
 - BuildNodeMap, 787
 - GetAnimation, 787
 - GetBindShapeTransform, 787
 - GetNodeByHandle, 787
 - GetNodeById, 787
 - GetNodeByName, 788
 - GetNodes, 788
 - GetNumAnimations, 788
 - GetNumJoints, 788
 - GetNumNodes, 788
 - GetNumVertNodeWeights, 788
 - GetRootNode, 789
 - GetVertNodeWeight, 789
 - nodes, 790
 - PrintTransforms, 789
 - rawNW, 790
 - root, 790
 - Scale, 789
 - SetBindShapeTransform, 789
 - SetNumVertAttached, 790
 - SetRootNode, 790
 - Skeleton, 786
- gazebo::common::SkeletonAnimation, 791
 - ~SkeletonAnimation, 792
 - AddKeyFrame, 792
 - animations, 794
 - GetLength, 792
 - GetName, 793
 - GetNodeCount, 793
 - GetNodePoseAt, 793
 - GetPoseAt, 793
 - GetPoseAtX, 794
 - HasNode, 794
 - length, 794
 - name, 795
 - Scale, 794
 - SetName, 794
 - SkeletonAnimation, 792
- gazebo::common::SkeletonNode, 795
 - ~SkeletonNode, 798
 - AddChild, 798
 - AddRawTransform, 798
 - children, 803
 - GetChild, 798
 - GetChildById, 798
 - GetChildByName, 799
 - GetChildCount, 799
 - GetHandle, 799
 - GetId, 799
 - GetInverseBindTransform, 799
 - GetModelTransform, 799
 - GetName, 800
 - GetNumRawTrans, 800
 - GetParent, 800
 - GetRawTransform, 800
 - GetRawTransforms, 800
 - GetTransform, 801
 - GetTransforms, 801
 - handle, 803
 - id, 803
 - initialTransform, 803
 - invBindTransform, 803
 - IsJoint, 801
 - IsRootNode, 801
 - JOINT, 797
 - modelTransform, 803
 - NODE, 797
 - name, 803
 - parent, 804
 - rawTransforms, 804
 - Reset, 801
 - SetHandle, 801
 - SetId, 801
 - SetInitialTransform, 802
 - SetInverseBindTransform, 802
 - SetModelTransform, 802
 - SetName, 802
 - SetParent, 802
 - SetTransform, 802
 - SetType, 803
 - SkeletonNode, 797
 - SkeletonNodeType, 797

- transform, 804
- type, 804
- UpdateChildrenTransforms, 803
- gazebo::common::SubMesh, 817
 - ~SubMesh, 819
 - AddIndex, 820
 - AddNodeAssignment, 820
 - AddNormal, 820
 - AddTexCoord, 820
 - AddVertex, 821
 - CopyNormals, 821
 - CopyVertices, 821
 - FillArrays, 821
 - GenSphericalTexCoord, 821
 - GetIndex, 822
 - GetIndexCount, 822
 - GetMaterialIndex, 822
 - GetMax, 822
 - GetMaxIndex, 822
 - GetMin, 822
 - GetNodeAssignment, 822
 - GetNodeAssignmentsCount, 822
 - GetNormal, 823
 - GetNormalCount, 823
 - GetPrimitiveType, 823
 - GetTexCoord, 823
 - GetTexCoordCount, 823
 - GetVertex, 823
 - GetVertexCount, 824
 - GetVertexIndex, 824
 - HasVertex, 824
 - LINES, 819
 - LINESTRIPS, 819
 - POINTS, 819
 - PrimitiveType, 819
 - RecalculateNormals, 824
 - Scale, 824
 - SetIndexCount, 824
 - SetMaterialIndex, 824
 - SetNormal, 825
 - SetNormalCount, 825
 - SetPrimitiveType, 825
 - SetSubMeshCenter, 825
 - SetTexCoord, 825
 - SetTexCoordCount, 826
 - SetVertex, 826
 - SetVertexCount, 826
 - SubMesh, 819
 - TRIANGLES, 819
 - TRIFANS, 819
 - TRISTRIPS, 819
- gazebo::common::SystemPaths, 834
 - AddGazeboPaths, 835
 - AddOgrePaths, 836
 - AddPluginPaths, 836
 - AddSearchPathSuffix, 836
 - ClearGazeboPaths, 836
 - ClearOgrePaths, 836
 - ClearPluginPaths, 836
 - FindFile, 836
 - FindFileURI, 836
 - gazeboPathsFromEnv, 838
 - GetGazeboPaths, 837
 - GetLogPath, 837
 - GetModelPaths, 837
 - GetOgrePaths, 837
 - GetPluginPaths, 837
 - GetWorldPathExtension, 837
 - modelPathsFromEnv, 838
 - ogrePathsFromEnv, 838
 - pluginPathsFromEnv, 838
- gazebo::common::Time, 840
 - ~Time, 844
 - Double, 844
 - Float, 845
 - GetWallTime, 845
 - MSleep, 845
 - MicToNano, 845
 - MilToNano, 845
 - NSleep, 846
 - nsec, 859
 - operator<, 852, 853
 - operator<<, 859
 - operator<=, 853, 854
 - operator>, 856, 857
 - operator>>, 859
 - operator>=, 857, 858
 - operator*, 847
 - operator*=:, 848
 - operator+, 848, 849
 - operator+=, 849
 - operator-, 850
 - operator-=, 850
 - operator/, 851
 - operator/=, 851, 852
 - operator=, 854, 855
 - operator==, 855, 856
 - sec, 859
 - SecToNano, 858
 - Set, 858
 - SetToWallTime, 858
 - Time, 843, 844
- gazebo::common::Timer, 859
 - ~Timer, 860
 - GetElapsed, 861
 - operator<<, 861
 - Start, 861
 - Timer, 860

- gazebo::common::Video, 924
 - ~Video, 925
 - GetHeight, 925
 - GetNextFrame, 925
 - GetWidth, 925
 - Load, 925
 - Video, 925
- gazebo::event, 98
 - Connection_V, 99
 - ConnectionPtr, 99
- gazebo::event::Connection, 287
 - ~Connection, 287
 - Connection, 287
 - GetId, 288
- gazebo::event::Event, 350
 - ~Event, 351
 - Disconnect, 351
- gazebo::event::EventT
 - operator(), 359–361
 - Signal, 362–364
- gazebo::event::EventT< T >, 356
- gazebo::event::Events, 352
- gazebo::math, 99
 - GeneratorType, 101
 - NRealGen, 101
 - NormalRealDist, 101
 - UIntGen, 101
 - URealGen, 101
 - UniformIntDist, 101
 - UniformRealDist, 101
- gazebo::math::Angle, 127
 - ~Angle, 129
 - Angle, 129
 - Degree, 129
 - GetAsDegree, 129
 - GetAsRadian, 129
 - Normalize, 130
 - operator<, 132
 - operator<<, 134
 - operator<=, 132
 - operator>, 133
 - operator>>, 134
 - operator>=, 133
 - operator*, 130
 - operator*=:, 130
 - operator+, 131
 - operator+=, 131
 - operator-, 131
 - operator-=, 131
 - operator/, 132
 - operator/=, 132
 - operator==, 133
 - Radian, 133
 - SetFromDegree, 134
 - SetFromRadian, 134
- gazebo::math::Box, 157
 - ~Box, 159
 - Box, 159
 - GetCenter, 159
 - GetSize, 159
 - GetXLength, 159
 - GetYLength, 160
 - GetZLength, 160
 - max, 162
 - Merge, 160
 - min, 162
 - operator<<, 161
 - operator+, 160
 - operator+=, 160
 - operator-, 161
 - operator=, 161
 - operator==, 161
- gazebo::math::Matrix3, 498
 - ~Matrix3, 499
 - m, 501
 - Matrix3, 499
 - operator<<, 501
 - operator==, 499
 - operator[], 500
 - SetCol, 500
 - SetFromAxes, 500
 - SetFromAxis, 501
- gazebo::math::Matrix4, 501
 - ~Matrix4, 504
 - GetAsPose, 504
 - GetEulerRotation, 504
 - GetRotation, 504
 - GetTranslation, 504
 - IDENTITY, 508
 - Inverse, 504
 - IsAffine, 505
 - m, 508
 - Matrix4, 503
 - operator<<, 508
 - operator*, 505
 - operator=, 505, 506
 - operator==, 506
 - operator[], 506
 - Set, 507
 - SetScale, 507
 - SetTranslate, 507
 - TransformAffine, 508
 - ZERO, 508
- gazebo::math::Plane, 668
 - ~Plane, 670
 - d, 670
 - Distance, 670
 - normal, 671

- operator=, 670
- Plane, 669
- Set, 670
- size, 671
- gazebo::math::Pose, 677
 - ~Pose, 679
 - CoordPoseSolve, 679
 - CoordPositionAdd, 679, 680
 - CoordPositionSub, 680
 - CoordRotationAdd, 680
 - CoordRotationSub, 681
 - Correct, 681
 - GetInverse, 681
 - IsFinite, 681
 - operator<<, 684
 - operator>>, 684
 - operator*, 681
 - operator+, 682
 - operator+=, 682
 - operator-, 682
 - operator-=, 682
 - operator==, 683
 - pos, 685
 - Pose, 679
 - Reset, 683
 - rot, 685
 - RotatePositionAboutOrigin, 683
 - Round, 683
 - Set, 683, 684
- gazebo::math::Quaternion, 697
 - ~Quaternion, 701
 - Correct, 701
 - Dot, 701
 - EulerToQuaternion, 701
 - GetAsAxis, 702
 - GetAsEuler, 702
 - GetAsMatrix3, 702
 - GetAsMatrix4, 702
 - GetExp, 702
 - GetInverse, 702
 - GetLog, 702
 - GetPitch, 703
 - GetRoll, 703
 - GetXAxis, 703
 - GetYAxis, 703
 - GetYaw, 703
 - GetZAxis, 703
 - Invert, 704
 - IsFinite, 704
 - Normalize, 704
 - operator<<, 709
 - operator>>, 709
 - operator*, 704, 705
 - operator*=:, 705
 - operator+, 705
 - operator+=, 705
 - operator-, 706
 - operator-=, 706
 - operator=, 706
 - operator==, 706
 - Quaternion, 700
 - RotateVector, 707
 - RotateVectorReverse, 707
 - Round, 707
 - Scale, 707
 - Set, 708
 - SetFromAxis, 708
 - SetFromEuler, 708
 - SetToIdentity, 708
 - Slerp, 708
 - Squad, 709
 - w, 710
 - x, 710
 - y, 710
 - z, 710
- gazebo::math::Rand, 710
 - GetDblNormal, 711
 - GetDblUniform, 711
 - GetIntNormal, 711
 - GetIntUniform, 711
- gazebo::math::RotationSpline, 738
 - ~RotationSpline, 739
 - AddPoint, 739
 - autoCalc, 742
 - Clear, 740
 - GetNumPoints, 740
 - GetPoint, 740
 - Interpolate, 740, 741
 - points, 742
 - RecalcTangents, 741
 - RotationSpline, 739
 - SetAutoCalculate, 741
 - tangents, 742
 - UpdatePoint, 741
- gazebo::math::Spline, 808
 - ~Spline, 809
 - AddPoint, 810
 - autoCalc, 812
 - Clear, 810
 - coeffs, 812
 - GetPoint, 810
 - GetPointCount, 810
 - GetTangent, 810
 - GetTension, 810
 - Interpolate, 811
 - points, 812
 - RecalcTangents, 811
 - SetAutoCalculate, 811

- SetTension, 811
- Spline, 809
- tangents, 812
- tension, 812
- UpdatePoint, 812
- gazebo::math::Vector2d, 885
 - ~Vector2d, 888
 - Cross, 888
 - Distance, 888
 - IsFinite, 888
 - Normalize, 888
 - operator<<, 893
 - operator>>, 893
 - operator*, 888, 889
 - operator*=: 889
 - operator+, 890
 - operator+=: 890
 - operator-, 890
 - operator-=: 890
 - operator/, 890, 891
 - operator/=, 891
 - operator=, 892
 - operator==, 892
 - operator[], 892
 - Set, 893
 - Vector2d, 887
 - x, 893
 - y, 893
- gazebo::math::Vector2i, 894
 - ~Vector2i, 896
 - Cross, 896
 - Distance, 896
 - IsFinite, 896
 - Normalize, 897
 - operator<<, 902
 - operator>>, 902
 - operator*, 897
 - operator*=: 898
 - operator+, 898
 - operator+=: 898
 - operator-, 899
 - operator-=: 899
 - operator/, 899
 - operator/=, 900
 - operator=, 900, 901
 - operator==, 901
 - operator[], 901
 - Set, 901
 - Vector2i, 895, 896
 - x, 902
 - y, 902
- gazebo::math::Vector3, 902
 - ~Vector3, 905
 - Correct, 906
 - Cross, 906
 - Distance, 906
 - Dot, 907
 - Equal, 907
 - GetAbs, 907
 - GetDistToLine, 907
 - GetLength, 907
 - GetMax, 908
 - GetMin, 908
 - GetNormal, 908
 - GetPerpendicular, 908
 - GetRounded, 908
 - GetSquaredLength, 908
 - GetSum, 909
 - IsFinite, 909
 - Normalize, 909
 - operator<<, 914
 - operator>>, 914
 - operator*, 909, 910
 - operator*=: 910
 - operator+, 910
 - operator+=: 911
 - operator-, 911
 - operator-=: 911
 - operator/, 911
 - operator/=, 912
 - operator=, 912
 - operator==, 913
 - operator[], 913
 - Round, 913
 - Set, 913
 - SetToMax, 913
 - SetToMin, 914
 - Vector3, 905
 - x, 914
 - y, 915
 - z, 915
- gazebo::math::Vector4, 915
 - ~Vector4, 917
 - Distance, 917
 - GetLength, 918
 - GetSquaredLength, 918
 - IsFinite, 918
 - Normalize, 918
 - operator<<, 923
 - operator>>, 923
 - operator*, 918, 919
 - operator*=: 919, 920
 - operator+, 920
 - operator+=: 920
 - operator-, 920
 - operator-=: 921
 - operator/, 921
 - operator/=, 921, 922

- operator=, 922
- operator==, 922
- operator[], 923
- Set, 923
- Vector4, 917
- w, 924
- x, 924
- y, 924
- z, 924
- gazebo::msgs, 101
- gazebo::physics, 103
 - Actor_V, 108
 - ActorPtr, 108
 - Base_V, 108
 - BasePtr, 108
 - BoxShapePtr, 108
 - BulletCollisionPtr, 108
 - BulletLinkPtr, 108
 - BulletPhysicsPtr, 108
 - BulletRayShapePtr, 108
 - Collision_V, 108
 - CollisionPtr, 108
 - ContactPtr, 108
 - CylinderShapePtr, 108
 - EntityPtr, 108
 - HeightmapShapePtr, 109
 - InertialPtr, 109
 - Joint_V, 109
 - JointPtr, 109
 - Link_V, 109
 - LinkPtr, 109
 - MeshShapePtr, 109
 - Model_V, 109
 - ModelPtr, 109
 - MultiRayShapePtr, 109
 - ODECollisionPtr, 109
 - ODELinkPtr, 109
 - ODERayShapePtr, 109
 - PhysicsEnginePtr, 109
 - RayShapePtr, 109
 - RoadPtr, 109
 - ShapePtr, 109
 - SphereShapePtr, 109
 - SurfaceParamsPtr, 109
 - WorldPtr, 109
- gazebo::physics::Actor, 121
 - ~Actor, 124
 - active, 125
 - Actor, 123
 - autoStart, 125
 - bonePosePub, 125
 - Fini, 124
 - GetSDF, 124
 - Init, 124
 - interpolateX, 125
 - isActive, 124
 - lastPos, 125
 - lastScriptTime, 125
 - lastTraj, 125
 - Load, 124
 - loop, 125
 - mainLink, 126
 - mesh, 126
 - oldAction, 126
 - pathLength, 126
 - Play, 124
 - playStartTime, 126
 - prevFrameTime, 126
 - scriptLength, 126
 - skelAnimation, 126
 - skelNodesMap, 126
 - skeleton, 126
 - skinFile, 126
 - skinScale, 127
 - startDelay, 127
 - Stop, 124
 - trajInfo, 127
 - trajectories, 127
 - Update, 124
 - UpdateParameters, 125
 - visualName, 127
- gazebo::physics::BallJoint
 - ~BallJoint, 144
 - BallJoint, 144
 - GetHighStop, 144
 - GetLowStop, 144
 - Load, 144
 - SetAxis, 144
 - SetHighStop, 144
 - SetLowStop, 144
- gazebo::physics::BallJoint< T >, 143
- gazebo::physics::Base, 145
 - ~Base, 150
 - ACTOR, 149
 - AddChild, 150
 - AddType, 150
 - BALL_JOINT, 149
 - BASE, 149
 - BOX_SHAPE, 149
 - Base, 150
 - COLLISION, 149
 - CYLINDER_SHAPE, 149
 - children, 157
 - childrenEnd, 157
 - ENTITY, 149
 - EntityType, 149
 - Fini, 150
 - GetById, 150

- GetByName, 151
- GetChild, 151
- GetChildCount, 151
- GetId, 152
- GetName, 152
- GetParent, 152
- GetParentId, 152
- GetSDF, 153
- GetSaveable, 152
- GetScopedName, 152
- GetType, 153
- GetWorld, 153
- HEIGHTMAP_SHAPE, 149
- HINGE2_JOINT, 149
- HINGE_JOINT, 149
- HasType, 153
- Init, 153
- IsSelected, 154
- JOINT, 149
- LIGHT, 149
- LINK, 149
- Load, 154
- MAP_SHAPE, 149
- MODEL, 149
- MULTIRAY_SHAPE, 149
- operator==, 154
- PLANE_SHAPE, 150
- parent, 157
- Print, 155
- RAY_SHAPE, 150
- RemoveChild, 155
- RemoveChildren, 155
- Reset, 155
- SCREW_JOINT, 149
- SHAPE, 149
- SLIDER_JOINT, 149
- SPHERE_SHAPE, 150
- sdf, 157
- SetName, 155
- SetParent, 156
- SetSaveable, 156
- SetSelected, 156
- SetWorld, 156
- TRIMESH_SHAPE, 150
- UNIVERSAL_JOINT, 149
- Update, 156
- UpdateParameters, 157
- VISUAL, 149
- world, 157
- gazebo::physics::BoxShape, 162
 - ~BoxShape, 164
 - BoxShape, 164
 - FillMsg, 164
 - FillShapeMsg, 164
- GetInertial, 164
- GetMass, 164
- GetSize, 165
- Init, 165
- ProcessMsg, 165
- SetSize, 165
- gazebo::physics::BulletBallJoint, 166
 - ~BulletBallJoint, 167
 - Attach, 168
 - BulletBallJoint, 167
 - GetAnchor, 168
 - GetAngle, 168
 - GetAngleImpl, 168
 - GetAxis, 168
 - GetGlobalAxis, 168
 - GetMaxForce, 168
 - GetVelocity, 168
 - SetAnchor, 168
 - SetDamping, 168
 - SetHighStop, 169
 - SetLowStop, 169
 - SetMaxForce, 169
 - SetVelocity, 169
- gazebo::physics::BulletBoxShape, 169
 - ~BulletBoxShape, 171
 - BulletBoxShape, 170
 - SetSize, 171
- gazebo::physics::BulletCollision, 171
 - ~BulletCollision, 173
 - BulletCollision, 173
 - collisionShape, 174
 - GetBoundingBox, 173
 - GetCollisionShape, 173
 - Load, 173
 - OnPoseChange, 173
 - SetCategoryBits, 173
 - SetCollideBits, 174
 - SetCollisionShape, 174
 - SetCompoundShapeIndex, 174
- gazebo::physics::BulletCylinderShape, 174
 - ~BulletCylinderShape, 176
 - BulletCylinderShape, 176
 - SetSize, 176
- gazebo::physics::BulletHeightmapShape, 176
 - ~BulletHeightmapShape, 178
 - BulletHeightmapShape, 178
 - Init, 178
- gazebo::physics::BulletHinge2Joint, 178
 - ~BulletHinge2Joint, 180
 - Attach, 181
 - BulletHinge2Joint, 180
 - GetAnchor, 181
 - GetAngle, 181
 - GetAngleImpl, 181

- GetAxis, 181
- GetGlobalAxis, 181
- GetHighStop, 181
- GetLowStop, 181
- GetMaxForce, 181
- GetVelocity, 182
- Load, 182
- SetAnchor, 182
- SetAxis, 182
- SetDamping, 182
- SetForce, 182
- SetHighStop, 182
- SetLowStop, 182
- SetMaxForce, 183
- SetVelocity, 183
- gazebo::physics::BulletHingeJoint, 183
 - ~BulletHingeJoint, 185
 - Attach, 186
 - BulletHingeJoint, 185
 - GetAnchor, 186
 - GetAngle, 186
 - GetAngleImpl, 186
 - GetForce, 186
 - GetGlobalAxis, 186
 - GetHighStop, 186
 - GetLowStop, 186
 - GetMaxForce, 186
 - GetVelocity, 187
 - Load, 187
 - SetAnchor, 187
 - SetAxis, 187
 - SetDamping, 187
 - SetForce, 187
 - SetHighStop, 187
 - SetLowStop, 187
 - SetMaxForce, 188
 - SetVelocity, 188
- gazebo::physics::BulletJoint, 188
 - ~BulletJoint, 189
 - AreConnected, 189
 - BulletJoint, 189
 - constraint, 191
 - Detach, 189
 - GetAnchor, 190
 - GetJointLink, 190
 - GetLinkForce, 190
 - GetLinkTorque, 190
 - Load, 190
 - Reset, 191
 - SetAnchor, 191
 - SetAttribute, 191
 - SetDamping, 191
 - world, 191
- gazebo::physics::BulletLink, 192
 - ~BulletLink, 194
 - AddForce, 194
 - AddForceAtRelativePosition, 194
 - AddForceAtWorldPosition, 194
 - AddRelativeForce, 194
 - AddRelativeTorque, 195
 - AddTorque, 195
 - BulletLink, 194
 - Finis, 195
 - GetBulletLink, 195
 - GetEnabled, 195
 - GetGravityMode, 195
 - GetWorldAngularVel, 195
 - GetWorldForce, 195
 - GetWorldLinearVel, 195
 - GetWorldTorque, 196
 - Init, 196
 - Load, 196
 - OnPoseChange, 196
 - pose, 198
 - SetAngularDamping, 196
 - SetAngularVel, 196
 - SetAutoDisable, 196
 - SetEnabled, 196
 - SetForce, 197
 - SetGravityMode, 197
 - SetLinearDamping, 197
 - SetLinearVel, 197
 - SetSelfCollide, 197
 - SetTorque, 197
 - Update, 197
 - UpdateCoM, 197
- gazebo::physics::BulletMotionState, 198
 - ~BulletMotionState, 199
 - BulletMotionState, 199
 - GetWorldPose, 199
 - getWorldTransform, 199
 - SetCoG, 199
 - SetWorldPose, 199
 - SetWorldPosition, 199
 - SetWorldRotation, 199
 - setWorldTransform, 200
- gazebo::physics::BulletMultiRayShape, 200
 - ~BulletMultiRayShape, 202
 - AddRay, 202
 - BulletMultiRayShape, 202
 - UpdateRays, 202
- gazebo::physics::BulletPhysics, 202
 - ~BulletPhysics, 204
 - BulletPhysics, 204
 - ConvertMass, 204
 - ConvertPose, 204, 205
 - CreateCollision, 205
 - CreateJoint, 205

- CreateLink, 205
- CreateShape, 205
- DebugPrint, 205
- Fini, 205
- GetDynamicsWorld, 205
- GetStepTime, 206
- Init, 206
- InitForThread, 206
- Load, 206
- SetGravity, 206
- SetStepTime, 206
- UpdateCollision, 206
- UpdatePhysics, 206
- gazebo::physics::BulletPlaneShape, 207
 - ~BulletPlaneShape, 208
 - BulletPlaneShape, 208
 - CreatePlane, 208
 - SetAltitude, 208
- gazebo::physics::BulletRaySensor, 208
 - ~BulletRaySensor, 209
 - AddRay, 210
 - BulletRaySensor, 209
 - GetCount, 210
 - GetFiducial, 210
 - GetRange, 210
 - GetRelativePoints, 210
 - GetRetro, 210
 - Update, 210
- gazebo::physics::BulletRayShape, 210
 - ~BulletRayShape, 212
 - BulletRayShape, 212
 - GetIntersection, 212
 - SetPoints, 212
 - Update, 212
- gazebo::physics::BulletScrewJoint, 213
 - ~BulletScrewJoint, 214
 - Attach, 215
 - BulletScrewJoint, 214
 - GetAngle, 215
 - GetAngleImpl, 215
 - GetGlobalAxis, 215
 - GetHighStop, 215
 - GetLowStop, 215
 - GetMaxForce, 215
 - GetVelocity, 215
 - Load, 215
 - SetAxis, 216
 - SetDamping, 216
 - SetForce, 216
 - SetHighStop, 216
 - SetLowStop, 216
 - SetMaxForce, 216
 - SetVelocity, 216
- gazebo::physics::BulletSliderJoint, 217
 - ~BulletSliderJoint, 218
 - Attach, 219
 - BulletSliderJoint, 218
 - GetAngle, 219
 - GetAngleImpl, 219
 - GetGlobalAxis, 219
 - GetHighStop, 219
 - GetLowStop, 219
 - GetMaxForce, 219
 - GetVelocity, 219
 - Load, 219
 - SetAxis, 220
 - SetDamping, 220
 - SetForce, 220
 - SetHighStop, 220
 - SetLowStop, 220
 - SetMaxForce, 220
 - SetVelocity, 220
- gazebo::physics::BulletSphereShape, 221
 - ~BulletSphereShape, 222
 - BulletSphereShape, 222
 - SetRadius, 222
- gazebo::physics::BulletTrimeshShape, 224
 - ~BulletTrimeshShape, 224
 - BulletTrimeshShape, 224
 - Init, 224
 - Load, 224
- gazebo::physics::BulletUniversalJoint, 224
 - ~BulletUniversalJoint, 226
 - Attach, 227
 - BulletUniversalJoint, 226
 - GetAnchor, 227
 - GetAngle, 227
 - GetAngleImpl, 227
 - GetAxis, 227
 - GetGlobalAxis, 227
 - GetHighStop, 227
 - GetLowStop, 227
 - GetMaxForce, 227
 - GetVelocity, 227
 - SetAnchor, 228
 - SetAxis, 228
 - SetDamping, 228
 - SetForce, 228
 - SetHighStop, 228
 - SetLowStop, 228
 - SetMaxForce, 228
 - SetVelocity, 228
- gazebo::physics::Collision, 262
 - ~Collision, 265
 - AddContact, 265
 - Collision, 265
 - ConnectContact, 265
 - DisconnectContact, 266

- FillCollisionMsg, 266
- FillMsg, 266
- Fini, 266
- GetBoundingBox, 266
- GetContactsEnabled, 267
- GetLaserRetro, 267
- GetLink, 267
- GetModel, 267
- GetRelativeAngularAccel, 267
- GetRelativeAngularVel, 267
- GetRelativeLinearAccel, 268
- GetRelativeLinearVel, 268
- GetShape, 268
- GetShapeType, 268
- GetState, 268
- GetSurface, 269
- GetWorldAngularAccel, 269
- GetWorldAngularVel, 269
- GetWorldLinearAccel, 269
- GetWorldLinearVel, 269
- Init, 269
- IsPlaceable, 270
- link, 272
- Load, 270
- placeable, 272
- ProcessMsg, 270
- SetCategoryBits, 270
- SetCollideBits, 270
- SetCollision, 271
- SetContactsEnabled, 271
- SetLaserRetro, 271
- SetShape, 271
- SetState, 271
- shape, 272
- UpdateParameters, 271
- gazebo::physics::CollisionState, 272
 - ~CollisionState, 273
 - CollisionState, 273
 - GetPose, 273
 - Load, 273
- gazebo::physics::Contact, 296
 - ~Contact, 298
 - Clone, 298
 - collision1, 298
 - collision2, 298
 - Contact, 297
 - count, 298
 - depths, 299
 - forces, 299
 - normals, 299
 - operator=, 298
 - positions, 299
 - Reset, 298
 - time, 299
- gazebo::physics::ContactFeedback, 299
 - contact, 300
 - feedbackCount, 300
 - feedbacks, 300
- gazebo::physics::CylinderShape, 307
 - ~CylinderShape, 309
 - CylinderShape, 309
 - FillMsg, 309
 - GetInertial, 309
 - GetLength, 309
 - GetMass, 310
 - GetRadius, 310
 - Init, 310
 - ProcessMsg, 310
 - SetLength, 310
 - SetRadius, 310
 - SetSize, 311
- gazebo::physics::Entity, 338
 - ~Entity, 341
 - animation, 348
 - animationConnection, 348
 - animationStartPose, 349
 - connections, 349
 - dirtyPose, 349
 - Entity, 341
 - Fini, 342
 - GetBoundingBox, 342
 - GetChildCollision, 342
 - GetChildLink, 342
 - GetCollisionBoundingBox, 342
 - GetDirtyPose, 343
 - GetNearestEntityBelow, 343
 - GetParentModel, 343
 - GetRelativeAngularAccel, 343
 - GetRelativeAngularVel, 343
 - GetRelativeLinearAccel, 343
 - GetRelativeLinearVel, 344
 - GetRelativePose, 344
 - GetWorldAngularAccel, 344
 - GetWorldAngularVel, 344
 - GetWorldLinearAccel, 344
 - GetWorldLinearVel, 345
 - GetWorldPose, 345
 - IsCanonicalLink, 345
 - IsStatic, 345
 - Load, 345
 - node, 349
 - OnPoseChange, 346
 - parentEntity, 349
 - PlaceOnEntity, 346
 - PlaceOnNearestEntityBelow, 346
 - poseMsg, 349
 - prevAnimationTime, 349
 - requestPub, 349

- Reset, 346
- SetAnimation, 346
- SetCanonicalLink, 347
- SetInitialRelativePose, 347
- SetName, 347
- SetRelativePose, 347
- SetStatic, 347
- SetWorldPose, 348
- SetWorldTwist, 348
- StopAnimation, 348
- UpdateParameters, 348
- visPub, 349
- visualMsg, 349
- gazebo::physics::Gripper, 393
 - ~Gripper, 393
 - Gripper, 393
 - Init, 393
 - Load, 393
- gazebo::physics::HeightmapShape, 399
 - ~HeightmapShape, 401
 - FillMsg, 401
 - GetHeight, 402
 - GetMaxHeight, 402
 - GetMinHeight, 402
 - GetPos, 402
 - GetSize, 402
 - GetURI, 402
 - GetVertexCount, 403
 - HeightmapShape, 401
 - heights, 403
 - img, 403
 - Init, 403
 - Load, 403
 - ProcessMsg, 403
 - scale, 404
 - subSampling, 404
 - vertSize, 404
- gazebo::physics::Hinge2Joint
 - ~Hinge2Joint, 405
 - Hinge2Joint, 405
 - Load, 405
- gazebo::physics::Hinge2Joint< T >, 404
- gazebo::physics::HingeJoint
 - ~HingeJoint, 406
 - HingeJoint, 406
 - Init, 407
 - Load, 407
- gazebo::physics::HingeJoint< T >, 405
- gazebo::physics::Inertial, 414
 - ~Inertial, 417
 - GetCoG, 417
 - GetIXX, 417
 - GetIXY, 417
 - GetIXZ, 417
 - GetIYY, 417
 - GetIYZ, 418
 - GetIZZ, 418
 - GetMass, 418
 - GetPose, 418
 - GetPrincipalMoments, 418
 - GetProductsofInertia, 418
 - Inertial, 416
 - Load, 418
 - operator<<, 422
 - operator+, 419
 - operator+=", 419
 - operator=, 419
 - ProcessMsg, 419
 - Reset, 420
 - Rotate, 420
 - SetCoG, 420
 - SetIXX, 420
 - SetIXY, 421
 - SetIXZ, 421
 - SetIYY, 421
 - SetIYZ, 421
 - SetIZZ, 421
 - SetInertiaMatrix, 420
 - SetMass, 421
 - UpdateParameters, 422
- gazebo::physics::Joint, 423
 - ~Joint, 427
 - anchorLink, 438
 - anchorPos, 438
 - AreConnected, 427
 - Attach, 427
 - Attribute, 426
 - CFM, 426
 - childLink, 438
 - ConnectJointUpdate, 427
 - damping_coefficient, 438
 - Detach, 428
 - DisconnectJointUpdate, 428
 - ERP, 426
 - FMAX, 426
 - FUDGE_FACTOR, 426
 - FillJointMsg, 428
 - FillMsg, 428
 - GetAnchor, 428
 - GetAngle, 429
 - GetAngleImpl, 429
 - GetChild, 429
 - GetForce, 429
 - GetGlobalAxis, 430
 - GetHighStop, 430
 - GetJointLink, 430
 - GetLinkForce, 431
 - GetLinkTorque, 431

- GetLocalAxis, 431
- GetLowStop, 432
- GetMaxForce, 432
- GetParent, 432
- GetState, 432
- GetVelocity, 433
- HI_STOP, 426
- Init, 433
- Joint, 427
- LO_STOP, 426
- Load, 433
- model, 438
- parentLink, 438
- Reset, 434
- STOP_CFM, 426
- STOP_ERP, 426
- SUSPENSION_CFM, 426
- SUSPENSION_ERP, 426
- SetAnchor, 434
- SetAngle, 434
- SetAttribute, 434
- SetAxis, 435
- SetDamping, 435
- SetForce, 435
- SetHighStop, 436
- SetLowStop, 436
- SetMaxForce, 436
- SetModel, 437
- SetState, 437
- SetVelocity, 437
- Update, 437
- UpdateParameters, 437
- VEL, 426
- gazebo::physics::JointController, 438
 - AddJoint, 439
 - JointController, 439
 - Reset, 439
 - SetJointPosition, 439
 - SetJointPositions, 440
 - Update, 440
- gazebo::physics::JointFeedback, 440
 - body1Force, 441
 - body1Torque, 441
 - body2Force, 441
 - body2Torque, 441
 - operator=, 441
- gazebo::physics::JointState, 442
 - ~JointState, 443
 - GetAngle, 443
 - GetAngleCount, 443
 - JointState, 442
 - Load, 443
- gazebo::physics::Link, 454
 - ~Link, 459
 - AddChildJoint, 459
 - AddForce, 459
 - AddForceAtRelativePosition, 459
 - AddForceAtWorldPosition, 460
 - AddParentJoint, 460
 - AddRelativeForce, 460
 - AddRelativeTorque, 460
 - AddTorque, 460
 - angularAccel, 473
 - AttachStaticModel, 461
 - attachedModelsOffset, 473
 - cgVisuals, 473
 - ConnectEnabled, 461
 - DetachAllStaticModels, 461
 - DetachStaticModel, 461
 - DisconnectEnabled, 461
 - FillLinkMsg, 462
 - FillMsg, 462
 - Fini, 462
 - GetAngularDamping, 462
 - GetBoundingBox, 462
 - GetChildJointsLinks, 462
 - GetCollision, 462, 463
 - GetCollisionById, 463
 - GetEnabled, 463
 - GetGravityMode, 463
 - GetInertial, 464
 - GetKinematic, 464
 - GetLinearDamping, 464
 - GetModel, 464
 - GetParentJointsLinks, 464
 - GetRelativeAngularAccel, 464
 - GetRelativeAngularVel, 465
 - GetRelativeForce, 465
 - GetRelativeLinearAccel, 465
 - GetRelativeLinearVel, 465
 - GetRelativeTorque, 465
 - GetSelfCollide, 466
 - GetSensorCount, 466
 - GetSensorName, 466
 - GetState, 466
 - GetWorldAngularAccel, 467
 - GetWorldForce, 467
 - GetWorldLinearAccel, 467
 - GetWorldTorque, 467
 - inertial, 473
 - Init, 467
 - linearAccel, 473
 - Link, 459
 - Load, 467
 - OnPoseChange, 468
 - ProcessMsg, 468
 - RemoveChildJoint, 468
 - RemoveParentJoint, 468

- Reset, 468
- SetAngularAccel, 468
- SetAngularDamping, 469
- SetAngularVel, 469
- SetAutoDisable, 469
- SetCollideMode, 469
- SetEnabled, 469
- SetForce, 470
- SetGravityMode, 470
- SetInertial, 470
- SetKinematic, 470
- SetLaserRetro, 470
- SetLinearAccel, 471
- SetLinearDamping, 471
- SetLinearVel, 471
- SetSelected, 471
- SetSelfCollide, 471
- SetState, 472
- SetTorque, 472
- Update, 472
- UpdateMass, 472
- UpdateParameters, 472
- UpdateSurface, 472
- visuals, 473
- gazebo::physics::LinkState, 473
 - ~LinkState, 475
 - FillStateSDF, 475
 - GetAcceleration, 475
 - GetCollisionState, 475, 476
 - GetCollisionStateCount, 476
 - GetForces, 476
 - GetPose, 476
 - GetVelocity, 477
 - LinkState, 475
 - Load, 477
 - UpdateLinkSDF, 477
- gazebo::physics::Logger, 477
 - Add, 478
 - Remove, 479
- gazebo::physics::Logplay, 479
 - Begin, 480
 - End, 480
 - Open, 480
 - Play, 481–483
- gazebo::physics::MapShape, 483
 - ~MapShape, 485
 - FillMsg, 485
 - GetGranularity, 485
 - GetHeight, 485
 - GetScale, 486
 - GetThreshold, 486
 - GetURI, 486
 - Init, 486
 - Load, 486
 - MapShape, 485
 - ProcessMsg, 487
 - Update, 487
- gazebo::physics::Model, 521
 - ~Model, 524
 - AttachStaticModel, 524
 - attachedModels, 533
 - attachedModelsOffset, 533
 - DetachStaticModel, 524
 - FillModelMsg, 525
 - FillMsg, 525
 - Fin, 525
 - GetAllLinks, 525
 - GetBoundingBox, 525
 - GetJoint, 525, 526
 - GetJointCount, 526
 - GetLink, 526
 - GetLinkById, 527
 - GetRelativeAngularAccel, 527
 - GetRelativeAngularVel, 527
 - GetRelativeLinearAccel, 527
 - GetRelativeLinearVel, 527
 - GetSDF, 527
 - GetState, 528
 - GetWorldAngularAccel, 528
 - GetWorldAngularVel, 528
 - GetWorldLinearAccel, 528
 - GetWorldLinearVel, 528
 - Init, 529
 - Load, 529
 - LoadPlugins, 529
 - Model, 524
 - OnPoseChange, 529
 - ProcessMsg, 529
 - RemoveChild, 529
 - Reset, 530
 - SetAngularAccel, 530
 - SetAngularVel, 530
 - SetAutoDisable, 530
 - SetCollideMode, 530
 - SetEnabled, 530
 - SetGravityMode, 531
 - SetJointAnimation, 531
 - SetJointPosition, 531
 - SetJointPositions, 531
 - SetLaserRetro, 531
 - SetLinearAccel, 532
 - SetLinearVel, 532
 - SetLinkWorldPose, 532
 - SetState, 532
 - StopAnimation, 533
 - Update, 533
 - UpdateParameters, 533
- gazebo::physics::ModelState, 536

- ~ModelState, 537
- FillStateSDF, 537
- GetJointState, 538
- GetJointStateCount, 538
- GetLinkState, 538, 539
- GetLinkStateCount, 539
- GetPose, 539
- Load, 539
- ModelState, 537
- UpdateModelSDF, 539
- gazebo::physics::MultiRayShape, 549
 - ~MultiRayShape, 551
 - AddRay, 551
 - ConnectNewLaserScans, 552
 - DisconnectNewLaserScans, 552
 - FillMsg, 552
 - GetFiducial, 552
 - GetMaxAngle, 553
 - GetMaxRange, 553
 - GetMinAngle, 553
 - GetMinRange, 553
 - GetRange, 553
 - GetResRange, 553
 - GetRetro, 554
 - GetSampleCount, 554
 - GetScanResolution, 554
 - GetVerticalMaxAngle, 554
 - GetVerticalMinAngle, 554
 - GetVerticalSampleCount, 554
 - GetVerticalScanResolution, 555
 - horzElem, 555
 - Init, 555
 - MultiRayShape, 551
 - newLaserScans, 555
 - offset, 556
 - ProcessMsg, 555
 - rangeElem, 556
 - rayElem, 556
 - rays, 556
 - scanElem, 556
 - Update, 555
 - UpdateRays, 555
 - vertElem, 556
- gazebo::physics::ODEBallJoint, 573
 - ~ODEBallJoint, 575
 - GetAnchor, 575
 - GetAngleImpl, 575
 - GetGlobalAxis, 575
 - GetMaxForce, 575
 - GetVelocity, 576
 - ODEBallJoint, 575
 - SetAnchor, 576
 - SetDamping, 576
 - SetMaxForce, 576
 - SetVelocity, 576
- gazebo::physics::ODEBoxShape, 576
 - ~ODEBoxShape, 577
 - ODEBoxShape, 577
 - SetSize, 578
- gazebo::physics::ODECollision, 578
 - ~ODECollision, 580
 - collisionId, 582
 - FinI, 580
 - GetBoundingBox, 580
 - GetCollisionClass, 580
 - GetCollisionId, 581
 - GetSpaceId, 581
 - Load, 581
 - ODECollision, 580
 - OnPoseChange, 581
 - SetCategoryBits, 581
 - SetCollideBits, 581
 - SetCollision, 582
 - SetSpaceId, 582
 - spaceId, 582
- gazebo::physics::ODECylinderShape, 582
 - ~ODECylinderShape, 584
 - ODECylinderShape, 584
 - SetSize, 584
- gazebo::physics::ODEHeightmapShape, 584
 - ~ODEHeightmapShape, 586
 - Init, 586
 - ODEHeightmapShape, 586
- gazebo::physics::ODEHinge2Joint, 586
 - ~ODEHinge2Joint, 588
 - GetAnchor, 589
 - GetAngleImpl, 589
 - GetGlobalAxis, 589
 - GetMaxForce, 589
 - GetParam, 589
 - GetVelocity, 589
 - Load, 589
 - ODEHinge2Joint, 588
 - SetAnchor, 589
 - SetAxis, 589
 - SetDamping, 590
 - SetForce, 590
 - SetMaxForce, 590
 - SetParam, 590
 - SetVelocity, 590
- gazebo::physics::ODEHingeJoint, 590
 - ~ODEHingeJoint, 592
 - ApplyDamping, 593
 - GetAnchor, 593
 - GetAngleImpl, 593
 - GetGlobalAxis, 593
 - GetMaxForce, 593
 - GetParam, 593

- GetVelocity, 593
- Load, 593
- ODEHingeJoint, 592
- SetAnchor, 593
- SetAxis, 594
- SetDamping, 594
- SetForce, 594
- SetMaxForce, 594
- SetParam, 594
- SetVelocity, 594
- gazebo::physics::ODEJoint, 594
 - ~ODEJoint, 596
 - AreConnected, 596
 - Attach, 596
 - Detach, 596
 - GetCFM, 597
 - GetERP, 597
 - GetFeedback, 597
 - GetHighStop, 597
 - GetJointLink, 597
 - GetLinkForce, 597
 - GetLinkTorque, 597
 - GetLowStop, 597
 - GetParam, 598
 - jointId, 599
 - Load, 598
 - ODEJoint, 596
 - Reset, 598
 - SetAttribute, 598
 - SetCFM, 598
 - SetERP, 598
 - SetHighStop, 598
 - SetLowStop, 598
 - SetParam, 599
- gazebo::physics::ODELink, 599
 - ~ODELink, 602
 - AddForce, 602
 - AddForceAtRelativePosition, 602
 - AddForceAtWorldPosition, 603
 - AddRelativeForce, 603
 - AddRelativeTorque, 603
 - AddTorque, 603
 - DisabledCallback, 603
 - Fini, 603
 - GetEnabled, 603
 - GetGravityMode, 603
 - GetKinematic, 604
 - GetODEId, 604
 - GetSpaceld, 604
 - GetWorldAngularVel, 604
 - GetWorldForce, 604
 - GetWorldLinearVel, 604
 - GetWorldTorque, 604
 - Init, 604
 - Load, 604
 - MoveCallback, 605
 - ODELink, 602
 - OnPoseChange, 605
 - pose, 607
 - SetAngularDamping, 605
 - SetAngularVel, 605
 - SetAutoDisable, 605
 - SetEnabled, 605
 - SetForce, 605
 - SetGravityMode, 606
 - SetKinematic, 606
 - SetLinearDamping, 606
 - SetLinearVel, 606
 - SetSelfCollide, 606
 - SetSpaceld, 606
 - SetTorque, 606
 - Update, 606
 - UpdateMass, 606
 - UpdateSurface, 607
- gazebo::physics::ODEMultiRayShape, 607
 - ~ODEMultiRayShape, 609
 - AddRay, 609
 - ODEMultiRayShape, 609
 - UpdateRays, 609
- gazebo::physics::ODEPhysics, 609
 - ~ODEPhysics, 612
 - Collide, 612
 - ConvertMass, 612, 613
 - CreateCollision, 613
 - CreateContact, 613
 - CreateJoint, 613
 - CreateLink, 613
 - CreateShape, 613
 - DebugPrint, 613
 - Fini, 613
 - GetContactMaxCorrectingVel, 613
 - GetContactSurfaceLayer, 614
 - GetGravity, 614
 - GetMaxContacts, 614
 - GetSORPGSIters, 614
 - GetSORPGSPreconlters, 614
 - GetSORPGSW, 614
 - GetSpaceld, 614
 - GetStepTime, 614
 - GetStepType, 614
 - GetWorldCFM, 615
 - GetWorldERP, 615
 - GetWorldId, 615
 - Init, 615
 - InitForThread, 615
 - Load, 615
 - ODEPhysics, 612
 - OnPhysicsMsg, 615

- OnRequest, 615
- ProcessContactFeedback, 615
- Reset, 616
- SetContactMaxCorrectingVel, 616
- SetContactSurfaceLayer, 616
- SetGravity, 616
- SetMaxContacts, 616
- SetSORPGSIters, 616
- SetSORPGSPreconIters, 616
- SetSORPGSW, 616
- SetStepTime, 616
- SetStepType, 617
- SetWorldCFM, 617
- SetWorldERP, 617
- UpdateCollision, 617
- UpdatePhysics, 617
- gazebo::physics::ODEPlaneShape, 617
 - ~ODEPlaneShape, 619
 - CreatePlane, 619
 - ODEPlaneShape, 619
 - SetAltitude, 619
- gazebo::physics::ODERayShape, 619
 - ~ODERayShape, 621
 - GetIntersection, 621
 - ODERayShape, 621
 - SetPoints, 621
 - Update, 621
- gazebo::physics::ODEScrewJoint, 622
 - ~ODEScrewJoint, 623
 - ApplyDamping, 624
 - GetAngleImpl, 624
 - GetGlobalAxis, 624
 - GetMaxForce, 624
 - GetParam, 624
 - GetVelocity, 624
 - Load, 624
 - ODEScrewJoint, 623
 - SetAxis, 624
 - SetDamping, 624
 - SetForce, 624
 - SetMaxForce, 625
 - SetParam, 625
 - SetThreadPitch, 625
 - SetVelocity, 625
- gazebo::physics::ODESliderJoint, 625
 - ~ODESliderJoint, 627
 - ApplyDamping, 627
 - GetAngleImpl, 627
 - GetGlobalAxis, 628
 - GetMaxForce, 628
 - GetParam, 628
 - GetVelocity, 628
 - Load, 628
 - ODESliderJoint, 627
- SetAxis, 628
- SetDamping, 628
- SetForce, 628
- SetMaxForce, 628
- SetParam, 629
- SetVelocity, 629
- gazebo::physics::ODESphereShape, 629
 - ~ODESphereShape, 631
 - ODESphereShape, 630
 - SetRadius, 631
- gazebo::physics::ODESurfaceParams, 631
 - ~ODESurfaceParams, 631
 - Load, 632
 - ODESurfaceParams, 631
- gazebo::physics::ODETrimeshShape, 632
 - ~ODETrimeshShape, 634
 - Init, 634
 - Load, 634
 - ODETrimeshShape, 634
 - Update, 634
- gazebo::physics::ODEUniversalJoint, 634
 - ~ODEUniversalJoint, 636
 - GetAnchor, 636
 - GetAngleImpl, 636
 - GetGlobalAxis, 636
 - GetMaxForce, 637
 - GetVelocity, 637
 - ODEUniversalJoint, 636
 - SetAnchor, 637
 - SetAxis, 637
 - SetDamping, 637
 - SetForce, 637
 - SetMaxForce, 637
 - SetParam, 637
 - SetVelocity, 638
- gazebo::physics::PhysicsEngine, 651
 - ~PhysicsEngine, 654
 - contactPub, 662
 - CreateCollision, 654, 655
 - CreateJoint, 655
 - CreateLink, 655
 - CreateShape, 655
 - DebugPrint, 655
 - Fini, 655
 - GetAutoDisableFlag, 656
 - GetContactMaxCorrectingVel, 656
 - GetContactSurfaceLayer, 656
 - GetGravity, 656
 - GetMaxContacts, 656
 - GetPhysicsUpdateMutex, 657
 - GetSORPGSIters, 657
 - GetSORPGSPreconIters, 657
 - GetSORPGSW, 657
 - GetStepTime, 657

- GetUpdatePeriod, 658
- GetUpdateRate, 658
- GetWorldCFM, 658
- GetWorldERP, 658
- Init, 658
- InitForThread, 659
- Load, 659
- node, 662
- OnPhysicsMsg, 659
- OnRequest, 659
- PhysicsEngine, 654
- physicsSub, 662
- physicsUpdateMutex, 663
- requestSub, 663
- Reset, 659
- responsePub, 663
- sdf, 663
- SetAutoDisableFlag, 659
- SetContactMaxCorrectingVel, 660
- SetContactSurfaceLayer, 660
- SetGravity, 660
- SetMaxContacts, 660
- SetSORPGSIters, 661
- SetSORPGSPreconIters, 661
- SetSORPGSW, 661
- SetStepTime, 661
- SetUpdateRate, 661
- SetWorldCFM, 662
- SetWorldERP, 662
- UpdateCollision, 662
- UpdatePhysics, 662
- world, 663
- gazebo::physics::PhysicsFactory, 663
 - NewPhysicsEngine, 664
 - RegisterAll, 664
 - RegisterPhysicsEngine, 664
- gazebo::physics::PlaneShape, 671
 - ~PlaneShape, 672
 - CreatePlane, 672
 - FillMsg, 673
 - GetNormal, 673
 - GetSize, 673
 - Init, 673
 - PlaneShape, 672
 - ProcessMsg, 673
 - SetAltitude, 673
 - SetNormal, 673
 - SetSize, 673
- gazebo::physics::RayShape, 718
 - ~RayShape, 720
 - contactFiducial, 722
 - contactLen, 722
 - contactRetro, 722
 - FillMsg, 720
 - GetFiducial, 720
 - GetGlobalPoints, 720
 - GetIntersection, 720
 - GetLength, 720
 - GetRelativePoints, 721
 - GetRetro, 721
 - globalEndPos, 722
 - globalStartPos, 722
 - Init, 721
 - ProcessMsg, 721
 - RayShape, 720
 - relativeEndPos, 722
 - relativeStartPos, 722
 - SetFiducial, 721
 - SetLength, 721
 - SetPoints, 721
 - SetRetro, 722
 - Update, 722
- gazebo::physics::Road, 736
 - ~Road, 737
 - Init, 737
 - Load, 737
 - Road, 737
- gazebo::physics::ScrewJoint
 - ~ScrewJoint, 761
 - fakeAnchor, 762
 - GetAnchor, 761
 - Load, 761
 - ScrewJoint, 761
 - SetAnchor, 761
 - SetThreadPitch, 761
 - threadPitch, 762
- gazebo::physics::ScrewJoint< T >, 760
- gazebo::physics::Shape, 779
 - ~Shape, 781
 - collisionParent, 782
 - FillMsg, 781
 - FillShapeMsg, 781
 - GetInertial, 781
 - GetMass, 781
 - Init, 781
 - ProcessMsg, 782
 - Shape, 781
- gazebo::physics::SliderJoint
 - ~SliderJoint, 805
 - fakeAnchor, 806
 - GetAnchor, 805
 - Load, 805
 - SetAnchor, 805
 - SliderJoint, 805
- gazebo::physics::SliderJoint< T >, 804
- gazebo::physics::SphereShape, 806
 - ~SphereShape, 807
 - FillMsg, 807

- GetInertial, 807
- GetMass, 807
- GetRadius, 808
- Init, 808
- ProcessMsg, 808
- SetRadius, 808
- SphereShape, 807
- gazebo::physics::State, 813
 - ~State, 814
 - GetName, 814
 - GetRealTime, 814
 - GetSimTime, 815
 - GetWallTime, 815
 - Load, 815
 - name, 815
 - realTime, 815
 - simTime, 815
 - State, 814
 - wallTime, 815
- gazebo::physics::SurfaceParams, 830
 - ~SurfaceParams, 831
 - bounce, 831
 - bounceThreshold, 831
 - cfm, 832
 - erp, 832
 - fdir1, 832
 - FillSurfaceMsg, 831
 - kd, 832
 - kp, 832
 - Load, 831
 - maxVel, 832
 - minDepth, 833
 - mu1, 833
 - mu2, 833
 - ProcessMsg, 831
 - slip1, 833
 - slip2, 833
 - SurfaceParams, 831
- gazebo::physics::TrajectoryInfo, 865
 - duration, 865
 - endTime, 865
 - id, 865
 - startTime, 866
 - translated, 866
 - type, 866
- gazebo::physics::TrimeshShape, 866
 - ~TrimeshShape, 867
 - FillMsg, 867
 - GetFilename, 867
 - GetMass, 867
 - GetSize, 868
 - Init, 868
 - mesh, 868
 - ProcessMsg, 868
 - SetFilename, 868
 - SetScale, 868
 - TrimeshShape, 867
 - Update, 868
- gazebo::physics::UniversalJoint
 - ~UniversalJoint, 869
 - Load, 870
 - UniversalJoint, 869
- gazebo::physics::UniversalJoint< T >, 868
- gazebo::physics::World, 954
 - ~World, 957
 - Clear, 957
 - dirtyPoses, 965
 - DisableAllModels, 957
 - EnableAllModels, 957
 - EnablePhysicsEngine, 957
 - Fini, 957
 - GetByName, 957
 - GetEnablePhysicsEngine, 958
 - GetEntity, 958
 - GetEntityBelowPoint, 958
 - GetEntityByName, 958
 - GetModel, 959
 - GetModelBelowPoint, 959
 - GetModelByName, 959
 - GetModelCount, 960
 - GetModels, 960
 - GetName, 960
 - GetPauseTime, 960
 - GetPhysicsEngine, 960
 - GetRealTime, 960
 - GetSelectedEntity, 961
 - GetSetWorldPoseMutex, 961
 - GetSimTime, 961
 - GetStartTime, 961
 - GetState, 961
 - Init, 961
 - InsertModelFile, 962
 - InsertModelSDF, 962
 - InsertModelString, 962
 - IsPaused, 962
 - Load, 962
 - LoadPlugin, 963
 - PrintEntityTree, 963
 - RemovePlugin, 963
 - Reset, 963
 - ResetEntities, 963
 - ResetTime, 963
 - Run, 963
 - Save, 964
 - SetPaused, 964
 - SetSimTime, 964
 - SetState, 964
 - StepWorld, 964

- Stop, 964
- StripWorldName, 965
- World, 957
- gazebo::physics::WorldState, 967
 - ~WorldState, 968
 - GetModelState, 968
 - GetModelStateCount, 968
 - GetSDF, 969
 - Load, 969
 - WorldState, 968
- gazebo::rendering, 109
 - ArrowVisualPtr, 112
 - AxisVisualPtr, 112
 - COMVisualPtr, 112
 - CameraPtr, 112
 - CameraVisualPtr, 112
 - ContactVisualPtr, 112
 - DepthCameraPtr, 112
 - DynamicLinesPtr, 112
 - GpuLaserPtr, 112
 - JointVisualPtr, 112
 - LaserVisualPtr, 112
 - LightPtr, 112
 - RENDERING_LINE_LIST, 113
 - RENDERING_LINE_STRIP, 113
 - RENDERING_MESH_RESOURCE, 113
 - RENDERING_POINT_LIST, 113
 - RENDERING_TRIANGLE_FAN, 113
 - RENDERING_TRIANGLE_LIST, 113
 - RENDERING_TRIANGLE_STRIP, 113
 - RFIDTagVisualPtr, 112
 - RFIDVisualPtr, 112
 - RenderOpType, 113
 - ScenePtr, 112
 - UserCameraPtr, 113
 - VisualPtr, 113
- gazebo::rendering::ArrowVisual, 139
 - ~ArrowVisual, 140
 - ArrowVisual, 140
 - Load, 140
 - ShowRotation, 140
- gazebo::rendering::AxisVisual, 140
 - ~AxisVisual, 142
 - AxisVisual, 142
 - Load, 142
 - ScaleXAxis, 142
 - ScaleYAxis, 142
 - ScaleZAxis, 142
 - SetAxisMaterial, 142
 - ShowRotation, 143
- gazebo::rendering::COMVisual, 285
 - ~COMVisual, 286
 - COMVisual, 286
 - Load, 286
- gazebo::rendering::Camera, 233
 - ~Camera, 238
 - animState, 254
 - AttachToVisual, 238
 - AttachToVisualImpl, 239
 - bayerFrameBuffer, 254
 - Camera, 238
 - camera, 254
 - captureData, 254
 - ConnectNewImageFrame, 239
 - connections, 254
 - CreateRenderTexture, 240
 - DisconnectNewImageFrame, 240
 - EnableSaveFrame, 240
 - Fini, 240
 - GetAspectRatio, 240
 - GetAvgFPS, 240
 - GetCameraToViewportRay, 241
 - GetDirection, 241
 - GetFarClip, 241
 - GetFrameFilename, 241
 - GetHFOV, 241
 - GetImageByteSize, 241, 242
 - GetImageData, 242
 - GetImageDepth, 242
 - GetImageFormat, 242
 - GetImageHeight, 243
 - GetImageWidth, 243
 - GetInitialized, 243
 - GetLastRenderWallTime, 243
 - GetName, 243
 - GetNearClip, 243
 - GetOgreCamera, 243
 - GetRenderTexture, 244
 - GetRight, 244
 - GetScene, 244
 - GetSceneNode, 244
 - GetTextureHeight, 244
 - GetTextureWidth, 244
 - GetTriangleCount, 245
 - GetUp, 245
 - GetVFOV, 245
 - GetViewport, 245
 - GetViewportHeight, 245
 - GetViewportWidth, 245
 - GetWindowId, 246
 - GetWorldPointOnPlane, 246
 - GetWorldPose, 246
 - GetWorldPosition, 246
 - GetWorldRotation, 246
 - GetZValue, 247
 - imageFormat, 254
 - imageHeight, 254
 - imageWidth, 254

- Init, 247
- initialized, 254
- IsInitialized, 247
- IsVisible, 247
- lastRenderWallTime, 254
- Load, 248
- MoveToPosition, 248
- MoveToPositions, 248
- name, 254
- newData, 254
- newImageFrame, 254
- onAnimationComplete, 254
- pitchNode, 254
- PostRender, 248
- prevAnimTime, 254
- Render, 249
- RenderImpl, 249
- renderTarget, 255
- renderTexture, 255
- requests, 255
- RotatePitch, 249
- RotateYaw, 249
- saveCount, 255
- SaveFrame, 249
- saveFrameBuffer, 255
- scene, 255
- sceneNode, 255
- sdf, 255
- SetAspectRatio, 250
- SetCaptureData, 250
- SetClipDist, 250
- SetHFOV, 250
- SetImageHeight, 250
- SetImageSize, 251
- SetImageWidth, 251
- SetName, 251
- SetRenderRate, 251
- SetRenderTarget, 251
- SetSaveFramePathname, 251
- SetScene, 252
- SetSceneNode, 252
- SetWindowId, 252
- SetWorldPose, 252
- SetWorldPosition, 252
- SetWorldRotation, 252
- ShowWireframe, 253
- textureHeight, 255
- textureWidth, 255
- ToggleShowWireframe, 253
- TrackVisual, 253
- TrackVisualImpl, 253
- Translate, 253
- Update, 254
- viewport, 255
- windowId, 255
- gazebo::rendering::CameraVisual, 259
 - ~CameraVisual, 261
 - CameraVisual, 260
 - Load, 261
- gazebo::rendering::ContactVisual, 304
 - ~ContactVisual, 305
 - ContactVisual, 305
- gazebo::rendering::Conversions, 305
 - Convert, 306
- gazebo::rendering::DepthCamera, 312
 - ~DepthCamera, 314
 - ConnectNewDepthFrame, 314
 - ConnectNewRGBPointCloud, 315
 - CreateDepthTexture, 315
 - DepthCamera, 314
 - depthTarget, 316
 - depthTexture, 317
 - depthViewport, 317
 - DisconnectNewDepthFrame, 315
 - DisconnectNewRGBPointCloud, 315
 - Fini, 316
 - GetDepthData, 316
 - Init, 316
 - Load, 316
 - PostRender, 316
 - SetDepthTarget, 316
- gazebo::rendering::DynamicLines, 324
 - ~DynamicLines, 326
 - AddPoint, 326
 - Clear, 326
 - CreateVertexDeclaration, 326
 - DynamicLines, 326
 - FillHardwareBuffers, 326
 - GetMovableType, 327
 - getMovableType, 327
 - GetPoint, 327
 - GetPointCount, 327
 - SetPoint, 327
 - Update, 328
- gazebo::rendering::DynamicRenderable, 328
 - ~DynamicRenderable, 329
 - CreateVertexDeclaration, 329
 - DynamicRenderable, 329
 - FillHardwareBuffers, 329
 - getBoundingRadius, 330
 - GetOperationType, 330
 - getSquaredViewDepth, 330
 - indexBufferCapacity, 331
 - Init, 330
 - PrepareHardwareBuffers, 331
 - SetOperationType, 331
 - vertexBufferCapacity, 331
- gazebo::rendering::Events, 354

- ConnectCreateScene, 355
- ConnectRemoveScene, 355
- createScene, 356
- DisconnectCreateScene, 355
- DisconnectRemoveScene, 356
- removeScene, 356
- gazebo::rendering::FPSViewController, 367
 - ~FPSViewController, 368
 - FPSViewController, 368
 - GetTypeString, 368
 - HandleKeyPressEvent, 368
 - HandleKeyReleaseEvent, 368
 - HandleMouseEvent, 369
 - Init, 369
 - Update, 369
- gazebo::rendering::GUIOverlay, 394
 - ~GUIOverlay, 395
 - AttachCameraToImage, 395
 - ButtonCallback, 395
 - CreateWindow, 396
 - GUIOverlay, 395
 - HandleKeyPressEvent, 396
 - HandleKeyReleaseEvent, 396
 - HandleMouseEvent, 396
 - Hide, 397
 - Init, 397
 - IsInitialized, 397
 - LoadLayout, 397
 - Resize, 397
 - Show, 397
 - Update, 397
- gazebo::rendering::GpuLaser, 375
 - ~GpuLaser, 376
 - ConnectNewLaserFrame, 376
 - CreateLaserTexture, 377
 - DisconnectNewLaserFrame, 377
 - Fini, 377
 - GetLaserData, 377
 - GpuLaser, 376
 - Init, 377
 - Load, 377
 - notifyRenderSingleObject, 378
 - PostRender, 378
 - SetParentSensor, 378
 - SetRangeCount, 378
- gazebo::rendering::Grid, 389
 - ~Grid, 390
 - Enable, 390
 - GetCellCount, 390
 - GetCellLength, 390
 - GetColor, 391
 - GetHeight, 391
 - GetLineWidth, 391
 - GetSceneNode, 391
 - Grid, 390
 - Init, 391
 - SetCellCount, 391
 - SetCellLength, 392
 - SetColor, 392
 - SetHeight, 392
 - SetLineWidth, 392
 - SetUserData, 392
- gazebo::rendering::Heightmap, 398
 - ~Heightmap, 398
 - GetHeight, 398
 - Heightmap, 398
 - Load, 399
 - LoadFromMsg, 399
- gazebo::rendering::JointVisual, 444
 - ~JointVisual, 445
 - JointVisual, 444
 - Load, 445
- gazebo::rendering::LaserVisual, 447
 - ~LaserVisual, 448
 - LaserVisual, 447
 - SetEmissive, 448
- gazebo::rendering::Light, 448
 - ~Light, 450
 - FillMsg, 450
 - GetDiffuseColor, 450
 - GetDirection, 450
 - GetName, 450
 - GetPosition, 451
 - GetSpecularColor, 451
 - GetType, 451
 - Light, 450
 - Load, 451
 - LoadFromMsg, 451
 - OnPoseChange, 451
 - SetAttenuation, 452
 - SetCastShadows, 452
 - SetDiffuseColor, 452
 - SetDirection, 452
 - SetLightType, 452
 - SetName, 452
 - SetPosition, 453
 - SetRange, 453
 - SetSelected, 453
 - SetSpecularColor, 453
 - SetSpotFalloff, 453
 - SetSpotInnerAngle, 453
 - SetSpotOuterAngle, 454
 - ShowVisual, 454
 - ToggleShowVisual, 454
 - UpdateFromMsg, 454
- gazebo::rendering::MovableText, 543
 - ~MovableText, 545
 - _setupGeometry, 545

- _updateColors, 545
- GetAABB, 545
- GetBaseline, 545
- getBoundingRadius, 545
- GetCharHeight, 546
- GetColor, 546
- GetFont, 546
- getLights, 546
- getMaterial, 546
- getRenderOperation, 546
- GetShowOnTop, 546
- GetSpaceWidth, 546
- getSquaredViewDepth, 546
- GetText, 546
- getWorldTransforms, 547
- H_CENTER, 545
- H_LEFT, 545
- HorizAlign, 545
- Load, 547
- MovableText, 545
- SetBaseline, 547
- SetCharHeight, 547
- SetColor, 547
- SetFontName, 547
- SetShowOnTop, 548
- SetSpaceWidth, 548
- SetText, 548
- SetTextAlignment, 548
- Update, 548
- V_ABOVE, 545
- V_BELOW, 545
- VertAlign, 545
- visitRenderables, 548
- gazebo::rendering::OrbitViewController, 638
 - ~OrbitViewController, 639
 - GetFocalPoint, 639
 - GetTypeString, 640
 - HandleKeyPressEvent, 640
 - HandleKeyReleaseEvent, 640
 - HandleMouseEvent, 640
 - Init, 640, 641
 - OrbitViewController, 639
 - SetDistance, 641
 - SetDistanceRange, 641
 - SetFocalPoint, 641
 - SetPitch, 641
 - SetYaw, 641
 - Update, 642
- gazebo::rendering::Projector, 689
 - ~Projector, 690
 - GetParent, 690
 - Load, 690, 691
 - Projector, 690
 - SetEnabled, 691
 - SetTexture, 691
 - Toggle, 691
- gazebo::rendering::RFIDTagVisual, 733
 - ~RFIDTagVisual, 734
 - RFIDTagVisual, 734
- gazebo::rendering::RFIDVisual, 734
 - ~RFIDVisual, 735
 - RFIDVisual, 735
- gazebo::rendering::RTShaderSystem, 742
 - AddScene, 744
 - ApplyShadows, 744
 - AttachEntity, 744
 - AttachViewport, 744
 - Clear, 745
 - DetachEntity, 745
 - DetachViewport, 745
 - Fini, 745
 - GenerateShaders, 745
 - Init, 745
 - LightingModel, 744
 - RemoveScene, 745
 - RemoveShadows, 745
 - SSLM_NormalMapLightingObjectSpace, 744
 - SSLM_NormalMapLightingTangentSpace, 744
 - SSLM_PerPixelLighting, 744
 - SSLM_PerVertexLighting, 744
 - SetPerPixelLighting, 746
 - UpdateShaders, 746
- gazebo::rendering::RenderEngine, 722
 - AddResourcePath, 724
 - CreateScene, 724
 - DEFERRED, 724
 - dummyContext, 726
 - dummyDisplay, 726
 - dummyWindowId, 726
 - FORWARD, 724
 - Fini, 725
 - GetRenderPathType, 725
 - GetScene, 725
 - GetSceneCount, 725
 - Init, 726
 - Load, 726
 - NONE, 724
 - RENDER_PATH_COUNT, 724
 - RemoveScene, 726
 - RenderPathType, 724
 - root, 726
 - VERTEX, 724
- gazebo::rendering::Road2d, 737
 - ~Road2d, 738
 - Load, 738
 - Road2d, 738
- gazebo::rendering::Scene, 746
 - ~Scene, 749

- AddVisual, 749
- Clear, 750
- CloneVisual, 750
- CreateCamera, 750
- CreateDepthCamera, 750
- CreateGpuLaser, 750
- CreateGrid, 751
- CreateUserCamera, 751
- DrawLine, 751
- GetAmbientColor, 751
- GetBackgroundColor, 752
- GetCamera, 752
- GetCameraCount, 752
- GetFirstContact, 752
- GetGrid, 753
- GetGridCount, 753
- GetHeightmap, 753
- GetId, 753
- GetIdString, 753
- GetLight, 754
- GetLightCount, 754
- GetManager, 754
- GetModelVisualAt, 754
- GetName, 755
- GetSelectedVisual, 755
- GetShadowsEnabled, 755
- GetUserCamera, 755
- GetUserCameraCount, 756
- GetVisual, 756
- GetVisualAt, 756
- GetVisualBelow, 756
- GetVisualsBelowPoint, 757
- GetWorldVisual, 757
- Init, 757
- Load, 757
- PreRender, 757
- PrintSceneGraph, 757
- RemoveVisual, 758
- Scene, 749
- SelectVisual, 758
- SetAmbientColor, 758
- SetBackgroundColor, 758
- SetFog, 758
- SetGrid, 759
- SetShadowsEnabled, 759
- SetVisible, 759
- skyx, 760
- SnapVisualToNearestBelow, 759
- StripSceneName, 759
- gazebo::rendering::SelectionObj, 763
 - ~SelectionObj, 764
 - Attach, 764
 - Clear, 764
 - GetVisualName, 765
 - Init, 765
 - IsActive, 765
 - SelectionObj, 764
 - SetActive, 765
 - SetHighlight, 765
- gazebo::rendering::UserCamera, 878
 - ~UserCamera, 880
 - AttachToVisualImpl, 880
 - EnableViewController, 881
 - Fini, 881
 - GetAvgFPS, 881
 - GetGUIOverlay, 881
 - GetTriangleCount, 881
 - GetVisual, 882
 - HandleKeyPressEvent, 882
 - HandleKeyReleaseEvent, 882
 - HandleMouseEvent, 882
 - Init, 883
 - Load, 883
 - MoveToPosition, 883
 - MoveToVisual, 883
 - PostRender, 883
 - Resize, 884
 - SetFocalPoint, 884
 - SetRenderTarget, 884
 - SetViewController, 884
 - SetViewportDimensions, 885
 - SetWorldPose, 885
 - TrackVisualImpl, 885
 - Update, 885
 - UserCamera, 880
- gazebo::rendering::VideoVisual, 926
 - ~VideoVisual, 927
 - VideoVisual, 927
- gazebo::rendering::ViewController, 927
 - ~ViewController, 928
 - camera, 930
 - enabled, 930
 - GetTypeString, 929
 - HandleKeyPressEvent, 929
 - HandleKeyReleaseEvent, 929
 - HandleMouseEvent, 929
 - Init, 929, 930
 - SetEnabled, 930
 - typeString, 930
 - Update, 930
 - ViewController, 928
- gazebo::rendering::Visual, 931
 - ~Visual, 936
 - AttachAxes, 936
 - AttachLineVertex, 936
 - AttachMesh, 936
 - AttachObject, 936
 - AttachVisual, 936

- ClearParent, 937
- Clone, 937
- CreateDynamicLine, 937
- DeleteDynamicLine, 937
- DetachObjects, 937
- DetachVisual, 937, 938
- DisableTrackVisual, 938
- EnableTrackVisual, 938
- Fini, 938
- GetAttachedObjectCount, 938
- GetBoundingBox, 938
- GetChild, 938
- GetChildCount, 939
- GetMaterialName, 939
- GetName, 939
- GetNormalMap, 939
- GetParent, 939
- GetPose, 939
- GetPosition, 940
- GetRootVisual, 940
- GetRotation, 940
- GetScale, 940
- GetScene, 940
- GetSceneNode, 940
- GetShaderType, 941
- GetTransparency, 941
- GetVisibilityFlags, 941
- GetVisible, 941
- GetWorldPose, 941
- HasAttachedObject, 942
- Init, 942
- InsertMesh, 942
- IsPlane, 942
- IsStatic, 942
- Load, 943
- LoadFromMsg, 943
- LoadPlugin, 943
- MakeStatic, 943
- MoveToPosition, 943
- MoveToPositions, 944
- parent, 949
- RemovePlugin, 944
- scene, 949
- sceneNode, 949
- SetAmbient, 944
- SetCastShadows, 944
- SetDiffuse, 944
- SetEmissive, 944
- SetHighlighted, 945
- SetMaterial, 945
- SetName, 945
- SetNormalMap, 945
- SetPose, 945
- SetPosition, 945
- SetRibbonTrail, 945
- SetRotation, 946
- SetScale, 946
- SetScene, 946
- SetShaderType, 946
- SetSkeletonPose, 946
- SetSpecular, 947
- SetTransparency, 947
- SetVisibilityFlags, 947
- SetVisible, 947
- SetWorldPose, 947
- SetWorldPosition, 948
- SetWorldRotation, 948
- ShowBoundingBox, 948
- ShowCOM, 948
- ShowCollision, 948
- ShowJoints, 948
- ShowSkeleton, 948
- ToggleVisible, 949
- Update, 949
- UpdateFromMsg, 949
- Visual, 935
- gazebo::rendering::WindowManager, 951
 - CreateWindow, 952
 - Fini, 952
 - GetAvgFPS, 952
 - GetTriangleCount, 952
 - GetWindow, 953
 - Moved, 953
 - Resize, 953
 - SetCamera, 953
- gazebo::sensors, 113
 - CameraSensor_V, 115
 - CameraSensorPtr, 115
 - ContactSensor_V, 115
 - ContactSensorPtr, 115
 - DepthCameraSensor_V, 115
 - DepthCameraSensorPtr, 115
 - GpuRaySensor_V, 115
 - GpuRaySensorPtr, 115
 - RFIDSensor_V, 115
 - RFIDSensorPtr, 115
 - RFIDTag_V, 115
 - RFIDTagPtr, 115
 - RaySensor_V, 115
 - RaySensorPtr, 115
 - Sensor_V, 115
 - SensorFactoryFn, 115
 - SensorPtr, 115
- gazebo::sensors::CameraSensor, 255
 - ~CameraSensor, 257
 - CameraSensor, 257
 - Fini, 257
 - GetCamera, 257

- GetImageData, 257
- GetImageHeight, 257
- GetImageWidth, 257
- GetTopic, 258
- Init, 258
- Load, 258
- SaveFrame, 258
- SetActive, 259
- SetParent, 259
- UpdateImpl, 259
- gazebo::sensors::ContactSensor, 300
 - ~ContactSensor, 301
 - ContactSensor, 301
 - Fini, 301
 - GetCollisionContact, 301
 - GetCollisionContactCount, 302
 - GetCollisionCount, 302
 - GetCollisionName, 302
 - GetContacts, 302
 - GetUpdateMutex, 303
 - Init, 303
 - Load, 303
 - UpdateImpl, 303
- gazebo::sensors::DepthCameraSensor, 317
 - ~DepthCameraSensor, 318
 - DepthCameraSensor, 318
 - Fini, 318
 - GetDepthCamera, 318
 - Init, 318
 - Load, 319
 - SaveFrame, 319
 - SetActive, 319
 - SetParent, 319
 - UpdateImpl, 320
- gazebo::sensors::GpuRaySensor, 378
 - ~GpuRaySensor, 381
 - cameraCount, 388
 - cameraElem, 388
 - chfov, 388
 - ConnectNewLaserFrame, 381
 - cvfov, 388
 - DisconnectNewLaserFrame, 381
 - far, 388
 - Fini, 382
 - Get1stRatio, 382
 - Get2ndRatio, 382
 - GetAngleMax, 382
 - GetAngleMin, 382
 - GetAngleResolution, 382
 - GetCHFOV, 383
 - GetCVFOV, 383
 - GetCameraCount, 382
 - GetFiducial, 383
 - GetHAngle, 383
 - GetHFOV, 383
 - GetLaserCamera, 383
 - GetRange, 383
 - GetRangeCount, 384
 - GetRangeMax, 384
 - GetRangeMin, 384
 - GetRangeResolution, 384
 - GetRanges, 385
 - GetRayCount, 385
 - GetRetro, 385
 - GetVAngle, 385
 - GetVFOV, 386
 - GetVerticalAngleMax, 385
 - GetVerticalAngleMin, 386
 - GetVerticalRangeCount, 386
 - GetVerticalRayCount, 386
 - GpuRaySensor, 381
 - hang, 388
 - height_1st, 388
 - height_2nd, 388
 - hfov, 388
 - horzElem, 388
 - Init, 386
 - IsHorizontal, 386
 - isHorizontal, 388
 - Load, 386, 387
 - near, 388
 - offset, 388
 - rangeElem, 388
 - ratio_1st, 388
 - ratio_2nd, 388
 - rayElem, 388
 - scanElem, 388
 - SetAngleMax, 387
 - SetAngleMin, 387
 - SetVerticalAngleMax, 387
 - SetVerticalAngleMin, 387
 - UpdateImpl, 388
 - vang, 388
 - vertElem, 389
 - vfov, 389
 - width_1st, 389
 - width_2nd, 389
- gazebo::sensors::ImuSensor, 412
 - ~ImuSensor, 413
 - FiniChild, 413
 - GetVelocity, 413
 - ImuSensor, 413
 - InitChild, 414
 - LoadChild, 414
 - SaveChild, 414
 - UpdateChild, 414
- gazebo::sensors::RFIDSensor, 727
 - ~RFIDSensor, 728

- Fini, 728
 - Init, 728
 - Load, 728
 - RFIDSensor, 728
 - UpdateImpl, 729
- gazebo::sensors::RFIDTag, 729
 - ~RFIDTag, 730
 - Fini, 730
 - GetTagPose, 730
 - Init, 731
 - Load, 731
 - RFIDTag, 730
 - UpdateImpl, 731
- gazebo::sensors::RFIDTagManager, 732
 - AddTaggedModel, 732
 - GetTags, 732
- gazebo::sensors::RaySensor, 711
 - ~RaySensor, 713
 - Fini, 713
 - GetAngleMax, 713
 - GetAngleMin, 714
 - GetAngleResolution, 714
 - GetFiducial, 714
 - GetLaserShape, 714
 - GetRange, 714
 - GetRangeCount, 715
 - GetRangeMax, 715
 - GetRangeMin, 715
 - GetRangeResolution, 715
 - GetRanges, 715
 - GetRayCount, 716
 - GetRetro, 716
 - GetTopic, 716
 - GetVerticalAngleMax, 716
 - GetVerticalAngleMin, 716
 - GetVerticalRangeCount, 717
 - GetVerticalRayCount, 717
 - Init, 717
 - Load, 717
 - RaySensor, 713
 - UpdateImpl, 718
- gazebo::sensors::Sensor, 765
 - ~Sensor, 767
 - active, 771
 - connections, 771
 - FillMsg, 768
 - Fini, 768
 - GetLastUpdateTime, 768
 - GetName, 768
 - GetParentName, 768
 - GetPose, 768
 - GetScopedName, 769
 - GetTopic, 769
 - GetType, 769
 - GetVisualize, 769
 - GetWorldName, 769
 - Init, 769
 - IsActive, 770
 - lastUpdateTime, 771
 - Load, 770
 - node, 771
 - parentName, 771
 - plugins, 772
 - pose, 772
 - poseSub, 772
 - sdf, 772
 - Sensor, 767
 - SetActive, 770
 - SetParent, 770
 - SetUpdateRate, 771
 - Update, 771
 - UpdateImpl, 771
 - updatePeriod, 772
 - world, 772
- gazebo::sensors::SensorFactory, 772
 - GetSensorTypes, 773
 - NewSensor, 773
 - RegisterAll, 773
 - RegisterSensor, 773
- gazebo::sensors::SensorManager, 774
 - CreateSensor, 775
 - Fini, 775
 - GetSensor, 775
 - GetSensorTypes, 775
 - Init, 776
 - RemoveSensor, 776
 - RemoveSensors, 776
 - Run, 776
 - SensorsInitialized, 776
 - Stop, 776
 - Update, 776
- gazebo::transport, 115
 - ConnectionPtr, 117
 - NodePtr, 117
 - PublicationPtr, 117
 - PublicationTransportPtr, 117
 - PublisherPtr, 117
 - SubscriberPtr, 117
 - SubscriptionTransportPtr, 117
- gazebo::transport::CallbackHelper, 230
 - ~CallbackHelper, 231
 - CallbackHelper, 231
 - GetLatching, 231
 - GetMsgType, 231
 - HandleData, 231
 - IsLocal, 231
 - latching, 231
- gazebo::transport::CallbackHelperT

- CallbackHelperT, 232
- GetMsgType, 232
- HandleData, 232
- IsLocal, 233
- gazebo::transport::CallbackHelperT< M >, 231
- gazebo::transport::Connection, 288
 - ~Connection, 290
 - AcceptCallback, 290
 - AsyncRead, 290
 - Cancel, 290
 - Connect, 290
 - ConnectToShutdown, 290
 - Connection, 290
 - DisconnectShutdown, 290
 - EnqueueMsg, 290
 - GetLocalAddress, 290
 - GetLocalHostname, 291
 - GetLocalPort, 291
 - GetLocalURI, 291
 - GetRemoteAddress, 291
 - GetRemoteHostname, 291
 - GetRemotePort, 291
 - GetRemoteURI, 291
 - id, 292
 - IsOpen, 291
 - Listen, 291
 - ProcessWriteQueue, 291
 - Read, 291
 - ReadCallback, 290
 - Shutdown, 292
 - StartRead, 292
 - StopRead, 292
 - writeCount, 292
- gazebo::transport::ConnectionManager, 292
 - Advertise, 293
 - ConnectToRemoteHost, 293
 - eventConnections, 294
 - Fini, 293
 - GetAllPublishers, 294
 - GetTopicNamespaces, 294
 - Init, 294
 - IsRunning, 294
 - RegisterTopicNamespace, 294
 - RemoveConnection, 294
 - Run, 294
 - RunUpdate, 294
 - Stop, 294
 - Subscribe, 294
 - Unadvertise, 294
 - Unsubscribe, 294
- gazebo::transport::DebugCallbackHelper, 311
 - DebugCallbackHelper, 312
 - GetMsgType, 312
 - HandleData, 312
 - IsLocal, 312
- gazebo::transport::IOManager, 422
 - ~IOManager, 423
 - DecCount, 423
 - GetCount, 423
 - GetIO, 423
 - IOManager, 423
 - IncCount, 423
 - Stop, 423
- gazebo::transport::Node, 556
 - ~Node, 558
 - Advertise, 558
 - DecodeTopicName, 558
 - EncodeTopicName, 558
 - Fini, 558
 - GetId, 558
 - GetMsgType, 558
 - GetTopicNamespace, 558
 - HandleData, 559
 - Init, 559
 - InsertLatchedMsg, 559
 - Node, 558
 - ProcessIncoming, 559
 - ProcessPublishers, 559
 - Subscribe, 559
- gazebo::transport::Publication, 692
 - ~Publication, 692
 - AddPublisher, 693
 - AddSubscription, 693
 - AddTransport, 693
 - GetCallbackCount, 693
 - GetLocallyAdvertised, 693
 - GetMsgType, 693
 - GetNodeCount, 693
 - GetRemoteSubscriptionCount, 693
 - GetTransportCount, 693
 - HasTransport, 693
 - LocalPublish, 693
 - Publication, 692
 - Publish, 693
 - RemoveSubscription, 693
 - RemoveTransport, 694
 - SetLocallyAdvertised, 694
- gazebo::transport::PublicationTransport, 694
 - ~PublicationTransport, 694
 - AddCallback, 694
 - Fini, 694
 - GetConnection, 694
 - GetMsgType, 695
 - GetTopic, 695
 - Init, 695
 - PublicationTransport, 694
- gazebo::transport::Publisher, 695
 - ~Publisher, 696

- GetLatching, 696
- GetMsgType, 696
- GetOutgoingCount, 696
- GetPrevMsg, 696
- GetTopic, 696
- HasConnections, 696
- Publish, 696
- Publisher, 696
- SendMessage, 696
- SetPublication, 696
- WaitForConnection, 696
- gazebo::transport::SubscribeOptions, 826
 - GetLatching, 827
 - GetMsgType, 827
 - GetNode, 827
 - GetTopic, 827
 - Init, 827
 - SubscribeOptions, 827
- gazebo::transport::Subscriber, 827
 - ~Subscriber, 828
 - GetTopic, 828
 - Subscriber, 828
 - Unsubscribe, 828
- gazebo::transport::SubscriptionTransport, 828
 - ~SubscriptionTransport, 829
 - GetConnection, 829
 - HandleData, 829
 - Init, 829
 - IsLocal, 829
 - SubscriptionTransport, 829
- gazebo::transport::TopicManager, 861
 - AddNode, 863
 - Advertise, 863
 - ClearBuffers, 863
 - ConnectPubToSub, 863
 - ConnectSubToPub, 863
 - ConnectSubscribers, 863
 - DisconnectPubFromSub, 863
 - DisconnectSubFromPub, 863
 - FindPublication, 863
 - Fini, 863
 - GetTopicNamespaces, 864
 - Init, 864
 - IsAdvertised, 864
 - PauseIncoming, 864
 - ProcessNodes, 864
 - Publish, 864
 - RegisterTopicNamespace, 864
 - RemoveNode, 864
 - SubNodeMap, 863
 - Subscribe, 864
 - Unadvertise, 865
 - Unsubscribe, 865
 - UpdatePublications, 865
- gazebo_core.hh, 1031
- gazebo_extensions_
 - urdf2gazebo::URDF2Gazebo, 878
- Gazebo_parser, 82
 - CollisionPtr, 82
 - VisualPtr, 82
- GazeboExtension
 - urdf2gazebo::GazeboExtension, 370
- GazeboGenerator
 - google::protobuf::compiler::cpp::GazeboGenerator, 374
- GazeboGenerator.hh, 1031
- gazeboPathsFromEnv
 - gazebo::common::SystemPaths, 838
- GenSphericalTexCoord
 - gazebo::common::Mesh, 511
 - gazebo::common::MeshManager, 519
 - gazebo::common::SubMesh, 821
- Generate
 - google::protobuf::compiler::cpp::GazeboGenerator, 374
- GenerateShaders
 - gazebo::rendering::RTShaderSystem, 745
- GeneratorType
 - gazebo::math, 101
- Get
 - gazebo::common::NodeTransform, 566
 - sdf::Param, 644, 645
- Get1stRatio
 - gazebo::sensors::GpuRaySensor, 382
- Get2ndRatio
 - gazebo::sensors::GpuRaySensor, 382
- get_master_uri
 - Transport, 90
- get_scene
 - Rendering, 81
- get_sensor
 - Sensors, 85
- get_topic_namespaces
 - Transport, 90
- get_world
 - Classes for physics and dynamics, 72
- GetAABB
 - gazebo::common::Mesh, 511
 - gazebo::rendering::MovableText, 545
- GetAbs
 - gazebo::math::Vector3, 907
- GetAcceleration
 - gazebo::physics::LinkState, 475
- GetAllLinks
 - gazebo::physics::Model, 525
- GetAllPublishers
 - gazebo::transport::ConnectionManager, 294
- GetAmbient

- gazebo::common::Material, 492
- GetAmbientColor
 - gazebo::rendering::Scene, 751
- GetAnchor
 - gazebo::physics::BulletBallJoint, 168
 - gazebo::physics::BulletHinge2Joint, 181
 - gazebo::physics::BulletHingeJoint, 186
 - gazebo::physics::BulletJoint, 190
 - gazebo::physics::BulletUniversalJoint, 227
 - gazebo::physics::Joint, 428
 - gazebo::physics::ODEBallJoint, 575
 - gazebo::physics::ODEHinge2Joint, 589
 - gazebo::physics::ODEHingeJoint, 593
 - gazebo::physics::ODEUniversalJoint, 636
 - gazebo::physics::ScrewJoint, 761
 - gazebo::physics::SliderJoint, 805
- GetAngle
 - gazebo::physics::BulletBallJoint, 168
 - gazebo::physics::BulletHinge2Joint, 181
 - gazebo::physics::BulletHingeJoint, 186
 - gazebo::physics::BulletScrewJoint, 215
 - gazebo::physics::BulletSliderJoint, 219
 - gazebo::physics::BulletUniversalJoint, 227
 - gazebo::physics::Joint, 429
 - gazebo::physics::JointState, 443
- GetAngleCount
 - gazebo::physics::JointState, 443
- GetAngleImpl
 - gazebo::physics::BulletBallJoint, 168
 - gazebo::physics::BulletHinge2Joint, 181
 - gazebo::physics::BulletHingeJoint, 186
 - gazebo::physics::BulletScrewJoint, 215
 - gazebo::physics::BulletSliderJoint, 219
 - gazebo::physics::BulletUniversalJoint, 227
 - gazebo::physics::Joint, 429
 - gazebo::physics::ODEBallJoint, 575
 - gazebo::physics::ODEHinge2Joint, 589
 - gazebo::physics::ODEHingeJoint, 593
 - gazebo::physics::ODEScrewJoint, 624
 - gazebo::physics::ODESliderJoint, 627
 - gazebo::physics::ODEUniversalJoint, 636
- GetAngleMax
 - gazebo::sensors::GpuRaySensor, 382
 - gazebo::sensors::RaySensor, 713
- GetAngleMin
 - gazebo::sensors::GpuRaySensor, 382
 - gazebo::sensors::RaySensor, 714
- GetAngleResolution
 - gazebo::sensors::GpuRaySensor, 382
 - gazebo::sensors::RaySensor, 714
- GetAngularDamping
 - gazebo::physics::Link, 462
- GetAnimation
 - gazebo::common::Skeleton, 787
- GetAsABGR
 - gazebo::common::Color, 277
- GetAsARGB
 - gazebo::common::Color, 277
- GetAsAxis
 - gazebo::math::Quaternion, 702
- GetAsBGRA
 - gazebo::common::Color, 277
- GetAsDegree
 - gazebo::math::Angle, 129
- GetAsEuler
 - gazebo::math::Quaternion, 702
- GetAsHSV
 - gazebo::common::Color, 278
- GetAsMatrix3
 - gazebo::math::Quaternion, 702
- GetAsMatrix4
 - gazebo::math::Quaternion, 702
- GetAsPose
 - gazebo::math::Matrix4, 504
- GetAsRGBA
 - gazebo::common::Color, 278
- GetAsRadian
 - gazebo::math::Angle, 129
- GetAsString
 - sdf::Param, 645
 - sdf::ParamT, 650
- GetAsYUV
 - gazebo::common::Color, 278
- GetAspectRatio
 - gazebo::rendering::Camera, 240
- GetAttachedObjectCount
 - gazebo::rendering::Visual, 938
- GetAttribute
 - sdf::Element, 334, 335
- GetAttributeCount
 - sdf::Element, 335
- GetAttributeSet
 - sdf::Element, 335
- GetAutoDisableFlag
 - gazebo::physics::PhysicsEngine, 656
- GetAvgColor
 - gazebo::common::Image, 409
- GetAvgFPS
 - gazebo::rendering::Camera, 240
 - gazebo::rendering::UserCamera, 881
 - gazebo::rendering::WindowManager, 952
- GetAxis
 - gazebo::physics::BulletBallJoint, 168
 - gazebo::physics::BulletHinge2Joint, 181
 - gazebo::physics::BulletUniversalJoint, 227
- GetBPP
 - gazebo::common::Image, 409
- GetBackgroundColor

- gazebo::rendering::Scene, 752
- GetBaseline
 - gazebo::rendering::MovableText, 545
- GetBindShapeTransform
 - gazebo::common::Skeleton, 787
- GetBlendFactors
 - gazebo::common::Material, 492
- GetBlendMode
 - gazebo::common::Material, 492
- GetBoundingBox
 - gazebo::physics::BulletCollision, 173
 - gazebo::physics::Collision, 266
 - gazebo::physics::Entity, 342
 - gazebo::physics::Link, 462
 - gazebo::physics::Model, 525
 - gazebo::physics::ODECollision, 580
 - gazebo::rendering::Visual, 938
- getBoundingBoxRadius
 - gazebo::rendering::DynamicRenderable, 330
 - gazebo::rendering::MovableText, 545
- GetBulletLink
 - gazebo::physics::BulletLink, 195
- GetById
 - gazebo::physics::Base, 150
- GetByName
 - gazebo::physics::Base, 151
 - gazebo::physics::World, 957
- GetCFM
 - gazebo::physics::ODEJoint, 597
- GetCHFOV
 - gazebo::sensors::GpuRaySensor, 383
- GetCVFOV
 - gazebo::sensors::GpuRaySensor, 383
- GetCallbackCount
 - gazebo::transport::Publication, 693
- GetCamera
 - gazebo::rendering::Scene, 752
 - gazebo::sensors::CameraSensor, 257
- GetCameraCount
 - gazebo::rendering::Scene, 752
 - gazebo::sensors::GpuRaySensor, 382
- GetCameraToViewportRay
 - gazebo::rendering::Camera, 241
- GetCellCount
 - gazebo::rendering::Grid, 390
- GetCellLength
 - gazebo::rendering::Grid, 390
- GetCenter
 - gazebo::math::Box, 159
- GetCharHeight
 - gazebo::rendering::MovableText, 546
- GetChild
 - gazebo::common::SkeletonNode, 798
 - gazebo::physics::Base, 151
 - gazebo::physics::Joint, 429
 - gazebo::rendering::Visual, 938
- GetChildById
 - gazebo::common::SkeletonNode, 798
- GetChildByName
 - gazebo::common::SkeletonNode, 799
- GetChildCollision
 - gazebo::physics::Entity, 342
- GetChildCount
 - gazebo::common::SkeletonNode, 799
 - gazebo::physics::Base, 151
 - gazebo::rendering::Visual, 939
- GetChildJointsLinks
 - gazebo::physics::Link, 462
- GetChildLink
 - gazebo::physics::Entity, 342
- GetCmd
 - gazebo::common::PID, 666
- GetCoG
 - gazebo::physics::Inertial, 417
- GetCollision
 - gazebo::physics::Link, 462, 463
- GetCollisionBoundingBox
 - gazebo::physics::Entity, 342
- GetCollisionById
 - gazebo::physics::Link, 463
- GetCollisionClass
 - gazebo::physics::ODECollision, 580
- GetCollisionContact
 - gazebo::sensors::ContactSensor, 301
- GetCollisionContactCount
 - gazebo::sensors::ContactSensor, 302
- GetCollisionCount
 - gazebo::sensors::ContactSensor, 302
- GetCollisionId
 - gazebo::physics::ODECollision, 581
- GetCollisionName
 - gazebo::sensors::ContactSensor, 302
- GetCollisionShape
 - gazebo::physics::BulletCollision, 173
- GetCollisionState
 - gazebo::physics::LinkState, 475, 476
- GetCollisionStateCount
 - gazebo::physics::LinkState, 476
- GetColor
 - gazebo::rendering::Grid, 391
 - gazebo::rendering::MovableText, 546
- GetConnection
 - gazebo::transport::PublicationTransport, 694
 - gazebo::transport::SubscriptionTransport, 829
- GetContactMaxCorrectingVel
 - gazebo::physics::ODEPhysics, 613
 - gazebo::physics::PhysicsEngine, 656
- GetContactSurfaceLayer

- gazebo::physics::ODEPhysics, 614
- gazebo::physics::PhysicsEngine, 656
- GetContacts
 - gazebo::sensors::ContactSensor, 302
- GetContactsEnabled
 - gazebo::physics::Collision, 267
- GetCopyChildren
 - sdf::Element, 335
- GetCount
 - gazebo::physics::BulletRaySensor, 210
 - gazebo::transport::LOManager, 423
- GetCurrentDir
 - SystemPaths.hh, 1168
- GetData
 - gazebo::common::Image, 409
- GetDbiNormal
 - gazebo::math::Rand, 711
- GetDbiUniform
 - gazebo::math::Rand, 711
- GetDefaultAsString
 - sdf::Param, 645
 - sdf::ParamT, 650
- GetDefaultValue
 - sdf::ParamT, 650
- GetDepthCamera
 - gazebo::sensors::DepthCameraSensor, 318
- GetDepthData
 - gazebo::rendering::DepthCamera, 316
- GetDepthWrite
 - gazebo::common::Material, 492
- GetDescription
 - sdf::Element, 335
 - sdf::Param, 645
- GetDiffuse
 - gazebo::common::Material, 493
- GetDiffuseColor
 - gazebo::rendering::Light, 450
- GetDirection
 - gazebo::rendering::Camera, 241
 - gazebo::rendering::Light, 450
- GetDirtyPose
 - gazebo::physics::Entity, 343
- GetDistToLine
 - gazebo::math::Vector3, 907
- GetDynamicsWorld
 - gazebo::physics::BulletPhysics, 205
- GetERP
 - gazebo::physics::ODEJoint, 597
- GetElapsed
 - gazebo::common::Timer, 861
- GetElement
 - sdf::Element, 335
- GetElementDescription
 - sdf::Element, 335
- GetElementDescriptionCount
 - sdf::Element, 335
- GetElementImpl
 - sdf::Element, 335
- GetEmissive
 - gazebo::common::Material, 493
- GetEnablePhysicsEngine
 - gazebo::physics::World, 958
- GetEnabled
 - gazebo::common::DiagnosticManager, 321
 - gazebo::physics::BulletLink, 195
 - gazebo::physics::Link, 463
 - gazebo::physics::ODELink, 603
- GetEntity
 - gazebo::physics::World, 958
- GetEntityBelowPoint
 - gazebo::physics::World, 958
- GetEntityByName
 - gazebo::physics::World, 958
- GetErrorFile
 - gazebo::common::Exception, 366
- GetErrorStr
 - gazebo::common::Exception, 366
- GetErrors
 - gazebo::common::PID, 666
- GetEulerRotation
 - gazebo::math::Matrix4, 504
- GetExp
 - gazebo::math::Quaternion, 702
- GetFarClip
 - gazebo::rendering::Camera, 241
- GetFeedback
 - gazebo::physics::ODEJoint, 597
- GetFiducial
 - gazebo::physics::BulletRaySensor, 210
 - gazebo::physics::MultiRayShape, 552
 - gazebo::physics::RayShape, 720
 - gazebo::sensors::GpuRaySensor, 383
 - gazebo::sensors::RaySensor, 714
- GetFilename
 - gazebo::common::Image, 409
 - gazebo::physics::TrimeshShape, 867
 - gazebo::PluginT, 676
- GetFirstContact
 - gazebo::rendering::Scene, 752
- GetFirstElement
 - sdf::Element, 336
- GetFocalPoint
 - gazebo::rendering::OrbitViewController, 639
- GetFont
 - gazebo::rendering::MovableText, 546
- GetForce
 - gazebo::physics::BulletHingeJoint, 186
 - gazebo::physics::Joint, 429

- GetForces
 - gazebo::physics::LinkState, 476
- GetFrameAt
 - gazebo::common::NodeAnimation, 561
- GetFrameCount
 - gazebo::common::NodeAnimation, 561
- GetFrameFilename
 - gazebo::rendering::Camera, 241
- GetGUIOverlay
 - gazebo::rendering::UserCamera, 881
- GetGazeboPaths
 - gazebo::common::SystemPaths, 837
- getGeometryBoundingBox
 - urdf2gazebo::URDF2Gazebo, 874
- GetGlobalAxis
 - gazebo::physics::BulletBallJoint, 168
 - gazebo::physics::BulletHinge2Joint, 181
 - gazebo::physics::BulletHingeJoint, 186
 - gazebo::physics::BulletScrewJoint, 215
 - gazebo::physics::BulletSliderJoint, 219
 - gazebo::physics::BulletUniversalJoint, 227
 - gazebo::physics::Joint, 430
 - gazebo::physics::ODEBallJoint, 575
 - gazebo::physics::ODEHinge2Joint, 589
 - gazebo::physics::ODEHingeJoint, 593
 - gazebo::physics::ODEScrewJoint, 624
 - gazebo::physics::ODESliderJoint, 628
 - gazebo::physics::ODEUniversalJoint, 636
- GetGlobalPoints
 - gazebo::physics::RayShape, 720
- GetGranularity
 - gazebo::physics::MapShape, 485
- GetGravity
 - gazebo::physics::ODEPhysics, 614
 - gazebo::physics::PhysicsEngine, 656
- GetGravityMode
 - gazebo::physics::BulletLink, 195
 - gazebo::physics::Link, 463
 - gazebo::physics::ODELink, 603
- GetGrid
 - gazebo::rendering::Scene, 753
- GetGridCount
 - gazebo::rendering::Scene, 753
- GetHAngle
 - gazebo::sensors::GpuRaySensor, 383
- GetHFOV
 - gazebo::rendering::Camera, 241
 - gazebo::sensors::GpuRaySensor, 383
- GetHandle
 - gazebo::common::SkeletonNode, 799
 - gazebo::PluginT, 676
- GetHeader
 - Messages, 64
- GetHeight
 - gazebo::common::Image, 410
 - gazebo::common::Video, 925
 - gazebo::physics::HeightmapShape, 402
 - gazebo::physics::MapShape, 485
 - gazebo::rendering::Grid, 391
 - gazebo::rendering::Heightmap, 398
- GetHeightmap
 - gazebo::rendering::Scene, 753
- GetHighStop
 - gazebo::physics::BallJoint, 144
 - gazebo::physics::BulletHinge2Joint, 181
 - gazebo::physics::BulletHingeJoint, 186
 - gazebo::physics::BulletScrewJoint, 215
 - gazebo::physics::BulletSliderJoint, 219
 - gazebo::physics::BulletUniversalJoint, 227
 - gazebo::physics::Joint, 430
 - gazebo::physics::ODEJoint, 597
- GetIO
 - gazebo::transport::IOManager, 423
- GetIXX
 - gazebo::physics::Inertial, 417
- GetIXY
 - gazebo::physics::Inertial, 417
- GetIXZ
 - gazebo::physics::Inertial, 417
- GetIYY
 - gazebo::physics::Inertial, 417
- GetIYZ
 - gazebo::physics::Inertial, 418
- GetIZZ
 - gazebo::physics::Inertial, 418
- GetId
 - gazebo::common::SkeletonNode, 799
 - gazebo::event::Connection, 288
 - gazebo::physics::Base, 152
 - gazebo::rendering::Scene, 753
 - gazebo::transport::Node, 558
- GetIdString
 - gazebo::rendering::Scene, 753
- GetImageByteSize
 - gazebo::rendering::Camera, 241, 242
- GetImageData
 - gazebo::rendering::Camera, 242
 - gazebo::sensors::CameraSensor, 257
- GetImageDepth
 - gazebo::rendering::Camera, 242
- GetImageFormat
 - gazebo::rendering::Camera, 242
- GetImageHeight
 - gazebo::rendering::Camera, 243
 - gazebo::sensors::CameraSensor, 257
- GetImageWidth
 - gazebo::rendering::Camera, 243
 - gazebo::sensors::CameraSensor, 257

- GetInclude
 - sdf::Element, 336
- GetIndex
 - gazebo::common::SubMesh, 822
- GetIndexCount
 - gazebo::common::Mesh, 511
 - gazebo::common::SubMesh, 822
- GetInertial
 - gazebo::physics::BoxShape, 164
 - gazebo::physics::CylinderShape, 309
 - gazebo::physics::Link, 464
 - gazebo::physics::Shape, 781
 - gazebo::physics::SphereShape, 807
- GetInitialized
 - gazebo::rendering::Camera, 243
 - gazebo::Server, 779
- GetIntNormal
 - gazebo::math::Rand, 711
- GetIntUniform
 - gazebo::math::Rand, 711
- GetInterpolatedKeyFrame
 - gazebo::common::NumericAnimation, 571
 - gazebo::common::PoseAnimation, 686, 687
- GetIntersection
 - gazebo::physics::BulletRayShape, 212
 - gazebo::physics::ODERayShape, 621
 - gazebo::physics::RayShape, 720
- GetInverse
 - gazebo::math::Pose, 681
 - gazebo::math::Quaternion, 702
- GetInverseBindTransform
 - gazebo::common::SkeletonNode, 799
- GetJoint
 - gazebo::physics::Model, 525, 526
- GetJointCount
 - gazebo::physics::Model, 526
- GetJointLink
 - gazebo::physics::BulletJoint, 190
 - gazebo::physics::Joint, 430
 - gazebo::physics::ODEJoint, 597
- GetJointState
 - gazebo::physics::ModelState, 538
- GetJointStateCount
 - gazebo::physics::ModelState, 538
- GetKey
 - sdf::Param, 645
- GetKeyFrame
 - gazebo::common::Animation, 137
 - gazebo::common::NodeAnimation, 562
- GetKeyFrameCount
 - gazebo::common::Animation, 137
- GetKeyFramesATime
 - gazebo::common::Animation, 137
- getKeyValueAsString
 - urdf2gazebo::URDF2Gazebo, 874
- GetKinematic
 - gazebo::physics::Link, 464
 - gazebo::physics::ODELink, 604
- GetLabel
 - gazebo::common::DiagnosticManager, 321
- GetLaserCamera
 - gazebo::sensors::GpuRaySensor, 383
- GetLaserData
 - gazebo::rendering::GpuLaser, 377
- GetLaserRetro
 - gazebo::physics::Collision, 267
- GetLaserShape
 - gazebo::sensors::RaySensor, 714
- GetLastRenderWallTime
 - gazebo::rendering::Camera, 243
- GetLastUpdateTime
 - gazebo::sensors::Sensor, 768
- GetLatching
 - gazebo::transport::CallbackHelper, 231
 - gazebo::transport::Publisher, 696
 - gazebo::transport::SubscribeOptions, 827
- GetLength
 - gazebo::common::Animation, 137
 - gazebo::common::NodeAnimation, 562
 - gazebo::common::SkeletonAnimation, 792
 - gazebo::math::Vector3, 907
 - gazebo::math::Vector4, 918
 - gazebo::physics::CylinderShape, 309
 - gazebo::physics::RayShape, 720
- GetLight
 - gazebo::rendering::Scene, 754
- GetLightCount
 - gazebo::rendering::Scene, 754
- GetLighting
 - gazebo::common::Material, 493
- getLights
 - gazebo::rendering::MovableText, 546
- GetLineWidth
 - gazebo::rendering::Grid, 391
- GetLinearDamping
 - gazebo::physics::Link, 464
- GetLink
 - gazebo::physics::Collision, 267
 - gazebo::physics::Model, 526
- GetLinkByld
 - gazebo::physics::Model, 527
- GetLinkForce
 - gazebo::physics::BulletJoint, 190
 - gazebo::physics::Joint, 431
 - gazebo::physics::ODEJoint, 597
- GetLinkState
 - gazebo::physics::ModelState, 538, 539
- GetLinkStateCount

- gazebo::physics::ModelState, 539
- GetLinkTorque
 - gazebo::physics::BulletJoint, 190
 - gazebo::physics::Joint, 431
 - gazebo::physics::ODEJoint, 597
- GetLocalAddress
 - gazebo::transport::Connection, 290
- GetLocalAxis
 - gazebo::physics::Joint, 431
- GetLocalHostname
 - gazebo::transport::Connection, 291
- GetLocalPort
 - gazebo::transport::Connection, 291
- GetLocalURI
 - gazebo::transport::Connection, 291
- GetLocallyAdvertised
 - gazebo::transport::Publication, 693
- GetLog
 - gazebo::math::Quaternion, 702
- GetLogPath
 - gazebo::common::SystemPaths, 837
- GetLowStop
 - gazebo::physics::BallJoint, 144
 - gazebo::physics::BulletHinge2Joint, 181
 - gazebo::physics::BulletHingeJoint, 186
 - gazebo::physics::BulletScrewJoint, 215
 - gazebo::physics::BulletSliderJoint, 219
 - gazebo::physics::BulletUniversalJoint, 227
 - gazebo::physics::Joint, 432
 - gazebo::physics::ODEJoint, 597
- GetManager
 - gazebo::rendering::Scene, 754
- GetManifest
 - Common, 37
- GetMass
 - gazebo::physics::BoxShape, 164
 - gazebo::physics::CylinderShape, 310
 - gazebo::physics::Inertial, 418
 - gazebo::physics::Shape, 781
 - gazebo::physics::SphereShape, 807
 - gazebo::physics::TrimeshShape, 867
- GetMaterial
 - gazebo::common::Mesh, 511
- getMaterial
 - gazebo::rendering::MovableText, 546
- GetMaterialCount
 - gazebo::common::Mesh, 512
- GetMaterialIndex
 - gazebo::common::SubMesh, 822
- GetMaterialName
 - gazebo::rendering::Visual, 939
- GetMax
 - gazebo::common::Mesh, 512
 - gazebo::common::SubMesh, 822
- gazebo::math::Vector3, 908
- GetMaxAngle
 - gazebo::physics::MultiRayShape, 553
- GetMaxColor
 - gazebo::common::Image, 410
- GetMaxContacts
 - gazebo::physics::ODEPhysics, 614
 - gazebo::physics::PhysicsEngine, 656
- GetMaxForce
 - gazebo::physics::BulletBallJoint, 168
 - gazebo::physics::BulletHinge2Joint, 181
 - gazebo::physics::BulletHingeJoint, 186
 - gazebo::physics::BulletScrewJoint, 215
 - gazebo::physics::BulletSliderJoint, 219
 - gazebo::physics::BulletUniversalJoint, 227
 - gazebo::physics::Joint, 432
 - gazebo::physics::ODEBallJoint, 575
 - gazebo::physics::ODEHinge2Joint, 589
 - gazebo::physics::ODEHingeJoint, 593
 - gazebo::physics::ODEScrewJoint, 624
 - gazebo::physics::ODESliderJoint, 628
 - gazebo::physics::ODEUniversalJoint, 637
- GetMaxHeight
 - gazebo::physics::HeightmapShape, 402
- GetMaxIndex
 - gazebo::common::SubMesh, 822
- GetMaxRange
 - gazebo::physics::MultiRayShape, 553
- GetMesh
 - gazebo::common::MeshManager, 519
- GetMeshAABB
 - gazebo::common::MeshManager, 520
- GetMin
 - gazebo::common::Mesh, 512
 - gazebo::common::SubMesh, 822
 - gazebo::math::Vector3, 908
- GetMinAngle
 - gazebo::physics::MultiRayShape, 553
- GetMinHeight
 - gazebo::physics::HeightmapShape, 402
- GetMinRange
 - gazebo::physics::MultiRayShape, 553
- GetModel
 - gazebo::physics::Collision, 267
 - gazebo::physics::Link, 464
 - gazebo::physics::World, 959
- GetModelBelowPoint
 - gazebo::physics::World, 959
- GetModelByName
 - gazebo::physics::World, 959
- GetModelCount
 - gazebo::physics::World, 960
- GetModelFile
 - Common, 37

- GetModelName
 - Common, 37
- GetModelPath
 - Common, 37
- GetModelPaths
 - gazebo::common::SystemPaths, 837
- GetModelState
 - gazebo::physics::WorldState, 968
- GetModelStateCount
 - gazebo::physics::WorldState, 968
- GetModelTransform
 - gazebo::common::SkeletonNode, 799
- GetModelVisualAt
 - gazebo::rendering::Scene, 754
- GetModels
 - Common, 38
 - gazebo::physics::World, 960
- GetMovableType
 - gazebo::rendering::DynamicLines, 327
- getMovableType
 - gazebo::rendering::DynamicLines, 327
- GetMsgType
 - gazebo::transport::CallbackHelper, 231
 - gazebo::transport::CallbackHelperT, 232
 - gazebo::transport::DebugCallbackHelper, 312
 - gazebo::transport::Node, 558
 - gazebo::transport::Publication, 693
 - gazebo::transport::PublicationTransport, 695
 - gazebo::transport::Publisher, 696
 - gazebo::transport::SubscribeOptions, 827
- GetName
 - gazebo::common::DiagnosticTimer, 324
 - gazebo::common::Material, 493
 - gazebo::common::Mesh, 512
 - gazebo::common::NodeAnimation, 562
 - gazebo::common::SkeletonAnimation, 793
 - gazebo::common::SkeletonNode, 800
 - gazebo::physics::Base, 152
 - gazebo::physics::State, 814
 - gazebo::physics::World, 960
 - gazebo::rendering::Camera, 243
 - gazebo::rendering::Light, 450
 - gazebo::rendering::Scene, 755
 - gazebo::rendering::Visual, 939
 - gazebo::sensors::Sensor, 768
 - sdf::Element, 336
- GetNearClip
 - gazebo::rendering::Camera, 243
- GetNearestEntityBelow
 - gazebo::physics::Entity, 343
- GetNextElement
 - sdf::Element, 336
- GetNextFrame
 - gazebo::common::Video, 925
- GetNode
 - gazebo::transport::SubscribeOptions, 827
- GetNodeAssignment
 - gazebo::common::SubMesh, 822
- GetNodeAssignmentsCount
 - gazebo::common::SubMesh, 822
- GetNodeByHandle
 - gazebo::common::Skeleton, 787
- GetNodeById
 - gazebo::common::Skeleton, 787
- GetNodeByName
 - gazebo::common::Skeleton, 788
- GetNodeCount
 - gazebo::common::SkeletonAnimation, 793
 - gazebo::transport::Publication, 693
- GetNodePoseAt
 - gazebo::common::SkeletonAnimation, 793
- GetNodes
 - gazebo::common::Skeleton, 788
- GetNormal
 - gazebo::common::SubMesh, 823
 - gazebo::math::Vector3, 908
 - gazebo::physics::PlaneShape, 673
- GetNormalCount
 - gazebo::common::Mesh, 512
 - gazebo::common::SubMesh, 823
- GetNormalMap
 - gazebo::rendering::Visual, 939
- GetNumAnimations
 - gazebo::common::Skeleton, 788
- GetNumJoints
 - gazebo::common::Skeleton, 788
- GetNumNodes
 - gazebo::common::Skeleton, 788
- GetNumPoints
 - gazebo::math::RotationSpline, 740
- GetNumRawTrans
 - gazebo::common::SkeletonNode, 800
- GetNumVertNodeWeights
 - gazebo::common::Skeleton, 788
- GetODEId
 - gazebo::physics::ODELink, 604
- GetOgreCamera
 - gazebo::rendering::Camera, 243
- GetOgrePaths
 - gazebo::common::SystemPaths, 837
- GetOperationType
 - gazebo::rendering::DynamicRenderable, 330
- GetOutgoingCount
 - gazebo::transport::Publisher, 696
- GetParam
 - gazebo::physics::ODEHinge2Joint, 589
 - gazebo::physics::ODEHingeJoint, 593
 - gazebo::physics::ODEJoint, 598

- gazebo::physics::ODEScrewJoint, 624
- gazebo::physics::ODESliderJoint, 628
- GetParent
 - gazebo::common::SkeletonNode, 800
 - gazebo::physics::Base, 152
 - gazebo::physics::Joint, 432
 - gazebo::rendering::Projector, 690
 - gazebo::rendering::Visual, 939
 - sdf::Element, 336
- GetParentId
 - gazebo::physics::Base, 152
- GetParentJointsLinks
 - gazebo::physics::Link, 464
- GetParentModel
 - gazebo::physics::Entity, 343
- GetParentName
 - gazebo::sensors::Sensor, 768
- GetPath
 - gazebo::common::Mesh, 512
- GetPauseTime
 - gazebo::physics::World, 960
- GetPerpendicular
 - gazebo::math::Vector3, 908
- GetPhysicsEngine
 - gazebo::physics::World, 960
- GetPhysicsUpdateMutex
 - gazebo::physics::PhysicsEngine, 657
- GetPitch
 - gazebo::common::Image, 410
 - gazebo::math::Quaternion, 703
- GetPixel
 - gazebo::common::Image, 410
- GetPixelFormat
 - gazebo::common::Image, 410
- GetPluginPaths
 - gazebo::common::SystemPaths, 837
- GetPoint
 - gazebo::math::RotationSpline, 740
 - gazebo::math::Spline, 810
 - gazebo::rendering::DynamicLines, 327
- GetPointCount
 - gazebo::math::Spline, 810
 - gazebo::rendering::DynamicLines, 327
- GetPointSize
 - gazebo::common::Material, 493
- GetPos
 - gazebo::physics::HeightmapShape, 402
- GetPose
 - gazebo::physics::CollisionState, 273
 - gazebo::physics::Inertial, 418
 - gazebo::physics::LinkState, 476
 - gazebo::physics::ModelState, 539
 - gazebo::rendering::Visual, 939
 - gazebo::sensors::Sensor, 768
- GetPoseAt
 - gazebo::common::SkeletonAnimation, 793
- GetPoseAtX
 - gazebo::common::SkeletonAnimation, 794
- GetPosition
 - gazebo::rendering::Light, 451
 - gazebo::rendering::Visual, 940
- GetPrevMsg
 - gazebo::transport::Publisher, 696
- GetPrimitiveType
 - gazebo::common::SubMesh, 823
- GetPrincipalMoments
 - gazebo::physics::Inertial, 418
- GetProductsofInertia
 - gazebo::physics::Inertial, 418
- GetRGBData
 - gazebo::common::Image, 410
- GetRadius
 - gazebo::physics::CylinderShape, 310
 - gazebo::physics::SphereShape, 808
- GetRange
 - gazebo::physics::BulletRaySensor, 210
 - gazebo::physics::MultiRayShape, 553
 - gazebo::sensors::GpuRaySensor, 383
 - gazebo::sensors::RaySensor, 714
- GetRangeCount
 - gazebo::sensors::GpuRaySensor, 384
 - gazebo::sensors::RaySensor, 715
- GetRangeMax
 - gazebo::sensors::GpuRaySensor, 384
 - gazebo::sensors::RaySensor, 715
- GetRangeMin
 - gazebo::sensors::GpuRaySensor, 384
 - gazebo::sensors::RaySensor, 715
- GetRangeResolution
 - gazebo::sensors::GpuRaySensor, 384
 - gazebo::sensors::RaySensor, 715
- GetRanges
 - gazebo::sensors::GpuRaySensor, 385
 - gazebo::sensors::RaySensor, 715
- GetRawTransform
 - gazebo::common::SkeletonNode, 800
- GetRawTransforms
 - gazebo::common::SkeletonNode, 800
- GetRayCount
 - gazebo::sensors::GpuRaySensor, 385
 - gazebo::sensors::RaySensor, 716
- GetRealTime
 - gazebo::physics::State, 814
 - gazebo::physics::World, 960
- GetRelativeAngularAccel
 - gazebo::physics::Collision, 267
 - gazebo::physics::Entity, 343
 - gazebo::physics::Link, 464

- gazebo::physics::Model, 527
- GetRelativeAngularVel
 - gazebo::physics::Collision, 267
 - gazebo::physics::Entity, 343
 - gazebo::physics::Link, 465
 - gazebo::physics::Model, 527
- GetRelativeForce
 - gazebo::physics::Link, 465
- GetRelativeLinearAccel
 - gazebo::physics::Collision, 268
 - gazebo::physics::Entity, 343
 - gazebo::physics::Link, 465
 - gazebo::physics::Model, 527
- GetRelativeLinearVel
 - gazebo::physics::Collision, 268
 - gazebo::physics::Entity, 344
 - gazebo::physics::Link, 465
 - gazebo::physics::Model, 527
- GetRelativePoints
 - gazebo::physics::BulletRaySensor, 210
 - gazebo::physics::RayShape, 721
- GetRelativePose
 - gazebo::physics::Entity, 344
- GetRelativeTorque
 - gazebo::physics::Link, 465
- GetRemoteAddress
 - gazebo::transport::Connection, 291
- GetRemoteHostname
 - gazebo::transport::Connection, 291
- GetRemotePort
 - gazebo::transport::Connection, 291
- GetRemoteSubscriptionCount
 - gazebo::transport::Publication, 693
- GetRemoteURI
 - gazebo::transport::Connection, 291
- getRenderOperation
 - gazebo::rendering::MovableText, 546
- GetRenderPathType
 - gazebo::rendering::RenderEngine, 725
- GetRenderTexture
 - gazebo::rendering::Camera, 244
- GetRequired
 - sdf::Element, 336
 - sdf::Param, 645
- GetResRange
 - gazebo::physics::MultiRayShape, 553
- GetRetro
 - gazebo::physics::BulletRaySensor, 210
 - gazebo::physics::MultiRayShape, 554
 - gazebo::physics::RayShape, 721
 - gazebo::sensors::GpuRaySensor, 385
 - gazebo::sensors::RaySensor, 716
- GetRight
 - gazebo::rendering::Camera, 244
- GetRoll
 - gazebo::math::Quaternion, 703
- GetRootNode
 - gazebo::common::Skeleton, 789
- GetRootVisual
 - gazebo::rendering::Visual, 940
- GetRotation
 - gazebo::common::PoseKeyFrame, 688
 - gazebo::math::Matrix4, 504
 - gazebo::rendering::Visual, 940
- GetRounded
 - gazebo::math::Vector3, 908
- GetSDF
 - gazebo::physics::Actor, 124
 - gazebo::physics::Base, 153
 - gazebo::physics::Model, 527
 - gazebo::physics::WorldState, 969
- GetSID
 - gazebo::common::NodeTransform, 567
- GetSORPGSIters
 - gazebo::physics::ODEPhysics, 614
 - gazebo::physics::PhysicsEngine, 657
- GetSORPGSPreconIters
 - gazebo::physics::ODEPhysics, 614
 - gazebo::physics::PhysicsEngine, 657
- GetSORPGSW
 - gazebo::physics::ODEPhysics, 614
 - gazebo::physics::PhysicsEngine, 657
- GetSampleCount
 - gazebo::physics::MultiRayShape, 554
- GetSaveable
 - gazebo::physics::Base, 152
- GetScale
 - gazebo::physics::MapShape, 486
 - gazebo::rendering::Visual, 940
- GetScanResolution
 - gazebo::physics::MultiRayShape, 554
- GetScene
 - gazebo::rendering::Camera, 244
 - gazebo::rendering::RenderEngine, 725
 - gazebo::rendering::Visual, 940
- GetSceneCount
 - gazebo::rendering::RenderEngine, 725
- GetSceneNode
 - gazebo::rendering::Camera, 244
 - gazebo::rendering::Grid, 391
 - gazebo::rendering::Visual, 940
- GetScopedName
 - gazebo::physics::Base, 152
 - gazebo::sensors::Sensor, 769
- GetSelectedEntity
 - gazebo::physics::World, 961
- GetSelectedVisual
 - gazebo::rendering::Scene, 755

- GetSelfCollide
 - gazebo::physics::Link, 466
- GetSensor
 - gazebo::sensors::SensorManager, 775
- GetSensorCount
 - gazebo::physics::Link, 466
- GetSensorName
 - gazebo::physics::Link, 466
- GetSensorTypes
 - gazebo::sensors::SensorFactory, 773
 - gazebo::sensors::SensorManager, 775
- GetSet
 - sdf::Param, 645
- GetSetWorldPoseMutex
 - gazebo::physics::World, 961
- GetShadeMode
 - gazebo::common::Material, 493
- GetShaderType
 - gazebo::rendering::Visual, 941
- GetShadowsEnabled
 - gazebo::rendering::Scene, 755
- GetShape
 - gazebo::physics::Collision, 268
- GetShapeType
 - gazebo::physics::Collision, 268
- GetShininess
 - gazebo::common::Material, 493
- GetShowOnTop
 - gazebo::rendering::MovableText, 546
- GetSimTime
 - gazebo::physics::State, 815
 - gazebo::physics::World, 961
- GetSize
 - gazebo::math::Box, 159
 - gazebo::physics::BoxShape, 165
 - gazebo::physics::HeightmapShape, 402
 - gazebo::physics::PlaneShape, 673
 - gazebo::physics::TrimeshShape, 868
- GetSkeleton
 - gazebo::common::Mesh, 512
- GetSpaceId
 - gazebo::physics::ODECollision, 581
 - gazebo::physics::ODELink, 604
 - gazebo::physics::ODEPhysics, 614
- GetSpaceWidth
 - gazebo::rendering::MovableText, 546
- GetSpecular
 - gazebo::common::Material, 494
- GetSpecularColor
 - gazebo::rendering::Light, 451
- GetSquaredLength
 - gazebo::math::Vector3, 908
 - gazebo::math::Vector4, 918
- getSquaredViewDepth
 - gazebo::rendering::DynamicRenderable, 330
 - gazebo::rendering::MovableText, 546
- GetStartTime
 - gazebo::physics::World, 961
- GetState
 - gazebo::physics::Collision, 268
 - gazebo::physics::Joint, 432
 - gazebo::physics::Link, 466
 - gazebo::physics::Model, 528
 - gazebo::physics::World, 961
- GetStepTime
 - gazebo::physics::BulletPhysics, 206
 - gazebo::physics::ODEPhysics, 614
 - gazebo::physics::PhysicsEngine, 657
- GetStepType
 - gazebo::physics::ODEPhysics, 614
- GetSubMesh
 - gazebo::common::Mesh, 513
- GetSubMeshCount
 - gazebo::common::Mesh, 513
- GetSum
 - gazebo::math::Vector3, 909
- GetSurface
 - gazebo::physics::Collision, 269
- GetTagPose
 - gazebo::sensors::RFIDTag, 730
- GetTags
 - gazebo::sensors::RFIDTagManager, 732
- GetTangent
 - gazebo::math::Spline, 810
- GetTension
 - gazebo::math::Spline, 810
- GetTexCoord
 - gazebo::common::SubMesh, 823
- GetTexCoordCount
 - gazebo::common::Mesh, 513
 - gazebo::common::SubMesh, 823
- GetText
 - gazebo::rendering::MovableText, 546
- GetTextureHeight
 - gazebo::rendering::Camera, 244
- GetTextureImage
 - gazebo::common::Material, 494
- GetTextureWidth
 - gazebo::rendering::Camera, 244
- GetThreshold
 - gazebo::physics::MapShape, 486
- GetTime
 - gazebo::common::Animation, 138
 - gazebo::common::DiagnosticManager, 322
 - gazebo::common::KeyFrame, 446
- GetTimeAtX
 - gazebo::common::NodeAnimation, 562
- GetTimerCount

- gazebo::common::DiagnosticManager, 322
- GetTopic
 - gazebo::sensors::CameraSensor, 258
 - gazebo::sensors::RaySensor, 716
 - gazebo::sensors::Sensor, 769
 - gazebo::transport::PublicationTransport, 695
 - gazebo::transport::Publisher, 696
 - gazebo::transport::SubscribeOptions, 827
 - gazebo::transport::Subscriber, 828
- GetTopicNamespace
 - gazebo::transport::Node, 558
- GetTopicNamespaces
 - gazebo::transport::ConnectionManager, 294
 - gazebo::transport::TopicManager, 864
- GetTransform
 - gazebo::common::SkeletonNode, 801
- GetTransforms
 - gazebo::common::SkeletonNode, 801
- GetTranslation
 - gazebo::common::PoseKeyFrame, 688
 - gazebo::math::Matrix4, 504
- GetTransparency
 - gazebo::common::Material, 494
 - gazebo::rendering::Visual, 941
- GetTransportCount
 - gazebo::transport::Publication, 693
- GetTriangleCount
 - gazebo::rendering::Camera, 245
 - gazebo::rendering::UserCamera, 881
 - gazebo::rendering::WindowManager, 952
- GetType
 - gazebo::common::NodeTransform, 567
 - gazebo::physics::Base, 153
 - gazebo::PluginT, 676
 - gazebo::rendering::Light, 451
 - gazebo::sensors::Sensor, 769
- GetTypeName
 - sdf::Param, 645
- GetTypeString
 - gazebo::rendering::FPSViewController, 368
 - gazebo::rendering::OrbitViewController, 640
 - gazebo::rendering::ViewController, 929
- GetURI
 - Common, 38
 - gazebo::physics::HeightmapShape, 402
 - gazebo::physics::MapShape, 486
- GetUp
 - gazebo::rendering::Camera, 245
- GetUpdateMutex
 - gazebo::sensors::ContactSensor, 303
- GetUpdatePeriod
 - gazebo::physics::PhysicsEngine, 658
- GetUpdateRate
 - gazebo::physics::PhysicsEngine, 658
- GetUserCamera
 - gazebo::rendering::Scene, 755
- GetUserCameraCount
 - gazebo::rendering::Scene, 756
- GetVAngle
 - gazebo::sensors::GpuRaySensor, 385
- GetVFOV
 - gazebo::rendering::Camera, 245
 - gazebo::sensors::GpuRaySensor, 386
- GetValue
 - gazebo::common::NumericKeyFrame, 572
 - sdf::Element, 336
 - sdf::ParamT, 650
- GetValueBool
 - sdf::Element, 336
- GetValueChar
 - sdf::Element, 336
- GetValueColor
 - sdf::Element, 336
- GetValueDouble
 - sdf::Element, 336
- GetValueFloat
 - sdf::Element, 336
- GetValueInt
 - sdf::Element, 336
- GetValuePose
 - sdf::Element, 336
- GetValueQuaternion
 - sdf::Element, 336
- GetValueString
 - sdf::Element, 336
- GetValueTime
 - sdf::Element, 336
- GetValueUInt
 - sdf::Element, 336
- GetValueVector2d
 - sdf::Element, 336
- GetValueVector3
 - sdf::Element, 336
- GetVelocity
 - gazebo::physics::BulletBallJoint, 168
 - gazebo::physics::BulletHinge2Joint, 182
 - gazebo::physics::BulletHingeJoint, 187
 - gazebo::physics::BulletScrewJoint, 215
 - gazebo::physics::BulletSliderJoint, 219
 - gazebo::physics::BulletUniversalJoint, 227
 - gazebo::physics::Joint, 433
 - gazebo::physics::LinkState, 477
 - gazebo::physics::ODEBallJoint, 576
 - gazebo::physics::ODEHinge2Joint, 589
 - gazebo::physics::ODEHingeJoint, 593
 - gazebo::physics::ODEScrewJoint, 624
 - gazebo::physics::ODESliderJoint, 628
 - gazebo::physics::ODEUniversalJoint, 637

- gazebo::sensors::ImuSensor, 413
- GetVertNodeWeight
 - gazebo::common::Skeleton, 789
- GetVertex
 - gazebo::common::SubMesh, 823
- GetVertexCount
 - gazebo::common::Mesh, 513
 - gazebo::common::SubMesh, 824
 - gazebo::physics::HeightmapShape, 403
- GetVertexIndex
 - gazebo::common::SubMesh, 824
- GetVerticalAngleMax
 - gazebo::sensors::GpuRaySensor, 385
 - gazebo::sensors::RaySensor, 716
- GetVerticalAngleMin
 - gazebo::sensors::GpuRaySensor, 386
 - gazebo::sensors::RaySensor, 716
- GetVerticalMaxAngle
 - gazebo::physics::MultiRayShape, 554
- GetVerticalMinAngle
 - gazebo::physics::MultiRayShape, 554
- GetVerticalRangeCount
 - gazebo::sensors::GpuRaySensor, 386
 - gazebo::sensors::RaySensor, 717
- GetVerticalRayCount
 - gazebo::sensors::GpuRaySensor, 386
 - gazebo::sensors::RaySensor, 717
- GetVerticalSampleCount
 - gazebo::physics::MultiRayShape, 554
- GetVerticalScanResolution
 - gazebo::physics::MultiRayShape, 555
- GetViewport
 - gazebo::rendering::Camera, 245
- GetViewportHeight
 - gazebo::rendering::Camera, 245
- GetViewportWidth
 - gazebo::rendering::Camera, 245
- GetVisibilityFlags
 - gazebo::rendering::Visual, 941
- GetVisible
 - gazebo::rendering::Visual, 941
- GetVisual
 - gazebo::rendering::Scene, 756
 - gazebo::rendering::UserCamera, 882
- GetVisualAt
 - gazebo::rendering::Scene, 756
- GetVisualBelow
 - gazebo::rendering::Scene, 756
- GetVisualName
 - gazebo::rendering::SelectionObj, 765
- GetVisualize
 - gazebo::sensors::Sensor, 769
- GetVisualsBelowPoint
 - gazebo::rendering::Scene, 757
- GetWallTime
 - gazebo::common::Time, 845
 - gazebo::physics::State, 815
- GetWidth
 - gazebo::common::Image, 411
 - gazebo::common::Video, 925
- GetWindow
 - gazebo::rendering::WindowManager, 953
- GetWindowId
 - gazebo::rendering::Camera, 246
- GetWorld
 - gazebo::physics::Base, 153
- GetWorldAngularAccel
 - gazebo::physics::Collision, 269
 - gazebo::physics::Entity, 344
 - gazebo::physics::Link, 467
 - gazebo::physics::Model, 528
- GetWorldAngularVel
 - gazebo::physics::BulletLink, 195
 - gazebo::physics::Collision, 269
 - gazebo::physics::Entity, 344
 - gazebo::physics::Model, 528
 - gazebo::physics::ODELink, 604
- GetWorldCFM
 - gazebo::physics::ODEPhysics, 615
 - gazebo::physics::PhysicsEngine, 658
- GetWorldERP
 - gazebo::physics::ODEPhysics, 615
 - gazebo::physics::PhysicsEngine, 658
- GetWorldForce
 - gazebo::physics::BulletLink, 195
 - gazebo::physics::Link, 467
 - gazebo::physics::ODELink, 604
- GetWorldId
 - gazebo::physics::ODEPhysics, 615
- GetWorldLinearAccel
 - gazebo::physics::Collision, 269
 - gazebo::physics::Entity, 344
 - gazebo::physics::Link, 467
 - gazebo::physics::Model, 528
- GetWorldLinearVel
 - gazebo::physics::BulletLink, 195
 - gazebo::physics::Collision, 269
 - gazebo::physics::Entity, 345
 - gazebo::physics::Model, 528
 - gazebo::physics::ODELink, 604
- GetWorldName
 - gazebo::sensors::Sensor, 769
- GetWorldPathExtension
 - gazebo::common::SystemPaths, 837
- GetWorldPointOnPlane
 - gazebo::rendering::Camera, 246
- GetWorldPose
 - gazebo::physics::BulletMotionState, 199

- gazebo::physics::Entity, 345
- gazebo::rendering::Camera, 246
- gazebo::rendering::Visual, 941
- GetWorldPosition
 - gazebo::rendering::Camera, 246
- GetWorldRotation
 - gazebo::rendering::Camera, 246
- GetWorldTorque
 - gazebo::physics::BulletLink, 196
 - gazebo::physics::Link, 467
 - gazebo::physics::ODELink, 604
- getWorldTransform
 - gazebo::physics::BulletMotionState, 199
- getWorldTransforms
 - gazebo::rendering::MovableText, 547
- GetWorldVisual
 - gazebo::rendering::Scene, 757
- GetXAxis
 - gazebo::math::Quaternion, 703
- GetXLength
 - gazebo::math::Box, 159
- GetYAxis
 - gazebo::math::Quaternion, 703
- GetYLength
 - gazebo::math::Box, 160
- GetYaw
 - gazebo::math::Quaternion, 703
- GetZAxis
 - gazebo::math::Quaternion, 703
- GetZLength
 - gazebo::math::Box, 160
- GetZValue
 - gazebo::rendering::Camera, 247
- globalEndPos
 - gazebo::physics::RayShape, 722
- globalStartPos
 - gazebo::physics::RayShape, 722
- google, 117
- google::protobuf, 117
- google::protobuf::compiler, 117
- google::protobuf::compiler::cpp, 118
- google::protobuf::compiler::cpp::GazeboGenerator, 374
 - ~GazeboGenerator, 374
 - GazeboGenerator, 374
 - Generate, 374
- GpuLaser
 - gazebo::rendering::GpuLaser, 376
- GpuLaser.hh, 1032
- GpuLaserPtr
 - gazebo::rendering, 112
- GpuRaySensor
 - gazebo::sensors::GpuRaySensor, 381
- GpuRaySensor.hh, 1033
- GpuRaySensor_V
 - gazebo::sensors, 115
- GpuRaySensorPtr
 - gazebo::sensors, 115
- gravity
 - urdf2gazebo::GazeboExtension, 371
- Green
 - gazebo::common::Color, 284
- Grid
 - gazebo::rendering::Grid, 390
- Grid.hh, 1034
- Gripper
 - gazebo::physics::Gripper, 393
- Gripper.hh, 1034
- gzclr_end
 - Common, 34
- gzclr_start
 - Common, 34
- gzdbg
 - Common, 34
- gzerr
 - Common, 34
- gzlog
 - Common, 35
- gzmsg
 - Common, 35
- gzthrow
 - Common, 35
- gzwarn
 - Common, 35
- H_CENTER
 - gazebo::rendering::MovableText, 545
- H_LEFT
 - gazebo::rendering::MovableText, 545
- HEADER_LENGTH
 - Connection.hh, 1012
- HEIGHTMAP_SHAPE
 - gazebo::physics::Base, 149
- HI_STOP
 - gazebo::physics::Joint, 426
- HINGE2_JOINT
 - gazebo::physics::Base, 149
- HINGE_JOINT
 - gazebo::physics::Base, 149
- handle
 - gazebo::common::SkeletonNode, 803
 - gazebo::PluginT, 677
- HandleData
 - gazebo::transport::CallbackHelper, 231
 - gazebo::transport::CallbackHelperT, 232
 - gazebo::transport::DebugCallbackHelper, 312
 - gazebo::transport::Node, 559
 - gazebo::transport::SubscriptionTransport, 829
- HandleKeyPressEvent

- gazebo::rendering::FPSViewController, 368
- gazebo::rendering::GUIOverlay, 396
- gazebo::rendering::OrbitViewController, 640
- gazebo::rendering::UserCamera, 882
- gazebo::rendering::ViewController, 929
- HandleKeyReleaseEvent
 - gazebo::rendering::FPSViewController, 368
 - gazebo::rendering::GUIOverlay, 396
 - gazebo::rendering::OrbitViewController, 640
 - gazebo::rendering::UserCamera, 882
 - gazebo::rendering::ViewController, 929
- HandleMouseEvent
 - gazebo::rendering::FPSViewController, 369
 - gazebo::rendering::GUIOverlay, 396
 - gazebo::rendering::OrbitViewController, 640
 - gazebo::rendering::UserCamera, 882
 - gazebo::rendering::ViewController, 929
- hang
 - gazebo::sensors::GpuRaySensor, 388
- HasAttachedObject
 - gazebo::rendering::Visual, 942
- HasAttribute
 - sdf::Element, 336
- HasConnections
 - gazebo::transport::Publisher, 696
- HasElement
 - sdf::Element, 336
- HasElementDescription
 - sdf::Element, 337
- HasMesh
 - gazebo::common::MeshManager, 520
- HasModel
 - Common, 38
- HasNode
 - gazebo::common::SkeletonAnimation, 794
- HasSkeleton
 - gazebo::common::Mesh, 513
- HasTransport
 - gazebo::transport::Publication, 693
- HasType
 - gazebo::physics::Base, 153
- HasVertex
 - gazebo::common::SubMesh, 824
- height_1st
 - gazebo::sensors::GpuRaySensor, 388
- height_2nd
 - gazebo::sensors::GpuRaySensor, 388
- Heightmap
 - gazebo::rendering::Heightmap, 398
- Heightmap.hh, 1036
- HeightmapShape
 - gazebo::physics::HeightmapShape, 401
- HeightmapShape.hh, 1037
- HeightmapShapePtr
 - gazebo::physics, 109
- heights
 - gazebo::physics::HeightmapShape, 403
- Helpers.hh, 1038
 - GZ_DBL_MAX, 1040
 - GZ_DBL_MIN, 1040
 - GZ_FLT_MAX, 1040
 - GZ_FLT_MIN, 1040
- hfov
 - gazebo::sensors::GpuRaySensor, 388
- Hide
 - gazebo::rendering::GUIOverlay, 397
- Hinge2Joint
 - gazebo::physics::Hinge2Joint, 405
- Hinge2Joint.hh, 1040
- HingeJoint
 - gazebo::physics::HingeJoint, 406
- HingeJoint.hh, 1041
- HorizAlign
 - gazebo::rendering::MovableText, 545
- horzElem
 - gazebo::physics::MultiRayShape, 555
 - gazebo::sensors::GpuRaySensor, 388
- IDENTITY
 - gazebo::math::Matrix4, 508
- IOManager
 - gazebo::transport::IOManager, 423
- IOManager.hh, 1045
- id
 - gazebo::common::SkeletonNode, 803
 - gazebo::physics::TrajectoryInfo, 865
 - gazebo::transport::Connection, 292
- Image
 - gazebo::common::Image, 409
- Image.hh, 1042
- imageFormat
 - gazebo::rendering::Camera, 254
- imageHeight
 - gazebo::rendering::Camera, 254
- imageWidth
 - gazebo::rendering::Camera, 254
- img
 - gazebo::physics::HeightmapShape, 403
- ImuSensor
 - gazebo::sensors::ImuSensor, 413
- ImuSensor.hh, 1043
- IncCount
 - gazebo::transport::IOManager, 423
- indexBufferCapacity
 - gazebo::rendering::DynamicRenderable, 331
- Inertial
 - gazebo::physics::Inertial, 416
- inertial

- gazebo::physics::Link, 473
- Inertial.hh, 1044
- InertialPtr
 - gazebo::physics, 109
- Init
 - gazebo::common::PID, 666
 - gazebo::Master, 488
 - gazebo::ModelPlugin, 535
 - gazebo::physics::Actor, 124
 - gazebo::physics::Base, 153
 - gazebo::physics::BoxShape, 165
 - gazebo::physics::BulletHeightmapShape, 178
 - gazebo::physics::BulletLink, 196
 - gazebo::physics::BulletPhysics, 206
 - gazebo::physics::BulletTrimeshShape, 224
 - gazebo::physics::Collision, 269
 - gazebo::physics::CylinderShape, 310
 - gazebo::physics::Gripper, 393
 - gazebo::physics::HeightmapShape, 403
 - gazebo::physics::HingeJoint, 407
 - gazebo::physics::Joint, 433
 - gazebo::physics::Link, 467
 - gazebo::physics::MapShape, 486
 - gazebo::physics::Model, 529
 - gazebo::physics::MultiRayShape, 555
 - gazebo::physics::ODEHeightmapShape, 586
 - gazebo::physics::ODELink, 604
 - gazebo::physics::ODEPhysics, 615
 - gazebo::physics::ODETrimeshShape, 634
 - gazebo::physics::PhysicsEngine, 658
 - gazebo::physics::PlaneShape, 673
 - gazebo::physics::RayShape, 721
 - gazebo::physics::Road, 737
 - gazebo::physics::Shape, 781
 - gazebo::physics::SphereShape, 808
 - gazebo::physics::TrimeshShape, 868
 - gazebo::physics::World, 961
 - gazebo::rendering::Camera, 247
 - gazebo::rendering::DepthCamera, 316
 - gazebo::rendering::DynamicRenderable, 330
 - gazebo::rendering::FPSViewController, 369
 - gazebo::rendering::GpuLaser, 377
 - gazebo::rendering::Grid, 391
 - gazebo::rendering::GUIOverlay, 397
 - gazebo::rendering::OrbitViewController, 640, 641
 - gazebo::rendering::RenderEngine, 726
 - gazebo::rendering::RTShaderSystem, 745
 - gazebo::rendering::Scene, 757
 - gazebo::rendering::SelectionObj, 765
 - gazebo::rendering::UserCamera, 883
 - gazebo::rendering::ViewController, 929, 930
 - gazebo::rendering::Visual, 942
 - gazebo::SensorPlugin, 778
 - gazebo::sensors::CameraSensor, 258
 - gazebo::sensors::ContactSensor, 303
 - gazebo::sensors::DepthCameraSensor, 318
 - gazebo::sensors::GpuRaySensor, 386
 - gazebo::sensors::RaySensor, 717
 - gazebo::sensors::RFIDSensor, 728
 - gazebo::sensors::RFIDTag, 731
 - gazebo::sensors::Sensor, 769
 - gazebo::sensors::SensorManager, 776
 - gazebo::Server, 779
 - gazebo::SystemPlugin, 839
 - gazebo::transport::ConnectionManager, 294
 - gazebo::transport::Node, 559
 - gazebo::transport::PublicationTransport, 695
 - gazebo::transport::SubscribeOptions, 827
 - gazebo::transport::SubscriptionTransport, 829
 - gazebo::transport::TopicManager, 864
 - gazebo::VisualPlugin, 950
 - gazebo::WorldPlugin, 966
 - Messages, 64
- init
 - gazebo, 95
 - Rendering, 81
 - sdf, 119
 - Sensors, 86
 - Transport, 90
- init_world
 - Classes for physics and dynamics, 72
- init_worlds
 - Classes for physics and dynamics, 72
- InitChild
 - gazebo::sensors::ImuSensor, 414
- initDoc
 - sdf, 119
- initFile
 - sdf, 119
- InitForThread
 - gazebo::physics::BulletPhysics, 206
 - gazebo::physics::ODEPhysics, 615
 - gazebo::physics::PhysicsEngine, 659
- initModelDoc
 - urdf2gazebo::URDF2Gazebo, 874
- initModelFile
 - urdf2gazebo::URDF2Gazebo, 874
- initModelString
 - urdf2gazebo::URDF2Gazebo, 874
- initString
 - sdf, 119
- initXml
 - sdf, 119
- initial_joint_position
 - urdf2gazebo::GazeboExtension, 371
- initialTransform
 - gazebo::common::SkeletonNode, 803
- initialized

- gazebo::rendering::Camera, 254
- InsertElement
 - sdf::Element, 337
- insertGazeboExtensionCollision
 - urdf2gazebo::URDF2Gazebo, 874
- insertGazeboExtensionJoint
 - urdf2gazebo::URDF2Gazebo, 874
- insertGazeboExtensionLink
 - urdf2gazebo::URDF2Gazebo, 874
- insertGazeboExtensionRobot
 - urdf2gazebo::URDF2Gazebo, 874
- insertGazeboExtensionVisual
 - urdf2gazebo::URDF2Gazebo, 874
- InsertLatchedMsg
 - gazebo::transport::Node, 559
- InsertMesh
 - gazebo::rendering::Visual, 942
- InsertModelFile
 - gazebo::physics::World, 962
- InsertModelSDF
 - gazebo::physics::World, 962
- InsertModelString
 - gazebo::physics::World, 962
- Instance
 - gazebo::common::Console, 296
 - SingletonT, 784
- Interpolate
 - gazebo::math::RotationSpline, 740, 741
 - gazebo::math::Spline, 811
- interpolateX
 - gazebo::physics::Actor, 125
- invBindTransform
 - gazebo::common::SkeletonNode, 803
- Inverse
 - gazebo::math::Matrix4, 504
- inverseTransformToParentFrame
 - urdf2gazebo::URDF2Gazebo, 875
- Invert
 - gazebo::math::Quaternion, 704
- is_damping_factor
 - urdf2gazebo::GazeboExtension, 371
- is_fudge_factor
 - urdf2gazebo::GazeboExtension, 371
- is_initial_joint_position
 - urdf2gazebo::GazeboExtension, 371
- is_kd
 - urdf2gazebo::GazeboExtension, 371
- is_kp
 - urdf2gazebo::GazeboExtension, 371
- is_laser_retro
 - urdf2gazebo::GazeboExtension, 371
- is_maxVel
 - urdf2gazebo::GazeboExtension, 371
- is_minDepth
 - urdf2gazebo::GazeboExtension, 372
- is_mu1
 - urdf2gazebo::GazeboExtension, 372
- is_mu2
 - urdf2gazebo::GazeboExtension, 372
- is_stop_cfm
 - urdf2gazebo::GazeboExtension, 372
- is_stop_erp
 - urdf2gazebo::GazeboExtension, 372
- is_stopped
 - Transport, 90
- IsActive
 - gazebo::physics::Actor, 124
 - gazebo::rendering::SelectionObj, 765
 - gazebo::sensors::Sensor, 770
- IsAdvertised
 - gazebo::transport::TopicManager, 864
- IsAffine
 - gazebo::math::Matrix4, 505
- IsBool
 - sdf::Param, 645
- IsCanonicalLink
 - gazebo::physics::Entity, 345
- IsChar
 - sdf::Param, 645
- IsColor
 - sdf::Param, 645
- IsDouble
 - sdf::Param, 646
- IsFinite
 - gazebo::math::Pose, 681
 - gazebo::math::Quaternion, 704
 - gazebo::math::Vector2d, 888
 - gazebo::math::Vector2i, 896
 - gazebo::math::Vector3, 909
 - gazebo::math::Vector4, 918
- IsFloat
 - sdf::Param, 646
- IsHorizontal
 - gazebo::sensors::GpuRaySensor, 386
- isHorizontal
 - gazebo::sensors::GpuRaySensor, 388
- IsInitialized
 - gazebo::rendering::Camera, 247
 - gazebo::rendering::GUIOverlay, 397
- IsInt
 - sdf::Param, 646
- IsJoint
 - gazebo::common::SkeletonNode, 801
- IsLocal
 - gazebo::transport::CallbackHelper, 231
 - gazebo::transport::CallbackHelperT, 233
 - gazebo::transport::DebugCallbackHelper, 312
 - gazebo::transport::SubscriptionTransport, 829

- IsOpen
 - gazebo::transport::Connection, 291
- IsPaused
 - gazebo::physics::World, 962
- IsPlaceable
 - gazebo::physics::Collision, 270
- IsPlane
 - gazebo::rendering::Visual, 942
- IsPose
 - sdf::Param, 646
- isPowerOfTwo
 - Math, 56
- IsQuaternion
 - sdf::Param, 646
- IsRootNode
 - gazebo::common::SkeletonNode, 801
- IsRunning
 - gazebo::transport::ConnectionManager, 294
- IsSelected
 - gazebo::physics::Base, 154
- IsStatic
 - gazebo::physics::Entity, 345
 - gazebo::rendering::Visual, 942
- IsStr
 - sdf::Param, 646
- IsTime
 - sdf::Param, 646
- IsUInt
 - sdf::Param, 646
- IsValidFilename
 - gazebo::common::MeshManager, 520
- IsVector2d
 - sdf::Param, 646
- IsVector2i
 - sdf::Param, 646
- IsVector3
 - sdf::Param, 646
- IsVisible
 - gazebo::rendering::Camera, 247
- isnan
 - Math, 55
- JOINT
 - gazebo::common::SkeletonNode, 797
 - gazebo::physics::Base, 149
- Joint
 - gazebo::physics::Joint, 427
- Joint.hh, 1046
- Joint_V
 - gazebo::physics, 109
- JointController
 - gazebo::physics::JointController, 439
- JointController.hh, 1047
- JointFeedback.hh, 1048
- jointId
 - gazebo::physics::ODEJoint, 599
- JointPtr
 - gazebo::physics, 109
- JointState
 - gazebo::physics::JointState, 442
- JointState.hh, 1049
- JointVisual
 - gazebo::rendering::JointVisual, 444
- JointVisual.hh, 1050
- JointVisualPtr
 - gazebo::rendering, 112
- kd
 - gazebo::physics::SurfaceParams, 832
 - urdf2gazebo::GazeboExtension, 372
- key
 - sdf::Param, 647
- KeyFrame
 - gazebo::common::KeyFrame, 446
- KeyFrame.hh, 1051
- KeyFrame_V
 - gazebo::common::Animation, 136
- keyFrames
 - gazebo::common::Animation, 138
 - gazebo::common::NodeAnimation, 563
- kp
 - gazebo::physics::SurfaceParams, 832
 - urdf2gazebo::GazeboExtension, 372
- L_INT16
 - gazebo::common::Image, 408
- L_INT8
 - gazebo::common::Image, 408
- LEFT
 - gazebo::common::MouseEvent, 541
- LIGHT
 - gazebo::physics::Base, 149
- LINE_MAX_LEN
 - STLLoader.hh, 1161
- LINES
 - gazebo::common::SubMesh, 819
- LINESTRIPS
 - gazebo::common::SubMesh, 819
- LINK
 - gazebo::physics::Base, 149
- LINUX
 - SystemPaths.hh, 1168
- LO_STOP
 - gazebo::physics::Joint, 426
- laser_retro
 - urdf2gazebo::GazeboExtension, 372
- LaserVisual
 - gazebo::rendering::LaserVisual, 447
- LaserVisual.hh, 1052

- LaserVisualPtr
 - gazebo::rendering, 112
- lastPos
 - gazebo::physics::Actor, 125
- lastRenderWallTime
 - gazebo::rendering::Camera, 254
- lastScriptTime
 - gazebo::physics::Actor, 125
- lastTraj
 - gazebo::physics::Actor, 125
- lastUpdateTime
 - gazebo::sensors::Sensor, 771
- latching
 - gazebo::transport::CallbackHelper, 231
- length
 - gazebo::common::Animation, 138
 - gazebo::common::NodeAnimation, 563
 - gazebo::common::SkeletonAnimation, 794
- Light
 - gazebo::rendering::Light, 450
- Light.hh, 1053
- LightFromSDF
 - Messages, 65
- LightPtr
 - gazebo::rendering, 112
- LightingModel
 - gazebo::rendering::RTShaderSystem, 744
- linearAccel
 - gazebo::physics::Link, 473
- Link
 - gazebo::physics::Link, 459
- link
 - gazebo::physics::Collision, 272
- Link.hh, 1054
- Link_V
 - gazebo::physics, 109
- LinkPtr
 - gazebo::physics, 109
- LinkState
 - gazebo::physics::LinkState, 475
- LinkState.hh, 1055
- listGazeboExtensions
 - urdf2gazebo::URDF2Gazebo, 875
- Listen
 - gazebo::transport::Connection, 291
- Load
 - gazebo::common::BVHLoader, 229
 - gazebo::common::ColladaLoader, 262
 - gazebo::common::Console, 296
 - gazebo::common::Image, 411
 - gazebo::common::MeshLoader, 515
 - gazebo::common::MeshManager, 520
 - gazebo::common::STLLoader, 817
 - gazebo::common::Video, 925
 - gazebo::ModelPlugin, 535
 - gazebo::physics::Actor, 124
 - gazebo::physics::BallJoint, 144
 - gazebo::physics::Base, 154
 - gazebo::physics::BulletCollision, 173
 - gazebo::physics::BulletHinge2Joint, 182
 - gazebo::physics::BulletHingeJoint, 187
 - gazebo::physics::BulletJoint, 190
 - gazebo::physics::BulletLink, 196
 - gazebo::physics::BulletPhysics, 206
 - gazebo::physics::BulletScrewJoint, 215
 - gazebo::physics::BulletSliderJoint, 219
 - gazebo::physics::BulletTrimeshShape, 224
 - gazebo::physics::Collision, 270
 - gazebo::physics::CollisionState, 273
 - gazebo::physics::Entity, 345
 - gazebo::physics::Gripper, 393
 - gazebo::physics::HeightmapShape, 403
 - gazebo::physics::Hinge2Joint, 405
 - gazebo::physics::HingeJoint, 407
 - gazebo::physics::Inertial, 418
 - gazebo::physics::Joint, 433
 - gazebo::physics::JointState, 443
 - gazebo::physics::Link, 467
 - gazebo::physics::LinkState, 477
 - gazebo::physics::MapShape, 486
 - gazebo::physics::Model, 529
 - gazebo::physics::ModelState, 539
 - gazebo::physics::ODECollision, 581
 - gazebo::physics::ODEHinge2Joint, 589
 - gazebo::physics::ODEHingeJoint, 593
 - gazebo::physics::ODEJoint, 598
 - gazebo::physics::ODELink, 604
 - gazebo::physics::ODEPhysics, 615
 - gazebo::physics::ODEScrewJoint, 624
 - gazebo::physics::ODESliderJoint, 628
 - gazebo::physics::ODESurfaceParams, 632
 - gazebo::physics::ODETrimeshShape, 634
 - gazebo::physics::PhysicsEngine, 659
 - gazebo::physics::Road, 737
 - gazebo::physics::ScrewJoint, 761
 - gazebo::physics::SliderJoint, 805
 - gazebo::physics::State, 815
 - gazebo::physics::SurfaceParams, 831
 - gazebo::physics::UniversalJoint, 870
 - gazebo::physics::World, 962
 - gazebo::physics::WorldState, 969
 - gazebo::rendering::ArrowVisual, 140
 - gazebo::rendering::AxisVisual, 142
 - gazebo::rendering::Camera, 248
 - gazebo::rendering::CameraVisual, 261
 - gazebo::rendering::COMVisual, 286
 - gazebo::rendering::DepthCamera, 316
 - gazebo::rendering::GpuLaser, 377

- gazebo::rendering::Heightmap, 399
- gazebo::rendering::JointVisual, 445
- gazebo::rendering::Light, 451
- gazebo::rendering::MovableText, 547
- gazebo::rendering::Projector, 690, 691
- gazebo::rendering::RenderEngine, 726
- gazebo::rendering::Road2d, 738
- gazebo::rendering::Scene, 757
- gazebo::rendering::UserCamera, 883
- gazebo::rendering::Visual, 943
- gazebo::SensorPlugin, 778
- gazebo::sensors::CameraSensor, 258
- gazebo::sensors::ContactSensor, 303
- gazebo::sensors::DepthCameraSensor, 319
- gazebo::sensors::GpuRaySensor, 386, 387
- gazebo::sensors::RaySensor, 717
- gazebo::sensors::RFIDSensor, 728
- gazebo::sensors::RFIDTag, 731
- gazebo::sensors::Sensor, 770
- gazebo::Server, 779
- gazebo::SystemPlugin, 839
- gazebo::VisualPlugin, 950
- gazebo::WorldPlugin, 966
- load
 - Classes for physics and dynamics, 73
 - gazebo, 95
 - Rendering, 81
 - Sensors, 86
- load_world
 - Classes for physics and dynamics, 73
- load_worlds
 - Classes for physics and dynamics, 73
- LoadChild
 - gazebo::sensors::ImuSensor, 414
- LoadFromMsg
 - gazebo::rendering::Heightmap, 399
 - gazebo::rendering::Light, 451
 - gazebo::rendering::Visual, 943
- LoadLayout
 - gazebo::rendering::GUIOverlay, 397
- LoadPlugin
 - gazebo::physics::World, 963
 - gazebo::rendering::Visual, 943
- LoadPlugins
 - gazebo::physics::Model, 529
- LocalPublish
 - gazebo::transport::Publication, 693
- Log
 - gazebo::common::Console, 296
- LogPlay.hh, 1058
- Logger.hh, 1057
- loop
 - gazebo::common::Animation, 138
 - gazebo::physics::Actor, 125
- m
 - gazebo::math::Matrix3, 501
 - gazebo::math::Matrix4, 508
- MAP_SHAPE
 - gazebo::physics::Base, 149
- MATRIX
 - gazebo::common::NodeTransform, 566
- MAX_COLLIDE_RETURNS
 - Contact.hh, 1016
- MAX_CONTACT_JOINTS
 - Contact.hh, 1016
- MIDDLE
 - gazebo::common::MouseEvent, 541
- MODEL
 - gazebo::physics::Base, 149
- MODEL_PLUGIN
 - Common, 36
- MODULATE
 - gazebo::common::Material, 491
- MOVE
 - gazebo::common::MouseEvent, 541
- MSleep
 - gazebo::common::Time, 845
- MULTIRAY_SHAPE
 - gazebo::physics::Base, 149
- mainLink
 - gazebo::physics::Actor, 126
- mainpage.html, 1058
- MakeStatic
 - gazebo::rendering::Visual, 943
- MapShape
 - gazebo::physics::MapShape, 485
- MapShape.hh, 1058
- Master
 - gazebo::Master, 488
- Master.hh, 1059
- Material
 - gazebo::common::Material, 492
- material
 - urdf2gazebo::GazeboExtension, 372
- Material.hh, 1060, 1062
- Math, 53
 - clamp, 55
 - equal, 55
 - isPowerOfTwo, 56
 - isnan, 55
 - max, 56
 - mean, 56
 - min, 56
 - NAN_D, 58
 - NAN_I, 58
 - parseFloat, 57
 - parseInt, 57
 - precision, 57

- variance, 57
- MathTypes.hh, 1062
- Matrix3
 - gazebo::math::Matrix3, 499
- Matrix3.hh, 1063
- Matrix4
 - gazebo::math::Matrix4, 503
- Matrix4.hh, 1064
- max
 - gazebo::math::Box, 162
 - Math, 56
- maxVel
 - gazebo::physics::SurfaceParams, 832
 - urdf2gazebo::GazeboExtension, 372
- mean
 - Math, 56
- Merge
 - gazebo::math::Box, 160
- Mesh
 - gazebo::common::Mesh, 510
- mesh
 - gazebo::physics::Actor, 126
 - gazebo::physics::TrimeshShape, 868
- Mesh.hh, 1065
- MeshLoader
 - gazebo::common::MeshLoader, 515
- MeshLoader.hh, 1066
- MeshManager.hh, 1068
- MeshShapePtr
 - gazebo::physics, 109
- Messages, 59
 - Convert, 60–63
 - CreateRequest, 63
 - FogFromSDF, 64
 - GUIFromSDF, 64
 - GetHeader, 64
 - Init, 64
 - LightFromSDF, 65
 - SceneFromSDF, 65
 - Set, 65–67
 - Stamp, 67
 - TrackVisualFromSDF, 67
 - VisualFromSDF, 67
- MicToNano
 - gazebo::common::Time, 845
- MilToNano
 - gazebo::common::Time, 845
- min
 - gazebo::math::Box, 162
 - Math, 56
- minDepth
 - gazebo::physics::SurfaceParams, 833
 - urdf2gazebo::GazeboExtension, 372
- Model
 - gazebo::physics::Model, 524
- model
 - gazebo::physics::Joint, 438
- Model.hh, 1069
- Model_V
 - gazebo::physics, 109
- ModelDatabase.hh, 1070
- modelPathsFromEnv
 - gazebo::common::SystemPaths, 838
- ModelPlugin
 - gazebo::ModelPlugin, 535
- ModelPluginPtr
 - gazebo, 95
- ModelPtr
 - gazebo::physics, 109
- ModelState
 - gazebo::physics::ModelState, 537
- ModelState.hh, 1071
- modelTransform
 - gazebo::common::SkeletonNode, 803
- MouseEvent
 - gazebo::common::MouseEvent, 541
- MouseEvent.hh, 1073
- MovableText
 - gazebo::rendering::MovableText, 545
- MovableText.hh, 1074
- MoveCallback
 - gazebo::physics::ODELink, 605
- moveScale
 - gazebo::common::MouseEvent, 542
- MoveToPosition
 - gazebo::rendering::Camera, 248
 - gazebo::rendering::UserCamera, 883
 - gazebo::rendering::Visual, 943
- MoveToPositions
 - gazebo::rendering::Camera, 248
 - gazebo::rendering::Visual, 944
- MoveToVisual
 - gazebo::rendering::UserCamera, 883
- Moved
 - gazebo::rendering::WindowManager, 953
- msgs.hh, 1075
- mu1
 - gazebo::physics::SurfaceParams, 833
 - urdf2gazebo::GazeboExtension, 373
- mu2
 - gazebo::physics::SurfaceParams, 833
 - urdf2gazebo::GazeboExtension, 373
- MultiRayShape
 - gazebo::physics::MultiRayShape, 551
- MultiRayShape.hh, 1077
- MultiRayShapePtr
 - gazebo::physics, 109

- NAN_D
 - Math, 58
- NAN_I
 - Math, 58
- NO_BUTTON
 - gazebo::common::MouseEvent, 541
- NO_EVENT
 - gazebo::common::MouseEvent, 541
- NODE
 - gazebo::common::SkeletonNode, 797
- NONE
 - gazebo::rendering::RenderEngine, 724
- NRealGen
 - gazebo::math, 101
- NSleep
 - gazebo::common::Time, 846
- NULL
 - CommonTypes.hh, 1009
- name
 - gazebo::common::Animation, 139
 - gazebo::common::Material, 497
 - gazebo::common::NodeAnimation, 563
 - gazebo::common::SkeletonAnimation, 795
 - gazebo::common::SkeletonNode, 803
 - gazebo::physics::State, 815
 - gazebo::rendering::Camera, 254
 - sdf::Plugin, 675
- near
 - gazebo::sensors::GpuRaySensor, 388
- newData
 - gazebo::rendering::Camera, 254
- newImageFrame
 - gazebo::rendering::Camera, 254
- newLaserScans
 - gazebo::physics::MultiRayShape, 555
- NewPhysicsEngine
 - gazebo::physics::PhysicsFactory, 664
- NewSensor
 - gazebo::sensors::SensorFactory, 773
- Node
 - gazebo::transport::Node, 558
- node
 - gazebo::physics::Entity, 349
 - gazebo::physics::PhysicsEngine, 662
 - gazebo::sensors::Sensor, 771
- Node.hh, 1078
- NodeAnimation
 - gazebo::common::NodeAnimation, 561
- nodeIndex
 - gazebo::common::NodeAssignment, 564
- NodeMap
 - gazebo::common, 98
- NodeMapItr
 - gazebo::common, 98
- NodePtr
 - gazebo::transport, 117
- NodeTransform
 - gazebo::common::NodeTransform, 566
- nodes
 - gazebo::common::Skeleton, 790
- normal
 - gazebo::math::Plane, 671
- NormalRealDist
 - gazebo::math, 101
- Normalize
 - gazebo::math::Angle, 130
 - gazebo::math::Quaternion, 704
 - gazebo::math::Vector2d, 888
 - gazebo::math::Vector2i, 897
 - gazebo::math::Vector3, 909
 - gazebo::math::Vector4, 918
- normals
 - gazebo::physics::Contact, 299
- notifyRenderSingleObject
 - gazebo::rendering::GpuLaser, 378
- nsec
 - gazebo::common::Time, 859
- NumericAnimation
 - gazebo::common::NumericAnimation, 570
- NumericAnimationPtr
 - gazebo::common, 98
- NumericKeyFrame
 - gazebo::common::NumericKeyFrame, 572
- ODE Physics, 77
 - ODEPhysicsPtr, 78
 - ODESurfaceParamsPtr, 78
- ODEBallJoint
 - gazebo::physics::ODEBallJoint, 575
- ODEBallJoint.hh, 1080
- ODEBoxShape
 - gazebo::physics::ODEBoxShape, 577
- ODEBoxShape.hh, 1081
- ODECollision
 - gazebo::physics::ODECollision, 580
- ODECollision.hh, 1082
- ODECollisionPtr
 - gazebo::physics, 109
- ODECylinderShape
 - gazebo::physics::ODECylinderShape, 584
- ODECylinderShape.hh, 1083
- ODEHeightmapShape
 - gazebo::physics::ODEHeightmapShape, 586
- ODEHeightmapShape.hh, 1083
- ODEHinge2Joint
 - gazebo::physics::ODEHinge2Joint, 588
- ODEHinge2Joint.hh, 1084
- ODEHingeJoint

- gazebo::physics::ODEHingeJoint, 592
- ODEHingeJoint.hh, 1085
- ODEJoint
 - gazebo::physics::ODEJoint, 596
- ODEJoint.hh, 1085
- ODELink
 - gazebo::physics::ODELink, 602
- ODELink.hh, 1086
- ODELinkPtr
 - gazebo::physics, 109
- ODEMultiRayShape
 - gazebo::physics::ODEMultiRayShape, 609
- ODEMultiRayShape.hh, 1087
- ODEPhysics
 - gazebo::physics::ODEPhysics, 612
- ODEPhysics.hh, 1087
- ODEPhysicsPtr
 - ODE Physics, 78
- ODEPlaneShape
 - gazebo::physics::ODEPlaneShape, 619
- ODEPlaneShape.hh, 1088
- ODERayShape
 - gazebo::physics::ODERayShape, 621
- ODERayShape.hh, 1089
- ODERayShapePtr
 - gazebo::physics, 109
- ODEScrewJoint
 - gazebo::physics::ODEScrewJoint, 623
- ODEScrewJoint.hh, 1090
- ODESliderJoint
 - gazebo::physics::ODESliderJoint, 627
- ODESliderJoint.hh, 1090
- ODESphereShape
 - gazebo::physics::ODESphereShape, 630
- ODESphereShape.hh, 1091
- ODESurfaceParams
 - gazebo::physics::ODESurfaceParams, 631
- ODESurfaceParams.hh, 1092
- ODESurfaceParamsPtr
 - ODE Physics, 78
- ODETrimeshShape
 - gazebo::physics::ODETrimeshShape, 634
- ODETrimeshShape.hh, 1092
- ODETypes.hh, 1093
- ODEUniversalJoint
 - gazebo::physics::ODEUniversalJoint, 636
- ODEUniversalJoint.hh, 1094
- ORDER_MAX
 - STLLoader.hh, 1161
- ode_inc.h, 1080
- offset
 - gazebo::physics::MultiRayShape, 556
 - gazebo::sensors::GpuRaySensor, 388
- Ogre, 118
- ogre, 118
- ogre_gazebo.h, 1095
- ogrePathsFromEnv
 - gazebo::common::SystemPaths, 838
- old_link_name
 - urdf2gazebo::GazeboExtension, 373
- oldAction
 - gazebo::physics::Actor, 126
- onAnimationComplete
 - gazebo::rendering::Camera, 254
- OnPhysicsMsg
 - gazebo::physics::ODEPhysics, 615
 - gazebo::physics::PhysicsEngine, 659
- OnPoseChange
 - gazebo::physics::BulletCollision, 173
 - gazebo::physics::BulletLink, 196
 - gazebo::physics::Entity, 346
 - gazebo::physics::Link, 468
 - gazebo::physics::Model, 529
 - gazebo::physics::ODECollision, 581
 - gazebo::physics::ODELink, 605
 - gazebo::rendering::Light, 451
- OnRequest
 - gazebo::physics::ODEPhysics, 615
 - gazebo::physics::PhysicsEngine, 659
- Open
 - gazebo::physics::Logplay, 480
- operator<
 - gazebo::common::Time, 852, 853
 - gazebo::math::Angle, 132
- operator<<
 - gazebo::common::Color, 284
 - gazebo::common::Exception, 366
 - gazebo::common::Material, 496
 - gazebo::common::Time, 859
 - gazebo::common::Timer, 861
 - gazebo::math::Angle, 134
 - gazebo::math::Box, 161
 - gazebo::math::Matrix3, 501
 - gazebo::math::Matrix4, 508
 - gazebo::math::Pose, 684
 - gazebo::math::Quaternion, 709
 - gazebo::math::Vector2d, 893
 - gazebo::math::Vector2i, 902
 - gazebo::math::Vector3, 914
 - gazebo::math::Vector4, 923
 - gazebo::physics::Inertial, 422
 - sdf::ParamT, 651
- operator<=
 - gazebo::common::Time, 853, 854
 - gazebo::math::Angle, 132
- operator>
 - gazebo::common::Time, 856, 857
 - gazebo::math::Angle, 133

- operator>>
 - gazebo::common::Color, 284
 - gazebo::common::Time, 859
 - gazebo::math::Angle, 134
 - gazebo::math::Pose, 684
 - gazebo::math::Quaternion, 709
 - gazebo::math::Vector2d, 893
 - gazebo::math::Vector2i, 902
 - gazebo::math::Vector3, 914
 - gazebo::math::Vector4, 923
- operator>=
 - gazebo::common::Time, 857, 858
 - gazebo::math::Angle, 133
- operator*
 - gazebo::common::Color, 278, 279
 - gazebo::common::NodeTransform, 567
 - gazebo::common::Time, 847
 - gazebo::math::Angle, 130
 - gazebo::math::Matrix4, 505
 - gazebo::math::Pose, 681
 - gazebo::math::Quaternion, 704, 705
 - gazebo::math::Vector2d, 888, 889
 - gazebo::math::Vector2i, 897
 - gazebo::math::Vector3, 909, 910
 - gazebo::math::Vector4, 918, 919
 - sdf::ParamT, 650
- operator*=
 - gazebo::common::Color, 279
 - gazebo::common::Time, 848
 - gazebo::math::Angle, 130
 - gazebo::math::Quaternion, 705
 - gazebo::math::Vector2d, 889
 - gazebo::math::Vector2i, 898
 - gazebo::math::Vector3, 910
 - gazebo::math::Vector4, 919, 920
- operator()
 - gazebo::common::NodeTransform, 567
 - gazebo::event::EventT, 359–361
- operator+
 - gazebo::common::Color, 279
 - gazebo::common::Time, 848, 849
 - gazebo::math::Angle, 131
 - gazebo::math::Box, 160
 - gazebo::math::Pose, 682
 - gazebo::math::Quaternion, 705
 - gazebo::math::Vector2d, 890
 - gazebo::math::Vector2i, 898
 - gazebo::math::Vector3, 910
 - gazebo::math::Vector4, 920
 - gazebo::physics::Inertial, 419
- operator+=
 - gazebo::common::Color, 280
 - gazebo::common::Time, 849
 - gazebo::math::Angle, 131
 - gazebo::math::Box, 160
 - gazebo::math::Pose, 682
 - gazebo::math::Quaternion, 705
 - gazebo::math::Vector2d, 890
 - gazebo::math::Vector2i, 899
 - gazebo::math::Vector3, 911
 - gazebo::math::Vector4, 920
- operator-
 - gazebo::common::Color, 280
 - gazebo::common::Time, 850
 - gazebo::math::Angle, 131
 - gazebo::math::Box, 161
 - gazebo::math::Pose, 682
 - gazebo::math::Quaternion, 706
 - gazebo::math::Vector2d, 890
 - gazebo::math::Vector2i, 899
 - gazebo::math::Vector3, 911
 - gazebo::math::Vector4, 920
- operator-=
 - gazebo::common::Color, 280
 - gazebo::common::Time, 850
 - gazebo::math::Angle, 131
 - gazebo::math::Pose, 682
 - gazebo::math::Quaternion, 706
 - gazebo::math::Vector2d, 890
 - gazebo::math::Vector2i, 899
 - gazebo::math::Vector3, 911
 - gazebo::math::Vector4, 921
- operator/
 - gazebo::common::Color, 281
 - gazebo::common::Time, 851
 - gazebo::math::Angle, 132
 - gazebo::math::Vector2d, 890, 891
 - gazebo::math::Vector2i, 899
 - gazebo::math::Vector3, 911
 - gazebo::math::Vector4, 921
- operator/=
 - gazebo::common::Color, 281
 - gazebo::common::Time, 851, 852
 - gazebo::math::Angle, 132
 - gazebo::math::Vector2d, 891
 - gazebo::math::Vector2i, 900
 - gazebo::math::Vector3, 912
 - gazebo::math::Vector4, 921, 922
- operator=
 - gazebo::common::Color, 281
 - gazebo::common::PID, 666
 - gazebo::common::Time, 854, 855
 - gazebo::math::Box, 161
 - gazebo::math::Matrix4, 505, 506
 - gazebo::math::Plane, 670
 - gazebo::math::Quaternion, 706
 - gazebo::math::Vector2d, 892

- gazebo::math::Vector2i, 900, 901
- gazebo::math::Vector3, 912
- gazebo::math::Vector4, 922
- gazebo::physics::Contact, 298
- gazebo::physics::Inertial, 419
- gazebo::physics::JointFeedback, 441
- operator==
 - gazebo::common::Color, 282
 - gazebo::common::Time, 855, 856
 - gazebo::math::Angle, 133
 - gazebo::math::Box, 161
 - gazebo::math::Matrix3, 499
 - gazebo::math::Matrix4, 506
 - gazebo::math::Pose, 683
 - gazebo::math::Quaternion, 706
 - gazebo::math::Vector2d, 892
 - gazebo::math::Vector2i, 901
 - gazebo::math::Vector3, 913
 - gazebo::math::Vector4, 922
 - gazebo::physics::Base, 154
- operator[]
 - gazebo::common::Color, 282
 - gazebo::math::Matrix3, 500
 - gazebo::math::Matrix4, 506
 - gazebo::math::Vector2d, 892
 - gazebo::math::Vector2i, 901
 - gazebo::math::Vector3, 913
 - gazebo::math::Vector4, 923
- OrbitViewController
 - gazebo::rendering::OrbitViewController, 639
- OrbitViewController.hh, 1096
- PHONG
 - gazebo::common::Material, 491
- PID
 - gazebo::common::PID, 665
- PID.hh, 1106
- PLANE_SHAPE
 - gazebo::physics::Base, 150
- POINTS
 - gazebo::common::SubMesh, 819
- PRESS
 - gazebo::common::MouseEvent, 541
- Param
 - sdf::Param, 644
- Param.hh, 1096
- Param_V
 - gazebo::common, 98
 - sdf, 119
- ParamPtr
 - sdf, 119
- ParamT
 - sdf::ParamT, 649
- ParamT< T >, 648
- parent
 - gazebo::common::SkeletonNode, 804
 - gazebo::physics::Base, 157
 - gazebo::rendering::Visual, 949
- parentEntity
 - gazebo::physics::Entity, 349
- parentLink
 - gazebo::physics::Joint, 438
- parentName
 - gazebo::sensors::Sensor, 771
- ParseArgs
 - gazebo::Server, 779
- parseFloat
 - Math, 57
- parseGazeboExtension
 - urdf2gazebo::URDF2Gazebo, 875
- parseInt
 - Math, 57
- parseVector3
 - urdf2gazebo::URDF2Gazebo, 875
- parser.hh, 1098
- parser_urdf.hh, 1099
- pathLength
 - gazebo::physics::Actor, 126
- pause
 - Events, 51
- pause_incoming
 - Transport, 90
- pause_world
 - Classes for physics and dynamics, 73
- pause_worlds
 - Classes for physics and dynamics, 73
- PauseIncoming
 - gazebo::transport::TopicManager, 864
- Physics.hh, 1100
- PhysicsEngine
 - gazebo::physics::PhysicsEngine, 654
- PhysicsEngine.hh, 1101
- PhysicsEnginePtr
 - gazebo::physics, 109
- PhysicsFactory.hh, 1102
- PhysicsFactoryFn
 - Classes for physics and dynamics, 72
- physicsSub
 - gazebo::physics::PhysicsEngine, 662
- PhysicsTypes.hh, 1103
 - GZ_ALL_COLLIDE, 1105
 - GZ_FIXED_COLLIDE, 1105
 - GZ_GHOST_COLLIDE, 1105
 - GZ_NONE_COLLIDE, 1105
 - GZ_SENSOR_COLLIDE, 1105
- physicsUpdateMutex
 - gazebo::physics::PhysicsEngine, 663
- pitchNode

- gazebo::rendering::Camera, 254
- PixelFormat
 - gazebo::common::Image, 408
- PlaceOnEntity
 - gazebo::physics::Entity, 346
- PlaceOnNearestEntityBelow
 - gazebo::physics::Entity, 346
- placeable
 - gazebo::physics::Collision, 272
- Plane
 - gazebo::math::Plane, 669
- Plane.hh, 1107
- PlaneShape
 - gazebo::physics::PlaneShape, 672
- PlaneShape.hh, 1108
- Play
 - gazebo::physics::Actor, 124
 - gazebo::physics::Logplay, 481–483
- playStartTime
 - gazebo::physics::Actor, 126
- Plugin
 - sdf::Plugin, 674
- Plugin.hh, 1109, 1113
- pluginPathsFromEnv
 - gazebo::common::SystemPaths, 838
- PluginType
 - Common, 36
- plugins
 - gazebo::sensors::Sensor, 772
- pointSize
 - gazebo::common::Material, 497
- points
 - gazebo::math::RotationSpline, 742
 - gazebo::math::Spline, 812
- pos
 - gazebo::common::MouseEvent, 542
 - gazebo::math::Pose, 685
- Pose
 - gazebo::math::Pose, 679
- pose
 - gazebo::physics::BulletLink, 198
 - gazebo::physics::ODELink, 607
 - gazebo::sensors::Sensor, 772
- Pose.hh, 1113
- PoseAnimation
 - gazebo::common::PoseAnimation, 686
- PoseAnimationPtr
 - gazebo::common, 98
- PoseKeyFrame
 - gazebo::common::PoseKeyFrame, 688
- poseMsg
 - gazebo::physics::Entity, 349
- poseSub
 - gazebo::sensors::Sensor, 772
- positions
 - gazebo::physics::Contact, 299
- PostRender
 - gazebo::rendering::Camera, 248
 - gazebo::rendering::DepthCamera, 316
 - gazebo::rendering::GpuLaser, 378
 - gazebo::rendering::UserCamera, 883
- postRender
 - Events, 51
- PreRender
 - gazebo::rendering::Scene, 757
- preRender
 - Events, 51
- precision
 - Math, 57
- PrepareHardwareBuffers
 - gazebo::rendering::DynamicRenderable, 331
- pressPos
 - gazebo::common::MouseEvent, 542
- prevAnimTime
 - gazebo::rendering::Camera, 254
- prevAnimationTime
 - gazebo::physics::Entity, 349
- prevFrameTime
 - gazebo::physics::Actor, 126
- prevPos
 - gazebo::common::MouseEvent, 542
- PrimitiveType
 - gazebo::common::SubMesh, 819
- Print
 - gazebo::physics::Base, 155
 - sdf::Plugin, 674
- print_version
 - gazebo, 95
- printCollisionGroups
 - urdf2gazebo::URDF2Gazebo, 875
- PrintDescription
 - sdf::Element, 337
 - sdf::SDF, 763
- PrintDoc
 - sdf::SDF, 763
- PrintDocLeftPane
 - sdf::Element, 337
- PrintDocRightPane
 - sdf::Element, 337
- PrintEntityTree
 - gazebo::physics::World, 963
- printMass
 - urdf2gazebo::URDF2Gazebo, 875
- PrintSceneGraph
 - gazebo::rendering::Scene, 757
- PrintSource
 - gazebo::common::NodeTransform, 568
- PrintTransforms

- gazebo::common::Skeleton, 789
- PrintUsage
 - gazebo::Server, 779
- PrintValues
 - sdf::Element, 337
 - sdf::SDF, 763
- PrintWiki
 - sdf::Element, 337
 - sdf::SDF, 763
- ProcessContactFeedback
 - gazebo::physics::ODEPhysics, 615
- ProcessIncoming
 - gazebo::transport::Node, 559
- ProcessMsg
 - gazebo::physics::BoxShape, 165
 - gazebo::physics::Collision, 270
 - gazebo::physics::CylinderShape, 310
 - gazebo::physics::HeightmapShape, 403
 - gazebo::physics::Inertial, 419
 - gazebo::physics::Link, 468
 - gazebo::physics::MapShape, 487
 - gazebo::physics::Model, 529
 - gazebo::physics::MultiRayShape, 555
 - gazebo::physics::PlaneShape, 673
 - gazebo::physics::RayShape, 721
 - gazebo::physics::Shape, 782
 - gazebo::physics::SphereShape, 808
 - gazebo::physics::SurfaceParams, 831
 - gazebo::physics::TrimeshShape, 868
- ProcessNodes
 - gazebo::transport::TopicManager, 864
- ProcessPublishers
 - gazebo::transport::Node, 559
- ProcessWriteQueue
 - gazebo::transport::Connection, 291
- Projector
 - gazebo::rendering::Projector, 690
- Projector.hh, 1114
- provideFeedback
 - urdf2gazebo::GazeboExtension, 373
- Publication
 - gazebo::transport::Publication, 692
- Publication.hh, 1115
- PublicationPtr
 - gazebo::transport, 117
- PublicationTransport
 - gazebo::transport::PublicationTransport, 694
- PublicationTransport.hh, 1117
- PublicationTransportPtr
 - gazebo::transport, 117
- Publish
 - gazebo::transport::Publication, 693
 - gazebo::transport::Publisher, 696
 - gazebo::transport::TopicManager, 864
- Publisher
 - gazebo::transport::Publisher, 696
- Publisher.hh, 1119
- PublisherPtr
 - gazebo::transport, 117
- Purple
 - gazebo::common::Color, 284
- Quaternion
 - gazebo::math::Quaternion, 700
- Quaternion.hh, 1121
- r
 - gazebo::common::Color, 284
- R_FLOAT16
 - gazebo::common::Image, 408
- R_FLOAT32
 - gazebo::common::Image, 408
- RAY_SHAPE
 - gazebo::physics::Base, 150
- RELEASE
 - gazebo::common::MouseEvent, 541
- RENDER_PATH_COUNT
 - gazebo::rendering::RenderEngine, 724
- RENDERING_LINE_LIST
 - gazebo::rendering, 113
- RENDERING_LINE_STRIP
 - gazebo::rendering, 113
- RENDERING_MESH_RESOURCE
 - gazebo::rendering, 113
- RENDERING_POINT_LIST
 - gazebo::rendering, 113
- RENDERING_TRIANGLE_FAN
 - gazebo::rendering, 113
- RENDERING_TRIANGLE_LIST
 - gazebo::rendering, 113
- RENDERING_TRIANGLE_STRIP
 - gazebo::rendering, 113
- REPLACE
 - gazebo::common::Material, 491
- RFIDSensor
 - gazebo::sensors::RFIDSensor, 728
- RFIDSensor.hh, 1130
- RFIDSensor_V
 - gazebo::sensors, 115
- RFIDSensorPtr
 - gazebo::sensors, 115
- RFIDTag
 - gazebo::sensors::RFIDTag, 730
- RFIDTag.hh, 1131
- RFIDTag_V
 - gazebo::sensors, 115
- RFIDTagManager.hh, 1132
- RFIDTagPtr
 - gazebo::sensors, 115

- RFIDTagVisual
 - gazebo::rendering::RFIDTagVisual, 734
- RFIDTagVisual.hh, 1133
- RFIDTagVisualPtr
 - gazebo::rendering, 112
- RFIDVisual
 - gazebo::rendering::RFIDVisual, 735
- RFIDVisual.hh, 1133
- RFIDVisualPtr
 - gazebo::rendering, 112
- RGB_FLOAT16
 - gazebo::common::Image, 408
- RGB_FLOAT32
 - gazebo::common::Image, 409
- RGB_INT16
 - gazebo::common::Image, 408
- RGB_INT32
 - gazebo::common::Image, 408
- RGB_INT8
 - gazebo::common::Image, 408
- RGBA
 - gazebo::common::Color, 276
- RGBA_INT8
 - gazebo::common::Image, 408
- RIGHT
 - gazebo::common::MouseEvent, 541
- ROTATE
 - gazebo::common::NodeTransform, 566
- RTShaderSystem.hh, 1137
- Radian
 - gazebo::math::Angle, 133
- Rand.hh, 1122
- rangeElem
 - gazebo::physics::MultiRayShape, 556
 - gazebo::sensors::GpuRaySensor, 388
- ratio_1st
 - gazebo::sensors::GpuRaySensor, 388
- ratio_2nd
 - gazebo::sensors::GpuRaySensor, 388
- rawNW
 - gazebo::common::Skeleton, 790
- RawNodeAnim
 - gazebo::common, 98
- RawNodeWeights
 - gazebo::common, 98
- RawSkeletonAnim
 - gazebo::common, 98
- rawTransforms
 - gazebo::common::SkeletonNode, 804
- rayElem
 - gazebo::physics::MultiRayShape, 556
 - gazebo::sensors::GpuRaySensor, 388
- RaySensor
 - gazebo::sensors::RaySensor, 713
- RaySensor.hh, 1123
- RaySensor_V
 - gazebo::sensors, 115
- RaySensorPtr
 - gazebo::sensors, 115
- RayShape
 - gazebo::physics::RayShape, 720
- RayShape.hh, 1124
- RayShapePtr
 - gazebo::physics, 109
- rays
 - gazebo::physics::MultiRayShape, 556
- Read
 - gazebo::transport::Connection, 291
- ReadCallback
 - gazebo::transport::Connection, 290
- readDoc
 - sdf, 120
- readFile
 - sdf, 120
- readString
 - sdf, 120
- readXml
 - sdf, 120
- realTime
 - gazebo::physics::State, 815
- RecalcTangents
 - gazebo::math::RotationSpline, 741
 - gazebo::math::Spline, 811
- RecalculateMatrix
 - gazebo::common::NodeTransform, 568
- RecalculateNormals
 - gazebo::common::Mesh, 513
 - gazebo::common::SubMesh, 824
- Red
 - gazebo::common::Color, 285
- reduceCollisionToParent
 - urdf2gazebo::URDF2Gazebo, 875
- reduceCollisionsToParent
 - urdf2gazebo::URDF2Gazebo, 875
- reduceFixedJoints
 - urdf2gazebo::URDF2Gazebo, 876
- reduceGazeboExtensionContactSensorFrameReplace
 - urdf2gazebo::URDF2Gazebo, 876
- reduceGazeboExtensionFrameReplace
 - urdf2gazebo::URDF2Gazebo, 876
- reduceGazeboExtensionGripperFrameReplace
 - urdf2gazebo::URDF2Gazebo, 876
- reduceGazeboExtensionJointFrameReplace
 - urdf2gazebo::URDF2Gazebo, 876
- reduceGazeboExtensionPluginFrameReplace
 - urdf2gazebo::URDF2Gazebo, 876
- reduceGazeboExtensionProjectorFrameReplace
 - urdf2gazebo::URDF2Gazebo, 876

- reduceGazeboExtensionProjectorTransformReduction
 - urdf2gazebo::URDF2Gazebo, 876
- reduceGazeboExtensionSensorTransformReduction
 - urdf2gazebo::URDF2Gazebo, 876
- reduceGazeboExtensionToParent
 - urdf2gazebo::URDF2Gazebo, 877
- reduceGazeboExtensionsTransformReduction
 - urdf2gazebo::URDF2Gazebo, 877
- reduceInertialToParent
 - urdf2gazebo::URDF2Gazebo, 877
- reduceJointsToParent
 - urdf2gazebo::URDF2Gazebo, 877
- reduceVisualToParent
 - urdf2gazebo::URDF2Gazebo, 877
- reduceVisualsToParent
 - urdf2gazebo::URDF2Gazebo, 877
- reduction_transform
 - urdf2gazebo::GazeboExtension, 373
- RegisterAll
 - gazebo::physics::PhysicsFactory, 664
 - gazebo::sensors::SensorFactory, 773
- RegisterPhysicsEngine
 - gazebo::physics::PhysicsFactory, 664
- RegisterSensor
 - gazebo::sensors::SensorFactory, 773
- RegisterTopicNamespace
 - gazebo::transport::ConnectionManager, 294
 - gazebo::transport::TopicManager, 864
- relativeEndPos
 - gazebo::physics::RayShape, 722
- relativeStartPos
 - gazebo::physics::RayShape, 722
- Remove
 - gazebo::physics::Logger, 479
- remove_scene
 - Rendering, 81
- remove_sensor
 - Sensors, 86
- remove_sensors
 - Sensors, 86
- remove_worlds
 - Classes for physics and dynamics, 73
- RemoveChild
 - gazebo::physics::Base, 155
 - gazebo::physics::Model, 529
- RemoveChildJoint
 - gazebo::physics::Link, 468
- RemoveChildren
 - gazebo::physics::Base, 155
- RemoveConnection
 - gazebo::transport::ConnectionManager, 294
- RemoveNode
 - gazebo::transport::TopicManager, 864
- RemoveParentJoint
 - gazebo::physics::Link, 468
- RemovePlugin
 - gazebo::physics::World, 963
 - gazebo::rendering::Visual, 944
- RemoveScene
 - gazebo::rendering::RenderEngine, 726
 - gazebo::rendering::RTShaderSystem, 745
- removeScene
 - gazebo::rendering::Events, 356
- RemoveSensor
 - gazebo::sensors::SensorManager, 776
- RemoveSensors
 - gazebo::sensors::SensorManager, 776
- RemoveShadows
 - gazebo::rendering::RTShaderSystem, 745
- RemoveSubscription
 - gazebo::transport::Publication, 693
- RemoveTransport
 - gazebo::transport::Publication, 694
- RemoveVisual
 - gazebo::rendering::Scene, 758
- Render
 - gazebo::rendering::Camera, 249
- render
 - Events, 51
- RenderEngine.hh, 1125
- RenderEvents.hh, 1126
- RenderImpl
 - gazebo::rendering::Camera, 249
- RenderOpType
 - gazebo::rendering, 113
- RenderPathType
 - gazebo::rendering::RenderEngine, 724
- renderTarget
 - gazebo::rendering::Camera, 255
- renderTexture
 - gazebo::rendering::Camera, 255
- RenderTypes.hh, 1128
 - GZ_VISIBILITY_ALL, 1129
 - GZ_VISIBILITY_GUI, 1129
 - GZ_VISIBILITY_NOT_SELECTABLE, 1129
- Rendering, 79
 - create_scene, 81
 - fini, 81
 - get_scene, 81
 - init, 81
 - load, 81
 - remove_scene, 81
- rendering.h, 1126
- Rendering.hh, 1127
- request
 - Transport, 90
- requestPub
 - gazebo::physics::Entity, 349

- requestSub
 - gazebo::physics::PhysicsEngine, 663
- requests
 - gazebo::rendering::Camera, 255
- required
 - sdf::Param, 647
- Rescale
 - gazebo::common::Image, 411
- Reset
 - gazebo::common::Color, 282
 - gazebo::common::PID, 666
 - gazebo::common::SkeletonNode, 801
 - gazebo::math::Pose, 683
 - gazebo::ModelPlugin, 536
 - gazebo::physics::Base, 155
 - gazebo::physics::BulletJoint, 191
 - gazebo::physics::Contact, 298
 - gazebo::physics::Entity, 346
 - gazebo::physics::Inertial, 420
 - gazebo::physics::Joint, 434
 - gazebo::physics::JointController, 439
 - gazebo::physics::Link, 468
 - gazebo::physics::Model, 530
 - gazebo::physics::ODEJoint, 598
 - gazebo::physics::ODEPhysics, 616
 - gazebo::physics::PhysicsEngine, 659
 - gazebo::physics::World, 963
 - gazebo::SensorPlugin, 778
 - gazebo::SystemPlugin, 840
 - gazebo::VisualPlugin, 951
 - gazebo::WorldPlugin, 966
 - sdf::Element, 337
 - sdf::Param, 646
 - sdf::ParamT, 650
- ResetEntities
 - gazebo::physics::World, 963
- ResetTime
 - gazebo::physics::World, 963
- Resize
 - gazebo::rendering::GUIOverlay, 397
 - gazebo::rendering::UserCamera, 884
 - gazebo::rendering::WindowManager, 953
- responsePub
 - gazebo::physics::PhysicsEngine, 663
- Road, 736
 - gazebo::physics::Road, 737
- Road.hh, 1134
- Road2d
 - gazebo::rendering::Road2d, 738
- Road2d.hh, 1135
- RoadPtr
 - gazebo::physics, 109
- root
 - gazebo::common::Skeleton, 790
 - gazebo::rendering::RenderEngine, 726
 - sdf::SDF, 763
- rot
 - gazebo::math::Pose, 685
- Rotate
 - gazebo::physics::Inertial, 420
- rotate
 - gazebo::common::PoseKeyFrame, 689
- RotatePitch
 - gazebo::rendering::Camera, 249
- RotatePositionAboutOrigin
 - gazebo::math::Pose, 683
- RotateVector
 - gazebo::math::Quaternion, 707
- RotateVectorReverse
 - gazebo::math::Quaternion, 707
- RotateYaw
 - gazebo::rendering::Camera, 249
- RotationSpline
 - gazebo::math::RotationSpline, 739
- RotationSpline.hh, 1135
- Round
 - gazebo::math::Pose, 683
 - gazebo::math::Quaternion, 707
 - gazebo::math::Vector3, 913
- Run
 - gazebo::Master, 488
 - gazebo::physics::World, 963
 - gazebo::sensors::SensorManager, 776
 - gazebo::Server, 779
 - gazebo::transport::ConnectionManager, 294
- run
 - gazebo, 95
 - Sensors, 86
 - Transport, 90
- run_once
 - Sensors, 86
- run_world
 - Classes for physics and dynamics, 73
- run_worlds
 - Classes for physics and dynamics, 74
- RunOnce
 - gazebo::Master, 488
- RunThread
 - gazebo::Master, 488
- RunUpdate
 - gazebo::transport::ConnectionManager, 294
- SCALE
 - gazebo::common::NodeTransform, 566
- SCREW_JOINT
 - gazebo::physics::Base, 149
- SCROLL
 - gazebo::common::MouseEvent, 541

- SDF
 - sdf::SDF, 763
- SDF.hh, 1140
 - SDF_VERSION, 1141
- SDF_VERSION
 - SDF.hh, 1141
- SDFPtr
 - sdf, 119
- SENSOR_PLUGIN
 - Common, 36
- SHADE_COUNT
 - gazebo::common::Material, 491
- SHAPE
 - gazebo::physics::Base, 149
- SLIDER_JOINT
 - gazebo::physics::Base, 149
- SPHERE_SHAPE
 - gazebo::physics::Base, 150
- SSLM_NormalMapLightingObjectSpace
 - gazebo::rendering::RTShaderSystem, 744
- SSLM_NormalMapLightingTangentSpace
 - gazebo::rendering::RTShaderSystem, 744
- SSLM_PerPixelLighting
 - gazebo::rendering::RTShaderSystem, 744
- SSLM_PerVertexLighting
 - gazebo::rendering::RTShaderSystem, 744
- STLLoader
 - gazebo::common::STLLoader, 816
- STLLoader.hh, 1159
 - COR3_MAX, 1160
 - FACE_MAX, 1160
 - LINE_MAX_LEN, 1161
 - ORDER_MAX, 1161
- STOP_CFM
 - gazebo::physics::Joint, 426
- STOP_ERP
 - gazebo::physics::Joint, 426
- SUSPENSION_CFM
 - gazebo::physics::Joint, 426
- SUSPENSION_ERP
 - gazebo::physics::Joint, 426
- SYSTEM_PLUGIN
 - Common, 36
- Save
 - gazebo::physics::World, 964
- SaveChild
 - gazebo::sensors::ImuSensor, 414
- saveCount
 - gazebo::rendering::Camera, 255
- SaveFrame
 - gazebo::rendering::Camera, 249
 - gazebo::sensors::CameraSensor, 258
 - gazebo::sensors::DepthCameraSensor, 319
- saveFrameBuffer
 - gazebo::rendering::Camera, 255
- SavePNG
 - gazebo::common::Image, 411
- Scale
 - gazebo::common::Mesh, 514
 - gazebo::common::NodeAnimation, 563
 - gazebo::common::Skeleton, 789
 - gazebo::common::SkeletonAnimation, 794
 - gazebo::common::SubMesh, 824
 - gazebo::math::Quaternion, 707
- scale
 - gazebo::physics::HeightmapShape, 404
- ScaleXAxis
 - gazebo::rendering::AxisVisual, 142
- ScaleYAxis
 - gazebo::rendering::AxisVisual, 142
- ScaleZAxis
 - gazebo::rendering::AxisVisual, 142
- scanElem
 - gazebo::physics::MultiRayShape, 556
 - gazebo::sensors::GpuRaySensor, 388
- Scene
 - gazebo::rendering::Scene, 749
- scene
 - gazebo::rendering::Camera, 255
 - gazebo::rendering::Visual, 949
- Scene.hh, 1137
- SceneFromSDF
 - Messages, 65
- sceneNode
 - gazebo::rendering::Camera, 255
 - gazebo::rendering::Visual, 949
- ScenePtr
 - gazebo::rendering, 112
- ScrewJoint
 - gazebo::physics::ScrewJoint, 761
- ScrewJoint.hh, 1138
- scriptLength
 - gazebo::physics::Actor, 126
- scroll
 - gazebo::common::MouseEvent, 542
- sdf, 118
 - addNestedModel, 119
 - copyChildren, 119
 - ElementPtr, 119
 - ElementPtr_V, 119
 - gazebo::physics::Base, 157
 - gazebo::physics::PhysicsEngine, 663
 - gazebo::rendering::Camera, 255
 - gazebo::sensors::Sensor, 772
 - init, 119
 - initDoc, 119
 - initFile, 119
 - initString, 119

initXml, 119
 Param_V, 119
 ParamPtr, 119
 readDoc, 120
 readFile, 120
 readString, 120
 readXml, 120
 SDFPtr, 119
 sdf.hh, 1139
 sdf::Converter, 307
 Convert, 307
 sdf::Element, 332
 ~Element, 334
 AddAttribute, 334
 AddElement, 334
 AddElementDescription, 334
 AddValue, 334
 ClearElements, 334
 Clone, 334
 Copy, 334
 Element, 334
 GetAttribute, 334, 335
 GetAttributeCount, 335
 GetAttributeSet, 335
 GetCopyChildren, 335
 GetDescription, 335
 GetElement, 335
 GetElementDescription, 335
 GetElementDescriptionCount, 335
 GetElementImpl, 335
 GetFirstElement, 336
 GetInclude, 336
 GetName, 336
 GetNextElement, 336
 GetParent, 336
 GetRequired, 336
 GetValue, 336
 GetValueBool, 336
 GetValueChar, 336
 GetValueColor, 336
 GetValueDouble, 336
 GetValueFloat, 336
 GetValueInt, 336
 GetValuePose, 336
 GetValueQuaternion, 336
 GetValueString, 336
 GetValueTime, 336
 GetValueUInt, 336
 GetValueVector2d, 336
 GetValueVector3, 336
 HasAttribute, 336
 HasElement, 336
 HasElementDescription, 337
 InsertElement, 337
 PrintDescription, 337
 PrintDocLeftPane, 337
 PrintDocRightPane, 337
 PrintValues, 337
 PrintWiki, 337
 Reset, 337
 Set, 337, 338
 SetCopyChildren, 338
 SetDescription, 338
 SetInclude, 338
 SetName, 338
 SetParent, 338
 SetRequired, 338
 ToString, 338
 Update, 338
 sdf::Param, 642
 ~Param, 644
 Clone, 644
 description, 647
 Get, 644, 645
 GetAsString, 645
 GetDefaultAsString, 645
 GetDescription, 645
 GetKey, 645
 GetRequired, 645
 GetSet, 645
 GetTypeName, 645
 IsBool, 645
 IsChar, 645
 IsColor, 645
 IsDouble, 646
 IsFloat, 646
 IsInt, 646
 IsPose, 646
 IsQuaternion, 646
 IsStr, 646
 IsTime, 646
 IsUInt, 646
 IsVector2d, 646
 IsVector2i, 646
 IsVector3, 646
 key, 647
 Param, 644
 required, 647
 Reset, 646
 Set, 646, 647
 set, 647
 SetDescription, 647
 SetFromString, 647
 SetUpdateFunc, 647
 typeName, 647
 Update, 647
 updateFunc, 648
 sdf::ParamT

- ~ParamT, 649
- Clone, 650
- defaultValue, 651
- GetAsString, 650
- GetDefaultAsString, 650
- GetDefaultValue, 650
- GetValue, 650
- operator<<, 651
- operator*, 650
- ParamT, 649
- Reset, 650
- Set, 650
- SetFromString, 650
- SetValue, 650
- Update, 651
- value, 651
- sdf::ParamT< T >, 648
- sdf::Plugin, 674
 - Clear, 674
 - data, 675
 - filename, 675
 - name, 675
 - Plugin, 674
 - Print, 674
- sdf::SDF, 762
 - PrintDescription, 763
 - PrintDoc, 763
 - PrintValues, 763
 - PrintWiki, 763
 - root, 763
 - SDF, 763
 - SetFromString, 763
 - ToString, 763
 - version, 763
 - Write, 763
- sec
 - gazebo::common::Time, 859
- SecToNano
 - gazebo::common::Time, 858
- SelectVisual
 - gazebo::rendering::Scene, 758
- SelectionObj
 - gazebo::rendering::SelectionObj, 764
- SelectionObj.hh, 1141
- self_collide
 - urdf2gazebo::GazeboExtension, 373
- SendMessage
 - gazebo::transport::Publisher, 696
- Sensor
 - gazebo::sensors::Sensor, 767
- Sensor.hh, 1142
- Sensor_V
 - gazebo::sensors, 115
- SensorFactory, 772
 - SensorFactory.hh, 1143
 - SensorFactoryFn
 - gazebo::sensors, 115
 - SensorManager.hh, 1144
 - SensorPlugin
 - gazebo::SensorPlugin, 777
 - SensorPluginPtr
 - gazebo, 95
 - SensorPtr
 - gazebo::sensors, 115
 - SensorTypes.hh, 1147
 - Sensors, 83
 - create_sensor, 85
 - fini, 85
 - GZ_REGISTER_STATIC_SENSOR, 85
 - get_sensor, 85
 - init, 86
 - load, 86
 - remove_sensor, 86
 - remove_sensors, 86
 - run, 86
 - run_once, 86
 - stop, 86
 - Sensors.hh, 1145
 - SensorsInitialized
 - gazebo::sensors::SensorManager, 776
 - Server
 - gazebo::Server, 779
 - Server.hh, 1148
 - Set
 - gazebo::common::Color, 282
 - gazebo::common::NodeTransform, 568
 - gazebo::common::Time, 858
 - gazebo::math::Matrix4, 507
 - gazebo::math::Plane, 670
 - gazebo::math::Pose, 683, 684
 - gazebo::math::Quaternion, 708
 - gazebo::math::Vector2d, 893
 - gazebo::math::Vector2i, 901
 - gazebo::math::Vector3, 913
 - gazebo::math::Vector4, 923
 - Messages, 65–67
 - sdf::Element, 337, 338
 - sdf::Param, 646, 647
 - sdf::ParamT, 650
 - set
 - sdf::Param, 647
 - SetActive
 - gazebo::rendering::SelectionObj, 765
 - gazebo::sensors::CameraSensor, 259
 - gazebo::sensors::DepthCameraSensor, 319
 - gazebo::sensors::Sensor, 770
 - SetAltitude
 - gazebo::physics::BulletPlaneShape, 208

- gazebo::physics::ODEPlaneShape, 619
- gazebo::physics::PlaneShape, 673
- SetAmbient
 - gazebo::common::Material, 494
 - gazebo::rendering::Visual, 944
- SetAmbientColor
 - gazebo::rendering::Scene, 758
- SetAnchor
 - gazebo::physics::BulletBallJoint, 168
 - gazebo::physics::BulletHinge2Joint, 182
 - gazebo::physics::BulletHingeJoint, 187
 - gazebo::physics::BulletJoint, 191
 - gazebo::physics::BulletUniversalJoint, 228
 - gazebo::physics::Joint, 434
 - gazebo::physics::ODEBallJoint, 576
 - gazebo::physics::ODEHinge2Joint, 589
 - gazebo::physics::ODEHingeJoint, 593
 - gazebo::physics::ODEUniversalJoint, 637
 - gazebo::physics::ScrewJoint, 761
 - gazebo::physics::SliderJoint, 805
- SetAngle
 - gazebo::physics::Joint, 434
- SetAngleMax
 - gazebo::sensors::GpuRaySensor, 387
- SetAngleMin
 - gazebo::sensors::GpuRaySensor, 387
- SetAngularAccel
 - gazebo::physics::Link, 468
 - gazebo::physics::Model, 530
- SetAngularDamping
 - gazebo::physics::BulletLink, 196
 - gazebo::physics::Link, 469
 - gazebo::physics::ODELink, 605
- SetAngularVel
 - gazebo::physics::BulletLink, 196
 - gazebo::physics::Link, 469
 - gazebo::physics::Model, 530
 - gazebo::physics::ODELink, 605
- SetAnimation
 - gazebo::physics::Entity, 346
- SetAspectRatio
 - gazebo::rendering::Camera, 250
- SetAttenuation
 - gazebo::rendering::Light, 452
- SetAttribute
 - gazebo::physics::BulletJoint, 191
 - gazebo::physics::Joint, 434
 - gazebo::physics::ODEJoint, 598
- SetAutoCalculate
 - gazebo::math::RotationSpline, 741
 - gazebo::math::Spline, 811
- SetAutoDisable
 - gazebo::physics::BulletLink, 196
 - gazebo::physics::Link, 469
- gazebo::physics::Model, 530
- gazebo::physics::ODELink, 605
- SetAutoDisableFlag
 - gazebo::physics::PhysicsEngine, 659
- SetAxis
 - gazebo::physics::BallJoint, 144
 - gazebo::physics::BulletHinge2Joint, 182
 - gazebo::physics::BulletHingeJoint, 187
 - gazebo::physics::BulletScrewJoint, 216
 - gazebo::physics::BulletSliderJoint, 220
 - gazebo::physics::BulletUniversalJoint, 228
 - gazebo::physics::Joint, 435
 - gazebo::physics::ODEHinge2Joint, 589
 - gazebo::physics::ODEHingeJoint, 594
 - gazebo::physics::ODEScrewJoint, 624
 - gazebo::physics::ODESliderJoint, 628
 - gazebo::physics::ODEUniversalJoint, 637
- SetAxisMaterial
 - gazebo::rendering::AxisVisual, 142
- SetBackgroundColor
 - gazebo::rendering::Scene, 758
- SetBaseline
 - gazebo::rendering::MovableText, 547
- SetBindShapeTransform
 - gazebo::common::Skeleton, 789
- SetBlendFactors
 - gazebo::common::Material, 494
- SetBlendMode
 - gazebo::common::Material, 495
- SetCFM
 - gazebo::physics::ODEJoint, 598
- SetCamera
 - gazebo::rendering::WindowManager, 953
- SetCanonicalLink
 - gazebo::physics::Entity, 347
- SetCaptureData
 - gazebo::rendering::Camera, 250
- SetCastShadows
 - gazebo::rendering::Light, 452
 - gazebo::rendering::Visual, 944
- SetCategoryBits
 - gazebo::physics::BulletCollision, 173
 - gazebo::physics::Collision, 270
 - gazebo::physics::ODECollision, 581
- SetCellCount
 - gazebo::rendering::Grid, 391
- SetCellLength
 - gazebo::rendering::Grid, 392
- SetCharHeight
 - gazebo::rendering::MovableText, 547
- SetClipDist
 - gazebo::rendering::Camera, 250
- SetCmd
 - gazebo::common::PID, 667

- SetCmdMax
 - gazebo::common::PID, 667
- SetCmdMin
 - gazebo::common::PID, 667
- SetCoG
 - gazebo::physics::BulletMotionState, 199
 - gazebo::physics::Inertial, 420
- SetCol
 - gazebo::math::Matrix3, 500
- SetCollideBits
 - gazebo::physics::BulletCollision, 174
 - gazebo::physics::Collision, 270
 - gazebo::physics::ODECollision, 581
- SetCollideMode
 - gazebo::physics::Link, 469
 - gazebo::physics::Model, 530
- SetCollision
 - gazebo::physics::Collision, 271
 - gazebo::physics::ODECollision, 582
- SetCollisionShape
 - gazebo::physics::BulletCollision, 174
- SetColor
 - gazebo::rendering::Grid, 392
 - gazebo::rendering::MovableText, 547
- SetComponent
 - gazebo::common::NodeTransform, 568
- SetCompoundShapeIndex
 - gazebo::physics::BulletCollision, 174
- SetContactMaxCorrectingVel
 - gazebo::physics::ODEPhysics, 616
 - gazebo::physics::PhysicsEngine, 660
- SetContactSurfaceLayer
 - gazebo::physics::ODEPhysics, 616
 - gazebo::physics::PhysicsEngine, 660
- SetContactsEnabled
 - gazebo::physics::Collision, 271
- SetCopyChildren
 - sdf::Element, 338
- SetDGain
 - gazebo::common::PID, 667
- SetDamping
 - gazebo::physics::BulletBallJoint, 168
 - gazebo::physics::BulletHinge2Joint, 182
 - gazebo::physics::BulletHingeJoint, 187
 - gazebo::physics::BulletJoint, 191
 - gazebo::physics::BulletScrewJoint, 216
 - gazebo::physics::BulletSliderJoint, 220
 - gazebo::physics::BulletUniversalJoint, 228
 - gazebo::physics::Joint, 435
 - gazebo::physics::ODEBallJoint, 576
 - gazebo::physics::ODEHinge2Joint, 590
 - gazebo::physics::ODEHingeJoint, 594
 - gazebo::physics::ODEScrewJoint, 624
 - gazebo::physics::ODESliderJoint, 628
 - gazebo::physics::ODEUniversalJoint, 637
- SetDepthTarget
 - gazebo::rendering::DepthCamera, 316
- SetDepthWrite
 - gazebo::common::Material, 495
- SetDescription
 - sdf::Element, 338
 - sdf::Param, 647
- SetDiffuse
 - gazebo::common::Material, 495
 - gazebo::rendering::Visual, 944
- SetDiffuseColor
 - gazebo::rendering::Light, 452
- SetDirection
 - gazebo::rendering::Light, 452
- SetDistance
 - gazebo::rendering::OrbitViewController, 641
- SetDistanceRange
 - gazebo::rendering::OrbitViewController, 641
- SetERP
 - gazebo::physics::ODEJoint, 598
- SetEmissive
 - gazebo::common::Material, 495
 - gazebo::rendering::LaserVisual, 448
 - gazebo::rendering::Visual, 944
- SetEnabled
 - gazebo::common::DiagnosticManager, 322
 - gazebo::physics::BulletLink, 196
 - gazebo::physics::Link, 469
 - gazebo::physics::Model, 530
 - gazebo::physics::ODELink, 605
 - gazebo::rendering::Projector, 691
 - gazebo::rendering::ViewController, 930
- SetFiducial
 - gazebo::physics::RayShape, 721
- SetFilename
 - gazebo::physics::TrimeshShape, 868
- SetFocalPoint
 - gazebo::rendering::OrbitViewController, 641
 - gazebo::rendering::UserCamera, 884
- SetFog
 - gazebo::rendering::Scene, 758
- SetFontName
 - gazebo::rendering::MovableText, 547
- SetForce
 - gazebo::physics::BulletHinge2Joint, 182
 - gazebo::physics::BulletHingeJoint, 187
 - gazebo::physics::BulletLink, 197
 - gazebo::physics::BulletScrewJoint, 216
 - gazebo::physics::BulletSliderJoint, 220
 - gazebo::physics::BulletUniversalJoint, 228
 - gazebo::physics::Joint, 435
 - gazebo::physics::Link, 470
 - gazebo::physics::ODEHinge2Joint, 590

- gazebo::physics::ODEHingeJoint, 594
- gazebo::physics::ODELink, 605
- gazebo::physics::ODEScrewJoint, 624
- gazebo::physics::ODESliderJoint, 628
- gazebo::physics::ODEUniversalJoint, 637
- SetFromABGR
 - gazebo::common::Color, 282
- SetFromARGB
 - gazebo::common::Color, 283
- SetFromAxes
 - gazebo::math::Matrix3, 500
- SetFromAxis
 - gazebo::math::Matrix3, 501
 - gazebo::math::Quaternion, 708
- SetFromBGRA
 - gazebo::common::Color, 283
- SetFromData
 - gazebo::common::Image, 411
- SetFromDegree
 - gazebo::math::Angle, 134
- SetFromEuler
 - gazebo::math::Quaternion, 708
- SetFromHSV
 - gazebo::common::Color, 283
- SetFromRGBA
 - gazebo::common::Color, 283
- SetFromRadian
 - gazebo::math::Angle, 134
- SetFromString
 - sdf::Param, 647
 - sdf::ParamT, 650
 - sdf::SDF, 763
- SetFromYUV
 - gazebo::common::Color, 283
- SetGravity
 - gazebo::physics::BulletPhysics, 206
 - gazebo::physics::ODEPhysics, 616
 - gazebo::physics::PhysicsEngine, 660
- SetGravityMode
 - gazebo::physics::BulletLink, 197
 - gazebo::physics::Link, 470
 - gazebo::physics::Model, 531
 - gazebo::physics::ODELink, 606
- SetGrid
 - gazebo::rendering::Scene, 759
- SetHFOV
 - gazebo::rendering::Camera, 250
- SetHandle
 - gazebo::common::SkeletonNode, 801
- SetHeight
 - gazebo::rendering::Grid, 392
- SetHighStop
 - gazebo::physics::BallJoint, 144
 - gazebo::physics::BulletBallJoint, 169
- gazebo::physics::BulletHinge2Joint, 182
- gazebo::physics::BulletHingeJoint, 187
- gazebo::physics::BulletScrewJoint, 216
- gazebo::physics::BulletSliderJoint, 220
- gazebo::physics::BulletUniversalJoint, 228
- gazebo::physics::Joint, 436
- gazebo::physics::ODEJoint, 598
- SetHighlight
 - gazebo::rendering::SelectionObj, 765
- SetHighlighted
 - gazebo::rendering::Visual, 945
- SetIGain
 - gazebo::common::PID, 667
- SetIMax
 - gazebo::common::PID, 667
- SetIMin
 - gazebo::common::PID, 668
- SetIXX
 - gazebo::physics::Inertial, 420
- SetIXY
 - gazebo::physics::Inertial, 421
- SetIXZ
 - gazebo::physics::Inertial, 421
- SetIYY
 - gazebo::physics::Inertial, 421
- SetIYZ
 - gazebo::physics::Inertial, 421
- SetIZZ
 - gazebo::physics::Inertial, 421
- SetId
 - gazebo::common::SkeletonNode, 801
- SetImageHeight
 - gazebo::rendering::Camera, 250
- SetImageSize
 - gazebo::rendering::Camera, 251
- SetImageWidth
 - gazebo::rendering::Camera, 251
- SetInclude
 - sdf::Element, 338
- SetIndexCount
 - gazebo::common::SubMesh, 824
- SetInertiaMatrix
 - gazebo::physics::Inertial, 420
- SetInertial
 - gazebo::physics::Link, 470
- SetInitialRelativePose
 - gazebo::physics::Entity, 347
- SetInitialTransform
 - gazebo::common::SkeletonNode, 802
- SetInverseBindTransform
 - gazebo::common::SkeletonNode, 802
- SetJointAnimation
 - gazebo::physics::Model, 531
- SetJointPosition

- gazebo::physics::JointController, 439
- gazebo::physics::Model, 531
- SetJointPositions
 - gazebo::physics::JointController, 440
 - gazebo::physics::Model, 531
- SetKinematic
 - gazebo::physics::Link, 470
 - gazebo::physics::ODELink, 606
- SetLaserRetro
 - gazebo::physics::Collision, 271
 - gazebo::physics::Link, 470
 - gazebo::physics::Model, 531
- SetLength
 - gazebo::common::Animation, 138
 - gazebo::physics::CylinderShape, 310
 - gazebo::physics::RayShape, 721
- SetLightType
 - gazebo::rendering::Light, 452
- SetLighting
 - gazebo::common::Material, 495
- SetLineWidth
 - gazebo::rendering::Grid, 392
- SetLinearAccel
 - gazebo::physics::Link, 471
 - gazebo::physics::Model, 532
- SetLinearDamping
 - gazebo::physics::BulletLink, 197
 - gazebo::physics::Link, 471
 - gazebo::physics::ODELink, 606
- SetLinearVel
 - gazebo::physics::BulletLink, 197
 - gazebo::physics::Link, 471
 - gazebo::physics::Model, 532
 - gazebo::physics::ODELink, 606
- SetLinkWorldPose
 - gazebo::physics::Model, 532
- SetLocallyAdvertised
 - gazebo::transport::Publication, 694
- SetLowStop
 - gazebo::physics::BallJoint, 144
 - gazebo::physics::BulletBallJoint, 169
 - gazebo::physics::BulletHinge2Joint, 182
 - gazebo::physics::BulletHingeJoint, 187
 - gazebo::physics::BulletScrewJoint, 216
 - gazebo::physics::BulletSliderJoint, 220
 - gazebo::physics::BulletUniversalJoint, 228
 - gazebo::physics::Joint, 436
 - gazebo::physics::ODEJoint, 598
- SetMass
 - gazebo::physics::Inertial, 421
- SetMaterial
 - gazebo::rendering::Visual, 945
- SetMaterialIndex
 - gazebo::common::SubMesh, 824
- SetMaxContacts
 - gazebo::physics::ODEPhysics, 616
 - gazebo::physics::PhysicsEngine, 660
- SetMaxForce
 - gazebo::physics::BulletBallJoint, 169
 - gazebo::physics::BulletHinge2Joint, 183
 - gazebo::physics::BulletHingeJoint, 188
 - gazebo::physics::BulletScrewJoint, 216
 - gazebo::physics::BulletSliderJoint, 220
 - gazebo::physics::BulletUniversalJoint, 228
 - gazebo::physics::Joint, 436
 - gazebo::physics::ODEBallJoint, 576
 - gazebo::physics::ODEHinge2Joint, 590
 - gazebo::physics::ODEHingeJoint, 594
 - gazebo::physics::ODEScrewJoint, 625
 - gazebo::physics::ODESliderJoint, 628
 - gazebo::physics::ODEUniversalJoint, 637
- SetModel
 - gazebo::physics::Joint, 437
- SetModelTransform
 - gazebo::common::SkeletonNode, 802
- SetName
 - gazebo::common::Mesh, 514
 - gazebo::common::NodeAnimation, 563
 - gazebo::common::SkeletonAnimation, 794
 - gazebo::common::SkeletonNode, 802
 - gazebo::physics::Base, 155
 - gazebo::physics::Entity, 347
 - gazebo::rendering::Camera, 251
 - gazebo::rendering::Light, 452
 - gazebo::rendering::Visual, 945
 - sdf::Element, 338
- SetNormal
 - gazebo::common::SubMesh, 825
 - gazebo::physics::PlaneShape, 673
- SetNormalCount
 - gazebo::common::SubMesh, 825
- SetNormalMap
 - gazebo::rendering::Visual, 945
- SetNumVertAttached
 - gazebo::common::Skeleton, 790
- SetOperationType
 - gazebo::rendering::DynamicRenderable, 331
- SetPGain
 - gazebo::common::PID, 668
- SetParam
 - gazebo::physics::ODEHinge2Joint, 590
 - gazebo::physics::ODEHingeJoint, 594
 - gazebo::physics::ODEJoint, 599
 - gazebo::physics::ODEScrewJoint, 625
 - gazebo::physics::ODESliderJoint, 629
 - gazebo::physics::ODEUniversalJoint, 637
- SetParams
 - gazebo::Server, 779

- SetParent
 - gazebo::common::SkeletonNode, 802
 - gazebo::physics::Base, 156
 - gazebo::sensors::CameraSensor, 259
 - gazebo::sensors::DepthCameraSensor, 319
 - gazebo::sensors::Sensor, 770
 - sdf::Element, 338
- SetParentSensor
 - gazebo::rendering::GpuLaser, 378
- SetPath
 - gazebo::common::Mesh, 514
- SetPaused
 - gazebo::physics::World, 964
- SetPerPixelLighting
 - gazebo::rendering::RTShaderSystem, 746
- SetPitch
 - gazebo::rendering::OrbitViewController, 641
- SetPoint
 - gazebo::rendering::DynamicLines, 327
- SetPointSize
 - gazebo::common::Material, 495
- SetPoints
 - gazebo::physics::BulletRayShape, 212
 - gazebo::physics::ODERayShape, 621
 - gazebo::physics::RayShape, 721
- SetPose
 - gazebo::rendering::Visual, 945
- SetPosition
 - gazebo::rendering::Light, 453
 - gazebo::rendering::Visual, 945
- SetPrimitiveType
 - gazebo::common::SubMesh, 825
- SetPublication
 - gazebo::transport::Publisher, 696
- SetQuiet
 - gazebo::common::Console, 296
- SetRadius
 - gazebo::physics::BulletSphereShape, 222
 - gazebo::physics::CylinderShape, 310
 - gazebo::physics::ODESphereShape, 631
 - gazebo::physics::SphereShape, 808
- SetRange
 - gazebo::rendering::Light, 453
- SetRangeCount
 - gazebo::rendering::GpuLaser, 378
- SetRelativePose
 - gazebo::physics::Entity, 347
- SetRenderRate
 - gazebo::rendering::Camera, 251
- SetRenderTarget
 - gazebo::rendering::Camera, 251
 - gazebo::rendering::UserCamera, 884
- SetRequired
 - sdf::Element, 338
- SetRetro
 - gazebo::physics::RayShape, 722
- SetRibbonTrail
 - gazebo::rendering::Visual, 945
- SetRootNode
 - gazebo::common::Skeleton, 790
- SetRotation
 - gazebo::common::PoseKeyFrame, 689
 - gazebo::rendering::Visual, 946
- SetSID
 - gazebo::common::NodeTransform, 568
- SetSORPGSIters
 - gazebo::physics::ODEPhysics, 616
 - gazebo::physics::PhysicsEngine, 661
- SetSORPGSPreconIters
 - gazebo::physics::ODEPhysics, 616
 - gazebo::physics::PhysicsEngine, 661
- SetSORPGSW
 - gazebo::physics::ODEPhysics, 616
 - gazebo::physics::PhysicsEngine, 661
- SetSaveFramePathname
 - gazebo::rendering::Camera, 251
- SetSaveable
 - gazebo::physics::Base, 156
- SetScale
 - gazebo::math::Matrix4, 507
 - gazebo::physics::TrimeshShape, 868
 - gazebo::rendering::Visual, 946
- SetScene
 - gazebo::rendering::Camera, 252
 - gazebo::rendering::Visual, 946
- SetSceneNode
 - gazebo::rendering::Camera, 252
- SetSelected
 - gazebo::physics::Base, 156
 - gazebo::physics::Link, 471
 - gazebo::rendering::Light, 453
- setSelectedEntity
 - Events, 51
- SetSelfCollide
 - gazebo::physics::BulletLink, 197
 - gazebo::physics::Link, 471
 - gazebo::physics::ODELink, 606
- SetShadeMode
 - gazebo::common::Material, 495
- SetShaderType
 - gazebo::rendering::Visual, 946
- SetShadowsEnabled
 - gazebo::rendering::Scene, 759
- SetShape
 - gazebo::physics::Collision, 271
- SetShininess
 - gazebo::common::Material, 496
- SetShowOnTop

- gazebo::rendering::MovableText, 548
- SetSimTime
 - gazebo::physics::World, 964
- SetSize
 - gazebo::physics::BoxShape, 165
 - gazebo::physics::BulletBoxShape, 171
 - gazebo::physics::BulletCylinderShape, 176
 - gazebo::physics::CylinderShape, 311
 - gazebo::physics::ODEBoxShape, 578
 - gazebo::physics::ODECylinderShape, 584
 - gazebo::physics::PlaneShape, 673
- SetSkeleton
 - gazebo::common::Mesh, 514
- SetSkeletonPose
 - gazebo::rendering::Visual, 946
- SetSourceValues
 - gazebo::common::NodeTransform, 568
- SetSpaceld
 - gazebo::physics::ODECollision, 582
 - gazebo::physics::ODELink, 606
- SetSpaceWidth
 - gazebo::rendering::MovableText, 548
- SetSpecular
 - gazebo::common::Material, 496
 - gazebo::rendering::Visual, 947
- SetSpecularColor
 - gazebo::rendering::Light, 453
- SetSpotFalloff
 - gazebo::rendering::Light, 453
- SetSpotInnerAngle
 - gazebo::rendering::Light, 453
- SetSpotOuterAngle
 - gazebo::rendering::Light, 454
- SetState
 - gazebo::physics::Collision, 271
 - gazebo::physics::Joint, 437
 - gazebo::physics::Link, 472
 - gazebo::physics::Model, 532
 - gazebo::physics::World, 964
- SetStatic
 - gazebo::physics::Entity, 347
- setStaticFlag
 - urdf2gazebo::GazeboExtension, 373
- SetStepTime
 - gazebo::physics::BulletPhysics, 206
 - gazebo::physics::ODEPhysics, 616
 - gazebo::physics::PhysicsEngine, 661
- SetStepType
 - gazebo::physics::ODEPhysics, 617
- SetSubMeshCenter
 - gazebo::common::SubMesh, 825
- SetTension
 - gazebo::math::Spline, 811
- SetTexCoord
 - gazebo::common::SubMesh, 825
- SetTexCoordCount
 - gazebo::common::SubMesh, 826
- SetText
 - gazebo::rendering::MovableText, 548
- SetTextAlignment
 - gazebo::rendering::MovableText, 548
- SetTexture
 - gazebo::rendering::Projector, 691
- SetTextureImage
 - gazebo::common::Material, 496
- SetThreadPitch
 - gazebo::physics::ODEScrewJoint, 625
 - gazebo::physics::ScrewJoint, 761
- SetTime
 - gazebo::common::Animation, 138
- SetTolIdentity
 - gazebo::math::Quaternion, 708
- SetToMax
 - gazebo::math::Vector3, 913
- SetToMin
 - gazebo::math::Vector3, 914
- SetToWallTime
 - gazebo::common::Time, 858
- SetTorque
 - gazebo::physics::BulletLink, 197
 - gazebo::physics::Link, 472
 - gazebo::physics::ODELink, 606
- SetTransform
 - gazebo::common::SkeletonNode, 802
- SetTranslate
 - gazebo::math::Matrix4, 507
- SetTranslation
 - gazebo::common::PoseKeyFrame, 689
- SetTransparency
 - gazebo::common::Material, 496
 - gazebo::rendering::Visual, 947
- SetType
 - gazebo::common::NodeTransform, 569
 - gazebo::common::SkeletonNode, 803
- SetUpdateFunc
 - sdf::Param, 647
- SetUpdateRate
 - gazebo::physics::PhysicsEngine, 661
 - gazebo::sensors::Sensor, 771
- SetUserData
 - gazebo::rendering::Grid, 392
- SetValue
 - gazebo::common::NumericKeyFrame, 572
 - sdf::ParamT, 650
- SetVelocity
 - gazebo::physics::BulletBallJoint, 169
 - gazebo::physics::BulletHinge2Joint, 183
 - gazebo::physics::BulletHingeJoint, 188

- gazebo::physics::BulletScrewJoint, 216
- gazebo::physics::BulletSliderJoint, 220
- gazebo::physics::BulletUniversalJoint, 228
- gazebo::physics::Joint, 437
- gazebo::physics::ODEBallJoint, 576
- gazebo::physics::ODEHinge2Joint, 590
- gazebo::physics::ODEHingeJoint, 594
- gazebo::physics::ODEScrewJoint, 625
- gazebo::physics::ODESliderJoint, 629
- gazebo::physics::ODEUniversalJoint, 638
- SetVertex
 - gazebo::common::SubMesh, 826
- SetVertexCount
 - gazebo::common::SubMesh, 826
- SetVerticalAngleMax
 - gazebo::sensors::GpuRaySensor, 387
- SetVerticalAngleMin
 - gazebo::sensors::GpuRaySensor, 387
- SetViewController
 - gazebo::rendering::UserCamera, 884
- SetViewportDimensions
 - gazebo::rendering::UserCamera, 885
- SetVisibilityFlags
 - gazebo::rendering::Visual, 947
- SetVisible
 - gazebo::rendering::Scene, 759
 - gazebo::rendering::Visual, 947
- SetWindowId
 - gazebo::rendering::Camera, 252
- SetWorld
 - gazebo::physics::Base, 156
- SetWorldCFM
 - gazebo::physics::ODEPhysics, 617
 - gazebo::physics::PhysicsEngine, 662
- SetWorldERP
 - gazebo::physics::ODEPhysics, 617
 - gazebo::physics::PhysicsEngine, 662
- SetWorldPose
 - gazebo::physics::BulletMotionState, 199
 - gazebo::physics::Entity, 348
 - gazebo::rendering::Camera, 252
 - gazebo::rendering::UserCamera, 885
 - gazebo::rendering::Visual, 947
- SetWorldPosition
 - gazebo::physics::BulletMotionState, 199
 - gazebo::rendering::Camera, 252
 - gazebo::rendering::Visual, 948
- SetWorldRotation
 - gazebo::physics::BulletMotionState, 199
 - gazebo::rendering::Camera, 252
 - gazebo::rendering::Visual, 948
- setWorldTransform
 - gazebo::physics::BulletMotionState, 200
- SetWorldTwist
 - gazebo::physics::Entity, 348
- SetYaw
 - gazebo::rendering::OrbitViewController, 641
- ShadeMode
 - gazebo::common::Material, 491
- shadeMode
 - gazebo::common::Material, 497
- ShadeModeStr
 - gazebo::common::Material, 497
- Shape
 - gazebo::physics::Shape, 781
- shape
 - gazebo::physics::Collision, 272
- Shape.hh, 1149
- ShapePtr
 - gazebo::physics, 109
- shift
 - gazebo::common::MouseEvent, 542
- shininess
 - gazebo::common::Material, 497
- Show
 - gazebo::rendering::GUIOverlay, 397
- ShowBoundingBox
 - gazebo::rendering::Visual, 948
- ShowCOM
 - gazebo::rendering::Visual, 948
- ShowCollision
 - gazebo::rendering::Visual, 948
- ShowJoints
 - gazebo::rendering::Visual, 948
- ShowRotation
 - gazebo::rendering::ArrowVisual, 140
 - gazebo::rendering::AxisVisual, 143
- ShowSkeleton
 - gazebo::rendering::Visual, 948
- ShowVisual
 - gazebo::rendering::Light, 454
- ShowWireframe
 - gazebo::rendering::Camera, 253
- Shutdown
 - gazebo::transport::Connection, 292
- sid
 - gazebo::common::NodeTransform, 569
- Signal
 - gazebo::event::EventT, 362–364
- simTime
 - gazebo::physics::State, 815
- SingletonT
 - ~SingletonT, 784
 - Instance, 784
 - SingletonT, 784
 - SingletonT, 784
- SingletonT< T >, 782
- SingletonT.hh, 1150

- size
 - gazebo::math::Plane, 671
- skelAnimation
 - gazebo::physics::Actor, 126
- skelNodesMap
 - gazebo::physics::Actor, 126
- Skeleton
 - gazebo::common::Skeleton, 786
- skeleton
 - gazebo::physics::Actor, 126
- Skeleton.hh, 1151
- SkeletonAnimation
 - gazebo::common::SkeletonAnimation, 792
- SkeletonAnimation.hh, 1153
- SkeletonNode
 - gazebo::common::SkeletonNode, 797
- SkeletonNodeType
 - gazebo::common::SkeletonNode, 797
- skinFile
 - gazebo::physics::Actor, 126
- skinScale
 - gazebo::physics::Actor, 127
- SkyX, 120
- skyx
 - gazebo::rendering::Scene, 760
- Slerp
 - gazebo::math::Quaternion, 708
- SliderJoint
 - gazebo::physics::SliderJoint, 805
- SliderJoint.hh, 1154
- slip1
 - gazebo::physics::SurfaceParams, 833
- slip2
 - gazebo::physics::SurfaceParams, 833
- SnapVisualToNearestBelow
 - gazebo::rendering::Scene, 759
- source
 - gazebo::common::NodeTransform, 569
- spaceld
 - gazebo::physics::ODECollision, 582
- specular
 - gazebo::common::Material, 497
- SphereShape
 - gazebo::physics::SphereShape, 807
- SphereShape.hh, 1155
- SphereShapePtr
 - gazebo::physics, 109
- Spline
 - gazebo::math::Spline, 809
- Spline.hh, 1156
- Squad
 - gazebo::math::Quaternion, 709
- Stamp
 - Messages, 67
- Start
 - gazebo::common::Timer, 861
- startDelay
 - gazebo::physics::Actor, 127
- StartRead
 - gazebo::transport::Connection, 292
- startTime
 - gazebo::physics::TrajectoryInfo, 866
- State
 - gazebo::physics::State, 814
- State.hh, 1158
- step
 - Events, 51
- StepWorld
 - gazebo::physics::World, 964
- Stop
 - gazebo::Master, 488
 - gazebo::physics::Actor, 124
 - gazebo::physics::World, 964
 - gazebo::sensors::SensorManager, 776
 - gazebo::Server, 779
 - gazebo::transport::ConnectionManager, 294
 - gazebo::transport::IOManager, 423
- stop
 - Events, 51
 - gazebo, 95
 - Sensors, 86
 - Transport, 91
- stop_cfm
 - urdf2gazebo::GazeboExtension, 373
- stop_erp
 - urdf2gazebo::GazeboExtension, 373
- stop_world
 - Classes for physics and dynamics, 74
- stop_worlds
 - Classes for physics and dynamics, 74
- StopAnimation
 - gazebo::physics::Entity, 348
 - gazebo::physics::Model, 533
- StopRead
 - gazebo::transport::Connection, 292
- StrStr_M
 - gazebo::common, 98
- StripSceneName
 - gazebo::rendering::Scene, 759
- StripWorldName
 - gazebo::physics::World, 965
- SubMesh
 - gazebo::common::SubMesh, 819
- SubNodeMap
 - gazebo::transport::TopicManager, 863
- subSampling
 - gazebo::physics::HeightmapShape, 404
- Subscribe

- gazebo::transport::ConnectionManager, 294
- gazebo::transport::Node, 559
- gazebo::transport::TopicManager, 864
- SubscribeOptions
 - gazebo::transport::SubscribeOptions, 827
- SubscribeOptions.hh, 1161
- Subscriber
 - gazebo::transport::Subscriber, 828
- Subscriber.hh, 1163
- SubscriberPtr
 - gazebo::transport, 117
- SubscriptionTransport
 - gazebo::transport::SubscriptionTransport, 829
- SubscriptionTransport.hh, 1165
- SubscriptionTransportPtr
 - gazebo::transport, 117
- SurfaceParams
 - gazebo::physics::SurfaceParams, 831
- SurfaceParams.hh, 1166
- SurfaceParamsPtr
 - gazebo::physics, 109
- SystemPaths.hh, 1167
 - GetCurrentDir, 1168
 - LINUX, 1168
- SystemPlugin
 - gazebo::SystemPlugin, 839
- SystemPluginPtr
 - gazebo, 95
- systemPluginsArgc
 - gazebo::Server, 779
- systemPluginsArgv
 - gazebo::Server, 779
- TPtr
 - gazebo::PluginT, 676
- TRANSLATE
 - gazebo::common::NodeTransform, 566
- TRIANGLES
 - gazebo::common::SubMesh, 819
- TRIFANS
 - gazebo::common::SubMesh, 819
- TRIMESH_SHAPE
 - gazebo::physics::Base, 150
- TRISTRIPS
 - gazebo::common::SubMesh, 819
- tangents
 - gazebo::math::RotationSpline, 742
 - gazebo::math::Spline, 812
- tension
 - gazebo::math::Spline, 812
- texImage
 - gazebo::common::Material, 497
- textureHeight
 - gazebo::rendering::Camera, 255
- textureWidth
 - gazebo::rendering::Camera, 255
- threadPitch
 - gazebo::physics::ScrewJoint, 762
- Time
 - gazebo::common::Time, 843, 844
- time
 - gazebo::common::KeyFrame, 446
 - gazebo::physics::Contact, 299
- Time.hh, 1168
- timePos
 - gazebo::common::Animation, 139
- Timer
 - gazebo::common::Timer, 860
- Timer.hh, 1169
- TimerStart
 - gazebo::common::DiagnosticManager, 322
- TimerStop
 - gazebo::common::DiagnosticManager, 323
- ToString
 - sdf::Element, 338
 - sdf::SDF, 763
- Toggle
 - gazebo::rendering::Projector, 691
- ToggleShowVisual
 - gazebo::rendering::Light, 454
- ToggleShowWireframe
 - gazebo::rendering::Camera, 253
- ToggleVisible
 - gazebo::rendering::Visual, 949
- TopicManager.hh, 1170
- TrackVisual
 - gazebo::rendering::Camera, 253
- TrackVisualFromSDF
 - Messages, 67
- TrackVisualImpl
 - gazebo::rendering::Camera, 253
 - gazebo::rendering::UserCamera, 885
- trajInfo
 - gazebo::physics::Actor, 127
- trajectories
 - gazebo::physics::Actor, 127
- transform
 - gazebo::common::NodeTransform, 569
 - gazebo::common::SkeletonNode, 804
- TransformAffine
 - gazebo::math::Matrix4, 508
- transformToParentFrame
 - urdf2gazebo::URDF2Gazebo, 877
- TransformType
 - gazebo::common::NodeTransform, 566
- Translate
 - gazebo::rendering::Camera, 253
- translate

- gazebo::common::PoseKeyFrame, 689
- translated
 - gazebo::physics::TrajectoryInfo, 866
- transparency
 - gazebo::common::Material, 498
- Transport, 88
 - CallbackHelperPtr, 89
 - clear_buffers, 89
 - fini, 89
 - get_master_uri, 90
 - get_topic_namespaces, 90
 - init, 90
 - is_stopped, 90
 - pause_incoming, 90
 - request, 90
 - run, 90
 - stop, 91
- Transport.hh, 1172
- TransportTypes.hh, 1174
- TrimeshShape
 - gazebo::physics::TrimeshShape, 867
- TrimeshShape.hh, 1175
- type
 - gazebo::common::MouseEvent, 542
 - gazebo::common::NodeTransform, 569
 - gazebo::common::SkeletonNode, 804
 - gazebo::physics::TrajectoryInfo, 866
 - gazebo::PluginT, 677
- typeName
 - sdf::Param, 647
- typeString
 - gazebo::rendering::ViewController, 930
- UIntGen
 - gazebo::math, 101
- UNIVERSAL_JOINT
 - gazebo::physics::Base, 149
- UNKNOWN
 - gazebo::common::Image, 408
- URDF2Gazebo
 - urdf2gazebo::URDF2Gazebo, 873
- URealGen
 - gazebo::math, 101
- Unadvertise
 - gazebo::transport::ConnectionManager, 294
 - gazebo::transport::TopicManager, 865
- UniformIntDist
 - gazebo::math, 101
- UniformRealDist
 - gazebo::math, 101
- UniversalJoint
 - gazebo::physics::UniversalJoint, 869
- UniversalJoint.hh, 1176
- Unsubscribe
 - gazebo::transport::ConnectionManager, 294
 - gazebo::transport::Subscriber, 828
 - gazebo::transport::TopicManager, 865
- Update
 - gazebo::common::PID, 668
 - gazebo::physics::Actor, 124
 - gazebo::physics::Base, 156
 - gazebo::physics::BulletLink, 197
 - gazebo::physics::BulletRaySensor, 210
 - gazebo::physics::BulletRayShape, 212
 - gazebo::physics::Joint, 437
 - gazebo::physics::JointController, 440
 - gazebo::physics::Link, 472
 - gazebo::physics::MapShape, 487
 - gazebo::physics::Model, 533
 - gazebo::physics::MultiRayShape, 555
 - gazebo::physics::ODELink, 606
 - gazebo::physics::ODERayShape, 621
 - gazebo::physics::ODETrimeshShape, 634
 - gazebo::physics::RayShape, 722
 - gazebo::physics::TrimeshShape, 868
 - gazebo::rendering::Camera, 254
 - gazebo::rendering::DynamicLines, 328
 - gazebo::rendering::FPSViewController, 369
 - gazebo::rendering::GUIOverlay, 397
 - gazebo::rendering::MovableText, 548
 - gazebo::rendering::OrbitViewController, 642
 - gazebo::rendering::UserCamera, 885
 - gazebo::rendering::ViewController, 930
 - gazebo::rendering::Visual, 949
 - gazebo::sensors::Sensor, 771
 - gazebo::sensors::SensorManager, 776
 - sdf::Element, 338
 - sdf::Param, 647
 - sdf::ParamT, 651
- UpdateChild
 - gazebo::sensors::ImuSensor, 414
- UpdateChildrenTransforms
 - gazebo::common::SkeletonNode, 803
- UpdateCoM
 - gazebo::physics::BulletLink, 197
- UpdateCollision
 - gazebo::physics::BulletPhysics, 206
 - gazebo::physics::ODEPhysics, 617
 - gazebo::physics::PhysicsEngine, 662
- UpdateFromMsg
 - gazebo::rendering::Light, 454
 - gazebo::rendering::Visual, 949
- updateFunc
 - sdf::Param, 648
- UpdateImpl
 - gazebo::sensors::CameraSensor, 259
 - gazebo::sensors::ContactSensor, 303
 - gazebo::sensors::DepthCameraSensor, 320

- gazebo::sensors::GpuRaySensor, 388
- gazebo::sensors::RaySensor, 718
- gazebo::sensors::RFIDSensor, 729
- gazebo::sensors::RFIDTag, 731
- gazebo::sensors::Sensor, 771
- UpdateLinkSDF
 - gazebo::physics::LinkState, 477
- UpdateMass
 - gazebo::physics::Link, 472
 - gazebo::physics::ODELink, 606
- UpdateModelSDF
 - gazebo::physics::ModelState, 539
- UpdateParameters
 - gazebo::physics::Actor, 125
 - gazebo::physics::Base, 157
 - gazebo::physics::Collision, 271
 - gazebo::physics::Entity, 348
 - gazebo::physics::Inertial, 422
 - gazebo::physics::Joint, 437
 - gazebo::physics::Link, 472
 - gazebo::physics::Model, 533
- updatePeriod
 - gazebo::sensors::Sensor, 772
- UpdatePhysics
 - gazebo::physics::BulletPhysics, 206
 - gazebo::physics::ODEPhysics, 617
 - gazebo::physics::PhysicsEngine, 662
- UpdatePoint
 - gazebo::math::RotationSpline, 741
 - gazebo::math::Spline, 812
- UpdatePublications
 - gazebo::transport::TopicManager, 865
- UpdateRays
 - gazebo::physics::BulletMultiRayShape, 202
 - gazebo::physics::MultiRayShape, 555
 - gazebo::physics::ODEMultiRayShape, 609
- UpdateShaders
 - gazebo::rendering::RTShaderSystem, 746
- UpdateSurface
 - gazebo::physics::Link, 472
 - gazebo::physics::ODELink, 607
- urdf2gazebo, 120
- urdf2gazebo::GazeboExtension, 369
 - blobs, 370
 - damping_factor, 370
 - fdir1, 371
 - fudge_factor, 371
 - GazeboExtension, 370
 - gravity, 371
 - initial_joint_position, 371
 - is_damping_factor, 371
 - is_fudge_factor, 371
 - is_initial_joint_position, 371
 - is_kd, 371
 - is_kp, 371
 - is_laser_retro, 371
 - is_maxVel, 371
 - is_minDepth, 372
 - is_mu1, 372
 - is_mu2, 372
 - is_stop_cfm, 372
 - is_stop_erp, 372
 - kd, 372
 - kp, 372
 - laser_retro, 372
 - material, 372
 - maxVel, 372
 - minDepth, 372
 - mu1, 373
 - mu2, 373
 - old_link_name, 373
 - provideFeedback, 373
 - reduction_transform, 373
 - self_collide, 373
 - setStaticFlag, 373
 - stop_cfm, 373
 - stop_erp, 373
- urdf2gazebo::URDF2Gazebo, 870
 - ~URDF2Gazebo, 873
 - addKeyValue, 873
 - addTransform, 873
 - copyPose, 873
 - createCollision, 873
 - createCollisions, 873
 - createGeometry, 873
 - createInertial, 873
 - createJoint, 873
 - createLink, 873
 - createSDF, 874
 - createVisual, 874
 - createVisuals, 874
 - gazebo_extensions_, 878
 - getGeometryBoundingBox, 874
 - getKeyValueAsString, 874
 - initModelDoc, 874
 - initModelFile, 874
 - initModelString, 874
 - insertGazeboExtensionCollision, 874
 - insertGazeboExtensionJoint, 874
 - insertGazeboExtensionLink, 874
 - insertGazeboExtensionRobot, 874
 - insertGazeboExtensionVisual, 874
 - inverseTransformToParentFrame, 875
 - listGazeboExtensions, 875
 - parseGazeboExtension, 875
 - parseVector3, 875
 - printCollisionGroups, 875
 - printMass, 875

- reduceCollisionToParent, 875
- reduceCollisionsToParent, 875
- reduceFixedJoints, 876
- reduceGazeboExtensionContactSensorFrame-Replace, 876
- reduceGazeboExtensionFrameReplace, 876
- reduceGazeboExtensionGripperFrameReplace, 876
- reduceGazeboExtensionJointFrameReplace, 876
- reduceGazeboExtensionPluginFrameReplace, 876
- reduceGazeboExtensionProjectorFrameReplace, 876
- reduceGazeboExtensionProjectorTransformReduction, 876
- reduceGazeboExtensionSensorTransformReduction, 876
- reduceGazeboExtensionToParent, 877
- reduceGazeboExtensionsTransformReduction, 877
- reduceInertialToParent, 877
- reduceJointsToParent, 877
- reduceVisualToParent, 877
- reduceVisualsToParent, 877
- transformToParentFrame, 877
- URDF2Gazebo, 873
- values2str, 877
- vector32str, 878
- UserCamera
 - gazebo::rendering::UserCamera, 880
- UserCamera.hh, 1177
- UserCameraPtr
 - gazebo::rendering, 113
- V_ABOVE
 - gazebo::rendering::MovableText, 545
- V_BELOW
 - gazebo::rendering::MovableText, 545
- VEL
 - gazebo::physics::Joint, 426
- VERTEX
 - gazebo::rendering::RenderEngine, 724
- VISUAL
 - gazebo::physics::Base, 149
- VISUAL_PLUGIN
 - Common, 36
- Valid
 - gazebo::common::Image, 412
- value
 - gazebo::common::NumericKeyFrame, 573
 - sdf::ParamT, 651
- values2str
 - urdf2gazebo::URDF2Gazebo, 877
- vang
 - gazebo::sensors::GpuRaySensor, 388
- variance
 - Math, 57
- Vector2d
 - gazebo::math::Vector2d, 887
- Vector2d.hh, 1178
- Vector2i
 - gazebo::math::Vector2i, 895, 896
- Vector2i.hh, 1179
- Vector3
 - gazebo::math::Vector3, 905
- Vector3.hh, 1180
- vector32str
 - urdf2gazebo::URDF2Gazebo, 878
- Vector4
 - gazebo::math::Vector4, 917
- Vector4.hh, 1181
- version
 - sdf::SDF, 763
- VertAlign
 - gazebo::rendering::MovableText, 545
- vertElem
 - gazebo::physics::MultiRayShape, 556
 - gazebo::sensors::GpuRaySensor, 389
- vertSize
 - gazebo::physics::HeightmapShape, 404
- vertexBufferCapacity
 - gazebo::rendering::DynamicRenderable, 331
- vertexIndex
 - gazebo::common::NodeAssignment, 564
- vfov
 - gazebo::sensors::GpuRaySensor, 389
- Video
 - gazebo::common::Video, 925
- Video.hh, 1183
- VideoVisual
 - gazebo::rendering::VideoVisual, 927
- VideoVisual.hh, 1184
- ViewController
 - gazebo::rendering::ViewController, 928
- ViewController.hh, 1185
- viewport
 - gazebo::rendering::Camera, 255
- visPub
 - gazebo::physics::Entity, 349
- visitRenderables
 - gazebo::rendering::MovableText, 548
- Visual
 - gazebo::rendering::Visual, 935
- Visual.hh, 1186
- VisualFromSDF
 - Messages, 67
- visualMsg
 - gazebo::physics::Entity, 349
- visualName
 - gazebo::physics::Actor, 127
- VisualPlugin

- gazebo::VisualPlugin, 950
- VisualPluginPtr
 - gazebo, 95
- VisualPtr
 - gazebo::rendering, 113
 - Gazebo_parser, 82
- visuals
 - gazebo::physics::Link, 473
- w
 - gazebo::math::Quaternion, 710
 - gazebo::math::Vector4, 924
- WORLD_PLUGIN
 - Common, 36
- WaitForConnection
 - gazebo::transport::Publisher, 696
- wallTime
 - gazebo::physics::State, 815
- weight
 - gazebo::common::NodeAssignment, 564
- White
 - gazebo::common::Color, 285
- width_1st
 - gazebo::sensors::GpuRaySensor, 389
- width_2nd
 - gazebo::sensors::GpuRaySensor, 389
- windowId
 - gazebo::rendering::Camera, 255
- WindowManager.hh, 1187
- World
 - gazebo::physics::World, 957
- world
 - gazebo::physics::Base, 157
 - gazebo::physics::BulletJoint, 191
 - gazebo::physics::PhysicsEngine, 663
 - gazebo::sensors::Sensor, 772
- World.hh, 1188
- worldCreated
 - Events, 52
- WorldPlugin
 - gazebo::WorldPlugin, 966
- WorldPluginPtr
 - gazebo, 95
- WorldPtr
 - gazebo::physics, 109
- WorldState
 - gazebo::physics::WorldState, 968
- WorldState.hh, 1190
- worldUpdateEnd
 - Events, 52
- worldUpdateStart
 - Events, 52
- Write
 - sdf::SDF, 763
- writeCount
 - gazebo::transport::Connection, 292
- x
 - gazebo::math::Quaternion, 710
 - gazebo::math::Vector2d, 893
 - gazebo::math::Vector2i, 902
 - gazebo::math::Vector3, 914
 - gazebo::math::Vector4, 924
- X_POSITION
 - BVHLoader.hh, 997
- X_ROTATION
 - BVHLoader.hh, 997
- y
 - gazebo::math::Quaternion, 710
 - gazebo::math::Vector2d, 893
 - gazebo::math::Vector2i, 902
 - gazebo::math::Vector3, 915
 - gazebo::math::Vector4, 924
- Y_POSITION
 - BVHLoader.hh, 997
- Y_ROTATION
 - BVHLoader.hh, 997
- Yellow
 - gazebo::common::Color, 285
- z
 - gazebo::math::Quaternion, 710
 - gazebo::math::Vector3, 915
 - gazebo::math::Vector4, 924
- Z_POSITION
 - BVHLoader.hh, 997
- Z_ROTATION
 - BVHLoader.hh, 997
- ZERO
 - gazebo::math::Matrix4, 508