Gazebo

1.6.3

Generated by Doxygen 1.8.2

Tue Apr 16 2013 16:08:28

Contents

1 Gazebo API Reference **1**

2 Todo List **3**

3 Module Index **5**

3.1 Modules 5

4 Namespace Index **7**

4.1 Namespace List 7

5 Hierarchical Index **9**

5.1 Class Hierarchy 9

6 Class Index **15**

6.1 Class List 15

7 File Index **23**

7.1 File List 23

8 Module Documentation **27**

8.1 Common 27

8.1.1 Detailed Description 31

8.1.2 Macro Definition Documentation 31

8.1.2.1 gzclr_end 31

8.1.2.2 gzclr_start 31

8.1.2.3 gzdbg 31

8.1.2.4 gzerr 31

8.1.2.5 gzlog 31

8.1.2.6 gzmsg 31

8.1.2.7 gzthrow 31

8.1.2.8 gzwarn 32

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

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

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

8.5.3.31 Stamp . . . 63
8.5.3.32 Stamp . . . 63
8.5.3.33 TrackVisualFromSDF . . . 63
8.5.3.34 VisualFromSDF . . . 64
8.6 Rendering . . . 65
8.6.1 Detailed Description . . . 67
8.6.2 Function Documentation . . . 67
8.6.2.1 create_scene . . . 67
8.6.2.2 fini . . . 67
8.6.2.3 get_scene . . . 67
8.6.2.4 init . . . 67
8.6.2.5 load . . . 67
8.6.2.6 remove_scene . . . 67
8.7 Gazebo_parser . . . 69
8.7.1 Detailed Description . . . 69
8.7.2 Typedef Documentation . . . 69
8.7.2.1 ConstUrdfLinkPtr . . . 69
8.7.2.2 UrdfCollisionPtr . . . 69
8.7.2.3 UrdfLinkPtr . . . 69
8.7.2.4 UrdfVisualPtr . . . 69
8.8 Sensors . . . 70
8.8.1 Detailed Description . . . 71
8.8.2 Macro Definition Documentation . . . 71
8.8.2.1 GZ_REGISTER_STATIC_SENSOR . . . 71
8.8.3 Function Documentation . . . 72
8.8.3.1 create_sensor . . . 72
8.8.3.2 fini . . . 72
8.8.3.3 get_sensor . . . 72
8.8.3.4 init . . . 72
8.8.3.5 load . . . 73
8.8.3.6 remove_sensor . . . 73
8.8.3.7 remove_sensors . . . 73
8.8.3.8 run . . . 73
8.8.3.9 run_once . . . 73
8.8.3.10 run_threads . . . 73
8.8.3.11 stop . . . 74
8.9 Transport . . . 75

8.9.1 Detailed Description . . . 76
8.9.2 Typedef Documentation . . . 76
8.9.2.1 CallbackHelperPtr . . . 76
8.9.3 Function Documentation . . . 77
8.9.3.1 clear_buffers . . . 77
8.9.3.2 fini . . . 77
8.9.3.3 get_master_uri . . . 77
8.9.3.4 get_topic_namespaces . . . 77
8.9.3.5 getAdvertisedTopics . . . 77
8.9.3.6 getAdvertisedTopics . . . 77
8.9.3.7 getTopicMsgType . . . 78
8.9.3.8 init . . . 78
8.9.3.9 is_stopped . . . 78
8.9.3.10 pause_incoming . . . 78
8.9.3.11 request . . . 79
8.9.3.12 requestNoReply . . . 79
8.9.3.13 requestNoReply . . . 79
8.9.3.14 run . . . 79
8.9.3.15 stop . . . 79
8.10 Utility . . . 80
8.10.1 Detailed Description . . . 80
8.10.2 Macro Definition Documentation . . . 80
8.10.2.1 DIAG_TIMER_LAP . . . 80
8.10.2.2 DIAG_TIMER_START . . . 80
8.10.2.3 DIAG_TIMER_STOP . . . 80

9 Namespace Documentation **81**

9.1 boost Namespace Reference . . . 81
9.2 gazebo Namespace Reference . . . 81
9.2.1 Detailed Description . . . 82
9.2.2 Typedef Documentation . . . 83
9.2.2.1 GUIPluginPtr . . . 83
9.2.2.2 ModelPluginPtr . . . 83
9.2.2.3 SensorPluginPtr . . . 83
9.2.2.4 SystemPluginPtr . . . 83
9.2.2.5 VisualPluginPtr . . . 83
9.2.2.6 WorldPluginPtr . . . 83

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

9.5.2.7 URealGen . . . 89
9.6 gazebo::msgs Namespace Reference . . . 89
9.6.1 Detailed Description . . . 91
9.6.2 Typedef Documentation . . . 91
9.6.2.1 MsgFactoryFn . . . 91
9.7 gazebo::physics Namespace Reference . . . 91
9.7.1 Detailed Description . . . 95
9.7.2 Typedef Documentation . . . 95
9.7.2.1 Actor_V . . . 95
9.7.2.2 ActorPtr . . . 95
9.7.2.3 Base_V . . . 95
9.7.2.4 BasePtr . . . 95
9.7.2.5 BoxShapePtr . . . 95
9.7.2.6 Collision_V . . . 95
9.7.2.7 CollisionPtr . . . 95
9.7.2.8 ContactPtr . . . 95
9.7.2.9 CylinderShapePtr . . . 95
9.7.2.10 EntityPtr . . . 95
9.7.2.11 HeightmapShapePtr . . . 95
9.7.2.12 InertialPtr . . . 95
9.7.2.13 Joint_V . . . 95
9.7.2.14 JointController_V . . . 95
9.7.2.15 JointControllerPtr . . . 95
9.7.2.16 JointPtr . . . 95
9.7.2.17 Link_V . . . 96
9.7.2.18 LinkPtr . . . 96
9.7.2.19 MeshShapePtr . . . 96
9.7.2.20 Model_V . . . 96
9.7.2.21 ModelPtr . . . 96
9.7.2.22 MultiRayShapePtr . . . 96
9.7.2.23 PhysicsEnginePtr . . . 96
9.7.2.24 RayShapePtr . . . 96
9.7.2.25 RoadPtr . . . 96
9.7.2.26 ShapePtr . . . 96
9.7.2.27 SphereShapePtr . . . 96
9.7.2.28 SurfaceParamsPtr . . . 96
9.7.2.29 WorldPtr . . . 96

9.8 gazebo::rendering Namespace Reference . . . 96
9.8.1 Detailed Description . . . 99
9.8.2 Typedef Documentation . . . 99
9.8.2.1 ArrowVisualPtr . . . 99
9.8.2.2 AxisVisualPtr . . . 99
9.8.2.3 CameraPtr . . . 99
9.8.2.4 CameraVisualPtr . . . 99
9.8.2.5 COMVisualPtr . . . 99
9.8.2.6 ContactVisualPtr . . . 99
9.8.2.7 DepthCameraPtr . . . 99
9.8.2.8 DynamicLinesPtr . . . 99
9.8.2.9 GpuLaserPtr . . . 99
9.8.2.10 JointVisualPtr . . . 99
9.8.2.11 LaserVisualPtr . . . 99
9.8.2.12 LightPtr . . . 99
9.8.2.13 RFIDTagVisualPtr . . . 99
9.8.2.14 RFIDVisualPtr . . . 99
9.8.2.15 ScenePtr . . . 99
9.8.2.16 UserCameraPtr . . . 99
9.8.2.17 VisualPtr . . . 99
9.8.3 Enumeration Type Documentation . . . 99
9.8.3.1 RenderOpType . . . 99
9.9 gazebo::sensors Namespace Reference . . . 100
9.9.1 Detailed Description . . . 102
9.9.2 Typedef Documentation . . . 102
9.9.2.1 CameraSensor_V . . . 102
9.9.2.2 CameraSensorPtr . . . 102
9.9.2.3 ContactSensor_V . . . 102
9.9.2.4 ContactSensorPtr . . . 102
9.9.2.5 DepthCameraSensor_V . . . 102
9.9.2.6 DepthCameraSensorPtr . . . 102
9.9.2.7 GpuRaySensor_V . . . 102
9.9.2.8 GpuRaySensorPtr . . . 102
9.9.2.9 ImuSensor_V . . . 102
9.9.2.10 ImuSensorPtr . . . 102
9.9.2.11 RaySensor_V . . . 102
9.9.2.12 RaySensorPtr . . . 102

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

9.18.3.1 addNestedModel . . . 107
9.18.3.2 copyChildren . . . 107
9.18.3.3 init . . . 107
9.18.3.4 initDoc . . . 108
9.18.3.5 initDoc . . . 108
9.18.3.6 initFile . . . 108
9.18.3.7 initFile . . . 108
9.18.3.8 initString . . . 108
9.18.3.9 initXml . . . 108
9.18.3.10 readDoc . . . 108
9.18.3.11 readDoc . . . 108
9.18.3.12 readFile . . . 108
9.18.3.13 readString . . . 108
9.18.3.14 readString . . . 108
9.18.3.15 readXml . . . 108
9.19 SkyX Namespace Reference . . . 108
9.20 urdf2gazebo Namespace Reference . . . 108
9.20.1 Detailed Description . . . 109

10 Class Documentation **111**

10.1 A Class Reference . . . 111
10.1.1 Detailed Description . . . 111
10.2 gazebo::physics::Actor Class Reference . . . 111
10.2.1 Detailed Description . . . 114
10.2.2 Constructor & Destructor Documentation . . . 114
10.2.2.1 Actor . . . 114
10.2.2.2 ∼Actor . . . 114
10.2.3 Member Function Documentation . . . 114
10.2.3.1 Fini . . . 114
10.2.3.2 GetSDF . . . 114
10.2.3.3 Init . . . 115
10.2.3.4 IsActive . . . 115
10.2.3.5 Load . . . 115
10.2.3.6 Play . . . 115
10.2.3.7 Stop . . . 115
10.2.3.8 Update . . . 115
10.2.3.9 UpdateParameters . . . 115

10.2.4 Member Data Documentation . . . 116
10.2.4.1 active . . . 116
10.2.4.2 autoStart . . . 116
10.2.4.3 bonePosePub . . . 116
10.2.4.4 interpolateX . . . 116
10.2.4.5 lastPos . . . 116
10.2.4.6 lastScriptTime . . . 116
10.2.4.7 lastTraj . . . 116
10.2.4.8 loop . . . 116
10.2.4.9 mainLink . . . 116
10.2.4.10 mesh . . . 116
10.2.4.11 oldAction . . . 116
10.2.4.12 pathLength . . . 117
10.2.4.13 playStartTime . . . 117
10.2.4.14 prevFrameTime . . . 117
10.2.4.15 scriptLength . . . 117
10.2.4.16 skelAnimation . . . 117
10.2.4.17 skeleton . . . 117
10.2.4.18 skelNodesMap . . . 117
10.2.4.19 skinFile . . . 117
10.2.4.20 skinScale . . . 117
10.2.4.21 startDelay . . . 117
10.2.4.22 trajectories . . . 117
10.2.4.23 trajInfo . . . 118
10.2.4.24 visualName . . . 118
10.3 gazebo::math::Angle Class Reference . . . 118
10.3.1 Detailed Description . . . 119
10.3.2 Constructor & Destructor Documentation . . . 119
10.3.2.1 Angle . . . 119
10.3.2.2 Angle . . . 119
10.3.2.3 Angle . . . 120
10.3.2.4 ∼Angle . . . 120
10.3.3 Member Function Documentation . . . 120
10.3.3.1 Degree . . . 120
10.3.3.2 Normalize . . . 120
10.3.3.3 operator!= . . . 120
10.3.3.4 operator∗ . . . 120

10.3.3.5 operator* . . . 120
10.3.3.6 operator*= . . . 121
10.3.3.7 operator+ . . . 121
10.3.3.8 operator+= . . . 121
10.3.3.9 operator- . . . 121
10.3.3.10 operator-= . . . 122
10.3.3.11 operator/ . . . 122
10.3.3.12 operator/= . . . 122
10.3.3.13 operator< . . . 122
10.3.3.14 operator<= . . . 123
10.3.3.15 operator== . . . 123
10.3.3.16 operator> . . . 123
10.3.3.17 operator>= . . . 123
10.3.3.18 Radian . . . 124
10.3.3.19 SetFromDegree . . . 124
10.3.3.20 SetFromRadian . . . 124
10.3.4 Friends And Related Function Documentation . . . 124
10.3.4.1 operator<< . . . 124
10.3.4.2 operator>> . . . 125
10.4 gazebo::common::Animation Class Reference . . . 125
10.4.1 Detailed Description . . . 126
10.4.2 Member Typedef Documentation . . . 126
10.4.2.1 KeyFrame_V . . . 126
10.4.3 Constructor & Destructor Documentation . . . 127
10.4.3.1 Animation . . . 127
10.4.3.2 ~Animation . . . 127
10.4.4 Member Function Documentation . . . 127
10.4.4.1 AddTime . . . 127
10.4.4.2 GetKeyFrame . . . 127
10.4.4.3 GetKeyFrameCount . . . 127
10.4.4.4 GetKeyFramesAtTime . . . 127
10.4.4.5 GetLength . . . 128
10.4.4.6 GetTime . . . 128
10.4.4.7 SetLength . . . 128
10.4.4.8 SetTime . . . 128
10.4.5 Member Data Documentation . . . 128
10.4.5.1 build . . . 128

10.4.5.2 keyFrames . . . 129
10.4.5.3 length . . . 129
10.4.5.4 loop . . . 129
10.4.5.5 name . . . 129
10.4.5.6 timePos . . . 129
10.5 gazebo::rendering::ArrowVisual Class Reference . . . 129
10.5.1 Detailed Description . . . 130
10.5.2 Constructor & Destructor Documentation . . . 130
10.5.2.1 ArrowVisual . . . 130
10.5.2.2 ∼ArrowVisual . . . 131
10.5.3 Member Function Documentation . . . 131
10.5.3.1 Load . . . 131
10.5.3.2 ShowRotation . . . 131
10.6 gazebo::common::AssertionInternalError Class Reference . . . 131
10.6.1 Detailed Description . . . 132
10.6.2 Constructor & Destructor Documentation . . . 132
10.6.2.1 AssertionInternalError . . . 132
10.6.2.2 ∼AssertionInternalError . . . 133
10.7 gazebo::rendering::AxisVisual Class Reference . . . 133
10.7.1 Detailed Description . . . 134
10.7.2 Constructor & Destructor Documentation . . . 134
10.7.2.1 AxisVisual . . . 134
10.7.2.2 ∼AxisVisual . . . 134
10.7.3 Member Function Documentation . . . 134
10.7.3.1 Load . . . 134
10.7.3.2 ScaleXAxis . . . 134
10.7.3.3 ScaleYAxis . . . 135
10.7.3.4 ScaleZAxis . . . 135
10.7.3.5 SetAxisMaterial . . . 135
10.7.3.6 ShowRotation . . . 135
10.8 gazebo::physics::BallJoint< T > Class Template Reference . . . 135
10.8.1 Detailed Description . . . 136
10.8.2 Constructor & Destructor Documentation . . . 136
10.8.2.1 BallJoint . . . 136
10.8.2.2 ∼BallJoint . . . 137
10.8.3 Member Function Documentation . . . 137
10.8.3.1 GetAngleCount . . . 137

10.8.3.2 GetHighStop . . . 137
10.8.3.3 GetLowStop . . . 137
10.8.3.4 Load . . . 137
10.8.3.5 SetAxis . . . 137
10.8.3.6 SetHighStop . . . 137
10.8.3.7 SetLowStop . . . 137
10.9 gazebo::physics::Base Class Reference . . . 137
10.9.1 Detailed Description . . . 140
10.9.2 Member Enumeration Documentation . . . 140
10.9.2.1 EntityType . . . 140
10.9.3 Constructor & Destructor Documentation . . . 141
10.9.3.1 Base . . . 141
10.9.3.2 ∼Base . . . 141
10.9.4 Member Function Documentation . . . 141
10.9.4.1 AddChild . . . 141
10.9.4.2 AddType . . . 142
10.9.4.3 Fini . . . 142
10.9.4.4 GetById . . . 142
10.9.4.5 GetByName . . . 142
10.9.4.6 GetChild . . . 142
10.9.4.7 GetChild . . . 143
10.9.4.8 GetChildCount . . . 143
10.9.4.9 GetId . . . 143
10.9.4.10 GetName . . . 143
10.9.4.11 GetParent . . . 143
10.9.4.12 GetParentId . . . 144
10.9.4.13 GetSaveable . . . 144
10.9.4.14 GetScopedName . . . 144
10.9.4.15 GetSDF . . . 144
10.9.4.16 GetType . . . 144
10.9.4.17 GetWorld . . . 144
10.9.4.18 HasType . . . 145
10.9.4.19 Init . . . 145
10.9.4.20 IsSelected . . . 145
10.9.4.21 Load . . . 145
10.9.4.22 operator== . . . 145
10.9.4.23 Print . . . 146

10.9.4.24 RemoveChild . . . 146
10.9.4.25 RemoveChild . . . 146
10.9.4.26 RemoveChildren . . . 146
10.9.4.27 Reset . . . 146
10.9.4.28 Reset . . . 146
10.9.4.29 SetName . . . 147
10.9.4.30 SetParent . . . 147
10.9.4.31 SetSaveable . . . 147
10.9.4.32 SetSelected . . . 147
10.9.4.33 SetWorld . . . 147
10.9.4.34 Update . . . 148
10.9.4.35 UpdateParameters . . . 148
10.9.5 Member Data Documentation . . . 148
10.9.5.1 children . . . 148
10.9.5.2 childrenEnd . . . 148
10.9.5.3 parent . . . 148
10.9.5.4 sdf . . . 148
10.9.5.5 world . . . 148
10.10 gazebo::math::Box Class Reference . . . 148
10.10.1 Detailed Description . . . 150
10.10.2 Constructor & Destructor Documentation . . . 150
10.10.2.1 Box . . . 150
10.10.2.2 Box . . . 150
10.10.2.3 Box . . . 150
10.10.2.4 ∼Box . . . 150
10.10.3 Member Function Documentation . . . 150
10.10.3.1 GetCenter . . . 150
10.10.3.2 GetSize . . . 150
10.10.3.3 GetXLength . . . 151
10.10.3.4 GetYLength . . . 151
10.10.3.5 GetZLength . . . 151
10.10.3.6 Merge . . . 151
10.10.3.7 operator+ . . . 151
10.10.3.8 operator+= . . . 151
10.10.3.9 operator- . . . 152
10.10.3.10 operator= . . . 152
10.10.3.11 operator== . . . 152

10.10.4 Friends And Related Function Documentation . . . 152
10.10.4.1 operator<< . . . 153
10.10.5 Member Data Documentation . . . 153
10.10.5.1 max . . . 153
10.10.5.2 min . . . 153
10.11 gazebo::physics::BoxShape Class Reference . . . 153
10.11.1 Detailed Description . . . 154
10.11.2 Constructor & Destructor Documentation . . . 155
10.11.2.1 BoxShape . . . 155
10.11.2.2 ~BoxShape . . . 155
10.11.3 Member Function Documentation . . . 155
10.11.3.1 FillMsg . . . 155
10.11.3.2 GetSize . . . 155
10.11.3.3 Init . . . 155
10.11.3.4 ProcessMsg . . . 155
10.11.3.5 SetSize . . . 156
10.12 gazebo::common::BVHLoader Class Reference . . . 156
10.12.1 Detailed Description . . . 156
10.12.2 Constructor & Destructor Documentation . . . 156
10.12.2.1 BVHLoader . . . 156
10.12.2.2 ~BVHLoader . . . 156
10.12.3 Member Function Documentation . . . 156
10.12.3.1 Load . . . 156
10.13 gazebo::transport::CallbackHelper Class Reference . . . 157
10.13.1 Detailed Description . . . 158
10.13.2 Constructor & Destructor Documentation . . . 158
10.13.2.1 CallbackHelper . . . 158
10.13.2.2 ~CallbackHelper . . . 158
10.13.3 Member Function Documentation . . . 158
10.13.3.1 GetId . . . 158
10.13.3.2 GetLatching . . . 158
10.13.3.3 GetMsgType . . . 159
10.13.3.4 HandleData . . . 159
10.13.3.5 HandleMessage . . . 159
10.13.3.6 IsLocal . . . 159
10.13.4 Member Data Documentation . . . 160
10.13.4.1 latching . . . 160

10.14gazebo::transport::CallbackHelperT< M > Class Template Reference . . . 160
10.14.1 Detailed Description . . . 161
10.14.2 Constructor & Destructor Documentation . . . 161
10.14.2.1 CallbackHelperT . . . 161
10.14.3 Member Function Documentation . . . 161
10.14.3.1 GetMsgType . . . 161
10.14.3.2 HandleData . . . 161
10.14.3.3 HandleMessage . . . 162
10.14.3.4 IsLocal . . . 162
10.15gazebo::rendering::Camera Class Reference . . . 162
10.15.1 Detailed Description . . . 169
10.15.2 Constructor & Destructor Documentation . . . 169
10.15.2.1 Camera . . . 169
10.15.2.2 ∼Camera . . . 169
10.15.3 Member Function Documentation . . . 169
10.15.3.1 AnimationComplete . . . 169
10.15.3.2 AttachToVisual . . . 169
10.15.3.3 AttachToVisualImpl . . . 169
10.15.3.4 AttachToVisualImpl . . . 170
10.15.3.5 ConnectNewImageFrame . . . 170
10.15.3.6 CreateRenderTexture . . . 170
10.15.3.7 DisconnectNewImageFrame . . . 171
10.15.3.8 EnableSaveFrame . . . 171
10.15.3.9 Fini . . . 171
10.15.3.10 GetAspectRatio . . . 171
10.15.3.11 GetAvgFPS . . . 171
10.15.3.12 GetCameraToViewportRay . . . 171
10.15.3.13 GetDirection . . . 172
10.15.3.14 GetFarClip . . . 172
10.15.3.15 GetFrameFilename . . . 172
10.15.3.16 GetHFOV . . . 172
10.15.3.17 GetImageByteSize . . . 172
10.15.3.18 GetImageByteSize . . . 173
10.15.3.19 GetImageData . . . 173
10.15.3.20 GetImageDepth . . . 173
10.15.3.21 GetImageFormat . . . 173
10.15.3.22 GetImageHeight . . . 173

10.15.3.23GetImageWidth . . . 174
10.15.3.24GetInitialized . . . 174
10.15.3.25GetLastRenderWallTime . . . 174
10.15.3.26GetName . . . 174
10.15.3.27GetNearClip . . . 174
10.15.3.28GetOgreCamera . . . 175
10.15.3.29GetPitchNode . . . 175
10.15.3.30GetRenderRate . . . 175
10.15.3.31GetRenderTexture . . . 175
10.15.3.32GetRight . . . 175
10.15.3.33GetScene . . . 175
10.15.3.34GetSceneNode . . . 176
10.15.3.35GetScreenshotPath . . . 176
10.15.3.36GetTextureHeight . . . 176
10.15.3.37GetTextureWidth . . . 176
10.15.3.38GetTriangleCount . . . 176
10.15.3.39GetUp . . . 176
10.15.3.40GetVFOV . . . 177
10.15.3.41GetViewport . . . 177
10.15.3.42GetViewportHeight . . . 177
10.15.3.43GetViewportWidth . . . 177
10.15.3.44GetWindowId . . . 177
10.15.3.45GetWorldPointOnPlane . . . 177
10.15.3.46GetWorldPose . . . 178
10.15.3.47GetWorldPosition . . . 178
10.15.3.48GetWorldRotation . . . 178
10.15.3.49GetZValue . . . 178
10.15.3.50Init . . . 178
10.15.3.51IsAnimating . . . 179
10.15.3.52IsInitialized . . . 179
10.15.3.53IsVisible . . . 179
10.15.3.54IsVisible . . . 179
10.15.3.55Load . . . 179
10.15.3.56Load . . . 179
10.15.3.57MoveToPosition . . . 180
10.15.3.58MoveToPositions . . . 180
10.15.3.59PostRender . . . 180

10.15.3.60Render . . . 180
10.15.3.61RenderImpl . . . 180
10.15.3.62RotatePitch . . . 181
10.15.3.63RotateYaw . . . 181
10.15.3.64SaveFrame . . . 181
10.15.3.65SaveFrame . . . 181
10.15.3.66SetAspectRatio . . . 181
10.15.3.67SetCaptureData . . . 182
10.15.3.68SetCaptureDataOnce . . . 182
10.15.3.69SetClipDist . . . 182
10.15.3.70SetHFOV . . . 182
10.15.3.71SetImageHeight . . . 182
10.15.3.72SetImageSize . . . 182
10.15.3.73SetImageWidth . . . 183
10.15.3.74SetName . . . 183
10.15.3.75SetRenderRate . . . 183
10.15.3.76SetRenderTarget . . . 183
10.15.3.77SetSaveFramePathname . . . 183
10.15.3.78SetScene . . . 183
10.15.3.79SetSceneNode . . . 184
10.15.3.80SetWindowId . . . 184
10.15.3.81SetWorldPose . . . 184
10.15.3.82SetWorldPosition . . . 184
10.15.3.83SetWorldRotation . . . 184
10.15.3.84ShowWireframe . . . 184
10.15.3.85ToggleShowWireframe . . . 184
10.15.3.86TrackVisual . . . 185
10.15.3.87TrackVisualImpl . . . 185
10.15.3.88TrackVisualImpl . . . 185
10.15.3.89Translate . . . 185
10.15.3.90Update . . . 185
10.15.4 Member Data Documentation . . . 185
10.15.4.1 animState . . . 186
10.15.4.2 bayerFrameBuffer . . . 186
10.15.4.3 camera . . . 186
10.15.4.4 captureData . . . 186
10.15.4.5 captureDataOnce . . . 186

10.15.4.6 connections . . . 186
10.15.4.7 imageFormat . . . 186
10.15.4.8 imageHeight . . . 186
10.15.4.9 imageWidth . . . 186
10.15.4.10 initialized . . . 186
10.15.4.11 lastRenderWallTime . . . 186
10.15.4.12 name . . . 187
10.15.4.13 newData . . . 187
10.15.4.14 newImageFrame . . . 187
10.15.4.15 onAnimationComplete . . . 187
10.15.4.16 pitchNode . . . 187
10.15.4.17 prevAnimTime . . . 187
10.15.4.18 renderTarget . . . 187
10.15.4.19 renderTexture . . . 187
10.15.4.20 requests . . . 187
10.15.4.21 saveCount . . . 187
10.15.4.22 saveFrameBuffer . . . 188
10.15.4.23 scene . . . 188
10.15.4.24 sceneNode . . . 188
10.15.4.25 screenshotPath . . . 188
10.15.4.26 sdf . . . 188
10.15.4.27 textureHeight . . . 188
10.15.4.28 textureWidth . . . 188
10.15.4.29 viewport . . . 188
10.15.4.30 windowId . . . 188
10.16 gazebo::sensors::CameraSensor Class Reference . . . 188
10.16.1 Detailed Description . . . 190
10.16.2 Constructor & Destructor Documentation . . . 190
10.16.2.1 CameraSensor . . . 190
10.16.2.2 ∼CameraSensor . . . 190
10.16.3 Member Function Documentation . . . 190
10.16.3.1 Fini . . . 190
10.16.3.2 GetCamera . . . 190
10.16.3.3 GetImageData . . . 190
10.16.3.4 GetImageHeight . . . 191
10.16.3.5 GetImageWidth . . . 191
10.16.3.6 GetTopic . . . 191

10.16.3.7 Init . . . 191
10.16.3.8 IsActive . . . 191
10.16.3.9 Load . . . 191
10.16.3.10Load . . . 192
10.16.3.11SaveFrame . . . 192
10.16.3.12SetParent . . . 192
10.16.3.13UpdateImpl . . . 192
10.17gazebo::rendering::CameraVisual Class Reference . . . 193
10.17.1 Detailed Description . . . 193
10.17.2 Constructor & Destructor Documentation . . . 193
10.17.2.1 CameraVisual . . . 194
10.17.2.2 ∼CameraVisual . . . 194
10.17.3 Member Function Documentation . . . 194
10.17.3.1 Load . . . 194
10.18gazebo::common::ColladaLoader Class Reference . . . 194
10.18.1 Detailed Description . . . 195
10.18.2 Constructor & Destructor Documentation . . . 195
10.18.2.1 ColladaLoader . . . 195
10.18.2.2 ∼ColladaLoader . . . 195
10.18.3 Member Function Documentation . . . 195
10.18.3.1 Load . . . 195
10.19gazebo::physics::Collision Class Reference . . . 195
10.19.1 Detailed Description . . . 198
10.19.2 Constructor & Destructor Documentation . . . 198
10.19.2.1 Collision . . . 198
10.19.2.2 ∼Collision . . . 198
10.19.3 Member Function Documentation . . . 198
10.19.3.1 AddContact . . . 198
10.19.3.2 ConnectContact . . . 199
10.19.3.3 DisconnectContact . . . 199
10.19.3.4 FillMsg . . . 199
10.19.3.5 Fini . . . 199
10.19.3.6 GetBoundingBox . . . 199
10.19.3.7 GetContactsEnabled . . . 199
10.19.3.8 GetLaserRetro . . . 199
10.19.3.9 GetLink . . . 200
10.19.3.10GetModel . . . 200

10.19.3.11GetRelativeAngularAccel . . . 200
10.19.3.12GetRelativeAngularVel . . . 200
10.19.3.13GetRelativeLinearAccel . . . 200
10.19.3.14GetRelativeLinearVel . . . 200
10.19.3.15GetShape . . . 201
10.19.3.16GetShapeType . . . 201
10.19.3.17GetState . . . 201
10.19.3.18GetSurface . . . 201
10.19.3.19GetWorldAngularAccel . . . 201
10.19.3.20GetWorldAngularVel . . . 202
10.19.3.21GetWorldLinearAccel . . . 202
10.19.3.22GetWorldLinearVel . . . 202
10.19.3.23Init . . . 202
10.19.3.24IsPlaceable . . . 202
10.19.3.25Load . . . 202
10.19.3.26ProcessMsg . . . 203
10.19.3.27SetCategoryBits . . . 203
10.19.3.28SetCollideBits . . . 203
10.19.3.29SetCollision . . . 203
10.19.3.30SetContactsEnabled . . . 203
10.19.3.31SetLaserRetro . . . 203
10.19.3.32SetShape . . . 204
10.19.3.33SetState . . . 204
10.19.3.34UpdateParameters . . . 204
10.19.4 Member Data Documentation . . . 204
10.19.4.1 link . . . 204
10.19.4.2 placeable . . . 204
10.19.4.3 shape . . . 204
10.20gazebo::physics::CollisionState Class Reference . . . 205
10.20.1 Detailed Description . . . 206
10.20.2 Constructor & Destructor Documentation . . . 206
10.20.2.1 CollisionState . . . 206
10.20.2.2 CollisionState . . . 206
10.20.2.3 CollisionState . . . 206
10.20.2.4 ∼CollisionState . . . 206
10.20.3 Member Function Documentation . . . 206
10.20.3.1 FillSDF . . . 206

10.20.3.2 GetPose . . . 207
10.20.3.3 IsZero . . . 207
10.20.3.4 Load . . . 207
10.20.3.5 operator+ . . . 207
10.20.3.6 operator- . . . 207
10.20.3.7 operator= . . . 208
10.20.4 Friends And Related Function Documentation . . . 208
10.20.4.1 operator<< . . . 208
10.21 gazebo::common::Color Class Reference . . . 208
10.21.1 Detailed Description . . . 211
10.21.2 Member Typedef Documentation . . . 211
10.21.2.1 ABGR . . . 211
10.21.2.2 ARGB . . . 211
10.21.2.3 BGRA . . . 211
10.21.2.4 RGBA . . . 211
10.21.3 Constructor & Destructor Documentation . . . 211
10.21.3.1 Color . . . 211
10.21.3.2 Color . . . 211
10.21.3.3 Color . . . 211
10.21.3.4 ∼Color . . . 212
10.21.4 Member Function Documentation . . . 212
10.21.4.1 GetAsABGR . . . 212
10.21.4.2 GetAsARGB . . . 212
10.21.4.3 GetAsBGRA . . . 212
10.21.4.4 GetAsHSV . . . 212
10.21.4.5 GetAsRGBA . . . 212
10.21.4.6 GetAsYUV . . . 213
10.21.4.7 operator!= . . . 213
10.21.4.8 operator* . . . 213
10.21.4.9 operator* . . . 213
10.21.4.10 operator*= . . . 213
10.21.4.11 operator+ . . . 214
10.21.4.12 operator+ . . . 214
10.21.4.13 operator+= . . . 214
10.21.4.14 operator- . . . 214
10.21.4.15 operator- . . . 215
10.21.4.16 operator-= . . . 215

10.21.4.17operator/ . . . 215
10.21.4.18operator/ . . . 215
10.21.4.19operator/= . . . 216
10.21.4.20operator= . . . 216
10.21.4.21operator== . . . 216
10.21.4.22operator[] . . . 216
10.21.4.23Reset . . . 217
10.21.4.24Set . . . 217
10.21.4.25SetFromABGR . . . 217
10.21.4.26SetFromARGB . . . 217
10.21.4.27SetFromBGRA . . . 217
10.21.4.28SetFromHSV . . . 218
10.21.4.29SetFromRGBA . . . 218
10.21.4.30SetFromYUV . . . 218
10.21.5 Friends And Related Function Documentation . . . 218
10.21.5.1 operator<< . . . 218
10.21.5.2 operator>> . . . 218
10.21.6 Member Data Documentation . . . 219
10.21.6.1 a . . . 219
10.21.6.2 b . . . 219
10.21.6.3 Black . . . 219
10.21.6.4 Blue . . . 219
10.21.6.5 g . . . 219
10.21.6.6 Green . . . 219
10.21.6.7 Purple . . . 219
10.21.6.8 r . . . 219
10.21.6.9 Red . . . 219
10.21.6.10White . . . 219
10.21.6.11Yellow . . . 219
10.22gazebo::rendering::COMVisual Class Reference . . . 219
10.22.1 Detailed Description . . . 220
10.22.2 Constructor & Destructor Documentation . . . 220
10.22.2.1 COMVisual . . . 220
10.22.2.2 ~COMVisual . . . 221
10.22.3 Member Function Documentation . . . 221
10.22.3.1 Load . . . 221
10.22.3.2 Load . . . 221

10.23gazebo::event::Connection Class Reference . . . 221
10.23.1 Detailed Description . . . 222
10.23.2 Constructor & Destructor Documentation . . . 222
10.23.2.1 Connection . . . 222
10.23.2.2 Connection . . . 222
10.23.2.3 ∼Connection . . . 222
10.23.3 Member Function Documentation . . . 222
10.23.3.1 GetId . . . 222
10.24gazebo::transport::Connection Class Reference . . . 222
10.24.1 Detailed Description . . . 224
10.24.2 Member Typedef Documentation . . . 224
10.24.2.1 AcceptCallback . . . 224
10.24.2.2 ReadCallback . . . 225
10.24.3 Constructor & Destructor Documentation . . . 225
10.24.3.1 Connection . . . 225
10.24.3.2 ∼Connection . . . 225
10.24.4 Member Function Documentation . . . 225
10.24.4.1 AsyncRead . . . 225
10.24.4.2 Cancel . . . 225
10.24.4.3 Connect . . . 225
10.24.4.4 ConnectToShutdown . . . 225
10.24.4.5 DisconnectShutdown . . . 226
10.24.4.6 EnqueueMsg . . . 226
10.24.4.7 GetId . . . 226
10.24.4.8 GetLocalAddress . . . 226
10.24.4.9 GetLocalHostname . . . 226
10.24.4.10 GetLocalPort . . . 227
10.24.4.11 GetLocalURI . . . 227
10.24.4.12 GetRemoteAddress . . . 227
10.24.4.13 GetRemoteHostname . . . 227
10.24.4.14 GetRemotePort . . . 227
10.24.4.15 GetRemoteURI . . . 227
10.24.4.16 IsOpen . . . 228
10.24.4.17 Listen . . . 228
10.24.4.18 ProcessWriteQueue . . . 228
10.24.4.19 Read . . . 228
10.24.4.20 Shutdown . . . 228

10.24.4.21StartRead . . . 228
10.24.4.22StopRead . . . 229
10.24.4.23ValidateIP . . . 229
10.25gazebo::transport::ConnectionManager Class Reference . . . 229
10.25.1 Detailed Description . . . 230
10.25.2 Member Function Documentation . . . 230
10.25.2.1 Advertise . . . 230
10.25.2.2 ConnectToRemoteHost . . . 231
10.25.2.3 Fini . . . 231
10.25.2.4 GetAllPublishers . . . 231
10.25.2.5 GetTopicNamespaces . . . 231
10.25.2.6 Init . . . 231
10.25.2.7 IsRunning . . . 232
10.25.2.8 RegisterTopicNamespace . . . 232
10.25.2.9 RemoveConnection . . . 232
10.25.2.10Run . . . 232
10.25.2.11RunUpdate . . . 232
10.25.2.12Stop . . . 232
10.25.2.13Subscribe . . . 232
10.25.2.14Unadvertise . . . 233
10.25.2.15Unsubscribe . . . 233
10.25.2.16Unsubscribe . . . 233
10.25.3 Member Data Documentation . . . 233
10.25.3.1 eventConnections . . . 233
10.26gazebo::transport::ConnectionReadTask Class Reference . . . 233
10.26.1 Detailed Description . . . 234
10.26.2 Constructor & Destructor Documentation . . . 234
10.26.2.1 ConnectionReadTask . . . 234
10.26.3 Member Function Documentation . . . 234
10.26.3.1 execute . . . 234
10.27gazebo::common::Console Class Reference . . . 235
10.27.1 Detailed Description . . . 235
10.28gazebo::physics::Contact Class Reference . . . 236
10.28.1 Detailed Description . . . 237
10.28.2 Constructor & Destructor Documentation . . . 237
10.28.2.1 Contact . . . 237
10.28.2.2 Contact . . . 237

10.28.2.3 ~Contact . . . 237
10.28.3 Member Function Documentation . . . 237
10.28.3.1 DebugString . . . 237
10.28.3.2 FillMsg . . . 237
10.28.3.3 operator= . . . 237
10.28.3.4 operator= . . . 238
10.28.3.5 Reset . . . 238
10.28.4 Member Data Documentation . . . 238
10.28.4.1 collision1 . . . 238
10.28.4.2 collision2 . . . 238
10.28.4.3 collisionPtr1 . . . 238
10.28.4.4 collisionPtr2 . . . 238
10.28.4.5 count . . . 238
10.28.4.6 depths . . . 239
10.28.4.7 normals . . . 239
10.28.4.8 positions . . . 239
10.28.4.9 time . . . 239
10.28.4.10 world . . . 239
10.28.4.11 wrench . . . 239
10.29 gazebo::physics::ContactManager Class Reference . . . 239
10.29.1 Detailed Description . . . 240
10.29.2 Constructor & Destructor Documentation . . . 240
10.29.2.1 ContactManager . . . 240
10.29.2.2 ~ContactManager . . . 240
10.29.3 Member Function Documentation . . . 240
10.29.3.1 Clear . . . 240
10.29.3.2 GetContact . . . 240
10.29.3.3 GetContactCount . . . 240
10.29.3.4 GetContacts . . . 241
10.29.3.5 Init . . . 241
10.29.3.6 NewContact . . . 241
10.29.3.7 PublishContacts . . . 241
10.29.3.8 ResetCount . . . 241
10.30 gazebo::sensors::ContactSensor Class Reference . . . 241
10.30.1 Detailed Description . . . 243
10.30.2 Constructor & Destructor Documentation . . . 243
10.30.2.1 ContactSensor . . . 243

10.30.2.2 ~ContactSensor . . . 243
10.30.3 Member Function Documentation . . . 243
10.30.3.1 Fini . . . 243
10.30.3.2 GetCollisionContactCount . . . 243
10.30.3.3 GetCollisionCount . . . 243
10.30.3.4 GetCollisionName . . . 244
10.30.3.5 GetContacts . . . 244
10.30.3.6 GetContacts . . . 245
10.30.3.7 Init . . . 245
10.30.3.8 IsActive . . . 245
10.30.3.9 Load . . . 245
10.30.3.10 Load . . . 245
10.30.3.11 UpdateImpl . . . 246
10.31 gazebo::rendering::ContactVisual Class Reference . . . 246
10.31.1 Detailed Description . . . 247
10.31.2 Constructor & Destructor Documentation . . . 247
10.31.2.1 ContactVisual . . . 247
10.31.2.2 ~ContactVisual . . . 247
10.31.3 Member Function Documentation . . . 247
10.31.3.1 SetEnabled . . . 247
10.32 gazebo::rendering::Conversions Class Reference . . . 247
10.32.1 Detailed Description . . . 248
10.32.2 Member Function Documentation . . . 248
10.32.2.1 Convert . . . 248
10.32.2.2 Convert . . . 248
10.32.2.3 Convert . . . 248
10.32.2.4 Convert . . . 249
10.32.2.5 Convert . . . 249
10.32.2.6 Convert . . . 249
10.33 sdf::Converter Class Reference . . . 249
10.33.1 Detailed Description . . . 250
10.33.2 Member Function Documentation . . . 250
10.33.2.1 Convert . . . 250
10.33.2.2 Convert . . . 250
10.34 gazebo::physics::CylinderShape Class Reference . . . 250
10.34.1 Detailed Description . . . 252
10.34.2 Constructor & Destructor Documentation . . . 252

10.34.2.1 CylinderShape . . . 252
10.34.2.2 ~CylinderShape . . . 252
10.34.3 Member Function Documentation . . . 252
10.34.3.1 FillMsg . . . 252
10.34.3.2 GetLength . . . 252
10.34.3.3 GetRadius . . . 252
10.34.3.4 Init . . . 253
10.34.3.5 ProcessMsg . . . 253
10.34.3.6 SetLength . . . 253
10.34.3.7 SetRadius . . . 253
10.34.3.8 SetSize . . . 253
10.35 gazebo::rendering::DepthCamera Class Reference . . . 253
10.35.1 Detailed Description . . . 255
10.35.2 Constructor & Destructor Documentation . . . 255
10.35.2.1 DepthCamera . . . 255
10.35.2.2 ~DepthCamera . . . 255
10.35.3 Member Function Documentation . . . 255
10.35.3.1 ConnectNewDepthFrame . . . 255
10.35.3.2 ConnectNewRGBPointCloud . . . 256
10.35.3.3 CreateDepthTexture . . . 256
10.35.3.4 DisconnectNewDepthFrame . . . 256
10.35.3.5 DisconnectNewRGBPointCloud . . . 256
10.35.3.6 Fini . . . 257
10.35.3.7 GetDepthData . . . 257
10.35.3.8 Init . . . 257
10.35.3.9 Load . . . 257
10.35.3.10 Load . . . 257
10.35.3.11 PostRender . . . 257
10.35.3.12 SetDepthTarget . . . 257
10.35.4 Member Data Documentation . . . 258
10.35.4.1 depthTarget . . . 258
10.35.4.2 depthTexture . . . 258
10.35.4.3 depthViewport . . . 258
10.36 gazebo::sensors::DepthCameraSensor Class Reference . . . 258
10.36.1 Constructor & Destructor Documentation . . . 259
10.36.1.1 DepthCameraSensor . . . 259
10.36.1.2 ~DepthCameraSensor . . . 259

10.36.2 Member Function Documentation . . . 259
10.36.2.1 Fini . . . 259
10.36.2.2 GetDepthCamera . . . 260
10.36.2.3 Init . . . 260
10.36.2.4 Load . . . 260
10.36.2.5 Load . . . 260
10.36.2.6 SaveFrame . . . 260
10.36.2.7 SetActive . . . 260
10.36.2.8 SetParent . . . 261
10.36.2.9 UpdateImpl . . . 261
10.37gazebo::util::DiagnosticManager Class Reference . . . 261
10.37.1 Detailed Description . . . 262
10.37.2 Member Function Documentation . . . 262
10.37.2.1 GetLabel . . . 262
10.37.2.2 GetLogPath . . . 262
10.37.2.3 GetTime . . . 263
10.37.2.4 GetTime . . . 263
10.37.2.5 GetTimerCount . . . 263
10.37.2.6 Init . . . 263
10.37.2.7 Lap . . . 263
10.37.2.8 StartTimer . . . 264
10.37.2.9 StopTimer . . . 264
10.38gazebo::util::DiagnosticTimer Class Reference . . . 264
10.38.1 Detailed Description . . . 265
10.38.2 Constructor & Destructor Documentation . . . 265
10.38.2.1 DiagnosticTimer . . . 265
10.38.2.2 ∼DiagnosticTimer . . . 265
10.38.3 Member Function Documentation . . . 265
10.38.3.1 GetName . . . 265
10.38.3.2 Lap . . . 265
10.38.3.3 Start . . . 266
10.38.3.4 Stop . . . 266
10.39gazebo::rendering::DynamicLines Class Reference . . . 266
10.39.1 Detailed Description . . . 267
10.39.2 Constructor & Destructor Documentation . . . 267
10.39.2.1 DynamicLines . . . 267
10.39.2.2 ∼DynamicLines . . . 268

10.39.3 Member Function Documentation . . . 268
10.39.3.1 AddPoint . . . 268
10.39.3.2 AddPoint . . . 268
10.39.3.3 Clear . . . 268
10.39.3.4 CreateVertexDeclaration . . . 268
10.39.3.5 FillHardwareBuffers . . . 268
10.39.3.6 GetMovableType . . . 268
10.39.3.7 getMovableType . . . 269
10.39.3.8 GetPoint . . . 269
10.39.3.9 GetPointCount . . . 269
10.39.3.10 SetPoint . . . 269
10.39.3.11 Update . . . 269
10.40 gazebo::rendering::DynamicRenderable Class Reference . . . 269
10.40.1 Detailed Description . . . 271
10.40.2 Constructor & Destructor Documentation . . . 271
10.40.2.1 DynamicRenderable . . . 271
10.40.2.2 ∼DynamicRenderable . . . 271
10.40.3 Member Function Documentation . . . 271
10.40.3.1 CreateVertexDeclaration . . . 271
10.40.3.2 FillHardwareBuffers . . . 271
10.40.3.3 getBoundingRadius . . . 272
10.40.3.4 GetOperationType . . . 272
10.40.3.5 getSquaredViewDepth . . . 272
10.40.3.6 Init . . . 272
10.40.3.7 PrepareHardwareBuffers . . . 272
10.40.3.8 SetOperationType . . . 273
10.40.4 Member Data Documentation . . . 273
10.40.4.1 indexBufferCapacity . . . 273
10.40.4.2 vertexBufferCapacity . . . 273
10.41 sdf::Element Class Reference . . . 273
10.41.1 Detailed Description . . . 276
10.41.2 Constructor & Destructor Documentation . . . 276
10.41.2.1 Element . . . 276
10.41.2.2 ∼Element . . . 276
10.41.3 Member Function Documentation . . . 276
10.41.3.1 AddAttribute . . . 276
10.41.3.2 AddElement . . . 276

10.41.3.3 AddElementDescription . . . 276
10.41.3.4 AddValue . . . 276
10.41.3.5 ClearElements . . . 276
10.41.3.6 Clone . . . 277
10.41.3.7 Copy . . . 277
10.41.3.8 GetAttribute . . . 277
10.41.3.9 GetAttribute . . . 277
10.41.3.10 GetAttributeCount . . . 277
10.41.3.11 GetAttributeSet . . . 277
10.41.3.12 GetCopyChildren . . . 277
10.41.3.13 GetDescription . . . 277
10.41.3.14 GetElement . . . 277
10.41.3.15 GetElement . . . 277
10.41.3.16 GetElementDescription . . . 277
10.41.3.17 GetElementDescription . . . 277
10.41.3.18 GetElementDescriptionCount . . . 278
10.41.3.19 GetElementImpl . . . 278
10.41.3.20 GetFirstElement . . . 278
10.41.3.21 GetInclude . . . 278
10.41.3.22 GetName . . . 278
10.41.3.23 GetNextElement . . . 278
10.41.3.24 GetParent . . . 278
10.41.3.25 GetRequired . . . 278
10.41.3.26 GetValue . . . 278
10.41.3.27 GetValueBool . . . 278
10.41.3.28 GetValueChar . . . 278
10.41.3.29 GetValueColor . . . 278
10.41.3.30 GetValueDouble . . . 278
10.41.3.31 GetValueFloat . . . 278
10.41.3.32 GetValueInt . . . 278
10.41.3.33 GetValuePose . . . 278
10.41.3.34 GetValueQuaternion . . . 278
10.41.3.35 GetValueString . . . 278
10.41.3.36 GetValueTime . . . 278
10.41.3.37 GetValueUInt . . . 278
10.41.3.38 GetValueVector2d . . . 278
10.41.3.39 GetValueVector3 . . . 278

10.41.3.40HasAttribute . . . 279
10.41.3.41HasElement . . . 279
10.41.3.42HasElementDescription . . . 279
10.41.3.43InsertElement . . . 279
10.41.3.44PrintDescription . . . 279
10.41.3.45PrintDocLeftPane . . . 279
10.41.3.46PrintDocRightPane . . . 279
10.41.3.47PrintValues . . . 279
10.41.3.48PrintWiki . . . 279
10.41.3.49RemoveChild . . . 279
10.41.3.50RemoveFromParent . . . 280
10.41.3.51Reset . . . 280
10.41.3.52Set . . . 280
10.41.3.53Set . . . 280
10.41.3.54Set . . . 280
10.41.3.55Set . . . 280
10.41.3.56Set . . . 280
10.41.3.57Set . . . 280
10.41.3.58Set . . . 280
10.41.3.59Set . . . 280
10.41.3.60Set . . . 280
10.41.3.61Set . . . 280
10.41.3.62Set . . . 280
10.41.3.63Set . . . 280
10.41.3.64Set . . . 280
10.41.3.65Set . . . 280
10.41.3.66Set . . . 280
10.41.3.67SetCopyChildren . . . 280
10.41.3.68SetDescription . . . 280
10.41.3.69SetInclude . . . 280
10.41.3.70SetName . . . 280
10.41.3.71SetParent . . . 280
10.41.3.72SetRequired . . . 280
10.41.3.73ToString . . . 281
10.41.3.74Update . . . 281
10.42gazebo::physics::Entity Class Reference . . . 281
10.42.1 Detailed Description . . . 284

10.42.2 Constructor & Destructor Documentation . . . 284
10.42.2.1 Entity . . . 284
10.42.2.2 ∼Entity . . . 284
10.42.3 Member Function Documentation . . . 284
10.42.3.1 Fini . . . 284
10.42.3.2 GetBoundingBox . . . 284
10.42.3.3 GetChildCollision . . . 284
10.42.3.4 GetChildLink . . . 285
10.42.3.5 GetCollisionBoundingBox . . . 285
10.42.3.6 GetDirtyPose . . . 285
10.42.3.7 GetInitialRelativePose . . . 285
10.42.3.8 GetNearestEntityBelow . . . 286
10.42.3.9 GetParentModel . . . 286
10.42.3.10 GetRelativeAngularAccel . . . 286
10.42.3.11 GetRelativeAngularVel . . . 286
10.42.3.12 GetRelativeLinearAccel . . . 286
10.42.3.13 GetRelativeLinearVel . . . 287
10.42.3.14 GetRelativePose . . . 287
10.42.3.15 GetWorldAngularAccel . . . 287
10.42.3.16 GetWorldAngularVel . . . 287
10.42.3.17 GetWorldLinearAccel . . . 287
10.42.3.18 GetWorldLinearVel . . . 288
10.42.3.19 GetWorldPose . . . 288
10.42.3.20 IsCanonicalLink . . . 288
10.42.3.21 IsStatic . . . 288
10.42.3.22 Load . . . 288
10.42.3.23 OnPoseChange . . . 289
10.42.3.24 PlaceOnEntity . . . 289
10.42.3.25 PlaceOnNearestEntityBelow . . . 289
10.42.3.26 Reset . . . 289
10.42.3.27 SetAnimation . . . 289
10.42.3.28 SetAnimation . . . 289
10.42.3.29 SetCanonicalLink . . . 289
10.42.3.30 SetInitialRelativePose . . . 290
10.42.3.31 SetName . . . 290
10.42.3.32 SetRelativePose . . . 290
10.42.3.33 SetStatic . . . 290

10.42.3.34 SetWorldPose . . . 290
10.42.3.35 SetWorldTwist . . . 291
10.42.3.36 StopAnimation . . . 291
10.42.3.37 UpdateParameters . . . 291
10.42.4 Member Data Documentation . . . 291
10.42.4.1 animation . . . 291
10.42.4.2 animationConnection . . . 291
10.42.4.3 animationStartPose . . . 291
10.42.4.4 connections . . . 292
10.42.4.5 dirtyPose . . . 292
10.42.4.6 node . . . 292
10.42.4.7 parentEntity . . . 292
10.42.4.8 poseMsg . . . 292
10.42.4.9 prevAnimationTime . . . 292
10.42.4.10 requestPub . . . 292
10.42.4.11 visPub . . . 292
10.42.4.12 visualMsg . . . 292
10.43 gazebo::event::Event Class Reference . . . 292
10.43.1 Detailed Description . . . 294
10.43.2 Constructor & Destructor Documentation . . . 294
10.43.2.1 ∼Event . . . 294
10.43.3 Member Function Documentation . . . 294
10.43.3.1 Disconnect . . . 294
10.43.3.2 Disconnect . . . 294
10.44 gazebo::rendering::Events Class Reference . . . 295
10.44.1 Detailed Description . . . 295
10.44.2 Member Function Documentation . . . 295
10.44.2.1 ConnectCreateScene . . . 295
10.44.2.2 ConnectRemoveScene . . . 296
10.44.2.3 DisconnectCreateScene . . . 296
10.44.2.4 DisconnectRemoveScene . . . 296
10.44.3 Member Data Documentation . . . 296
10.44.3.1 createScene . . . 296
10.44.3.2 removeScene . . . 297
10.45 gazebo::event::Events Class Reference . . . 297
10.45.1 Detailed Description . . . 299
10.45.2 Member Function Documentation . . . 299

10.45.2.1 ConnectAddEntity . . . 299
10.45.2.2 ConnectCreateEntity . . . 300
10.45.2.3 ConnectDeleteEntity . . . 300
10.45.2.4 ConnectDiagTimerStart . . . 300
10.45.2.5 ConnectDiagTimerStop . . . 301
10.45.2.6 ConnectPause . . . 301
10.45.2.7 ConnectPostRender . . . 301
10.45.2.8 ConnectPreRender . . . 301
10.45.2.9 ConnectRender . . . 302
10.45.2.10 ConnectSetSelectedEntity . . . 302
10.45.2.11 ConnectStep . . . 302
10.45.2.12 ConnectStop . . . 303
10.45.2.13 ConnectWorldCreated . . . 303
10.45.2.14 ConnectWorldUpdateBegin . . . 303
10.45.2.15 ConnectWorldUpdateEnd . . . 303
10.45.2.16 ConnectWorldUpdateStart . . . 304
10.45.2.17 DisconnectAddEntity . . . 304
10.45.2.18 DisconnectCreateEntity . . . 304
10.45.2.19 DisconnectDeleteEntity . . . 304
10.45.2.20 DisconnectDiagTimerStart . . . 305
10.45.2.21 DisconnectDiagTimerStop . . . 305
10.45.2.22 DisconnectPause . . . 305
10.45.2.23 DisconnectPostRender . . . 305
10.45.2.24 DisconnectPreRender . . . 306
10.45.2.25 DisconnectRender . . . 306
10.45.2.26 DisconnectSetSelectedEntity . . . 306
10.45.2.27 DisconnectStep . . . 306
10.45.2.28 DisconnectStop . . . 306
10.45.2.29 DisconnectWorldCreated . . . 307
10.45.2.30 DisconnectWorldUpdateBegin . . . 307
10.45.2.31 DisconnectWorldUpdateEnd . . . 307
10.45.2.32 DisconnectWorldUpdateStart . . . 307
10.45.3 Member Data Documentation . . . 307
10.45.3.1 addEntity . . . 307
10.45.3.2 deleteEntity . . . 307
10.45.3.3 diagTimerStart . . . 307
10.45.3.4 diagTimerStop . . . 308

10.45.3.5 entityCreated . . . 308
10.45.3.6 pause . . . 308
10.45.3.7 postRender . . . 308
10.45.3.8 preRender . . . 308
10.45.3.9 render . . . 308
10.45.3.10 setSelectedEntity . . . 308
10.45.3.11 step . . . 308
10.45.3.12 stop . . . 309
10.45.3.13 worldCreated . . . 309
10.45.3.14 worldUpdateBegin . . . 309
10.45.3.15 worldUpdateEnd . . . 309
10.45.3.16 worldUpdateStart . . . 309
10.46 gazebo::event::EventT< T > Class Template Reference . . . 309
10.46.1 Detailed Description . . . 312
10.46.2 Member Function Documentation . . . 312
10.46.2.1 operator() . . . 312
10.46.2.2 operator() . . . 312
10.46.2.3 operator() . . . 312
10.46.2.4 operator() . . . 312
10.46.2.5 operator() . . . 313
10.46.2.6 operator() . . . 313
10.46.2.7 operator() . . . 313
10.46.2.8 operator() . . . 313
10.46.2.9 operator() . . . 314
10.46.2.10 operator() . . . 314
10.46.2.11 operator() . . . 315
10.46.2.12 Signal . . . 315
10.46.2.13 Signal . . . 315
10.46.2.14 Signal . . . 315
10.46.2.15 Signal . . . 315
10.46.2.16 Signal . . . 316
10.46.2.17 Signal . . . 316
10.46.2.18 Signal . . . 316
10.46.2.19 Signal . . . 317
10.46.2.20 Signal . . . 317
10.46.2.21 Signal . . . 317
10.46.2.22 Signal . . . 318

10.47gazebo::common::Exception Class Reference . . . 318
10.47.1 Detailed Description . . . 319
10.47.2 Constructor & Destructor Documentation . . . 319
10.47.2.1 Exception . . . 319
10.47.2.2 Exception . . . 319
10.47.2.3 ∼Exception . . . 319
10.47.3 Member Function Documentation . . . 319
10.47.3.1 GetErrorFile . . . 320
10.47.3.2 GetErrorStr . . . 320
10.47.3.3 Print . . . 320
10.47.4 Friends And Related Function Documentation . . . 320
10.47.4.1 operator<< . . . 320
10.48gazebo::rendering::FPSViewController Class Reference . . . 320
10.48.1 Detailed Description . . . 321
10.48.2 Constructor & Destructor Documentation . . . 322
10.48.2.1 FPSViewController . . . 322
10.48.2.2 ∼FPSViewController . . . 322
10.48.3 Member Function Documentation . . . 322
10.48.3.1 GetTypeString . . . 322
10.48.3.2 HandleKeyPressEvent . . . 322
10.48.3.3 HandleKeyReleaseEvent . . . 322
10.48.3.4 HandleMouseEvent . . . 322
10.48.3.5 Init . . . 323
10.48.3.6 Update . . . 323
10.49urdf2gazebo::GazeboExtension Class Reference . . . 323
10.50google::protobuf::compiler::cpp::GazeboGenerator Class Reference . . . 323
10.50.1 Detailed Description . . . 324
10.50.2 Constructor & Destructor Documentation . . . 324
10.50.2.1 GazeboGenerator . . . 324
10.50.2.2 ∼GazeboGenerator . . . 324
10.50.3 Member Function Documentation . . . 324
10.50.3.1 Generate . . . 324
10.51gazebo::rendering::GpuLaser Class Reference . . . 324
10.51.1 Detailed Description . . . 326
10.51.2 Constructor & Destructor Documentation . . . 326
10.51.2.1 GpuLaser . . . 326
10.51.2.2 ∼GpuLaser . . . 326

10.51.3 Member Function Documentation . . . 326
10.51.3.1 ConnectNewLaserFrame . . . 326
10.51.3.2 CreateLaserTexture . . . 326
10.51.3.3 DisconnectNewLaserFrame . . . 327
10.51.3.4 Fini . . . 327
10.51.3.5 GetLaserData . . . 327
10.51.3.6 Init . . . 327
10.51.3.7 Load . . . 327
10.51.3.8 Load . . . 327
10.51.3.9 notifyRenderSingleObject . . . 327
10.51.3.10 PostRender . . . 327
10.51.3.11 SetParentSensor . . . 328
10.51.3.12 SetRangeCount . . . 328
10.52 gazebo::sensors::GpuRaySensor Class Reference . . . 328
10.52.1 Constructor & Destructor Documentation . . . 331
10.52.1.1 GpuRaySensor . . . 331
10.52.1.2 ∼GpuRaySensor . . . 331
10.52.2 Member Function Documentation . . . 332
10.52.2.1 ConnectNewLaserFrame . . . 332
10.52.2.2 DisconnectNewLaserFrame . . . 332
10.52.2.3 Fini . . . 332
10.52.2.4 GetAngleMax . . . 332
10.52.2.5 GetAngleMin . . . 332
10.52.2.6 GetAngleResolution . . . 332
10.52.2.7 GetCameraCount . . . 332
10.52.2.8 GetCosHorzFOV . . . 333
10.52.2.9 GetCosVertFOV . . . 333
10.52.2.10 GetFiducial . . . 333
10.52.2.11 GetHorzFOV . . . 333
10.52.2.12 GetHorzHalfAngle . . . 333
10.52.2.13 GetLaserCamera . . . 334
10.52.2.14 GetRange . . . 334
10.52.2.15 GetRangeCount . . . 334
10.52.2.16 GetRangeCountRatio . . . 334
10.52.2.17 GetRangeMax . . . 334
10.52.2.18 GetRangeMin . . . 335
10.52.2.19 GetRangeResolution . . . 335

10.52.2.20GetRanges . . . 335
10.52.2.21GetRayCount . . . 335
10.52.2.22GetRayCountRatio . . . 335
10.52.2.23GetRetro . . . 335
10.52.2.24GetVertFOV . . . 336
10.52.2.25GetVertHalfAngle . . . 336
10.52.2.26GetVerticalAngleMax . . . 336
10.52.2.27GetVerticalAngleMin . . . 336
10.52.2.28GetVerticalRangeCount . . . 336
10.52.2.29GetVerticalRayCount . . . 337
10.52.2.30Init . . . 337
10.52.2.31IsHorizontal . . . 337
10.52.2.32Load . . . 337
10.52.2.33Load . . . 337
10.52.2.34SetAngleMax . . . 337
10.52.2.35SetAngleMin . . . 338
10.52.2.36SetVerticalAngleMax . . . 338
10.52.2.37SetVerticalAngleMin . . . 338
10.52.2.38UpdateImpl . . . 338
10.52.3 Member Data Documentation . . . 338
10.52.3.1 cameraCount . . . 338
10.52.3.2 cameraElem . . . 338
10.52.3.3 chfov . . . 338
10.52.3.4 cvfov . . . 339
10.52.3.5 far . . . 339
10.52.3.6 hfov . . . 339
10.52.3.7 horzElem . . . 339
10.52.3.8 horzHalfAngle . . . 339
10.52.3.9 horzRangeCount . . . 339
10.52.3.10horzRayCount . . . 339
10.52.3.11isHorizontal . . . 339
10.52.3.12near . . . 339
10.52.3.13rangeCountRatio . . . 339
10.52.3.14rangeElem . . . 339
10.52.3.15rayCountRatio . . . 340
10.52.3.16scanElem . . . 340
10.52.3.17vertElem . . . 340

10.52.3.18 vertHalfAngle . . . 340
10.52.3.19 vertRangeCount . . . 340
10.52.3.20 vertRayCount . . . 340
10.52.3.21 vfov . . . 340
10.53 gazebo::rendering::Grid Class Reference . . . 340
10.53.1 Detailed Description . . . 341
10.53.2 Constructor & Destructor Documentation . . . 341
10.53.2.1 Grid . . . 341
10.53.2.2 ∼Grid . . . 342
10.53.3 Member Function Documentation . . . 342
10.53.3.1 Enable . . . 342
10.53.3.2 GetCellCount . . . 342
10.53.3.3 GetCellLength . . . 342
10.53.3.4 GetColor . . . 342
10.53.3.5 GetHeight . . . 342
10.53.3.6 GetLineWidth . . . 342
10.53.3.7 GetSceneNode . . . 343
10.53.3.8 Init . . . 343
10.53.3.9 SetCellCount . . . 343
10.53.3.10 SetCellLength . . . 343
10.53.3.11 SetColor . . . 343
10.53.3.12 SetHeight . . . 343
10.53.3.13 SetLineWidth . . . 343
10.53.3.14 SetUserData . . . 344
10.54 gazebo::physics::Gripper Class Reference . . . 344
10.54.1 Detailed Description . . . 344
10.54.2 Constructor & Destructor Documentation . . . 344
10.54.2.1 Gripper . . . 344
10.54.2.2 ∼Gripper . . . 345
10.54.3 Member Function Documentation . . . 345
10.54.3.1 Init . . . 345
10.54.3.2 Load . . . 345
10.55 gazebo::rendering::GUIOverlay Class Reference . . . 345
10.55.1 Detailed Description . . . 346
10.55.2 Constructor & Destructor Documentation . . . 346
10.55.2.1 GUIOverlay . . . 346
10.55.2.2 ∼GUIOverlay . . . 346

10.55.3 Member Function Documentation . . . 346
10.55.3.1 AttachCameraToImage . . . 346
10.55.3.2 AttachCameraToImage . . . 347
10.55.3.3 ButtonCallback . . . 347
10.55.3.4 CreateWindow . . . 347
10.55.3.5 HandleKeyPressEvent . . . 347
10.55.3.6 HandleKeyReleaseEvent . . . 348
10.55.3.7 HandleMouseEvent . . . 348
10.55.3.8 Hide . . . 348
10.55.3.9 Init . . . 348
10.55.3.10 IsInitialized . . . 348
10.55.3.11 LoadLayout . . . 349
10.55.3.12 Resize . . . 349
10.55.3.13 Show . . . 349
10.55.3.14 Update . . . 349
10.56 gazebo::rendering::GzTerrainMatGen Class Reference . . . 349
10.56.1 Constructor & Destructor Documentation . . . 350
10.56.1.1 GzTerrainMatGen . . . 350
10.56.1.2 ∼GzTerrainMatGen . . . 350
10.57 gazebo::rendering::Heightmap Class Reference . . . 350
10.57.1 Detailed Description . . . 350
10.57.2 Constructor & Destructor Documentation . . . 351
10.57.2.1 Heightmap . . . 351
10.57.2.2 ∼Heightmap . . . 351
10.57.3 Member Function Documentation . . . 351
10.57.3.1 GetHeight . . . 351
10.57.3.2 GetOgreTerrain . . . 351
10.57.3.3 Load . . . 351
10.57.3.4 LoadFromMsg . . . 351
10.58 gazebo::physics::HeightmapShape Class Reference . . . 352
10.58.1 Detailed Description . . . 353
10.58.2 Constructor & Destructor Documentation . . . 353
10.58.2.1 HeightmapShape . . . 353
10.58.2.2 ∼HeightmapShape . . . 354
10.58.3 Member Function Documentation . . . 354
10.58.3.1 FillMsg . . . 354
10.58.3.2 GetHeight . . . 354

10.58.3.3 GetMaxHeight . . . 354
10.58.3.4 GetMinHeight . . . 354
10.58.3.5 GetPos . . . 354
10.58.3.6 GetSize . . . 355
10.58.3.7 GetSubSampling . . . 355
10.58.3.8 GetURI . . . 355
10.58.3.9 GetVertexCount . . . 355
10.58.3.10 Init . . . 355
10.58.3.11 Load . . . 355
10.58.3.12 ProcessMsg . . . 356
10.58.4 Member Data Documentation . . . 356
10.58.4.1 heights . . . 356
10.58.4.2 img . . . 356
10.58.4.3 scale . . . 356
10.58.4.4 subSampling . . . 356
10.58.4.5 vertSize . . . 356
10.59 gazebo::physics::Hinge2Joint< T > Class Template Reference . . . 356
10.59.1 Detailed Description . . . 357
10.59.2 Constructor & Destructor Documentation . . . 357
10.59.2.1 Hinge2Joint . . . 357
10.59.2.2 ∼Hinge2Joint . . . 358
10.59.3 Member Function Documentation . . . 358
10.59.3.1 GetAngleCount . . . 358
10.59.3.2 Load . . . 358
10.60 gazebo::physics::HingeJoint< T > Class Template Reference . . . 358
10.60.1 Detailed Description . . . 359
10.60.2 Constructor & Destructor Documentation . . . 359
10.60.2.1 HingeJoint . . . 359
10.60.2.2 ∼HingeJoint . . . 359
10.60.3 Member Function Documentation . . . 359
10.60.3.1 GetAngleCount . . . 359
10.60.3.2 Init . . . 359
10.60.3.3 Load . . . 360
10.61 gazebo::common::Image Class Reference . . . 360
10.61.1 Detailed Description . . . 361
10.61.2 Member Enumeration Documentation . . . 361
10.61.2.1 PixelFormat . . . 361

10.61.3 Constructor & Destructor Documentation . . . 362
10.61.3.1 Image . . . 362
10.61.3.2 ∼Image . . . 362
10.61.4 Member Function Documentation . . . 362
10.61.4.1 ConvertPixelFormat . . . 362
10.61.4.2 GetAvgColor . . . 362
10.61.4.3 GetBPP . . . 363
10.61.4.4 GetData . . . 363
10.61.4.5 GetFilename . . . 363
10.61.4.6 GetHeight . . . 363
10.61.4.7 GetMaxColor . . . 363
10.61.4.8 GetPitch . . . 363
10.61.4.9 GetPixel . . . 364
10.61.4.10 GetPixelFormat . . . 364
10.61.4.11 GetRGBData . . . 364
10.61.4.12 GetWidth . . . 364
10.61.4.13 Load . . . 364
10.61.4.14 Rescale . . . 364
10.61.4.15 SavePNG . . . 365
10.61.4.16 SetFromData . . . 365
10.61.4.17 Valid . . . 365
10.62 gazebo::sensors::ImuSensor Class Reference . . . 365
10.62.1 Detailed Description . . . 367
10.62.2 Constructor & Destructor Documentation . . . 367
10.62.2.1 ImuSensor . . . 367
10.62.2.2 ∼ImuSensor . . . 367
10.62.3 Member Function Documentation . . . 367
10.62.3.1 Fini . . . 367
10.62.3.2 GetAngularVelocity . . . 367
10.62.3.3 GetImuMessage . . . 367
10.62.3.4 GetLinearAcceleration . . . 368
10.62.3.5 GetOrientation . . . 368
10.62.3.6 Init . . . 368
10.62.3.7 Load . . . 368
10.62.3.8 Load . . . 368
10.62.3.9 SetReferencePose . . . 368
10.62.3.10 UpdateImpl . . . 369

10.63gazebo::physics::Inertial Class Reference . . . 369
10.63.1 Detailed Description . . . 371
10.63.2 Constructor & Destructor Documentation . . . 371
10.63.2.1 Inertial . . . 371
10.63.2.2 Inertial . . . 371
10.63.2.3 Inertial . . . 371
10.63.2.4 ∼Inertial . . . 371
10.63.3 Member Function Documentation . . . 372
10.63.3.1 GetCoG . . . 372
10.63.3.2 GetInertial . . . 372
10.63.3.3 GetIXX . . . 372
10.63.3.4 GetIXY . . . 372
10.63.3.5 GetIXZ . . . 372
10.63.3.6 GetIYY . . . 373
10.63.3.7 GetIYZ . . . 373
10.63.3.8 GetIZZ . . . 373
10.63.3.9 GetMass . . . 373
10.63.3.10GetMOI . . . 373
10.63.3.11GetMOI . . . 373
10.63.3.12GetPose . . . 374
10.63.3.13GetPrincipalMoments . . . 374
10.63.3.14GetProductsofInertia . . . 374
10.63.3.15Load . . . 374
10.63.3.16operator+ . . . 374
10.63.3.17operator+= . . . 375
10.63.3.18operator= . . . 375
10.63.3.19ProcessMsg . . . 375
10.63.3.20Reset . . . 375
10.63.3.21Rotate . . . 375
10.63.3.22SetCoG . . . 376
10.63.3.23SetCoG . . . 376
10.63.3.24SetCoG . . . 376
10.63.3.25SetCoG . . . 376
10.63.3.26SetInertiaMatrix . . . 376
10.63.3.27SetIXX . . . 377
10.63.3.28SetIXY . . . 377
10.63.3.29SetIXZ . . . 377

10.63.3.30 SetIYY . . . 377
10.63.3.31 SetIYZ . . . 377
10.63.3.32 SetIZZ . . . 377
10.63.3.33 SetMass . . . 378
10.63.3.34 SetMOI . . . 378
10.63.3.35 UpdateParameters . . . 378
10.63.4 Friends And Related Function Documentation . . . 378
10.63.4.1 operator<< . . . 378
10.64 gazebo::common::InternalError Class Reference . . . 378
10.64.1 Detailed Description . . . 379
10.64.2 Constructor & Destructor Documentation . . . 379
10.64.2.1 InternalError . . . 379
10.64.2.2 InternalError . . . 379
10.64.2.3 ∼InternalError . . . 380
10.65 gazebo::transport::IOManager Class Reference . . . 380
10.65.1 Detailed Description . . . 380
10.65.2 Constructor & Destructor Documentation . . . 380
10.65.2.1 IOManager . . . 380
10.65.2.2 ∼IOManager . . . 381
10.65.3 Member Function Documentation . . . 381
10.65.3.1 DecCount . . . 381
10.65.3.2 GetCount . . . 381
10.65.3.3 GetIO . . . 381
10.65.3.4 IncCount . . . 381
10.65.3.5 Stop . . . 381
10.66 gazebo::physics::Joint Class Reference . . . 381
10.66.1 Detailed Description . . . 385
10.66.2 Member Enumeration Documentation . . . 385
10.66.2.1 Attribute . . . 385
10.66.3 Constructor & Destructor Documentation . . . 386
10.66.3.1 Joint . . . 386
10.66.3.2 ∼Joint . . . 386
10.66.4 Member Function Documentation . . . 386
10.66.4.1 ApplyDamping . . . 386
10.66.4.2 AreConnected . . . 386
10.66.4.3 Attach . . . 386
10.66.4.4 ConnectJointUpdate . . . 387

10.66.4.5 Detach . . . 387
10.66.4.6 DisconnectJointUpdate . . . 387
10.66.4.7 FillMsg . . . 387
10.66.4.8 GetAnchor . . . 387
10.66.4.9 GetAngle . . . 388
10.66.4.10 GetAngleCount . . . 388
10.66.4.11 GetAngleImpl . . . 388
10.66.4.12 GetAttribute . . . 388
10.66.4.13 GetChild . . . 389
10.66.4.14 GetEffortLimit . . . 389
10.66.4.15 GetForce . . . 389
10.66.4.16 GetForce . . . 389
10.66.4.17 GetForceTorque . . . 390
10.66.4.18 GetForceTorque . . . 390
10.66.4.19 GetGlobalAxis . . . 390
10.66.4.20 GetHighStop . . . 390
10.66.4.21 GetInertiaRatio . . . 391
10.66.4.22 GetJointLink . . . 391
10.66.4.23 GetLinkForce . . . 391
10.66.4.24 GetLinkTorque . . . 391
10.66.4.25 GetLocalAxis . . . 392
10.66.4.26 GetLowerLimit . . . 392
10.66.4.27 GetLowStop . . . 392
10.66.4.28 GetMaxForce . . . 393
10.66.4.29 GetParent . . . 393
10.66.4.30 GetUpperLimit . . . 393
10.66.4.31 GetVelocity . . . 393
10.66.4.32 GetVelocityLimit . . . 394
10.66.4.33 Init . . . 394
10.66.4.34 Load . . . 394
10.66.4.35 Load . . . 394
10.66.4.36 Load . . . 394
10.66.4.37 Reset . . . 395
10.66.4.38 SetAnchor . . . 395
10.66.4.39 SetAngle . . . 395
10.66.4.40 SetAttribute . . . 395
10.66.4.41 SetAxis . . . 395

10.66.4.42 SetDamping . . . 396
10.66.4.43 SetForce . . . 396
10.66.4.44 SetHighStop . . . 396
10.66.4.45 SetLowStop . . . 396
10.66.4.46 SetMaxForce . . . 396
10.66.4.47 SetModel . . . 397
10.66.4.48 SetState . . . 397
10.66.4.49 SetVelocity . . . 397
10.66.4.50 Update . . . 397
10.66.4.51 UpdateParameters . . . 397
10.66.5 Member Data Documentation . . . 397
10.66.5.1 anchorLink . . . 397
10.66.5.2 anchorPos . . . 398
10.66.5.3 anchorPose . . . 398
10.66.5.4 applyDamping . . . 398
10.66.5.5 childLink . . . 398
10.66.5.6 dampingCoefficient . . . 398
10.66.5.7 effortLimit . . . 398
10.66.5.8 forceApplied . . . 398
10.66.5.9 inertiaRatio . . . 398
10.66.5.10 lowerLimit . . . 398
10.66.5.11 model . . . 398
10.66.5.12 parentLink . . . 399
10.66.5.13 upperLimit . . . 399
10.66.5.14 useCFMDamping . . . 399
10.66.5.15 velocityLimit . . . 399
10.67 gazebo::physics::JointController Class Reference . . . 399
10.67.1 Detailed Description . . . 399
10.67.2 Constructor & Destructor Documentation . . . 400
10.67.2.1 JointController . . . 400
10.67.3 Member Function Documentation . . . 400
10.67.3.1 AddJoint . . . 400
10.67.3.2 Reset . . . 400
10.67.3.3 SetJointPosition . . . 400
10.67.3.4 SetJointPosition . . . 400
10.67.3.5 SetJointPositions . . . 400
10.67.3.6 Update . . . 401

10.68gazebo::physics::JointState Class Reference . . . 401
10.68.1 Detailed Description . . . 402
10.68.2 Constructor & Destructor Documentation . . . 402
10.68.2.1 JointState . . . 402
10.68.2.2 JointState . . . 402
10.68.2.3 JointState . . . 402
10.68.2.4 ∼JointState . . . 403
10.68.3 Member Function Documentation . . . 403
10.68.3.1 FillSDF . . . 403
10.68.3.2 GetAngle . . . 403
10.68.3.3 GetAngleCount . . . 403
10.68.3.4 GetAngles . . . 403
10.68.3.5 IsZero . . . 404
10.68.3.6 Load . . . 404
10.68.3.7 operator+ . . . 404
10.68.3.8 operator- . . . 404
10.68.3.9 operator= . . . 404
10.68.4 Friends And Related Function Documentation . . . 405
10.68.4.1 operator<< . . . 405
10.69gazebo::rendering::JointVisual Class Reference . . . 405
10.69.1 Detailed Description . . . 406
10.69.2 Constructor & Destructor Documentation . . . 406
10.69.2.1 JointVisual . . . 406
10.69.2.2 ∼JointVisual . . . 407
10.69.3 Member Function Documentation . . . 407
10.69.3.1 Load . . . 407
10.70gazebo::physics::JointWrench Class Reference . . . 407
10.70.1 Detailed Description . . . 407
10.70.2 Member Function Documentation . . . 408
10.70.2.1 operator+ . . . 408
10.70.2.2 operator- . . . 408
10.70.2.3 operator= . . . 408
10.70.3 Member Data Documentation . . . 408
10.70.3.1 body1Force . . . 408
10.70.3.2 body1Torque . . . 409
10.70.3.3 body2Force . . . 409
10.70.3.4 body2Torque . . . 409

10.71 gazebo::common::KeyFrame Class Reference . . . 409
10.71.1 Detailed Description . . . 410
10.71.2 Constructor & Destructor Documentation . . . 410
10.71.2.1 KeyFrame . . . 410
10.71.2.2 ∼KeyFrame . . . 410
10.71.3 Member Function Documentation . . . 410
10.71.3.1 GetTime . . . 410
10.71.4 Member Data Documentation . . . 410
10.71.4.1 time . . . 410
10.72 gazebo::rendering::LaserVisual Class Reference . . . 410
10.72.1 Detailed Description . . . 411
10.72.2 Constructor & Destructor Documentation . . . 411
10.72.2.1 LaserVisual . . . 411
10.72.2.2 ∼LaserVisual . . . 412
10.72.3 Member Function Documentation . . . 412
10.72.3.1 SetEmissive . . . 412
10.73 gazebo::rendering::Light Class Reference . . . 412
10.73.1 Detailed Description . . . 413
10.73.2 Constructor & Destructor Documentation . . . 414
10.73.2.1 Light . . . 414
10.73.2.2 ∼Light . . . 414
10.73.3 Member Function Documentation . . . 414
10.73.3.1 FillMsg . . . 414
10.73.3.2 GetDiffuseColor . . . 414
10.73.3.3 GetDirection . . . 414
10.73.3.4 GetName . . . 414
10.73.3.5 GetPosition . . . 415
10.73.3.6 GetSpecularColor . . . 415
10.73.3.7 GetType . . . 415
10.73.3.8 Load . . . 415
10.73.3.9 Load . . . 415
10.73.3.10 LoadFromMsg . . . 415
10.73.3.11 OnPoseChange . . . 415
10.73.3.12 SetAttenuation . . . 416
10.73.3.13 SetCastShadows . . . 416
10.73.3.14 SetDiffuseColor . . . 416
10.73.3.15 SetDirection . . . 416

10.73.3.16SetLightType . . . 416
10.73.3.17SetName . . . 416
10.73.3.18SetPosition . . . 417
10.73.3.19SetRange . . . 417
10.73.3.20SetSelected . . . 417
10.73.3.21SetSpecularColor . . . 417
10.73.3.22SetSpotFalloff . . . 417
10.73.3.23SetSpotInnerAngle . . . 417
10.73.3.24SetSpotOuterAngle . . . 418
10.73.3.25ShowVisual . . . 418
10.73.3.26ToggleShowVisual . . . 418
10.73.3.27UpdateFromMsg . . . 418
10.74gazebo::physics::Link Class Reference . . . 418
10.74.1 Detailed Description . . . 423
10.74.2 Constructor & Destructor Documentation . . . 423
10.74.2.1 Link . . . 423
10.74.2.2 ∼Link . . . 423
10.74.3 Member Function Documentation . . . 423
10.74.3.1 AddChildJoint . . . 423
10.74.3.2 AddForce . . . 424
10.74.3.3 AddForceAtRelativePosition . . . 424
10.74.3.4 AddForceAtWorldPosition . . . 424
10.74.3.5 AddParentJoint . . . 424
10.74.3.6 AddRelativeForce . . . 424
10.74.3.7 AddRelativeTorque . . . 424
10.74.3.8 AddTorque . . . 425
10.74.3.9 AttachStaticModel . . . 425
10.74.3.10ConnectEnabled . . . 425
10.74.3.11DetachAllStaticModels . . . 425
10.74.3.12DetachStaticModel . . . 425
10.74.3.13DisconnectEnabled . . . 426
10.74.3.14FillMsg . . . 426
10.74.3.15Fini . . . 426
10.74.3.16GetAngularDamping . . . 426
10.74.3.17GetBoundingBox . . . 426
10.74.3.18GetChildJointsLinks . . . 426
10.74.3.19GetCollision . . . 427

10.74.3.20GetCollision . . . 427
10.74.3.21GetCollisionById . . . 427
10.74.3.22GetCollisions . . . 427
10.74.3.23GetEnabled . . . 427
10.74.3.24GetGravityMode . . . 428
10.74.3.25GetInertial . . . 428
10.74.3.26GetKinematic . . . 428
10.74.3.27GetLinearDamping . . . 428
10.74.3.28GetModel . . . 428
10.74.3.29GetParentJointsLinks . . . 429
10.74.3.30GetRelativeAngularAccel . . . 429
10.74.3.31GetRelativeAngularVel . . . 429
10.74.3.32GetRelativeForce . . . 429
10.74.3.33GetRelativeLinearAccel . . . 429
10.74.3.34GetRelativeLinearVel . . . 429
10.74.3.35GetRelativeTorque . . . 430
10.74.3.36GetSelfCollide . . . 430
10.74.3.37GetSensorCount . . . 430
10.74.3.38GetSensorName . . . 430
10.74.3.39GetWorldAngularAccel . . . 430
10.74.3.40GetWorldCoGLinearVel . . . 431
10.74.3.41GetWorldCoGPose . . . 431
10.74.3.42GetWorldForce . . . 431
10.74.3.43GetWorldLinearAccel . . . 431
10.74.3.44GetWorldLinearVel . . . 431
10.74.3.45GetWorldLinearVel . . . 432
10.74.3.46GetWorldTorque . . . 432
10.74.3.47Init . . . 432
10.74.3.48Load . . . 432
10.74.3.49OnPoseChange . . . 432
10.74.3.50ProcessMsg . . . 433
10.74.3.51RemoveChildJoint . . . 433
10.74.3.52RemoveChildJoint . . . 433
10.74.3.53RemoveParentJoint . . . 433
10.74.3.54RemoveParentJoint . . . 433
10.74.3.55Reset . . . 433
10.74.3.56ResetPhysicsStates . . . 434

10.74.3.57SetAngularAccel . . . 434
10.74.3.58SetAngularDamping . . . 434
10.74.3.59SetAngularVel . . . 434
10.74.3.60SetAutoDisable . . . 434
10.74.3.61SetCollideMode . . . 434
10.74.3.62SetEnabled . . . 435
10.74.3.63SetForce . . . 435
10.74.3.64SetGravityMode . . . 435
10.74.3.65SetInertial . . . 435
10.74.3.66SetKinematic . . . 435
10.74.3.67SetLaserRetro . . . 435
10.74.3.68SetLinearAccel . . . 436
10.74.3.69SetLinearDamping . . . 436
10.74.3.70SetLinearVel . . . 436
10.74.3.71SetSelected . . . 436
10.74.3.72SetSelfCollide . . . 436
10.74.3.73SetState . . . 436
10.74.3.74SetTorque . . . 437
10.74.3.75Update . . . 437
10.74.3.76UpdateMass . . . 437
10.74.3.77UpdateParameters . . . 437
10.74.3.78UpdateSurface . . . 437
10.74.4 Member Data Documentation . . . 437
10.74.4.1 angularAccel . . . 437
10.74.4.2 attachedModelsOffset . . . 437
10.74.4.3 cgVisuals . . . 437
10.74.4.4 inertial . . . 438
10.74.4.5 linearAccel . . . 438
10.74.4.6 visuals . . . 438
10.75gazebo::physics::LinkState Class Reference . . . 438
10.75.1 Detailed Description . . . 439
10.75.2 Constructor & Destructor Documentation . . . 440
10.75.2.1 LinkState . . . 440
10.75.2.2 LinkState . . . 440
10.75.2.3 LinkState . . . 440
10.75.2.4 ∼LinkState . . . 440
10.75.3 Member Function Documentation . . . 440

10.75.3.1 FillSDF . . . 440
10.75.3.2 GetAcceleration . . . 440
10.75.3.3 GetCollisionState . . . 441
10.75.3.4 GetCollisionState . . . 441
10.75.3.5 GetCollisionStateCount . . . 441
10.75.3.6 GetCollisionStates . . . 441
10.75.3.7 GetPose . . . 442
10.75.3.8 GetVelocity . . . 442
10.75.3.9 GetWrench . . . 442
10.75.3.10 IsZero . . . 442
10.75.3.11 Load . . . 442
10.75.3.12 operator+ . . . 443
10.75.3.13 operator- . . . 443
10.75.3.14 operator= . . . 443
10.75.4 Friends And Related Function Documentation . . . 443
10.75.4.1 operator<< . . . 443
10.76 gazebo::common::LogPlay Class Reference . . . 444
10.76.1 Member Function Documentation . . . 445
10.76.1.1 GetChunk . . . 445
10.76.1.2 GetChunkCount . . . 445
10.76.1.3 GetEncoding . . . 445
10.76.1.4 GetGazeboVersion . . . 445
10.76.1.5 GetLogVersion . . . 445
10.76.1.6 GetRandSeed . . . 446
10.76.1.7 IsOpen . . . 446
10.76.1.8 Open . . . 446
10.76.1.9 Step . . . 446
10.77 Logplay Class Reference . . . 446
10.77.1 Detailed Description . . . 447
10.78 gazebo::common::LogRecord Class Reference . . . 447
10.78.1 Detailed Description . . . 448
10.78.2 Member Function Documentation . . . 448
10.78.2.1 Add . . . 448
10.78.2.2 Fini . . . 449
10.78.2.3 GetBasePath . . . 449
10.78.2.4 GetEncoding . . . 449
10.78.2.5 GetFilename . . . 449

10.78.2.6 GetFileSize . . . 449
10.78.2.7 GetFirstUpdate . . . 450
10.78.2.8 GetPaused . . . 450
10.78.2.9 GetRunning . . . 450
10.78.2.10 GetRunTime . . . 450
10.78.2.11 Init . . . 450
10.78.2.12 Remove . . . 451
10.78.2.13 SetBasePath . . . 451
10.78.2.14 SetPaused . . . 451
10.78.2.15 Start . . . 451
10.78.2.16 Stop . . . 452
10.79 gazebo::Master Class Reference . . . 452
10.79.1 Detailed Description . . . 452
10.79.2 Constructor & Destructor Documentation . . . 453
10.79.2.1 Master . . . 453
10.79.2.2 ∼Master . . . 453
10.79.3 Member Function Documentation . . . 453
10.79.3.1 Fini . . . 453
10.79.3.2 Init . . . 453
10.79.3.3 Run . . . 453
10.79.3.4 RunOnce . . . 453
10.79.3.5 RunThread . . . 453
10.79.3.6 Stop . . . 453
10.80 gazebo::common::Material Class Reference . . . 453
10.80.1 Detailed Description . . . 456
10.80.2 Member Enumeration Documentation . . . 456
10.80.2.1 BlendMode . . . 456
10.80.2.2 ShadeMode . . . 456
10.80.3 Constructor & Destructor Documentation . . . 456
10.80.3.1 Material . . . 456
10.80.3.2 ∼Material . . . 457
10.80.3.3 Material . . . 457
10.80.4 Member Function Documentation . . . 457
10.80.4.1 GetAmbient . . . 457
10.80.4.2 GetBlendFactors . . . 457
10.80.4.3 GetBlendMode . . . 457
10.80.4.4 GetDepthWrite . . . 457

10.80.4.5 GetDiffuse . . . 457
10.80.4.6 GetEmissive . . . 458
10.80.4.7 GetLighting . . . 458
10.80.4.8 GetName . . . 458
10.80.4.9 GetPointSize . . . 458
10.80.4.10 GetShadeMode . . . 458
10.80.4.11 GetShininess . . . 458
10.80.4.12 GetSpecular . . . 459
10.80.4.13 GetTextureImage . . . 459
10.80.4.14 GetTransparency . . . 459
10.80.4.15 SetAmbient . . . 459
10.80.4.16 SetBlendFactors . . . 459
10.80.4.17 SetBlendMode . . . 459
10.80.4.18 SetDepthWrite . . . 460
10.80.4.19 SetDiffuse . . . 460
10.80.4.20 SetEmissive . . . 460
10.80.4.21 SetLighting . . . 460
10.80.4.22 SetPointSize . . . 460
10.80.4.23 SetShadeMode . . . 460
10.80.4.24 SetShininess . . . 461
10.80.4.25 SetSpecular . . . 461
10.80.4.26 SetTextureImage . . . 461
10.80.4.27 SetTextureImage . . . 461
10.80.4.28 SetTransparency . . . 461
10.80.5 Friends And Related Function Documentation . . . 461
10.80.5.1 operator<< . . . 461
10.80.6 Member Data Documentation . . . 462
10.80.6.1 ambient . . . 462
10.80.6.2 blendMode . . . 462
10.80.6.3 BlendModeStr . . . 462
10.80.6.4 diffuse . . . 462
10.80.6.5 emissive . . . 462
10.80.6.6 name . . . 462
10.80.6.7 pointSize . . . 462
10.80.6.8 shadeMode . . . 462
10.80.6.9 ShadeModeStr . . . 462
10.80.6.10 shininess . . . 462

10.80.6.11specular . . . 462
10.80.6.12texImage . . . 462
10.80.6.13transparency . . . 463
10.81gazebo::math::Matrix3 Class Reference . . . 463
10.81.1 Detailed Description . . . 464
10.81.2 Constructor & Destructor Documentation . . . 464
10.81.2.1 Matrix3 . . . 464
10.81.2.2 Matrix3 . . . 464
10.81.2.3 Matrix3 . . . 464
10.81.2.4 ∼Matrix3 . . . 465
10.81.3 Member Function Documentation . . . 465
10.81.3.1 operator∗ . . . 465
10.81.3.2 operator∗ . . . 465
10.81.3.3 operator+ . . . 465
10.81.3.4 operator- . . . 465
10.81.3.5 operator== . . . 465
10.81.3.6 operator[] . . . 466
10.81.3.7 operator[] . . . 466
10.81.3.8 SetCol . . . 466
10.81.3.9 SetFromAxes . . . 466
10.81.3.10SetFromAxis . . . 466
10.81.4 Friends And Related Function Documentation . . . 467
10.81.4.1 operator∗ . . . 467
10.81.4.2 operator<< . . . 467
10.81.5 Member Data Documentation . . . 467
10.81.5.1 m . . . 467
10.82gazebo::math::Matrix4 Class Reference . . . 467
10.82.1 Detailed Description . . . 469
10.82.2 Constructor & Destructor Documentation . . . 469
10.82.2.1 Matrix4 . . . 469
10.82.2.2 Matrix4 . . . 469
10.82.2.3 Matrix4 . . . 469
10.82.2.4 ∼Matrix4 . . . 470
10.82.3 Member Function Documentation . . . 470
10.82.3.1 GetAsPose . . . 470
10.82.3.2 GetEulerRotation . . . 470
10.82.3.3 GetRotation . . . 470

10.82.3.4 GetTranslation . . . 470
10.82.3.5 Inverse . . . 471
10.82.3.6 IsAffine . . . 471
10.82.3.7 operator∗ . . . 471
10.82.3.8 operator∗ . . . 471
10.82.3.9 operator∗ . . . 471
10.82.3.10 operator= . . . 472
10.82.3.11 operator= . . . 472
10.82.3.12 operator== . . . 472
10.82.3.13 operator[] . . . 472
10.82.3.14 operator[] . . . 473
10.82.3.15 Set . . . 473
10.82.3.16 SetScale . . . 473
10.82.3.17 SetTranslate . . . 473
10.82.3.18 TransformAffine . . . 474
10.82.4 Friends And Related Function Documentation . . . 474
10.82.4.1 operator<< . . . 474
10.82.5 Member Data Documentation . . . 474
10.82.5.1 IDENTITY . . . 474
10.82.5.2 m . . . 474
10.82.5.3 ZERO . . . 474
10.83 gazebo::common::Mesh Class Reference . . . 475
10.83.1 Detailed Description . . . 476
10.83.2 Constructor & Destructor Documentation . . . 476
10.83.2.1 Mesh . . . 476
10.83.2.2 ∼Mesh . . . 476
10.83.3 Member Function Documentation . . . 476
10.83.3.1 AddMaterial . . . 476
10.83.3.2 AddSubMesh . . . 477
10.83.3.3 Center . . . 477
10.83.3.4 FillArrays . . . 477
10.83.3.5 GenSphericalTexCoord . . . 477
10.83.3.6 GetAABB . . . 477
10.83.3.7 GetIndexCount . . . 478
10.83.3.8 GetMaterial . . . 478
10.83.3.9 GetMaterialCount . . . 478
10.83.3.10 GetMax . . . 478

10.83.3.11 GetMin . . . 478
10.83.3.12 GetName . . . 478
10.83.3.13 GetNormalCount . . . 479
10.83.3.14 GetPath . . . 479
10.83.3.15 GetSkeleton . . . 479
10.83.3.16 GetSubMesh . . . 479
10.83.3.17 GetSubMesh . . . 479
10.83.3.18 GetSubMeshCount . . . 480
10.83.3.19 GetTexCoordCount . . . 480
10.83.3.20 GetVertexCount . . . 480
10.83.3.21 HasSkeleton . . . 480
10.83.3.22 RecalculateNormals . . . 480
10.83.3.23 Scale . . . 480
10.83.3.24 SetName . . . 480
10.83.3.25 SetPath . . . 481
10.83.3.26 SetScale . . . 481
10.83.3.27 SetSkeleton . . . 481
10.83.3.28 Translate . . . 481
10.84 gazebo::common::MeshCSG Class Reference . . . 481
10.84.1 Detailed Description . . . 482
10.84.2 Member Enumeration Documentation . . . 482
10.84.2.1 BooleanOperation . . . 482
10.84.3 Constructor & Destructor Documentation . . . 482
10.84.3.1 MeshCSG . . . 482
10.84.3.2 ∼MeshCSG . . . 482
10.84.4 Member Function Documentation . . . 482
10.84.4.1 CreateBoolean . . . 482
10.85 gazebo::common::MeshLoader Class Reference . . . 483
10.85.1 Detailed Description . . . 483
10.85.2 Constructor & Destructor Documentation . . . 483
10.85.2.1 MeshLoader . . . 483
10.85.2.2 ∼MeshLoader . . . 483
10.85.3 Member Function Documentation . . . 484
10.85.3.1 Load . . . 484
10.86 gazebo::common::MeshManager Class Reference . . . 484
10.86.1 Detailed Description . . . 485
10.86.2 Member Function Documentation . . . 485

10.86.2.1 AddMesh . . . 485
10.86.2.2 CreateBox . . . 486
10.86.2.3 CreateCamera . . . 486
10.86.2.4 CreateCone . . . 486
10.86.2.5 CreateCylinder . . . 486
10.86.2.6 CreatePlane . . . 486
10.86.2.7 CreatePlane . . . 487
10.86.2.8 CreateSphere . . . 487
10.86.2.9 CreateTube . . . 487
10.86.2.10 GenSphericalTexCoord . . . 488
10.86.2.11 GetMesh . . . 488
10.86.2.12 GetMeshAABB . . . 488
10.86.2.13 HasMesh . . . 488
10.86.2.14 IsValidFilename . . . 488
10.86.2.15 Load . . . 488
10.87 gazebo::physics::Model Class Reference . . . 489
10.87.1 Detailed Description . . . 492
10.87.2 Constructor & Destructor Documentation . . . 492
10.87.2.1 Model . . . 492
10.87.2.2 ∼Model . . . 492
10.87.3 Member Function Documentation . . . 492
10.87.3.1 AttachStaticModel . . . 493
10.87.3.2 DetachStaticModel . . . 493
10.87.3.3 FillMsg . . . 493
10.87.3.4 Fini . . . 493
10.87.3.5 GetAutoDisable . . . 493
10.87.3.6 GetBoundingBox . . . 494
10.87.3.7 GetJoint . . . 494
10.87.3.8 GetJointController . . . 494
10.87.3.9 GetJointCount . . . 494
10.87.3.10 GetJoints . . . 494
10.87.3.11 GetLink . . . 495
10.87.3.12 GetLinkById . . . 495
10.87.3.13 GetLinks . . . 495
10.87.3.14 GetPluginCount . . . 495
10.87.3.15 GetRelativeAngularAccel . . . 495
10.87.3.16 GetRelativeAngularVel . . . 496

10.87.3.17GetRelativeLinearAccel . . . 496
10.87.3.18GetRelativeLinearVel . . . 496
10.87.3.19GetSDF . . . 496
10.87.3.20GetSensorCount . . . 496
10.87.3.21GetWorldAngularAccel . . . 497
10.87.3.22GetWorldAngularVel . . . 497
10.87.3.23GetWorldLinearAccel . . . 497
10.87.3.24GetWorldLinearVel . . . 497
10.87.3.25Init . . . 497
10.87.3.26Load . . . 497
10.87.3.27LoadJoints . . . 498
10.87.3.28LoadPlugins . . . 498
10.87.3.29OnPoseChange . . . 498
10.87.3.30ProcessMsg . . . 498
10.87.3.31RemoveChild . . . 498
10.87.3.32Reset . . . 498
10.87.3.33SetAngularAccel . . . 498
10.87.3.34SetAngularVel . . . 499
10.87.3.35SetAutoDisable . . . 499
10.87.3.36SetCollideMode . . . 499
10.87.3.37SetEnabled . . . 499
10.87.3.38SetGravityMode . . . 499
10.87.3.39SetJointAnimation . . . 500
10.87.3.40SetJointPosition . . . 500
10.87.3.41SetJointPositions . . . 500
10.87.3.42SetLaserRetro . . . 500
10.87.3.43SetLinearAccel . . . 500
10.87.3.44SetLinearVel . . . 501
10.87.3.45SetLinkWorldPose . . . 501
10.87.3.46SetLinkWorldPose . . . 501
10.87.3.47SetState . . . 501
10.87.3.48StopAnimation . . . 501
10.87.3.49Update . . . 502
10.87.3.50UpdateParameters . . . 502
10.87.4 Member Data Documentation . . . 502
10.87.4.1 attachedModels . . . 502
10.87.4.2 attachedModelsOffset . . . 502

10.88gazebo::common::ModelDatabase Class Reference . . . 502
10.88.1 Detailed Description . . . 503
10.89gazebo::ModelPlugin Class Reference . . . 504
10.89.1 Detailed Description . . . 504
10.89.2 Constructor & Destructor Documentation . . . 504
10.89.2.1 ModelPlugin . . . 505
10.89.2.2 ∼ModelPlugin . . . 505
10.89.3 Member Function Documentation . . . 505
10.89.3.1 Init . . . 505
10.89.3.2 Load . . . 505
10.89.3.3 Reset . . . 505
10.90gazebo::physics::ModelState Class Reference . . . 505
10.90.1 Detailed Description . . . 507
10.90.2 Constructor & Destructor Documentation . . . 507
10.90.2.1 ModelState . . . 507
10.90.2.2 ModelState . . . 507
10.90.2.3 ModelState . . . 507
10.90.2.4 ∼ModelState . . . 508
10.90.3 Member Function Documentation . . . 508
10.90.3.1 FillSDF . . . 508
10.90.3.2 GetJointState . . . 508
10.90.3.3 GetJointState . . . 508
10.90.3.4 GetJointStateCount . . . 509
10.90.3.5 GetJointStates . . . 509
10.90.3.6 GetLinkState . . . 509
10.90.3.7 GetLinkState . . . 509
10.90.3.8 GetLinkStateCount . . . 510
10.90.3.9 GetLinkStates . . . 510
10.90.3.10 GetPose . . . 510
10.90.3.11 HasJointState . . . 510
10.90.3.12 HasLinkState . . . 511
10.90.3.13 IsZero . . . 511
10.90.3.14 Load . . . 511
10.90.3.15 operator+ . . . 511
10.90.3.16 operator- . . . 512
10.90.3.17 operator= . . . 512
10.90.4 Friends And Related Function Documentation . . . 512

10.90.4.1 operator<< . . . 512
10.91 gazebo::common::MouseEvent Class Reference . . . 512
10.91.1 Detailed Description . . . 513
10.91.2 Member Enumeration Documentation . . . 514
10.91.2.1 Buttons . . . 514
10.91.2.2 EventType . . . 514
10.91.3 Constructor & Destructor Documentation . . . 514
10.91.3.1 MouseEvent . . . 514
10.91.4 Member Data Documentation . . . 514
10.91.4.1 alt . . . 514
10.91.4.2 button . . . 514
10.91.4.3 buttons . . . 514
10.91.4.4 control . . . 514
10.91.4.5 dragging . . . 515
10.91.4.6 moveScale . . . 515
10.91.4.7 pos . . . 515
10.91.4.8 pressPos . . . 515
10.91.4.9 prevPos . . . 515
10.91.4.10 scroll . . . 515
10.91.4.11 shift . . . 515
10.91.4.12 type . . . 515
10.92 gazebo::rendering::MovableText Class Reference . . . 515
10.92.1 Detailed Description . . . 517
10.92.2 Member Enumeration Documentation . . . 517
10.92.2.1 HorizAlign . . . 517
10.92.2.2 VertAlign . . . 518
10.92.3 Constructor & Destructor Documentation . . . 518
10.92.3.1 MovableText . . . 518
10.92.3.2 ~MovableText . . . 518
10.92.4 Member Function Documentation . . . 518
10.92.4.1 _setupGeometry . . . 518
10.92.4.2 _updateColors . . . 518
10.92.4.3 GetAABB . . . 518
10.92.4.4 GetBaseline . . . 518
10.92.4.5 getBoundingRadius . . . 518
10.92.4.6 GetCharHeight . . . 518
10.92.4.7 GetColor . . . 519

10.92.4.8 GetFont . . . 519
10.92.4.9 getLights . . . 519
10.92.4.10 getMaterial . . . 519
10.92.4.11 getRenderOperation . . . 519
10.92.4.12 GetShowOnTop . . . 519
10.92.4.13 GetSpaceWidth . . . 519
10.92.4.14 getSquaredViewDepth . . . 519
10.92.4.15 GetText . . . 519
10.92.4.16 getWorldTransforms . . . 520
10.92.4.17 Load . . . 520
10.92.4.18 SetBaseline . . . 520
10.92.4.19 SetCharHeight . . . 520
10.92.4.20 SetColor . . . 520
10.92.4.21 SetFontName . . . 520
10.92.4.22 SetShowOnTop . . . 521
10.92.4.23 SetSpaceWidth . . . 521
10.92.4.24 SetText . . . 521
10.92.4.25 SetTextAlignment . . . 521
10.92.4.26 Update . . . 521
10.92.4.27 visitRenderables . . . 521
10.93 gazebo::msgs::MsgFactory Class Reference . . . 521
10.93.1 Detailed Description . . . 522
10.93.2 Member Function Documentation . . . 522
10.93.2.1 GetMsgTypes . . . 522
10.93.2.2 NewMsg . . . 522
10.93.2.3 RegisterMsg . . . 522
10.94 gazebo::sensors::MultiCameraSensor Class Reference . . . 523
10.94.1 Detailed Description . . . 524
10.94.2 Constructor & Destructor Documentation . . . 524
10.94.2.1 MultiCameraSensor . . . 524
10.94.2.2 ∼MultiCameraSensor . . . 524
10.94.3 Member Function Documentation . . . 524
10.94.3.1 Fini . . . 524
10.94.3.2 GetCamera . . . 524
10.94.3.3 GetCameraCount . . . 525
10.94.3.4 GetImageData . . . 525
10.94.3.5 GetImageHeight . . . 525

10.94.3.6 GetImageWidth . . . 526
10.94.3.7 GetTopic . . . 526
10.94.3.8 Init . . . 526
10.94.3.9 IsActive . . . 526
10.94.3.10 Load . . . 526
10.94.3.11 SaveFrame . . . 527
10.94.3.12 UpdateImpl . . . 527
10.95 gazebo::physics::MultiRayShape Class Reference . . . 527
10.95.1 Detailed Description . . . 530
10.95.2 Constructor & Destructor Documentation . . . 530
10.95.2.1 MultiRayShape . . . 530
10.95.2.2 ∼MultiRayShape . . . 530
10.95.3 Member Function Documentation . . . 530
10.95.3.1 AddRay . . . 530
10.95.3.2 ConnectNewLaserScans . . . 530
10.95.3.3 DisconnectNewLaserScans . . . 531
10.95.3.4 FillMsg . . . 531
10.95.3.5 GetFiducial . . . 531
10.95.3.6 GetMaxAngle . . . 531
10.95.3.7 GetMaxRange . . . 531
10.95.3.8 GetMinAngle . . . 532
10.95.3.9 GetMinRange . . . 532
10.95.3.10 GetRange . . . 532
10.95.3.11 GetResRange . . . 532
10.95.3.12 GetRetro . . . 532
10.95.3.13 GetSampleCount . . . 533
10.95.3.14 GetScanResolution . . . 533
10.95.3.15 GetVerticalMaxAngle . . . 533
10.95.3.16 GetVerticalMinAngle . . . 533
10.95.3.17 GetVerticalSampleCount . . . 533
10.95.3.18 GetVerticalScanResolution . . . 533
10.95.3.19 Init . . . 534
10.95.3.20 ProcessMsg . . . 534
10.95.3.21 Update . . . 534
10.95.3.22 UpdateRays . . . 534
10.95.4 Member Data Documentation . . . 534
10.95.4.1 horzElem . . . 534

10.95.4.2 newLaserScans . . . 534
10.95.4.3 offset . . . 534
10.95.4.4 rangeElem . . . 534
10.95.4.5 rayElem . . . 535
10.95.4.6 rays . . . 535
10.95.4.7 scanElem . . . 535
10.95.4.8 vertElem . . . 535
10.96 gazebo::transport::Node Class Reference . . . 535
10.96.1 Detailed Description . . . 537
10.96.2 Constructor & Destructor Documentation . . . 537
10.96.2.1 Node . . . 537
10.96.2.2 ∼Node . . . 537
10.96.3 Member Function Documentation . . . 537
10.96.3.1 Advertise . . . 537
10.96.3.2 DecodeTopicName . . . 537
10.96.3.3 EncodeTopicName . . . 537
10.96.3.4 Fini . . . 538
10.96.3.5 GetId . . . 538
10.96.3.6 GetMsgType . . . 538
10.96.3.7 GetTopicNamespace . . . 538
10.96.3.8 HandleData . . . 538
10.96.3.9 HandleMessage . . . 539
10.96.3.10 HasLatchedSubscriber . . . 539
10.96.3.11 Init . . . 539
10.96.3.12 InsertLatchedMsg . . . 539
10.96.3.13 InsertLatchedMsg . . . 540
10.96.3.14 ProcessIncoming . . . 540
10.96.3.15 ProcessPublishers . . . 540
10.96.3.16 Publish . . . 540
10.96.3.17 RemoveCallback . . . 540
10.96.3.18 Subscribe . . . 540
10.96.3.19 Subscribe . . . 541
10.96.3.20 Subscribe . . . 541
10.96.3.21 Subscribe . . . 541
10.97 gazebo::common::NodeAnimation Class Reference . . . 542
10.97.1 Detailed Description . . . 543
10.97.2 Constructor & Destructor Documentation . . . 543

10.97.2.1 NodeAnimation . . . 543
10.97.2.2 ∼NodeAnimation . . . 543
10.97.3 Member Function Documentation . . . 543
10.97.3.1 AddKeyFrame . . . 543
10.97.3.2 AddKeyFrame . . . 543
10.97.3.3 GetFrameAt . . . 544
10.97.3.4 GetFrameCount . . . 544
10.97.3.5 GetKeyFrame . . . 544
10.97.3.6 GetKeyFrame . . . 544
10.97.3.7 GetLength . . . 544
10.97.3.8 GetName . . . 545
10.97.3.9 GetTimeAtX . . . 545
10.97.3.10Scale . . . 545
10.97.3.11SetName . . . 545
10.97.4 Member Data Documentation . . . 545
10.97.4.1 keyFrames . . . 545
10.97.4.2 length . . . 545
10.97.4.3 name . . . 546
10.98gazebo::common::NodeAssignment Struct Reference . . . 546
10.98.1 Detailed Description . . . 546
10.98.2 Member Data Documentation . . . 546
10.98.2.1 nodeIndex . . . 546
10.98.2.2 vertexIndex . . . 546
10.98.2.3 weight . . . 546
10.99gazebo::common::NodeTransform Class Reference . . . 547
10.99.1 Detailed Description . . . 548
10.99.2 Member Enumeration Documentation . . . 548
10.99.2.1 TransformType . . . 548
10.99.3 Constructor & Destructor Documentation . . . 548
10.99.3.1 NodeTransform . . . 548
10.99.3.2 NodeTransform . . . 548
10.99.3.3 ∼NodeTransform . . . 549
10.99.4 Member Function Documentation . . . 549
10.99.4.1 Get . . . 549
10.99.4.2 GetSID . . . 549
10.99.4.3 GetType . . . 549
10.99.4.4 operator() . . . 549

10.99.4.5 operator* . . . 549
10.99.4.6 operator* . . . 550
10.99.4.7 PrintSource . . . 550
10.99.4.8 RecalculateMatrix . . . 550
10.99.4.9 Set . . . 550
10.99.4.10 SetComponent . . . 550
10.99.4.11 SetSID . . . 550
10.99.4.12 SetSourceValues . . . 551
10.99.4.13 SetSourceValues . . . 551
10.99.4.14 SetSourceValues . . . 551
10.99.4.15 SetType . . . 551
10.99.5 Member Data Documentation . . . 551
10.99.5.1 sid . . . 551
10.99.5.2 source . . . 551
10.99.5.3 transform . . . 551
10.99.5.4 type . . . 551
10.100 gazebo::common::NumericAnimation Class Reference . . . 552
10.100.1 Detailed Description . . . 552
10.100.2 Constructor & Destructor Documentation . . . 552
10.100.2.1 NumericAnimation . . . 552
10.100.2.2 ~NumericAnimation . . . 553
10.100.3 Member Function Documentation . . . 553
10.100.3.1 CreateKeyFrame . . . 553
10.100.3.2 GetInterpolatedKeyFrame . . . 553
10.101 gazebo::common::NumericKeyFrame Class Reference . . . 553
10.101.1 Detailed Description . . . 554
10.101.2 Constructor & Destructor Documentation . . . 554
10.101.2.1 NumericKeyFrame . . . 554
10.101.2.2 ~NumericKeyFrame . . . 555
10.101.3 Member Function Documentation . . . 555
10.101.3.1 GetValue . . . 555
10.101.3.2 SetValue . . . 555
10.101.4 Member Data Documentation . . . 555
10.101.4.1 value . . . 555
10.102 gazebo::rendering::OrbitViewController Class Reference . . . 555
10.102.1 Detailed Description . . . 557
10.102.2 Constructor & Destructor Documentation . . . 557

10.102.2.1 OrbitViewController . . . 557
10.102.2.2 ~OrbitViewController . . . 557
10.102.3 Member Function Documentation . . . 557
10.102.3.1 GetFocalPoint . . . 557
10.102.3.2 GetTypeString . . . 557
10.102.3.3 HandleKeyPressEvent . . . 557
10.102.3.4 HandleKeyReleaseEvent . . . 558
10.102.3.5 HandleMouseEvent . . . 558
10.102.3.6 Init . . . 558
10.102.3.7 Init . . . 558
10.102.3.8 SetDistance . . . 558
10.102.3.9 SetFocalPoint . . . 558
10.102.3.10 Update . . . 559
10.103 sdf::Param Class Reference . . . 559
10.103.1 Detailed Description . . . 561
10.103.2 Constructor & Destructor Documentation . . . 561
10.103.2.1 Param . . . 561
10.103.2.2 ~Param . . . 561
10.103.3 Member Function Documentation . . . 561
10.103.3.1 Clone . . . 561
10.103.3.2 Get . . . 561
10.103.3.3 Get . . . 561
10.103.3.4 Get . . . 561
10.103.3.5 Get . . . 561
10.103.3.6 Get . . . 561
10.103.3.7 Get . . . 561
10.103.3.8 Get . . . 562
10.103.3.9 Get . . . 562
10.103.3.10 Get . . . 562
10.103.3.11 Get . . . 562
10.103.3.12 Get . . . 562
10.103.3.13 Get . . . 562
10.103.3.14 Get . . . 562
10.103.3.15 Get . . . 562
10.103.3.16 GetAsString . . . 562
10.103.3.17 GetDefaultAsString . . . 562
10.103.3.18 GetDescription . . . 562

10.103.3.19 GetKey . . . 562
10.103.3.20 GetRequired . . . 562
10.103.3.21 GetSet . . . 562
10.103.3.22 GetTypeName . . . 562
10.103.3.23 IsBool . . . 562
10.103.3.24 IsChar . . . 562
10.103.3.25 IsColor . . . 563
10.103.3.26 IsDouble . . . 563
10.103.3.27 IsFloat . . . 563
10.103.3.28 IsInt . . . 563
10.103.3.29 IsPose . . . 563
10.103.3.30 IsQuaternion . . . 563
10.103.3.31 IsStr . . . 563
10.103.3.32 IsTime . . . 563
10.103.3.33 IsUInt . . . 563
10.103.3.34 IsVector2d . . . 563
10.103.3.35 IsVector2i . . . 563
10.103.3.36 IsVector3 . . . 563
10.103.3.37 Reset . . . 563
10.103.3.38 Set . . . 563
10.103.3.39 Set . . . 563
10.103.3.40 Set . . . 563
10.103.3.41 Set . . . 563
10.103.3.42 Set . . . 563
10.103.3.43 Set . . . 563
10.103.3.44 Set . . . 563
10.103.3.45 Set . . . 563
10.103.3.46 Set . . . 563
10.103.3.47 Set . . . 563
10.103.3.48 Set . . . 564
10.103.3.49 Set . . . 564
10.103.3.50 Set . . . 564
10.103.3.51 Set . . . 564
10.103.3.52 Set . . . 564
10.103.3.53 SetDescription . . . 564
10.103.3.54 SetFromString . . . 564
10.103.3.55 SetUpdateFunc . . . 564

10.103.3.56 Update . . . 564
10.103.4 Member Data Documentation . . . 564
10.103.4.1 description . . . 564
10.103.4.2 key . . . 564
10.103.4.3 required . . . 564
10.103.4.4 set . . . 564
10.103.4.5 typeName . . . 565
10.103.4.6 updateFunc . . . 565
10.104 ParamT< T > Class Template Reference . . . 565
10.105 sdf::ParamT< T > Class Template Reference . . . 565
10.105.1 Detailed Description . . . 566
10.105.2 Constructor & Destructor Documentation . . . 566
10.105.2.1 ParamT . . . 566
10.105.2.2 ~ParamT . . . 566
10.105.3 Member Function Documentation . . . 567
10.105.3.1 Clone . . . 567
10.105.3.2 GetAsString . . . 567
10.105.3.3 GetDefaultAsString . . . 567
10.105.3.4 GetDefaultValue . . . 567
10.105.3.5 GetValue . . . 567
10.105.3.6 operator* . . . 567
10.105.3.7 Reset . . . 567
10.105.3.8 Set . . . 567
10.105.3.9 SetFromString . . . 567
10.105.3.10 SetValue . . . 568
10.105.3.11 Update . . . 568
10.105.4 Friends And Related Function Documentation . . . 568
10.105.4.1 operator<< . . . 568
10.105.5 Member Data Documentation . . . 568
10.105.5.1 defaultValue . . . 568
10.105.5.2 value . . . 568
10.106 gazebo::physics::PhysicsEngine Class Reference . . . 568
10.106.1 Detailed Description . . . 571
10.106.2 Constructor & Destructor Documentation . . . 571
10.106.2.1 PhysicsEngine . . . 571
10.106.2.2 ~PhysicsEngine . . . 572
10.106.3 Member Function Documentation . . . 572

10.106.3.1 CreateCollision . . . 572
10.106.3.2 CreateCollision . . . 572
10.106.3.3 CreateJoint . . . 572
10.106.3.4 CreateLink . . . 572
10.106.3.5 CreateShape . . . 572
10.106.3.6 DebugPrint . . . 573
10.106.3.7 Fini . . . 573
10.106.3.8 GetAutoDisableFlag . . . 573
10.106.3.9 GetContactManager . . . 573
10.106.3.10 GetContactMaxCorrectingVel . . . 573
10.106.3.11 GetContactSurfaceLayer . . . 573
10.106.3.12 GetGravity . . . 574
10.106.3.13 GetMaxContacts . . . 574
10.106.3.14 GetMaxStepSize . . . 574
10.106.3.15 GetParam . . . 574
10.106.3.16 GetPhysicsUpdateMutex . . . 574
10.106.3.17 GetRealTimeUpdateRate . . . 574
10.106.3.18 GetSORPGSIters . . . 575
10.106.3.19 GetSORPGSPreconIters . . . 575
10.106.3.20 GetSORPGSW . . . 575
10.106.3.21 GetStepTime . . . 575
10.106.3.22 GetTargetRealTimeFactor . . . 575
10.106.3.23 GetType . . . 576
10.106.3.24 GetUpdatePeriod . . . 576
10.106.3.25 GetUpdateRate . . . 576
10.106.3.26 GetWorldCFM . . . 576
10.106.3.27 GetWorldERP . . . 576
10.106.3.28 Init . . . 576
10.106.3.29 InitForThread . . . 577
10.106.3.30 Load . . . 577
10.106.3.31 OnPhysicsMsg . . . 577
10.106.3.32 OnRequest . . . 577
10.106.3.33 Reset . . . 577
10.106.3.34 SetAutoDisableFlag . . . 577
10.106.3.35 SetContactMaxCorrectingVel . . . 577
10.106.3.36 SetContactSurfaceLayer . . . 578
10.106.3.37 SetGravity . . . 578

10.106.3.38 SetMaxContacts . . . 578
10.106.3.39 SetMaxStepSize . . . 578
10.106.3.40 SetParam . . . 578
10.106.3.41 SetRealTimeUpdateRate . . . 579
10.106.3.42 SetSeed . . . 579
10.106.3.43 SetSORPGSIters . . . 579
10.106.3.44 SetSORPGSPreconIters . . . 579
10.106.3.45 SetSORPGSW . . . 579
10.106.3.46 SetStepTime . . . 579
10.106.3.47 SetTargetRealTimeFactor . . . 580
10.106.3.48 SetUpdateRate . . . 580
10.106.3.49 SetWorldCFM . . . 580
10.106.3.50 SetWorldERP . . . 580
10.106.3.51 UpdateCollision . . . 580
10.106.3.52 UpdatePhysics . . . 581
10.106.4 Member Data Documentation . . . 581
10.106.4.1 contactManager . . . 581
10.106.4.2 maxStepSize . . . 581
10.106.4.3 node . . . 581
10.106.4.4 physicsSub . . . 581
10.106.4.5 physicsUpdateMutex . . . 581
10.106.4.6 realTimeUpdateRate . . . 581
10.106.4.7 requestSub . . . 581
10.106.4.8 responsePub . . . 581
10.106.4.9 sdf . . . 581
10.106.4.10 targetRealTimeFactor . . . 581
10.106.4.11 world . . . 582
10.107 gazebo::physics::PhysicsFactory Class Reference . . . 582
10.107.1 Detailed Description . . . 582
10.107.2 Member Function Documentation . . . 582
10.107.2.1 IsRegistered . . . 582
10.107.2.2 NewPhysicsEngine . . . 582
10.107.2.3 RegisterAll . . . 583
10.107.2.4 RegisterPhysicsEngine . . . 583
10.108 gazebo::common::PID Class Reference . . . 583
10.108.1 Detailed Description . . . 584
10.108.2 Constructor & Destructor Documentation . . . 584

10.108.2.1 PID . . . 584
10.108.2.2 ~PID . . . 584
10.108.3 Member Function Documentation . . . 584
10.108.3.1 GetCmd . . . 585
10.108.3.2 GetErrors . . . 585
10.108.3.3 Init . . . 585
10.108.3.4 operator= . . . 585
10.108.3.5 Reset . . . 585
10.108.3.6 SetCmd . . . 586
10.108.3.7 SetCmdMax . . . 586
10.108.3.8 SetCmdMin . . . 586
10.108.3.9 SetDGain . . . 586
10.108.3.10 SetIGain . . . 586
10.108.3.11 SetIMax . . . 586
10.108.3.12 SetIMin . . . 587
10.108.3.13 SetPGain . . . 587
10.108.3.14 Update . . . 587
10.109 gazebo::math::Plane Class Reference . . . 587
10.109.1 Detailed Description . . . 588
10.109.2 Constructor & Destructor Documentation . . . 588
10.109.2.1 Plane . . . 588
10.109.2.2 Plane . . . 588
10.109.2.3 Plane . . . 588
10.109.2.4 ~Plane . . . 589
10.109.3 Member Function Documentation . . . 589
10.109.3.1 Distance . . . 589
10.109.3.2 operator= . . . 589
10.109.3.3 Set . . . 589
10.109.4 Member Data Documentation . . . 589
10.109.4.1 d . . . 589
10.109.4.2 normal . . . 590
10.109.4.3 size . . . 590
10.110 gazebo::physics::PlaneShape Class Reference . . . 590
10.110.1 Detailed Description . . . 591
10.110.2 Constructor & Destructor Documentation . . . 591
10.110.2.1 PlaneShape . . . 591
10.110.2.2 ~PlaneShape . . . 591

10.110.3 Member Function Documentation . . . 591
10.110.3.1 CreatePlane . . . 591
10.110.3.2 FillMsg . . . 592
10.110.3.3 GetNormal . . . 592
10.110.3.4 GetSize . . . 592
10.110.3.5 Init . . . 592
10.110.3.6 ProcessMsg . . . 592
10.110.3.7 SetAltitude . . . 592
10.110.3.8 SetNormal . . . 593
10.110.3.9 SetSize . . . 593
10.111 sdf::Plugin Class Reference . . . 593
10.111.1 Constructor & Destructor Documentation . . . 594
10.111.1.1 Plugin . . . 594
10.111.2 Member Function Documentation . . . 594
10.111.2.1 Clear . . . 594
10.111.2.2 Print . . . 594
10.111.3 Member Data Documentation . . . 594
10.111.3.1 data . . . 594
10.111.3.2 filename . . . 594
10.111.3.3 name . . . 594
10.112 gazebo::PluginT< T > Class Template Reference . . . 594
10.112.1 Detailed Description . . . 595
10.112.2 Member Typedef Documentation . . . 595
10.112.2.1 TPtr . . . 595
10.112.3 Member Function Documentation . . . 595
10.112.3.1 Create . . . 595
10.112.3.2 GetFilename . . . 596
10.112.3.3 GetHandle . . . 596
10.112.3.4 GetType . . . 596
10.112.4 Member Data Documentation . . . 596
10.112.4.1 filename . . . 596
10.112.4.2 handle . . . 596
10.112.4.3 type . . . 596
10.113 gazebo::math::Pose Class Reference . . . 596
10.113.1 Detailed Description . . . 598
10.113.2 Constructor & Destructor Documentation . . . 598
10.113.2.1 Pose . . . 598

10.113.2.2 Pose . . . 598
10.113.2.3 Pose . . . 599
10.113.2.4 Pose . . . 599
10.113.2.5 ~Pose . . . 599
10.113.3 Member Function Documentation . . . 599
10.113.3.1 CoordPoseSolve . . . 599
10.113.3.2 CoordPositionAdd . . . 599
10.113.3.3 CoordPositionAdd . . . 600
10.113.3.4 CoordPositionSub . . . 600
10.113.3.5 CoordRotationAdd . . . 600
10.113.3.6 CoordRotationSub . . . 600
10.113.3.7 Correct . . . 601
10.113.3.8 GetInverse . . . 601
10.113.3.9 IsFinite . . . 601
10.113.3.10 operator!= . . . 601
10.113.3.11 operator* . . . 601
10.113.3.12 operator+ . . . 602
10.113.3.13 operator+= . . . 602
10.113.3.14 operator- . . . 602
10.113.3.15 operator- . . . 602
10.113.3.16 operator-= . . . 602
10.113.3.17 operator== . . . 603
10.113.3.18 Reset . . . 603
10.113.3.19 RotatePositionAboutOrigin . . . 603
10.113.3.20 Round . . . 603
10.113.3.21 Set . . . 603
10.113.3.22 Set . . . 604
10.113.4 Friends And Related Function Documentation . . . 604
10.113.4.1 operator<< . . . 604
10.113.4.2 operator>> . . . 604
10.113.5 Member Data Documentation . . . 604
10.113.5.1 pos . . . 604
10.113.5.2 rot . . . 605
10.113.5.3 Zero . . . 605
10.114 gazebo::common::PoseAnimation Class Reference . . . 605
10.114.1 Detailed Description . . . 606
10.114.2 Constructor & Destructor Documentation . . . 606

10.114.2.1 PoseAnimation . . . 606
10.114.2.2 ~PoseAnimation . . . 606
10.114.3 Member Function Documentation . . . 606
10.114.3.1 BuildInterpolationSplines . . . 606
10.114.3.2 CreateKeyFrame . . . 606
10.114.3.3 GetInterpolatedKeyFrame . . . 607
10.114.3.4 GetInterpolatedKeyFrame . . . 607
10.115 gazebo::common::PoseKeyFrame Class Reference . . . 607
10.115.1 Detailed Description . . . 608
10.115.2 Constructor & Destructor Documentation . . . 608
10.115.2.1 PoseKeyFrame . . . 608
10.115.2.2 ~PoseKeyFrame . . . 608
10.115.3 Member Function Documentation . . . 608
10.115.3.1 GetRotation . . . 608
10.115.3.2 GetTranslation . . . 608
10.115.3.3 SetRotation . . . 609
10.115.3.4 SetTranslation . . . 609
10.115.4 Member Data Documentation . . . 609
10.115.4.1 rotate . . . 609
10.115.4.2 translate . . . 609
10.116 gazebo::rendering::Projector Class Reference . . . 609
10.116.1 Detailed Description . . . 610
10.116.2 Constructor & Destructor Documentation . . . 610
10.116.2.1 Projector . . . 610
10.116.2.2 ~Projector . . . 610
10.116.3 Member Function Documentation . . . 610
10.116.3.1 GetParent . . . 610
10.116.3.2 Load . . . 610
10.116.3.3 Load . . . 611
10.116.3.4 Load . . . 611
10.116.3.5 SetEnabled . . . 611
10.116.3.6 SetTexture . . . 611
10.116.3.7 Toggle . . . 611
10.117 gazebo::transport::Publication Class Reference . . . 612
10.117.1 Detailed Description . . . 613
10.117.2 Constructor & Destructor Documentation . . . 613
10.117.2.1 Publication . . . 613

10.117.2.2 ~Publication . . . 613
10.117.3 Member Function Documentation . . . 613
10.117.3.1 AddPublisher . . . 613
10.117.3.2 AddSubscription . . . 613
10.117.3.3 AddSubscription . . . 613
10.117.3.4 AddTransport . . . 614
10.117.3.5 GetCallbackCount . . . 614
10.117.3.6 GetLocallyAdvertised . . . 614
10.117.3.7 GetMsgType . . . 614
10.117.3.8 GetNodeCount . . . 614
10.117.3.9 GetRemoteSubscriptionCount . . . 614
10.117.3.10 GetTransportCount . . . 615
10.117.3.11 HasTransport . . . 615
10.117.3.12 LocalPublish . . . 615
10.117.3.13 Publish . . . 615
10.117.3.14 RemoveSubscription . . . 615
10.117.3.15 RemoveSubscription . . . 616
10.117.3.16 RemoveTransport . . . 616
10.117.3.17 SetLocallyAdvertised . . . 616
10.118 gazebo::transport::PublicationTransport Class Reference . . . 616
10.118.1 Detailed Description . . . 617
10.118.2 Constructor & Destructor Documentation . . . 617
10.118.2.1 PublicationTransport . . . 617
10.118.2.2 ~PublicationTransport . . . 617
10.118.3 Member Function Documentation . . . 617
10.118.3.1 AddCallback . . . 617
10.118.3.2 Fini . . . 617
10.118.3.3 GetConnection . . . 617
10.118.3.4 GetMsgType . . . 618
10.118.3.5 GetTopic . . . 618
10.118.3.6 Init . . . 618
10.119 gazebo::transport::Publisher Class Reference . . . 618
10.119.1 Detailed Description . . . 619
10.119.2 Constructor & Destructor Documentation . . . 619
10.119.2.1 Publisher . . . 619
10.119.2.2 Publisher . . . 619
10.119.2.3 ~Publisher . . . 620

10.119.3 Member Function Documentation . . . 620
10.119.3.1 GetLatching . . . 620
10.119.3.2 GetMsgType . . . 620
10.119.3.3 GetOutgoingCount . . . 620
10.119.3.4 GetPrevMsg . . . 620
10.119.3.5 GetPrevMsgPtr . . . 620
10.119.3.6 GetTopic . . . 620
10.119.3.7 HasConnections . . . 621
10.119.3.8 Publish . . . 621
10.119.3.9 Publish . . . 621
10.119.3.10 SendMessage . . . 621
10.119.3.11 SetPublication . . . 621
10.119.3.12 SetPublication . . . 621
10.119.3.13 WaitForConnection . . . 622
10.120 gazebo::transport::PublishTask Class Reference . . . 622
10.120.1 Detailed Description . . . 622
10.120.2 Constructor & Destructor Documentation . . . 622
10.120.2.1 PublishTask . . . 622
10.120.3 Member Function Documentation . . . 623
10.120.3.1 execute . . . 623
10.121 gazebo::math::Quaternion Class Reference . . . 623
10.121.1 Detailed Description . . . 626
10.121.2 Constructor & Destructor Documentation . . . 626
10.121.2.1 Quaternion . . . 626
10.121.2.2 Quaternion . . . 626
10.121.2.3 Quaternion . . . 626
10.121.2.4 Quaternion . . . 626
10.121.2.5 Quaternion . . . 627
10.121.2.6 Quaternion . . . 627
10.121.2.7 ~Quaternion . . . 627
10.121.3 Member Function Documentation . . . 627
10.121.3.1 Correct . . . 627
10.121.3.2 Dot . . . 627
10.121.3.3 EulerToQuaternion . . . 627
10.121.3.4 EulerToQuaternion . . . 628
10.121.3.5 GetAsAxis . . . 628
10.121.3.6 GetAsEuler . . . 628

10.121.3.7 GetAsMatrix3 . . . 628
10.121.3.8 GetAsMatrix4 . . . 628
10.121.3.9 GetExp . . . 628
10.121.3.10 GetInverse . . . 629
10.121.3.11 GetLog . . . 629
10.121.3.12 GetPitch . . . 629
10.121.3.13 GetRoll . . . 629
10.121.3.14 GetXAxis . . . 629
10.121.3.15 GetYaw . . . 629
10.121.3.16 GetYAxis . . . 630
10.121.3.17 GetZAxis . . . 630
10.121.3.18 Invert . . . 630
10.121.3.19 IsFinite . . . 630
10.121.3.20 Normalize . . . 630
10.121.3.21 operator!= . . . 630
10.121.3.22 operator* . . . 630
10.121.3.23 operator* . . . 631
10.121.3.24 operator* . . . 631
10.121.3.25 operator*= . . . 631
10.121.3.26 operator+ . . . 631
10.121.3.27 operator+= . . . 632
10.121.3.28 operator- . . . 632
10.121.3.29 operator- . . . 632
10.121.3.30 operator-= . . . 632
10.121.3.31 operator= . . . 633
10.121.3.32 operator== . . . 633
10.121.3.33 RotateVector . . . 633
10.121.3.34 RotateVectorReverse . . . 633
10.121.3.35 Round . . . 633
10.121.3.36 Scale . . . 634
10.121.3.37 Set . . . 634
10.121.3.38 SetFromAxis . . . 634
10.121.3.39 SetFromAxis . . . 634
10.121.3.40 SetFromEuler . . . 634
10.121.3.41 SetFromEuler . . . 635
10.121.3.42 SetToIdentity . . . 635
10.121.3.43 Slerp . . . 635

10.121.3.4 squad . . . 635
10.121.4 Friends And Related Function Documentation . . . 635
10.121.4.1 operator<< . . . 635
10.121.4.2 operator>> . . . 636
10.121.5 Member Data Documentation . . . 636
10.121.5.1 w . . . 636
10.121.5.2 x . . . 636
10.121.5.3 y . . . 636
10.121.5.4 z . . . 636
10.122 gazebo::math::Rand Class Reference . . . 637
10.122.1 Detailed Description . . . 637
10.122.2 Member Function Documentation . . . 637
10.122.2.1 GetDblNormal . . . 637
10.122.2.2 GetDblUniform . . . 637
10.122.2.3 GetIntNormal . . . 637
10.122.2.4 GetIntUniform . . . 638
10.122.2.5 GetSeed . . . 638
10.122.2.6 SetSeed . . . 638
10.123 gazebo::transport::RawCallbackHelper Class Reference . . . 638
10.123.1 Detailed Description . . . 639
10.123.2 Constructor & Destructor Documentation . . . 639
10.123.2.1 RawCallbackHelper . . . 639
10.123.3 Member Function Documentation . . . 640
10.123.3.1 GetMsgType . . . 640
10.123.3.2 HandleData . . . 640
10.123.3.3 HandleMessage . . . 640
10.123.3.4 IsLocal . . . 640
10.124 gazebo::sensors::RaySensor Class Reference . . . 641
10.124.1 Detailed Description . . . 642
10.124.2 Constructor & Destructor Documentation . . . 643
10.124.2.1 RaySensor . . . 643
10.124.2.2 ~RaySensor . . . 643
10.124.3 Member Function Documentation . . . 643
10.124.3.1 Fini . . . 643
10.124.3.2 GetAngleMax . . . 643
10.124.3.3 GetAngleMin . . . 643
10.124.3.4 GetAngleResolution . . . 643

10.124.3.5 GetFiducial . . . 643
10.124.3.6 GetLaserShape . . . 644
10.124.3.7 GetRange . . . 644
10.124.3.8 GetRangeCount . . . 644
10.124.3.9 GetRangeMax . . . 644
10.124.3.10 GetRangeMin . . . 645
10.124.3.11 GetRangeResolution . . . 645
10.124.3.12 GetRanges . . . 645
10.124.3.13 GetRayCount . . . 645
10.124.3.14 GetRetro . . . 645
10.124.3.15 GetTopic . . . 646
10.124.3.16 GetVerticalAngleMax . . . 646
10.124.3.17 GetVerticalAngleMin . . . 646
10.124.3.18 GetVerticalRangeCount . . . 646
10.124.3.19 GetVerticalRayCount . . . 646
10.124.3.20 Init . . . 646
10.124.3.21 IsActive . . . 647
10.124.3.22 Load . . . 647
10.124.3.23 UpdateImpl . . . 647
10.125 gazebo::physics::RayShape Class Reference . . . 647
10.125.1 Detailed Description . . . 649
10.125.2 Constructor & Destructor Documentation . . . 649
10.125.2.1 RayShape . . . 649
10.125.2.2 RayShape . . . 650
10.125.2.3 ~RayShape . . . 650
10.125.3 Member Function Documentation . . . 650
10.125.3.1 FillMsg . . . 650
10.125.3.2 GetFiducial . . . 650
10.125.3.3 GetGlobalPoints . . . 650
10.125.3.4 GetIntersection . . . 650
10.125.3.5 GetLength . . . 651
10.125.3.6 GetRelativePoints . . . 651
10.125.3.7 GetRetro . . . 651
10.125.3.8 Init . . . 651
10.125.3.9 ProcessMsg . . . 651
10.125.3.10 SetFiducial . . . 652
10.125.3.11 SetLength . . . 652

10.125.3.12SetPoints . . . 652
10.125.3.13SetRetro . . . 652
10.125.3.14Update . . . 652
10.125.4Member Data Documentation . . . 652
10.125.4.1contactFiducial . . . 652
10.125.4.2contactLen . . . 653
10.125.4.3contactRetro . . . 653
10.125.4.4globalEndPos . . . 653
10.125.4.5globalStartPos . . . 653
10.125.4.6relativeEndPos . . . 653
10.125.4.7relativeStartPos . . . 653
10.126gazebo::rendering::RenderEngine Class Reference . . . 653
10.126.1Detailed Description . . . 655
10.126.2Member Enumeration Documentation . . . 655
10.126.2.1RenderPathType . . . 655
10.126.3Member Function Documentation . . . 655
10.126.3.1AddResourcePath . . . 655
10.126.3.2CreateScene . . . 656
10.126.3.3Fini . . . 656
10.126.3.4GetRenderPathType . . . 656
10.126.3.5GetScene . . . 656
10.126.3.6GetScene . . . 656
10.126.3.7GetSceneCount . . . 657
10.126.3.8Init . . . 657
10.126.3.9Load . . . 657
10.126.3.10RemoveScene . . . 657
10.126.4Member Data Documentation . . . 657
10.126.4.1dummyContext . . . 657
10.126.4.2dummyDisplay . . . 657
10.126.4.3dummyWindowId . . . 657
10.126.4.4root . . . 657
10.127gazebo::sensors::RFIDSensor Class Reference . . . 658
10.127.1Detailed Description . . . 659
10.127.2Constructor & Destructor Documentation . . . 659
10.127.2.1RFIDSensor . . . 659
10.127.2.2~RFIDSensor . . . 659
10.127.3Member Function Documentation . . . 659

10.127.3.1 AddTag . . . 659
10.127.3.2 Fini . . . 659
10.127.3.3 Init . . . 659
10.127.3.4 Load . . . 659
10.127.3.5 Load . . . 659
10.127.3.6 UpdateImpl . . . 660
10.128 gazebo::sensors::RFIDTag Class Reference . . . 660
10.128.1 Detailed Description . . . 661
10.128.2 Constructor & Destructor Documentation . . . 661
10.128.2.1 RFIDTag . . . 661
10.128.2.2 ~RFIDTag . . . 661
10.128.3 Member Function Documentation . . . 661
10.128.3.1 Fini . . . 661
10.128.3.2 GetTagPose . . . 662
10.128.3.3 Init . . . 662
10.128.3.4 Load . . . 662
10.128.3.5 Load . . . 662
10.128.3.6 UpdateImpl . . . 662
10.129 gazebo::rendering::RFIDTagVisual Class Reference . . . 662
10.129.1 Detailed Description . . . 663
10.129.2 Constructor & Destructor Documentation . . . 663
10.129.2.1 RFIDTagVisual . . . 663
10.129.2.2 ~RFIDTagVisual . . . 664
10.130 gazebo::rendering::RFIDVisual Class Reference . . . 664
10.130.1 Detailed Description . . . 665
10.130.2 Constructor & Destructor Documentation . . . 665
10.130.2.1 RFIDVisual . . . 665
10.130.2.2 ~RFIDVisual . . . 665
10.131 gazebo::physics::Road Class Reference . . . 665
10.131.1 Detailed Description . . . 666
10.131.2 Constructor & Destructor Documentation . . . 666
10.131.2.1 Road . . . 666
10.131.2.2 ~Road . . . 667
10.131.3 Member Function Documentation . . . 667
10.131.3.1 Init . . . 667
10.131.3.2 Load . . . 667
10.132 Road Class Reference . . . 667

10.132.1 Detailed Description . . . 667
10.133 gazebo::rendering::Road2d Class Reference . . . 667
10.133.1 Constructor & Destructor Documentation . . . 668
10.133.1.1 Road2d . . . 668
10.133.1.2 ~Road2d . . . 668
10.133.2 Member Function Documentation . . . 668
10.133.2.1 Load . . . 668
10.134 gazebo::math::RotationSpline Class Reference . . . 668
10.134.1 Detailed Description . . . 669
10.134.2 Constructor & Destructor Documentation . . . 669
10.134.2.1 RotationSpline . . . 669
10.134.2.2 ~RotationSpline . . . 669
10.134.3 Member Function Documentation . . . 669
10.134.3.1 AddPoint . . . 669
10.134.3.2 Clear . . . 670
10.134.3.3 GetNumPoints . . . 670
10.134.3.4 GetPoint . . . 670
10.134.3.5 Interpolate . . . 670
10.134.3.6 Interpolate . . . 671
10.134.3.7 RecalcTangents . . . 671
10.134.3.8 SetAutoCalculate . . . 671
10.134.3.9 UpdatePoint . . . 671
10.134.4 Member Data Documentation . . . 672
10.134.4.1 autoCalc . . . 672
10.134.4.2 points . . . 672
10.134.4.3 tangents . . . 672
10.135 gazebo::rendering::RTShaderSystem Class Reference . . . 672
10.135.1 Detailed Description . . . 673
10.135.2 Member Enumeration Documentation . . . 674
10.135.2.1 LightingModel . . . 674
10.135.3 Member Function Documentation . . . 674
10.135.3.1 AddScene . . . 674
10.135.3.2 ApplyShadows . . . 674
10.135.3.3 AttachEntity . . . 674
10.135.3.4 AttachViewport . . . 674
10.135.3.5 Clear . . . 675
10.135.3.6 DetachEntity . . . 675

10.135.3.7 DetachViewport . . . 675
10.135.3.8 Fini . . . 675
10.135.3.9 GenerateShaders . . . 675
10.135.3.10 GetPSSMShadowCameraSetup . . . 675
10.135.3.11 Init . . . 675
10.135.3.12 RemoveScene . . . 676
10.135.3.13 RemoveShadows . . . 676
10.135.3.14 SetPerPixelLighting . . . 676
10.135.3.15 UpdateShaders . . . 676
10.136 gazebo::rendering::Scene Class Reference . . . 676
10.136.1 Detailed Description . . . 680
10.136.2 Constructor & Destructor Documentation . . . 680
10.136.2.1 Scene . . . 680
10.136.2.2 ~Scene . . . 680
10.136.3 Member Function Documentation . . . 680
10.136.3.1 AddVisual . . . 680
10.136.3.2 Clear . . . 680
10.136.3.3 CloneVisual . . . 680
10.136.3.4 CreateCamera . . . 681
10.136.3.5 CreateDepthCamera . . . 681
10.136.3.6 CreateGrid . . . 681
10.136.3.7 CreateUserCamera . . . 681
10.136.3.8 DrawLine . . . 682
10.136.3.9 GetAmbientColor . . . 682
10.136.3.10 GetBackgroundColor . . . 682
10.136.3.11 GetCamera . . . 682
10.136.3.12 GetCamera . . . 683
10.136.3.13 GetCameraCount . . . 683
10.136.3.14 GetFirstContact . . . 683
10.136.3.15 GetGrid . . . 683
10.136.3.16 GetGridCount . . . 684
10.136.3.17 GetHeightBelowPoint . . . 684
10.136.3.18 GetHeightmap . . . 684
10.136.3.19 GetId . . . 684
10.136.3.20 GetIdString . . . 684
10.136.3.21 GetInitialized . . . 684
10.136.3.22 GetLight . . . 685

10.136.3.23 GetLight . . . 685
10.136.3.24 GetLightCount . . . 685
10.136.3.25 GetManager . . . 685
10.136.3.26 GetModelVisualAt . . . 685
10.136.3.27 GetName . . . 686
10.136.3.28 GetSelectedVisual . . . 686
10.136.3.29 GetShadowsEnabled . . . 686
10.136.3.30 GetUserCamera . . . 686
10.136.3.31 GetUserCameraCount . . . 686
10.136.3.32 GetVisual . . . 687
10.136.3.33 GetVisualAt . . . 687
10.136.3.34 GetVisualAt . . . 687
10.136.3.35 GetVisualBelow . . . 687
10.136.3.36 GetVisualsBelowPoint . . . 687
10.136.3.37 GetWorldVisual . . . 688
10.136.3.38 Init . . . 688
10.136.3.39 Load . . . 688
10.136.3.40 Load . . . 688
10.136.3.41 PreRender . . . 688
10.136.3.42 PrintSceneGraph . . . 688
10.136.3.43 RemoveVisual . . . 688
10.136.3.44 SelectVisual . . . 689
10.136.3.45 SetAmbientColor . . . 689
10.136.3.46 SetBackgroundColor . . . 689
10.136.3.47 SetFog . . . 689
10.136.3.48 SetGrid . . . 689
10.136.3.49 SetShadowsEnabled . . . 690
10.136.3.50 SetTransparent . . . 690
10.136.3.51 SetVisible . . . 690
10.136.3.52 SetWireframe . . . 690
10.136.3.53 ShowCollisions . . . 690
10.136.3.54 ShowCOMs . . . 690
10.136.3.55 ShowContacts . . . 691
10.136.3.56 ShowJoints . . . 691
10.136.3.57 SnapVisualToNearestBelow . . . 691
10.136.3.58 StripSceneName . . . 691
10.136.4 Member Data Documentation . . . 691

10.136.4.1 skyx . . . 691
10.137 gazebo::physics::ScrewJoint< T > Class Template Reference . . . 691
10.137.1 Detailed Description . . . 692
10.137.2 Constructor & Destructor Documentation . . . 693
10.137.2.1 ScrewJoint . . . 693
10.137.2.2 ~ScrewJoint . . . 693
10.137.3 Member Function Documentation . . . 693
10.137.3.1 GetAnchor . . . 693
10.137.3.2 GetAngleCount . . . 693
10.137.3.3 GetThreadPitch . . . 693
10.137.3.4 Load . . . 694
10.137.3.5 SetAnchor . . . 694
10.137.3.6 SetThreadPitch . . . 694
10.137.4 Member Data Documentation . . . 694
10.137.4.1 fakeAnchor . . . 694
10.137.4.2 threadPitch . . . 694
10.138 sdf::SDF Class Reference . . . 695
10.138.1 Detailed Description . . . 695
10.138.2 Constructor & Destructor Documentation . . . 695
10.138.2.1 SDF . . . 695
10.138.3 Member Function Documentation . . . 695
10.138.3.1 PrintDescription . . . 695
10.138.3.2 PrintDoc . . . 695
10.138.3.3 PrintValues . . . 695
10.138.3.4 PrintWiki . . . 695
10.138.3.5 SetFromString . . . 695
10.138.3.6 ToString . . . 696
10.138.3.7 Write . . . 696
10.138.4 Member Data Documentation . . . 696
10.138.4.1 root . . . 696
10.138.4.2 version . . . 696
10.139 gazebo::rendering::SelectionObj Class Reference . . . 696
10.139.1 Detailed Description . . . 696
10.139.2 Constructor & Destructor Documentation . . . 697
10.139.2.1 SelectionObj . . . 697
10.139.2.2 ~SelectionObj . . . 697
10.139.3 Member Function Documentation . . . 697

10.139.3.1Attach . . . 697
10.139.3.2Clear . . . 697
10.139.3.3GetVisualName . . . 697
10.139.3.4Init . . . 697
10.139.3.5IsActive . . . 697
10.139.3.6SetActive . . . 698
10.139.3.7SetHighlight . . . 698
10.140Gazebo::sensors::Sensor Class Reference . . . 698
10.140.1Detailed Description . . . 700
10.140.2Constructor & Destructor Documentation . . . 701
10.140.2.1Sensor . . . 701
10.140.2.2~Sensor . . . 701
10.140.3Member Function Documentation . . . 701
10.140.3.1ConnectUpdated . . . 701
10.140.3.2DisconnectUpdated . . . 701
10.140.3.3FillMsg . . . 702
10.140.3.4Fini . . . 702
10.140.3.5GetCategory . . . 702
10.140.3.6GetLastMeasurementTime . . . 702
10.140.3.7GetLastUpdateTime . . . 702
10.140.3.8GetName . . . 702
10.140.3.9GetParentName . . . 703
10.140.3.10GetPose . . . 703
10.140.3.11GetScopedName . . . 703
10.140.3.12GetTopic . . . 703
10.140.3.13GetType . . . 703
10.140.3.14GetUpdateRate . . . 703
10.140.3.15GetVisualize . . . 704
10.140.3.16GetWorldName . . . 704
10.140.3.17Init . . . 704
10.140.3.18IsActive . . . 704
10.140.3.19Load . . . 704
10.140.3.20Load . . . 705
10.140.3.21ResetLastUpdateTime . . . 705
10.140.3.22SetActive . . . 705
10.140.3.23SetParent . . . 705
10.140.3.24SetUpdateRate . . . 705

10.140.3.25 Update . . . 705
10.140.3.26 UpdateImpl . . . 706
10.140.4 Member Data Documentation . . . 706
10.140.4.1 active . . . 706
10.140.4.2 connections . . . 706
10.140.4.3 lastMeasurementTime . . . 706
10.140.4.4 lastUpdateTime . . . 706
10.140.4.5 node . . . 706
10.140.4.6 parentName . . . 706
10.140.4.7 plugins . . . 707
10.140.4.8 pose . . . 707
10.140.4.9 poseSub . . . 707
10.140.4.10 sdf . . . 707
10.140.4.11 updatePeriod . . . 707
10.140.4.12 world . . . 707
10.141 SensorFactor Class Reference . . . 707
10.141.1 Detailed Description . . . 707
10.142 gazebo::sensors::SensorFactory Class Reference . . . 707
10.142.1 Member Function Documentation . . . 708
10.142.1.1 GetSensorTypes . . . 708
10.142.1.2 NewSensor . . . 708
10.142.1.3 RegisterAll . . . 708
10.142.1.4 RegisterSensor . . . 709
10.143 gazebo::sensors::SensorManager Class Reference . . . 709
10.143.1 Detailed Description . . . 710
10.143.2 Member Function Documentation . . . 710
10.143.2.1 CreateSensor . . . 710
10.143.2.2 Fini . . . 710
10.143.2.3 GetSensor . . . 711
10.143.2.4 GetSensors . . . 711
10.143.2.5 GetSensorTypes . . . 711
10.143.2.6 Init . . . 711
10.143.2.7 RemoveSensor . . . 711
10.143.2.8 RemoveSensors . . . 711
10.143.2.9 ResetLastUpdateTimes . . . 711
10.143.2.10 Run . . . 712
10.143.2.11 RunThreads . . . 712

10.143.2.12 SensorsInitialized . . . 712
10.143.2.13 Stop . . . 712
10.143.2.14 Update . . . 712
10.144 gazebo::SensorPlugin Class Reference . . . 712
10.144.1 Detailed Description . . . 713
10.144.2 Constructor & Destructor Documentation . . . 713
10.144.2.1 SensorPlugin . . . 713
10.144.2.2 ~SensorPlugin . . . 713
10.144.3 Member Function Documentation . . . 714
10.144.3.1 Init . . . 714
10.144.3.2 Load . . . 714
10.144.3.3 Reset . . . 714
10.145 gazebo::Server Class Reference . . . 714
10.145.1 Constructor & Destructor Documentation . . . 715
10.145.1.1 Server . . . 715
10.145.1.2 ~Server . . . 715
10.145.2 Member Function Documentation . . . 715
10.145.2.1 Fini . . . 715
10.145.2.2 GetInitialized . . . 715
10.145.2.3 Init . . . 715
10.145.2.4 LoadFile . . . 715
10.145.2.5 LoadString . . . 715
10.145.2.6 ParseArgs . . . 715
10.145.2.7 PrintUsage . . . 715
10.145.2.8 Run . . . 715
10.145.2.9 SetParams . . . 715
10.145.2.10 Stop . . . 715
10.145.3 Member Data Documentation . . . 715
10.145.3.1 systemPluginsArgc . . . 715
10.145.3.2 systemPluginsArgv . . . 715
10.146 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference . . . 715
10.146.1 Detailed Description . . . 716
10.146.2 Member Function Documentation . . . 717
10.146.2.1 defaultVpParams . . . 717
10.146.2.2 generateFragmentProgram . . . 717
10.146.2.3 generateVertexProgram . . . 717
10.146.2.4 generateVertexProgramSource . . . 717

10.146.2.5 generateVpDynamicShadows . . . 717
10.146.2.6 generateVpDynamicShadowsParams . . . 717
10.146.2.7 generateVpFooter . . . 717
10.146.2.8 generateVpHeader . . . 717
10.147 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference . . . 717
10.147.1 Detailed Description . . . 719
10.147.2 Member Function Documentation . . . 719
10.147.2.1 defaultVpParams . . . 719
10.147.2.2 generateFpDynamicShadows . . . 719
10.147.2.3 generateFpDynamicShadowsHelpers . . . 719
10.147.2.4 generateFpDynamicShadowsParams . . . 719
10.147.2.5 generateFpFooter . . . 719
10.147.2.6 generateFpHeader . . . 719
10.147.2.7 generateFpLayer . . . 719
10.147.2.8 generateFragmentProgram . . . 719
10.147.2.9 generateFragmentProgramSource . . . 720
10.147.2.10 generateVertexProgram . . . 720
10.147.2.11 generateVertexProgramSource . . . 720
10.147.2.12 generateVpDynamicShadows . . . 720
10.147.2.13 generateVpDynamicShadowsParams . . . 720
10.147.2.14 generateVpFooter . . . 720
10.147.2.15 generateVpHeader . . . 720
10.147.2.16 updateParams . . . 720
10.147.2.17 updateVpParams . . . 720
10.148 gazebo::physics::Shape Class Reference . . . 720
10.148.1 Detailed Description . . . 721
10.148.2 Constructor & Destructor Documentation . . . 721
10.148.2.1 Shape . . . 722
10.148.2.2 ~Shape . . . 722
10.148.3 Member Function Documentation . . . 722
10.148.3.1 FillMsg . . . 722
10.148.3.2 Init . . . 722
10.148.3.3 ProcessMsg . . . 722
10.148.4 Member Data Documentation . . . 723
10.148.4.1 collisionParent . . . 723
10.149 gazebo::sensors::SimTimeEvent Class Reference . . . 723
10.149.1 Detailed Description . . . 723

10.149.2 Member Data Documentation . . . 723
10.149.2.1 condition . . . 723
10.149.2.2 time . . . 723
10.150 gazebo::sensors::SimTimeEventHandler Class Reference . . . 723
10.150.1 Detailed Description . . . 724
10.150.2 Constructor & Destructor Documentation . . . 724
10.150.2.1 SimTimeEventHandler . . . 724
10.150.2.2 ~SimTimeEventHandler . . . 724
10.150.3 Member Function Documentation . . . 724
10.150.3.1 AddRelativeEvent . . . 725
10.151 SingletonT< T > Class Template Reference . . . 725
10.151.1 Detailed Description . . . 727
10.151.2 Constructor & Destructor Documentation . . . 727
10.151.2.1 SingletonT . . . 727
10.151.2.2 ~SingletonT . . . 727
10.151.3 Member Function Documentation . . . 727
10.151.3.1 Instance . . . 727
10.152 gazebo::common::Skeleton Class Reference . . . 727
10.152.1 Detailed Description . . . 729
10.152.2 Constructor & Destructor Documentation . . . 729
10.152.2.1 Skeleton . . . 729
10.152.2.2 Skeleton . . . 729
10.152.2.3 ~Skeleton . . . 729
10.152.3 Member Function Documentation . . . 729
10.152.3.1 AddAnimation . . . 729
10.152.3.2 AddVertNodeWeight . . . 730
10.152.3.3 BuildNodeMap . . . 730
10.152.3.4 GetAnimation . . . 730
10.152.3.5 GetBindShapeTransform . . . 730
10.152.3.6 GetNodeByHandle . . . 730
10.152.3.7 GetNodeById . . . 730
10.152.3.8 GetNodeByName . . . 731
10.152.3.9 GetNodes . . . 731
10.152.3.10 GetNumAnimations . . . 731
10.152.3.11 GetNumJoints . . . 731
10.152.3.12 GetNumNodes . . . 731
10.152.3.13 GetNumVertNodeWeights . . . 732

10.152.3.14GetRootNode . . . 732
10.152.3.15GetVertNodeWeight . . . 732
10.152.3.16PrintTransforms . . . 732
10.152.3.17Scale . . . 732
10.152.3.18SetBindShapeTransform . . . 732
10.152.3.19SetNumVertAttached . . . 733
10.152.3.20SetRootNode . . . 733
10.152.4Member Data Documentation . . . 733
10.152.4.1anims . . . 733
10.152.4.2bindShapeTransform . . . 733
10.152.4.3nodes . . . 733
10.152.4.4rawNW . . . 733
10.152.4.5root . . . 733
10.153gazebo::common::SkeletonAnimation Class Reference . . . 734
10.153.1Detailed Description . . . 735
10.153.2Constructor & Destructor Documentation . . . 735
10.153.2.1SkeletonAnimation . . . 735
10.153.2.2~SkeletonAnimation . . . 735
10.153.3Member Function Documentation . . . 735
10.153.3.1AddKeyFrame . . . 735
10.153.3.2AddKeyFrame . . . 735
10.153.3.3GetLength . . . 735
10.153.3.4GetName . . . 736
10.153.3.5GetNodeCount . . . 736
10.153.3.6GetNodePoseAt . . . 736
10.153.3.7GetPoseAt . . . 736
10.153.3.8GetPoseAtX . . . 737
10.153.3.9HasNode . . . 737
10.153.3.10Scale . . . 737
10.153.3.11SetName . . . 737
10.153.4Member Data Documentation . . . 737
10.153.4.1animations . . . 737
10.153.4.2length . . . 738
10.153.4.3name . . . 738
10.154gazebo::common::SkeletonNode Class Reference . . . 738
10.154.1Detailed Description . . . 740
10.154.2Member Enumeration Documentation . . . 740

10.154.2.1 SkeletonNodeType . . . 740
10.154.3 Constructor & Destructor Documentation . . . 740
10.154.3.1 SkeletonNode . . . 740
10.154.3.2 SkeletonNode . . . 741
10.154.3.3 ~SkeletonNode . . . 741
10.154.4 Member Function Documentation . . . 741
10.154.4.1 AddChild . . . 741
10.154.4.2 AddRawTransform . . . 741
10.154.4.3 GetChild . . . 741
10.154.4.4 GetChildById . . . 741
10.154.4.5 GetChildByName . . . 742
10.154.4.6 GetChildCount . . . 742
10.154.4.7 GetHandle . . . 742
10.154.4.8 GetId . . . 742
10.154.4.9 GetInverseBindTransform . . . 742
10.154.4.10 GetModelTransform . . . 743
10.154.4.11 GetName . . . 743
10.154.4.12 GetNumRawTrans . . . 743
10.154.4.13 GetParent . . . 743
10.154.4.14 GetRawTransform . . . 743
10.154.4.15 GetRawTransforms . . . 743
10.154.4.16 GetTransform . . . 744
10.154.4.17 GetTransforms . . . 744
10.154.4.18 IsJoint . . . 744
10.154.4.19 IsRootNode . . . 744
10.154.4.20 Reset . . . 744
10.154.4.21 SetHandle . . . 744
10.154.4.22 SetId . . . 745
10.154.4.23 SetInitialTransform . . . 745
10.154.4.24 SetInverseBindTransform . . . 745
10.154.4.25 SetModelTransform . . . 745
10.154.4.26 SetName . . . 745
10.154.4.27 SetParent . . . 745
10.154.4.28 SetTransform . . . 746
10.154.4.29 SetType . . . 746
10.154.4.30 UpdateChildrenTransforms . . . 746
10.154.5 Member Data Documentation . . . 746

10.154.5.1 children . . . 746
10.154.5.2 handle . . . 746
10.154.5.3 id . . . 746
10.154.5.4 initialTransform . . . 746
10.154.5.5 invBindTransform . . . 746
10.154.5.6 modelTransform . . . 746
10.154.5.7 name . . . 747
10.154.5.8 parent . . . 747
10.154.5.9 rawTransforms . . . 747
10.154.5.10 transform . . . 747
10.154.5.11 type . . . 747
10.155 gazebo::physics::SliderJoint< T > Class Template Reference . . . 747
10.155.1 Detailed Description . . . 748
10.155.2 Constructor & Destructor Documentation . . . 748
10.155.2.1 SliderJoint . . . 748
10.155.2.2 ~SliderJoint . . . 748
10.155.3 Member Function Documentation . . . 748
10.155.3.1 GetAnchor . . . 748
10.155.3.2 GetAngleCount . . . 749
10.155.3.3 Load . . . 749
10.155.3.4 SetAnchor . . . 749
10.155.4 Member Data Documentation . . . 749
10.155.4.1 fakeAnchor . . . 749
10.156 gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference . . . 749
10.156.1 Detailed Description . . . 750
10.156.2 Constructor & Destructor Documentation . . . 751
10.156.2.1 SM2Profile . . . 751
10.156.2.2 ~SM2Profile . . . 751
10.156.3 Member Function Documentation . . . 751
10.156.3.1 addTechnique . . . 751
10.156.3.2 generate . . . 751
10.156.3.3 generateForCompositeMap . . . 751
10.156.3.4 UpdateParams . . . 751
10.156.3.5 UpdateParamsForCompositeMap . . . 751
10.157 gazebo::physics::SphereShape Class Reference . . . 751
10.157.1 Detailed Description . . . 752
10.157.2 Constructor & Destructor Documentation . . . 753

10.157.2.1 SphereShape . . . 753
10.157.2.2 ~SphereShape . . . 753
10.157.3 Member Function Documentation . . . 753
10.157.3.1 FillMsg . . . 753
10.157.3.2 GetRadius . . . 753
10.157.3.3 Init . . . 753
10.157.3.4 ProcessMsg . . . 753
10.157.3.5 SetRadius . . . 754
10.158 gazebo::math::Spline Class Reference . . . 754
10.158.1 Detailed Description . . . 755
10.158.2 Constructor & Destructor Documentation . . . 755
10.158.2.1 Spline . . . 755
10.158.2.2 ~Spline . . . 755
10.158.3 Member Function Documentation . . . 755
10.158.3.1 AddPoint . . . 755
10.158.3.2 Clear . . . 755
10.158.3.3 GetPoint . . . 755
10.158.3.4 GetPointCount . . . 756
10.158.3.5 GetTangent . . . 756
10.158.3.6 GetTension . . . 756
10.158.3.7 Interpolate . . . 756
10.158.3.8 Interpolate . . . 756
10.158.3.9 RecalcTangents . . . 757
10.158.3.10 SetAutoCalculate . . . 757
10.158.3.11 SetTension . . . 757
10.158.3.12 UpdatePoint . . . 757
10.158.4 Member Data Documentation . . . 758
10.158.4.1 autoCalc . . . 758
10.158.4.2 coeffs . . . 758
10.158.4.3 points . . . 758
10.158.4.4 tangents . . . 758
10.158.4.5 tension . . . 758
10.159 gazebo::physics::State Class Reference . . . 758
10.159.1 Detailed Description . . . 760
10.159.2 Constructor & Destructor Documentation . . . 760
10.159.2.1 State . . . 760
10.159.2.2 State . . . 760

10.159.2.3 ~State . . . 760
10.159.3 Member Function Documentation . . . 760
10.159.3.1 GetName . . . 760
10.159.3.2 GetRealTime . . . 761
10.159.3.3 GetSimTime . . . 761
10.159.3.4 GetWallTime . . . 761
10.159.3.5 Load . . . 761
10.159.3.6 operator- . . . 761
10.159.3.7 operator= . . . 762
10.159.3.8 SetName . . . 762
10.159.4 Member Data Documentation . . . 762
10.159.4.1 name . . . 762
10.159.4.2 realTime . . . 762
10.159.4.3 simTime . . . 762
10.159.4.4 wallTime . . . 762
10.160 gazebo::common::STLLoader Class Reference . . . 762
10.160.1 Detailed Description . . . 763
10.160.2 Constructor & Destructor Documentation . . . 763
10.160.2.1 STLLoader . . . 763
10.160.2.2 ~STLLoader . . . 763
10.160.3 Member Function Documentation . . . 763
10.160.3.1 Load . . . 763
10.161 gazebo::common::SubMesh Class Reference . . . 764
10.161.1 Detailed Description . . . 766
10.161.2 Member Enumeration Documentation . . . 766
10.161.2.1 PrimitiveType . . . 766
10.161.3 Constructor & Destructor Documentation . . . 766
10.161.3.1 SubMesh . . . 766
10.161.3.2 SubMesh . . . 767
10.161.3.3 ~SubMesh . . . 767
10.161.4 Member Function Documentation . . . 767
10.161.4.1 AddIndex . . . 767
10.161.4.2 AddNodeAssignment . . . 767
10.161.4.3 AddNormal . . . 767
10.161.4.4 AddNormal . . . 767
10.161.4.5 AddTexCoord . . . 768
10.161.4.6 AddVertex . . . 768

10.161.4.7 AddVertex . . . 768
10.161.4.8 Center . . . 768
10.161.4.9 CopyNormals . . . 768
10.161.4.10 CopyVertices . . . 768
10.161.4.11 FillArrays . . . 769
10.161.4.12 GenSphericalTexCoord . . . 769
10.161.4.13 GetIndex . . . 769
10.161.4.14 GetIndexCount . . . 769
10.161.4.15 GetMaterialIndex . . . 769
10.161.4.16 GetMax . . . 769
10.161.4.17 GetMaxIndex . . . 769
10.161.4.18 GetMin . . . 770
10.161.4.19 GetName . . . 770
10.161.4.20 GetNodeAssignment . . . 770
10.161.4.21 GetNodeAssignmentsCount . . . 770
10.161.4.22 GetNormal . . . 770
10.161.4.23 GetNormalCount . . . 770
10.161.4.24 GetPrimitiveType . . . 770
10.161.4.25 GetTexCoord . . . 771
10.161.4.26 GetTexCoordCount . . . 771
10.161.4.27 GetVertex . . . 771
10.161.4.28 GetVertexCount . . . 771
10.161.4.29 GetVertexIndex . . . 771
10.161.4.30 HasVertex . . . 771
10.161.4.31 RecalculateNormals . . . 772
10.161.4.32 Scale . . . 772
10.161.4.33 SetIndexCount . . . 772
10.161.4.34 SetMaterialIndex . . . 772
10.161.4.35 SetName . . . 772
10.161.4.36 SetNormal . . . 772
10.161.4.37 SetNormalCount . . . 773
10.161.4.38 SetPrimitiveType . . . 773
10.161.4.39 SetScale . . . 773
10.161.4.40 SetSubMeshCenter . . . 773
10.161.4.41 SetTexCoord . . . 773
10.161.4.42 SetTexCoordCount . . . 774
10.161.4.43 SetVertex . . . 774

10.161.4.43 GetVertexCount . . . 774
10.161.4.44 Translate . . . 774
10.162 gazebo::transport::SubscribeOptions Class Reference . . . 774
10.162.1 Detailed Description . . . 775
10.162.2 Constructor & Destructor Documentation . . . 775
10.162.2.1 SubscribeOptions . . . 775
10.162.3 Member Function Documentation . . . 775
10.162.3.1 GetLatching . . . 775
10.162.3.2 GetMsgType . . . 775
10.162.3.3 GetNode . . . 775
10.162.3.4 GetTopic . . . 776
10.162.3.5 Init . . . 776
10.162.3.6 Init . . . 776
10.163 gazebo::transport::Subscriber Class Reference . . . 776
10.163.1 Detailed Description . . . 777
10.163.2 Constructor & Destructor Documentation . . . 777
10.163.2.1 Subscriber . . . 777
10.163.2.2 ~Subscriber . . . 777
10.163.3 Member Function Documentation . . . 777
10.163.3.1 GetCallbackId . . . 777
10.163.3.2 GetTopic . . . 777
10.163.3.3 SetCallbackId . . . 777
10.163.3.4 Unsubscribe . . . 777
10.164 gazebo::transport::SubscriptionTransport Class Reference . . . 778
10.164.1 Detailed Description . . . 778
10.164.2 Constructor & Destructor Documentation . . . 779
10.164.2.1 SubscriptionTransport . . . 779
10.164.2.2 ~SubscriptionTransport . . . 779
10.164.3 Member Function Documentation . . . 779
10.164.3.1 GetConnection . . . 779
10.164.3.2 HandleData . . . 779
10.164.3.3 HandleMessage . . . 779
10.164.3.4 Init . . . 780
10.164.3.5 IsLocal . . . 780
10.165 gazebo::physics::SurfaceParams Class Reference . . . 780
10.165.1 Detailed Description . . . 781
10.165.2 Constructor & Destructor Documentation . . . 781

10.165.2.1 SurfaceParams . . . 781
10.165.2.2 ~SurfaceParams . . . 781
10.165.3 Member Function Documentation . . . 781
10.165.3.1 FillMsg . . . 781
10.165.3.2 Load . . . 782
10.165.3.3 ProcessMsg . . . 782
10.165.4 Member Data Documentation . . . 782
10.165.4.1 bounce . . . 782
10.165.4.2 bounceThreshold . . . 782
10.165.4.3 cfm . . . 782
10.165.4.4 erp . . . 782
10.165.4.5 fdir1 . . . 782
10.165.4.6 kd . . . 783
10.165.4.7 kp . . . 783
10.165.4.8 maxVel . . . 783
10.165.4.9 minDepth . . . 783
10.165.4.10 mu1 . . . 783
10.165.4.11 mu2 . . . 784
10.165.4.12 slip1 . . . 784
10.165.4.13 slip2 . . . 784
10.166 gazebo::common::SystemPaths Class Reference . . . 784
10.166.1 Detailed Description . . . 786
10.166.2 Member Function Documentation . . . 786
10.166.2.1 AddGazeboPaths . . . 786
10.166.2.2 AddModelPaths . . . 786
10.166.2.3 AddOgrePaths . . . 787
10.166.2.4 AddPluginPaths . . . 787
10.166.2.5 AddSearchPathSuffix . . . 787
10.166.2.6 ClearGazeboPaths . . . 787
10.166.2.7 ClearModelPaths . . . 787
10.166.2.8 ClearOgrePaths . . . 787
10.166.2.9 ClearPluginPaths . . . 787
10.166.2.10 FindFile . . . 787
10.166.2.11 FindFileURI . . . 788
10.166.2.12 GetGazeboPaths . . . 788
10.166.2.13 GetLogPath . . . 788
10.166.2.14 GetModelPaths . . . 788

10.166.2.15 GetOgrePaths . . . 788
10.166.2.16 GetPluginPaths . . . 789
10.166.2.17 GetWorldPathExtension . . . 789
10.166.3 Member Data Documentation . . . 789
10.166.3.1 gazeboPathsFromEnv . . . 789
10.166.3.2 modelPathsFromEnv . . . 789
10.166.3.3 ogrePathsFromEnv . . . 789
10.166.3.4 pluginPathsFromEnv . . . 789
10.167 gazebo::SystemPlugin Class Reference . . . 789
10.167.1 Detailed Description . . . 790
10.167.2 Constructor & Destructor Documentation . . . 790
10.167.2.1 SystemPlugin . . . 790
10.167.2.2 ~SystemPlugin . . . 790
10.167.3 Member Function Documentation . . . 791
10.167.3.1 Init . . . 791
10.167.3.2 Load . . . 791
10.167.3.3 Reset . . . 791
10.168 gazebo::common::Time Class Reference . . . 791
10.168.1 Detailed Description . . . 795
10.168.2 Constructor & Destructor Documentation . . . 795
10.168.2.1 Time . . . 795
10.168.2.2 Time . . . 795
10.168.2.3 Time . . . 795
10.168.2.4 Time . . . 795
10.168.2.5 Time . . . 796
10.168.2.6 Time . . . 796
10.168.2.7 ~Time . . . 796
10.168.3 Member Function Documentation . . . 796
10.168.3.1 Double . . . 796
10.168.3.2 Float . . . 796
10.168.3.3 GetWallTime . . . 796
10.168.3.4 GetWallTimeAsISOString . . . 797
10.168.3.5 MicToNano . . . 797
10.168.3.6 MilToNano . . . 797
10.168.3.7 MSleep . . . 797
10.168.3.8 NSleep . . . 797
10.168.3.9 NSleep . . . 798

10.168.3.10operator!= . . . 798
10.168.3.11operator!= . . . 798
10.168.3.12operator!= . . . 798
10.168.3.13operator!= . . . 799
10.168.3.14operator∗ . . . 799
10.168.3.15operator∗ . . . 799
10.168.3.16operator∗ . . . 799
10.168.3.17operator∗= . . . 800
10.168.3.18operator∗= . . . 800
10.168.3.19operator∗= . . . 800
10.168.3.20operator+ . . . 800
10.168.3.21operator+ . . . 801
10.168.3.22operator+ . . . 801
10.168.3.23operator+= . . . 801
10.168.3.24operator+= . . . 801
10.168.3.25operator+= . . . 802
10.168.3.26operator- . . . 802
10.168.3.27operator- . . . 802
10.168.3.28operator- . . . 802
10.168.3.29operator-= . . . 803
10.168.3.30operator-= . . . 803
10.168.3.31operator-= . . . 803
10.168.3.32operator/ . . . 803
10.168.3.33operator/ . . . 804
10.168.3.34operator/ . . . 804
10.168.3.35operator/= . . . 804
10.168.3.36operator/= . . . 804
10.168.3.37operator/= . . . 805
10.168.3.38operator< . . . 805
10.168.3.39operator< . . . 805
10.168.3.40operator< . . . 805
10.168.3.41operator< . . . 806
10.168.3.42operator<= . . . 806
10.168.3.43operator<= . . . 806
10.168.3.44operator<= . . . 806
10.168.3.45operator<= . . . 807
10.168.3.46operator= . . . 807

10.168.3.47 operator= . . . 807
10.168.3.48 operator= . . . 807
10.168.3.49 operator== . . . 808
10.168.3.50 operator== . . . 808
10.168.3.51 operator== . . . 808
10.168.3.52 operator== . . . 808
10.168.3.53 operator> . . . 809
10.168.3.54 operator> . . . 809
10.168.3.55 operator> . . . 809
10.168.3.56 operator> . . . 809
10.168.3.57 operator>= . . . 810
10.168.3.58 operator>= . . . 810
10.168.3.59 operator>= . . . 810
10.168.3.60 operator>= . . . 810
10.168.3.61 SecToNano . . . 811
10.168.3.62 Set . . . 811
10.168.3.63 Set . . . 811
10.168.3.64 SetToWallTime . . . 811
10.168.3.65 Sleep . . . 811
10.168.4 Friends And Related Function Documentation . . . 812
10.168.4.1 operator<< . . . 812
10.168.4.2 operator>> . . . 812
10.168.5 Member Data Documentation . . . 812
10.168.5.1 nsec . . . 812
10.168.5.2 sec . . . 812
10.168.5.3 Zero . . . 812
10.169 gazebo::common::Timer Class Reference . . . 813
10.169.1 Detailed Description . . . 813
10.169.2 Constructor & Destructor Documentation . . . 814
10.169.2.1 Timer . . . 814
10.169.2.2 ~Timer . . . 814
10.169.3 Member Function Documentation . . . 814
10.169.3.1 GetElapsed . . . 814
10.169.3.2 GetRunning . . . 814
10.169.3.3 Start . . . 814
10.169.3.4 Stop . . . 814
10.169.4 Friends And Related Function Documentation . . . 814

10.169.4.1 operator<< . . . 814
10.170 gazebo::transport::TopicManager Class Reference . . . 815
10.170.1 Detailed Description . . . 816
10.170.2 Member Typedef Documentation . . . 816
10.170.2.1 SubNodeMap . . . 816
10.170.3 Member Function Documentation . . . 817
10.170.3.1 AddNode . . . 817
10.170.3.2 Advertise . . . 817
10.170.3.3 ClearBuffers . . . 817
10.170.3.4 ConnectPubToSub . . . 817
10.170.3.5 ConnectSubscribers . . . 817
10.170.3.6 ConnectSubToPub . . . 818
10.170.3.7 DisconnectPubFromSub . . . 818
10.170.3.8 DisconnectSubFromPub . . . 818
10.170.3.9 FindPublication . . . 818
10.170.3.10 Fini . . . 818
10.170.3.11 GetAdvertisedTopics . . . 819
10.170.3.12 GetTopicNamespaces . . . 819
10.170.3.13 Init . . . 819
10.170.3.14 IsAdvertised . . . 819
10.170.3.15 PauseIncoming . . . 819
10.170.3.16 ProcessNodes . . . 819
10.170.3.17 Publish . . . 820
10.170.3.18 RegisterTopicNamespace . . . 820
10.170.3.19 RemoveNode . . . 820
10.170.3.20 Subscribe . . . 820
10.170.3.21 Unadvertise . . . 820
10.170.3.22 Unsubscribe . . . 821
10.170.3.23 UpdatePublications . . . 821
10.171 gazebo::physics::TrajectoryInfo Struct Reference . . . 821
10.171.1 Member Data Documentation . . . 822
10.171.1.1 duration . . . 822
10.171.1.2 endTime . . . 822
10.171.1.3 id . . . 822
10.171.1.4 startTime . . . 822
10.171.1.5 translated . . . 822
10.171.1.6 type . . . 822

10.172 gazebo::physics::TrimeshShape Class Reference . . . 822
10.172.1 Detailed Description . . . 823
10.172.2 Constructor & Destructor Documentation . . . 823
10.172.2.1 TrimeshShape . . . 823
10.172.2.2 ~TrimeshShape . . . 824
10.172.3 Member Function Documentation . . . 824
10.172.3.1 FillMsg . . . 824
10.172.3.2 GetFilename . . . 824
10.172.3.3 GetMeshURI . . . 824
10.172.3.4 GetSize . . . 824
10.172.3.5 Init . . . 824
10.172.3.6 ProcessMsg . . . 825
10.172.3.7 SetFilename . . . 825
10.172.3.8 SetMesh . . . 825
10.172.3.9 SetScale . . . 825
10.172.3.10 Update . . . 825
10.172.4 Member Data Documentation . . . 825
10.172.4.1 mesh . . . 825
10.172.4.2 submesh . . . 826
10.173 gazebo::physics::UniversalJoint< T > Class Template Reference . . . 826
10.173.1 Detailed Description . . . 826
10.173.2 Constructor & Destructor Documentation . . . 827
10.173.2.1 UniversalJoint . . . 827
10.173.2.2 ~UniversalJoint . . . 827
10.173.3 Member Function Documentation . . . 827
10.173.3.1 GetAngleCount . . . 827
10.173.3.2 Load . . . 827
10.174 gazebo::common::UpdateInfo Class Reference . . . 827
10.174.1 Detailed Description . . . 828
10.174.2 Member Data Documentation . . . 828
10.174.2.1 realTime . . . 828
10.174.2.2 simTime . . . 828
10.174.2.3 worldName . . . 828
10.175 urdf2gazebo::URDF2Gazebo Class Reference . . . 828
10.175.1 Constructor & Destructor Documentation . . . 828
10.175.1.1 URDF2Gazebo . . . 828
10.175.1.2 ~URDF2Gazebo . . . 829

10.175.2 Member Function Documentation . . . 829
10.175.2.1 InitModelDoc . . . 829
10.175.2.2 InitModelFile . . . 829
10.175.2.3 InitModelString . . . 829
10.176 gazebo::rendering::UserCamera Class Reference . . . 830
10.176.1 Detailed Description . . . 832
10.176.2 Constructor & Destructor Documentation . . . 832
10.176.2.1 UserCamera . . . 832
10.176.2.2 ~UserCamera . . . 832
10.176.3 Member Function Documentation . . . 832
10.176.3.1 AnimationComplete . . . 832
10.176.3.2 AttachToVisualImpl . . . 832
10.176.3.3 EnableViewController . . . 833
10.176.3.4 Fini . . . 833
10.176.3.5 GetAvgFPS . . . 833
10.176.3.6 GetGUIOverlay . . . 833
10.176.3.7 GetImageHeight . . . 833
10.176.3.8 GetImageWidth . . . 834
10.176.3.9 GetTriangleCount . . . 834
10.176.3.10 GetViewControllerTypeString . . . 834
10.176.3.11 GetVisual . . . 834
10.176.3.12 GetVisual . . . 834
10.176.3.13 HandleKeyPressEvent . . . 835
10.176.3.14 HandleKeyReleaseEvent . . . 835
10.176.3.15 HandleMouseEvent . . . 835
10.176.3.16 Init . . . 835
10.176.3.17 Load . . . 835
10.176.3.18 Load . . . 835
10.176.3.19 MoveToPosition . . . 836
10.176.3.20 MoveToVisual . . . 836
10.176.3.21 MoveToVisual . . . 836
10.176.3.22 PostRender . . . 836
10.176.3.23 Resize . . . 836
10.176.3.24 SetFocalPoint . . . 836
10.176.3.25 SetRenderTarget . . . 837
10.176.3.26 SetViewController . . . 837
10.176.3.27 SetViewController . . . 837

10.176.3.28 SetViewportDimensions . . . 837
10.176.3.29 SetWorldPose . . . 837
10.176.3.30 TrackVisualImpl . . . 838
10.176.3.31 Update . . . 838
10.177 gazebo::math::Vector2d Class Reference . . . 838
10.177.1 Detailed Description . . . 840
10.177.2 Constructor & Destructor Documentation . . . 840
10.177.2.1 Vector2d . . . 840
10.177.2.2 Vector2d . . . 840
10.177.2.3 Vector2d . . . 840
10.177.2.4 ~Vector2d . . . 840
10.177.3 Member Function Documentation . . . 840
10.177.3.1 Cross . . . 840
10.177.3.2 Distance . . . 841
10.177.3.3 IsFinite . . . 841
10.177.3.4 Normalize . . . 841
10.177.3.5 operator!= . . . 841
10.177.3.6 operator* . . . 841
10.177.3.7 operator* . . . 842
10.177.3.8 operator*= . . . 842
10.177.3.9 operator*= . . . 842
10.177.3.10 operator+ . . . 842
10.177.3.11 operator+= . . . 843
10.177.3.12 operator- . . . 843
10.177.3.13 operator-= . . . 843
10.177.3.14 operator/ . . . 843
10.177.3.15 operator/ . . . 844
10.177.3.16 operator/= . . . 844
10.177.3.17 operator/= . . . 844
10.177.3.18 operator= . . . 844
10.177.3.19 operator= . . . 845
10.177.3.20 operator== . . . 845
10.177.3.21 operator[] . . . 845
10.177.3.22 Set . . . 845
10.177.4 Friends And Related Function Documentation . . . 846
10.177.4.1 operator<< . . . 846
10.177.4.2 operator>> . . . 846

10.177.5 Member Data Documentation . . . 846
10.177.5.1 x . . . 846
10.177.5.2 y . . . 846
10.178 gazebo::math::Vector2i Class Reference . . . 846
10.178.1 Detailed Description . . . 848
10.178.2 Constructor & Destructor Documentation . . . 848
10.178.2.1 Vector2i . . . 848
10.178.2.2 Vector2i . . . 848
10.178.2.3 Vector2i . . . 848
10.178.2.4 ~Vector2i . . . 849
10.178.3 Member Function Documentation . . . 849
10.178.3.1 Cross . . . 849
10.178.3.2 Distance . . . 849
10.178.3.3 IsFinite . . . 849
10.178.3.4 Normalize . . . 849
10.178.3.5 operator!= . . . 849
10.178.3.6 operator* . . . 850
10.178.3.7 operator* . . . 850
10.178.3.8 operator*= . . . 850
10.178.3.9 operator*= . . . 851
10.178.3.10 operator+ . . . 851
10.178.3.11 operator+= . . . 851
10.178.3.12 operator- . . . 851
10.178.3.13 operator-= . . . 852
10.178.3.14 operator/ . . . 852
10.178.3.15 operator/ . . . 852
10.178.3.16 operator/= . . . 853
10.178.3.17 operator/= . . . 853
10.178.3.18 operator= . . . 853
10.178.3.19 operator= . . . 853
10.178.3.20 operator== . . . 854
10.178.3.21 operator[] . . . 854
10.178.3.22 Set . . . 854
10.178.4 Friends And Related Function Documentation . . . 854
10.178.4.1 operator<< . . . 854
10.178.4.2 operator>> . . . 855
10.178.5 Member Data Documentation . . . 855

10.178.5.1 x . . . 855
10.178.5.2 y . . . 855
10.179 gazebo::math::Vector3 Class Reference . . . 855
10.179.1 Detailed Description . . . 858
10.179.2 Constructor & Destructor Documentation . . . 858
10.179.2.1 Vector3 . . . 858
10.179.2.2 Vector3 . . . 858
10.179.2.3 Vector3 . . . 859
10.179.2.4 ~Vector3 . . . 859
10.179.3 Member Function Documentation . . . 859
10.179.3.1 Correct . . . 859
10.179.3.2 Cross . . . 859
10.179.3.3 Distance . . . 859
10.179.3.4 Distance . . . 859
10.179.3.5 Dot . . . 860
10.179.3.6 Equal . . . 860
10.179.3.7 GetAbs . . . 860
10.179.3.8 GetDistToLine . . . 860
10.179.3.9 GetLength . . . 861
10.179.3.10 GetMax . . . 861
10.179.3.11 GetMin . . . 861
10.179.3.12 GetNormal . . . 861
10.179.3.13 GetPerpendicular . . . 861
10.179.3.14 GetRounded . . . 861
10.179.3.15 GetSquaredLength . . . 862
10.179.3.16 GetSum . . . 862
10.179.3.17 IsFinite . . . 862
10.179.3.18 Normalize . . . 862
10.179.3.19 operator!= . . . 862
10.179.3.20 operator* . . . 862
10.179.3.21 operator* . . . 863
10.179.3.22 operator*= . . . 863
10.179.3.23 operator*= . . . 863
10.179.3.24 operator+ . . . 863
10.179.3.25 operator+= . . . 864
10.179.3.26 operator- . . . 864
10.179.3.27 operator- . . . 864

10.179.3.28 operator-= . . . 864
10.179.3.29 operator/ . . . 865
10.179.3.30 operator/ . . . 865
10.179.3.31 operator/= . . . 865
10.179.3.32 operator/= . . . 865
10.179.3.33 operator= . . . 866
10.179.3.34 operator= . . . 866
10.179.3.35 operator== . . . 866
10.179.3.36 operator[] . . . 866
10.179.3.37 Round . . . 866
10.179.3.38 Round . . . 867
10.179.3.39 Set . . . 867
10.179.3.40 SetToMax . . . 867
10.179.3.41 SetToMin . . . 867
10.179.4 Friends And Related Function Documentation . . . 867
10.179.4.1 operator* . . . 867
10.179.4.2 operator<< . . . 868
10.179.4.3 operator>> . . . 868
10.179.5 Member Data Documentation . . . 868
10.179.5.1 One . . . 868
10.179.5.2 UnitX . . . 868
10.179.5.3 UnitY . . . 868
10.179.5.4 UnitZ . . . 868
10.179.5.5 x . . . 868
10.179.5.6 y . . . 869
10.179.5.7 z . . . 869
10.179.5.8 Zero . . . 869
10.180 gazebo::math::Vector4 Class Reference . . . 869
10.180.1 Detailed Description . . . 871
10.180.2 Constructor & Destructor Documentation . . . 871
10.180.2.1 Vector4 . . . 871
10.180.2.2 Vector4 . . . 871
10.180.2.3 Vector4 . . . 871
10.180.2.4 ~Vector4 . . . 871
10.180.3 Member Function Documentation . . . 871
10.180.3.1 Distance . . . 871
10.180.3.2 GetLength . . . 872

10.180.3.3 GetSquaredLength . . . 872
10.180.3.4 IsFinite . . . 872
10.180.3.5 Normalize . . . 872
10.180.3.6 operator!= . . . 872
10.180.3.7 operator* . . . 872
10.180.3.8 operator* . . . 873
10.180.3.9 operator* . . . 873
10.180.3.10 operator*= . . . 873
10.180.3.11 operator*= . . . 874
10.180.3.12 operator+ . . . 874
10.180.3.13 operator+= . . . 874
10.180.3.14 operator- . . . 874
10.180.3.15 operator-= . . . 875
10.180.3.16 operator/ . . . 875
10.180.3.17 operator/ . . . 875
10.180.3.18 operator/= . . . 876
10.180.3.19 operator/= . . . 876
10.180.3.20 operator= . . . 876
10.180.3.21 operator= . . . 876
10.180.3.22 operator== . . . 877
10.180.3.23 operator[] . . . 877
10.180.3.24 Set . . . 877
10.180.4 Friends And Related Function Documentation . . . 877
10.180.4.1 operator<< . . . 877
10.180.4.2 operator>> . . . 877
10.180.5 Member Data Documentation . . . 878
10.180.5.1 w . . . 878
10.180.5.2 x . . . 878
10.180.5.3 y . . . 878
10.180.5.4 z . . . 878
10.181 gazebo::common::Video Class Reference . . . 878
10.181.1 Detailed Description . . . 879
10.181.2 Constructor & Destructor Documentation . . . 879
10.181.2.1 Video . . . 879
10.181.2.2 ~Video . . . 879
10.181.3 Member Function Documentation . . . 879
10.181.3.1 GetHeight . . . 879

10.181.3.2 GetNextFrame . . . 879
10.181.3.3 GetWidth . . . 879
10.181.3.4 Load . . . 880
10.182 gazebo::rendering::VideoVisual Class Reference . . . 880
10.182.1 Detailed Description . . . 881
10.182.2 Constructor & Destructor Documentation . . . 881
10.182.2.1 VideoVisual . . . 881
10.182.2.2 ~VideoVisual . . . 881
10.183 gazebo::rendering::ViewController Class Reference . . . 881
10.183.1 Detailed Description . . . 882
10.183.2 Constructor & Destructor Documentation . . . 882
10.183.2.1 ViewController . . . 882
10.183.2.2 ~ViewController . . . 883
10.183.3 Member Function Documentation . . . 883
10.183.3.1 GetTypeString . . . 883
10.183.3.2 HandleKeyPressEvent . . . 883
10.183.3.3 HandleKeyReleaseEvent . . . 883
10.183.3.4 HandleMouseEvent . . . 883
10.183.3.5 Init . . . 884
10.183.3.6 Init . . . 884
10.183.3.7 SetEnabled . . . 884
10.183.3.8 Update . . . 884
10.183.4 Member Data Documentation . . . 884
10.183.4.1 camera . . . 884
10.183.4.2 enabled . . . 884
10.183.4.3 typeString . . . 884
10.184 gazebo::rendering::Visual Class Reference . . . 885
10.184.1 Detailed Description . . . 889
10.184.2 Constructor & Destructor Documentation . . . 890
10.184.2.1 Visual . . . 890
10.184.2.2 Visual . . . 890
10.184.2.3 ~Visual . . . 890
10.184.3 Member Function Documentation . . . 890
10.184.3.1 AttachAxes . . . 890
10.184.3.2 AttachLineVertex . . . 890
10.184.3.3 AttachMesh . . . 890
10.184.3.4 AttachObject . . . 891

10.184.3.5 AttachVisual . . . 891
10.184.3.6 ClearParent . . . 891
10.184.3.7 Clone . . . 891
10.184.3.8 CreateDynamicLine . . . 891
10.184.3.9 DeleteDynamicLine . . . 892
10.184.3.10 DetachObjects . . . 892
10.184.3.11 DetachVisual . . . 892
10.184.3.12 DetachVisual . . . 892
10.184.3.13 DisableTrackVisual . . . 892
10.184.3.14 EnableTrackVisual . . . 892
10.184.3.15 Fini . . . 892
10.184.3.16 GetAttachedObjectCount . . . 893
10.184.3.17 GetBoundingBox . . . 893
10.184.3.18 GetChild . . . 893
10.184.3.19 GetChildCount . . . 893
10.184.3.20 GetMaterialName . . . 893
10.184.3.21 GetMeshName . . . 893
10.184.3.22 GetName . . . 894
10.184.3.23 GetNormalMap . . . 894
10.184.3.24 GetParent . . . 894
10.184.3.25 GetPose . . . 894
10.184.3.26 GetPosition . . . 894
10.184.3.27 GetRootVisual . . . 894
10.184.3.28 GetRotation . . . 895
10.184.3.29 GetScale . . . 895
10.184.3.30 GetScene . . . 895
10.184.3.31 GetSceneNode . . . 895
10.184.3.32 GetShaderType . . . 895
10.184.3.33 GetSubMeshName . . . 895
10.184.3.34 GetTransparency . . . 896
10.184.3.35 GetVisibilityFlags . . . 896
10.184.3.36 GetVisible . . . 896
10.184.3.37 GetWorldPose . . . 896
10.184.3.38 HasAttachedObject . . . 896
10.184.3.39 Init . . . 897
10.184.3.40 InsertMesh . . . 897
10.184.3.41 InsertMesh . . . 897

10.184.3.42 IsPlane . . . 897
10.184.3.43 IsStatic . . . 897
10.184.3.44 Load . . . 897
10.184.3.45 Load . . . 898
10.184.3.46 LoadFromMsg . . . 898
10.184.3.47 LoadPlugin . . . 898
10.184.3.48 MakeStatic . . . 898
10.184.3.49 MoveToPosition . . . 898
10.184.3.50 MoveToPositions . . . 898
10.184.3.51 RemovePlugin . . . 899
10.184.3.52 SetAmbient . . . 899
10.184.3.53 SetCastShadows . . . 899
10.184.3.54 SetDiffuse . . . 899
10.184.3.55 SetEmissive . . . 899
10.184.3.56 SetHighlighted . . . 899
10.184.3.57 SetMaterial . . . 900
10.184.3.58 SetName . . . 900
10.184.3.59 SetNormalMap . . . 900
10.184.3.60 SetPose . . . 900
10.184.3.61 SetPosition . . . 900
10.184.3.62 SetRibbonTrail . . . 901
10.184.3.63 SetRotation . . . 901
10.184.3.64 SetScale . . . 901
10.184.3.65 SetScene . . . 901
10.184.3.66 SetShaderType . . . 901
10.184.3.67 SetSkeletonPose . . . 901
10.184.3.68 SetSpecular . . . 902
10.184.3.69 SetTransparency . . . 902
10.184.3.70 SetVisibilityFlags . . . 902
10.184.3.71 SetVisible . . . 902
10.184.3.72 SetWireframe . . . 902
10.184.3.73 SetWorldPose . . . 903
10.184.3.74 SetWorldPosition . . . 903
10.184.3.75 SetWorldRotation . . . 903
10.184.3.76 ShowBoundingBox . . . 903
10.184.3.77 ShowCollision . . . 903
10.184.3.78 ShowCOM . . . 903

10.184.3.7 ShowJoints . . . 904
10.184.3.8 ShowSkeleton . . . 904
10.184.3.9 ToggleVisible . . . 904
10.184.3.10 Update . . . 904
10.184.3.11 UpdateFromMsg . . . 904
10.184.4 Member Data Documentation . . . 904
10.184.4.1 parent . . . 904
10.184.4.2 scene . . . 904
10.184.4.3 sceneNode . . . 904
10.185 gazebo::VisualPlugin Class Reference . . . 905
10.185.1 Detailed Description . . . 905
10.185.2 Constructor & Destructor Documentation . . . 905
10.185.2.1 VisualPlugin . . . 905
10.185.3 Member Function Documentation . . . 905
10.185.3.1 Init . . . 906
10.185.3.2 Load . . . 906
10.185.3.3 Reset . . . 906
10.186 gazebo::rendering::WindowManager Class Reference . . . 906
10.186.1 Detailed Description . . . 907
10.186.2 Member Function Documentation . . . 907
10.186.2.1 CreateWindow . . . 907
10.186.2.2 Fini . . . 907
10.186.2.3 GetAvgFPS . . . 907
10.186.2.4 GetTriangleCount . . . 908
10.186.2.5 GetWindow . . . 908
10.186.2.6 Moved . . . 908
10.186.2.7 Resize . . . 908
10.186.2.8 SetCamera . . . 909
10.187 gazebo::rendering::WireBox Class Reference . . . 909
10.187.1 Detailed Description . . . 909
10.187.2 Constructor & Destructor Documentation . . . 909
10.187.2.1 WireBox . . . 909
10.187.2.2 ~WireBox . . . 909
10.187.3 Member Function Documentation . . . 910
10.187.3.1 Init . . . 910
10.187.3.2 SetVisible . . . 910
10.188 gazebo::physics::World Class Reference . . . 910

10.188.1 Detailed Description . . . 913
10.188.2 Constructor & Destructor Documentation . . . 913
10.188.2.1 World . . . 913
10.188.2.2 ~World . . . 913
10.188.3 Member Function Documentation . . . 913
10.188.3.1 Clear . . . 913
10.188.3.2 DisableAllModels . . . 913
10.188.3.3 EnableAllModels . . . 913
10.188.3.4 EnablePhysicsEngine . . . 914
10.188.3.5 Fini . . . 914
10.188.3.6 GetByName . . . 914
10.188.3.7 GetEnablePhysicsEngine . . . 914
10.188.3.8 GetEntity . . . 914
10.188.3.9 GetEntityBelowPoint . . . 915
10.188.3.10 GetModel . . . 915
10.188.3.11 GetModel . . . 915
10.188.3.12 GetModelBelowPoint . . . 915
10.188.3.13 GetModelCount . . . 916
10.188.3.14 GetModels . . . 916
10.188.3.15 GetName . . . 916
10.188.3.16 GetPauseTime . . . 916
10.188.3.17 GetPhysicsEngine . . . 916
10.188.3.18 GetRealTime . . . 917
10.188.3.19 GetSelectedEntity . . . 917
10.188.3.20 GetSetWorldPoseMutex . . . 917
10.188.3.21 GetSimTime . . . 917
10.188.3.22 GetStartTime . . . 917
10.188.3.23 Init . . . 917
10.188.3.24 InsertModelFile . . . 918
10.188.3.25 InsertModelSDF . . . 918
10.188.3.26 InsertModelString . . . 918
10.188.3.27 IsLoaded . . . 918
10.188.3.28 IsPaused . . . 918
10.188.3.29 Load . . . 918
10.188.3.30 LoadPlugin . . . 919
10.188.3.31 PrintEntityTree . . . 919
10.188.3.32 PublishModelPose . . . 919

10.188.3.3RemovePlugin . . . 919
10.188.3.4Reset . . . 919
10.188.3.5ResetEntities . . . 919
10.188.3.6ResetTime . . . 920
10.188.3.7Run . . . 920
10.188.3.8Save . . . 920
10.188.3.9SetPaused . . . 920
10.188.3.10SetSimTime . . . 920
10.188.3.11SetState . . . 920
10.188.3.12StepWorld . . . 921
10.188.3.13Stop . . . 921
10.188.3.14StripWorldName . . . 921
10.188.3.15UpdateStateSDF . . . 921
10.188.4Member Data Documentation . . . 921
10.188.4.1dirtyPoses . . . 921
10.189gazebo::WorldPlugin Class Reference . . . 921
10.189.1Detailed Description . . . 922
10.189.2Constructor & Destructor Documentation . . . 922
10.189.2.1WorldPlugin . . . 922
10.189.2.2~WorldPlugin . . . 922
10.189.3Member Function Documentation . . . 923
10.189.3.1Init . . . 923
10.189.3.2Load . . . 923
10.189.3.3Reset . . . 923
10.190gazebo::physics::WorldState Class Reference . . . 923
10.190.1Detailed Description . . . 924
10.190.2Constructor & Destructor Documentation . . . 924
10.190.2.1WorldState . . . 924
10.190.2.2WorldState . . . 925
10.190.2.3WorldState . . . 925
10.190.2.4~WorldState . . . 925
10.190.3Member Function Documentation . . . 925
10.190.3.1FillSDF . . . 925
10.190.3.2GetModelState . . . 925
10.190.3.3GetModelState . . . 926
10.190.3.4GetModelStateCount . . . 926
10.190.3.5GetModelStates . . . 926

10.190.3.6 HasModelState . . . 926
10.190.3.7 IsZero . . . 926
10.190.3.8 Load . . . 927
10.190.3.9 operator+ . . . 927
10.190.3.10 operator- . . . 927
10.190.3.11 operator= . . . 927
10.190.3.12 SetWorld . . . 928
10.190.4 Friends And Related Function Documentation . . . 928
10.190.4.1 operator<< . . . 928

11 File Documentation **929**

11.1 Actor.hh File Reference . . . 929
11.2 Angle.hh File Reference . . . 930
11.2.1 Macro Definition Documentation . . . 932
11.2.1.1 GZ_DTOR . . . 932
11.2.1.2 GZ_NORMALIZE . . . 932
11.2.1.3 GZ_RTOD . . . 932
11.3 Animation.hh File Reference . . . 933
11.4 ArrowVisual.hh File Reference . . . 934
11.5 Assert.hh File Reference . . . 935
11.5.1 Macro Definition Documentation . . . 936
11.5.1.1 GZ_ASSERT . . . 936
11.6 AxisVisual.hh File Reference . . . 937
11.7 BallJoint.hh File Reference . . . 937
11.8 Base.hh File Reference . . . 938
11.9 Box.hh File Reference . . . 939
11.10 BoxShape.hh File Reference . . . 940
11.11 BVHLoader.hh File Reference . . . 941
11.11.1 Macro Definition Documentation . . . 943
11.11.1.1 X_POSITION . . . 943
11.11.1.2 X_ROTATION . . . 943
11.11.1.3 Y_POSITION . . . 943
11.11.1.4 Y_ROTATION . . . 943
11.11.1.5 Z_POSITION . . . 943
11.11.1.6 Z_ROTATION . . . 943
11.12 CallbackHelper.hh File Reference . . . 943
11.13 Camera.hh File Reference . . . 945

11.14CameraSensor.hh File Reference . . . 946
11.15CameraVisual.hh File Reference . . . 946
11.16cegui.h File Reference . . . 947
11.17ColladaLoader.hh File Reference . . . 948
11.18Collision.hh File Reference . . . 949
11.19CollisionState.hh File Reference . . . 951
11.20Color.hh File Reference . . . 952
11.21Common.hh File Reference . . . 953
11.22CommonTypes.hh File Reference . . . 954
11.22.1 Macro Definition Documentation . . . 955
11.22.1.1 GAZEBO_DEPRECATED . . . 955
11.22.1.2 GAZEBO_FORCEINLINE . . . 955
11.22.1.3 NULL . . . 955
11.23COMVisual.hh File Reference . . . 956
11.24Connection.hh File Reference . . . 956
11.24.1 Macro Definition Documentation . . . 958
11.24.1.1 HEADER_LENGTH . . . 958
11.25ConnectionManager.hh File Reference . . . 958
11.26Console.hh File Reference . . . 959
11.27Contact.hh File Reference . . . 961
11.27.1 Macro Definition Documentation . . . 962
11.27.1.1 MAX_COLLIDE_RETURNS . . . 962
11.27.1.2 MAX_CONTACT_JOINTS . . . 962
11.28ContactManager.hh File Reference . . . 962
11.29ContactSensor.hh File Reference . . . 963
11.30ContactVisual.hh File Reference . . . 964
11.31Conversions.hh File Reference . . . 964
11.32Converter.hh File Reference . . . 965
11.33CylinderShape.hh File Reference . . . 966
11.34DepthCamera.hh File Reference . . . 967
11.35DepthCameraSensor.hh File Reference . . . 968
11.36Diagnostics.hh File Reference . . . 968
11.37DynamicLines.hh File Reference . . . 969
11.38DynamicRenderable.hh File Reference . . . 970
11.39Entity.hh File Reference . . . 971
11.40Event.hh File Reference . . . 973
11.41Events.hh File Reference . . . 974

11.42Exception.hh File Reference . . . 974
11.43FPSViewController.hh File Reference . . . 976
11.44gazebo.hh File Reference . . . 977
11.45gazebo_core.hh File Reference . . . 978
11.46GazeboGenerator.hh File Reference . . . 978
11.47GpuLaser.hh File Reference . . . 979
11.48GpuRaySensor.hh File Reference . . . 980
11.49Grid.hh File Reference . . . 981
11.50Gripper.hh File Reference . . . 981
11.51GUIOverlay.hh File Reference . . . 983
11.52Heightmap.hh File Reference . . . 983
11.53HeightmapShape.hh File Reference . . . 984
11.54Helpers.hh File Reference . . . 985
11.54.1 Macro Definition Documentation . . . 987
11.54.1.1 GZ_DBL_MAX . . . 987
11.54.1.2 GZ_DBL_MIN . . . 987
11.54.1.3 GZ_FLT_MAX . . . 987
11.54.1.4 GZ_FLT_MIN . . . 987
11.55Hinge2Joint.hh File Reference . . . 987
11.56HingeJoint.hh File Reference . . . 989
11.57Image.hh File Reference . . . 990
11.58ImuSensor.hh File Reference . . . 991
11.59Inertial.hh File Reference . . . 991
11.60IOManager.hh File Reference . . . 992
11.61Joint.hh File Reference . . . 994
11.61.1 Macro Definition Documentation . . . 995
11.61.1.1 MAX_JOINT_AXIS . . . 995
11.62JointController.hh File Reference . . . 995
11.63JointState.hh File Reference . . . 996
11.64JointVisual.hh File Reference . . . 997
11.65JointWrench.hh File Reference . . . 998
11.66KeyFrame.hh File Reference . . . 999
11.67LaserVisual.hh File Reference . . . 1000
11.68Light.hh File Reference . . . 1001
11.69Link.hh File Reference . . . 1002
11.70LinkState.hh File Reference . . . 1003
11.71LogPlay.hh File Reference . . . 1005

11.72LogRecord.hh File Reference . . . 1006
11.72.1 Macro Definition Documentation . . . 1008
11.72.1.1 GZ_LOG_VERSION . . . 1008
11.73mainpage.html File Reference . . . 1008
11.74MapShape.hh File Reference . . . 1008
11.75Master.hh File Reference . . . 1009
11.76Material.hh File Reference . . . 1010
11.77Material.hh File Reference . . . 1012
11.78MathTypes.hh File Reference . . . 1012
11.78.1 Detailed Description . . . 1012
11.79Matrix3.hh File Reference . . . 1013
11.80Matrix4.hh File Reference . . . 1013
11.81Mesh.hh File Reference . . . 1014
11.82MeshCSG.hh File Reference . . . 1016
11.82.1 Typedef Documentation . . . 1016
11.82.1.1 GPtrArray . . . 1016
11.82.1.2 GtsSurface . . . 1017
11.83MeshLoader.hh File Reference . . . 1017
11.84MeshManager.hh File Reference . . . 1018
11.85Model.hh File Reference . . . 1019
11.86ModelDatabase.hh File Reference . . . 1021
11.86.1 Macro Definition Documentation . . . 1021
11.86.1.1 GZ_MODEL_DB_MANIFEST_FILENAME . . . 1021
11.86.1.2 GZ_MODEL_MANIFEST_FILENAME . . . 1022
11.87ModelState.hh File Reference . . . 1022
11.88MouseEvent.hh File Reference . . . 1024
11.89MovableText.hh File Reference . . . 1025
11.90MsgFactory.hh File Reference . . . 1025
11.91msgs.hh File Reference . . . 1026
11.92MultiCameraSensor.hh File Reference . . . 1029
11.93MultiRayShape.hh File Reference . . . 1030
11.94Node.hh File Reference . . . 1031
11.95ogre_gazebo.h File Reference . . . 1032
11.96OrbitViewController.hh File Reference . . . 1033
11.97Param.hh File Reference . . . 1034
11.98parser.hh File Reference . . . 1035
11.99parser_urdf.hh File Reference . . . 1036

11.100Physics.hh File Reference . . . 1037
11.101PhysicsEngine.hh File Reference . . . 1039
11.102PhysicsFactory.hh File Reference . . . 1040
11.103PhysicsTypes.hh File Reference . . . 1042
11.103.1Detailed Description . . . 1044
11.103.2Macro Definition Documentation . . . 1044
11.103.2.1GZ_ALL_COLLIDE . . . 1044
11.103.2.2GZ_FIXED_COLLIDE . . . 1044
11.103.2.3GZ_GHOST_COLLIDE . . . 1044
11.103.2.4GZ_NONE_COLLIDE . . . 1044
11.103.2.5GZ_SENSOR_COLLIDE . . . 1044
11.104PID.hh File Reference . . . 1044
11.105Plane.hh File Reference . . . 1045
11.106PlaneShape.hh File Reference . . . 1046
11.107Plugin.hh File Reference . . . 1048
11.107.1Macro Definition Documentation . . . 1050
11.107.1.1GZ_REGISTER_MODEL_PLUGIN . . . 1050
11.107.1.2GZ_REGISTER_SENSOR_PLUGIN . . . 1050
11.107.1.3GZ_REGISTER_SYSTEM_PLUGIN . . . 1051
11.107.1.4GZ_REGISTER_VISUAL_PLUGIN . . . 1051
11.107.1.5GZ_REGISTER_WORLD_PLUGIN . . . 1051
11.108Plugin.hh File Reference . . . 1052
11.109Pose.hh File Reference . . . 1052
11.110Projector.hh File Reference . . . 1053
11.111Publication.hh File Reference . . . 1054
11.112PublicationTransport.hh File Reference . . . 1055
11.113Publisher.hh File Reference . . . 1057
11.114Quaternion.hh File Reference . . . 1059
11.115Rand.hh File Reference . . . 1060
11.116RaySensor.hh File Reference . . . 1061
11.117RayShape.hh File Reference . . . 1062
11.118RenderEngine.hh File Reference . . . 1063
11.119RenderEvents.hh File Reference . . . 1064
11.120Rendering.hh File Reference . . . 1065
11.121RenderTypes.hh File Reference . . . 1066
11.121.1Macro Definition Documentation . . . 1067
11.121.1.1GZ_VISIBILITY_ALL . . . 1067

11.121.1.2GZ_VISIBILITY_GUI . . . 1067
11.121.1.3GZ_VISIBILITY_NOT_SELECTABLE . . . 1067
11.121.1.4GZ_VISIBILITY_SELECTION . . . 1068
11.122RFIDSensor.hh File Reference . . . 1068
11.123RFIDTag.hh File Reference . . . 1068
11.124RFIDTagVisual.hh File Reference . . . 1069
11.125RFIDVisual.hh File Reference . . . 1070
11.126Road.hh File Reference . . . 1070
11.127Road2d.hh File Reference . . . 1072
11.128RotationSpline.hh File Reference . . . 1072
11.129RTShaderSystem.hh File Reference . . . 1073
11.130Scene.hh File Reference . . . 1074
11.131ScrewJoint.hh File Reference . . . 1075
11.132sdf.hh File Reference . . . 1076
11.133SDF.hh File Reference . . . 1077
11.133.1Macro Definition Documentation . . . 1078
11.133.1.1SDF_VERSION . . . 1078
11.134SelectionObj.hh File Reference . . . 1078
11.135Sensor.hh File Reference . . . 1079
11.136SensorFactory.hh File Reference . . . 1080
11.137SensorManager.hh File Reference . . . 1081
11.138Sensors.hh File Reference . . . 1082
11.139SensorTypes.hh File Reference . . . 1084
11.139.1Detailed Description . . . 1085
11.140Server.hh File Reference . . . 1085
11.141Shape.hh File Reference . . . 1086
11.142SingletonT.hh File Reference . . . 1088
11.143Skeleton.hh File Reference . . . 1088
11.144SkeletonAnimation.hh File Reference . . . 1090
11.145SliderJoint.hh File Reference . . . 1091
11.146SphereShape.hh File Reference . . . 1093
11.147Spline.hh File Reference . . . 1094
11.148State.hh File Reference . . . 1095
11.149STLLoader.hh File Reference . . . 1096
11.149.1Macro Definition Documentation . . . 1097
11.149.1.1COR3_MAX . . . 1097
11.149.1.2FACE_MAX . . . 1098

11.149.1.3 LINE_MAX_LEN . . . 1098
11.149.1.4 ORDER_MAX . . . 1098
11.150 SubscribeOptions.hh File Reference . . . 1098
11.151 Subscriber.hh File Reference . . . 1099
11.152 SubscriptionTransport.hh File Reference . . . 1101
11.153 SurfaceParams.hh File Reference . . . 1102
11.154 SystemPaths.hh File Reference . . . 1104
11.154.1 Macro Definition Documentation . . . 1105
11.154.1.1 GetCurrentDir . . . 1105
11.154.1.2 LINUX . . . 1105
11.155 Time.hh File Reference . . . 1105
11.156 Timer.hh File Reference . . . 1106
11.157 TopicManager.hh File Reference . . . 1107
11.158 Transport.hh File Reference . . . 1109
11.159 TransportTypes.hh File Reference . . . 1111
11.159.1 Detailed Description . . . 1112
11.160 TrimeshShape.hh File Reference . . . 1113
11.161 UniversalJoint.hh File Reference . . . 1114
11.162 UpdateInfo.hh File Reference . . . 1115
11.163 UserCamera.hh File Reference . . . 1116
11.164 UtilTypes.hh File Reference . . . 1116
11.165 Vector2d.hh File Reference . . . 1117
11.166 Vector2i.hh File Reference . . . 1118
11.167 Vector3.hh File Reference . . . 1119
11.168 Vector4.hh File Reference . . . 1120
11.169 Video.hh File Reference . . . 1122
11.170 VideoVisual.hh File Reference . . . 1123
11.171 ViewController.hh File Reference . . . 1124
11.172 Visual.hh File Reference . . . 1125
11.173 WindowManager.hh File Reference . . . 1126
11.174 WireBox.hh File Reference . . . 1126
11.175 World.hh File Reference . . . 1127
11.176 WorldState.hh File Reference . . . 1129

Index **1130**

Chapter 1

Gazebo API Reference

This documentation provides useful information about the Gazebo API. The code reference is divided into the groups below. Should you find problems with this documentation - typos, unclear phrases, or insufficient detail - please create a `new bitbucket issue`. Include sufficient detail to quickly locate the problematic documentation, and set the issue's fields accordingly: Assignee - blank; Kind - bug; Priority - minor; Version - blank.

Class `List` - Index of all classes in Gazebo, organized alphabetically

`Hierarchy` - Index of classes, organized hierachically according to their inheritance

`Modules Common`: Classes and files used ubiquitously across Gazebo

`Events`: For creating and destroying Gazebo events

`Math`: **A** (p. 111) set of classes that encapsulate math related properties and functions.

`Messages`: All messages and helper functions.

`Physics`: Classes for physics and dynamics

`Rendering`: **A** (p. 111) set of rendering related class, functions, and definitions.

`Sensors`: **A** (p. 111) set of sensor classes, functions, and definitions.

`Transport`: Handles transportation of messages.

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

`Wiki`: **A** (p. 111) collection of user supported documentation.

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

`Download`: How to download and install Gazebo

Chapter 2

Todo List

Member gazebo::physics::Joint::GetForce (p. **389) (int _index) GAZEBO_DEPRECATED(1.5)**

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

Member gazebo::physics::Joint::GetForce (p. **389) (unsigned int _index)**

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

Member gazebo::sensors::CameraSensor::GetTopic (p. **191) () const**

to be implemented

Class gazebo::SystemPlugin (p. **789)**

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

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Common 27
Events 38
Classes for physics and dynamics 41
Math 48
Messages 54
Rendering 65
Gazebo_parser 69
Sensors 70
Transport 75
Utility 80

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost . . . 81
gazebo
Forward declarations for the common classes . . . 81
gazebo::common
Common namespace . . . 83
gazebo::event
Event (p. 292) namespace . . . 86
gazebo::math
Math namespace . . . 87
gazebo::msgs
Messages namespace . . . 89
gazebo::physics
Namespace for physics . . . 91
gazebo::rendering
Rendering namespace . . . 96
gazebo::sensors
Sensors namespace . . . 100
gazebo::transport . . . 103
gazebo::util . . . 105
google . . . 105
google::protobuf . . . 105
google::protobuf::compiler . . . 106
google::protobuf::compiler::cpp . . . 106
Ogre . . . 106
ogre . . . 106
sdf
Namespace for Simulation Description Format parser . . . 106
SkyX . . . 108
urdf2gazebo
Namespace for URDF to SDF parser . . . 108

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

A . . . 111
gazebo::math::Angle . . . 118
gazebo::common::Animation . . . 125
  gazebo::common::NumericAnimation . . . 552
  gazebo::common::PoseAnimation . . . 605
gazebo::math::Box . . . 148
gazebo::common::BVHLoader . . . 156
gazebo::transport::CallbackHelper . . . 157
  gazebo::transport::CallbackHelperT< M > . . . 160
  gazebo::transport::RawCallbackHelper . . . 638
  gazebo::transport::SubscriptionTransport . . . 778
CodeGenerator
  google::protobuf::compiler::cpp::GazeboGenerator . . . 323
gazebo::common::Color . . . 208
gazebo::event::Connection . . . 221
gazebo::physics::Contact . . . 236
gazebo::physics::ContactManager . . . 239
gazebo::rendering::Conversions . . . 247
sdf::Converter . . . 249
enable_shared_from_this
  gazebo::physics::Base . . . 137
    gazebo::physics::Entity . . . 281
      gazebo::physics::Collision . . . 195
      gazebo::physics::Link . . . 418
      gazebo::physics::Model . . . 489
        gazebo::physics::Actor . . . 111
    gazebo::physics::Joint . . . 381
    gazebo::physics::Road . . . 665
    gazebo::physics::Shape . . . 720
      gazebo::physics::BoxShape . . . 153
      gazebo::physics::CylinderShape . . . 250
      gazebo::physics::HeightmapShape . . . 352
      gazebo::physics::MultiRayShape . . . 527

gazebo::physics::PlaneShape . . . 590
gazebo::physics::RayShape . . . 647
gazebo::physics::SphereShape . . . 751
gazebo::physics::TrimeshShape . . . 822
gazebo::physics::World . . . 910
gazebo::rendering::Camera . . . 162
gazebo::rendering::DepthCamera . . . 253
gazebo::rendering::GpuLaser . . . 324
gazebo::rendering::UserCamera . . . 830
gazebo::rendering::Scene . . . 676
gazebo::rendering::Visual . . . 885
gazebo::rendering::ArrowVisual . . . 129
gazebo::rendering::AxisVisual . . . 133
gazebo::rendering::CameraVisual . . . 193
gazebo::rendering::COMVisual . . . 219
gazebo::rendering::ContactVisual . . . 246
gazebo::rendering::JointVisual . . . 405
gazebo::rendering::LaserVisual . . . 410
gazebo::rendering::RFIDTagVisual . . . 662
gazebo::rendering::RFIDVisual . . . 664
gazebo::rendering::VideoVisual . . . 880
gazebo::sensors::Sensor . . . 698
gazebo::sensors::CameraSensor . . . 188
gazebo::sensors::ContactSensor . . . 241
gazebo::sensors::DepthCameraSensor . . . 258
gazebo::sensors::GpuRaySensor . . . 328
gazebo::sensors::ImuSensor . . . 365
gazebo::sensors::MultiCameraSensor . . . 523
gazebo::sensors::RaySensor . . . 641
gazebo::sensors::RFIDSensor . . . 658
gazebo::sensors::RFIDTag . . . 660
gazebo::transport::Connection . . . 222
gazebo::transport::Node . . . 535
sdf::Element . . . 273
gazebo::event::Event . . . 292
gazebo::event::EventT< void() > . . . 309
gazebo::event::EventT< void(bool) > . . . 309
gazebo::event::EventT< void(const common::UpdateInfo &) > . . . 309
gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &) > . . . 309
gazebo::event::EventT< void(const std::string &) > . . . 309
gazebo::event::EventT< void(const std::string &, const Contact &) > . . . 309
gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &) > . . . 309
gazebo::event::EventT< void(std::string) > . . . 309
gazebo::event::EventT< void(std::string, std::string) > . . . 309
gazebo::event::EventT< T > . . . 309
gazebo::rendering::Events . . . 295
gazebo::event::Events . . . 297
gazebo::common::Exception . . . 318
gazebo::common::InternalError . . . 378
gazebo::common::AssertionInternalError . . . 131
urdf2gazebo::GazeboExtension . . . 323
gazebo::rendering::Grid . . . 340

gazebo::physics::Gripper . . . 344
gazebo::rendering::GUIOverlay . . . 345
gazebo::rendering::Heightmap . . . 350
gazebo::common::Image . . . 360
gazebo::physics::Inertial . . . 369
gazebo::transport::IOManager . . . 380
gazebo::physics::JointController . . . 399
gazebo::physics::JointWrench . . . 407
gazebo::common::KeyFrame . . . 409
  gazebo::common::NumericKeyFrame . . . 553
  gazebo::common::PoseKeyFrame . . . 607
gazebo::rendering::Light . . . 412
Logplay . . . 446
gazebo::Master . . . 452
gazebo::common::Material . . . 453
gazebo::math::Matrix3 . . . 463
gazebo::math::Matrix4 . . . 467
gazebo::common::Mesh . . . 475
gazebo::common::MeshCSG . . . 481
gazebo::common::MeshLoader . . . 483
  gazebo::common::ColladaLoader . . . 194
  gazebo::common::STLLoader . . . 762
gazebo::common::MouseEvent . . . 512
MovableObject
  gazebo::rendering::MovableText . . . 515
gazebo::msgs::MsgFactory . . . 521
gazebo::common::NodeAnimation . . . 542
gazebo::common::NodeAssignment . . . 546
gazebo::common::NodeTransform . . . 547
sdf::Param . . . 559
  sdf::ParamT< std::string > . . . 565
  sdf::ParamT< T > . . . 565
ParamT< T > . . . 565
gazebo::physics::PhysicsEngine . . . 568
gazebo::physics::PhysicsFactory . . . 582
gazebo::common::PID . . . 583
gazebo::math::Plane . . . 587
gazebo::PluginT< T > . . . 594
gazebo::PluginT< ModelPlugin > . . . 594
  gazebo::ModelPlugin . . . 504
gazebo::PluginT< SensorPlugin > . . . 594
  gazebo::SensorPlugin . . . 712
gazebo::PluginT< SystemPlugin > . . . 594
  gazebo::SystemPlugin . . . 789
gazebo::PluginT< VisualPlugin > . . . 594
  gazebo::VisualPlugin . . . 905
gazebo::PluginT< WorldPlugin > . . . 594
  gazebo::WorldPlugin . . . 921
gazebo::math::Pose . . . 596
gazebo::rendering::Projector . . . 609
gazebo::transport::Publication . . . 612
gazebo::transport::PublicationTransport . . . 616

gazebo::transport::Publisher 618
gazebo::math::Quaternion 623
gazebo::math::Rand 637
Renderable
  gazebo::rendering::MovableText 515
RenderObjectListener
  gazebo::rendering::GpuLaser 324
Road 667
gazebo::rendering::Road2d 667
gazebo::math::RotationSpline 668
sdf::SDF 695
SDFBase
  sdf::Plugin 593
gazebo::rendering::SelectionObj 696
SensorFactor 707
gazebo::sensors::SensorFactory 707
gazebo::Server 714
ShaderHelperCg
  gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg 715
ShaderHelperGLSL
  gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL 717
SimpleRenderable
  gazebo::rendering::DynamicRenderable 269
    gazebo::rendering::DynamicLines 266
gazebo::sensors::SimTimeEvent 723
SingletonT< T > 725
  gazebo::common::Console 235
  gazebo::common::LogPlay 444
  gazebo::common::LogRecord 447
  gazebo::common::MeshManager 484
  gazebo::common::ModelDatabase 502
  gazebo::common::SystemPaths 784
  gazebo::rendering::RenderEngine 653
  gazebo::rendering::RTShaderSystem 672
  gazebo::sensors::SensorManager 709
  gazebo::sensors::SimTimeEventHandler 723
  gazebo::transport::ConnectionManager 229
  gazebo::transport::TopicManager 815
  gazebo::util::DiagnosticManager 261
SingletonT< ConnectionManager > 725
SingletonT< Console > 725
SingletonT< DiagnosticManager > 725
SingletonT< LogPlay > 725
SingletonT< LogRecord > 725
SingletonT< MeshManager > 725
SingletonT< ModelDatabase > 725
SingletonT< RenderEngine > 725
SingletonT< RTShaderSystem > 725
SingletonT< SensorManager > 725
SingletonT< SimTimeEventHandler > 725
SingletonT< SystemPaths > 725
SingletonT< TopicManager > 725
SingletonT< WindowManager > 725
  gazebo::rendering::WindowManager 906

gazebo::common::Skeleton . . . 727
gazebo::common::SkeletonAnimation . . . 734
gazebo::common::SkeletonNode . . . 738
SM2Profile
  gazebo::rendering::GzTerrainMatGen::SM2Profile . . . 749
gazebo::math::Spline . . . 754
gazebo::physics::State . . . 758
  gazebo::physics::CollisionState . . . 205
  gazebo::physics::JointState . . . 401
  gazebo::physics::LinkState . . . 438
  gazebo::physics::ModelState . . . 505
  gazebo::physics::WorldState . . . 923
gazebo::common::SubMesh . . . 764
gazebo::transport::SubscribeOptions . . . 774
gazebo::transport::Subscriber . . . 776
gazebo::physics::SurfaceParams . . . 780
T
  gazebo::physics::BallJoint< T > . . . 135
  gazebo::physics::Hinge2Joint< T > . . . 356
  gazebo::physics::HingeJoint< T > . . . 358
  gazebo::physics::ScrewJoint< T > . . . 691
  gazebo::physics::SliderJoint< T > . . . 747
  gazebo::physics::UniversalJoint< T > . . . 826
task
  gazebo::transport::ConnectionReadTask . . . 233
  gazebo::transport::PublishTask . . . 622
TerrainMaterialGeneratorA
  gazebo::rendering::GzTerrainMatGen . . . 349
gazebo::common::Time . . . 791
gazebo::common::Timer . . . 813
  gazebo::util::DiagnosticTimer . . . 264
gazebo::physics::TrajectoryInfo . . . 821
gazebo::common::UpdateInfo . . . 827
urdf2gazebo::URDF2Gazebo . . . 828
gazebo::math::Vector2d . . . 838
gazebo::math::Vector2i . . . 846
gazebo::math::Vector3 . . . 855
gazebo::math::Vector4 . . . 869
gazebo::common::Video . . . 878
gazebo::rendering::ViewController . . . 881
  gazebo::rendering::FPSViewController . . . 320
  gazebo::rendering::OrbitViewController . . . 555
gazebo::rendering::WireBox . . . 909

Chapter 6

Class Index

6.1 Class List

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

A
Holding gazebo extension elements in urdf . . . 111
gazebo::physics::Actor
Actor (p. 111) class enables GPU based mesh model / skeleton scriptable animation . . . 111
gazebo::math::Angle
An angle and related functions . . . 118
gazebo::common::Animation
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes . . . 125
gazebo::rendering::ArrowVisual
Basic arrow visualization . . . 129
gazebo::common::AssertionInternalError
Class for generating Exceptions which come from gazebo assertions . . . 131
gazebo::rendering::AxisVisual
Basic axis visualization . . . 133
gazebo::physics::BallJoint< T >
Base (p. 137) class for a ball joint . . . 135
gazebo::physics::Base
Base (p. 137) class for most physics classes . . . 137
gazebo::math::Box
Mathematical representation of a box and related functions . . . 148
gazebo::physics::BoxShape
Box geometry primitive . . . 153
gazebo::common::BVHLoader
Handles loading BVH animation files . . . 156
gazebo::transport::CallbackHelper
A (p. 111) helper class to handle callbacks when messages arrive . . . 157
gazebo::transport::CallbackHelperT< M >
Callback helper Template . . . 160
gazebo::rendering::Camera
Basic camera sensor . . . 162
gazebo::sensors::CameraSensor
Basic camera sensor . . . 188

gazebo::rendering::CameraVisual
Basic camera visualization . . . 193
gazebo::common::ColladaLoader
Class used to load Collada mesh files . . . 194
gazebo::physics::Collision
Base (p. 137) class for all collision entities . . . 195
gazebo::physics::CollisionState
Store state information of a **physics::Collision** (p. 195) object . . . 205
gazebo::common::Color
Defines a color . . . 208
gazebo::rendering::COMVisual
Basic Center of Mass visualization . . . 219
gazebo::event::Connection
A (p. 111) class that encapsulates a connection . . . 221
gazebo::transport::Connection
Single TCP/IP connection manager . . . 222
gazebo::transport::ConnectionManager
Manager of connections . . . 229
gazebo::transport::ConnectionReadTask
233
gazebo::common::Console
Message, error, warning functionality . . . 235
gazebo::physics::Contact
A (p. 111) contact between two collisions . . . 236
gazebo::physics::ContactManager
Aggregates all the contact information generated by the collision detection engine . . . 239
gazebo::sensors::ContactSensor
Contact sensor . . . 241
gazebo::rendering::ContactVisual
Contact visualization . . . 246
gazebo::rendering::Conversions
Conversions (p. 247) **Conversions.hh** (p. 964) **rendering/Conversions.hh** (p. 964) . . . 247
sdf::Converter
Convert from one version of **SDF** (p. 695) to another . . . 249
gazebo::physics::CylinderShape
Cylinder collision . . . 250
gazebo::rendering::DepthCamera
Depth camera used to render depth data into an image buffer . . . 253
gazebo::sensors::DepthCameraSensor . . . 258
gazebo::util::DiagnosticManager
A (p. 111) diagnostic manager class . . . 261
gazebo::util::DiagnosticTimer
A (p. 111) timer designed for diagnostics . . . 264
gazebo::rendering::DynamicLines
Class for drawing lines that can change . . . 266
gazebo::rendering::DynamicRenderable
Abstract base class providing mechanisms for dynamically growing hardware buffers . . . 269
sdf::Element
SDF (p. 695) **Element** (p. 273) class . . . 273
gazebo::physics::Entity
Base (p. 137) class for all physics objects in Gazebo . . . 281
gazebo::event::Event
Base class for all events . . . 292

gazebo::rendering::Events
Base class for rendering events . . . 295
gazebo::event::Events
An **Event** (p. 292) class to get notifications for simulator events . . . 297
gazebo::event::EventT< T >
A (p. 111) class for event processing . . . 309
gazebo::common::Exception
Class for generating exceptions . . . 318
gazebo::rendering::FPSViewController
First Person Shooter style view controller . . . 320
urdf2gazebo::GazeboExtension . . . 323
google::protobuf::compiler::cpp::GazeboGenerator
Google protobuf message generator for **gazebo::msgs** (p. 89) . . . 323
gazebo::rendering::GpuLaser
GPU based laser distance sensor . . . 324
gazebo::sensors::GpuRaySensor . . . 328
gazebo::rendering::Grid
Displays a grid of cells, drawn with lines . . . 340
gazebo::physics::Gripper
A (p. 111) gripper abstraction . . . 344
gazebo::rendering::GUIOverlay
A (p. 111) class that creates a CEGUI overlay on a render window . . . 345
gazebo::rendering::GzTerrainMatGen . . . 349
gazebo::rendering::Heightmap
Rendering a terrain using heightmap information . . . 350
gazebo::physics::HeightmapShape
HeightmapShape (p. 352) collision shape builds a heightmap from an image . . . 352
gazebo::physics::Hinge2Joint< T >
A (p. 111) two axis hinge joint . . . 356
gazebo::physics::HingeJoint< T >
A (p. 111) single axis hinge joint . . . 358
gazebo::common::Image
Encapsulates an image . . . 360
gazebo::sensors::ImuSensor
An IMU sensor . . . 365
gazebo::physics::Inertial
A (p. 111) class for inertial information about a link . . . 369
gazebo::common::InternalError
Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs . . . 378
gazebo::transport::IOManager
Manages boost::asio IO . . . 380
gazebo::physics::Joint
Base (p. 137) class for all joints . . . 381
gazebo::physics::JointController
A (p. 111) class for manipulating **physics::Joint** (p. 381) . . . 399
gazebo::physics::JointState
Keeps track of state of a **physics::Joint** (p. 381) . . . 401
gazebo::rendering::JointVisual
Visualization for joints . . . 405
gazebo::physics::JointWrench
Wrench information from a joint . . . 407
gazebo::common::KeyFrame
A (p. 111) key frame in an animation . . . 409

gazebo::rendering::LaserVisual
Visualization for laser data . . . 410
gazebo::rendering::Light
A (p. 111) light source . . . 412
gazebo::physics::Link
Link (p. 418) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body . . . 418
gazebo::physics::LinkState
Store state information of a **physics::Link** (p. 418) object . . . 438
gazebo::common::LogPlay . . . 444
Logplay
Open and playback log files that were recorded using LogRecord . . . 446
gazebo::common::LogRecord
Addtogroup gazebo_common . . . 447
gazebo::Master
A (p. 111) ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication . . . 452
gazebo::common::Material
Encapsulates description of a material . . . 453
gazebo::math::Matrix3
A (p. 111) 3x3 matrix class . . . 463
gazebo::math::Matrix4
A (p. 111) 3x3 matrix class . . . 467
gazebo::common::Mesh
A (p. 111) 3D mesh . . . 475
gazebo::common::MeshCSG
Creates CSG meshes . . . 481
gazebo::common::MeshLoader
Base class for loading meshes . . . 483
gazebo::common::MeshManager
Maintains and manages all meshes . . . 484
gazebo::physics::Model
A (p. 111) model is a collection of links, joints, and plugins . . . 489
gazebo::common::ModelDatabase
Connects to model database, and has utility functions to find models . . . 502
gazebo::ModelPlugin
A (p. 111) plugin with access to **physics::Model** (p. 489) . . . 504
gazebo::physics::ModelState
Store state information of a **physics::Model** (p. 489) object . . . 505
gazebo::common::MouseEvent
Generic description of a mouse event . . . 512
gazebo::rendering::MovableText
Movable text . . . 515
gazebo::msgs::MsgFactory
A (p. 111) factory that generates protobuf message based on a string type . . . 521
gazebo::sensors::MultiCameraSensor
Multiple camera sensor . . . 523
gazebo::physics::MultiRayShape
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner . . . 527
gazebo::transport::Node
A (p. 111) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics . . . 535
gazebo::common::NodeAnimation
Node animation . . . 542

gazebo::common::NodeAssignment
Vertex to node weighted assignement for skeleton animation visualization . . . 546
gazebo::common::NodeTransform
NodeTransform (p. 547) **Skeleton.hh** (p. 1088) common/common.hh . . . 547
gazebo::common::NumericAnimation
A (p. 111) numeric animation . . . 552
gazebo::common::NumericKeyFrame
A (p. 111) keyframe for a **NumericAnimation** (p. 552) . . . 553
gazebo::rendering::OrbitViewController
Orbit view controller . . . 555
sdf::Param
A (p. 111) parameter class . . . 559
ParamT< T > . . . 565
sdf::ParamT< T >
Templatized parameter class . . . 565
gazebo::physics::PhysicsEngine
Base (p. 137) class for a physics engine . . . 568
gazebo::physics::PhysicsFactory
The physics factory instantiates different physics engines . . . 582
gazebo::common::PID
Generic **PID** (p. 583) controller class . . . 583
gazebo::math::Plane
A (p. 111) plane and related functions . . . 587
gazebo::physics::PlaneShape
Collision (p. 195) for an infinite plane . . . 590
sdf::Plugin . . . 593
gazebo::PluginT< T >
A (p. 111) class which all plugins must inherit from . . . 594
gazebo::math::Pose
Encapsulates a position and rotation in three space . . . 596
gazebo::common::PoseAnimation
A (p. 111) pose animation . . . 605
gazebo::common::PoseKeyFrame
A (p. 111) keyframe for a **PoseAnimation** (p. 605) . . . 607
gazebo::rendering::Projector
Projects a material onto surface, light a light projector . . . 609
gazebo::transport::Publication
A (p. 111) publication for a topic . . . 612
gazebo::transport::PublicationTransport
Transport/transport.hh . . . 616
gazebo::transport::Publisher
A (p. 111) publisher of messages on a topic . . . 618
gazebo::transport::PublishTask
622
gazebo::math::Quaternion
A (p. 111) quaternion class . . . 623
gazebo::math::Rand
Random number generator class . . . 637
gazebo::transport::RawCallbackHelper
Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher . . . 638
gazebo::sensors::RaySensor
Sensor (p. 698) with one or more rays . . . 641

gazebo::physics::RayShape
Base (p. 137) class for Ray collision geometry . . . 647
gazebo::rendering::RenderEngine
Adaptor to Ogre3d . . . 653
gazebo::sensors::RFIDSensor
Sensor (p. 698) class for RFID type of sensor . . . 658
gazebo::sensors::RFIDTag
RFIDTag (p. 660) to interact with RFIDTagSensors . . . 660
gazebo::rendering::RFIDTagVisual
Visualization for RFID tags sensor . . . 662
gazebo::rendering::RFIDVisual
Visualization for RFID sensor . . . 664
gazebo::physics::Road
For building a **Road** (p. 665) from SDF . . . 665
Road
Used to render a strip of road . . . 667
gazebo::rendering::Road2d . . . 667
gazebo::math::RotationSpline
Spline (p. 754) for rotations . . . 668
gazebo::rendering::RTShaderSystem
Implements **Ogre** (p. 106)'s Run-Time Shader system . . . 672
gazebo::rendering::Scene
Representation of an entire scene graph . . . 676
gazebo::physics::ScrewJoint< T >
A (p. 111) screw joint, which has both prismatic and rotational DOFs . . . 691
sdf::SDF
Base **SDF** (p. 695) class . . . 695
gazebo::rendering::SelectionObj
A (p. 111) graphical selection object . . . 696
gazebo::sensors::Sensor
Base class for sensors . . . 698
SensorFactor
The sensor factory; the class is just for namespacing purposes . . . 707
gazebo::sensors::SensorFactory . . . 707
gazebo::sensors::SensorManager
Class to manage and update all sensors . . . 709
gazebo::SensorPlugin
A (p. 111) plugin with access to physics::Sensor . . . 712
gazebo::Server . . . 714
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg
Keeping the CG shader for reference . . . 715
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL
Utility class to help with generating shaders for GLSL . . . 717
gazebo::physics::Shape
Base (p. 137) class for all shapes . . . 720
gazebo::sensors::SimTimeEvent
723
gazebo::sensors::SimTimeEventHandler
Monitors simulation time, and notifies conditions when a specified time has been reached . . . 723
SingletonT< T >
Singleton template class . . . 725
gazebo::common::Skeleton
A (p. 111) skeleton . . . 727

gazebo::common::SkeletonAnimation
Skeleton (p. 727) animation . . . 734
gazebo::common::SkeletonNode
A (p. 111) skeleton node . . . 738
gazebo::physics::SliderJoint< T >
A (p. 111) slider joint . . . 747
gazebo::rendering::GzTerrainMatGen::SM2Profile
Shader model 2 profile target . . . 749
gazebo::physics::SphereShape
Sphere collision shape . . . 751
gazebo::math::Spline
Splines . . . 754
gazebo::physics::State
State (p. 758) of an entity . . . 758
gazebo::common::STLLoader
Class used to load STL mesh files . . . 762
gazebo::common::SubMesh
A (p. 111) child mesh . . . 764
gazebo::transport::SubscribeOptions
Options for a subscription . . . 774
gazebo::transport::Subscriber
A (p. 111) subscriber to a topic . . . 776
gazebo::transport::SubscriptionTransport
Transport/transport.hh . . . 778
gazebo::physics::SurfaceParams
SurfaceParams (p. 780) defines various Surface contact parameters . . . 780
gazebo::common::SystemPaths
Functions to handle getting system paths, keeps track of: . . . 784
gazebo::SystemPlugin
A (p. 111) plugin loaded within the gzserver on startup . . . 789
gazebo::common::Time
A (p. 111) **Time** (p. 791) class, can be used to hold wall- or sim-time . . . 791
gazebo::common::Timer
A (p. 111) timer class, used to time things in real world walltime . . . 813
gazebo::transport::TopicManager
Manages topics and their subscriptions . . . 815
gazebo::physics::TrajectoryInfo . . . 821
gazebo::physics::TrimeshShape
Triangle mesh collision shape . . . 822
gazebo::physics::UniversalJoint< T >
A (p. 111) universal joint . . . 826
gazebo::common::UpdateInfo
Information for use in an update event . . . 827
urdf2gazebo::URDF2Gazebo . . . 828
gazebo::rendering::UserCamera
A (p. 111) camera used for user visualization of a scene . . . 830
gazebo::math::Vector2d
Generic double x, y vector . . . 838
gazebo::math::Vector2i
Generic integer x, y vector . . . 846
gazebo::math::Vector3
Generic vector containing 3 elements . . . 855
gazebo::math::Vector4
Double Generic x, y, z, w vector . . . 869

gazebo::common::Video
Handle video encoding and decoding using libavcodec . . . 878
gazebo::rendering::VideoVisual
A (p. 111) visual element that displays a video as a texture . . . 880
gazebo::rendering::ViewController
Base class for view controllers . . . 881
gazebo::rendering::Visual
A (p. 111) renderable object . . . 885
gazebo::VisualPlugin
A (p. 111) plugin loaded within the gzserver on startup . . . 905
gazebo::rendering::WindowManager
Class to mangage render windows . . . 906
gazebo::rendering::WireBox
Draws a wireframe box . . . 909
gazebo::physics::World
The world provides access to all other object within a simulated environment . . . 910
gazebo::WorldPlugin
A (p. 111) plugin with access to **physics::World** (p. 910) . . . 921
gazebo::physics::WorldState
Store state information of a **physics::World** (p. 910) object . . . 923

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Actor.hh . . . 929
Angle.hh . . . 930
Animation.hh . . . 933
ArrowVisual.hh . . . 934
Assert.hh . . . 935
AxisVisual.hh . . . 937
BallJoint.hh . . . 937
Base.hh . . . 938
Box.hh . . . 939
BoxShape.hh . . . 940
BVHLoader.hh . . . 941
CallbackHelper.hh . . . 943
Camera.hh . . . 945
CameraSensor.hh . . . 946
CameraVisual.hh . . . 946
cegui.h . . . 947
ColladaLoader.hh . . . 948
Collision.hh . . . 949
CollisionState.hh . . . 951
Color.hh . . . 952
Common.hh . . . 953
CommonTypes.hh . . . 954
COMVisual.hh . . . 956
Connection.hh . . . 956
ConnectionManager.hh . . . 958
Console.hh . . . 959
Contact.hh . . . 961
ContactManager.hh . . . 962
ContactSensor.hh . . . 963
ContactVisual.hh . . . 964
Conversions.hh . . . 964
Converter.hh . . . 965
CylinderShape.hh . . . 966
DepthCamera.hh . . . 967

DepthCameraSensor.hh . . . 968
Diagnostics.hh . . . 968
DynamicLines.hh . . . 969
DynamicRenderable.hh . . . 970
Entity.hh . . . 971
Event.hh . . . 973
Events.hh . . . 974
Exception.hh . . . 974
FPSViewController.hh . . . 976
gazebo.hh . . . 977
gazebo_core.hh . . . 978
GazeboGenerator.hh . . . 978
GpuLaser.hh . . . 979
GpuRaySensor.hh . . . 980
Grid.hh . . . 981
Gripper.hh . . . 981
GUIOverlay.hh . . . 983
Heightmap.hh . . . 983
HeightmapShape.hh . . . 984
Helpers.hh . . . 985
Hinge2Joint.hh . . . 987
HingeJoint.hh . . . 989
Image.hh . . . 990
ImuSensor.hh . . . 991
Inertial.hh . . . 991
IOManager.hh . . . 992
Joint.hh . . . 994
JointController.hh . . . 995
JointState.hh . . . 996
JointVisual.hh . . . 997
JointWrench.hh . . . 998
KeyFrame.hh . . . 999
LaserVisual.hh . . . 1000
Light.hh . . . 1001
Link.hh . . . 1002
LinkState.hh . . . 1003
LogPlay.hh . . . 1005
LogRecord.hh . . . 1006
mainpage.html . . . 1008
MapShape.hh . . . 1008
Master.hh . . . 1009
common/Material.hh . . . 1010
rendering/Material.hh . . . 1012
MathTypes.hh
Forward declarations for the math classes . . . 1012
Matrix3.hh . . . 1013
Matrix4.hh . . . 1013
Mesh.hh . . . 1014
MeshCSG.hh . . . 1016
MeshLoader.hh . . . 1017
MeshManager.hh . . . 1018
Model.hh . . . 1019
ModelDatabase.hh . . . 1021
ModelState.hh . . . 1022

MouseEvent.hh . . . 1024
MovableText.hh . . . 1025
MsgFactory.hh . . . 1025
msgs.hh . . . 1026
MultiCameraSensor.hh . . . 1029
MultiRayShape.hh . . . 1030
Node.hh . . . 1031
ogre_gazebo.h . . . 1032
OrbitViewController.hh . . . 1033
Param.hh . . . 1034
parser.hh . . . 1035
parser_urdf.hh . . . 1036
Physics.hh . . . 1037
PhysicsEngine.hh . . . 1039
PhysicsFactory.hh . . . 1040
PhysicsTypes.hh
Default namespace for gazebo . . . 1042
PID.hh . . . 1044
Plane.hh . . . 1045
PlaneShape.hh . . . 1046
common/Plugin.hh . . . 1048
sdf/interface/Plugin.hh . . . 1052
Pose.hh . . . 1052
Projector.hh . . . 1053
Publication.hh . . . 1054
PublicationTransport.hh . . . 1055
Publisher.hh . . . 1057
Quaternion.hh . . . 1059
Rand.hh . . . 1060
RaySensor.hh . . . 1061
RayShape.hh . . . 1062
RenderEngine.hh . . . 1063
RenderEvents.hh . . . 1064
Rendering.hh . . . 1065
RenderTypes.hh . . . 1066
RFIDSensor.hh . . . 1068
RFIDTag.hh . . . 1068
RFIDTagVisual.hh . . . 1069
RFIDVisual.hh . . . 1070
Road.hh . . . 1070
Road2d.hh . . . 1072
RotationSpline.hh . . . 1072
RTShaderSystem.hh . . . 1073
Scene.hh . . . 1074
ScrewJoint.hh . . . 1075
sdf.hh . . . 1076
SDF.hh . . . 1077
SelectionObj.hh . . . 1078
Sensor.hh . . . 1079
SensorFactory.hh . . . 1080
SensorManager.hh . . . 1081
Sensors.hh . . . 1082
SensorTypes.hh
Forward declarations and typedefs for sensors . . . 1084

Server.hh . . . 1085
Shape.hh . . . 1086
SingletonT.hh . . . 1088
Skeleton.hh . . . 1088
SkeletonAnimation.hh . . . 1090
SliderJoint.hh . . . 1091
SphereShape.hh . . . 1093
Spline.hh . . . 1094
State.hh . . . 1095
STLLoader.hh . . . 1096
SubscribeOptions.hh . . . 1098
Subscriber.hh . . . 1099
SubscriptionTransport.hh . . . 1101
SurfaceParams.hh . . . 1102
SystemPaths.hh . . . 1104
Time.hh . . . 1105
Timer.hh . . . 1106
TopicManager.hh . . . 1107
Transport.hh . . . 1109
TransportTypes.hh
Forward declarations for transport . . . 1111
TrimeshShape.hh . . . 1113
UniversalJoint.hh . . . 1114
UpdateInfo.hh . . . 1115
UserCamera.hh . . . 1116
UtilTypes.hh . . . 1116
Vector2d.hh . . . 1117
Vector2i.hh . . . 1118
Vector3.hh . . . 1119
Vector4.hh . . . 1120
Video.hh . . . 1122
VideoVisual.hh . . . 1123
ViewController.hh . . . 1124
Visual.hh . . . 1125
WindowManager.hh . . . 1126
WireBox.hh . . . 1126
World.hh . . . 1127
WorldState.hh . . . 1129

Chapter 8

Module Documentation

8.1 Common

Files

- file **CommonTypes.hh**

Namespaces

- namespace **gazebo::common**

  Common namespace.

Classes

- class **gazebo::common::Animation**

  Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.
- class **gazebo::common::AssertionInternalError**

  Class for generating Exceptions which come from gazebo assertions.
- class **gazebo::common::BVHLoader**

  Handles loading BVH animation files.
- class **gazebo::common::ColladaLoader**

  Class used to load Collada mesh files.
- class **gazebo::common::Color**

  Defines a color.
- class **gazebo::common::Console**

  Message, error, warning functionality.
- class **gazebo::common::Exception**

  Class for generating exceptions.
- class **gazebo::common::Image**

  Encapsulates an image.
- class **gazebo::common::InternalError**

  Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.
- class **gazebo::common::KeyFrame**

A *(p. 111) key frame in an animation.*

- class **gazebo::common::Material**

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

  A *(p. 111) 3D mesh.*
- class **gazebo::common::MeshCSG**

  Creates CSG meshes.
- class **gazebo::common::MeshLoader**

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

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

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

  A *(p. 111) plugin with access to* ***physics::Model*** *(p. 489).*
- class **gazebo::common::MouseEvent**

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

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

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

  NodeTransform *(p. 547)* ***Skeleton.hh*** *(p. 1088) common/common.hh*
- class **gazebo::common::NumericAnimation**

  A *(p. 111) numeric animation.*
- class **gazebo::common::NumericKeyFrame**

  A *(p. 111) keyframe for a* ***NumericAnimation*** *(p. 552).*
- class **gazebo::common::PID**

  Generic ***PID*** *(p. 583) controller class.*
- class **gazebo::PluginT< T >**

  A *(p. 111) class which all plugins must inherit from.*
- class **gazebo::common::PoseAnimation**

  A *(p. 111) pose animation.*
- class **gazebo::common::PoseKeyFrame**

  A *(p. 111) keyframe for a* ***PoseAnimation*** *(p. 605).*
- class **gazebo::SensorPlugin**

  A *(p. 111) plugin with access to physics::Sensor.*
- class **SingletonT< T >**

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

  A *(p. 111) skeleton.*
- class **gazebo::common::SkeletonAnimation**

  Skeleton *(p. 727) animation.*
- class **gazebo::common::SkeletonNode**

  A *(p. 111) skeleton node.*
- class **gazebo::common::STLLoader**

  Class used to load STL mesh files.

- class **gazebo::common::SubMesh**

  A *(*p. *111) child mesh.*
- class **gazebo::common::SystemPaths**

  Functions to handle getting system paths, keeps track of:
- class **gazebo::SystemPlugin**

  A *(*p. *111) plugin loaded within the gzserver on startup.*
- class **gazebo::common::Time**

  A *(*p. *111)* ***Time*** *(*p. *791) class, can be used to hold wall- or sim-time.*
- class **gazebo::common::Timer**

  A *(*p. *111) timer class, used to time things in real world walltime.*
- class **gazebo::common::Video**

  Handle video encoding and decoding using libavcodec.
- class **gazebo::VisualPlugin**

  A *(*p. *111) plugin loaded within the gzserver on startup.*
- class **gazebo::WorldPlugin**

  A *(*p. *111) plugin with access to* ***physics::World*** *(*p. *910).*

Macros

- #define **gzclr_end** "\033[0m"

  End marker.
- #define **gzclr_start**(clr) "\033[1;33m"

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

  Output a debug message.
- #define **gzerr**

  Output an error message.
- #define **gzlog** (**gazebo::common::Console::Instance**()->Log())

  Output a message to a log file.
- #define **gzmsg** (**gazebo::common::Console::Instance**()->ColorMsg("Msg", 32))

  Output a message.
- #define **gzthrow**(msg)

  This macro logs an error to the throw stream and throws an exception that contains the file name and line number.
- #define **gzwarn**

  Output a warning message.

Enumerations

- enum **gazebo::PluginType** {
  gazebo::WORLD_PLUGIN, **gazebo::MODEL_PLUGIN**, **gazebo::SENSOR_PLUGIN**, **gazebo::SYSTEM_PLUGIN**,
  gazebo::VISUAL_PLUGIN }

  Used to specify the type of plugin.

Functions

- **gazebo::common::Console::NullStream::NullStream** ()

  constructor
- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)

  *add path prefix to **common::SystemPaths** (p. 784)*
- std::ostream & **gazebo::common::Console::ColorErr** (const std::string &_lbl, const std::string &_file, unsigned int _line, int _color)

  Use this to output an error to the terminal.
- std::ostream & **gazebo::common::Console::ColorMsg** (const std::string &_lbl, int _color)

  Use this to output a colored message to the terminal.
- void **gazebo::common::ModelDatabase::DownloadDependencies** (const std::string &_path)

  Download all dependencies for a give model path.
- std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath=true)

  *search for file in **common::SystemPaths** (p. 784)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)

  *search for a file in **common::SystemPaths** (p. 784)*
- std::string **gazebo::common::ModelDatabase::GetDBConfig** (const std::string &_uri)

  Return the database.config file as a string.
- std::string **gazebo::common::ModelDatabase::GetManifest** (const std::string &_uri) **GAZEBO_DEPRECATED**(1.5)

  Deprecated.
- std::string **gazebo::common::ModelDatabase::GetModelConfig** (const std::string &_uri)

  Return the model.config file as a string.
- std::string **gazebo::common::ModelDatabase::GetModelFile** (const std::string &_uri)

  Get a model's SDF file based on a URI.
- std::string **gazebo::common::ModelDatabase::GetModelName** (const std::string &_uri)

  Get the name of a model based on a URI.
- std::string **gazebo::common::ModelDatabase::GetModelPath** (const std::string &_uri, bool _forceDownload=false)

  Get the local path to a model.
- std::map< std::string, std::string > **gazebo::common::ModelDatabase::GetModels** ()

  Returns the dictionary of all the model names.
- void **gazebo::common::ModelDatabase::GetModels** (boost::function< void(const std::map< std::string, std::string > &)> _func)

  Get the dictionary of all model names via a callback.
- std::string **gazebo::common::ModelDatabase::GetURI** ()

  Returns the the global model database URI.
- bool **gazebo::common::ModelDatabase::HasModel** (const std::string &_modelName)

  Returns true if the model exists on the database.
- void **gazebo::common::Console::Init** (const std::string &_logFilename)

  Load the message parameters.
- bool **gazebo::common::Console::IsInitialized** () const

  Return true if Init has been called.
- std::ofstream & **gazebo::common::Console::Log** ()

  Use this to output a colored message to the terminal.
- void **gazebo::common::Console::SetQuiet** (bool _q)

  Set quiet output.

Variables

- static std::string **gazebo::common::PixelFormatNames** [ ]

  String names for the pixel formats.

8.1.1 Detailed Description

8.1.2 Macro Definition Documentation

8.1.2.1 #define gzclr_end "\033[0m"

End marker.

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

Start marker.

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

Output a debug message.

8.1.2.4 #define gzerr

Value:

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

Output an error message.

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

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

Output a message to a log file.

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

Output a message.

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

8.1.2.7 #define gzthrow(*msg*)

Value:

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

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

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

8.1.2.8 #define gzwarn

Value:

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

Output a warning message.

Referenced by gazebo::physics::Joint::GetLowerLimit(), and gazebo::physics::Joint::GetUpperLimit().

8.1.3 Enumeration Type Documentation

8.1.3.1 enum gazebo::PluginType

Used to specify the type of plugin.

Enumerator:

WORLD_PLUGIN **A** (p. 111) World plugin.

MODEL_PLUGIN **A** (p. 111) Model plugin.

SENSOR_PLUGIN **A** (p. 111) Sensor plugin.

SYSTEM_PLUGIN **A** (p. 111) System plugin.

VISUAL_PLUGIN **A** (p. 111) Visual plugin.

8.1.4 Function Documentation

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

constructor

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

add path prefix to **common::SystemPaths** (p. 784)

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

Use this to output an error to the terminal.

Parameters

in	*_lbl*	Text label
in	*_file*	File containing the error
in	*_line*	Line containing the error
in	*_color*	**Color** (p. 208) to make the label

Returns

Reference to an output stream

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

Use this to output a colored message to the terminal.

Parameters

in	*_lbl*	Text label
in	*_color*	**Color** (p. 208) to make the label

Returns

Reference to an output stream

8.1.4.5 void gazebo::common::ModelDatabase::DownloadDependencies (const std::string & *_path*)

Download all dependencies for a give model path.

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

Parameters

in	*_path*	Path to a model.

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

search for file in **common::SystemPaths** (p. 784)

Parameters

in	*_file*	Name of the file to find.
in	*_searchLocal-Path*	True to search in the current working directory.

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

search for a file in **common::SystemPaths** (p. 784)

Parameters

in	*_file*	the file name to look for

Returns

The path containing the file

8.1.4.8 std::string gazebo::common::ModelDatabase::GetDBConfig (const std::string & _uri)

Return the database.config file as a string.

Returns

The database config file from the model database.

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

Deprecated.

See Also

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

8.1.4.10 std::string gazebo::common::ModelDatabase::GetModelConfig (const std::string & _uri)

Return the model.config file as a string.

Returns

The model config file from the model database.

8.1.4.11 std::string gazebo::common::ModelDatabase::GetModelFile (const std::string & _uri)

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

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

Parameters

in	*_uri*	The URI of the model

Returns

The full path and filename to the SDF file

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

Get the name of a model based on a URI.

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

Parameters

in	*_uri*	the model uri

Returns

the model's name.

8.1.4.13 std::string gazebo::common::ModelDatabase::GetModelPath (const std::string & _uri, bool _forceDownload = false)

Get the local path to a model.

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

Parameters

in	*_uri*	the model uri
in	*_forceDownload*	True to skip searching local paths.

Returns

path to a model directory

8.1.4.14 std::map<std::string, std::string> gazebo::common::ModelDatabase::GetModels ( )

Returns the dictionary of all the model names.

This is a blocking call. Which means it will wait for the **ModelDatabase** (p. 502) to download the model list.

Returns

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

8.1.4.15 void gazebo::common::ModelDatabase::GetModels (boost::function< void(const std::map< std::string, std::string > &)> _func)

Get the dictionary of all model names via a callback.

This is the non-blocking version of **ModelDatabase::GetModels** (p. 35)

Parameters

in	*_func*	Callback function that receives the list of models.

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

Returns the the global model database URI.

Returns

the URI.

8.1.4.17 bool gazebo::common::ModelDatabase::HasModel (const std::string & _modelName)

Returns true if the model exists on the database.

Parameters

in	*_modelName*	URI of the model (eg: model://my_model_name).

Returns

True if the model was found.

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

Load the message parameters.

8.1.4.19 bool gazebo::common::Console::IsInitialized ( ) const

Return true if Init has been called.

Returns

True is initialized.

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

Use this to output a colored message to the terminal.

Parameters

in	*_lbl*	Text label

Returns

Reference to an output stream

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

Set quiet output.

Parameters

in	*q*	True to prevent warning

8.1.5 Variable Documentation

8.1.5.1 std::string gazebo::common::PixelFormatNames[] [static]

Initial value:

```
=
    {
      "UNKNOWN_PIXEL_FORMAT",
      "L_INT8",
      "L_INT16",
```

```
  "RGB_INT8",
  "RGBA_INT8",
  "BGRA_INT8",
  "RGB_INT16",
  "RGB_INT32",
  "BGR_INT8",
  "BGR_INT16",
  "BGR_INT32",
  "R_FLOAT16",
  "RGB_FLOAT16",
  "R_FLOAT32",
  "RGB_FLOAT32",
  "BAYER_RGGB8",
  "BAYER_RGGR8",
  "BAYER_GBRG8",
  "BAYER_GRBG8"
}
```

String names for the pixel formats.

See Also

Image::PixelFormat (p. 361).

8.2 Events

Namespaces

- namespace **gazebo::event**

  Event *(p. 292) namespace.*

Classes

- class **gazebo::event::Connection**

  A *(p. 111) class that encapsulates a connection.*
- class **gazebo::event::Event**

  Base class for all events.
- class **gazebo::event::Events**

  An ***Event*** *(p. 292) class to get notifications for simulator events.*
- class **gazebo::event::EventT< T >**

  A *(p. 111) class for event processing.*

Functions

- virtual **gazebo::event::EventT< T >::~EventT** ()

  Destructor.
- ConnectionPtr **gazebo::event::EventT< T >::Connect** (const boost::function< T > &_subscriber)

  Connect a callback to this event.
- unsigned int **gazebo::event::EventT< T >::ConnectionCount** () const

  Get the number of connections.
- virtual void **gazebo::event::EventT< T >::Disconnect** (ConnectionPtr _c)

  Disconnect a callback to this event.
- virtual void **gazebo::event::EventT< T >::Disconnect** (int _id)

  Disconnect a callback to this event.

8.2.1 Detailed Description

8.2.2 Function Documentation

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

Destructor.

Destructor. Deletes all the associated connections.

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

Connect a callback to this event.

Adds a connection.

Parameters

in	*_subscriber*	Pointer to a callback function

Returns

A (p. 111) **Connection** (p. 221) object, which will automatically call Disconnect when it goes out of scope

Parameters

in	*_subscriber*	the subscriber to connect

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

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

Get the number of connections.

Returns

Number of connection to this **Event** (p. 292).

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

Disconnect a callback to this event.

Removes a connection.

Parameters

in	*_c*	The connection to disconnect
in	*_c*	the connection

Implements **gazebo::event::Event** (p. 294).

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

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

NewImageFrame(), gazebo::rendering::GpuLaser::DisconnectNewLaserFrame(), gazebo::physics::MultiRayShape-::DisconnectNewLaserScans(), gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud(), gazebo::event-::Events::DisconnectPause(), gazebo::event::Events::DisconnectPostRender(), gazebo::event::Events::Disconnect-PreRender(), gazebo::rendering::Events::DisconnectRemoveScene(), gazebo::event::Events::DisconnectRender(), gazebo::event::Events::DisconnectSetSelectedEntity(), gazebo::transport::Connection::DisconnectShutdown(), gazebo-::event::Events::DisconnectStep(), gazebo::event::Events::DisconnectStop(), gazebo::sensors::Sensor::Disconnect-Updated(), gazebo::event::Events::DisconnectWorldCreated(), and gazebo::event::Events::DisconnectWorldUpdate-End().

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

Disconnect a callback to this event.

Removes a connection.

Parameters

in	*_id*	The id of the connection to disconnect
in	*_id*	the connection index

Implements **gazebo::event::Event** (p. 294).

8.3 Classes for physics and dynamics

Files

- file **PhysicsTypes.hh**

  default namespace for gazebo

Namespaces

- namespace **gazebo::physics**

  namespace for physics

Classes

- class **gazebo::physics::Actor**

  Actor *(*p. *111) class enables GPU based mesh model / skeleton scriptable animation.*
- class **gazebo::physics::BallJoint< T >**

  Base *(*p. *137) class for a ball joint.*
- class **gazebo::physics::Base**

  Base *(*p. *137) class for most physics classes.*
- class **gazebo::physics::BoxShape**

  Box geometry primitive.
- class **gazebo::physics::Collision**

  Base *(*p. *137) class for all collision entities.*
- class **gazebo::physics::CollisionState**

  Store state information of a ***physics::Collision*** *(*p. *195) object.*
- class **gazebo::physics::Contact**

  A *(*p. *111) contact between two collisions.*
- class **gazebo::physics::ContactManager**

  Aggregates all the contact information generated by the collision detection engine.
- class **gazebo::physics::CylinderShape**

  Cylinder collision.
- class **gazebo::physics::Entity**

  Base *(*p. *137) class for all physics objects in Gazebo.*
- class **gazebo::physics::Gripper**

  A *(*p. *111) gripper abstraction.*
- class **gazebo::physics::HeightmapShape**

  HeightmapShape *(*p. *352) collision shape builds a heightmap from an image.*
- class **gazebo::physics::Hinge2Joint< T >**

  A *(*p. *111) two axis hinge joint.*
- class **gazebo::physics::HingeJoint< T >**

  A *(*p. *111) single axis hinge joint.*
- class **gazebo::physics::Inertial**

  A *(*p. *111) class for inertial information about a link.*
- class **gazebo::physics::Joint**

  Base *(*p. *137) class for all joints.*
- class **gazebo::physics::JointController**

A *(p. 111) class for manipulating* ***physics::Joint*** *(p. 381).*

- class **gazebo::physics::JointState**

  keeps track of state of a ***physics::Joint*** *(p. 381)*

- class **gazebo::physics::JointWrench**

  Wrench information from a joint.

- class **gazebo::physics::Link**

  Link *(p. 418) 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. 418) object.*

- class **Logplay**

  Open and playback log files that were recorded using LogRecord.

- class **gazebo::common::LogPlay**
- class **gazebo::physics::Model**

  A *(p. 111) model is a collection of links, joints, and plugins.*

- class **gazebo::physics::ModelState**

  Store state information of a ***physics::Model*** *(p. 489) 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. 137) class for a physics engine.*

- class **gazebo::physics::PhysicsFactory**

  The physics factory instantiates different physics engines.

- class **gazebo::physics::PlaneShape**

  Collision *(p. 195) for an infinite plane.*

- class **gazebo::physics::RayShape**

  Base *(p. 137) class for Ray collision geometry.*

- class **gazebo::physics::Road**

  for building a ***Road*** *(p. 665) from SDF*

- class **gazebo::physics::ScrewJoint< T >**

  A *(p. 111) screw joint, which has both prismatic and rotational DOFs.*

- class **gazebo::physics::Shape**

  Base *(p. 137) class for all shapes.*

- class **gazebo::physics::SliderJoint< T >**

  A *(p. 111) slider joint.*

- class **gazebo::physics::SphereShape**

  Sphere collision shape.

- class **gazebo::physics::State**

  State *(p. 758) of an entity.*

- class **gazebo::physics::SurfaceParams**

  SurfaceParams *(p. 780) defines various Surface contact parameters.*

- class **gazebo::physics::TrimeshShape**

  Triangle mesh collision shape.

- class **gazebo::physics::UniversalJoint< T >**

  A *(p. 111) 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. 910) 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. 77).*
- 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. 789)'s and call **gazebo::transport::init** (p. 78).*
- void **gazebo::physics::load_world** (WorldPtr _world, **sdf::ElementPtr** _sdf)

  *Load world from **sdf::Element** (p. 273) pointer.*
- void **gazebo::physics::load_worlds** (**sdf::ElementPtr** _sdf)

  *load multiple worlds from single **sdf::Element** (p. 273) pointer*
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)

  *Pause world by calling **World::SetPaused** (p. 920).*
- 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. 920) 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. 921) given a pointer to it.*
- void **gazebo::physics::stop_worlds** ()

  stop multiple worlds stored in static variable gazebo::g_worlds

Variables

- static std::string **gazebo::physics::EntityTypename** [ ]

  String names for the different entity types.

8.3.1 Detailed Description

8.3.2 Macro Definition Documentation

8.3.2.1 #define GZ_REGISTER_PHYSICS_ENGINE(*name, classname*)

Value:

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

Static physics registration macro.

Use this macro to register physics engine with the server.

Parameters

in	*name*	Physics type name, as it appears in the world file.
in	*classname*	C++ class name for the physics engine.

8.3.3 Typedef Documentation

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

8.3.4 Function Documentation

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

Create a world given a name.

Parameters

in	*_name*	Name of the world to create.

Returns

Pointer to the new world.

8.3.4.2 bool gazebo::physics::fini ( )

Finalize transport by calling **gazebo::transport::fini** (p. 77).

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

Returns a pointer to a world by name.

Parameters

in	*_name*	Name of the world to get.

Returns

Pointer to the world.

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

Init world given a pointer to it.

Parameters

in	*_world*	**World** (p. 910) to initialize.

8.3.4.5 void gazebo::physics::init_worlds ( )

initialize multiple worlds stored in static variable gazebo::g_worlds

8.3.4.6 bool gazebo::physics::load ( )

Setup **gazebo::SystemPlugin** (p. 789)'s and call **gazebo::transport::init** (p. 78).

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

Load world from **sdf::Element** (p. 273) pointer.

Parameters

in	*_world*	Pointer to a world.
in	*_sdf*	SDF values to load from.

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

load multiple worlds from single **sdf::Element** (p. 273) pointer

Parameters

in	*_sdf*	SDF values used to create worlds.

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

Pause world by calling **World::SetPaused** (p. 920).

Parameters

in	*_world*	**World** (p. 910) to pause or unpause.
in	*_pause*	True to pause, False to unpause.

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

pause multiple worlds stored in static variable gazebo::g_worlds

Parameters

in	*_pause*	True to pause, False to unpause.

8.3.4.11 void gazebo::physics::remove_worlds ( )

remove multiple worlds stored in static variable gazebo::g_worlds

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

Run world by calling **World::Run()** (p. 920) given a pointer to it.

Parameters

in	*_world*	**World** (p. 910) to run.

8.3.4.13 void gazebo::physics::run_worlds ( )

run multiple worlds stored in static variable gazebo::g_worlds

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

Stop world by calling **World::Stop()** (p. 921) given a pointer to it.

Parameters

in	*_world*	**World** (p. 910) to stop.

8.3.4.15 void gazebo::physics::stop_worlds ( )

stop multiple worlds stored in static variable gazebo::g_worlds

8.3.5 Variable Documentation

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

Initial value:

```
= {
      "common",
      "entity",
      "model",
      "actor",
      "link",
      "collision",
      "light",
      "visual",
```

```
    "joint",
    "ball",
    "hinge2",
    "hinge",
    "slider",
    "universal",
    "shape",
    "box",
    "cylinder",
    "heightmap",
    "map",
    "multiray",
    "ray",
    "plane",
    "sphere",
    "trimesh"
  }
```

String names for the different entity types.

8.4 Math

A (p. 111) set of classes that encapsulate math related properties and functions.

Files

- file **MathTypes.hh**

  Forward declarations for the math classes.

Namespaces

- namespace **gazebo::math**

  Math namespace.

Classes

- class **gazebo::math::Angle**

  An angle and related functions.
- class **gazebo::math::Box**

  Mathematical representation of a box and related functions.
- class **gazebo::math::Matrix3**

  A *(p. 111) 3x3 matrix class.*
- class **gazebo::math::Matrix4**

  A *(p. 111) 3x3 matrix class.*
- class **gazebo::math::Plane**

  A *(p. 111) plane and related functions.*
- class **gazebo::math::Pose**

  Encapsulates a position and rotation in three space.
- class **gazebo::math::Quaternion**

  A *(p. 111) quaternion class.*
- class **gazebo::math::Rand**

  Random number generator class.
- class **gazebo::math::RotationSpline**

  Spline *(p. 754) 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. 855) class represents the generic vector containing 3 elements.*
- class **gazebo::math::Vector4**

  double Generic x, y, z, w vector

Functions

- template<typename T >
  T **gazebo::math::clamp** (T _v, T _min, T _max)

  Simple clamping function.
- template<typename T >
  bool **gazebo::math::equal** (const T &_a, const T &_b, const T &_epsilon=1e-6)

  check if two values are equal, within a tolerance
- bool **gazebo::math::isnan** (float _v)

  check if a float is NaN
- bool **gazebo::math::isnan** (double _v)

  check if a double is NaN
- bool **gazebo::math::isPowerOfTwo** (unsigned int _x)

  is this a power of 2?
- template<typename T >
  T **gazebo::math::max** (const std::vector< T > &_values)

  get the maximum value of vector of values
- template<typename T >
  T **gazebo::math::mean** (const std::vector< T > &_values)

  get mean of vector of values
- template<typename T >
  T **gazebo::math::min** (const std::vector< T > &_values)

  get the minimum value of vector of values
- double **gazebo::math::parseFloat** (const std::string &_input)

  parse string into float
- int **gazebo::math::parseInt** (const std::string &_input)

  parse string into an integer
- template<typename T >
  T **gazebo::math::precision** (const T &_a, const unsigned int &_precision)

  get value at a specified precision
- template<typename T >
  T **gazebo::math::variance** (const std::vector< T > &_values)

  get variance of vector of values

Variables

- static const double **gazebo::math::NAN_D** = std::numeric_limits<double>::quiet_NaN()

  Returns the representation of a quiet not a number (NAN)
- static const int **gazebo::math::NAN_I** = std::numeric_limits<int>::quiet_NaN()

  Returns the representation of a quiet not a number (NAN)

8.4.1 Detailed Description

A (p. 111) set of classes that encapsulate math related properties and functions.

8.4.2 Function Documentation

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

Simple clamping function.

Parameters

in	*_v*	value
in	*_min*	minimum
in	*_max*	maximum

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

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

check if two values are equal, within a tolerance

Parameters

in	*_a*	the first value
in	*_b*	the second value
in	*_epsilon*	the tolerance

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

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

check if a float is NaN

Parameters

in	*_v*	the value

Returns

true if _v is not a number, false otherwise

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

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

check if a double is NaN

Parameters

in	*_v*	the value

Returns

true if _v is not a number, false otherwise

References gazebo::math::isnan().

8.4.2.5 bool gazebo::math::isPowerOfTwo (unsigned int *_x*) [inline]

is this a power of 2?

Parameters

in	*_x*	the number

Returns

true if _x is a power of 2, false otherwise

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

get the maximum value of vector of values

Parameters

in	*_values*	the vector of values

Returns

maximum

References gazebo::math::min().

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

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

get mean of vector of values

Parameters

in	*_values*	the vector of values

Returns

the mean

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

get the minimum value of vector of values

Parameters

in	*_values*	the vector of values

Returns

minimum

References gazebo::math::max().

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

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

parse string into float

Parameters

_input	the string

Returns

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

References gazebo::math::NAN_D.

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

parse string into an integer

Parameters

in	*_input*	the string

Returns

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

References gazebo::math::NAN_I.

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

get value at a specified precision

Parameters

in	*_a*	the number
in	*_precision*	the precision

Returns

the value for the specified precision

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

get variance of vector of values

Parameters

in	*_values*	the vector of values

Returns

the squared deviation

8.4.3 Variable Documentation

8.4.3.1 const double gazebo::math::NAN_D = std::numeric_limits<double>::quiet_NaN() [static]

Returns the representation of a quiet not a number (NAN)

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

8.4.3.2 const int gazebo::math::NAN_I = std::numeric_limits<int>::quiet_NaN() [static]

Returns the representation of a quiet not a number (NAN)

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

8.5 Messages

All messages and helper functions.

Namespaces

- namespace **gazebo::msgs**

  Messages namespace.

Classes

- class **google::protobuf::compiler::cpp::GazeboGenerator**

  *Google protobuf message generator for **gazebo::msgs** (p. 89).*
- class **gazebo::msgs::MsgFactory**

  ***A** (p. 111) factory that generates protobuf message based on a string type.*

Macros

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

  Static message registration macro.

Functions

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

  *Convert a **math::Vector3** (p. 855) to a msgs::Vector3d.*
- msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)

  *Convert a **math::Quaternion** (p. 623) to a msgs::Quaternion.*
- msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)

  *Convert a **math::Pose** (p. 596) to a msgs::Pose.*
- msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)

  *Convert a **common::Color** (p. 208) to a msgs::Color.*
- msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)

  *Convert a **common::Time** (p. 791) to a msgs::Time.*
- msgs::PlaneGeom **gazebo::msgs::Convert** (const math::Plane &_p)

  *Convert a **math::Plane** (p. 587) 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. 623).*
- math::Pose **gazebo::msgs::Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose** (p. 596).*
- common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color** (p. 208).*
- common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time** (p. 791).*
- math::Plane **gazebo::msgs::Convert** (const msgs::PlaneGeom &_p)

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

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

  Create a request message.
- msgs::Fog **gazebo::msgs::FogFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Fog from a fog SDF element.
- msgs::Geometry **gazebo::msgs::GeometryFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Geometry from a geometry SDF element.
- msgs::Header * **gazebo::msgs::GetHeader** (google::protobuf::Message &_message)

  Get the header from a protobuf message.
- msgs::GUI **gazebo::msgs::GUIFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::GUI from a GUI SDF element.
- void **gazebo::msgs::Init** (google::protobuf::Message &_message, const std::string &_id="")

  Initialize a message.
- msgs::Light **gazebo::msgs::LightFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Light from a light SDF element.
- msgs::MeshGeom **gazebo::msgs::MeshFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::MeshGeom from a mesh SDF element.
- msgs::Scene **gazebo::msgs::SceneFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Scene from a scene SDF element.
- void **gazebo::msgs::Set** (common::Image &_img, const msgs::Image &_msg)

  *Convert a msgs::Image to a **common::Image** (p. 360).*
- void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)

  *Set a msgs::Image from a **common::Image** (p. 360).*
- void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 855).*
- void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 855).*
- void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 623).*
- void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 596).*
- void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)

  *Set a msgs::Color from a **common::Color** (p. 208).*
- void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)

  *Set a msgs::Time from a **common::Time** (p. 791).*
- void **gazebo::msgs::Set** (msgs::PlaneGeom *_p, const math::Plane &_v)

  *Set a msgs::Plane from a **math::Plane** (p. 587).*
- void **gazebo::msgs::Stamp** (msgs::Header *_header)

  Time stamp a header.
- void **gazebo::msgs::Stamp** (msgs::Time *_time)

  Set the time in a time message.
- msgs::TrackVisual **gazebo::msgs::TrackVisualFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::TrackVisual from a track visual SDF element.
- msgs::Visual **gazebo::msgs::VisualFromSDF** (**sdf::ElementPtr** _sdf)

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

8.5.1 Detailed Description

All messages and helper functions.

8.5.2 Macro Definition Documentation

8.5.2.1 #define GZ_REGISTER_STATIC_MSG(_msgtype, _classname)

Value:

```
google::protobuf::Message* New##_classname() \
    { \
      return gazebo::msgs::_classname*(\
          new gazebo::msgs::_classname); \
    } \
    class Msg##_classname \
    { \
      public: Msg##_classname() \
      { \
        gazebo::msgs::MsgFactory::RegisterMsg(_msgtype, New##_classname);\
      } \
    }; \
    static Msg##_classname GzMsgInitializer;
```

Static message registration macro.

Use this macro to register messages.

Parameters

in	*_msgtype*	Message type name.
in	*_classname*	Class name for message.

8.5.3 Function Documentation

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

Convert a **math::Vector3** (p. 855) to a msgs::Vector3d.

Parameters

in	*_v*	The vector to convert

Returns

A (p. 111) msgs::Vector3d object

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

Convert a **math::Quaternion** (p. 623) to a msgs::Quaternion.

Parameters

in	*_q*	The quaternion to convert

Returns

A (p. 111) msgs::Quaternion object

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

Convert a **math::Pose** (p. 596) to a msgs::Pose.

Parameters

in	_p	The pose to convert

Returns

A (p. 111) msgs::Pose object

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

Convert a **common::Color** (p. 208) to a msgs::Color.

Parameters

in	_c	The color to convert

Returns

A (p. 111) msgs::Color object

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

Convert a **common::Time** (p. 791) to a msgs::Time.

Parameters

in	_t	The time to convert

Returns

A (p. 111) msgs::Time object

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

Convert a **math::Plane** (p. 587) to a msgs::PlaneGeom.

Parameters

in	_p	The plane to convert

Returns

A (p. 111) msgs::PlaneGeom object

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

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

Parameters

in	*_v*	The plane to convert

Returns

A (p. 111) **math::Vector3** (p. 855) object

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

Convert a msgs::Quaternion to a **math::Quaternion** (p. 623).

Parameters

in	*_q*	The quaternion to convert

Returns

A (p. 111) **math::Quaternion** (p. 623) object

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

Convert a msgs::Pose to a **math::Pose** (p. 596).

Parameters

in	*_q*	The pose to convert

Returns

A (p. 111) **math::Pose** (p. 596) object

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

Convert a msgs::Color to a **common::Color** (p. 208).

Parameters

in	*_c*	The color to convert

Returns

A (p. 111) **common::Color** (p. 208) object

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

Convert a msgs::Time to a **common::Time** (p. 791).

Parameters

in	*_t*	The time to convert

Returns

A (p. 111) **common::Time** (p. 791) object

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

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

Parameters

in	*_p*	The plane to convert

Returns

A (p. 111) common::Plane object

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

Create a request message.

Parameters

in	*_request*	Request string
in	*_data*	Optional data string

Returns

A (p. 111) Request message

8.5.3.14 msgs::Fog gazebo::msgs::FogFromSDF (sdf::ElementPtr _sdf)

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

Parameters

in	*_sdf*	The sdf element

Returns

The new msgs::Fog object

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

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

Parameters

in	*_sdf*	The sdf element

Returns

The new msgs::Geometry object

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

Get the header from a protobuf message.

Parameters

in	*_message*	**A** (p. 111) google protobuf message

Returns

A (p. 111) pointer to the message's header

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

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

Parameters

in	*_sdf*	The sdf element

Returns

The new msgs::GUI object

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

Initialize a message.

Parameters

in	*_message*	Message to initialize
in	*_id*	Optional string id

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

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

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

Parameters

in	*_sdf*	The sdf element

Returns

The new msgs::Light object

8.5.3.20 msgs::MeshGeom gazebo::msgs::MeshFromSDF (sdf::ElementPtr _sdf)

Create a msgs::MeshGeom from a mesh SDF element.

Parameters

in	*_sdf*	The sdf element

Returns

The new msgs::MeshGeom object

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

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

Parameters

in	*_sdf*	The sdf element

Returns

The new msgs::Scene object

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

Convert a msgs::Image to a **common::Image** (p. 360).

Parameters

out	*_img*	The **common::Image** (p. 360) container
in	*_msg*	The Image message to convert

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

Set a msgs::Image from a **common::Image** (p. 360).

Parameters

out	*_msg*	**A** (p. 111) msgs::Image pointer
in	*_i*	**A** (p. 111) **common::Image** (p. 360) reference

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

Set a msgs::Vector3d from a **math::Vector3** (p. 855).

Parameters

out	*_pt*	**A** (p. 111) msgs::Vector3d pointer
in	*_v*	**A** (p. 111) **math::Vector3** (p. 855) reference

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

Set a msgs::Vector2d from a **math::Vector3** (p. 855).

Parameters

out	*_pt*	**A** (p. 111) msgs::Vector2d pointer
in	*_v*	**A** (p. 111) **math::Vector2d** (p. 838) reference

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

Set a msgs::Quaternion from a **math::Quaternion** (p. 623).

Parameters

out	*_q*	**A** (p. 111) msgs::Quaternion pointer
in	*_v*	**A** (p. 111) **math::Quaternion** (p. 623) reference

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

Set a msgs::Pose from a **math::Pose** (p. 596).

Parameters

out	*_p*	**A** (p. 111) msgs::Pose pointer
in	*_v*	**A** (p. 111) **math::Pose** (p. 596) reference

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

Set a msgs::Color from a **common::Color** (p. 208).

Parameters

out	*_p*	**A** (p. 111) msgs::Color pointer
in	*_v*	**A** (p. 111) **common::Color** (p. 208) reference

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

Set a msgs::Time from a **common::Time** (p. 791).

Parameters

out	*_p*	**A** (p. 111) msgs::Time pointer
in	*_v*	**A** (p. 111) **common::Time** (p. 791) reference

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

Set a msgs::Plane from a **math::Plane** (p. 587).

Parameters

out	*_p*	**A** (p. 111) msgs::Plane pointer
in	*_v*	**A** (p. 111) **math::Plane** (p. 587) reference

8.5.3.31 void gazebo::msgs::Stamp (msgs::Header * *_header*)

Time stamp a header.

Parameters

in	*_header*	Header to stamp

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

Set the time in a time message.

Parameters

in	*_time*	**A** (p. 111) Time message

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

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

Parameters

in	*_sdf*	The sdf element

Returns

The new msgs::TrackVisual object

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

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

Parameters

in	*_sdf*	The sdf element

Returns

The new msgs::Visual object

8.6 Rendering

A (p. 111) 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. 247)* ***Conversions.hh*** *(p. 964)* ***rendering/Conversions.hh*** *(p. 964).*
- class **gazebo::rendering::DepthCamera**

  Depth camera used to render depth data into an image buffer.
- class **gazebo::rendering::DynamicLines**

  Class for drawing lines that can change.
- class **gazebo::rendering::DynamicRenderable**

  Abstract base class providing mechanisms for dynamically growing hardware buffers.
- class **gazebo::rendering::Events**

  Base class for rendering events.
- class **gazebo::rendering::FPSViewController**

  First Person Shooter style view controller.
- class **gazebo::rendering::GpuLaser**

  GPU based laser distance sensor.
- class **gazebo::rendering::Grid**

  Displays a grid of cells, drawn with lines.
- class **gazebo::rendering::GUIOverlay**

  A *(p. 111) class that creates a CEGUI overlay on a render window.*
- class **gazebo::rendering::Heightmap**

  Rendering a terrain using heightmap information.
- class **gazebo::rendering::JointVisual**

  Visualization for joints.
- class **gazebo::rendering::LaserVisual**

Visualization for laser data.

- class **gazebo::rendering::Light**

  A (p. *111) light source.*
- class **gazebo::rendering::MovableText**

  Movable text.
- class **gazebo::rendering::OrbitViewController**

  Orbit view controller.
- class **gazebo::rendering::Projector**

  Projects a material onto surface, light a light projector.
- class **gazebo::rendering::RenderEngine**

  Adaptor to Ogre3d.
- class **gazebo::rendering::RFIDTagVisual**

  Visualization for RFID tags sensor.
- class **gazebo::rendering::RFIDVisual**

  Visualization for RFID sensor.
- class **Road**

  Used to render a strip of road.
- class **gazebo::rendering::Road2d**
- class **gazebo::rendering::RTShaderSystem**

  *Implements **Ogre*** (p. *106)'s Run-Time Shader system.*
- class **gazebo::rendering::Scene**

  Representation of an entire scene graph.
- class **gazebo::rendering::SelectionObj**

  A (p. *111) graphical selection object.*
- class **gazebo::rendering::UserCamera**

  A (p. *111) camera used for user visualization of a scene.*
- class **gazebo::rendering::VideoVisual**

  A (p. *111) visual element that displays a video as a texture.*
- class **gazebo::rendering::ViewController**

  Base class for view controllers.
- class **gazebo::rendering::Visual**

  A (p. *111) renderable object.*
- class **gazebo::rendering::WindowManager**

  Class to mangage render windows.
- class **gazebo::rendering::WireBox**

  Draws a wireframe box.

Functions

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

  *create **rendering::Scene*** (p. *676) 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. *676) 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. 676) by name*

8.6.1 Detailed Description

A (p. 111) set of rendering related class, functions, and definitions.

8.6.2 Function Documentation

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

create **rendering::Scene** (p. 676) by name.

Parameters

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

8.6.2.2 bool gazebo::rendering::fini ( )

teardown rendering engine.

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

get pointer to **rendering::Scene** (p. 676) by name.

Parameters

in	*_name*	Name of the scene to retreive.

8.6.2.4 bool gazebo::rendering::init ( )

init rendering engine.

8.6.2.5 bool gazebo::rendering::load ( )

load rendering engine.

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

remove a **rendering::Scene** (p. 676) by name

Parameters

in	*_name*	The name of the scene to remove.

8.7 Gazebo_parser

Namespaces

- namespace **sdf**

  namespace for Simulation Description Format parser
- namespace **urdf2gazebo**

  namespace for URDF to SDF parser

Classes

- class **A**

  holding gazebo extension elements in urdf
- class **sdf::Element**

  SDF *(*p. *695)* ***Element*** *(*p. *273) class.*
- class **urdf2gazebo::GazeboExtension**
- class **sdf::SDF**

  Base ***SDF*** *(*p. *695) class.*
- class **urdf2gazebo::URDF2Gazebo**

Typedefs

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

8.7.1 Detailed Description

8.7.2 Typedef Documentation

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

8.7.2.2 typedef urdf::Collision∗ urdf2gazebo::UrdfCollisionPtr

8.7.2.3 typedef urdf::Link∗ urdf2gazebo::UrdfLinkPtr

8.7.2.4 typedef urdf::Visual∗ urdf2gazebo::UrdfVisualPtr

8.8 Sensors

A (p. 111) set of sensor classes, functions, and definitions.

Files

- file **SensorTypes.hh**

  Forward declarations and typedefs for sensors.

Namespaces

- namespace **gazebo::sensors**

  Sensors namespace.

Classes

- class **gazebo::sensors::CameraSensor**

  Basic camera sensor.
- class **gazebo::sensors::ContactSensor**

  Contact sensor.
- class **gazebo::sensors::DepthCameraSensor**
- class **gazebo::sensors::GpuRaySensor**
- class **gazebo::sensors::ImuSensor**

  An IMU sensor.
- class **gazebo::sensors::MultiCameraSensor**

  Multiple camera sensor.
- class **gazebo::sensors::RaySensor**

  Sensor *(p. 698) with one or more rays.*
- class **gazebo::sensors::RFIDSensor**

  Sensor *(p. 698) class for RFID type of sensor.*
- class **gazebo::sensors::RFIDTag**

  RFIDTag *(p. 660) to interact with RFIDTagSensors.*
- class **gazebo::sensors::Sensor**

  Base class for sensors.
- class **SensorFactor**

  The sensor factory; the class is just for namespacing purposes.
- class **gazebo::sensors::SensorFactory**
- class **gazebo::sensors::SensorManager**

  Class to manage and update all sensors.

Macros

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

  Static sensor registration macro.

Functions

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

  Create a sensor using SDF.
- bool **gazebo::sensors::fini** ()

  shutdown the sensor generation loop.
- SensorPtr **gazebo::sensors::get_sensor** (const std::string &_name)

  Get a sensor using by name.
- bool **gazebo::sensors::init** ()

  initialize the sensor generation loop.
- bool **gazebo::sensors::load** ()

  Load the sensor library.
- void **gazebo::sensors::remove_sensor** (const std::string &_sensorName)

  Remove a sensor by name.
- bool **gazebo::sensors::remove_sensors** ()

  Remove all sensors.
- void **gazebo::sensors::run** () **GAZEBO_DEPRECATED**(1.5)

  Deprecated.
- void **gazebo::sensors::run_once** (bool _force=false)

  Run the sensor generation one step.
- void **gazebo::sensors::run_threads** ()

  Run sensors in a threads. This is a non-blocking call.
- void **gazebo::sensors::stop** ()

  Stop the sensor generation loop.

8.8.1 Detailed Description

A (p. 111) set of sensor classes, functions, and definitions. GPU based laser sensor.

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

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

8.8.2 Macro Definition Documentation

8.8.2.1 #define GZ_REGISTER_STATIC_SENSOR(*name, classname*)

Value:

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

Static sensor registration macro.

Use this macro to register sensors with the server.

Parameters

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

8.8.3 Function Documentation

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

Create a sensor using SDF.

Parameters

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

Returns

The name of the new sensor.

8.8.3.2 bool gazebo::sensors::fini ( )

shutdown the sensor generation loop.

Returns

True if successfully finalized, false if not

8.8.3.3 SensorPtr gazebo::sensors::get_sensor (const std::string & *_name*)

Get a sensor using by name.

The given name should have: world_name::model_name::link_name::sensor_name

Parameters

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

Returns

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

8.8.3.4 bool gazebo::sensors::init ( )

initialize the sensor generation loop.

Returns

True if successfully initialized, false if not

8.8.3.5 bool gazebo::sensors::load ( )

Load the sensor library.

Returns

True if successfully loaded, false if not.

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

Remove a sensor by name.

Parameters

in	*_sensorName*	Name of sensor to remove

8.8.3.7 bool gazebo::sensors::remove_sensors ( )

Remove all sensors.

Returns

True if all successfully removed, false if not

8.8.3.8 void gazebo::sensors::run ( )

Deprecated.

See Also

run_threads (p. 73)

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

Run the sensor generation one step.

Parameters

_force,:	If true, all sensors are forced to update. Otherwise a sensor will update based on it's Hz rate.

8.8.3.10 void gazebo::sensors::run_threads ( )

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

8.8.3.11 void gazebo::sensors::stop ( )

Stop the sensor generation loop.

8.9 Transport

Handles transportation of messages.

Files

- file **TransportTypes.hh**

  Forward declarations for transport.

Classes

- class **gazebo::transport::CallbackHelper**

  A *(p. 111) helper class to handle callbacks when messages arrive.*
- class **gazebo::transport::CallbackHelperT< M >**

  Callback helper Template.
- class **gazebo::transport::Connection**

  Single TCP/IP connection manager.
- class **gazebo::transport::ConnectionManager**

  Manager of connections.
- class **gazebo::transport::IOManager**

  Manages boost::asio IO.
- class **gazebo::transport::Node**

  A *(p. 111) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.*
- class **gazebo::transport::Publication**

  A *(p. 111) publication for a topic.*
- class **gazebo::transport::PublicationTransport**

  transport/transport.hh
- class **gazebo::transport::Publisher**

  A *(p. 111) publisher of messages on a topic.*
- class **gazebo::transport::RawCallbackHelper**

  Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.
- class **gazebo::transport::SubscribeOptions**

  Options for a subscription.
- class **gazebo::transport::Subscriber**

  A *(p. 111) subscriber to a topic.*
- class **gazebo::transport::SubscriptionTransport**

  transport/transport.hh
- class **gazebo::transport::TopicManager**

  Manages topics and their subscriptions.

Typedefs

- typedef CallbackHelper * **gazebo::transport::CallbackHelperPtr**

  boost shared pointer to ***transport::CallbackHelper*** *(p. 157)*

Functions

- void **gazebo::transport::clear_buffers** ()

  Clear any remaining communication buffers.
- void **gazebo::transport::fini** ()

  Cleanup the transport component.
- bool **gazebo::transport::get_master_uri** (std::string &_master_host, unsigned int &_master_port)

  Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.
- void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)

  Return all the namespace (world names) on the master.
- std::map< std::string, std::list< std::string > > **gazebo::transport::getAdvertisedTopics** ()

  Get a list of all the topics and their message types.
- std::list< std::string > **gazebo::transport::getAdvertisedTopics** (const std::string &_msgType)

  Get a list of all the unique advertised topic names.
- std::string **gazebo::transport::getTopicMsgType** (const std::string &_topicName)

  Get the message typename that is published on the given topic.
- bool **gazebo::transport::init** (const std::string &_master_host="", unsigned int _master_port=0)

  Initialize the transport system.
- bool **gazebo::transport::is_stopped** ()

  Is the transport system stopped?
- void **gazebo::transport::pause_incoming** (bool _pause)

  Pause or unpause incoming messages.
- msgs::Response * **gazebo::transport::request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  Send a request and receive a response.
- void **gazebo::transport::requestNoReply** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  Send a request and don't wait for a response.
- void **gazebo::transport::requestNoReply** (NodePtr _node, const std::string &_request, const std::string &_data="")

  Send a request and don't wait for a response.
- void **gazebo::transport::run** ()

  Run the transport component.
- void **gazebo::transport::stop** ()

  Stop the transport component from running.

8.9.1 Detailed Description

Handles transportation of messages.

8.9.2 Typedef Documentation

8.9.2.1 typedef CallbackHelper* gazebo::transport::CallbackHelperPtr

boost shared pointer to **transport::CallbackHelper** (p. 157)

8.9.3 Function Documentation

8.9.3.1 void gazebo::transport::clear_buffers ( )

Clear any remaining communication buffers.

8.9.3.2 void gazebo::transport::fini ( )

Cleanup the transport component.

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

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

Parameters

out	*_master_host*	The hostname of the master is set to this param
out	*_master_port*	The port of the master is set to this param

Returns

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

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

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

Parameters

out	*_namespaces*	The list of namespace will be written here

8.9.3.5 std::map<std::string, std::list<std::string> > gazebo::transport::getAdvertisedTopics ( )

Get a list of all the topics and their message types.

Returns

A (p. 111) map where keys are message types, and values are a list of topic names.

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

Get a list of all the unique advertised topic names.

Parameters

in	*_msgType*	Type of message to filter the result on. If empty, then a list of all the topics is returned.

Returns

A (p. 111) list of the advertised topics that publish messages of the type specified by _msgType.

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

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

Parameters

in	*_topicName*	Name of the topic to query.

Returns

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

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

Initialize the transport system.

Parameters

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

Returns

true if initialization succeeded; false otherwise

8.9.3.9 bool gazebo::transport::is_stopped ( )

Is the transport system stopped?

Returns

true if the transport system is stopped; false otherwise

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

Pause or unpause incoming messages.

When paused, messages are queued for later delivery

Parameters

in	*_pause*	If true, pause; otherwise unpause

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

Send a request and receive a response.

This call will block until a response is received.

Parameters

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

Returns

The response to the request. Can be empty.

8.9.3.12 void gazebo::transport::requestNoReply (const std::string & _worldName, const std::string & _request, const std::string & _data = " ")

Send a request and don't wait for a response.

This is non-blocking.

Parameters

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

8.9.3.13 void gazebo::transport::requestNoReply (NodePtr _node, const std::string & _request, const std::string & _data = " ")

Send a request and don't wait for a response.

This is non-blocking.

Parameters

in	*_node*	Pointer to a node that provides communication.
in	*_request*	The type request.
in	*_data*	Optional data string.

8.9.3.14 void gazebo::transport::run ( )

Run the transport component.

Creates a thread to handle message passing. This call will block until the master can be contacted or until a retry limit is reached

8.9.3.15 void gazebo::transport::stop ( )

Stop the transport component from running.

8.10 Utility

Files

- file **UtilTypes.hh**

Classes

- class **gazebo::util::DiagnosticManager**

  A *(*p. *111) diagnostic manager class.*
- class **gazebo::util::DiagnosticTimer**

  A *(*p. *111) timer designed for diagnostics.*

Macros

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

8.10.1 Detailed Description

8.10.2 Macro Definition Documentation

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

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

8.10.2.3 #define DIAG_TIMER_STOP(*_name*) ((void) 0)

Chapter 9

Namespace Documentation

9.1 boost Namespace Reference

9.2 gazebo Namespace Reference

Forward declarations for the common classes.

Namespaces

- namespace **common**

  Common namespace.
- namespace **event**

  Event *(*p. *292) namespace.*
- namespace **math**

  Math namespace.
- namespace **msgs**

  Messages namespace.
- namespace **physics**

  namespace for physics
- namespace **rendering**

  Rendering namespace.
- namespace **sensors**

  Sensors namespace.
- namespace **transport**
- namespace **util**

Classes

- class **Master**

  A *(*p. *111) ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.*
- class **ModelPlugin**

  A *(*p. *111) plugin with access to* ***physics::Model*** *(*p. *489).*

- class **PluginT**

  A (p. *111) class which all plugins must inherit from.*
- class **SensorPlugin**

  A (p. *111) plugin with access to physics::Sensor.*
- class **Server**
- class **SystemPlugin**

  A (p. *111) plugin loaded within the gzserver on startup.*
- class **VisualPlugin**

  A (p. *111) plugin loaded within the gzserver on startup.*
- class **WorldPlugin**

  A (p. *111) plugin with access to* ***physics::World*** (p. *910).*

Typedefs

- typedef GUIPlugin ∗ **GUIPluginPtr**
- typedef **ModelPlugin** ∗ **ModelPluginPtr**
- typedef **SensorPlugin** ∗ **SensorPluginPtr**
- typedef **SystemPlugin** ∗ **SystemPluginPtr**
- typedef **VisualPlugin** ∗ **VisualPluginPtr**
- typedef **WorldPlugin** ∗ **WorldPluginPtr**

Enumerations

- enum **PluginType** {
  WORLD_PLUGIN, **MODEL_PLUGIN**, **SENSOR_PLUGIN**, **SYSTEM_PLUGIN**,
  VISUAL_PLUGIN }

  Used to specify the type of plugin.

Functions

- void **add_plugin** (const std::string &_filename)
- std::string **find_file** (const std::string &_file)

  Find a file in the gazebo search paths.
- void **fini** ()
- bool **init** ()
- bool **load** (int _argc=0, char ∗∗_argv=0)
- void **print_version** ()
- void **run** ()
- void **stop** ()

9.2.1 Detailed Description

Forward declarations for the common classes. Forward declarations for the util classes.

9.2.2 Typedef Documentation

9.2.2.1 typedef GUIPlugin* gazebo::GUIPluginPtr

9.2.2.2 typedef ModelPlugin* gazebo::ModelPluginPtr

9.2.2.3 typedef SensorPlugin* gazebo::SensorPluginPtr

9.2.2.4 typedef SystemPlugin* gazebo::SystemPluginPtr

9.2.2.5 typedef VisualPlugin* gazebo::VisualPluginPtr

9.2.2.6 typedef WorldPlugin* gazebo::WorldPluginPtr

9.2.3 Function Documentation

9.2.3.1 void gazebo::add_plugin (const std::string & *_filename*)

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

Find a file in the gazebo search paths.

9.2.3.3 void gazebo::fini ( )

9.2.3.4 bool gazebo::init ( )

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

9.2.3.6 void gazebo::print_version ( )

9.2.3.7 void gazebo::run ( )

9.2.3.8 void gazebo::stop ( )

9.3 gazebo::common Namespace Reference

Common namespace.

Classes

- class **Animation**

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

  Class for generating Exceptions which come from gazebo assertions.
- class **BVHLoader**

  Handles loading BVH animation files.
- class **ColladaLoader**

  Class used to load Collada mesh files.
- class **Color**

Defines a color.

- class **Console**

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

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

  Encapsulates an image.
- class **InternalError**

  Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.
- class **KeyFrame**

  A *(*p. *111) key frame in an animation.*
- class **LogPlay**
- class **LogRecord**

  addtogroup gazebo_common
- class **Material**

  Encapsulates description of a material.
- class **Mesh**

  A *(*p. *111) 3D mesh.*
- class **MeshCSG**

  Creates CSG meshes.
- class **MeshLoader**

  Base class for loading meshes.
- class **MeshManager**

  Maintains and manages all meshes.
- class **ModelDatabase**

  Connects to model database, and has utility functions to find models.
- class **MouseEvent**

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

  Node animation.
- struct **NodeAssignment**

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

  NodeTransform *(*p. *547)* ***Skeleton.hh*** *(*p. *1088) common/common.hh*
- class **NumericAnimation**

  A *(*p. *111) numeric animation.*
- class **NumericKeyFrame**

  A *(*p. *111) keyframe for a* ***NumericAnimation*** *(*p. *552).*
- class **PID**

  Generic ***PID*** *(*p. *583) controller class.*
- class **PoseAnimation**

  A *(*p. *111) pose animation.*
- class **PoseKeyFrame**

  A *(*p. *111) keyframe for a* ***PoseAnimation*** *(*p. *605).*
- class **Skeleton**

  A *(*p. *111) skeleton.*
- class **SkeletonAnimation**

Skeleton *(p. 727) animation.*

- class **SkeletonNode**

  A *(p. 111) skeleton node.*

- class **STLLoader**

  Class used to load STL mesh files.

- class **SubMesh**

  A *(p. 111) child mesh.*

- class **SystemPaths**

  Functions to handle getting system paths, keeps track of:

- class **Time**

  A *(p. 111)* ***Time*** *(p. 791) class, can be used to hold wall- or sim-time.*

- class **Timer**

  A *(p. 111) timer class, used to time things in real world walltime.*

- class **UpdateInfo**

  Information for use in an update event.

- class **Video**

  Handle video encoding and decoding using libavcodec.

Typedefs

- typedef **Animation** ∗ **AnimationPtr**
- typedef DiagnosticTimer ∗ **DiagnosticTimerPtr**
- typedef std::map< unsigned int, **SkeletonNode** ∗ > **NodeMap**
- typedef std::map< unsigned int, **SkeletonNode** ∗ >::iterator **NodeMapIter**
- typedef **NumericAnimation** ∗ **NumericAnimationPtr**
- typedef std::vector < common::Param ∗ > **Param_V**
- typedef **PoseAnimation** ∗ **PoseAnimationPtr**
- typedef std::map< double, std::vector< **NodeTransform** > > **RawNodeAnim**
- typedef std::vector < std::vector< std::pair < std::string, double > > > **RawNodeWeights**
- typedef std::map< std::string, **RawNodeAnim** > **RawSkeletonAnim**
- typedef std::map< std::string, std::string > **StrStr_M**

Functions

- void **add_search_path_suffix** (const std::string &_suffix)

  add path prefix to ***common::SystemPaths*** *(p. 784)*

- std::string **find_file** (const std::string &_file, bool _searchLocalPath=true)

  search for file in ***common::SystemPaths*** *(p. 784)*

- std::string **find_file_path** (const std::string &_file)

  search for a file in ***common::SystemPaths*** *(p. 784)*

Variables

- static std::string **PixelFormatNames** [ ]

  String names for the pixel formats.

9.3.1 Detailed Description

Common namespace.

9.3.2 Typedef Documentation

9.3.2.1 typedef Animation* gazebo::common::AnimationPtr

9.3.2.2 typedef DiagnosticTimer* gazebo::common::DiagnosticTimerPtr

9.3.2.3 typedef std::map<unsigned int, SkeletonNode*> gazebo::common::NodeMap

9.3.2.4 typedef std::map<unsigned int, SkeletonNode*>::iterator gazebo::common::NodeMapIter

9.3.2.5 typedef NumericAnimation* gazebo::common::NumericAnimationPtr

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

9.3.2.7 typedef PoseAnimation* gazebo::common::PoseAnimationPtr

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

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

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

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

9.4 gazebo::event Namespace Reference

Event (p. 292) namespace.

Classes

- class **Connection**

  A *(p. 111) class that encapsulates a connection.*
- class **Event**

  Base class for all events.
- class **Events**

  An ***Event*** *(p. 292) class to get notifications for simulator events.*
- class **EventT**

  A *(p. 111) class for event processing.*

Typedefs

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

9.4.1 Detailed Description

Event (p. 292) namespace.

9.4.2 Typedef Documentation

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

9.4.2.2 typedef Connection* gazebo::event::ConnectionPtr

9.5 gazebo::math Namespace Reference

Math namespace.

Classes

- class **Angle**

  An angle and related functions.
- class **Box**

  Mathematical representation of a box and related functions.
- class **Matrix3**

  A *(p. 111) 3x3 matrix class.*
- class **Matrix4**

  A *(p. 111) 3x3 matrix class.*
- class **Plane**

  A *(p. 111) plane and related functions.*
- class **Pose**

  Encapsulates a position and rotation in three space.
- class **Quaternion**

  A *(p. 111) quaternion class.*
- class **Rand**

  Random number generator class.
- class **RotationSpline**

  Spline *(p. 754) for rotations.*
- class **Spline**

  Splines.
- class **Vector2d**

  Generic double x, y vector.
- class **Vector2i**

  Generic integer x, y vector.
- class **Vector3**

*The **Vector3** (p. 855) class represents the generic vector containing 3 elements.*

- class **Vector4**

  double Generic x, y, z, w vector

Typedefs

- typedef boost::mt19937 **GeneratorType**
- typedef boost::normal_distribution < double > **NormalRealDist**
- typedef boost::variate_generator < **GeneratorType** &, **NormalRealDist** > **NRealGen**
- typedef boost::variate_generator < **GeneratorType** &, **UniformIntDist** > **UIntGen**
- typedef boost::uniform_int< int > **UniformIntDist**
- typedef boost::uniform_real < double > **UniformRealDist**
- typedef boost::variate_generator < **GeneratorType** &, **UniformRealDist** > **URealGen**

Functions

- template<typename T >
  T **clamp** (T _v, T _min, T _max)

  Simple clamping function.
- template<typename T >
  bool **equal** (const T &_a, const T &_b, const T &_epsilon=1e-6)

  check if two values are equal, within a tolerance
- bool **isnan** (float _v)

  check if a float is NaN
- bool **isnan** (double _v)

  check if a double is NaN
- bool **isPowerOfTwo** (unsigned int _x)

  is this a power of 2?
- template<typename T >
  T **max** (const std::vector< T > &_values)

  get the maximum value of vector of values
- template<typename T >
  T **mean** (const std::vector< T > &_values)

  get mean of vector of values
- template<typename T >
  T **min** (const std::vector< T > &_values)

  get the minimum value of vector of values

- double **parseFloat** (const std::string &_input)

  parse string into float
- int **parseInt** (const std::string &_input)

  parse string into an integer
- template<typename T >
  T **precision** (const T &_a, const unsigned int &_precision)

  get value at a specified precision
- template<typename T >
  T **variance** (const std::vector< T > &_values)

  get variance of vector of values

Variables

- static const double **NAN_D** = std::numeric_limits<double>::quiet_NaN()

  Returns the representation of a quiet not a number (NAN)
- static const int **NAN_I** = std::numeric_limits<int>::quiet_NaN()

  Returns the representation of a quiet not a number (NAN)

9.5.1 Detailed Description

Math namespace.

9.5.2 Typedef Documentation

9.5.2.1 typedef boost::mt19937 gazebo::math::GeneratorType

9.5.2.2 typedef boost::normal_distribution<double> gazebo::math::NormalRealDist

9.5.2.3 typedef boost::variate_generator<GeneratorType&, NormalRealDist > gazebo::math::NRealGen

9.5.2.4 typedef boost::variate_generator<GeneratorType&, UniformIntDist > gazebo::math::UIntGen

9.5.2.5 typedef boost::uniform_int<int> gazebo::math::UniformIntDist

9.5.2.6 typedef boost::uniform_real<double> gazebo::math::UniformRealDist

9.5.2.7 typedef boost::variate_generator<GeneratorType&, UniformRealDist > gazebo::math::URealGen

9.6 gazebo::msgs Namespace Reference

Messages namespace.

Classes

- class **MsgFactory**

  A *(p. 111) factory that generates protobuf message based on a string type.*

Typedefs

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

Functions

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

  *Convert a **math::Vector3*** (p. *855) to a msgs::Vector3d.*
- msgs::Quaternion **Convert** (const **math::Quaternion** &_q)

  *Convert a **math::Quaternion*** (p. *623) to a msgs::Quaternion.*
- msgs::Pose **Convert** (const **math::Pose** &_p)

  *Convert a **math::Pose*** (p. *596) to a msgs::Pose.*
- msgs::Color **Convert** (const **common::Color** &_c)

  *Convert a **common::Color*** (p. *208) to a msgs::Color.*
- msgs::Time **Convert** (const **common::Time** &_t)

  *Convert a **common::Time*** (p. *791) to a msgs::Time.*
- msgs::PlaneGeom **Convert** (const **math::Plane** &_p)

  *Convert a **math::Plane*** (p. *587) 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. *623).*
- **math::Pose Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose*** (p. *596).*
- **common::Color Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color*** (p. *208).*
- **common::Time Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time*** (p. *791).*
- **math::Plane Convert** (const msgs::PlaneGeom &_p)

  Convert a msgs::PlaneGeom to a common::Plane.
- msgs::Request ∗ **CreateRequest** (const std::string &_request, const std::string &_data="")

  Create a request message.
- msgs::Fog **FogFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Fog from a fog SDF element.
- msgs::Geometry **GeometryFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Geometry from a geometry SDF element.
- msgs::Header ∗ **GetHeader** (google::protobuf::Message &_message)

  Get the header from a protobuf message.
- msgs::GUI **GUIFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::GUI from a GUI SDF element.
- void **Init** (google::protobuf::Message &_message, const std::string &_id="")

  Initialize a message.
- msgs::Light **LightFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Light from a light SDF element.
- msgs::MeshGeom **MeshFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::MeshGeom from a mesh SDF element.

- msgs::Scene **SceneFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Scene from a scene SDF element.
- void **Set** (**common::Image** &_img, const msgs::Image &_msg)

  *Convert a msgs::Image to a **common::Image*** (p. *360).*
- void **Set** (msgs::Image ∗_msg, const **common::Image** &_i)

  *Set a msgs::Image from a **common::Image*** (p. *360).*
- void **Set** (msgs::Vector3d ∗_pt, const **math::Vector3** &_v)

  *Set a msgs::Vector3d from a **math::Vector3*** (p. *855).*
- void **Set** (msgs::Vector2d ∗_pt, const **math::Vector2d** &_v)

  *Set a msgs::Vector2d from a **math::Vector3*** (p. *855).*
- void **Set** (msgs::Quaternion ∗_q, const **math::Quaternion** &_v)

  *Set a msgs::Quaternion from a **math::Quaternion*** (p. *623).*
- void **Set** (msgs::Pose ∗_p, const **math::Pose** &_v)

  *Set a msgs::Pose from a **math::Pose*** (p. *596).*
- void **Set** (msgs::Color ∗_c, const **common::Color** &_v)

  *Set a msgs::Color from a **common::Color*** (p. *208).*
- void **Set** (msgs::Time ∗_t, const **common::Time** &_v)

  *Set a msgs::Time from a **common::Time*** (p. *791).*
- void **Set** (msgs::PlaneGeom ∗_p, const **math::Plane** &_v)

  *Set a msgs::Plane from a **math::Plane*** (p. *587).*
- void **Stamp** (msgs::Header ∗_header)

  Time stamp a header.
- void **Stamp** (msgs::Time ∗_time)

  Set the time in a time message.
- msgs::TrackVisual **TrackVisualFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::TrackVisual from a track visual SDF element.
- msgs::Visual **VisualFromSDF** (**sdf::ElementPtr** _sdf)

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

9.6.1 Detailed Description

Messages namespace.

9.6.2 Typedef Documentation

9.6.2.1 typedef google::protobuf::Message∗(∗ gazebo::msgs::MsgFactoryFn)()

9.7 gazebo::physics Namespace Reference

namespace for physics

Classes

- class **Actor**

  Actor *(p. 111) class enables GPU based mesh model / skeleton scriptable animation.*
- class **BallJoint**

  Base *(p. 137) class for a ball joint.*
- class **Base**

  Base *(p. 137) class for most physics classes.*
- class **BoxShape**

  Box geometry primitive.
- class **Collision**

  Base *(p. 137) class for all collision entities.*
- class **CollisionState**

  Store state information of a ***physics::Collision*** *(p. 195) object.*
- class **Contact**

  A *(p. 111) contact between two collisions.*
- class **ContactManager**

  Aggregates all the contact information generated by the collision detection engine.
- class **CylinderShape**

  Cylinder collision.
- class **Entity**

  Base *(p. 137) class for all physics objects in Gazebo.*
- class **Gripper**

  A *(p. 111) gripper abstraction.*
- class **HeightmapShape**

  HeightmapShape *(p. 352) collision shape builds a heightmap from an image.*
- class **Hinge2Joint**

  A *(p. 111) two axis hinge joint.*
- class **HingeJoint**

  A *(p. 111) single axis hinge joint.*
- class **Inertial**

  A *(p. 111) class for inertial information about a link.*
- class **Joint**

  Base *(p. 137) class for all joints.*
- class **JointController**

  A *(p. 111) class for manipulating* ***physics::Joint*** *(p. 381).*
- class **JointState**

  keeps track of state of a ***physics::Joint*** *(p. 381)*
- class **JointWrench**

  Wrench information from a joint.
- class **Link**

  Link *(p. 418) 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. 418) object.*
- class **Model**

  A *(p. 111) model is a collection of links, joints, and plugins.*
- class **ModelState**

*Store state information of a **physics::Model** (p. 489) object.*

- class **MultiRayShape**

  Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.
- class **PhysicsEngine**

  ***Base** (p. 137) class for a physics engine.*
- class **PhysicsFactory**

  The physics factory instantiates different physics engines.
- class **PlaneShape**

  ***Collision** (p. 195) for an infinite plane.*
- class **RayShape**

  ***Base** (p. 137) class for Ray collision geometry.*
- class **Road**

  *for building a **Road** (p. 665) from SDF*
- class **ScrewJoint**

  ***A** (p. 111) screw joint, which has both prismatic and rotational DOFs.*
- class **Shape**

  ***Base** (p. 137) class for all shapes.*
- class **SliderJoint**

  ***A** (p. 111) slider joint.*
- class **SphereShape**

  Sphere collision shape.
- class **State**

  ***State** (p. 758) of an entity.*
- class **SurfaceParams**

  ***SurfaceParams** (p. 780) defines various Surface contact parameters.*
- struct **TrajectoryInfo**
- class **TrimeshShape**

  Triangle mesh collision shape.
- class **UniversalJoint**

  ***A** (p. 111) 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. 910) object.*

Typedefs

- typedef std::vector< **ActorPtr** > **Actor_V**
- typedef **Actor** * **ActorPtr**
- typedef std::vector< **BasePtr** > **Base_V**
- typedef **Base** * **BasePtr**
- typedef **BoxShape** * **BoxShapePtr**
- typedef std::vector< **CollisionPtr** > **Collision_V**
- typedef **Collision** * **CollisionPtr**
- typedef **Contact** * **ContactPtr**
- typedef **CylinderShape** * **CylinderShapePtr**
- typedef **Entity** * **EntityPtr**
- typedef **HeightmapShape** * **HeightmapShapePtr**

- typedef **Inertial** ∗ **InertialPtr**
- typedef std::vector< **JointPtr** > **Joint_V**
- typedef std::vector < **JointControllerPtr** > **JointController_V**
- typedef **JointController** ∗ **JointControllerPtr**
- typedef **Joint** ∗ **JointPtr**
- typedef std::vector< **LinkPtr** > **Link_V**
- typedef **Link** ∗ **LinkPtr**
- typedef MeshShape ∗ **MeshShapePtr**
- typedef std::vector< **ModelPtr** > **Model_V**
- typedef **Model** ∗ **ModelPtr**
- typedef **MultiRayShape** ∗ **MultiRayShapePtr**
- typedef **PhysicsEngine** ∗ **PhysicsEnginePtr**
- typedef **PhysicsEnginePtr**(∗ **PhysicsFactoryFn**)(**WorldPtr** world)
- typedef **RayShape** ∗ **RayShapePtr**
- typedef **Road** ∗ **RoadPtr**
- typedef **Shape** ∗ **ShapePtr**
- typedef **SphereShape** ∗ **SphereShapePtr**
- typedef **SurfaceParams** ∗ **SurfaceParamsPtr**
- typedef **World** ∗ **WorldPtr**

Functions

- **WorldPtr create_world** (const std::string &_name="")

  Create a world given a name.
- bool **fini** ()

  *Finalize transport by calling **gazebo::transport::fini** (p. 77).*
- **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. 789)'s and call **gazebo::transport::init** (p. 78).*
- void **load_world** (**WorldPtr** _world, **sdf::ElementPtr** _sdf)

  *Load world from **sdf::Element** (p. 273) pointer.*
- void **load_worlds** (**sdf::ElementPtr** _sdf)

  *load multiple worlds from single **sdf::Element** (p. 273) pointer*
- void **pause_world** (**WorldPtr** _world, bool _pause)

  *Pause world by calling **World::SetPaused** (p. 920).*
- 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. 920) 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. 921) given a pointer to it.*

- void **stop_worlds** ()

  stop multiple worlds stored in static variable gazebo::g_worlds

Variables

- static std::string **EntityTypename** []

  String names for the different entity types.

9.7.1 Detailed Description

namespace for physics Physics forward declarations and type defines.

physics namespace

9.7.2 Typedef Documentation

9.7.2.1 typedef std::vector<ActorPtr> gazebo::physics::Actor_V

9.7.2.2 typedef Actor* gazebo::physics::ActorPtr

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

9.7.2.4 typedef Base* gazebo::physics::BasePtr

9.7.2.5 typedef BoxShape* gazebo::physics::BoxShapePtr

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

9.7.2.7 typedef Collision* gazebo::physics::CollisionPtr

9.7.2.8 typedef Contact* gazebo::physics::ContactPtr

9.7.2.9 typedef CylinderShape* gazebo::physics::CylinderShapePtr

9.7.2.10 typedef Entity* gazebo::physics::EntityPtr

9.7.2.11 typedef HeightmapShape* gazebo::physics::HeightmapShapePtr

9.7.2.12 typedef Inertial* gazebo::physics::InertialPtr

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

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

9.7.2.15 typedef JointController* gazebo::physics::JointControllerPtr

9.7.2.16 typedef Joint* gazebo::physics::JointPtr

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

9.7.2.18 typedef Link* gazebo::physics::LinkPtr

9.7.2.19 typedef MeshShape* gazebo::physics::MeshShapePtr

9.7.2.20 typedef std::vector<ModelPtr> gazebo::physics::Model_V

9.7.2.21 typedef Model* gazebo::physics::ModelPtr

9.7.2.22 typedef MultiRayShape* gazebo::physics::MultiRayShapePtr

9.7.2.23 typedef PhysicsEngine* gazebo::physics::PhysicsEnginePtr

9.7.2.24 typedef RayShape* gazebo::physics::RayShapePtr

9.7.2.25 typedef Road* gazebo::physics::RoadPtr

9.7.2.26 typedef Shape* gazebo::physics::ShapePtr

9.7.2.27 typedef SphereShape* gazebo::physics::SphereShapePtr

9.7.2.28 typedef SurfaceParams* gazebo::physics::SurfaceParamsPtr

9.7.2.29 typedef World* gazebo::physics::WorldPtr

9.8 gazebo::rendering Namespace Reference

Rendering namespace.

Classes

- class **ArrowVisual**

  Basic arrow visualization.
- class **AxisVisual**

  Basic axis visualization.
- class **Camera**

  Basic camera sensor.
- class **CameraVisual**

  Basic camera visualization.
- class **COMVisual**

  Basic Center of Mass visualization.
- class **ContactVisual**

  Contact visualization.
- class **Conversions**

  Conversions *(*p. *247)* ***Conversions.hh*** *(*p. *964)* ***rendering/Conversions.hh*** *(*p. *964).*
- class **DepthCamera**

  Depth camera used to render depth data into an image buffer.
- class **DynamicLines**

Class for drawing lines that can change.

- class **DynamicRenderable**

  Abstract base class providing mechanisms for dynamically growing hardware buffers.
- class **Events**

  Base class for rendering events.
- class **FPSViewController**

  First Person Shooter style view controller.
- class **GpuLaser**

  GPU based laser distance sensor.
- class **Grid**

  Displays a grid of cells, drawn with lines.
- class **GUIOverlay**

  A *(*p. *111) class that creates a CEGUI overlay on a render window.*
- class **GzTerrainMatGen**
- class **Heightmap**

  Rendering a terrain using heightmap information.
- class **JointVisual**

  Visualization for joints.
- class **LaserVisual**

  Visualization for laser data.
- class **Light**

  A *(*p. *111) light source.*
- class **MovableText**

  Movable text.
- class **OrbitViewController**

  Orbit view controller.
- class **Projector**

  Projects a material onto surface, light a light projector.
- class **RenderEngine**

  Adaptor to Ogre3d.
- class **RFIDTagVisual**

  Visualization for RFID tags sensor.
- class **RFIDVisual**

  Visualization for RFID sensor.
- class **Road2d**
- class **RTShaderSystem**

  Implements ***Ogre*** *(*p. *106)'s Run-Time Shader system.*
- class **Scene**

  Representation of an entire scene graph.
- class **SelectionObj**

  A *(*p. *111) graphical selection object.*
- class **UserCamera**

  A *(*p. *111) camera used for user visualization of a scene.*
- class **VideoVisual**

  A *(*p. *111) visual element that displays a video as a texture.*
- class **ViewController**

  Base class for view controllers.

- class **Visual**

  A *(p. 111) renderable object.*
- class **WindowManager**

  Class to mangage render windows.
- class **WireBox**

  Draws a wireframe box.

Typedefs

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

Enumerations

- enum **RenderOpType** {
  RENDERING_POINT_LIST = 0, **RENDERING_LINE_LIST** = 1, **RENDERING_LINE_STRIP** = 2, **RENDERING-_TRIANGLE_LIST** = 3,
  RENDERING_TRIANGLE_STRIP = 4, **RENDERING_TRIANGLE_FAN** = 5, **RENDERING_MESH_RESOURCE** = 6 }

  Type of render operation for a drawable.

Functions

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

  create ***rendering::Scene*** *(p. 676) by name.*
- bool **fini** ()

  teardown rendering engine.
- **rendering::ScenePtr get_scene** (const std::string &_name)

  get pointer to ***rendering::Scene*** *(p. 676) by name.*
- bool **init** ()

  init rendering engine.
- bool **load** ()

  load rendering engine.
- void **remove_scene** (const std::string &_name)

  remove a ***rendering::Scene*** *(p. 676) by name*

9.8.1 Detailed Description

Rendering namespace.

9.8.2 Typedef Documentation

9.8.2.1 typedef ArrowVisual* gazebo::rendering::ArrowVisualPtr

9.8.2.2 typedef AxisVisual* gazebo::rendering::AxisVisualPtr

9.8.2.3 typedef Camera* gazebo::rendering::CameraPtr

9.8.2.4 typedef CameraVisual* gazebo::rendering::CameraVisualPtr

9.8.2.5 typedef COMVisual* gazebo::rendering::COMVisualPtr

9.8.2.6 typedef ContactVisual* gazebo::rendering::ContactVisualPtr

9.8.2.7 typedef DepthCamera* gazebo::rendering::DepthCameraPtr

9.8.2.8 typedef DynamicLines* gazebo::rendering::DynamicLinesPtr

9.8.2.9 typedef GpuLaser* gazebo::rendering::GpuLaserPtr

9.8.2.10 typedef JointVisual* gazebo::rendering::JointVisualPtr

9.8.2.11 typedef LaserVisual* gazebo::rendering::LaserVisualPtr

9.8.2.12 typedef Light* gazebo::rendering::LightPtr

9.8.2.13 typedef RFIDTagVisual* gazebo::rendering::RFIDTagVisualPtr

9.8.2.14 typedef RFIDVisual* gazebo::rendering::RFIDVisualPtr

9.8.2.15 typedef Scene* gazebo::rendering::ScenePtr

9.8.2.16 typedef UserCamera* gazebo::rendering::UserCameraPtr

9.8.2.17 typedef Visual* gazebo::rendering::VisualPtr

9.8.3 Enumeration Type Documentation

9.8.3.1 enum gazebo::rendering::RenderOpType

Type of render operation for a drawable.

Enumerator:

RENDERING_POINT_LIST **A** (p. 111) list of points, 1 vertex per point.

RENDERING_LINE_LIST **A** (p. 111) list of lines, 2 vertices per line.

RENDERING_LINE_STRIP **A** (p. 111) strip of connected lines, 1 vertex per line plus 1 start vertex.

RENDERING_TRIANGLE_LIST **A** (p. 111) list of triangles, 3 vertices per triangle.

RENDERING_TRIANGLE_STRIP **A** (p. 111) strip of triangles, 3 vertices for the first triangle, and 1 per triangle after that.

RENDERING_TRIANGLE_FAN **A** (p. 111) fan of triangles, 3 vertices for the first triangle, and 1 per triangle after that.

RENDERING_MESH_RESOURCE N/A.

9.9 gazebo::sensors Namespace Reference

Sensors namespace.

Classes

- class **CameraSensor**

  Basic camera sensor.
- class **ContactSensor**

  Contact sensor.
- class **DepthCameraSensor**
- class **GpuRaySensor**
- class **ImuSensor**

  An IMU sensor.
- class **MultiCameraSensor**

  Multiple camera sensor.
- class **RaySensor**

  Sensor *(p. 698) with one or more rays.*
- class **RFIDSensor**

  Sensor *(p. 698) class for RFID type of sensor.*
- class **RFIDTag**

  RFIDTag *(p. 660) to interact with RFIDTagSensors.*
- class **Sensor**

  Base class for sensors.
- class **SensorFactory**
- class **SensorManager**

  Class to manage and update all sensors.
- class **SimTimeEvent**

- class **SimTimeEventHandler**

  Monitors simulation time, and notifies conditions when a specified time has been reached.

Typedefs

- typedef std::vector
  < **CameraSensorPtr** > **CameraSensor_V**
- typedef **CameraSensor** ∗ **CameraSensorPtr**
- typedef std::vector
  < **ContactSensorPtr** > **ContactSensor_V**

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

Enumerations

- enum **SensorCategory** { **IMAGE** = 0, **RAY** = 1, **OTHER** = 2, **CATEGORY_COUNT** = 3 }

  SensorClass is used to categorize sensors.

Functions

- std::string **create_sensor** (**sdf::ElementPtr** _elem, const std::string &_worldName, const std::string &_parentName)

  Create a sensor using SDF.
- bool **fini** ()

  shutdown the sensor generation loop.
- **SensorPtr get_sensor** (const std::string &_name)

  Get a sensor using by name.
- bool **init** ()

  initialize the sensor generation loop.
- bool **load** ()

  Load the sensor library.
- void **remove_sensor** (const std::string &_sensorName)

  Remove a sensor by name.
- bool **remove_sensors** ()

  Remove all sensors.
- void **run** () **GAZEBO_DEPRECATED**(1.5)

  Deprecated.
- void **run_once** (bool _force=false)

  Run the sensor generation one step.
- void **run_threads** ()

  Run sensors in a threads. This is a non-blocking call.
- void **stop** ()

  Stop the sensor generation loop.

9.9.1 Detailed Description

Sensors namespace.

9.9.2 Typedef Documentation

9.9.2.1 typedef std::vector<CameraSensorPtr> gazebo::sensors::CameraSensor_V

9.9.2.2 typedef CameraSensor* gazebo::sensors::CameraSensorPtr

9.9.2.3 typedef std::vector<ContactSensorPtr> gazebo::sensors::ContactSensor_V

9.9.2.4 typedef ContactSensor* gazebo::sensors::ContactSensorPtr

9.9.2.5 typedef std::vector<DepthCameraSensorPtr> gazebo::sensors::DepthCameraSensor_V

9.9.2.6 typedef DepthCameraSensor* gazebo::sensors::DepthCameraSensorPtr

9.9.2.7 typedef std::vector<GpuRaySensorPtr> gazebo::sensors::GpuRaySensor_V

9.9.2.8 typedef GpuRaySensor* gazebo::sensors::GpuRaySensorPtr

9.9.2.9 typedef std::vector<ImuSensorPtr> gazebo::sensors::ImuSensor_V

9.9.2.10 typedef ImuSensor* gazebo::sensors::ImuSensorPtr

9.9.2.11 typedef std::vector<RaySensorPtr> gazebo::sensors::RaySensor_V

9.9.2.12 typedef RaySensor* gazebo::sensors::RaySensorPtr

9.9.2.13 typedef std::vector<RFIDSensor> gazebo::sensors::RFIDSensor_V

9.9.2.14 typedef RFIDSensor* gazebo::sensors::RFIDSensorPtr

9.9.2.15 typedef std::vector<RFIDTag> gazebo::sensors::RFIDTag_V

9.9.2.16 typedef RFIDTag* gazebo::sensors::RFIDTagPtr

9.9.2.17 typedef std::vector<SensorPtr> gazebo::sensors::Sensor_V

9.9.2.18 typedef Sensor*(* gazebo::sensors::SensorFactoryFn)()

9.9.2.19 typedef Sensor* gazebo::sensors::SensorPtr

9.9.3 Enumeration Type Documentation

9.9.3.1 enum gazebo::sensors::SensorCategory

SensorClass is used to categorize sensors.

This is used to put sensors into different threads.

Enumerator:

IMAGE Image based sensor class. This type requires the rendering engine.

RAY Ray based sensor class.

OTHER **A** (p. 111) type of sensor is not a RAY or IMAGE sensor.

CATEGORY_COUNT Number of **Sensor** (p. 698) Categories.

9.10 gazebo::transport Namespace Reference

Classes

- class **CallbackHelper**

  A *(p. 111) helper class to handle callbacks when messages arrive.*
- class **CallbackHelperT**

  Callback helper Template.
- class **Connection**

  Single TCP/IP connection manager.
- class **ConnectionManager**

  Manager of connections.
- class **ConnectionReadTask**
- class **IOManager**

  Manages boost::asio IO.
- class **Node**

  A *(p. 111) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.*
- class **Publication**

  A *(p. 111) publication for a topic.*
- class **PublicationTransport**

  transport/transport.hh
- class **Publisher**

  A *(p. 111) publisher of messages on a topic.*
- class **PublishTask**
- class **RawCallbackHelper**

  Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.
- class **SubscribeOptions**

  Options for a subscription.
- class **Subscriber**

  A *(p. 111) subscriber to a topic.*
- class **SubscriptionTransport**

  transport/transport.hh
- class **TopicManager**

  Manages topics and their subscriptions.

Typedefs

- typedef **CallbackHelper** ∗ **CallbackHelperPtr**

  *boost shared pointer to **transport::CallbackHelper** (p. 157)*
- typedef **Connection** ∗ **ConnectionPtr**
- typedef google::protobuf::Message ∗ **MessagePtr**
- typedef **Node** ∗ **NodePtr**
- typedef **Publication** ∗ **PublicationPtr**
- typedef **PublicationTransport** ∗ **PublicationTransportPtr**
- typedef **Publisher** ∗ **PublisherPtr**
- typedef **Subscriber** ∗ **SubscriberPtr**
- typedef **SubscriptionTransport** ∗ **SubscriptionTransportPtr**

Functions

- void **clear_buffers** ()

  Clear any remaining communication buffers.
- void **fini** ()

  Cleanup the transport component.
- bool **get_master_uri** (std::string &_master_host, unsigned int &_master_port)

  Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.
- void **get_topic_namespaces** (std::list< std::string > &_namespaces)

  Return all the namespace (world names) on the master.
- std::map< std::string,
  std::list< std::string > > **getAdvertisedTopics** ()

  Get a list of all the topics and their message types.
- std::list< std::string > **getAdvertisedTopics** (const std::string &_msgType)

  Get a list of all the unique advertised topic names.
- std::string **getTopicMsgType** (const std::string &_topicName)

  Get the message typename that is published on the given topic.
- bool **init** (const std::string &_master_host="", unsigned int _master_port=0)

  Initialize the transport system.
- bool **is_stopped** ()

  Is the transport system stopped?
- void **pause_incoming** (bool _pause)

  Pause or unpause incoming messages.
- msgs::Response ∗ **request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  Send a request and receive a response.
- void **requestNoReply** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  Send a request and don't wait for a response.
- void **requestNoReply** (**NodePtr** _node, const std::string &_request, const std::string &_data="")

  Send a request and don't wait for a response.
- void **run** ()

  Run the transport component.
- void **stop** ()

  Stop the transport component from running.

9.10.1 Typedef Documentation

9.10.1.1 typedef Connection* gazebo::transport::ConnectionPtr

9.10.1.2 typedef google::protobuf::Message* gazebo::transport::MessagePtr

9.10.1.3 typedef Node* gazebo::transport::NodePtr

9.10.1.4 typedef Publication* gazebo::transport::PublicationPtr

9.10.1.5 typedef PublicationTransport* gazebo::transport::PublicationTransportPtr

9.10.1.6 typedef Publisher* gazebo::transport::PublisherPtr

9.10.1.7 typedef Subscriber* gazebo::transport::SubscriberPtr

9.10.1.8 typedef SubscriptionTransport* gazebo::transport::SubscriptionTransportPtr

9.11 gazebo::util Namespace Reference

Classes

- class **DiagnosticManager**

  A (p. *111*) *diagnostic manager class.*
- class **DiagnosticTimer**

  A (p. *111*) *timer designed for diagnostics.*

Typedefs

- typedef **DiagnosticTimer** * **DiagnosticTimerPtr**

9.11.1 Typedef Documentation

9.11.1.1 typedef DiagnosticTimer* gazebo::util::DiagnosticTimerPtr

9.12 google Namespace Reference

Namespaces

- namespace **protobuf**

9.13 google::protobuf Namespace Reference

Namespaces

- namespace **compiler**

9.14 google::protobuf::compiler Namespace Reference

Namespaces

- namespace **cpp**

9.15 google::protobuf::compiler::cpp Namespace Reference

Classes

- class **GazeboGenerator**

  *Google protobuf message generator for **gazebo::msgs** (p. 89).*

9.16 Ogre Namespace Reference

9.17 ogre Namespace Reference

9.18 sdf Namespace Reference

namespace for Simulation Description Format parser

Classes

- class **Converter**

  *Convert from one version of **SDF** (p. 695) to another.*
- class **Element**

  ***SDF** (p. 695) **Element** (p. 273) class.*
- class **Param**

  ***A** (p. 111) parameter class.*
- class **ParamT**

  Templatized parameter class.
- class **Plugin**
- class **SDF**

  *Base **SDF** (p. 695) 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. 695) 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. 695) values from a file.*
- bool **readString** (const std::string &_xmlString, **SDFPtr** _sdf)

  *Populate the **SDF** (p. 695) values from a string.*
- bool **readString** (const std::string &_xmlString, **ElementPtr** _sdf)
- bool **readXml** (TiXmlElement *_xml, **ElementPtr** _sdf)

9.18.1 Detailed Description

namespace for Simulation Description Format parser

9.18.2 Typedef Documentation

9.18.2.1 typedef Element* sdf::ElementPtr

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

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

9.18.2.4 typedef Param* sdf::ParamPtr

9.18.2.5 typedef SDF* sdf::SDFPtr

9.18.3 Function Documentation

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

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

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

Init based on the installed sdf_format.xml file.

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

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

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

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

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

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

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

Populate the **SDF** (p. 695) values from a TinyXML document.

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

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

Populate the **SDF** (p. 695) values from a file.

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

Populate the **SDF** (p. 695) values from a string.

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

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

9.19 SkyX Namespace Reference

9.20 urdf2gazebo Namespace Reference

namespace for URDF to SDF parser

Classes

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

Typedefs

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

9.20.1 Detailed Description

namespace for URDF to SDF parser

Chapter 10

Class Documentation

10.1 A Class Reference

holding gazebo extension elements in urdf

10.1.1 Detailed Description

holding gazebo extension elements in urdf

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

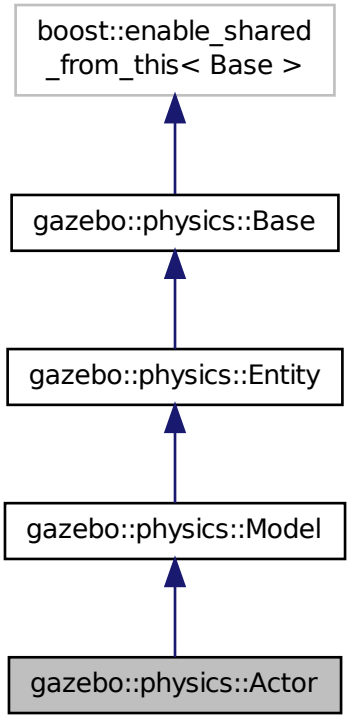
- **parser_urdf.hh**

10.2 gazebo::physics::Actor Class Reference

Actor (p. 111) 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 scipt was last updated.
- unsigned int **lastTraj**

  THe last trajectory.
- bool **loop**

  True if the animation should loop.
- **LinkPtr mainLink**

  Base *(p. 137) 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 scipt.
- std::map< std::string, **common::SkeletonAnimation** * > **skelAnimation**

  Skeleton animations.
- **common::Skeleton** * **skeleton**

  The actor's skeleton.
- std::map< std::string, std::map< std::string, std::string > > **skelNodesMap**

Skeleton to naode map.

- std::string **skinFile**

  Filename for the skin.

- double **skinScale**

  Scaling factor to apply to the skin.

- double **startDelay**

  Amount of time to delay start by.

- std::map< unsigned int, **common::PoseAnimation** * > **trajectories**

  All the trajectories.

- std::vector< **TrajectoryInfo** > **trajInfo**

  Trajectory information.

- std::string **visualName**

  Name of the visual.

Additional Inherited Members

10.2.1 Detailed Description

Actor (p. 111) class enables GPU based mesh model / skeleton scriptable animation.

10.2.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_parent*	Parent object

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

Destructor.

10.2.3 Member Function Documentation

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

Finalize the actor.

Reimplemented from **gazebo::physics::Model** (p. 493).

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

Get the SDF values for the actor.

Returns

Pointer to the SDF values.

Reimplemented from **gazebo::physics::Model** (p. 496).

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

Initialize the actor.

Reimplemented from **gazebo::physics::Model** (p. 497).

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

Returns true when actor is playing animation.

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

Load the actor.

Parameters

in	*_sdf*	SDF parameters

Reimplemented from **gazebo::physics::Entity** (p. 288).

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

Start playing the script.

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

Stop playing the script.

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

Update the actor.

Reimplemented from **gazebo::physics::Base** (p. 148).

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

update the parameters using new sdf values.

Parameters

in	*_sdf*	SDF values to update from.

Reimplemented from **gazebo::physics::Model** (p. 502).

10.2.4 Member Data Documentation

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

True if the actor is being updated.

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

True if the actor should start running automatically.

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

Where to send bone info.

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

True to interpolate along x direction.

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

Last position of the actor.

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

Time the scipt was last updated.

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

THe last trajectory.

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

True if the animation should loop.

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

Base (p. 137) link.

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

Pointer to the actor's mesh.

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

THe old action.

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

Length of the actor's path.

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

Time when the animation was started.

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

Time of the previous frame.

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

Time length of a scipt.

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

Skeleton animations.

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

The actor's skeleton.

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

Skeleton to naode map.

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

Filename for the skin.

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

Scaling factor to apply to the skin.

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

Amount of time to delay start by.

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

All the trajectories.

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

Trajectory information.

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

Name of the visual.

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

- **Actor.hh**

10.3 gazebo::math::Angle Class Reference

An angle and related functions.

`#include <math/gzmath.hh>`

Public Member Functions

- **Angle** ()
  Constructor.
- **Angle** (double _radian)
  Copy Constructor.
- **Angle** (const **Angle** &_angle)
  Copy constructor.
- virtual ~**Angle** ()
  Destructor.
- double **Degree** () const
  Get the angle in degrees.
- void **Normalize** ()
  Normalize the angle in the range -Pi to Pi.
- bool **operator!=** (const **Angle** &_angle) const
  Inequality.
- double **operator*** () const
  Dereference operator.
- **Angle operator*** (const **Angle** &_angle) const
  *Multiplication operator, result = this * _angle.*
- **Angle operator*=** (const **Angle** &_angle)
  *Multiplication set, this = this * _angle.*
- **Angle operator+** (const **Angle** &_angle) const
  Addition operator, result = this + _angle.
- **Angle operator+=** (const **Angle** &_angle)
  Addition set, this = this + _angle.
- **Angle operator-** (const **Angle** &_angle) const
  Substraction, result = this - _angle.
- **Angle operator-=** (const **Angle** &_angle)

Subtraction set, this = this - _angle.

- **Angle operator**/ (const **Angle** &_angle) const

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

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

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

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

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

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

  Greater or equal operator.
- double **Radian** () const

  Get the angle in radians.
- void **SetFromDegree** (double _degree)

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

  Set the value from an angle in radians.

Friends

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

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

  Stream extraction operator.

10.3.1 Detailed Description

An angle and related functions.

10.3.2 Constructor & Destructor Documentation

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

Constructor.

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

Copy Constructor.

Parameters

in	*_radian*	Radians

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

Copy constructor.

Parameters

in	_angle	**Angle** (p. 118) to copy

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

Destructor.

10.3.3 Member Function Documentation

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

Get the angle in degrees.

Returns

double containing the angle's degree value

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

Normalize the angle in the range -Pi to Pi.

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

Inequality.

Parameters

in	_angle	**Angle** (p. 118) to check for inequality

Returns

true if this != _angle

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

Dereference operator.

Returns

Double containing the angle's radian value

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

Multiplication operator, result = this * _angle.

Parameters

in	*_angle*	**Angle** (p. 118) for multiplication

Returns

the new angle

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

Multiplication set, this = this * _angle.

Parameters

in	*_angle*	**Angle** (p. 118) for multiplication

Returns

angle

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

Addition operator, result = this + _angle.

Parameters

in	*_angle*	**Angle** (p. 118) for addition

Returns

the new angle

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

Addition set, this = this + _angle.

Parameters

in	*_angle*	**Angle** (p. 118) for addition

Returns

angle

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

Substraction, result = this - _angle.

Parameters

in	*_angle*	**Angle** (p. 118) for substraction

Returns

the new angle

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

Subtraction set, this = this - _angle.

Parameters

in	*_angle*	**Angle** (p. 118) for subtraction

Returns

angle

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

Division, result = this / _angle.

Parameters

in	*_angle*	**Angle** (p. 118) for division

Returns

the new angle

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

Division set, this = this / _angle.

Parameters

in	*_angle*	**Angle** (p. 118) for division

Returns

angle

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

Less than operator.

Parameters

in	*_angle*	**Angle** (p. 118) to check

Returns

true if this < _angle

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

Less or equal operator.

Parameters

in	*_angle*	**Angle** (p. 118) to check

Returns

true if this <= _angle

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

Equality operator, result = this == _angle.

Parameters

in	*_angle*	**Angle** (p. 118) to check for equality

Returns

true if this == _angle

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

Greater than operator.

Parameters

in	*_angle*	**Angle** (p. 118) to check

Returns

true if this > _angle

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

Greater or equal operator.

Parameters

in	*_angle*	**Angle** (p. 118) to check

Returns

true if this >= _angle

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

Get the angle in radians.

Returns

double containing the angle's radian value

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

Set the value from an angle in degrees.

Parameters

in	*_degree*	Degree value

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

Set the value from an angle in radians.

Parameters

in	*_radian*	Radian value

10.3.4 Friends And Related Function Documentation

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

Stream insertion operator.

Outputs in degrees

Parameters

in	*_out*	output stream
in	*_a*	angle to output

Returns

The output stream

10.3.4.2 std::istream& operator>> (std::istream & *in*, gazebo::math::Angle & *a*) [friend]

Stream extraction operator.

Assumes input is in degrees

Parameters

in	input stream
pt	angle to read value into

Returns

The input stream

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

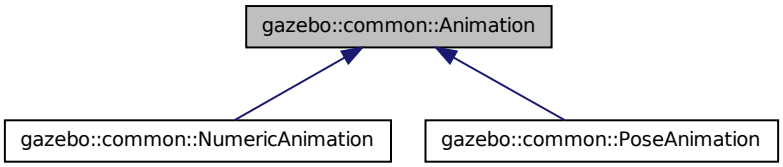
- **Angle.hh**

10.4 gazebo::common::Animation Class Reference

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

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::Animation:



Public Member Functions

- **Animation** (const std::string &_name, double _length, bool _loop)

  Constructor.
- virtual ~**Animation** ()

  Destructor.
- void **AddTime** (double _time)

  Add time to the animation.
- **KeyFrame** * **GetKeyFrame** (unsigned int _index) const

  Get a key frame using an index value.
- unsigned int **GetKeyFrameCount** () const

  Return the number of key frames in the animation.
- double **GetLength** () const

Return the duration of the animation.

- double **GetTime** () const

  Return the current time position.

- void **SetLength** (double _len)

  Set the duration of the animation.

- void **SetTime** (double _time)

  Set the current time position of the animation.

Protected Types

- typedef std::vector< **KeyFrame** * > **KeyFrame_V**

  array of keyframe type alias

Protected Member Functions

- double **GetKeyFramesAtTime** (double _time, **KeyFrame** **_kf1, **KeyFrame** **_kf2, unsigned int &_firstKeyIndex) const

  Get the two key frames that bound a time value.

Protected Attributes

- bool **build**

  determines if the interpolation splines need building

- **KeyFrame_V keyFrames**

  array of key frames

- double **length**

  animation duration

- bool **loop**

  true if animation repeats

- std::string **name**

  animation name

- double **timePos**

  current time position

10.4.1 Detailed Description

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

10.4.2 Member Typedef Documentation

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

array of keyframe type alias

10.4.3 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_name*	Name of the animation, should be unique
in	*_length*	Duration of the animation in seconds
in	*_loop*	Set to true if the animation should repeat

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

Destructor.

10.4.4 Member Function Documentation

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

Add time to the animation.

Parameters

in	*_time*	The amount of time to add in seconds

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

Get a key frame using an index value.

Parameters

in	*_index*	The index of the key frame

Returns

A (p. 111) pointer the keyframe, NULL if the _index is invalid

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

Return the number of key frames in the animation.

Returns

The number of keyframes

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

Get the two key frames that bound a time value.

Parameters

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

Returns

The time between the two keyframe

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

Return the duration of the animation.

Returns

Duration of the animation in seconds

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

Return the current time position.

Returns

The time position in seconds

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

Set the duration of the animation.

Parameters

in	*_len*	The length of the animation in seconds

10.4.4.8 void gazebo::common::Animation::SetTime (double *_time*)

Set the current time position of the animation.

Parameters

in	*_time*	The time position in seconds

10.4.5 Member Data Documentation

10.4.5.1 bool gazebo::common::Animation::build [mutable], [protected]

determines if the interpolation splines need building

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

array of key frames

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

animation duration

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

true if animation repeats

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

animation name

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

current time position

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

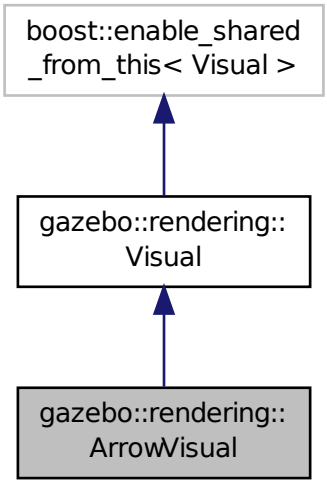
- **Animation.hh**

10.5 gazebo::rendering::ArrowVisual Class Reference

Basic arrow visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::ArrowVisual:



Public Member Functions

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

  Constructor.
- virtual ~**ArrowVisual** ()

  Destructor.
- virtual void **Load** ()

  Load the visual with default parameters.
- void **ShowRotation** ()

  Show the rotation of the visual.

Additional Inherited Members

10.5.1 Detailed Description

Basic arrow visualization.

10.5.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_name*	Name of the arrow visual
in	*_vis*	Pointer to the parent visual

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

Destructor.

10.5.3 Member Function Documentation

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

Load the visual with default parameters.

Reimplemented from **gazebo::rendering::Visual** (p. 898).

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

Show the rotation of the visual.

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

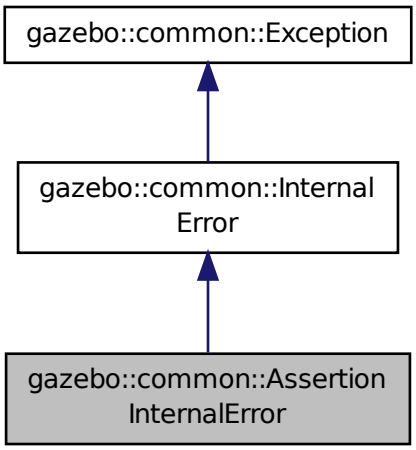
- **ArrowVisual.hh**

10.6 gazebo::common::AssertionInternalError Class Reference

Class for generating Exceptions which come from gazebo assertions.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::AssertionInternalError:



Public Member Functions

- **AssertionInternalError** (const char ∗_file, int _line, const std::string &_expr, const std::string &_function, const std::string &_msg="")

  Constructor for assertions.
- virtual ∼**AssertionInternalError** ()

  Destructor.

10.6.1 Detailed Description

Class for generating Exceptions which come from gazebo assertions.

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

10.6.2 Constructor & Destructor Documentation

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

Constructor for assertions.

Parameters

in	*_file*	File name
in	*_line*	Line number where the error occurred
in	*_expr*	Assertion expression failed resulting in an internal error

in	*_function*	Function where assertion failed
in	*_msg*	Function where assertion failed

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

Destructor.

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

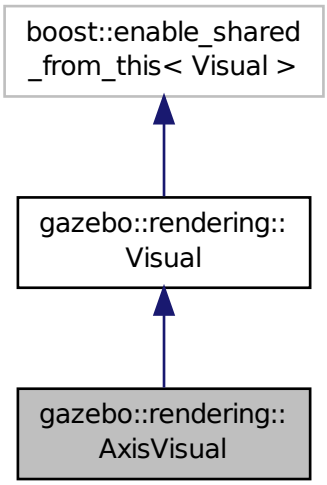
- **Exception.hh**

10.7 gazebo::rendering::AxisVisual Class Reference

Basic axis visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::AxisVisual:



Public Member Functions

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

  Constructor.
- virtual ~**AxisVisual** ()

  Destructor.
- virtual void **Load** ()

Load the axis visual.

- void **ScaleXAxis** (const **math::Vector3** &_scale)

  Scale the X axis.
- void **ScaleYAxis** (const **math::Vector3** &_scale)

  Scale the Y axis.
- void **ScaleZAxis** (const **math::Vector3** &_scale)

  Scale the Z axis.
- void **SetAxisMaterial** (unsigned int _axis, const std::string &_material)

  Set the material used to render and axis.
- void **ShowRotation** (unsigned int _axis)

  Load the rotation tube.

Additional Inherited Members

10.7.1 Detailed Description

Basic axis visualization.

10.7.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_name*	Name of the **AxisVisual** (p. 133)
in	*_vis*	Parent visual

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

Destructor.

10.7.3 Member Function Documentation

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

Load the axis visual.

Reimplemented from **gazebo::rendering::Visual** (p. 898).

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

Scale the X axis.

Parameters

in	*_scale*	Scaling factor

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

Scale the Y axis.

Parameters

in	*_scale*	Scaling factor

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

Scale the Z axis.

Parameters

in	*_scale*	Scaling factor

10.7.3.5 void gazebo::rendering::AxisVisual::SetAxisMaterial (unsigned int _axis, const std::string & _material)

Set the material used to render and axis.

Parameters

in	*_axis*	The number of the axis (0, 1, 2 = x,y,z)
in	*_material*	The name of the material to apply to the axis

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

Load the rotation tube.

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

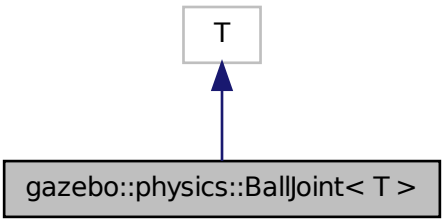
- **AxisVisual.hh**

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

Base (p. 137) class for a ball joint.

`#include <physics/physics.hh>`

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



Public Member Functions

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

  Constructor.
- virtual ~**BallJoint** ()

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

  *Template to ::Load the **BallJoint** (p. 135).*
- virtual void **SetAxis** (int, const **math::Vector3** &)
- virtual void **SetHighStop** (int, **math::Angle**)
- virtual void **SetLowStop** (int, **math::Angle**)

10.8.1 Detailed Description

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

Base (p. 137) class for a ball joint.

Each physics engine should implement this class.

10.8.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_parent*	Pointer to the parent link.

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

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

Destructor.

10.8.3 Member Function Documentation

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

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

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

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

Template to ::Load the **BallJoint** (p. 135).

Parameters

in	*_sdf*	SDF to load the joint from.

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

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

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

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

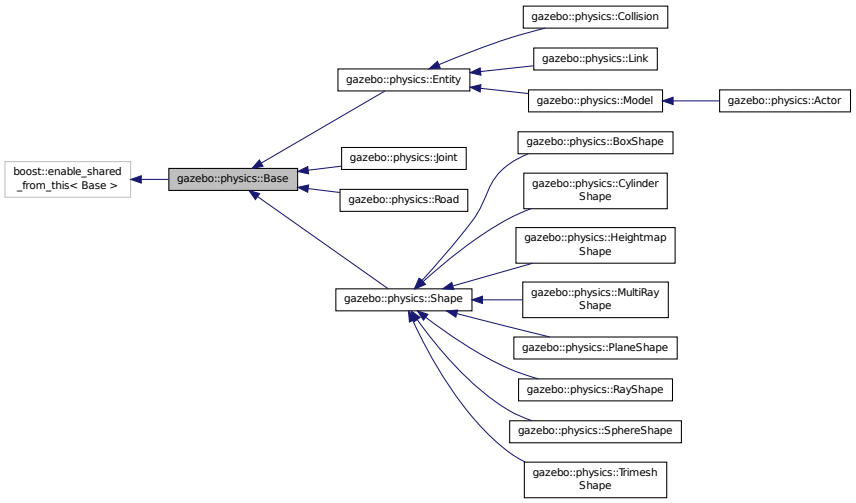
- **BallJoint.hh**

10.9 gazebo::physics::Base Class Reference

Base (p. 137) 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** ()

  Finialize the object.
- **BasePtr GetById** (unsigned int _id) const

This is an internal function.

- **BasePtr GetByName** (const std::string &_name)

  Get by name.
- **BasePtr GetChild** (unsigned int _i) const

  Get a child by index.
- **BasePtr GetChild** (const std::string &_name)

  Get a child by name.
- unsigned int **GetChildCount** () const

  Get the number of children.
- unsigned int **GetId** () const

  Return the ID of this entity.
- std::string **GetName** () const

  Return the name of the entity.
- **BasePtr GetParent** () const

  Get the parent.
- int **GetParentId** () const

  Return the ID of the parent.
- bool **GetSaveable** () const

  Get whether the object should be "saved", when the user selects to save the world to xml.
- std::string **GetScopedName** () const

  Return the name of this entity with the model scope world::model1::...::modelN::entityName.
- virtual const **sdf::ElementPtr GetSDF** ()

  Get the SDF values for the object.
- unsigned int **GetType** () const

  Get the full type definition.
- const **WorldPtr** & **GetWorld** () const

  *Get the **World** (p. 910) 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. 140)'s.*
- virtual void **SetName** (const std::string &_name)

  Set the name of the entity.
- void **SetParent** (**BasePtr** _parent)

  Set the parent.
- void **SetSaveable** (bool _v)

  Set whether the object should be "saved", when the user selects to save the world to xml.
- virtual bool **SetSelected** (bool _show)

  Set whether this entity has been selected by the user through the gui.
- void **SetWorld** (const **WorldPtr** &_newWorld)

  Set the world this object belongs to.
- virtual void **Update** ()

  Update the object.
- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  Update the parameters using new sdf values.

Protected Attributes

- **Base_V children**

  Children of this entity.
- Base_V::iterator **childrenEnd**

  End of the children vector.
- **BasePtr parent**

  Parent of this entity.
- **sdf::ElementPtr sdf**

  The SDF values for this object.
- **WorldPtr world**

  Pointer to the world.

10.9.1 Detailed Description

Base (p. 137) class for most physics classes.

10.9.2 Member Enumeration Documentation

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

Unique identifiers for all entity types.

Enumerator:

BASE **Base** (p. 137) type.

ENTITY **Entity** (p. 281) type.

MODEL **Model** (p. 489) type.

LINK **Link** (p. 418) type.

COLLISION **Collision** (p. 195) type.

ACTOR **Actor** (p. 111) type.

LIGHT Light type.

VISUAL Visual type.

JOINT **Joint** (p. 381) type.

BALL_JOINT **BallJoint** (p. 135) type.

HINGE2_JOINT Hing2Joint type.

HINGE_JOINT **HingeJoint** (p. 358) type.

SLIDER_JOINT **SliderJoint** (p. 747) type.

SCREW_JOINT **ScrewJoint** (p. 691) type.

UNIVERSAL_JOINT **UniversalJoint** (p. 826) type.

SHAPE **Shape** (p. 720) type.

BOX_SHAPE **BoxShape** (p. 153) type.

CYLINDER_SHAPE **CylinderShape** (p. 250) type.

HEIGHTMAP_SHAPE **HeightmapShape** (p. 352) type.

MAP_SHAPE MapShape type.

MULTIRAY_SHAPE **MultiRayShape** (p. 527) type.

RAY_SHAPE **RayShape** (p. 647) type.

PLANE_SHAPE **PlaneShape** (p. 590) type.

SPHERE_SHAPE **SphereShape** (p. 751) type.

TRIMESH_SHAPE **TrimeshShape** (p. 822) type.

10.9.3 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_parent*	Parent of this object

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

Destructor.

10.9.4 Member Function Documentation

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

Add a child to this entity.

Parameters

in	*_child*	Child entity.

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

Add a type specifier.

Parameters

in	*_type*	New type to append to this objects type definition.

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

Finialize the object.

Reimplemented in **gazebo::physics::Actor** (p. 114), **gazebo::physics::Model** (p. 493), **gazebo::physics::Entity** (p. 284), **gazebo::physics::Link** (p. 426), and **gazebo::physics::Collision** (p. 199).

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

This is an internal function.

Get a child or self by id.

Parameters

in	*_id*	ID of the object to retreive.

Returns

A (p. 111) pointer to the object, NULL if not found

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

Get by name.

Parameters

in	*_name*	Get a child (or self) object by name

Returns

A (p. 111) pointer to the object, NULL if not found

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

Get a child by index.

Parameters

in	*_i*	Index of the child to retreive.

Returns

A (p. 111) pointer to the object, NULL if the index is invalid.

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

Get a child by name.

Parameters

in	*_name*	Name of the child.

Returns

A (p. 111) pointer to the object, NULL if not found

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

Get the number of children.

Returns

The number of children.

10.9.4.9 unsigned int gazebo::physics::Base::GetId ( ) const

Return the ID of this entity.

This id is unique.

Returns

Integer ID.

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

Return the name of the entity.

Returns

Name of the entity.

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

Get the parent.

Returns

Pointer to the parent entity.

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

Return the ID of the parent.

Returns

Integer ID.

10.9.4.13 bool gazebo::physics::Base::GetSaveable ( ) const

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

Returns

True if the object is saveable.

10.9.4.14 std::string gazebo::physics::Base::GetScopedName ( ) const

Return the name of this entity with the model scope world::model1::...::modelN::entityName.

Returns

The scoped name.

10.9.4.15 virtual const sdf::ElementPtr gazebo::physics::Base::GetSDF ( ) [virtual]

Get the SDF values for the object.

Returns

The SDF values for the object.

Reimplemented in **gazebo::physics::Actor** (p. 114), and **gazebo::physics::Model** (p. 496).

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

Get the full type definition.

Returns

The full type definition.

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

Get the **World** (p. 910) this object is in.

Returns

The **World** (p. 910) this object is part of.

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

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

Parameters

in	*_t*	Type to check.

Returns

True if this object's type definition has the.

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

Initialize the object.

Reimplemented in **gazebo::physics::Joint** (p. 394), **gazebo::physics::RayShape** (p. 651), **gazebo::physics::Actor** (p. 115), **gazebo::physics::Model** (p. 497), **gazebo::physics::Link** (p. 432), **gazebo::physics::Collision** (p. 202), **gazebo::physics::HeightmapShape** (p. 355), **gazebo::physics::TrimeshShape** (p. 824), **gazebo::physics::MultiRayShape** (p. 534), **gazebo::physics::PlaneShape** (p. 592), **gazebo::physics::Road** (p. 667), **gazebo::physics::Shape** (p. 722), **gazebo::physics::SphereShape** (p. 753), **gazebo::physics::BoxShape** (p. 155), and **gazebo::physics::CylinderShape** (p. 253).

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

True if the entity is selected by the user.

Returns

True if the entity is selected.

10.9.4.21 virtual void gazebo::physics::Base::Load (sdf::ElementPtr _sdf) [virtual]

Load.

Parameters

in	*node*	Pointer to an SDF parameters

Reimplemented in **gazebo::physics::Joint** (p. 394), **gazebo::physics::Actor** (p. 115), **gazebo::physics::Entity** (p. 288), **gazebo::physics::Model** (p. 497), **gazebo::physics::Link** (p. 432), **gazebo::physics::Collision** (p. 202), **gazebo::physics::HeightmapShape** (p. 355), and **gazebo::physics::Road** (p. 667).

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

Returns true if the entities are the same.

Checks only the name.

Parameters

in	*_ent*	**Base** (p. 137) object to compare with.

Returns

True if the entities are the same.

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

Print this object to screen via gzmsg.

Parameters

in	*_prefix*	Usually a set of spaces.

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

Remove a child from this entity.

Parameters

in	*_id*	ID of the child to remove.

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

Remove a child by name.

Parameters

in	*_name*	Name of the child.

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

Remove all children.

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

Reset the object.

Reimplemented in **gazebo::physics::Joint** (p. 395), **gazebo::physics::Model** (p. 498), **gazebo::physics::Entity** (p. 289), and **gazebo::physics::Link** (p. 433).

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

Calls recursive Reset on one of the **Base::EntityType** (p. 140)'s.

Parameters

in	*_resetType*	The type of reset operation

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

Set the name of the entity.

Parameters

in	*_name*	New name.

Reimplemented in **gazebo::physics::Entity** (p. 290).

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

Set the parent.

Parameters

in	*_parent*	Parent object.

10.9.4.31 void gazebo::physics::Base::SetSaveable (bool *_v*)

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

Parameters

in	*_v*	Set to True if the object should be saved.

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

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

Parameters

in	*_show*	True to set this entity as selected.

Reimplemented in **gazebo::physics::Link** (p. 436).

10.9.4.33 void gazebo::physics::Base::SetWorld (const WorldPtr & *_newWorld*)

Set the world this object belongs to.

This will also set the world for all children.

Parameters

in	*_newWorld*	The new **World** (p. 910) this object is part of.

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

Update the object.

Reimplemented in **gazebo::physics::Joint** (p. 397), **gazebo::physics::MultiRayShape** (p. 534), **gazebo::physics::Actor** (p. 115), **gazebo::physics::RayShape** (p. 652), **gazebo::physics::Link** (p. 437), **gazebo::physics::Model** (p. 502), and **gazebo::physics::TrimeshShape** (p. 825).

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

Update the parameters using new sdf values.

Parameters

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

Reimplemented in **gazebo::physics::Joint** (p. 397), **gazebo::physics::Actor** (p. 115), **gazebo::physics::Model** (p. 502), **gazebo::physics::Link** (p. 437), **gazebo::physics::Entity** (p. 291), and **gazebo::physics::Collision** (p. 204).

10.9.5 Member Data Documentation

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

Children of this entity.

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

End of the children vector.

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

Parent of this entity.

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

The SDF values for this object.

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

Pointer to the world.

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

- **Base.hh**

10.10 gazebo::math::Box Class Reference

Mathematical representation of a box and related functions.

`#include <math/gzmath.hh>`

Public Member Functions

- **Box** ()
  Default constructor.
- **Box** (const **Vector3** &_min, const **Vector3** &_max)
  Constructor.
- **Box** (const **Box** &_b)
  Copy Constructor.
- virtual ~**Box** ()
  Destructor.
- **math::Vector3 GetCenter** () const
  Get the box center.
- **math::Vector3 GetSize** () const
  Get the size of the box.
- double **GetXLength** () const
  Get the length along the x dimension.
- double **GetYLength** () const
  Get the length along the y dimension.
- double **GetZLength** () const
  Get the length along the z dimension.
- void **Merge** (const **Box** &_box)
  Merge a box with this box.
- **Box operator+** (const **Box** &_b) const
  Addition operator.
- const **Box** & **operator+=** (const **Box** &_b)
  Addition set operator.
- **Box operator-** (const **Vector3** &_v)
  Subtract a vector from the min and max values.
- **Box** & **operator=** (const **Box** &_b)
  Assignment operator.
- bool **operator==** (const **Box** &_b)
  Equality test operatoer.

Public Attributes

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

Friends

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

10.10.1 Detailed Description

Mathematical representation of a box and related functions.

10.10.2 Constructor & Destructor Documentation

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

Default constructor.

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

Constructor.

Parameters

in	*_min*	Minimum corner of the box
in	*_max*	Maximum corner of the box

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

Copy Constructor.

Parameters

in	*_b*	**Box** (p. 148) to copy

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

Destructor.

10.10.3 Member Function Documentation

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

Get the box center.

Returns

The center position of the box

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

Get the size of the box.

Returns

Size of the box

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

Get the length along the x dimension.

Returns

Double value of the length in the x dimension

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

Get the length along the y dimension.

Returns

Double value of the length in the y dimension

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

Get the length along the z dimension.

Returns

Double value of the length in the z dimension

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

Merge a box with this box.

Parameters

in	*_box*	**Box** (p. 148) to add to this box

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

Addition operator.

result = this + _b

Parameters

in	*_b*	**Box** (p. 148) to add

Returns

The new box

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

Addition set operator.

this = this + _b

Parameters

in	*_b*	**Box** (p. 148) to add

Returns

This new box

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

Subtract a vector from the min and max values.

Parameters

_v	The vector to use during subtraction

Returns

The new box

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

Assignment operator.

Set this box to the parameter

Parameters

in	*_b*	**Box** (p. 148) to copy

Returns

The new box.

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

Equality test operatoer.

Parameters

in	*_b*	**Box** (p. 148) to test

Returns

True if equal

10.10.4 Friends And Related Function Documentation

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

Output operator.

Parameters

in	*_out*	Output stream
in	*_b*	**Box** (p. 148) to output to the stream

Returns

The stream

10.10.5 Member Data Documentation

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

Maximum corner of the box.

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

Minimum corner of the box.

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

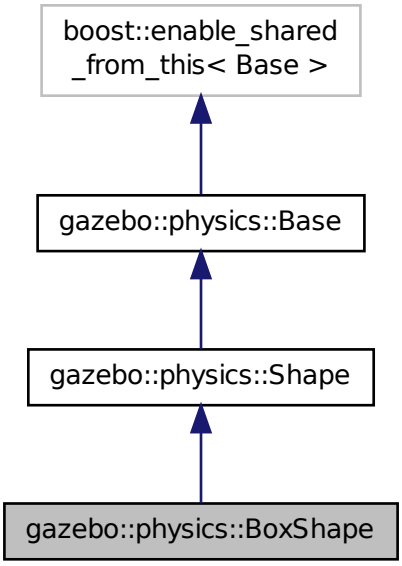
- **Box.hh**

10.11 gazebo::physics::BoxShape Class Reference

Box geometry primitive.

`#include <physics/physcs.hh>`

Inheritance diagram for gazebo::physics::BoxShape:



Public Member Functions

- **BoxShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**BoxShape** ()

  Destructor.
- void **FillMsg** (msgs::Geometry &_msg)

  Fill in the values for a geomertry message.
- **math::Vector3 GetSize** () const

  Get the size of the box.
- virtual void **Init** ()

  Initialize the box.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  Process a geometry message.
- virtual void **SetSize** (const **math::Vector3** &_size)

  Set the size of the box.

Additional Inherited Members

10.11.1 Detailed Description

Box geometry primitive.

10.11.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_parent*	Parent **Collision** (p. 195).

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

Destructor.

10.11.3 Member Function Documentation

10.11.3.1 void gazebo::physics::BoxShape::FillMsg (msgs::Geometry & _msg) [virtual]

Fill in the values for a geomertry message.

Parameters

out	*_msg*	The geometry message to fill.

Implements **gazebo::physics::Shape** (p. 722).

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

Get the size of the box.

Returns

The size of each side of the box.

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

Initialize the box.

Implements **gazebo::physics::Shape** (p. 722).

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

Process a geometry message.

Parameters

in	*_msg*	The message to set values from.

Implements **gazebo::physics::Shape** (p. 722).

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

Set the size of the box.

Parameters

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

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

- **BoxShape.hh**

10.12 gazebo::common::BVHLoader Class Reference

Handles loading BVH animation files.

`#include <common/common.hh>`

Public Member Functions

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

10.12.1 Detailed Description

Handles loading BVH animation files.

10.12.2 Constructor & Destructor Documentation

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

Constructor.

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

Desutrctor.

10.12.3 Member Function Documentation

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

Load a BVH file.

Parameters

in	*_filename*	BVH file to load
in	*_scale*	Scaling factor to apply to the skeleton

Returns

A (p. 111) pointer to a new **Skeleton** (p. 727)

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

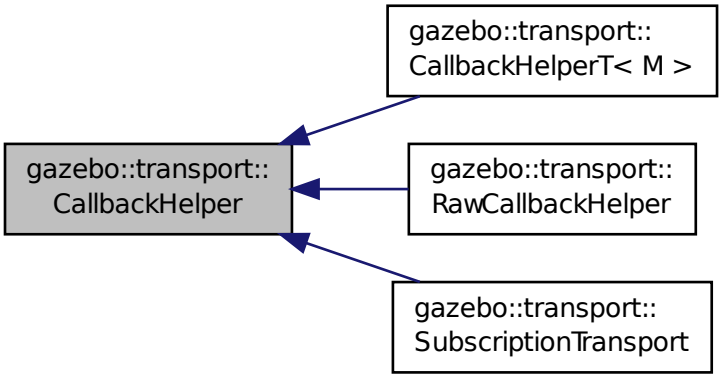
- **BVHLoader.hh**

10.13 gazebo::transport::CallbackHelper Class Reference

A (p. 111) helper class to handle callbacks when messages arrive.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::CallbackHelper:



Public Member Functions

- **CallbackHelper** (bool _latching=false)

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

  Destructor.
- unsigned int **GetId** () const

  Get the unique ID of this callback.
- bool **GetLatching** () const

  Is the callback latching?

- virtual std::string **GetMsgType** () const

  Get the typename of the message that is handled.
- virtual bool **HandleData** (const std::string &_newdata)=0

  Process new incoming data.
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)=0

  Process new incoming message.
- virtual bool **IsLocal** () const =0

  Is the callback local?

Protected Attributes

- bool **latching**

  True means that the callback helper will get the last published message on the topic.

10.13.1 Detailed Description

A (p. 111) helper class to handle callbacks when messages arrive.

10.13.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_latching*	Set to true to make the callback helper latching.

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

Destructor.

10.13.3 Member Function Documentation

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

Get the unique ID of this callback.

Returns

The unique ID of this callback.

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

Is the callback latching?

Returns

true if the callback is latching, false otherwise

10.13.3.3 virtual std::string gazebo::transport::CallbackHelper::GetMsgType ( ) const [virtual]

Get the typename of the message that is handled.

Returns

String representation of the message type

Reimplemented in **gazebo::transport::RawCallbackHelper** (p. 640), and **gazebo::transport::CallbackHelperT< M >** (p. 161).

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

Process new incoming data.

Parameters

in	*_newdata*	Incoming data to be processed

Returns

true if successfully processed; false otherwise

Implemented in **gazebo::transport::RawCallbackHelper** (p. 640), **gazebo::transport::CallbackHelperT< M >** (p. 161), and **gazebo::transport::SubscriptionTransport** (p. 779).

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

Process new incoming message.

Parameters

in	*_newMsg*	Incoming message to be processed

Returns

true if successfully processed; false otherwise

Implemented in **gazebo::transport::RawCallbackHelper** (p. 640), **gazebo::transport::CallbackHelperT< M >** (p. 162), and **gazebo::transport::SubscriptionTransport** (p. 779).

10.13.3.6 virtual bool gazebo::transport::CallbackHelper::IsLocal ( ) const [pure virtual]

Is the callback local?

Returns

true if the callback is local, false if the callback is tied to a remote connection

Implemented in **gazebo::transport::RawCallbackHelper** (p. 640), **gazebo::transport::CallbackHelperT< M >** (p. 162), and **gazebo::transport::SubscriptionTransport** (p. 780).

10.13.4 Member Data Documentation

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

True means that the callback helper will get the last published message on the topic.

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

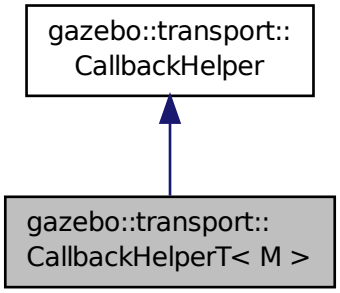
- **CallbackHelper.hh**

10.14 gazebo::transport::CallbackHelperT< M > Class Template Reference

Callback helper Template.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::CallbackHelperT< M >:



Public Member Functions

- **CallbackHelperT** (const boost::function< void(const M const > &)*&_cb, bool _latching=false)

  Constructor.
- std::string **GetMsgType** () const

  Get the typename of the message that is handled.
- virtual bool **HandleData** (const std::string &_newdata)

  Process new incoming data.
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)

Process new incoming message.

- virtual bool **IsLocal** () const

  Is the callback local?

Additional Inherited Members

10.14.1 Detailed Description

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

Callback helper Template.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 template<class M > gazebo::transport::CallbackHelperT< M >::CallbackHelperT (const boost::function< void(const M const > &) [inline]

Constructor.

Parameters

in	*_cb*	boost function to call on incoming messages
in	*_latching*	Set to true to make the callback helper latching.

10.14.3 Member Function Documentation

10.14.3.1 template<class M > std::string gazebo::transport::CallbackHelperT< M >::GetMsgType ( ) const [inline], [virtual]

Get the typename of the message that is handled.

Returns

String representation of the message type

Reimplemented from **gazebo::transport::CallbackHelper** (p. 159).

References gzthrow, and NULL.

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

Process new incoming data.

Parameters

in	*_newdata*	Incoming data to be processed

Returns

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 159).

10.14.3.3 template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::HandleMessage (MessagePtr _newMsg) [inline], [virtual]

Process new incoming message.

Parameters

in	*_newMsg*	Incoming message to be processed

Returns

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 159).

10.14.3.4 template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::IsLocal ( ) const [inline], [virtual]

Is the callback local?

Returns

true if the callback is local, false if the callback is tied to a remote connection

Implements **gazebo::transport::CallbackHelper** (p. 159).

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

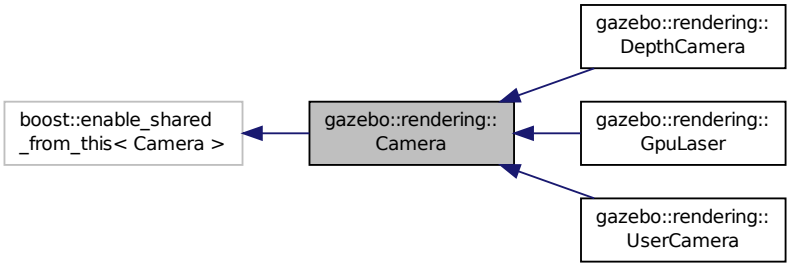
- **CallbackHelper.hh**

10.15 gazebo::rendering::Camera Class Reference

Basic camera sensor.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::Camera:



Public Member Functions

- **Camera** (const std::string &_namePrefix, **ScenePtr** _scene, bool _autoRender=true)

  Constructor.
- virtual ~**Camera** ()

  Destructor.
- void **AttachToVisual** (const std::string &_visualName, bool _inheritOrientation, double _minDist=0.0, double _maxDist=0.0)

  Attach the camera to a scene node.
- template<typename T >
  event::ConnectionPtr ConnectNewImageFrame (T _subscriber)

  Connect to the new image signal.
- void **CreateRenderTexture** (const std::string &_textureName)

  Set the render target.
- void **DisconnectNewImageFrame** (**event::ConnectionPtr** &_c)

  Disconnect from an image frame.
- void **EnableSaveFrame** (bool _enable)

  Enable or disable saving.
- virtual void **Fini** ()

  Finalize the camera.
- float **GetAspectRatio** () const

  Get the apect ratio.
- virtual float **GetAvgFPS** ()

  Get the average FPS.
- void **GetCameraToViewportRay** (int _screenx, int _screeny, **math::Vector3** &_origin, **math::Vector3** &_dir)

  Get a world space ray as cast from the camera through the viewport.
- **math::Vector3 GetDirection** () const

  Get the camera's direction vector.
- double **GetFarClip** ()

  Get the far clip distance.
- **math::Angle GetHFOV** () const

  Get the camera FOV (horizontal)

- size_t **GetImageByteSize** () const

  Get the image size in bytes.
- virtual const unsigned char * **GetImageData** (unsigned int i=0)

  Get a pointer to the image data.
- unsigned int **GetImageDepth** () const

  Get the depth of the image.
- std::string **GetImageFormat** () const

  Get the string representation of the image format.
- virtual unsigned int **GetImageHeight** () const

  Get the height of the image.
- virtual unsigned int **GetImageWidth** () const

  Get the width of the image.
- bool **GetInitialized** () const

  Return true if the camera has been initialized.
- **common::Time GetLastRenderWallTime** ()

  Get the last time the camera was rendered.
- std::string **GetName** () const

  Get the camera's name.
- double **GetNearClip** ()

  Get the near clip distance.
- Ogre::Camera * **GetOgreCamera** () const

  Get a pointer to the ogre camera.
- Ogre::SceneNode * **GetPitchNode** () const

  Get the camera's pitch scene node.
- double **GetRenderRate** () const

  Get the render Hz rate.
- Ogre::Texture * **GetRenderTexture** () const

  Get the render texture.
- **math::Vector3 GetRight** ()

  Get the viewport right vector.
- **ScenePtr GetScene** () const

  Get the scene this camera is in.
- Ogre::SceneNode * **GetSceneNode** () const

  Get the camera's scene node.
- std::string **GetScreenshotPath** () const

  Get the path to saved screenshots.
- unsigned int **GetTextureHeight** () const

  Get the height of the off-screen render texture.
- unsigned int **GetTextureWidth** () const

  Get the width of the off-screen render texture.
- virtual unsigned int **GetTriangleCount** ()

  Get the triangle count.
- **math::Vector3 GetUp** ()

  Get the viewport up vector.
- **math::Angle GetVFOV** () const

  Get the camera FOV (vertical)
- Ogre::Viewport * **GetViewport** () const

Get a pointer to the Ogre::Viewport.

- unsigned int **GetViewportHeight** () const

  Get the viewport height in pixels.

- unsigned int **GetViewportWidth** () const

  Get the viewport width in pixels.

- unsigned int **GetWindowId** () const

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

- bool **GetWorldPointOnPlane** (int _x, int _y, const **math::Plane** &_plane, **math::Vector3** &_result)

  Get point on a plane.

- **math::Pose GetWorldPose** ()

  Get the global pose of the camera.

- **math::Vector3 GetWorldPosition** () const

  Get the camera position in the world.

- **math::Quaternion GetWorldRotation** () const

  Get the camera's orientation in the world.

- double **GetZValue** (int _x, int _y)

  Get the Z-buffer value at the given image coordinate.

- virtual void **Init** ()

  Initialize the camera.

- bool **IsAnimating** () const

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

- bool **IsInitialized** () const **GAZEBO_DEPRECATED**(1.5)

  Deprecated.

- bool **IsVisible** (**VisualPtr** _visual)

  Return true if the visual is within the camera's view frustum.

- bool **IsVisible** (const std::string &_visualName)

  Return true if the visual is within the camera's view frustum.

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the camera with a set of parmeters.

- virtual void **Load** ()

  Load the camera with default parmeters.

- virtual bool **MoveToPosition** (const **math::Pose** &_pose, double _time)

  Move the camera to a position (this is an animated motion).

- bool **MoveToPositions** (const std::vector< **math::Pose** > &_pts, double _time, boost::function< void()> _onComplete=**NULL**)

  Move the camera to a series of poses (this is an animated motion).

- virtual void **PostRender** ()

  Post render.

- void **Render** ()

  Render the camera.

- void **RotatePitch** (**math::Angle** _angle)

  Rotate the camera around the pitch axis.

- void **RotateYaw** (**math::Angle** _angle)

  Rotate the camera around the yaw axis.

- bool **SaveFrame** (const std::string &_filename)

  Save the last frame to disk.

- void **SetAspectRatio** (float _ratio)

Set the aspect ratio.

- void **SetCaptureData** (bool _value)

  Set whether to capture data.

- void **SetCaptureDataOnce** ()

  Capture data once and save to disk.

- void **SetClipDist** (float _near, float _far)

  Set the clip distances.

- void **SetHFOV** (**math::Angle** _angle)

  Set the camera FOV (horizontal)

- void **SetImageHeight** (unsigned int _h)

  Set the image height.

- void **SetImageSize** (unsigned int _w, unsigned int _h)

  Set the image size.

- void **SetImageWidth** (unsigned int _w)

  Set the image height.

- void **SetName** (const std::string &_name)

  Set the camera's name.

- void **SetRenderRate** (double _hz)

  Set the render Hz rate.

- virtual void **SetRenderTarget** (Ogre::RenderTarget *_target)

  Set the camera's render target.

- void **SetSaveFramePathname** (const std::string &_pathname)

  Set the save frame pathname.

- void **SetScene** (**ScenePtr** _scene)

  Set the scene this camera is viewing.

- void **SetSceneNode** (Ogre::SceneNode *_node)

  Set the camera's scene node.

- void **SetWindowId** (unsigned int _windowId)

- virtual void **SetWorldPose** (const **math::Pose** &_pose)

  Set the global pose of the camera.

- void **SetWorldPosition** (const **math::Vector3** &_pos)

  Set the world position.

- void **SetWorldRotation** (const **math::Quaternion** &_quat)

  Set the world orientation.

- void **ShowWireframe** (bool _s)

  Set whether to view the world in wireframe.

- void **ToggleShowWireframe** ()

  Toggle whether to view the world in wireframe.

- void **TrackVisual** (const std::string &_visualName)

  Set the camera to track a scene node.

- void **Translate** (const **math::Vector3** &_direction)

  Translate the camera.

- virtual void **Update** ()

Static Public Member Functions

- static size_t **GetImageByteSize** (unsigned int _width, unsigned int _height, const std::string &_format)

  Calculate image byte size base on a few parameters.
- static bool **SaveFrame** (const unsigned char *_image, unsigned int _width, unsigned int _height, int _depth, const std::string &_format, const std::string &_filename)

  Save a frame using an image buffer.

Protected Member Functions

- virtual void **AnimationComplete** ()

  Internal function used to indicate that an animation has completed.
- virtual bool **AttachToVisualImpl** (const std::string &_name, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  Attach the camera to a scene node.
- virtual bool **AttachToVisualImpl** (**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. *185) call.*
- virtual bool **TrackVisualImpl** (**VisualPtr** _visual)

  Set the camera to track a scene node.

Protected Attributes

- Ogre::AnimationState * **animState**

  Animation state, used to animate the camera.
- unsigned char * **bayerFrameBuffer**

  Buffer for a bayer image frame.
- Ogre::Camera * **camera**

  The OGRE camera.
- bool **captureData**

  True to capture frames into an image buffer.
- bool **captureDataOnce**

  True to capture a frame once and save to disk.
- std::vector< **event::ConnectionPtr** > **connections**

  The camera's event connections.
- int **imageFormat**

  Format for saving images.
- int **imageHeight**

  Save image height.
- int **imageWidth**

  Save image width.

- bool **initialized**

  True if initialized.
- **common::Time lastRenderWallTime**

  Time the last frame was rendered.
- std::string **name**

  Name of the camera.
- bool **newData**

  True if new data is available.
- **event::EventT**< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)> **newImageFrame**

  Event triggered when a new frame is generated.
- boost::function< void()> **onAnimationComplete**

  User callback for when an animation completes.
- Ogre::SceneNode * **pitchNode**

  Scene *(p. 676) nod that controls camera pitch.*
- **common::Time prevAnimTime**

  Previous time the camera animation was updated.
- Ogre::RenderTarget * **renderTarget**

  Target that renders frames.
- Ogre::Texture * **renderTexture**

  Texture that receives results from rendering.
- std::list< msgs::Request > **requests**

  List of requests.
- unsigned int **saveCount**

  Number of saved frames.
- unsigned char * **saveFrameBuffer**
- **ScenePtr scene**

  Pointer to the scene.
- Ogre::SceneNode * **sceneNode**

  Scene *(p. 676) node that controls camera position.*
- std::string **screenshotPath**

  Path to saved screenshots.
- **sdf::ElementPtr sdf**

  Camera *(p. 162)'s SDF values.*
- unsigned int **textureHeight**

  Height of the render texture.
- unsigned int **textureWidth**

  Width of the render texture.
- Ogre::Viewport * **viewport**

  Viewport the ogre camera uses.
- unsigned int **windowId**

  ID of the window that the camera is attached to.

10.15.1 Detailed Description

Basic camera sensor.

This is the base class for all cameras.

10.15.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_namePrefix*	Unique prefix name for the camera.
in	*_scene*	**Scene** (p. 676) that will contain the camera
in	*_autoRender*	Almost everyone should leave this as true.

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

Destructor.

10.15.3 Member Function Documentation

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

Internal function used to indicate that an animation has completed.

Reimplemented in **gazebo::rendering::UserCamera** (p. 832).

10.15.3.2 void gazebo::rendering::Camera::AttachToVisual (const std::string & *_visualName,* bool *_inheritOrientation,* double *_minDist* = `0.0`, double *_maxDist* = `0.0`)

Attach the camera to a scene node.

Parameters

in	*_visualName*	Name of the visual to attach the camera to
in	*_inherit-Orientation*	True means camera acquires the visual's orientation
in	*_minDist*	Minimum distance the camera is allowed to get to the visual
in	*_maxDist*	Maximum distance the camera is allowd to get from the visual

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

Attach the camera to a scene node.

Parameters

in	*_visualName*	Name of the visual to attach the camera to
in	*_inherit-Orientation*	True means camera acquires the visual's orientation
in	*_minDist*	Minimum distance the camera is allowed to get to the visual
in	*_maxDist*	Maximum distance the camera is allowd to get from the visual

Returns

True on success

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

Attach the camera to a visual.

Parameters

in	*_visual*	The visual to attach the camera to
in	*_inherit-Orientation*	True means camera acquires the visual's orientation
in	*_minDist*	Minimum distance the camera is allowed to get to the visual
in	*_maxDist*	Maximum distance the camera is allowd to get from the visual

Returns

True on success

Reimplemented in **gazebo::rendering::UserCamera** (p. 832).

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

Connect to the new image signal.

Parameters

in	*_subscriber*	Callback that is called when a new image is generated

Returns

A (p. 111) pointer to the connection. This must be kept in scope.

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

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

Set the render target.

Parameters

in	*_textureName*	Name of the new render texture

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

Disconnect from an image frame.

Parameters

in	*_c*	The connection to disconnect

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

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

Enable or disable saving.

Parameters

in	*_enable*	Set to True to enable saving of frames

10.15.3.9 virtual void gazebo::rendering::Camera::Fini ( ) [virtual]

Finalize the camera.

This function is called before the camera is destructed

Reimplemented in **gazebo::rendering::GpuLaser** (p. 327), **gazebo::rendering::DepthCamera** (p. 257), and **gazebo::rendering::UserCamera** (p. 833).

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

Get the apect ratio.

Returns

The aspect ratio (width / height) in pixels

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

Get the average FPS.

Returns

The average frames per second

10.15.3.12 void gazebo::rendering::Camera::GetCameraToViewportRay (int *_screenx*, int *_screeny*, math::Vector3 & *_origin*, math::Vector3 & *_dir*)

Get a world space ray as cast from the camera through the viewport.

Parameters

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

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

Get the camera's direction vector.

Returns

Direction the camera is facing

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

Get the far clip distance.

Returns

Far clip distance

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

Get the next frame filename based on SDF parameters.

Returns

The frame's filename

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

Get the camera FOV (horizontal)

Returns

The horizontal field of view

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

Get the image size in bytes.

Returns

Size in bytes

10.15.3.18 static size_t gazebo::rendering::Camera::GetImageByteSize (unsigned int *_width*, unsigned int *_height*, const std::string & *_format*) [static]

Calculate image byte size base on a few parameters.

Parameters

in	*_width*	Width of an image
in	*_height*	Height of an image
in	*_format*	Image format

Returns

Size of an image based on the parameters

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

Get a pointer to the image data.

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

Parameters

in	*_i*	Index of the camera's texture (0 = RGB, 1 = depth).

Returns

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

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

Get the depth of the image.

Returns

Depth of the image

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

Get the string representation of the image format.

Returns

String representation of the image format.

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

Get the height of the image.

Returns

Image height

Reimplemented in **gazebo::rendering::UserCamera** (p. 833).

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

Get the width of the image.

Returns

Image width

Reimplemented in **gazebo::rendering::UserCamera** (p. 834).

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

Return true if the camera has been initialized.

Returns

True if initialized was successful

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

Get the last time the camera was rendered.

Returns

Time the camera was last rendered

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

Get the camera's name.

Returns

The name of the camera

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

Get the near clip distance.

Returns

Near clip distance

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

Get a pointer to the ogre camera.

Returns

Pointer to the OGRE camera

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

Get the camera's pitch scene node.

Returns

The pitch node the camera is attached to

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

Get the render Hz rate.

Returns

The Hz rate

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

Get the render texture.

Returns

Pointer to the render texture

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

Get the viewport right vector.

Returns

The viewport right vector

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

Get the scene this camera is in.

Returns

Pointer to scene containing this camera

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

Get the camera's scene node.

Returns

The scene node the camera is attached to

10.15.3.35 std::string gazebo::rendering::Camera::GetScreenshotPath ( ) const

Get the path to saved screenshots.

Returns

Path to saved screenshots.

10.15.3.36 unsigned int gazebo::rendering::Camera::GetTextureHeight ( ) const

Get the height of the off-screen render texture.

Returns

Render texture height

10.15.3.37 unsigned int gazebo::rendering::Camera::GetTextureWidth ( ) const

Get the width of the off-screen render texture.

Returns

Render texture width

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

Get the triangle count.

Returns

The current triangle count

10.15.3.39 math::Vector3 gazebo::rendering::Camera::GetUp ( )

Get the viewport up vector.

Returns

The viewport up vector

10.15.3.40 math::Angle gazebo::rendering::Camera::GetVFOV ( ) const

Get the camera FOV (vertical)

Returns

The vertical field of view

10.15.3.41 Ogre::Viewport* gazebo::rendering::Camera::GetViewport ( ) const

Get a pointer to the Ogre::Viewport.

Returns

Pointer to the Ogre::Viewport

10.15.3.42 unsigned int gazebo::rendering::Camera::GetViewportHeight ( ) const

Get the viewport height in pixels.

Returns

The viewport height

10.15.3.43 unsigned int gazebo::rendering::Camera::GetViewportWidth ( ) const

Get the viewport width in pixels.

Returns

The viewport width

10.15.3.44 unsigned int gazebo::rendering::Camera::GetWindowId ( ) const

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

Returns

The ID of the window.

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

Get point on a plane.

Parameters

in	*_x*	X cooridnate in camera's viewport, in pixels
in	*_y*	Y cooridnate in camera's viewport, in pixels
in	*_plane*	Plane on which to find the intersecting point
out	*_result*	Point on the plane

Returns

True if a valid point was found

10.15.3.46 math::Pose gazebo::rendering::Camera::GetWorldPose ( )

Get the global pose of the camera.

Returns

Pose of the camera in the world coordinate frame

10.15.3.47 math::Vector3 gazebo::rendering::Camera::GetWorldPosition ( ) const

Get the camera position in the world.

Returns

The world position of the camera

10.15.3.48 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. 623)

10.15.3.49 double gazebo::rendering::Camera::GetZValue (int _x, int _y)

Get the Z-buffer value at the given image coordinate.

Parameters

in	_x	Image coordinate; (0, 0) specifies the top-left corner.
in	_y	Image coordinate; (0, 0) specifies the top-left corner.

Returns

Image z value; note that this is abitrarily scaled and is *not* the same as the depth value.

10.15.3.50 virtual void gazebo::rendering::Camera::Init ( ) [virtual]

Initialize the camera.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 327), **gazebo::rendering::DepthCamera** (p. 257), and **gazebo::rendering::UserCamera** (p. 835).

10.15.3.51 bool gazebo::rendering::Camera::IsAnimating ( ) const

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

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

Deprecated.

See Also

GetInitialized (p. 174)

10.15.3.53 bool gazebo::rendering::Camera::IsVisible (VisualPtr *_visual*)

Return true if the visual is within the camera's view frustum.

Parameters

in	*_visual*	The visual to check for visibility

Returns

True if the _visual is in the camera's frustum

10.15.3.54 bool gazebo::rendering::Camera::IsVisible (const std::string & *_visualName*)

Return true if the visual is within the camera's view frustum.

Parameters

in	*_visualName*	Name of the visual to check for visibility

Returns

True if the _visual is in the camera's frustum

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

Load the camera with a set of parmeters.

Parameters

in	*_sdf*	The SDF camera info

Reimplemented in **gazebo::rendering::UserCamera** (p. 835).

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

Load the camera with default parmeters.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 327), **gazebo::rendering::DepthCamera** (p. 257), and **gazebo::rendering::UserCamera** (p. 835).

10.15.3.57 virtual bool gazebo::rendering::Camera::MoveToPosition (const math::Pose & _pose, double _time) [virtual]

Move the camera to a position (this is an animated motion).

See Also

Camera::MoveToPositions (p. 180)

Parameters

in	*_pose*	End position of the camera
in	*_time*	Duration of the camera's movement

Reimplemented in **gazebo::rendering::UserCamera** (p. 836).

10.15.3.58 bool gazebo::rendering::Camera::MoveToPositions (const std::vector< math::Pose > & _pts, double _time, boost::function< void() > _onComplete = NULL)

Move the camera to a series of poses (this is an animated motion).

See Also

Camera::MoveToPosition (p. 180)

Parameters

in	*_pts*	Vector of poses to move to
in	*_time*	Duration of the entire move
in	*_onComplete*	Callback that is called when the move is complete

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

Post render.

Called afer the render signal.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 327), **gazebo::rendering::DepthCamera** (p. 257), and **gazebo::rendering::UserCamera** (p. 836).

10.15.3.60 void gazebo::rendering::Camera::Render ( )

Render the camera.

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

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

Implementation of the render call.

10.15.3.62 void gazebo::rendering::Camera::RotatePitch (math::Angle _angle)

Rotate the camera around the pitch axis.

Parameters

in	*_angle*	Pitch amount

10.15.3.63 void gazebo::rendering::Camera::RotateYaw (math::Angle _angle)

Rotate the camera around the yaw axis.

Parameters

in	*_angle*	Rotation amount

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

Save the last frame to disk.

Parameters

in	*_filename*	File in which to save a single frame

Returns

True if saving was successful

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

Save a frame using an image buffer.

Parameters

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

Returns

True if saving was successful

10.15.3.66 void gazebo::rendering::Camera::SetAspectRatio (float _ratio)

Set the aspect ratio.

Parameters

in	*_ratio*	The aspect ratio (width / height) in pixels

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

Set whether to capture data.

Parameters

in	*_value*	Set to true to capture data into a memory buffer.

10.15.3.68 void gazebo::rendering::Camera::SetCaptureDataOnce ( )

Capture data once and save to disk.

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

Set the clip distances.

Parameters

in	*_near*	Near clip distance in meters
in	*_far*	Far clip distance in meters

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

Set the camera FOV (horizontal)

Parameters

in	*_radians*	Horizontal field of view

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

Set the image height.

Parameters

in	*_h*	Image height

10.15.3.72 void gazebo::rendering::Camera::SetImageSize (unsigned int *_w,* unsigned int *_h*)

Set the image size.

Parameters

in	*_w*	Image width
in	*_h*	Image height

10.15.3.73 void gazebo::rendering::Camera::SetImageWidth (unsigned int _w)

Set the image height.

Parameters

in	*_w*	Image width

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

Set the camera's name.

Parameters

in	*_name*	New name for the camera

10.15.3.75 void gazebo::rendering::Camera::SetRenderRate (double _hz)

Set the render Hz rate.

Parameters

in	*_hz*	The Hz rate

10.15.3.76 virtual void gazebo::rendering::Camera::SetRenderTarget (Ogre::RenderTarget * _target) [virtual]

Set the camera's render target.

Parameters

in	*_target*	Pointer to the render target

Reimplemented in **gazebo::rendering::UserCamera** (p. 837).

10.15.3.77 void gazebo::rendering::Camera::SetSaveFramePathname (const std::string & _pathname)

Set the save frame pathname.

Parameters

in	*_pathname*	Directory in which to store saved image frames

10.15.3.78 void gazebo::rendering::Camera::SetScene (ScenePtr _scene)

Set the scene this camera is viewing.

Parameters

in	*_scene*	Pointer to the scene

10.15.3.79 void gazebo::rendering::Camera::SetSceneNode (Ogre::SceneNode * _node)

Set the camera's scene node.

Parameters

in	*_node*	The scene nodes to attach the camera to

10.15.3.80 void gazebo::rendering::Camera::SetWindowId (unsigned int _windowId)

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

Set the global pose of the camera.

Parameters

in	*_pose*	The new **math::Pose** (p. 596) of the camera

Reimplemented in **gazebo::rendering::UserCamera** (p. 837).

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

Set the world position.

Parameters

in	*_pos*	The new position of the camera

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

Set the world orientation.

Parameters

in	*_quat*	The new orientation of the camera

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

Set whether to view the world in wireframe.

Parameters

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

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

Toggle whether to view the world in wireframe.

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

Set the camera to track a scene node.

Parameters

in	*_visualName*	Name of the visual to track

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

Implementation of the **Camera::TrackVisual** (p. 185) call.

Parameters

in	*_visualName*	Name of the visual to track

Returns

True if able to track the visual

10.15.3.88 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. 838).

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

Translate the camera.

Parameters

in	*_direction*	The translation vector

10.15.3.90 virtual void gazebo::rendering::Camera::Update ( ) [virtual]

Reimplemented in **gazebo::rendering::UserCamera** (p. 838).

10.15.4 Member Data Documentation

10.15.4.1 Ogre::AnimationState* gazebo::rendering::Camera::animState [protected]

Animation state, used to animate the camera.

10.15.4.2 unsigned char* gazebo::rendering::Camera::bayerFrameBuffer [protected]

Buffer for a bayer image frame.

10.15.4.3 Ogre::Camera* gazebo::rendering::Camera::camera [protected]

The OGRE camera.

10.15.4.4 bool gazebo::rendering::Camera::captureData [protected]

True to capture frames into an image buffer.

10.15.4.5 bool gazebo::rendering::Camera::captureDataOnce [protected]

True to capture a frame once and save to disk.

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

The camera's event connections.

10.15.4.7 int gazebo::rendering::Camera::imageFormat [protected]

Format for saving images.

10.15.4.8 int gazebo::rendering::Camera::imageHeight [protected]

Save image height.

10.15.4.9 int gazebo::rendering::Camera::imageWidth [protected]

Save image width.

10.15.4.10 bool gazebo::rendering::Camera::initialized [protected]

True if initialized.

10.15.4.11 common::Time gazebo::rendering::Camera::lastRenderWallTime [protected]

Time the last frame was rendered.

10.15.4.12 std::string gazebo::rendering::Camera::name [protected]

Name of the camera.

10.15.4.13 bool gazebo::rendering::Camera::newData [protected]

True if new data is available.

10.15.4.14 event::EventT<void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)> gazebo::rendering::Camera::newImageFrame [protected]

Event triggered when a new frame is generated.

Referenced by ConnectNewImageFrame(), and DisconnectNewImageFrame().

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

User callback for when an animation completes.

10.15.4.16 Ogre::SceneNode* gazebo::rendering::Camera::pitchNode [protected]

Scene (p. 676) nod that controls camera pitch.

10.15.4.17 common::Time gazebo::rendering::Camera::prevAnimTime [protected]

Previous time the camera animation was updated.

10.15.4.18 Ogre::RenderTarget* gazebo::rendering::Camera::renderTarget [protected]

Target that renders frames.

10.15.4.19 Ogre::Texture* gazebo::rendering::Camera::renderTexture [protected]

Texture that receives results from rendering.

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

List of requests.

10.15.4.21 unsigned int gazebo::rendering::Camera::saveCount [protected]

Number of saved frames.

10.15.4.22 unsigned char* gazebo::rendering::Camera::saveFrameBuffer [protected]

10.15.4.23 ScenePtr gazebo::rendering::Camera::scene [protected]

Pointer to the scene.

10.15.4.24 Ogre::SceneNode* gazebo::rendering::Camera::sceneNode [protected]

Scene (p. 676) node that controls camera position.

10.15.4.25 std::string gazebo::rendering::Camera::screenshotPath [protected]

Path to saved screenshots.

10.15.4.26 sdf::ElementPtr gazebo::rendering::Camera::sdf [protected]

Camera (p. 162)'s SDF values.

10.15.4.27 unsigned int gazebo::rendering::Camera::textureHeight [protected]

Height of the render texture.

10.15.4.28 unsigned int gazebo::rendering::Camera::textureWidth [protected]

Width of the render texture.

10.15.4.29 Ogre::Viewport* gazebo::rendering::Camera::viewport [protected]

Viewport the ogre camera uses.

10.15.4.30 unsigned int gazebo::rendering::Camera::windowId [protected]

ID of the window that the camera is attached to.

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

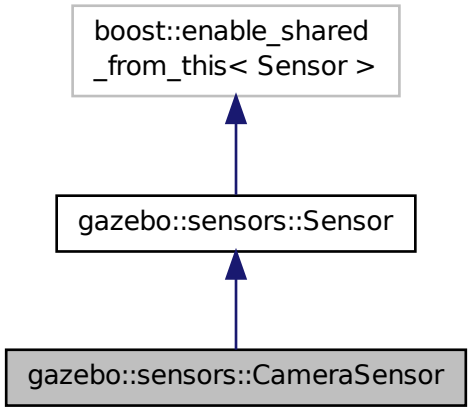
- **Camera.hh**

10.16 gazebo::sensors::CameraSensor Class Reference

Basic camera sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::CameraSensor:



Public Member Functions

- **CameraSensor** ()
  Constructor.
- virtual ~**CameraSensor** ()
  Destructor.
- **rendering::CameraPtr GetCamera** () const
  *Returns a pointer to the **rendering::Camera*** (p. *162*).
- const unsigned char * **GetImageData** ()
  Gets the raw image data from the sensor.
- unsigned int **GetImageHeight** () const
  Gets the height of the image in pixels.
- unsigned int **GetImageWidth** () const
  Gets the width of the image in pixels.
- virtual std::string **GetTopic** () const
  Gets the topic name of the sensor.
- virtual void **Init** ()
  Initialize the camera.
- virtual bool **IsActive** ()
  Returns true if sensor generation is active.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** _sdf)
  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
  Load the sensor with default parameters.
- bool **SaveFrame** (const std::string &_filename)
  Saves the image to the disk.
- virtual void **SetParent** (const std::string &_name)
  Set the parent of the sensor.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the camera.
- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.

Additional Inherited Members

10.16.1 Detailed Description

Basic camera sensor.

This sensor is used for simulating standard monocular cameras

10.16.2 Constructor & Destructor Documentation

10.16.2.1 gazebo::sensors::CameraSensor::CameraSensor ( )

Constructor.

10.16.2.2 virtual gazebo::sensors::CameraSensor::~CameraSensor ( ) [virtual]

Destructor.

10.16.3 Member Function Documentation

10.16.3.1 virtual void gazebo::sensors::CameraSensor::Fini ( ) [protected], [virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 702).

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

Returns a pointer to the **rendering::Camera** (p. 162).

Returns

The Pointer to the camera sensor.

10.16.3.3 const unsigned char* gazebo::sensors::CameraSensor::GetImageData ( )

Gets the raw image data from the sensor.

Returns

The pointer to the image data array.

10.16.3.4 unsigned int gazebo::sensors::CameraSensor::GetImageHeight ( ) const

Gets the height of the image in pixels.

Returns

The image height in pixels.

10.16.3.5 unsigned int gazebo::sensors::CameraSensor::GetImageWidth ( ) const

Gets the width of the image in pixels.

Returns

The image width in pixels.

10.16.3.6 virtual std::string gazebo::sensors::CameraSensor::GetTopic ( ) const [virtual]

Gets the topic name of the sensor.

Returns

Topic name

Todo to be implemented

Reimplemented from **gazebo::sensors::Sensor** (p. 703).

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

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.16.3.8 virtual bool gazebo::sensors::CameraSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

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

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 698) parameters
in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.16.3.10 virtual void gazebo::sensors::CameraSensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

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

Saves the image to the disk.

Parameters

in	*_filename*	The name of the file to be saved.

Returns

True if successful, false if unsuccessful.

10.16.3.12 virtual void gazebo::sensors::CameraSensor::SetParent (const std::string & *_name*) [virtual]

Set the parent of the sensor.

Parameters

_name	The name of the parent

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

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

Update the sensor information.

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 706).

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

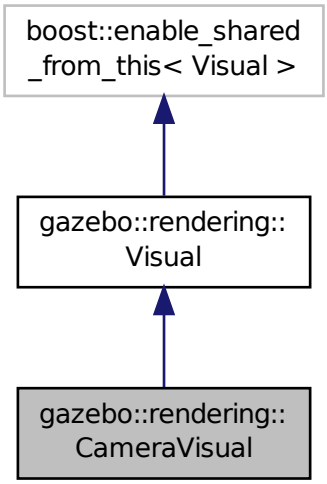
- **CameraSensor.hh**

10.17 gazebo::rendering::CameraVisual Class Reference

Basic camera visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::CameraVisual:



Public Member Functions

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

  Constructor.
- virtual ~**CameraVisual** ()

  Destructor.
- void **Load** (unsigned int _width, unsigned int _height)

  *Load the **Visual** (*p. *885).*

Additional Inherited Members

10.17.1 Detailed Description

Basic camera visualization.

This class is used to visualize a camera image generated from a CameraSensor. The sensor's image is drawn on a billboard in the 3D environment.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 gazebo::rendering::CameraVisual::CameraVisual (const std::string & *_name*, VisualPtr *_vis*)

Constructor.

Parameters

in	*_name*	Name of the **Visual** (p. 885)
in	*_vis*	Pointer to the parent **Visual** (p. 885)

10.17.2.2 virtual gazebo::rendering::CameraVisual::~CameraVisual ( ) [virtual]

Destructor.

10.17.3 Member Function Documentation

10.17.3.1 void gazebo::rendering::CameraVisual::Load (unsigned int *_width*, unsigned int *_height*)

Load the **Visual** (p. 885).

Parameters

in	*_width*	Width of the **Camera** (p. 162) image
in	*_height*	Height of the **Camera** (p. 162) image

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

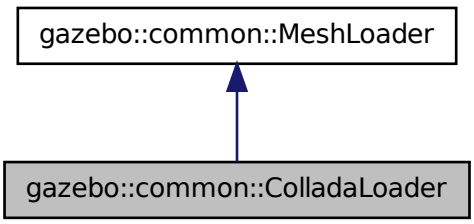
- **CameraVisual.hh**

10.18 gazebo::common::ColladaLoader Class Reference

Class used to load Collada mesh files.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::ColladaLoader:



Public Member Functions

- **ColladaLoader** ()

  Constructor.
- virtual ∼**ColladaLoader** ()

  Destructor.
- virtual **Mesh** ∗ **Load** (const std::string &_filename)

  Load a mesh.

10.18.1 Detailed Description

Class used to load Collada mesh files.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 gazebo::common::ColladaLoader::ColladaLoader ( )

Constructor.

10.18.2.2 virtual gazebo::common::ColladaLoader::∼ColladaLoader ( ) [virtual]

Destructor.

10.18.3 Member Function Documentation

10.18.3.1 virtual Mesh∗ gazebo::common::ColladaLoader::Load (const std::string & *_filename*) [virtual]

Load a mesh.

Parameters

in	*_filename*	Collada file to load

Returns

Pointer to a new **Mesh** (p. 475)

Implements **gazebo::common::MeshLoader** (p. 484).

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

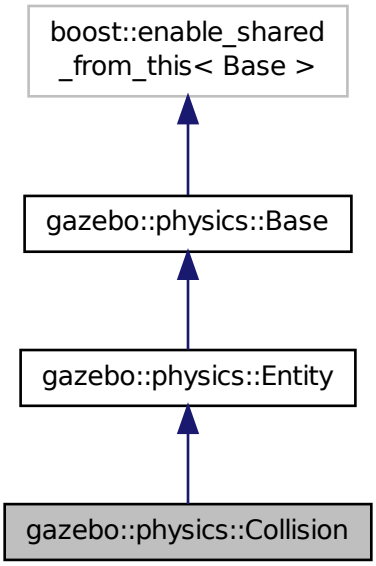
- **ColladaLoader.hh**

10.19 gazebo::physics::Collision Class Reference

Base (p. 137) 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 occurance of a contact to this collision.
- template<typename T >
  event::ConnectionPtr ConnectContact (T _subscriber)

  Deprecated.
- void **DisconnectContact** (**event::ConnectionPtr** &_conn)

  Deprecated.
- void **FillMsg** (msgs::Collision &_msg)

  Fill a collision message.
- virtual void **Fini** ()

  Finalize the collision.
- virtual **math::Box GetBoundingBox** () const =0

  Get the bounding box for this collision.
- bool **GetContactsEnabled** () const

  Return true of 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 collsion 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. 720).

Additional Inherited Members

10.19.1 Detailed Description

Base (p. 137) class for all collision entities.

10.19.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_link*	**Link** (p. 418) that contains this collision object.

10.19.2.2 virtual gazebo::physics::Collision::~Collision ( ) [virtual]

Destructor.

10.19.3 Member Function Documentation

10.19.3.1 void gazebo::physics::Collision::AddContact (const Contact & *_contact*)

Add an occurance of a contact to this collision.

Parameters

in	*_contact*	The contact which was detected by a collision engine.

10.19.3.2 template<typename T > event::ConnectionPtr gazebo::physics::Collision::ConnectContact (T _subscriber) [inline]

Deprecated.

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

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

Deprecated.

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

10.19.3.4 void gazebo::physics::Collision::FillMsg (msgs::Collision & _msg)

Fill a collision message.

Parameters

out	*_msg*	The message to fill with this collision's data.

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

Finalize the collision.

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

10.19.3.6 virtual math::Box gazebo::physics::Collision::GetBoundingBox ( ) const [pure virtual]

Get the bounding box for this collision.

Returns

The bounding box.

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

10.19.3.7 bool gazebo::physics::Collision::GetContactsEnabled ( ) const

Return true of contacts are on.

Returns

True of contact are on.

10.19.3.8 float gazebo::physics::Collision::GetLaserRetro ( ) const

Get the laser retro reflectiveness.

Returns

The laser retro value.

10.19.3.9 LinkPtr gazebo::physics::Collision::GetLink ( ) const

Get the link this collision belongs to.

Returns

The parent **Link** (p. 418).

10.19.3.10 ModelPtr gazebo::physics::Collision::GetModel ( ) const

Get the model this collision belongs to.

Returns

The parent model.

10.19.3.11 virtual math::Vector3 gazebo::physics::Collision::GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the collision.

Returns

The angular acceleration of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 286).

10.19.3.12 virtual math::Vector3 gazebo::physics::Collision::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the collision.

Returns

The angular velocity of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 286).

10.19.3.13 virtual math::Vector3 gazebo::physics::Collision::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the collision.

Returns

The linear acceleration of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 286).

10.19.3.14 virtual math::Vector3 gazebo::physics::Collision::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the collision.

Returns

The linear velocity relative to the parent model.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.19.3.15 ShapePtr gazebo::physics::Collision::GetShape ( ) const

Get the collision shape.

Returns

The collision shape.

10.19.3.16 unsigned int gazebo::physics::Collision::GetShapeType ( )

Get the shape type.

Returns

The shape type.

See Also

EntityType (p. 140)

10.19.3.17 CollisionState gazebo::physics::Collision::GetState ( )

Get the collision state.

Returns

The collision state.

10.19.3.18 SurfaceParamsPtr gazebo::physics::Collision::GetSurface ( ) const [inline]

Get the surface parameters.

Returns

The surface parameters.

10.19.3.19 virtual math::Vector3 gazebo::physics::Collision::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the collision in the world frame.

Returns

The angular acceleration of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.19.3.20 virtual math::Vector3 gazebo::physics::Collision::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the collision in the world frame.

Returns

The angular velocity of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.19.3.21 virtual math::Vector3 gazebo::physics::Collision::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the collision in the world frame.

Returns

The linear acceleration of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.19.3.22 virtual math::Vector3 gazebo::physics::Collision::GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the collision in the world frame.

Returns

The linear velocity of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 288).

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

Initialize the collision.

Reimplemented from **gazebo::physics::Base** (p. 145).

10.19.3.24 bool gazebo::physics::Collision::IsPlaceable ( ) const

Return whether this collision is movable.

Example on an immovable object is a ray.

Returns

True if the object is immovable.

10.19.3.25 virtual void gazebo::physics::Collision::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the collision.

Parameters

in	*_sdf*	SDF to load from.

Reimplemented from **gazebo::physics::Entity** (p. 288).

10.19.3.26 void gazebo::physics::Collision::ProcessMsg (const msgs::Collision & *_msg*)

Update parameters from a message.

Parameters

in	*_msg*	Message to update from.

10.19.3.27 virtual void gazebo::physics::Collision::SetCategoryBits (unsigned int *_bits*) [pure virtual]

Set the category bits, used during collision detection.

Parameters

in	*_bits*	The bits to set.

10.19.3.28 virtual void gazebo::physics::Collision::SetCollideBits (unsigned int *_bits*) [pure virtual]

Set the collide bits, used during collision detection.

Parameters

in	*_bits*	The bits to set.

10.19.3.29 void gazebo::physics::Collision::SetCollision (bool *_placeable*)

Set the encapsulated collsion object.

Parameters

in	*_placeable*	True to make the object movable.

10.19.3.30 void gazebo::physics::Collision::SetContactsEnabled (bool *_enable*)

Turn contact recording on or off.

Parameters

in	*_enable*	True to enable collision contacts.

10.19.3.31 void gazebo::physics::Collision::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	*_retro*	The laser retro value.

10.19.3.32 void gazebo::physics::Collision::SetShape (ShapePtr _shape)

Set the shape for this collision.

Parameters

in	*_shape*	The shape for this collision object.

10.19.3.33 void gazebo::physics::Collision::SetState (const CollisionState & _state)

Set the current collision state.

Parameters

in	*The*	collision state.

10.19.3.34 virtual void gazebo::physics::Collision::UpdateParameters (sdf::ElementPtr _sdf) [virtual]

Update the parameters using new sdf values.

Parameters

in	*_sdf*	SDF values to update from.

Reimplemented from **gazebo::physics::Entity** (p. 291).

10.19.4 Member Data Documentation

10.19.4.1 LinkPtr gazebo::physics::Collision::link [protected]

The link this collision belongs to.

10.19.4.2 bool gazebo::physics::Collision::placeable [protected]

Flag for placeable.

10.19.4.3 ShapePtr gazebo::physics::Collision::shape [protected]

Pointer to **physics::Shape** (p. 720).

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

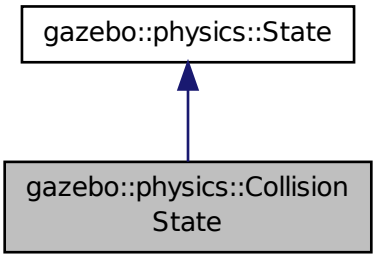
- **Collision.hh**

10.20 gazebo::physics::CollisionState Class Reference

Store state information of a **physics::Collision** (p. 195) object.

`#include <physics/phyiscs.hh>`

Inheritance diagram for gazebo::physics::CollisionState:



Public Member Functions

- **CollisionState** ()

  Default constructor.
- **CollisionState** (const **CollisionPtr** _collision)

  Constructor.
- **CollisionState** (const **sdf::ElementPtr** _sdf)

  Constructor.
- virtual ~**CollisionState** ()

  Destructor.
- void **FillSDF** (**sdf::ElementPtr** _sdf)

  Populate a state SDF element with data from the object.
- const **math::Pose** & **GetPose** () const

  *Get the **Collision** (p. 195) pose.*
- bool **IsZero** () const

  Return true if the values in the state are zero.
- virtual void **Load** (const **sdf::ElementPtr** _elem)

  Load state from SDF element.
- **CollisionState operator+** (const **CollisionState** &_state) const

  Addition operator.
- **CollisionState operator-** (const **CollisionState** &_state) const

  Subtraction operator.
- **CollisionState** & **operator=** (const **CollisionState** &_state)

  Assignment operator.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::CollisionState** &_state)
  Stream insertion operator.

Additional Inherited Members

10.20.1 Detailed Description

Store state information of a **physics::Collision** (p. 195) object.

This class captures the entire state of a **Collision** (p. 195) at one specific time during a simulation run.

State (p. 758) of a **Collision** (p. 195) is its Pose.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 gazebo::physics::CollisionState::CollisionState ( )

Default constructor.

10.20.2.2 gazebo::physics::CollisionState::CollisionState (const **CollisionPtr** *_collision*) [explicit]

Constructor.

Build a **CollisionState** (p. 205) from an existing **Collision** (p. 195).

Parameters

in	*_model*	Pointer to the **Link** (p. 418) from which to gather state info.

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

Constructor.

Build a **CollisionState** (p. 205) from SDF data

Parameters

in	*_sdf*	SDF data to load a collision state from.

10.20.2.4 virtual gazebo::physics::CollisionState::~CollisionState ( ) [virtual]

Destructor.

10.20.3 Member Function Documentation

10.20.3.1 void gazebo::physics::CollisionState::FillSDF (**sdf::ElementPtr** *_sdf*)

Populate a state SDF element with data from the object.

Parameters

out	*_sdf*	SDF element to populate.

10.20.3.2 const math::Pose& gazebo::physics::CollisionState::GetPose ( ) const

Get the **Collision** (p. 195) pose.

Returns

The pose of the **CollisionState** (p. 205)

10.20.3.3 bool gazebo::physics::CollisionState::IsZero ( ) const

Return true if the values in the state are zero.

Returns

True if the values in the state are zero.

10.20.3.4 virtual void gazebo::physics::CollisionState::Load (const sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Load **CollisionState** (p. 205) information from stored data in and SDF::Element

Parameters

in	*_elem*	Pointer to the SDF::Element containing state info.

Reimplemented from **gazebo::physics::State** (p. 761).

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

Addition operator.

Parameters

in	*_pt*	**A** (p. 111) state to add.

Returns

The resulting state.

10.20.3.6 CollisionState gazebo::physics::CollisionState::operator- (const CollisionState & *_state*) const

Subtraction operator.

Parameters

in	*_pt*	**A** (p. 111) state to substract.

Returns

The resulting state.

10.20.3.7 CollisionState& gazebo::physics::CollisionState::operator= (const CollisionState & *_state*)

Assignment operator.

Parameters

in	*_state*	**State** (p. 758) value

Returns

Reference to this

10.20.4 Friends And Related Function Documentation

10.20.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::physics::CollisionState & *_state*) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_state*	**Collision** (p. 195) state to output

Returns

the stream

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

- **CollisionState.hh**

10.21 gazebo::common::Color Class Reference

Defines a color.

`#include <common/common.hh>`

Public Types

- typedef unsigned int **ABGR**
- typedef unsigned int **ARGB**
- typedef unsigned int **BGRA**
- typedef unsigned int **RGBA**

Public Member Functions

- **Color** ()

  Constructor.
- **Color** (float _r, float _g, float _b, float _a=1.0)

  Constructor.
- **Color** (const **Color** &_clr)

  Copy Constructor.
- virtual ∼**Color** ()

  Destructor.
- **ABGR GetAsABGR** () const

  Get as uint32 ABGR packed value.
- **ARGB GetAsARGB** () const

  Get as uint32 ARGB packed value.
- **BGRA GetAsBGRA** () const

  Get as uint32 BGRA packed value.
- **math::Vector3 GetAsHSV** () const

  Get the color in HSV colorspace.
- **RGBA GetAsRGBA** () const

  Get as uint32 RGBA packed value.
- **math::Vector3 GetAsYUV** () const

  Get the color in YUV colorspace.
- bool **operator!=** (const **Color** &_pt) const

  Inequality operator.
- const **Color operator**∗ (const **Color** &_pt) const

  Multiplication operator.
- const **Color operator**∗ (const float &_v) const

  Multiply all color components by _v.
- const **Color** & **operator**∗**=** (const **Color** &_pt)

  Multiplication equal operator.
- **Color operator+** (const **Color** &_pt) const

  Addition operator (this + _pt)
- **Color operator+** (const float &_v) const

  Add _v to all color components.
- const **Color** & **operator+=** (const **Color** &_pt)

  Addition equal operator.
- **Color operator-** (const **Color** &_pt) const

  Subtraction operator.
- **Color operator-** (const float &_v) const

  Subtract _v from all color components.
- const **Color** & **operator-=** (const **Color** &_pt)

  Subtraction equal operator.
- const **Color operator**/ (const **Color** &_pt) const

  Division operator.
- const **Color operator**/ (const float &_v) const

  Divide all color component by _v.
- const **Color** & **operator**/**=** (const **Color** &_pt)

Division equal operator.

- **Color** & **operator=** (const **Color** &_pt)

  Equal operator.
- bool **operator==** (const **Color** &_pt) const

  Equality operator.
- float **operator[]** (unsigned int _index)

  Array index operator.
- void **Reset** ()

  Reset the color to default values.
- void **Set** (float _r=1, float _g=1, float _b=1, float _a=1)

  Set the contents of the vector.
- void **SetFromABGR** (const **ABGR** _v)

  Set from uint32 ABGR packed value.
- void **SetFromARGB** (const **ARGB** _v)

  Set from uint32 ARGB packed value.
- void **SetFromBGRA** (const **BGRA** _v)

  Set from uint32 BGRA packed value.
- void **SetFromHSV** (float _h, float _s, float _v)

  Set a color based on HSV values.
- void **SetFromRGBA** (const **RGBA** _v)

  Set from uint32 RGBA packed value.
- void **SetFromYUV** (float _y, float _u, float _v)

  Set from yuv.

Public Attributes

- float **a**
- float **b**
- float **g**
- float **r**

Static Public Attributes

- static const **Color Black**

  (0, 0, 0)
- static const **Color Blue**

  (0, 0, 1)
- static const **Color Green**

  (0, 1, 0)
- static const **Color Purple**

  (1, 0, 1)
- static const **Color Red**

  (1, 0, 0)
- static const **Color White**

  (1, 1, 1)
- static const **Color Yellow**

  (1, 1, 0)

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **Color** &_pt)

  Stream insertion operator.
- std::istream & **operator**>> (std::istream &_in, **Color** &_pt)

  Stream insertion operator.

10.21.1 Detailed Description

Defines a color.

10.21.2 Member Typedef Documentation

10.21.2.1 typedef unsigned int gazebo::common::Color::ABGR

10.21.2.2 typedef unsigned int gazebo::common::Color::ARGB

10.21.2.3 typedef unsigned int gazebo::common::Color::BGRA

10.21.2.4 typedef unsigned int gazebo::common::Color::RGBA

10.21.3 Constructor & Destructor Documentation

10.21.3.1 gazebo::common::Color::Color ( )

Constructor.

10.21.3.2 gazebo::common::Color::Color (float *_r*, float *_g*, float *_b*, float *_a* = `1.0`)

Constructor.

Parameters

in	*_r*	Red value (range 0 to 1)
in	*_g*	Green value (range 0 to 1
in	*_b*	Blue value (range 0 to 1
in	*_a*	Alpha value (0=transparent, 1=opaque)

10.21.3.3 gazebo::common::Color::Color (const **Color** & *_clr*)

Copy Constructor.

Parameters

in	*_clr*	**Color** (p. 208) to copy

10.21.3.4 virtual gazebo::common::Color::~Color ( ) [virtual]

Destructor.

10.21.4 Member Function Documentation

10.21.4.1 ABGR gazebo::common::Color::GetAsABGR ( ) const

Get as uint32 ABGR packed value.

Returns

the color

10.21.4.2 ARGB gazebo::common::Color::GetAsARGB ( ) const

Get as uint32 ARGB packed value.

Returns

the color

10.21.4.3 BGRA gazebo::common::Color::GetAsBGRA ( ) const

Get as uint32 BGRA packed value.

Returns

the color

10.21.4.4 math::Vector3 gazebo::common::Color::GetAsHSV ( ) const

Get the color in HSV colorspace.

Returns

HSV values in a **math::Vector3** (p. 855) format

10.21.4.5 RGBA gazebo::common::Color::GetAsRGBA ( ) const

Get as uint32 RGBA packed value.

Returns

the color

10.21.4.6 math::Vector3 gazebo::common::Color::GetAsYUV ( ) const

Get the color in YUV colorspace.

Returns

the YUV color

10.21.4.7 bool gazebo::common::Color::operator!= (const Color & _pt) const

Inequality operator.

Parameters

in	*_pt*	The color to check for inequality

Returns

True if the this color does not equal _pt

10.21.4.8 const Color gazebo::common::Color::operator* (const Color & _pt) const

Multiplication operator.

Parameters

in	*_pt*	The color to muliply by

Returns

The resulting color

10.21.4.9 const Color gazebo::common::Color::operator* (const float & _v) const

Multiply all color components by _v.

Parameters

in	*_v*	The value to multiply by

Returns

The resulting color

10.21.4.10 const Color& gazebo::common::Color::operator*= (const Color & _pt)

Multiplication equal operator.

Parameters

in	*_pt*	The color to muliply by

Returns

The resulting color

10.21.4.11 Color gazebo::common::Color::operator+ (const Color & _pt) const

Addition operator (this + _pt)

Parameters

in	*_pt*	**Color** (p. 208) to add

Returns

The resulting color

10.21.4.12 Color gazebo::common::Color::operator+ (const float & _v) const

Add _v to all color components.

Parameters

in	*_v*	Value to add to each color component

Returns

The resulting color

10.21.4.13 const Color& gazebo::common::Color::operator+= (const Color & _pt)

Addition equal operator.

Parameters

in	*_pt*	**Color** (p. 208) to add

Returns

The resulting color

10.21.4.14 Color gazebo::common::Color::operator- (const Color & _pt) const

Subtraction operator.

Parameters

in	*_pt*	The color to substract

Returns

The resulting color

10.21.4.15 Color gazebo::common::Color::operator- (const float & _v) const

Subtract _v from all color components.

Parameters

in	*_v*	Value to subtract

Returns

The resulting color

10.21.4.16 const Color& gazebo::common::Color::operator-= (const Color & _pt)

Subtraction equal operator.

Parameters

in	*_pt*	**Color** (p. 208) to subtract

Returns

The resulting color

10.21.4.17 const Color gazebo::common::Color::operator/ (const Color & _pt) const

Division operator.

Parameters

in	*_pt*	**Color** (p. 208) to divide by

Returns

The resulting color

10.21.4.18 const Color gazebo::common::Color::operator/ (const float & _v) const

Divide all color component by _v.

Parameters

in	*_v*	The value to divide by

Returns

The resulting color

10.21.4.19 const Color& gazebo::common::Color::operator/= (const Color & *_pt*)

Division equal operator.

Parameters

in	*_pt*	**Color** (p. 208) to divide by

Returns

The resulting color

10.21.4.20 Color& gazebo::common::Color::operator= (const Color & *_pt*)

Equal operator.

Parameters

in	*_pt*	**Color** (p. 208) to copy

Returns

Reference to this color

10.21.4.21 bool gazebo::common::Color::operator== (const Color & *_pt*) const

Equality operator.

Parameters

in	*_pt*	The color to check for equality

Returns

True if the this color equals _pt

10.21.4.22 float gazebo::common::Color::operator[] (unsigned int *_index*)

Array index operator.

Parameters

in	*_index*	**Color** (p. 208) component index(0=red, 1=green, 2=blue)

Returns

r, g, b, or a when _index is 0, 1, 2 or 3

10.21.4.23 void gazebo::common::Color::Reset ( )

Reset the color to default values.

10.21.4.24 void gazebo::common::Color::Set (float *_r* = 1, float *_g* = 1, float *_b* = 1, float *_a* = 1)

Set the contents of the vector.

Parameters

in	*_r*	Red value (range 0 to 1)
in	*_g*	Green value (range 0 to 1)
in	*_b*	Blue value (range 0 to 1)
in	*_a*	Alpha value (0=transparent, 1=opaque)

10.21.4.25 void gazebo::common::Color::SetFromABGR (const **ABGR** *_v*)

Set from uint32 ABGR packed value.

Parameters

in	*_v*	the new color

10.21.4.26 void gazebo::common::Color::SetFromARGB (const **ARGB** *_v*)

Set from uint32 ARGB packed value.

Parameters

in	*_v*	the new color

10.21.4.27 void gazebo::common::Color::SetFromBGRA (const **BGRA** *_v*)

Set from uint32 BGRA packed value.

Parameters

in	*_v*	the new color

10.21.4.28 void gazebo::common::Color::SetFromHSV (float _h, float _s, float _v)

Set a color based on HSV values.

Parameters

in	_h	Hue(0..360)
in	_s	Saturation(0..1)
in	_v	Value(0..1)

10.21.4.29 void gazebo::common::Color::SetFromRGBA (const RGBA _v)

Set from uint32 RGBA packed value.

Parameters

in	_v	the new color

10.21.4.30 void gazebo::common::Color::SetFromYUV (float _y, float _u, float _v)

Set from yuv.

Parameters

in	_y	value
in	_u	value
in	_v	value

10.21.5 Friends And Related Function Documentation

10.21.5.1 std::ostream& operator<< (std::ostream & _out, const Color & _pt) [friend]

Stream insertion operator.

Parameters

in	_out	the output stream
in	_pt	the color

Returns

the output stream

10.21.5.2 std::istream& operator>> (std::istream & _in, Color & _pt) [friend]

Stream insertion operator.

Parameters

in	_in	the input stream
in	_pt	

10.21.6 Member Data Documentation

10.21.6.1 float gazebo::common::Color::a

10.21.6.2 float gazebo::common::Color::b

10.21.6.3 const Color gazebo::common::Color::Black [static]

(0, 0, 0)

10.21.6.4 const Color gazebo::common::Color::Blue [static]

(0, 0, 1)

10.21.6.5 float gazebo::common::Color::g

10.21.6.6 const Color gazebo::common::Color::Green [static]

(0, 1, 0)

10.21.6.7 const Color gazebo::common::Color::Purple [static]

(1, 0, 1)

10.21.6.8 float gazebo::common::Color::r

10.21.6.9 const Color gazebo::common::Color::Red [static]

(1, 0, 0)

10.21.6.10 const Color gazebo::common::Color::White [static]

(1, 1, 1)

10.21.6.11 const Color gazebo::common::Color::Yellow [static]

(1, 1, 0)

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

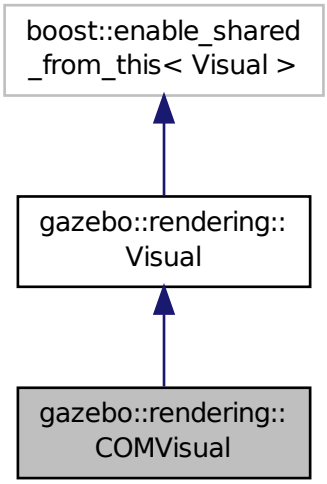
- **Color.hh**

10.22 gazebo::rendering::COMVisual Class Reference

Basic Center of Mass visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::COMVisual:



Public Member Functions

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

  Constructor.
- virtual ~**COMVisual** ()

  Destructor.
- virtual void **Load** (**sdf::ElementPtr** _elem)

  *Load the **Visual** (p. 885) from an SDF pointer.*
- virtual void **Load** (ConstLinkPtr &_msg)

  Load from a message.

Additional Inherited Members

10.22.1 Detailed Description

Basic Center of Mass visualization.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 gazebo::rendering::COMVisual::COMVisual (const std::string & *_name*, VisualPtr *_vis*)

Constructor.

Parameters

in	*_name*	Name of the **Visual** (p. 885)
in	*_vis*	Parent **Visual** (p. 885)

10.22.2.2 virtual gazebo::rendering::COMVisual::~COMVisual ( ) [virtual]

Destructor.

10.22.3 Member Function Documentation

10.22.3.1 virtual void gazebo::rendering::COMVisual::Load (sdf::ElementPtr *_elem*) [virtual]

Load the **Visual** (p. 885) from an SDF pointer.

Parameters

in	*_elem*	SDF Element pointer

10.22.3.2 virtual void gazebo::rendering::COMVisual::Load (ConstLinkPtr & *_msg*) [virtual]

Load from a message.

Parameters

in	*_msg*	Pointer to the message

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

- **COMVisual.hh**

10.23 gazebo::event::Connection Class Reference

A (p. 111) class that encapsulates a connection.

`#include <Event.hh>`

Public Member Functions

- **Connection** ()

  Constructor.
- **Connection** (**Event** *_e, int _i)

  Constructor.
- ~**Connection** ()

  Destructor.
- int **GetId** () const

  Get the id of this connection.

10.23.1 Detailed Description

A (p. 111) class that encapsulates a connection.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 gazebo::event::Connection::Connection ( ) [inline]

Constructor.

10.23.2.2 gazebo::event::Connection::Connection (Event * _e, int _i)

Constructor.

Parameters

in	*_e*	**Event** (p. 292) pointer to connect with
in	*_i*	Unique id

10.23.2.3 gazebo::event::Connection::~Connection ( )

Destructor.

10.23.3 Member Function Documentation

10.23.3.1 int gazebo::event::Connection::GetId ( ) const

Get the id of this connection.

Returns

The id of this connection

Referenced by gazebo::event::EventT< T >::Disconnect().

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

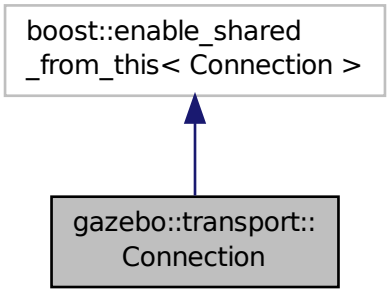
- **Event.hh**

10.24 gazebo::transport::Connection Class Reference

Single TCP/IP connection manager.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::Connection:



Public Types

- typedef boost::function< void(const **ConnectionPtr** &)> **AcceptCallback**

  The signature of a connection accept callback.
- typedef boost::function< void(const std::string &_data)> **ReadCallback**

  The signature of a connection read callback.

Public Member Functions

- **Connection** ()

  Constructor.
- virtual ~**Connection** ()

  Destructor.
- template<typename Handler >
  void **AsyncRead** (Handler _handler)

  Peform an asyncronous read param[in] _handler Callback to invoke on received data.
- void **Cancel** ()

  Cancel all async operations on an open socket.
- bool **Connect** (const std::string &_host, unsigned int _port)

  Connect to a remote host.
- **event::ConnectionPtr ConnectToShutdown** (boost::function< void()> _subscriber)

  Register a function to be called when the connection is shut down.
- void **DisconnectShutdown** (**event::ConnectionPtr** _subscriber)

  Unregister a function to be called when the connection is shut down.
- void **EnqueueMsg** (const std::string &_buffer, bool _force=false)

  Write data to the socket.
- unsigned int **GetId** () const

  Get the ID of the connection.

- std::string **GetLocalAddress** () const

  Get the local address of this connection.
- unsigned int **GetLocalPort** () const

  Get the port of this connection.
- std::string **GetLocalURI** () const

  Get the local URI.
- std::string **GetRemoteAddress** () const

  Get the remote address.
- std::string **GetRemoteHostname** () const

  Get the remote hostname.
- unsigned int **GetRemotePort** () const

  Get the remote port number.
- std::string **GetRemoteURI** () const

  Get the remote URI.
- bool **IsOpen** () const

  Is the connection open?
- void **Listen** (unsigned int _port, const **AcceptCallback** &_acceptCB)

  Start a server that listens on a port.
- void **ProcessWriteQueue** (bool _blocking=false)

  Handle on-write callbacks.
- bool **Read** (std::string &_data)

  Read data from the socket.
- void **Shutdown** ()

  Shutdown the socket.
- void **StartRead** (const **ReadCallback** &_cb)

  Start a thread that reads from the connection and passes new message to the ReadCallback.
- void **StopRead** ()

  Stop the read loop.

Static Public Member Functions

- static std::string **GetLocalHostname** ()

  Get the local hostname.
- static bool **ValidateIP** (const std::string &_ip)

  Return true if the _ip is a valid.

10.24.1 Detailed Description

Single TCP/IP connection manager.

10.24.2 Member Typedef Documentation

10.24.2.1 typedef boost::function<void(const ConnectionPtr&)> gazebo::transport::Connection::AcceptCallback

The signature of a connection accept callback.

10.24.2.2 typedef boost::function<void(const std::string &_data)> gazebo::transport::Connection::ReadCallback

The signature of a connection read callback.

10.24.3 Constructor & Destructor Documentation

10.24.3.1 gazebo::transport::Connection::Connection ( )

Constructor.

10.24.3.2 virtual gazebo::transport::Connection::~Connection ( ) [virtual]

Destructor.

10.24.4 Member Function Documentation

10.24.4.1 template<typename Handler > void gazebo::transport::Connection::AsyncRead (Handler *_handler*) [inline]

Peform an asyncronous read param[in] _handler Callback to invoke on received data.

References gzerr, HEADER_LENGTH, and IsOpen().

10.24.4.2 void gazebo::transport::Connection::Cancel ( )

Cancel all async operations on an open socket.

10.24.4.3 bool gazebo::transport::Connection::Connect (const std::string & *_host*, unsigned int *_port*)

Connect to a remote host.

Parameters

in	*_host*	The host to connect to
in	*_port*	The port to connect to

Returns

true if connection succeeded, false otherwise

10.24.4.4 event::ConnectionPtr gazebo::transport::Connection::ConnectToShutdown (boost::function< void()> *_subscriber*) [inline]

Register a function to be called when the connection is shut down.

Parameters

in	*_subscriber*	Function to be called

Returns

Handle that can be used to unregister the function

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

10.24.4.5 void gazebo::transport::Connection::DisconnectShutdown (event::ConnectionPtr *_subscriber*) [inline]

Unregister a function to be called when the connection is shut down.

Parameters

in	*_subscriber*	Handle previously returned by **ConnectToShutdown()** (p. 225)

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

10.24.4.6 void gazebo::transport::Connection::EnqueueMsg (const std::string & *_buffer*, bool *_force* = false)

Write data to the socket.

Parameters

in	*_buffer*	Data to write
in	*_force*	If true, block until the data has been written to the socket, otherwise just enqueue the data for asynchronous write

10.24.4.7 unsigned int gazebo::transport::Connection::GetId ( ) const

Get the ID of the connection.

Returns

The connection's unique ID.

10.24.4.8 std::string gazebo::transport::Connection::GetLocalAddress ( ) const

Get the local address of this connection.

Returns

The local address

10.24.4.9 static std::string gazebo::transport::Connection::GetLocalHostname ( ) [static]

Get the local hostname.

Returns

The local hostname

10.24.4.10 unsigned int gazebo::transport::Connection::GetLocalPort ( ) const

Get the port of this connection.

Returns

The local port

10.24.4.11 std::string gazebo::transport::Connection::GetLocalURI ( ) const

Get the local URI.

Returns

The local URI

10.24.4.12 std::string gazebo::transport::Connection::GetRemoteAddress ( ) const

Get the remote address.

Returns

The remote address

10.24.4.13 std::string gazebo::transport::Connection::GetRemoteHostname ( ) const

Get the remote hostname.

Returns

The remote hostname

10.24.4.14 unsigned int gazebo::transport::Connection::GetRemotePort ( ) const

Get the remote port number.

Returns

The remote port

10.24.4.15 std::string gazebo::transport::Connection::GetRemoteURI ( ) const

Get the remote URI.

Returns

The remote URI

10.24.4.16 bool gazebo::transport::Connection::IsOpen ( ) const

Is the connection open?

Returns

true if the connection is open; false otherwise

Referenced by AsyncRead().

10.24.4.17 void gazebo::transport::Connection::Listen (unsigned int *_port*, const **AcceptCallback** & *_acceptCB*)

Start a server that listens on a port.

Parameters

in	*_port*	The port to listen on
in	*_acceptCB*	The callback to invoke when a new connection has been accepted

10.24.4.18 void gazebo::transport::Connection::ProcessWriteQueue (bool *_blocking* = `false`)

Handle on-write callbacks.

10.24.4.19 bool gazebo::transport::Connection::Read (std::string & *_data*)

Read data from the socket.

Parameters

out	*_data*	Destination for data that is read

Returns

true if data was successfully read, false otherwise

10.24.4.20 void gazebo::transport::Connection::Shutdown ( )

Shutdown the socket.

10.24.4.21 void gazebo::transport::Connection::StartRead (const **ReadCallback** & *_cb*)

Start a thread that reads from the connection and passes new message to the ReadCallback.

Parameters

in	*_cb*	The callback to invoke when a new message is received

10.24.4.22 void gazebo::transport::Connection::StopRead ( )

Stop the read loop.

10.24.4.23 static bool gazebo::transport::Connection::ValidateIP (const std::string & *_ip*) [static]

Return true if the _ip is a valid.

Parameters

in	*_ip*	Dotted quad to validate.

Returns

True if the _ip is a valid.

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

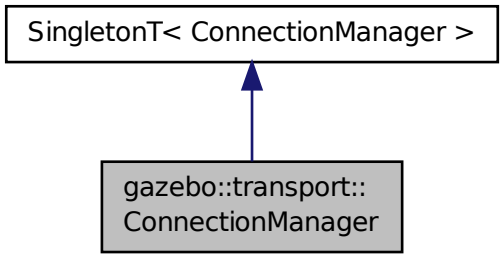
- **Connection.hh**

10.25 gazebo::transport::ConnectionManager Class Reference

Manager of connections.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::ConnectionManager:



Public Member Functions

- void **Advertise** (const std::string &_topic, const std::string &_msgType)

  Advertise a topic.
- **ConnectionPtr ConnectToRemoteHost** (const std::string &_host, unsigned int _port)

  Connect to a remote server.
- void **Fini** ()

Finalize the connection manager.

- void **GetAllPublishers** (std::list< msgs::Publish > &_publishers)

  Explicitly update the publisher list.
- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

  Get all the topic namespaces.
- bool **Init** (const std::string &_masterHost, unsigned int _masterPort)

  Initialize the connection manager.
- bool **IsRunning** () const

  Is the manager running?
- void **RegisterTopicNamespace** (const std::string &_name)

  Register a new topic namespace.
- void **RemoveConnection** (**ConnectionPtr** &_conn)

  Remove a connection from the manager.
- void **Run** ()

  Run the connection manager loop.
- void **RunUpdate** ()

  Run the manager update loop once.
- void **Stop** ()

  Stop the conneciton manager.
- void **Subscribe** (const std::string &_topic, const std::string &_msgType, bool _latching)

  Subscribe to a topic.
- void **Unadvertise** (const std::string &_topic)

  Unadvertise a topic.
- void **Unsubscribe** (const msgs::Subscribe &_sub)

  Unsubscribe from a topic.
- void **Unsubscribe** (const std::string &_topic, const std::string &_msgType)

  Unsubscribe from a topic.

Protected Attributes

- std::vector< **event::ConnectionPtr** > **eventConnections**

Additional Inherited Members

10.25.1 Detailed Description

Manager of connections.

10.25.2 Member Function Documentation

10.25.2.1 void gazebo::transport::ConnectionManager::Advertise (const std::string & *_topic*, const std::string & *_msgType*)

Advertise a topic.

Parameters

in	*_topic*	The topic to advertise
in	*_msgType*	The type of the topic

10.25.2.2 ConnectionPtr gazebo::transport::ConnectionManager::ConnectToRemoteHost (const std::string & *_host,* unsigned int *_port*)

Connect to a remote server.

Parameters

in	*_host*	Host to connect to
in	*_port*	Port to connect to

Returns

Pointer to the connection; can be null (if connection failed)

10.25.2.3 void gazebo::transport::ConnectionManager::Fini ( )

Finalize the connection manager.

10.25.2.4 void gazebo::transport::ConnectionManager::GetAllPublishers (std::list< msgs::Publish > & *_publishers*)

Explicitly update the publisher list.

Parameters

out	*_publishers*	The updated list of publishers is written here

10.25.2.5 void gazebo::transport::ConnectionManager::GetTopicNamespaces (std::list< std::string > & *_namespaces*)

Get all the topic namespaces.

Parameters

out	*_namespaces*	The list of namespace is written here

10.25.2.6 bool gazebo::transport::ConnectionManager::Init (const std::string & *_masterHost,* unsigned int *_masterPort*)

Initialize the connection manager.

Parameters

in	*_masterHost*	Host where the master is running
in	*_masterPort*	Port where the master is running

Returns

true if initialization succeeded, false otherwise

10.25.2.7 bool gazebo::transport::ConnectionManager::IsRunning ( ) const

Is the manager running?

Returns

true if running, false otherwise

10.25.2.8 void gazebo::transport::ConnectionManager::RegisterTopicNamespace (const std::string & *_name*)

Register a new topic namespace.

Parameters

in	*_name*	The name of the topic namespace to be registered

10.25.2.9 void gazebo::transport::ConnectionManager::RemoveConnection (ConnectionPtr & *_conn*)

Remove a connection from the manager.

Parameters

in	*_conn*	The connection to be removed

10.25.2.10 void gazebo::transport::ConnectionManager::Run ( )

Run the connection manager loop.

Does not return until stopped.

10.25.2.11 void gazebo::transport::ConnectionManager::RunUpdate ( )

Run the manager update loop once.

10.25.2.12 void gazebo::transport::ConnectionManager::Stop ( )

Stop the conneciton manager.

10.25.2.13 void gazebo::transport::ConnectionManager::Subscribe (const std::string & *_topic,* const std::string & *_msgType,* bool *_latching*)

Subscribe to a topic.

Parameters

in	*_topic*	The topic to subscribe to
in	*_msgType*	The type of the topic
in	*_latching*	If true, latch the latest incoming message; otherwise don't

10.25.2.14 void gazebo::transport::ConnectionManager::Unadvertise (const std::string & _topic)

Unadvertise a topic.

Parameters

in	*_topic*	The topic to unadvertise

10.25.2.15 void gazebo::transport::ConnectionManager::Unsubscribe (const msgs::Subscribe & _sub)

Unsubscribe from a topic.

Parameters

in	*_sub*	**A** (p. 111) subscription object

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

Unsubscribe from a topic.

Parameters

in	*_topic*	The topic to unsubscribe from
in	*_msgType*	The type of the topic

10.25.3 Member Data Documentation

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

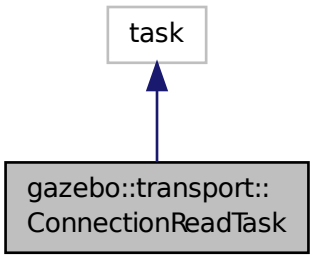
The documentation for this class was generated from the following file:

- **ConnectionManager.hh**

10.26 gazebo::transport::ConnectionReadTask Class Reference

`#include <Connection.hh>`

Inheritance diagram for gazebo::transport::ConnectionReadTask:



Public Member Functions

- **ConnectionReadTask** (boost::function< void(const std::string &)> _func, const std::string &_data)

  Constructor.
- tbb::task * **execute** ()

  Overridden function from tbb::task that exectues the data callback.

10.26.1 Detailed Description

A (p. 111) task instance that is created when data is read from a socket and used by TBB

10.26.2 Constructor & Destructor Documentation

10.26.2.1 gazebo::transport::ConnectionReadTask::ConnectionReadTask (boost::function< void(const std::string &)> *_func*, const std::string & *_data*) [inline]

Constructor.

Parameters

	_in]	_func Boost function pointer, which is the function that receives the data.
in	*_data*	Data to send to the boost function pointer.

10.26.3 Member Function Documentation

10.26.3.1 tbb::task* gazebo::transport::ConnectionReadTask::execute ( ) [inline]

Overridden function from tbb::task that exectues the data callback.

References NULL.

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

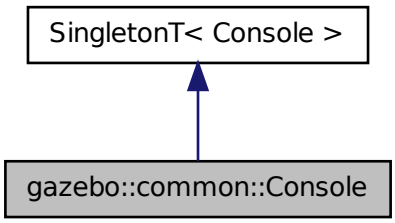
- **Connection.hh**

10.27 gazebo::common::Console Class Reference

Message, error, warning functionality.

`#include <common/commom.hh>`

Inheritance diagram for gazebo::common::Console:



Public Member Functions

- std::ostream & **ColorErr** (const std::string &_lbl, const std::string &_file, unsigned int _line, int _color)

  Use this to output an error to the terminal.
- std::ostream & **ColorMsg** (const std::string &_lbl, int _color)

  Use this to output a colored message to the terminal.
- void **Init** (const std::string &_logFilename)

  Load the message parameters.
- bool **IsInitialized** () const

  Return true if Init has been called.
- std::ofstream & **Log** ()

  Use this to output a colored message to the terminal.
- void **SetQuiet** (bool _q)

  Set quiet output.

Additional Inherited Members

10.27.1 Detailed Description

Message, error, warning functionality.

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

- **Console.hh**

10.28 gazebo::physics::Contact Class Reference

A (p. 111) contact between two collisions.

`#include <physics/physics.hh>`

Public Member Functions

- **Contact** ()
  Constructor.
- **Contact** (const **Contact** &_contact)
  Copy constructor.
- virtual ~**Contact** ()
  Destructor.
- std::string **DebugString** () const
  Produce a debug string.
- void **FillMsg** (msgs::Contact &_msg) const
  Populate a msgs::Contact with data from this.
- **Contact** & **operator=** (const **Contact** &_contact)
  Operator =.
- **Contact** & **operator=** (const msgs::Contact &_contact)
  Operator =.
- void **Reset** ()
  Reset to default values.

Public Attributes

- std::string **collision1**
  Name of the first collision object.
- std::string **collision2**
  Name of the second collision object.
- **Collision** * **collisionPtr1**
  Pointer to the first collision object.
- **Collision** * **collisionPtr2**
  Pointer to the second collision object.
- int **count**
  Length of all the arrays.
- double **depths** [32]
  Array of contact depths.
- **math::Vector3 normals** [32]
  Array of force normals.
- **math::Vector3 positions** [32]
  Array of force positions.
- **common::Time time**
  Time at which the contact occurred.
- **WorldPtr world**
  World (p. *910) in which the contact occurred.*
- **JointWrench wrench** [32]
  Array of forces for the contact.

10.28.1 Detailed Description

A (p. 111) contact between two collisions.

Each contact can consist of a number of contact points

10.28.2 Constructor & Destructor Documentation

10.28.2.1 gazebo::physics::Contact::Contact ( )

Constructor.

10.28.2.2 gazebo::physics::Contact::Contact (const Contact & *_contact*)

Copy constructor.

Parameters

in	*_contact*	**Contact** (p. 236) to copy.

10.28.2.3 virtual gazebo::physics::Contact::~Contact ( ) [virtual]

Destructor.

10.28.3 Member Function Documentation

10.28.3.1 std::string gazebo::physics::Contact::DebugString ( ) const

Produce a debug string.

Returns

A (p. 111) string that contains the values of the contact.

10.28.3.2 void gazebo::physics::Contact::FillMsg (msgs::Contact & *_msg*) const

Populate a msgs::Contact with data from this.

Parameters

out	*_msg*	**Contact** (p. 236) message the will hold the data.

10.28.3.3 Contact& gazebo::physics::Contact::operator= (const Contact & *_contact*)

Operator =.

Parameters

in	*_contact*	**Contact** (p. 236) to copy.

Returns

Reference to this contact

10.28.3.4 Contact& gazebo::physics::Contact::operator= (const msgs::Contact & *_contact*)

Operator =.

Parameters

in	*_contact*	msgs::Contact to copy.

Returns

Reference to this contact

10.28.3.5 void gazebo::physics::Contact::Reset ( )

Reset to default values.

10.28.4 Member Data Documentation

10.28.4.1 std::string gazebo::physics::Contact::collision1

Name of the first collision object.

10.28.4.2 std::string gazebo::physics::Contact::collision2

Name of the second collision object.

10.28.4.3 Collision* gazebo::physics::Contact::collisionPtr1

Pointer to the first collision object.

10.28.4.4 Collision* gazebo::physics::Contact::collisionPtr2

Pointer to the second collision object.

10.28.4.5 int gazebo::physics::Contact::count

Length of all the arrays.

10.28.4.6 double gazebo::physics::Contact::depths[32]

Array of contact depths.

10.28.4.7 math::Vector3 gazebo::physics::Contact::normals[32]

Array of force normals.

10.28.4.8 math::Vector3 gazebo::physics::Contact::positions[32]

Array of force positions.

10.28.4.9 common::Time gazebo::physics::Contact::time

Time at which the contact occurred.

10.28.4.10 WorldPtr gazebo::physics::Contact::world

World (p. 910) in which the contact occurred.

10.28.4.11 JointWrench gazebo::physics::Contact::wrench[32]

Array of forces for the contact.

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

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

- **Contact.hh**

10.29 gazebo::physics::ContactManager Class Reference

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

`#include <physics/physics.hh>`

Public Member Functions

- **ContactManager** ()
  Constructor.
- virtual ∼**ContactManager** ()
  Destructor.
- void **Clear** ()
  Clear all stored contacts.
- **Contact** ∗ **GetContact** (unsigned int _index) const
  Get a single contact by index.
- unsigned int **GetContactCount** () const
  Return the number of valid contacts.

- const std::vector< **Contact** * > & **GetContacts** () const

  Get all the contacts.
- void **Init** (**WorldPtr** _world)

  *Initialize the **ContactManager** (p. 239).*
- **Contact** * **NewContact** (**Collision** *_collision1, **Collision** *_collision2, const **common::Time** &_time)

  Add a new contact.
- void **PublishContacts** ()

  Publish all contacts in a msgs::Contacts message.
- void **ResetCount** ()

  Set the contact count to zero.

10.29.1 Detailed Description

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

10.29.2 Constructor & Destructor Documentation

10.29.2.1 gazebo::physics::ContactManager::ContactManager ( )

Constructor.

10.29.2.2 virtual gazebo::physics::ContactManager::~ContactManager ( ) [virtual]

Destructor.

10.29.3 Member Function Documentation

10.29.3.1 void gazebo::physics::ContactManager::Clear ( )

Clear all stored contacts.

10.29.3.2 Contact* gazebo::physics::ContactManager::GetContact (unsigned int *_index*) const

Get a single contact by index.

The index must be between 0 and **ContactManager::GetContactCount** (p. 240).

Parameters

in	*_index*	Index of the **Contact** (p. 236) to return.

Returns

Pointer to a contact, NULL If index is invalid.

10.29.3.3 unsigned int gazebo::physics::ContactManager::GetContactCount ( ) const

Return the number of valid contacts.

10.29.3.4 const std::vector<Contact*>& gazebo::physics::ContactManager::GetContacts ( ) const

Get all the contacts.

The return vector may have invalid contacts. Only use contents of the vector between 0 and **ContactManager::GetContactCount** (p. 240)

Returns

Vector of contact pointers.

10.29.3.5 void gazebo::physics::ContactManager::Init (WorldPtr _world)

Initialize the **ContactManager** (p. 239).

This is required in order to publish contact messages via the **ContactManager::PublishContacts** (p. 241) method.

Parameters

in	*_world*	Pointer to the world that is initializing the contact manager.

10.29.3.6 Contact* gazebo::physics::ContactManager::NewContact (Collision * _collision1, Collision * _collision2, const common::Time & _time)

Add a new contact.

Noramlly this is only used by a Physics/Collision engine when a new contact is generated. All other users should just make use of the accessor functions.

If no one is listening, then the return value will be NULL. This is a signal to the Physics engine that it can skip the extra processing necessary to get back contact information.

Returns

The new contact. The physics engine should populate the contact's parameters. NULL will be returned if there are no subscribers to the contact topic.

10.29.3.7 void gazebo::physics::ContactManager::PublishContacts ( )

Publish all contacts in a msgs::Contacts message.

10.29.3.8 void gazebo::physics::ContactManager::ResetCount ( )

Set the contact count to zero.

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

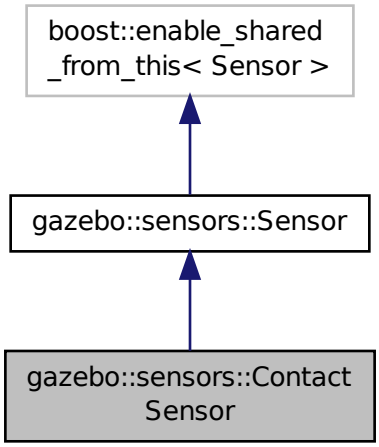
- **ContactManager.hh**

10.30 gazebo::sensors::ContactSensor Class Reference

Contact sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::ContactSensor:



Public Member Functions

- **ContactSensor** ()

  Constructor.
- virtual ~**ContactSensor** ()

  Destructor.
- unsigned int **GetCollisionContactCount** (const std::string &_collisionName) const

  Return the number of contacts for an observed collision.
- unsigned int **GetCollisionCount** () const

  Get the number of collisions that the sensor is observing.
- std::string **GetCollisionName** (unsigned int _index) const

  Get a collision name at index _index.
- msgs::Contacts **GetContacts** () const

  *Get all the contacts for the **ContactSensor** (p. 241).*
- std::map< std::string,
  physics::Contact > **GetContacts** (const std::string &_collisionName)

  Gets contacts of a collision.
- virtual void **Init** ()

  Initialize the sensor.
- virtual bool **IsActive** ()

  Returns true if sensor generation is active.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** _sdf)

  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the sensor.
- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.

Additional Inherited Members

10.30.1 Detailed Description

Contact sensor.

This sensor detects and reports contacts between objects

10.30.2 Constructor & Destructor Documentation

10.30.2.1 gazebo::sensors::ContactSensor::ContactSensor ( )

Constructor.

10.30.2.2 virtual gazebo::sensors::ContactSensor::~ContactSensor ( ) [virtual]

Destructor.

10.30.3 Member Function Documentation

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

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 702).

10.30.3.2 unsigned int gazebo::sensors::ContactSensor::GetCollisionContactCount (const std::string & *_collisionName*) const

Return the number of contacts for an observed collision.

Parameters

in	*_collisionName*	The name of the observed collision.

Returns

The collision contact count.

10.30.3.3 unsigned int gazebo::sensors::ContactSensor::GetCollisionCount ( ) const

Get the number of collisions that the sensor is observing.

Returns

Number of collisions.

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

Get a collision name at index _index.

Parameters

in	*_index*	Index of collision in collection of collisions.

Returns

name of collision.

10.30.3.5 msgs::Contacts gazebo::sensors::ContactSensor::GetContacts ( ) const

Get all the contacts for the **ContactSensor** (p. 241).

Returns

Message that contains contact information between collision pairs.

During ODEPhysics::UpdateCollisions, all collision pairs in the world are pushed into a buffer within ContactManager. Subsequently, World::Update invokes ContactManager::PublishContacts to publish all contacts generated within a timestep onto Gazebo topic ~/physics/contacts.

Each **ContactSensor** (p. 241) subscribes to the Gazebo ~/physics/contacts topic, retrieves all contact pairs in a time step and filters them wthin ContactSensor::OnContacts against <collision> body name specified by the **ContactSensor** (p. 241) SDF. All collision pairs between **ContactSensor** (p. 241) <collision> body and other bodies in the world are stored in an array inside contacts.proto.

Within each element of the contact.proto array inside contacts.proto, list of collisions between collision bodies (collision1 and collision 2) are stored in an array of elements, (position, normal, depth, wrench). **A** (p. 111) timestamp has also been added (time). Details are described below:

- string collision1 name of the first collision object.
- string collision2 name of the second collision object.
- Vector3d position position of the contact joint in inertial frame.
- Vector3d normal normal of the contact joint in inertial frame.
- double depth intersection (penetration) depth of two collision bodies.
- JointWrench wrench Forces and torques acting on both collision bodies. See joint_wrench.proto for details. The forces and torques are applied at the CG of perspective links for each collision body, specified in the inertial frame.
- Time time time at which this contact happened.

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

Gets contacts of a collision.

Parameters

in	*_collisionName*	Name of collision

Returns

Container of contacts

10.30.3.7 virtual void gazebo::sensors::ContactSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.30.3.8 virtual bool gazebo::sensors::ContactSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

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

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 698) parameters
in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.30.3.10 virtual void gazebo::sensors::ContactSensor::Load (const std::string & _worldName) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

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

Update the sensor information.

Parameters

in	*_force*	True if update is forced, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 706).

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

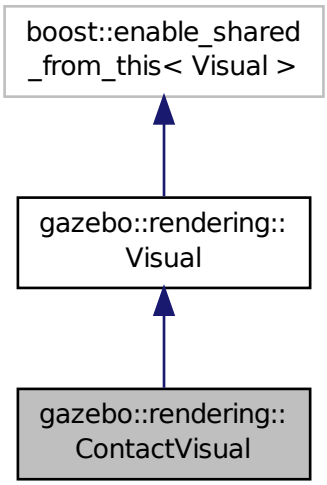
- **ContactSensor.hh**

10.31 gazebo::rendering::ContactVisual Class Reference

Contact visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::ContactVisual:



Public Member Functions

- **ContactVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)
  Constructor.
- virtual ~**ContactVisual** ()
  Destructor.
- void **SetEnabled** (bool _enabled)
  Set to true to enable contact visualization.

Additional Inherited Members

10.31.1 Detailed Description

Contact visualization.

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

10.31.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_name*	Name of the **ContactVisual** (p. 246)
in	*_vis*	Pointer the parent **Visual** (p. 885)
in	*_topicName*	Name of the topic which publishes the contact information.

10.31.2.2 virtual gazebo::rendering::ContactVisual::~ContactVisual ( ) [virtual]

Destructor.

10.31.3 Member Function Documentation

10.31.3.1 void gazebo::rendering::ContactVisual::SetEnabled (bool *_enabled*)

Set to true to enable contact visualization.

Parameters

in	*_enabled*	True to show contacts, false to hide.

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

- **ContactVisual.hh**

10.32 gazebo::rendering::Conversions Class Reference

Conversions (p. 247) **Conversions.hh** (p. 964) **rendering/Conversions.hh** (p. 964).

```
#include <Conversions.hh>
```

Static Public Member Functions

- static Ogre::ColourValue **Convert** (const **common::Color** &_clr)

  Return the equivalent ogre color.

- static **common::Color Convert** (const Ogre::ColourValue &_clr)

  Return the equivalent gazebo color.
- static Ogre::Vector3 **Convert** (const **math::Vector3** &_v)

  *return **Ogre** (p. 106) Vector from Gazebo Vector3*
- static **math::Vector3 Convert** (const Ogre::Vector3 &_v)

  return gazebo Vector from ogre Vector3
- static Ogre::Quaternion **Convert** (const **math::Quaternion** &_v)

  *Gazebo quaternion to **Ogre** (p. 106) quaternion.*
- static **math::Quaternion Convert** (const Ogre::Quaternion &_v)

  ***Ogre** (p. 106) quaternion to Gazebo quaternion.*

10.32.1 Detailed Description

Conversions (p. 247) **Conversions.hh** (p. 964) **rendering/Conversions.hh** (p. 964).

A (p. 111) set of utility function to convert between Gazebo and **Ogre** (p. 106) data types

10.32.2 Member Function Documentation

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

Return the equivalent ogre color.

Parameters

in	*_clr*	Gazebo color to convert

Returns

Ogre (p. 106) color value

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

Return the equivalent gazebo color.

Parameters

in	*_clr*	**Ogre** (p. 106) color to convert

Returns

Gazebo color value

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

return **Ogre** (p. 106) Vector from Gazebo Vector3

Parameters

in	_v	Gazebo vector

Returns

Ogre (p. 106) vector

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

return gazebo Vector from ogre Vector3

Parameters

in	_v	**Ogre** (p. 106) vector

Returns

Gazebo vector

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

Gazebo quaternion to **Ogre** (p. 106) quaternion.

Parameters

in	_v	Gazebo quaternion

Returns

Ogre (p. 106) quaternion

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

Ogre (p. 106) quaternion to Gazebo quaternion.

Parameters

in	_v	**Ogre** (p. 106) quaternion return Gazebo quaternion

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

- **Conversions.hh**

10.33 sdf::Converter Class Reference

Convert from one version of **SDF** (p. 695) to another.

`#include <Converter.hh>`

Static Public Member Functions

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

  *Convert **SDF** (p. 695) to the specified version.*
- static void **Convert** (TiXmlDocument *_doc, TiXmlDocument *_convertDoc)

  This is an internal function.

10.33.1 Detailed Description

Convert from one version of **SDF** (p. 695) to another.

10.33.2 Member Function Documentation

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

Convert **SDF** (p. 695) to the specified version.

Parameters

in	*_doc*	**SDF** (p. 695) xml doc
in	*_toVersion*	Version number in string format
in	*_quiet*	False to be more verbose

10.33.2.2 static void sdf::Converter::Convert (TiXmlDocument * *_doc*, TiXmlDocument * *_convertDoc*) [static]

This is an internal function.

Generic convert function that converts the **SDF** (p. 695) based on the given Convert file.

Parameters

in	*_doc*	**SDF** (p. 695) xml doc
in	*_convertDoc*	Convert xml doc

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

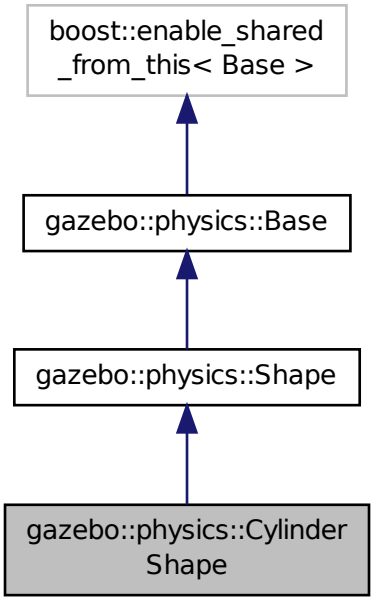
- **Converter.hh**

10.34 gazebo::physics::CylinderShape Class Reference

Cylinder collision.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::CylinderShape:



Public Member Functions

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

Additional Inherited Members

10.34.1 Detailed Description

Cylinder collision.

10.34.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_parent*	Parent of the shape.

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

Destructor.

10.34.3 Member Function Documentation

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

Fill in the values for a geomertry message.

Parameters

out	*_msg*	The geometry message to fill.

Implements **gazebo::physics::Shape** (p. 722).

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

Get length.

Returns

The cylinder length.

10.34.3.3 double gazebo::physics::CylinderShape::GetRadius ( ) const

Get radius.

Returns

The cylinder radius.

10.34.3.4 void gazebo::physics::CylinderShape::Init ( ) [virtual]

Initialize the cylinder.

Implements **gazebo::physics::Shape** (p. 722).

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

Update values based on a message.

Parameters

in	*_msg*	Message to update from.

Implements **gazebo::physics::Shape** (p. 722).

10.34.3.6 void gazebo::physics::CylinderShape::SetLength (double *_length*)

Set length.

Parameters

in	*_length*	New length of the cylinder.

10.34.3.7 void gazebo::physics::CylinderShape::SetRadius (double *_radius*)

Set radius.

Parameters

in}	_radius New radius of the cylinder.

10.34.3.8 virtual void gazebo::physics::CylinderShape::SetSize (double *_radius,* double *_length*) [virtual]

Set the size of the cylinder.

Parameters

in	*_radius*	New radius.
in	*_lenght*	New length.

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

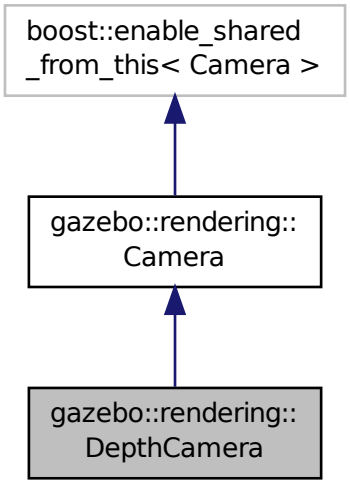
- **CylinderShape.hh**

10.35 gazebo::rendering::DepthCamera Class Reference

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

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::DepthCamera:



Public Member Functions

- **DepthCamera** (const std::string &_namePrefix, **ScenePtr** _scene, bool _autoRender=true)

  Constructor.
- virtual ~**DepthCamera** ()

  Destructor.
- template<typename T >
  event::ConnectionPtr ConnectNewDepthFrame (T _subscriber)

  Connect a to the new depth image signal.
- template<typename T >
  event::ConnectionPtr ConnectNewRGBPointCloud (T _subscriber)

  Connect a to the new rgb point cloud signal.
- void **CreateDepthTexture** (const std::string &_textureName)

  Create a texture which will hold the depth data.
- void **DisconnectNewDepthFrame** (**event::ConnectionPtr** &_c)

  Disconnect from an depth image singal.
- void **DisconnectNewRGBPointCloud** (**event::ConnectionPtr** &c)

  Disconnect from an rgb point cloud singal.
- void **Fini** ()

  Finalize the camera.
- virtual const float * **GetDepthData** ()

  All things needed to get back z buffer for depth data.
- void **Init** ()

  Initialize the camera.
- void **Load** (**sdf::ElementPtr** &_sdf)

Load the camera with a set of parmeters.

- void **Load** ()

  Load the camera with default parmeters.

- virtual void **PostRender** ()

  Render the camera.

- virtual void **SetDepthTarget** (Ogre::RenderTarget ∗_target)

  Set the render target, which renders the depth data.

Protected Attributes

- Ogre::RenderTarget ∗ **depthTarget**

  Pointer to the depth target.

- Ogre::Texture ∗ **depthTexture**

  Pointer to the depth texture.

- Ogre::Viewport ∗ **depthViewport**

  Pointer to the depth viewport.

Additional Inherited Members

10.35.1 Detailed Description

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

10.35.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_namePrefix*	Unique prefix name for the camera.
in	*_scene*	**Scene** (p. 676) that will contain the camera
in	*_autoRender*	Almost everyone should leave this as true.

10.35.2.2 virtual gazebo::rendering::DepthCamera::∼DepthCamera ( ) `[virtual]`

Destructor.

10.35.3 Member Function Documentation

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

Connect a to the new depth image signal.

Parameters

in	*_subscriber*	Subscriber callback function

Returns

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

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

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

Connect a to the new rgb point cloud signal.

Parameters

in	*_subscriber*	Subscriber callback function

Returns

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

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

10.35.3.3 void gazebo::rendering::DepthCamera::CreateDepthTexture (const std::string & *_textureName*)

Create a texture which will hold the depth data.

Parameters

in	*_textureName*	Name of the texture to create

10.35.3.4 void gazebo::rendering::DepthCamera::DisconnectNewDepthFrame (event::ConnectionPtr & *_c*) [inline]

Disconnect from an depth image singal.

Parameters

in	*_c*	The connection to disconnect

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

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

Disconnect from an rgb point cloud singal.

Parameters

in	*_c*	The connection to disconnect

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

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

Finalize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 171).

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

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

Returns

The z-buffer as a float array

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

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 178).

10.35.3.9 void gazebo::rendering::DepthCamera::Load (sdf::ElementPtr & *_sdf*)

Load the camera with a set of parmeters.

Parameters

in	*_sdf*	The SDF camera info

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

Load the camera with default parmeters.

Reimplemented from **gazebo::rendering::Camera** (p. 179).

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

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 180).

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

Set the render target, which renders the depth data.

Parameters

in	*_target*	Pointer to the render target

10.35.4 Member Data Documentation

10.35.4.1 Ogre::RenderTarget* gazebo::rendering::DepthCamera::depthTarget [protected]

Pointer to the depth target.

10.35.4.2 Ogre::Texture* gazebo::rendering::DepthCamera::depthTexture [protected]

Pointer to the depth texture.

10.35.4.3 Ogre::Viewport* gazebo::rendering::DepthCamera::depthViewport [protected]

Pointer to the depth viewport.

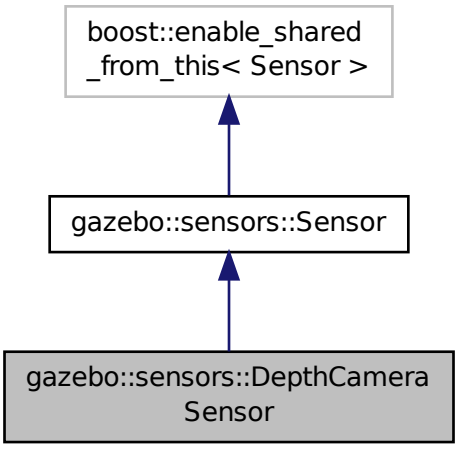
The documentation for this class was generated from the following file:

- **DepthCamera.hh**

10.36 gazebo::sensors::DepthCameraSensor Class Reference

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::DepthCameraSensor:



Public Member Functions

- **DepthCameraSensor** ()

  Constructor.

- virtual ~**DepthCameraSensor** ()

  Destructor.
- **rendering::DepthCameraPtr GetDepthCamera** () const

  *Returns a pointer to the **rendering::DepthCamera** (p. 253).*
- bool **SaveFrame** (const std::string &_filename)

  Saves an image frame of depth camera sensor to file.
- virtual void **SetActive** (bool _value)

  Set whether the sensor is active or not.
- virtual void **SetParent** (const std::string &_name)

  Set the parent of the sensor.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the camera.
- virtual void **Init** ()

  Initialize the camera.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** &_sdf)

  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.
- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.

Additional Inherited Members

10.36.1 Constructor & Destructor Documentation

10.36.1.1 gazebo::sensors::DepthCameraSensor::DepthCameraSensor ( )

Constructor.

10.36.1.2 virtual gazebo::sensors::DepthCameraSensor::~DepthCameraSensor ( ) [virtual]

Destructor.

10.36.2 Member Function Documentation

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

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 702).

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

Returns a pointer to the **rendering::DepthCamera** (p. 253).

Returns

Depth Camera pointer

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

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

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

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 698) parameters
in	*_worldName*	Name of world to load from

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

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

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

Saves an image frame of depth camera sensor to file.

Parameters

in	*Name*	of file to save as

Returns

True if saved, false if not

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

Set whether the sensor is active or not.

Parameters

in	*_value*	True if active, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

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

Set the parent of the sensor.

Parameters

in	*_name*	Name of parent

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

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

Update the sensor information.

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 706).

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

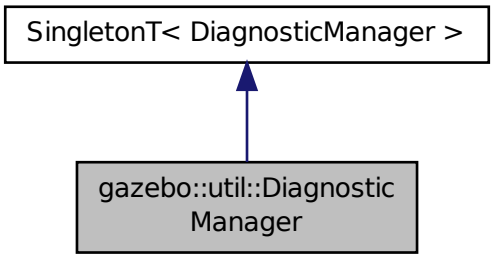
- **DepthCameraSensor.hh**

10.37 gazebo::util::DiagnosticManager Class Reference

A (p. 111) diagnostic manager class.

`#include <util/util.hh>`

Inheritance diagram for gazebo::util::DiagnosticManager:



Public Member Functions

- std::string **GetLabel** (int _index) const

  Get a label for a timer.
- boost::filesystem::path **GetLogPath** () const

  Get the path in which logs are stored.
- **common::Time GetTime** (int _index) const

  Get the time of a timer instance.
- **common::Time GetTime** (const std::string &_label) const

  Get a time based on a label.
- int **GetTimerCount** () const

  Get the number of timers.
- void **Init** (const std::string &_worldName)

  Initialize to report diagnostics about a world.
- void **Lap** (const std::string &_name, const std::string &_prefix)

  Output the current elapsed time of an active timer with a prefix string.
- void **StartTimer** (const std::string &_name)

  Start a new timer instance.
- void **StopTimer** (const std::string &_name)

  Stop a currently running timer.

Additional Inherited Members

10.37.1 Detailed Description

A (p. 111) diagnostic manager class.

10.37.2 Member Function Documentation

10.37.2.1 std::string gazebo::util::DiagnosticManager::GetLabel (int *_index*) const

Get a label for a timer.

Parameters

in	*_index*	Index of a timer instance

Returns

Label of the specified timer

10.37.2.2 boost::filesystem::path gazebo::util::DiagnosticManager::GetLogPath ( ) const

Get the path in which logs are stored.

Returns

The path in which logs are stored.

10.37.2.3 common::Time gazebo::util::DiagnosticManager::GetTime (int _index) const

Get the time of a timer instance.

Parameters

in	*_index*	The index of a timer instance

Returns

Time of the specified timer

10.37.2.4 common::Time gazebo::util::DiagnosticManager::GetTime (const std::string & _label) const

Get a time based on a label.

Parameters

in	*_label*	Name of the timer instance

Returns

Time of the specified timer

10.37.2.5 int gazebo::util::DiagnosticManager::GetTimerCount ( ) const

Get the number of timers.

Returns

The number of timers

10.37.2.6 void gazebo::util::DiagnosticManager::Init (const std::string & _worldName)

Initialize to report diagnostics about a world.

Parameters

in	*_worldName*	Name of the world.

10.37.2.7 void gazebo::util::DiagnosticManager::Lap (const std::string & _name, const std::string & _prefix)

Output the current elapsed time of an active timer with a prefix string.

This also resets the timer and keeps it running.

Parameters

in	*_name*	Name of the timer to access.
in	*_prefix*	Informational string that is output with the elapsed time.

10.37.2.8 void gazebo::util::DiagnosticManager::StartTimer (const std::string & _name)

Start a new timer instance.

Parameters

in	*_name*	Name of the timer.

Returns

A (p. 111) pointer to the new diagnostic timer

10.37.2.9 void gazebo::util::DiagnosticManager::StopTimer (const std::string & _name)

Stop a currently running timer.

Parameters

in	*_name*	Name of the timer to stop.

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

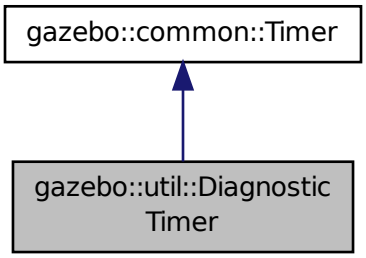
- **Diagnostics.hh**

10.38 gazebo::util::DiagnosticTimer Class Reference

A (p. 111) timer designed for diagnostics.

`#include <util/util.hh>`

Inheritance diagram for gazebo::util::DiagnosticTimer:



Public Member Functions

- **DiagnosticTimer** (const std::string &_name)

Constructor.

- virtual ~**DiagnosticTimer** ()

  Destructor.
- const std::string **GetName** () const

  Get the name of the timer.
- void **Lap** (const std::string &_prefix)

  Output a lap time.
- virtual void **Start** ()

  Start the timer.
- virtual void **Stop** ()

  Stop the timer.

10.38.1 Detailed Description

A (p. 111) timer designed for diagnostics.

10.38.2 Constructor & Destructor Documentation

10.38.2.1 gazebo::util::DiagnosticTimer::DiagnosticTimer (const std::string & *_name*)

Constructor.

Parameters

in	*_name*	Name of the timer

10.38.2.2 virtual gazebo::util::DiagnosticTimer::~DiagnosticTimer ( ) [virtual]

Destructor.

10.38.3 Member Function Documentation

10.38.3.1 const std::string gazebo::util::DiagnosticTimer::GetName ( ) const [inline]

Get the name of the timer.

Returns

The name of timer

10.38.3.2 void gazebo::util::DiagnosticTimer::Lap (const std::string & *_prefix*)

Output a lap time.

Parameters

in	*_prefix*	Annotation to output with the elapsed time.

10.38.3.3 virtual void gazebo::util::DiagnosticTimer::Start ( ) [virtual]

Start the timer.

Reimplemented from **gazebo::common::Timer** (p. 814).

10.38.3.4 virtual void gazebo::util::DiagnosticTimer::Stop ( ) [virtual]

Stop the timer.

Reimplemented from **gazebo::common::Timer** (p. 814).

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

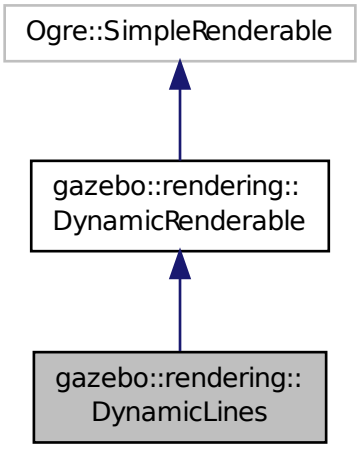
- **Diagnostics.hh**

10.39 gazebo::rendering::DynamicLines Class Reference

Class for drawing lines that can change.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::DynamicLines:



Public Member Functions

- **DynamicLines** (**RenderOpType** _opType=**RENDERING_LINE_STRIP**)

  Constructor.
- virtual ~**DynamicLines** ()

  Destructor.

- void **AddPoint** (const **math::Vector3** &_pt)

  Add a point to the point list.
- void **AddPoint** (double _x, double _y, double _z)

  Add a point to the point list.
- void **Clear** ()

  Remove all points from the point list.
- virtual const Ogre::String & **getMovableType** () const

  *Overridden function from **Ogre** (p. 106)'s base class.*
- const **math::Vector3** & **GetPoint** (unsigned int _index) const

  Return the location of an existing point in the point list.
- unsigned int **GetPointCount** () const

  Return the total number of points in the point list.
- void **SetPoint** (unsigned int _index, const **math::Vector3** &_value)

  Change the location of an existing point in the point list.
- void **Update** ()

  Call this to update the hardware buffer after making changes.

Static Public Member Functions

- static std::string **GetMovableType** ()

  Get type of movable.

Protected Member Functions

- virtual void **CreateVertexDeclaration** ()

  *Implementation **DynamicRenderable** (p. 269), creates a simple vertex-only decl.*
- virtual void **FillHardwareBuffers** ()

  *Implementation **DynamicRenderable** (p. 269), pushes point list out to hardware memory.*

Additional Inherited Members

10.39.1 Detailed Description

Class for drawing lines that can change.

10.39.2 Constructor & Destructor Documentation

10.39.2.1 gazebo::rendering::DynamicLines::DynamicLines (RenderOpType *_opType* = RENDERING_LINE_STRIP)

Constructor.

Parameters

in	*_opType*	The type of Line

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

Destructor.

10.39.3 Member Function Documentation

10.39.3.1 void gazebo::rendering::DynamicLines::AddPoint (const math::Vector3 & _pt)

Add a point to the point list.

Parameters

in	*pt*	**math::Vector3** (p. 855) point

10.39.3.2 void gazebo::rendering::DynamicLines::AddPoint (double _x, double _y, double _z)

Add a point to the point list.

Parameters

in	*_x*	X position.
in	*_y*	Y position.
in	*_z*	Z position.

10.39.3.3 void gazebo::rendering::DynamicLines::Clear ( )

Remove all points from the point list.

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

Implementation **DynamicRenderable** (p. 269), creates a simple vertex-only decl.

Implements **gazebo::rendering::DynamicRenderable** (p. 271).

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

Implementation **DynamicRenderable** (p. 269), pushes point list out to hardware memory.

Implements **gazebo::rendering::DynamicRenderable** (p. 271).

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

Get type of movable.

Returns

This returns "gazebo::dynamiclines"

10.39.3.7 virtual const Ogre::String& gazebo::rendering::DynamicLines::getMovableType ( ) const [virtual]

Overridden function from **Ogre** (p. 106)'s base class.

Returns

Returns "gazebo::ogredynamicslines"

10.39.3.8 const math::Vector3& gazebo::rendering::DynamicLines::GetPoint (unsigned int _index) const

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

Parameters

in	*_index*	Number of the point to return

Returns

math::Vector3 (p. 855) value of the point

10.39.3.9 unsigned int gazebo::rendering::DynamicLines::GetPointCount ( ) const

Return the total number of points in the point list.

Returns

Number of points

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

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

Parameters

in	*_index*	Index of the point to set
in	*_value*	**math::Vector3** (p. 855) value to set the point to

10.39.3.11 void gazebo::rendering::DynamicLines::Update ( )

Call this to update the hardware buffer after making changes.

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

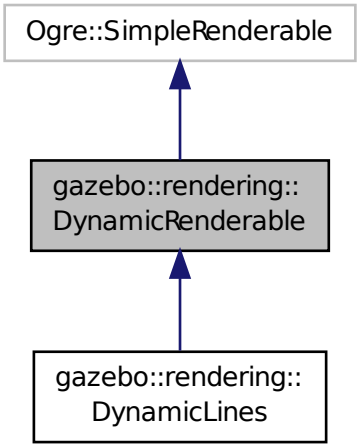
- **DynamicLines.hh**

10.40 gazebo::rendering::DynamicRenderable Class Reference

Abstract base class providing mechanisms for dynamically growing hardware buffers.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::DynamicRenderable:



Public Member Functions

- **DynamicRenderable** ()
  Constructor.
- virtual ~**DynamicRenderable** ()
  Virtual destructor.
- virtual Ogre::Real **getBoundingRadius** () const
  Implementation of Ogre::SimpleRenderable.
- **RenderOpType GetOperationType** () const
  Get the render operation type.
- virtual Ogre::Real **getSquaredViewDepth** (const Ogre::Camera *_cam) const
  Implementation of Ogre::SimpleRenderable.
- void **Init** (**RenderOpType** _opType, bool _useIndices=false)
  Initializes the dynamic renderable.
- void **SetOperationType** (**RenderOpType** _opType)
  Set the render operation type.

Protected Member Functions

- virtual void **CreateVertexDeclaration** ()=0
  Creates the vertex declaration.
- virtual void **FillHardwareBuffers** ()=0
  Fills the hardware vertex and index buffers with data.
- void **PrepareHardwareBuffers** (size_t _vertexCount, size_t _indexCount)
  Prepares the hardware buffers for the requested vertex and index counts.

Protected Attributes

- size_t **indexBufferCapacity**

  Maximum capacity of the currently allocated index buffer.
- size_t **vertexBufferCapacity**

  Maximum capacity of the currently allocated vertex buffer.

10.40.1 Detailed Description

Abstract base class providing mechanisms for dynamically growing hardware buffers.

10.40.2 Constructor & Destructor Documentation

10.40.2.1 gazebo::rendering::DynamicRenderable::DynamicRenderable ( )

Constructor.

10.40.2.2 virtual gazebo::rendering::DynamicRenderable::~DynamicRenderable ( ) [virtual]

Virtual destructor.

10.40.3 Member Function Documentation

10.40.3.1 virtual void gazebo::rendering::DynamicRenderable::CreateVertexDeclaration ( ) [protected], [pure virtual]

Creates the vertex declaration.

Remarks

Override and set mRenderOp.vertexData->vertexDeclaration here. mRenderOp.vertexData will be created for you before this method is called.

Implemented in **gazebo::rendering::DynamicLines** (p. 268).

10.40.3.2 virtual void gazebo::rendering::DynamicRenderable::FillHardwareBuffers ( ) [protected], [pure virtual]

Fills the hardware vertex and index buffers with data.

Remarks

This function must call prepareHardwareBuffers() before locking the buffers to ensure 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. 268).

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

Implementation of Ogre::SimpleRenderable.

Returns

The bounding radius

10.40.3.4 RenderOpType gazebo::rendering::DynamicRenderable::GetOperationType ( ) const

Get the render operation type.

Returns

The render operation type.

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

Implementation of Ogre::SimpleRenderable.

Parameters

in	*_cam*	Pointer to the **Ogre** (p. 106) camera that views the renderable.

Returns

The squared depth in the **Camera** (p. 162)'s view

10.40.3.6 void gazebo::rendering::DynamicRenderable::Init (RenderOpType *_opType*, bool *_useIndices* = false)

Initializes the dynamic renderable.

Remarks

This function should only be called once. It initializes the render operation, and calls the abstract function **CreateVertexDeclaration()** (p. 271).

Parameters

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

10.40.3.7 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	*_vertexCount*	The number of vertices the buffer must hold.
in	*_indexCount*	The number of indices the buffer must hold. This parameter is ignored if not using indices.

10.40.3.8 void gazebo::rendering::DynamicRenderable::SetOperationType (RenderOpType *_opType*)

Set the render operation type.

Parameters

in	*_opType*	The type of render operation to perform.

10.40.4 Member Data Documentation

10.40.4.1 size_t gazebo::rendering::DynamicRenderable::indexBufferCapacity [protected]

Maximum capacity of the currently allocated index buffer.

10.40.4.2 size_t gazebo::rendering::DynamicRenderable::vertexBufferCapacity [protected]

Maximum capacity of the currently allocated vertex buffer.

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

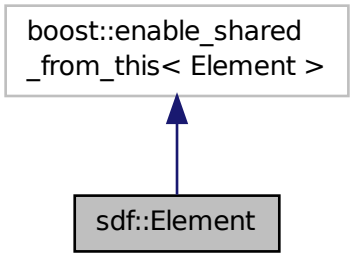
- **DynamicRenderable.hh**

10.41 sdf::Element Class Reference

SDF (p. 695) **Element** (p. 273) class.

`#include <SDF.hh>`

Inheritance diagram for sdf::Element:



Public Member Functions

- **Element** ()
- virtual ~**Element** ()
- void **AddAttribute** (const std::string &_key, const std::string &_type, const std::string &_defaultvalue, bool _required, const std::string &_description="")
- **ElementPtr AddElement** (const std::string &_name)
- void **AddElementDescription** (**ElementPtr** _elem)

  Add a new element description.
- void **AddValue** (const std::string &_type, const std::string &_defaultValue, bool _required, const std::string &_description="")
- void **ClearElements** ()

  Remove all child elements.
- **Element** * **Clone** () const
- void **Copy** (const **ElementPtr** _elem)

  *Copy values from an **Element** (p. 273).*
- **ParamPtr GetAttribute** (const std::string &_key)

  Get the param of an attribute.
- **ParamPtr GetAttribute** (unsigned int _index) const

  Get an attribute using an index.
- unsigned int **GetAttributeCount** () const

  Get the number of attributes.
- bool **GetAttributeSet** (const std::string &_key)

  Return true if the attribute was set (i.e. not default value)
- bool **GetCopyChildren** () const
- std::string **GetDescription** () const

  Get a text description of the element.
- **ElementPtr GetElement** (const std::string &_name) const
- **ElementPtr GetElement** (const std::string &_name)
- **ElementPtr GetElementDescription** (unsigned int _index) const

  Get an element description using an index.

- **ElementPtr GetElementDescription** (const std::string &_key) const

  Get an element descriptio using a key.
- unsigned int **GetElementDescriptionCount** () const

  Get the number of element descriptions.
- **ElementPtr GetElementImpl** (const std::string &_name) const
- **ElementPtr GetFirstElement** () const
- std::string **GetInclude** () const
- const std::string & **GetName** () const
- **ElementPtr GetNextElement** (const std::string &_name="") const
- **ElementPtr GetParent** () const
- const std::string & **GetRequired** () const
- **ParamPtr GetValue** ()

  Get the param of the elements value.
- bool **GetValueBool** (const std::string &_key="")
- char **GetValueChar** (const std::string &_key="")
- **gazebo::common::Color GetValueColor** (const std::string &_key="")
- double **GetValueDouble** (const std::string &_key="")
- float **GetValueFloat** (const std::string &_key="")
- int **GetValueInt** (const std::string &_key="")
- **gazebo::math::Pose GetValuePose** (const std::string &_key="")
- **gazebo::math::Quaternion GetValueQuaternion** (const std::string &_key="")
- std::string **GetValueString** (const std::string &_key="")
- **gazebo::common::Time GetValueTime** (const std::string &_key="")
- unsigned int **GetValueUInt** (const std::string &_key="")
- **gazebo::math::Vector2d GetValueVector2d** (const std::string &_key="")
- **gazebo::math::Vector3 GetValueVector3** (const std::string &_key="")
- bool **HasAttribute** (const std::string &_key)
- bool **HasElement** (const std::string &_name) const
- bool **HasElementDescription** (const std::string &_name)

  Return true if an element description exists.
- void **InsertElement** (**ElementPtr** _elem)
- void **PrintDescription** (std::string _prefix)
- void **PrintDocLeftPane** (std::string &_html, int _spacing, int &_index)

  *Helper function for **SDF::PrintDoc*** (p. *695*).
- void **PrintDocRightPane** (std::string &_html, int _spacing, int &_index)

  *Helper function for **SDF::PrintDoc*** (p. *695*).
- void **PrintValues** (std::string _prefix)
- void **PrintWiki** (std::string _prefix)
- void **RemoveChild** (**ElementPtr** _child)

  Remove a child element.
- void **RemoveFromParent** ()

  Remove this element from its parent.
- void **Reset** ()
- bool **Set** (const bool &_value)
- bool **Set** (const int &_value)
- bool **Set** (const unsigned int &_value)
- bool **Set** (const float &_value)
- bool **Set** (const double &_value)
- bool **Set** (const char &_value)

- bool **Set** (const std::string &_value)
- bool **Set** (const char *_value)
- bool **Set** (const **gazebo::math::Vector3** &_value)
- bool **Set** (const **gazebo::math::Vector2i** &_value)
- bool **Set** (const **gazebo::math::Vector2d** &_value)
- bool **Set** (const **gazebo::math::Quaternion** &_value)
- bool **Set** (const **gazebo::math::Pose** &_value)
- bool **Set** (const **gazebo::common::Color** &_value)
- bool **Set** (const **gazebo::common::Time** &_value)
- void **SetCopyChildren** (bool _value)
- void **SetDescription** (const std::string &_desc)

  Set a text description for the element.
- void **SetInclude** (const std::string &_filename)
- void **SetName** (const std::string &_name)
- void **SetParent** (const **ElementPtr** _parent)
- void **SetRequired** (const std::string &_req)
- std::string **ToString** (const std::string &_prefix) const
- void **Update** ()

10.41.1 Detailed Description

SDF (p. 695) **Element** (p. 273) class.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 sdf::Element::Element ( )

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

10.41.3 Member Function Documentation

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

10.41.3.2 ElementPtr sdf::Element::AddElement (const std::string & *_name*)

10.41.3.3 void sdf::Element::AddElementDescription (ElementPtr *_elem*)

Add a new element description.

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

10.41.3.5 void sdf::Element::ClearElements ( )

Remove all child elements.

10.41.3.6 Element* sdf::Element::Clone ( ) const

10.41.3.7 void sdf::Element::Copy (const ElementPtr _elem)

Copy values from an **Element** (p. 273).

10.41.3.8 ParamPtr sdf::Element::GetAttribute (const std::string & _key)

Get the param of an attribute.

Parameters

_key	the name of the attribute

10.41.3.9 ParamPtr sdf::Element::GetAttribute (unsigned int _index) const

Get an attribute using an index.

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

Get the number of attributes.

10.41.3.11 bool sdf::Element::GetAttributeSet (const std::string & _key)

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

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

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

Get a text description of the element.

10.41.3.14 ElementPtr sdf::Element::GetElement (const std::string & _name) const

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

10.41.3.15 ElementPtr sdf::Element::GetElement (const std::string & _name)

10.41.3.16 ElementPtr sdf::Element::GetElementDescription (unsigned int _index) const

Get an element description using an index.

10.41.3.17 ElementPtr sdf::Element::GetElementDescription (const std::string & _key) const

Get an element descriptio using a key.

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

Get the number of element descriptions.

10.41.3.19 **ElementPtr** sdf::Element::GetElementImpl (const std::string & *_name*) const

10.41.3.20 **ElementPtr** sdf::Element::GetFirstElement ( ) const

10.41.3.21 std::string sdf::Element::GetInclude ( ) const

10.41.3.22 const std::string& sdf::Element::GetName ( ) const

10.41.3.23 **ElementPtr** sdf::Element::GetNextElement (const std::string & *_name* = " ") const

10.41.3.24 **ElementPtr** sdf::Element::GetParent ( ) const

10.41.3.25 const std::string& sdf::Element::GetRequired ( ) const

10.41.3.26 **ParamPtr** sdf::Element::GetValue ( )

Get the param of the elements value.

10.41.3.27 bool sdf::Element::GetValueBool (const std::string & *_key* = " ")

10.41.3.28 char sdf::Element::GetValueChar (const std::string & *_key* = " ")

10.41.3.29 **gazebo::common::Color** sdf::Element::GetValueColor (const std::string & *_key* = " ")

10.41.3.30 double sdf::Element::GetValueDouble (const std::string & *_key* = " ")

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

10.41.3.31 float sdf::Element::GetValueFloat (const std::string & *_key* = " ")

10.41.3.32 int sdf::Element::GetValueInt (const std::string & *_key* = " ")

10.41.3.33 **gazebo::math::Pose** sdf::Element::GetValuePose (const std::string & *_key* = " ")

10.41.3.34 **gazebo::math::Quaternion** sdf::Element::GetValueQuaternion (const std::string & *_key* = " ")

10.41.3.35 std::string sdf::Element::GetValueString (const std::string & *_key* = " ")

10.41.3.36 **gazebo::common::Time** sdf::Element::GetValueTime (const std::string & *_key* = " ")

10.41.3.37 unsigned int sdf::Element::GetValueUInt (const std::string & *_key* = " ")

10.41.3.38 **gazebo::math::Vector2d** sdf::Element::GetValueVector2d (const std::string & *_key* = " ")

10.41.3.39 **gazebo::math::Vector3** sdf::Element::GetValueVector3 (const std::string & *_key* = " ")

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

10.41.3.40 bool sdf::Element::HasAttribute (const std::string & *_key*)

10.41.3.41 bool sdf::Element::HasElement (const std::string & *_name*) const

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

10.41.3.42 bool sdf::Element::HasElementDescription (const std::string & *_name*)

Return true if an element description exists.

10.41.3.43 void sdf::Element::InsertElement (ElementPtr *_elem*)

10.41.3.44 void sdf::Element::PrintDescription (std::string *_prefix*)

10.41.3.45 void sdf::Element::PrintDocLeftPane (std::string & *_html*, int *_spacing*, int & *_index*)

Helper function for **SDF::PrintDoc** (p. 695).

This generates the **SDF** (p. 695) html documentation.

Parameters

out	*_html*	Accumulated HTML for output.
in	*_spacing*	Amount of spacing for this element.
in	*_index*	Unique index for this element.

10.41.3.46 void sdf::Element::PrintDocRightPane (std::string & *_html*, int *_spacing*, int & *_index*)

Helper function for **SDF::PrintDoc** (p. 695).

This generates the **SDF** (p. 695) html documentation.

Parameters

out	*_html*	Accumulated HTML for output
in	*_spacing*	Amount of spacing for this element.

10.41.3.47 void sdf::Element::PrintValues (std::string *_prefix*)

10.41.3.48 void sdf::Element::PrintWiki (std::string *_prefix*)

10.41.3.49 void sdf::Element::RemoveChild (ElementPtr *_child*)

Remove a child element.

Parameters

in	*_child*	Pointer to the child to remove.

10.41.3.50 void sdf::Element::RemoveFromParent ( )

Remove this element from its parent.

10.41.3.51 void sdf::Element::Reset ( )

10.41.3.52 bool sdf::Element::Set (const bool & *_value*)

10.41.3.53 bool sdf::Element::Set (const int & *_value*)

10.41.3.54 bool sdf::Element::Set (const unsigned int & *_value*)

10.41.3.55 bool sdf::Element::Set (const float & *_value*)

10.41.3.56 bool sdf::Element::Set (const double & *_value*)

10.41.3.57 bool sdf::Element::Set (const char & *_value*)

10.41.3.58 bool sdf::Element::Set (const std::string & *_value*)

10.41.3.59 bool sdf::Element::Set (const char * *_value*)

10.41.3.60 bool sdf::Element::Set (const **gazebo::math::Vector3** & *_value*)

10.41.3.61 bool sdf::Element::Set (const **gazebo::math::Vector2i** & *_value*)

10.41.3.62 bool sdf::Element::Set (const **gazebo::math::Vector2d** & *_value*)

10.41.3.63 bool sdf::Element::Set (const **gazebo::math::Quaternion** & *_value*)

10.41.3.64 bool sdf::Element::Set (const **gazebo::math::Pose** & *_value*)

10.41.3.65 bool sdf::Element::Set (const **gazebo::common::Color** & *_value*)

10.41.3.66 bool sdf::Element::Set (const **gazebo::common::Time** & *_value*)

10.41.3.67 void sdf::Element::SetCopyChildren (bool *_value*)

10.41.3.68 void sdf::Element::SetDescription (const std::string & *_desc*)

Set a text description for the element.

10.41.3.69 void sdf::Element::SetInclude (const std::string & *_filename*)

10.41.3.70 void sdf::Element::SetName (const std::string & *_name*)

10.41.3.71 void sdf::Element::SetParent (const **ElementPtr** *_parent*)

10.41.3.72 void sdf::Element::SetRequired (const std::string & *_req*)

10.41.3.73 std::string sdf::Element::ToString (const std::string & *_prefix*) const

10.41.3.74 void sdf::Element::Update ( )

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

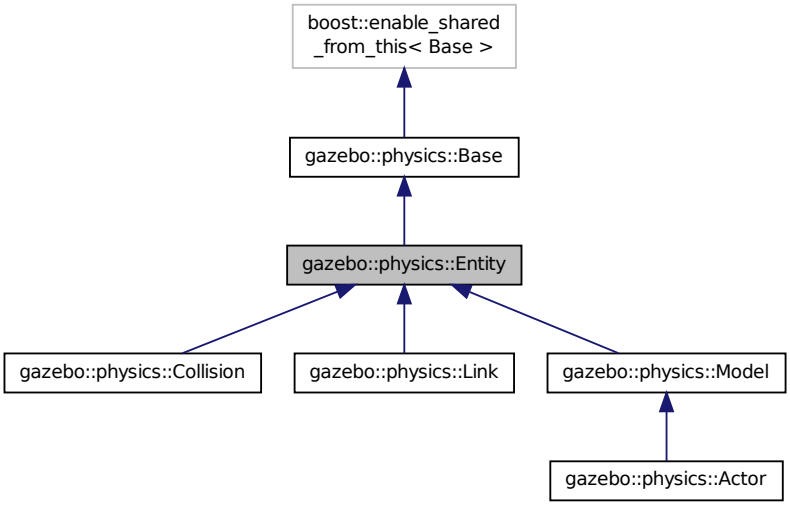
- **SDF.hh**

10.42 gazebo::physics::Entity Class Reference

Base (p. 137) class for all physics objects in Gazebo.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Entity:



Public Member Functions

- **Entity** (**BasePtr** _parent)

  Constructor.
- virtual ~**Entity** ()

  Destructor.
- virtual void **Fini** ()

  Finalize the entity.
- virtual **math::Box GetBoundingBox** () const

  Return the bounding box for the entity.
- **CollisionPtr GetChildCollision** (const std::string &_name)

  Get a child collision entity, if one exists.
- **LinkPtr GetChildLink** (const std::string &_name)

Get a child linke entity, if one exists.

- **math::Box GetCollisionBoundingBox** () const

  Returns collision bounding box.

- const **math::Pose** & **GetDirtyPose** () const

  *Returns **Entity::dirtyPose** (p. 292).*

- **math::Pose GetInitialRelativePose** () const

  Get the initial relative pose.

- void **GetNearestEntityBelow** (double &_distBelow, std::string &_entityName)

  Get the distance to the nearest entity below (along the Z-axis) this entity.

- **ModelPtr GetParentModel** ()

  Get the parent model, if one exists.

- virtual **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the entity.

- virtual **math::Vector3 GetRelativeAngularVel** () const

  Get the angular velocity of the entity.

- virtual **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the entity.

- virtual **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the entity.

- **math::Pose GetRelativePose** () const

  Get the pose of the entity relative to its parent.

- virtual **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the entity in the world frame.

- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the entity in the world frame.

- virtual **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the entity in the world frame.

- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the entity in the world frame.

- const **math::Pose** & **GetWorldPose** () const

  Get the absolute pose of the entity.

- bool **IsCanonicalLink** () const

  ***A** (p. 111) 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. 281).
- virtual void **StopAnimation** ()

  Stop the current animation, if any.
- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  Update the parameters using new sdf values.

Protected Member Functions

- virtual void **OnPoseChange** ()=0

  This function is called when the entity's (or one of its parents) pose of the parent has changed.

Protected Attributes

- **common::PoseAnimationPtr animation**

  Current pose animation.
- **event::ConnectionPtr animationConnection**

  Connection used to update an animation.
- **math::Pose animationStartPose**

  Start pose of an animation.
- std::vector< **event::ConnectionPtr** > **connections**

  All our event connections.
- **math::Pose dirtyPose**

  The pose set by a physics engine.
- **transport::NodePtr node**

  Communication node.
- **EntityPtr parentEntity**

  A (p. 111) *helper that prevents numerous dynamic_casts.*
- msgs::Pose * **poseMsg**

  Pose message container.
- **common::Time prevAnimationTime**

  Previous time an animation was updated.
- **transport::PublisherPtr requestPub**

  Request publisher.

- **transport::PublisherPtr visPub**

  Visual publisher.
- msgs::Visual * **visualMsg**

  Visual message container.

Additional Inherited Members

10.42.1 Detailed Description

Base (p. 137) class for all physics objects in Gazebo.

10.42.2 Constructor & Destructor Documentation

10.42.2.1 gazebo::physics::Entity::Entity (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of the entity.

10.42.2.2 virtual gazebo::physics::Entity::~Entity ( ) [virtual]

Destructor.

10.42.3 Member Function Documentation

10.42.3.1 virtual void gazebo::physics::Entity::Fini ( ) [virtual]

Finalize the entity.

Reimplemented from **gazebo::physics::Base** (p. 142).

Reimplemented in **gazebo::physics::Actor** (p. 114), **gazebo::physics::Model** (p. 493), **gazebo::physics::Link** (p. 426), and **gazebo::physics::Collision** (p. 199).

10.42.3.2 virtual math::Box gazebo::physics::Entity::GetBoundingBox ( ) const [virtual]

Return the bounding box for the entity.

Returns

The bounding box.

Reimplemented in **gazebo::physics::Link** (p. 426), **gazebo::physics::Model** (p. 494), and **gazebo::physics::Collision** (p. 199).

10.42.3.3 CollisionPtr gazebo::physics::Entity::GetChildCollision (const std::string & *_name*)

Get a child collision entity, if one exists.

Parameters

in	*_name*	Name of the child collision object.

Returns

Pointer to the **Collision** (p. 195) object, or NULL if not found.

10.42.3.4 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. 418) object.

Returns

Pointer to the **Link** (p. 418) object, or NULL if not found.

10.42.3.5 math::Box gazebo::physics::Entity::GetCollisionBoundingBox ( ) const

Returns collision bounding box.

Returns

Collsiion boundin box.

10.42.3.6 const math::Pose& gazebo::physics::Entity::GetDirtyPose ( ) const

Returns **Entity::dirtyPose** (p. 292).

The dirty pose is the pose set by the physics engine before it's value is propagated to the rest of the simulator.

Returns

The dirty pose of the entity.

10.42.3.7 math::Pose gazebo::physics::Entity::GetInitialRelativePose ( ) const

Get the initial relative pose.

Returns

The initial relative pose.

10.42.3.8 void gazebo::physics::Entity::GetNearestEntityBelow (double & *_distBelow,* std::string & *_entityName*)

Get the distance to the nearest entity below (along the Z-axis) this entity.

Parameters

out	*_distBelow*	The distance to the nearest entity below.
out	*_entityName*	The name of the nearest entity below.

10.42.3.9 ModelPtr gazebo::physics::Entity::GetParentModel ( )

Get the parent model, if one exists.

Returns

Pointer to a model, or NULL if no parent model exists.

10.42.3.10 virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularAccel ( ) const [inline], [virtual]

Get the angular acceleration of the entity.

Returns

A (p. 111) **math::Vector3** (p. 855) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 429), **gazebo::physics::Collision** (p. 200), and **gazebo::physics::Model** (p. 495).

10.42.3.11 virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularVel ( ) const [inline], [virtual]

Get the angular velocity of the entity.

Returns

A (p. 111) **math::Vector3** (p. 855) for the velocity.

Reimplemented in **gazebo::physics::Link** (p. 429), **gazebo::physics::Collision** (p. 200), and **gazebo::physics::Model** (p. 496).

10.42.3.12 virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearAccel ( ) const [inline], [virtual]

Get the linear acceleration of the entity.

Returns

A (p. 111) **math::Vector3** (p. 855) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 429), **gazebo::physics::Collision** (p. 200), and **gazebo::physics::Model** (p. 496).

10.42.3.13 virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearVel ( ) const [inline], [virtual]

Get the linear velocity of the entity.

Returns

A (p. 111) **math::Vector3** (p. 855) for the linear velocity.

Reimplemented in **gazebo::physics::Link** (p. 429), **gazebo::physics::Collision** (p. 200), and **gazebo::physics::Model** (p. 496).

10.42.3.14 math::Pose gazebo::physics::Entity::GetRelativePose ( ) const

Get the pose of the entity relative to its parent.

Returns

The pose of the entity relative to its parent.

10.42.3.15 virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularAccel ( ) const [inline], [virtual]

Get the angular acceleration of the entity in the world frame.

Returns

A (p. 111) **math::Vector3** (p. 855) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 430), **gazebo::physics::Collision** (p. 201), and **gazebo::physics::Model** (p. 497).

10.42.3.16 virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularVel ( ) const [inline], [virtual]

Get the angular velocity of the entity in the world frame.

Returns

A (p. 111) **math::Vector3** (p. 855) for the velocity.

Reimplemented in **gazebo::physics::Collision** (p. 202), and **gazebo::physics::Model** (p. 497).

10.42.3.17 virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearAccel ( ) const [inline], [virtual]

Get the linear acceleration of the entity in the world frame.

Returns

A (p. 111) **math::Vector3** (p. 855) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 431), **gazebo::physics::Collision** (p. 202), and **gazebo::physics::Model** (p. 497).

10.42.3.18 virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearVel ( ) const [inline], [virtual]

Get the linear velocity of the entity in the world frame.

Returns

A (p. 111) **math::Vector3** (p. 855) for the linear velocity.

Reimplemented in **gazebo::physics::Collision** (p. 202), and **gazebo::physics::Model** (p. 497).

10.42.3.19 const math::Pose& gazebo::physics::Entity::GetWorldPose ( ) const [inline]

Get the absolute pose of the entity.

Returns

The absolute pose of the entity.

Referenced by gazebo::sensors::RFIDTag::GetTagPose().

10.42.3.20 bool gazebo::physics::Entity::IsCanonicalLink ( ) const [inline]

A (p. 111) helper function that checks if this is a canonical body.

Returns

True if the link is canonical.

10.42.3.21 bool gazebo::physics::Entity::IsStatic ( ) const

Return whether this entity is static.

Returns

True if static.

10.42.3.22 virtual void gazebo::physics::Entity::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the entity.

Parameters

in	*_sdf*	Pointer to an SDF element.

Reimplemented from **gazebo::physics::Base** (p. 145).

Reimplemented in **gazebo::physics::Actor** (p. 115), **gazebo::physics::Model** (p. 497), **gazebo::physics::Link** (p. 432), and **gazebo::physics::Collision** (p. 202).

10.42.3.23 virtual void gazebo::physics::Entity::OnPoseChange ( ) [protected], [pure virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implemented in **gazebo::physics::Link** (p. 432), and **gazebo::physics::Model** (p. 498).

10.42.3.24 void gazebo::physics::Entity::PlaceOnEntity (const std::string & *_entityName*)

Move this entity to be ontop of another entity by name.

Parameters

in	*_entityName*	Name of the **Entity** (p. 281) this **Entity** (p. 281) should be ontop of.

10.42.3.25 void gazebo::physics::Entity::PlaceOnNearestEntityBelow ( )

Move this entity to be ontop of the nearest entity below.

10.42.3.26 virtual void gazebo::physics::Entity::Reset ( ) [virtual]

Reset the entity.

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

Reimplemented in **gazebo::physics::Model** (p. 498), and **gazebo::physics::Link** (p. 433).

10.42.3.27 void gazebo::physics::Entity::SetAnimation (const common::PoseAnimationPtr & *_anim*, boost::function< void() > *_onComplete*)

Set an animation for this entity.

Parameters

in	*_anim*	Pose animation.
in	*_onComplete*	Callback for when the animation completes.

10.42.3.28 void gazebo::physics::Entity::SetAnimation (common::PoseAnimationPtr *_anim*)

Set an animation for this entity.

Parameters

in	*_anim*	Pose animation.

10.42.3.29 void gazebo::physics::Entity::SetCanonicalLink (bool *_value*)

Set to true if this entity is a canonical link for a model.

Parameters

in	*_value*	True if the link is canonical.

10.42.3.30 void gazebo::physics::Entity::SetInitialRelativePose (const math::Pose & *_pose*)

Set the initial pose.

Parameters

in	*_pose*	The initial pose.

10.42.3.31 virtual void gazebo::physics::Entity::SetName (const std::string & *_name*) [virtual]

Set the name of the entity.

Parameters

in	*_name*	The new name.

Reimplemented from **gazebo::physics::Base** (p. 147).

10.42.3.32 void gazebo::physics::Entity::SetRelativePose (const math::Pose & *_pose*, bool *_notify* = true, bool *_publish* = true)

Set the pose of the entity relative to its parent.

Parameters

in	*_pose*	The new pose.
in	*_notify*	True = tell children of the pose change.
in	*_publish*	True to publish the pose.

10.42.3.33 void gazebo::physics::Entity::SetStatic (const bool & *_static*)

Set whether this entity is static: immovable.

Parameters

in	*_static*	True = static.

10.42.3.34 void gazebo::physics::Entity::SetWorldPose (const math::Pose & *_pose*, bool *_notify* = true, bool *_publish* = true)

Set the world pose of the entity.

Parameters

in	*_pose*	The new world pose.
in	*_notify*	True = tell children of the pose change.

in	*_publish*	True to publish the pose.

10.42.3.35 void gazebo::physics::Entity::SetWorldTwist (const math::Vector3 & *_linear,* const math::Vector3 & *_angular,* bool *_updateChildren* = true)

Set angular and linear rates of an **physics::Entity** (p. 281).

Parameters

in	*_linear*	Linear twist.
in	*_angular*	Angular twist.
in	*_updateChildren*	True to pass this update to child entities.

10.42.3.36 virtual void gazebo::physics::Entity::StopAnimation ( ) [virtual]

Stop the current animation, if any.

Reimplemented in **gazebo::physics::Model** (p. 501).

10.42.3.37 virtual void gazebo::physics::Entity::UpdateParameters (sdf::ElementPtr *_sdf*) [virtual]

Update the parameters using new sdf values.

Parameters

in	*_sdf*	SDF to update from.

Reimplemented from **gazebo::physics::Base** (p. 148).

Reimplemented in **gazebo::physics::Actor** (p. 115), **gazebo::physics::Model** (p. 502), **gazebo::physics::Link** (p. 437), and **gazebo::physics::Collision** (p. 204).

10.42.4 Member Data Documentation

10.42.4.1 common::PoseAnimationPtr gazebo::physics::Entity::animation [protected]

Current pose animation.

10.42.4.2 event::ConnectionPtr gazebo::physics::Entity::animationConnection [protected]

Connection used to update an animation.

10.42.4.3 math::Pose gazebo::physics::Entity::animationStartPose [protected]

Start pose of an animation.

10.42.4.4 std::vector<event::ConnectionPtr> gazebo::physics::Entity::connections [protected]

All our event connections.

10.42.4.5 math::Pose gazebo::physics::Entity::dirtyPose [protected]

The pose set by a physics engine.

10.42.4.6 transport::NodePtr gazebo::physics::Entity::node [protected]

Communication node.

10.42.4.7 EntityPtr gazebo::physics::Entity::parentEntity [protected]

A (p. 111) helper that prevents numerous dynamic_casts.

10.42.4.8 msgs::Pose* gazebo::physics::Entity::poseMsg [protected]

Pose message container.

10.42.4.9 common::Time gazebo::physics::Entity::prevAnimationTime [protected]

Previous time an animation was updated.

10.42.4.10 transport::PublisherPtr gazebo::physics::Entity::requestPub [protected]

Request publisher.

10.42.4.11 transport::PublisherPtr gazebo::physics::Entity::visPub [protected]

Visual publisher.

10.42.4.12 msgs::Visual* gazebo::physics::Entity::visualMsg [protected]

Visual message container.

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

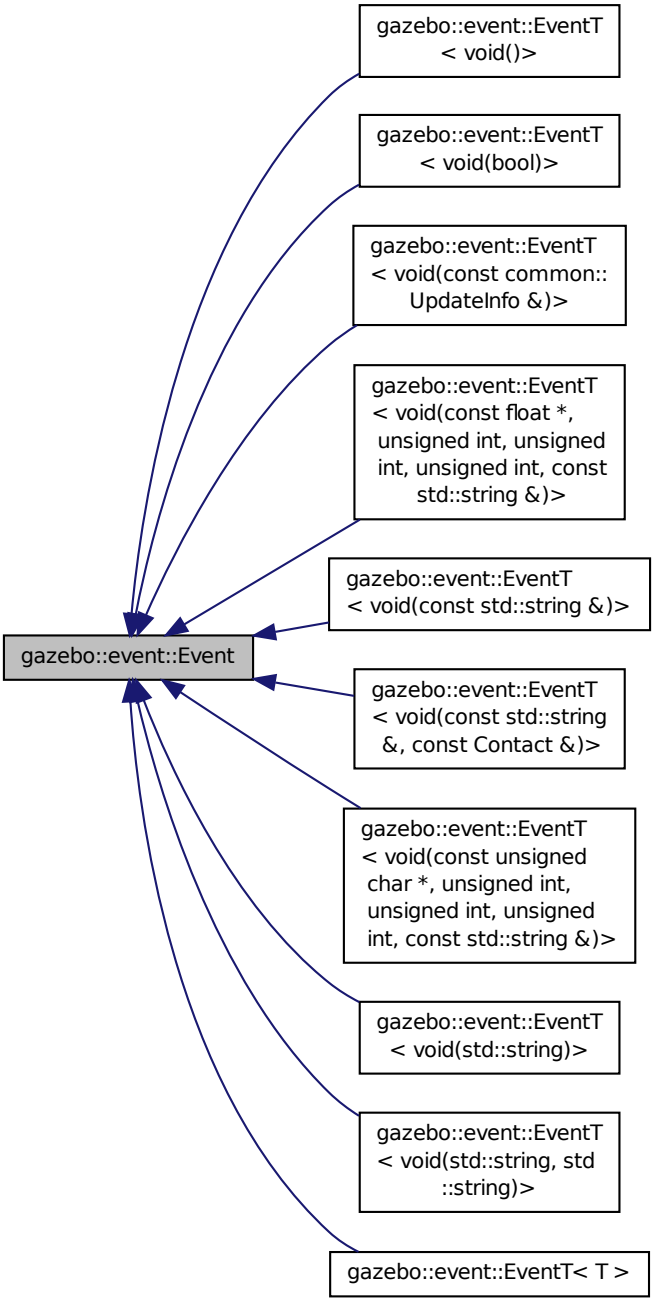
- **Entity.hh**

10.43 gazebo::event::Event Class Reference

Base class for all events.

`#include <common/common.hh>`

Inheritance diagram for gazebo::event::Event:



Public Member Functions

- virtual ~**Event** ()

Constructor.

- virtual void **Disconnect** (**ConnectionPtr** _c)=0

  Disconnect.

- virtual void **Disconnect** (int _id)=0

  Disconnect.

10.43.1 Detailed Description

Base class for all events.

10.43.2 Constructor & Destructor Documentation

10.43.2.1 virtual gazebo::event::Event::~Event ( ) [inline], [virtual]

Constructor.

10.43.3 Member Function Documentation

10.43.3.1 virtual void gazebo::event::Event::Disconnect (ConnectionPtr *_c*) [pure virtual]

Disconnect.

Parameters

in	*_c*	**A** (p. 111) pointer to a connection

Implemented in **gazebo::event::EventT< T >** (p. 39), **gazebo::event::EventT< void(std::string)>** (p. 39), **gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 39), **gazebo::event::EventT< void(const std::string &)>** (p. 39), **gazebo::event::EventT< void()>** (p. 39), **gazebo::event::EventT< void(const common::UpdateInfo &)>** (p. 39), **gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 39), **gazebo::event::EventT< void(const std::string &, const Contact &)>** (p. 39), **gazebo::event::EventT< void(std::string, std::string)>** (p. 39), and **gazebo::event::EventT< void(bool)>** (p. 39).

10.43.3.2 virtual void gazebo::event::Event::Disconnect (int *_id*) [pure virtual]

Disconnect.

Parameters

in	*_id*	Integer ID of a connection

Implemented in **gazebo::event::EventT< T >** (p. 40), **gazebo::event::EventT< void(std::string)>** (p. 40), **gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 40), **gazebo::event::EventT< void(const std::string &)>** (p. 40), **gazebo::event::EventT< void()>** (p. 40), **gazebo::event::EventT< void(const common::UpdateInfo &)>** (p. 40), **gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 40), **gazebo::event::EventT< void(const std::string &, const Contact &)>** (p. 40), **gazebo::event::EventT< void(std::string, std::string)>** (p. 40), and **gazebo::event::EventT< void(bool)>** (p. 40).

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

- **Event.hh**

10.44 gazebo::rendering::Events Class Reference

Base class for rendering events.

`#include <rendering/rendering.hh>`

Static Public Member Functions

- template<typename T >
  static **event::ConnectionPtr ConnectCreateScene** (T _subscriber)

  Connect to a scene created event.
- template<typename T >
  static **event::ConnectionPtr ConnectRemoveScene** (T _subscriber)

  Connect to a scene removed event.
- static void **DisconnectCreateScene** (**event::ConnectionPtr** _connection)

  Disconnect from a scene created event.
- static void **DisconnectRemoveScene** (**event::ConnectionPtr** _connection)

  Disconnect from a scene removed event.

Static Public Attributes

- static **event::EventT**< void(const std::string &)> **createScene**

  The event used to trigger a create scene event.
- static **event::EventT**< void(const std::string &)> **removeScene**

  The event used to trigger a remve scene event.

10.44.1 Detailed Description

Base class for rendering events.

10.44.2 Member Function Documentation

10.44.2.1 template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectCreateScene (T _subscriber) [inline], [static]

Connect to a scene created event.

Parameters

in	*_subscriber*	Callback to trigger when event occurs.

Returns

Pointer the connection. This must stay in scope.

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

10.44.2.2 template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectRemoveScene (T _subscriber) [inline], [static]

Connect to a scene removed event.

Parameters

in	*_subscriber*	Callback to trigger when event occurs.

Returns

Pointer the connection. This must stay in scope.

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

10.44.2.3 static void gazebo::rendering::Events::DisconnectCreateScene (event::ConnectionPtr _connection) [inline], [static]

Disconnect from a scene created event.

Parameters

in	*_connection*	The connection to disconnect.

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

10.44.2.4 static void gazebo::rendering::Events::DisconnectRemoveScene (event::ConnectionPtr _connection) [inline], [static]

Disconnect from a scene removed event.

Parameters

in	*_connection*	The connection to disconnect.

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

10.44.3 Member Data Documentation

10.44.3.1 event::EventT<void (const std::string &)> gazebo::rendering::Events::createScene [static]

The event used to trigger a create scene event.

Referenced by ConnectCreateScene(), and DisconnectCreateScene().

10.44.3.2 event::EventT<void (const std::string &)> gazebo::rendering::Events::removeScene [static]

The event used to trigger a remve scene event.

Referenced by ConnectRemoveScene(), and DisconnectRemoveScene().

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

- **RenderEvents.hh**

10.45 gazebo::event::Events Class Reference

An **Event** (p. 292) class to get notifications for simulator events.

`#include <common/common.hh>`

Static Public Member Functions

- template<typename T >
  static **ConnectionPtr ConnectAddEntity** (T _subscriber)

  Connect a boost::slot the the add entity signal.
- template<typename T >
  static **ConnectionPtr ConnectCreateEntity** (T _subscriber)

  Connect a boost::slot the the add entity signal.
- template<typename T >
  static **ConnectionPtr ConnectDeleteEntity** (T _subscriber)

  Connect a boost::slot the delete entity.
- template<typename T >
  static **ConnectionPtr ConnectDiagTimerStart** (T _subscriber)

  Connect a boost::slot the diagnostic timer start signal.
- template<typename T >
  static **ConnectionPtr ConnectDiagTimerStop** (T _subscriber)

  Connect a boost::slot the diagnostic timer stop signal.
- template<typename T >
  static **ConnectionPtr ConnectPause** (T _subscriber)

  Connect a boost::slot the the pause signal.
- template<typename T >
  static **ConnectionPtr ConnectPostRender** (T _subscriber)

  Connect a boost::slot the post render update signal.
- template<typename T >
  static **ConnectionPtr ConnectPreRender** (T _subscriber)

  Render start signal.
- template<typename T >
  static **ConnectionPtr ConnectRender** (T _subscriber)

  Connect a boost::slot the render update signal.
- template<typename T >
  static **ConnectionPtr ConnectSetSelectedEntity** (T _subscriber)

  Connect a boost::slot the set selected entity.
- template<typename T >
  static **ConnectionPtr ConnectStep** (T _subscriber)

Connect a boost::slot the the step signal.

- template<typename T >
  static **ConnectionPtr ConnectStop** (T _subscriber)

  Connect a boost::slot the the stop signal.
- template<typename T >
  static **ConnectionPtr ConnectWorldCreated** (T _subscriber)

  Connect a boost::slot the the world created signal.
- template<typename T >
  static **ConnectionPtr ConnectWorldUpdateBegin** (T _subscriber)

  Connect a boost::slot the the world update start signal.
- template<typename T >
  static **ConnectionPtr ConnectWorldUpdateEnd** (T _subscriber)

  Connect a boost::slot the the world update end signal.
- template<typename T >
  static **ConnectionPtr ConnectWorldUpdateStart** (T _subscriber)

  Connect a boost::slot the the world update start signal.
- static void **DisconnectAddEntity** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the add entity signal.
- static void **DisconnectCreateEntity** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the add entity signal.
- static void **DisconnectDeleteEntity** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the delete entity.
- static void **DisconnectDiagTimerStart** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the diagnostic timer start signal.
- static void **DisconnectDiagTimerStop** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the diagnostic timer stop signal.
- static void **DisconnectPause** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the pause signal.
- static void **DisconnectPostRender** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the post render update signal.
- static void **DisconnectPreRender** (**ConnectionPtr** _subscriber)

  Disconnect a render start signal.
- static void **DisconnectRender** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the render update signal.
- static void **DisconnectSetSelectedEntity** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the set selected entity.
- static void **DisconnectStep** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the step signal.
- static void **DisconnectStop** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the stop signal.
- static void **DisconnectWorldCreated** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the world created signal.
- static void **DisconnectWorldUpdateBegin** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the world update start signal.
- static void **DisconnectWorldUpdateEnd** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the world update end signal.
- static void **DisconnectWorldUpdateStart** (**ConnectionPtr** _subscriber) **GAZEBO_DEPRECATED**(1.5)

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

Static Public Attributes

- static **EventT**< void(std::string)> **addEntity**

  An entity has been added.
- static **EventT**< void(std::string)> **deleteEntity**

  An entity has been deleted.
- static **EventT**< void(std::string)> **diagTimerStart**

  Diagnostic timer start.
- static **EventT**< void(std::string)> **diagTimerStop**

  Diagnostic timer stop.
- static **EventT**< void(std::string)> **entityCreated**

  An entity has been created.
- static **EventT**< void(bool)> **pause**

  Pause signal.
- static **EventT**< void()> **postRender**

  Post-Render.
- static **EventT**< void()> **preRender**

  Pre-render.
- static **EventT**< void()> **render**

  Render.
- static **EventT**< void(std::string, std::string)> **setSelectedEntity**

  An entity has been selected.
- static **EventT**< void()> **step**

  Step the simulation once signal.
- static **EventT**< void()> **stop**

  Simulation stop signal.
- static **EventT**< void(std::string)> **worldCreated**

  A *(*p. *111) world has been created.*
- static **EventT**< void(const **common::UpdateInfo** &)> **worldUpdateBegin**

  World update has started.
- static **EventT**< void()> **worldUpdateEnd**

  World update has ended.
- static **EventT**< void()> **worldUpdateStart**

  World update has started.

10.45.1 Detailed Description

An **Event** (p. 292) class to get notifications for simulator events.

10.45.2 Member Function Documentation

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

Connect a boost::slot the delete entity.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

Connect a boost::slot the the pause signal.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

Render start signal.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

Connect a boost::slot the render update signal.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

10.45.2.10 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSetSelectedEntity (T *_subscriber*) [inline], [static]

Connect a boost::slot the set selected entity.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

10.45.2.11 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStep (T *_subscriber*) [inline], [static]

Connect a boost::slot the the step signal.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

10.45.2.12 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStop (T _subscriber) [inline], [static]

Connect a boost::slot the the stop signal.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

10.45.2.14 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectWorldUpdateBegin (T _subscriber) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

10.45.2.17 static void gazebo::event::Events::DisconnectAddEntity (ConnectionPtr *_subscriber*) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

10.45.2.18 static void gazebo::event::Events::DisconnectCreateEntity (ConnectionPtr *_subscriber*) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

10.45.2.19 static void gazebo::event::Events::DisconnectDeleteEntity (ConnectionPtr *_subscriber*) [inline], [static]

Disconnect a boost::slot the delete entity.

Parameters

in	*_subscriber*	the subscriber to this event

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

10.45.2.20 static void gazebo::event::Events::DisconnectDiagTimerStart (ConnectionPtr *_subscriber*) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

10.45.2.21 static void gazebo::event::Events::DisconnectDiagTimerStop (ConnectionPtr *_subscriber*) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

10.45.2.22 static void gazebo::event::Events::DisconnectPause (ConnectionPtr *_subscriber*) [inline], [static]

Disconnect a boost::slot the the pause signal.

Parameters

in	*_subscriber*	the subscriber to this event

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

10.45.2.23 static void gazebo::event::Events::DisconnectPostRender (ConnectionPtr *_subscriber*) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

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

Disconnect a render start signal.

Parameters

in	*_subscriber*	the subscriber to this event

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

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

Disconnect a boost::slot the render update signal.

Parameters

in	*_subscriber*	the subscriber to this event

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

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

Disconnect a boost::slot the set selected entity.

Parameters

in	*_subscriber*	the subscriber to this event

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

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

Disconnect a boost::slot the the step signal.

Parameters

in	*_subscriber*	the subscriber to this event

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

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

Disconnect a boost::slot the the stop signal.

Parameters

in	*_subscriber*	the subscriber to this event

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

10.45.2.29 static void gazebo::event::Events::DisconnectWorldCreated (ConnectionPtr _subscriber) [inline], [static]

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

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

10.45.2.30 static void gazebo::event::Events::DisconnectWorldUpdateBegin (ConnectionPtr _subscriber) [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

10.45.2.31 static void gazebo::event::Events::DisconnectWorldUpdateEnd (ConnectionPtr _subscriber) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

10.45.2.32 static void gazebo::event::Events::DisconnectWorldUpdateStart (ConnectionPtr _subscriber) [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

10.45.3 Member Data Documentation

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

An entity has been added.

Referenced by ConnectAddEntity(), and DisconnectAddEntity().

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

An entity has been deleted.

Referenced by ConnectDeleteEntity(), and DisconnectDeleteEntity().

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

Diagnostic timer start.

Referenced by ConnectDiagTimerStart(), and DisconnectDiagTimerStart().

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

Diagnostic timer stop.

Referenced by ConnectDiagTimerStop(), and DisconnectDiagTimerStop().

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

An entity has been created.

Referenced by ConnectCreateEntity(), and DisconnectCreateEntity().

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

Pause signal.

Referenced by ConnectPause(), and DisconnectPause().

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

Post-Render.

Referenced by ConnectPostRender(), and DisconnectPostRender().

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

Pre-render.

Referenced by ConnectPreRender(), and DisconnectPreRender().

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

Render.

Referenced by ConnectRender(), and DisconnectRender().

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

An entity has been selected.

Referenced by ConnectSetSelectedEntity(), and DisconnectSetSelectedEntity().

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

Step the simulation once signal.

Referenced by ConnectStep(), and DisconnectStep().

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

Simulation stop signal.

Referenced by ConnectStop(), and DisconnectStop().

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

A (p. 111) world has been created.

Referenced by ConnectWorldCreated(), and DisconnectWorldCreated().

10.45.3.14 EventT<void (const common::UpdateInfo &)> gazebo::event::Events::worldUpdateBegin [static]

World update has started.

Referenced by ConnectWorldUpdateBegin().

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

World update has ended.

Referenced by ConnectWorldUpdateEnd(), and DisconnectWorldUpdateEnd().

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

World update has started.

Referenced by ConnectWorldUpdateStart().

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

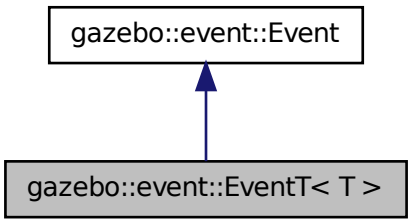
- **Events.hh**

10.46 gazebo::event::EventT< T > Class Template Reference

A (p. 111) class for event processing.

`#include <common/common.hh>`

Inheritance diagram for gazebo::event::EventT< T >:



Public Member Functions

- virtual ~**EventT** ()

  Destructor.
- **ConnectionPtr Connect** (const boost::function< T > &_subscriber)

  Connect a callback to this event.
- unsigned int **ConnectionCount** () const

  Get the number of connections.
- virtual void **Disconnect** (**ConnectionPtr** _c)

  Disconnect a callback to this event.
- virtual void **Disconnect** (int _id)

  Disconnect a callback to this event.
- void **operator()** ()

  Access the signal.
- template<typename P >
  void **operator()** (const P &_p)

  Signal the event with one parameter.
- template<typename P1 , typename P2 >
  void **operator()** (const P1 &_p1, const P2 &_p2)

  Signal the event with two parameters.
- template<typename P1 , typename P2 , typename P3 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3)

  Signal the event with three parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4)

  Signal the event with four parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5)

  Signal the event with five parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6)

  Signal the event with six parameters.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7)

  Signal the event with seven parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8)

  Signal the event with eight parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)

  Signal the event with nine parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)

  Signal the event with ten parameters.
- void **Signal** ()

  Signal the event for all subscribers.
- template<typename P >
  void **Signal** (const P &_p)

  Signal the event with one parameter.
- template<typename P1 , typename P2 >
  void **Signal** (const P1 &_p1, const P2 &_p2)

  Signal the event with two parameter.
- template<typename P1 , typename P2 , typename P3 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3)

  Signal the event with three parameter.
- template<typename P1 , typename P2 , typename P3 , typename P4 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4)

  Signal the event with four parameter.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5)

  Signal the event with five parameter.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6)

  Signal the event with six parameter.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7)

  Signal the event with seven parameter.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8)

  Signal the event with eight parameter.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)

Signal the event with nine parameter.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)

  Signal the event with ten parameter.

10.46.1 Detailed Description

template<typename T>class gazebo::event::EventT< T >

A (p. 111) class for event processing.

10.46.2 Member Function Documentation

10.46.2.1 template<typename T> void gazebo::event::EventT< T >::operator() ( ) [inline]

Access the signal.

10.46.2.2 template<typename T> template<typename P > void gazebo::event::EventT< T >::operator() (const P & *_p*) [inline]

Signal the event with one parameter.

Parameters

in	*_p*	the parameter

10.46.2.3 template<typename T> template<typename P1 , typename P2 > void gazebo::event::EventT< T >::operator() (const P1 & *_p1,* const P2 & *_p2*) [inline]

Signal the event with two parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter

10.46.2.4 template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::operator() (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3*) [inline]

Signal the event with three parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter

10.46.2.5 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4) [inline]

Signal the event with four parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter

10.46.2.6 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5) [inline]

Signal the event with five parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fift parameter

10.46.2.7 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6) [inline]

Signal the event with six parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fift parameter
in	*_p6*	the sixt parameter

10.46.2.8 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7) [inline]

Signal the event with seven parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fifth parameter
in	*_p6*	the sixth parameter
in	*_p7*	the seventh parameter

10.46.2.9 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 > void gazebo::event::EventT< T >::operator() (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6,* const P7 & *_p7,* const P8 & *_p8*) [inline]

Signal the event with eight parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fifth parameter
in	*_p6*	the sixth parameter
in	*_p7*	the seventh parameter
in	*_p8*	the eighth parameter

10.46.2.10 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 > void gazebo::event::EventT< T >::operator() (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6,* const P7 & *_p7,* const P8 & *_p8,* const P9 & *_p9*) [inline]

Signal the event with nine parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fifth parameter
in	*_p6*	the sixth parameter
in	*_p7*	the seventh parameter
in	*_p8*	the eighth parameter
in	*_p9*	the ninth parameter

10.46.2.11 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9, const P10 & _p10) [inline]

Signal the event with ten parameters.

Parameters

in	_p1	the first parameter
in	_p2	the second parameter
in	_p3	the second parameter
in	_p4	the first parameter
in	_p5	the fifth parameter
in	_p6	the sixth parameter
in	_p7	the seventh parameter
in	_p8	the eighth parameter
in	_p9	the ninth parameter
in	_p10	the tenth parameter

10.46.2.12 template<typename T> void gazebo::event::EventT< T >::Signal ( ) [inline]

Signal the event for all subscribers.

Referenced by gazebo::event::EventT< void(bool)>::operator()().

10.46.2.13 template<typename T> template<typename P > void gazebo::event::EventT< T >::Signal (const P & _p) [inline]

Signal the event with one parameter.

Parameters

in	_p	parameter

10.46.2.14 template<typename T> template<typename P1 , typename P2 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2) [inline]

Signal the event with two parameter.

Parameters

in	_p1	the first parameter
in	_p2	the second parameter

10.46.2.15 template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3) [inline]

Signal the event with three parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter

10.46.2.16 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 > void gazebo::event::EventT< T >::Signal (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4*) [inline]

Signal the event with four parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter

10.46.2.17 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 > void gazebo::event::EventT< T >::Signal (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5*) [inline]

Signal the event with five parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fifth parameter

10.46.2.18 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 > void gazebo::event::EventT< T >::Signal (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6*) [inline]

Signal the event with six parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fifth parameter
in	*_p6*	the sixth parameter

10.46.2.19 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7) [inline]

Signal the event with seven parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fifth parameter
in	*_p6*	the sixth parameter
in	*_p7*	the seventh parameter

10.46.2.20 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8) [inline]

Signal the event with eight parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fifth parameter
in	*_p6*	the sixth parameter
in	*_p7*	the seventh parameter
in	*_p8*	the eighth parameter

10.46.2.21 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9) [inline]

Signal the event with nine parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fifth parameter
in	*_p6*	the sixth parameter
in	*_p7*	the seventh parameter
in	*_p8*	the eighth parameter
in	*_p9*	the ninth parameter

10.46.2.22 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 > void gazebo::event::EventT< T >::Signal (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6,* const P7 & *_p7,* const P8 & *_p8,* const P9 & *_p9,* const P10 & *_p10*) [inline]

Signal the event with ten parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fifth parameter
in	*_p6*	the sixth parameter
in	*_p7*	the seventh parameter
in	*_p8*	the eighth parameter
in	*_p9*	the ninth parameter
in	*_p10*	the tenth parameter

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

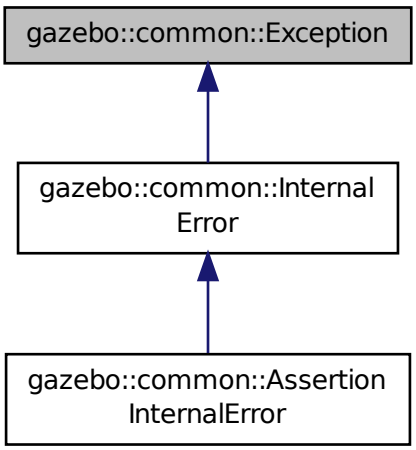
- **Event.hh**

10.47 gazebo::common::Exception Class Reference

Class for generating exceptions.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::Exception:



Public Member Functions

- **Exception** ()

  Constructor.
- **Exception** (const char *_file, int _line, std::string _msg)

  Default constructor.
- virtual ~**Exception** ()

  Destructor.
- std::string **GetErrorFile** () const

  Return the error function.
- std::string **GetErrorStr** () const

  Return the error string.
- void **Print** () const

  Print the exception to std out.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Exception** &_err)

  stream insertion operator for Gazebo Error

10.47.1 Detailed Description

Class for generating exceptions.

10.47.2 Constructor & Destructor Documentation

10.47.2.1 gazebo::common::Exception::Exception ( )

Constructor.

10.47.2.2 gazebo::common::Exception::Exception (const char * *_file*, int *_line*, std::string *_msg*)

Default constructor.

Parameters

in	*_file*	File name
in	*_line*	Line number where the error occurred
in	*_msg*	Error message

10.47.2.3 virtual gazebo::common::Exception::~Exception ( ) [virtual]

Destructor.

10.47.3 Member Function Documentation

10.47.3.1 std::string gazebo::common::Exception::GetErrorFile ( ) const

Return the error function.

Returns

The error function name

10.47.3.2 std::string gazebo::common::Exception::GetErrorStr ( ) const

Return the error string.

Returns

The error string

10.47.3.3 void gazebo::common::Exception::Print ( ) const

Print the exception to std out.

10.47.4 Friends And Related Function Documentation

10.47.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::common::Exception & *_err*) [friend]

stream insertion operator for Gazebo Error

Parameters

in	*_out*	the output stream
in	*_err*	the exception

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

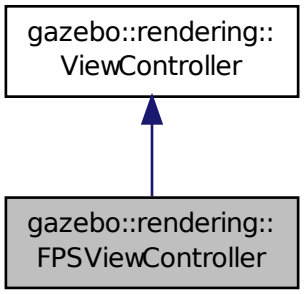
- **Exception.hh**

10.48 gazebo::rendering::FPSViewController Class Reference

First Person Shooter style view controller.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::FPSViewController:



Public Member Functions

- **FPSViewController** (**UserCameraPtr** _camera)

  Constructor.
- virtual ~**FPSViewController** ()

  Destructor.
- void **HandleKeyPressEvent** (const std::string &_key)

  Handle a key press event.
- void **HandleKeyReleaseEvent** (const std::string &_key)

  Handle a key release event.
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)

  Handle a mouse event.
- virtual void **Init** ()

  Initialize the controller.
- virtual void **Update** ()

  Update the camera position.

Static Public Member Functions

- static std::string **GetTypeString** ()

  Get the type name of this view controller.

Additional Inherited Members

10.48.1 Detailed Description

First Person Shooter style view controller.

10.48.2 Constructor & Destructor Documentation

10.48.2.1 gazebo::rendering::FPSViewController::FPSViewController (UserCameraPtr _camera)

Constructor.

Parameters

in	***Camera*** *(p. 162)*	to controll

10.48.2.2 virtual gazebo::rendering::FPSViewController::~FPSViewController ( ) [virtual]

Destructor.

10.48.3 Member Function Documentation

10.48.3.1 static std::string gazebo::rendering::FPSViewController::GetTypeString ( ) [static]

Get the type name of this view controller.

Returns

The name of the controller type: "fps"

10.48.3.2 void gazebo::rendering::FPSViewController::HandleKeyPressEvent (const std::string & _key) [virtual]

Handle a key press event.

Parameters

in	*_key*	The key that was pressed.

Implements **gazebo::rendering::ViewController** (p. 883).

10.48.3.3 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. 883).

10.48.3.4 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. 883).

10.48.3.5 virtual void gazebo::rendering::FPSViewController::Init ( ) [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 884).

10.48.3.6 virtual void gazebo::rendering::FPSViewController::Update ( ) [virtual]

Update the camera position.

Implements **gazebo::rendering::ViewController** (p. 884).

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

- **FPSViewController.hh**

10.49 urdf2gazebo::GazeboExtension Class Reference

`#include <parser_urdf.hh>`

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

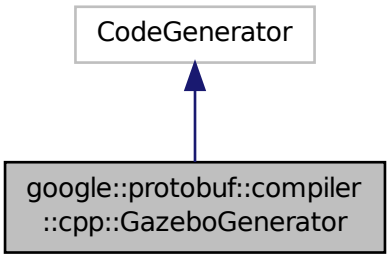
- **parser_urdf.hh**

10.50 google::protobuf::compiler::cpp::GazeboGenerator Class Reference

Google protobuf message generator for **gazebo::msgs** (p. 89).

`#include <GazeboGenerator.hh>`

Inheritance diagram for google::protobuf::compiler::cpp::GazeboGenerator:



Public Member Functions

- **GazeboGenerator** (const std::string &_name)
- virtual ~**GazeboGenerator** ()
- virtual bool **Generate** (const FileDescriptor *file, const string &parameter, OutputDirectory *directory, string *error) const

10.50.1 Detailed Description

Google protobuf message generator for **gazebo::msgs** (p. 89).

10.50.2 Constructor & Destructor Documentation

10.50.2.1 google::protobuf::compiler::cpp::GazeboGenerator::GazeboGenerator (const std::string & *_name*)

10.50.2.2 virtual google::protobuf::compiler::cpp::GazeboGenerator::~GazeboGenerator ( ) [virtual]

10.50.3 Member Function Documentation

10.50.3.1 virtual bool google::protobuf::compiler::cpp::GazeboGenerator::Generate (const FileDescriptor * *file,* const string & *parameter,* OutputDirectory * *directory,* string * *error*) const [virtual]

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

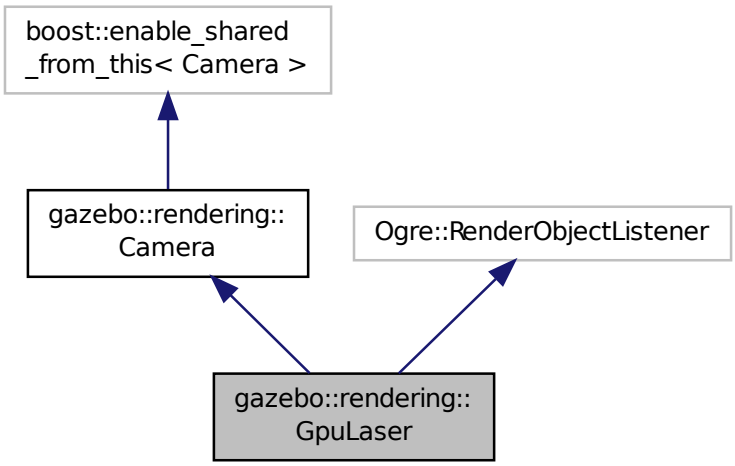
- **GazeboGenerator.hh**

10.51 gazebo::rendering::GpuLaser Class Reference

GPU based laser distance sensor.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::GpuLaser:



Public Member Functions

- **GpuLaser** (const std::string &_namePrefix, **Scene** *_scene, bool _autoRender=true)

  Constructor.
- virtual ~**GpuLaser** ()

  Destructor.
- template<typename T >
  event::ConnectionPtr ConnectNewLaserFrame (T _subscriber)

  Connect to a laser frame signal.
- void **CreateLaserTexture** (const std::string &_textureName)

  Create the texture which is used to render laser data.
- void **DisconnectNewLaserFrame** (**event::ConnectionPtr** &_c)

  Disconnect from a laser frame signal.
- virtual void **Fini** ()

  Finalize the camera.
- const float * **GetLaserData** ()

  All things needed to get back z buffer for laser data.
- virtual void **Init** ()

  Initialize the camera.
- virtual void **Load** (**sdf::ElementPtr** &_sdf)
- virtual void **Load** ()

  Load the camera with default parmeters.
- virtual void **notifyRenderSingleObject** (Ogre::Renderable *_rend, const Ogre::Pass *_p, const Ogre::AutoParamDataSource *_s, const Ogre::LightList *_ll, bool _supp)
- virtual void **PostRender** ()

Post render.

- void **SetParentSensor** (**sensors::GpuRaySensor** *_parent)

  Set the parent sensor.

- void **SetRangeCount** (unsigned int _w, unsigned int _h=1)

  Set the number of laser samples in the width and height.

Additional Inherited Members

10.51.1 Detailed Description

GPU based laser distance sensor.

10.51.2 Constructor & Destructor Documentation

10.51.2.1 gazebo::rendering::GpuLaser::GpuLaser (const std::string & *_namePrefix,* Scene * *_scene,* bool *_autoRender* = `true`)

Constructor.

Parameters

in	*_namePrefix*	Unique prefix name for the camera.
in	*_scene*	**Scene** (p. 676) that will contain the camera
in	*_autoRender*	Almost everyone should leave this as true.

10.51.2.2 virtual gazebo::rendering::GpuLaser::~GpuLaser ( ) `[virtual]`

Destructor.

10.51.3 Member Function Documentation

10.51.3.1 template<typename T > event::ConnectionPtr gazebo::rendering::GpuLaser::ConnectNewLaserFrame (T *_subscriber*) `[inline]`

Connect to a laser frame signal.

Parameters

in	*_subscriber*	Callback that is called when a new image is generated

Returns

A (p. 111) pointer to the connection. This must be kept in scope.

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

10.51.3.2 void gazebo::rendering::GpuLaser::CreateLaserTexture (const std::string & *_textureName*)

Create the texture which is used to render laser data.

Parameters

in	*_textureName*	Name of the new texture.

10.51.3.3 void gazebo::rendering::GpuLaser::DisconnectNewLaserFrame (event::ConnectionPtr & *_c*) [inline]

Disconnect from a laser frame signal.

Parameters

in	*_c*	The connection to disconnect

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

10.51.3.4 virtual void gazebo::rendering::GpuLaser::Fini ( ) [virtual]

Finalize the camera.

This function is called before the camera is destructed

Reimplemented from **gazebo::rendering::Camera** (p. 171).

10.51.3.5 const float* gazebo::rendering::GpuLaser::GetLaserData ( )

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

Returns

Array of laser data.

10.51.3.6 virtual void gazebo::rendering::GpuLaser::Init ( ) [virtual]

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 178).

10.51.3.7 virtual void gazebo::rendering::GpuLaser::Load (sdf::ElementPtr & *_sdf*) [virtual]

10.51.3.8 virtual void gazebo::rendering::GpuLaser::Load ( ) [virtual]

Load the camera with default parmeters.

Reimplemented from **gazebo::rendering::Camera** (p. 179).

10.51.3.9 virtual void gazebo::rendering::GpuLaser::notifyRenderSingleObject (Ogre::Renderable * *_rend*, const Ogre::Pass * *_p*, const Ogre::AutoParamDataSource * *_s*, const Ogre::LightList * *_ll*, bool *_supp*) [virtual]

10.51.3.10 virtual void gazebo::rendering::GpuLaser::PostRender ( ) [virtual]

Post render.

Called afer the render signal.

Reimplemented from **gazebo::rendering::Camera** (p. 180).

10.51.3.11 void gazebo::rendering::GpuLaser::SetParentSensor (sensors::GpuRaySensor * _parent)

Set the parent sensor.

Parameters

in	_parent	Pointer to a **sensors::GpuRaySensor** (p. 328)

10.51.3.12 void gazebo::rendering::GpuLaser::SetRangeCount (unsigned int _w, unsigned int _h = 1)

Set the number of laser samples in the width and height.

Parameters

in	_w	Number of samples in the horizontal sweep
in	_h	Number of samples in the vertical sweep

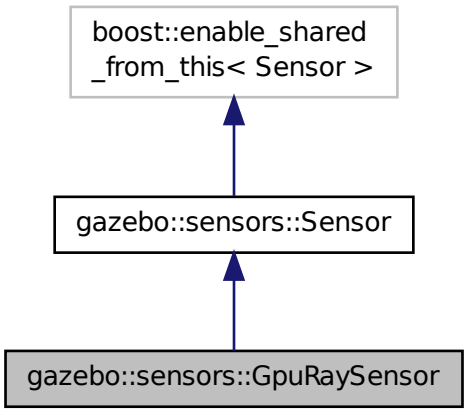
The documentation for this class was generated from the following file:

- **GpuLaser.hh**

10.52 gazebo::sensors::GpuRaySensor Class Reference

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::GpuRaySensor:



Public Member Functions

- **GpuRaySensor** ()

Constructor.

- virtual ~**GpuRaySensor** ()

  Destructor.

- **event::ConnectionPtr ConnectNewLaserFrame** (boost::function< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)> _subscriber)

  Connect to the new laser frame event.

- void **DisconnectNewLaserFrame** (**event::ConnectionPtr** &_conn)

  Disconnect Laser Frame.

- **math::Angle GetAngleMax** () const

  Get the maximum angle.

- **math::Angle GetAngleMin** () const

  Get the minimum angle.

- double **GetAngleResolution** () const

  Get radians between each range.

- unsigned int **GetCameraCount** () const

  Gets the camera count.

- double **GetCosHorzFOV** () const

  Get Cos Horz field-of-view.

- double **GetCosVertFOV** () const

  Get Cos Vert field-of-view.

- int **GetFiducial** (int _index) const

  Get detected fiducial value for a ray.

- double **GetHorzFOV** () const

  Get the horizontal field of view of the laser sensor.

- double **GetHorzHalfAngle** () const

  *Get (horizontal_max_angle + horizontal_min_angle) * 0.5.*

- **rendering::GpuLaserPtr GetLaserCamera** () const

  *Returns a pointer to the internally kept **rendering::GpuLaser** (p. 324).*

- double **GetRange** (int _index)

  Get detected range for a ray.

- int **GetRangeCount** () const

  Get the range count.

- double **GetRangeCountRatio** () const

  Return the ratio of horizontal range count to vertical range count.

- double **GetRangeMax** () const

  Get the maximum range.

- double **GetRangeMin** () const

  Get the minimum range.

- double **GetRangeResolution** () const

  Get the range resolution If RangeResolution is 1, the number of simulated rays is equal to the number of returned range readings.

- void **GetRanges** (std::vector< double > &_ranges) const

  Get all the ranges.

- int **GetRayCount** () const

  Get the ray count.

- double **GetRayCountRatio** () const

  Return the ratio of horizontal ray count to vertical ray count.

- double **GetRetro** (int _index) const

  Get detected retro (intensity) value for a ray.
- double **GetVertFOV** () const

  Get the vertical field-of-view.
- double **GetVertHalfAngle** () const

  Get (vertical_max_angle + vertical_min_angle) ∗ 0.5.
- **math::Angle GetVerticalAngleMax** () const

  Get the vertical scan line top angle.
- **math::Angle GetVerticalAngleMin** () const

  Get the vertical scan bottom angle.
- int **GetVerticalRangeCount** () const

  Get the vertical scan line count.
- int **GetVerticalRayCount** () const

  Get the vertical scan line count.
- virtual void **Init** ()

  Initialize the ray.
- bool **IsHorizontal** () const

  Gets if sensor is horizontal.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** &_sdf)

  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.
- void **SetAngleMax** (double _angle)

  Set the scan maximum angle.
- void **SetAngleMin** (double _angle)

  Set the scan minimum angle.
- void **SetVerticalAngleMax** (double _angle)

  Set the vertical scan line top angle.
- void **SetVerticalAngleMin** (double _angle)

  Set the vertical scan bottom angle.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the ray.
- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.

Protected Attributes

- unsigned int **cameraCount**

  Number of cameras.
- **sdf::ElementPtr cameraElem**

  Camera SDF element.
- double **chfov**

  Cos horizontal field-of-view.

- double **cvfov**

  Cos vertical field-of-view.
- double **far**

  Far clip plane.
- double **hfov**

  Horizontal field-of-view.
- **sdf::ElementPtr horzElem**

  Horizontal SDF element.
- double **horzHalfAngle**

  Horizontal half angle.
- unsigned int **horzRangeCount**

  Horizontal range count.
- unsigned int **horzRayCount**

  Horizontal ray count.
- bool **isHorizontal**

  True if the sensor is horizontal only.
- double **near**

  Near clip plane.
- double **rangeCountRatio**

  Range count ratio.
- **sdf::ElementPtr rangeElem**

  Range SDF element.
- double **rayCountRatio**

  Ray count ratio.
- **sdf::ElementPtr scanElem**

  Scan SDF elementz.
- **sdf::ElementPtr vertElem**

  Vertical SDF element.
- double **vertHalfAngle**

  Vertical half angle.
- unsigned int **vertRangeCount**

  Vertical range count.
- unsigned int **vertRayCount**

  Vertical ray count.
- double **vfov**

  Vertical field-of-view.

10.52.1 Constructor & Destructor Documentation

10.52.1.1 gazebo::sensors::GpuRaySensor::GpuRaySensor ( )

Constructor.

10.52.1.2 virtual gazebo::sensors::GpuRaySensor::~GpuRaySensor ( ) [virtual]

Destructor.

10.52.2 Member Function Documentation

10.52.2.1 event::ConnectionPtr gazebo::sensors::GpuRaySensor::ConnectNewLaserFrame (boost::function< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)> _subscriber)

Connect to the new laser frame event.

Parameters

in	*_subscriber*	Event callback.

10.52.2.2 void gazebo::sensors::GpuRaySensor::DisconnectNewLaserFrame (event::ConnectionPtr & _conn)

Disconnect Laser Frame.

Parameters

in,out	*_conn*	Connection pointer to disconnect.

10.52.2.3 virtual void gazebo::sensors::GpuRaySensor::Fini ( ) [protected], [virtual]

Finalize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 702).

10.52.2.4 math::Angle gazebo::sensors::GpuRaySensor::GetAngleMax ( ) const

Get the maximum angle.

Returns

the maximum angle

10.52.2.5 math::Angle gazebo::sensors::GpuRaySensor::GetAngleMin ( ) const

Get the minimum angle.

Returns

The minimum angle

10.52.2.6 double gazebo::sensors::GpuRaySensor::GetAngleResolution ( ) const

Get radians between each range.

10.52.2.7 unsigned int gazebo::sensors::GpuRaySensor::GetCameraCount ( ) const

Gets the camera count.

Returns

Number of cameras

10.52.2.8 double gazebo::sensors::GpuRaySensor::GetCosHorzFOV ( ) const

Get Cos Horz field-of-view.

Returns

2 ∗ atan(tan(this->hfov/2) / cos(this->vfov/2))

10.52.2.9 double gazebo::sensors::GpuRaySensor::GetCosVertFOV ( ) const

Get Cos Vert field-of-view.

Returns

2 ∗ atan(tan(this->vfov/2) / cos(this->hfov/2))

10.52.2.10 int gazebo::sensors::GpuRaySensor::GetFiducial (int *_index*) const

Get detected fiducial value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index of specific ray

Returns

Fiducial value of ray

10.52.2.11 double gazebo::sensors::GpuRaySensor::GetHorzFOV ( ) const

Get the horizontal field of view of the laser sensor.

Returns

The horizontal field of view of the laser sensor.

10.52.2.12 double gazebo::sensors::GpuRaySensor::GetHorzHalfAngle ( ) const

Get (horizontal_max_angle + horizontal_min_angle) ∗ 0.5.

Returns

(horizontal_max_angle + horizontal_min_angle) ∗ 0.5

10.52.2.13 rendering::GpuLaserPtr gazebo::sensors::GpuRaySensor::GetLaserCamera ( ) const [inline]

Returns a pointer to the internally kept **rendering::GpuLaser** (p. 324).

Returns

Pointer to GpuLaser

10.52.2.14 double gazebo::sensors::GpuRaySensor::GetRange (int *_index*)

Get detected range for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index of specific ray

Returns

Returns DBL_MAX for no detection.

10.52.2.15 int gazebo::sensors::GpuRaySensor::GetRangeCount ( ) const

Get the range count.

Returns

The number of ranges

10.52.2.16 double gazebo::sensors::GpuRaySensor::GetRangeCountRatio ( ) const

Return the ratio of horizontal range count to vertical range count.

A (p. 111) ray count is the number of simulated rays. Whereas a range count is the total number of data points returned. When range count != ray count, then values are interpolated between rays.

10.52.2.17 double gazebo::sensors::GpuRaySensor::GetRangeMax ( ) const

Get the maximum range.

Returns

The maximum range

10.52.2.18 double gazebo::sensors::GpuRaySensor::GetRangeMin ( ) const

Get the minimum range.

Returns

The minimum range

10.52.2.19 double gazebo::sensors::GpuRaySensor::GetRangeResolution ( ) const

Get the range resolution If RangeResolution is 1, the number of simulated rays is equal to the number of returned range readings.

If it's less than 1, fewer simulated rays than actual returned range readings are used, the results are interpolated from two nearest neighbors, and vice versa.

Returns

The Range Resolution

10.52.2.20 void gazebo::sensors::GpuRaySensor::GetRanges (std::vector< double > & *_ranges*) const

Get all the ranges.

Parameters

out	*_range*	**A** (p. 111) vector that will contain all the range data

10.52.2.21 int gazebo::sensors::GpuRaySensor::GetRayCount ( ) const

Get the ray count.

Returns

The number of rays

10.52.2.22 double gazebo::sensors::GpuRaySensor::GetRayCountRatio ( ) const

Return the ratio of horizontal ray count to vertical ray count.

A (p. 111) ray count is the number of simulated rays. Whereas a range count is the total number of data points returned. When range count != ray count, then values are interpolated between rays.

10.52.2.23 double gazebo::sensors::GpuRaySensor::GetRetro (int *_index*) const

Get detected retro (intensity) value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index of specific ray

Returns

Intensity value of ray

10.52.2.24 double gazebo::sensors::GpuRaySensor::GetVertFOV ( ) const

Get the vertical field-of-view.

10.52.2.25 double gazebo::sensors::GpuRaySensor::GetVertHalfAngle ( ) const

Get (vertical_max_angle + vertical_min_angle) $*$ 0.5.

Returns

(vertical_max_angle + vertical_min_angle) $*$ 0.5

10.52.2.26 math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMax ( ) const

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.52.2.27 math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMin ( ) const

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.52.2.28 int gazebo::sensors::GpuRaySensor::GetVerticalRangeCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.52.2.29 int gazebo::sensors::GpuRaySensor::GetVerticalRayCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.52.2.30 virtual void gazebo::sensors::GpuRaySensor::Init ( ) [virtual]

Initialize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.52.2.31 bool gazebo::sensors::GpuRaySensor::IsHorizontal ( ) const

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

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

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 698) parameters
in	*_worldName*	Name of world to load from

10.52.2.33 virtual void gazebo::sensors::GpuRaySensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

10.52.2.34 void gazebo::sensors::GpuRaySensor::SetAngleMax (double *_angle*)

Set the scan maximum angle.

Parameters

in	*_angle*	The maximum angle

10.52.2.35 void gazebo::sensors::GpuRaySensor::SetAngleMin (double *_angle*)

Set the scan minimum angle.

Parameters

in	*_angle*	The minimum angle

10.52.2.36 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMax (double *_angle*)

Set the vertical scan line top angle.

Parameters

in	*_angle*	The Maximum angle of the scan block

10.52.2.37 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMin (double *_angle*)

Set the vertical scan bottom angle.

Parameters

in	*_angle*	The minimum angle of the scan block

10.52.2.38 virtual void gazebo::sensors::GpuRaySensor::UpdateImpl (bool *_force*) [protected], [virtual]

Update the sensor information.

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 706).

10.52.3 Member Data Documentation

10.52.3.1 unsigned int gazebo::sensors::GpuRaySensor::cameraCount [protected]

Number of cameras.

10.52.3.2 sdf::ElementPtr gazebo::sensors::GpuRaySensor::cameraElem [protected]

Camera SDF element.

10.52.3.3 double gazebo::sensors::GpuRaySensor::chfov [protected]

Cos horizontal field-of-view.

10.52.3.4 double gazebo::sensors::GpuRaySensor::cvfov [protected]

Cos vertical field-of-view.

10.52.3.5 double gazebo::sensors::GpuRaySensor::far [protected]

Far clip plane.

10.52.3.6 double gazebo::sensors::GpuRaySensor::hfov [protected]

Horizontal field-of-view.

10.52.3.7 sdf::ElementPtr gazebo::sensors::GpuRaySensor::horzElem [protected]

Horizontal SDF element.

10.52.3.8 double gazebo::sensors::GpuRaySensor::horzHalfAngle [protected]

Horizontal half angle.

10.52.3.9 unsigned int gazebo::sensors::GpuRaySensor::horzRangeCount [protected]

Horizontal range count.

10.52.3.10 unsigned int gazebo::sensors::GpuRaySensor::horzRayCount [protected]

Horizontal ray count.

10.52.3.11 bool gazebo::sensors::GpuRaySensor::isHorizontal [protected]

True if the sensor is horizontal only.

10.52.3.12 double gazebo::sensors::GpuRaySensor::near [protected]

Near clip plane.

10.52.3.13 double gazebo::sensors::GpuRaySensor::rangeCountRatio [protected]

Range count ratio.

10.52.3.14 sdf::ElementPtr gazebo::sensors::GpuRaySensor::rangeElem [protected]

Range SDF element.

10.52.3.15 double gazebo::sensors::GpuRaySensor::rayCountRatio [protected]

Ray count ratio.

10.52.3.16 sdf::ElementPtr gazebo::sensors::GpuRaySensor::scanElem [protected]

Scan SDF elementz.

10.52.3.17 sdf::ElementPtr gazebo::sensors::GpuRaySensor::vertElem [protected]

Vertical SDF element.

10.52.3.18 double gazebo::sensors::GpuRaySensor::vertHalfAngle [protected]

Vertical half angle.

10.52.3.19 unsigned int gazebo::sensors::GpuRaySensor::vertRangeCount [protected]

Vertical range count.

10.52.3.20 unsigned int gazebo::sensors::GpuRaySensor::vertRayCount [protected]

Vertical ray count.

10.52.3.21 double gazebo::sensors::GpuRaySensor::vfov [protected]

Vertical field-of-view.

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

- **GpuRaySensor.hh**

10.53 gazebo::rendering::Grid Class Reference

Displays a grid of cells, drawn with lines.

`#include <rendering/rendering.hh>`

Public Member Functions

- **Grid** (**Scene** *_scene, uint32_t _cellCount, float _cellLength, float _lineWidth, const **common::Color** &_color)
  Constructor.
- ~**Grid** ()
  Destructor.
- void **Enable** (bool _enable)
  Enable or disable the grid.
- uint32_t **GetCellCount** () const

Get the number of cells.

- float **GetCellLength** () const

  Get the cell length.

- **common::Color GetColor** () const

  Return the grid color.

- uint32_t **GetHeight** () const

  Get the height of the grid.

- float **GetLineWidth** () const

  Get the width of the grid line.

- Ogre::SceneNode * **GetSceneNode** ()

  *Get the **Ogre** (p. 106) scene node associated with this grid.*

- void **Init** ()

  Initialize the grid.

- void **SetCellCount** (uint32_t _count)

  Set the number of cells.

- void **SetCellLength** (float _len)

  Set the cell length.

- void **SetColor** (const **common::Color** &_color)

  Sets the color of the grid.

- void **SetHeight** (uint32_t _count)

  Set the height of the grid.

- void **SetLineWidth** (float _width)

  Set the line width.

- void **SetUserData** (const Ogre::Any &_data)

  Sets user data on all ogre objects we own.

10.53.1 Detailed Description

Displays a grid of cells, drawn with lines.

Displays a grid of cells, drawn with lines. **A** (p. 111) grid with an identity orientation is drawn along the XY plane.

10.53.2 Constructor & Destructor Documentation

10.53.2.1 gazebo::rendering::Grid::Grid (Scene * *_scene*, uint32_t *_cellCount*, float *_cellLength*, float *_lineWidth*, const common::Color & *_color*)

Constructor.

Parameters

in	*_scene*	The scene this object is part of
in	*_cellCount*	The number of cells to draw
in	*_cellLength*	The size of each cell
in	*_lineWidth*	The width of the lines to use
in	*_color*	The color of the grid

10.53.2.2 gazebo::rendering::Grid::~Grid ( )

Destructor.

10.53.3 Member Function Documentation

10.53.3.1 void gazebo::rendering::Grid::Enable (bool *_enable*)

Enable or disable the grid.

Parameters

in	*_enable*	Set to true to view the grid, false to make invisible.

10.53.3.2 uint32_t gazebo::rendering::Grid::GetCellCount ( ) const [inline]

Get the number of cells.

10.53.3.3 float gazebo::rendering::Grid::GetCellLength ( ) const [inline]

Get the cell length.

Returns

The cell length

10.53.3.4 common::Color gazebo::rendering::Grid::GetColor ( ) const [inline]

Return the grid color.

Returns

The grid color

10.53.3.5 uint32_t gazebo::rendering::Grid::GetHeight ( ) const [inline]

Get the height of the grid.

Returns

The height

10.53.3.6 float gazebo::rendering::Grid::GetLineWidth ( ) const [inline]

Get the width of the grid line.

Returns

The line width

10.53.3.7 Ogre::SceneNode* gazebo::rendering::Grid::GetSceneNode ( ) [inline]

Get the **Ogre** (p. 106) scene node associated with this grid.

Returns

The **Ogre** (p. 106) scene node associated with this grid

10.53.3.8 void gazebo::rendering::Grid::Init ( )

Initialize the grid.

10.53.3.9 void gazebo::rendering::Grid::SetCellCount (uint32_t *_count*)

Set the number of cells.

Parameters

in	*_count*	The number of cells

10.53.3.10 void gazebo::rendering::Grid::SetCellLength (float *_len*)

Set the cell length.

Parameters

in	*_len*	The cell length

10.53.3.11 void gazebo::rendering::Grid::SetColor (const common::Color & *_color*)

Sets the color of the grid.

Parameters

in	*_color*	The grid color

10.53.3.12 void gazebo::rendering::Grid::SetHeight (uint32_t *_count*)

Set the height of the grid.

Parameters

in	*_count*	**Grid** (p. 340) height

10.53.3.13 void gazebo::rendering::Grid::SetLineWidth (float *_width*)

Set the line width.

Parameters

in	*_width*	The width of the grid

10.53.3.14 void gazebo::rendering::Grid::SetUserData (const Ogre::Any & *_data*)

Sets user data on all ogre objects we own.

Parameters

in	*_data*	The user data

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

- **Grid.hh**

10.54 gazebo::physics::Gripper Class Reference

A (p. 111) gripper abstraction.

`#include <physics/physics.hh>`

Public Member Functions

- **Gripper** (**ModelPtr** _model)
  Constructor.
- virtual ~**Gripper** ()
  Destructor.
- virtual void **Init** ()
  Initialize.
- virtual void **Load** (**sdf::ElementPtr** _sdf)
  Load the gripper.

10.54.1 Detailed Description

A (p. 111) gripper abstraction.

A (p. 111) gripper is a collection of links that act as a gripper. This class will intelligently generate fixed joints between the gripper and an object within the gripper. This allows the object to be manipulated without falling or behaving poorly.

10.54.2 Constructor & Destructor Documentation

10.54.2.1 gazebo::physics::Gripper::Gripper (**ModelPtr** *_model*) [explicit]

Constructor.

Parameters

in	*_model*	The model which contains the **Gripper** (p. 344).

10.54.2.2 virtual gazebo::physics::Gripper::~Gripper ( ) [virtual]

Destructor.

10.54.3 Member Function Documentation

10.54.3.1 virtual void gazebo::physics::Gripper::Init ( ) [virtual]

Initialize.

10.54.3.2 virtual void gazebo::physics::Gripper::Load (sdf::ElementPtr _sdf) [virtual]

Load the gripper.

Parameters

in	*_sdf*	Shared point to an sdf element that contains the list of links in the gripper.

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

- **Gripper.hh**

10.55 gazebo::rendering::GUIOverlay Class Reference

A (p. 111) 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. 162) to and overlay window.*
- bool **AttachCameraToImage** (**DepthCameraPtr** &_camera, const std::string &_windowName)

  *Use this function to draw the output from a **rendering::DepthCamera** (p. 253) to and overlay window.*
- template<typename T >
  void **ButtonCallback** (const std::string &_buttonName, void(T::*_fp)(), T *_obj)

  Register a CEGUI button callback.
- void **CreateWindow** (const std::string &_type, const std::string &_name, const std::string &_parent, const **math::Vector2d** &_position, const **math::Vector2d** &_size, const std::string &_text)

  Create a new window on the overlay.
- bool **HandleKeyPressEvent** (const std::string &_key)

  Handle a key press event.
- bool **HandleKeyReleaseEvent** (const std::string &_key)

  Handle a key release event.

- bool **HandleMouseEvent** (const **common::MouseEvent** &_evt)

  Handle a mouse event.
- void **Hide** ()

  Make the overlay invisible.
- void **Init** (Ogre::RenderTarget ∗_renderTarget)

  Initialize the overlay.
- bool **IsInitialized** ()

  Return true if the overlay has been initialized.
- void **LoadLayout** (const std::string &_filename)

  Load a CEGUI layout file.
- void **Resize** (unsigned int _width, unsigned int _height)

  Resize the window.
- void **Show** ()

  Make the overlay visible.
- void **Update** ()

  Update the overlay's objects.

10.55.1 Detailed Description

A (p. 111) class that creates a CEGUI overlay on a render window.

10.55.2 Constructor & Destructor Documentation

10.55.2.1 gazebo::rendering::GUIOverlay::GUIOverlay ( )

Constructor.

10.55.2.2 virtual gazebo::rendering::GUIOverlay::∼GUIOverlay ( ) [virtual]

Destructor.

10.55.3 Member Function Documentation

10.55.3.1 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (CameraPtr & *_camera*, const std::string & *_windowName*)

Use this function to draw the output from a **rendering::Camera** (p. 162) to and overlay window.

Parameters

in	*_camera*	Pointer to the camera.
in	*_windowName*	Name of the window to receive the camera image

Returns

True if successful

10.55.3.2 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (DepthCameraPtr & _camera, const std::string & _windowName)

Use this function to draw the output from a **rendering::DepthCamera** (p. 253) to and overlay window.

Parameters

in	*_camera*	Pointer to the camera.
in	*_windowName*	Name of the window to receive the camera image

Returns

True if successful

10.55.3.3 template<typename T > void gazebo::rendering::GUIOverlay::ButtonCallback (const std::string & _buttonName, void(T::*)() _fp, T * _obj) [inline]

Register a CEGUI button callback.

Assign a callback to a name button.

Parameters

in	*_buttonName*	Name of the button.
in	*_fp*	Function pointer to the callback.
in	*_obj*	Class pointer that contains _fp.

10.55.3.4 void gazebo::rendering::GUIOverlay::CreateWindow (const std::string & _type, const std::string & _name, const std::string & _parent, const math::Vector2d & _position, const math::Vector2d & _size, const std::string & _text)

Create a new window on the overlay.

Parameters

in	*_type*	The window type. This should match a CEGUI window type. See CEGUI::WindowManager::getSingleton().createWindow().
in	*_name*	Unique name for the window.
in	*_parent*	Name of the parent window.
in	*_position*	Position of the window within the parent.
in	*_size*	Size of the window.
in	*_text*	Display title of the window.

10.55.3.5 bool gazebo::rendering::GUIOverlay::HandleKeyPressEvent (const std::string & _key)

Handle a key press event.

Parameters

in	*_key*	The key pressed.

Returns

True if the key press event was handled.

10.55.3.6 bool gazebo::rendering::GUIOverlay::HandleKeyReleaseEvent (const std::string & *_key*)

Handle a key release event.

Parameters

in	*_key*	The key released.

Returns

True if the key release event was handled.

10.55.3.7 bool gazebo::rendering::GUIOverlay::HandleMouseEvent (const common::MouseEvent & *_evt*)

Handle a mouse event.

Parameters

in	*_evt*	The mouse event.

Returns

True if the mouse event was handled.

10.55.3.8 void gazebo::rendering::GUIOverlay::Hide ( )

Make the overlay invisible.

10.55.3.9 void gazebo::rendering::GUIOverlay::Init (Ogre::RenderTarget * *_renderTarget*)

Initialize the overlay.

Parameters

in	*_renderTarget*	The render target which will have the overlay.

10.55.3.10 bool gazebo::rendering::GUIOverlay::IsInitialized ( )

Return true if the overlay has been initialized.

Returns

True if initialized

10.55.3.11 void gazebo::rendering::GUIOverlay::LoadLayout (const std::string & *_filename*)

Load a CEGUI layout file.

Parameters

in	*_filename*	Name of the layout file.

10.55.3.12 void gazebo::rendering::GUIOverlay::Resize (unsigned int *_width*, unsigned int *_height*)

Resize the window.

10.55.3.13 void gazebo::rendering::GUIOverlay::Show ( )

Make the overlay visible.

10.55.3.14 void gazebo::rendering::GUIOverlay::Update ( )

Update the overlay's objects.

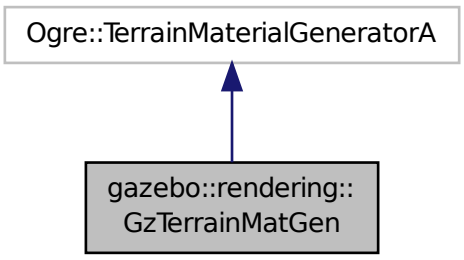
The documentation for this class was generated from the following file:

- **GUIOverlay.hh**

10.56 gazebo::rendering::GzTerrainMatGen Class Reference

`#include <Heightmap.hh>`

Inheritance diagram for gazebo::rendering::GzTerrainMatGen:



Classes

- class **SM2Profile**

  Shader model 2 profile target.

Public Member Functions

- **GzTerrainMatGen** ()

  Constructor.
- virtual ∼**GzTerrainMatGen** ()

  Destructor.

10.56.1 Constructor & Destructor Documentation

10.56.1.1 gazebo::rendering::GzTerrainMatGen::GzTerrainMatGen ( )

Constructor.

10.56.1.2 virtual gazebo::rendering::GzTerrainMatGen::∼GzTerrainMatGen ( ) [virtual]

Destructor.

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

- **Heightmap.hh**

10.57 gazebo::rendering::Heightmap Class Reference

Rendering a terrain using heightmap information.

`#include <rendering/rendering.hh>`

Public Member Functions

- **Heightmap** (**ScenePtr** _scene)

  Constructor.
- virtual ∼**Heightmap** ()

  Destructor.
- double **GetHeight** (double _x, double _y, double _z=1000)

  Get the height at a location.
- Ogre::TerrainGroup ∗ **GetOgreTerrain** () const

  Get a pointer to the OGRE terrain group object.
- void **Load** ()

  Load the heightmap.
- void **LoadFromMsg** (ConstVisualPtr &_msg)

  Load the heightmap from a visual message.

10.57.1 Detailed Description

Rendering a terrain using heightmap information.

10.57.2 Constructor & Destructor Documentation

10.57.2.1 gazebo::rendering::Heightmap::Heightmap (ScenePtr _scene)

Constructor.

Parameters

in	*_scene*	Pointer to the scene that will contain the heightmap

10.57.2.2 virtual gazebo::rendering::Heightmap::~Heightmap ( ) [virtual]

Destructor.

10.57.3 Member Function Documentation

10.57.3.1 double gazebo::rendering::Heightmap::GetHeight (double _x, double _y, double _z = 1000)

Get the height at a location.

Parameters

in	*_x*	X location
in	*_y*	Y location
in	*_z*	Z location

Returns

The height at the specified location

10.57.3.2 Ogre::TerrainGroup* gazebo::rendering::Heightmap::GetOgreTerrain ( ) const

Get a pointer to the OGRE terrain group object.

Returns

Pointer to the OGRE terrain.

10.57.3.3 void gazebo::rendering::Heightmap::Load ( )

Load the heightmap.

10.57.3.4 void gazebo::rendering::Heightmap::LoadFromMsg (ConstVisualPtr & _msg)

Load the heightmap from a visual message.

Parameters

in	*_msg*	The visual message containing heightmap info

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

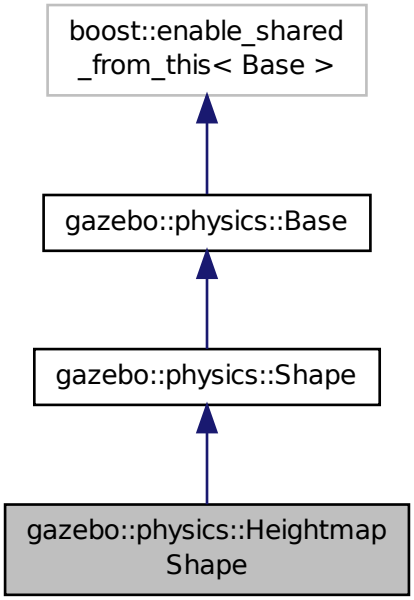
- **Heightmap.hh**

10.58 gazebo::physics::HeightmapShape Class Reference

HeightmapShape (p. 352) collision shape builds a heightmap from an image.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::HeightmapShape:



Public Member Functions

- **HeightmapShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**HeightmapShape** ()

  Destructor.
- void **FillMsg** (msgs::Geometry &_msg)

  Fill a geometry message with this shape's data.
- float **GetHeight** (int _x, int _y)

  Get a height at a position.
- float **GetMaxHeight** () const

  Get the maximum height.

- float **GetMinHeight** () const

  Get the minimum height.
- **math::Vector3 GetPos** () const

  Get the origin in world coordinate frame.
- **math::Vector3 GetSize** () const

  Get the size in meters.
- int **GetSubSampling** () const

  Get the amount of subsampling.
- std::string **GetURI** () const

  Get the URI of the heightmap image.
- **math::Vector2i GetVertexCount** () const

  Return the number of vertices, which equals the size of the image used to load the heightmap.
- virtual void **Init** ()

  Initialize the heightmap.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the heightmap.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  Update the heightmap from a message.

Protected Attributes

- std::vector< float > **heights**

  Lookup table of heights.
- **common::Image img**

  Image used to generate the heights.
- **math::Vector3 scale**

  Scaling factor.
- int **subSampling**

  Level of subsampling.
- unsigned int **vertSize**

  Size of the height lookup table.

Additional Inherited Members

10.58.1 Detailed Description

HeightmapShape (p. 352) collision shape builds a heightmap from an image.

The supplied image must be square with N∗N+1 pixels per side, where N is an integer.

10.58.2 Constructor & Destructor Documentation

10.58.2.1 gazebo::physics::HeightmapShape::HeightmapShape (**CollisionPtr** *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent **Collision** (p. 195) object.

10.58.2.2 virtual gazebo::physics::HeightmapShape::~HeightmapShape ( ) [virtual]

Destructor.

10.58.3 Member Function Documentation

10.58.3.1 void gazebo::physics::HeightmapShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Fill a geometry message with this shape's data.

Parameters

in	*_msg*	Message to fill.

Implements **gazebo::physics::Shape** (p. 722).

10.58.3.2 float gazebo::physics::HeightmapShape::GetHeight (int *_x*, int *_y*)

Get a height at a position.

Parameters

in	*_x*	X position.
in	*_y*	Y position.

Returns

The height at a the specified location.

10.58.3.3 float gazebo::physics::HeightmapShape::GetMaxHeight ( ) const

Get the maximum height.

Returns

The maximum height.

10.58.3.4 float gazebo::physics::HeightmapShape::GetMinHeight ( ) const

Get the minimum height.

Returns

The minimum height.

10.58.3.5 math::Vector3 gazebo::physics::HeightmapShape::GetPos ( ) const

Get the origin in world coordinate frame.

Returns

The origin in world coordinate frame.

10.58.3.6 math::Vector3 gazebo::physics::HeightmapShape::GetSize ( ) const

Get the size in meters.

Returns

The size in meters.

10.58.3.7 int gazebo::physics::HeightmapShape::GetSubSampling ( ) const

Get the amount of subsampling.

Returns

Amount of subsampling.

10.58.3.8 std::string gazebo::physics::HeightmapShape::GetURI ( ) const

Get the URI of the heightmap image.

Returns

The heightmap image URI.

10.58.3.9 math::Vector2i gazebo::physics::HeightmapShape::GetVertexCount ( ) const

Return the number of vertices, which equals the size of the image used to load the heightmap.

Returns

math::Vector2i (p. 846), result.x = width, result.y = length/height.

10.58.3.10 virtual void gazebo::physics::HeightmapShape::Init ( ) [virtual]

Initialize the heightmap.

Implements **gazebo::physics::Shape** (p. 722).

10.58.3.11 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. 145).

10.58.3.12 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. 722).

10.58.4 Member Data Documentation

10.58.4.1 std::vector<float> gazebo::physics::HeightmapShape::heights [protected]

Lookup table of heights.

10.58.4.2 common::Image gazebo::physics::HeightmapShape::img [protected]

Image used to generate the heights.

10.58.4.3 math::Vector3 gazebo::physics::HeightmapShape::scale [protected]

Scaling factor.

10.58.4.4 int gazebo::physics::HeightmapShape::subSampling [protected]

Level of subsampling.

10.58.4.5 unsigned int gazebo::physics::HeightmapShape::vertSize [protected]

Size of the height lookup table.

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

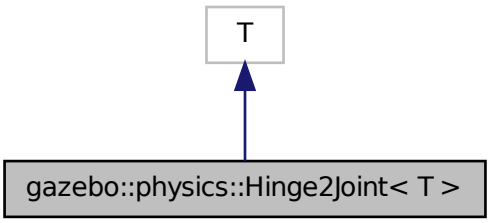
- **HeightmapShape.hh**

10.59 gazebo::physics::Hinge2Joint< T > Class Template Reference

A (p. 111) two axis hinge joint.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Hinge2Joint< T >:



Public Member Functions

- **Hinge2Joint** (**BasePtr** _parent)
  Constructor.
- virtual ~**Hinge2Joint** ()
  Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (**sdf::ElementPtr** _sdf)
  Load the joint.

10.59.1 Detailed Description

template<class T>class gazebo::physics::Hinge2Joint< T >

A (p. 111) two axis hinge joint.

10.59.2 Constructor & Destructor Documentation

10.59.2.1 template<class T > gazebo::physics::Hinge2Joint< T >::Hinge2Joint (BasePtr *_parent*) [inline], [explicit]

Constructor.

Parameters

in	*_parent*	Parent link.

References gazebo::physics::Base::HINGE2_JOINT.

10.59.2.2 template<class T > virtual gazebo::physics::Hinge2Joint< T >::~Hinge2Joint ( ) [inline], [virtual]

Destructor.

10.59.3 Member Function Documentation

10.59.3.1 template<class T > virtual unsigned int gazebo::physics::Hinge2Joint< T >::GetAngleCount ( ) const [inline], [virtual]

10.59.3.2 template<class T > virtual void gazebo::physics::Hinge2Joint< T >::Load (sdf::ElementPtr _sdf) [inline], [virtual]

Load the joint.

Parameters

in	*_sdf*	SDF values to load from.

References sdf::Element::GetElement(), and sdf::Element::GetValueVector3().

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

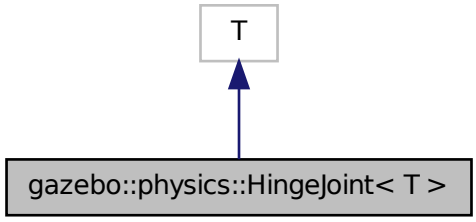
- **Hinge2Joint.hh**

10.60 gazebo::physics::HingeJoint< T > Class Template Reference

A (p. 111) single axis hinge joint.

`#include <physics/physics.hh>`

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



Public Member Functions

- **HingeJoint** (**BasePtr** _parent)

  Constructor.

- virtual ~**HingeJoint** ()

  Destructor.
- virtual unsigned int **GetAngleCount** () const

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load joint.

Protected Member Functions

- virtual void **Init** ()

  Initialize joint.

10.60.1 Detailed Description

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

A (p. 111) single axis hinge joint.

10.60.2 Constructor & Destructor Documentation

10.60.2.1 template<class T > gazebo::physics::HingeJoint< T >::HingeJoint (BasePtr *_parent*) [inline]

Constructor.

Parameters

in	*_parent*	Parent link

References gazebo::physics::Base::HINGE_JOINT.

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

Destructor.

10.60.3 Member Function Documentation

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

10.60.3.2 template<class T > virtual void gazebo::physics::HingeJoint< T >::Init ( ) [inline], [protected], [virtual]

Initialize joint.

References gazebo::msgs::Init().

10.60.3.3 template<class T > virtual void gazebo::physics::HingeJoint< T >::Load (sdf::ElementPtr _sdf) [inline], [virtual]

Load joint.

Parameters

in	*_sdf*	Pointer to SDF element

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

- **HingeJoint.hh**

10.61 gazebo::common::Image Class Reference

Encapsulates an image.

`#include <common/common.hh>`

Public Types

- enum **PixelFormat** {
  UNKNOWN_PIXEL_FORMAT = 0, **L_INT8**, **L_INT16**, **RGB_INT8**,
  RGBA_INT8, **BGRA_INT8**, **RGB_INT16**, **RGB_INT32**,
  BGR_INT8, **BGR_INT16**, **BGR_INT32**, **R_FLOAT16**,
  RGB_FLOAT16, **R_FLOAT32**, **RGB_FLOAT32**, **BAYER_RGGB8**,
  BAYER_RGGR8, **BAYER_GBRG8**, **BAYER_GRBG8**, **PIXEL_FORMAT_COUNT** }

  Pixel formats enumeration.

Public Member Functions

- **Image** (const std::string &_filename="")

  Constructor.
- virtual ~**Image** ()

  Destructor.
- **Color GetAvgColor** ()

  Get the average color.
- unsigned int **GetBPP** () const

  Get the size of one pixel in bits.
- void **GetData** (unsigned char **_data, unsigned int &_count) const

  Get the image as a data array.
- std::string **GetFilename** () const

  Get the full filename of the image.
- unsigned int **GetHeight** () const

  Get the height.
- **Color GetMaxColor** ()

  Get the max color.
- int **GetPitch** () const
- **Color GetPixel** (unsigned int _x, unsigned int _y)

Get a pixel color value.

- **PixelFormat GetPixelFormat** () const

  Get the pixel format.

- void **GetRGBData** (unsigned char **_data, unsigned int &_count) const

  Get only the RGB data from the image.

- unsigned int **GetWidth** () const

  Get the width.

- int **Load** (const std::string &_filename)

  Load an image.

- void **Rescale** (int _width, int _height)

  Rescale the image.

- void **SavePNG** (const std::string &_filename)

  Save the image in PNG format.

- void **SetFromData** (const unsigned char *_data, unsigned int _width, unsigned int _height, **Image::PixelFormat** _format)

  Set the image from raw data.

- bool **Valid** () const

  Returns whether this is a valid image.

Static Public Member Functions

- static **Image::PixelFormat ConvertPixelFormat** (const std::string &_format)

  *Convert a string to a **Image::PixelFormat** (p. 361).*

10.61.1 Detailed Description

Encapsulates an image.

10.61.2 Member Enumeration Documentation

10.61.2.1 enum gazebo::common::Image::PixelFormat

Pixel formats enumeration.

Enumerator:

UNKNOWN_PIXEL_FORMAT

L_INT8

L_INT16

RGB_INT8

RGBA_INT8

BGRA_INT8

RGB_INT16

RGB_INT32

BGR_INT8

BGR_INT16

BGR_INT32

R_FLOAT16

RGB_FLOAT16

R_FLOAT32

RGB_FLOAT32

BAYER_RGGB8

BAYER_RGGR8

BAYER_GBRG8

BAYER_GRBG8

PIXEL_FORMAT_COUNT

10.61.3 Constructor & Destructor Documentation

10.61.3.1 gazebo::common::Image::Image (const std::string & *_filename* = " ") [explicit]

Constructor.

Parameters

in	*_filename*	the path to the image

10.61.3.2 virtual gazebo::common::Image::~Image ( ) [virtual]

Destructor.

10.61.4 Member Function Documentation

10.61.4.1 static Image::PixelFormat gazebo::common::Image::ConvertPixelFormat (const std::string & *_format*) [static]

Convert a string to a **Image::PixelFormat** (p. 361).

Parameters

in	*_format*	Pixel format string.

See Also

Image::PixelFormatNames

Returns

Image::PixelFormat (p. 361)

10.61.4.2 Color gazebo::common::Image::GetAvgColor ( )

Get the average color.

Returns

The average color

10.61.4.3 unsigned int gazebo::common::Image::GetBPP ( ) const

Get the size of one pixel in bits.

Returns

The BPP of the image

10.61.4.4 void gazebo::common::Image::GetData (unsigned char ** *_data*, unsigned int & *_count*) const

Get the image as a data array.

Parameters

out	*_data*	Pointer to a NULL array of char.
out	*_count*	The resulting data array size

10.61.4.5 std::string gazebo::common::Image::GetFilename ( ) const

Get the full filename of the image.

Returns

The filename used to load the image

10.61.4.6 unsigned int gazebo::common::Image::GetHeight ( ) const

Get the height.

Returns

The image height

10.61.4.7 Color gazebo::common::Image::GetMaxColor ( )

Get the max color.

Returns

The max color

10.61.4.8 int gazebo::common::Image::GetPitch ( ) const

Returns

The pitch of the image

10.61.4.9 Color gazebo::common::Image::GetPixel (unsigned int _x, unsigned int _y)

Get a pixel color value.

Parameters

in	_x	Column location in the image
in	_y	Row location in the image

10.61.4.10 PixelFormat gazebo::common::Image::GetPixelFormat ( ) const

Get the pixel format.

Returns

PixelFormat

10.61.4.11 void gazebo::common::Image::GetRGBData (unsigned char ** _data, unsigned int & _count) const

Get only the RGB data from the image.

This will drop the alpha channel if one is present.

Parameters

out	_data	Pointer to a NULL array of char.
out	_count	The resulting data array size

10.61.4.12 unsigned int gazebo::common::Image::GetWidth ( ) const

Get the width.

Returns

The image width

10.61.4.13 int gazebo::common::Image::Load (const std::string & _filename)

Load an image.

Return 0 on success

Parameters

in	_filename	the path to the image file

10.61.4.14 void gazebo::common::Image::Rescale (int _width, int _height)

Rescale the image.

Parameters

in	*_width*	New image width
in	*_height*	New image height

10.61.4.15 void gazebo::common::Image::SavePNG (const std::string & *_filename*)

Save the image in PNG format.

Parameters

in	*_filename*	The name of the saved image

10.61.4.16 void gazebo::common::Image::SetFromData (const unsigned char * *_data*, unsigned int *_width*, unsigned int *_height*, Image::PixelFormat *_format*)

Set the image from raw data.

Parameters

in	*_data*	Pointer to the raw image data
in	*_width*	Width in pixels
in	*_height*	Height in pixels
in	*_format*	Pixel format of the provided data

10.61.4.17 bool gazebo::common::Image::Valid ( ) const

Returns whether this is a valid image.

Returns

true if image has a bitmap

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

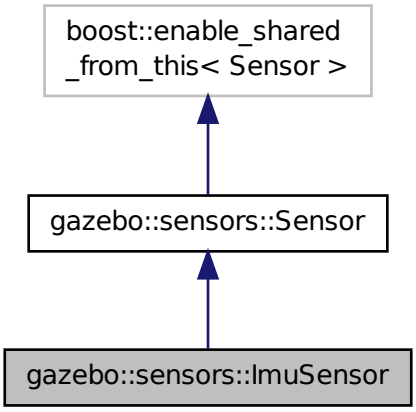
- **Image.hh**

10.62 gazebo::sensors::ImuSensor Class Reference

An IMU sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::ImuSensor:



Public Member Functions

- **ImuSensor** ()

  Constructor.
- virtual ~**ImuSensor** ()

  Destructor.
- **math::Vector3 GetAngularVelocity** () const

  Returns the angular velocity.
- msgs::IMU **GetImuMessage** () const

  Returns the imu message.
- **math::Vector3 GetLinearAcceleration** () const

  Returns the imu linear acceleration.
- **math::Quaternion GetOrientation** () const

  get orientation of the IMU relative to the reference pose
- virtual void **Init** ()

  Initialize the IMU.
- void **SetReferencePose** ()

  Sets the current pose as the IMU reference pose.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the sensor.
- void **Load** (const std::string &_worldName, **sdf::ElementPtr** _sdf)

  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)

Load the sensor with default parameters.

- virtual void **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

Additional Inherited Members

10.62.1 Detailed Description

An IMU sensor.

10.62.2 Constructor & Destructor Documentation

10.62.2.1 gazebo::sensors::ImuSensor::ImuSensor ( )

Constructor.

10.62.2.2 virtual gazebo::sensors::ImuSensor::~ImuSensor ( ) [virtual]

Destructor.

10.62.3 Member Function Documentation

10.62.3.1 virtual void gazebo::sensors::ImuSensor::Fini ( ) [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 702).

10.62.3.2 math::Vector3 gazebo::sensors::ImuSensor::GetAngularVelocity ( ) const

Returns the angular velocity.

Returns

Angular velocity.

10.62.3.3 msgs::IMU gazebo::sensors::ImuSensor::GetImuMessage ( ) const

Returns the imu message.

Returns

Imu message.

10.62.3.4 math::Vector3 gazebo::sensors::ImuSensor::GetLinearAcceleration ( ) const

Returns the imu linear acceleration.

Returns

Linear acceleration.

10.62.3.5 math::Quaternion gazebo::sensors::ImuSensor::GetOrientation ( ) const

get orientation of the IMU relative to the reference pose

Returns

returns the orientation quaternion of the IMU relative to the imu reference pose.

10.62.3.6 virtual void gazebo::sensors::ImuSensor::Init ( ) [virtual]

Initialize the IMU.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.62.3.7 void gazebo::sensors::ImuSensor::Load (const std::string & *_worldName,* sdf::ElementPtr *_sdf*) [protected], [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 698) parameters.
in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.62.3.8 virtual void gazebo::sensors::ImuSensor::Load (const std::string & *_worldName*) [protected], [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

10.62.3.9 void gazebo::sensors::ImuSensor::SetReferencePose ( )

Sets the current pose as the IMU reference pose.

10.62.3.10 virtual void gazebo::sensors::ImuSensor::UpdateImpl (bool) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 706).

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

- **ImuSensor.hh**

10.63 gazebo::physics::Inertial Class Reference

A (p. 111) class for inertial information about a link.

`#include <physics/physics.hh>`

Public Member Functions

- **Inertial** ()

  Default Constructor.
- **Inertial** (double _mass)

  Constructor.
- **Inertial** (const **Inertial** &_inertial)

  Copy constructor.
- virtual ~**Inertial** ()

  Destructor.
- const **math::Vector3** & **GetCoG** () const

  Get the center of gravity.
- **Inertial GetInertial** (const **math::Pose** &_frameOffset) const

  *Get equivalent Inertia values with the **Link** (p. 418) frame offset, while holding the Pose of CoG constant in the world frame.*
- double **GetIXX** () const

  Get IXX.
- double **GetIXY** () const

  Get IXY.
- double **GetIXZ** () const

  Get IXZ.
- double **GetIYY** () const

  Get IYY.
- double **GetIYZ** () const

  Get IXZ.

- double **GetIZZ** () const

  Get IZZ.
- double **GetMass** () const

  Get the mass.
- **math::Matrix3 GetMOI** (const **math::Pose** &_pose) const

  *Get the equivalent inertia from a point in local **Link** (p. 418) frame If you specify GetMOI(this->**GetPose()** (p. 374)), you should get back the Moment of Inertia (MOI) exactly as specified in the SDF.*
- **math::Matrix3 GetMOI** () const

  returns Moments of Inertia as a Matrix3
- const **math::Pose GetPose** () const

  *Get the pose about which the mass and inertia matrix is specified in the **Link** (p. 418) frame.*
- **math::Vector3 GetPrincipalMoments** () const

  Get the principal moments of inertia (Ixx, Iyy, Izz).
- **math::Vector3 GetProductsofInertia** () const

  Get the products of inertia (Ixy, Ixz, Iyz).
- void **Load** (**sdf::ElementPtr** _sdf)

  Load from SDF values.
- **Inertial operator+** (const **Inertial** &_inertial) const

  Addition operator.
- const **Inertial** & **operator+=** (const **Inertial** &_inertial)

  Addition equal operator.
- **Inertial** & **operator=** (const **Inertial** &_inertial)

  Equal operator.
- void **ProcessMsg** (const msgs::Inertial &_msg)

  Update parameters from a message.
- void **Reset** ()

  Reset all the mass properties.
- void **Rotate** (const **math::Quaternion** &_rot)

  Rotate this mass.
- void **SetCoG** (double _cx, double _cy, double _cz)

  Set the center of gravity.
- void **SetCoG** (const **math::Vector3** &_center)

  Set the center of gravity.
- void **SetCoG** (double _cx, double _cy, double _cz, double _rx, double _ry, double _rz)

  *Set the center of gravity and rotation offset of inertial coordinate frame relative to **Link** (p. 418) frame.*
- void **SetCoG** (const **math::Pose** &_c)

  Set the center of gravity.
- void **SetInertiaMatrix** (double _ixx, double _iyy, double _izz, double _ixy, double _ixz, double iyz)

  Set the mass matrix.
- void **SetIXX** (double _v)

  Set IXX.
- void **SetIXY** (double _v)

  Set IXY.
- void **SetIXZ** (double _v)

  Set IXZ.
- void **SetIYY** (double _v)

  Set IYY.

- void **SetIYZ** (double _v)

  Set IYZ.
- void **SetIZZ** (double _v)

  Set IZZ.
- void **SetMass** (double m)

  Set the mass.
- void **SetMOI** (const **math::Matrix3** &_moi)

  Sets Moments of Inertia (MOI) from a Matrix3.
- void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  update the parameters using new sdf values.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::Inertial** &_inertial)

  Output operator.

10.63.1 Detailed Description

A (p. 111) class for inertial information about a link.

10.63.2 Constructor & Destructor Documentation

10.63.2.1 gazebo::physics::Inertial::Inertial ( )

Default Constructor.

10.63.2.2 gazebo::physics::Inertial::Inertial (double *_mass*) [explicit]

Constructor.

Parameters

in	*_mass*	Mass value in kg if using metric.

10.63.2.3 gazebo::physics::Inertial::Inertial (const **Inertial** & *_inertial*)

Copy constructor.

Parameters

in	*_inertial*	**Inertial** (p. 369) element to copy

10.63.2.4 virtual gazebo::physics::Inertial::~Inertial ( ) [virtual]

Destructor.

10.63.3 Member Function Documentation

10.63.3.1 const math::Vector3& gazebo::physics::Inertial::GetCoG ( ) const [inline]

Get the center of gravity.

Returns

The center of gravity.

References gazebo::math::Pose::pos.

10.63.3.2 Inertial gazebo::physics::Inertial::GetInertial (const math::Pose & *_frameOffset*) const

Get equivalent Inertia values with the **Link** (p. 418) frame offset, while holding the Pose of CoG constant in the world frame.

Parameters

in	*_frameOffset*	amount to offset the **Link** (p. 418) frame by, this is a transform defined in the **Link** (p. 418) frame.

Returns

Inertial (p. 369) parameters with the shifted frame.

10.63.3.3 double gazebo::physics::Inertial::GetIXX ( ) const

Get IXX.

Returns

IXX value

10.63.3.4 double gazebo::physics::Inertial::GetIXY ( ) const

Get IXY.

Returns

IXY value

10.63.3.5 double gazebo::physics::Inertial::GetIXZ ( ) const

Get IXZ.

Returns

IXZ value

10.63.3.6 double gazebo::physics::Inertial::GetIYY ( ) const

Get IYY.

Returns

IYY value

10.63.3.7 double gazebo::physics::Inertial::GetIYZ ( ) const

Get IXZ.

Returns

IYZ value

10.63.3.8 double gazebo::physics::Inertial::GetIZZ ( ) const

Get IZZ.

Returns

IZZ value

10.63.3.9 double gazebo::physics::Inertial::GetMass ( ) const

Get the mass.

10.63.3.10 math::Matrix3 gazebo::physics::Inertial::GetMOI (const math::Pose & *_pose*) const

Get the equivalent inertia from a point in local **Link** (p. 418) frame If you specify GetMOI(this->**GetPose()** (p. 374)), you should get back the Moment of Inertia (MOI) exactly as specified in the SDF.

If _pose is different from pose of the **Inertial** (p. 369) block, then the MOI is rotated accordingly, and contributions from changes in MOI location location due to point mass is added to the final MOI.

Parameters

in	*_pose*	location in **Link** (p. 418) local frame

Returns

equivalent inertia at _pose

10.63.3.11 math::Matrix3 gazebo::physics::Inertial::GetMOI ( ) const

returns Moments of Inertia as a Matrix3

Returns

Moments of Inertia as a Matrix3

10.63.3.12 const math::Pose gazebo::physics::Inertial::GetPose ( ) const [inline]

Get the pose about which the mass and inertia matrix is specified in the **Link** (p. 418) frame.

Returns

The inertial pose.

10.63.3.13 math::Vector3 gazebo::physics::Inertial::GetPrincipalMoments ( ) const

Get the principal moments of inertia (Ixx, Iyy, Izz).

Returns

The principal moments.

10.63.3.14 math::Vector3 gazebo::physics::Inertial::GetProductsofInertia ( ) const

Get the products of inertia (Ixy, Ixz, Iyz).

Returns

The products of inertia.

10.63.3.15 void gazebo::physics::Inertial::Load (sdf::ElementPtr *_sdf*)

Load from SDF values.

Parameters

in	*_sdf*	SDF value to load from.

10.63.3.16 Inertial gazebo::physics::Inertial::operator+ (const Inertial & *_inertial*) const

Addition operator.

Assuming both CG and Moment of Inertia (MOI) are defined in the same reference **Link** (p. 418) frame. New CG is computed from masses and perspective offsets, and both MOI contributions relocated to the new cog.

Parameters

in	*_inertial*	**Inertial** (p. 369) to add.

Returns

The result of the addition.

10.63.3.17 const Inertial& gazebo::physics::Inertial::operator+= (const Inertial & _inertial)

Addition equal operator.

Parameters

in	*_inertial*	**Inertial** (p. 369) to add.

Returns

Reference to this object.

10.63.3.18 Inertial& gazebo::physics::Inertial::operator= (const Inertial & _inertial)

Equal operator.

Parameters

in	*_inertial*	**Inertial** (p. 369) to copy.

Returns

Reference to this object.

10.63.3.19 void gazebo::physics::Inertial::ProcessMsg (const msgs::Inertial & _msg)

Update parameters from a message.

Parameters

in	*_msg*	Message to read

10.63.3.20 void gazebo::physics::Inertial::Reset ( )

Reset all the mass properties.

10.63.3.21 void gazebo::physics::Inertial::Rotate (const math::Quaternion & _rot)

Rotate this mass.

Parameters

in	*_rot*	Rotation amount.

10.63.3.22 void gazebo::physics::Inertial::SetCoG (double *_cx*, double *_cy*, double *_cz*)

Set the center of gravity.

Parameters

in	*_cx*	X position.
in	*_cy*	Y position.
in	*_cz*	Z position.

10.63.3.23 void gazebo::physics::Inertial::SetCoG (const math::Vector3 & *_center*)

Set the center of gravity.

Parameters

in	*_center*	Center of the gravity.

10.63.3.24 void gazebo::physics::Inertial::SetCoG (double *_cx*, double *_cy*, double *_cz*, double *_rx*, double *_ry*, double *_rz*)

Set the center of gravity and rotation offset of inertial coordinate frame relative to **Link** (p. 418) frame.

Parameters

in	*_cx*	Center offset in x-direction in **Link** (p. 418) frame
in	*_cy*	Center offset in y-direction in **Link** (p. 418) frame
in	*_cz*	Center offset in z-direction in **Link** (p. 418) frame
in	*_rx*	Roll angle offset of inertial coordinate frame.
in	*_ry*	Pitch angle offset of inertial coordinate frame.
in	*_rz*	Yaw angle offset of inertial coordinate frame.

10.63.3.25 void gazebo::physics::Inertial::SetCoG (const math::Pose & *_c*)

Set the center of gravity.

Parameters

in	*_c*	Transform to center of gravity.

10.63.3.26 void gazebo::physics::Inertial::SetInertiaMatrix (double *_ixx*, double *_iyy*, double *_izz*, double *_ixy*, double *_ixz*, double *iyz*)

Set the mass matrix.

Parameters

in	*_ixx*	X second moment of inertia (MOI) about x axis.
in	*_iyy*	Y second moment of inertia about y axis.
in	*_izz*	Z second moment of inertia about z axis.
in	*_ixy*	XY inertia.

in	*_ixz*	XZ inertia.
in	*_iyz*	YZ inertia.

10.63.3.27 void gazebo::physics::Inertial::SetIXX (double _v)

Set IXX.

Parameters

in	*_v*	IXX value

10.63.3.28 void gazebo::physics::Inertial::SetIXY (double _v)

Set IXY.

Parameters

in	*_v*	IXY value

10.63.3.29 void gazebo::physics::Inertial::SetIXZ (double _v)

Set IXZ.

Parameters

in	*_v*	IXZ value

10.63.3.30 void gazebo::physics::Inertial::SetIYY (double _v)

Set IYY.

Parameters

in	*_v*	IYY value

10.63.3.31 void gazebo::physics::Inertial::SetIYZ (double _v)

Set IYZ.

Parameters

in	*_v*	IXX value

10.63.3.32 void gazebo::physics::Inertial::SetIZZ (double _v)

Set IZZ.

Parameters

in	*_v*	IZZ value

10.63.3.33 void gazebo::physics::Inertial::SetMass (double *m*)

Set the mass.

10.63.3.34 void gazebo::physics::Inertial::SetMOI (const **math::Matrix3** & *_moi*)

Sets Moments of Inertia (MOI) from a Matrix3.

Parameters

in	*Moments*	of Inertia as a Matrix3

10.63.3.35 void gazebo::physics::Inertial::UpdateParameters (**sdf::ElementPtr** *_sdf*)

update the parameters using new sdf values.

Parameters

in	*_sdf*	Update values from.

10.63.4 Friends And Related Function Documentation

10.63.4.1 std::ostream& operator<< (std::ostream & *_out,* const **gazebo::physics::Inertial** & *_inertial*) [friend]

Output operator.

Parameters

in	*_out*	Output stream.
in	*_inertial*	**Inertial** (p. 369) object to output.

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

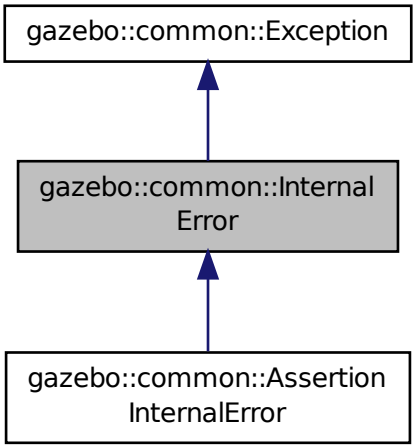
- **Inertial.hh**

10.64 gazebo::common::InternalError Class Reference

Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::InternalError:



Public Member Functions

- **InternalError** ()

  Constructor.
- **InternalError** (const char ∗_file, int _line, const std::string &_msg)

  Default constructor.
- virtual ∼**InternalError** ()

  Destructor.

10.64.1 Detailed Description

Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.

10.64.2 Constructor & Destructor Documentation

10.64.2.1 gazebo::common::InternalError::InternalError ( )

Constructor.

10.64.2.2 gazebo::common::InternalError::InternalError (const char ∗ *_file*, int *_line*, const std::string & *_msg*)

Default constructor.

Parameters

in	*_file*	File name
in	*_line*	Line number where the error occurred
in	*_msg*	Error message

10.64.2.3 virtual gazebo::common::InternalError::~InternalError ( ) [virtual]

Destructor.

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

- **Exception.hh**

10.65 gazebo::transport::IOManager Class Reference

Manages boost::asio IO.

`#include <transport/transport.hh>`

Public Member Functions

- **IOManager** ()
  Constructor.
- ~**IOManager** ()
  Destructor.
- void **DecCount** ()
  Decrement the event count by 1.
- unsigned int **GetCount** () const
  Get the event count.
- boost::asio::io_service & **GetIO** ()
  Get handle to boost::asio IO service.
- void **IncCount** ()
  Increment the event count by 1.
- void **Stop** ()
  Stop the IO service.

10.65.1 Detailed Description

Manages boost::asio IO.

10.65.2 Constructor & Destructor Documentation

10.65.2.1 gazebo::transport::IOManager::IOManager ( )

Constructor.

10.65.2.2 gazebo::transport::IOManager::~IOManager ( )

Destructor.

10.65.3 Member Function Documentation

10.65.3.1 void gazebo::transport::IOManager::DecCount ( )

Decrement the event count by 1.

10.65.3.2 unsigned int gazebo::transport::IOManager::GetCount ( ) const

Get the event count.

Returns

The event count

10.65.3.3 boost::asio::io_service& gazebo::transport::IOManager::GetIO ( )

Get handle to boost::asio IO service.

Returns

Handle to boost::asio IO service

10.65.3.4 void gazebo::transport::IOManager::IncCount ( )

Increment the event count by 1.

10.65.3.5 void gazebo::transport::IOManager::Stop ( )

Stop the IO service.

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

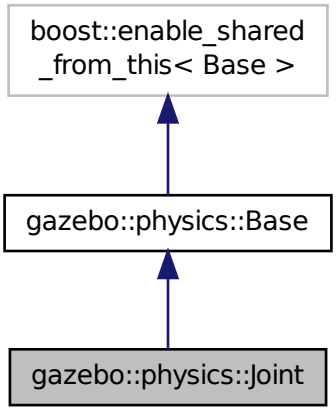
- **IOManager.hh**

10.66 gazebo::physics::Joint Class Reference

Base (p. 137) 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. 381) attribute types.*

Public Member Functions

- **Joint** (**BasePtr** _parent)

  Constructor.
- virtual ~**Joint** ()

  Destructor.
- virtual void **ApplyDamping** ()

  Callback to apply damping force to joint.
- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const =0

  Determines of 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 the the joint update signal.
- virtual void **Detach** ()

  Detach this joint from all links.
- void **DisconnectJointUpdate** (**event::ConnectionPtr** &_conn)

  Disconnect a boost::slot the the joint update signal.

- void **FillMsg** (msgs::Joint &_msg)

  Fill a joint message.
- virtual **math::Vector3 GetAnchor** (int _index) const =0

  Get the anchor point.
- **math::Angle GetAngle** (int _index) const

  Get the angle of rotation of an axis(index)
- virtual unsigned int **GetAngleCount** () const =0

  Get the angle count.
- virtual double **GetAttribute** (const std::string &_key, unsigned int _index)=0

  Get a non-generic parameter for the joint.
- **LinkPtr GetChild** () const

  Get the child link.
- virtual double **GetEffortLimit** (int _index)

  Get the effort limit on axis(index).
- virtual double **GetForce** (int _index) **GAZEBO_DEPRECATED**(1.5)
- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench GetForceTorque** (int _index) **GAZEBO_DEPRECATED**(1.5)=0

  get internal force and torque values at a joint Note that you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.
- virtual **JointWrench GetForceTorque** (unsigned int _index)=0

  get internal force and torque values at a joint Note that you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const =0

  Get the axis of rotation in global cooridnate frame.
- virtual **math::Angle GetHighStop** (int _index)=0

  Get the high stop of an axis(index).
- double **GetInertiaRatio** (unsigned int _index) const

  Accessor to inertia ratio across this joint.
- virtual **LinkPtr GetJointLink** (int _index) const =0

  Get the link to which the joint is attached according the _index.
- virtual **math::Vector3 GetLinkForce** (unsigned int _index) const =0

  *Get the forces applied to the center of mass of a **physics::Link** (p. 418) due to the existence of this **Joint** (p. 381).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const =0

  *Get the torque applied to the center of mass of a **physics::Link** (p. 418) due to the existence of this **Joint** (p. 381).*
- **math::Vector3 GetLocalAxis** (int _index) const

  Get the axis of rotation.
- **math::Angle GetLowerLimit** (unsigned int _index) const

  : get the joint upper limit (replaces GetLowStop and GetHighStop)
- virtual **math::Angle GetLowStop** (int _index)=0

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)=0

  Get the max allowed force of an axis(index).
- **LinkPtr GetParent** () const

  Get the parent link.
- **math::Angle GetUpperLimit** (unsigned int _index) const

  : get the joint lower limit (replacee GetLowStop and GetHighStop)
- virtual double **GetVelocity** (int _index) const =0

Get the rotation rate of an axis(index)

- virtual double **GetVelocityLimit** (int _index)

  Get the velocity limit on axis(index).

- virtual void **Init** ()

  Initialize a joint.

- void **Load** (**LinkPtr** _parent, **LinkPtr** _child, const **math::Pose** &_pose)

  *Set pose, parent and child links of a **physics::Joint*** (p. *381).*

- void **Load** (**LinkPtr** _parent, **LinkPtr** _child, const **math::Vector3** &_pos) **GAZEBO_DEPRECATED**(1.5)

  *Set parent and child links of a **physics::Joint*** (p. *381) and its anchor offset position.*

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load **physics::Joint*** (p. *381) from a SDF **sdf::Element*** (p. *273).*

- 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. *381) is static, Gazebo stores the state of this **Joint*** (p. *381) as a scalar inside the **Joint*** (p. *381) class, so this call will NOT move the joint dynamically for a static **Model*** (p. *489).*

- virtual void **SetAttribute** (const std::string &_key, int _index, const boost::any &_value)=0

  Set a non-generic parameter for the joint.

- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)=0

  Set the axis of rotation.

- virtual void **SetDamping** (int _index, double _damping)=0

  Set the joint damping.

- virtual void **SetForce** (int _index, double _force)

  *Set the force applied to this **physics::Joint*** (p. *381).*

- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).

- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).

- virtual void **SetMaxForce** (int _index, double _force)=0

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

- void **SetModel** (**ModelPtr** _model)

  Set the model this joint belongs too.

- void **SetState** (const **JointState** &_state)

  Set the joint state.

- virtual void **SetVelocity** (int _index, double _vel)=0

  Set the velocity of an axis(index).

- void **Update** ()

  Update the joint.

- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  Update the parameters using new sdf values.

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (int _index) const =0

  Get the angle of an axis helper function.

Protected Attributes

- **LinkPtr anchorLink**

  Anchor link.
- **math::Vector3 anchorPos**

  Anchor pose.
- **math::Pose anchorPose**

  Anchor pose specified in SDF <joint><pose> tag.
- **gazebo::event::ConnectionPtr applyDamping**

  apply damping for adding viscous damping forces on updates
- **LinkPtr childLink**

  The first link this joint connects to.
- double **dampingCoefficient**

  joint dampingCoefficient
- double **effortLimit** [2]

  *Store **Joint** (p. 381) effort limit as specified in SDF.*
- double **forceApplied** [2]

  Save force applied by user This plus the joint feedback (joint contstraint forces) is the equivalent of simulated force torque sensor reading Allocate a 2 vector in case hinge2 joint is used.
- double **inertiaRatio** [2]

  *Store **Joint** (p. 381) inertia ratio.*
- **math::Angle lowerLimit** [2]

  *Store **Joint** (p. 381) position lower limit as specified in SDF.*
- **ModelPtr model**

  Pointer to the parent model.
- **LinkPtr parentLink**

  The second link this joint connects to.
- **math::Angle upperLimit** [2]

  *Store **Joint** (p. 381) position upper limit as specified in SDF.*
- bool **useCFMDamping**

  option to use CFM damping
- double **velocityLimit** [2]

  *Store **Joint** (p. 381) velocity limit as specified in SDF.*

10.66.1 Detailed Description

Base (p. 137) class for all joints.

10.66.2 Member Enumeration Documentation

10.66.2.1 enum gazebo::physics::Joint::Attribute

Joint (p. 381) attribute types.

Enumerator:

FUDGE_FACTOR Fudge factor.

SUSPENSION_ERP Suspension error reduction parameter.

SUSPENSION_CFM Suspension constraint force mixing.

STOP_ERP Stop limit error reduction parameter.

STOP_CFM Stop limit constraint force mixing.

ERP Error reduction parameter.

CFM Constraint force mixing.

FMAX Maximum force.

VEL Velocity.

HI_STOP High stop angle.

LO_STOP Low stop angle.

10.66.3 Constructor & Destructor Documentation

10.66.3.1 gazebo::physics::Joint::Joint (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	***Joint*** (p. *381*)	parent

10.66.3.2 virtual gazebo::physics::Joint::~Joint ( ) [virtual]

Destructor.

10.66.4 Member Function Documentation

10.66.4.1 virtual void gazebo::physics::Joint::ApplyDamping ( ) [virtual]

Callback to apply damping force to joint.

10.66.4.2 virtual bool gazebo::physics::Joint::AreConnected (LinkPtr *_one,* LinkPtr *_two*) const [pure virtual]

Determines of the two bodies are connected by a joint.

Parameters

in	*_one*	First link.
in	*_two*	Second link.

Returns

True if the two links are connected by a joint.

10.66.4.3 virtual void gazebo::physics::Joint::Attach (LinkPtr *_parent,* LinkPtr *_child*) [virtual]

Attach the two bodies with this joint.

Parameters

in	*_parent*	Parent link.
in	*_child*	Child link.

10.66.4.4 template<typename T > event::ConnectionPtr gazebo::physics::Joint::ConnectJointUpdate (T *_subscriber*) [inline]

Connect a boost::slot the the joint update signal.

Parameters

in	*_subscriber*	Callback for the connection.

Returns

Connection pointer, which must be kept in scope.

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

10.66.4.5 virtual void gazebo::physics::Joint::Detach ( ) [virtual]

Detach this joint from all links.

10.66.4.6 void gazebo::physics::Joint::DisconnectJointUpdate (event::ConnectionPtr & *_conn*) [inline]

Disconnect a boost::slot the the joint update signal.

Parameters

in	*_conn*	Connection to disconnect.

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

10.66.4.7 void gazebo::physics::Joint::FillMsg (msgs::Joint & *_msg*)

Fill a joint message.

Parameters

out	*_msg*	Message to fill with this joint's properties.

10.66.4.8 virtual math::Vector3 gazebo::physics::Joint::GetAnchor (int *_index*) const [pure virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

10.66.4.9 math::Angle gazebo::physics::Joint::GetAngle (int _index) const

Get the angle of rotation of an axis(index)

Parameters

in	_index	Index of the axis.

Returns

Angle of the axis.

10.66.4.10 virtual unsigned int gazebo::physics::Joint::GetAngleCount ( ) const [pure virtual]

Get the angle count.

Returns

The number of DOF for the joint.

Referenced by GetLowerLimit(), and GetUpperLimit().

10.66.4.11 virtual math::Angle gazebo::physics::Joint::GetAngleImpl (int _index) const [protected], [pure virtual]

Get the angle of an axis helper function.

Parameters

in	_index	Index of the axis.

Returns

Angle of the axis.

10.66.4.12 virtual double gazebo::physics::Joint::GetAttribute (const std::string & _key, unsigned int _index) [pure virtual]

Get a non-generic parameter for the joint.

Parameters

in	_key	String key.
in	_index	Index of the axis.
in	_value	Value of the attribute.

10.66.4.13 LinkPtr gazebo::physics::Joint::GetChild ( ) const

Get the child link.

Returns

Pointer to the child link.

10.66.4.14 virtual double gazebo::physics::Joint::GetEffortLimit (int _index) [virtual]

Get the effort limit on axis(index).

Parameters

in	*_index*	Index of axis, where 0=first axis and 1=second axis

Returns

Effort limit specified in SDF

10.66.4.15 virtual double gazebo::physics::Joint::GetForce (int _index) [virtual]

Todo : not yet implemented. Get the forces applied at this **Joint** (p. 381). Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

in	*_index*	Index of the axis.

Returns

The force applied to an axis.

10.66.4.16 virtual double gazebo::physics::Joint::GetForce (unsigned int _index) [virtual]

Todo : not yet implemented. Get the forces applied at this **Joint** (p. 381). Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

in	*_index*	Index of the axis.

Returns

The force applied to an axis.

10.66.4.17 virtual JointWrench gazebo::physics::Joint::GetForceTorque (int _index) [pure virtual]

get internal force and torque values at a joint Note that you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

Parameters

in	_index	Force and torque on child link if _index = 0 and on parent link of _index = 1

Returns

The force and torque at the joint

10.66.4.18 virtual JointWrench gazebo::physics::Joint::GetForceTorque (unsigned int _index) [pure virtual]

get internal force and torque values at a joint Note that you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

Parameters

in	_index	Force and torque on child link if _index = 0 and on parent link of _index = 1

Returns

The force and torque at the joint

10.66.4.19 virtual math::Vector3 gazebo::physics::Joint::GetGlobalAxis (int _index) const [pure virtual]

Get the axis of rotation in global cooridnate frame.

Parameters

in	_index	Index of the axis to get.

Returns

Axis value for the provided index.

10.66.4.20 virtual math::Angle gazebo::physics::Joint::GetHighStop (int _index) [pure virtual]

Get the high stop of an axis(index).

This function is replaced by GetUpperLimit(unsigned int). If you are interested in getting the value of dParamHiStop∗, use GetAttribute(hi_stop, _index)

Parameters

in	_index	Index of the axis.

Returns

Angle of the high stop value.

10.66.4.21 double gazebo::physics::Joint::GetInertiaRatio (unsigned int _index) const

Accessor to inertia ratio across this joint.

Parameters

in	*_index*	**Joint** (p. 381) axis index.

Returns

returns the inertia ratio across specified joint axis.

10.66.4.22 virtual LinkPtr gazebo::physics::Joint::GetJointLink (int _index) const [pure virtual]

Get the link to which the joint is attached according the _index.

Parameters

in	*_index*	Index of the link to retreive.

Returns

Pointer to the request link. NULL if the index was invalid.

10.66.4.23 virtual math::Vector3 gazebo::physics::Joint::GetLinkForce (unsigned int _index) const [pure virtual]

Get the forces applied to the center of mass of a **physics::Link** (p. 418) due to the existence of this **Joint** (p. 381).

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

Parameters

in	*index*	The index of the link(0 or 1).

Returns

Force applied to the link.

10.66.4.24 virtual math::Vector3 gazebo::physics::Joint::GetLinkTorque (unsigned int _index) const [pure virtual]

Get the torque applied to the center of mass of a **physics::Link** (p. 418) due to the existence of this **Joint** (p. 381).

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

in	*index*	The index of the link(0 or 1)

Returns

Torque applied to the link.

10.66.4.25 math::Vector3 gazebo::physics::Joint::GetLocalAxis (int _index) const

Get the axis of rotation.

Parameters

in	*_index*	Index of the axis to get.

Returns

Axis value for the provided index.

10.66.4.26 math::Angle gazebo::physics::Joint::GetLowerLimit (unsigned int _index) const [inline]

: get the joint upper limit (replaces GetLowStop and GetHighStop)

Parameters

in	*_index*	Index of the axis.

Returns

Upper limit of the axis.

References GetAngleCount(), gzwarn, and lowerLimit.

10.66.4.27 virtual math::Angle gazebo::physics::Joint::GetLowStop (int _index) [pure virtual]

Get the low stop of an axis(index).

This function is replaced by GetLowerLimit(unsigned int). If you are interested in getting the value of dParamHiStop∗, use GetAttribute(hi_stop, _index)

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the low stop value.

10.66.4.28 virtual double gazebo::physics::Joint::GetMaxForce (int _index) [pure virtual]

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

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

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

10.66.4.29 LinkPtr gazebo::physics::Joint::GetParent ( ) const

Get the parent link.

Returns

Pointer to the parent link.

10.66.4.30 math::Angle gazebo::physics::Joint::GetUpperLimit (unsigned int _index) const [inline]

: get the joint lower limit (replacee GetLowStop and GetHighStop)

Parameters

in	*_index*	Index of the axis.

Returns

Upper limit of the axis.

References GetAngleCount(), gzwarn, and upperLimit.

10.66.4.31 virtual double gazebo::physics::Joint::GetVelocity (int _index) const [pure virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

10.66.4.32 virtual double gazebo::physics::Joint::GetVelocityLimit (int _index) [virtual]

Get the velocity limit on axis(index).

Parameters

in	*_index*	Index of axis, where 0=first axis and 1=second axis

Returns

Velocity limit specified in SDF

10.66.4.33 virtual void gazebo::physics::Joint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Base** (p. 145).

10.66.4.34 void gazebo::physics::Joint::Load (LinkPtr _parent, LinkPtr _child, const math::Pose & _pose)

Set pose, parent and child links of a **physics::Joint** (p. 381).

Parameters

in	*_parent*	Parent link.
in	*_child*	Child link.
in	*_pose*	Pose containing **Joint** (p. 381) Anchor offset from child link.

10.66.4.35 void gazebo::physics::Joint::Load (LinkPtr _parent, LinkPtr _child, const math::Vector3 & _pos)

Set parent and child links of a **physics::Joint** (p. 381) and its anchor offset position.

This funciton is deprecated, use **Load(LinkPtr _parent, LinkPtr _child, const math::Pose &_pose)** (p. 394)

Parameters

in	*_parent*	Parent link.
in	*_child*	Child link.
in	*_pos*	**Joint** (p. 381) Anchor offset from child link.

10.66.4.36 virtual void gazebo::physics::Joint::Load (sdf::ElementPtr _sdf) [virtual]

Load **physics::Joint** (p. 381) from a SDF **sdf::Element** (p. 273).

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::Base** (p. 145).

10.66.4.37 virtual void gazebo::physics::Joint::Reset ( ) [virtual]

Reset the joint.

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

10.66.4.38 virtual void gazebo::physics::Joint::SetAnchor (int *_index*, const math::Vector3 & *_anchor*) [pure virtual]

Set the anchor point.

Parameters

in	*_index*	Indx of the axis.
in	*_anchor*	Anchor value.

10.66.4.39 void gazebo::physics::Joint::SetAngle (int *_index*, math::Angle *_angle*)

If the **Joint** (p. 381) is static, Gazebo stores the state of this **Joint** (p. 381) as a scalar inside the **Joint** (p. 381) class, so this call will NOT move the joint dynamically for a static **Model** (p. 489).

But if this **Model** (p. 489) is not static, then it is updated dynamically, all the conencted children **Link** (p. 418)'s are moved as a result of the **Joint** (p. 381) angle setting. Dynamic **Joint** (p. 381) angle update is accomplished by calling **JointController::SetJointPosition** (p. 400).

Parameters

in	*_index*	Index of the axis.
in	*_angle*	Angle to set the joint to.

10.66.4.40 virtual void gazebo::physics::Joint::SetAttribute (const std::string & *_key*, int *_index*, const boost::any & *_value*) [pure virtual]

Set a non-generic parameter for the joint.

replaces SetAttribute(Attribute, int, double)

Parameters

in	*_key*	String key.
in	*_index*	Index of the axis.
in	*_value*	Value of the attribute.

10.66.4.41 virtual void gazebo::physics::Joint::SetAxis (int *_index*, const math::Vector3 & *_axis*) [pure virtual]

Set the axis of rotation.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in world frame of axis direction (must have length greater than zero).

10.66.4.42 virtual void gazebo::physics::Joint::SetDamping (int *_index*, double *_damping*) [pure virtual]

Set the joint damping.

Parameters

in	*_index*	Index of the axis to set.
in	*_damping*	Damping value for the axis.

10.66.4.43 virtual void gazebo::physics::Joint::SetForce (int *_index*, double *_force*) [virtual]

Set the force applied to this **physics::Joint** (p. 381).

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

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value.

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

Set the high stop of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_angle*	High stop angle.

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

Set the low stop of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_angle*	Low stop angle.

10.66.4.46 virtual void gazebo::physics::Joint::SetMaxForce (int *_index*, double *_force*) [pure virtual]

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

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

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

10.66.4.47 void gazebo::physics::Joint::SetModel (ModelPtr _model)

Set the model this joint belongs too.

Parameters

in	*_model*	Pointer to a model.

10.66.4.48 void gazebo::physics::Joint::SetState (const JointState & _state)

Set the joint state.

Parameters

in	*_state*	**Joint** (p. 381) state

10.66.4.49 virtual void gazebo::physics::Joint::SetVelocity (int _index, double _vel) [pure virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

10.66.4.50 void gazebo::physics::Joint::Update ( ) [virtual]

Update the joint.

Reimplemented from **gazebo::physics::Base** (p. 148).

10.66.4.51 virtual void gazebo::physics::Joint::UpdateParameters (sdf::ElementPtr _sdf) [virtual]

Update the parameters using new sdf values.

Parameters

in	*_sdf*	SDF values to update from.

Reimplemented from **gazebo::physics::Base** (p. 148).

10.66.5 Member Data Documentation

10.66.5.1 LinkPtr gazebo::physics::Joint::anchorLink [protected]

Anchor link.

10.66.5.2 math::Vector3 gazebo::physics::Joint::anchorPos [protected]

Anchor pose.

This is the xyz offset of the joint frame from child frame specified in the parent link frame

10.66.5.3 math::Pose gazebo::physics::Joint::anchorPose [protected]

Anchor pose specified in SDF <joint><pose> tag.

AnchorPose is the transform from child link frame to joint frame specified in the child link frame. AnchorPos is more relevant in normal usage, but sometimes, we do need this (e.g. GetForceTorque and joint visualization).

10.66.5.4 gazebo::event::ConnectionPtr gazebo::physics::Joint::applyDamping [protected]

apply damping for adding viscous damping forces on updates

10.66.5.5 LinkPtr gazebo::physics::Joint::childLink [protected]

The first link this joint connects to.

10.66.5.6 double gazebo::physics::Joint::dampingCoefficient [protected]

joint dampingCoefficient

10.66.5.7 double gazebo::physics::Joint::effortLimit[2] [protected]

Store **Joint** (p. 381) effort limit as specified in SDF.

10.66.5.8 double gazebo::physics::Joint::forceApplied[2] [protected]

Save force applied by user This plus the joint feedback (joint contstraint forces) is the equivalent of simulated force torque sensor reading Allocate a 2 vector in case hinge2 joint is used.

10.66.5.9 double gazebo::physics::Joint::inertiaRatio[2] [protected]

Store **Joint** (p. 381) inertia ratio.

This is a measure of how well this model behaves using interative LCP solvers.

10.66.5.10 math::Angle gazebo::physics::Joint::lowerLimit[2] [protected]

Store **Joint** (p. 381) position lower limit as specified in SDF.

Referenced by GetLowerLimit().

10.66.5.11 ModelPtr gazebo::physics::Joint::model [protected]

Pointer to the parent model.

10.66.5.12 LinkPtr gazebo::physics::Joint::parentLink [protected]

The second link this joint connects to.

10.66.5.13 math::Angle gazebo::physics::Joint::upperLimit[2] [protected]

Store **Joint** (p. 381) position upper limit as specified in SDF.

Referenced by GetUpperLimit().

10.66.5.14 bool gazebo::physics::Joint::useCFMDamping [protected]

option to use CFM damping

10.66.5.15 double gazebo::physics::Joint::velocityLimit[2] [protected]

Store **Joint** (p. 381) velocity limit as specified in SDF.

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

- **Joint.hh**

10.67 gazebo::physics::JointController Class Reference

A (p. 111) class for manipulating **physics::Joint** (p. 381).

`#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. 381) by name.*
- void **SetJointPosition** (**JointPtr** _joint, double _position)
  *Set the positions of a **Joint** (p. 381) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 747) and radians for **HingeJoint** (p. 358), etc.*
- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)
  *Set the positions of a set of **Joint** (p. 381)'s.*
- void **Update** ()
  Update the joint control.

10.67.1 Detailed Description

A (p. 111) class for manipulating **physics::Joint** (p. 381).

10.67.2 Constructor & Destructor Documentation

10.67.2.1 gazebo::physics::JointController::JointController (ModelPtr _model) [explicit]

Constructor.

Parameters

in	*_model*	**Model** (p. 489) that uses this joint controller.

10.67.3 Member Function Documentation

10.67.3.1 void gazebo::physics::JointController::AddJoint (JointPtr _joint)

Add a joint to control.

Parameters

in	*_joint*	**Joint** (p. 381) to control.

10.67.3.2 void gazebo::physics::JointController::Reset ( )

Reset all commands.

10.67.3.3 void gazebo::physics::JointController::SetJointPosition (const std::string & _name, double _position)

Set the positions of a **Joint** (p. 381) by name.

See Also

JointController::SetJointPosition(JointPtr, double) (p. 400)

10.67.3.4 void gazebo::physics::JointController::SetJointPosition (JointPtr _joint, double _position)

Set the positions of a **Joint** (p. 381) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 747) and radians for **HingeJoint** (p. 358), etc.

Implementation: In order to change the position of a **Joint** (p. 381) inside a **Model** (p. 489), this call must recursively crawl through all the connected children **Link** (p. 418)'s in this **Model** (p. 489), and update each **Link** (p. 418) Pose affected by this **Joint** (p. 381) angle update. Warning: There is no constraint satisfaction being done here, traversal through the kinematic graph has unexpected behavior if you try to set the joint position of a link inside a loop structure.

Parameters

in	*_joint*	**Joint** (p. 381) to set.
in	*_position*	Position of the joint.

10.67.3.5 void gazebo::physics::JointController::SetJointPositions (const std::map< std::string, double > & _jointPositions)

Set the positions of a set of **Joint** (p. 381)'s.

See Also

JointController::SetJointPosition(JointPtr, double) (p. 400)

10.67.3.6 void gazebo::physics::JointController::Update ( )

Update the joint control.

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

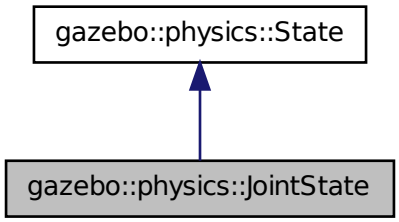
- **JointController.hh**

10.68 gazebo::physics::JointState Class Reference

keeps track of state of a **physics::Joint** (p. 381)

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::JointState:



Public Member Functions

- **JointState** ()
  Default constructor.
- **JointState** (**JointPtr** _joint)
  Constructor.
- **JointState** (const **sdf::ElementPtr** _sdf)
  Constructor.
- virtual ~**JointState** ()
  Destructor.
- void **FillSDF** (**sdf::ElementPtr** _sdf)
  Populate a state SDF element with data from the object.
- **math::Angle GetAngle** (unsigned int _axis) const
  Get the joint angle.
- unsigned int **GetAngleCount** () const

Get the number of angles.

- const std::vector< **math::Angle** > & **GetAngles** () const

  Get the angles.

- bool **IsZero** () const

  Return true if the values in the state are zero.

- virtual void **Load** (const **sdf::ElementPtr** _elem)

  Load state from SDF element.

- **JointState operator+** (const **JointState** &_state) const

  Addition operator.

- **JointState operator-** (const **JointState** &_state) const

  Subtraction operator.

- **JointState** & **operator=** (const **JointState** &_state)

  Assignment operator.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::JointState** &_state)

  Stream insertion operator.

Additional Inherited Members

10.68.1 Detailed Description

keeps track of state of a **physics::Joint** (p. 381)

10.68.2 Constructor & Destructor Documentation

10.68.2.1 gazebo::physics::JointState::JointState ( )

Default constructor.

10.68.2.2 gazebo::physics::JointState::JointState (**JointPtr** *_joint*) [explicit]

Constructor.

Parameters

in	*_joint*	**Joint** (p. 381) to get the state of.

10.68.2.3 gazebo::physics::JointState::JointState (const **sdf::ElementPtr** *_sdf*) [explicit]

Constructor.

Build a **JointState** (p. 401) from SDF data

Parameters

in	*_sdf*	SDF data to load a joint state from.

10.68.2.4 virtual gazebo::physics::JointState::~JointState ( ) [virtual]

Destructor.

10.68.3 Member Function Documentation

10.68.3.1 void gazebo::physics::JointState::FillSDF (sdf::ElementPtr *_sdf*)

Populate a state SDF element with data from the object.

Parameters

out	*_sdf*	SDF element to populate.

10.68.3.2 math::Angle gazebo::physics::JointState::GetAngle (unsigned int *_axis*) const

Get the joint angle.

Parameters

in	*_axis*	The axis index.

Returns

Angle of the axis.

Exceptions

common::Exception *(p. 318)*	When _axis is invalid.

10.68.3.3 unsigned int gazebo::physics::JointState::GetAngleCount ( ) const

Get the number of angles.

Returns

The number of angles.

10.68.3.4 const std::vector<math::Angle>& gazebo::physics::JointState::GetAngles ( ) const

Get the angles.

Returns

Vector of angles.

10.68.3.5 bool gazebo::physics::JointState::IsZero ( ) const

Return true if the values in the state are zero.

Returns

True if the values in the state are zero.

10.68.3.6 virtual void gazebo::physics::JointState::Load (const sdf::ElementPtr _elem) [virtual]

Load state from SDF element.

Parameters

in	*_elem*	SDf values to load from.

Reimplemented from **gazebo::physics::State** (p. 761).

10.68.3.7 JointState gazebo::physics::JointState::operator+ (const JointState & _state) const

Addition operator.

Parameters

in	*_pt*	**A** (p. 111) state to add.

Returns

The resulting state.

10.68.3.8 JointState gazebo::physics::JointState::operator- (const JointState & _state) const

Subtraction operator.

Parameters

in	*_pt*	**A** (p. 111) state to substract.

Returns

The resulting state.

10.68.3.9 JointState& gazebo::physics::JointState::operator= (const JointState & _state)

Assignment operator.

Parameters

in	*_state*	**State** (p. 758) value

Returns

this

10.68.4 Friends And Related Function Documentation

10.68.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::physics::JointState & *_state*) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream.
in	*_state*	**Joint** (p. 381) state to output.

Returns

The stream.

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

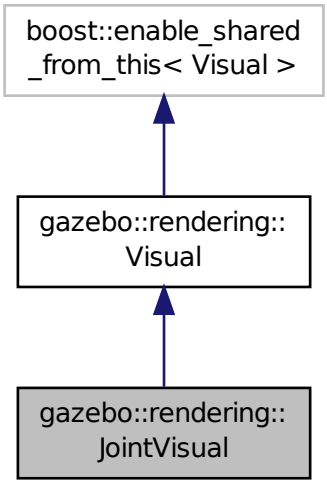
- **JointState.hh**

10.69 gazebo::rendering::JointVisual Class Reference

Visualization for joints.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::JointVisual:



Public Member Functions

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

  Constructor.
- virtual ~**JointVisual** ()

  Destructor.
- void **Load** (ConstJointPtr &_msg)

  Load the visual based on a message.

Additional Inherited Members

10.69.1 Detailed Description

Visualization for joints.

10.69.2 Constructor & Destructor Documentation

10.69.2.1 gazebo::rendering::JointVisual::JointVisual (const std::string & *_name*, VisualPtr *_vis*)

Constructor.

Parameters

in	*_name*	Name of the visual
in	*_vis*	Pointer to the parent visual

10.69.2.2 virtual gazebo::rendering::JointVisual::~JointVisual ( ) [virtual]

Destructor.

10.69.3 Member Function Documentation

10.69.3.1 void gazebo::rendering::JointVisual::Load (ConstJointPtr & _msg)

Load the visual based on a message.

Parameters

in	*_msg*	Joint message

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

- **JointVisual.hh**

10.70 gazebo::physics::JointWrench Class Reference

Wrench information from a joint.

`#include <physics/physics.hh>`

Public Member Functions

- **JointWrench** & **operator+** (const **JointWrench** &_wrench)
  Operator +.
- **JointWrench** & **operator-** (const **JointWrench** &_wrench)
  Operator -.
- **JointWrench** & **operator=** (const **JointWrench** &_wrench)
  Operator =.

Public Attributes

- **math::Vector3 body1Force**
  Force on the first link.
- **math::Vector3 body1Torque**
  Torque on the first link.
- **math::Vector3 body2Force**
  Force on the second link.
- **math::Vector3 body2Torque**
  Torque on the second link.

10.70.1 Detailed Description

Wrench information from a joint.

These are forces and torques on parent and child Links, relative to the **Link** (p. 418)'s center of mass.

10.70.2 Member Function Documentation

10.70.2.1 JointWrench& gazebo::physics::JointWrench::operator+ (const JointWrench & _wrench) [inline]

Operator +.

Parameters

in	*_wrench*	**Joint** (p. 381) wrench to add

Returns

∗this

References body1Force, body1Torque, body2Force, and body2Torque.

10.70.2.2 JointWrench& gazebo::physics::JointWrench::operator- (const JointWrench & _wrench) [inline]

Operator -.

Parameters

in	*_wrench*	**Joint** (p. 381) wrench to subtract

Returns

∗this

References body1Force, body1Torque, body2Force, and body2Torque.

10.70.2.3 JointWrench& gazebo::physics::JointWrench::operator= (const JointWrench & _wrench) [inline]

Operator =.

Parameters

in	*_wrench*	**Joint** (p. 381) wrench to set from.

Returns

∗this

References body1Force, body1Torque, body2Force, and body2Torque.

10.70.3 Member Data Documentation

10.70.3.1 math::Vector3 gazebo::physics::JointWrench::body1Force

Force on the first link.

Referenced by operator+(), operator-(), and operator=().

10.70.3.2 math::Vector3 gazebo::physics::JointWrench::body1Torque

Torque on the first link.

Referenced by operator+(), operator-(), and operator=().

10.70.3.3 math::Vector3 gazebo::physics::JointWrench::body2Force

Force on the second link.

Referenced by operator+(), operator-(), and operator=().

10.70.3.4 math::Vector3 gazebo::physics::JointWrench::body2Torque

Torque on the second link.

Referenced by operator+(), operator-(), and operator=().

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

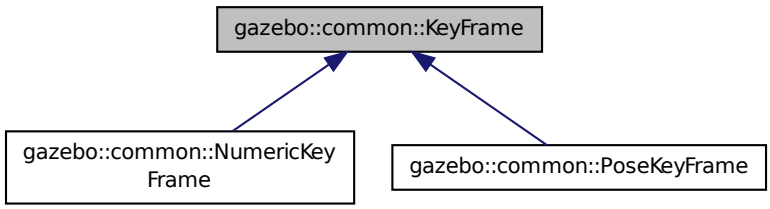
- **JointWrench.hh**

10.71 gazebo::common::KeyFrame Class Reference

A (p. 111) key frame in an animation.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::KeyFrame:



Public Member Functions

- **KeyFrame** (double _time)

  Constructor.
- virtual ~**KeyFrame** ()

  Destructor.
- double **GetTime** () const

  Get the time of the keyframe.

Protected Attributes

- double **time**

  time of key frame

10.71.1 Detailed Description

A (p. 111) key frame in an animation.

10.71.2 Constructor & Destructor Documentation

10.71.2.1 gazebo::common::KeyFrame::KeyFrame (double *_time*)

Constructor.

Parameters

in	*_time*	**Time** (p. 791) of the keyframe in seconds

10.71.2.2 virtual gazebo::common::KeyFrame::~KeyFrame ( ) [virtual]

Destructor.

10.71.3 Member Function Documentation

10.71.3.1 double gazebo::common::KeyFrame::GetTime ( ) const

Get the time of the keyframe.

Returns

the time

10.71.4 Member Data Documentation

10.71.4.1 double gazebo::common::KeyFrame::time [protected]

time of key frame

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

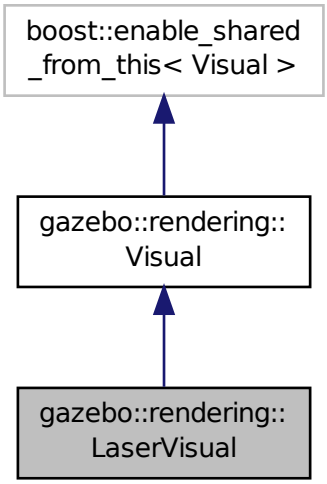
- **KeyFrame.hh**

10.72 gazebo::rendering::LaserVisual Class Reference

Visualization for laser data.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::LaserVisual:



Public Member Functions

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

  Constructor.
- virtual ~**LaserVisual** ()

  Destructor.
- virtual void **SetEmissive** (const **common::Color** &_color)

  Documentation inherited from parent.

Additional Inherited Members

10.72.1 Detailed Description

Visualization for laser data.

10.72.2 Constructor & Destructor Documentation

10.72.2.1 gazebo::rendering::LaserVisual::LaserVisual (const std::string & *_name,* VisualPtr *_vis,* const std::string & *_topicName*)

Constructor.

Parameters

in	*_name*	Name of the visual.
in	*_vis*	Pointer to the parent **Visual** (p. 885).
in	*_topicName*	Name of the topic that has laser data.

10.72.2.2 virtual gazebo::rendering::LaserVisual::~LaserVisual ( ) [virtual]

Destructor.

10.72.3 Member Function Documentation

10.72.3.1 virtual void gazebo::rendering::LaserVisual::SetEmissive (const common::Color & *_color*) [virtual]

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 899).

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

- **LaserVisual.hh**

10.73 gazebo::rendering::Light Class Reference

A (p. 111) light source.

`#include <rendering/rendering.hh>`

Public Member Functions

- **Light** (**ScenePtr** _scene)
  Constructor.
- virtual ~**Light** ()
  Destructor.
- void **FillMsg** (msgs::Light &_msg) const
  Fill the contents of a light message.
- **common::Color GetDiffuseColor** () const
  Get the diffuse color.
- **math::Vector3 GetDirection** () const
  Get the direction.
- std::string **GetName** () const
  Get the name of the visual.
- **math::Vector3 GetPosition** () const
  Get the position of the light.
- **common::Color GetSpecularColor** () const
  Get the specular color.
- std::string **GetType** () const
  Get the type of the light.

- void **Load** (**sdf::ElementPtr** _sdf)

  Load the light using a set of SDF parameters.
- void **Load** ()

  Load the light using default parameters.
- void **LoadFromMsg** (ConstLightPtr &_msg)

  Load from a light message.
- void **SetAttenuation** (double _constant, double _linear, double _quadratic)

  Set the attenuation.
- void **SetCastShadows** (const bool &_cast)

  Set cast shadows.
- void **SetDiffuseColor** (const **common::Color** &_color)

  Set the diffuse color.
- void **SetDirection** (const **math::Vector3** &_dir)

  Set the direction.
- void **SetLightType** (const std::string &_type)

  Set the light type.
- void **SetName** (const std::string &_name)

  Set the name of the visual.
- void **SetPosition** (const **math::Vector3** &_p)

  Set the position of the light.
- void **SetRange** (const double &_range)

  Set the range.
- virtual bool **SetSelected** (bool _s)

  Set whether this entity has been selected by the user through the gui.
- void **SetSpecularColor** (const **common::Color** &_color)

  Set the specular color.
- void **SetSpotFalloff** (const double &_value)

  Set the spot light falloff.
- void **SetSpotInnerAngle** (const double &_angle)

  Set the spot light inner angle.
- void **SetSpotOuterAngle** (const double &_angle)

  Set the spot light outer angle.
- void **ShowVisual** (bool _s)

  Set whether to show the visual.
- void **ToggleShowVisual** ()
- void **UpdateFromMsg** (ConstLightPtr &_msg)

  Update a light source from a message.

Protected Member Functions

- virtual void **OnPoseChange** ()

  On pose change callback.

10.73.1 Detailed Description

A (p. 111) light source.

There are three types of lights: Point, Spot, and Directional. This class encapsulates all three. Point lights are light light bulbs, spot lights project a cone of light, and directional lights are light sun light.

10.73.2 Constructor & Destructor Documentation

10.73.2.1 gazebo::rendering::Light::Light (ScenePtr _scene)

Constructor.

Parameters

in	*_scene*	Pointer to the scene that contains the **Light** (p. 412).

10.73.2.2 virtual gazebo::rendering::Light::~Light ( ) [virtual]

Destructor.

10.73.3 Member Function Documentation

10.73.3.1 void gazebo::rendering::Light::FillMsg (msgs::Light & _msg) const

Fill the contents of a light message.

Parameters

out	*_msg*	Message to fill.

10.73.3.2 common::Color gazebo::rendering::Light::GetDiffuseColor ( ) const

Get the diffuse color.

Returns

The light's diffuse color.

10.73.3.3 math::Vector3 gazebo::rendering::Light::GetDirection ( ) const

Get the direction.

Returns

The light's direction.

10.73.3.4 std::string gazebo::rendering::Light::GetName ( ) const

Get the name of the visual.

Returns

The light's name.

10.73.3.5 math::Vector3 gazebo::rendering::Light::GetPosition ( ) const

Get the position of the light.

Returns

The position of the light

10.73.3.6 common::Color gazebo::rendering::Light::GetSpecularColor ( ) const

Get the specular color.

Returns

The specular color

10.73.3.7 std::string gazebo::rendering::Light::GetType ( ) const

Get the type of the light.

Returns

The light type: "point", "spot", "directional".

10.73.3.8 void gazebo::rendering::Light::Load (sdf::ElementPtr *_sdf*)

Load the light using a set of SDF parameters.

Parameters

in	*_sdf*	Pointer to the SDF containing the **Light** (p. 412) description.

10.73.3.9 void gazebo::rendering::Light::Load ( )

Load the light using default parameters.

10.73.3.10 void gazebo::rendering::Light::LoadFromMsg (ConstLightPtr & *_msg*)

Load from a light message.

Parameters

in	*_msg*	Containing the light information.

10.73.3.11 virtual void gazebo::rendering::Light::OnPoseChange ( ) [inline], [protected], [virtual]

On pose change callback.

10.73.3.12 void gazebo::rendering::Light::SetAttenuation (double *_constant,* double *_linear,* double *_quadratic*)

Set the attenuation.

Parameters

in	*_contant*	Constant attenuation
in	*_linear*	Linear attenuation
in	*_quadratic*	Quadratic attenuation

10.73.3.13 void gazebo::rendering::Light::SetCastShadows (const bool & *_cast*)

Set cast shadows.

Parameters

in	*_cast*	Set to true to cast shadows.

10.73.3.14 void gazebo::rendering::Light::SetDiffuseColor (const common::Color & *_color*)

Set the diffuse color.

Parameters

in	*_color*	**Light** (p. 412) diffuse color.

10.73.3.15 void gazebo::rendering::Light::SetDirection (const math::Vector3 & *_dir*)

Set the direction.

Parameters

in	*_dir*	Set the light's direction. Only applicable to spot and directional lights.

10.73.3.16 void gazebo::rendering::Light::SetLightType (const std::string & *_type*)

Set the light type.

Parameters

in	*_type*	The light type: "point", "spot", "directional"

10.73.3.17 void gazebo::rendering::Light::SetName (const std::string & *_name*)

Set the name of the visual.

Parameters

in	*_name*	Name of the light source.

10.73.3.18 void gazebo::rendering::Light::SetPosition (const math::Vector3 & _p)

Set the position of the light.

Parameters

in	*_p*	New position for the light

10.73.3.19 void gazebo::rendering::Light::SetRange (const double & _range)

Set the range.

Parameters

in	*_range*	Rage of the light in meters.

10.73.3.20 virtual bool gazebo::rendering::Light::SetSelected (bool _s) [virtual]

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

Parameters

in	*_s*	Set to True when the light is selected by the user.

10.73.3.21 void gazebo::rendering::Light::SetSpecularColor (const common::Color & _color)

Set the specular color.

Parameters

in	*_color*	The specular color

10.73.3.22 void gazebo::rendering::Light::SetSpotFalloff (const double & _value)

Set the spot light falloff.

Parameters

in	*_value*	Falloff value

10.73.3.23 void gazebo::rendering::Light::SetSpotInnerAngle (const double & _angle)

Set the spot light inner angle.

Parameters

in	*_angle*	Inner angle in radians

10.73.3.24 void gazebo::rendering::Light::SetSpotOuterAngle (const double & _angle)

Set the spot light outer angle.

Parameters

in	*_angle*	Outer angle in radians

10.73.3.25 void gazebo::rendering::Light::ShowVisual (bool _s)

Set whether to show the visual.

Parameters

in	*_s*	Set to true to draw a representation of the light.

10.73.3.26 void gazebo::rendering::Light::ToggleShowVisual ( )

10.73.3.27 void gazebo::rendering::Light::UpdateFromMsg (ConstLightPtr & _msg)

Update a light source from a message.

Parameters

in	*_msg*	**Light** (p. 412) message to update from

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

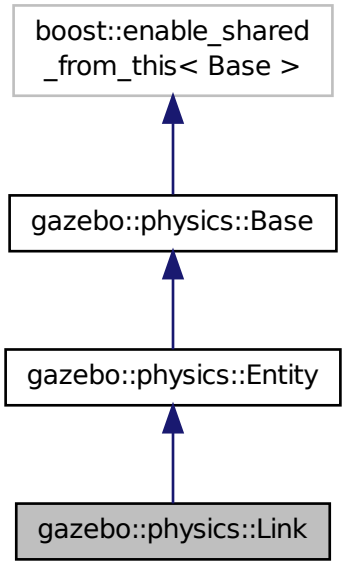
- **Light.hh**

10.74 gazebo::physics::Link Class Reference

Link (p. 418) 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. 418) as a parent **Link** (p. 418).*
- 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. 418) as a child **Link** (p. 418).*
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)=0

  Add a force to the body, components are relative to the body's own frame of reference.
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)=0

  Add a torque to the body, components are relative to the body's own frame of reference.
- virtual void **AddTorque** (const **math::Vector3** &_torque)=0

  Add a torque to the body.

- void **AttachStaticModel** (**ModelPtr** &_model, const **math::Pose** &_offset)

  Attach a static model to this link.
- template<typename T >
  event::ConnectionPtr ConnectEnabled (T _subscriber)

  Connect to the add entity signal.
- void **DetachAllStaticModels** ()

  Detach all static models from this link.
- void **DetachStaticModel** (const std::string &_modelName)

  Detach a static model from this link.
- void **DisconnectEnabled** (**event::ConnectionPtr** &_conn)

  Disconnect to the add entity signal.
- void **FillMsg** (msgs::Link &_msg)

  Fill a link message.
- void **Fini** ()

  Finalize the body.
- double **GetAngularDamping** () const

  Get the angular damping factor.
- virtual **math::Box GetBoundingBox** () const

  Get the bounding box for the link and all the child elements.
- **Link_V GetChildJointsLinks** () const

  Returns a vector of children Links connected by joints.
- **CollisionPtr GetCollision** (const std::string &_name)

  Get a child collision by name.
- **CollisionPtr GetCollision** (unsigned int _index) const

  Get a child collision by index.
- **CollisionPtr GetCollisionById** (unsigned int _id) const

  This is an internal function
- **Collision_V GetCollisions** () const

  Get all the child collisions.
- virtual bool **GetEnabled** () const =0

  Get whether this body is enabled in the physics engine.
- virtual bool **GetGravityMode** () const =0

  Get the gravity mode.
- **InertialPtr GetInertial** () const

  Get the inertia of the link.
- virtual bool **GetKinematic** () const

  Implement this function.
- double **GetLinearDamping** () const

  Get the linear damping factor.
- **ModelPtr GetModel** () const

  Get the model that this body belongs to.
- **Link_V GetParentJointsLinks** () const

  Returns a vector of parent Links connected by joints.
- **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the body.
- **math::Vector3 GetRelativeAngularVel** () const

  Get the angular velocity of the body.

- **math::Vector3 GetRelativeForce** () const

  Get the force applied to the body.
- **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the body.
- **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the body.
- **math::Vector3 GetRelativeTorque** () const

  Get the torque applied to the body.
- bool **GetSelfCollide** ()

  Get Self-Collision Flag, if this is true, this body will collide with other bodies even if they share the same parent.
- unsigned int **GetSensorCount** () const

  Get sensor count.
- std::string **GetSensorName** (unsigned int _index) const

  Get sensor name.
- **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the body in the world frame.
- virtual **math::Vector3 GetWorldCoGLinearVel** () const =0

  Get the linear velocity at the body's center of gravity in the world frame.
- **math::Pose GetWorldCoGPose** () const

  Get the pose of the body's center of gravity in the world coordinate frame.
- virtual **math::Vector3 GetWorldForce** () const =0

  Get the force applied to the body in the world frame.
- **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the body in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset=**math::Vector3**(0, 0, 0)) const =0

  Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset, const **math::Quaternion** &_q) const =0

  Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.
- virtual **math::Vector3 GetWorldTorque** () const =0

  Get the torque applied to the body in the world frame.
- virtual void **Init** ()

  Initialize the body.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the body based on an SDF element.
- virtual void **OnPoseChange** ()

  This function is called when the entity's (or one of its parents) pose of the parent has changed.
- void **ProcessMsg** (const msgs::Link &_msg)

  Update parameters from a message.
- void **RemoveChildJoint** (**JointPtr** _joint) **GAZEBO_DEPRECATED**(1.5)

  *Remove Joints that have this **Link** (p. 418) as a parent **Link** (p. 418).*
- void **RemoveChildJoint** (const std::string &_jointName)

  *Remove Joints that have this **Link** (p. 418) as a parent **Link** (p. 418).*
- void **RemoveParentJoint** (**JointPtr** _joint) **GAZEBO_DEPRECATED**(1.5)

  *Remove Joints that have this **Link** (p. 418) as a child **Link** (p. 418).*
- void **RemoveParentJoint** (const std::string &_jointName)

  *Remove Joints that have this **Link** (p. 418) as a child **Link** (p. 418).*

- void **Reset** ()

  Reset the link.
- void **ResetPhysicsStates** ()

  Reset the link.
- void **SetAngularAccel** (const **math::Vector3** &_accel)

  Set the angular acceleration of the body.
- virtual void **SetAngularDamping** (double _damping)=0

  Set the angular damping factor.
- virtual void **SetAngularVel** (const **math::Vector3** &_vel)=0

  Set the angular velocity of the body.
- virtual void **SetAutoDisable** (bool _disable)=0

  Allow the link to auto disable.
- void **SetCollideMode** (const std::string &_mode)

  Set the collide mode of the body.
- virtual void **SetEnabled** (bool _enable) const =0

  Set whether this body is enabled.
- virtual void **SetForce** (const **math::Vector3** &_force)=0

  Set the force applied to the body.
- virtual void **SetGravityMode** (bool _mode)=0

  Set whether gravity affects this body.
- void **SetInertial** (const **InertialPtr** &_inertial)

  Set the mass of the link.
- virtual void **SetKinematic** (const bool &_kinematic)

  Implement this function.
- void **SetLaserRetro** (float _retro)

  Set the laser retro reflectiveness.
- void **SetLinearAccel** (const **math::Vector3** &_accel)

  Set the linear acceleration of the body.
- virtual void **SetLinearDamping** (double _damping)=0

  Set the linear damping factor.
- virtual void **SetLinearVel** (const **math::Vector3** &_vel)=0

  Set the linear velocity of the body.
- virtual bool **SetSelected** (bool _set)

  Set whether this entity has been selected by the user through the gui.
- virtual void **SetSelfCollide** (bool _collide)=0

  Set whether this body will collide with others in the model.
- void **SetState** (const **LinkState** &_state)

  Set the current link state.
- virtual void **SetTorque** (const **math::Vector3** &_torque)=0

  Set the torque applied to the body.
- virtual void **Update** ()

  Update the body.
- virtual void **UpdateMass** ()

  Update the mass matrix.
- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  Update the parameters using new sdf values.
- virtual void **UpdateSurface** ()

  Update surface parameters.

Protected Attributes

- **math::Vector3 angularAccel**

  Angular acceleration.
- std::vector< **math::Pose** > **attachedModelsOffset**

  Offsets for the attached models.
- std::vector< std::string > **cgVisuals**

  Center of gravity visual elements.
- **InertialPtr inertial**

  Inertial *(p. 369) properties.*
- **math::Vector3 linearAccel**

  Linear acceleration.
- std::vector< std::string > **visuals**

  Link *(p. 418) visual elements.*

Additional Inherited Members

10.74.1 Detailed Description

Link (p. 418) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

10.74.2 Constructor & Destructor Documentation

10.74.2.1 gazebo::physics::Link::Link (EntityPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of this link.

10.74.2.2 virtual gazebo::physics::Link::~Link ( ) [virtual]

Destructor.

10.74.3 Member Function Documentation

10.74.3.1 void gazebo::physics::Link::AddChildJoint (JointPtr *_joint*)

Joints that have this **Link** (p. 418) as a parent **Link** (p. 418).

Parameters

in	*_joint*	**Joint** (p. 381) that is a child of this link.

10.74.3.2 virtual void gazebo::physics::Link::AddForce (const math::Vector3 & _force) [pure virtual]

Add a force to the body.

Parameters

in	*_force*	Force to add.

10.74.3.3 virtual void gazebo::physics::Link::AddForceAtRelativePosition (const math::Vector3 & _force, const math::Vector3 & _relPos) [pure virtual]

Add a force to the body at position expressed to the body's own frame of reference.

Parameters

in	*_force*	Force to add.
in	*_relPos*	Position on the link to add the force.

10.74.3.4 virtual void gazebo::physics::Link::AddForceAtWorldPosition (const math::Vector3 & _force, const math::Vector3 & _pos) [pure virtual]

Add a force to the body using a global position.

Parameters

in	*_force*	Force to add.
in	*_pos*	Position in global coord frame to add the force.

10.74.3.5 void gazebo::physics::Link::AddParentJoint (JointPtr _joint)

Joints that have this **Link** (p. 418) as a child **Link** (p. 418).

Parameters

in	*_joint*	**Joint** (p. 381) that is a parent of this link.

10.74.3.6 virtual void gazebo::physics::Link::AddRelativeForce (const math::Vector3 & _force) [pure virtual]

Add a force to the body, components are relative to the body's own frame of reference.

Parameters

in	*_force*	Force to add.

10.74.3.7 virtual void gazebo::physics::Link::AddRelativeTorque (const math::Vector3 & _torque) [pure virtual]

Add a torque to the body, components are relative to the body's own frame of reference.

Parameters

in	*_torque*	Torque value to add.

10.74.3.8 virtual void gazebo::physics::Link::AddTorque (const math::Vector3 & *_torque*) [pure virtual]

Add a torque to the body.

Parameters

in	*_torque*	Torque value to add to the link.

10.74.3.9 void gazebo::physics::Link::AttachStaticModel (ModelPtr & *_model,* const math::Pose & *_offset*)

Attach a static model to this link.

Parameters

in	*_model*	Pointer to a static model.
in	*_offset*	Pose relative to this link to place the model.

10.74.3.10 template<typename T > event::ConnectionPtr gazebo::physics::Link::ConnectEnabled (T *_subscriber*) [inline]

Connect to the add entity signal.

Parameters

in	*_subscriber*	Subsciber callback function.

Returns

Pointer to the connection, which must be kept in scope.

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

10.74.3.11 void gazebo::physics::Link::DetachAllStaticModels ( )

Detach all static models from this link.

10.74.3.12 void gazebo::physics::Link::DetachStaticModel (const std::string & *_modelName*)

Detach a static model from this link.

Parameters

in	*_modelName*	Name of an attached model to detach.

10.74.3.13 void gazebo::physics::Link::DisconnectEnabled (event::ConnectionPtr & *_conn*) [inline]

Disconnect to the add entity signal.

Parameters

in	*_conn*	Connection pointer to disconnect.

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

10.74.3.14 void gazebo::physics::Link::FillMsg (msgs::Link & *_msg*)

Fill a link message.

Parameters

out	*_msg*	Message to fill

10.74.3.15 void gazebo::physics::Link::Fini ( ) [virtual]

Finalize the body.

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

10.74.3.16 double gazebo::physics::Link::GetAngularDamping ( ) const

Get the angular damping factor.

Returns

Angular damping.

10.74.3.17 virtual math::Box gazebo::physics::Link::GetBoundingBox ( ) const [virtual]

Get the bounding box for the link and all the child elements.

Returns

The link's bounding box.

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

10.74.3.18 Link_V gazebo::physics::Link::GetChildJointsLinks ( ) const

Returns a vector of children Links connected by joints.

Returns

A (p. 111) vector of children Links connected by joints.

10.74.3.19 CollisionPtr gazebo::physics::Link::GetCollision (const std::string & _name)

Get a child collision by name.

Parameters

in	*_name*	Name of the collision object.

Returns

Pointer to the collision, NULL if the name was not found.

10.74.3.20 CollisionPtr gazebo::physics::Link::GetCollision (unsigned int _index) const

Get a child collision by index.

Parameters

in	*_index*	Index of the collision object.

Returns

Pointer to the collision, NULL if the name was not found.

10.74.3.21 CollisionPtr gazebo::physics::Link::GetCollisionById (unsigned int _id) const

This is an internal function

Get a collision by id.

Parameters

in	*_id*	Id of the collision object to find.

Returns

Pointer to the collision, NULL if the id is invalid.

10.74.3.22 Collision_V gazebo::physics::Link::GetCollisions ( ) const

Get all the child collisions.

Returns

A (p. 111) std::vector of all the child collisions.

10.74.3.23 virtual bool gazebo::physics::Link::GetEnabled ( ) const [pure virtual]

Get whether this body is enabled in the physics engine.

Returns

True if the link is enabled.

10.74.3.24 virtual bool gazebo::physics::Link::GetGravityMode ( ) const [pure virtual]

Get the gravity mode.

Returns

True if gravity is enabled.

10.74.3.25 InertialPtr gazebo::physics::Link::GetInertial ( ) const [inline]

Get the inertia of the link.

Returns

Inertia of the link.

References inertial.

10.74.3.26 virtual bool gazebo::physics::Link::GetKinematic ( ) const [inline], [virtual]

Implement this function.

Get whether this body is in the kinematic state.

Returns

True if the link is kinematic only.

10.74.3.27 double gazebo::physics::Link::GetLinearDamping ( ) const

Get the linear damping factor.

Returns

Linear damping.

10.74.3.28 ModelPtr gazebo::physics::Link::GetModel ( ) const

Get the model that this body belongs to.

Returns

Model (p. 489) that this body belongs to.

10.74.3.29 Link_V gazebo::physics::Link::GetParentJointsLinks ( ) const

Returns a vector of parent Links connected by joints.

Returns

Vector of parent Links connected by joints.

10.74.3.30 math::Vector3 gazebo::physics::Link::GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the body.

Returns

Angular acceleration of the body.

Reimplemented from **gazebo::physics::Entity** (p. 286).

10.74.3.31 math::Vector3 gazebo::physics::Link::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the body.

Returns

Angular velocity of the body.

Reimplemented from **gazebo::physics::Entity** (p. 286).

10.74.3.32 math::Vector3 gazebo::physics::Link::GetRelativeForce ( ) const

Get the force applied to the body.

Returns

Force applied to the body.

10.74.3.33 math::Vector3 gazebo::physics::Link::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the body.

Returns

Linear acceleration of the body.

Reimplemented from **gazebo::physics::Entity** (p. 286).

10.74.3.34 math::Vector3 gazebo::physics::Link::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the body.

Returns

Linear velocity of the body.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.74.3.35 math::Vector3 gazebo::physics::Link::GetRelativeTorque ( ) const

Get the torque applied to the body.

Returns

Torque applied to the body.

10.74.3.36 bool gazebo::physics::Link::GetSelfCollide ( )

Get Self-Collision Flag, if this is true, this body will collide with other bodies even if they share the same parent.

Returns

True if self collision is enabled.

10.74.3.37 unsigned int gazebo::physics::Link::GetSensorCount ( ) const

Get sensor count.

This will return the number of sensors created by the link when it was loaded. This function is commonly used with **Link::GetSensorName** (p. 430).

Returns

The number of sensors created by the link.

10.74.3.38 std::string gazebo::physics::Link::GetSensorName (unsigned int *_index*) const

Get sensor name.

Get the name of a sensor based on an index. The index should be in the range of 0...**Link::GetSensorCount()** (p. 430).

Note

A (p. 111) **Link** (p. 418) does not manage or maintain a pointer to a **sensors::Sensor** (p. 698). Access to a Sensor object is accomplished through the **sensors::SensorManager** (p. 709). This was done to separate the physics engine from the sensor engine.

Parameters

in	*_index*	Index of the sensor name.

Returns

The name of the sensor, or empty string if the index is out of bounds.

10.74.3.39 math::Vector3 gazebo::physics::Link::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the body in the world frame.

Returns

Angular acceleration of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.74.3.40 virtual math::Vector3 gazebo::physics::Link::GetWorldCoGLinearVel ( ) const [pure virtual]

Get the linear velocity at the body's center of gravity in the world frame.

Returns

Linear velocity at the body's center of gravity in the world frame.

10.74.3.41 math::Pose gazebo::physics::Link::GetWorldCoGPose ( ) const

Get the pose of the body's center of gravity in the world coordinate frame.

Returns

Pose of the body's center of gravity in the world coordinate frame.

10.74.3.42 virtual math::Vector3 gazebo::physics::Link::GetWorldForce ( ) const [pure virtual]

Get the force applied to the body in the world frame.

Returns

Force applied to the body in the world frame.

10.74.3.43 math::Vector3 gazebo::physics::Link::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the body in the world frame.

Returns

Linear acceleration of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.74.3.44 virtual math::Vector3 gazebo::physics::Link::GetWorldLinearVel (const math::Vector3 & *_offset* = math::Vector3(0, 0, 0)) const [pure virtual]

Get the linear velocity of a point on the body in the world frame, using an offset expressed in a body-fixed frame.

If no offset is given, the velocity at the origin of the **Link** (p. 418) frame will be returned.

Parameters

in	*_offset*	Offset of the point from the origin of the **Link** (p. 418) frame, expressed in the body-fixed frame.

Returns

Linear velocity of the point on the body

10.74.3.45 virtual math::Vector3 gazebo::physics::Link::GetWorldLinearVel (const math::Vector3 & *_offset*, const math::Quaternion & *_q*) const [pure virtual]

Get the linear velocity of a point on the body in the world frame, using an offset expressed in an arbitrary frame.

Parameters

in	*_offset*	Offset from the origin of the link frame expressed in a frame defined by _q.
in	*_q*	Describes the rotation of a reference frame relative to the world reference frame.

Returns

Linear velocity of the point on the body in the world frame.

10.74.3.46 virtual math::Vector3 gazebo::physics::Link::GetWorldTorque ( ) const [pure virtual]

Get the torque applied to the body in the world frame.

Returns

Torque applied to the body in the world frame.

10.74.3.47 virtual void gazebo::physics::Link::Init ( ) [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Base** (p. 145).

10.74.3.48 virtual void gazebo::physics::Link::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the body based on an SDF element.

Parameters

in	*_sdf*	SDF parameters.

Reimplemented from **gazebo::physics::Entity** (p. 288).

10.74.3.49 virtual void gazebo::physics::Link::OnPoseChange ( ) [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implements **gazebo::physics::Entity** (p. 289).

10.74.3.50 void gazebo::physics::Link::ProcessMsg (const msgs::Link & _msg)

Update parameters from a message.

Parameters

in	*_msg*	Message to read.

10.74.3.51 void gazebo::physics::Link::RemoveChildJoint (JointPtr _joint)

Remove Joints that have this **Link** (p. 418) as a parent **Link** (p. 418).

Parameters

in	*_joint*	**Joint** (p. 381) that is a child of this link.

10.74.3.52 void gazebo::physics::Link::RemoveChildJoint (const std::string & _jointName)

Remove Joints that have this **Link** (p. 418) as a parent **Link** (p. 418).

Parameters

in	*_jointName*	Child **Joint** (p. 381) name.

10.74.3.53 void gazebo::physics::Link::RemoveParentJoint (JointPtr _joint)

Remove Joints that have this **Link** (p. 418) as a child **Link** (p. 418).

Parameters

in	*_joint*	**Joint** (p. 381) that is a parent of this link.

10.74.3.54 void gazebo::physics::Link::RemoveParentJoint (const std::string & _jointName)

Remove Joints that have this **Link** (p. 418) as a child **Link** (p. 418).

Parameters

in	*_jointName*	Parent **Joint** (p. 381) name.

10.74.3.55 void gazebo::physics::Link::Reset ( ) [virtual]

Reset the link.

Reimplemented from **gazebo::physics::Entity** (p. 289).

10.74.3.56 void gazebo::physics::Link::ResetPhysicsStates ( )

Reset the link.

10.74.3.57 void gazebo::physics::Link::SetAngularAccel (const math::Vector3 & _accel)

Set the angular acceleration of the body.

Parameters

in	*_accel*	Angular acceleration.

10.74.3.58 virtual void gazebo::physics::Link::SetAngularDamping (double _damping) [pure virtual]

Set the angular damping factor.

Parameters

in	*_damping*	Angular damping factor.

10.74.3.59 virtual void gazebo::physics::Link::SetAngularVel (const math::Vector3 & _vel) [pure virtual]

Set the angular velocity of the body.

Parameters

in	*_vel*	Angular velocity.

10.74.3.60 virtual void gazebo::physics::Link::SetAutoDisable (bool _disable) [pure virtual]

Allow the link to auto disable.

Parameters

in	*_disable*	If true, the link is allowed to auto disable.

10.74.3.61 void gazebo::physics::Link::SetCollideMode (const std::string & _mode)

Set the collide mode of the body.

Parameters

in	*_mode*	**Collision** (p. 195) Mode, this can be: [all|none|sensors|fixed|ghost] all: collides with everything none: collides with nothing sensors: collides with everything else but other sensors fixed: collides with everything else but other fixed ghost: collides with everything else but other ghost

10.74.3.62 virtual void gazebo::physics::Link::SetEnabled (bool *_enable*) const [pure virtual]

Set whether this body is enabled.

Parameters

in	*_enable*	True to enable the link in the physics engine.

10.74.3.63 virtual void gazebo::physics::Link::SetForce (const math::Vector3 & *_force*) [pure virtual]

Set the force applied to the body.

Parameters

in	*_force*	Force value.

10.74.3.64 virtual void gazebo::physics::Link::SetGravityMode (bool *_mode*) [pure virtual]

Set whether gravity affects this body.

Parameters

in	*_mode*	True to enable gravity.

10.74.3.65 void gazebo::physics::Link::SetInertial (const InertialPtr & *_inertial*)

Set the mass of the link.

[in] _inertial **Inertial** (p. 369) value for the link.

10.74.3.66 virtual void gazebo::physics::Link::SetKinematic (const bool & *_kinematic*) [virtual]

Implement this function.

Set whether this body is in the kinematic state.

Parameters

in	*_kinematic*	True to make the link kinematic only.

10.74.3.67 void gazebo::physics::Link::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	*_retro*	Retro value for all child collisions.

10.74.3.68 void gazebo::physics::Link::SetLinearAccel (const math::Vector3 & _accel)

Set the linear acceleration of the body.

Parameters

in	*_accel*	Linear acceleration.

10.74.3.69 virtual void gazebo::physics::Link::SetLinearDamping (double _damping) [pure virtual]

Set the linear damping factor.

Parameters

in	*_damping*	Linear damping factor.

10.74.3.70 virtual void gazebo::physics::Link::SetLinearVel (const math::Vector3 & _vel) [pure virtual]

Set the linear velocity of the body.

Parameters

in	*_vel*	Linear velocity.

10.74.3.71 virtual bool gazebo::physics::Link::SetSelected (bool _set) [virtual]

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

Parameters

in	*_set*	True to set the link as selected.

Reimplemented from **gazebo::physics::Base** (p. 147).

10.74.3.72 virtual void gazebo::physics::Link::SetSelfCollide (bool _collide) [pure virtual]

Set whether this body will collide with others in the model.

Parameters

in	*_collid*	True to enable collisions.

10.74.3.73 void gazebo::physics::Link::SetState (const LinkState & _state)

Set the current link state.

Parameters

in	*_state*	The state to set the link to.

10.74.3.74 virtual void gazebo::physics::Link::SetTorque (const math::Vector3 & _torque) [pure virtual]

Set the torque applied to the body.

Parameters

in	*_torque*	Torque value.

10.74.3.75 virtual void gazebo::physics::Link::Update ( ) [virtual]

Update the body.

Reimplemented from **gazebo::physics::Base** (p. 148).

10.74.3.76 virtual void gazebo::physics::Link::UpdateMass ( ) [inline], [virtual]

Update the mass matrix.

10.74.3.77 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. 291).

10.74.3.78 virtual void gazebo::physics::Link::UpdateSurface ( ) [inline], [virtual]

Update surface parameters.

10.74.4 Member Data Documentation

10.74.4.1 math::Vector3 gazebo::physics::Link::angularAccel [protected]

Angular acceleration.

10.74.4.2 std::vector<math::Pose> gazebo::physics::Link::attachedModelsOffset [protected]

Offsets for the attached models.

10.74.4.3 std::vector<std::string> gazebo::physics::Link::cgVisuals [protected]

Center of gravity visual elements.

10.74.4.4 InertialPtr gazebo::physics::Link::inertial [protected]

Inertial (p. 369) properties.

Referenced by GetInertial().

10.74.4.5 math::Vector3 gazebo::physics::Link::linearAccel [protected]

Linear acceleration.

10.74.4.6 std::vector<std::string> gazebo::physics::Link::visuals [protected]

Link (p. 418) visual elements.

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

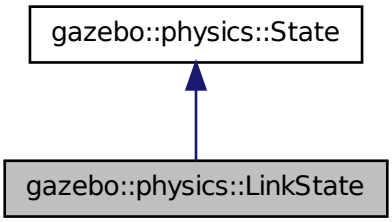
- **Link.hh**

10.75 gazebo::physics::LinkState Class Reference

Store state information of a **physics::Link** (p. 418) object.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::LinkState:



Public Member Functions

- **LinkState** ()

  Default constructor.
- **LinkState** (const **LinkPtr** _link)

  Constructor.
- **LinkState** (const **sdf::ElementPtr** _sdf)

  Constructor.
- virtual ~**LinkState** ()

Destructor.

- void **FillSDF** (**sdf::ElementPtr** _sdf)

  Populate a state SDF element with data from the object.
- const **math::Pose** & **GetAcceleration** () const

  Get the link acceleration.
- **CollisionState GetCollisionState** (unsigned int _index) const

  Get a collision state.
- **CollisionState GetCollisionState** (const std::string &_collisionName) const

  Get a link state by link name.
- unsigned int **GetCollisionStateCount** () const

  Get the number of link states.
- const std::vector < **CollisionState** > & **GetCollisionStates** () const

  Get the collision states.
- const **math::Pose** & **GetPose** () const

  Get the link pose.
- const **math::Pose** & **GetVelocity** () const

  Get the link velocity.
- const **math::Pose** & **GetWrench** () const

  *Get the force applied to the **Link** (p. 418).*
- bool **IsZero** () const

  Return true if the values in the state are zero.
- virtual void **Load** (const **sdf::ElementPtr** _elem)

  Load state from SDF element.
- **LinkState operator+** (const **LinkState** &_state) const

  Addition operator.
- **LinkState operator-** (const **LinkState** &_state) const

  Subtraction operator.
- **LinkState** & **operator=** (const **LinkState** &_state)

  Assignment operator.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::LinkState** &_state)

  Stream insertion operator.

Additional Inherited Members

10.75.1 Detailed Description

Store state information of a **physics::Link** (p. 418) object.

This class captures the entire state of a **Link** (p. 418) at one specific time during a simulation run.

State (p. 758) of a **Link** (p. 418) includes the state of itself all its child **Collision** (p. 195) entities.

10.75.2 Constructor & Destructor Documentation

10.75.2.1 gazebo::physics::LinkState::LinkState ( )

Default constructor.

10.75.2.2 gazebo::physics::LinkState::LinkState (const LinkPtr _link) [explicit]

Constructor.

Build a **LinkState** (p. 438) from an existing **Link** (p. 418).

Parameters

in	*_model*	Pointer to the **Link** (p. 418) from which to gather state info.

10.75.2.3 gazebo::physics::LinkState::LinkState (const sdf::ElementPtr _sdf) [explicit]

Constructor.

Build a **LinkState** (p. 438) from SDF data

Parameters

in	*_sdf*	SDF data to load a link state from.

10.75.2.4 virtual gazebo::physics::LinkState::~LinkState ( ) [virtual]

Destructor.

10.75.3 Member Function Documentation

10.75.3.1 void gazebo::physics::LinkState::FillSDF (sdf::ElementPtr _sdf)

Populate a state SDF element with data from the object.

Parameters

out	*_sdf*	SDF element to populate.

10.75.3.2 const math::Pose& gazebo::physics::LinkState::GetAcceleration ( ) const

Get the link acceleration.

Returns

The acceleration represented as a **math::Pose** (p. 596).

10.75.3.3 CollisionState gazebo::physics::LinkState::GetCollisionState (unsigned int _index) const

Get a collision state.

Get a **Collision** (p. 195) **State** (p. 758) based on an index, where index is in the range of 0...**LinkState::GetCollisionStateCount** (p. 441).

Parameters

in	*_index*	Index of the **CollisionState** (p. 205).

Returns

State (p. 758) of the **Collision** (p. 195).

Exceptions

common::Exception *(p. 318)*	When _index is invalid.

10.75.3.4 CollisionState gazebo::physics::LinkState::GetCollisionState (const std::string & _collisionName) const

Get a link state by link name.

Searches through all CollisionStates. Returns the **CollisionState** (p. 205) with the matching name, if any.

Parameters

in	*_collisionName*	Name of the **CollisionState** (p. 205)

Returns

State (p. 758) of the **Collision** (p. 195).

Exceptions

common::Exception *(p. 318)*	When _collisionName is invalid

10.75.3.5 unsigned int gazebo::physics::LinkState::GetCollisionStateCount ( ) const

Get the number of link states.

This returns the number of Collisions recorded.

Returns

Number of **CollisionState** (p. 205) recorded.

10.75.3.6 const std::vector<CollisionState>& gazebo::physics::LinkState::GetCollisionStates ( ) const

Get the collision states.

Returns

A (p. 111) vector of collision states.

10.75.3.7 const math::Pose& gazebo::physics::LinkState::GetPose ( ) const

Get the link pose.

Returns

The **math::Pose** (p. 596) of the **Link** (p. 418).

10.75.3.8 const math::Pose& gazebo::physics::LinkState::GetVelocity ( ) const

Get the link velocity.

Returns

The velocity represented as a **math::Pose** (p. 596).

10.75.3.9 const math::Pose& gazebo::physics::LinkState::GetWrench ( ) const

Get the force applied to the **Link** (p. 418).

Returns

Magnitude of the force.

10.75.3.10 bool gazebo::physics::LinkState::IsZero ( ) const

Return true if the values in the state are zero.

Returns

True if the values in the state are zero.

10.75.3.11 virtual void gazebo::physics::LinkState::Load (const sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Load **LinkState** (p. 438) information from stored data in and SDF::Element.

Parameters

in	*_elem*	Pointer to the SDF::Element containing state info.

Reimplemented from **gazebo::physics::State** (p. 761).

10.75.3.12 LinkState gazebo::physics::LinkState::operator+ (const LinkState & _state) const

Addition operator.

Parameters

in	*_pt*	**A** (p. 111) state to add.

Returns

The resulting state.

10.75.3.13 LinkState gazebo::physics::LinkState::operator- (const LinkState & _state) const

Subtraction operator.

Parameters

in	*_pt*	**A** (p. 111) state to substract.

Returns

The resulting state.

10.75.3.14 LinkState& gazebo::physics::LinkState::operator= (const LinkState & _state)

Assignment operator.

Parameters

in	*_state*	**State** (p. 758) value

Returns

this

10.75.4 Friends And Related Function Documentation

10.75.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::physics::LinkState & _state) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_state*	**Link** (p. 418) state to output

Returns

the stream

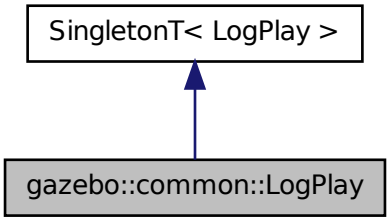
The documentation for this class was generated from the following file:

- **LinkState.hh**

10.76 gazebo::common::LogPlay Class Reference

`#include <LogPlay.hh>`

Inheritance diagram for gazebo::common::LogPlay:



Public Member Functions

- bool **GetChunk** (unsigned int _index, std::string &_data)

  Get data for a particular chunk index.
- unsigned int **GetChunkCount** () const

  Get the number of chunks (steps) in the open log file.
- std::string **GetEncoding** () const

  Get the type of encoding used for current chunck in the open log file.
- std::string **GetGazeboVersion** () const

  Get the Gazebo version number of the open log file.
- std::string **GetLogVersion** () const

  Get the log version number of the open log file.
- uint32_t **GetRandSeed** () const

  Get the random number seed of the open log file.
- bool **IsOpen** () const

  Return true if a file is open.
- void **Open** (const std::string &_logFile)

  Open a log file for reading.
- bool **Step** (std::string &_data)

  Step through the open log file.

Additional Inherited Members

10.76.1 Member Function Documentation

10.76.1.1 bool gazebo::common::LogPlay::GetChunk (unsigned int *_index*, std::string & *_data*)

Get data for a particular chunk index.

Parameters

in	*_index*	Index of the chunk.
out	*_data*	Storage for the chunk's data.

Returns

True if the _index was valid.

10.76.1.2 unsigned int gazebo::common::LogPlay::GetChunkCount ( ) const

Get the number of chunks (steps) in the open log file.

Returns

The number of recorded states in the log file.

10.76.1.3 std::string gazebo::common::LogPlay::GetEncoding ( ) const

Get the type of encoding used for current chunck in the open log file.

Returns

The type of encoding. An empty string will be returned if **LogPlay::Step** (p. 446) has not been called at least once.

10.76.1.4 std::string gazebo::common::LogPlay::GetGazeboVersion ( ) const

Get the Gazebo version number of the open log file.

Returns

The Gazebo version of the open log file. Empty string if a log file is not open.

10.76.1.5 std::string gazebo::common::LogPlay::GetLogVersion ( ) const

Get the log version number of the open log file.

Returns

The log version of the open log file. Empty string if a log file is not open.

10.76.1.6 uint32_t gazebo::common::LogPlay::GetRandSeed ( ) const

Get the random number seed of the open log file.

Returns

The random number seed the open log file. The current random number seed, as defined in **math::Rand::GetSeed** (p. 638).

10.76.1.7 bool gazebo::common::LogPlay::IsOpen ( ) const

Return true if a file is open.

Returns

True if a log file is open.

10.76.1.8 void gazebo::common::LogPlay::Open (const std::string & _logFile)

Open a log file for reading.

Open a log file that was previously recorded.

Parameters

in	*_logFile*	The file to load

Exceptions

Exception *(p. 318)*	

10.76.1.9 bool gazebo::common::LogPlay::Step (std::string & _data)

Step through the open log file.

Parameters

out	*_data*	Data from next entry in the log file.

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

- **LogPlay.hh**

10.77 Logplay Class Reference

Open and playback log files that were recorded using LogRecord.

10.77.1 Detailed Description

Open and playback log files that were recorded using LogRecord.

Use **Logplay** (p. 446) to open a log file (Logplay::Open), and access the recorded state information. Iterators are available to step through the state information. It is also possible to replay the data in a World using the Play functions. Replay involves reading and applying state information to a World.

See Also

LogRecord, State

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

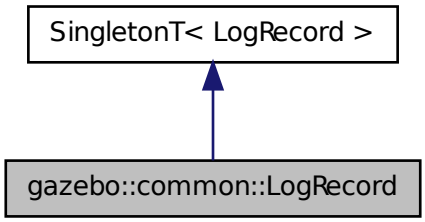
- **LogPlay.hh**

10.78 gazebo::common::LogRecord Class Reference

addtogroup gazebo_common

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::LogRecord:



Public Member Functions

- void **Add** (const std::string &_name, const std::string &_filename, boost::function< bool(std::ostringstream &)> _logCallback)

  Add an object to a log file.
- void **Fini** ()

  Finialize, and shutdown.
- std::string **GetBasePath** () const

  Get the base path for a log recording.
- const std::string & **GetEncoding** () const

  Get the encoding used.
- std::string **GetFilename** (const std::string &_name) const

  Get the filename for a log object.

- unsigned int **GetFileSize** (const std::string &_name) const

  Get the file size for a log object.
- bool **GetFirstUpdate** () const

  Return true if an Update has not yet been completed.
- bool **GetPaused** () const

  Get whether logging is paused.
- bool **GetRunning** () const

  Get whether logging is running.
- **common::Time GetRunTime** () const

  Get the run time in sim time.
- bool **Init** (const std::string &_subdir)

  Initialize logging into a subdirectory.
- bool **Remove** (const std::string &_name)

  Remove an entity from a log.
- void **SetBasePath** (const std::string &_path)

  Set the base path.
- void **SetPaused** (bool _paused)

  Set whether logging should pause.
- bool **Start** (const std::string &_encoding="bz2")

  Start the logger.
- void **Stop** ()

  Stop the logger.

Additional Inherited Members

10.78.1 Detailed Description

addtogroup gazebo_common

Handles logging of data to disk

The **LogRecord** (p. 447) class is a Singleton that manages data logging of any entity within a running simulation. An entity may be a World, Model, or any of their child entities. This class only writes log files, see **LogPlay** (p. 444) for playback functionality.

State information for an entity may be logged through the **LogRecord::Add** (p. 448) function, and stopped through the **LogRecord::Remove** (p. 451) function. Data may be logged into a single file, or split into many separate files by specifying different filenames for the **LogRecord::Add** (p. 448) function.

The **LogRecord** (p. 447) is updated at the start of each simulation step. This guarantees that all data is stored.

See Also

Logplay (p. 446), State

10.78.2 Member Function Documentation

10.78.2.1 void gazebo::common::LogRecord::Add (const std::string & *_name,* const std::string & *_filename,* boost::function< bool(std::ostringstream &)> *_logCallback*)

Add an object to a log file.

Add a new object to a log. An object can be any valid named object in simulation, including the world itself. Duplicate additions are ignored. Objects can be added to the same file by specifying the same _filename.

Parameters

in	*_name*	Name of the object to log.
in	*_filename*	Filename of the log file.
in	*_logCallback*	Function used to log data for the object. Typically an object will have a log function that outputs data to the provided ofstream.

Exceptions

Exception *(p. 318)*	

10.78.2.2 void gazebo::common::LogRecord::Fini ( )

Finialize, and shutdown.

10.78.2.3 std::string gazebo::common::LogRecord::GetBasePath ( ) const

Get the base path for a log recording.

Returns

Path for log recording.

10.78.2.4 const std::string& gazebo::common::LogRecord::GetEncoding ( ) const

Get the encoding used.

Returns

Either [txt, or bz2], where txt is plain txt and bz2 is bzip2 compressed data with Base64 encoding.

10.78.2.5 std::string gazebo::common::LogRecord::GetFilename (const std::string & *_name*) const

Get the filename for a log object.

Parameters

in	*_name*	Name of the log object.

Returns

Filename, empty string if not found.

10.78.2.6 unsigned int gazebo::common::LogRecord::GetFileSize (const std::string & *_name*) const

Get the file size for a log object.

Parameters

in	*_name*	Name of the log object.

Returns

Size in bytes.

10.78.2.7 bool gazebo::common::LogRecord::GetFirstUpdate ( ) const

Return true if an Update has not yet been completed.

Returns

True if an Update has not yet been completed.

10.78.2.8 bool gazebo::common::LogRecord::GetPaused ( ) const

Get whether logging is paused.

Returns

True if logging is paused.

See Also

LogRecord::SetPaused (p. 451)

10.78.2.9 bool gazebo::common::LogRecord::GetRunning ( ) const

Get whether logging is running.

Returns

True if logging has been started.

10.78.2.10 common::Time gazebo::common::LogRecord::GetRunTime ( ) const

Get the run time in sim time.

Returns

Run sim time.

10.78.2.11 bool gazebo::common::LogRecord::Init (const std::string & *_subdir*)

Initialize logging into a subdirectory.

Init may only be called once, False will be returned if called multiple times.

Parameters

in	*_subdir*	Directory to record to

Returns

True if successful.

10.78.2.12 bool gazebo::common::LogRecord::Remove (const std::string & *_name*)

Remove an entity from a log.

Removes an entity from the logger. The stops data recording for the entity and all its children. For example, specifying a world will stop all data logging.

Parameters

in	*_name*	Name of the log

Returns

True if the entity existed and was removed. False if the entity was not registered with the logger.

10.78.2.13 void gazebo::common::LogRecord::SetBasePath (const std::string & *_path*)

Set the base path.

Parameters

in	*_path*	Path to the new logging location.

10.78.2.14 void gazebo::common::LogRecord::SetPaused (bool *_paused*)

Set whether logging should pause.

A (p. 111) paused state means the log file is still open, but data is not written to it.

Parameters

in	*_paused*	True to pause data logging.

See Also

LogRecord::GetPaused (p. 450)

10.78.2.15 bool gazebo::common::LogRecord::Start (const std::string & *_encoding* = "bz2")

Start the logger.

Parameters

in	*_encoding*	The type of encoding (txt, or bz2).

10.78.2.16 void gazebo::common::LogRecord::Stop ( )

Stop the logger.

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

- **LogRecord.hh**

10.79 gazebo::Master Class Reference

A (p. 111) 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.

10.79.1 Detailed Description

A (p. 111) 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. 452), runs World update and sensor generation loops.

10.79.2 Constructor & Destructor Documentation

10.79.2.1 gazebo::Master::Master ( )

Constructor.

10.79.2.2 virtual gazebo::Master::~Master ( ) [virtual]

Destructor.

10.79.3 Member Function Documentation

10.79.3.1 void gazebo::Master::Fini ( )

Finalize the master.

10.79.3.2 void gazebo::Master::Init (uint16_t *_port*)

Initialize.

Parameters

in	*_port*	The master's port

10.79.3.3 void gazebo::Master::Run ( )

Run the master.

10.79.3.4 void gazebo::Master::RunOnce ( )

Run the master one iteration.

10.79.3.5 void gazebo::Master::RunThread ( )

Run the master in a new thread.

10.79.3.6 void gazebo::Master::Stop ( )

Stop the master.

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

- **Master.hh**

10.80 gazebo::common::Material Class Reference

Encapsulates description of a material.

`#include <common/common.hh>`

Public Types

- enum **BlendMode** { **ADD**, **MODULATE**, **REPLACE**, **BLEND_COUNT** }
- enum **ShadeMode** {
  FLAT, **GOURAUD**, **PHONG**, **BLINN**,
  SHADE_COUNT }

Public Member Functions

- **Material** ()

  Constructor.
- **Material** (const **Color** &_clr)

  Create a material with a default color.
- virtual ~**Material** ()

  Destructor.
- **Color GetAmbient** () const

  Get the ambient color.
- void **GetBlendFactors** (double &_srcFactor, double &_dstFactor)

  Get the blend factors.
- **BlendMode GetBlendMode** () const

  Get the blending mode.
- bool **GetDepthWrite** () const

  Get depth write.
- **Color GetDiffuse** () const

  Get the diffuse color.
- **Color GetEmissive** () const

  Get the emissive color.
- bool **GetLighting** () const

  Get lighting enabled.
- std::string **GetName** () const

  Get the name of the material.
- double **GetPointSize** () const

  Get the point size.
- **ShadeMode GetShadeMode** () const

  Get the shading mode.
- double **GetShininess** () const

  Get the shininess.
- **Color GetSpecular** () const

  Get the specular color.
- std::string **GetTextureImage** () const

  Get a texture image.
- double **GetTransparency** () const

  Get the transparency percentage (0..1)
- void **SetAmbient** (const **Color** &_clr)

  Set the ambient color.

- void **SetBlendFactors** (double _srcFactor, double _dstFactor)

  Set the blende factors.
- void **SetBlendMode** (**BlendMode** _b)

  Set the blending mode.
- void **SetDepthWrite** (bool _value)

  Set depth write.
- void **SetDiffuse** (const **Color** &_clr)

  Set the diffuse color.
- void **SetEmissive** (const **Color** &_clr)

  Set the emissive color.
- void **SetLighting** (bool _value)

  Set lighting enabled.
- void **SetPointSize** (double _size)

  Set the point size.
- void **SetShadeMode** (**ShadeMode** _b)

  Set the shading mode param[in] the shading mode.
- void **SetShininess** (double _t)

  Set the shininess.
- void **SetSpecular** (const **Color** &_clr)

  Set the specular color.
- void **SetTextureImage** (const std::string &_tex)

  Set a texture image.
- void **SetTextureImage** (const std::string &_tex, const std::string &_resourcePath)

  Set a texture image.
- void **SetTransparency** (double _t)

  Set the transparency percentage (0..1)

Static Public Attributes

- static std::string **BlendModeStr** [**BLEND_COUNT**]
- static std::string **ShadeModeStr** [**SHADE_COUNT**]

Protected Attributes

- **Color ambient**

  the ambient light color
- **BlendMode blendMode**

  blend mode
- **Color diffuse**

  the diffuse ligth color
- **Color emissive**

  the emissive light color
- std::string **name**

  the name of the material
- double **pointSize**

  point size
- **ShadeMode shadeMode**

the shade mode

- double **shininess**

  shininess value (0 to 1)

- **Color specular**

  the specular light color

- std::string **texImage**

  the texture image file name

- double **transparency**

  transparency value in the range 0 to 1

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Material** &_m)

  Stream insertion operator param[in] _out the output stream to extract from param[out] _m the material information.

10.80.1 Detailed Description

Encapsulates description of a material.

10.80.2 Member Enumeration Documentation

10.80.2.1 enum gazebo::common::Material::BlendMode

Enumerator:

ADD

MODULATE

REPLACE

BLEND_COUNT

10.80.2.2 enum gazebo::common::Material::ShadeMode

Enumerator:

FLAT

GOURAUD

PHONG

BLINN

SHADE_COUNT

10.80.3 Constructor & Destructor Documentation

10.80.3.1 gazebo::common::Material::Material ( )

Constructor.

10.80.3.2 virtual gazebo::common::Material::~Material ( ) [virtual]

Destructor.

10.80.3.3 gazebo::common::Material::Material (const Color & _clr)

Create a material with a default color.

Parameters

in	*_clr*	**Color** (p. 208) of the material

10.80.4 Member Function Documentation

10.80.4.1 Color gazebo::common::Material::GetAmbient ( ) const

Get the ambient color.

Returns

The ambient color

10.80.4.2 void gazebo::common::Material::GetBlendFactors (double & _srcFactor, double & _dstFactor)

Get the blend factors.

Parameters

in	*_srcFactor*	Source factor is returned in this variable
in	*_dstFactor*	Destination factor is returned in this variable

10.80.4.3 BlendMode gazebo::common::Material::GetBlendMode ( ) const

Get the blending mode.

Returns

the blend mode

10.80.4.4 bool gazebo::common::Material::GetDepthWrite ( ) const

Get depth write.

Returns

the depth write enabled state

10.80.4.5 Color gazebo::common::Material::GetDiffuse ( ) const

Get the diffuse color.

Returns

The diffuse color

10.80.4.6 Color gazebo::common::Material::GetEmissive ( ) const

Get the emissive color.

Returns

The emissive color

10.80.4.7 bool gazebo::common::Material::GetLighting ( ) const

Get lighting enabled.

Returns

the lighting enabled state

10.80.4.8 std::string gazebo::common::Material::GetName ( ) const

Get the name of the material.

Returns

The name of the material

10.80.4.9 double gazebo::common::Material::GetPointSize ( ) const

Get the point size.

Returns

the point size

10.80.4.10 ShadeMode gazebo::common::Material::GetShadeMode ( ) const

Get the shading mode.

Returns

the shading mode

10.80.4.11 double gazebo::common::Material::GetShininess ( ) const

Get the shininess.

Returns

The shininess value

10.80.4.12 Color gazebo::common::Material::GetSpecular ( ) const

Get the specular color.

Returns

The specular color

10.80.4.13 std::string gazebo::common::Material::GetTextureImage ( ) const

Get a texture image.

Returns

The name of the texture image (if one exists) or an empty string

10.80.4.14 double gazebo::common::Material::GetTransparency ( ) const

Get the transparency percentage (0..1)

Returns

The transparency percentage

10.80.4.15 void gazebo::common::Material::SetAmbient (const Color & *_clr*)

Set the ambient color.

Parameters

in	*_clr*	The ambient color

10.80.4.16 void gazebo::common::Material::SetBlendFactors (double *_srcFactor*, double *_dstFactor*)

Set the blende factors.

Will be interpreted as: (texture * _srcFactor) + (scene_pixel * _dstFactor)

Parameters

in	*_srcFactor*	The source factor
in	*_dstFactor*	The destination factor

10.80.4.17 void gazebo::common::Material::SetBlendMode (BlendMode *_b*)

Set the blending mode.

Parameters

in	*_b*	the blend mode

10.80.4.18 void gazebo::common::Material::SetDepthWrite (bool *_value*)

Set depth write.

Parameters

in	*_value*	the depth write enabled state

10.80.4.19 void gazebo::common::Material::SetDiffuse (const Color & *_clr*)

Set the diffuse color.

Parameters

in	*_clr*	The diffuse color

10.80.4.20 void gazebo::common::Material::SetEmissive (const Color & *_clr*)

Set the emissive color.

Parameters

in	*_clr*	The emissive color

10.80.4.21 void gazebo::common::Material::SetLighting (bool *_value*)

Set lighting enabled.

Parameters

in	*_value*	the lighting enabled state

10.80.4.22 void gazebo::common::Material::SetPointSize (double *_size*)

Set the point size.

Parameters

in	*_size*	the size

10.80.4.23 void gazebo::common::Material::SetShadeMode (ShadeMode *_b*)

Set the shading mode param[in] the shading mode.

10.80.4.24 void gazebo::common::Material::SetShininess (double _t)

Set the shininess.

Parameters

in	_t	The shininess value

10.80.4.25 void gazebo::common::Material::SetSpecular (const Color & _clr)

Set the specular color.

Parameters

in	_clr	The specular color

10.80.4.26 void gazebo::common::Material::SetTextureImage (const std::string & _tex)

Set a texture image.

Parameters

in	_tex	The name of the texture, which must be in Gazebo's resource path

10.80.4.27 void gazebo::common::Material::SetTextureImage (const std::string & _tex, const std::string & _resourcePath)

Set a texture image.

Parameters

in	_tex	The name of the texture
in	_resourcePath	Path which contains _tex

10.80.4.28 void gazebo::common::Material::SetTransparency (double _t)

Set the transparency percentage (0..1)

Parameters

in	_t	The amount of transparency (0..1)

10.80.5 Friends And Related Function Documentation

10.80.5.1 std::ostream& operator<< (std::ostream & _out, const gazebo::common::Material & _m) [friend]

Stream insertion operator param[in] _out the output stream to extract from param[out] _m the material information.

10.80.6 Member Data Documentation

10.80.6.1 Color gazebo::common::Material::ambient [protected]

the ambient light color

10.80.6.2 BlendMode gazebo::common::Material::blendMode [protected]

blend mode

10.80.6.3 std::string gazebo::common::Material::BlendModeStr[BLEND_COUNT] [static]

10.80.6.4 Color gazebo::common::Material::diffuse [protected]

the diffuse ligth color

10.80.6.5 Color gazebo::common::Material::emissive [protected]

the emissive light color

10.80.6.6 std::string gazebo::common::Material::name [protected]

the name of the material

10.80.6.7 double gazebo::common::Material::pointSize [protected]

point size

10.80.6.8 ShadeMode gazebo::common::Material::shadeMode [protected]

the shade mode

10.80.6.9 std::string gazebo::common::Material::ShadeModeStr[SHADE_COUNT] [static]

10.80.6.10 double gazebo::common::Material::shininess [protected]

shininess value (0 to 1)

10.80.6.11 Color gazebo::common::Material::specular [protected]

the specular light color

10.80.6.12 std::string gazebo::common::Material::texImage [protected]

the texture image file name

10.80.6.13 double gazebo::common::Material::transparency [protected]

transparency value in the range 0 to 1

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

- **common/Material.hh**

10.81 gazebo::math::Matrix3 Class Reference

A (p. 111) 3x3 matrix class.

`#include <Matrix3.hh>`

Public Member Functions

- **Matrix3** ()

  Constructor.
- **Matrix3** (const **Matrix3** &_m)

  Copy constructor.
- **Matrix3** (double _v00, double _v01, double _v02, double _v10, double _v11, double _v12, double _v20, double _v21, double _v22)

  Constructor.
- virtual ∼**Matrix3** ()

  Desctructor.
- **Matrix3 operator**∗ (const double &_s) const

  returns the element wise scalar multiplication
- **Matrix3 operator**∗ (const **Matrix3** &_m) const

  Matrix multiplication operator.
- **Matrix3 operator+** (const **Matrix3** &_m) const

  returns the element wise sum of two matrices
- **Matrix3 operator-** (const **Matrix3** &_m) const

  returns the element wise difference of two matrices
- bool **operator==** (const **Matrix3** &_m) const

  Equality test operator.
- const double ∗ **operator[]** (size_t _row) const

  Array subscript operator.
- double ∗ **operator[]** (size_t _row)

  Array subscript operator.
- void **SetCol** (unsigned int _c, const **Vector3** &_v)

  Set a column.
- void **SetFromAxes** (const **Vector3** &_xAxis, const **Vector3** &_yAxis, const **Vector3** &_zAxis)

  Set the matrix from three axis (1 per column)
- void **SetFromAxis** (const **Vector3** &_axis, double _angle)

  Set the matrix from an axis and angle.

Protected Attributes

- double **m** [3][3]

  the 3x3 matrix

Friends

- **Matrix3 operator**∗ (double _s, const **Matrix3** &_m)

  Multiplication operators.
- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Matrix3** &_m)

  Stream insertion operator.

10.81.1 Detailed Description

A (p. 111) 3x3 matrix class.

10.81.2 Constructor & Destructor Documentation

10.81.2.1 gazebo::math::Matrix3::Matrix3 ( )

Constructor.

Referenced by operator∗(), operator+(), and operator-().

10.81.2.2 gazebo::math::Matrix3::Matrix3 (const Matrix3 & _m)

Copy constructor.

Parameters

_m	Matrix to copy

10.81.2.3 gazebo::math::Matrix3::Matrix3 (double _v00, double _v01, double _v02, double _v10, double _v11, double _v12, double _v20, double _v21, double _v22)

Constructor.

Parameters

in	*_v00*	Row 0, Col 0 value
in	*_v01*	Row 0, Col 1 value
in	*_v02*	Row 0, Col 2 value
in	*_v10*	Row 1, Col 0 value
in	*_v11*	Row 1, Col 1 value
in	*_v12*	Row 1, Col 2 value
in	*_v20*	Row 2, Col 0 value
in	*_v21*	Row 2, Col 1 value
in	*_v22*	Row 2, Col 2 value

10.81.2.4 virtual gazebo::math::Matrix3::~Matrix3 ( ) [virtual]

Desctructor.

10.81.3 Member Function Documentation

10.81.3.1 Matrix3 gazebo::math::Matrix3::operator* (const double & _s) const [inline]

returns the element wise scalar multiplication

References m, and Matrix3().

10.81.3.2 Matrix3 gazebo::math::Matrix3::operator* (const Matrix3 & _m) const [inline]

Matrix multiplication operator.

Parameters

in	*_m*	**Matrix3** (p. 463) to multiply

Returns

product of this * _m

References m, and Matrix3().

10.81.3.3 Matrix3 gazebo::math::Matrix3::operator+ (const Matrix3 & _m) const [inline]

returns the element wise sum of two matrices

References m, and Matrix3().

10.81.3.4 Matrix3 gazebo::math::Matrix3::operator- (const Matrix3 & _m) const [inline]

returns the element wise difference of two matrices

References m, and Matrix3().

10.81.3.5 bool gazebo::math::Matrix3::operator== (const Matrix3 & _m) const

Equality test operator.

Parameters

in	*_m*	**Matrix3** (p. 463) to test

Returns

True if equal (using the default tolerance of 1e-6)

10.81.3.6 const double* gazebo::math::Matrix3::operator[] (size_t _row) const [inline]

Array subscript operator.

Parameters

in	*_row*	row index

Returns

a pointer to the row

References m.

10.81.3.7 double* gazebo::math::Matrix3::operator[] (size_t _row) [inline]

Array subscript operator.

Parameters

in	*_row*	row index

Returns

a pointer to the row

References m.

10.81.3.8 void gazebo::math::Matrix3::SetCol (unsigned int _c, const Vector3 & _v)

Set a column.

Parameters

in	*_c*	The colum index (0, 1, 2)
in	*_v*	The value to set in each row of the column

10.81.3.9 void gazebo::math::Matrix3::SetFromAxes (const Vector3 & _xAxis, const Vector3 & _yAxis, const Vector3 & _zAxis)

Set the matrix from three axis (1 per column)

Parameters

in	*_xAxis*	The x axis
in	*_yAxis*	The y axis
in	*_zAxis*	The z axis

10.81.3.10 void gazebo::math::Matrix3::SetFromAxis (const Vector3 & _axis, double _angle)

Set the matrix from an axis and angle.

Parameters

in	*_axis*	the axis
in	*_angle*	ccw rotation around the axis in radians

10.81.4 Friends And Related Function Documentation

10.81.4.1 Matrix3 operator∗ (double *_s*, const Matrix3 & *_m*) [friend]

Multiplication operators.

Parameters

in	*_s*	the scaling factor
in	*_m*	input matrix

Returns

a scaled matrix

10.81.4.2 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Matrix3 & *_m*) [friend]

Stream insertion operator.

Parameters

in	*_out*	Output stream
in	*_m*	Matrix to output

Returns

the stream

10.81.5 Member Data Documentation

10.81.5.1 double gazebo::math::Matrix3::m[3][3] [protected]

the 3x3 matrix

Referenced by operator∗(), operator+(), operator-(), and operator[]().

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

- **Matrix3.hh**

10.82 gazebo::math::Matrix4 Class Reference

A (p. 111) 3x3 matrix class.

`#include <math/gzmath.hh>`

Public Member Functions

- **Matrix4** ()

  Constructor.
- **Matrix4** (const **Matrix4** &_m)

  Copy constructor.
- **Matrix4** (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

  Constructor.
- virtual ∼**Matrix4** ()

  Destructor.
- **math::Pose GetAsPose** () const

  *Get the transformation as **math::Pose** (*p. *596).*
- **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. *855).*
- **Matrix4 Inverse** () const

  Return the inverse matrix.
- bool **IsAffine** () const

  Return true if the matrix is affine.
- **Matrix4 operator**∗ (const **Matrix4** &_mat) const

  Multiplication operator.
- **Matrix4 operator**∗ (const **Matrix3** &_mat) const

  Multiplication operator.
- **Vector3 operator**∗ (const **Vector3** &_vec) const

  Multiplication operator.
- **Matrix4** & **operator=** (const **Matrix4** &_mat)

  Equal operator.
- const **Matrix4** & **operator=** (const **Matrix3** &_mat)

  Equal operator for 3x3 matrix.
- bool **operator==** (const **Matrix4** &_m) const

  Equality operator.
- double ∗ **operator[]** (size_t _row)

  Array subscript operator.
- const double ∗ **operator[]** (size_t _row) const
- void **Set** (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

  Change the values.
- void **SetScale** (const **Vector3** &_s)

  Set the scale.
- void **SetTranslate** (const **Vector3** &_t)

  Set the translational values [(0, 3) (1, 3) (2, 3)].
- **Vector3 TransformAffine** (const **Vector3** &_v) const

  Perform an affine transformation.

Static Public Attributes

- static const **Matrix4 IDENTITY**

  Identity matrix.
- static const **Matrix4 ZERO**

  Zero matrix.

Protected Attributes

- double **m** [4][4]

  The 4x4 matrix.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Matrix4** &_m)

  Stream insertion operator.

10.82.1 Detailed Description

A (p. 111) 3x3 matrix class.

10.82.2 Constructor & Destructor Documentation

10.82.2.1 gazebo::math::Matrix4::Matrix4 ( )

Constructor.

10.82.2.2 gazebo::math::Matrix4::Matrix4 (const Matrix4 & *_m*)

Copy constructor.

Parameters

_m	Matrix to copy

10.82.2.3 gazebo::math::Matrix4::Matrix4 (double *_v00*, double *_v01*, double *_v02*, double *_v03*, double *_v10*, double *_v11*, double *_v12*, double *_v13*, double *_v20*, double *_v21*, double *_v22*, double *_v23*, double *_v30*, double *_v31*, double *_v32*, double *_v33*)

Constructor.

Parameters

in	*_v00*	Row 0, Col 0 value
in	*_v01*	Row 0, Col 1 value
in	*_v02*	Row 0, Col 2 value
in	*_v03*	Row 0, Col 3 value
in	*_v10*	Row 1, Col 0 value

in	*_v11*	Row 1, Col 1 value
in	*_v12*	Row 1, Col 2 value
in	*_v13*	Row 1, Col 3 value
in	*_v20*	Row 2, Col 0 value
in	*_v21*	Row 2, Col 1 value
in	*_v22*	Row 2, Col 2 value
in	*_v23*	Row 2, Col 3 value
in	*_v30*	Row 3, Col 0 value
in	*_v31*	Row 3, Col 1 value
in	*_v32*	Row 3, Col 2 value
in	*_v33*	Row 3, Col 3 value

10.82.2.4 virtual gazebo::math::Matrix4::~Matrix4 ( ) [virtual]

Destructor.

10.82.3 Member Function Documentation

10.82.3.1 math::Pose gazebo::math::Matrix4::GetAsPose ( ) const

Get the transformation as **math::Pose** (p. 596).

Returns

the pose

10.82.3.2 Vector3 gazebo::math::Matrix4::GetEulerRotation (unsigned int *solution_number* = 1) const

Get the rotation as a Euler angles.

Returns

the rotation

10.82.3.3 Quaternion gazebo::math::Matrix4::GetRotation ( ) const

Get the rotation as a quaternion.

Returns

the rotation

10.82.3.4 Vector3 gazebo::math::Matrix4::GetTranslation ( ) const

Get the translational values as a **Vector3** (p. 855).

Returns

x,y,z

10.82.3.5 Matrix4 gazebo::math::Matrix4::Inverse ( ) const

Return the inverse matrix.

10.82.3.6 bool gazebo::math::Matrix4::IsAffine ( ) const

Return true if the matrix is affine.

Returns

true if the matrix is affine, false otherwise

10.82.3.7 Matrix4 gazebo::math::Matrix4::operator* (const Matrix4 & *_mat*) const

Multiplication operator.

Parameters

_mat	Incoming matrix

Returns

This matrix * _mat

10.82.3.8 Matrix4 gazebo::math::Matrix4::operator* (const Matrix3 & *_mat*) const

Multiplication operator.

Parameters

_mat	Incoming matrix

Returns

This matrix * _mat

10.82.3.9 Vector3 gazebo::math::Matrix4::operator* (const Vector3 & *_vec*) const

Multiplication operator.

Parameters

_vec	**Vector3** (p. 855)

Returns

Resulting vector from multiplication

10.82.3.10 Matrix4& gazebo::math::Matrix4::operator= (const Matrix4 & _mat)

Equal operator.

this = _mat

Parameters

_mat	Incoming matrix

Returns

itself

10.82.3.11 const Matrix4& gazebo::math::Matrix4::operator= (const Matrix3 & _mat)

Equal operator for 3x3 matrix.

Parameters

_mat	Incoming matrix

Returns

itself

10.82.3.12 bool gazebo::math::Matrix4::operator== (const Matrix4 & _m) const

Equality operator.

Parameters

in	*_m*	**Matrix3** (p. 463) to test

Returns

true if the 2 matrices are equal (using the tolerance 1e-6), false otherwise

10.82.3.13 double* gazebo::math::Matrix4::operator[] (size_t _row) [inline]

Array subscript operator.

Parameters

in	*_row*	the row index

Returns

the row

References m.

10.82.3.14 const double* gazebo::math::Matrix4::operator[] (size_t _row) const [inline]

Parameters

in	*_row*	the row index

Returns

the row

References m.

10.82.3.15 void gazebo::math::Matrix4::Set (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

Change the values.

Parameters

in	*_v00*	Row 0, Col 0 value
in	*_v01*	Row 0, Col 1 value
in	*_v02*	Row 0, Col 2 value
in	*_v03*	Row 0, Col 3 value
in	*_v10*	Row 1, Col 0 value
in	*_v11*	Row 1, Col 1 value
in	*_v12*	Row 1, Col 2 value
in	*_v13*	Row 1, Col 3 value
in	*_v20*	Row 2, Col 0 value
in	*_v21*	Row 2, Col 1 value
in	*_v22*	Row 2, Col 2 value
in	*_v23*	Row 2, Col 3 value
in	*_v30*	Row 3, Col 0 value
in	*_v31*	Row 3, Col 1 value
in	*_v32*	Row 3, Col 2 value
in	*_v33*	Row 3, Col 3 value

10.82.3.16 void gazebo::math::Matrix4::SetScale (const Vector3 & _s)

Set the scale.

Parameters

in	*_s*	scale

10.82.3.17 void gazebo::math::Matrix4::SetTranslate (const Vector3 & _t)

Set the translational values [(0, 3) (1, 3) (2, 3)].

Parameters

in	*_t*	Values to set

10.82.3.18 Vector3 gazebo::math::Matrix4::TransformAffine (const Vector3 & _v) const

Perform an affine transformation.

Parameters

_v	**Vector3** (p. 855) value for the transformation

Returns

The result of the transformation

10.82.4 Friends And Related Function Documentation

10.82.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Matrix4 & _m) [friend]

Stream insertion operator.

Parameters

_out	output stream
_m	Matrix to output

Returns

the stream

10.82.5 Member Data Documentation

10.82.5.1 const Matrix4 gazebo::math::Matrix4::IDENTITY [static]

Identity matrix.

10.82.5.2 double gazebo::math::Matrix4::m[4][4] [protected]

The 4x4 matrix.

Referenced by operator[]().

10.82.5.3 const Matrix4 gazebo::math::Matrix4::ZERO [static]

Zero matrix.

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

- **Matrix4.hh**

10.83 gazebo::common::Mesh Class Reference

A (p. 111) 3D mesh.

`#include <common/common.hh>`

Public Member Functions

- **Mesh** ()

  Constructor.
- virtual ∼**Mesh** ()

  Destructor.
- int **AddMaterial** (**Material** ∗_mat)

  Add a material to the mesh.
- void **AddSubMesh** (**SubMesh** ∗_child)

  Add a submesh mesh.
- void **Center** (const **math::Vector3** &_center=**math::Vector3::Zero**)

  Move the center of the mesh to the given coordinate.
- void **FillArrays** (float ∗∗_vertArr, int ∗∗_indArr) const

  Put all the data into flat arrays.
- void **GenSphericalTexCoord** (const **math::Vector3** &_center)

  Generate texture coordinates using spherical projection from center.
- void **GetAABB** (**math::Vector3** &_center, **math::Vector3** &_min_xyz, **math::Vector3** &_max_xyz) const

  Get AABB coordinate.
- unsigned int **GetIndexCount** () const

  Return the number of indices.
- const **Material** ∗ **GetMaterial** (int _index) const

  Get a material.
- unsigned int **GetMaterialCount** () const

  Get the number of materials.
- **math::Vector3 GetMax** () const

  Get the maximun X, Y, Z values.
- **math::Vector3 GetMin** () const

  Get the minimum X, Y, Z values.
- std::string **GetName** () const

  Get the name of this mesh.
- unsigned int **GetNormalCount** () const

  Return the number of normals.
- std::string **GetPath** () const

  Get the path which contains the mesh resource.
- **Skeleton** ∗ **GetSkeleton** () const

  Get the skeleton to which this mesh is attached.
- const **SubMesh** ∗ **GetSubMesh** (unsigned int _i) const

  Get a child mesh.
- const **SubMesh** ∗ **GetSubMesh** (const std::string &_name) const

  Get a child mesh by name.
- unsigned int **GetSubMeshCount** () const

Get the number of children.

- unsigned int **GetTexCoordCount** () const

  Return the number of texture coordinates.

- unsigned int **GetVertexCount** () const

  Return the number of vertices.

- bool **HasSkeleton** () const

  Return true if mesh is attached to a skeleton.

- void **RecalculateNormals** ()

  Recalculate all the normals of each face defined by three indices.

- void **Scale** (double _factor)

  Scale all vertices by _factor.

- void **SetName** (const std::string &_n)

  Set the name of this mesh.

- void **SetPath** (const std::string &_path)

  Set the path which contains the mesh resource.

- void **SetScale** (const **math::Vector3** &_factor)

  Scale all vertices by the _factor vector.

- void **SetSkeleton** (**Skeleton** ∗_skel)

  Set the mesh skeleton.

- void **Translate** (const **math::Vector3** &_vec)

  Move all vertices in all submeshes by _vec.

10.83.1 Detailed Description

A (p. 111) 3D mesh.

10.83.2 Constructor & Destructor Documentation

10.83.2.1 gazebo::common::Mesh::Mesh ( )

Constructor.

10.83.2.2 virtual gazebo::common::Mesh::∼Mesh ( ) [virtual]

Destructor.

10.83.3 Member Function Documentation

10.83.3.1 int gazebo::common::Mesh::AddMaterial (**Material** ∗ *_mat*)

Add a material to the mesh.

Parameters

in	*_mat*	the material

Returns

Index of this material

10.83.3.2 void gazebo::common::Mesh::AddSubMesh (SubMesh * _child)

Add a submesh mesh.

The **Mesh** (p. 475) object takes ownership of the submesh.

Parameters

in	*_child*	the submesh

10.83.3.3 void gazebo::common::Mesh::Center (const math::Vector3 & _center = math::Vector3::Zero)

Move the center of the mesh to the given coordinate.

This will move all the vertices in all submeshes.

Parameters

in	*_center*	Location of the mesh center.

10.83.3.4 void gazebo::common::Mesh::FillArrays (float ** _vertArr, int ** _indArr) const

Put all the data into flat arrays.

Parameters

out	*_vertArr*	the vertex array
out	*_indArr*	the index array

10.83.3.5 void gazebo::common::Mesh::GenSphericalTexCoord (const math::Vector3 & _center)

Generate texture coordinates using spherical projection from center.

Parameters

in	*_center*	the center of the projection

10.83.3.6 void gazebo::common::Mesh::GetAABB (math::Vector3 & _center, math::Vector3 & _min_xyz, math::Vector3 & _max_xyz) const

Get AABB coordinate.

Parameters

out	*_center*	of the bounding box
out	*_min_xyz*	bounding box minimum values
out	*_max_xyz*	bounding box maximum values

10.83.3.7 unsigned int gazebo::common::Mesh::GetIndexCount ( ) const

Return the number of indices.

Returns

the count

10.83.3.8 const Material* gazebo::common::Mesh::GetMaterial (int *_index*) const

Get a material.

Parameters

in	*_index*	the index

Returns

the material or NULL if the index is out of bounds

10.83.3.9 unsigned int gazebo::common::Mesh::GetMaterialCount ( ) const

Get the number of materials.

Returns

the count

10.83.3.10 math::Vector3 gazebo::common::Mesh::GetMax ( ) const

Get the maximun X, Y, Z values.

Returns

the upper bounds of the bounding box

10.83.3.11 math::Vector3 gazebo::common::Mesh::GetMin ( ) const

Get the minimum X, Y, Z values.

Returns

the lower bounds of the bounding box

10.83.3.12 std::string gazebo::common::Mesh::GetName ( ) const

Get the name of this mesh.

Returns

the name

10.83.3.13 unsigned int gazebo::common::Mesh::GetNormalCount ( ) const

Return the number of normals.

Returns

the count

10.83.3.14 std::string gazebo::common::Mesh::GetPath ( ) const

Get the path which contains the mesh resource.

Returns

the path to the mesh resource

10.83.3.15 Skeleton* gazebo::common::Mesh::GetSkeleton ( ) const

Get the skeleton to which this mesh is attached.

Returns

pointer to skeleton, or NULL if none is present.

10.83.3.16 const SubMesh* gazebo::common::Mesh::GetSubMesh (unsigned int _i) const

Get a child mesh.

Parameters

in	*_i*	the index

Returns

the submesh. An exception is thrown if the index is out of bounds

10.83.3.17 const SubMesh* gazebo::common::Mesh::GetSubMesh (const std::string & _name) const

Get a child mesh by name.

Parameters

in	*_name*	Name of the submesh.

Returns

The submesh, NULL if the _name is not found.

10.83.3.18 unsigned int gazebo::common::Mesh::GetSubMeshCount ( ) const

Get the number of children.

Returns

the count

10.83.3.19 unsigned int gazebo::common::Mesh::GetTexCoordCount ( ) const

Return the number of texture coordinates.

Returns

the count

10.83.3.20 unsigned int gazebo::common::Mesh::GetVertexCount ( ) const

Return the number of vertices.

Returns

the count

10.83.3.21 bool gazebo::common::Mesh::HasSkeleton ( ) const

Return true if mesh is attached to a skeleton.

10.83.3.22 void gazebo::common::Mesh::RecalculateNormals ( )

Recalculate all the normals of each face defined by three indices.

10.83.3.23 void gazebo::common::Mesh::Scale (double *_factor*)

Scale all vertices by _factor.

Parameters

_factor	Scaling factor

10.83.3.24 void gazebo::common::Mesh::SetName (const std::string & *_n*)

Set the name of this mesh.

Parameters

in	*_n*	the name to set

10.83.3.25 void gazebo::common::Mesh::SetPath (const std::string & _path)

Set the path which contains the mesh resource.

Parameters

in	*_path*	the file path

10.83.3.26 void gazebo::common::Mesh::SetScale (const math::Vector3 & _factor)

Scale all vertices by the _factor vector.

Parameters

in	*_factor*	Scaling vector

10.83.3.27 void gazebo::common::Mesh::SetSkeleton (Skeleton * _skel)

Set the mesh skeleton.

10.83.3.28 void gazebo::common::Mesh::Translate (const math::Vector3 & _vec)

Move all vertices in all submeshes by _vec.

Parameters

in	*_vec*	Amount to translate vertices.

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

- **Mesh.hh**

10.84 gazebo::common::MeshCSG Class Reference

Creates CSG meshes.

`#include <common/common.hh>`

Public Types

- enum **BooleanOperation** { **UNION**, **INTERSECTION**, **DIFFERENCE** }

  An enumeration of the boolean operations.

Public Member Functions

- **MeshCSG** ()

  Constructor.
- virtual ~**MeshCSG** ()

Destructor.

- **Mesh** ∗ **CreateBoolean** (const **Mesh** ∗_m1, const **Mesh** ∗_m2, const int _operation, const **math::Pose** &_offset=**math::Pose::Zero**)

  Create a boolean mesh from two meshes.

10.84.1 Detailed Description

Creates CSG meshes.

10.84.2 Member Enumeration Documentation

10.84.2.1 enum gazebo::common::MeshCSG::BooleanOperation

An enumeration of the boolean operations.

Enumerator:

UNION

INTERSECTION

DIFFERENCE

10.84.3 Constructor & Destructor Documentation

10.84.3.1 gazebo::common::MeshCSG::MeshCSG ( )

Constructor.

10.84.3.2 virtual gazebo::common::MeshCSG::∼MeshCSG ( ) [virtual]

Destructor.

10.84.4 Member Function Documentation

10.84.4.1 Mesh∗ gazebo::common::MeshCSG::CreateBoolean (const Mesh ∗ *_m1*, const Mesh ∗ *_m2*, const int *_operation*, const math::Pose & *_offset* = math::Pose::Zero)

Create a boolean mesh from two meshes.

Parameters

in	*_m1*	the parent mesh in the boolean operation
in	*_m2*	the child mesh in the boolean operation
in	*_operation*	the boolean operation applied to the two meshes
in	*_offset*	_m2's pose offset from _m1

Returns

a pointer to the created mesh

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

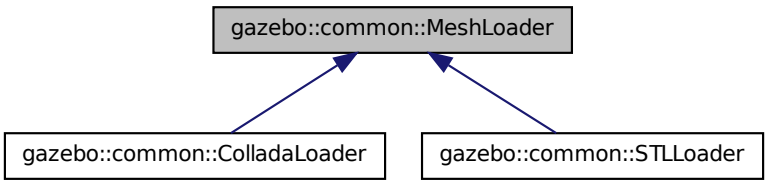
- **MeshCSG.hh**

10.85 gazebo::common::MeshLoader Class Reference

Base class for loading meshes.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::MeshLoader:



Public Member Functions

- **MeshLoader** ()

  Constructor.
- virtual ∼**MeshLoader** ()

  Destructor.
- virtual **Mesh** ∗ **Load** (const std::string &_filename)=0

  Load a 3D mesh.

10.85.1 Detailed Description

Base class for loading meshes.

10.85.2 Constructor & Destructor Documentation

10.85.2.1 gazebo::common::MeshLoader::MeshLoader ( )

Constructor.

10.85.2.2 virtual gazebo::common::MeshLoader::∼MeshLoader ( ) [virtual]

Destructor.

10.85.3 Member Function Documentation

10.85.3.1 virtual Mesh* gazebo::common::MeshLoader::Load (const std::string & *_filename*) [pure virtual]

Load a 3D mesh.

Parameters

in	*_filename*	the path to the mesh

Returns

a pointer to the created mesh

Implemented in **gazebo::common::ColladaLoader** (p. 195), and **gazebo::common::STLLoader** (p. 763).

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

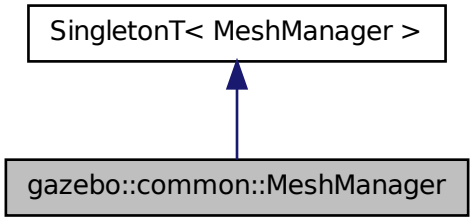
- **MeshLoader.hh**

10.86 gazebo::common::MeshManager Class Reference

Maintains and manages all meshes.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::MeshManager:



Public Member Functions

- void **AddMesh** (**Mesh** *_mesh)
  Add a mesh to the manager.
- void **CreateBox** (const std::string &_name, const **math::Vector3** &_sides, const **math::Vector2d** &_uvCoords)
  Create a Box mesh.
- void **CreateCamera** (const std::string &_name, float _scale)
  Create a Camera mesh.
- void **CreateCone** (const std::string &_name, float _radius, float _height, int _rings, int _segments)

Create a cone mesh.

- void **CreateCylinder** (const std::string &_name, float _radius, float _height, int _rings, int _segments)

  Create a cylinder mesh.
- void **CreatePlane** (const std::string &_name, const **math::Plane** &_plane, const **math::Vector2d** &_segments, const **math::Vector2d** &_uvTile)

  Create mesh for a plane.
- void **CreatePlane** (const std::string &_name, const **math::Vector3** &_normal, double _d, const **math::Vector2d** &_size, const **math::Vector2d** &_segments, const **math::Vector2d** &_uvTile)

  Create mesh for a plane.
- void **CreateSphere** (const std::string &_name, float _radius, int _rings, int _segments)

  Create a sphere mesh.
- void **CreateTube** (const std::string &_name, float _innerRadius, float _outterRadius, float _height, int _rings, int _segments)

  Create a tube mesh.
- void **GenSphericalTexCoord** (const **Mesh** *_mesh, **math::Vector3** _center)

  generate spherical texture coordinates
- const **Mesh** * **GetMesh** (const std::string &_name) const

  Get a mesh by name.
- void **GetMeshAABB** (const **Mesh** *_mesh, **math::Vector3** &_center, **math::Vector3** &_min_xyz, **math::Vector3** &_max_xyz)

  Get mesh aabb and center.
- bool **HasMesh** (const std::string &_name) const

  Return true if the mesh exists.
- bool **IsValidFilename** (const std::string &_filename)

  Checks a path extension against the list of valid extensions.
- const **Mesh** * **Load** (const std::string &_filename)

  Load a mesh from a file.

Additional Inherited Members

10.86.1 Detailed Description

Maintains and manages all meshes.

10.86.2 Member Function Documentation

10.86.2.1 void gazebo::common::MeshManager::AddMesh (Mesh * *_mesh*)

Add a mesh to the manager.

This **MeshManager** (p. 484) takes ownership of the mesh and will destroy it. See ~MeshManager.

Parameters

in	*the*	mesh to add.

10.86.2.2 void gazebo::common::MeshManager::CreateBox (const std::string & *_name,* const math::Vector3 & *_sides,* const math::Vector2d & *_uvCoords*)

Create a Box mesh.

Parameters

in	*_name*	the name of the new mesh
in	*_sides*	the x y x dimentions of eah side in meter
in	*_uvCoords*	the texture coordinates

10.86.2.3 void gazebo::common::MeshManager::CreateCamera (const std::string & *_name,* float *_scale*)

Create a Camera mesh.

Parameters

in	*_name*	name of the new mesh
in	*_scale*	scaling factor for the camera

10.86.2.4 void gazebo::common::MeshManager::CreateCone (const std::string & *_name,* float *_radius,* float *_height,* int *_rings,* int *_segments*)

Create a cone mesh.

Parameters

in	*_name*	the name of the new mesh
in	*_radius*	the radius of the cylinder in the x y plane
in	*_height*	the height along z
in	*_rings*	the number of circles along the height
in	*_segments*	the number of segment per circle

10.86.2.5 void gazebo::common::MeshManager::CreateCylinder (const std::string & *_name,* float *_radius,* float *_height,* int *_rings,* int *_segments*)

Create a cylinder mesh.

Parameters

in	*_name*	the name of the new mesh
in	*_radius*	the radius of the cylinder in the x y plane
in	*_height*	the height along z
in	*_rings*	the number of circles along the height
in	*_segments*	the number of segment per circle

10.86.2.6 void gazebo::common::MeshManager::CreatePlane (const std::string & *_name,* const math::Plane & *_plane,* const math::Vector2d & *_segments,* const math::Vector2d & *_uvTile*)

Create mesh for a plane.

Parameters

in	*_name*	
in	*_plane*	plane parameters
in	*_segments*	number of segments in x and y
in	*_uvTile*	the texture tile size in x and y

10.86.2.7 void gazebo::common::MeshManager::CreatePlane (const std::string & *_name*, const math::Vector3 & *_normal*, double *_d*, const math::Vector2d & *_size*, const math::Vector2d & *_segments*, const math::Vector2d & *_uvTile*)

Create mesh for a plane.

Parameters

in	*_name*	the name of the new mesh
in	*_normal*	the normal to the plane
in	*_d*	distance from the origin along normal
in	*_size*	the size of the plane in x and y
in	*_segments*	the number of segments in x and y
in	*_uvTile*	the texture tile size in x and y

10.86.2.8 void gazebo::common::MeshManager::CreateSphere (const std::string & *_name*, float *_radius*, int *_rings*, int *_segments*)

Create a sphere mesh.

Parameters

in	*_name*	the name of the mesh
in	*_radius*	radius of the sphere in meter
in	*_rings*	number of circles on th y axis
in	*_segments*	number of segment per circle

10.86.2.9 void gazebo::common::MeshManager::CreateTube (const std::string & *_name*, float *_innerRadius*, float *_outterRadius*, float *_height*, int *_rings*, int *_segments*)

Create a tube mesh.

Generates rings inside and outside the cylinder Needs at least two rings and 3 segments

Parameters

in	*_name*	the name of the new mesh
in	*_innerRadius*	the inner radius of the tube in the x y plane
in	*_outterRadius*	the outer radius of the tube in the x y plane
in	*_height*	the height along z
in	*_rings*	the number of circles along the height
in	*_segments*	the number of segment per circle

10.86.2.10 void gazebo::common::MeshManager::GenSphericalTexCoord (const Mesh * _mesh, math::Vector3 _center)

generate spherical texture coordinates

10.86.2.11 const Mesh* gazebo::common::MeshManager::GetMesh (const std::string & _name) const

Get a mesh by name.

Parameters

in	_name	the name of the mesh to look for

Returns

the mesh or NULL if not found

10.86.2.12 void gazebo::common::MeshManager::GetMeshAABB (const Mesh * _mesh, math::Vector3 & _center, math::Vector3 & _min_xyz, math::Vector3 & _max_xyz)

Get mesh aabb and center.

Parameters

in	_mesh	the mesh
out	_center	the AAB center position
out	_min_xyz	the bounding box minimum
out	_max_xyz	the bounding box maximum

10.86.2.13 bool gazebo::common::MeshManager::HasMesh (const std::string & _name) const

Return true if the mesh exists.

Parameters

in	_name	the name of the mesh

10.86.2.14 bool gazebo::common::MeshManager::IsValidFilename (const std::string & _filename)

Checks a path extension against the list of valid extensions.

Returns

true if the file extension is loadable

10.86.2.15 const Mesh* gazebo::common::MeshManager::Load (const std::string & _filename)

Load a mesh from a file.

Parameters

in	*_filename*	the path to the mesh

Returns

a pointer to the created mesh

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

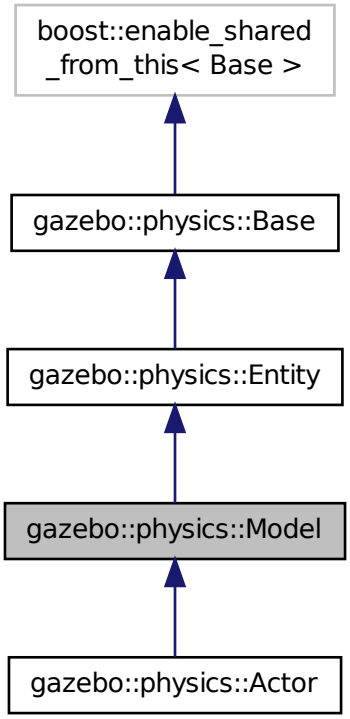
- **MeshManager.hh**

10.87 gazebo::physics::Model Class Reference

A (p. 111) model is a collection of links, joints, and plugins.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Model:



Public Member Functions

- **Model** (**BasePtr** _parent)

Constructor.

- virtual ~**Model** ()

  Destructor.

- void **AttachStaticModel** (**ModelPtr** &_model, **math::Pose** _offset)

  Attach a static model to this model.

- void **DetachStaticModel** (const std::string &_model)

  Detach a static model from this model.

- void **FillMsg** (msgs::Model &_msg)

  Fill a model message.

- virtual void **Fini** ()

  Finalize the model.

- bool **GetAutoDisable** () const

  Return the value of the SDF <allow_auto_disable> element.

- virtual **math::Box GetBoundingBox** () const

  Get the size of the bounding box.

- **JointPtr GetJoint** (const std::string &name)

  Get a joint.

- **JointControllerPtr GetJointController** ()

  Get a handle to the Controller for the joints in this model.

- unsigned int **GetJointCount** () const

  Get the number of joints.

- const **Joint_V** & **GetJoints** () const

  Get the joints.

- **LinkPtr GetLink** (const std::string &_name="canonical") const

  Get a link by name.

- **LinkPtr GetLinkById** (unsigned int _id) const

  This is an internal function

- **Link_V GetLinks** () const

  *Construct and return a vector of **Link** (p. 418)'s in this model Note this constructs the vector of **Link** (p. 418)'s on the fly, could be costly.*

- unsigned int **GetPluginCount** () const

  Get the number of plugins this model has.

- virtual **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the entity.

- virtual **math::Vector3 GetRelativeAngularVel** () const

  Get the angular velocity of the entity.

- virtual **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the entity.

- virtual **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the entity.

- virtual const **sdf::ElementPtr GetSDF** ()

  Get the SDF values for the model.

- unsigned int **GetSensorCount** () const

  Get the number of sensors attached to this model.

- virtual **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the entity in the world frame.

- virtual **math::Vector3 GetWorldAngularVel** () const

Get the angular velocity of the entity in the world frame.

- virtual **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the entity in the world frame.

- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the entity in the world frame.

- virtual void **Init** ()

  Initialize the model.

- void **Load** (**sdf::ElementPtr** _sdf)

  Load the model.

- void **LoadJoints** ()

  Load all the joints.

- void **LoadPlugins** ()

  Load all plugins.

- void **ProcessMsg** (const msgs::Model &_msg)

  Update parameters from a model message.

- virtual void **RemoveChild** (**EntityPtr** _child)

  Remove a child.

- void **Reset** ()

  Reset the model.

- void **SetAngularAccel** (const **math::Vector3** &_vel)

  Set the angular acceleration of the model, and all its links.

- void **SetAngularVel** (const **math::Vector3** &_vel)

  Set the angular velocity of the model, and all its links.

- void **SetAutoDisable** (bool _disable)

  Allow the model the auto disable.

- void **SetCollideMode** (const std::string &_mode)

  *This is not implemented in **Link** (p. 418), 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. 381) Animation.*

- void **SetJointPosition** (const std::string &_jointName, double _position)

  *Set the positions of a **Joint** (p. 381) 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. 489) by specifying desired Pose of a **Link** (p. 418) within the **Model** (p. 489).*

- void **SetLinkWorldPose** (const **math::Pose** &_pose, const **LinkPtr** &_link)

*Set the Pose of the entire **Model** (p. 489) by specifying desired Pose of a **Link** (p. 418) within the **Model** (p. 489).*

- 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. 493)*
- std::vector< **math::Pose** > **attachedModelsOffset**

  *used by **Model::AttachStaticModel** (p. 493)*

Additional Inherited Members

10.87.1 Detailed Description

A (p. 111) model is a collection of links, joints, and plugins.

10.87.2 Constructor & Destructor Documentation

10.87.2.1 gazebo::physics::Model::Model (**BasePtr** *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent object.

10.87.2.2 virtual gazebo::physics::Model::~Model ( ) [virtual]

Destructor.

10.87.3 Member Function Documentation

10.87.3.1 void gazebo::physics::Model::AttachStaticModel (ModelPtr & _model, math::Pose _offset)

Attach a static model to this model.

This function takes as input a static **Model** (p. 489), which is a **Model** (p. 489) that has been marked as static (no physics simulation), and attaches it to this **Model** (p. 489) with a given offset.

This function is useful when you want to simulate a grasp of a static object, or move a static object around using a dynamic model.

If you are in doubt, do not use this function.

Parameters

in	*_model*	Pointer to the static model.
in	*_offset*	Offset, relative to this **Model** (p. 489), to place _model.

10.87.3.2 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. 493).

10.87.3.3 void gazebo::physics::Model::FillMsg (msgs::Model & _msg)

Fill a model message.

Parameters

in	*_msg*	Message to fill using this model's data.

10.87.3.4 virtual void gazebo::physics::Model::Fini ( ) [virtual]

Finalize the model.

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

Reimplemented in **gazebo::physics::Actor** (p. 114).

10.87.3.5 bool gazebo::physics::Model::GetAutoDisable ( ) const

Return the value of the SDF <allow_auto_disable> element.

Returns

True if auto disable is allowed for this model.

10.87.3.6 virtual math::Box gazebo::physics::Model::GetBoundingBox ( ) const [virtual]

Get the size of the bounding box.

Returns

The bounding box.

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

10.87.3.7 JointPtr gazebo::physics::Model::GetJoint (const std::string & *name*)

Get a joint.

Parameters

name	The name of the joint, specified in the world file

Returns

Pointer to the joint

10.87.3.8 JointControllerPtr gazebo::physics::Model::GetJointController ( ) [inline]

Get a handle to the Controller for the joints in this model.

Returns

A (p. 111) handle to the Controller for the joints in this model.

10.87.3.9 unsigned int gazebo::physics::Model::GetJointCount ( ) const

Get the number of joints.

Returns

Get the number of joints.

10.87.3.10 const Joint_V& gazebo::physics::Model::GetJoints ( ) const

Get the joints.

Returns

Vector of joints.

10.87.3.11 LinkPtr gazebo::physics::Model::GetLink (const std::string & *_name* = "canonical") const

Get a link by name.

Parameters

in	*_name*	Name of the link to get.

Returns

Pointer to the link, NULL if the name is invalid.

10.87.3.12 LinkPtr gazebo::physics::Model::GetLinkById (unsigned int *_id*) const

This is an internal function

Get a link by id.

Returns

Pointer to the link, NULL if the id is invalid.

10.87.3.13 Link_V gazebo::physics::Model::GetLinks ( ) const

Construct and return a vector of **Link** (p. 418)'s in this model Note this constructs the vector of **Link** (p. 418)'s on the fly, could be costly.

Returns

a vector of **Link** (p. 418)'s in this model

10.87.3.14 unsigned int gazebo::physics::Model::GetPluginCount ( ) const

Get the number of plugins this model has.

Returns

Number of plugins associated with this model.

10.87.3.15 virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity.

Returns

math::Vector3 (p. 855), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 286).

10.87.3.16 virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the entity.

Returns

math::Vector3 (p. 855), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 286).

10.87.3.17 virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity.

Returns

math::Vector3 (p. 855), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 286).

10.87.3.18 virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the entity.

Returns

math::Vector3 (p. 855), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.87.3.19 virtual const sdf::ElementPtr gazebo::physics::Model::GetSDF ( ) [virtual]

Get the SDF values for the model.

Returns

The SDF value for this model.

Reimplemented from **gazebo::physics::Base** (p. 144).

Reimplemented in **gazebo::physics::Actor** (p. 114).

10.87.3.20 unsigned int gazebo::physics::Model::GetSensorCount ( ) const

Get the number of sensors attached to this model.

This will count all the sensors attached to all the links.

Returns

Number of sensors.

10.87.3.21 virtual math::Vector3 gazebo::physics::Model::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 855), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.87.3.22 virtual math::Vector3 gazebo::physics::Model::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the entity in the world frame.

Returns

math::Vector3 (p. 855), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.87.3.23 virtual math::Vector3 gazebo::physics::Model::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 855), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 287).

10.87.3.24 virtual math::Vector3 gazebo::physics::Model::GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the entity in the world frame.

Returns

math::Vector3 (p. 855), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 288).

10.87.3.25 virtual void gazebo::physics::Model::Init ( ) [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Base** (p. 145).

Reimplemented in **gazebo::physics::Actor** (p. 115).

10.87.3.26 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. 288).

10.87.3.27 void gazebo::physics::Model::LoadJoints ( )

Load all the joints.

10.87.3.28 void gazebo::physics::Model::LoadPlugins ( )

Load all plugins.

Load all plugins specified in the SDF for the model.

10.87.3.29 virtual void gazebo::physics::Model::OnPoseChange ( ) [protected], [virtual]

Callback when the pose of the model has been changed.

Implements **gazebo::physics::Entity** (p. 289).

10.87.3.30 void gazebo::physics::Model::ProcessMsg (const msgs::Model & *_msg*)

Update parameters from a model message.

Parameters

in	*_msg*	Message to process.

10.87.3.31 virtual void gazebo::physics::Model::RemoveChild (EntityPtr *_child*) [virtual]

Remove a child.

Parameters

in	*_child*	Remove a child entity.

10.87.3.32 void gazebo::physics::Model::Reset ( ) [virtual]

Reset the model.

Reimplemented from **gazebo::physics::Entity** (p. 289).

10.87.3.33 void gazebo::physics::Model::SetAngularAccel (const math::Vector3 & *_vel*)

Set the angular acceleration of the model, and all its links.

Parameters

in	*_vel*	The new angular acceleration

10.87.3.34 void gazebo::physics::Model::SetAngularVel (const math::Vector3 & *_vel*)

Set the angular velocity of the model, and all its links.

Parameters

in	*_vel*	The new angular velocity.

10.87.3.35 void gazebo::physics::Model::SetAutoDisable (bool *_disable*)

Allow the model the auto disable.

This is ignored if the model has joints.

Parameters

in	*_disable*	If true, the model is allowed to auto disable.

10.87.3.36 void gazebo::physics::Model::SetCollideMode (const std::string & *_mode*)

This is not implemented in **Link** (p. 418), which means this function doesn't do anything.

Set the collide mode of the model.

Parameters

in	*_mode*	The collision mode

10.87.3.37 void gazebo::physics::Model::SetEnabled (bool *_enabled*)

Enable all the links in all the models.

Parameters

in	*_enabled*	True to enable all the links.

10.87.3.38 void gazebo::physics::Model::SetGravityMode (const bool & *_value*)

Set the gravity mode of the model.

Parameters

in	*_value*	False to turn gravity on for the model.

10.87.3.39 void gazebo::physics::Model::SetJointAnimation (const std::map< std::string, common::NumericAnimationPtr > _anim, boost::function< void() > _onComplete = NULL)

Joint (p. 381) Animation.

Parameters

in	*_anim*	Map of joint names to their position animation.
in	*_onComplete*	Callback function for when the animation completes.

10.87.3.40 void gazebo::physics::Model::SetJointPosition (const std::string & _jointName, double _position)

Set the positions of a **Joint** (p. 381) by name.

See Also

JointController::SetJointPosition (p. 400)

Parameters

in	*_jointName*	Name of the joint to set.
in	*_position*	Position to set the joint to.

10.87.3.41 void gazebo::physics::Model::SetJointPositions (const std::map< std::string, double > & _jointPositions)

Set the positions of a set of joints.

See Also

JointController::SetJointPositions (p. 400).

Parameters

in	*_jointPositions*	Map of joint names to their positions.

10.87.3.42 void gazebo::physics::Model::SetLaserRetro (const float _retro)

Set the laser retro reflectiveness of the model.

Parameters

in	*_retro*	Retro reflectance value.

10.87.3.43 void gazebo::physics::Model::SetLinearAccel (const math::Vector3 & _vel)

Set the linear acceleration of the model, and all its links.

Parameters

in	*_vel*	The new linear acceleration.

10.87.3.44 void gazebo::physics::Model::SetLinearVel (const math::Vector3 & *_vel*)

Set the linear velocity of the model, and all its links.

Parameters

in	*_vel*	The new linear velocity.

10.87.3.45 void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & *_pose*, std::string *_linkName*)

Set the Pose of the entire **Model** (p. 489) by specifying desired Pose of a **Link** (p. 418) within the **Model** (p. 489).

Doing so, keeps the configuration of the **Model** (p. 489) unchanged, i.e. all **Joint** (p. 381) angles are unchanged.

Parameters

in	*_pose*	Pose to set the link to.
in	*_linkName*	Name of the link to set.

10.87.3.46 void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & *_pose*, const LinkPtr & *_link*)

Set the Pose of the entire **Model** (p. 489) by specifying desired Pose of a **Link** (p. 418) within the **Model** (p. 489).

Doing so, keeps the configuration of the **Model** (p. 489) unchanged, i.e. all **Joint** (p. 381) angles are unchanged.

Parameters

in	*_pose*	Pose to set the link to.
in	*_link*	Pointer to the link to set.

10.87.3.47 void gazebo::physics::Model::SetState (const ModelState & *_state*)

Set the current model state.

Parameters

in	*_state*	**State** (p. 758) to set the model to.

10.87.3.48 virtual void gazebo::physics::Model::StopAnimation ( ) [virtual]

Stop the current animations.

Reimplemented from **gazebo::physics::Entity** (p. 291).

10.87.3.49 void gazebo::physics::Model::Update ( ) [virtual]

Update the model.

Reimplemented from **gazebo::physics::Base** (p. 148).

10.87.3.50 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. 291).

Reimplemented in **gazebo::physics::Actor** (p. 115).

10.87.4 Member Data Documentation

10.87.4.1 std::vector<ModelPtr> gazebo::physics::Model::attachedModels [protected]

used by **Model::AttachStaticModel** (p. 493)

10.87.4.2 std::vector<math::Pose> gazebo::physics::Model::attachedModelsOffset [protected]

used by **Model::AttachStaticModel** (p. 493)

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

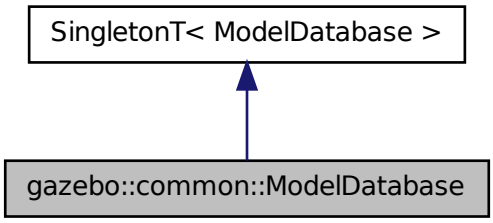
- **Model.hh**

10.88 gazebo::common::ModelDatabase Class Reference

Connects to model database, and has utility functions to find models.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::ModelDatabase:



Public Member Functions

- void **DownloadDependencies** (const std::string &_path)
  Download all dependencies for a give model path.
- std::string **GetDBConfig** (const std::string &_uri)
  Return the database.config file as a string.
- std::string **GetManifest** (const std::string &_uri) **GAZEBO_DEPRECATED**(1.5)
  Deprecated.
- std::string **GetModelConfig** (const std::string &_uri)
  Return the model.config file as a string.
- std::string **GetModelFile** (const std::string &_uri)
  Get a model's SDF file based on a URI.
- std::string **GetModelName** (const std::string &_uri)
  Get the name of a model based on a URI.
- std::string **GetModelPath** (const std::string &_uri, bool _forceDownload=false)
  Get the local path to a model.
- std::map< std::string, std::string > **GetModels** ()
  Returns the dictionary of all the model names.
- void **GetModels** (boost::function< void(const std::map< std::string, std::string > &)> _func)
  Get the dictionary of all model names via a callback.
- std::string **GetURI** ()
  Returns the the global model database URI.
- bool **HasModel** (const std::string &_modelName)
  Returns true if the model exists on the database.

Additional Inherited Members

10.88.1 Detailed Description

Connects to model database, and has utility functions to find models.

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

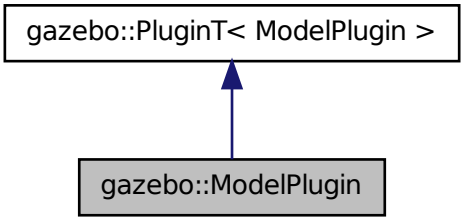
- **ModelDatabase.hh**

10.89 gazebo::ModelPlugin Class Reference

A (p. 111) plugin with access to **physics::Model** (p. 489).

`#include <Plugin.hh>`

Inheritance diagram for gazebo::ModelPlugin:



Public Member Functions

- **ModelPlugin** ()

  Constructor.
- virtual ~**ModelPlugin** ()

  Destructor.
- virtual void **Init** ()

  Override this method for custom plugin initialization behavior.
- virtual void **Load** (**physics::ModelPtr** _model, **sdf::ElementPtr** _sdf)=0

  Load function.
- virtual void **Reset** ()

  Override this method for custom plugin reset behavior.

Additional Inherited Members

10.89.1 Detailed Description

A (p. 111) plugin with access to **physics::Model** (p. 489).

See `reference`.

10.89.2 Constructor & Destructor Documentation

10.89.2.1 gazebo::ModelPlugin::ModelPlugin ( ) [inline]

Constructor.

References gazebo::MODEL_PLUGIN, and gazebo::PluginT< ModelPlugin >::type.

10.89.2.2 virtual gazebo::ModelPlugin::~ModelPlugin ( ) [inline], [virtual]

Destructor.

10.89.3 Member Function Documentation

10.89.3.1 virtual void gazebo::ModelPlugin::Init ( ) [inline], [virtual]

Override this method for custom plugin initialization behavior.

10.89.3.2 virtual void gazebo::ModelPlugin::Load (physics::ModelPtr *_model*, sdf::ElementPtr *_sdf*) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	*_model*	Pointer to the Model
in	*_sdf*	Pointer to the SDF element of the plugin.

10.89.3.3 virtual void gazebo::ModelPlugin::Reset ( ) [inline], [virtual]

Override this method for custom plugin reset behavior.

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

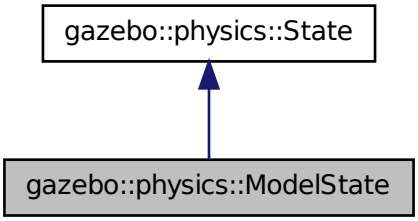
- **common/Plugin.hh**

10.90 gazebo::physics::ModelState Class Reference

Store state information of a **physics::Model** (p. 489) object.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::ModelState:



Public Member Functions

- **ModelState** ()

  Default constructor.
- **ModelState** (const **ModelPtr** _model)

  Constructor.
- **ModelState** (const **sdf::ElementPtr** _sdf)

  Constructor.
- virtual ∼**ModelState** ()

  Destructor.
- void **FillSDF** (**sdf::ElementPtr** _sdf)

  Populate a state SDF element with data from the object.
- **JointState GetJointState** (unsigned int _index) const

  *Get a **Joint** (p. 381) state.*
- **JointState GetJointState** (const std::string &_jointName) const

  *Get a **Joint** (p. 381) state by **Joint** (p. 381) name.*
- unsigned int **GetJointStateCount** () const

  Get the number of joint states.
- const std::vector< **JointState** > & **GetJointStates** () const

  Get the joint states.
- **LinkState GetLinkState** (unsigned int _index) const

  Get a link state.
- **LinkState GetLinkState** (const std::string &_linkName) const

  *Get a link state by **Link** (p. 418) name.*
- unsigned int **GetLinkStateCount** () const

  Get the number of link states.
- const std::vector< **LinkState** > & **GetLinkStates** () const

  Get the link states.
- const **math::Pose** & **GetPose** () const

  Get the stored model pose.
- bool **HasJointState** (const std::string &_jointName) const

Return true if there is a joint with the specified name.

- bool **HasLinkState** (const std::string &_linkName) const

  Return true if there is a link with the specified name.

- bool **IsZero** () const

  Return true if the values in the state are zero.

- virtual void **Load** (const **sdf::ElementPtr** _elem)

  Load state from SDF element.

- **ModelState operator+** (const **ModelState** &_state) const

  Addition operator.

- **ModelState operator-** (const **ModelState** &_state) const

  Subtraction operator.

- **ModelState** & **operator=** (const **ModelState** &_state)

  Assignment operator.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::ModelState** &_state)

  Stream insertion operator.

Additional Inherited Members

10.90.1 Detailed Description

Store state information of a **physics::Model** (p. 489) object.

This class captures the entire state of a **Model** (p. 489) at one specific time during a simulation run.

State (p. 758) of a **Model** (p. 489) includes the state of all its child Links and Joints.

10.90.2 Constructor & Destructor Documentation

10.90.2.1 gazebo::physics::ModelState::ModelState ( )

Default constructor.

10.90.2.2 gazebo::physics::ModelState::ModelState (const **ModelPtr** *_model*) [explicit]

Constructor.

Build a **ModelState** (p. 505) from an existing **Model** (p. 489).

Parameters

in	*_model*	Pointer to the model from which to gather state info.

10.90.2.3 gazebo::physics::ModelState::ModelState (const **sdf::ElementPtr** *_sdf*) [explicit]

Constructor.

Build a **ModelState** (p. 505) from SDF data

Parameters

in	*_sdf*	SDF data to load a model state from.

10.90.2.4 virtual gazebo::physics::ModelState::~ModelState ( ) [virtual]

Destructor.

10.90.3 Member Function Documentation

10.90.3.1 void gazebo::physics::ModelState::FillSDF (sdf::ElementPtr *_sdf*)

Populate a state SDF element with data from the object.

Parameters

out	*_sdf*	SDF element to populate.

10.90.3.2 JointState gazebo::physics::ModelState::GetJointState (unsigned int *_index*) const

Get a **Joint** (p. 381) state.

Return a **JointState** (p. 401) based on a index, where index is between 0...**ModelState::GetJointStateCount()** (p. 509).

Parameters

in	*_index*	Index of a **JointState** (p. 401).

Returns

State (p. 758) of a **Joint** (p. 381).

Exceptions

common::Exception *(p. 318)*	When _index is out of range.

10.90.3.3 JointState gazebo::physics::ModelState::GetJointState (const std::string & *_jointName*) const

Get a **Joint** (p. 381) state by **Joint** (p. 381) name.

Searches through all JointStates. Returns the **JointState** (p. 401) with the matching name, if any.

Parameters

in	*_jointName*	Name of the **JointState** (p. 401).

Returns

State (p. 758) of the **Joint** (p. 381).

Exceptions

common::Exception *(p. 318)*	When _jointName is invalid.

10.90.3.4 unsigned int gazebo::physics::ModelState::GetJointStateCount ( ) const

Get the number of joint states.

Returns the number of JointStates recorded.

Returns

Number of JointStates.

10.90.3.5 const std::vector<**JointState**>& gazebo::physics::ModelState::GetJointStates ( ) const

Get the joint states.

Returns

A (p. 111) vector of joint states.

10.90.3.6 **LinkState** gazebo::physics::ModelState::GetLinkState (unsigned int *_index*) const

Get a link state.

Get a **Link** (p. 418) **State** (p. 758) based on an index, where index is in the range of 0...**ModelState::GetLinkStateCount** (p. 510)

Parameters

in	*_index*	Index of the **LinkState** (p. 438)

Returns

State (p. 758) of the **Link** (p. 418).

Exceptions

common::Exception *(p. 318)*	When _index is out of range.

10.90.3.7 **LinkState** gazebo::physics::ModelState::GetLinkState (const std::string & *_linkName*) const

Get a link state by **Link** (p. 418) name.

Searches through all LinkStates. Returns the **LinkState** (p. 438) with the matching name, if any.

Parameters

in	*_linkName*	Name of the **LinkState** (p. 438)

Returns

State (p. 758) of the **Link** (p. 418).

Exceptions

common::Exception *(p. 318)*	When _linkName is invalid.

10.90.3.8 unsigned int gazebo::physics::ModelState::GetLinkStateCount ( ) const

Get the number of link states.

This returns the number of Links recorded.

Returns

Number of **LinkState** (p. 438) recorded.

10.90.3.9 const std::vector<LinkState>& gazebo::physics::ModelState::GetLinkStates ( ) const

Get the link states.

Returns

A (p. 111) vector of link states.

10.90.3.10 const math::Pose& gazebo::physics::ModelState::GetPose ( ) const

Get the stored model pose.

Returns

The **math::Pose** (p. 596) of the **Model** (p. 489).

10.90.3.11 bool gazebo::physics::ModelState::HasJointState (const std::string & *_jointName*) const

Return true if there is a joint with the specified name.

Parameters

in	*_jointName*	Name of the Jointtate.

Returns

True if the joint exists in the model.

10.90.3.12 bool gazebo::physics::ModelState::HasLinkState (const std::string & _linkName) const

Return true if there is a link with the specified name.

Parameters

in	*_linkName*	Name of the **LinkState** (p. 438).

Returns

True if the link exists in the model.

10.90.3.13 bool gazebo::physics::ModelState::IsZero ( ) const

Return true if the values in the state are zero.

Returns

True if the values in the state are zero.

10.90.3.14 virtual void gazebo::physics::ModelState::Load (const sdf::ElementPtr _elem) [virtual]

Load state from SDF element.

Load **ModelState** (p. 505) information from stored data in and SDF::Element

Parameters

in	*_elem*	Pointer to the SDF::Element containing state info.

Reimplemented from **gazebo::physics::State** (p. 761).

10.90.3.15 ModelState gazebo::physics::ModelState::operator+ (const ModelState & _state) const

Addition operator.

Parameters

in	*_pt*	**A** (p. 111) state to substract.

Returns

The resulting state.

10.90.3.16 ModelState gazebo::physics::ModelState::operator- (const ModelState & _state) const

Subtraction operator.

Parameters

in	*_pt*	**A** (p. 111) state to substract.

Returns

The resulting state.

10.90.3.17 ModelState& gazebo::physics::ModelState::operator= (const ModelState & _state)

Assignment operator.

Parameters

in	*_state*	**State** (p. 758) value

Returns

this

10.90.4 Friends And Related Function Documentation

10.90.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::physics::ModelState & _state) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream.
in	*_state*	**Model** (p. 489) state to output.

Returns

The stream.

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

- **ModelState.hh**

10.91 gazebo::common::MouseEvent Class Reference

Generic description of a mouse event.

`#include <common/common.hh>`

Public Types

- enum **Buttons** { **NO_BUTTON** = 0x0, **LEFT** = 0x1, **MIDDLE** = 0x2, **RIGHT** = 0x4 }

  Standard mouse buttons enumeration.
- enum **EventType** {
  NO_EVENT, **MOVE**, **PRESS**, **RELEASE**,
  SCROLL }

  Mouse event types enumeration.

Public Member Functions

- **MouseEvent** ()

  Constructor.

Public Attributes

- bool **alt**

  Alt key press flag.
- unsigned int **button**

  The button which caused the event.
- unsigned int **buttons**

  State of the buttons when the event was generated.
- bool **control**

  Control key press flag.
- bool **dragging**

  Flag for mouse drag motion.
- float **moveScale**

  Scaling factor.
- **math::Vector2i pos**

  Mouse pointer position on the screen.
- **math::Vector2i pressPos**

  Position of button press.
- **math::Vector2i prevPos**

  Previous position.
- **math::Vector2i scroll**

  Scroll position.
- bool **shift**

  Shift key press flag.
- **EventType type**

  Event type.

10.91.1 Detailed Description

Generic description of a mouse event.

10.91.2 Member Enumeration Documentation

10.91.2.1 enum gazebo::common::MouseEvent::Buttons

Standard mouse buttons enumeration.

Enumerator:

NO_BUTTON
LEFT
MIDDLE
RIGHT

10.91.2.2 enum gazebo::common::MouseEvent::EventType

Mouse event types enumeration.

Enumerator:

NO_EVENT
MOVE
PRESS
RELEASE
SCROLL

10.91.3 Constructor & Destructor Documentation

10.91.3.1 gazebo::common::MouseEvent::MouseEvent ( ) [inline]

Constructor.

10.91.4 Member Data Documentation

10.91.4.1 bool gazebo::common::MouseEvent::alt

Alt key press flag.

10.91.4.2 unsigned int gazebo::common::MouseEvent::button

The button which caused the event.

10.91.4.3 unsigned int gazebo::common::MouseEvent::buttons

State of the buttons when the event was generated.

10.91.4.4 bool gazebo::common::MouseEvent::control

Control key press flag.

10.91.4.5 bool gazebo::common::MouseEvent::dragging

Flag for mouse drag motion.

10.91.4.6 float gazebo::common::MouseEvent::moveScale

Scaling factor.

10.91.4.7 math::Vector2i gazebo::common::MouseEvent::pos

Mouse pointer position on the screen.

10.91.4.8 math::Vector2i gazebo::common::MouseEvent::pressPos

Position of button press.

10.91.4.9 math::Vector2i gazebo::common::MouseEvent::prevPos

Previous position.

10.91.4.10 math::Vector2i gazebo::common::MouseEvent::scroll

Scroll position.

10.91.4.11 bool gazebo::common::MouseEvent::shift

Shift key press flag.

10.91.4.12 EventType gazebo::common::MouseEvent::type

Event type.

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

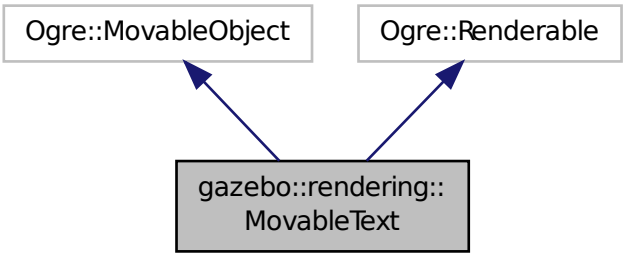
- **MouseEvent.hh**

10.92 gazebo::rendering::MovableText Class Reference

Movable text.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::MovableText:



Public Types

- enum **HorizAlign** { **H_LEFT**, **H_CENTER** }

  Horizontal alignment.
- enum **VertAlign** { **V_BELOW**, **V_ABOVE** }

  vertical alignment

Public Member Functions

- **MovableText** ()

  Constructor.
- virtual ~**MovableText** ()

  Destructor.
- **math::Box GetAABB** ()

  Get the axis aligned bounding box of the text.
- float **GetBaseline** () const

  Get the baseline height.
- float **GetCharHeight** () const

  Set the height of a characters return Height of the characters.
- const **common::Color** & **GetColor** () const

  Get the text color.
- const std::string & **GetFont** () const

  Get the font.
- bool **GetShowOnTop** () const

  True = text is displayed on top.
- float **GetSpaceWidth** () const

  Get the width of a space.
- const std::string & **GetText** () const

  Get the displayed text.
- void **Load** (const std::string &_name, const std::string &_text, const std::string &_fontName="Arial", float _charHeight=1.0, const **common::Color** &_color=**common::Color::White**)

Loads text and font info.

- void **SetBaseline** (float _height)

  Set the baseline height of the text.
- void **SetCharHeight** (float _height)

  Set the height of a character.
- void **SetColor** (const **common::Color** &_color)

  Set the text color.
- void **SetFontName** (const std::string &_font)

  Set the font.
- void **SetShowOnTop** (bool _show)

  True = text always is displayed ontop.
- void **SetSpaceWidth** (float _width)

  Set the width of a space.
- void **SetText** (const std::string &_text)

  Set the text to display.
- void **SetTextAlignment** (const **HorizAlign** &_hAlign, const **VertAlign** &_vAlign)

  Set the alignment of the text.
- void **Update** ()

  Update the text.
- virtual void **visitRenderables** (Ogre::Renderable::Visitor ∗_visitor, bool _debug=false)

Protected Member Functions

- void **_setupGeometry** ()
- void **_updateColors** ()
- float **getBoundingRadius** () const
- const Ogre::LightList & **getLights** (void) const
- const Ogre::MaterialPtr & **getMaterial** (void) const
- void **getRenderOperation** (Ogre::RenderOperation &op)
- float **getSquaredViewDepth** (const Ogre::Camera ∗cam) const
- void **getWorldTransforms** (Ogre::Matrix4 ∗xform) const

10.92.1 Detailed Description

Movable text.

10.92.2 Member Enumeration Documentation

10.92.2.1 enum gazebo::rendering::MovableText::HorizAlign

Horizontal alignment.

Enumerator:

H_LEFT Left alignment.

H_CENTER Center alignment.

10.92.2.2 enum gazebo::rendering::MovableText::VertAlign

vertical alignment

Enumerator:

V_BELOW Align below.

V_ABOVE Align above.

10.92.3 Constructor & Destructor Documentation

10.92.3.1 gazebo::rendering::MovableText::MovableText ( )

Constructor.

10.92.3.2 virtual gazebo::rendering::MovableText::~MovableText ( ) [virtual]

Destructor.

10.92.4 Member Function Documentation

10.92.4.1 void gazebo::rendering::MovableText::_setupGeometry ( ) [protected]

10.92.4.2 void gazebo::rendering::MovableText::_updateColors ( ) [protected]

10.92.4.3 math::Box gazebo::rendering::MovableText::GetAABB ( )

Get the axis aligned bounding box of the text.

Returns

The axis aligned bounding box.

10.92.4.4 float gazebo::rendering::MovableText::GetBaseline ( ) const

Get the baseline height.

Returns

Baseline height

10.92.4.5 float gazebo::rendering::MovableText::getBoundingRadius ( ) const [protected]

10.92.4.6 float gazebo::rendering::MovableText::GetCharHeight ( ) const

Set the height of a characters return Height of the characters.

10.92.4.7 const common::Color& gazebo::rendering::MovableText::GetColor ( ) const

Get the text color.

Returns

Texture color.

10.92.4.8 const std::string& gazebo::rendering::MovableText::GetFont ( ) const

Get the font.

Returns

The font name

10.92.4.9 const Ogre::LightList& gazebo::rendering::MovableText::getLights (void) const [protected]

10.92.4.10 const Ogre::MaterialPtr& gazebo::rendering::MovableText::getMaterial (void) const [protected]

10.92.4.11 void gazebo::rendering::MovableText::getRenderOperation (Ogre::RenderOperation & *op*) [protected]

10.92.4.12 bool gazebo::rendering::MovableText::GetShowOnTop ( ) const

True = text is displayed on top.

Returns

True if MovableText::SetShownOnTop(true) was called.

10.92.4.13 float gazebo::rendering::MovableText::GetSpaceWidth ( ) const

Get the width of a space.

Returns

Space width

10.92.4.14 float gazebo::rendering::MovableText::getSquaredViewDepth (const Ogre::Camera * *cam*) const [protected]

10.92.4.15 const std::string& gazebo::rendering::MovableText::GetText ( ) const

Get the displayed text.

Returns

The displayed text.

10.92.4.16 void gazebo::rendering::MovableText::getWorldTransforms (Ogre::Matrix4 * *xform*) const [protected]

10.92.4.17 void gazebo::rendering::MovableText::Load (const std::string & *_name*, const std::string & *_text*, const std::string & *_fontName* = "Arial", float *_charHeight* = 1.0, const common::Color & *_color* = common::Color::White)

Loads text and font info.

Parameters

in	*_name*	Name of the text object
in	*_text*	Text to render
in	*_fontName*	Font to use
in	*_charHeight*	Height of the characters
in	*_color*	Text color

10.92.4.18 void gazebo::rendering::MovableText::SetBaseline (float *_height*)

Set the baseline height of the text.

Parameters

in	*_height*	Baseline height

10.92.4.19 void gazebo::rendering::MovableText::SetCharHeight (float *_height*)

Set the height of a character.

Parameters

in	*_height*	Height of the characters.

10.92.4.20 void gazebo::rendering::MovableText::SetColor (const common::Color & *_color*)

Set the text color.

Parameters

in	*_color*	Text color.

10.92.4.21 void gazebo::rendering::MovableText::SetFontName (const std::string & *_font*)

Set the font.

Parameters

in	*_font*	Name of the font

10.92.4.22 void gazebo::rendering::MovableText::SetShowOnTop (bool *_show*)

True = text always is displayed ontop.

Parameters

in	*_show*	Set to true to render the text on top of all other drawables.

10.92.4.23 void gazebo::rendering::MovableText::SetSpaceWidth (float *_width*)

Set the width of a space.

Parameters

in	*_width*	space width

10.92.4.24 void gazebo::rendering::MovableText::SetText (const std::string & *_text*)

Set the text to display.

Parameters

in	*_text*	The text to display.

10.92.4.25 void gazebo::rendering::MovableText::SetTextAlignment (const HorizAlign & *_hAlign,* const VertAlign & *_vAlign*)

Set the alignment of the text.

Parameters

in	*_hAlign*	Horizontal alignment
in	*_vAlign*	Vertical alignment

10.92.4.26 void gazebo::rendering::MovableText::Update ( )

Update the text.

10.92.4.27 virtual void gazebo::rendering::MovableText::visitRenderables (Ogre::Renderable::Visitor * *_visitor,* bool *_debug* = `false`) [virtual]

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

- **MovableText.hh**

10.93 gazebo::msgs::MsgFactory Class Reference

A (p. 111) factory that generates protobuf message based on a string type.

`#include <msgs/msgs.hh>`

Static Public Member Functions

- static void **GetMsgTypes** (std::vector< std::string > &_types)

  Get all the message types.
- static google::protobuf::Message * **NewMsg** (const std::string &_msgType)

  Create a new instance of a message.
- static void **RegisterMsg** (const std::string &_msgType, **MsgFactoryFn** _factoryfn)

  Register a message.

10.93.1 Detailed Description

A (p. 111) factory that generates protobuf message based on a string type.

10.93.2 Member Function Documentation

10.93.2.1 static void gazebo::msgs::MsgFactory::GetMsgTypes (std::vector< std::string > & *_types*) [static]

Get all the message types.

Parameters

out	*_types*	Vector of strings of the message types.

10.93.2.2 static google::protobuf::Message* gazebo::msgs::MsgFactory::NewMsg (const std::string & *_msgType*) [static]

Create a new instance of a message.

Parameters

in	*_msgType*	Type of message to create.

Returns

Pointer to a google protobuf message. Null if the message type could not be handled.

10.93.2.3 static void gazebo::msgs::MsgFactory::RegisterMsg (const std::string & *_msgType,* **MsgFactoryFn** *_factoryfn*) [static]

Register a message.

Parameters

in	*_msgType*	Type of message to register.
in	*_factoryfn*	Function that generates the message.

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

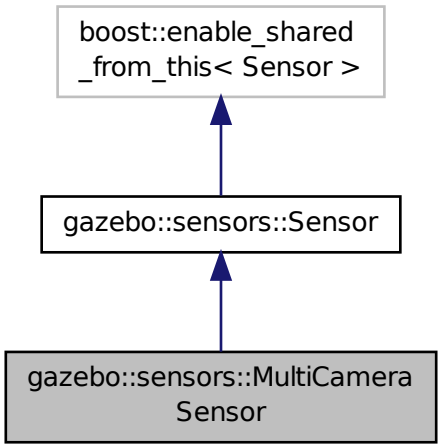
- **MsgFactory.hh**

10.94 gazebo::sensors::MultiCameraSensor Class Reference

Multiple camera sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::MultiCameraSensor:



Public Member Functions

- **MultiCameraSensor** ()
  Constructor.
- virtual ~**MultiCameraSensor** ()
  Destructor.
- **rendering::CameraPtr GetCamera** (unsigned int _index) const
  *Returns a pointer to a **rendering::Camera** (p. 162).*
- unsigned int **GetCameraCount** () const
  Get the number of cameras.
- const unsigned char * **GetImageData** (unsigned int _index)
  Gets the raw image data from the sensor.
- unsigned int **GetImageHeight** (unsigned int _index) const
  Gets the height of the image in pixels.
- unsigned int **GetImageWidth** (unsigned int _index) const
  Gets the width of the image in pixels.
- virtual std::string **GetTopic** () const
  Returns the topic name as set in SDF.

- virtual void **Init** ()

  Initialize the sensor.
- virtual bool **IsActive** ()

  Returns true if sensor generation is active.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.
- bool **SaveFrame** (const std::vector< std::string > &_filenames)

  Saves the camera image(s) to the disk.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the sensor.
- virtual void **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

Additional Inherited Members

10.94.1 Detailed Description

Multiple camera sensor.

This sensor type can create one or more synchronized cameras.

10.94.2 Constructor & Destructor Documentation

10.94.2.1 gazebo::sensors::MultiCameraSensor::MultiCameraSensor ( )

Constructor.

10.94.2.2 virtual gazebo::sensors::MultiCameraSensor::~MultiCameraSensor ( ) [virtual]

Destructor.

10.94.3 Member Function Documentation

10.94.3.1 virtual void gazebo::sensors::MultiCameraSensor::Fini ( ) [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 702).

10.94.3.2 rendering::CameraPtr gazebo::sensors::MultiCameraSensor::GetCamera (unsigned int *_index*) const

Returns a pointer to a **rendering::Camera** (p. 162).

Parameters

in	*_index*	Index of the camera to get

Returns

The Pointer to the camera sensor.

See Also

MultiCameraSensor::GetCameraCount (p. 525)

10.94.3.3 unsigned int gazebo::sensors::MultiCameraSensor::GetCameraCount ( ) const

Get the number of cameras.

Returns

The number of cameras.

10.94.3.4 const unsigned char* gazebo::sensors::MultiCameraSensor::GetImageData (unsigned int *_index*)

Gets the raw image data from the sensor.

Parameters

in	*_index*	Index of the camera

Returns

The pointer to the image data array.

See Also

MultiCameraSensor::GetCameraCount (p. 525)

10.94.3.5 unsigned int gazebo::sensors::MultiCameraSensor::GetImageHeight (unsigned int *_index*) const

Gets the height of the image in pixels.

Parameters

in	*_index*	Index of the camera

Returns

The image height in pixels.

See Also

MultiCameraSensor::GetCameraCount (p. 525)

10.94.3.6 unsigned int gazebo::sensors::MultiCameraSensor::GetImageWidth (unsigned int *_index*) const

Gets the width of the image in pixels.

Parameters

in	*_index*	Index of the camera

Returns

The image width in pixels.

See Also

MultiCameraSensor::GetCameraCount (p. 525)

10.94.3.7 virtual std::string gazebo::sensors::MultiCameraSensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 703).

10.94.3.8 virtual void gazebo::sensors::MultiCameraSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.94.3.9 virtual bool gazebo::sensors::MultiCameraSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.94.3.10 virtual void gazebo::sensors::MultiCameraSensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

10.94.3.11 bool gazebo::sensors::MultiCameraSensor::SaveFrame (const std::vector< std::string > & *_filenames*)

Saves the camera image(s) to the disk.

Parameters

in	*_filenames*	The name of the files for each camera.

Returns

True if successful, false if unsuccessful.

See Also

MultiCameraSensor::GetCameraCount (p. 525)

10.94.3.12 virtual void gazebo::sensors::MultiCameraSensor::UpdateImpl (bool) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 706).

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

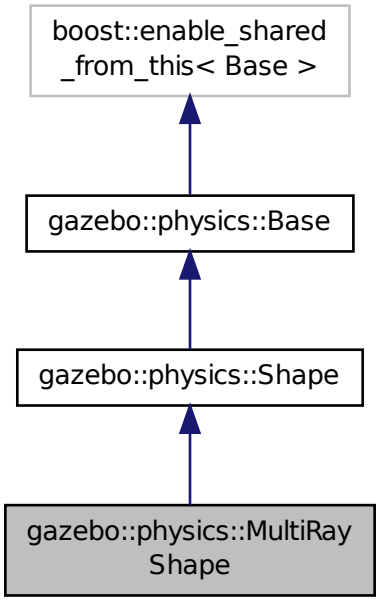
- **MultiCameraSensor.hh**

10.95 gazebo::physics::MultiRayShape Class Reference

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::MultiRayShape:



Public Member Functions

- **MultiRayShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**MultiRayShape** ()

  Destructor.
- template<typename T >
  event::ConnectionPtr ConnectNewLaserScans (T _subscriber)

  Connect a to the new laser scan signal.
- void **DisconnectNewLaserScans** (**event::ConnectionPtr** &_conn)

  Disconnect from the new laser scans signal.
- void **FillMsg** (msgs::Geometry &_msg)

  This function is not implemented.
- int **GetFiducial** (int _index)

  Get detected fiducial value for a ray.
- **math::Angle GetMaxAngle** () const

  Get the maximum angle.
- double **GetMaxRange** () const

  Get the maximum range.
- **math::Angle GetMinAngle** () const

  Get the minimum angle.

- double **GetMinRange** () const

  Get the minimum range.
- double **GetRange** (int _index)

  Get detected range for a ray.
- double **GetResRange** () const

  Get the range resolution.
- double **GetRetro** (int _index)

  Get detected retro (intensity) value for a ray.
- int **GetSampleCount** () const

  Get the horizontal sample count.
- double **GetScanResolution** () const

  Get the horizontal resolution.
- **math::Angle GetVerticalMaxAngle** () const

  Get the vertical max angle.
- **math::Angle GetVerticalMinAngle** () const

  Get the vertical min angle.
- int **GetVerticalSampleCount** () const

  Get the vertical sample count.
- double **GetVerticalScanResolution** () const

  Get the vertical range resolution.
- virtual void **Init** ()

  Init the shape.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  This function is not implemented.
- void **Update** ()

  Update the ray collisions.

Protected Member Functions

- virtual void **AddRay** (const **math::Vector3** &_start, const **math::Vector3** &_end)

  Add a ray to the collision.
- virtual void **UpdateRays** ()=0

  Physics engine specific method for updating the rays.

Protected Attributes

- **sdf::ElementPtr horzElem**

  Horizontal SDF element pointer.
- **event::EventT**< void()> **newLaserScans**

  New laser scans event.
- **math::Pose offset**

  Pose offset of all the rays.
- **sdf::ElementPtr rangeElem**

  Range SDF element pointer.
- **sdf::ElementPtr rayElem**

  Ray SDF element pointer.

- std::vector< **RayShapePtr** > **rays**

  Ray data.
- **sdf::ElementPtr scanElem**

  Scan SDF element pointer.
- **sdf::ElementPtr vertElem**

  Vertical SDF element pointer.

Additional Inherited Members

10.95.1 Detailed Description

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

10.95.2 Constructor & Destructor Documentation

10.95.2.1 gazebo::physics::MultiRayShape::MultiRayShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent collision shape.

10.95.2.2 virtual gazebo::physics::MultiRayShape::~MultiRayShape ( ) [virtual]

Destructor.

10.95.3 Member Function Documentation

10.95.3.1 virtual void gazebo::physics::MultiRayShape::AddRay (const math::Vector3 & *_start*, const math::Vector3 & *_end*) [protected], [virtual]

Add a ray to the collision.

Parameters

in	*_start*	Start of the ray.
in	*_end*	End of the ray.

10.95.3.2 template<typename T > event::ConnectionPtr gazebo::physics::MultiRayShape::ConnectNewLaserScans (T *_subscriber*) [inline]

Connect a to the new laser scan signal.

Parameters

in	*_subscriber*	Callback function.

Returns

The connection, which must be kept in scope.

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

10.95.3.3 void gazebo::physics::MultiRayShape::DisconnectNewLaserScans (event::ConnectionPtr & _conn) [inline]

Disconnect from the new laser scans signal.

Parameters

in	*_conn*	Connection to remove.

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

10.95.3.4 void gazebo::physics::MultiRayShape::FillMsg (msgs::Geometry & _msg) [virtual]

This function is not implemented.

Fill a message with this shape's values.

Parameters

out	*_msg*	Message that contains the shape's values.

Implements **gazebo::physics::Shape** (p. 722).

10.95.3.5 int gazebo::physics::MultiRayShape::GetFiducial (int _index)

Get detected fiducial value for a ray.

Parameters

in	*_index*	Index of the ray.

Returns

Fiducial value for the ray.

10.95.3.6 math::Angle gazebo::physics::MultiRayShape::GetMaxAngle ( ) const

Get the maximum angle.

Returns

Maximum angle of ray scan.

10.95.3.7 double gazebo::physics::MultiRayShape::GetMaxRange ( ) const

Get the maximum range.

Returns

Maximum range of all the rays.

10.95.3.8 math::Angle gazebo::physics::MultiRayShape::GetMinAngle ( ) const

Get the minimum angle.

Returns

Minimum angle of ray scan.

10.95.3.9 double gazebo::physics::MultiRayShape::GetMinRange ( ) const

Get the minimum range.

Returns

Minimum range of all the rays.

10.95.3.10 double gazebo::physics::MultiRayShape::GetRange (int *_index*)

Get detected range for a ray.

Parameters

in	*_index*	Index of the ray.

Returns

Returns DBL_MAX for no detection.

10.95.3.11 double gazebo::physics::MultiRayShape::GetResRange ( ) const

Get the range resolution.

Returns

Range resolution of all the rays.

10.95.3.12 double gazebo::physics::MultiRayShape::GetRetro (int *_index*)

Get detected retro (intensity) value for a ray.

Parameters

in	*_index*	Index of the ray.

Returns

Retro value for the ray.

10.95.3.13 int gazebo::physics::MultiRayShape::GetSampleCount ( ) const

Get the horizontal sample count.

Returns

Horizontal sample count.

10.95.3.14 double gazebo::physics::MultiRayShape::GetScanResolution ( ) const

Get the horizontal resolution.

Returns

Horizontal resolution.

10.95.3.15 math::Angle gazebo::physics::MultiRayShape::GetVerticalMaxAngle ( ) const

Get the vertical max angle.

Returns

Verticam max angle.

10.95.3.16 math::Angle gazebo::physics::MultiRayShape::GetVerticalMinAngle ( ) const

Get the vertical min angle.

Returns

Vertical min angle.

10.95.3.17 int gazebo::physics::MultiRayShape::GetVerticalSampleCount ( ) const

Get the vertical sample count.

Returns

Verical sample count.

10.95.3.18 double gazebo::physics::MultiRayShape::GetVerticalScanResolution ( ) const

Get the vertical range resolution.

Returns

Vertical range resolution.

10.95.3.19 virtual void gazebo::physics::MultiRayShape::Init ( ) [virtual]

Init the shape.

Implements **gazebo::physics::Shape** (p. 722).

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

This function is not implemented.

Update the ray based on a message.

Parameters

in	*_msg*	Message to update from.

Implements **gazebo::physics::Shape** (p. 722).

10.95.3.21 void gazebo::physics::MultiRayShape::Update ( ) [virtual]

Update the ray collisions.

Reimplemented from **gazebo::physics::Base** (p. 148).

10.95.3.22 virtual void gazebo::physics::MultiRayShape::UpdateRays ( ) [protected], [pure virtual]

Physics engine specific method for updating the rays.

10.95.4 Member Data Documentation

10.95.4.1 sdf::ElementPtr gazebo::physics::MultiRayShape::horzElem [protected]

Horizontal SDF element pointer.

10.95.4.2 event::EventT<void()> gazebo::physics::MultiRayShape::newLaserScans [protected]

New laser scans event.

Referenced by ConnectNewLaserScans(), and DisconnectNewLaserScans().

10.95.4.3 math::Pose gazebo::physics::MultiRayShape::offset [protected]

Pose offset of all the rays.

10.95.4.4 sdf::ElementPtr gazebo::physics::MultiRayShape::rangeElem [protected]

Range SDF element pointer.

10.95.4.5 sdf::ElementPtr gazebo::physics::MultiRayShape::rayElem [protected]

Ray SDF element pointer.

10.95.4.6 std::vector<RayShapePtr> gazebo::physics::MultiRayShape::rays [protected]

Ray data.

10.95.4.7 sdf::ElementPtr gazebo::physics::MultiRayShape::scanElem [protected]

Scan SDF element pointer.

10.95.4.8 sdf::ElementPtr gazebo::physics::MultiRayShape::vertElem [protected]

Vertical SDF element pointer.

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

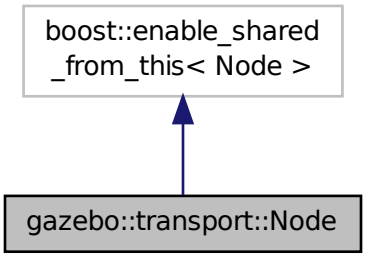
- **MultiRayShape.hh**

10.96 gazebo::transport::Node Class Reference

A (p. 111) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::Node:



Public Member Functions

- **Node** ()

  Constructor.
- virtual ~**Node** ()

  Destructor.

- template<typename M >
  transport::PublisherPtr Advertise (const std::string &_topic, unsigned int _queueLimit=1000, double _hzRate=0)

  Adverise a topic.
- std::string **DecodeTopicName** (const std::string &_topic)

  Decode a topic name.
- std::string **EncodeTopicName** (const std::string &_topic)

  Encode a topic name.
- void **Fini** ()

  Finalize the node.
- unsigned int **GetId** () const

  Get the unique ID of the node.
- std::string **GetMsgType** (const std::string &_topic) const

  Get the message type for a topic.
- std::string **GetTopicNamespace** () const

  Get the topic namespace for this node.
- bool **HandleData** (const std::string &_topic, const std::string &_msg)

  Handle incoming data.
- bool **HandleMessage** (const std::string &_topic, **MessagePtr** _msg)

  Handle incoming msg.
- bool **HasLatchedSubscriber** (const std::string &_topic) const

  Return true if a subscriber on a specific topic is latched.
- void **Init** (const std::string &_space="")

  Init the node.
- void **InsertLatchedMsg** (const std::string &_topic, const std::string &_msg)

  Add a latched message to the node for publication.
- void **InsertLatchedMsg** (const std::string &_topic, **MessagePtr** _msg)

  Add a latched message to the node for publication.
- void **ProcessIncoming** ()

  Process incoming messages.
- void **ProcessPublishers** ()

  Process all publishers, which has each publisher send it's most recent message over the wire.
- template<typename M >
  void **Publish** (const std::string &_topic, const google::protobuf::Message &_message)

  A *(*p. *111**) convenience function for a one-time publication of a message.*
- void **RemoveCallback** (const std::string &_topic, unsigned int _id)
- template<typename M , typename T >
  SubscriberPtr Subscribe (const std::string &_topic, void(T::*_fp)(const M const *&), T *_obj, bool _latching=false)

  Subscribe to a topic using a class method as the callback.
- template<typename M >
  SubscriberPtr Subscribe (const std::string &_topic, void(*_fp)(const M const *&), bool _latching=false)

  Subscribe to a topic using a bare function as the callback.
- template<typename T >
  SubscriberPtr Subscribe (const std::string &_topic, void(T::*_fp)(const std::string &), T *_obj, bool _latching=false)

  Subscribe to a topic using a class method as the callback.
- **SubscriberPtr Subscribe** (const std::string &_topic, void(*_fp)(const std::string &), bool _latching=false)

  Subscribe to a topic using a bare function as the callback.

10.96.1 Detailed Description

A (p. 111) node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

10.96.2 Constructor & Destructor Documentation

10.96.2.1 gazebo::transport::Node::Node ( )

Constructor.

10.96.2.2 virtual gazebo::transport::Node::~Node ( ) [virtual]

Destructor.

10.96.3 Member Function Documentation

10.96.3.1 template<typename M > transport::PublisherPtr gazebo::transport::Node::Advertise (const std::string & *_topic*, unsigned int *_queueLimit* = 1000, double *_hzRate* = 0) [inline]

Adverise a topic.

Parameters

in	*_topic*	The topic to advertise
in	*_queueLimit*	The maximum number of outgoing messages to queue for delivery
in	*_hz*	Update rate for the publisher. Units are 1.0/seconds.

Returns

Pointer to new publisher object

References DecodeTopicName(), and SingletonT< T >::Instance().

10.96.3.2 std::string gazebo::transport::Node::DecodeTopicName (const std::string & *_topic*)

Decode a topic name.

Parameters

in	*The*	encoded name

Returns

The decoded name

Referenced by Advertise(), and Subscribe().

10.96.3.3 std::string gazebo::transport::Node::EncodeTopicName (const std::string & *_topic*)

Encode a topic name.

Parameters

in	*The*	decoded name

Returns

The encoded name

10.96.3.4 void gazebo::transport::Node::Fini ( )

Finalize the node.

10.96.3.5 unsigned int gazebo::transport::Node::GetId ( ) const

Get the unique ID of the node.

Returns

The unique ID of the node

Referenced by Subscribe().

10.96.3.6 std::string gazebo::transport::Node::GetMsgType (const std::string & *_topic*) const

Get the message type for a topic.

Parameters

in	*_topic*	The topic

Returns

The message type

10.96.3.7 std::string gazebo::transport::Node::GetTopicNamespace ( ) const

Get the topic namespace for this node.

Returns

The namespace

10.96.3.8 bool gazebo::transport::Node::HandleData (const std::string & *_topic*, const std::string & *_msg*)

Handle incoming data.

Parameters

in	*_topic*	Topic for which the data was received
in	*_msg*	The message that was received

Returns

true if the message was handled successfully, false otherwise

10.96.3.9 bool gazebo::transport::Node::HandleMessage (const std::string & _topic, MessagePtr _msg)

Handle incoming msg.

Parameters

in	*_topic*	Topic for which the data was received
in	*_msg*	The message that was received

Returns

true if the message was handled successfully, false otherwise

10.96.3.10 bool gazebo::transport::Node::HasLatchedSubscriber (const std::string & _topic) const

Return true if a subscriber on a specific topic is latched.

Parameters

in	*_topic*	Name of the topic to check.

Returns

True if a latched subscriber exists.

10.96.3.11 void gazebo::transport::Node::Init (const std::string & _space = " ")

Init the node.

Parameters

in	*_space*	Set the global namespace of all topics. If left blank, the topic will initialize to the first namespace on the **Master** (p. 452)

10.96.3.12 void gazebo::transport::Node::InsertLatchedMsg (const std::string & _topic, const std::string & _msg)

Add a latched message to the node for publication.

This is called when a subscription is connected to a publication.

Parameters

in	*_topic*	Name of the topic to publish data on.
in	*_msg*	The message to publish.

10.96.3.13 void gazebo::transport::Node::InsertLatchedMsg (const std::string & *_topic*, MessagePtr *_msg*)

Add a latched message to the node for publication.

This is called when a subscription is connected to a publication.

Parameters

in	*_topic*	Name of the topic to publish data on.
in	*_msg*	The message to publish.

10.96.3.14 void gazebo::transport::Node::ProcessIncoming ( )

Process incoming messages.

10.96.3.15 void gazebo::transport::Node::ProcessPublishers ( )

Process all publishers, which has each publisher send it's most recent message over the wire.

This is for internal use only

10.96.3.16 template<typename M > void gazebo::transport::Node::Publish (const std::string & *_topic*, const google::protobuf::Message & *_message*) [inline]

A (p. 111) convenience function for a one-time publication of a message.

This is inefficient, compared to **Node::Advertise** (p. 537) followed by **Publisher::Publish** (p. 621). This function should only be used when sending a message very infrequently.

Parameters

in	*_topic*	The topic to advertise
in	*_message*	Message to be published

10.96.3.17 void gazebo::transport::Node::RemoveCallback (const std::string & *_topic*, unsigned int *_id*)

10.96.3.18 template<typename M , typename T > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & *_topic*, void(T::*)(const M const *&) *_fp*, T * *_obj*, bool *_latching* = false) [inline]

Subscribe to a topic using a class method as the callback.

Parameters

in	*_topic*	The topic to subscribe to
in	*_fp*	Class method to be called on receipt of new message
in	*_obj*	Class instance to be used on receipt of new message
in	*_latching*	If true, latch latest incoming message; otherwise don't latch

Returns

Pointer to new **Subscriber** (p. 776) object

References DecodeTopicName(), GetId(), SingletonT< T >::Instance(), and gazebo::transport::Subscriber::SetCallbackId().

10.96.3.19 template<typename M > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & _topic, void(*)(const M const *&) _fp, bool _latching = false) [inline]

Subscribe to a topic using a bare function as the callback.

Parameters

in	*_topic*	The topic to subscribe to
in	*_fp*	Function to be called on receipt of new message
in	*_latching*	If true, latch latest incoming message; otherwise don't latch

Returns

Pointer to new **Subscriber** (p. 776) object

References DecodeTopicName(), GetId(), SingletonT< T >::Instance(), and gazebo::transport::Subscriber::SetCallbackId().

10.96.3.20 template<typename T > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & _topic, void(T::*)(const std::string &) _fp, T * _obj, bool _latching = false) [inline]

Subscribe to a topic using a class method as the callback.

Parameters

in	*_topic*	The topic to subscribe to
in	*_fp*	Class method to be called on receipt of new message
in	*_obj*	Class instance to be used on receipt of new message
in	*_latching*	If true, latch latest incoming message; otherwise don't latch

Returns

Pointer to new **Subscriber** (p. 776) object

References DecodeTopicName(), GetId(), gazebo::transport::SubscribeOptions::Init(), SingletonT< T >::Instance(), and gazebo::transport::Subscriber::SetCallbackId().

10.96.3.21 SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & _topic, void(*)(const std::string &) _fp, bool _latching = false) [inline]

Subscribe to a topic using a bare function as the callback.

Parameters

in	*_topic*	The topic to subscribe to
in	*_fp*	Function to be called on receipt of new message
in	*_latching*	If true, latch latest incoming message; otherwise don't latch

Returns

Pointer to new **Subscriber** (p. 776) object

References DecodeTopicName(), GetId(), gazebo::transport::SubscribeOptions::Init(), SingletonT< T >::Instance(), and gazebo::transport::Subscriber::SetCallbackId().

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

- **Node.hh**

10.97 gazebo::common::NodeAnimation Class Reference

Node animation.

`#include <common/common.hh>`

Public Member Functions

- **NodeAnimation** (const std::string &_name)

  constructor
- ~**NodeAnimation** ()

  Destructor. It empties the key frames list.
- void **AddKeyFrame** (const double _time, const **math::Matrix4** _trans)

  Adds a key frame at a specific time.
- void **AddKeyFrame** (const double _time, const **math::Pose** _pose)

  Adds a key fram at a specific time.
- **math::Matrix4 GetFrameAt** (double _time, bool _loop=true) const

  Returns a frame transformation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.
- unsigned int **GetFrameCount** () const

  Returns the number of key frames.
- void **GetKeyFrame** (const unsigned int _i, double &_time, **math::Matrix4** &_trans) const

  Finds a key frame using the index.
- std::pair< double, **math::Matrix4** > **GetKeyFrame** (const unsigned int _i) const

  Returns a key frame using the index.
- double **GetLength** () const

  Returns the duration of the animations.
- std::string **GetName** () const

  Returns the name.
- double **GetTimeAtX** (const double _x) const

  Returns the time where a transformation's translational value along the X axis is equal to _x.
- void **Scale** (const double _scale)

  Scales each transformation in the key frames.
- void **SetName** (const std::string &_name)

  Changes the name of the animation.

Protected Attributes

- std::map< double, **math::Matrix4** > **keyFrames**

  the dictionary of key frames, indexed by time
- double **length**

  the duration of the animations (time of last key frame)
- std::string **name**

  the name of the animation

10.97.1 Detailed Description

Node animation.

10.97.2 Constructor & Destructor Documentation

10.97.2.1 gazebo::common::NodeAnimation::NodeAnimation (const std::string & *_name*)

constructor

Parameters

in	*_name*	the name of the node

10.97.2.2 gazebo::common::NodeAnimation::~NodeAnimation ( )

Destructor. It empties the key frames list.

10.97.3 Member Function Documentation

10.97.3.1 void gazebo::common::NodeAnimation::AddKeyFrame (const double *_time,* const **math::Matrix4** *_trans*)

Adds a key frame at a specific time.

Parameters

in	*_time*	the time of the key frame
in	*_trans*	the transformation

10.97.3.2 void gazebo::common::NodeAnimation::AddKeyFrame (const double *_time,* const **math::Pose** *_pose*)

Adds a key fram at a specific time.

Parameters

in	*_time*	the tiem of the key frame
in	*_pose*	the pose

10.97.3.3 math::Matrix4 gazebo::common::NodeAnimation::GetFrameAt (double *_time*, bool *_loop* = true) const

Returns a frame transformation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.

Parameters

in	*_time*	the time
in	*_loop*	when true, the time is divided by the duration (see GetLength)

10.97.3.4 unsigned int gazebo::common::NodeAnimation::GetFrameCount ( ) const

Returns the number of key frames.

Returns

the count

10.97.3.5 void gazebo::common::NodeAnimation::GetKeyFrame (const unsigned int *_i*, double & *_time*, math::Matrix4 & *_trans*) const

Finds a key frame using the index.

Note the index of a key frame can change as frames are added.

Parameters

in	*_i*	the index
out	*_time*	the time of the frame, or -1 if the index id is out of bounds
out	*_trans*	the transformation for this key frame

10.97.3.6 std::pair<double, math::Matrix4> gazebo::common::NodeAnimation::GetKeyFrame (const unsigned int *_i*) const

Returns a key frame using the index.

Note the index of a key frame can change as frames are added.

Parameters

in	*_i*	the index

Returns

a pair that contains the time and transformation. **Time** (p. 791) is -1 if the index is out of bounds

10.97.3.7 double gazebo::common::NodeAnimation::GetLength ( ) const

Returns the duration of the animations.

Returns

the time of the last animation

10.97.3.8 std::string gazebo::common::NodeAnimation::GetName ( ) const

Returns the name.

Returns

the name

10.97.3.9 double gazebo::common::NodeAnimation::GetTimeAtX (const double *_x*) const

Returns the time where a transformation's translational value along the X axis is equal to _x.

When no transformation is found (within a tolerance of 1e-6), the time is interpolated.

Parameters

in	*_x*	the value along x. You must ensure that _x is within a valid range.

10.97.3.10 void gazebo::common::NodeAnimation::Scale (const double *_scale*)

Scales each transformation in the key frames.

This only affects the translational values.

Parameters

in	*_scale*	the scaling factor

10.97.3.11 void gazebo::common::NodeAnimation::SetName (const std::string & *_name*)

Changes the name of the animation.

Parameters

in	*the*	new name

10.97.4 Member Data Documentation

10.97.4.1 std::map<double, math::Matrix4> gazebo::common::NodeAnimation::keyFrames [protected]

the dictionary of key frames, indexed by time

10.97.4.2 double gazebo::common::NodeAnimation::length [protected]

the duration of the animations (time of last key frame)

10.97.4.3 std::string gazebo::common::NodeAnimation::name [protected]

the name of the animation

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

- **SkeletonAnimation.hh**

10.98 gazebo::common::NodeAssignment Struct Reference

Vertex to node weighted assignement for skeleton animation visualization.

`#include <Mesh.hh>`

Public Attributes

- unsigned int **nodeIndex**

  node (or bone) index
- unsigned int **vertexIndex**

  index of the vertex
- float **weight**

  the weight (between 0 and 1)

10.98.1 Detailed Description

Vertex to node weighted assignement for skeleton animation visualization.

10.98.2 Member Data Documentation

10.98.2.1 unsigned int gazebo::common::NodeAssignment::nodeIndex

node (or bone) index

10.98.2.2 unsigned int gazebo::common::NodeAssignment::vertexIndex

index of the vertex

10.98.2.3 float gazebo::common::NodeAssignment::weight

the weight (between 0 and 1)

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

- **Mesh.hh**

10.99 gazebo::common::NodeTransform Class Reference

NodeTransform (p. 547) **Skeleton.hh** (p. 1088) common/common.hh

```
#include <Skeleton.hh>
```

Public Types

- enum **TransformType** { **TRANSLATE**, **ROTATE**, **SCALE**, **MATRIX** }

  Enumeration of the transform types.

Public Member Functions

- **NodeTransform** (**TransformType** _type=**MATRIX**)

  Constructor.
- **NodeTransform** (**math::Matrix4** _mat, std::string _sid="_default_", TransformType _type=**MATRIX**)

  Constructor.
- ∼**NodeTransform** ()

  Destructor. It does nothing.
- **math::Matrix4 Get** ()

  Returns the transformation matrix.
- std::string **GetSID** ()

  Returns thr SID.
- **TransformType GetType** ()

  Returns the transformation type.
- **math::Matrix4 operator()** ()

  Matrix cast operator.
- **math::Matrix4 operator**∗ (**NodeTransform** _t)

  Node transform multiplication operator.
- **math::Matrix4 operator**∗ (**math::Matrix4** _m)

  Matrix multiplication operator.
- void **PrintSource** ()

  Prints the transform matrix to std::err stream.
- void **RecalculateMatrix** ()

  Sets the transform matrix from the source according to the type.
- void **Set** (**math::Matrix4** _mat)

  Assign a transformation.
- void **SetComponent** (unsigned int _idx, double _value)

  Set a transformation matrix component value.
- void **SetSID** (std::string _sid)

  Set the SID.
- void **SetSourceValues** (**math::Matrix4** _mat)

  Set source data values _ param[in] _mat the values.
- void **SetSourceValues** (**math::Vector3** _vec)

  Set source data values.
- void **SetSourceValues** (**math::Vector3** _axis, double _angle)

  Sets source matrix values from roation.
- void **SetType** (**TransformType** _type)

  Set transform type.

Protected Attributes

- std::string **sid**

  the sid
- std::vector< double > **source**

  source data values (can be a matrix, a position or rotation)
- **math::Matrix4 transform**

  transform
- **TransformType type**

  transform type

10.99.1 Detailed Description

NodeTransform (p. 547) **Skeleton.hh** (p. 1088) common/common.hh

A (p. 111) transformation node

10.99.2 Member Enumeration Documentation

10.99.2.1 enum gazebo::common::NodeTransform::TransformType

Enumeration of the transform types.

Enumerator:

TRANSLATE

ROTATE

SCALE

MATRIX

10.99.3 Constructor & Destructor Documentation

10.99.3.1 gazebo::common::NodeTransform::NodeTransform (TransformType *_type* = MATRIX)

Constructor.

Parameters

in	*_type*	the type of transform

10.99.3.2 gazebo::common::NodeTransform::NodeTransform (math::Matrix4 *_mat*, std::string *_sid* = "_default_", TransformType *_type* = MATRIX)

Constructor.

Parameters

in	*_mat*	the matrix
in	*_sid*	identifier
in	*_type*	the type of transform

10.99.3.3 gazebo::common::NodeTransform::~NodeTransform ( )

Destructor. It does nothing.

10.99.4 Member Function Documentation

10.99.4.1 math::Matrix4 gazebo::common::NodeTransform::Get ( )

Returns the transformation matrix.

Returns

the matrix

10.99.4.2 std::string gazebo::common::NodeTransform::GetSID ( )

Returns thr SID.

Returns

the SID

10.99.4.3 TransformType gazebo::common::NodeTransform::GetType ( )

Returns the transformation type.

Returns

the type

10.99.4.4 math::Matrix4 gazebo::common::NodeTransform::operator() ( )

Matrix cast operator.

Returns

the transform

10.99.4.5 math::Matrix4 gazebo::common::NodeTransform::operator* (NodeTransform _t)

Node transform multiplication operator.

Parameters

in	_t	a transform

Returns

transform matrix multiplied by _t's transform

10.99.4.6 math::Matrix4 gazebo::common::NodeTransform::operator∗ (math::Matrix4 _m)

Matrix multiplication operator.

Parameters

in	*_m*	a matrix

Returns

transform matrix multiplied by _m

10.99.4.7 void gazebo::common::NodeTransform::PrintSource ( )

Prints the transform matrix to std::err stream.

10.99.4.8 void gazebo::common::NodeTransform::RecalculateMatrix ( )

Sets the transform matrix from the source according to the type.

10.99.4.9 void gazebo::common::NodeTransform::Set (math::Matrix4 _mat)

Assign a transformation.

Parameters

in	*_mat*	the transform

10.99.4.10 void gazebo::common::NodeTransform::SetComponent (unsigned int _idx, double _value)

Set a transformation matrix component value.

Parameters

in	*_idx*	the component index
in	*_value*	the value

10.99.4.11 void gazebo::common::NodeTransform::SetSID (std::string _sid)

Set the SID.

Parameters

in	*_sid*	the sid

10.99.4.12 void gazebo::common::NodeTransform::SetSourceValues (math::Matrix4 _mat)

Set source data values _ param[in] _mat the values.

10.99.4.13 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 _vec)

Set source data values.

10.99.4.14 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 _axis, double _angle)

Sets source matrix values from roation.

Parameters

in	*_axis*	of rotation
in	*_angle*	of rotation

10.99.4.15 void gazebo::common::NodeTransform::SetType (TransformType _type)

Set transform type.

Parameters

in	*_type*	the type

10.99.5 Member Data Documentation

10.99.5.1 std::string gazebo::common::NodeTransform::sid [protected]

the sid

10.99.5.2 std::vector<double> gazebo::common::NodeTransform::source [protected]

source data values (can be a matrix, a position or rotation)

10.99.5.3 math::Matrix4 gazebo::common::NodeTransform::transform [protected]

transform

10.99.5.4 TransformType gazebo::common::NodeTransform::type [protected]

transform type

The documentation for this class was generated from the following file:

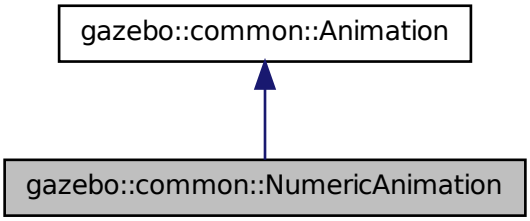
- **Skeleton.hh**

10.100 gazebo::common::NumericAnimation Class Reference

A (p. 111) numeric animation.

`#include <Animation.hh>`

Inheritance diagram for gazebo::common::NumericAnimation:



Public Member Functions

- **NumericAnimation** (const std::string &_name, double _length, bool _loop)

  Constructor.
- virtual ~**NumericAnimation** ()

  Destructor.
- **NumericKeyFrame** * **CreateKeyFrame** (double _time)

  Create a numeric keyframe at the given time.
- void **GetInterpolatedKeyFrame** (**NumericKeyFrame** &_kf) const

  Get a keyframe using the animation's current time.

Additional Inherited Members

10.100.1 Detailed Description

A (p. 111) numeric animation.

10.100.2 Constructor & Destructor Documentation

10.100.2.1 gazebo::common::NumericAnimation::NumericAnimation (const std::string & *_name,* double *_length,* bool *_loop*)

Constructor.

Parameters

in	*_name*	String name of the animation. This should be unique.
in	*_length*	Length of the animation in seconds
in	*_loop*	True == loop the animation

10.100.2.2 virtual gazebo::common::NumericAnimation::~NumericAnimation ( ) [virtual]

Destructor.

10.100.3 Member Function Documentation

10.100.3.1 NumericKeyFrame* gazebo::common::NumericAnimation::CreateKeyFrame (double _time)

Create a numeric keyframe at the given time.

Parameters

in	*_time*	**Time** (p. 791) at which to create the keyframe

Returns

Pointer to the new keyframe

10.100.3.2 void gazebo::common::NumericAnimation::GetInterpolatedKeyFrame (NumericKeyFrame & _kf) const

Get a keyframe using the animation's current time.

Parameters

out	*_kf*	**NumericKeyFrame** (p. 553) reference to hold the interpolated result

The documentation for this class was generated from the following file:

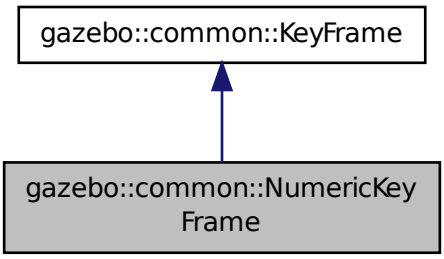
- **Animation.hh**

10.101 gazebo::common::NumericKeyFrame Class Reference

A (p. 111) keyframe for a **NumericAnimation** (p. 552).

`#include <KeyFrame.hh>`

Inheritance diagram for gazebo::common::NumericKeyFrame:



Public Member Functions

- **NumericKeyFrame** (double _time)

  Constructor.
- virtual ~**NumericKeyFrame** ()

  Destructor.
- const double & **GetValue** () const

  Get the value of the keyframe.
- void **SetValue** (const double &_value)

  Set the value of the keyframe.

Protected Attributes

- double **value**

  numeric value

10.101.1 Detailed Description

A (p. 111) keyframe for a **NumericAnimation** (p. 552).

10.101.2 Constructor & Destructor Documentation

10.101.2.1 gazebo::common::NumericKeyFrame::NumericKeyFrame (double *_time*)

Constructor.

Parameters

in	*_time*	**Time** (p. 791) of the keyframe

10.101.2.2 virtual gazebo::common::NumericKeyFrame::~NumericKeyFrame ( ) [virtual]

Destructor.

10.101.3 Member Function Documentation

10.101.3.1 const double& gazebo::common::NumericKeyFrame::GetValue ( ) const

Get the value of the keyframe.

Returns

the value of the keyframe

10.101.3.2 void gazebo::common::NumericKeyFrame::SetValue (const double & *_value*)

Set the value of the keyframe.

Parameters

in	*_value*	The new value

10.101.4 Member Data Documentation

10.101.4.1 double gazebo::common::NumericKeyFrame::value [protected]

numeric value

The documentation for this class was generated from the following file:

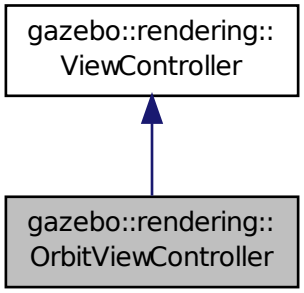
- **KeyFrame.hh**

10.102 gazebo::rendering::OrbitViewController Class Reference

Orbit view controller.

`#include <OrbitViewController.hh>`

Inheritance diagram for gazebo::rendering::OrbitViewController:



Public Member Functions

- **OrbitViewController** (**UserCameraPtr** _camera)

  Constructor.
- virtual ~**OrbitViewController** ()

  Destructor.
- **math::Vector3 GetFocalPoint** () const

  Get the focal point.
- virtual void **HandleKeyPressEvent** (const std::string &_key)

  Handle a key press event.
- void **HandleKeyReleaseEvent** (const std::string &_key)

  Handle a key release event.
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)

  Handle a mouse event.
- virtual void **Init** ()

  Initialize the controller.
- virtual void **Init** (const **math::Vector3** &_focalPoint)

  Initialze the controller with a focal point.
- void **SetDistance** (float _d)

  Set the distance to the focal point.
- void **SetFocalPoint** (const **math::Vector3** &_fp)

  Set the focal point.
- virtual void **Update** ()

  Update.

Static Public Member Functions

- static std::string **GetTypeString** ()

  Get the type name of this view controller.

Additional Inherited Members

10.102.1 Detailed Description

Orbit view controller.

10.102.2 Constructor & Destructor Documentation

10.102.2.1 gazebo::rendering::OrbitViewController::OrbitViewController (UserCameraPtr *_camera*)

Constructor.

Parameters

in	*_camera*	Pointer to the camera to control.

10.102.2.2 virtual gazebo::rendering::OrbitViewController::~OrbitViewController ( ) [virtual]

Destructor.

10.102.3 Member Function Documentation

10.102.3.1 math::Vector3 gazebo::rendering::OrbitViewController::GetFocalPoint ( ) const

Get the focal point.

Returns

The focal point

10.102.3.2 static std::string gazebo::rendering::OrbitViewController::GetTypeString ( ) [static]

Get the type name of this view controller.

Returns

The view controller name: "orbit".

10.102.3.3 virtual void gazebo::rendering::OrbitViewController::HandleKeyPressEvent (const std::string & *_key*) [virtual]

Handle a key press event.

Parameters

in	*_key*	The key that was pressed.

Implements **gazebo::rendering::ViewController** (p. 883).

10.102.3.4 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. 883).

10.102.3.5 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. 883).

10.102.3.6 virtual void gazebo::rendering::OrbitViewController::Init ( ) [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 884).

10.102.3.7 virtual void gazebo::rendering::OrbitViewController::Init (const math::Vector3 & _focalPoint) [virtual]

Initialze the controller with a focal point.

Parameters

in	*_focalPoint*	Point to look at.

Reimplemented from **gazebo::rendering::ViewController** (p. 884).

10.102.3.8 void gazebo::rendering::OrbitViewController::SetDistance (float _d)

Set the distance to the focal point.

Parameters

in	*_d*	The distance from the focal point.

10.102.3.9 void gazebo::rendering::OrbitViewController::SetFocalPoint (const math::Vector3 & _fp)

Set the focal point.

Parameters

in	*_fp*	The focal point

10.102.3.10 virtual void gazebo::rendering::OrbitViewController::Update ( ) [virtual]

Update.

Implements **gazebo::rendering::ViewController** (p. 884).

The documentation for this class was generated from the following file:

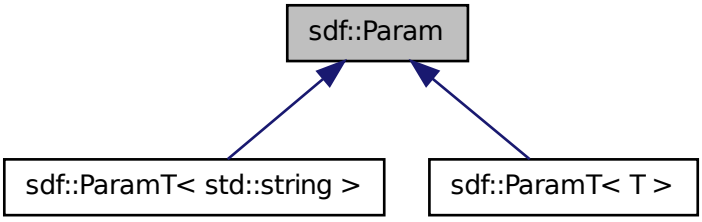
- **OrbitViewController.hh**

10.103 sdf::Param Class Reference

A (p. 111) 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 **GetTypeName** () const
- bool **IsBool** () const
- bool **IsChar** () const
- bool **IsColor** () const
- bool **IsDouble** () const
- bool **IsFloat** () const
- bool **IsInt** () const
- bool **IsPose** () const
- bool **IsQuaternion** () const
- bool **IsStr** () const
- bool **IsTime** () const
- bool **IsUInt** () const
- bool **IsVector2d** () const
- bool **IsVector2i** () const
- bool **IsVector3** () const
- virtual void **Reset** ()=0

  Reset the parameter.
- bool **Set** (const bool &_value)
- bool **Set** (const int &_value)
- bool **Set** (const unsigned int &_value)
- bool **Set** (const float &_value)
- bool **Set** (const double &_value)
- bool **Set** (const char &_value)
- bool **Set** (const std::string &_value)
- bool **Set** (const char ∗_value)
- bool **Set** (const **gazebo::math::Vector3** &_value)
- bool **Set** (const **gazebo::math::Vector2i** &_value)
- bool **Set** (const **gazebo::math::Vector2d** &_value)
- bool **Set** (const **gazebo::math::Quaternion** &_value)
- bool **Set** (const **gazebo::math::Pose** &_value)
- bool **Set** (const **gazebo::common::Color** &_value)
- bool **Set** (const **gazebo::common::Time** &_value)
- void **SetDescription** (const std::string &_desc)

  Set the description of the parameter.

- virtual bool **SetFromString** (const std::string &)

  Set the parameter value from a string.
- template<typename T >
  void **SetUpdateFunc** (T _updateFunc)

  Update function.
- virtual void **Update** ()=0

Protected Attributes

- std::string **description**
- std::string **key**
- bool **required**
- bool **set**
- std::string **typeName**
- boost::function< boost::any()> **updateFunc**

10.103.1 Detailed Description

A (p. 111) parameter class.

10.103.2 Constructor & Destructor Documentation

10.103.2.1 sdf::Param::Param (Param * *_newParam*)

Constructor.

10.103.2.2 virtual sdf::Param::~Param ( ) [virtual]

Destructor.

10.103.3 Member Function Documentation

10.103.3.1 virtual Param* sdf::Param::Clone ( ) const [pure virtual]

Implemented in **sdf::ParamT**< **T** > (p. 567), and **sdf::ParamT**< **std::string** > (p. 567).

10.103.3.2 bool sdf::Param::Get (bool & *_value*)

10.103.3.3 bool sdf::Param::Get (int & *_value*)

10.103.3.4 bool sdf::Param::Get (unsigned int & *_value*)

10.103.3.5 bool sdf::Param::Get (float & *_value*)

10.103.3.6 bool sdf::Param::Get (double & *_value*)

10.103.3.7 bool sdf::Param::Get (char & *_value*)

10.103.3.8 bool sdf::Param::Get (std::string & *_value*)

10.103.3.9 bool sdf::Param::Get (gazebo::math::Vector3 & *_value*)

10.103.3.10 bool sdf::Param::Get (gazebo::math::Vector2i & *_value*)

10.103.3.11 bool sdf::Param::Get (gazebo::math::Vector2d & *_value*)

10.103.3.12 bool sdf::Param::Get (gazebo::math::Quaternion & *_value*)

10.103.3.13 bool sdf::Param::Get (gazebo::math::Pose & *_value*)

10.103.3.14 bool sdf::Param::Get (gazebo::common::Color & *_value*)

10.103.3.15 bool sdf::Param::Get (gazebo::common::Time & *_value*)

10.103.3.16 virtual std::string sdf::Param::GetAsString ( ) const [inline], [virtual]

Get the type.

Reimplemented in **sdf::ParamT< T >** (p. 567), and **sdf::ParamT< std::string >** (p. 567).

10.103.3.17 virtual std::string sdf::Param::GetDefaultAsString ( ) const [inline], [virtual]

Reimplemented in **sdf::ParamT< T >** (p. 567), and **sdf::ParamT< std::string >** (p. 567).

10.103.3.18 std::string sdf::Param::GetDescription ( ) const

Get the description of the parameter.

10.103.3.19 const std::string& sdf::Param::GetKey ( ) const [inline]

References key.

Referenced by sdf::ParamT< std::string >::Clone(), and sdf::ParamT< std::string >::Set().

10.103.3.20 bool sdf::Param::GetRequired ( ) const [inline]

References required.

10.103.3.21 bool sdf::Param::GetSet ( ) const [inline]

Return true if the parameter has been set.

10.103.3.22 std::string sdf::Param::GetTypeName ( ) const

10.103.3.23 bool sdf::Param::IsBool ( ) const

10.103.3.24 bool sdf::Param::IsChar ( ) const

10.103.3.25 bool sdf::Param::IsColor ( ) const

10.103.3.26 bool sdf::Param::IsDouble ( ) const

10.103.3.27 bool sdf::Param::IsFloat ( ) const

10.103.3.28 bool sdf::Param::IsInt ( ) const

10.103.3.29 bool sdf::Param::IsPose ( ) const

10.103.3.30 bool sdf::Param::IsQuaternion ( ) const

10.103.3.31 bool sdf::Param::IsStr ( ) const

10.103.3.32 bool sdf::Param::IsTime ( ) const

10.103.3.33 bool sdf::Param::IsUInt ( ) const

10.103.3.34 bool sdf::Param::IsVector2d ( ) const

10.103.3.35 bool sdf::Param::IsVector2i ( ) const

10.103.3.36 bool sdf::Param::IsVector3 ( ) const

10.103.3.37 virtual void sdf::Param::Reset ( ) [pure virtual]

Reset the parameter.

Implemented in **sdf::ParamT< T >** (p. 567), and **sdf::ParamT< std::string >** (p. 567).

10.103.3.38 bool sdf::Param::Set (const bool & *_value*)

Referenced by sdf::ParamT< std::string >::Update().

10.103.3.39 bool sdf::Param::Set (const int & *_value*)

10.103.3.40 bool sdf::Param::Set (const unsigned int & *_value*)

10.103.3.41 bool sdf::Param::Set (const float & *_value*)

10.103.3.42 bool sdf::Param::Set (const double & *_value*)

10.103.3.43 bool sdf::Param::Set (const char & *_value*)

10.103.3.44 bool sdf::Param::Set (const std::string & *_value*)

10.103.3.45 bool sdf::Param::Set (const char * *_value*)

10.103.3.46 bool sdf::Param::Set (const gazebo::math::Vector3 & *_value*)

10.103.3.47 bool sdf::Param::Set (const gazebo::math::Vector2i & *_value*)

10.103.3.48 bool sdf::Param::Set (const gazebo::math::Vector2d & *_value*)

10.103.3.49 bool sdf::Param::Set (const gazebo::math::Quaternion & *_value*)

10.103.3.50 bool sdf::Param::Set (const gazebo::math::Pose & *_value*)

10.103.3.51 bool sdf::Param::Set (const gazebo::common::Color & *_value*)

10.103.3.52 bool sdf::Param::Set (const gazebo::common::Time & *_value*)

10.103.3.53 void sdf::Param::SetDescription (const std::string & *_desc*)

Set the description of the parameter.

10.103.3.54 virtual bool sdf::Param::SetFromString (const std::string &) [inline], [virtual]

Set the parameter value from a string.

Reimplemented in **sdf::ParamT< T >** (p. 567), and **sdf::ParamT< std::string >** (p. 567).

10.103.3.55 template<typename T > void sdf::Param::SetUpdateFunc (T *_updateFunc*) [inline]

Update function.

References updateFunc.

10.103.3.56 virtual void sdf::Param::Update ( ) [pure virtual]

Implemented in **sdf::ParamT< T >** (p. 568), and **sdf::ParamT< std::string >** (p. 568).

10.103.4 Member Data Documentation

10.103.4.1 std::string sdf::Param::description [protected]

Referenced by sdf::ParamT< std::string >::Clone(), and sdf::ParamT< std::string >::ParamT().

10.103.4.2 std::string sdf::Param::key [protected]

Referenced by GetKey(), sdf::ParamT< std::string >::ParamT(), and sdf::ParamT< std::string >::Set().

10.103.4.3 bool sdf::Param::required [protected]

Referenced by sdf::ParamT< std::string >::Clone(), GetRequired(), sdf::ParamT< std::string >::ParamT(), and sdf::-ParamT< std::string >::Set().

10.103.4.4 bool sdf::Param::set [protected]

10.103.4.5 std::string sdf::Param::typeName [protected]

Referenced by sdf::ParamT< std::string >::Clone(), and sdf::ParamT< std::string >::ParamT().

10.103.4.6 boost::function<boost::any ()> sdf::Param::updateFunc [protected]

Referenced by SetUpdateFunc(), and sdf::ParamT< std::string >::Update().

The documentation for this class was generated from the following file:

- **Param.hh**

10.104 ParamT< T > Class Template Reference

`#include <CommonTypes.hh>`

The documentation for this class was generated from the following file:

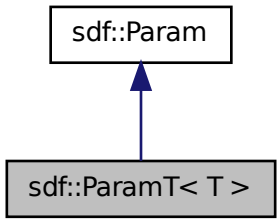
- **CommonTypes.hh**

10.105 sdf::ParamT< T > Class Template Reference

Templatized parameter class.

`#include <Param.hh>`

Inheritance diagram for sdf::ParamT< T >:



Public Member Functions

- **ParamT** (const std::string &_key, const std::string &_default, bool _required, const std::string &_typeName="", const std::string &_description="")

  Constructor.
- virtual ~**ParamT** ()

  Destructor.

- virtual **Param** ∗ **Clone** () const
- virtual std::string **GetAsString** () const

  Get the parameter value as a string.
- virtual std::string **GetDefaultAsString** () const
- T **GetDefaultValue** () const

  Get the value.
- T **GetValue** () const

  Get the value.
- T **operator**∗ () const
- virtual void **Reset** ()

  Reset to default value.
- virtual bool **Set** (const std::string &_str)

  Set the parameter value from a string.
- virtual bool **SetFromString** (const std::string &_value)

  Set the parameter value from a string.
- void **SetValue** (const T &_value)

  Set the value of the parameter.
- virtual void **Update** ()

  Update param value.

Protected Attributes

- T **defaultValue**
- T **value**

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **ParamT**< T > &_p)

10.105.1 Detailed Description

template<**typename T**>**class sdf::ParamT**< T >

Templatized parameter class.

10.105.2 Constructor & Destructor Documentation

10.105.2.1 template<typename T> sdf::ParamT< T >::ParamT (const std::string & *_key,* const std::string & *_default,* bool *_required,* const std::string & *_typeName = "",* const std::string & *_description = "")* [inline]

Constructor.

10.105.2.2 template<typename T> virtual sdf::ParamT< T >::~ParamT ( ) [inline], [virtual]

Destructor.

10.105.3 Member Function Documentation

10.105.3.1 template<typename T> virtual Param* sdf::ParamT< T >::Clone ( ) const [inline], [virtual]

Implements **sdf::Param** (p. 561).

10.105.3.2 template<typename T> virtual std::string sdf::ParamT< T >::GetAsString ( ) const [inline], [virtual]

Get the parameter value as a string.

Reimplemented from **sdf::Param** (p. 562).

Referenced by sdf::ParamT< std::string >::Clone().

10.105.3.3 template<typename T> virtual std::string sdf::ParamT< T >::GetDefaultAsString ( ) const [inline], [virtual]

Reimplemented from **sdf::Param** (p. 562).

10.105.3.4 template<typename T> T sdf::ParamT< T >::GetDefaultValue ( ) const [inline]

Get the value.

10.105.3.5 template<typename T> T sdf::ParamT< T >::GetValue ( ) const [inline]

Get the value.

10.105.3.6 template<typename T> T sdf::ParamT< T >::operator* ( ) const [inline]

10.105.3.7 template<typename T> virtual void sdf::ParamT< T >::Reset ( ) [inline], [virtual]

Reset to default value.

Implements **sdf::Param** (p. 563).

10.105.3.8 template<typename T> virtual bool sdf::ParamT< T >::Set (const std::string & *_str*) [inline], [virtual]

Set the parameter value from a string.

Referenced by sdf::ParamT< std::string >::ParamT(), and sdf::ParamT< std::string >::SetFromString().

10.105.3.9 template<typename T> virtual bool sdf::ParamT< T >::SetFromString (const std::string & *_value*) [inline], [virtual]

Set the parameter value from a string.

Reimplemented from **sdf::Param** (p. 564).

10.105.3.10 template<typename T> void sdf::ParamT< T >::SetValue (const T & _value) [inline]

Set the value of the parameter.

10.105.3.11 template<typename T> virtual void sdf::ParamT< T >::Update ( ) [inline], [virtual]

Update param value.

Implements **sdf::Param** (p. 564).

10.105.4 Friends And Related Function Documentation

10.105.4.1 template<typename T> std::ostream& operator<< (std::ostream & _out, const ParamT< T > & _p) [friend]

10.105.5 Member Data Documentation

10.105.5.1 template<typename T> T sdf::ParamT< T >::defaultValue [protected]

Referenced by sdf::ParamT< std::string >::GetDefaultAsString(), sdf::ParamT< std::string >::GetDefaultValue(), sdf::ParamT< std::string >::ParamT(), sdf::ParamT< std::string >::Reset(), and sdf::ParamT< std::string >::Set().

10.105.5.2 template<typename T> T sdf::ParamT< T >::value [protected]

Referenced by sdf::ParamT< std::string >::GetAsString(), sdf::ParamT< std::string >::GetValue(), sdf::ParamT< std::string >::operator*(), sdf::ParamT< std::string >::ParamT(), sdf::ParamT< std::string >::Reset(), sdf::ParamT< std::string >::Set(), and sdf::ParamT< std::string >::SetValue().

The documentation for this class was generated from the following file:

- **Param.hh**

10.106 gazebo::physics::PhysicsEngine Class Reference

Base (p. 137) class for a physics engine.

`#include <physics/physics.hh>`

Public Member Functions

- **PhysicsEngine** (**WorldPtr** _world)

  Default constructor.
- virtual ~**PhysicsEngine** ()

  Destructor.
- virtual **CollisionPtr CreateCollision** (const std::string &_shapeType, **LinkPtr** _link)=0

  Create a collision.
- **CollisionPtr CreateCollision** (const std::string &_shapeType, const std::string &_linkName)

  Create a collision.
- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent)=0

  Create a new joint.

- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)=0

  Create a new body.
- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)=0

  *Create a **physics::Shape** (p. 720) object.*
- virtual void **DebugPrint** () const =0

  Debug print out of the physic engine state.
- virtual void **Fini** ()

  Finilize the physics engine.
- virtual bool **GetAutoDisableFlag** ()

  : Remove this function, and replace it with a more generic property map
- **ContactManager** * **GetContactManager** () const

  Get a pointer to the contact manger.
- virtual double **GetContactMaxCorrectingVel** ()

  : Remove this function, and replace it with a more generic property map.
- virtual double **GetContactSurfaceLayer** ()

  : Remove this function, and replace it with a more generic property map.
- virtual **math::Vector3 GetGravity** () const

  Return the gavity vector.
- virtual int **GetMaxContacts** ()

  : Remove this function, and replace it with a more generic property map.
- double **GetMaxStepSize** () const

  Get max step size.
- virtual boost::any **GetParam** (std::string _key) const

  Get an parameter of the physics engine.
- boost::recursive_mutex * **GetPhysicsUpdateMutex** () const

  *returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 581).*
- double **GetRealTimeUpdateRate** () const

  Get real time update rate.
- virtual int **GetSORPGSIters** ()

  : Remove this function, and replace it with a more generic property map
- virtual int **GetSORPGSPreconIters** ()

  : Remove this function, and replace it with a more generic property map
- virtual double **GetSORPGSW** ()

  : Remove this function, and replace it with a more generic property map.
- virtual double **GetStepTime** () **GAZEBO_DEPRECATED**(1.5)

  Get the simulation step time.
- double **GetTargetRealTimeFactor** () const

  Get target real time factor.
- virtual std::string **GetType** () const =0

  Return the type of the physics engine (ode|bullet).
- double **GetUpdatePeriod** ()

  Get the simulation update period.
- double **GetUpdateRate** () **GAZEBO_DEPRECATED**(1.5)

  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
- void **SetMaxStepSize** (double _stepSize)

  Set max step size.
- virtual void **SetParam** (std::string _key, const boost::any &_value)

  Set a parameter of the physics engine.
- void **SetRealTimeUpdateRate** (double _rate)

  Set real time update rate.
- virtual void **SetSeed** (uint32_t _seed)=0

  Set the random number seed for the physics engine.
- virtual void **SetSORPGSIters** (unsigned int _iters)

  : Remove this function, and replace it with a more generic property map
- virtual void **SetSORPGSPreconIters** (unsigned int _iters)

  : Remove this function, and replace it with a more generic property map
- virtual void **SetSORPGSW** (double _w)

  : Remove this function, and replace it with a more generic property map
- virtual void **SetStepTime** (double _value) **GAZEBO_DEPRECATED**(1.5)

  Set the simulation step time.
- void **SetTargetRealTimeFactor** (double _factor)

  Set target real time factor.
- void **SetUpdateRate** (double _value) **GAZEBO_DEPRECATED**(1.5)

  Set the simulation update rate.
- virtual void **SetWorldCFM** (double _cfm)

  : Remove this function, and replace it with a more generic property map
- virtual void **SetWorldERP** (double _erp)

  : Remove this function, and replace it with a more generic property map
- virtual void **UpdateCollision** ()=0

  Update the physics engine collision.
- virtual void **UpdatePhysics** ()

  Update the physics engine.

Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &_msg)

  virtual callback for gztopic "~/physics".
- virtual void **OnRequest** (ConstRequestPtr &_msg)

  virtual callback for gztopic "~/request".

Protected Attributes

- **ContactManager** * **contactManager**

  Class that handles all contacts generated by the physics engine.
- double **maxStepSize**

  Real time update rate.
- **transport::NodePtr node**

  Node for communication.
- **transport::SubscriberPtr physicsSub**

  Subscribe to the physics topic.
- boost::recursive_mutex * **physicsUpdateMutex**

  Mutex to protect the update cycle.
- double **realTimeUpdateRate**

  Real time update rate.
- **transport::SubscriberPtr requestSub**

  Subscribe to the request topic.
- **transport::PublisherPtr responsePub**

  Response publisher.
- **sdf::ElementPtr sdf**

  Our SDF values.
- double **targetRealTimeFactor**

  Target real time factor.
- **WorldPtr world**

  Pointer to the world.

10.106.1 Detailed Description

Base (p. 137) class for a physics engine.

10.106.2 Constructor & Destructor Documentation

10.106.2.1 gazebo::physics::PhysicsEngine::PhysicsEngine (WorldPtr *_world*) [explicit]

Default constructor.

Parameters

in	*_world*	Pointer to the world.

10.106.2.2 virtual gazebo::physics::PhysicsEngine::~PhysicsEngine ( ) [virtual]

Destructor.

10.106.3 Member Function Documentation

10.106.3.1 virtual CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & *_shapeType*, LinkPtr *_link*) [pure virtual]

Create a collision.

Parameters

in	*_shapeType*	Type of collision to create.
in	*_link*	Parent link.

10.106.3.2 CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & *_shapeType*, const std::string & *_linkName*)

Create a collision.

Parameters

in	*_shapeType*	Type of collision to create.
in	*_linkName*	Name of the parent link.

10.106.3.3 virtual JointPtr gazebo::physics::PhysicsEngine::CreateJoint (const std::string & *_type*, ModelPtr *_parent*) [pure virtual]

Create a new joint.

Parameters

in	*_type*	Type of joint to create.
in	*_parent*	**Model** (p. 489) parent.

10.106.3.4 virtual LinkPtr gazebo::physics::PhysicsEngine::CreateLink (ModelPtr *_parent*) [pure virtual]

Create a new body.

Parameters

in	*_parent*	Parent model for the link.

10.106.3.5 virtual ShapePtr gazebo::physics::PhysicsEngine::CreateShape (const std::string & *_shapeType*, CollisionPtr *_collision*) [pure virtual]

Create a **physics::Shape** (p. 720) object.

Parameters

in	*_shapeType*	Type of shape to create.
in	*_collision*	**Collision** (p. 195) parent.

10.106.3.6 virtual void gazebo::physics::PhysicsEngine::DebugPrint ( ) const [pure virtual]

Debug print out of the physic engine state.

10.106.3.7 virtual void gazebo::physics::PhysicsEngine::Fini ( ) [virtual]

Finilize the physics engine.

10.106.3.8 virtual bool gazebo::physics::PhysicsEngine::GetAutoDisableFlag ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

access functions to set ODE parameters..

Returns

Auto disable flag.

10.106.3.9 ContactManager* gazebo::physics::PhysicsEngine::GetContactManager ( ) const

Get a pointer to the contact manger.

Returns

Pointer to the contact manager.

10.106.3.10 virtual double gazebo::physics::PhysicsEngine::GetContactMaxCorrectingVel ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters.

Returns

Max correcting velocity.

10.106.3.11 virtual double gazebo::physics::PhysicsEngine::GetContactSurfaceLayer ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters.

Returns

Contact (p. 236) suerface layer depth.

10.106.3.12 virtual math::Vector3 gazebo::physics::PhysicsEngine::GetGravity ( ) const [virtual]

Return the gavity vector.

Returns

The gavity vector.

10.106.3.13 virtual int gazebo::physics::PhysicsEngine::GetMaxContacts ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters.

Returns

Maximum number of allows contacts.

10.106.3.14 double gazebo::physics::PhysicsEngine::GetMaxStepSize ( ) const

Get max step size.

Returns

Max step size.

10.106.3.15 virtual boost::any gazebo::physics::PhysicsEngine::GetParam (std::string *_key*) const [virtual]

Get an parameter of the physics engine.

Parameters

in	*_attr*	String key

Returns

The value of the parameter

10.106.3.16 boost::recursive_mutex* gazebo::physics::PhysicsEngine::GetPhysicsUpdateMutex ( ) const [inline]

returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 581).

Returns

Pointer to the physics mutex.

References physicsUpdateMutex.

10.106.3.17 double gazebo::physics::PhysicsEngine::GetRealTimeUpdateRate ( ) const

Get real time update rate.

Returns

Update rate

10.106.3.18 virtual int gazebo::physics::PhysicsEngine::GetSORPGSIters ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

access functions to set ODE parameters.

Returns

SORPGS iterations.

10.106.3.19 virtual int gazebo::physics::PhysicsEngine::GetSORPGSPreconIters ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

access functions to set ODE parameters.

Returns

SORPGS precondition iterations.

10.106.3.20 virtual double gazebo::physics::PhysicsEngine::GetSORPGSW ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters

Returns

SORPGSW value.

10.106.3.21 virtual double gazebo::physics::PhysicsEngine::GetStepTime ( ) [virtual]

Get the simulation step time.

This funciton is deprecated, use World::GetMaxStepSize.

Returns

Simulation step time.

10.106.3.22 double gazebo::physics::PhysicsEngine::GetTargetRealTimeFactor ( ) const

Get target real time factor.

Returns

Target real time factor

10.106.3.23 virtual std::string gazebo::physics::PhysicsEngine::GetType ( ) const [pure virtual]

Return the type of the physics engine (ode|bullet).

Returns

Type of the physics engine.

10.106.3.24 double gazebo::physics::PhysicsEngine::GetUpdatePeriod ( )

Get the simulation update period.

Returns

Simulation update period.

10.106.3.25 double gazebo::physics::PhysicsEngine::GetUpdateRate ( )

Get the simulation update rate.

This funciton is deprecated, use **PhysicsEngine::GetRealTimeUpdateRate** (p. 574).

Returns

Update rate.

10.106.3.26 virtual double gazebo::physics::PhysicsEngine::GetWorldCFM ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 910) CFM.

Returns

World (p. 910) CFM.

10.106.3.27 virtual double gazebo::physics::PhysicsEngine::GetWorldERP ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 910) ERP.

Returns

World (p. 910) ERP.

10.106.3.28 virtual void gazebo::physics::PhysicsEngine::Init ( ) [pure virtual]

Initialize the physics engine.

10.106.3.29 virtual void gazebo::physics::PhysicsEngine::InitForThread ( ) [pure virtual]

Init the engine for threads.

10.106.3.30 virtual void gazebo::physics::PhysicsEngine::Load (sdf::ElementPtr _sdf) [virtual]

Load the physics engine.

Parameters

in	*_sdf*	Pointer to the SDF parameters.

10.106.3.31 virtual void gazebo::physics::PhysicsEngine::OnPhysicsMsg (ConstPhysicsPtr & _msg) [protected], [virtual]

virtual callback for gztopic "~/physics".

Parameters

in	*_msg*	Physics message.

10.106.3.32 virtual void gazebo::physics::PhysicsEngine::OnRequest (ConstRequestPtr & _msg) [protected], [virtual]

virtual callback for gztopic "~/request".

Parameters

in	*_msg*	Request message.

10.106.3.33 virtual void gazebo::physics::PhysicsEngine::Reset ( ) [inline], [virtual]

Rest the physics engine.

10.106.3.34 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	*_autoDisable*	True to enable auto disabling of bodies.

10.106.3.35 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	*_vel*	Max correcting velocity.

10.106.3.36 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	*_layerDepth*	Surface layer depth

10.106.3.37 virtual void gazebo::physics::PhysicsEngine::SetGravity (const gazebo::math::Vector3 & *_gravity*) [pure virtual]

Set the gavity vector.

Parameters

in	*_gravity*	New gravity vector.

10.106.3.38 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.

10.106.3.39 void gazebo::physics::PhysicsEngine::SetMaxStepSize (double *_stepSize*)

Set max step size.

Parameters

in	*_stepSize*	Max step size.

10.106.3.40 virtual void gazebo::physics::PhysicsEngine::SetParam (std::string *_key,* const boost::any & *_value*) [virtual]

Set a parameter of the physics engine.

Parameters

in	*_key*	String key
in	*_value*	The value to set to

10.106.3.41 void gazebo::physics::PhysicsEngine::SetRealTimeUpdateRate (double _rate)

Set real time update rate.

Parameters

in	*_rate*	Update rate

10.106.3.42 virtual void gazebo::physics::PhysicsEngine::SetSeed (uint32_t _seed) [pure virtual]

Set the random number seed for the physics engine.

Parameters

in	*_seed*	The random number seed.

10.106.3.43 virtual void gazebo::physics::PhysicsEngine::SetSORPGSIters (unsigned int _iters) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_iter*	Number of iterations.

10.106.3.44 virtual void gazebo::physics::PhysicsEngine::SetSORPGSPreconIters (unsigned int _iters) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_iter*	Number of iterations.

10.106.3.45 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.

10.106.3.46 virtual void gazebo::physics::PhysicsEngine::SetStepTime (double _value) [virtual]

Set the simulation step time.

This funciton is deprecated, use World::SetMaxStepSize.

Parameters

in	*_value*	Value of the step time.

10.106.3.47 void gazebo::physics::PhysicsEngine::SetTargetRealTimeFactor (double *_factor*)

Set target real time factor.

Parameters

in	*_factor*	Target real time factor

10.106.3.48 void gazebo::physics::PhysicsEngine::SetUpdateRate (double *_value*)

Set the simulation update rate.

This funciton is deprecated, use **PhysicsEngine::SetRealTimeUpdateRate** (p. 579).

Parameters

in	*_value*	Value of the update rate.

10.106.3.49 virtual void gazebo::physics::PhysicsEngine::SetWorldCFM (double *_cfm*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_cfm*	Constraint force mixing.

10.106.3.50 virtual void gazebo::physics::PhysicsEngine::SetWorldERP (double *_erp*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_erp*	Error reduction parameter.

10.106.3.51 virtual void gazebo::physics::PhysicsEngine::UpdateCollision ( ) [pure virtual]

Update the physics engine collision.

10.106.3.52 virtual void gazebo::physics::PhysicsEngine::UpdatePhysics ( ) [inline], [virtual]

Update the physics engine.

10.106.4 Member Data Documentation

10.106.4.1 ContactManager* gazebo::physics::PhysicsEngine::contactManager [protected]

Class that handles all contacts generated by the physics engine.

10.106.4.2 double gazebo::physics::PhysicsEngine::maxStepSize [protected]

Real time update rate.

10.106.4.3 transport::NodePtr gazebo::physics::PhysicsEngine::node [protected]

Node for communication.

10.106.4.4 transport::SubscriberPtr gazebo::physics::PhysicsEngine::physicsSub [protected]

Subscribe to the physics topic.

10.106.4.5 boost::recursive_mutex* gazebo::physics::PhysicsEngine::physicsUpdateMutex [protected]

Mutex to protect the update cycle.

Referenced by GetPhysicsUpdateMutex().

10.106.4.6 double gazebo::physics::PhysicsEngine::realTimeUpdateRate [protected]

Real time update rate.

10.106.4.7 transport::SubscriberPtr gazebo::physics::PhysicsEngine::requestSub [protected]

Subscribe to the request topic.

10.106.4.8 transport::PublisherPtr gazebo::physics::PhysicsEngine::responsePub [protected]

Response publisher.

10.106.4.9 sdf::ElementPtr gazebo::physics::PhysicsEngine::sdf [protected]

Our SDF values.

10.106.4.10 double gazebo::physics::PhysicsEngine::targetRealTimeFactor [protected]

Target real time factor.

10.106.4.11 WorldPtr gazebo::physics::PhysicsEngine::world [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **PhysicsEngine.hh**

10.107 gazebo::physics::PhysicsFactory Class Reference

The physics factory instantiates different physics engines.

`#include <physics/physics.hh>`

Static Public Member Functions

- static bool **IsRegistered** (const std::string &_name)

  Check if a physics engine is registered.
- static **PhysicsEnginePtr NewPhysicsEngine** (const std::string &_className, **WorldPtr** _world)

  Create a new instance of a physics engine.
- static void **RegisterAll** ()

  Register everything.
- static void **RegisterPhysicsEngine** (std::string _className, **PhysicsFactoryFn** _factoryfn)

  Register a physics class.

10.107.1 Detailed Description

The physics factory instantiates different physics engines.

10.107.2 Member Function Documentation

10.107.2.1 static bool gazebo::physics::PhysicsFactory::IsRegistered (const std::string & *_name*) [static]

Check if a physics engine is registered.

Parameters

in	*_name*	Name of the physics engine.

Returns

True if physics engine is registered, false otherwise.

10.107.2.2 static PhysicsEnginePtr gazebo::physics::PhysicsFactory::NewPhysicsEngine (const std::string & *_className*, WorldPtr *_world*) [static]

Create a new instance of a physics engine.

Parameters

in	*_className*	Name of the physics class.
in	*_world*	**World** (p. 910) to pass to the created physics engine.

10.107.2.3 static void gazebo::physics::PhysicsFactory::RegisterAll ( ) [static]

Register everything.

10.107.2.4 static void gazebo::physics::PhysicsFactory::RegisterPhysicsEngine (std::string *_className,* **PhysicsFactoryFn** *_factoryfn*) [static]

Register a physics class.

Parameters

in	*_className*	Name of the physics class.
in	*_factoryfn*	Function pointer used to create a physics engine.

The documentation for this class was generated from the following file:

- **PhysicsFactory.hh**

10.108 gazebo::common::PID Class Reference

Generic **PID** (p. 583) controller class.

`#include <common/common.hh>`

Public Member Functions

- **PID** (double _p=0.0, double _i=0.0, double _d=0.0, double _imax=0.0, double _imin=0.0, double _cmdMax=0.0, double _cmdMin=0.0)

  Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[I1:I2].
- virtual ~**PID** ()

  Destructor.
- double **GetCmd** ()

  *Return current command for this **PID** (p. 583) controller.*
- void **GetErrors** (double &_pe, double &_ie, double &_de)

  *Return **PID** (p. 583) error terms for the controller.*
- void **Init** (double _p=0.0, double _i=0.0, double _d=0.0, double _imax=0.0, double _imin=0.0, double _cmdMax=0.0, double _cmdMin=0.0)

  Initialize PID-gains and integral term limits:[iMax:iMin]-[I1:I2].
- **PID** & **operator=** (const **PID** &_p)

  Assignment operator.
- void **Reset** ()

  Reset the errors and command.
- void **SetCmd** (double _cmd)

*Set current target command for this **PID** (p. 583) controller.*

- void **SetCmdMax** (double _c)

  Set the maximum value for the command.
- void **SetCmdMin** (double _c)

  Set the maximum value for the command.
- void **SetDGain** (double _d)

  Set the derivtive Gain.
- void **SetIGain** (double _i)

  Set the integral Gain.
- void **SetIMax** (double _i)

  Set the integral upper limit.
- void **SetIMin** (double _i)

  Set the integral lower limit.
- void **SetPGain** (double _p)

  Set the proportional Gain.
- double **Update** (double _error, **common::Time** _dt)

  Update the Pid loop with nonuniform time step size.

10.108.1 Detailed Description

Generic **PID** (p. 583) controller class.

Generic proportiolnal-integral-derivative controller class that keeps track of PID-error states and control inputs given the state of a system and a user specified target state.

10.108.2 Constructor & Destructor Documentation

10.108.2.1 gazebo::common::PID::PID (double _p = 0.0, double _i = 0.0, double _d = 0.0, double _imax = 0.0, double _imin = 0.0, double _cmdMax = 0.0, double _cmdMin = 0.0)

Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[I1:I2].

Parameters

in	*_p*	The proportional gain.
in	*_i*	The integral gain.
in	*_d*	The derivative gain.
in	*_imax*	The integral upper limit.
in	*_imin*	The integral lower limit.

10.108.2.2 virtual gazebo::common::PID::~PID ( ) [virtual]

Destructor.

10.108.3 Member Function Documentation

10.108.3.1 double gazebo::common::PID::GetCmd ( )

Return current command for this **PID** (p. 583) controller.

Returns

the command value

10.108.3.2 void gazebo::common::PID::GetErrors (double & *_pe*, double & *_ie*, double & *_de*)

Return **PID** (p. 583) error terms for the controller.

Parameters

in	*_pe*	The proportional error.
in	*_ie*	The integral error.
in	*_de*	The derivative error.

10.108.3.3 void gazebo::common::PID::Init (double *_p* = 0.0, double *_i* = 0.0, double *_d* = 0.0, double *_imax* = 0.0, double *_imin* = 0.0, double *_cmdMax* = 0.0, double *_cmdMin* = 0.0)

Initialize PID-gains and integral term limits:[iMax:iMin]-[I1:I2].

Parameters

in	*_p*	The proportional gain.
in	*_i*	The integral gain.
in	*_d*	The derivative gain.
in	*_imax*	The integral upper limit.
in	*_imin*	The integral lower limit.

10.108.3.4 PID& gazebo::common::PID::operator= (const PID & *_p*) [inline]

Assignment operator.

Parameters

in	*_p*	a reference to a **PID** (p. 583) to assign values from

Returns

reference to this instance

References Reset().

10.108.3.5 void gazebo::common::PID::Reset ( )

Reset the errors and command.

Referenced by operator=().

10.108.3.6 void gazebo::common::PID::SetCmd (double _cmd)

Set current target command for this **PID** (p. 583) controller.

Parameters

in	*_cmd*	New command

10.108.3.7 void gazebo::common::PID::SetCmdMax (double _c)

Set the maximum value for the command.

Parameters

in	*_c*	The maximum value

10.108.3.8 void gazebo::common::PID::SetCmdMin (double _c)

Set the maximum value for the command.

Parameters

in	*_c*	The maximum value

10.108.3.9 void gazebo::common::PID::SetDGain (double _d)

Set the derivtive Gain.

Parameters

in	*_p*	dertivative gain value

10.108.3.10 void gazebo::common::PID::SetIGain (double _i)

Set the integral Gain.

Parameters

in	*_p*	integral gain value

10.108.3.11 void gazebo::common::PID::SetIMax (double _i)

Set the integral upper limit.

Parameters

in	*_p*	integral upper limit value

10.108.3.12 void gazebo::common::PID::SetIMin (double _i)

Set the integral lower limit.

Parameters

in	*_p*	integral lower limit value

10.108.3.13 void gazebo::common::PID::SetPGain (double _p)

Set the proportional Gain.

Parameters

in	*_p*	proportional gain value

10.108.3.14 double gazebo::common::PID::Update (double _error, common::Time _dt)

Update the Pid loop with nonuniform time step size.

Parameters

_in]	_error Error since last call (p_state - p_target).
_in]	_dt Change in time since last update call. Normally, this is called at every time step, The return value is an updated command to be passed to the object being controlled.

Returns

the command value

The documentation for this class was generated from the following file:

- **PID.hh**

10.109 gazebo::math::Plane Class Reference

A (p. 111) plane and related functions.

`#include <math/gzmath.hh>`

Public Member Functions

- **Plane** ()

  Constructor.
- **Plane** (const **Vector3** &_normal, double _offset=0.0)

  Constructor from a normal and a distanec.
- **Plane** (const **Vector3** &_normal, const **Vector2d** &_size, double _offset)

  Constructor.
- virtual ∼**Plane** ()

Destructor.

- double **Distance** (const **Vector3** &_origin, const **Vector3** &_dir) const

  Get distance to the plane give an origin and direction.

- **Plane** & **operator=** (const **Plane** &_p)

  Equal operator.

- void **Set** (const **Vector3** &_normal, const **Vector2d** &_size, double offset)

  Set the plane.

Public Attributes

- double **d**

  Plane *(*p. *587) offset.*

- **Vector3 normal**

  Plane *(*p. *587) normal.*

- **Vector2d size**

  Plane *(*p. *587) size.*

10.109.1 Detailed Description

A (p. 111) plane and related functions.

10.109.2 Constructor & Destructor Documentation

10.109.2.1 gazebo::math::Plane::Plane ( )

Constructor.

10.109.2.2 gazebo::math::Plane::Plane (const Vector3 & *_normal,* double *_offset* = `0.0`)

Constructor from a normal and a distanec.

Parameters

in	*_normal*	The plane normal
in	*_offset*	Offset along the normal

10.109.2.3 gazebo::math::Plane::Plane (const Vector3 & *_normal,* const Vector2d & *_size,* double *_offset*)

Constructor.

Parameters

in	*_normal*	The plane normal
in	*_size*	Size of the plane
in	*_offset*	Offset along the normal

10.109.2.4 virtual gazebo::math::Plane::~Plane ( ) [virtual]

Destructor.

10.109.3 Member Function Documentation

10.109.3.1 double gazebo::math::Plane::Distance (const Vector3 & _origin, const Vector3 & _dir) const

Get distance to the plane give an origin and direction.

Parameters

in	*_origin*	the origin
in	*_dir*	a direction

Returns

the shortest distance

10.109.3.2 Plane& gazebo::math::Plane::operator= (const Plane & _p)

Equal operator.

Parameters

_p	another plane

Returns

itself

10.109.3.3 void gazebo::math::Plane::Set (const Vector3 & _normal, const Vector2d & _size, double *offset*)

Set the plane.

Parameters

in	*_normal*	The plane normal
in	*_size*	Size of the plane
in	*_offset*	Offset along the normal

10.109.4 Member Data Documentation

10.109.4.1 double gazebo::math::Plane::d

Plane (p. 587) offset.

10.109.4.2 Vector3 gazebo::math::Plane::normal

Plane (p. 587) normal.

10.109.4.3 Vector2d gazebo::math::Plane::size

Plane (p. 587) size.

The documentation for this class was generated from the following file:

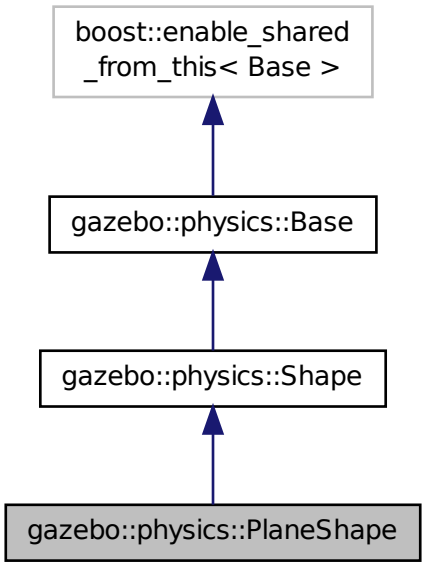
- **Plane.hh**

10.110 gazebo::physics::PlaneShape Class Reference

Collision (p. 195) for an infinite plane.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::PlaneShape:



Public Member Functions

- **PlaneShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**PlaneShape** ()

  Destructor.

- virtual void **CreatePlane** ()

  Create the plane.
- void **FillMsg** (msgs::Geometry &_msg)

  Fill a geometry message with data from this object.
- **math::Vector3 GetNormal** () const

  Get the plane normal.
- **math::Vector2d GetSize** () const

  Get the size.
- virtual void **Init** ()

  Initialize the plane.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  Process a geometry message and use the data to update this object.
- virtual void **SetAltitude** (const **math::Vector3** &_pos)

  Set the altitude of the plane.
- void **SetNormal** (const **math::Vector3** &_norm)

  Set the normal.
- void **SetSize** (const **math::Vector2d** &_size)

  Set the size.

Additional Inherited Members

10.110.1 Detailed Description

Collision (p. 195) for an infinite plane.

This collision is used primarily for ground planes. Note that while the plane in infinite, only the part near the camera is drawn.

10.110.2 Constructor & Destructor Documentation

10.110.2.1 gazebo::physics::PlaneShape::PlaneShape (**CollisionPtr** *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	**Link** (p. 418) to which we are attached.

10.110.2.2 virtual gazebo::physics::PlaneShape::~PlaneShape ( ) [virtual]

Destructor.

10.110.3 Member Function Documentation

10.110.3.1 virtual void gazebo::physics::PlaneShape::CreatePlane ( ) [virtual]

Create the plane.

10.110.3.2 void gazebo::physics::PlaneShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Fill a geometry message with data from this object.

Parameters

out	*_msg*	Message to fill.

Implements **gazebo::physics::Shape** (p. 722).

10.110.3.3 math::Vector3 gazebo::physics::PlaneShape::GetNormal ( ) const

Get the plane normal.

Returns

The plane normal.

10.110.3.4 math::Vector2d gazebo::physics::PlaneShape::GetSize ( ) const

Get the size.

Returns

Size of the plane.

10.110.3.5 virtual void gazebo::physics::PlaneShape::Init ( ) [virtual]

Initialize the plane.

Implements **gazebo::physics::Shape** (p. 722).

10.110.3.6 virtual void gazebo::physics::PlaneShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Process a geometry message and use the data to update this object.

Parameters

in	*_msg*	Message to update from.

Implements **gazebo::physics::Shape** (p. 722).

10.110.3.7 virtual void gazebo::physics::PlaneShape::SetAltitude (const math::Vector3 & *_pos*) [virtual]

Set the altitude of the plane.

Parameters

in	*_pos*	Position of the plane.

10.110.3.8 void gazebo::physics::PlaneShape::SetNormal (const math::Vector3 & _norm)

Set the normal.

Parameters

in	*_norm*	Plane normal.

10.110.3.9 void gazebo::physics::PlaneShape::SetSize (const math::Vector2d & _size)

Set the size.

Parameters

in	*_size*	2D size of the plane.

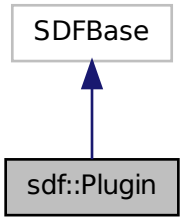
The documentation for this class was generated from the following file:

- **PlaneShape.hh**

10.111 sdf::Plugin Class Reference

`#include <Plugin.hh>`

Inheritance diagram for sdf::Plugin:



Public Member Functions

- **Plugin** ()
- void **Clear** ()
- void **Print** (const std::string &prefix)

Public Attributes

- std::vector< **ParamT**
  < std::string > > **data**

- **ParamT**< std::string > **filename**
- **ParamT**< std::string > **name**

10.111.1 Constructor & Destructor Documentation

10.111.1.1 sdf::Plugin::Plugin ( ) [inline]

10.111.2 Member Function Documentation

10.111.2.1 void sdf::Plugin::Clear ( ) [inline]

References data.

10.111.2.2 void sdf::Plugin::Print (const std::string & *prefix*) [inline]

References filename, and name.

10.111.3 Member Data Documentation

10.111.3.1 std::vector<**ParamT**<std::string> > sdf::Plugin::data

Referenced by Clear().

10.111.3.2 **ParamT**<std::string> sdf::Plugin::filename

Referenced by Print().

10.111.3.3 **ParamT**<std::string> sdf::Plugin::name

Referenced by Print().

The documentation for this class was generated from the following file:

- **sdf/interface/Plugin.hh**

10.112 gazebo::PluginT< T > Class Template Reference

A (p. 111) class which all plugins must inherit from.

`#include <common/common.hh>`

Public Types

- typedef T * **TPtr**

  plugin pointer type definition

Public Member Functions

- std::string **GetFilename** () const

  Get the name of the handler.
- std::string **GetHandle** () const

  Get the short name of the handler.
- **PluginType GetType** () const

  Returns the type of the plugin.

Static Public Member Functions

- static **TPtr Create** (const std::string &_filename, const std::string &_handle)

  a class method that creates a plugin from a file name.

Protected Attributes

- std::string **filename**

  Path to the shared library file.
- std::string **handle**

  Short name.
- **PluginType type**

  Type of plugin.

10.112.1 Detailed Description

template<class T>class gazebo::PluginT< T >

A (p. 111) class which all plugins must inherit from.

10.112.2 Member Typedef Documentation

10.112.2.1 template<class T> typedef T* gazebo::PluginT< T >::TPtr

plugin pointer type definition

10.112.3 Member Function Documentation

10.112.3.1 template<class T> static TPtr gazebo::PluginT< T >::Create (const std::string & *_filename,* const std::string & *_handle*) [inline], [static]

a class method that creates a plugin from a file name.

It locates the shared library and loads it dynamically.

Parameters

in	*_filename*	the path to the shared library.
in	*_handle*	short name of the handler

Returns

Shared Pointer to this class type

10.112.3.2 template<class T> std::string gazebo::PluginT< T >::GetFilename ( ) const [inline]

Get the name of the handler.

10.112.3.3 template<class T> std::string gazebo::PluginT< T >::GetHandle ( ) const [inline]

Get the short name of the handler.

10.112.3.4 template<class T> PluginType gazebo::PluginT< T >::GetType ( ) const [inline]

Returns the type of the plugin.

Returns

type of the plugin

10.112.4 Member Data Documentation

10.112.4.1 template<class T> std::string gazebo::PluginT< T >::filename [protected]

Path to the shared library file.

Referenced by gazebo::PluginT< ModelPlugin >::Create(), and gazebo::PluginT< ModelPlugin >::GetFilename().

10.112.4.2 template<class T> std::string gazebo::PluginT< T >::handle [protected]

Short name.

Referenced by gazebo::PluginT< ModelPlugin >::Create(), and gazebo::PluginT< ModelPlugin >::GetHandle().

10.112.4.3 template<class T> PluginType gazebo::PluginT< T >::type [protected]

Type of plugin.

Referenced by gazebo::PluginT< ModelPlugin >::GetType().

The documentation for this class was generated from the following file:

- **common/Plugin.hh**

10.113 gazebo::math::Pose Class Reference

Encapsulates a position and rotation in three space.

`#include <math/gzmath.hh>`

Public Member Functions

- **Pose** ()

  Default constructors.
- **Pose** (const **Vector3** &_pos, const **Quaternion** &_rot)

  Constructor.
- **Pose** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

  Constructor.
- **Pose** (const **Pose** &_pose)

  Copy constructor.
- virtual ∼**Pose** ()

  Destructor.
- **Pose CoordPoseSolve** (const **Pose** &_b) const

  Find the inverse of a pose; i.e., if b = this + a, given b and this, find a.
- **Vector3 CoordPositionAdd** (const **Vector3** &_pos) const

  Add one point to a vector: result = this + pos.
- **Vector3 CoordPositionAdd** (const **Pose** &_pose) const

  Add one point to another: result = this + pose.
- **Vector3 CoordPositionSub** (const **Pose** &_pose) const

  Subtract one position from another: result = this - pose.
- **Quaternion CoordRotationAdd** (const **Quaternion** &_rot) const

  Add one rotation to another: result = this->rot + rot.
- **Quaternion CoordRotationSub** (const **Quaternion** &_rot) const

  Subtract one rotation from another: result = this->rot - rot.
- void **Correct** ()

  Fix any nan values.
- **Pose GetInverse** () const

  Get the inverse of this pose.
- bool **IsFinite** () const

  See if a pose is finite (e.g., not nan)
- bool **operator!=** (const **Pose** &_pose) const

  Inequality operator.
- **Pose operator**∗ (const **Pose** &_pose)

  Multiplication operator.
- **Pose operator+** (const **Pose** &_pose) const

  Addition operator.
- const **Pose** & **operator+=** (const **Pose** &_pose)

  Add-Equals operator.
- **Pose operator-** () const

  Negation operator.
- **Pose operator-** (const **Pose** &_pose) const

  Subtraction operator.
- const **Pose** & **operator-=** (const **Pose** &_pose)

  Subtraction operator.
- bool **operator==** (const **Pose** &_pose) const

  Equality operator.
- void **Reset** ()

Reset the pose.

- **Pose RotatePositionAboutOrigin** (const **Quaternion** &_rot) const

  Rotate vector part of a pose about the origin.

- void **Round** (int _precision)

  Round all values to _precision decimal places.

- void **Set** (const **Vector3** &_pos, const **Quaternion** &_rot)

  *Set the pose from a **Vector3*** (p. *855) and a **Quaternion*** (p. *623).*

- void **Set** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

  Set the pose from a six tuple.

Public Attributes

- **Vector3 pos**

  The position.

- **Quaternion rot**

  The rotation.

Static Public Attributes

- static const **Pose Zero**

  math::Pose(0, 0, 0, 0, 0, 0)

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Pose** &_pose)

  Stream insertion operator.

- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Pose** &_pose)

  Stream extraction operator.

10.113.1 Detailed Description

Encapsulates a position and rotation in three space.

10.113.2 Constructor & Destructor Documentation

10.113.2.1 gazebo::math::Pose::Pose ( )

Default constructors.

Referenced by operator-().

10.113.2.2 gazebo::math::Pose::Pose (const Vector3 & *_pos,* const Quaternion & *_rot*)

Constructor.

Parameters

in	*_pos*	**A** (p. 111) position
in	*_rot*	**A** (p. 111) rotation

10.113.2.3 gazebo::math::Pose::Pose (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

Constructor.

Parameters

in	*_x*	x position in meters.
in	*_y*	y position in meters.
in	*_z*	z position in meters.
in	*_roll*	Roll (rotation about X-axis) in radians.
in	*_pitch*	Pitch (rotation about y-axis) in radians.
in	*_yaw*	Yaw (rotation about z-axis) in radians.

10.113.2.4 gazebo::math::Pose::Pose (const Pose & _pose)

Copy constructor.

Parameters

in	*_pose*	**Pose** (p. 596) to copy

10.113.2.5 virtual gazebo::math::Pose::~Pose ( ) [virtual]

Destructor.

10.113.3 Member Function Documentation

10.113.3.1 Pose gazebo::math::Pose::CoordPoseSolve (const Pose & _b) const

Find the inverse of a pose; i.e., if b = this + a, given b and this, find a.

Parameters

in	*_b*	the other pose

10.113.3.2 Vector3 gazebo::math::Pose::CoordPositionAdd (const Vector3 & _pos) const

Add one point to a vector: result = this + pos.

Parameters

in	*_pos*	Position to add to this pose

Returns

the resulting position

10.113.3.3 Vector3 gazebo::math::Pose::CoordPositionAdd (const Pose & *_pose*) const

Add one point to another: result = this + pose.

Parameters

in	*_pose*	The **Pose** (p. 596) to add

Returns

The resulting position

10.113.3.4 Vector3 gazebo::math::Pose::CoordPositionSub (const Pose & *_pose*) const [inline]

Subtract one position from another: result = this - pose.

Parameters

in	*_pose*	**Pose** (p. 596) to subtract

Returns

The resulting position

References gazebo::math::Quaternion::GetInverse(), pos, rot, gazebo::math::Vector3::x, gazebo::math::Quaternion::x, gazebo::math::Vector3::y, gazebo::math::Quaternion::y, gazebo::math::Vector3::z, and gazebo::math::Quaternion::z.

Referenced by operator-().

10.113.3.5 Quaternion gazebo::math::Pose::CoordRotationAdd (const Quaternion & *_rot*) const

Add one rotation to another: result = this->rot + rot.

Parameters

in	*_rot*	Rotation to add

Returns

The resulting rotation

10.113.3.6 Quaternion gazebo::math::Pose::CoordRotationSub (const Quaternion & *_rot*) const [inline]

Subtract one rotation from another: result = this->rot - rot.

Parameters

in	*_rot*	The rotation to subtract

Returns

The resulting rotation

References gazebo::math::Quaternion::GetInverse(), gazebo::math::Quaternion::Normalize(), and rot.

Referenced by operator-().

10.113.3.7 void gazebo::math::Pose::Correct ( ) [inline]

Fix any nan values.

References gazebo::math::Quaternion::Correct(), gazebo::math::Vector3::Correct(), pos, and rot.

10.113.3.8 Pose gazebo::math::Pose::GetInverse ( ) const

Get the inverse of this pose.

Returns

the inverse pose

10.113.3.9 bool gazebo::math::Pose::IsFinite ( ) const

See if a pose is finite (e.g., not nan)

10.113.3.10 bool gazebo::math::Pose::operator!= (const Pose & *_pose*) const

Inequality operator.

Parameters

in	*_pose*	**Pose** (p. 596) for comparison

Returns

True if not equal

10.113.3.11 Pose gazebo::math::Pose::operator* (const Pose & *_pose*)

Multiplication operator.

Parameters

in	*_pose*	the other pose

Returns

itself

10.113.3.12 Pose gazebo::math::Pose::operator+ (const Pose & _pose) const

Addition operator.

Parameters

in	*_pose*	**Pose** (p. 596) to add to this pose

Returns

The resulting pose

10.113.3.13 const Pose& gazebo::math::Pose::operator+= (const Pose & _pose)

Add-Equals operator.

Parameters

in	*_pose*	**Pose** (p. 596) to add to this pose

Returns

The resulting pose

10.113.3.14 Pose gazebo::math::Pose::operator- ( ) const [inline]

Negation operator.

Returns

The resulting pose

References Pose().

10.113.3.15 Pose gazebo::math::Pose::operator- (const Pose & _pose) const [inline]

Subtraction operator.

Parameters

in	*_pose*	**Pose** (p. 596) to subtract from this one

Returns

The resulting pose

References CoordPositionSub(), CoordRotationSub(), Pose(), and rot.

10.113.3.16 const Pose& gazebo::math::Pose::operator-= (const Pose & _pose)

Subtraction operator.

Parameters

in	*_pose*	**Pose** (p. 596) to subtract from this one

Returns

The resulting pose

10.113.3.17 bool gazebo::math::Pose::operator== (const Pose & *_pose*) const

Equality operator.

Parameters

in	*_pose*	**Pose** (p. 596) for comparison

Returns

True if equal

10.113.3.18 void gazebo::math::Pose::Reset ( )

Reset the pose.

10.113.3.19 Pose gazebo::math::Pose::RotatePositionAboutOrigin (const Quaternion & *_rot*) const

Rotate vector part of a pose about the origin.

Parameters

in	*_rot*	rotation

Returns

the rotated pose

10.113.3.20 void gazebo::math::Pose::Round (int *_precision*)

Round all values to _precision decimal places.

Parameters

in	*_precision*	

10.113.3.21 void gazebo::math::Pose::Set (const Vector3 & *_pos,* const Quaternion & *_rot*)

Set the pose from a **Vector3** (p. 855) and a **Quaternion** (p. 623).

Parameters

in	*_pos*	The position.
in	*_rot*	The rotation.

10.113.3.22 void gazebo::math::Pose::Set (double *_x*, double *_y*, double *_z*, double *_roll*, double *_pitch*, double *_yaw*)

Set the pose from a six tuple.

Parameters

in	*_x*	x position in meters.
in	*_y*	y position in meters.
in	*_z*	z position in meters.
in	*_roll*	Roll (rotation about X-axis) in radians.
in	*_pitch*	Pitch (rotation about y-axis) in radians.
in	*_yaw*	Pitch (rotation about z-axis) in radians.

10.113.4 Friends And Related Function Documentation

10.113.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Pose & *_pose*) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_pose*	pose to output

Returns

the stream

10.113.4.2 std::istream& operator>> (std::istream & *_in*, gazebo::math::Pose & *_pose*) [friend]

Stream extraction operator.

Parameters

in	*_in*	the input stream
in	*_pose*	the pose

Returns

the stream

10.113.5 Member Data Documentation

10.113.5.1 Vector3 gazebo::math::Pose::pos

The position.

Referenced by CoordPositionSub(), Correct(), and gazebo::physics::Inertial::GetCoG().

10.113.5.2 Quaternion gazebo::math::Pose::rot

The rotation.

Referenced by CoordPositionSub(), CoordRotationSub(), Correct(), and operator-().

10.113.5.3 const Pose gazebo::math::Pose::Zero [static]

math::Pose(0, 0, 0, 0, 0, 0)

The documentation for this class was generated from the following file:

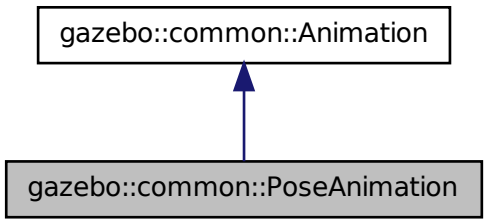
- **Pose.hh**

10.114 gazebo::common::PoseAnimation Class Reference

A (p. 111) pose animation.

`#include <Animation.hh>`

Inheritance diagram for gazebo::common::PoseAnimation:



Public Member Functions

- **PoseAnimation** (const std::string &_name, double _length, bool _loop)

  Constructor.
- virtual ∼**PoseAnimation** ()

  Destructor.
- **PoseKeyFrame** ∗ **CreateKeyFrame** (double _time)

  Create a pose keyframe at the given time.
- void **GetInterpolatedKeyFrame** (**PoseKeyFrame** &_kf) const

  Get a keyframe using the animation's current time.

Protected Member Functions

- void **BuildInterpolationSplines** () const

  Update the pose splines.
- void **GetInterpolatedKeyFrame** (double _time, **PoseKeyFrame** &_kf) const

  Get a keyframe using a passed in time.

Additional Inherited Members

10.114.1 Detailed Description

A (p. 111) pose animation.

10.114.2 Constructor & Destructor Documentation

10.114.2.1 gazebo::common::PoseAnimation::PoseAnimation (const std::string & *_name,* double *_length,* bool *_loop*)

Constructor.

Parameters

in	*_name*	String name of the animation. This should be unique.
in	*_length*	Length of the animation in seconds
in	*_loop*	True == loop the animation

10.114.2.2 virtual gazebo::common::PoseAnimation::~PoseAnimation ( ) [virtual]

Destructor.

10.114.3 Member Function Documentation

10.114.3.1 void gazebo::common::PoseAnimation::BuildInterpolationSplines ( ) const [protected]

Update the pose splines.

10.114.3.2 PoseKeyFrame* gazebo::common::PoseAnimation::CreateKeyFrame (double *_time*)

Create a pose keyframe at the given time.

Parameters

in	*_time*	**Time** (p. 791) at which to create the keyframe

Returns

Pointer to the new keyframe

10.114.3.3 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (PoseKeyFrame & _kf) const

Get a keyframe using the animation's current time.

Parameters

out	*_kf*	**PoseKeyFrame** (p. 607) reference to hold the interpolated result

10.114.3.4 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (double _time, PoseKeyFrame & _kf) const [protected]

Get a keyframe using a passed in time.

Parameters

in	*_time*	**Time** (p. 791) in seconds
out	*_kf*	**PoseKeyFrame** (p. 607) reference to hold the interpolated result

The documentation for this class was generated from the following file:

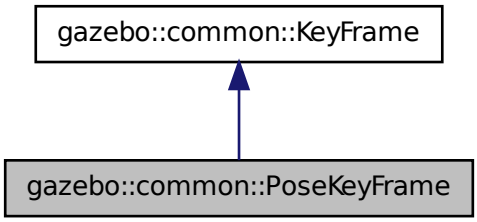
- **Animation.hh**

10.115 gazebo::common::PoseKeyFrame Class Reference

A (p. 111) keyframe for a **PoseAnimation** (p. 605).

`#include <KeyFrame.hh>`

Inheritance diagram for gazebo::common::PoseKeyFrame:



Public Member Functions

- **PoseKeyFrame** (double _time)

  Constructor.
- virtual ~**PoseKeyFrame** ()

  Destructor.

- const **math::Quaternion** & **GetRotation** () const

  Get the rotation of the keyframe.
- const **math::Vector3** & **GetTranslation** () const

  Get the translation of the keyframe.
- void **SetRotation** (const **math::Quaternion** &_rot)

  Set the rotation for the keyframe.
- void **SetTranslation** (const **math::Vector3** &_trans)

  Set the translation for the keyframe.

Protected Attributes

- **math::Quaternion rotate**

  the rotation quaternion
- **math::Vector3 translate**

  the translation vector

10.115.1 Detailed Description

A (p. 111) keyframe for a **PoseAnimation** (p. 605).

10.115.2 Constructor & Destructor Documentation

10.115.2.1 gazebo::common::PoseKeyFrame::PoseKeyFrame (double *_time*)

Constructor.

Parameters

in	*_time*	of the keyframe

10.115.2.2 virtual gazebo::common::PoseKeyFrame::~PoseKeyFrame ( ) [virtual]

Destructor.

10.115.3 Member Function Documentation

10.115.3.1 const math::Quaternion& gazebo::common::PoseKeyFrame::GetRotation ( ) const

Get the rotation of the keyframe.

Returns

The rotation amount

10.115.3.2 const math::Vector3& gazebo::common::PoseKeyFrame::GetTranslation ( ) const

Get the translation of the keyframe.

Returns

The translation amount

10.115.3.3 void gazebo::common::PoseKeyFrame::SetRotation (const math::Quaternion & *_rot*)

Set the rotation for the keyframe.

Parameters

in	*_rot*	Rotation amount

10.115.3.4 void gazebo::common::PoseKeyFrame::SetTranslation (const math::Vector3 & *_trans*)

Set the translation for the keyframe.

Parameters

in	*_trans*	Translation amount

10.115.4 Member Data Documentation

10.115.4.1 math::Quaternion gazebo::common::PoseKeyFrame::rotate [protected]

the rotation quaternion

10.115.4.2 math::Vector3 gazebo::common::PoseKeyFrame::translate [protected]

the translation vector

The documentation for this class was generated from the following file:

- **KeyFrame.hh**

10.116 gazebo::rendering::Projector Class Reference

Projects a material onto surface, light a light projector.

`#include <rendering/rendering.hh>`

Public Member Functions

- **Projector** (**VisualPtr** _parent)
  Constructor.
- virtual ~**Projector** ()
  Destructor.
- **VisualPtr GetParent** ()
  Get the parent visual.

- void **Load** (**sdf::ElementPtr** _sdf)

  Load from an sdf pointer.
- void **Load** (const msgs::Projector &_msg)

  Load from a message.
- void **Load** (const std::string &_name, const **math::Pose** &_pose=**math::Pose**(0, 0, 0, 0, 0, 0), const std::string &_textureName="", double _nearClip=0.25, double _farClip=15.0, double _fov=M_PI ∗0.25)

  Load the projector.
- void **SetEnabled** (bool _enabled)

  Set whether the projector is enabled or disabled.
- void **SetTexture** (const std::string &_textureName)

  Load a texture into the projector.
- void **Toggle** ()

  Toggle the activation of the projector.

10.116.1 Detailed Description

Projects a material onto surface, light a light projector.

10.116.2 Constructor & Destructor Documentation

10.116.2.1 gazebo::rendering::Projector::Projector (VisualPtr *_parent*)

Constructor.

Parameters

in	*_parent*	Name of the parent visual.

10.116.2.2 virtual gazebo::rendering::Projector::∼Projector ( ) [virtual]

Destructor.

10.116.3 Member Function Documentation

10.116.3.1 VisualPtr gazebo::rendering::Projector::GetParent ( )

Get the parent visual.

Returns

Pointer ot the parent visual.

10.116.3.2 void gazebo::rendering::Projector::Load (sdf::ElementPtr *_sdf*)

Load from an sdf pointer.

Parameters

in	*_sdf*	Pointer to the SDF element.

10.116.3.3 void gazebo::rendering::Projector::Load (const msgs::Projector & *_msg*)

Load from a message.

Parameters

in	*_msg*	Load from a message.

10.116.3.4 void gazebo::rendering::Projector::Load (const std::string & *_name*, const math::Pose & *_pose* = math::Pose(0, 0, 0, 0, 0, 0), const std::string & *_textureName* = "", double *_nearClip* = 0.25, double *_farClip* = 15.0, double *_fov* = M_PI *0.25)

Load the projector.

Parameters

in	*_name*	Name of the projector.
in	*_pos*	Pose of the projector.
in	*_textureName*	Name of the texture to project.
in	*_nearClip*	Near clip distance.
in	*_farClip*	Far clip distance.
in	*_fov*	Field of view.

10.116.3.5 void gazebo::rendering::Projector::SetEnabled (bool *_enabled*)

Set whether the projector is enabled or disabled.

Parameters

in	*_enabled*	True to enable the projector.

10.116.3.6 void gazebo::rendering::Projector::SetTexture (const std::string & *_textureName*)

Load a texture into the projector.

Parameters

in	*_textureName*	Name of the texture to project.

10.116.3.7 void gazebo::rendering::Projector::Toggle ( )

Toggle the activation of the projector.

The documentation for this class was generated from the following file:

- **Projector.hh**

10.117 gazebo::transport::Publication Class Reference

A (p. 111) publication for a topic.

`#include <transport/transport.hh>`

Public Member Functions

- **Publication** (const std::string &_topic, const std::string &_msgType)
  Constructor.
- virtual ~**Publication** ()
  Destructor.
- void **AddPublisher** (**PublisherPtr** _pub)
  Add a publisher.
- void **AddSubscription** (const **CallbackHelperPtr** _callback)
  Subscribe a callback to our topic.
- void **AddSubscription** (const **NodePtr** &_node)
  Subscribe a node to our topic.
- void **AddTransport** (const **PublicationTransportPtr** &_publink)
  Add a transport.
- unsigned int **GetCallbackCount** () const
  Get the number of callbacks.
- bool **GetLocallyAdvertised** () const
  Was the topic has been advertised from this process?
- std::string **GetMsgType** () const
  Get the type of message.
- unsigned int **GetNodeCount** () const
  Get the number of nodes.
- unsigned int **GetRemoteSubscriptionCount** ()
  Get the number of remote subscriptions.
- unsigned int **GetTransportCount** () const
  Get the number of transports.
- bool **HasTransport** (const std::string &_host, unsigned int _port)
  Does a given transport exist?
- void **LocalPublish** (const std::string &_data)
  Publish data to local subscribers (skip serialization)
- void **Publish** (**MessagePtr** _msg, const boost::function< void()> &_cb=**NULL**)
  Publish data to remote subscribers.
- void **RemoveSubscription** (const **NodePtr** &_node)
  Unsubscribe a node from our topic.
- void **RemoveSubscription** (const std::string &_host, unsigned int _port)
  Unsubscribe a a node by host/port from our topic.
- void **RemoveTransport** (const std::string &_host, unsigned int _port)
  Remove a transport.
- void **SetLocallyAdvertised** (bool _value)
  Set whether this topic has been advertised from this process.

10.117.1 Detailed Description

A (p. 111) publication for a topic.

This facilitates transport of messages

10.117.2 Constructor & Destructor Documentation

10.117.2.1 gazebo::transport::Publication::Publication (const std::string & *_topic,* const std::string & *_msgType*)

Constructor.

Parameters

in	*_topic*	The topic we're publishing
in	*_msgType*	The type of the topic we're publishing

10.117.2.2 virtual gazebo::transport::Publication::~Publication ( ) [virtual]

Destructor.

10.117.3 Member Function Documentation

10.117.3.1 void gazebo::transport::Publication::AddPublisher (PublisherPtr *_pub*)

Add a publisher.

Parameters

in,out	*_pub*	Pointer to publisher object to be added

Referenced by gazebo::transport::TopicManager::Advertise().

10.117.3.2 void gazebo::transport::Publication::AddSubscription (const CallbackHelperPtr *_callback*)

Subscribe a callback to our topic.

Parameters

in	*_callback*	The callback

Referenced by gazebo::transport::TopicManager::Advertise().

10.117.3.3 void gazebo::transport::Publication::AddSubscription (const NodePtr & *_node*)

Subscribe a node to our topic.

Parameters

in	*_node*	The node

10.117.3.4 void gazebo::transport::Publication::AddTransport (const PublicationTransportPtr & _publink)

Add a transport.

Parameters

in	*_publink*	Pointer to publication transport object to be added

10.117.3.5 unsigned int gazebo::transport::Publication::GetCallbackCount ( ) const

Get the number of callbacks.

Returns

The number of callbacks

10.117.3.6 bool gazebo::transport::Publication::GetLocallyAdvertised ( ) const

Was the topic has been advertised from this process?

Returns

true if the topic has been advertised from this process, false otherwise

Referenced by gazebo::transport::TopicManager::Advertise().

10.117.3.7 std::string gazebo::transport::Publication::GetMsgType ( ) const

Get the type of message.

Returns

The type of message

10.117.3.8 unsigned int gazebo::transport::Publication::GetNodeCount ( ) const

Get the number of nodes.

Returns

The number of nodes

10.117.3.9 unsigned int gazebo::transport::Publication::GetRemoteSubscriptionCount ( )

Get the number of remote subscriptions.

Returns

The number of remote subscriptions

10.117.3.10 unsigned int gazebo::transport::Publication::GetTransportCount ( ) const

Get the number of transports.

Returns

The number of transports

10.117.3.11 bool gazebo::transport::Publication::HasTransport (const std::string & *_host*, unsigned int *_port*)

Does a given transport exist?

Parameters

in	*_host*	Hostname of the transport
in	*_port*	Port of the transport

Returns

true if the transport exists, false otherwise

10.117.3.12 void gazebo::transport::Publication::LocalPublish (const std::string & *_data*)

Publish data to local subscribers (skip serialization)

Parameters

in	*_data*	The data to be published

10.117.3.13 void gazebo::transport::Publication::Publish (MessagePtr *_msg*, const boost::function< void() > & *_cb* = NULL)

Publish data to remote subscribers.

Parameters

in	*_msg*	Message to be published
in	*_cb*	If non-null, callback to be invoked after publishing is completed

10.117.3.14 void gazebo::transport::Publication::RemoveSubscription (const NodePtr & *_node*)

Unsubscribe a node from our topic.

Parameters

in	*_node*	The node

10.117.3.15 void gazebo::transport::Publication::RemoveSubscription (const std::string & *_host,* unsigned int *_port*)

Unsubscribe a a node by host/port from our topic.

Parameters

in	*_host*	The node's hostname
in	*_port*	The node's port

10.117.3.16 void gazebo::transport::Publication::RemoveTransport (const std::string & *_host,* unsigned int *_port*)

Remove a transport.

Parameters

in	*_host*	The transport's hostname
in	*_port*	The transport's port

10.117.3.17 void gazebo::transport::Publication::SetLocallyAdvertised (bool *_value*)

Set whether this topic has been advertised from this process.

Parameters

in	*_value*	If true, the topic was locally advertise, otherwise it was not

Referenced by gazebo::transport::TopicManager::Advertise().

The documentation for this class was generated from the following file:

- **Publication.hh**

10.118 gazebo::transport::PublicationTransport Class Reference

transport/transport.hh

```
#include <PublicationTransport.hh>
```

Public Member Functions

- **PublicationTransport** (const std::string &_topic, const std::string &_msgType)
  Constructor.
- virtual ~**PublicationTransport** ()
  Destructor.
- void **AddCallback** (const boost::function< void(const std::string &)> &_cb)
  Add a callback to the transport.
- void **Fini** ()
  Finalize the transport.
- const **ConnectionPtr GetConnection** () const

Get the underlying connection.

- std::string **GetMsgType** () const

  Get the topic type.

- std::string **GetTopic** () const

  Get the topic name.

- void **Init** (const **ConnectionPtr** &_conn, bool _latched)

  Initialize the transport.

10.118.1 Detailed Description

transport/transport.hh

Reads data from a remote advertiser, and passes the data along to local subscribers

10.118.2 Constructor & Destructor Documentation

10.118.2.1 gazebo::transport::PublicationTransport::PublicationTransport (const std::string & *_topic,* const std::string & *_msgType*)

Constructor.

Parameters

in	*_topic*	Topic that we're publishing
in	*_topic*	Type of the topic that we're publishing

10.118.2.2 virtual gazebo::transport::PublicationTransport::~PublicationTransport ( ) [virtual]

Destructor.

10.118.3 Member Function Documentation

10.118.3.1 void gazebo::transport::PublicationTransport::AddCallback (const boost::function< void(const std::string &)> & *_cb*)

Add a callback to the transport.

Parameters

in	*_cb*	The callback to be added

10.118.3.2 void gazebo::transport::PublicationTransport::Fini ( )

Finalize the transport.

10.118.3.3 const ConnectionPtr gazebo::transport::PublicationTransport::GetConnection ( ) const

Get the underlying connection.

Returns

Pointer to the underlying connection

10.118.3.4 std::string gazebo::transport::PublicationTransport::GetMsgType ( ) const

Get the topic type.

Returns

The topic type

10.118.3.5 std::string gazebo::transport::PublicationTransport::GetTopic ( ) const

Get the topic name.

Returns

The topic name

10.118.3.6 void gazebo::transport::PublicationTransport::Init (const ConnectionPtr & *_conn,* bool *_latched*)

Initialize the transport.

Parameters

in	*_conn*	The underlying connection.
in	*_latched*	True to grab the last message sent on the topic.

The documentation for this class was generated from the following file:

- **PublicationTransport.hh**

10.119 gazebo::transport::Publisher Class Reference

A (p. 111) publisher of messages on a topic.

`#include <transport/transport.hh>`

Public Member Functions

- **Publisher** (const std::string &_topic, const std::string &_msgType, unsigned int _limit, bool _latch) **GAZEBO_DEPRECATED**(1.5)

  Deprecated.
- **Publisher** (const std::string &_topic, const std::string &_msgType, unsigned int _limit, double _hzRate)

  Constructor.
- virtual ~**Publisher** ()

  Destructor.

- bool **GetLatching** () const **GAZEBO_DEPRECATED**(1.5)

  Deprecated.
- std::string **GetMsgType** () const

  Get the message type.
- unsigned int **GetOutgoingCount** () const

  Get the number of outgoing messages.
- std::string **GetPrevMsg** () const

  Get the previously published message.
- **MessagePtr GetPrevMsgPtr** () const

  Get the previously published message.
- std::string **GetTopic** () const

  Get the topic name.
- bool **HasConnections** () const

  Are there any connections?
- void **Publish** (const google::protobuf::Message &_message, bool _block=false)

  Publish a protobuf message on the topic.
- template<typename M >
  void **Publish** (M _message, bool _block=false)

  Publish an arbitrary message on the topic.
- void **SendMessage** ()

  Send latest message over the wire. For internal use only.
- void **SetPublication** (**PublicationPtr** &_publication, int _i) **GAZEBO_DEPRECATED**(1.5)

  DEPRECATED in version 1.6.
- void **SetPublication** (**PublicationPtr** _publication)

  Set the publication object for a particular publication.
- void **WaitForConnection** () const

  Block until a connection has been established with this publisher.

10.119.1 Detailed Description

A (p. 111) publisher of messages on a topic.

10.119.2 Constructor & Destructor Documentation

10.119.2.1 gazebo::transport::Publisher::Publisher (const std::string & *_topic*, const std::string & *_msgType*, unsigned int *_limit*, bool *_latch*)

Deprecated.

10.119.2.2 gazebo::transport::Publisher::Publisher (const std::string & *_topic*, const std::string & *_msgType*, unsigned int *_limit*, double *_hzRate*)

Constructor.

Parameters

in	*_topic*	Name of topic to be published
in	*_msgType*	Type of the message to be published
in	*_limit*	Maximum number of outgoing messages to queue
in	*_hz*	Update rate for the publisher. Units are 1.0/seconds.

10.119.2.3 virtual gazebo::transport::Publisher::~Publisher ( ) [virtual]

Destructor.

10.119.3 Member Function Documentation

10.119.3.1 bool gazebo::transport::Publisher::GetLatching ( ) const

Deprecated.

10.119.3.2 std::string gazebo::transport::Publisher::GetMsgType ( ) const

Get the message type.

Returns

The message type

10.119.3.3 unsigned int gazebo::transport::Publisher::GetOutgoingCount ( ) const

Get the number of outgoing messages.

Returns

The number of outgoing messages

10.119.3.4 std::string gazebo::transport::Publisher::GetPrevMsg ( ) const

Get the previously published message.

Returns

The previously published message, if any

10.119.3.5 MessagePtr gazebo::transport::Publisher::GetPrevMsgPtr ( ) const

Get the previously published message.

Returns

The previously published message, if any

10.119.3.6 std::string gazebo::transport::Publisher::GetTopic ( ) const

Get the topic name.

Returns

The topic name

10.119.3.7 bool gazebo::transport::Publisher::HasConnections ( ) const

Are there any connections?

Returns

true if there are any connections, false otherwise

10.119.3.8 void gazebo::transport::Publisher::Publish (const google::protobuf::Message & *_message,* bool *_block* = `false`) [inline]

Publish a protobuf message on the topic.

Parameters

in	*_message*	Message to be published
in	*_block*	Whether to block until the message is actually written out

Referenced by gazebo::transport::PublishTask::execute().

10.119.3.9 template<typename M > void gazebo::transport::Publisher::Publish (M *_message,* bool *_block* = `false`) [inline]

Publish an arbitrary message on the topic.

Parameters

in	*_message*	Message to be published
in	*_block*	Whether to block until the message is actually written out

10.119.3.10 void gazebo::transport::Publisher::SendMessage ( )

Send latest message over the wire. For internal use only.

10.119.3.11 void gazebo::transport::Publisher::SetPublication (PublicationPtr & *_publication,* int *_i*)

DEPRECATED in version 1.6.

See Also

SetPublication (p. 621)

10.119.3.12 void gazebo::transport::Publisher::SetPublication (PublicationPtr *_publication*)

Set the publication object for a particular publication.

Parameters

in	*_publication*	Pointer to the publication object to be set

10.119.3.13 void gazebo::transport::Publisher::WaitForConnection ( ) const

Block until a connection has been established with this publisher.

Referenced by gazebo::transport::PublishTask::execute().

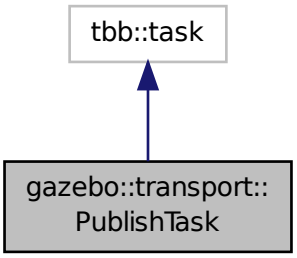
The documentation for this class was generated from the following file:

- **Publisher.hh**

10.120 gazebo::transport::PublishTask Class Reference

`#include <Node.hh>`

Inheritance diagram for gazebo::transport::PublishTask:



Public Member Functions

- **PublishTask** (**transport::PublisherPtr** _pub, const google::protobuf::Message &_message)

  Constructor.
- tbb::task * **execute** ()

  Overridden function from tbb::task that exectues the publish task.

10.120.1 Detailed Description

Task used by **Node::Publish** (p. 540) to publish on a one-time publisher

10.120.2 Constructor & Destructor Documentation

10.120.2.1 gazebo::transport::PublishTask::PublishTask (transport::PublisherPtr *_pub,* const google::protobuf::Message & *_message*) [inline]

Constructor.

Parameters

in	*_pub*	**Publisher** (p. 618) to publish the message on.
in	*_message*	Message to publish

10.120.3 Member Function Documentation

10.120.3.1 tbb::task* gazebo::transport::PublishTask::execute ( ) [inline]

Overridden function from tbb::task that exectues the publish task.

References NULL, gazebo::transport::Publisher::Publish(), and gazebo::transport::Publisher::WaitForConnection().

The documentation for this class was generated from the following file:

- **Node.hh**

10.121 gazebo::math::Quaternion Class Reference

A (p. 111) quaternion class.

`#include <math/gzmath.hh>`

Public Member Functions

- **Quaternion** ()
  Default Constructor.
- **Quaternion** (const double &_w, const double &_x, const double &_y, const double &_z)
  Constructor.
- **Quaternion** (const double &_roll, const double &_pitch, const double &_yaw)
  Constructor from Euler angles in radians.
- **Quaternion** (const **Vector3** &_axis, const double &_angle)
  Constructor from axis angle.
- **Quaternion** (const **Vector3** &_rpy)
  Constructor.
- **Quaternion** (const **Quaternion** &_qt)
  Copy constructor.
- ~**Quaternion** ()
  Destructor.
- void **Correct** ()
  Correct any nan.
- double **Dot** (const **Quaternion** &_q) const
  Dot product.
- void **GetAsAxis** (**Vector3** &_axis, double &_angle) const
  Return rotation as axis and angle.
- **Vector3 GetAsEuler** () const
  Return the rotation in Euler angles.
- **Matrix3 GetAsMatrix3** () const
  Get the quaternion as a 3x3 matrix.

- **Matrix4 GetAsMatrix4** () const

  Get the quaternion as a 4x4 matrix.
- **Quaternion GetExp** () const

  Return the exponent.
- **Quaternion GetInverse** () const

  Get the inverse of this quaternion.
- **Quaternion GetLog** () const

  Return the logarithm.
- double **GetPitch** ()

  Get the Euler pitch angle in radians.
- double **GetRoll** ()

  Get the Euler roll angle in radians.
- **Vector3 GetXAxis** () const

  Return the X axis.
- double **GetYaw** ()

  Get the Euler yaw angle in radians.
- **Vector3 GetYAxis** () const

  Return the Y axis.
- **Vector3 GetZAxis** () const

  Return the Z axis.
- void **Invert** ()

  Invert the quaternion.
- bool **IsFinite** () const

  See if a quatern is finite (e.g., not nan)
- void **Normalize** ()

  Normalize the quaternion.
- bool **operator!=** (const **Quaternion** &_qt) const

  Not equal to operator.
- **Quaternion operator**∗ (const **Quaternion** &_q) const

  Multiplication operator.
- **Quaternion operator**∗ (const double &_f) const

  Multiplication operator.
- **Vector3 operator**∗ (const **Vector3** &_v) const

  Vector3 *(*p. *855) multiplication operator.*
- **Quaternion operator**∗**=** (const **Quaternion** &qt)

  Multiplication operator.
- **Quaternion operator+** (const **Quaternion** &_qt) const

  Addition operator.
- **Quaternion operator+=** (const **Quaternion** &_qt)

  Addition operator.
- **Quaternion operator-** (const **Quaternion** &_qt) const

  Substraction operator.
- **Quaternion operator-** () const

  Unary minus operator.
- **Quaternion operator-=** (const **Quaternion** &_qt)

  Substraction operator.
- **Quaternion** & **operator=** (const **Quaternion** &_qt)

Equal operator.

- bool **operator==** (const **Quaternion** &_qt) const

  Equal to operator.
- **Vector3 RotateVector** (const **Vector3** &_vec) const

  Rotate a vector using the quaternion.
- **Vector3 RotateVectorReverse** (**Vector3** _vec) const

  Do the reverse rotation of a vector by this quaternion.
- void **Round** (int _precision)

  Round all values to _precision decimal places.
- void **Scale** (double _scale)

  Scale a Quaternionion.
- void **Set** (double _u, double _x, double _y, double _z)

  Set this quaternion from 4 floating numbers.
- void **SetFromAxis** (double _x, double _y, double _z, double _a)

  Set the quaternion from an axis and angle.
- void **SetFromAxis** (const **Vector3** &_axis, double _a)

  Set the quaternion from an axis and angle.
- void **SetFromEuler** (const **Vector3** &_vec)

  Set the quaternion from Euler angles.
- void **SetFromEuler** (double _roll, double _pitch, double _yaw)

  Set the quaternion from Euler angles.
- void **SetToIdentity** ()

  Set the quatern to the identity.

Static Public Member Functions

- static **Quaternion EulerToQuaternion** (const **Vector3** &_vec)

  Convert euler angles to quatern.
- static **Quaternion EulerToQuaternion** (double _x, double _y, double _z)

  Convert euler angles to quatern.
- static **Quaternion Slerp** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkQ, bool _shortestPath=false)

  Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.
- static **Quaternion Squad** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkA, const **Quaternion** &_rkB, const **Quaternion** &_rkQ, bool _shortestPath=false)

  Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.

Public Attributes

- double **w**

  Attributes of the quaternion.
- double **x**

  Attributes of the quaternion.
- double **y**

  Attributes of the quaternion.
- double **z**

  Attributes of the quaternion.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Quaternion** &_q)

  Stream insertion operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Quaternion** &_q)

  Stream extraction operator.

10.121.1 Detailed Description

A (p. 111) quaternion class.

10.121.2 Constructor & Destructor Documentation

10.121.2.1 gazebo::math::Quaternion::Quaternion ( )

Default Constructor.

Referenced by operator∗().

10.121.2.2 gazebo::math::Quaternion::Quaternion (const double & *_w,* const double & *_x,* const double & *_y,* const double & *_z*)

Constructor.

Parameters

in	*_w*	W param
in	*_x*	X param
in	*_y*	Y param
in	*_z*	Z param

10.121.2.3 gazebo::math::Quaternion::Quaternion (const double & *_roll,* const double & *_pitch,* const double & *_yaw*)

Constructor from Euler angles in radians.

Parameters

in	*_roll*	roll
in	*_pitch*	pitch
in	*_yaw*	yaw

10.121.2.4 gazebo::math::Quaternion::Quaternion (const Vector3 & *_axis,* const double & *_angle*)

Constructor from axis angle.

Parameters

in	*_axis*	the rotation axis
in	*_angle*	the rotation angle in radians

10.121.2.5 gazebo::math::Quaternion::Quaternion (const Vector3 & _rpy)

Constructor.

Parameters

in	*_rpy*	euler angles

10.121.2.6 gazebo::math::Quaternion::Quaternion (const Quaternion & _qt)

Copy constructor.

Parameters

qt	**Quaternion** (p. 623) to copy

10.121.2.7 gazebo::math::Quaternion::~Quaternion ( )

Destructor.

10.121.3 Member Function Documentation

10.121.3.1 void gazebo::math::Quaternion::Correct ( ) [inline]

Correct any nan.

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.121.3.2 double gazebo::math::Quaternion::Dot (const Quaternion & _q) const

Dot product.

Parameters

in	*_q*	the other quaternion

Returns

the product

10.121.3.3 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (const Vector3 & _vec) [static]

Convert euler angles to quatern.

Parameters

in		

10.121.3.4 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (double _x, double _y, double _z) [static]

Convert euler angles to quatern.

Parameters

in	*_x*	rotation along x
in	*_y*	rotation along y
in	*_z*	rotation along z

10.121.3.5 void gazebo::math::Quaternion::GetAsAxis (Vector3 & _axis, double & _angle) const

Return rotation as axis and angle.

Parameters

in	*_axis*	rotation axis
in	*_angle*	ccw angle in radians

10.121.3.6 Vector3 gazebo::math::Quaternion::GetAsEuler ( ) const

Return the rotation in Euler angles.

Returns

This quaternion as an Euler vector

10.121.3.7 Matrix3 gazebo::math::Quaternion::GetAsMatrix3 ( ) const

Get the quaternion as a 3x3 matrix.

10.121.3.8 Matrix4 gazebo::math::Quaternion::GetAsMatrix4 ( ) const

Get the quaternion as a 4x4 matrix.

Returns

a 4x4 matrix

10.121.3.9 Quaternion gazebo::math::Quaternion::GetExp ( ) const

Return the exponent.

Returns

the exp

10.121.3.10 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().

10.121.3.11 Quaternion gazebo::math::Quaternion::GetLog ( ) const

Return the logarithm.

Returns

the log

10.121.3.12 double gazebo::math::Quaternion::GetPitch ( )

Get the Euler pitch angle in radians.

Returns

the pitch

10.121.3.13 double gazebo::math::Quaternion::GetRoll ( )

Get the Euler roll angle in radians.

Returns

the roll

10.121.3.14 Vector3 gazebo::math::Quaternion::GetXAxis ( ) const

Return the X axis.

Returns

the vector

10.121.3.15 double gazebo::math::Quaternion::GetYaw ( )

Get the Euler yaw angle in radians.

Returns

the yaw

10.121.3.16 Vector3 gazebo::math::Quaternion::GetYAxis ( ) const

Return the Y axis.

Returns

the vector

10.121.3.17 Vector3 gazebo::math::Quaternion::GetZAxis ( ) const

Return the Z axis.

Returns

the vector

10.121.3.18 void gazebo::math::Quaternion::Invert ( )

Invert the quaternion.

10.121.3.19 bool gazebo::math::Quaternion::IsFinite ( ) const

See if a quatern is finite (e.g., not nan)

Returns

True if quatern is finite

10.121.3.20 void gazebo::math::Quaternion::Normalize ( )

Normalize the quaternion.

Referenced by gazebo::math::Pose::CoordRotationSub().

10.121.3.21 bool gazebo::math::Quaternion::operator!= (const Quaternion & _qt) const

Not equal to operator.

Parameters

in	*_qt*	**Quaternion** (p. 623) for comparison

Returns

True if not equal

10.121.3.22 Quaternion gazebo::math::Quaternion::operator* (const Quaternion & _q) const [inline]

Multiplication operator.

Parameters

in	*_qt*	**Quaternion** (p. 623) for multiplication

Returns

This quaternion multiplied by the parameter

References Quaternion(), w, x, y, and z.

10.121.3.23 Quaternion gazebo::math::Quaternion::operator∗ (const double & _f) const

Multiplication operator.

Parameters

in	*_f*	factor

Returns

quaternion multiplied by _f

10.121.3.24 Vector3 gazebo::math::Quaternion::operator∗ (const Vector3 & _v) const

Vector3 (p. 855) multiplication operator.

Parameters

in	*_v*	vector to multiply

10.121.3.25 Quaternion gazebo::math::Quaternion::operator∗= (const Quaternion & *qt*)

Multiplication operator.

Parameters

in	*_qt*	**Quaternion** (p. 623) for multiplication

Returns

This quatern multiplied by the parameter

10.121.3.26 Quaternion gazebo::math::Quaternion::operator+ (const Quaternion & _qt) const

Addition operator.

Parameters

in	*_qt*	quaternion for addition

Returns

this quaternion + _qt

10.121.3.27 Quaternion gazebo::math::Quaternion::operator+= (const Quaternion & _qt)

Addition operator.

Parameters

in	*_qt*	quaternion for addition

Returns

this quaternion + qt

10.121.3.28 Quaternion gazebo::math::Quaternion::operator- (const Quaternion & _qt) const

Substraction operator.

Parameters

in	*_qt*	quaternion to substract

Returns

this quaternion - _qt

10.121.3.29 Quaternion gazebo::math::Quaternion::operator- ( ) const

Unary minus operator.

Returns

negates each component of the quaternion

10.121.3.30 Quaternion gazebo::math::Quaternion::operator-= (const Quaternion & _qt)

Substraction operator.

Parameters

in	*_qt*	**Quaternion** (p. 623) for substraction

Returns

This quatern - qt

10.121.3.31 Quaternion& gazebo::math::Quaternion::operator= (const Quaternion & _qt)

Equal operator.

Parameters

in	*_qt*	**Quaternion** (p. 623) to copy

10.121.3.32 bool gazebo::math::Quaternion::operator== (const Quaternion & _qt) const

Equal to operator.

Parameters

in	*_qt*	**Quaternion** (p. 623) for comparison

Returns

True if equal

10.121.3.33 Vector3 gazebo::math::Quaternion::RotateVector (const Vector3 & _vec) const [inline]

Rotate a vector using the quaternion.

Parameters

in	*_vec*	vector to rotate

Returns

the rotated vector

References GetInverse(), gazebo::math::Vector3::x, x, gazebo::math::Vector3::y, y, gazebo::math::Vector3::z, and z.

10.121.3.34 Vector3 gazebo::math::Quaternion::RotateVectorReverse (Vector3 _vec) const

Do the reverse rotation of a vector by this quaternion.

Parameters

in	*_vec*	the vector

Returns

the

10.121.3.35 void gazebo::math::Quaternion::Round (int _precision)

Round all values to _precision decimal places.

Parameters

in	*_precision*	the precision

10.121.3.36 void gazebo::math::Quaternion::Scale (double *_scale*)

Scale a Quaternionion.

Parameters

in	*_scale*	Amount to scale this rotation

10.121.3.37 void gazebo::math::Quaternion::Set (double *_u*, double *_x*, double *_y*, double *_z*)

Set this quaternion from 4 floating numbers.

Parameters

in	*_u*	u
in	*_x*	x
in	*_y*	y
in	*_z*	z

10.121.3.38 void gazebo::math::Quaternion::SetFromAxis (double *_x*, double *_y*, double *_z*, double *_a*)

Set the quaternion from an axis and angle.

Parameters

in	*_x*	X axis
in	*_y*	Y axis
in	*_z*	Z axis
in	*_a*	**Angle** (p. 118) in radians

10.121.3.39 void gazebo::math::Quaternion::SetFromAxis (const Vector3 & *_axis*, double *_a*)

Set the quaternion from an axis and angle.

Parameters

in	*_axis*	Axis
in	*_a*	**Angle** (p. 118) in radians

10.121.3.40 void gazebo::math::Quaternion::SetFromEuler (const Vector3 & *_vec*)

Set the quaternion from Euler angles.

Parameters

in	*vec*	Euler angle

10.121.3.41 void gazebo::math::Quaternion::SetFromEuler (double *_roll*, double *_pitch*, double *_yaw*)

Set the quaternion from Euler angles.

Parameters

in	*_roll*	Roll angle (radians).
in	*_pitch*	Roll angle (radians).
in	*_yaw*	Roll angle (radians).

10.121.3.42 void gazebo::math::Quaternion::SetToIdentity ( )

Set the quatern to the identity.

10.121.3.43 static Quaternion gazebo::math::Quaternion::Slerp (double *_fT*, const Quaternion & *_rkP*, const Quaternion & *_rkQ*, bool *_shortestPath* = `false`) [static]

Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.

Parameters

in	*_ft*	the interpolation parameter
in	*_rkP*	the beginning quaternion
in	*_rkQ*	the end quaternion
in	*_shortestPath*	when true, the rotation may be inverted to get to minimize rotation

10.121.3.44 static Quaternion gazebo::math::Quaternion::Squad (double *_fT*, const Quaternion & *_rkP*, const Quaternion & *_rkA*, const Quaternion & *_rkB*, const Quaternion & *_rkQ*, bool *_shortestPath* = `false`) [static]

Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.

Parameters

in	*_ft*	the interpolation parameter
in	*_rkP*	the beginning quaternion
in	*_rkA*	first intermediate quaternion
in	*_rkB*	second intermediate quaternion
in	*_rkQ*	the end quaternion
in	*_shortestPath*	when true, the rotation may be inverted to get to minimize rotation

10.121.4 Friends And Related Function Documentation

10.121.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Quaternion & *_q*) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_q*	quaternion to output

Returns

the stream

10.121.4.2 std::istream& operator>> (std::istream & *_in,* gazebo::math::Quaternion & *_q*) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_q*	**Quaternion** (p. 623) to read values into

Returns

The istream

10.121.5 Member Data Documentation

10.121.5.1 double gazebo::math::Quaternion::w

Attributes of the quaternion.

Referenced by Correct(), GetInverse(), and operator∗().

10.121.5.2 double gazebo::math::Quaternion::x

Attributes of the quaternion.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), GetInverse(), operator∗(), and RotateVector().

10.121.5.3 double gazebo::math::Quaternion::y

Attributes of the quaternion.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), GetInverse(), operator∗(), and RotateVector().

10.121.5.4 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**

10.122 gazebo::math::Rand Class Reference

Random number generator class.

`#include <gzmath/gzmath.hh>`

Static Public Member Functions

- static double **GetDblNormal** (double _mean=0, double _sigma=1)

  Get a double from a normal distribution.
- static double **GetDblUniform** (double _min=0, double _max=1)

  Get a double from a uniform distribution.
- static int **GetIntNormal** (int _mean, int _sigma)

  Get a double from a normal distribution.
- static int **GetIntUniform** (int _min, int _max)

  Get a integer from a uniform distribution.
- static uint32_t **GetSeed** ()

  Get the seed value.
- static void **SetSeed** (uint32_t _seed)

  Set the seed value.

10.122.1 Detailed Description

Random number generator class.

10.122.2 Member Function Documentation

10.122.2.1 static double gazebo::math::Rand::GetDblNormal (double *_mean* = 0, double *_sigma* = 1) [static]

Get a double from a normal distribution.

Parameters

in	*_mean*	Mean value for the distribution
in	*_sigma*	Sigma value for the distribution

10.122.2.2 static double gazebo::math::Rand::GetDblUniform (double *_min* = 0, double *_max* = 1) [static]

Get a double from a uniform distribution.

Parameters

in	*_min*	Minimum bound for the random number
in	*_max*	Maximum bound for the random number

10.122.2.3 static int gazebo::math::Rand::GetIntNormal (int *_mean*, int *_sigma*) [static]

Get a double from a normal distribution.

Parameters

in	*_mean*	Mean value for the distribution
in	*_sigma*	Sigma value for the distribution

10.122.2.4 static int gazebo::math::Rand::GetIntUniform (int *_min*, int *_max*) [static]

Get a integer from a uniform distribution.

Parameters

in	*_min*	Minimum bound for the random number
in	*_max*	Maximum bound for the random number

10.122.2.5 static uint32_t gazebo::math::Rand::GetSeed ( ) [static]

Get the seed value.

Returns

The seed value used to initialize the random number generator.

10.122.2.6 static void gazebo::math::Rand::SetSeed (uint32_t *_seed*) [static]

Set the seed value.

Parameters

in	*_seed*	The seed used to initialize the randon number generator.

The documentation for this class was generated from the following file:

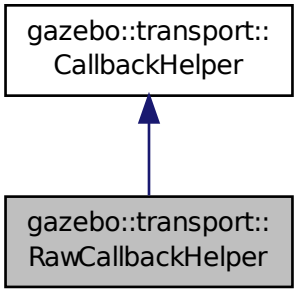
- **Rand.hh**

10.123 gazebo::transport::RawCallbackHelper Class Reference

Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.

`#include <CallbackHelper.hh>`

Inheritance diagram for gazebo::transport::RawCallbackHelper:



Public Member Functions

- **RawCallbackHelper** (const boost::function< void(const std::string &)> &_cb, bool _latching=false)

  Constructor.
- std::string **GetMsgType** () const

  Get the typename of the message that is handled.
- virtual bool **HandleData** (const std::string &_newdata)

  Process new incoming data.
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)

  Process new incoming message.
- virtual bool **IsLocal** () const

  Is the callback local?

Additional Inherited Members

10.123.1 Detailed Description

Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.

Raw means that the data has not been converted into a protobuf message.

10.123.2 Constructor & Destructor Documentation

10.123.2.1 gazebo::transport::RawCallbackHelper::RawCallbackHelper (const boost::function< void(const std::string &)> & *_cb*, bool *_latching* = `false`) [inline]

Constructor.

Parameters

in	*_cb*	boost function to call on incoming messages
in	*_latching*	Set to true to make the callback helper latching.

10.123.3 Member Function Documentation

10.123.3.1 std::string gazebo::transport::RawCallbackHelper::GetMsgType ( ) const [inline], [virtual]

Get the typename of the message that is handled.

Returns

String representation of the message type

Reimplemented from **gazebo::transport::CallbackHelper** (p. 159).

10.123.3.2 virtual bool gazebo::transport::RawCallbackHelper::HandleData (const std::string & *_newdata*) [inline], [virtual]

Process new incoming data.

Parameters

in	*_newdata*	Incoming data to be processed

Returns

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 159).

10.123.3.3 virtual bool gazebo::transport::RawCallbackHelper::HandleMessage (MessagePtr *_newMsg*) [inline], [virtual]

Process new incoming message.

Parameters

in	*_newMsg*	Incoming message to be processed

Returns

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 159).

10.123.3.4 virtual bool gazebo::transport::RawCallbackHelper::IsLocal ( ) const [inline], [virtual]

Is the callback local?

Returns

true if the callback is local, false if the callback is tied to a remote connection

Implements **gazebo::transport::CallbackHelper** (p. 159).

The documentation for this class was generated from the following file:

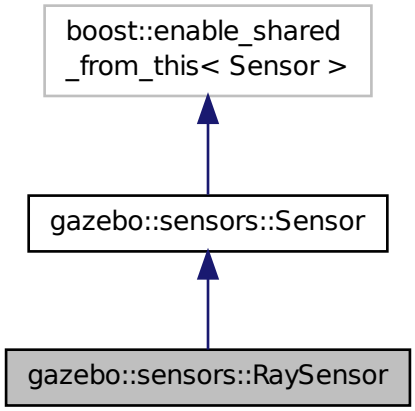
- **CallbackHelper.hh**

10.124 gazebo::sensors::RaySensor Class Reference

Sensor (p. 698) with one or more rays.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::RaySensor:



Public Member Functions

- **RaySensor** ()
  Constructor.
- virtual ~**RaySensor** ()
  Destructor.
- **math::Angle GetAngleMax** () const
  Get the maximum angle.
- **math::Angle GetAngleMin** () const
  Get the minimum angle.
- double **GetAngleResolution** () const
  Get the angle in radians between each range.
- int **GetFiducial** (int _index)
  Get detected fiducial value for a ray.
- **physics::MultiRayShapePtr GetLaserShape** () const
  *Returns a pointer to the internal **physics::MultiRayShape** (p. 527).*
- double **GetRange** (int _index)
  Get detected range for a ray.
- int **GetRangeCount** () const

Get the range count.

- double **GetRangeMax** () const

  Get the maximum range.

- double **GetRangeMin** () const

  Get the minimum range.

- double **GetRangeResolution** () const

  Get the range resolution.

- void **GetRanges** (std::vector< double > &_ranges)

  Get all the ranges.

- int **GetRayCount** () const

  Get the ray count.

- double **GetRetro** (int _index)

  Get detected retro (intensity) value for a ray.

- virtual std::string **GetTopic** () const

  Returns the topic name as set in SDF.

- **math::Angle GetVerticalAngleMax** () const

  Get the vertical scan line top angle.

- **math::Angle GetVerticalAngleMin** () const

  Get the vertical scan bottom angle.

- int **GetVerticalRangeCount** () const

  Get the vertical scan line count.

- int **GetVerticalRayCount** () const

  Get the vertical scan line count.

- virtual void **Init** ()

  Initialize the sensor.

- virtual bool **IsActive** ()

  Returns true if sensor generation is active.

- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the sensor.

- virtual void **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

Additional Inherited Members

10.124.1 Detailed Description

Sensor (p. 698) with one or more rays.

This sensor cast rays into the world, tests for intersections, and reports the range to the nearest object. It is used by ranging sensor models (e.g., sonars and scanning laser range finders).

10.124.2 Constructor & Destructor Documentation

10.124.2.1 gazebo::sensors::RaySensor::RaySensor ( )

Constructor.

10.124.2.2 virtual gazebo::sensors::RaySensor::~RaySensor ( ) [virtual]

Destructor.

10.124.3 Member Function Documentation

10.124.3.1 virtual void gazebo::sensors::RaySensor::Fini ( ) [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 702).

10.124.3.2 math::Angle gazebo::sensors::RaySensor::GetAngleMax ( ) const

Get the maximum angle.

Returns

the maximum angle object

10.124.3.3 math::Angle gazebo::sensors::RaySensor::GetAngleMin ( ) const

Get the minimum angle.

Returns

The minimum angle object

10.124.3.4 double gazebo::sensors::RaySensor::GetAngleResolution ( ) const

Get the angle in radians between each range.

Returns

Resolution of the angle

10.124.3.5 int gazebo::sensors::RaySensor::GetFiducial (int *_index*)

Get detected fiducial value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index value of specific ray

Returns

Fiducial value

10.124.3.6 physics::MultiRayShapePtr gazebo::sensors::RaySensor::GetLaserShape ( ) const [inline]

Returns a pointer to the internal **physics::MultiRayShape** (p. 527).

Returns

Pointer to ray shape

10.124.3.7 double gazebo::sensors::RaySensor::GetRange (int *_index*)

Get detected range for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index of specific ray

Returns

Returns DBL_MAX for no detection.

10.124.3.8 int gazebo::sensors::RaySensor::GetRangeCount ( ) const

Get the range count.

Returns

The number of ranges

10.124.3.9 double gazebo::sensors::RaySensor::GetRangeMax ( ) const

Get the maximum range.

Returns

The maximum range

10.124.3.10 double gazebo::sensors::RaySensor::GetRangeMin ( ) const

Get the minimum range.

Returns

The minimum range

10.124.3.11 double gazebo::sensors::RaySensor::GetRangeResolution ( ) const

Get the range resolution.

Returns

Resolution of the range

10.124.3.12 void gazebo::sensors::RaySensor::GetRanges (std::vector< double > & *_ranges*)

Get all the ranges.

Parameters

_ranges	**A** (p. 111) vector that will contain all the range data

10.124.3.13 int gazebo::sensors::RaySensor::GetRayCount ( ) const

Get the ray count.

Returns

The number of rays

10.124.3.14 double gazebo::sensors::RaySensor::GetRetro (int *_index*)

Get detected retro (intensity) value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index of specific ray

Returns

Retro (intensity) value for ray

10.124.3.15 virtual std::string gazebo::sensors::RaySensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 703).

10.124.3.16 math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMax ( ) const

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.124.3.17 math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMin ( ) const

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.124.3.18 int gazebo::sensors::RaySensor::GetVerticalRangeCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.124.3.19 int gazebo::sensors::RaySensor::GetVerticalRayCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.124.3.20 virtual void gazebo::sensors::RaySensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.124.3.21 virtual bool gazebo::sensors::RaySensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.124.3.22 virtual void gazebo::sensors::RaySensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

10.124.3.23 virtual void gazebo::sensors::RaySensor::UpdateImpl (bool) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 706).

The documentation for this class was generated from the following file:

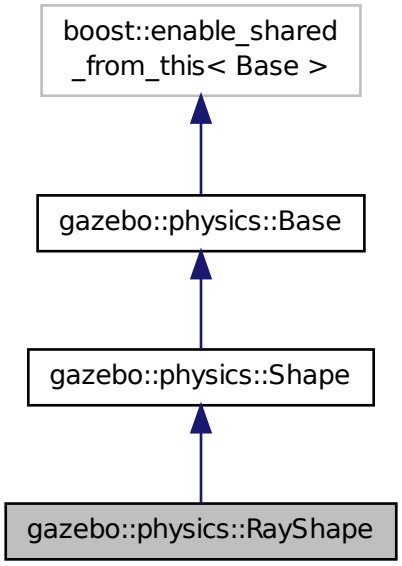
- **RaySensor.hh**

10.125 gazebo::physics::RayShape Class Reference

Base (p. 137) class for Ray collision geometry.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::RayShape:



Public Member Functions

- **RayShape** (**PhysicsEnginePtr** _physicsEngine)

  Constructor for a global ray.
- **RayShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**RayShape** ()

  Destructor.
- void **FillMsg** (msgs::Geometry &_msg)

  Fill a message with data from this object.
- int **GetFiducial** () const

  Get the fiducial id detected by this ray.
- virtual void **GetGlobalPoints** (**math::Vector3** &_posA, **math::Vector3** &_posB)

  Get the global starting and ending points.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)=0

  Get the nearest intersection.
- double **GetLength** () const

  Get the length of the ray.
- virtual void **GetRelativePoints** (**math::Vector3** &_posA, **math::Vector3** &_posB)

  Get the relative starting and ending points.
- float **GetRetro** () const

  Get the retro-reflectivness detected by this ray.

- virtual void **Init** ()

  In the ray.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  Update this shape from a message.
- void **SetFiducial** (int _fid)

  Set the fiducial id detected by this ray.
- virtual void **SetLength** (double _len)

  Set the length of the ray.
- virtual void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)

  Set the ray based on starting and ending points relative to the body.
- void **SetRetro** (float _retro)

  Set the retro-reflectivness detected by this ray.
- virtual void **Update** ()=0

  Update the ray collision.

Protected Attributes

- int **contactFiducial**

  Fiducial ID value.
- double **contactLen**

  Length of the ray.
- double **contactRetro**

  Retro reflectance value.
- **math::Vector3 globalEndPos**

  End position of the ray in global cs.
- **math::Vector3 globalStartPos**

  Start position of the ray in global cs.
- **math::Vector3 relativeEndPos**

  End position of the ray, relative to the body.
- **math::Vector3 relativeStartPos**

  Start position of the ray, relative to the body.

Additional Inherited Members

10.125.1 Detailed Description

Base (p. 137) class for Ray collision geometry.

10.125.2 Constructor & Destructor Documentation

10.125.2.1 gazebo::physics::RayShape::RayShape (**PhysicsEnginePtr** *_physicsEngine*) [explicit]

Constructor for a global ray.

Parameters

in	*_physicsEngine*	Pointer to the physics engine.

10.125.2.2 gazebo::physics::RayShape::RayShape (CollisionPtr _parent) [explicit]

Constructor.

Parameters

in	*_parent*	**Collision** (p. 195) parent of the shape.

10.125.2.3 virtual gazebo::physics::RayShape::~RayShape ( ) [virtual]

Destructor.

10.125.3 Member Function Documentation

10.125.3.1 void gazebo::physics::RayShape::FillMsg (msgs::Geometry & _msg) [virtual]

Fill a message with data from this object.

Parameters

out	*_msg*	Message to fill. Implement this function.

Implements **gazebo::physics::Shape** (p. 722).

10.125.3.2 int gazebo::physics::RayShape::GetFiducial ( ) const

Get the fiducial id detected by this ray.

Returns

Fiducial id detected.

10.125.3.3 virtual void gazebo::physics::RayShape::GetGlobalPoints (math::Vector3 & _posA, math::Vector3 & _posB) [virtual]

Get the global starting and ending points.

Parameters

out	*_posA*	Returns the starting point.
out	*_posB*	Returns the ending point.

10.125.3.4 virtual void gazebo::physics::RayShape::GetIntersection (double & _dist, std::string & _entity) [pure virtual]

Get the nearest intersection.

Parameters

out	*_dist*	Distance to the intersection.
out	*_entity*	Name of the entity the ray intersected with.

10.125.3.5 double gazebo::physics::RayShape::GetLength ( ) const

Get the length of the ray.

Returns

The ray length.

10.125.3.6 virtual void gazebo::physics::RayShape::GetRelativePoints (math::Vector3 & *_posA*, math::Vector3 & *_posB*) [virtual]

Get the relative starting and ending points.

Parameters

in	*_posA*	Returns the starting point.
in	*_posB*	Returns the ending point.

10.125.3.7 float gazebo::physics::RayShape::GetRetro ( ) const

Get the retro-reflectivness detected by this ray.

Returns

Retro reflectance value.

10.125.3.8 virtual void gazebo::physics::RayShape::Init ( ) [virtual]

In the ray.

Implements **gazebo::physics::Shape** (p. 722).

10.125.3.9 virtual void gazebo::physics::RayShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Update this shape from a message.

Parameters

in	*_msg*	Message to update from. Implement this function.

Implements **gazebo::physics::Shape** (p. 722).

10.125.3.10 void gazebo::physics::RayShape::SetFiducial (int *_fid*)

Set the fiducial id detected by this ray.

Parameters

in	*_fid*	Fiducial id detected by this ray.

10.125.3.11 virtual void gazebo::physics::RayShape::SetLength (double *_len*) [virtual]

Set the length of the ray.

Parameters

in	*_len*	Length of the array.

10.125.3.12 virtual void gazebo::physics::RayShape::SetPoints (const math::Vector3 & *_posStart*, const math::Vector3 & *_posEnd*) [virtual]

Set the ray based on starting and ending points relative to the body.

Parameters

in	*_posStart*	Start position, relative the body.
in	*_posEnd*	End position, relative to the body.

10.125.3.13 void gazebo::physics::RayShape::SetRetro (float *_retro*)

Set the retro-reflectivness detected by this ray.

Parameters

in	*_retro*	Retro reflectance value.

10.125.3.14 virtual void gazebo::physics::RayShape::Update ( ) [pure virtual]

Update the ray collision.

Reimplemented from **gazebo::physics::Base** (p. 148).

10.125.4 Member Data Documentation

10.125.4.1 int gazebo::physics::RayShape::contactFiducial [protected]

Fiducial ID value.

10.125.4.2 double gazebo::physics::RayShape::contactLen [protected]

Length of the ray.

10.125.4.3 double gazebo::physics::RayShape::contactRetro [protected]

Retro reflectance value.

10.125.4.4 math::Vector3 gazebo::physics::RayShape::globalEndPos [protected]

End position of the ray in global cs.

10.125.4.5 math::Vector3 gazebo::physics::RayShape::globalStartPos [protected]

Start position of the ray in global cs.

10.125.4.6 math::Vector3 gazebo::physics::RayShape::relativeEndPos [protected]

End position of the ray, relative to the body.

10.125.4.7 math::Vector3 gazebo::physics::RayShape::relativeStartPos [protected]

Start position of the ray, relative to the body.

The documentation for this class was generated from the following file:

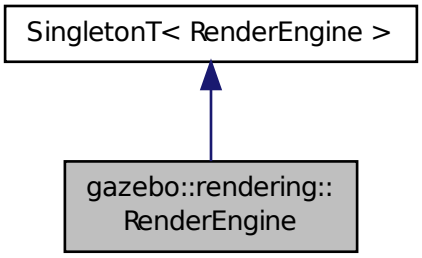
- **RayShape.hh**

10.126 gazebo::rendering::RenderEngine Class Reference

Adaptor to Ogre3d.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RenderEngine:



Public Types

- enum **RenderPathType** {
  NONE = 0, **VERTEX** = 1, **FORWARD** = 2, **DEFERRED** = 3,
  RENDER_PATH_COUNT }

  The type of rendering path used by the rendering engine.

Public Member Functions

- void **AddResourcePath** (const std::string &_uri)

  *Add a new path for **Ogre** (p. 106) to search for resources.*
- **ScenePtr CreateScene** (const std::string &_name, bool _enableVisualizations)

  Create a scene.
- void **Fini** ()

  Tears down the rendering engine.
- **RenderPathType GetRenderPathType** () const

  Get the type of rendering path to use.
- **ScenePtr GetScene** (const std::string &_name)

  Get a scene by name.
- **ScenePtr GetScene** (unsigned int _index)

  Get a scene by index.
- unsigned int **GetSceneCount** () const

  Get the number of scenes.
- void **Init** ()

  *Initialize **Ogre** (p. 106). Load must happen before Init.*
- void **Load** ()

  *Load the parameters for **Ogre** (p. 106). Load must happen before Init.*
- void **RemoveScene** (const std::string &_name)

  Remove a scene.

Public Attributes

- Ogre::Root ∗ **root**

  Pointer to the root scene node.

Protected Attributes

- void ∗ **dummyContext**

  GLX context used to render the scenes.Used for gui-less operation.
- void ∗ **dummyDisplay**

  Pointer to the dummy display.Used for gui-less operation.
- uint64_t **dummyWindowId**

  ID for a dummy window. Used for gui-less operation.

Additional Inherited Members

10.126.1 Detailed Description

Adaptor to Ogre3d.

Provides the interface to load, initialize the rendering engine.

10.126.2 Member Enumeration Documentation

10.126.2.1 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.

10.126.3 Member Function Documentation

10.126.3.1 void gazebo::rendering::RenderEngine::AddResourcePath (const std::string & *_uri*)

Add a new path for **Ogre** (p. 106) to search for resources.

Parameters

in	*_uri*	URI of the path. The uri should be of the form `file://` or model://

10.126.3.2 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.

10.126.3.3 void gazebo::rendering::RenderEngine::Fini ( )

Tears down the rendering engine.

10.126.3.4 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

10.126.3.5 ScenePtr gazebo::rendering::RenderEngine::GetScene (const std::string & *_name*)

Get a scene by name.

Parameters

in	*_name*	Name of the scene to retreive.

Returns

A (p. 111) pointer to the **Scene** (p. 676), or NULL if the scene doesn't exist.

10.126.3.6 ScenePtr gazebo::rendering::RenderEngine::GetScene (unsigned int *_index*)

Get a scene by index.

The index should be between 0 and **GetSceneCount()** (p. 657).

Parameters

in	*_index*	The index of the scene.

Returns

A (p. 111) pointer to a **Scene** (p. 676), or NULL if the index was invalid.

10.126.3.7 unsigned int gazebo::rendering::RenderEngine::GetSceneCount ( ) const

Get the number of scenes.

Returns

The number of scenes created by the **RenderEngine** (p. 653).

10.126.3.8 void gazebo::rendering::RenderEngine::Init ( )

Initialize **Ogre** (p. 106). Load must happen before Init.

10.126.3.9 void gazebo::rendering::RenderEngine::Load ( )

Load the parameters for **Ogre** (p. 106). Load must happen before Init.

10.126.3.10 void gazebo::rendering::RenderEngine::RemoveScene (const std::string & *_name*)

Remove a scene.

Parameters

in	*_name*	The name of the scene to remove.

10.126.4 Member Data Documentation

10.126.4.1 void* gazebo::rendering::RenderEngine::dummyContext [protected]

GLX context used to render the scenes.Used for gui-less operation.

10.126.4.2 void* gazebo::rendering::RenderEngine::dummyDisplay [protected]

Pointer to the dummy display.Used for gui-less operation.

10.126.4.3 uint64_t gazebo::rendering::RenderEngine::dummyWindowId [protected]

ID for a dummy window. Used for gui-less operation.

10.126.4.4 Ogre::Root* gazebo::rendering::RenderEngine::root

Pointer to the root scene node.

The documentation for this class was generated from the following file:

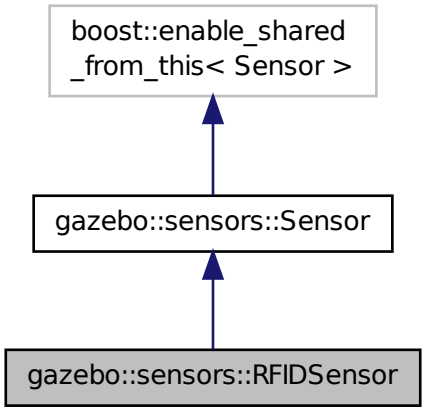
- **RenderEngine.hh**

10.127 gazebo::sensors::RFIDSensor Class Reference

Sensor (p. 698) class for RFID type of sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::RFIDSensor:



Public Member Functions

- **RFIDSensor** ()

  Constructor.
- virtual ~**RFIDSensor** ()

  Destructor.
- void **AddTag** (**RFIDTag** *_tag)
- virtual void **Fini** ()

  Finalize the sensor.
- virtual void **Init** ()

  Initialize the sensor.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** _sdf)

  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

Additional Inherited Members

10.127.1 Detailed Description

Sensor (p. 698) class for RFID type of sensor.

10.127.2 Constructor & Destructor Documentation

10.127.2.1 gazebo::sensors::RFIDSensor::RFIDSensor ( )

Constructor.

10.127.2.2 virtual gazebo::sensors::RFIDSensor::~RFIDSensor ( ) [virtual]

Destructor.

10.127.3 Member Function Documentation

10.127.3.1 void gazebo::sensors::RFIDSensor::AddTag (RFIDTag * *_tag*)

10.127.3.2 virtual void gazebo::sensors::RFIDSensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 702).

10.127.3.3 virtual void gazebo::sensors::RFIDSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.127.3.4 virtual void gazebo::sensors::RFIDSensor::Load (const std::string & *_worldName*, sdf::ElementPtr *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 698) parameters.
in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.127.3.5 virtual void gazebo::sensors::RFIDSensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

10.127.3.6 virtual void gazebo::sensors::RFIDSensor::UpdateImpl (bool) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 706).

The documentation for this class was generated from the following file:

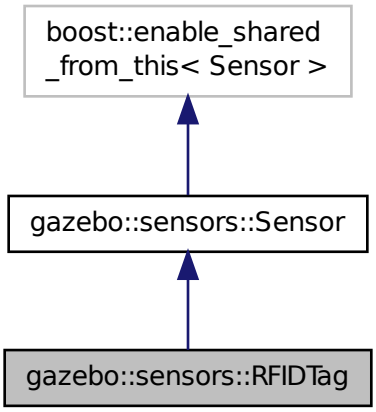
- **RFIDSensor.hh**

10.128 gazebo::sensors::RFIDTag Class Reference

RFIDTag (p. 660) to interact with RFIDTagSensors.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::RFIDTag:



Public Member Functions

- **RFIDTag** ()

  Constructor.
- virtual ∼**RFIDTag** ()

  Destructor.
- virtual void **Fini** ()

  Finalize the sensor.
- **math::Pose GetTagPose** () const

  Returns pose of tag in world coordinate.
- virtual void **Init** ()

  Initialize the sensor.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** &_sdf)
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

Additional Inherited Members

10.128.1 Detailed Description

RFIDTag (p. 660) to interact with RFIDTagSensors.

10.128.2 Constructor & Destructor Documentation

10.128.2.1 gazebo::sensors::RFIDTag::RFIDTag ( )

Constructor.

10.128.2.2 virtual gazebo::sensors::RFIDTag::∼RFIDTag ( ) [virtual]

Destructor.

10.128.3 Member Function Documentation

10.128.3.1 virtual void gazebo::sensors::RFIDTag::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 702).

10.128.3.2 math::Pose gazebo::sensors::RFIDTag::GetTagPose ( ) const [inline]

Returns pose of tag in world coordinate.

Returns

Pose of object.

References gazebo::physics::Entity::GetWorldPose().

10.128.3.3 virtual void gazebo::sensors::RFIDTag::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 704).

10.128.3.4 virtual void gazebo::sensors::RFIDTag::Load (const std::string & *_worldName*, sdf::ElementPtr & *_sdf*) [virtual]

10.128.3.5 virtual void gazebo::sensors::RFIDTag::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 705).

10.128.3.6 virtual void gazebo::sensors::RFIDTag::UpdateImpl (bool) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 706).

The documentation for this class was generated from the following file:

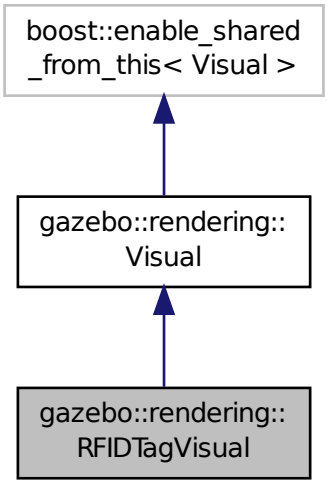
- **RFIDTag.hh**

10.129 gazebo::rendering::RFIDTagVisual Class Reference

Visualization for RFID tags sensor.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RFIDTagVisual:



Public Member Functions

- **RFIDTagVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

  Constructor.
- virtual ~**RFIDTagVisual** ()

  Destructor.

Additional Inherited Members

10.129.1 Detailed Description

Visualization for RFID tags sensor.

10.129.2 Constructor & Destructor Documentation

10.129.2.1 gazebo::rendering::RFIDTagVisual::RFIDTagVisual (const std::string & *_name*, VisualPtr *_vis*, const std::string & *_topicName*)

Constructor.

Parameters

in	*_name*	Name of the visual.
in	*_vis*	Parent visual.
in	*_topicName*	Name of the topic that publishes RFID data.

See Also

sensors::RFIDSensor (p. 658)

10.129.2.2 virtual gazebo::rendering::RFIDTagVisual::~RFIDTagVisual ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

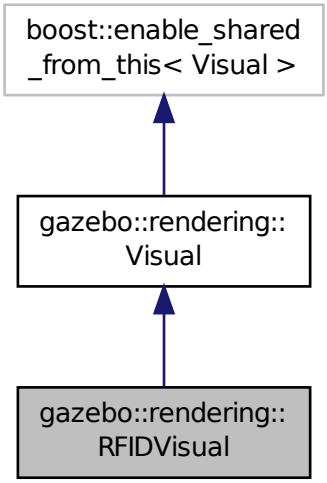
- **RFIDTagVisual.hh**

10.130 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

10.130.1 Detailed Description

Visualization for RFID sensor.

10.130.2 Constructor & Destructor Documentation

10.130.2.1 gazebo::rendering::RFIDVisual::RFIDVisual (const std::string & *_name,* VisualPtr *_vis,* const std::string & *_topicName*)

Constructor.

Parameters

in	*_name*	Name of the **Visual** (p. 885).
in	*_vis*	Parent **Visual** (p. 885).
in	*_topicName*	Name of the topic which publishes RFID data.

10.130.2.2 virtual gazebo::rendering::RFIDVisual::~RFIDVisual ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

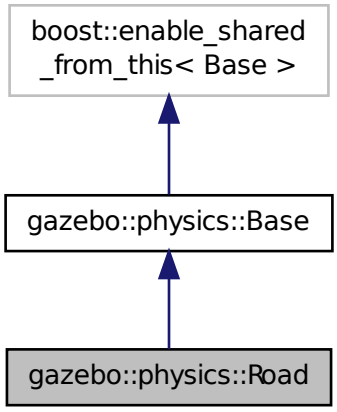
- **RFIDVisual.hh**

10.131 gazebo::physics::Road Class Reference

for building a **Road** (p. 665) from SDF

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Road:



Public Member Functions

- **Road** (**BasePtr** _parent)
  Constructor.
- virtual ~**Road** ()
  Destructor.
- virtual void **Init** ()
  Initialize the road.
- void **Load** (**sdf::ElementPtr** _sdf)
  Load the road from SDF.

Additional Inherited Members

10.131.1 Detailed Description

for building a **Road** (p. 665) from SDF

10.131.2 Constructor & Destructor Documentation

10.131.2.1 gazebo::physics::Road::Road (**BasePtr** *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of this road object.

10.131.2.2 virtual gazebo::physics::Road::~Road ( ) [virtual]

Destructor.

10.131.3 Member Function Documentation

10.131.3.1 virtual void gazebo::physics::Road::Init ( ) [virtual]

Initialize the road.

Reimplemented from **gazebo::physics::Base** (p. 145).

10.131.3.2 void gazebo::physics::Road::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the road from SDF.

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::Base** (p. 145).

The documentation for this class was generated from the following file:

- **Road.hh**

10.132 Road Class Reference

Used to render a strip of road.

`#include <rendering/rendering.hh>`

10.132.1 Detailed Description

Used to render a strip of road.

The documentation for this class was generated from the following file:

- **Road2d.hh**

10.133 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.

10.133.1 Constructor & Destructor Documentation

10.133.1.1 gazebo::rendering::Road2d::Road2d ( )

Constructor.

10.133.1.2 virtual gazebo::rendering::Road2d::~Road2d ( ) [virtual]

Destructor.

10.133.2 Member Function Documentation

10.133.2.1 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**

10.134 gazebo::math::RotationSpline Class Reference

Spline (p. 754) for rotations.

`#include <math/gzmath.hh>`

Public Member Functions

- **RotationSpline** ()

  Constructor. Sets the autoCalc to true.

- ~**RotationSpline** ()

  Destructor. Nothing is done.

- void **AddPoint** (const **Quaternion** &_p)

  Adds a control point to the end of the spline.

- void **Clear** ()

  Clears all the points in the spline.

- unsigned int **GetNumPoints** () const

  Gets the number of control points in the spline.

- const **Quaternion** & **GetPoint** (unsigned int _index) const

Gets the detail of one of the control points of the spline.

- **Quaternion Interpolate** (double _t, bool _useShortestPath=true)

  Returns an interpolated point based on a parametric value over the whole series.
- **Quaternion Interpolate** (unsigned int _fromIndex, double _t, bool _useShortestPath=true)

  Interpolates a single segment of the spline given a parametric value.
- void **RecalcTangents** ()

  Recalculates the tangents associated with this spline.
- void **SetAutoCalculate** (bool _autoCalc)

  Tells the spline whether it should automatically calculate tangents on demand as points are added.
- void **UpdatePoint** (unsigned int _index, const **Quaternion** &_value)

  Updates a single point in the spline.

Protected Attributes

- bool **autoCalc**

  Automatic recalcultation of tangeants when control points are updated.
- std::vector< **Quaternion** > **points**

  the control points
- std::vector< **Quaternion** > **tangents**

  the tangents

10.134.1 Detailed Description

Spline (p. 754) for rotations.

10.134.2 Constructor & Destructor Documentation

10.134.2.1 gazebo::math::RotationSpline::RotationSpline ( )

Constructor. Sets the autoCalc to true.

10.134.2.2 gazebo::math::RotationSpline::~RotationSpline ( )

Destructor. Nothing is done.

10.134.3 Member Function Documentation

10.134.3.1 void gazebo::math::RotationSpline::AddPoint (const Quaternion & _p)

Adds a control point to the end of the spline.

Parameters

in	*_p*	control point

10.134.3.2 void gazebo::math::RotationSpline::Clear ( )

Clears all the points in the spline.

10.134.3.3 unsigned int gazebo::math::RotationSpline::GetNumPoints ( ) const

Gets the number of control points in the spline.

Returns

the count

10.134.3.4 const Quaternion& gazebo::math::RotationSpline::GetPoint (unsigned int *_index*) const

Gets the detail of one of the control points of the spline.

Parameters

in	*_index*	the index of the control point.

Remarks

This point must already exist in the spline.

Returns

a quaternion (out of bound index result in assertion)

10.134.3.5 Quaternion gazebo::math::RotationSpline::Interpolate (double *_t*, bool *_useShortestPath* = `true`)

Returns an interpolated point based on a parametric value over the whole series.

Remarks

Given a t value between 0 and 1 representing the parametric distance along the whole length of the spline, this method returns an interpolated point.

Parameters

in	*_t*	Parametric value.
in	*_useShortest-Path*	Defines if rotation should take the shortest possible path

Returns

the rotation

10.134.3.6 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	*_fromIndex*	The point index to treat as t = 0. _fromIndex + 1 is deemed to be t = 1
in	*_t*	Parametric value
in	*_useShortest-Path*	Defines if rotation should take the shortest possible path

Returns

the rotation

10.134.3.7 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.

10.134.3.8 void gazebo::math::RotationSpline::SetAutoCalculate (bool *_autoCalc*)

Tells the spline whether it should automatically calculate tangents on demand as points are added.

Remarks

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the recalcTangents method when you are done.

Parameters

in	*_autoCalc*	If true, tangents are calculated for you whenever a point changes. If false, you must call reclacTangents to recalculate them when it best suits.

10.134.3.9 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	*_index*	index
in	*_value*	the new control point value

10.134.4 Member Data Documentation

10.134.4.1 bool gazebo::math::RotationSpline::autoCalc [protected]

Automatic recalcultation of tangeants when control points are updated.

10.134.4.2 std::vector<Quaternion> gazebo::math::RotationSpline::points [protected]

the control points

10.134.4.3 std::vector<Quaternion> gazebo::math::RotationSpline::tangents [protected]

the tangents

The documentation for this class was generated from the following file:

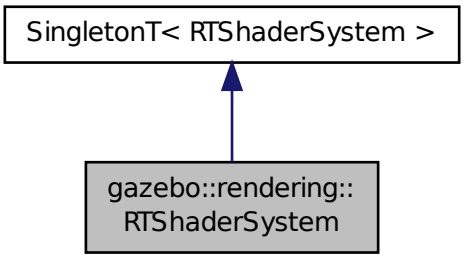
- **RotationSpline.hh**

10.135 gazebo::rendering::RTShaderSystem Class Reference

Implements **Ogre** (p. 106)'s Run-Time Shader system.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RTShaderSystem:



Public Types

- enum **LightingModel** { **SSLM_PerVertexLighting**, **SSLM_PerPixelLighting**, **SSLM_NormalMapLighting-TangentSpace**, **SSLM_NormalMapLightingObjectSpace** }

Public Member Functions

- void **AddScene** (**ScenePtr** _scene)
  Add a scene manager.
- void **ApplyShadows** (**ScenePtr** _scene)
  Apply shadows to a scene.
- void **AttachEntity** (**Visual** ∗vis)
  Set an Ogre::Entity to use RT shaders.
- void **Clear** ()
  Clear the shader system.
- void **DetachEntity** (**Visual** ∗_vis)
  Remove and entity.
- void **Fini** ()
  Finalize the shader system.
- void **GenerateShaders** (**Visual** ∗_vis)
  Generate shaders for an entity.
- Ogre::PSSMShadowCameraSetup ∗ **GetPSSMShadowCameraSetup** () const
  *Get the **Ogre** (p. 106) PSSM Shadows camera setup.*
- void **Init** ()
  Init the run time shader system.
- void **RemoveScene** (**ScenePtr** _scene)
  Remove a scene.
- void **RemoveShadows** (**ScenePtr** _scene)
  Remove shadows from a scene.
- void **SetPerPixelLighting** (bool _set)
  Set the lighting model to per pixel or per vertex.
- void **UpdateShaders** ()
  Update the shaders. This should not be called frequently.

Static Public Member Functions

- static void **AttachViewport** (Ogre::Viewport ∗_viewport, **ScenePtr** _scene)
  Set a viewport to use shaders.
- static void **DetachViewport** (Ogre::Viewport ∗_viewport, **ScenePtr** _scene)
  Set a viewport to not use shaders.

Additional Inherited Members

10.135.1 Detailed Description

Implements **Ogre** (p. 106)'s Run-Time Shader system.

This class allows Gazebo to generate per-pixel shaders for every material at run-time.

10.135.2 Member Enumeration Documentation

10.135.2.1 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.

10.135.3 Member Function Documentation

10.135.3.1 void gazebo::rendering::RTShaderSystem::AddScene (ScenePtr *_scene*)

Add a scene manager.

Parameters

in	*_scene*	The scene to process

10.135.3.2 void gazebo::rendering::RTShaderSystem::ApplyShadows (ScenePtr *_scene*)

Apply shadows to a scene.

Parameters

in	*_scene*	The scene to receive shadows.

10.135.3.3 void gazebo::rendering::RTShaderSystem::AttachEntity (Visual * *vis*)

Set an Ogre::Entity to use RT shaders.

Parameters

in	*_vis*	**Visual** (p. 885) that will use the **RTShaderSystem** (p. 672).

10.135.3.4 static void gazebo::rendering::RTShaderSystem::AttachViewport (Ogre::Viewport * *_viewport,* ScenePtr *_scene*) [static]

Set a viewport to use shaders.

Parameters

in	*_viewport*	The viewport to add.
in	*_scene*	The scene that the viewport uses.

10.135.3.5 void gazebo::rendering::RTShaderSystem::Clear ( )

Clear the shader system.

10.135.3.6 void gazebo::rendering::RTShaderSystem::DetachEntity (Visual * _vis)

Remove and entity.

Parameters

in	*_vis*	Remove this visual.

10.135.3.7 static void gazebo::rendering::RTShaderSystem::DetachViewport (Ogre::Viewport * _viewport, ScenePtr _scene) [static]

Set a viewport to not use shaders.

Parameters

in	*_viewport*	The viewport to remove.
in	*_scene*	The scene that the viewport uses.

10.135.3.8 void gazebo::rendering::RTShaderSystem::Fini ( )

Finalize the shader system.

10.135.3.9 void gazebo::rendering::RTShaderSystem::GenerateShaders (Visual * _vis)

Generate shaders for an entity.

Parameters

in	*_vis*	The visual to generate shaders for.

10.135.3.10 Ogre::PSSMShadowCameraSetup* gazebo::rendering::RTShaderSystem::GetPSSMShadowCameraSetup ( ) const

Get the **Ogre** (p. 106) PSSM Shadows camera setup.

Returns

The **Ogre** (p. 106) PSSM Shadows camera setup.

10.135.3.11 void gazebo::rendering::RTShaderSystem::Init ( )

Init the run time shader system.

10.135.3.12 void gazebo::rendering::RTShaderSystem::RemoveScene (ScenePtr _scene)

Remove a scene.

Parameters

in	*The*	scene to remove

10.135.3.13 void gazebo::rendering::RTShaderSystem::RemoveShadows (ScenePtr _scene)

Remove shadows from a scene.

Parameters

in	*_scene*	The scene to remove shadows from.

10.135.3.14 void gazebo::rendering::RTShaderSystem::SetPerPixelLighting (bool _set)

Set the lighting model to per pixel or per vertex.

Parameters

in	*_set*	True means to use per-pixel shaders.

10.135.3.15 void gazebo::rendering::RTShaderSystem::UpdateShaders ( )

Update the shaders. This should not be called frequently.

The documentation for this class was generated from the following file:

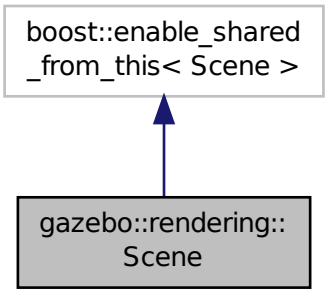
- **RTShaderSystem.hh**

10.136 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. 676).*
- **VisualPtr CloneVisual** (const std::string &_visualName, const std::string &_newName)

  Clone a visual.
- **CameraPtr CreateCamera** (const std::string &_name, bool _autoRender=true)

  Create a camera.
- **DepthCameraPtr CreateDepthCamera** (const std::string &_name, bool _autoRender=true)

  Create depth camera.
- void **CreateGrid** (uint32_t _cellCount, float _cellLength, float _lineWidth, const **common::Color** &_color)

  Create a square grid of cells.
- **UserCameraPtr CreateUserCamera** (const std::string &_name)

  Create a user camera.
- void **DrawLine** (const **math::Vector3** &_start, const **math::Vector3** &_end, const std::string &_name)

  Draw a named line.
- **common::Color GetAmbientColor** () const

  Get the ambient color.
- **common::Color GetBackgroundColor** () const

  Get the background color.
- **CameraPtr GetCamera** (uint32_t _index) const

  Get a camera based on an index.
- **CameraPtr GetCamera** (const std::string &_name) const

Get a camera by name.

- uint32_t **GetCameraCount** () const

  Create laser that generates data from rendering.

- bool **GetFirstContact** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos, **math::Vector3** &_position)

  Get the world pos of a the first contact at a pixel location.

- **Grid** ∗ **GetGrid** (uint32_t _index) const

  Get a grid based on an index.

- uint32_t **GetGridCount** () const

  Get the number of grids.

- double **GetHeightBelowPoint** (const **math::Vector3** &_pt)

  Get the Z-value of the first object below the given point.

- **Heightmap** ∗ **GetHeightmap** () const

  Get a pointer to the heightmap.

- uint32_t **GetId** () const

  Get the scene ID.

- std::string **GetIdString** () const

  Get the scene Id as a string.

- bool **GetInitialized** () const

  *Return true if the **Scene** (p. 676) has been initialized.*

- **LightPtr GetLight** (const std::string &_name) const

  Get a light by name.

- **LightPtr GetLight** (uint32_t _index) const

  Get a light based on an index.

- uint32_t **GetLightCount** () const

  Get the count of the lights.

- Ogre::SceneManager ∗ **GetManager** () const

  Get the OGRE scene manager.

- **VisualPtr GetModelVisualAt** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos)

  Get a model's visual at a mouse position.

- std::string **GetName** () const

  Get the name of the scene.

- **VisualPtr GetSelectedVisual** () const

  Get the currently selected visual.

- bool **GetShadowsEnabled** () const

  Get whether shadows are on or off.

- **UserCameraPtr GetUserCamera** (uint32_t _index) const

  Get a user camera by index.

- uint32_t **GetUserCameraCount** () const

  Get the number of user cameras in this scene.

- **VisualPtr GetVisual** (const std::string &_name) const

  Get a visual by name.

- **VisualPtr GetVisualAt** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos, std::string &_mod)

  Get an entity at a pixel location using a camera.

- **VisualPtr GetVisualAt** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos)

  Get a visual at a mouse position.

- **VisualPtr GetVisualBelow** (const std::string &_visualName)

  Get the closest visual below a given visual.

- void **GetVisualsBelowPoint** (const **math::Vector3** &_pt, std::vector< **VisualPtr** > &_visuals)

  Get a visual directly below a point.
- **VisualPtr GetWorldVisual** () const

  Get the top level world visual.
- void **Init** ()

  *Init **rendering::Scene** (p. 676).*
- void **Load** (**sdf::ElementPtr** _scene)

  Load the scene from a set of parameters.
- void **Load** ()

  Load the scene with default parameters.
- void **PreRender** ()

  Process all received messages.
- void **PrintSceneGraph** ()

  Print the scene graph to std_out.
- void **RemoveVisual** (**VisualPtr** _vis)

  Remove a visual from the scene.
- void **SelectVisual** (const std::string &_name, const std::string &_mode)

  Select a visual by name.
- void **SetAmbientColor** (const **common::Color** &_color)

  Set the ambient color.
- void **SetBackgroundColor** (const **common::Color** &_color)

  Set the background color.
- void **SetFog** (const std::string &_type, const **common::Color** &_color, double _density, double _start, double _end)

  Set the fog parameters.
- void **SetGrid** (bool _enabled)

  Set the grid on or off.
- void **SetShadowsEnabled** (bool _value)

  Set whether shadows are on or off.
- void **SetTransparent** (bool _show)

  Enable or disable transparency for all visuals.
- void **SetVisible** (const std::string &_name, bool _visible)

  Hide or show a visual.
- void **SetWireframe** (bool _show)

  Enable or disable wireframe for all visuals.
- void **ShowCollisions** (bool _show)

  Enable or disable collision visualization.
- void **ShowCOMs** (bool _show)

  Enable or disable center of mass visualization.
- void **ShowContacts** (bool _show)

  Enable or disable contact visualization.
- void **ShowJoints** (bool _show)

  Enable or disable joint visualization.
- void **SnapVisualToNearestBelow** (const std::string &_visualName)

  Move the visual to be ontop of the nearest visual below it.
- std::string **StripSceneName** (const std::string &_name) const

  Remove the name of scene from a string.

Public Attributes

- SkyX::SkyX * **skyx**

  Pointer to the sky.

10.136.1 Detailed Description

Representation of an entire scene graph.

Maintains all the Visuals, Lights, and Cameras for a World.

10.136.2 Constructor & Destructor Documentation

10.136.2.1 gazebo::rendering::Scene::Scene (const std::string & *_name,* bool *_enableVisualizations* = `false`)

Constructor.

Parameters

in	*_name*	Name of the scene.
in	*_enableVisualizations*	True to enable visualizations, this should be set to true for user interfaces, and false for sensor generation.

10.136.2.2 virtual gazebo::rendering::Scene::~Scene ( ) [virtual]

Destructor.

10.136.3 Member Function Documentation

10.136.3.1 void gazebo::rendering::Scene::AddVisual (VisualPtr *_vis*)

Add a visual to the scene.

Parameters

in	*_vis*	**Visual** (p. 885) to add.

10.136.3.2 void gazebo::rendering::Scene::Clear ( )

Clear **rendering::Scene** (p. 676).

10.136.3.3 VisualPtr gazebo::rendering::Scene::CloneVisual (const std::string & *_visualName,* const std::string & *_newName*)

Clone a visual.

Parameters

in	*_visualName*	Name of the visual to clone.
in	*_newName*	New name of the visual.

Returns

Pointer to the cloned visual.

10.136.3.4 CameraPtr gazebo::rendering::Scene::CreateCamera (const std::string & *_name,* bool *_autoRender* = `true`)

Create a camera.

Parameters

in	*_name*	Name of the new camera.
in	*_autoRender*	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.136.3.5 DepthCameraPtr gazebo::rendering::Scene::CreateDepthCamera (const std::string & *_name,* bool *_autoRender* = `true`)

Create depth camera.

Parameters

in	*_name*	Name of the new camera.
in	*_autoRender*	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.136.3.6 void gazebo::rendering::Scene::CreateGrid (uint32_t *_cellCount,* float *_cellLength,* float *_lineWidth,* const common::Color & *_color*)

Create a square grid of cells.

Parameters

in	*_cellCount*	Number of grid cells in one direction.
in	*_cellLength*	Length of one grid cell.
in	*_lineWidth*	Width of the grid lines.
in	*_color*	Color of the grid lines.

10.136.3.7 UserCameraPtr gazebo::rendering::Scene::CreateUserCamera (const std::string & *_name*)

Create a user camera.

A (p. 111) user camera is one design for use with a GUI.

Parameters

in	*_name*	Name of the **UserCamera** (p. 830).

Returns

A (p. 111) pointer to the new **UserCamera** (p. 830).

10.136.3.8 void gazebo::rendering::Scene::DrawLine (const math::Vector3 & *_start,* const math::Vector3 & *_end,* const std::string & *_name*)

Draw a named line.

Parameters

in	*_start*	Start position of the line.
in	*_end*	End position of the line.
in	*_name*	Name of the line.

10.136.3.9 common::Color gazebo::rendering::Scene::GetAmbientColor ( ) const

Get the ambient color.

Returns

The scene's ambient color.

10.136.3.10 common::Color gazebo::rendering::Scene::GetBackgroundColor ( ) const

Get the background color.

Returns

The background color.

10.136.3.11 CameraPtr gazebo::rendering::Scene::GetCamera (uint32_t *_index*) const

Get a camera based on an index.

Index must be between 0 and **Scene::GetCameraCount** (p. 683).

Parameters

in	*_index*	Index of the camera to get.

Returns

Pointer to the camera. Or NULL if the index is invalid.

10.136.3.12 CameraPtr gazebo::rendering::Scene::GetCamera (const std::string & _name) const

Get a camera by name.

Parameters

in	*_name*	Name of the camera.

Returns

Pointer to the camera. Or NULL if the name is invalid.

10.136.3.13 uint32_t gazebo::rendering::Scene::GetCameraCount ( ) const

Create laser that generates data from rendering.

Parameters

in	*_name*	Name of the new laser.
in	*_autoRender*	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new laser. Get the number of cameras in this scene
Number of lasers.

10.136.3.14 bool gazebo::rendering::Scene::GetFirstContact (CameraPtr _camera, const math::Vector2i & _mousePos, math::Vector3 & _position)

Get the world pos of a the first contact at a pixel location.

Parameters

in	*_camera*	Pointer to the camera.
in	*_mousePos*	2D position of the mouse in pixels.
out	*_position*	3D position of the first contact point.

Returns

True if a valid object was hit by the raycast.

10.136.3.15 Grid* gazebo::rendering::Scene::GetGrid (uint32_t _index) const

Get a grid based on an index.

Index must be between 0 and **Scene::GetGridCount** (p. 684).

Parameters

in	*_index*	Index of the grid.

10.136.3.16 uint32_t gazebo::rendering::Scene::GetGridCount ( ) const

Get the number of grids.

Returns

The number of grids.

10.136.3.17 double gazebo::rendering::Scene::GetHeightBelowPoint (const math::Vector3 & _pt)

Get the Z-value of the first object below the given point.

Parameters

in	*_pt*	Position to search below for a visual.

Returns

The Z-value of the nearest visual below the point. Zero is returned if no visual is found.

10.136.3.18 Heightmap* gazebo::rendering::Scene::GetHeightmap ( ) const

Get a pointer to the heightmap.

Returns

Pointer to the heightmap, NULL if no heightmap.

10.136.3.19 uint32_t gazebo::rendering::Scene::GetId ( ) const

Get the scene ID.

Returns

The ID of the scene.

10.136.3.20 std::string gazebo::rendering::Scene::GetIdString ( ) const

Get the scene Id as a string.

Returns

The ID as a string.

10.136.3.21 bool gazebo::rendering::Scene::GetInitialized ( ) const

Return true if the **Scene** (p. 676) has been initialized.

10.136.3.22 LightPtr gazebo::rendering::Scene::GetLight (const std::string & _name) const

Get a light by name.

Parameters

in	*_name*	Name of the light to get.

Returns

Pointer to the light, or NULL if the light was not found.

10.136.3.23 LightPtr gazebo::rendering::Scene::GetLight (uint32_t _index) const

Get a light based on an index.

The index must be between 0 and **Scene::GetLightCount** (p. 685).

Parameters

in	*_index*	Index of the light.

Returns

Pointer to the **Light** (p. 412) or NULL if index was invalid.

10.136.3.24 uint32_t gazebo::rendering::Scene::GetLightCount ( ) const

Get the count of the lights.

Returns

The number of lights.

10.136.3.25 Ogre::SceneManager* gazebo::rendering::Scene::GetManager ( ) const

Get the OGRE scene manager.

Returns

Pointer to the **Ogre** (p. 106) SceneManager.

10.136.3.26 VisualPtr gazebo::rendering::Scene::GetModelVisualAt (CameraPtr _camera, const math::Vector2i & _mousePos)

Get a model's visual at a mouse position.

Parameters

in	*_camera*	Pointer to the camera used to project the mouse position.
in	*_mousePos*	The 2d position of the mouse in pixels.

Returns

Pointer to the visual, NULL if none found.

10.136.3.27 std::string gazebo::rendering::Scene::GetName ( ) const

Get the name of the scene.

Returns

Name of the scene.

10.136.3.28 VisualPtr gazebo::rendering::Scene::GetSelectedVisual ( ) const

Get the currently selected visual.

Returns

Pointer to the currently selected visual, or NULL if nothing is selected.

10.136.3.29 bool gazebo::rendering::Scene::GetShadowsEnabled ( ) const

Get whether shadows are on or off.

Returns

True if shadows are enabled.

10.136.3.30 UserCameraPtr gazebo::rendering::Scene::GetUserCamera (uint32_t *_index*) const

Get a user camera by index.

The index value must be between 0 and **Scene::GetUserCameraCount** (p. 686).

Parameters

in	*_index*	Index of the **UserCamera** (p. 830) to get.

Returns

Pointer to the **UserCamera** (p. 830), or NULL if the index was invalid.

10.136.3.31 uint32_t gazebo::rendering::Scene::GetUserCameraCount ( ) const

Get the number of user cameras in this scene.

Returns

The number of user cameras.

10.136.3.32 VisualPtr gazebo::rendering::Scene::GetVisual (const std::string & _name) const

Get a visual by name.

10.136.3.33 VisualPtr gazebo::rendering::Scene::GetVisualAt (CameraPtr _camera, const math::Vector2i & _mousePos, std::string & _mod)

Get an entity at a pixel location using a camera.

Used for mouse picking.

Parameters

in	*_camera*	The ogre camera, used to do mouse picking
in	*_mousePos*	The position of the mouse in screen coordinates
out	*_mod*	Used for object manipulation

Returns

The selected entity, or NULL

10.136.3.34 VisualPtr gazebo::rendering::Scene::GetVisualAt (CameraPtr _camera, const math::Vector2i & _mousePos)

Get a visual at a mouse position.

Parameters

in	*_camera*	Pointer to the camera used to project the mouse position.
in	*_mousePos*	The 2d position of the mouse in pixels.

Returns

Pointer to the visual, NULL if none found.

10.136.3.35 VisualPtr gazebo::rendering::Scene::GetVisualBelow (const std::string & _visualName)

Get the closest visual below a given visual.

Parameters

in	*_visualName*	Name of the visual to search below.

Returns

Pointer to the visual below, or NULL if no visual.

10.136.3.36 void gazebo::rendering::Scene::GetVisualsBelowPoint (const math::Vector3 & _pt, std::vector< VisualPtr > & _visuals)

Get a visual directly below a point.

Parameters

in	*_pt*	3D point to get the visual below.
out	*_visuals*	The visuals below the point order in proximity.

10.136.3.37 VisualPtr gazebo::rendering::Scene::GetWorldVisual ( ) const

Get the top level world visual.

Returns

Pointer to the world visual.

10.136.3.38 void gazebo::rendering::Scene::Init ( )

Init **rendering::Scene** (p. 676).

10.136.3.39 void gazebo::rendering::Scene::Load (sdf::ElementPtr *_scene*)

Load the scene from a set of parameters.

Parameters

in	*_scene*	SDF scene element to load.

10.136.3.40 void gazebo::rendering::Scene::Load ( )

Load the scene with default parameters.

10.136.3.41 void gazebo::rendering::Scene::PreRender ( )

Process all received messages.

10.136.3.42 void gazebo::rendering::Scene::PrintSceneGraph ( )

Print the scene graph to std_out.

10.136.3.43 void gazebo::rendering::Scene::RemoveVisual (VisualPtr *_vis*)

Remove a visual from the scene.

Parameters

in	*_vis*	**Visual** (p. 885) to remove.

10.136.3.44 void gazebo::rendering::Scene::SelectVisual (const std::string & *_name,* const std::string & *_mode*)

Select a visual by name.

Parameters

in	*_name*	Name of the visual to select.
in	*_mode*	Selection mode (normal, or move).

10.136.3.45 void gazebo::rendering::Scene::SetAmbientColor (const common::Color & *_color*)

Set the ambient color.

Parameters

in	*_color*	The ambient color to use.

10.136.3.46 void gazebo::rendering::Scene::SetBackgroundColor (const common::Color & *_color*)

Set the background color.

Parameters

in	*_color*	The background color.

10.136.3.47 void gazebo::rendering::Scene::SetFog (const std::string & *_type,* const common::Color & *_color,* double *_density,* double *_start,* double *_end*)

Set the fog parameters.

Parameters

in	*_type*	Type of fog: "linear", "exp", or "exp2".
in	*_color*	Color of the fog.
in	*_density*	Fog density.
in	*_start*	Distance from camera to start the fog.
in	*_end*	Distance from camera at which the fog is at max density.

10.136.3.48 void gazebo::rendering::Scene::SetGrid (bool *_enabled*)

Set the grid on or off.

Parameters

in	*_enabled*	Set to true to turn on the grid

10.136.3.49 void gazebo::rendering::Scene::SetShadowsEnabled (bool *_value*)

Set whether shadows are on or off.

Parameters

in	*_value*	True to enable shadows, False to disable

10.136.3.50 void gazebo::rendering::Scene::SetTransparent (bool *_show*)

Enable or disable transparency for all visuals.

Parameters

in	*_show*	True to enable transparency for all visuals.

10.136.3.51 void gazebo::rendering::Scene::SetVisible (const std::string & *_name,* bool *_visible*)

Hide or show a visual.

Parameters

in	*_name*	Name of the visual to change.
in	*_visible*	True to make visual visible, False to make it invisible.

10.136.3.52 void gazebo::rendering::Scene::SetWireframe (bool *_show*)

Enable or disable wireframe for all visuals.

Parameters

in	*_show*	True to enable wireframe for all visuals.

10.136.3.53 void gazebo::rendering::Scene::ShowCollisions (bool *_show*)

Enable or disable collision visualization.

Parameters

in	*_show*	True to enable collision visualization.

10.136.3.54 void gazebo::rendering::Scene::ShowCOMs (bool *_show*)

Enable or disable center of mass visualization.

Parameters

in	*_show*	True to enable center of mass visualization.

10.136.3.55 void gazebo::rendering::Scene::ShowContacts (bool _show)

Enable or disable contact visualization.

Parameters

in	*_show*	True to enable contact visualization.

10.136.3.56 void gazebo::rendering::Scene::ShowJoints (bool _show)

Enable or disable joint visualization.

Parameters

in	*_show*	True to enable joint visualization.

10.136.3.57 void gazebo::rendering::Scene::SnapVisualToNearestBelow (const std::string & _visualName)

Move the visual to be ontop of the nearest visual below it.

Parameters

in	*_visualName*	Name of the visual to move.

10.136.3.58 std::string gazebo::rendering::Scene::StripSceneName (const std::string & _name) const

Remove the name of scene from a string.

Parameters

in	*_name*	Name to string the scene name from.

Returns

The stripped name.

10.136.4 Member Data Documentation

10.136.4.1 SkyX::SkyX* gazebo::rendering::Scene::skyx

Pointer to the sky.

The documentation for this class was generated from the following file:

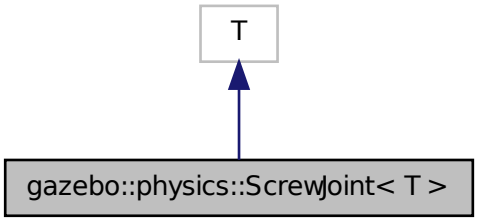
- **Scene.hh**

10.137 gazebo::physics::ScrewJoint< T > Class Template Reference

A (p. 111) screw joint, which has both prismatic and rotational DOFs.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::ScrewJoint< T >:



Public Member Functions

- **ScrewJoint** (**BasePtr** _parent)
  Constructor.
- virtual ~**ScrewJoint** ()
  Destructor.
- virtual **math::Vector3 GetAnchor** (int _index) const
  Get the anchor.
- virtual unsigned int **GetAngleCount** () const
- virtual double **GetThreadPitch** (unsigned int _index)=0
  Get screw joint thread pitch.
- virtual void **Load** (**sdf::ElementPtr** _sdf)
  *Load a **ScrewJoint** (p. 691).*
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)
  Set the anchor.
- virtual void **SetThreadPitch** (int _index, double _threadPitch)=0
  Set screw joint thread pitch.

Protected Attributes

- **math::Vector3 fakeAnchor**
  The anchor value is not used internally.
- double **threadPitch**
  Pitch of the thread.

10.137.1 Detailed Description

template<class T>class gazebo::physics::ScrewJoint< T >

A (p. 111) screw joint, which has both prismatic and rotational DOFs.

10.137.2 Constructor & Destructor Documentation

10.137.2.1 template<class T > gazebo::physics::ScrewJoint< T >::ScrewJoint (BasePtr *_parent*) [inline], [explicit]

Constructor.

Parameters

in	*_parent*	Parent of the joint.

References gazebo::physics::Base::SCREW_JOINT.

10.137.2.2 template<class T > virtual gazebo::physics::ScrewJoint< T >::~ScrewJoint ( ) [inline], [virtual]

Destructor.

10.137.3 Member Function Documentation

10.137.3.1 template<class T > math::Vector3 gazebo::physics::ScrewJoint< T >::GetAnchor (int *_index*) const [virtual]

Get the anchor.

Parameters

in	*_index*	Index of the axis. Not Used.

Returns

Anchor for the joint.

10.137.3.2 template<class T > virtual unsigned int gazebo::physics::ScrewJoint< T >::GetAngleCount ( ) const [inline], [virtual]

10.137.3.3 template<class T > virtual double gazebo::physics::ScrewJoint< T >::GetThreadPitch (unsigned int *_index*) [pure virtual]

Get screw joint thread pitch.

This must be implemented in a child class

Parameters

in	*_index*	Index of the axis.

Returns

_threadPitch Thread pitch value.

10.137.3.4 template<class T > virtual void gazebo::physics::ScrewJoint< T >::Load (sdf::ElementPtr _sdf) [inline], [virtual]

Load a **ScrewJoint** (p. 691).

Parameters

in	*_sdf*	SDF value to load from

References sdf::Element::GetElement(), sdf::Element::GetValueDouble(), sdf::Element::GetValueVector3(), gzerr, sdf::Element::HasElement(), and gazebo::physics::ScrewJoint< T >::threadPitch.

10.137.3.5 template<class T > void gazebo::physics::ScrewJoint< T >::SetAnchor (int _index, const math::Vector3 & _anchor) [virtual]

Set the anchor.

Parameters

in	*_index*	Index of the axis. Not Used.
in	*_anchor*	Anchor value for the joint.

10.137.3.6 template<class T > virtual void gazebo::physics::ScrewJoint< T >::SetThreadPitch (int _index, double _threadPitch) [pure virtual]

Set screw joint thread pitch.

This must be implemented in a child class

Parameters

in	*_index*	Index of the axis.
in	*_threadPitch*	Thread pitch value.

10.137.4 Member Data Documentation

10.137.4.1 template<class T > math::Vector3 gazebo::physics::ScrewJoint< T >::fakeAnchor [protected]

The anchor value is not used internally.

10.137.4.2 template<class T > double gazebo::physics::ScrewJoint< T >::threadPitch [protected]

Pitch of the thread.

Referenced by gazebo::physics::ScrewJoint< T >::Load().

The documentation for this class was generated from the following file:

- **ScrewJoint.hh**

10.138 sdf::SDF Class Reference

Base **SDF** (p. 695) 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. 695) 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**

10.138.1 Detailed Description

Base **SDF** (p. 695) class.

10.138.2 Constructor & Destructor Documentation

10.138.2.1 sdf::SDF::SDF ( )

10.138.3 Member Function Documentation

10.138.3.1 void sdf::SDF::PrintDescription ( )

10.138.3.2 void sdf::SDF::PrintDoc ( )

10.138.3.3 void sdf::SDF::PrintValues ( )

10.138.3.4 void sdf::SDF::PrintWiki ( )

10.138.3.5 void sdf::SDF::SetFromString (const std::string & *_sdfData*)

Set **SDF** (p. 695) values from a string.

10.138.3.6 std::string sdf::SDF::ToString ( ) const

10.138.3.7 void sdf::SDF::Write (const std::string & *_filename*)

10.138.4 Member Data Documentation

10.138.4.1 ElementPtr sdf::SDF::root

10.138.4.2 std::string sdf::SDF::version [static]

The documentation for this class was generated from the following file:

- **SDF.hh**

10.139 gazebo::rendering::SelectionObj Class Reference

A (p. 111) 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. 696) object.*
- std::string **GetVisualName** () const

  Get the name of the visual the selection obj is attached to.
- void **Init** ()

  *Initialize the **rendering::SelectionObj** (p. 696) 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.

10.139.1 Detailed Description

A (p. 111) graphical selection object.

Used to draw a visual around a selected object.

10.139.2 Constructor & Destructor Documentation

10.139.2.1 gazebo::rendering::SelectionObj::SelectionObj (Scene * _scene)

Constructor.

Parameters

in	*_scene*	**Scene** (p. 676) to use.

10.139.2.2 virtual gazebo::rendering::SelectionObj::~SelectionObj ( ) [virtual]

Destructor.

10.139.3 Member Function Documentation

10.139.3.1 void gazebo::rendering::SelectionObj::Attach (VisualPtr _visual)

Set the position of the node.

Parameters

in	*This*	draws the selection object around the passed in visual.

10.139.3.2 void gazebo::rendering::SelectionObj::Clear ( )

Clear the **rendering::SelectionObj** (p. 696) object.

10.139.3.3 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.

10.139.3.4 void gazebo::rendering::SelectionObj::Init ( )

Initialize the **rendering::SelectionObj** (p. 696) object.

10.139.3.5 bool gazebo::rendering::SelectionObj::IsActive ( ) const

Return true if the user is move the selection obj.

Returns

True if something is selected.

10.139.3.6 void gazebo::rendering::SelectionObj::SetActive (bool *_active*)

Set true if the user is moving the selection obj.

Parameters

in	*_active*	True if the user is interacting with the selection object.

10.139.3.7 void gazebo::rendering::SelectionObj::SetHighlight (const std::string & *_mod*)

Highlight the selection object based on a modifier.

Parameters

in	*_mod*	Modifier used when highlighting the selection object.

The documentation for this class was generated from the following file:

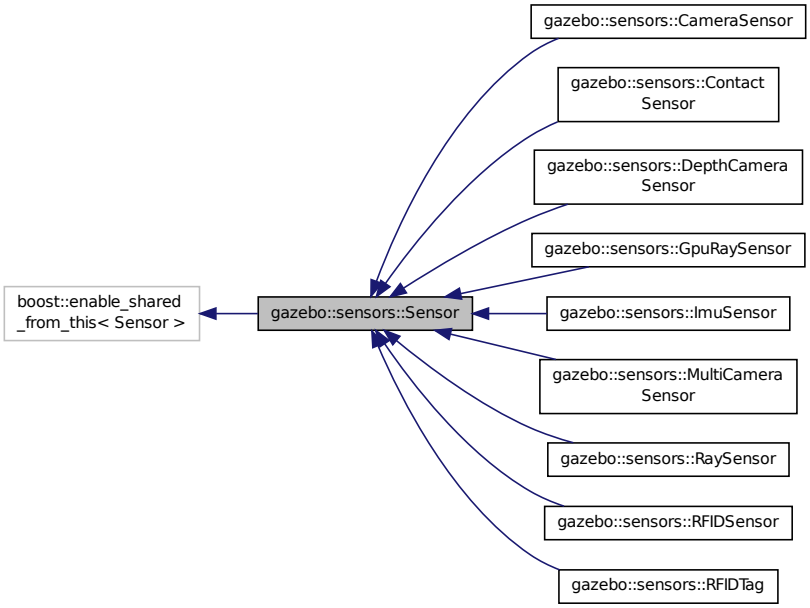
- **SelectionObj.hh**

10.140 gazebo::sensors::Sensor Class Reference

Base class for sensors.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::Sensor:



Public Member Functions

- **Sensor** (**SensorCategory** _cat)

  Constructor.
- virtual ~**Sensor** ()

  Destructor.
- template<typename T >
  event::ConnectionPtr ConnectUpdated (T _subscriber)

  Connect a signal that is triggered when the sensor is updated.
- void **DisconnectUpdated** (**event::ConnectionPtr** &_c)

  Disconnect from a the updated signal.
- void **FillMsg** (msgs::Sensor &_msg)

  fills a msgs::Sensor message.
- virtual void **Fini** ()

  Finalize the sensor.
- **SensorCategory GetCategory** () const

  Get the category of the sensor.
- **common::Time GetLastMeasurementTime** ()

  Return last measurement time.
- **common::Time GetLastUpdateTime** ()

  Return last update time.
- std::string **GetName** () const

  Get name.
- std::string **GetParentName** () const

  Returns the name of the sensor parent.
- virtual **math::Pose GetPose** () const

  Get the current pose.
- std::string **GetScopedName** () const

  Get fully scoped name of the sensor.
- virtual std::string **GetTopic** () const

  Returns the topic name as set in SDF.
- std::string **GetType** () const

  Get sensor type.
- double **GetUpdateRate** ()

  Get the update rate of the sensor.
- bool **GetVisualize** () const

  Return true if user requests the sensor to be visualized via tag: <visualize>true</visualize> in SDF.
- std::string **GetWorldName** () const

  Returns the name of the world the sensor is in.
- virtual void **Init** ()

  Initialize the sensor.
- virtual bool **IsActive** ()

  Returns true if sensor generation is active.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** _sdf)

  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

- void **ResetLastUpdateTime** ()

  Reset the lastUpdateTime to zero.
- virtual void **SetActive** (bool _value)

  Set whether the sensor is active or not.
- virtual void **SetParent** (const std::string &_name)

  Set the parent of the sensor.
- void **SetUpdateRate** (double _hz)

  Set the update rate of the sensor.
- void **Update** (bool _force)

  Update the sensor.

Protected Member Functions

- virtual void **UpdateImpl** (bool)

  This gets overwritten by derived sensor types.

Protected Attributes

- bool **active**

  True if sensor generation is active.
- std::vector< **event::ConnectionPtr** > **connections**

  All event connections.
- **common::Time lastMeasurementTime**

  Stores last time that a sensor measurement was generated; this value must be updated within each sensor's UpdateImpl.
- **common::Time lastUpdateTime**

  Time of the last update.
- **transport::NodePtr node**

  Node for communication.
- std::string **parentName**

  Name of the parent.
- std::vector< **SensorPluginPtr** > **plugins**

  All the plugins for the sensor.
- **math::Pose pose**

  Pose of the sensor.
- **transport::SubscriberPtr poseSub**

  Subscribe to pose updates.
- **sdf::ElementPtr sdf**

  Pointer the the SDF element for the sensor.
- **common::Time updatePeriod**

  Desired time between updates, set indirectly by ***Sensor::SetUpdateRate*** *(*p. 705*).*
- **gazebo::physics::WorldPtr world**

  Pointer to the world.

10.140.1 Detailed Description

Base class for sensors.

10.140.2 Constructor & Destructor Documentation

10.140.2.1 gazebo::sensors::Sensor::Sensor (SensorCategory _cat) [explicit]

Constructor.

Parameters

in	*_class*	

10.140.2.2 virtual gazebo::sensors::Sensor::~Sensor ( ) [virtual]

Destructor.

10.140.3 Member Function Documentation

10.140.3.1 template<typename T > event::ConnectionPtr gazebo::sensors::Sensor::ConnectUpdated (T _subscriber) [inline]

Connect a signal that is triggered when the sensor is updated.

Parameters

in	*_subscriber*	Callback that receives the signal.

Returns

A (p. 111) pointer to the connection. This must be kept in scope.

See Also

Sensor::DisconnectUpdated (p. 701)

References gazebo::event::EventT< T >::Connect().

10.140.3.2 void gazebo::sensors::Sensor::DisconnectUpdated (event::ConnectionPtr & _c) [inline]

Disconnect from a the updated signal.

Parameters

in	*_c*	The connection to disconnect

See Also

Sensor::ConnectUpdated (p. 701)

References gazebo::event::EventT< T >::Disconnect().

10.140.3.3 void gazebo::sensors::Sensor::FillMsg (msgs::Sensor & *_msg*)

fills a msgs::Sensor message.

Parameters

out	*_msg*	Message to fill.

10.140.3.4 virtual void gazebo::sensors::Sensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented in **gazebo::sensors::MultiCameraSensor** (p. 524), **gazebo::sensors::CameraSensor** (p. 190), **gazebo::sensors::GpuRaySensor** (p. 332), **gazebo::sensors::ContactSensor** (p. 243), **gazebo::sensors::DepthCameraSensor** (p. 259), **gazebo::sensors::RFIDSensor** (p. 659), **gazebo::sensors::RaySensor** (p. 643), **gazebo::sensors::RFIDTag** (p. 661), and **gazebo::sensors::ImuSensor** (p. 367).

10.140.3.5 SensorCategory gazebo::sensors::Sensor::GetCategory ( ) const

Get the category of the sensor.

Returns

The category of the sensor.

See Also

SensorCategory (p. 102)

10.140.3.6 common::Time gazebo::sensors::Sensor::GetLastMeasurementTime ( )

Return last measurement time.

Returns

Time of last measurement.

10.140.3.7 common::Time gazebo::sensors::Sensor::GetLastUpdateTime ( )

Return last update time.

Returns

Time of last update.

10.140.3.8 std::string gazebo::sensors::Sensor::GetName ( ) const

Get name.

Returns

Name of sensor.

10.140.3.9 std::string gazebo::sensors::Sensor::GetParentName ( ) const

Returns the name of the sensor parent.

The parent name is set by **Sensor::SetParent** (p. 705).

Returns

Name of Parent.

10.140.3.10 virtual math::Pose gazebo::sensors::Sensor::GetPose ( ) const [virtual]

Get the current pose.

Returns

Current pose of the sensor.

10.140.3.11 std::string gazebo::sensors::Sensor::GetScopedName ( ) const

Get fully scoped name of the sensor.

Returns

world_name::parent_name::sensor_name.

10.140.3.12 virtual std::string gazebo::sensors::Sensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented in **gazebo::sensors::RaySensor** (p. 646), **gazebo::sensors::CameraSensor** (p. 191), and **gazebo::sensors::MultiCameraSensor** (p. 526).

10.140.3.13 std::string gazebo::sensors::Sensor::GetType ( ) const

Get sensor type.

Returns

Type of sensor.

10.140.3.14 double gazebo::sensors::Sensor::GetUpdateRate ( )

Get the update rate of the sensor.

Returns

_hz update rate of sensor. Returns 0 if unthrottled.

10.140.3.15 bool gazebo::sensors::Sensor::GetVisualize ( ) const

Return true if user requests the sensor to be visualized via tag: <visualize>true</visualize> in SDF.

Returns

True if visualized, false if not.

10.140.3.16 std::string gazebo::sensors::Sensor::GetWorldName ( ) const

Returns the name of the world the sensor is in.

Returns

Name of the world.

10.140.3.17 virtual void gazebo::sensors::Sensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented in **gazebo::sensors::GpuRaySensor** (p. 337), **gazebo::sensors::ContactSensor** (p. 245), **gazebo::sensors::CameraSensor** (p. 191), **gazebo::sensors::DepthCameraSensor** (p. 260), **gazebo::sensors::RFIDSensor** (p. 659), **gazebo::sensors::RaySensor** (p. 646), **gazebo::sensors::RFIDTag** (p. 662), **gazebo::sensors::MultiCameraSensor** (p. 526), and **gazebo::sensors::ImuSensor** (p. 368).

10.140.3.18 virtual bool gazebo::sensors::Sensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented in **gazebo::sensors::RaySensor** (p. 647), **gazebo::sensors::ContactSensor** (p. 245), **gazebo::sensors::CameraSensor** (p. 191), and **gazebo::sensors::MultiCameraSensor** (p. 526).

10.140.3.19 virtual void gazebo::sensors::Sensor::Load (const std::string & *_worldName*, sdf::ElementPtr *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 698) parameters.
in	*_worldName*	Name of world to load from.

Reimplemented in **gazebo::sensors::ContactSensor** (p. 245), **gazebo::sensors::CameraSensor** (p. 191), **gazebo::sensors::RFIDSensor** (p. 659), and **gazebo::sensors::ImuSensor** (p. 368).

10.140.3.20 virtual void gazebo::sensors::Sensor::Load (const std::string & _worldName) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from.

Reimplemented in **gazebo::sensors::GpuRaySensor** (p. 337), **gazebo::sensors::ContactSensor** (p. 245), **gazebo::sensors::CameraSensor** (p. 192), **gazebo::sensors::DepthCameraSensor** (p. 260), **gazebo::sensors::RFIDSensor** (p. 659), **gazebo::sensors::RaySensor** (p. 647), **gazebo::sensors::RFIDTag** (p. 662), **gazebo::sensors::MultiCameraSensor** (p. 526), and **gazebo::sensors::ImuSensor** (p. 368).

10.140.3.21 void gazebo::sensors::Sensor::ResetLastUpdateTime ( )

Reset the lastUpdateTime to zero.

10.140.3.22 virtual void gazebo::sensors::Sensor::SetActive (bool _value) [virtual]

Set whether the sensor is active or not.

Parameters

in	*_value*	True if active, false if not.

Reimplemented in **gazebo::sensors::DepthCameraSensor** (p. 260).

10.140.3.23 virtual void gazebo::sensors::Sensor::SetParent (const std::string & _name) [virtual]

Set the parent of the sensor.

Parameters

in	*_name*	Name of the parent.

Reimplemented in **gazebo::sensors::CameraSensor** (p. 192), and **gazebo::sensors::DepthCameraSensor** (p. 261).

10.140.3.24 void gazebo::sensors::Sensor::SetUpdateRate (double _hz)

Set the update rate of the sensor.

Parameters

in	*_hz*	update rate of sensor.

10.140.3.25 void gazebo::sensors::Sensor::Update (bool _force)

Update the sensor.

Parameters

in	*_force*	True to force update, false otherwise.

10.140.3.26 virtual void gazebo::sensors::Sensor::UpdateImpl (bool) [inline], [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented in **gazebo::sensors::MultiCameraSensor** (p. 527), **gazebo::sensors::CameraSensor** (p. 192), **gazebo::sensors::GpuRaySensor** (p. 338), **gazebo::sensors::ContactSensor** (p. 246), **gazebo::sensors::DepthCameraSensor** (p. 261), **gazebo::sensors::RFIDSensor** (p. 660), **gazebo::sensors::RaySensor** (p. 647), **gazebo::sensors::RFIDTag** (p. 662), and **gazebo::sensors::ImuSensor** (p. 369).

10.140.4 Member Data Documentation

10.140.4.1 bool gazebo::sensors::Sensor::active [protected]

True if sensor generation is active.

10.140.4.2 std::vector<event::ConnectionPtr> gazebo::sensors::Sensor::connections [protected]

All event connections.

10.140.4.3 common::Time gazebo::sensors::Sensor::lastMeasurementTime [protected]

Stores last time that a sensor measurement was generated; this value must be updated within each sensor's UpdateImpl.

10.140.4.4 common::Time gazebo::sensors::Sensor::lastUpdateTime [protected]

Time of the last update.

10.140.4.5 transport::NodePtr gazebo::sensors::Sensor::node [protected]

Node for communication.

10.140.4.6 std::string gazebo::sensors::Sensor::parentName [protected]

Name of the parent.

10.140.4.7 std::vector<SensorPluginPtr> gazebo::sensors::Sensor::plugins [protected]

All the plugins for the sensor.

10.140.4.8 math::Pose gazebo::sensors::Sensor::pose [protected]

Pose of the sensor.

10.140.4.9 transport::SubscriberPtr gazebo::sensors::Sensor::poseSub [protected]

Subscribe to pose updates.

10.140.4.10 sdf::ElementPtr gazebo::sensors::Sensor::sdf [protected]

Pointer the the SDF element for the sensor.

10.140.4.11 common::Time gazebo::sensors::Sensor::updatePeriod [protected]

Desired time between updates, set indirectly by **Sensor::SetUpdateRate** (p. 705).

10.140.4.12 gazebo::physics::WorldPtr gazebo::sensors::Sensor::world [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **Sensor.hh**

10.141 SensorFactor Class Reference

The sensor factory; the class is just for namespacing purposes.

`#include <sensors/sensors.hh>`

10.141.1 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**

10.142 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).

10.142.1 Member Function Documentation

10.142.1.1 static void gazebo::sensors::SensorFactory::GetSensorTypes (std::vector< std::string > & *_types*) [static]

Get all the sensor types.

Parameters

_types	Vector of strings of the sensor types, populated by function

10.142.1.2 static SensorPtr gazebo::sensors::SensorFactory::NewSensor (const std::string & *_className*) [static]

Create a new instance of a sensor.

Used by the world when reading the world file.

Parameters

in	*_className*	Name of sensor class

Returns

Pointer to **Sensor** (p. 698)

10.142.1.3 static void gazebo::sensors::SensorFactory::RegisterAll ( ) [static]

Register all known sensors.

- **sensors::CameraSensor** (p. 188)
- **sensors::DepthCameraSensor** (p. 258)
- **sensors::GpuRaySensor** (p. 328)
- **sensors::RaySensor** (p. 641)
- **sensors::ContactSensor** (p. 241)
- **sensors::RFIDSensor** (p. 658)
- **sensors::RFIDTag** (p. 660)

10.142.1.4 static void gazebo::sensors::SensorFactory::RegisterSensor (const std::string & *_className*, SensorFactoryFn *_factoryfn*) [static]

Register a sensor class (called by sensor registration function).

Parameters

in	*_className*	Name of class of sensor to register.
in	*_factoryfn*	Function handle for registration.

The documentation for this class was generated from the following file:

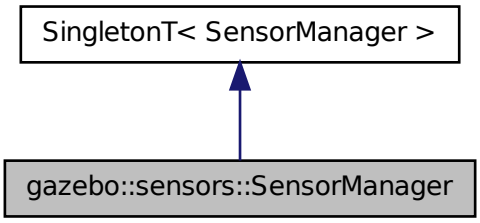
- **SensorFactory.hh**

10.143 gazebo::sensors::SensorManager Class Reference

Class to manage and update all sensors.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::SensorManager:



Public Member Functions

- std::string **CreateSensor** (**sdf::ElementPtr** _elem, const std::string &_worldName, const std::string &_parentName)

  Add a sensor from an SDF element.
- void **Fini** ()

  Finalize all the sensors.
- **SensorPtr GetSensor** (const std::string &_name) const

  Get a sensor.
- **Sensor_V GetSensors** () const

  Get all the sensors.
- void **GetSensorTypes** (std::vector< std::string > &_types) const

  Get all the sensor types.
- void **Init** ()

Init all the sensors.

- void **RemoveSensor** (const std::string &_name)

  Remove a sensor.
- void **RemoveSensors** ()

  Remove all sensors.
- void **ResetLastUpdateTimes** ()

  Reset last update times in all sensors.
- void **Run** () **GAZEBO_DEPRECATED**(1.5)

  Deprecated.
- void **RunThreads** ()

  Run sensor updates in separate threads.
- bool **SensorsInitialized** ()

  True if SensorManager::initSensors queue is empty i.e.
- void **Stop** ()

  Stop the run thread.
- void **Update** (bool _force=false)

  Update all the sensors.

Additional Inherited Members

10.143.1 Detailed Description

Class to manage and update all sensors.

10.143.2 Member Function Documentation

10.143.2.1 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

in	*_elem*	The SDF element that describes the sensor
in	*_worldName*	Name of the world in which to create the sensor
in	*_parentName*	The name of the parent link which the sensor is attached to.

Returns

The name of the sensor

10.143.2.2 void gazebo::sensors::SensorManager::Fini ( )

Finalize all the sensors.

10.143.2.3 SensorPtr gazebo::sensors::SensorManager::GetSensor (const std::string & _name) const

Get a sensor.

Parameters

in	*_name*	The name of a sensor to find.

Returns

A (p. 111) pointer to the sensor. NULL if not found.

10.143.2.4 Sensor_V gazebo::sensors::SensorManager::GetSensors ( ) const

Get all the sensors.

Returns

Vector of all the sensors.

10.143.2.5 void gazebo::sensors::SensorManager::GetSensorTypes (std::vector< std::string > & _types) const

Get all the sensor types.

Parameters

out	*All*	the sensor types.

10.143.2.6 void gazebo::sensors::SensorManager::Init ( )

Init all the sensors.

10.143.2.7 void gazebo::sensors::SensorManager::RemoveSensor (const std::string & _name)

Remove a sensor.

Parameters

in	*_name*	The name of the sensor to remove.

10.143.2.8 void gazebo::sensors::SensorManager::RemoveSensors ( )

Remove all sensors.

10.143.2.9 void gazebo::sensors::SensorManager::ResetLastUpdateTimes ( )

Reset last update times in all sensors.

10.143.2.10 void gazebo::sensors::SensorManager::Run ( )

Deprecated.

See Also

RunThreads (p. 712)

10.143.2.11 void gazebo::sensors::SensorManager::RunThreads ( )

Run sensor updates in separate threads.

This will only run non-image based sensor updates.

10.143.2.12 bool gazebo::sensors::SensorManager::SensorsInitialized ( )

True if SensorManager::initSensors queue is empty i.e.

all sensors managed by **SensorManager** (p. 709) have been initialized

10.143.2.13 void gazebo::sensors::SensorManager::Stop ( )

Stop the run thread.

10.143.2.14 void gazebo::sensors::SensorManager::Update (bool *_force* = `false`)

Update all the sensors.

Checks to see if any sensor need to be initialized first, then updates all sensors once.

Parameters

in	*_force*	True force update, false if not

The documentation for this class was generated from the following file:

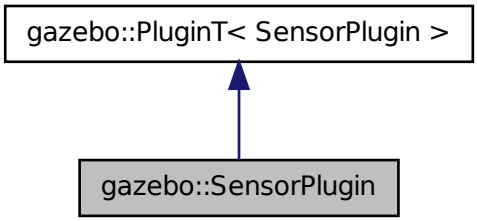
- **SensorManager.hh**

10.144 gazebo::SensorPlugin Class Reference

A (p. 111) plugin with access to physics::Sensor.

`#include <common/common.hh>`

Inheritance diagram for gazebo::SensorPlugin:



Public Member Functions

- **SensorPlugin** ()

  Constructor.
- virtual ~**SensorPlugin** ()

  Destructor.
- virtual void **Init** ()

  Override this method for custom plugin initialization behavior.
- virtual void **Load** (**sensors::SensorPtr** _sensor, **sdf::ElementPtr** _sdf)=0

  Load function.
- virtual void **Reset** ()

  Override this method for custom plugin reset behavior.

Additional Inherited Members

10.144.1 Detailed Description

A (p. 111) plugin with access to physics::Sensor.

See `reference`.

10.144.2 Constructor & Destructor Documentation

10.144.2.1 gazebo::SensorPlugin::SensorPlugin ( ) [inline]

Constructor.

References gazebo::SENSOR_PLUGIN, and gazebo::PluginT< SensorPlugin >::type.

10.144.2.2 virtual gazebo::SensorPlugin::~SensorPlugin ( ) [inline], [virtual]

Destructor.

10.144.3 Member Function Documentation

10.144.3.1 virtual void gazebo::SensorPlugin::Init ( ) [inline], [virtual]

Override this method for custom plugin initialization behavior.

10.144.3.2 virtual void gazebo::SensorPlugin::Load (sensors::SensorPtr *_sensor,* sdf::ElementPtr *_sdf*) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	*_sensor*	Pointer the Sensor.
in	*_sdf*	Pointer the the SDF element of the plugin.

10.144.3.3 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**

10.145 gazebo::Server Class Reference

`#include <Server.hh>`

Public Member Functions

- **Server** ()
- virtual ~**Server** ()
- void **Fini** ()
- bool **GetInitialized** () const
- void **Init** ()
- bool **LoadFile** (const std::string &_filename="worlds/empty.world", const std::string &_physics="")

  Load a world file and optionally override physics engine type.
- bool **LoadString** (const std::string &_sdfString)
- bool **ParseArgs** (int argc, char **argv)
- void **PrintUsage** ()
- void **Run** ()
- void **SetParams** (const **common::StrStr_M** &params)
- void **Stop** ()

Public Attributes

- int **systemPluginsArgc**
- char ** **systemPluginsArgv**

10.145.1 Constructor & Destructor Documentation

10.145.1.1 gazebo::Server::Server ( )

10.145.1.2 virtual gazebo::Server::~Server ( ) [virtual]

10.145.2 Member Function Documentation

10.145.2.1 void gazebo::Server::Fini ( )

10.145.2.2 bool gazebo::Server::GetInitialized ( ) const

10.145.2.3 void gazebo::Server::Init ( )

10.145.2.4 bool gazebo::Server::LoadFile (const std::string & *_filename* = "worlds/empty.world", const std::string & *_physics* = "")

Load a world file and optionally override physics engine type.

Parameters

in	*_filename*	Name of the world file to load.
in	*_physics*	Type of physics engine to use (ode|bullet).

10.145.2.5 bool gazebo::Server::LoadString (const std::string & *_sdfString*)

10.145.2.6 bool gazebo::Server::ParseArgs (int *argc*, char ** *argv*)

10.145.2.7 void gazebo::Server::PrintUsage ( )

10.145.2.8 void gazebo::Server::Run ( )

10.145.2.9 void gazebo::Server::SetParams (const common::StrStr_M & *params*)

10.145.2.10 void gazebo::Server::Stop ( )

10.145.3 Member Data Documentation

10.145.3.1 int gazebo::Server::systemPluginsArgc

10.145.3.2 char** gazebo::Server::systemPluginsArgv

The documentation for this class was generated from the following file:

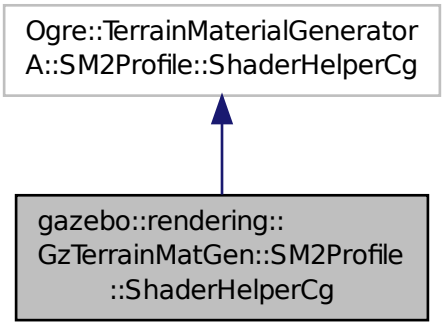
- **Server.hh**

10.146 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference

Keeping the CG shader for reference.

`#include <Heightmap.hh>`

Inheritance diagram for gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg:



Public Member Functions

- virtual Ogre::HighLevelGpuProgramPtr **generateFragmentProgram** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt)
- virtual Ogre::HighLevelGpuProgramPtr **generateVertexProgram** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt)

Protected Member Functions

- virtual void **defaultVpParams** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, const Ogre::HighLevelGpuProgramPtr &_prog)
- virtual void **generateVertexProgramSource** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpDynamicShadows** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual unsigned int **generateVpDynamicShadowsParams** (unsigned int _texCoordStart, const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpFooter** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpHeader** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)

10.146.1 Detailed Description

Keeping the CG shader for reference.

Utility class to help with generating shaders for Cg / HLSL.

10.146.2 Member Function Documentation

10.146.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::defaultVpParams (const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* const Ogre::HighLevelGpuProgramPtr & *_prog*) [protected], [virtual]

10.146.2.2 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateFragmentProgram (const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt*) [virtual]

10.146.2.3 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgram (const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt*) [virtual]

10.146.2.4 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgramSource (const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.146.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadows (const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.146.2.6 virtual unsigned int gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadowsParams (unsigned int *_texCoordStart,* const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.146.2.7 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpFooter (const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.146.2.8 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpHeader (const SM2Profile * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

The documentation for this class was generated from the following file:

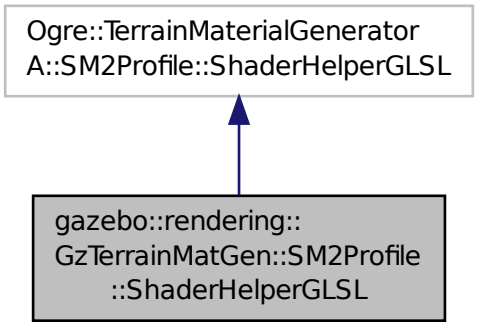
- **Heightmap.hh**

10.147 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference

Utility class to help with generating shaders for GLSL.

`#include <Heightmap.hh>`

Inheritance diagram for gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL:



Public Member Functions

- virtual Ogre::HighLevelGpuProgramPtr **generateFragmentProgram** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt)
- virtual Ogre::HighLevelGpuProgramPtr **generateVertexProgram** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt)
- virtual void **updateParams** (const **SM2Profile** ∗_prof, const Ogre::MaterialPtr &_mat, const Ogre::Terrain ∗_terrain, bool _compositeMap)

Protected Member Functions

- virtual void **defaultVpParams** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, const Ogre::HighLevelGpuProgramPtr &_prog)
- void **generateFpDynamicShadows** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFpDynamicShadowsHelpers** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFpDynamicShadowsParams** (Ogre::uint ∗_texCoord, Ogre::uint ∗_sampler, const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFpFooter** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFpHeader** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFpLayer** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType tt, Ogre::uint _layer, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateFragmentProgramSource** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVertexProgramSource** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)

- virtual void **generateVpDynamicShadows** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual unsigned int **generateVpDynamicShadowsParams** (unsigned int _texCoordStart, const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpFooter** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **generateVpHeader** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType &_outStream)
- virtual void **updateVpParams** (const **SM2Profile** ∗_prof, const Ogre::Terrain ∗_terrain, TechniqueType _tt, const Ogre::GpuProgramParametersSharedPtr &_params)

10.147.1 Detailed Description

Utility class to help with generating shaders for GLSL.

10.147.2 Member Function Documentation

10.147.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::defaultVpParams (const **SM2Profile** ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *_tt,* const Ogre::HighLevelGpuProgramPtr & *_prog*) [protected], [virtual]

10.147.2.2 void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadows (const **SM2Profile** ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected]

10.147.2.3 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadowsHelpers (const **SM2Profile** ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.4 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadowsParams (Ogre::uint ∗ *_texCoord,* Ogre::uint ∗ *_sampler,* const **SM2Profile** ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpFooter (const **SM2Profile** ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.6 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpHeader (const **SM2Profile** ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.7 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpLayer (const **SM2Profile** ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *tt,* Ogre::uint *_layer,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.8 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgram (const **SM2Profile** ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *_tt*) [virtual]

10.147.2.9 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgramSource (const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.10 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL-::generateVertexProgram (const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt*) [virtual]

10.147.2.11 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgramSource (const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.12 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamicShadows (const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.13 virtual unsigned int gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamic-ShadowsParams (unsigned int *_texCoordStart,* const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.14 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpFooter (const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.15 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpHeader (const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected], [virtual]

10.147.2.16 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::updateParams (const **SM2Profile** * *_prof,* const Ogre::MaterialPtr & *_mat,* const Ogre::Terrain * *_terrain,* bool *_compositeMap*) [virtual]

10.147.2.17 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::updateVpParams (const **SM2Profile** * *_prof,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt,* const Ogre::GpuProgramParametersSharedPtr & *_params*) [protected], [virtual]

The documentation for this class was generated from the following file:

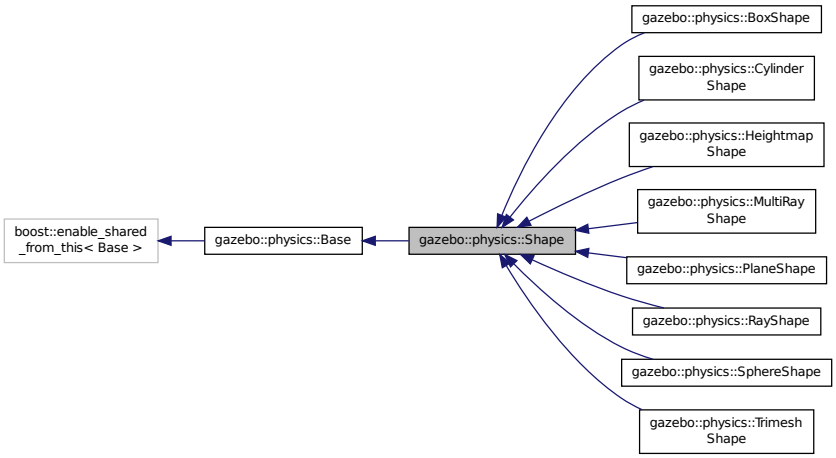
- **Heightmap.hh**

10.148 gazebo::physics::Shape Class Reference

Base (p. 137) class for all shapes.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Shape:



Public Member Functions

- **Shape** (**CollisionPtr** _parent)

  Constructor.
- virtual ∼**Shape** ()

  Destructor.
- virtual void **FillMsg** (msgs::Geometry &_msg)=0

  Fill in the values for a geometry message.
- virtual void **Init** ()=0

  Initialize the shape.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)=0

  Process a geometry message.

Protected Attributes

- **CollisionPtr collisionParent**

  This shape's collision parent.

Additional Inherited Members

10.148.1 Detailed Description

Base (p. 137) class for all shapes.

10.148.2 Constructor & Destructor Documentation

10.148.2.1 gazebo::physics::Shape::Shape (CollisionPtr _parent) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of the shape.

10.148.2.2 virtual gazebo::physics::Shape::~Shape ( ) [virtual]

Destructor.

10.148.3 Member Function Documentation

10.148.3.1 virtual void gazebo::physics::Shape::FillMsg (msgs::Geometry & _msg) [pure virtual]

Fill in the values for a geometry message.

Parameters

out	*_msg*	The geometry message to fill.

Implemented in **gazebo::physics::MultiRayShape** (p. 531), **gazebo::physics::RayShape** (p. 650), **gazebo::physics::TrimeshShape** (p. 824), **gazebo::physics::HeightmapShape** (p. 354), **gazebo::physics::PlaneShape** (p. 592), **gazebo::physics::CylinderShape** (p. 252), **gazebo::physics::SphereShape** (p. 753), and **gazebo::physics::BoxShape** (p. 155).

10.148.3.2 virtual void gazebo::physics::Shape::Init ( ) [pure virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::Base** (p. 145).

Implemented in **gazebo::physics::RayShape** (p. 651), **gazebo::physics::HeightmapShape** (p. 355), **gazebo::physics::TrimeshShape** (p. 824), **gazebo::physics::MultiRayShape** (p. 534), **gazebo::physics::PlaneShape** (p. 592), **gazebo::physics::SphereShape** (p. 753), **gazebo::physics::BoxShape** (p. 155), and **gazebo::physics::CylinderShape** (p. 253).

10.148.3.3 virtual void gazebo::physics::Shape::ProcessMsg (const msgs::Geometry & _msg) [pure virtual]

Process a geometry message.

Parameters

in	*_msg*	The message to set values from.

Implemented in **gazebo::physics::MultiRayShape** (p. 534), **gazebo::physics::RayShape** (p. 651), **gazebo::physics::TrimeshShape** (p. 825), **gazebo::physics::HeightmapShape** (p. 356), **gazebo::physics::PlaneShape** (p. 592), **gazebo::physics::CylinderShape** (p. 253), **gazebo::physics::SphereShape** (p. 753), and **gazebo::physics::BoxShape** (p. 155).

10.148.4 Member Data Documentation

10.148.4.1 CollisionPtr gazebo::physics::Shape::collisionParent [protected]

This shape's collision parent.

The documentation for this class was generated from the following file:

- **Shape.hh**

10.149 gazebo::sensors::SimTimeEvent Class Reference

`#include <SensorManager.hh>`

Public Attributes

- boost::condition_variable * **condition**

  The condition to notify.
- **common::Time time**

  The time at which to trigger the condition.

10.149.1 Detailed Description

A (p. 111) simulation time event

10.149.2 Member Data Documentation

10.149.2.1 boost::condition_variable* gazebo::sensors::SimTimeEvent::condition

The condition to notify.

10.149.2.2 common::Time gazebo::sensors::SimTimeEvent::time

The time at which to trigger the condition.

The documentation for this class was generated from the following file:

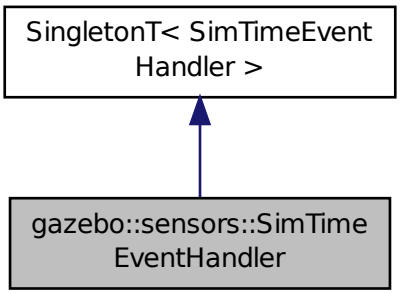
- **SensorManager.hh**

10.150 gazebo::sensors::SimTimeEventHandler Class Reference

Monitors simulation time, and notifies conditions when a specified time has been reached.

`#include <SensorManager.hh>`

Inheritance diagram for gazebo::sensors::SimTimeEventHandler:



Public Member Functions

- **SimTimeEventHandler** ()

  Constructor.
- virtual ∼**SimTimeEventHandler** ()

  Destructor.
- void **AddRelativeEvent** (const **common::Time** &_time, boost::condition_variable ∗_var)

  Add a new event to the handler.

Additional Inherited Members

10.150.1 Detailed Description

Monitors simulation time, and notifies conditions when a specified time has been reached.

10.150.2 Constructor & Destructor Documentation

10.150.2.1 gazebo::sensors::SimTimeEventHandler::SimTimeEventHandler ( )

Constructor.

10.150.2.2 virtual gazebo::sensors::SimTimeEventHandler::∼SimTimeEventHandler ( ) [virtual]

Destructor.

10.150.3 Member Function Documentation

10.150.3.1 void gazebo::sensors::SimTimeEventHandler::AddRelativeEvent (const common::Time & _time, boost::condition_variable * _var)

Add a new event to the handler.

Parameters

in	*_time*	Time of the new event. The current sim time will be add to this time.
in	*_var*	Condition to notify when the time has been reached.

The documentation for this class was generated from the following file:

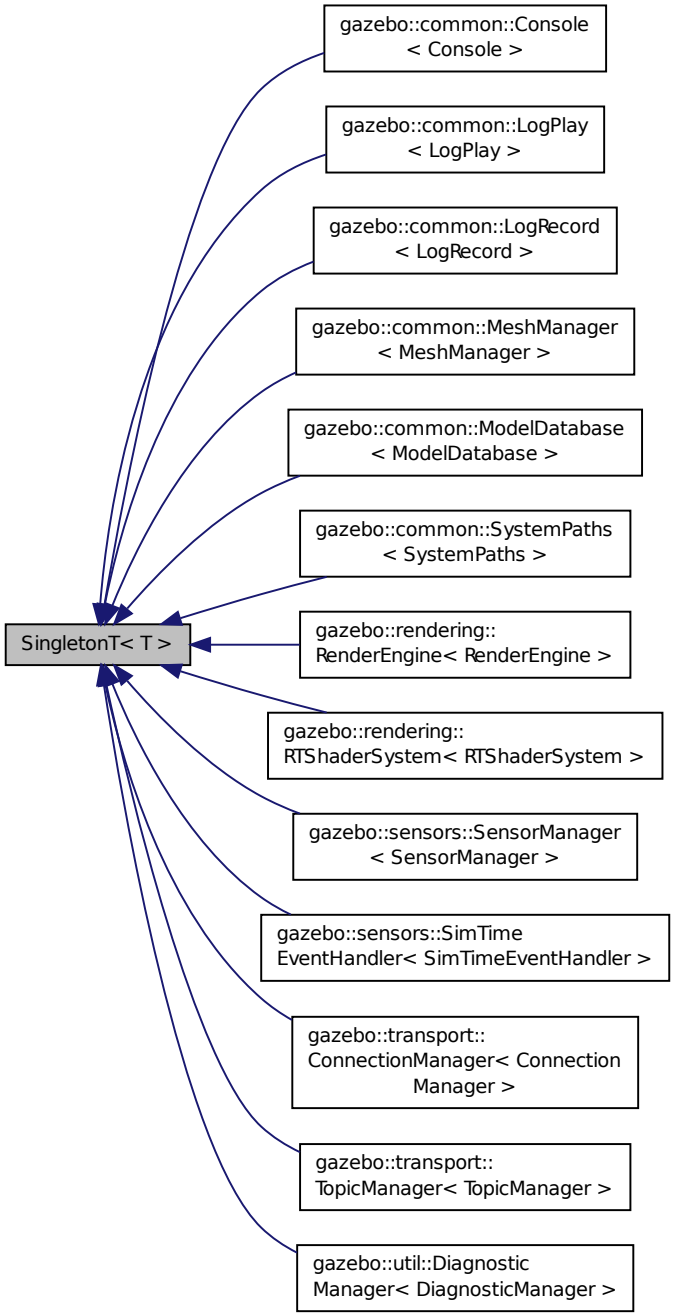
- **SensorManager.hh**

10.151 SingletonT< T > Class Template Reference

Singleton template class.

`#include <common/common.hh>`

Inheritance diagram for SingletonT< T >:



Static Public Member Functions

- static T * **Instance** ()

Get an instance of the singleton.

Protected Member Functions

- **SingletonT** ()
  Constructor.
- virtual ~**SingletonT** ()
  Destructor.

10.151.1 Detailed Description

template<class T>class SingletonT< T >

Singleton template class.

10.151.2 Constructor & Destructor Documentation

10.151.2.1 template<class T> SingletonT< T >::SingletonT ( ) [inline], [protected]

Constructor.

10.151.2.2 template<class T> virtual SingletonT< T >::~SingletonT ( ) [inline], [protected], [virtual]

Destructor.

10.151.3 Member Function Documentation

10.151.3.1 template<class T> static T* SingletonT< T >::Instance ( ) [inline], [static]

Get an instance of the singleton.

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::transport::Node::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), and gazebo::transport::Node::Subscribe().

The documentation for this class was generated from the following file:

- **SingletonT.hh**

10.152 gazebo::common::Skeleton Class Reference

A (p. 111) skeleton.

`#include <common/common.hh>`

Public Member Functions

- **Skeleton** ()
  Constructor.

- **Skeleton** (**SkeletonNode** ∗_root)

  Constructor.
- virtual ∼**Skeleton** ()

  Destructor.
- void **AddAnimation** (**SkeletonAnimation** ∗_anim)

  Add an animation.
- void **AddVertNodeWeight** (unsigned int _vertex, std::string _node, double _weight)

  Add a new weight to a node (bone)
- **SkeletonAnimation** ∗ **GetAnimation** (const unsigned int _i)

  Find animation.
- **math::Matrix4 GetBindShapeTransform** ()

  Return bind pose skeletal transform.
- **SkeletonNode** ∗ **GetNodeByHandle** (unsigned int _handle)

  Find or create node with handle.
- **SkeletonNode** ∗ **GetNodeById** (std::string _id)

  Find node by index.
- **SkeletonNode** ∗ **GetNodeByName** (std::string _name)

  Find a node.
- **NodeMap GetNodes** ()

  Get a copy or the node dictionary.
- unsigned int **GetNumAnimations** ()

  Returns the number of animations.
- unsigned int **GetNumJoints** ()

  Returns the number of joints.
- unsigned int **GetNumNodes** ()

  Returns the node count.
- unsigned int **GetNumVertNodeWeights** (unsigned int _vertex)

  Returns the number of bone weights for a vertex.
- **SkeletonNode** ∗ **GetRootNode** ()

  Return the root.
- std::pair< std::string, double > **GetVertNodeWeight** (unsigned int _v, unsigned int _i)

  Weight of a bone for a vertex.
- void **PrintTransforms** ()

  Outputs the transforms to std::err stream.
- void **Scale** (double _scale)

  Scale all nodes, transforms and animation data.
- void **SetBindShapeTransform** (**math::Matrix4** _trans)

  Set the bind pose skeletal transform.
- void **SetNumVertAttached** (unsigned int _vertices)

  Resizes the raw node weight array.
- void **SetRootNode** (**SkeletonNode** ∗_node)

  Change the root node.

Protected Member Functions

- void **BuildNodeMap** ()

  Initializes the hande numbers for each node in the map using breadth first traversal.

Protected Attributes

- std::vector< **SkeletonAnimation** * > **anims**

  the array of animations
- **math::Matrix4 bindShapeTransform**

  the bind pose skeletal transform
- **NodeMap nodes**

  The dictionary of nodes, indexed by name.
- **RawNodeWeights rawNW**

  the node weight table
- **SkeletonNode** * **root**

  the root node

10.152.1 Detailed Description

A (p. 111) skeleton.

10.152.2 Constructor & Destructor Documentation

10.152.2.1 gazebo::common::Skeleton::Skeleton ( )

Constructor.

10.152.2.2 gazebo::common::Skeleton::Skeleton (**SkeletonNode** * *_root*)

Constructor.

Parameters

in	*_root*	node

10.152.2.3 virtual gazebo::common::Skeleton::~Skeleton ( ) [virtual]

Destructor.

10.152.3 Member Function Documentation

10.152.3.1 void gazebo::common::Skeleton::AddAnimation (**SkeletonAnimation** * *_anim*)

Add an animation.

The skeleton does not take ownership of the animation

Parameters

in	*_anim*	the animation to add

10.152.3.2 void gazebo::common::Skeleton::AddVertNodeWeight (unsigned int _vertex, std::string _node, double _weight)

Add a new weight to a node (bone)

Parameters

in	*_vertex*	index of the vertex
in	*_node*	name of the bone
in	*_weight*	the new weight (range 0 to 1)

10.152.3.3 void gazebo::common::Skeleton::BuildNodeMap ( ) [protected]

Initializes the hande numbers for each node in the map using breadth first traversal.

10.152.3.4 SkeletonAnimation* gazebo::common::Skeleton::GetAnimation (const unsigned int _i)

Find animation.

Parameters

in	*_i*	the animation index

Returns

the animation, or NULL if _i is out of bounds

10.152.3.5 math::Matrix4 gazebo::common::Skeleton::GetBindShapeTransform ( )

Return bind pose skeletal transform.

Returns

a matrix

10.152.3.6 SkeletonNode* gazebo::common::Skeleton::GetNodeByHandle (unsigned int _handle)

Find or create node with handle.

Parameters

in	*_handle*	

Returns

the node. **A** (p. 111) new node is created if it didn't exist

10.152.3.7 SkeletonNode* gazebo::common::Skeleton::GetNodeById (std::string _id)

Find node by index.

Parameters

in	*_id*	the index

Returns

the node, or NULL if not found

10.152.3.8 SkeletonNode* gazebo::common::Skeleton::GetNodeByName (std::string *_name*)

Find a node.

Parameters

in	*_name*	the name of the node to look for

Returns

the node, or NULL if not found

10.152.3.9 NodeMap gazebo::common::Skeleton::GetNodes ( )

Get a copy or the node dictionary.

10.152.3.10 unsigned int gazebo::common::Skeleton::GetNumAnimations ( )

Returns the number of animations.

Returns

the count

10.152.3.11 unsigned int gazebo::common::Skeleton::GetNumJoints ( )

Returns the number of joints.

Returns

the count

10.152.3.12 unsigned int gazebo::common::Skeleton::GetNumNodes ( )

Returns the node count.

Returns

the count

10.152.3.13 unsigned int gazebo::common::Skeleton::GetNumVertNodeWeights (unsigned int *_vertex*)

Returns the number of bone weights for a vertex.

Parameters

in	*_vertex*	the index of the vertex

Returns

the count

10.152.3.14 SkeletonNode* gazebo::common::Skeleton::GetRootNode ( )

Return the root.

Returns

the root

10.152.3.15 std::pair<std::string, double> gazebo::common::Skeleton::GetVertNodeWeight (unsigned int *_v*, unsigned int *_i*)

Weight of a bone for a vertex.

Parameters

in	*_v*	the index of the vertex
in	*_i*	the index of the weight for that vertex

Returns

a pair containing the name of the node and the weight

10.152.3.16 void gazebo::common::Skeleton::PrintTransforms ( )

Outputs the transforms to std::err stream.

10.152.3.17 void gazebo::common::Skeleton::Scale (double *_scale*)

Scale all nodes, transforms and animation data.

Parameters

in	*the*	scaling factor

10.152.3.18 void gazebo::common::Skeleton::SetBindShapeTransform (math::Matrix4 *_trans*)

Set the bind pose skeletal transform.

Parameters

in	*_trans*	the transform

10.152.3.19 void gazebo::common::Skeleton::SetNumVertAttached (unsigned int *_vertices*)

Resizes the raw node weight array.

Parameters

in	*_vertices*	the new size

10.152.3.20 void gazebo::common::Skeleton::SetRootNode (SkeletonNode * *_node*)

Change the root node.

Parameters

in	*_node*	the new node

10.152.4 Member Data Documentation

10.152.4.1 std::vector<SkeletonAnimation*> gazebo::common::Skeleton::anims [protected]

the array of animations

10.152.4.2 math::Matrix4 gazebo::common::Skeleton::bindShapeTransform [protected]

the bind pose skeletal transform

10.152.4.3 NodeMap gazebo::common::Skeleton::nodes [protected]

The dictionary of nodes, indexed by name.

10.152.4.4 RawNodeWeights gazebo::common::Skeleton::rawNW [protected]

the node weight table

10.152.4.5 SkeletonNode* gazebo::common::Skeleton::root [protected]

the root node

The documentation for this class was generated from the following file:

- **Skeleton.hh**

10.153 gazebo::common::SkeletonAnimation Class Reference

Skeleton (p. 727) 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

10.153.1 Detailed Description

Skeleton (p. 727) animation.

10.153.2 Constructor & Destructor Documentation

10.153.2.1 gazebo::common::SkeletonAnimation::SkeletonAnimation (const std::string & *_name*)

The Constructor.

Parameters

in	*_name*	the name of the animation

10.153.2.2 gazebo::common::SkeletonAnimation::~SkeletonAnimation ( )

The destructor.

Clears the list without destroying the animations

10.153.3 Member Function Documentation

10.153.3.1 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	*_node*	the name of the new or existing node
in	*_time*	the time
in	*_mat*	the key frame transformation

10.153.3.2 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	*_node*	the name of the new or existing node
in	*_time*	the time
in	*_pose*	the key frame transformation as a **math::Pose** (p. 596)

10.153.3.3 double gazebo::common::SkeletonAnimation::GetLength ( ) const

Returns the duration of the animations.

Returns

the duration in seconds

10.153.3.4 std::string gazebo::common::SkeletonAnimation::GetName ( ) const

Returns the name.

Returns

the name

10.153.3.5 unsigned int gazebo::common::SkeletonAnimation::GetNodeCount ( ) const

Returns the number of animation nodes.

Returns

the count

10.153.3.6 math::Matrix4 gazebo::common::SkeletonAnimation::GetNodePoseAt (const std::string & *_node*, const double *_time*, const bool *_loop* = `true`)

Returns the key frame transformation for a named animation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.

Parameters

in	*_node*	the name of the animation node
in	*_time*	the time
in	*_loop*	when true, the time is divided by the duration (see GetLength)

Returns

the transformation

10.153.3.7 std::map<std::string, math::Matrix4> gazebo::common::SkeletonAnimation::GetPoseAt (const double *_time*, const bool *_loop* = `true`) const

Returns a dictionary of transformations indexed by name at a specific time if a node does not exist at that specific time (with tolerance of 1e-6 sec), the transformation is interpolated.

Parameters

in	*_time*	the time
in	*_loop*	when true, the time is divided by the duration (see GetLength)

Returns

the transformation for every node

10.153.3.8 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	*_x*	the value along x. You must ensure that _x is within a valid range.
in	*_node*	the name of the animation node
in	*_loop*	when true, the time is divided by the duration (see GetLength)

10.153.3.9 bool gazebo::common::SkeletonAnimation::HasNode (const std::string & _node) const

Looks for a node with a specific name in the animations.

Parameters

in	*_node*	the name of the node

Returns

true if the node exits

10.153.3.10 void gazebo::common::SkeletonAnimation::Scale (const double _scale)

Scales every animation in the animations list.

Parameters

in	*_scale*	the scaling factor

10.153.3.11 void gazebo::common::SkeletonAnimation::SetName (const std::string & _name)

Changes the name.

Parameters

in	*_name*	the new name

10.153.4 Member Data Documentation

10.153.4.1 std::map<std::string, NodeAnimation*> gazebo::common::SkeletonAnimation::animations [protected]

a dictionary of node animations

10.153.4.2 double gazebo::common::SkeletonAnimation::length [protected]

the duration of the longest animation

10.153.4.3 std::string gazebo::common::SkeletonAnimation::name [protected]

the node name

The documentation for this class was generated from the following file:

- **SkeletonAnimation.hh**

10.154 gazebo::common::SkeletonNode Class Reference

A (p. 111) skeleton node.

`#include <common/common.hh>`

Public Types

- enum **SkeletonNodeType** { **NODE**, **JOINT** }
  enumeration of node types

Public Member Functions

- **SkeletonNode** (**SkeletonNode** *_parent)
  Constructor.
- **SkeletonNode** (**SkeletonNode** *_parent, std::string _name, std::string _id, **SkeletonNodeType** _type=**JOINT**)
  Constructor.
- virtual ~**SkeletonNode** ()
  Destructor.
- void **AddChild** (**SkeletonNode** *_child)
  Add a new child.
- void **AddRawTransform** (**NodeTransform** _t)
  Add a raw transform.
- **SkeletonNode** * **GetChild** (unsigned int _index)
  Find a child by index.
- **SkeletonNode** * **GetChildById** (std::string _id)
  Get child by string id.
- **SkeletonNode** * **GetChildByName** (std::string _name)
  Get child by name.
- unsigned int **GetChildCount** ()
  Returns the children count.
- unsigned int **GetHandle** ()
  Get the handle index.
- std::string **GetId** ()
  Returns the index.

- **math::Matrix4 GetInverseBindTransform** ()

  Retrieve the inverse of the bind pose skeletal transform.
- **math::Matrix4 GetModelTransform** ()

  Retrieve the model transform.
- std::string **GetName** ()

  Returns the name.
- unsigned int **GetNumRawTrans** ()

  Return the raw transformations count.
- **SkeletonNode** ∗ **GetParent** ()

  Returns the parent node.
- **NodeTransform GetRawTransform** (unsigned int _i)

  Find a raw transformation.
- std::vector< **NodeTransform** > **GetRawTransforms** ()

  Retrieve the raw transformations.
- **math::Matrix4 GetTransform** ()

  Get transform relative to parent.
- std::vector< **NodeTransform** > **GetTransforms** ()

  Returns a copy of the array of transformations.
- bool **IsJoint** ()

  Is a joint query.
- bool **IsRootNode** ()

  Queries wether a node has no parent parent.
- void **Reset** (bool _resetChildren)

  Reset the transformation to the initial transformation.
- void **SetHandle** (unsigned int _h)

  Assign a handle number.
- void **SetId** (std::string _id)

  Change the id string.
- void **SetInitialTransform** (**math::Matrix4** _tras)

  Sets the initial transformation.
- void **SetInverseBindTransform** (**math::Matrix4** _invBM)

  Assign the inverse of the bind pose skeletal transform.
- void **SetModelTransform** (**math::Matrix4** _trans, bool _updateChildren=true)

  Set the model transformation.
- void **SetName** (std::string _name)

  Change the name.
- void **SetParent** (**SkeletonNode** ∗_parent)

  Set the parent node.
- void **SetTransform** (**math::Matrix4** _trans, bool _updateChildren=true)

  Set a transformation.
- void **SetType** (**SkeletonNodeType** _type)

  Change the skeleton node type.
- void **UpdateChildrenTransforms** ()

  Apply model transformations in order for each node in the tree.

Protected Attributes

- std::vector< **SkeletonNode** * > **children**

  the children nodes
- unsigned int **handle**

  handle index number
- std::string **id**

  a string identifier
- **math::Matrix4 initialTransform**

  the initial transformation
- **math::Matrix4 invBindTransform**

  the inverse of the bind pose skeletal transform
- **math::Matrix4 modelTransform**

  the model transformation
- std::string **name**

  the name of the skeletal node
- **SkeletonNode** * **parent**

  the parent node
- std::vector< **NodeTransform** > **rawTransforms**

  the raw transformation
- **math::Matrix4 transform**

  the transform
- **SkeletonNodeType type**

  the type fo node

10.154.1 Detailed Description

A (p. 111) skeleton node.

10.154.2 Member Enumeration Documentation

10.154.2.1 enum gazebo::common::SkeletonNode::SkeletonNodeType

enumeration of node types

Enumerator:

NODE

JOINT

10.154.3 Constructor & Destructor Documentation

10.154.3.1 gazebo::common::SkeletonNode::SkeletonNode (SkeletonNode * *_parent*)

Constructor.

Parameters

in	*_parent*	The parent node

10.154.3.2 gazebo::common::SkeletonNode::SkeletonNode (SkeletonNode * _parent, std::string _name, std::string _id, SkeletonNodeType _type = JOINT)

Constructor.

Parameters

in	*_parent*	the parent node
in	*_name*	name of node
in	*_id*	Id of node
in	*_type*	The type of this node

10.154.3.3 virtual gazebo::common::SkeletonNode::~SkeletonNode ( ) [virtual]

Destructor.

10.154.4 Member Function Documentation

10.154.4.1 void gazebo::common::SkeletonNode::AddChild (SkeletonNode * _child)

Add a new child.

Parameters

in	*_child*	a child

10.154.4.2 void gazebo::common::SkeletonNode::AddRawTransform (NodeTransform _t)

Add a raw transform.

Parameters

in	*_t*	the transform

10.154.4.3 SkeletonNode* gazebo::common::SkeletonNode::GetChild (unsigned int _index)

Find a child by index.

Parameters

in	*_index*	the index

Returns

the child skeleton. NO BOUNDS CHECKING

10.154.4.4 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

10.154.4.5 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

10.154.4.6 unsigned int gazebo::common::SkeletonNode::GetChildCount ( )

Returns the children count.

Returns

the count

10.154.4.7 unsigned int gazebo::common::SkeletonNode::GetHandle ( )

Get the handle index.

Returns

the handle index

10.154.4.8 std::string gazebo::common::SkeletonNode::GetId ( )

Returns the index.

Returns

the id string

10.154.4.9 math::Matrix4 gazebo::common::SkeletonNode::GetInverseBindTransform ( )

Retrieve the inverse of the bind pose skeletal transform.

Returns

the transform

10.154.4.10 math::Matrix4 gazebo::common::SkeletonNode::GetModelTransform ( )

Retrieve the model transform.

Returns

the transform

10.154.4.11 std::string gazebo::common::SkeletonNode::GetName ( )

Returns the name.

Returns

the name

10.154.4.12 unsigned int gazebo::common::SkeletonNode::GetNumRawTrans ( )

Return the raw transformations count.

Returns

the count

10.154.4.13 SkeletonNode* gazebo::common::SkeletonNode::GetParent ( )

Returns the parent node.

Returns

the parent

10.154.4.14 NodeTransform gazebo::common::SkeletonNode::GetRawTransform (unsigned int _i)

Find a raw transformation.

Parameters

in	_i	the index of the transformation

Returns

the node transform. NO BOUNDS CHECKING PERFORMED

10.154.4.15 std::vector<NodeTransform> gazebo::common::SkeletonNode::GetRawTransforms ( )

Retrieve the raw transformations.

Returns

an array of transformations

10.154.4.16 math::Matrix4 gazebo::common::SkeletonNode::GetTransform ( )

Get transform relative to parent.

10.154.4.17 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)

10.154.4.18 bool gazebo::common::SkeletonNode::IsJoint ( )

Is a joint query.

Returns

true if the skeleton type is a joint, false otherwise

10.154.4.19 bool gazebo::common::SkeletonNode::IsRootNode ( )

Queries wether a node has no parent parent.

Returns

true if the node has no parent, fasle otherwise

10.154.4.20 void gazebo::common::SkeletonNode::Reset (bool *_resetChildren*)

Reset the transformation to the initial transformation.

Parameters

in	*_resetChildren*	when true, performs the operation for every node in the tree

10.154.4.21 void gazebo::common::SkeletonNode::SetHandle (unsigned int *_h*)

Assign a handle number.

Parameters

in	*_h*	the handle

10.154.4.22 void gazebo::common::SkeletonNode::SetId (std::string _id)

Change the id string.

Parameters

in	*_id*	the new id string

10.154.4.23 void gazebo::common::SkeletonNode::SetInitialTransform (math::Matrix4 _tras)

Sets the initial transformation.

Parameters

in	*_tras*	the transfromation matrix

10.154.4.24 void gazebo::common::SkeletonNode::SetInverseBindTransform (math::Matrix4 _invBM)

Assign the inverse of the bind pose skeletal transform.

Parameters

in	*_invBM*	the transform

10.154.4.25 void gazebo::common::SkeletonNode::SetModelTransform (math::Matrix4 _trans, bool _updateChildren = true)

Set the model transformation.

Parameters

in	*_trans*	the transformation
in	*_updateChildren*	when true the UpdateChildrenTransforms operation is performed

10.154.4.26 void gazebo::common::SkeletonNode::SetName (std::string _name)

Change the name.

Parameters

in	*_name*	the new name

10.154.4.27 void gazebo::common::SkeletonNode::SetParent (SkeletonNode * _parent)

Set the parent node.

Parameters

in	*_parent*	the new parent

10.154.4.28 void gazebo::common::SkeletonNode::SetTransform (math::Matrix4 _trans, bool _updateChildren = true)

Set a transformation.

Parameters

in	*_trans*	the transformation
in	*_updateChildren*	when true the UpdateChildrenTransforms operation is performed

10.154.4.29 void gazebo::common::SkeletonNode::SetType (SkeletonNodeType _type)

Change the skeleton node type.

Parameters

in	*_type*	the new type

10.154.4.30 void gazebo::common::SkeletonNode::UpdateChildrenTransforms ( )

Apply model transformations in order for each node in the tree.

10.154.5 Member Data Documentation

10.154.5.1 std::vector<SkeletonNode*> gazebo::common::SkeletonNode::children [protected]

the children nodes

10.154.5.2 unsigned int gazebo::common::SkeletonNode::handle [protected]

handle index number

10.154.5.3 std::string gazebo::common::SkeletonNode::id [protected]

a string identifier

10.154.5.4 math::Matrix4 gazebo::common::SkeletonNode::initialTransform [protected]

the initial transformation

10.154.5.5 math::Matrix4 gazebo::common::SkeletonNode::invBindTransform [protected]

the inverse of the bind pose skeletal transform

10.154.5.6 math::Matrix4 gazebo::common::SkeletonNode::modelTransform [protected]

the model transformation

10.154.5.7 std::string gazebo::common::SkeletonNode::name [protected]

the name of the skeletal node

10.154.5.8 SkeletonNode* gazebo::common::SkeletonNode::parent [protected]

the parent node

10.154.5.9 std::vector<NodeTransform> gazebo::common::SkeletonNode::rawTransforms [protected]

the raw transformation

10.154.5.10 math::Matrix4 gazebo::common::SkeletonNode::transform [protected]

the transform

10.154.5.11 SkeletonNodeType gazebo::common::SkeletonNode::type [protected]

the type fo node

The documentation for this class was generated from the following file:

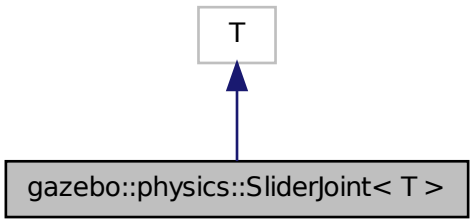
- **Skeleton.hh**

10.155 gazebo::physics::SliderJoint< T > Class Template Reference

A (p. 111) slider joint.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::SliderJoint< T >:



Public Member Functions

- **SliderJoint** (**BasePtr** _parent)

Constructor.

- virtual ~**SliderJoint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (int _index) const

  Get the anchor.
- virtual unsigned int **GetAngleCount** () const

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load a **SliderJoint** (p. 747).*
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)

  Set the anchor.

Protected Attributes

- **math::Vector3 fakeAnchor**

  The anchor value is not used internally.

10.155.1 Detailed Description

template<class T>class gazebo::physics::SliderJoint< T >

A (p. 111) slider joint.

10.155.2 Constructor & Destructor Documentation

10.155.2.1 template<class T > gazebo::physics::SliderJoint< T >::SliderJoint (BasePtr *_parent*) [inline], [explicit]

Constructor.

Parameters

in	*_parent*	Parent of the joint.

References gazebo::physics::Base::SLIDER_JOINT.

10.155.2.2 template<class T > virtual gazebo::physics::SliderJoint< T >::~SliderJoint ( ) [inline], [virtual]

Destructor.

10.155.3 Member Function Documentation

10.155.3.1 template<class T > math::Vector3 gazebo::physics::SliderJoint< T >::GetAnchor (int *_index*) const [virtual]

Get the anchor.

Parameters

in	*_index*	Index of the axis. Not used.

Returns

Anchor for the joint.

10.155.3.2 template<class T > virtual unsigned int gazebo::physics::SliderJoint< T >::GetAngleCount ( ) const [inline], [virtual]

10.155.3.3 template<class T > virtual void gazebo::physics::SliderJoint< T >::Load (sdf::ElementPtr *_sdf*) [inline], [virtual]

Load a **SliderJoint** (p. 747).

Parameters

in	*_sdf*	SDF values to load from

10.155.3.4 template<class T > void gazebo::physics::SliderJoint< T >::SetAnchor (int *_index*, const math::Vector3 & *_anchor*) [virtual]

Set the anchor.

Parameters

in	*_index*	Index of the axis. Not used.
in	*_anchor*	Anchor for the axis.

10.155.4 Member Data Documentation

10.155.4.1 template<class T > math::Vector3 gazebo::physics::SliderJoint< T >::fakeAnchor [protected]

The anchor value is not used internally.

The documentation for this class was generated from the following file:

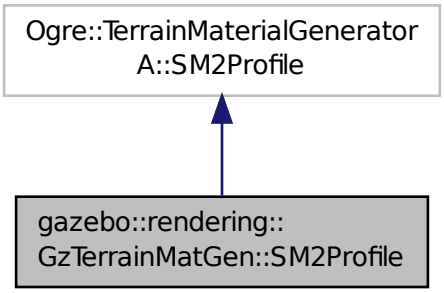
- **SliderJoint.hh**

10.156 gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference

Shader model 2 profile target.

`#include <Heightmap.hh>`

Inheritance diagram for gazebo::rendering::GzTerrainMatGen::SM2Profile:



Classes

- class **ShaderHelperCg**

  Keeping the CG shader for reference.
- class **ShaderHelperGLSL**

  Utility class to help with generating shaders for GLSL.

Public Member Functions

- **SM2Profile** (Ogre::TerrainMaterialGenerator ∗_parent, const Ogre::String &_name, const Ogre::String &_desc)

  Constructor.
- virtual ∼**SM2Profile** ()

  Destructor.
- Ogre::MaterialPtr **generate** (const Ogre::Terrain ∗_terrain)
- Ogre::MaterialPtr **generateForCompositeMap** (const Ogre::Terrain ∗_terrain)
- void **UpdateParams** (const Ogre::MaterialPtr &_mat, const Ogre::Terrain ∗_terrain)
- void **UpdateParamsForCompositeMap** (const Ogre::MaterialPtr &_mat, const Ogre::Terrain ∗_terrain)

Protected Member Functions

- virtual void **addTechnique** (const Ogre::MaterialPtr &_mat, const Ogre::Terrain ∗_terrain, TechniqueType _tt)

10.156.1 Detailed Description

Shader model 2 profile target.

10.156.2 Constructor & Destructor Documentation

10.156.2.1 gazebo::rendering::GzTerrainMatGen::SM2Profile::SM2Profile (Ogre::TerrainMaterialGenerator * *_parent,* const Ogre::String & *_name,* const Ogre::String & *_desc*)

Constructor.

10.156.2.2 virtual gazebo::rendering::GzTerrainMatGen::SM2Profile::~SM2Profile ( ) [virtual]

Destructor.

10.156.3 Member Function Documentation

10.156.3.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::addTechnique (const Ogre::MaterialPtr & *_mat,* const Ogre::Terrain * *_terrain,* TechniqueType *_tt*) [protected], [virtual]

10.156.3.2 Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generate (const Ogre::Terrain * *_terrain*)

10.156.3.3 Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generateForCompositeMap (const Ogre::Terrain * *_terrain*)

10.156.3.4 void gazebo::rendering::GzTerrainMatGen::SM2Profile::UpdateParams (const Ogre::MaterialPtr & *_mat,* const Ogre::Terrain * *_terrain*)

10.156.3.5 void gazebo::rendering::GzTerrainMatGen::SM2Profile::UpdateParamsForCompositeMap (const Ogre::MaterialPtr & *_mat,* const Ogre::Terrain * *_terrain*)

The documentation for this class was generated from the following file:

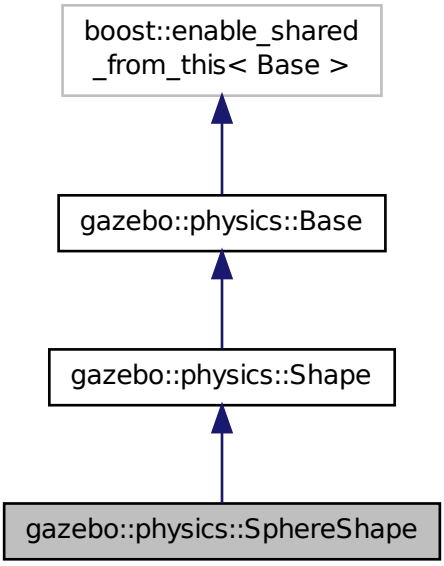
- **Heightmap.hh**

10.157 gazebo::physics::SphereShape Class Reference

Sphere collision shape.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::SphereShape:



Public Member Functions

- **SphereShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**SphereShape** ()

  Destructor.
- virtual void **FillMsg** (msgs::Geometry &_msg)

  Fill in the values for a geomertry message.
- double **GetRadius** () const

  Get the sphere's radius.
- virtual void **Init** ()

  Initialize the sphere.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  Process a geometry message.
- virtual void **SetRadius** (double _radius)

  Set the size.

Additional Inherited Members

10.157.1 Detailed Description

Sphere collision shape.

10.157.2 Constructor & Destructor Documentation

10.157.2.1 gazebo::physics::SphereShape::SphereShape (CollisionPtr _parent) [explicit]

Constructor.

Parameters

in	*_parent*	Parent collision object.

10.157.2.2 virtual gazebo::physics::SphereShape::~SphereShape ( ) [virtual]

Destructor.

10.157.3 Member Function Documentation

10.157.3.1 virtual void gazebo::physics::SphereShape::FillMsg (msgs::Geometry & _msg) [virtual]

Fill in the values for a geomertry message.

Parameters

out	*_msg*	The geometry message to fill.

Implements **gazebo::physics::Shape** (p. 722).

10.157.3.2 double gazebo::physics::SphereShape::GetRadius ( ) const

Get the sphere's radius.

Returns

Radius of the sphere.

10.157.3.3 virtual void gazebo::physics::SphereShape::Init ( ) [virtual]

Initialize the sphere.

Implements **gazebo::physics::Shape** (p. 722).

10.157.3.4 virtual void gazebo::physics::SphereShape::ProcessMsg (const msgs::Geometry & _msg) [virtual]

Process a geometry message.

Parameters

in	*_msg*	The message to set values from.

Implements **gazebo::physics::Shape** (p. 722).

10.157.3.5 virtual void gazebo::physics::SphereShape::SetRadius (double *_radius*) [virtual]

Set the size.

Parameters

in	*_radius*	Radius of the sphere.

The documentation for this class was generated from the following file:

- **SphereShape.hh**

10.158 gazebo::math::Spline Class Reference

Splines.

`#include <math/gzmath.hh>`

Public Member Functions

- **Spline** ()

  constructor
- ~**Spline** ()

  destructor
- void **AddPoint** (const **Vector3** &_pt)

  Adds a control point to the end of the spline.
- void **Clear** ()

  Clears all the points in the spline.
- **Vector3 GetPoint** (unsigned int _index) const

  Gets the detail of one of the control points of the spline.
- unsigned int **GetPointCount** () const

  Gets the number of control points in the spline.
- **Vector3 GetTangent** (unsigned int _index) const

  Get the tangent value for a point.
- double **GetTension** () const

  Get the tension value.
- **Vector3 Interpolate** (double _t) const

  Returns an interpolated point based on a parametric value over the whole series.
- **Vector3 Interpolate** (unsigned int _fromIndex, double _t) const

  Interpolates a single segment of the spline given a parametric value.
- void **RecalcTangents** ()

  Recalculates the tangents associated with this spline.
- void **SetAutoCalculate** (bool _autoCalc)

  Tells the spline whether it should automatically calculate tangents on demand as points are added.
- void **SetTension** (double _t)

  Set the tension parameter.
- void **UpdatePoint** (unsigned int _index, const **Vector3** &_value)

  Updates a single point in the spline.

Protected Attributes

- bool **autoCalc**

  when true, the tangents are recalculated when the control point change
- **Matrix4 coeffs**

  Matrix of coefficients.
- std::vector< **Vector3** > **points**

  control points
- std::vector< **Vector3** > **tangents**

  tangents
- double **tension**

  Tension of 0 = Catmull-Rom spline, otherwise a Cardinal spline.

10.158.1 Detailed Description

Splines.

10.158.2 Constructor & Destructor Documentation

10.158.2.1 gazebo::math::Spline::Spline ( )

constructor

10.158.2.2 gazebo::math::Spline::~Spline ( )

destructor

10.158.3 Member Function Documentation

10.158.3.1 void gazebo::math::Spline::AddPoint (const Vector3 & *_pt*)

Adds a control point to the end of the spline.

Parameters

in	*_pt*	point to add

10.158.3.2 void gazebo::math::Spline::Clear ( )

Clears all the points in the spline.

10.158.3.3 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

10.158.3.4 unsigned int gazebo::math::Spline::GetPointCount ( ) const

Gets the number of control points in the spline.

Returns

the count

10.158.3.5 Vector3 gazebo::math::Spline::GetTangent (unsigned int *_index*) const

Get the tangent value for a point.

Parameters

in	*_index*	the control point index

10.158.3.6 double gazebo::math::Spline::GetTension ( ) const

Get the tension value.

Returns

The value of the tension, which is between 0.0 and 1.0

10.158.3.7 Vector3 gazebo::math::Spline::Interpolate (double *_t*) const

Returns an interpolated point based on a parametric value over the whole series.

Parameters

in	*_t*	parameter (range 0 to 1)

10.158.3.8 Vector3 gazebo::math::Spline::Interpolate (unsigned int *_fromIndex,* double *_t*) const

Interpolates a single segment of the spline given a parametric value.

Parameters

in	*_fromIndex*	The point index to treat as t = 0. fromIndex + 1 is deemed to be t = 1
in	*_t*	Parametric value

10.158.3.9 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.

10.158.3.10 void gazebo::math::Spline::SetAutoCalculate (bool *_autoCalc*)

Tells the spline whether it should automatically calculate tangents on demand as points are added.

Remarks

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the recalcTangents method when you are done.

Parameters

in	*_autoCalc*	If true, tangents are calculated for you whenever a point changes. If false, you must call reclacTangents to recalculate them when it best suits.

10.158.3.11 void gazebo::math::Spline::SetTension (double *_t*)

Set the tension parameter.

A (p. 111) value of 0 = Catmull-Rom spline.

Parameters

in	*_t*	Tension value between 0.0 and 1.0

10.158.3.12 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

10.158.4 Member Data Documentation

10.158.4.1 bool gazebo::math::Spline::autoCalc [protected]

when true, the tangents are recalculated when the control point change

10.158.4.2 Matrix4 gazebo::math::Spline::coeffs [protected]

Matrix of coefficients.

10.158.4.3 std::vector<Vector3> gazebo::math::Spline::points [protected]

control points

10.158.4.4 std::vector<Vector3> gazebo::math::Spline::tangents [protected]

tangents

10.158.4.5 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:

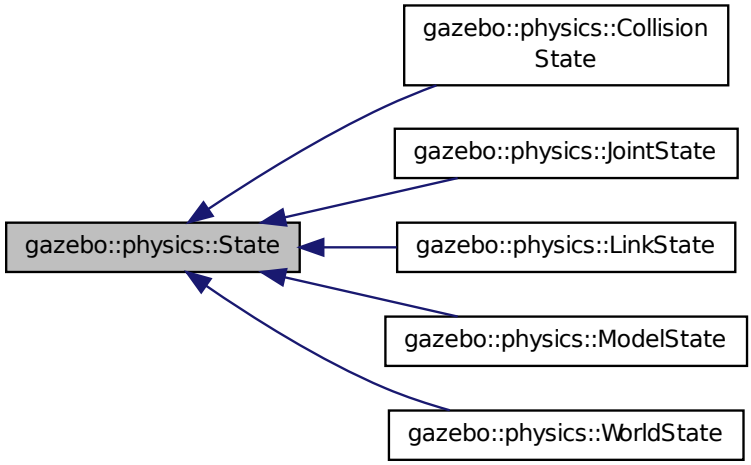
- **Spline.hh**

10.159 gazebo::physics::State Class Reference

State (p. 758) of an entity.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::State:



Public Member Functions

- **State** ()

  Default constructor.
- **State** (const std::string &_name, const **common::Time** &_realTime, const **common::Time** &_simTime)

  Constructor.
- virtual ~**State** ()

  Destructor.
- std::string **GetName** () const

  *Get the name associated with this **State** (p. 758).*
- **common::Time GetRealTime** () const

  Get the real time when this state was generated.
- **common::Time GetSimTime** () const

  Get the sim time when this state was generated.
- **common::Time GetWallTime** () const

  Get the wall time when this state was generated.
- virtual void **Load** (const **sdf::ElementPtr** _elem)

  Load state from SDF element.
- **State operator-** (const **State** &_state) const

  Subtraction operator.
- **State** & **operator=** (const **State** &_state)

  Assignment operator.
- void **SetName** (const std::string &_name)

  *Set the name associated with this **State** (p. 758).*

Protected Attributes

- std::string **name**

  *Name associated with this **State** (p. 758).*
- **common::Time realTime**
- **common::Time simTime**
- **common::Time wallTime**

  Times for the state data.

10.159.1 Detailed Description

State (p. 758) of an entity.

This is the base class for all **State** (p. 758) information.

10.159.2 Constructor & Destructor Documentation

10.159.2.1 gazebo::physics::State::State ( )

Default constructor.

10.159.2.2 gazebo::physics::State::State (const std::string & *_name,* const **common::Time** & *_realTime,* const **common::Time** & *_simTime*)

Constructor.

Construct a **State** (p. 758) object using some basic information.

Parameters

_name	Name associated with the **State** (p. 758) information. This is typically the name of an **Entity** (p. 281). _realTime Clock time since simulation started.
_simTime	Simulation time associated with this **State** (p. 758) info.

10.159.2.3 virtual gazebo::physics::State::~State ( ) [virtual]

Destructor.

10.159.3 Member Function Documentation

10.159.3.1 std::string gazebo::physics::State::GetName ( ) const

Get the name associated with this **State** (p. 758).

Returns

Name associated with this state information. Typically a name of an **Entity** (p. 281).

10.159.3.2 common::Time gazebo::physics::State::GetRealTime ( ) const

Get the real time when this state was generated.

Returns

Clock time since simulation was stated.

10.159.3.3 common::Time gazebo::physics::State::GetSimTime ( ) const

Get the sim time when this state was generated.

Returns

Simulation time when the data was recorded.

10.159.3.4 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. 758) data was recorded.

10.159.3.5 virtual void gazebo::physics::State::Load (const sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Populates the **State** (p. 758) information from data stored in an SDF::Element

Parameters

_elem	Pointer to the SDF::Element

Reimplemented in **gazebo::physics::ModelState** (p. 511), **gazebo::physics::LinkState** (p. 442), **gazebo::physics::WorldState** (p. 927), **gazebo::physics::CollisionState** (p. 207), and **gazebo::physics::JointState** (p. 404).

10.159.3.6 State gazebo::physics::State::operator- (const State & *_state*) const

Subtraction operator.

Parameters

in	*_pt*	**A** (p. 111) state to substract.

Returns

The resulting state.

10.159.3.7 State& gazebo::physics::State::operator= (const State & *_state*)

Assignment operator.

Parameters

in	*_state*	**State** (p. 758) value

Returns

this

10.159.3.8 void gazebo::physics::State::SetName (const std::string & *_name*)

Set the name associated with this **State** (p. 758).

Parameters

in	*_name*	Name associated with this state information. Typically the name of an **Entity** (p. 281).

10.159.4 Member Data Documentation

10.159.4.1 std::string gazebo::physics::State::name [protected]

Name associated with this **State** (p. 758).

10.159.4.2 common::Time gazebo::physics::State::realTime [protected]

10.159.4.3 common::Time gazebo::physics::State::simTime [protected]

10.159.4.4 common::Time gazebo::physics::State::wallTime [protected]

Times for the state data.

The documentation for this class was generated from the following file:

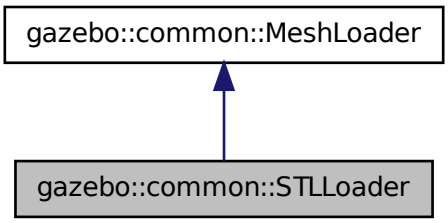
- **State.hh**

10.160 gazebo::common::STLLoader Class Reference

Class used to load STL mesh files.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::STLLoader:



Public Member Functions

- **STLLoader** ()
  Constructor.
- virtual ∼**STLLoader** ()
  Destructor.
- virtual **Mesh** ∗ **Load** (const std::string &_filename)
  Creates a new mesh and loads the data from a file.

10.160.1 Detailed Description

Class used to load STL mesh files.

10.160.2 Constructor & Destructor Documentation

10.160.2.1 gazebo::common::STLLoader::STLLoader ( )

Constructor.

10.160.2.2 virtual gazebo::common::STLLoader::∼STLLoader ( ) [virtual]

Destructor.

10.160.3 Member Function Documentation

10.160.3.1 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. 484).

The documentation for this class was generated from the following file:

- **STLLoader.hh**

10.161 gazebo::common::SubMesh Class Reference

A (p. 111) child mesh.

`#include <Mesh.hh>`

Public Types

- enum **PrimitiveType** {
  POINTS, **LINES**, **LINESTRIPS**, **TRIANGLES**,
  TRIFANS, **TRISTRIPS** }

  An enumeration of the geometric mesh primitives.

Public Member Functions

- **SubMesh** ()

  Constructor.
- **SubMesh** (const **SubMesh** ∗_mesh)

  Copy Constructor.
- virtual ∼**SubMesh** ()

  Destructor.
- void **AddIndex** (unsigned int _i)

  Add an index to the mesh.
- void **AddNodeAssignment** (unsigned int _vertex, unsigned int _node, float _weight)

  Add a vertex - skeleton node assignment.
- void **AddNormal** (const **math::Vector3** &_n)

  Add a normal to the mesh.
- void **AddNormal** (double _x, double _y, double _z)

  Add a normal to the mesh.
- void **AddTexCoord** (double _u, double _v)

  Add a texture coord to the mesh.
- void **AddVertex** (const **math::Vector3** &_v)

  Add a vertex to the mesh.
- void **AddVertex** (double _x, double _y, double _z)

  Add a vertex to the mesh.
- void **Center** (const **math::Vector3** &_center=**math::Vector3::Zero**)

  Move the center of the submesh to the given coordinate.
- void **CopyNormals** (const std::vector< **math::Vector3** > &_norms)

  Copy normals from a vector.
- void **CopyVertices** (const std::vector< **math::Vector3** > &_verts)

  Copy vertices from a vector.
- void **FillArrays** (float ∗∗_vertArr, int ∗∗_indArr) const

Put all the data into flat arrays.

- void **GenSphericalTexCoord** (const **math::Vector3** &_center)

  Generate texture coordinates using spherical projection from center.
- unsigned int **GetIndex** (unsigned int _i) const

  Get an index.
- unsigned int **GetIndexCount** () const

  Return the number of indicies.
- unsigned int **GetMaterialIndex** () const

  Get the material index.
- **math::Vector3 GetMax** () const

  Get the maximun X, Y, Z values.
- unsigned int **GetMaxIndex** () const

  Get the highest index value.
- **math::Vector3 GetMin** () const

  Get the minimum X, Y, Z values.
- std::string **GetName** () const

  Get the name of this mesh.
- **NodeAssignment GetNodeAssignment** (unsigned int _i) const

  Get a vertex - skeleton node assignment.
- unsigned int **GetNodeAssignmentsCount** () const

  Return the number of vertex - skeleton node assignments.
- **math::Vector3 GetNormal** (unsigned int _i) const

  Get a normal.
- unsigned int **GetNormalCount** () const

  Return the number of normals.
- **PrimitiveType GetPrimitiveType** () const

  Get the primitive type.
- **math::Vector2d GetTexCoord** (unsigned int _i) const

  Get a tex coord.
- unsigned int **GetTexCoordCount** () const

  Return the number of texture coordinates.
- **math::Vector3 GetVertex** (unsigned int _i) const

  Get a vertex.
- unsigned int **GetVertexCount** () const

  Return the number of vertices.
- unsigned int **GetVertexIndex** (const **math::Vector3** &_v) const

  Get the index of the vertex.
- bool **HasVertex** (const **math::Vector3** &_v) const

  Return true if this submesh has the vertex.
- void **RecalculateNormals** ()

  Recalculate all the normals.
- void **Scale** (double _factor)

  Scale all vertices by _factor.
- void **SetIndexCount** (unsigned int _count)

  Resize the index array.
- void **SetMaterialIndex** (unsigned int _index)

  Set the material index.

- void **SetName** (const std::string &_n)

  Set the name of this mesh.
- void **SetNormal** (unsigned int _i, const **math::Vector3** &_n)

  Set a normal.
- void **SetNormalCount** (unsigned int _count)

  Resize the normal array.
- void **SetPrimitiveType** (**PrimitiveType** _type)

  Set the primitive type.
- void **SetScale** (const **math::Vector3** &_factor)

  Scale all vertices by the _factor vector.
- void **SetSubMeshCenter** (**math::Vector3** _center)

  Reset mesh center to geometric center.
- void **SetTexCoord** (unsigned int _i, const **math::Vector2d** &_t)

  Set a tex coord.
- void **SetTexCoordCount** (unsigned int _count)

  Resize the texture coordinate array.
- void **SetVertex** (unsigned int _i, const **math::Vector3** &_v)

  Set a vertex.
- void **SetVertexCount** (unsigned int _count)

  Resize the vertex array.
- void **Translate** (const **math::Vector3** &_vec)

  Move all vertices by _vec.

10.161.1 Detailed Description

A (p. 111) child mesh.

10.161.2 Member Enumeration Documentation

10.161.2.1 enum gazebo::common::SubMesh::PrimitiveType

An enumeration of the geometric mesh primitives.

Enumerator:

POINTS

LINES

LINESTRIPS

TRIANGLES

TRIFANS

TRISTRIPS

10.161.3 Constructor & Destructor Documentation

10.161.3.1 gazebo::common::SubMesh::SubMesh ( )

Constructor.

10.161.3.2 gazebo::common::SubMesh::SubMesh (const SubMesh * _mesh)

Copy Constructor.

10.161.3.3 virtual gazebo::common::SubMesh::~SubMesh ( ) [virtual]

Destructor.

10.161.4 Member Function Documentation

10.161.4.1 void gazebo::common::SubMesh::AddIndex (unsigned int _i)

Add an index to the mesh.

Parameters

in	*_i*	the new vertex index

10.161.4.2 void gazebo::common::SubMesh::AddNodeAssignment (unsigned int _vertex, unsigned int _node, float _weight)

Add a vertex - skeleton node assignment.

Parameters

in	*_vertex*	the vertex index
in	*_node*	the node index
in	*_weight*	the weight (between 0 and 1)

10.161.4.3 void gazebo::common::SubMesh::AddNormal (const math::Vector3 & _n)

Add a normal to the mesh.

Parameters

in	*_n*	the normal

10.161.4.4 void gazebo::common::SubMesh::AddNormal (double _x, double _y, double _z)

Add a normal to the mesh.

Parameters

in	*_x*	position along x
in	*_y*	position along y
in	*_z*	position along z

10.161.4.5 void gazebo::common::SubMesh::AddTexCoord (double _u, double _v)

Add a texture coord to the mesh.

Parameters

in	_u	position along u
in	_v	position along v

10.161.4.6 void gazebo::common::SubMesh::AddVertex (const math::Vector3 & _v)

Add a vertex to the mesh.

Parameters

in	_v	the new position

10.161.4.7 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

10.161.4.8 void gazebo::common::SubMesh::Center (const math::Vector3 & _center = math::Vector3::Zero)

Move the center of the submesh to the given coordinate.

This will move all the vertices.

Parameters

in	_center	Location of the mesh center.

10.161.4.9 void gazebo::common::SubMesh::CopyNormals (const std::vector< math::Vector3 > & _norms)

Copy normals from a vector.

Parameters

in	_norms	to copy from

10.161.4.10 void gazebo::common::SubMesh::CopyVertices (const std::vector< math::Vector3 > & _verts)

Copy vertices from a vector.

Parameters

in	*_verts*	the vertices to copy from

10.161.4.11 void gazebo::common::SubMesh::FillArrays (float ** *_vertArr*, int ** *_indArr*) const

Put all the data into flat arrays.

Parameters

in	*_verArr*	
in	*_indArr*	

10.161.4.12 void gazebo::common::SubMesh::GenSphericalTexCoord (const math::Vector3 & *_center*)

Generate texture coordinates using spherical projection from center.

Parameters

in	*_center*	

10.161.4.13 unsigned int gazebo::common::SubMesh::GetIndex (unsigned int *_i*) const

Get an index.

Parameters

in	*_i*	

10.161.4.14 unsigned int gazebo::common::SubMesh::GetIndexCount ( ) const

Return the number of indicies.

10.161.4.15 unsigned int gazebo::common::SubMesh::GetMaterialIndex ( ) const

Get the material index.

10.161.4.16 math::Vector3 gazebo::common::SubMesh::GetMax ( ) const

Get the maximun X, Y, Z values.

Returns

10.161.4.17 unsigned int gazebo::common::SubMesh::GetMaxIndex ( ) const

Get the highest index value.

10.161.4.18 math::Vector3 gazebo::common::SubMesh::GetMin ( ) const

Get the minimum X, Y, Z values.

Returns

10.161.4.19 std::string gazebo::common::SubMesh::GetName ( ) const

Get the name of this mesh.

Returns

the name

10.161.4.20 NodeAssignment gazebo::common::SubMesh::GetNodeAssignment (unsigned int _i) const

Get a vertex - skeleton node assignment.

Parameters

in	_i	the index of the assignment

10.161.4.21 unsigned int gazebo::common::SubMesh::GetNodeAssignmentsCount ( ) const

Return the number of vertex - skeleton node assignments.

10.161.4.22 math::Vector3 gazebo::common::SubMesh::GetNormal (unsigned int _i) const

Get a normal.

Parameters

in	_i	the normal index

Returns

the orientation of the normal, or throws an exception

10.161.4.23 unsigned int gazebo::common::SubMesh::GetNormalCount ( ) const

Return the number of normals.

10.161.4.24 PrimitiveType gazebo::common::SubMesh::GetPrimitiveType ( ) const

Get the primitive type.

Returns

the primitive type

10.161.4.25 math::Vector2d gazebo::common::SubMesh::GetTexCoord (unsigned int _i) const

Get a tex coord.

Parameters

in	_i	the texture index

Returns

the texture coordinates

10.161.4.26 unsigned int gazebo::common::SubMesh::GetTexCoordCount ( ) const

Return the number of texture coordinates.

10.161.4.27 math::Vector3 gazebo::common::SubMesh::GetVertex (unsigned int _i) const

Get a vertex.

Parameters

in	_i	the vertex index

Returns

the position or throws an exception

10.161.4.28 unsigned int gazebo::common::SubMesh::GetVertexCount ( ) const

Return the number of vertices.

10.161.4.29 unsigned int gazebo::common::SubMesh::GetVertexIndex (const math::Vector3 & _v) const

Get the index of the vertex.

Parameters

in	_v	

10.161.4.30 bool gazebo::common::SubMesh::HasVertex (const math::Vector3 & _v) const

Return true if this submesh has the vertex.

Parameters

in	*_v*	

10.161.4.31 void gazebo::common::SubMesh::RecalculateNormals ( )

Recalculate all the normals.

10.161.4.32 void gazebo::common::SubMesh::Scale (double *_factor*)

Scale all vertices by _factor.

Parameters

in	*_factor*	Scaling factor

10.161.4.33 void gazebo::common::SubMesh::SetIndexCount (unsigned int *_count*)

Resize the index array.

Parameters

in	*_count*	the new size of the array

10.161.4.34 void gazebo::common::SubMesh::SetMaterialIndex (unsigned int *_index*)

Set the material index.

Relates to the parent mesh material list

Parameters

in	*_index*	

10.161.4.35 void gazebo::common::SubMesh::SetName (const std::string & *_n*)

Set the name of this mesh.

Parameters

in	*_n*	the name to set

10.161.4.36 void gazebo::common::SubMesh::SetNormal (unsigned int *_i*, const math::Vector3 & *_n*)

Set a normal.

Parameters

in	*_i*	the normal index
in	*_n*	the normal direction

10.161.4.37 void gazebo::common::SubMesh::SetNormalCount (unsigned int *_count*)

Resize the normal array.

Parameters

in	*_count*	the new size of the array

10.161.4.38 void gazebo::common::SubMesh::SetPrimitiveType (PrimitiveType *_type*)

Set the primitive type.

Parameters

in	*_type*	the type

10.161.4.39 void gazebo::common::SubMesh::SetScale (const math::Vector3 & *_factor*)

Scale all vertices by the _factor vector.

Parameters

in	*_factor*	Scaling vector

10.161.4.40 void gazebo::common::SubMesh::SetSubMeshCenter (math::Vector3 *_center*)

Reset mesh center to geometric center.

Parameters

in	*_center*	

10.161.4.41 void gazebo::common::SubMesh::SetTexCoord (unsigned int *_i,* const math::Vector2d & *_t*)

Set a tex coord.

Parameters

in	*_i*	
in	*_t*	

10.161.4.42 void gazebo::common::SubMesh::SetTexCoordCount (unsigned int _count)

Resize the texture coordinate array.

Parameters

in	_count	

10.161.4.43 void gazebo::common::SubMesh::SetVertex (unsigned int _i, const math::Vector3 & _v)

Set a vertex.

Parameters

in	_i	the index
in	_v	the position

10.161.4.44 void gazebo::common::SubMesh::SetVertexCount (unsigned int _count)

Resize the vertex array.

Parameters

in	_count	the new size of the array

10.161.4.45 void gazebo::common::SubMesh::Translate (const math::Vector3 & _vec)

Move all vertices by _vec.

Parameters

in	_vec	Amount to translate vertices.

The documentation for this class was generated from the following file:

- **Mesh.hh**

10.162 gazebo::transport::SubscribeOptions Class Reference

Options for a subscription.

`#include <transport/transport.hh>`

Public Member Functions

- **SubscribeOptions** ()

  Constructor.
- bool **GetLatching** () const

Are we latching?

- std::string **GetMsgType** () const

  Get the type of the topic we're subscribed to.
- **NodePtr GetNode** () const

  Get the node we're subscribed to.
- std::string **GetTopic** () const

  Get the topic we're subscribed to.
- template<class M >
  void **Init** (const std::string &_topic, **NodePtr** _node, bool _latching)

  Initialize the options.
- void **Init** (const std::string &_topic, **NodePtr** _node, bool _latching)

  Initialize the options.

10.162.1 Detailed Description

Options for a subscription.

10.162.2 Constructor & Destructor Documentation

10.162.2.1 gazebo::transport::SubscribeOptions::SubscribeOptions ( ) [inline]

Constructor.

10.162.3 Member Function Documentation

10.162.3.1 bool gazebo::transport::SubscribeOptions::GetLatching ( ) const [inline]

Are we latching?

Returns

true if we're latching the latest message, false otherwise

10.162.3.2 std::string gazebo::transport::SubscribeOptions::GetMsgType ( ) const [inline]

Get the type of the topic we're subscribed to.

Returns

The type of the topic we're subscribed to

10.162.3.3 NodePtr gazebo::transport::SubscribeOptions::GetNode ( ) const [inline]

Get the node we're subscribed to.

Returns

The associated node

10.162.3.4 std::string gazebo::transport::SubscribeOptions::GetTopic ( ) const [inline]

Get the topic we're subscribed to.

Returns

The topic we're subscribed to

10.162.3.5 template<class M > void gazebo::transport::SubscribeOptions::Init (const std::string & *_topic,* NodePtr *_node,* bool *_latching*) [inline]

Initialize the options.

Parameters

in	*_topic*	Topic we're subscribing to
in,out	*_node*	The associated node
in	*_latching*	If true, latch the latest message; if false, don't latch

References gzthrow, and NULL.

Referenced by gazebo::transport::Node::Subscribe().

10.162.3.6 void gazebo::transport::SubscribeOptions::Init (const std::string & *_topic,* NodePtr *_node,* bool *_latching*) [inline]

Initialize the options.

This version of init is only used when creating subscribers of raw data.

Parameters

in	*_topic*	Topic we're subscribing to
in,out	*_node*	The associated node
in	*_latching*	If true, latch the latest message; if false, don't latch

The documentation for this class was generated from the following file:

- **SubscribeOptions.hh**

10.163 gazebo::transport::Subscriber Class Reference

A (p. 111) subscriber to a topic.

`#include <transport/transport.hh>`

Public Member Functions

- **Subscriber** (const std::string &_topic, **NodePtr** _node)

  Constructor.
- virtual ~**Subscriber** ()

Destructor.

- unsigned int **GetCallbackId** () const
- std::string **GetTopic** () const

  Get the topic name.
- void **SetCallbackId** (unsigned int _id)
- void **Unsubscribe** () const

  Unsubscribe from the topic.

10.163.1 Detailed Description

A (p. 111) subscriber to a topic.

10.163.2 Constructor & Destructor Documentation

10.163.2.1 gazebo::transport::Subscriber::Subscriber (const std::string & *_topic,* NodePtr *_node*)

Constructor.

Parameters

in	*_topic*	The topic we're subscribing to
in	*_node*	The associated node

10.163.2.2 virtual gazebo::transport::Subscriber::~Subscriber ( ) [virtual]

Destructor.

10.163.3 Member Function Documentation

10.163.3.1 unsigned int gazebo::transport::Subscriber::GetCallbackId ( ) const

10.163.3.2 std::string gazebo::transport::Subscriber::GetTopic ( ) const

Get the topic name.

Returns

The topic name

10.163.3.3 void gazebo::transport::Subscriber::SetCallbackId (unsigned int *_id*)

Referenced by gazebo::transport::Node::Subscribe().

10.163.3.4 void gazebo::transport::Subscriber::Unsubscribe ( ) const

Unsubscribe from the topic.

The documentation for this class was generated from the following file:

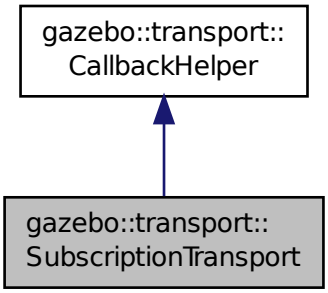
- **Subscriber.hh**

10.164 gazebo::transport::SubscriptionTransport Class Reference

transport/transport.hh

`#include <SubscriptionTransport.hh>`

Inheritance diagram for gazebo::transport::SubscriptionTransport:



Public Member Functions

- **SubscriptionTransport** ()

  Constructor.
- virtual ~**SubscriptionTransport** ()

  Destructor.
- const **ConnectionPtr** & **GetConnection** () const

  Get the connection we're using.
- virtual bool **HandleData** (const std::string &_newdata)

  Output a message to a connection.
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)

  Process new incoming message.
- void **Init** (const **ConnectionPtr** &_conn, bool _latching)

  Initialize the publication link.
- virtual bool **IsLocal** () const

  Is the callback local?

Additional Inherited Members

10.164.1 Detailed Description

transport/transport.hh

Handles sending data over the wire to remote subscribers

10.164.2 Constructor & Destructor Documentation

10.164.2.1 gazebo::transport::SubscriptionTransport::SubscriptionTransport ( )

Constructor.

10.164.2.2 virtual gazebo::transport::SubscriptionTransport::~SubscriptionTransport ( ) [virtual]

Destructor.

10.164.3 Member Function Documentation

10.164.3.1 const ConnectionPtr& gazebo::transport::SubscriptionTransport::GetConnection ( ) const

Get the connection we're using.

Returns

Pointer to the connection we're using

10.164.3.2 virtual bool gazebo::transport::SubscriptionTransport::HandleData (const std::string & *_newdata*) [virtual]

Output a message to a connection.

Parameters

in	*_newdata*	The message to be handled

Returns

true if the message was handled successfully, false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 159).

10.164.3.3 virtual bool gazebo::transport::SubscriptionTransport::HandleMessage (MessagePtr *_newMsg*) [virtual]

Process new incoming message.

Parameters

in	*_newMsg*	Incoming message to be processed

Returns

true if successfully processed; false otherwise

Implements **gazebo::transport::CallbackHelper** (p. 159).

10.164.3.4 void gazebo::transport::SubscriptionTransport::Init (const ConnectionPtr & *_conn,* bool *_latching*)

Initialize the publication link.

Parameters

in	*_conn*	The connection to use
in	*_latching*	If true, latch the latest message; if false, don't latch

10.164.3.5 virtual bool gazebo::transport::SubscriptionTransport::IsLocal ( ) const [virtual]

Is the callback local?

Returns

true if the callback is local, false if the callback is tied to a remote connection

Implements **gazebo::transport::CallbackHelper** (p. 159).

The documentation for this class was generated from the following file:

- **SubscriptionTransport.hh**

10.165 gazebo::physics::SurfaceParams Class Reference

SurfaceParams (p. 780) defines various Surface contact parameters.

`#include <physics/physics.hh>`

Public Member Functions

- **SurfaceParams** ()
  Constructor.
- virtual ~**SurfaceParams** ()
  Destructor.
- void **FillMsg** (msgs::Surface &_msg)
  Fill in a surface message.
- virtual void **Load** (**sdf::ElementPtr** _sdf)
  Load the contact params.
- virtual void **ProcessMsg** (const msgs::Surface &_msg)

Public Attributes

- double **bounce**
  bounce restitution coefficient [0,1], with 0 being inelastic, and 1 being perfectly elastic.
- double **bounceThreshold**
  minimum contact velocity for bounce to take effect, otherwise the collision is treated as an inelastic collision.
- double **cfm**
  Constraint Force Mixing parameter.

- double **erp**

  Error Reduction Parameter.
- **math::Vector3 fdir1**

  *Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 783)) of the friction pyramid.*
- double **kd**

  *spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 782) and **SurfaceParams::erp** (p. 782).*
- double **kp**

  *spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 782) and **SurfaceParams::erp** (p. 782).*
- double **maxVel**

  Maximum interpenetration error correction velocity.
- double **minDepth**

  Minimum depth before ERP takes effect.
- double **mu1**

  Dry friction coefficient in the primary friction direction as defined by the friction pyramid.
- double **mu2**

  Dry friction coefficient in the second friction direction as defined by the friction pyramid.
- double **slip1**

  Artificial contact slip in the primary friction direction.
- double **slip2**

  Artificial contact slip in the secondary friction dirction.

10.165.1 Detailed Description

SurfaceParams (p. 780) defines various Surface contact parameters.

These parameters defines the properties of a **physics::Contact** (p. 236) constraint.

10.165.2 Constructor & Destructor Documentation

10.165.2.1 gazebo::physics::SurfaceParams::SurfaceParams ( )

Constructor.

10.165.2.2 virtual gazebo::physics::SurfaceParams::~SurfaceParams ( ) [virtual]

Destructor.

10.165.3 Member Function Documentation

10.165.3.1 void gazebo::physics::SurfaceParams::FillMsg (msgs::Surface & *_msg*)

Fill in a surface message.

Parameters

in	*_msg*	Message to fill with this object's values.

10.165.3.2 virtual void gazebo::physics::SurfaceParams::Load (sdf::ElementPtr _sdf) [virtual]

Load the contact params.

Parameters

in	*_sdf*	SDF values to load from.

10.165.3.3 virtual void gazebo::physics::SurfaceParams::ProcessMsg (const msgs::Surface & _msg) [virtual]

10.165.4 Member Data Documentation

10.165.4.1 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

10.165.4.2 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

10.165.4.3 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.

10.165.4.4 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.

10.165.4.5 math::Vector3 gazebo::physics::SurfaceParams::fdir1

Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 783)) of the friction pyramid.

```
If undefined, a vector consstrained 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

10.165.4.6 double gazebo::physics::SurfaceParams::kd

spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 782) and **SurfaceParams::erp** (p. 782).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.165.4.7 double gazebo::physics::SurfaceParams::kp

spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 782) and **SurfaceParams::erp** (p. 782).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.165.4.8 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)

10.165.4.9 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)

10.165.4.10 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 consstrained 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

10.165.4.11 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 consstrained 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

10.165.4.12 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

10.165.4.13 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:

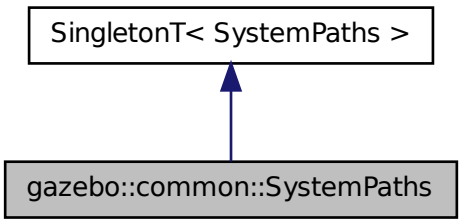
- **SurfaceParams.hh**

10.166 gazebo::common::SystemPaths Class Reference

Functions to handle getting system paths, keeps track of:

#include <common/common.hh>

Inheritance diagram for gazebo::common::SystemPaths:



Public Member Functions

- void **AddGazeboPaths** (const std::string &_path)

  Add colon delimited paths to Gazebo install.
- void **AddModelPaths** (const std::string &_path)

  Add colon delimited paths to modelPaths.
- void **AddOgrePaths** (const std::string &_path)

  Add colon delimited paths to ogre install.
- void **AddPluginPaths** (const std::string &_path)

  Add colon delimited paths to plugins.
- void **AddSearchPathSuffix** (const std::string &_suffix)

  add _suffix to the list of path search suffixes
- void **ClearGazeboPaths** ()

  clear out SystemPaths::gazeboPaths
- void **ClearModelPaths** ()

  clear out SystemPaths::modelPaths
- void **ClearOgrePaths** ()

  clear out SystemPaths::ogrePaths
- void **ClearPluginPaths** ()

  clear out SystemPaths::pluginPaths
- std::string **FindFile** (const std::string &_filename, bool _searchLocalPath=true)

  Find a file in the gazebo paths.
- std::string **FindFileURI** (const std::string &_uri)

  Find a file or path using a URI.
- const std::list< std::string > & **GetGazeboPaths** ()

  Get the gazebo install paths.
- std::string **GetLogPath** () const

  Get the log path.
- const std::list< std::string > & **GetModelPaths** ()

  Get the model paths.
- const std::list< std::string > & **GetOgrePaths** ()

Get the ogre install paths.

- const std::list< std::string > & **GetPluginPaths** ()

  Get the plugin paths.

- std::string **GetWorldPathExtension** ()

  Returns the world path extension.

Public Attributes

- bool **gazeboPathsFromEnv**

  *if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 788)*

- bool **modelPathsFromEnv**

  *if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 788)*

- bool **ogrePathsFromEnv**

  *if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 788)*

- bool **pluginPathsFromEnv**

  *if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 789)*

Additional Inherited Members

10.166.1 Detailed Description

Functions to handle getting system paths, keeps track of:

- SystemPaths::gazeboPaths - media paths containing worlds, models, sdf descriptions, material scripts, textures.
- SystemPaths::ogrePaths - ogre library paths. Should point to **Ogre** (p. 106) RenderSystem_GL.so et. al.
- SystemPaths::pluginPaths - plugin library paths for common::WorldPlugin

10.166.2 Member Function Documentation

10.166.2.1 void gazebo::common::SystemPaths::AddGazeboPaths (const std::string & *_path*)

Add colon delimited paths to Gazebo install.

Parameters

in	*_path*	the directory to add

10.166.2.2 void gazebo::common::SystemPaths::AddModelPaths (const std::string & *_path*)

Add colon delimited paths to modelPaths.

Parameters

in	*_path*	the directory to add

10.166.2.3 void gazebo::common::SystemPaths::AddOgrePaths (const std::string & _path)

Add colon delimited paths to ogre install.

Parameters

in	*_path*	the directory to add

10.166.2.4 void gazebo::common::SystemPaths::AddPluginPaths (const std::string & _path)

Add colon delimited paths to plugins.

Parameters

in	*_path*	the directory to add

10.166.2.5 void gazebo::common::SystemPaths::AddSearchPathSuffix (const std::string & _suffix)

add _suffix to the list of path search suffixes

Parameters

in	*_suffix*	The suffix to add

10.166.2.6 void gazebo::common::SystemPaths::ClearGazeboPaths ( )

clear out SystemPaths::gazeboPaths

10.166.2.7 void gazebo::common::SystemPaths::ClearModelPaths ( )

clear out SystemPaths::modelPaths

10.166.2.8 void gazebo::common::SystemPaths::ClearOgrePaths ( )

clear out SystemPaths::ogrePaths

10.166.2.9 void gazebo::common::SystemPaths::ClearPluginPaths ( )

clear out SystemPaths::pluginPaths

10.166.2.10 std::string gazebo::common::SystemPaths::FindFile (const std::string & _filename, bool _searchLocalPath = true)

Find a file in the gazebo paths.

Parameters

in	*_filename*	Name of the file to find.
in	*_searchLocal-Path*	True to search in the current working directory.

Returns

Returns full path name to file

10.166.2.11 std::string gazebo::common::SystemPaths::FindFileURI (const std::string & *_uri*)

Find a file or path using a URI.

Parameters

in	*_uri*	the uniform resource identifier

Returns

Returns full path name to file

10.166.2.12 const std::list<std::string>& gazebo::common::SystemPaths::GetGazeboPaths ( )

Get the gazebo install paths.

Returns

a list of paths

10.166.2.13 std::string gazebo::common::SystemPaths::GetLogPath ( ) const

Get the log path.

Returns

the path

10.166.2.14 const std::list<std::string>& gazebo::common::SystemPaths::GetModelPaths ( )

Get the model paths.

Returns

a list of paths

10.166.2.15 const std::list<std::string>& gazebo::common::SystemPaths::GetOgrePaths ( )

Get the ogre install paths.

Returns

a list of paths

10.166.2.16 const std::list<std::string>& gazebo::common::SystemPaths::GetPluginPaths ( )

Get the plugin paths.

Returns

a list of paths

10.166.2.17 std::string gazebo::common::SystemPaths::GetWorldPathExtension ( )

Returns the world path extension.

Returns

Right now, it just returns "/worlds"

10.166.3 Member Data Documentation

10.166.3.1 bool gazebo::common::SystemPaths::gazeboPathsFromEnv

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 788)

10.166.3.2 bool gazebo::common::SystemPaths::modelPathsFromEnv

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 788)

10.166.3.3 bool gazebo::common::SystemPaths::ogrePathsFromEnv

if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 788)

10.166.3.4 bool gazebo::common::SystemPaths::pluginPathsFromEnv

if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 789)

The documentation for this class was generated from the following file:

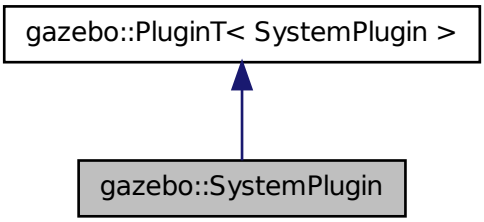
- **SystemPaths.hh**

10.167 gazebo::SystemPlugin Class Reference

A (p. 111) 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

10.167.1 Detailed Description

A (p. 111) plugin loaded within the gzserver on startup.

See `reference`.

Todo how to make doxygen reference to the file gazebo.cc::g_plugins?

10.167.2 Constructor & Destructor Documentation

10.167.2.1 gazebo::SystemPlugin::SystemPlugin ( ) [inline]

Constructor.

References gazebo::SYSTEM_PLUGIN, and gazebo::PluginT< SystemPlugin >::type.

10.167.2.2 virtual gazebo::SystemPlugin::∼SystemPlugin ( ) [inline], [virtual]

Destructor.

10.167.3 Member Function Documentation

10.167.3.1 virtual void gazebo::SystemPlugin::Init ( ) [inline], [virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.167.3.2 virtual void gazebo::SystemPlugin::Load (int *_argc* = 0, char ** *_argv* = NULL) [pure virtual]

Load function.

Called before Gazebo is loaded. Must not block.

Parameters

_argc	Number of command line arguments.
_argv	Array of command line arguments.

10.167.3.3 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**

10.168 gazebo::common::Time Class Reference

A (p. 111) **Time** (p. 791) class, can be used to hold wall- or sim-time.

`#include <common/common.hh>`

Public Member Functions

- **Time** ()
  Constructors.
- **Time** (const **Time** &_time)
  Copy constructor.
- **Time** (const struct timeval &_tv)
  Constructor.
- **Time** (const struct timespec &_tv)
  Constructor.
- **Time** (int32_t _sec, int32_t _nsec)
  Constructor.
- **Time** (double _time)
  Constuctor.
- 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 assignemtn operator.
- **Time operator-** (const struct timeval &_tv) const

  Subtraction operator.
- **Time operator-** (const struct timespec &_tv) const

  Subtraction operator.
- **Time operator-** (const **Time** &_time) const

  Subtraction operator.
- const **Time** & **operator-=** (const struct timeval &_tv)

  Subtraction assignment operator.
- const **Time** & **operator-=** (const struct timespec &_tv)

  Subtraction assignment operator.
- const **Time** & **operator-=** (const **Time** &_time)

Subtraction assignment operator.

- **Time operator**/ (const struct timeval &_tv) const

  Division operator.
- **Time operator**/ (const struct timespec &_tv) const

  Division operator.
- **Time operator**/ (const **Time** &_time) const

  Division operator.
- const **Time** & **operator/=** (const struct timeval &_tv)

  Division assignment operator.
- const **Time** & **operator/=** (const struct timespec &_tv)

  Division assignment operator.
- const **Time** & **operator/=** (const **Time** &time)

  Division assignment operator.
- bool **operator**< (const struct timeval &_tv) const

  Less than operator.
- bool **operator**< (const struct timespec &_tv) const

  Less than operator.
- bool **operator**< (const **Time** &_time) const

  Less than operator.
- bool **operator**< (double _time) const

  Less than operator.
- bool **operator**<**=** (const struct timeval &_tv) const

  Less than or equal to operator.
- bool **operator**<**=** (const struct timespec &_tv) const

  Less than or equal to operator.
- bool **operator**<**=** (const **Time** &_time) const

  Less than or equal to operator.
- bool **operator**<**=** (double _time) const

  Less than or equal to operator.
- **Time** & **operator=** (const struct timeval &_tv)

  Assignment operator.
- **Time** & **operator=** (const struct timespec &_tv)

  Assignment operator.
- **Time** & **operator=** (const **Time** &_time)

  Assignment operator.
- bool **operator==** (const struct timeval &_tv) const

  Equal to operator.
- bool **operator==** (const struct timespec &_tv) const

  Equal to operator.
- bool **operator==** (const **Time** &_time) const

  Equal to operator.
- bool **operator==** (double _time) const

  Equal to operator.
- bool **operator**> (const struct timeval &_tv) const

  Greater than operator.
- bool **operator**> (const struct timespec &_tv) const

  Greater than operator.

- bool **operator**> (const **Time** &_time) const

  Greater than operator.
- bool **operator**> (double _time) const

  Greater than operator.
- bool **operator**>= (const struct timeval &_tv) const

  Greater than or equal operator.
- bool **operator**>= (const struct timespec &_tv) const

  Greater than or equal operator.
- bool **operator**>= (const **Time** &_time) const

  Greater than or equal operator.
- bool **operator**>= (double _time) const

  Greater than or equal operator.
- void **Set** (int32_t _sec, int32_t _nsec)

  Set to sec and nsec.
- void **Set** (double _seconds)

  Set to seconds.
- void **SetToWallTime** ()

  Set the time to the wall time.

Static Public Member Functions

- static const **Time** & **GetWallTime** ()

  Get the wall time.
- static const std::string & **GetWallTimeAsISOString** ()

  Get the wall time as an ISO string: YYYY-MM-DDTHH:MM:SS.
- static double **MicToNano** (double _ms)

  Convert microseconds to nanoseconds.
- static double **MilToNano** (double _ms)

  Convert milliseconds to nanoseconds.
- static **Time MSleep** (unsigned int _ms)

  Millisecond sleep.
- static **Time NSleep** (unsigned int _ns)

  Nano sleep.
- static **Time NSleep** (**Time** _time) **GAZEBO_DEPRECATED**(1.5)

  Nano sleep.
- static double **SecToNano** (double _sec)

  Convert seconds to nanoseconds.
- static **Time Sleep** (const **common::Time** &_time)

  Sleep for the specified time.

Public Attributes

- int32_t **nsec**

  Microseconds.
- int32_t **sec**

  Seconds.

Static Public Attributes

- static const **Time Zero**

  A *(p. 111) static zero time variable set to common::Time(0, 0).*

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Time** &_time)

  Stream insertion operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::common::Time** &_time)

  Stream extraction operator.

10.168.1 Detailed Description

A (p. 111) **Time** (p. 791) class, can be used to hold wall- or sim-time.

stored as sec and nano-sec.

10.168.2 Constructor & Destructor Documentation

10.168.2.1 gazebo::common::Time::Time ( )

Constructors.

10.168.2.2 gazebo::common::Time::Time (const Time & *_time*)

Copy constructor.

Parameters

in	*time*	**Time** (p. 791) to copy

10.168.2.3 gazebo::common::Time::Time (const struct timeval & *_tv*)

Constructor.

Parameters

in	*_tv*	**Time** (p. 791) to initialize to

10.168.2.4 gazebo::common::Time::Time (const struct timespec & *_tv*)

Constructor.

Parameters

in	*_tv*	**Time** (p. 791) to initialize to

10.168.2.5 gazebo::common::Time::Time (int32_t _sec, int32_t _nsec)

Constructor.

Parameters

in	*_sec*	Seconds
in	*_nsec*	Nanoseconds

10.168.2.6 gazebo::common::Time::Time (double _time)

Constuctor.

Parameters

in	*_time*	**Time** (p. 791) in double format sec.nsec

10.168.2.7 virtual gazebo::common::Time::~Time ( ) [virtual]

Destructor.

10.168.3 Member Function Documentation

10.168.3.1 double gazebo::common::Time::Double ( ) const

Get the time as a double.

Returns

Time (p. 791) as a double in seconds

10.168.3.2 float gazebo::common::Time::Float ( ) const

Get the time as a float.

Returns

Time (p. 791) as a float in seconds

10.168.3.3 static const Time& gazebo::common::Time::GetWallTime ( ) [static]

Get the wall time.

Returns

the current time

10.168.3.4 static const std::string& gazebo::common::Time::GetWallTimeAsISOString ( ) [static]

Get the wall time as an ISO string: YYYY-MM-DDTHH:MM:SS.

Returns

The current wall time as an ISO string.

10.168.3.5 static double gazebo::common::Time::MicToNano (double _ms) [inline], [static]

Convert microseconds to nanoseconds.

Parameters

_ms	microseconds

Returns

nanoseconds

10.168.3.6 static double gazebo::common::Time::MilToNano (double _ms) [inline], [static]

Convert milliseconds to nanoseconds.

Parameters

in	*_ms*	milliseconds

Returns

nanoseconds

10.168.3.7 static Time gazebo::common::Time::MSleep (unsigned int _ms) [static]

Millisecond sleep.

Parameters

in	*_ms*	milliseconds

Returns

Time (p. 791) actually slept

10.168.3.8 static Time gazebo::common::Time::NSleep (unsigned int _ns) [static]

Nano sleep.

Parameters

in	*_ns*	nanoseconds

Returns

Time (p. 791) actually slept

10.168.3.9 static Time gazebo::common::Time::NSleep (Time *_time*) [static]

Nano sleep.

Parameters

in	*_time*	is a **Time** (p. 791)

Returns

Time (p. 791) actually slept

10.168.3.10 bool gazebo::common::Time::operator!= (const struct timeval & *_tv*) const

Equal to operator.

Parameters

in	*_tv*	the time to compare to

Returns

true if values are the same, false otherwise

10.168.3.11 bool gazebo::common::Time::operator!= (const struct timespec & *_tv*) const

Equal to operator.

Parameters

in	*_tv*	the time to compare to

Returns

true if values are the same, false otherwise

10.168.3.12 bool gazebo::common::Time::operator!= (const Time & *_time*) const

Equal to operator.

Parameters

in	*_time*	the time to compare to

Returns

true if values are the same, false otherwise

10.168.3.13 bool gazebo::common::Time::operator!= (double *_time*) const

Equal to operator.

Parameters

in	*_time*	the time to compare to

Returns

true if values are the same, false otherwise

10.168.3.14 Time gazebo::common::Time::operator* (const struct timeval & *_tv*) const

Multiplication operator.

Parameters

in	*_tv*	The scaling duration

Returns

Time (p. 791) instance

10.168.3.15 Time gazebo::common::Time::operator* (const struct timespec & *_tv*) const

Multiplication operator.

Parameters

in	*_tv*	the scaling duration

Returns

Time (p. 791) instance

10.168.3.16 Time gazebo::common::Time::operator* (const Time & *_time*) const

Multiplication operators.

Parameters

in	*_time*	the scaling factor

Returns

a scaled **Time** (p. 791) instance

10.168.3.17 const Time& gazebo::common::Time::operator*= (const struct timeval & *_tv*)

Multiplication assignment operator.

Parameters

in	*_tv*	the scaling duration

Returns

a reference to this instance

10.168.3.18 const Time& gazebo::common::Time::operator*= (const struct timespec & *_tv*)

Multiplication assignment operator.

Parameters

in	*_tv*	the scaling duration

Returns

a reference to this instance

10.168.3.19 const Time& gazebo::common::Time::operator*= (const Time & *_time*)

Multiplication operators.

Parameters

in	*_time*	scale factor

Returns

a scaled **Time** (p. 791) instance

10.168.3.20 Time gazebo::common::Time::operator+ (const struct timeval & *_tv*) const

Addition operators.

Parameters

in	*_tv*	the time to add

Returns

a **Time** (p. 791) instance

10.168.3.21 Time gazebo::common::Time::operator+ (const struct timespec & _tv) const

Addition operators.

Parameters

in	*_tv*	the time to add

Returns

a **Time** (p. 791) instance

10.168.3.22 Time gazebo::common::Time::operator+ (const Time & _time) const

Addition operators.

Parameters

in	*_time*	The time to add

Returns

a **Time** (p. 791) instance

10.168.3.23 const Time& gazebo::common::Time::operator+= (const struct timeval & _tv)

Addition assignment operator.

Parameters

in	*_tv*	the time to add

Returns

a reference to this instance

10.168.3.24 const Time& gazebo::common::Time::operator+= (const struct timespec & _tv)

Addition assignment operator.

Parameters

in	*_tv*	the time to add

Returns

a reference to this instance

10.168.3.25 const Time& gazebo::common::Time::operator+= (const Time & *_time*)

Addition assignemtn operator.

Parameters

in	*_time*	The time to add

Returns

a **Time** (p. 791) instance

10.168.3.26 Time gazebo::common::Time::operator- (const struct timeval & *_tv*) const

Subtraction operator.

Parameters

in	*_tv*	The time to subtract

Returns

a **Time** (p. 791) instance

10.168.3.27 Time gazebo::common::Time::operator- (const struct timespec & *_tv*) const

Subtraction operator.

Parameters

in	*_tv*	The time to subtract

Returns

a **Time** (p. 791) instance

10.168.3.28 Time gazebo::common::Time::operator- (const Time & *_time*) const

Subtraction operator.

Parameters

in	*_time*	The time to subtract

Returns

a **Time** (p. 791) instance

10.168.3.29 const Time& gazebo::common::Time::operator-= (const struct timeval & _tv)

Subtraction assignment operator.

Parameters

in	*_tv*	The time to subtract

Returns

a **Time** (p. 791) instance

10.168.3.30 const Time& gazebo::common::Time::operator-= (const struct timespec & _tv)

Subtraction assignment operator.

Parameters

in	*_tv*	The time to subtract

Returns

a **Time** (p. 791) instance

10.168.3.31 const Time& gazebo::common::Time::operator-= (const Time & _time)

Subtraction assignment operator.

Parameters

in	*_time*	The time to subtract

Returns

a reference to this instance

10.168.3.32 Time gazebo::common::Time::operator/ (const struct timeval & _tv) const

Division operator.

Parameters

in	*_tv*	a timeval divisor

Returns

a **Time** (p. 791) instance

10.168.3.33 Time gazebo::common::Time::operator/ (const struct timespec & _tv) const

Division operator.

Parameters

in	*_tv*	a timespec divisor

Returns

a **Time** (p. 791) instance

10.168.3.34 Time gazebo::common::Time::operator/ (const Time & _time) const

Division operator.

Parameters

in	*_time*	the divisor

Returns

a **Time** (p. 791) instance

10.168.3.35 const Time& gazebo::common::Time::operator/= (const struct timeval & _tv)

Division assignment operator.

Parameters

in	*_tv*	a divisor

Returns

a **Time** (p. 791) instance

10.168.3.36 const Time& gazebo::common::Time::operator/= (const struct timespec & _tv)

Division assignment operator.

Parameters

in	*_tv*	a divisor

Returns

a **Time** (p. 791) instance

10.168.3.37 const Time& gazebo::common::Time::operator/= (const Time & *time*)

Division assignment operator.

Parameters

in	*time*	the divisor

Returns

a **Time** (p. 791) instance

10.168.3.38 bool gazebo::common::Time::operator< (const struct timeval & *_tv*) const

Less than operator.

Parameters

in	*_tv*	the time to compare with

Returns

true if tv is shorter than this, false otherwise

10.168.3.39 bool gazebo::common::Time::operator< (const struct timespec & *_tv*) const

Less than operator.

Parameters

in	*_tv*	the time to compare with

Returns

true if tv is shorter than this, false otherwise

10.168.3.40 bool gazebo::common::Time::operator< (const Time & *_time*) const

Less than operator.

Parameters

in	*_time*	the time to compare with

Returns

true if time is shorter than this, false otherwise

10.168.3.41 bool gazebo::common::Time::operator< (double *_time*) const

Less than operator.

Parameters

in	*_time*	the time to compare with

Returns

true if time is shorter than this, false otherwise

10.168.3.42 bool gazebo::common::Time::operator<= (const struct timeval & *_tv*) const

Less than or equal to operator.

Parameters

in	*_tv*	the time to compare with

Returns

true if tv is shorter than or equal to this, false otherwise

10.168.3.43 bool gazebo::common::Time::operator<= (const struct timespec & *_tv*) const

Less than or equal to operator.

Parameters

in	*_tv*	the time to compare with

Returns

true if tv is shorter than or equal to this, false otherwise

10.168.3.44 bool gazebo::common::Time::operator<= (const Time & *_time*) const

Less than or equal to operator.

Parameters

in	*_time*	the time to compare with

Returns

true if time is shorter than or equal to this, false otherwise

10.168.3.45 bool gazebo::common::Time::operator<= (double *_time*) const

Less than or equal to operator.

Parameters

in	*_time*	the time to compare with

Returns

true if time is shorter than or equal to this, false otherwise

10.168.3.46 Time& gazebo::common::Time::operator= (const struct timeval & *_tv*)

Assignment operator.

Parameters

in	*_tv*	the new time

Returns

a reference to this instance

10.168.3.47 Time& gazebo::common::Time::operator= (const struct timespec & *_tv*)

Assignment operator.

Parameters

in	*_tv*	the new time

Returns

a reference to this instance

10.168.3.48 Time& gazebo::common::Time::operator= (const Time & *_time*)

Assignment operator.

Parameters

in	*_time*	the new time

Returns

a reference to this instance

10.168.3.49 bool gazebo::common::Time::operator== (const struct timeval & *_tv*) const

Equal to operator.

Parameters

in	*_tv*	the time to compare to

Returns

true if values are the same, false otherwise

10.168.3.50 bool gazebo::common::Time::operator== (const struct timespec & *_tv*) const

Equal to operator.

Parameters

in	*_tv*	the time to compare to

Returns

true if values are the same, false otherwise

10.168.3.51 bool gazebo::common::Time::operator== (const Time & *_time*) const

Equal to operator.

Parameters

in	*_time*	the time to compare to

Returns

true if values are the same, false otherwise

10.168.3.52 bool gazebo::common::Time::operator== (double *_time*) const

Equal to operator.

Parameters

in	*_time*	the time to compare to

Returns

true if values are the same, false otherwise

10.168.3.53 bool gazebo::common::Time::operator> (const struct timeval & *_tv*) const

Greater than operator.

Parameters

in	*_tv*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.168.3.54 bool gazebo::common::Time::operator> (const struct timespec & *_tv*) const

Greater than operator.

Parameters

in	*_tv*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.168.3.55 bool gazebo::common::Time::operator> (const Time & *_time*) const

Greater than operator.

Parameters

in	*_time*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.168.3.56 bool gazebo::common::Time::operator> (double *_time*) const

Greater than operator.

Parameters

in	*_time*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.168.3.57 bool gazebo::common::Time::operator>= (const struct timeval & *_tv*) const

Greater than or equal operator.

Parameters

in	*_tv*	the time to compare with

Returns

true if tv is greater than or equal to this, false otherwise

10.168.3.58 bool gazebo::common::Time::operator>= (const struct timespec & *_tv*) const

Greater than or equal operator.

Parameters

in	*_tv*	the time to compare with

Returns

true if tv is greater than or equal to this, false otherwise

10.168.3.59 bool gazebo::common::Time::operator>= (const Time & *_time*) const

Greater than or equal operator.

Parameters

in	*_time*	the time to compare with

Returns

true if time is greater than or equal to this, false otherwise

10.168.3.60 bool gazebo::common::Time::operator>= (double *_time*) const

Greater than or equal operator.

Parameters

in	*_time*	the time to compare with

Returns

true if time is greater than or equal to this, false otherwise

10.168.3.61 static double gazebo::common::Time::SecToNano (double *_sec*) [inline],[static]

Convert seconds to nanoseconds.

Parameters

in	*_sec*	duration in seconds

Returns

nanoseconds

10.168.3.62 void gazebo::common::Time::Set (int32_t *_sec*, int32_t *_nsec*)

Set to sec and nsec.

Parameters

in	*_sec*	Seconds
in	*_nsec*	Nanoseconds

10.168.3.63 void gazebo::common::Time::Set (double *_seconds*)

Set to seconds.

Parameters

in	*_seconds*	Number of seconds

10.168.3.64 void gazebo::common::Time::SetToWallTime ( )

Set the time to the wall time.

10.168.3.65 static Time gazebo::common::Time::Sleep (const common::Time & *_time*) [static]

Sleep for the specified time.

Parameters

in	*_time*	Sleep time

Returns

Time (p. 791) actually slept

10.168.4 Friends And Related Function Documentation

10.168.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::common::Time & *_time*) [friend]

Stream insertion operator.

Parameters

in	*_out*	the output stream
in	*_time*	time to write to the stream

Returns

the output stream

10.168.4.2 std::istream& operator>> (std::istream & *_in,* gazebo::common::Time & *_time*) [friend]

Stream extraction operator.

Parameters

in	*_in*	the input stream
in	*_time*	time to read from to the stream

Returns

the input stream

10.168.5 Member Data Documentation

10.168.5.1 int32_t gazebo::common::Time::nsec

Microseconds.

10.168.5.2 int32_t gazebo::common::Time::sec

Seconds.

10.168.5.3 const Time gazebo::common::Time::Zero [static]

A (p. 111) static zero time variable set to common::Time(0, 0).

The documentation for this class was generated from the following file:

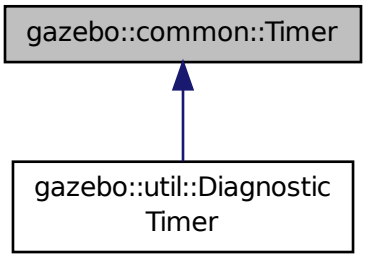
- **Time.hh**

10.169 gazebo::common::Timer Class Reference

A (p. 111) timer class, used to time things in real world walltime.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::Timer:



Public Member Functions

- **Timer** ()

  Constructor.
- virtual ~**Timer** ()

  Destructor.
- **Time GetElapsed** () const

  Get the elapsed time.
- bool **GetRunning** () const

  Returns true if the timer is running.
- virtual void **Start** ()

  Start the timer.
- virtual void **Stop** ()

  Stop the timer.

Friends

- std::ostream & **operator**<< (std::ostream &out, const **gazebo::common::Timer** &t)

  Stream operator friendly.

10.169.1 Detailed Description

A (p. 111) timer class, used to time things in real world walltime.

10.169.2 Constructor & Destructor Documentation

10.169.2.1 gazebo::common::Timer::Timer ( )

Constructor.

10.169.2.2 virtual gazebo::common::Timer::~Timer ( ) [virtual]

Destructor.

10.169.3 Member Function Documentation

10.169.3.1 Time gazebo::common::Timer::GetElapsed ( ) const

Get the elapsed time.

Returns

The time

10.169.3.2 bool gazebo::common::Timer::GetRunning ( ) const

Returns true if the timer is running.

Returns

Tue if the timer has been started and not stopped.

10.169.3.3 virtual void gazebo::common::Timer::Start ( ) [virtual]

Start the timer.

Reimplemented in **gazebo::util::DiagnosticTimer** (p. 266).

10.169.3.4 virtual void gazebo::common::Timer::Stop ( ) [virtual]

Stop the timer.

Reimplemented in **gazebo::util::DiagnosticTimer** (p. 266).

10.169.4 Friends And Related Function Documentation

10.169.4.1 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:

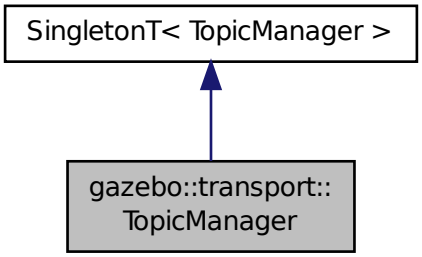
- **Timer.hh**

10.170 gazebo::transport::TopicManager Class Reference

Manages topics and their subscriptions.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::TopicManager:



Public Types

- typedef std::map< std::string,
  std::list< **NodePtr** > > **SubNodeMap**

  A (p. *111) map of string->list of* ***Node*** (p. *535) pointers.*

Public Member Functions

- void **AddNode** (**NodePtr** _node)

  Add a node to the manager.
- template<typename M >
  PublisherPtr Advertise (const std::string &_topic, unsigned int _queueLimit, double _hzRate)

  Advertise on a topic.
- void **ClearBuffers** ()

  Clear all buffers.
- void **ConnectPubToSub** (const std::string &_topic, const **SubscriptionTransportPtr** _sublink)

  Connection (p. *222) a local* ***Publisher*** (p. *618) to a remote* ***Subscriber*** (p. *776).*
- 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. *776) to a remote* ***Publisher*** (p. *618).*
- void **DisconnectPubFromSub** (const std::string &_topic, const std::string &_host, unsigned int _port)

  Disconnect a local publisher from a remote subscriber.
- void **DisconnectSubFromPub** (const std::string &_topic, const std::string &_host, unsigned int _port)

  Disconnect all local subscribers from a remote publisher.
- **PublicationPtr FindPublication** (const std::string &_topic)

Find a publication object by topic.

- void **Fini** ()

  Finalize the manager.

- std::map< std::string, std::list< std::string > > **GetAdvertisedTopics** () const **GAZEBO_DEPRECATED**(1.5)

  Get a list of all the topics.

- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

  Get all the topic namespaces.

- void **Init** ()

  Initialize the manager.

- bool **IsAdvertised** (const std::string &_topic)

  Has the topic been advertised?

- void **PauseIncoming** (bool _pause)

  Pause or unpause processing of incoming messages.

- void **ProcessNodes** (bool _onlyOut=false)

  Process all nodes under management.

- void **Publish** (const std::string &_topic, **MessagePtr** _message, const boost::function< void()> &_cb=**NULL**)

  Send a message.

- void **RegisterTopicNamespace** (const std::string &_name)

  Register a new topic namespace.

- void **RemoveNode** (unsigned int _id)

  Remove a node by its id.

- **SubscriberPtr Subscribe** (const **SubscribeOptions** &_options)

  Subscribe to a topic.

- void **Unadvertise** (const std::string &_topic)

  Unadvertise a topic.

- void **Unsubscribe** (const std::string &_topic, const **NodePtr** &_sub)

  Unsubscribe from a topic.

- **PublicationPtr UpdatePublications** (const std::string &_topic, const std::string &_msgType)

  Update our list of advertised topics.

Additional Inherited Members

10.170.1 Detailed Description

Manages topics and their subscriptions.

10.170.2 Member Typedef Documentation

10.170.2.1 typedef std::map<std::string, std::list<NodePtr> > gazebo::transport::TopicManager::SubNodeMap

A (p. 111) map of string->list of **Node** (p. 535) pointers.

10.170.3 Member Function Documentation

10.170.3.1 void gazebo::transport::TopicManager::AddNode (NodePtr _node)

Add a node to the manager.

Parameters

in,out	*_node*	The node to be added

10.170.3.2 template<typename M > PublisherPtr gazebo::transport::TopicManager::Advertise (const std::string & _topic, unsigned int _queueLimit, double _hzRate) [inline]

Advertise on a topic.

Parameters

in	*_topic*	The name of the topic
in	*_queueLimit*	The maximum number of outgoing messages to queue
in	*_hz*	Update rate for the publisher. Units are 1.0/seconds.

Returns

Pointer to the newly created **Publisher** (p. 618)

References gazebo::transport::Publication::AddPublisher(), gazebo::transport::Publication::AddSubscription(), FindPublication(), gazebo::transport::Publication::GetLocallyAdvertised(), GZ_ASSERT, gzthrow, SingletonT< T >::Instance(), NULL, gazebo::transport::Publication::SetLocallyAdvertised(), and UpdatePublications().

10.170.3.3 void gazebo::transport::TopicManager::ClearBuffers ( )

Clear all buffers.

10.170.3.4 void gazebo::transport::TopicManager::ConnectPubToSub (const std::string & _topic, const SubscriptionTransportPtr _sublink)

Connection (p. 222) a local **Publisher** (p. 618) to a remote **Subscriber** (p. 776).

Parameters

in	*_topic*	The topic to use
in	*_sublink*	The subscription transport object to use

10.170.3.5 void gazebo::transport::TopicManager::ConnectSubscribers (const std::string & _topic)

Connect all subscribers on a topic to known publishers.

Parameters

in	*_topic*	The topic to be connected

10.170.3.6 void gazebo::transport::TopicManager::ConnectSubToPub (const msgs::Publish & *_pub*)

Connect a local **Subscriber** (p. 776) to a remote **Publisher** (p. 618).

Parameters

in	*_pub*	The publish object to use

10.170.3.7 void gazebo::transport::TopicManager::DisconnectPubFromSub (const std::string & *_topic,* const std::string & *_host,* unsigned int *_port*)

Disconnect a local publisher from a remote subscriber.

Parameters

in	*_topic*	The topic to be disconnected
in	*_host*	The host to be disconnected
in	*_port*	The port to be disconnected

10.170.3.8 void gazebo::transport::TopicManager::DisconnectSubFromPub (const std::string & *_topic,* const std::string & *_host,* unsigned int *_port*)

Disconnect all local subscribers from a remote publisher.

Parameters

in	*_topic*	The topic to be disconnected
in	*_host*	The host to be disconnected
in	*_port*	The port to be disconnected

10.170.3.9 PublicationPtr gazebo::transport::TopicManager::FindPublication (const std::string & *_topic*)

Find a publication object by topic.

Parameters

in	*_topic*	The topic to search for

Returns

Pointer to the publication object, if found (can be null)

Referenced by Advertise().

10.170.3.10 void gazebo::transport::TopicManager::Fini ( )

Finalize the manager.

10.170.3.11 std::map<std::string, std::list<std::string> > gazebo::transport::TopicManager::GetAdvertisedTopics ( ) const

Get a list of all the topics.

Returns

A (p. 111) map where keys are message types, and values are a list of topic names.

See Also

transport::GetAdvertisedTopics

10.170.3.12 void gazebo::transport::TopicManager::GetTopicNamespaces (std::list< std::string > & *_namespaces*)

Get all the topic namespaces.

Parameters

out	*_namespaces*	The list of namespaces will be written here

10.170.3.13 void gazebo::transport::TopicManager::Init ( )

Initialize the manager.

10.170.3.14 bool gazebo::transport::TopicManager::IsAdvertised (const std::string & *_topic*)

Has the topic been advertised?

Parameters

in	*_topic*	The name of the topic to check

Returns

true if the topic has been advertised, false otherwise

10.170.3.15 void gazebo::transport::TopicManager::PauseIncoming (bool *_pause*)

Pause or unpause processing of incoming messages.

Parameters

in	*_pause*	If true pause processing; otherwse unpause

10.170.3.16 void gazebo::transport::TopicManager::ProcessNodes (bool *_onlyOut* = false)

Process all nodes under management.

Parameters

in	*_onlyOut*	True means only outbound messages on nodes will be sent. False means nodes process both outbound and inbound messages

10.170.3.17 void gazebo::transport::TopicManager::Publish (const std::string & *_topic*, MessagePtr *_message*, const boost::function< void() > & *_cb* = NULL)

Send a message.

Use a **Publisher** (p. 618) instead of calling this function directly.

Parameters

_topic	Name of the topic
_message	The message to send.
_cb	Callback, used when the publish is completed.

10.170.3.18 void gazebo::transport::TopicManager::RegisterTopicNamespace (const std::string & *_name*)

Register a new topic namespace.

Parameters

in	*_name*	The name of the new namespace

10.170.3.19 void gazebo::transport::TopicManager::RemoveNode (unsigned int *_id*)

Remove a node by its id.

Parameters

in	*_id*	The ID of the node to be removed

10.170.3.20 SubscriberPtr gazebo::transport::TopicManager::Subscribe (const SubscribeOptions & *_options*)

Subscribe to a topic.

Parameters

in	*_options*	The options to use for the subscription

Returns

Pointer to the newly created subscriber

10.170.3.21 void gazebo::transport::TopicManager::Unadvertise (const std::string & *_topic*)

Unadvertise a topic.

Parameters

in	*_topic*	The topic to be unadvertised

10.170.3.22 void gazebo::transport::TopicManager::Unsubscribe (const std::string & *_topic*, const NodePtr & *_sub*)

Unsubscribe from a topic.

Use a **Subscriber** (p. 776) rather than calling this function directly

Parameters

in	*_topic*	The topic to unsubscribe from
in	*_sub*	The node to unsubscribe

10.170.3.23 PublicationPtr gazebo::transport::TopicManager::UpdatePublications (const std::string & *_topic*, const std::string & *_msgType*)

Update our list of advertised topics.

Parameters

in	*_topic*	The topic to be updated
in	*_msgType*	The type of the topic to be updated

Returns

True if the provided params define a new publisher, false otherwise

Referenced by Advertise().

The documentation for this class was generated from the following file:

- **TopicManager.hh**

10.171 gazebo::physics::TrajectoryInfo Struct Reference

`#include <Actor.hh>`

Public Attributes

- double **duration**
- double **endTime**
- unsigned int **id**
- double **startTime**
- bool **translated**
- std::string **type**

10.171.1 Member Data Documentation

10.171.1.1 double gazebo::physics::TrajectoryInfo::duration

10.171.1.2 double gazebo::physics::TrajectoryInfo::endTime

10.171.1.3 unsigned int gazebo::physics::TrajectoryInfo::id

10.171.1.4 double gazebo::physics::TrajectoryInfo::startTime

10.171.1.5 bool gazebo::physics::TrajectoryInfo::translated

10.171.1.6 std::string gazebo::physics::TrajectoryInfo::type

The documentation for this struct was generated from the following file:

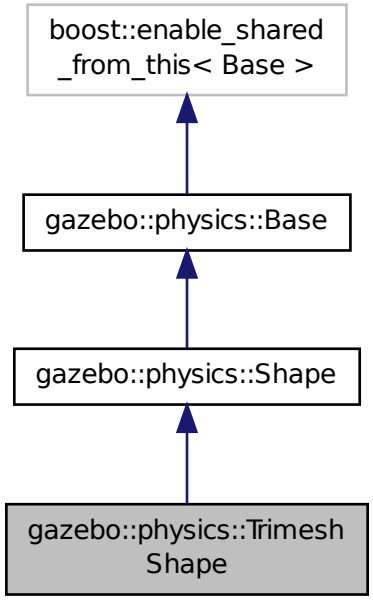
- **Actor.hh**

10.172 gazebo::physics::TrimeshShape Class Reference

Triangle mesh collision shape.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::TrimeshShape:



Public Member Functions

- **TrimeshShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**TrimeshShape** ()

  Destructor.
- void **FillMsg** (msgs::Geometry &_msg)

  Populate a msgs::Geometry message with data from this shape.
- std::string **GetFilename** () const **GAZEBO_DEPRECATED**(1.5)

  Deprecated.
- std::string **GetMeshURI** () const

  Get the URI of the mesh data.
- virtual **math::Vector3 GetSize** () const

  Get the size of the triangle mesh.
- virtual void **Init** ()

  Initialize the shape.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  Update this shape from a message.
- void **SetFilename** (const std::string &_filename) **GAZEBO_DEPRECATED**(1.5)

  Deprecated.
- void **SetMesh** (const std::string &_uri, const std::string &_submesh="", bool _center=false)

  Set the mesh uri and submesh name.
- void **SetScale** (const **math::Vector3** &_scale)

  Set the scaling factor.
- virtual void **Update** ()

  Update the tri mesh.

Protected Attributes

- const **common::Mesh** * **mesh**

  Pointer to the mesh data.
- **common::SubMesh** * **submesh**

  The submesh to use from within the parent mesh.

Additional Inherited Members

10.172.1 Detailed Description

Triangle mesh collision shape.

10.172.2 Constructor & Destructor Documentation

10.172.2.1 gazebo::physics::TrimeshShape::TrimeshShape (**CollisionPtr** *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent collision.

10.172.2.2 virtual gazebo::physics::TrimeshShape::~TrimeshShape ( ) [virtual]

Destructor.

10.172.3 Member Function Documentation

10.172.3.1 void gazebo::physics::TrimeshShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Populate a msgs::Geometry message with data from this shape.

Parameters

out	*_msg*	Message to fill.

Implements **gazebo::physics::Shape** (p. 722).

10.172.3.2 std::string gazebo::physics::TrimeshShape::GetFilename ( ) const

Deprecated.

See Also

GetMeshURI (p. 824)

10.172.3.3 std::string gazebo::physics::TrimeshShape::GetMeshURI ( ) const

Get the URI of the mesh data.

Returns

The URI of the mesh data.

10.172.3.4 virtual math::Vector3 gazebo::physics::TrimeshShape::GetSize ( ) const [virtual]

Get the size of the triangle mesh.

Returns

The size of the triangle mesh.

10.172.3.5 virtual void gazebo::physics::TrimeshShape::Init ( ) [virtual]

Initialize the shape.

Implements **gazebo::physics::Shape** (p. 722).

10.172.3.6 virtual void gazebo::physics::TrimeshShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Update this shape from a message.

Parameters

in	*_msg*	Message that contains triangle mesh info.

Implements **gazebo::physics::Shape** (p. 722).

10.172.3.7 void gazebo::physics::TrimeshShape::SetFilename (const std::string & *_filename*)

Deprecated.

See Also

SetMesh (p. 825)

10.172.3.8 void gazebo::physics::TrimeshShape::SetMesh (const std::string & *_uri*, const std::string & *_submesh* = " ", bool *_center* = false)

Set the mesh uri and submesh name.

Parameters

in	*_uri*	Filename of the mesh file to load from.
in	*_submesh*	Name of the submesh to use within the mesh
in	*_center*	True to center the submesh. specified in the _uri.

10.172.3.9 void gazebo::physics::TrimeshShape::SetScale (const math::Vector3 & *_scale*)

Set the scaling factor.

Parameters

in	*_scale*	Scaling factor.

10.172.3.10 virtual void gazebo::physics::TrimeshShape::Update ( ) [inline], [virtual]

Update the tri mesh.

Reimplemented from **gazebo::physics::Base** (p. 148).

10.172.4 Member Data Documentation

10.172.4.1 const common::Mesh* gazebo::physics::TrimeshShape::mesh [protected]

Pointer to the mesh data.

10.172.4.2 common::SubMesh* gazebo::physics::TrimeshShape::submesh [protected]

The submesh to use from within the parent mesh.

The documentation for this class was generated from the following file:

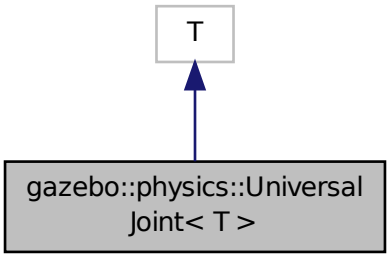
- **TrimeshShape.hh**

10.173 gazebo::physics::UniversalJoint< T > Class Template Reference

A (p. 111) universal joint.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::UniversalJoint< T >:



Public Member Functions

- **UniversalJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**UniversalJoint** ()

  Destuctor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load a **UniversalJoint** (p. 826).*

10.173.1 Detailed Description

template<class T>class gazebo::physics::UniversalJoint< T >

A (p. 111) universal joint.

10.173.2 Constructor & Destructor Documentation

10.173.2.1 template<class T > gazebo::physics::UniversalJoint< T >::UniversalJoint (BasePtr _parent) [inline], [explicit]

Constructor.

Parameters

in	*_parent*	Parent link of the univeral joint.

References gazebo::physics::Base::UNIVERSAL_JOINT.

10.173.2.2 template<class T > virtual gazebo::physics::UniversalJoint< T >::~UniversalJoint ( ) [inline], [virtual]

Destuctor.

10.173.3 Member Function Documentation

10.173.3.1 template<class T > virtual unsigned int gazebo::physics::UniversalJoint< T >::GetAngleCount ( ) const [inline], [virtual]

10.173.3.2 template<class T > virtual void gazebo::physics::UniversalJoint< T >::Load (sdf::ElementPtr _sdf) [inline], [virtual]

Load a **UniversalJoint** (p. 826).

Parameters

in	*_sdf*	SDF values to load from.

The documentation for this class was generated from the following file:

- **UniversalJoint.hh**

10.174 gazebo::common::UpdateInfo Class Reference

Information for use in an update event.

`#include <common/common.hh>`

Public Attributes

- **common::Time realTime**

  Current real time.
- **common::Time simTime**

  Current simulation time.
- std::string **worldName**

  Name of the world.

10.174.1 Detailed Description

Information for use in an update event.

10.174.2 Member Data Documentation

10.174.2.1 common::Time gazebo::common::UpdateInfo::realTime

Current real time.

10.174.2.2 common::Time gazebo::common::UpdateInfo::simTime

Current simulation time.

10.174.2.3 std::string gazebo::common::UpdateInfo::worldName

Name of the world.

The documentation for this class was generated from the following file:

- **UpdateInfo.hh**

10.175 urdf2gazebo::URDF2Gazebo Class Reference

`#include <parser_urdf.hh>`

Public Member Functions

- **URDF2Gazebo** ()
  constructor
- ∼**URDF2Gazebo** ()
  destructor
- TiXmlDocument **InitModelDoc** (TiXmlDocument ∗_xmlDoc)
  convert urdf xml document string to sdf xml document
- TiXmlDocument **InitModelFile** (const std::string &_filename)
  convert urdf file to sdf xml document
- TiXmlDocument **InitModelString** (const std::string &_urdfStr, bool _enforceLimits=true)
  convert urdf string to sdf xml document, with option to enforce limits.

10.175.1 Constructor & Destructor Documentation

10.175.1.1 urdf2gazebo::URDF2Gazebo::URDF2Gazebo ( )

constructor

10.175.1.2 urdf2gazebo::URDF2Gazebo::~URDF2Gazebo ( )

destructor

10.175.2 Member Function Documentation

10.175.2.1 TiXmlDocument urdf2gazebo::URDF2Gazebo::InitModelDoc (TiXmlDocument * *_xmlDoc*)

convert urdf xml document string to sdf xml document

Parameters

in	*_xmlDoc*	a tinyxml document containing the urdf model

Returns

a tinyxml document containing sdf of the model

10.175.2.2 TiXmlDocument urdf2gazebo::URDF2Gazebo::InitModelFile (const std::string & *_filename*)

convert urdf file to sdf xml document

Parameters

in	*_urdfStr*	a string containing filename of the urdf model

Returns

a tinyxml document containing sdf of the model

10.175.2.3 TiXmlDocument urdf2gazebo::URDF2Gazebo::InitModelString (const std::string & *_urdfStr,* bool *_enforceLimits* = `true`)

convert urdf string to sdf xml document, with option to enforce limits.

Parameters

in	*_urdfStr*	a string containing model urdf
in	*_enforceLimits*	option to enforce joint limits

Returns

a tinyxml document containing sdf of the model

The documentation for this class was generated from the following file:

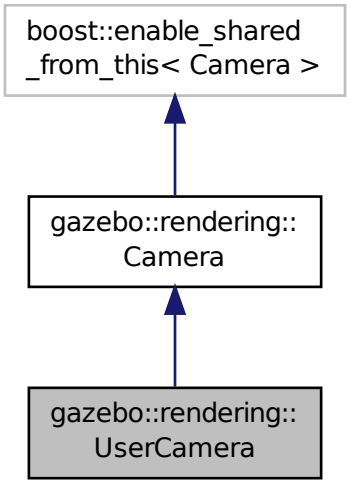
- **parser_urdf.hh**

10.176 gazebo::rendering::UserCamera Class Reference

A (p. 111) camera used for user visualization of a scene.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::UserCamera:



Public Member Functions

- **UserCamera** (const std::string &_name, **ScenePtr** _scene)

  Constructor.
- virtual ~**UserCamera** ()

  Destructor.
- void **EnableViewController** (bool _value) const

  Set whether the view controller is enabled.
- void **Fini** ()

  Finialize.
- float **GetAvgFPS** () const

  Get the average frames per second.
- **GUIOverlay** * **GetGUIOverlay** ()

  Get the GUI overlay.
- virtual unsigned int **GetImageHeight** () const

  Get the height of the image.
- virtual unsigned int **GetImageWidth** () const

  Get the width of the image.
- float **GetTriangleCount** () const

  Get the triangle count.

- std::string **GetViewControllerTypeString** ()

  Get current view controller type.
- **VisualPtr GetVisual** (const **math::Vector2i** &_mousePos, std::string &_mod)

  Get an entity at a pixel location using a camera.
- **VisualPtr GetVisual** (const **math::Vector2i** &_mousePos) const

  Get a visual at a mouse position.
- void **HandleKeyPressEvent** (const std::string &_key)

  Handle a key press.
- void **HandleKeyReleaseEvent** (const std::string &_key)

  Handle a key release.
- void **HandleMouseEvent** (const **common::MouseEvent** &_evt)

  Handle a mouse event.
- void **Init** ()

  Initialize.
- void **Load** (**sdf::ElementPtr** _sdf)

  Load the user camera.
- void **Load** ()

  Generic load function.
- virtual bool **MoveToPosition** (const **math::Pose** &_pose, double _time)

  Move the camera to a position (this is an animated motion).
- void **MoveToVisual** (**VisualPtr** _visual)

  Move the camera to focus on a visual.
- void **MoveToVisual** (const std::string &_visualName)

  Move the camera to focus on a visual.
- virtual void **PostRender** ()

  Post render.
- void **Resize** (unsigned int _w, unsigned int _h)

  Resize the camera.
- void **SetFocalPoint** (const **math::Vector3** &_pt)

  Set the point the camera should orbit around.
- virtual void **SetRenderTarget** (Ogre::RenderTarget ∗_target)

  Set to true to enable rendering.
- void **SetViewController** (const std::string &_type)

  Set view controller.
- void **SetViewController** (const std::string &_type, const **math::Vector3** &_pos)

  Set view controller.
- void **SetViewportDimensions** (float _x, float _y, float _w, float _h)

  Set the dimensions of the viewport.
- virtual void **SetWorldPose** (const **math::Pose** &_pose)

  Set the pose in the world coordinate frame.
- virtual void **Update** ()

  Render the camera.

Protected Member Functions

- virtual void **AnimationComplete** ()

  Internal function used to indicate that an animation has completed.
- virtual bool **AttachToVisualImpl** (**VisualPtr** _visual, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  Set the camera to be attached to a visual.
- virtual bool **TrackVisualImpl** (**VisualPtr** _visual)

  Set the camera to track a scene node.

Additional Inherited Members

10.176.1 Detailed Description

A (p. 111) camera used for user visualization of a scene.

10.176.2 Constructor & Destructor Documentation

10.176.2.1 gazebo::rendering::UserCamera::UserCamera (const std::string & *_name,* ScenePtr *_scene*)

Constructor.

Parameters

in	*_name*	Name of the camera.
in	*_scene*	**Scene** (p. 676) to put the camera in.

10.176.2.2 virtual gazebo::rendering::UserCamera::~UserCamera ( ) [virtual]

Destructor.

10.176.3 Member Function Documentation

10.176.3.1 virtual void gazebo::rendering::UserCamera::AnimationComplete ( ) [protected], [virtual]

Internal function used to indicate that an animation has completed.

Reimplemented from **gazebo::rendering::Camera** (p. 169).

10.176.3.2 virtual bool gazebo::rendering::UserCamera::AttachToVisualImpl (VisualPtr *_visual,* bool *_inheritOrientation,* double *_minDist* = 0, double *_maxDist* = 0) [protected], [virtual]

Set the camera to be attached to a visual.

This causes the camera to move in relation to the specified visual.

Parameters

in	*_visual*	The visual to attach to.
in	*_inherit-Orientation*	True if the camera should also rotate when the visual rotates.
in	*_minDist*	Minimum distance the camera can get to the visual.
in	*_maxDist*	Maximum distance the camera can get from the visual.

Returns

True if successfully attach to the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 170).

10.176.3.3 void gazebo::rendering::UserCamera::EnableViewController (bool *_value*) const

Set whether the view controller is enabled.

The view controller is used to handle user camera movements.

Parameters

in	*_value*	True to enable viewcontroller, False to disable.

10.176.3.4 void gazebo::rendering::UserCamera::Fini ( ) [virtual]

Finialize.

Reimplemented from **gazebo::rendering::Camera** (p. 171).

10.176.3.5 float gazebo::rendering::UserCamera::GetAvgFPS ( ) const

Get the average frames per second.

Returns

The average rendering frames per second

10.176.3.6 GUIOverlay* gazebo::rendering::UserCamera::GetGUIOverlay ( )

Get the GUI overlay.

An overlay allows you to draw 2D elements on the viewport.

Returns

Pointer to the **GUIOverlay** (p. 345).

10.176.3.7 virtual unsigned int gazebo::rendering::UserCamera::GetImageHeight ( ) const [virtual]

Get the height of the image.

Returns

Image height

Reimplemented from **gazebo::rendering::Camera** (p. 173).

10.176.3.8 virtual unsigned int gazebo::rendering::UserCamera::GetImageWidth ( ) const [virtual]

Get the width of the image.

Returns

Image width

Reimplemented from **gazebo::rendering::Camera** (p. 174).

10.176.3.9 float gazebo::rendering::UserCamera::GetTriangleCount ( ) const

Get the triangle count.

Returns

The number of triangles currently being rendered.

10.176.3.10 std::string gazebo::rendering::UserCamera::GetViewControllerTypeString ( )

Get current view controller type.

Returns

Type of the current view controller: "orbit", "fps"

10.176.3.11 VisualPtr gazebo::rendering::UserCamera::GetVisual (const math::Vector2i & *_mousePos*, std::string & *_mod*)

Get an entity at a pixel location using a camera.

Used for mouse picking.

Parameters

in	*_mousePos*	The position of the mouse in screen coordinates
out	*_mod*	Used for object manipulation

Returns

The selected entity, or NULL

10.176.3.12 VisualPtr gazebo::rendering::UserCamera::GetVisual (const math::Vector2i & *_mousePos*) const

Get a visual at a mouse position.

Parameters

in	*_mousePos*	2D position of the mouse in pixels.

10.176.3.13 void gazebo::rendering::UserCamera::HandleKeyPressEvent (const std::string & *_key*)

Handle a key press.

Parameters

in	*_key*	The key pressed.

10.176.3.14 void gazebo::rendering::UserCamera::HandleKeyReleaseEvent (const std::string & *_key*)

Handle a key release.

Parameters

in	*_key*	The key released.

10.176.3.15 void gazebo::rendering::UserCamera::HandleMouseEvent (const common::MouseEvent & *_evt*)

Handle a mouse event.

Parameters

in	*_evt*	The mouse event.

10.176.3.16 void gazebo::rendering::UserCamera::Init ( ) [virtual]

Initialize.

Reimplemented from **gazebo::rendering::Camera** (p. 178).

10.176.3.17 void gazebo::rendering::UserCamera::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the user camera.

Parameters

in	*_sdf*	Parameters for the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 179).

10.176.3.18 void gazebo::rendering::UserCamera::Load ( ) [virtual]

Generic load function.

Reimplemented from **gazebo::rendering::Camera** (p. 179).

10.176.3.19 virtual bool gazebo::rendering::UserCamera::MoveToPosition (const math::Pose & _pose, double _time) [virtual]

Move the camera to a position (this is an animated motion).

See Also

Camera::MoveToPositions (p. 180)

Parameters

in	*_pose*	End position of the camera
in	*_time*	Duration of the camera's movement

Reimplemented from **gazebo::rendering::Camera** (p. 180).

10.176.3.20 void gazebo::rendering::UserCamera::MoveToVisual (VisualPtr _visual)

Move the camera to focus on a visual.

Parameters

in	*_visual*	**Visual** (p. 885) to move the camera to.

10.176.3.21 void gazebo::rendering::UserCamera::MoveToVisual (const std::string & _visualName)

Move the camera to focus on a visual.

Parameters

in	*_visualName*	Name of the visual to move the camera to.

10.176.3.22 virtual void gazebo::rendering::UserCamera::PostRender ( ) [virtual]

Post render.

Reimplemented from **gazebo::rendering::Camera** (p. 180).

10.176.3.23 void gazebo::rendering::UserCamera::Resize (unsigned int _w, unsigned int _h)

Resize the camera.

Parameters

in	*_w*	Width of the camera image.
in	*_h*	Height of the camera image.

10.176.3.24 void gazebo::rendering::UserCamera::SetFocalPoint (const math::Vector3 & _pt)

Set the point the camera should orbit around.

Parameters

in	*_pt*	The focal point

10.176.3.25 virtual void gazebo::rendering::UserCamera::SetRenderTarget (Ogre::RenderTarget * *_target*) [virtual]

Set to true to enable rendering.

Use this only if you really know what you're doing.

Parameters

in	*_target*	The new rendering target.

Reimplemented from **gazebo::rendering::Camera** (p. 183).

10.176.3.26 void gazebo::rendering::UserCamera::SetViewController (const std::string & *_type*)

Set view controller.

Parameters

in	*_type*	The type of view controller: "orbit", "fps"

10.176.3.27 void gazebo::rendering::UserCamera::SetViewController (const std::string & *_type,* const **math::Vector3** & *_pos*)

Set view controller.

Parameters

in	*_type*	The type of view controller: "orbit", "fps"
in	*_pos*	The initial pose of the camera.

10.176.3.28 void gazebo::rendering::UserCamera::SetViewportDimensions (float *_x,* float *_y,* float *_w,* float *_h*)

Set the dimensions of the viewport.

Parameters

in	*_x*	X position of the viewport.
in	*_y*	Y position of the viewport.
in	*_w*	Width of the viewport.
in	*_h*	Height of the viewport.

10.176.3.29 virtual void gazebo::rendering::UserCamera::SetWorldPose (const **math::Pose** & *_pose*) [virtual]

Set the pose in the world coordinate frame.

Parameters

in	*_pose*	New pose of the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 184).

10.176.3.30 virtual bool gazebo::rendering::UserCamera::TrackVisualImpl (VisualPtr *_visual*) [protected], [virtual]

Set the camera to track a scene node.

Tracking just causes the camera to rotate to follow the visual.

Parameters

in	*_visual*	**Visual** (p. 885) to track.

Returns

True if the camera is now tracking the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 185).

10.176.3.31 virtual void gazebo::rendering::UserCamera::Update ( ) [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 185).

The documentation for this class was generated from the following file:

- **UserCamera.hh**

10.177 gazebo::math::Vector2d Class Reference

Generic double x, y vector.

`#include <Vector2d.hh>`

Public Member Functions

- **Vector2d** ()

  Constructor.
- **Vector2d** (const double &_x, const double &_y)

  Constructor.
- **Vector2d** (const **Vector2d** &_v)

  Copy constructor.
- virtual ~**Vector2d** ()

  Destructor.
- **Vector2d Cross** (const **Vector2d** &_v) const

  Return the cross product of this vector and _v.

- double **Distance** (const **Vector2d** &_pt) const
  Calc distance to the given point.
- bool **IsFinite** () const
  See if a point is finite (e.g., not nan)
- void **Normalize** ()
  Normalize the vector length.
- bool **operator!=** (const **Vector2d** &_v) const
  Not equal to operator.
- const **Vector2d operator**∗ (const **Vector2d** &_v) const
  Multiplication operators.
- const **Vector2d operator**∗ (double _v) const
  Multiplication operators.
- const **Vector2d** & **operator**∗**=** (const **Vector2d** &_v)
  Multiplication assignment operator.
- const **Vector2d** & **operator**∗**=** (double _v)
  Multiplication assignment operator.
- **Vector2d operator+** (const **Vector2d** &_v) const
  Addition operator.
- const **Vector2d** & **operator+=** (const **Vector2d** &_v)
  Addition assignment operator.
- **Vector2d operator-** (const **Vector2d** &_v) const
  Subtraction operator.
- const **Vector2d** & **operator-=** (const **Vector2d** &_v)
  Subtraction assignment operator.
- const **Vector2d operator**/ (const **Vector2d** &_v) const
  Division operator.
- const **Vector2d operator**/ (double _v) const
  Division operator.
- const **Vector2d** & **operator**/**=** (const **Vector2d** &_v)
  Division operator.
- const **Vector2d** & **operator**/**=** (double _v)
  Division operator.
- **Vector2d** & **operator=** (const **Vector2d** &_v)
  Assignment operator.
- const **Vector2d** & **operator=** (double _v)
  Assignment operator.
- bool **operator==** (const **Vector2d** &_v) const
  Equal to operator.
- double **operator[ ]** (unsigned int _index) const
  Array subscript operator.
- void **Set** (double _x, double _y)
  Set the contents of the vector.

Public Attributes

- double **x**
  x data
- double **y**
  y data

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector2d** &_pt)

  Stream extraction operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector2d** &_pt)

  Stream extraction operator.

10.177.1 Detailed Description

Generic double x, y vector.

10.177.2 Constructor & Destructor Documentation

10.177.2.1 gazebo::math::Vector2d::Vector2d ( )

Constructor.

10.177.2.2 gazebo::math::Vector2d::Vector2d (const double & *_x*, const double & *_y*)

Constructor.

Parameters

in	*_x*	value along x
in	*_y*	value along y

10.177.2.3 gazebo::math::Vector2d::Vector2d (const **Vector2d** & *_v*)

Copy constructor.

Parameters

in	*_v*	the value

10.177.2.4 virtual gazebo::math::Vector2d::~Vector2d ( ) [virtual]

Destructor.

10.177.3 Member Function Documentation

10.177.3.1 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

10.177.3.2 double gazebo::math::Vector2d::Distance (const Vector2d & _pt) const

Calc distance to the given point.

Parameters

in	_pt	The point to measure to

Returns

the distance

10.177.3.3 bool gazebo::math::Vector2d::IsFinite ( ) const

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.177.3.4 void gazebo::math::Vector2d::Normalize ( )

Normalize the vector length.

10.177.3.5 bool gazebo::math::Vector2d::operator!= (const Vector2d & _v) const

Not equal to operator.

Returns

true if elements are of diffent values (tolerence 1e-6)

10.177.3.6 const Vector2d gazebo::math::Vector2d::operator* (const Vector2d & _v) const

Multiplication operators.

Parameters

in	_v	the vector

Returns

the result

10.177.3.7 const Vector2d gazebo::math::Vector2d::operator∗ (double _v) const

Multiplication operators.

Parameters

in	_v	the scaling factor

Returns

a scaled vector

10.177.3.8 const Vector2d& gazebo::math::Vector2d::operator∗= (const Vector2d & _v)

Multiplication assignment operator.

Remarks

this is an element wise multiplication

Parameters

in	_v	the vector

Returns

this

10.177.3.9 const Vector2d& gazebo::math::Vector2d::operator∗= (double _v)

Multiplication assignment operator.

Parameters

in	_v	the scaling factor

Returns

a scaled vector

10.177.3.10 Vector2d gazebo::math::Vector2d::operator+ (const Vector2d & _v) const

Addition operator.

Parameters

in	_v	vector to add

Returns

sum vector

10.177.3.11 const Vector2d& gazebo::math::Vector2d::operator+= (const Vector2d & _v)

Addition assignment operator.

Parameters

in	_v	the vector to add

10.177.3.12 Vector2d gazebo::math::Vector2d::operator- (const Vector2d & _v) const

Subtraction operator.

Parameters

in	_v	the vector to substract

Returns

the subtracted vector

10.177.3.13 const Vector2d& gazebo::math::Vector2d::operator-= (const Vector2d & _v)

Subtraction assignment operator.

Parameters

in	_v	the vector to substract

Returns

this

10.177.3.14 const Vector2d gazebo::math::Vector2d::operator/ (const Vector2d & _v) const

Division operator.

Remarks

this is an element wise division

Parameters

in	_v	a vector

Returns

a result

10.177.3.15 const Vector2d gazebo::math::Vector2d::operator/ (double _v) const

Division operator.

Parameters

in	_v	the value

Returns

a vector

10.177.3.16 const Vector2d& gazebo::math::Vector2d::operator/= (const Vector2d & _v)

Division operator.

Remarks

this is an element wise division

Parameters

in	_v	a vector

Returns

this

10.177.3.17 const Vector2d& gazebo::math::Vector2d::operator/= (double _v)

Division operator.

Parameters

in	_v	the divisor

Returns

a vector

10.177.3.18 Vector2d& gazebo::math::Vector2d::operator= (const Vector2d & _v)

Assignment operator.

Parameters

in	*_v*	a value for x and y element

Returns

this

10.177.3.19 const Vector2d& gazebo::math::Vector2d::operator= (double _v)

Assignment operator.

Parameters

in	*_v*	the value for x and y element

Returns

this

10.177.3.20 bool gazebo::math::Vector2d::operator== (const Vector2d & _v) const

Equal to operator.

Parameters

in	*_v*	the vector to compare to

Returns

true if the elements of the 2 vectors are equal within a tolerence (1e-6)

10.177.3.21 double gazebo::math::Vector2d::operator[] (unsigned int _index) const

Array subscript operator.

Parameters

in	*_index*	the index

Returns

the value, or 0 if _index is out of bounds

10.177.3.22 void gazebo::math::Vector2d::Set (double _x, double _y)

Set the contents of the vector.

Parameters

in	*_x*	value along x
in	*_y*	value along y

10.177.4 Friends And Related Function Documentation

10.177.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::math::Vector2d & *_pt*) [friend]

Stream extraction operator.

Parameters

in	*_out*	output stream
in	*_pt*	**Vector2d** (p. 838) to output

Returns

The stream

10.177.4.2 std::istream& operator>> (std::istream & *_in,* gazebo::math::Vector2d & *_pt*) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_pt*	**Vector3** (p. 855) to read values into

Returns

The stream

10.177.5 Member Data Documentation

10.177.5.1 double gazebo::math::Vector2d::x

x data

10.177.5.2 double gazebo::math::Vector2d::y

y data

The documentation for this class was generated from the following file:

- **Vector2d.hh**

10.178 gazebo::math::Vector2i Class Reference

Generic integer x, y vector.

`#include <math/gzmath.hh>`

Public Member Functions

- **Vector2i** ()

  Constructor.
- **Vector2i** (const int &_x, const int &_y)

  Constructor.
- **Vector2i** (const **Vector2i** &_pt)

  Copy onstructor.
- virtual ∼**Vector2i** ()

  Destructor.
- **Vector2i Cross** (const **Vector2i** &_pt) const

  Return the cross product of this vector and _pt.
- int **Distance** (const **Vector2i** &_pt) const

  Calc distance to the given point.
- bool **IsFinite** () const

  See if a point is finite (e.g., not nan)
- void **Normalize** ()

  Normalize the vector length.
- bool **operator!=** (const **Vector2i** &_v) const

  Equality operators.
- const **Vector2i operator**∗ (const **Vector2i** &_v) const

  Multiplication operator.
- const **Vector2i operator**∗ (int _v) const

  Multiplication operator.
- const **Vector2i** & **operator**∗**=** (const **Vector2i** &_v)

  Multiplication operators.
- const **Vector2i** & **operator**∗**=** (int _v)

  Multiplication operator.
- **Vector2i operator+** (const **Vector2i** &_v) const

  Addition operator.
- const **Vector2i** & **operator+=** (const **Vector2i** &_v)

  Addition assignment operator.
- **Vector2i operator-** (const **Vector2i** &_v) const

  Subtraction operator.
- const **Vector2i** & **operator-=** (const **Vector2i** &_v)

  Subtraction operators.
- const **Vector2i operator**/ (const **Vector2i** &_v) const

  Division operator.
- const **Vector2i operator**/ (int _v) const

  Division operator.
- const **Vector2i** & **operator**/**=** (const **Vector2i** &_v)

  Division operator.
- const **Vector2i** & **operator**/**=** (int _v)

  Division operator.
- **Vector2i** & **operator=** (const **Vector2i** &_v)

Assignment operator.

- const **Vector2i** & **operator=** (int _value)

  Assignment operator.
- bool **operator==** (const **Vector2i** &_v) const

  Equality operator.
- int **operator[ ]** (unsigned int _index) const

  Array subscript operator.
- void **Set** (int _x, int _y)

  Set the contents of the vector.

Public Attributes

- int **x**

  x data
- int **y**

  y data

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector2i** &_pt)

  Stream insertion operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector2i** &_pt)

  Stream extraction operator.

10.178.1 Detailed Description

Generic integer x, y vector.

10.178.2 Constructor & Destructor Documentation

10.178.2.1 gazebo::math::Vector2i::Vector2i ( )

Constructor.

10.178.2.2 gazebo::math::Vector2i::Vector2i (const int & *_x*, const int & *_y*)

Constructor.

Parameters

in	*_x*	value along x
in	*_y*	value along y

10.178.2.3 gazebo::math::Vector2i::Vector2i (const Vector2i & *_pt*)

Copy onstructor.

Parameters

in	*_pt*	a point

10.178.2.4 virtual gazebo::math::Vector2i::~Vector2i ( ) [virtual]

Destructor.

10.178.3 Member Function Documentation

10.178.3.1 Vector2i gazebo::math::Vector2i::Cross (const Vector2i & *_pt*) const

Return the cross product of this vector and _pt.

Parameters

in	*_pt*	the other vector

Returns

the product

10.178.3.2 int gazebo::math::Vector2i::Distance (const Vector2i & *_pt*) const

Calc distance to the given point.

Parameters

in	*_pt*	a point

Returns

the distance

10.178.3.3 bool gazebo::math::Vector2i::IsFinite ( ) const

See if a point is finite (e.g., not nan)

Returns

the result

10.178.3.4 void gazebo::math::Vector2i::Normalize ( )

Normalize the vector length.

10.178.3.5 bool gazebo::math::Vector2i::operator!= (const Vector2i & *_v*) const

Equality operators.

Parameters

_v	the vector to compare with

Returns

true if component have different values, false otherwise

10.178.3.6 const Vector2i gazebo::math::Vector2i::operator∗ (const Vector2i & _v) const

Multiplication operator.

Remarks

this is an element wise multiplication

Parameters

in	_v	the vector

Returns

the result

10.178.3.7 const Vector2i gazebo::math::Vector2i::operator∗ (int _v) const

Multiplication operator.

Parameters

in	_v	the scaling factor

Returns

the result

10.178.3.8 const Vector2i& gazebo::math::Vector2i::operator∗= (const Vector2i & _v)

Multiplication operators.

Remarks

this is an element wise multiplication

Parameters

in	_v	the vector

Returns

this

10.178.3.9 const Vector2i& gazebo::math::Vector2i::operator*= (int _v)

Multiplication operator.

Parameters

in	_v	scaling factor

Returns

this

10.178.3.10 Vector2i gazebo::math::Vector2i::operator+ (const Vector2i & _v) const

Addition operator.

Parameters

in	_v	the vector to add

Returns

the sum vector

10.178.3.11 const Vector2i& gazebo::math::Vector2i::operator+= (const Vector2i & _v)

Addition assignment operator.

Parameters

in	_v	the vector to add

Returns

this

10.178.3.12 Vector2i gazebo::math::Vector2i::operator- (const Vector2i & _v) const

Subtraction operator.

Parameters

in	_v	the vector to substract

Returns

the result vector

10.178.3.13 const Vector2i& gazebo::math::Vector2i::operator-= (const Vector2i & _v)

Subtraction operators.

Parameters

in	_v	the vector to substract

Returns

this

10.178.3.14 const Vector2i gazebo::math::Vector2i::operator/ (const Vector2i & _v) const

Division operator.

Remarks

this is an element wise division.

Parameters

in	_v	the vector to divide

Returns

the result

10.178.3.15 const Vector2i gazebo::math::Vector2i::operator/ (int _v) const

Division operator.

Remarks

this is an element wise division.

Parameters

in	_v	the vector to divide

Returns

the result

10.178.3.16 const Vector2i& gazebo::math::Vector2i::operator/= (const Vector2i & _v)

Division operator.

Remarks

this is an element wise division.

Parameters

in	_v	the vector to divide

Returns

this

10.178.3.17 const Vector2i& gazebo::math::Vector2i::operator/= (int _v)

Division operator.

Remarks

this is an element wise division.

Parameters

in	_v	the vector to divide

Returns

this

10.178.3.18 Vector2i& gazebo::math::Vector2i::operator= (const Vector2i & _v)

Assignment operator.

Parameters

in	_v	the value

Returns

this

10.178.3.19 const Vector2i& gazebo::math::Vector2i::operator= (int _value)

Assignment operator.

Parameters

in	*_value*	the value for x and y

Returns

this

10.178.3.20 bool gazebo::math::Vector2i::operator== (const Vector2i & _v) const

Equality operator.

Parameters

_v	the vector to compare with

Returns

true if component have the same values, false otherwise

10.178.3.21 int gazebo::math::Vector2i::operator[] (unsigned int _index) const

Array subscript operator.

Parameters

in	*_index*	the array index

10.178.3.22 void gazebo::math::Vector2i::Set (int _x, int _y)

Set the contents of the vector.

Parameters

in	*_x*	value along x
in	*_y*	value along y

10.178.4 Friends And Related Function Documentation

10.178.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector2i & _pt) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*pt*	**Vector2i** (p. 846) to output

Returns

the stream

10.178.4.2 std::istream& operator>> (std::istream & *_in,* gazebo::math::Vector2i & *_pt*) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_pt*	**Vector3** (p. 855) to read values into

Returns

The stream

10.178.5 Member Data Documentation

10.178.5.1 int gazebo::math::Vector2i::x

x data

10.178.5.2 int gazebo::math::Vector2i::y

y data

The documentation for this class was generated from the following file:

- **Vector2i.hh**

10.179 gazebo::math::Vector3 Class Reference

The **Vector3** (p. 855) class represents the generic vector containing 3 elements.

`#include <math/gzmath.hh>`

Public Member Functions

- **Vector3** ()
  Constructor.
- **Vector3** (const double &_x, const double &_y, const double &_z)
  Constructor.
- **Vector3** (const **Vector3** &_v)
  Copy constructor.
- virtual ~**Vector3** ()
  Destructor.
- void **Correct** ()
  Corrects any nan values.

- **Vector3 Cross** (const **Vector3** &_pt) const

  Return the cross product of this vector and pt.
- double **Distance** (const **Vector3** &_pt) const

  Calc distance to the given point.
- double **Distance** (double _x, double _y, double _z) const

  Calc distance to the given point.
- double **Dot** (const **Vector3** &_pt) const

  Return the dot product of this vector and pt.
- bool **Equal** (const **Vector3** &_v) const

  Equality test.
- **Vector3 GetAbs** () const

  Get the absolute value of the vector.
- double **GetDistToLine** (const **Vector3** &_pt1, const **Vector3** &_pt2)

  Get distance to a line.
- double **GetLength** () const

  Returns the length (magnitude) of the vector \ return the length.
- double **GetMax** () const

  Get the maximum value in the vector.
- double **GetMin** () const

  Get the minimum value in the vector.
- **Vector3 GetPerpendicular** () const

  Return a vector that is perpendicular to this one.
- **Vector3 GetRounded** () const

  Get a rounded version of this vector.
- double **GetSquaredLength** () const

  Return the square of the length (magnitude) of the vector.
- double **GetSum** () const

  Return the sum of the values.
- bool **IsFinite** () const

  See if a point is finite (e.g., not nan)
- **Vector3 Normalize** ()

  Normalize the vector length.
- bool **operator!=** (const **Vector3** &_v) const

  Not equal to operator.
- **Vector3 operator*** (const **Vector3** &_p) const

  Multiplication operator.
- **Vector3 operator*** (double _v) const

  Multiplication operators.
- const **Vector3** & **operator*=** (const **Vector3** &_v)

  Multiplication operators.
- const **Vector3** & **operator*=** (double _v)

  Multiplication operator.
- **Vector3 operator+** (const **Vector3** &_v) const

  Addition operator.
- const **Vector3** & **operator+=** (const **Vector3** &_v)

  Addition assignment operator.
- **Vector3 operator-** () const

Negation operator.

- **Vector3 operator-** (const **Vector3** &_pt) const

  Subtraction operators.
- const **Vector3** & **operator-=** (const **Vector3** &_pt)

  Subtraction operators.
- const **Vector3 operator**/ (const **Vector3** &_pt) const

  Division operator.
- const **Vector3 operator**/ (double _v) const

  Division operator.
- const **Vector3** & **operator**/**=** (const **Vector3** &_pt)

  Division assignment operator.
- const **Vector3** & **operator**/**=** (double _v)

  Division operator.
- **Vector3** & **operator=** (const **Vector3** &_v)

  Assignment operator.
- **Vector3** & **operator=** (double _value)

  Assignment operator.
- bool **operator==** (const **Vector3** &_pt) const

  Equal to operator.
- double **operator[]** (unsigned int index) const

  [] operator
- **Vector3 Round** ()

  Round to near whole number, return the result.
- void **Round** (int _precision)

  Round all values to _precision decimal places.
- void **Set** (double _x=0, double _y=0, double _z=0)

  Set the contents of the vector.
- void **SetToMax** (const **Vector3** &_v)

  Set this vector's components to the maximum of itself and the passed in vector.
- void **SetToMin** (const **Vector3** &_v)

  Set this vector's components to the minimum of itself and the passed in vector.

Static Public Member Functions

- static **Vector3 GetNormal** (const **Vector3** &_v1, const **Vector3** &_v2, const **Vector3** &_v3)

  Get a normal vector to a triangle.

Public Attributes

- double **x**

  X location.
- double **y**

  Y location.
- double **z**

  Z location.

Static Public Attributes

- static const **Vector3 One**

  math::Vector3(1, 1, 1)
- static const **Vector3 UnitX**

  math::Vector3(1, 0, 0)
- static const **Vector3 UnitY**

  math::Vector3(0, 1, 0)
- static const **Vector3 UnitZ**

  math::Vector3(0, 0, 1)
- static const **Vector3 Zero**

  math::Vector3(0, 0, 0)

Friends

- **Vector3 operator**∗ (double _s, const **Vector3** &_v)

  Multiplication operators.
- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector3** &_pt)

  Stream insertion operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector3** &_pt)

  Stream extraction operator.

10.179.1 Detailed Description

The **Vector3** (p. 855) 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.

10.179.2 Constructor & Destructor Documentation

10.179.2.1 gazebo::math::Vector3::Vector3 ( )

Constructor.

Referenced by operator-().

10.179.2.2 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

10.179.2.3 gazebo::math::Vector3::Vector3 (const Vector3 & _v)

Copy constructor.

Parameters

in	_v	a vector

10.179.2.4 virtual gazebo::math::Vector3::~Vector3 ( ) [virtual]

Destructor.

10.179.3 Member Function Documentation

10.179.3.1 void gazebo::math::Vector3::Correct ( ) [inline]

Corrects any nan values.

References x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.179.3.2 Vector3 gazebo::math::Vector3::Cross (const Vector3 & _pt) const

Return the cross product of this vector and pt.

Returns

the product

10.179.3.3 double gazebo::math::Vector3::Distance (const Vector3 & _pt) const

Calc distance to the given point.

Parameters

in	_pt	the point

Returns

the distance

10.179.3.4 double gazebo::math::Vector3::Distance (double _x, double _y, double _z) const

Calc distance to the given point.

Parameters

in	_x	value along x
in	_y	value along y
in	_z	value along z

Returns

the distance

10.179.3.5 double gazebo::math::Vector3::Dot (const Vector3 & _pt) const

Return the dot product of this vector and pt.

Returns

the product

10.179.3.6 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

10.179.3.7 Vector3 gazebo::math::Vector3::GetAbs ( ) const

Get the absolute value of the vector.

Returns

a vector with positive elements

10.179.3.8 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

10.179.3.9 double gazebo::math::Vector3::GetLength ( ) const

Returns the length (magnitude) of the vector \ return the length.

10.179.3.10 double gazebo::math::Vector3::GetMax ( ) const

Get the maximum value in the vector.

Returns

the maximum element

10.179.3.11 double gazebo::math::Vector3::GetMin ( ) const

Get the minimum value in the vector.

Returns

the minimum element

10.179.3.12 static Vector3 gazebo::math::Vector3::GetNormal (const Vector3 & *_v1*, const Vector3 & *_v2*, const Vector3 & *_v3*) [static]

Get a normal vector to a triangle.

Parameters

in	*_v1*	first vertex of the triangle
in	*_v2*	second vertex
in	*_v3*	third vertex

Returns

the normal

10.179.3.13 Vector3 gazebo::math::Vector3::GetPerpendicular ( ) const

Return a vector that is perpendicular to this one.

Returns

an orthogonal vector

10.179.3.14 Vector3 gazebo::math::Vector3::GetRounded ( ) const

Get a rounded version of this vector.

Returns

a rounded vector

10.179.3.15 double gazebo::math::Vector3::GetSquaredLength ( ) const

Return the square of the length (magnitude) of the vector.

Returns

the squared length

10.179.3.16 double gazebo::math::Vector3::GetSum ( ) const

Return the sum of the values.

Returns

the sum

10.179.3.17 bool gazebo::math::Vector3::IsFinite ( ) const

See if a point is finite (e.g., not nan)

10.179.3.18 Vector3 gazebo::math::Vector3::Normalize ( )

Normalize the vector length.

Returns

unit length vector

10.179.3.19 bool gazebo::math::Vector3::operator!= (const Vector3 & _v) const

Not equal to operator.

Parameters

in	_v	The vector to compare against

Returns

true if each component is equal withing a default tolerence (1e-6), false otherwise

10.179.3.20 Vector3 gazebo::math::Vector3::operator* (const Vector3 & _p) const

Multiplication operator.

Remarks

this is an element wise multiplication, not a cross product

Parameters

in	_v	

10.179.3.21 Vector3 gazebo::math::Vector3::operator∗ (double _v) const

Multiplication operators.

Parameters

in	_v	the scaling factor

Returns

a scaled vector

10.179.3.22 const Vector3& gazebo::math::Vector3::operator∗= (const Vector3 & _v)

Multiplication operators.

Remarks

this is an element wise multiplication, not a cross product

Parameters

in	_v	a vector

Returns

this

10.179.3.23 const Vector3& gazebo::math::Vector3::operator∗= (double _v)

Multiplication operator.

Parameters

in	_v	scaling factor

Returns

this

10.179.3.24 Vector3 gazebo::math::Vector3::operator+ (const Vector3 & _v) const

Addition operator.

Parameters

in	*_v*	vector to add

Returns

the sum vector

10.179.3.25 const Vector3& gazebo::math::Vector3::operator+= (const Vector3 & _v)

Addition assignment operator.

Parameters

in	*_v*	vector to add

10.179.3.26 Vector3 gazebo::math::Vector3::operator- ( ) const [inline]

Negation operator.

Returns

negative of this vector

References Vector3(), x, y, and z.

10.179.3.27 Vector3 gazebo::math::Vector3::operator- (const Vector3 & _pt) const [inline]

Subtraction operators.

Parameters

in	*_pt*	a vector to substract

Returns

a vector

References Vector3(), x, y, and z.

10.179.3.28 const Vector3& gazebo::math::Vector3::operator-= (const Vector3 & _pt)

Subtraction operators.

Parameters

in	*_pt*	subtrahend

10.179.3.29 const Vector3 gazebo::math::Vector3::operator/ (const Vector3 & _pt) const

Division operator.

[in] _pt the vector divisor

Remarks

this is an element wise division

Returns

a vector

10.179.3.30 const Vector3 gazebo::math::Vector3::operator/ (double _v) const

Division operator.

Remarks

this is an element wise division

Returns

a vector

10.179.3.31 const Vector3& gazebo::math::Vector3::operator/= (const Vector3 & _pt)

Division assignment operator.

[in] _pt the vector divisor

Remarks

this is an element wise division

Returns

a vector

10.179.3.32 const Vector3& gazebo::math::Vector3::operator/= (double _v)

Division operator.

Remarks

this is an element wise division

Returns

this

10.179.3.33 Vector3& gazebo::math::Vector3::operator= (const Vector3 & _v)

Assignment operator.

Parameters

in	_v	a new value

Returns

this

10.179.3.34 Vector3& gazebo::math::Vector3::operator= (double _value)

Assignment operator.

Parameters

in	_value	assigned to all elements

Returns

this

10.179.3.35 bool gazebo::math::Vector3::operator== (const Vector3 & _pt) const

Equal to operator.

Parameters

in	_pt	The vector to compare against

Returns

true if each component is equal withing a default tolerence (1e-6), false otherwise

10.179.3.36 double gazebo::math::Vector3::operator[] (unsigned int index) const

[] operator

10.179.3.37 Vector3 gazebo::math::Vector3::Round ( )

Round to near whole number, return the result.

Returns

the result

10.179.3.38 void gazebo::math::Vector3::Round (int _precision)

Round all values to _precision decimal places.

Parameters

in	*_precision*	the decimal places

10.179.3.39 void gazebo::math::Vector3::Set (double _x = 0, double _y = 0, double _z = 0) [inline]

Set the contents of the vector.

Parameters

in	*_x*	value along x
in	*_y*	value along y
in	*_z*	value aling z

References x, y, and z.

10.179.3.40 void gazebo::math::Vector3::SetToMax (const Vector3 & _v)

Set this vector's components to the maximum of itself and the passed in vector.

Parameters

in	*_v*	the maximum clamping vector

10.179.3.41 void gazebo::math::Vector3::SetToMin (const Vector3 & _v)

Set this vector's components to the minimum of itself and the passed in vector.

Parameters

in	*_v*	the minimum clamping vector

10.179.4 Friends And Related Function Documentation

10.179.4.1 Vector3 operator* (double _s, const Vector3 & _v) [friend]

Multiplication operators.

Parameters

in	*_s*	the scaling factor
in	*_v*	input vector

Returns

a scaled vector

10.179.4.2 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector3 & _pt) [friend]

Stream insertion operator.

Parameters

_out	output stream
_pt	**Vector3** (p. 855) to output

Returns

the stream

10.179.4.3 std::istream& operator>> (std::istream & _in, gazebo::math::Vector3 & _pt) [friend]

Stream extraction operator.

Parameters

_in	input stream
_pt	vector3 to read values into

Returns

the stream

10.179.5 Member Data Documentation

10.179.5.1 const Vector3 gazebo::math::Vector3::One [static]

math::Vector3(1, 1, 1)

10.179.5.2 const Vector3 gazebo::math::Vector3::UnitX [static]

math::Vector3(1, 0, 0)

10.179.5.3 const Vector3 gazebo::math::Vector3::UnitY [static]

math::Vector3(0, 1, 0)

10.179.5.4 const Vector3 gazebo::math::Vector3::UnitZ [static]

math::Vector3(0, 0, 1)

10.179.5.5 double gazebo::math::Vector3::x

X location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), operator-(), gazebo::math::Quaternion::RotateVector(), and Set().

10.179.5.6 double gazebo::math::Vector3::y

Y location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), operator-(), gazebo::math::Quaternion::RotateVector(), and Set().

10.179.5.7 double gazebo::math::Vector3::z

Z location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), operator-(), gazebo::math::Quaternion::RotateVector(), and Set().

10.179.5.8 const Vector3 gazebo::math::Vector3::Zero [static]

math::Vector3(0, 0, 0)

The documentation for this class was generated from the following file:

- **Vector3.hh**

10.180 gazebo::math::Vector4 Class Reference

double Generic x, y, z, w vector

`#include <math/gzmath.hh>`

Public Member Functions

- **Vector4** ()

  Constructor.
- **Vector4** (const double &_x, const double &_y, const double &_z, const double &_w)

  Constructor with component values.
- **Vector4** (const **Vector4** &_v)

  Copy constructor.
- virtual ~**Vector4** ()

  Destructor.
- double **Distance** (const **Vector4** &_pt) const

  Calc distance to the given point.
- double **GetLength** () const

  Returns the length (magnitude) of the vector.
- double **GetSquaredLength** () const

  Return the square of the length (magnitude) of the vector.
- bool **IsFinite** () const

  See if a point is finite (e.g., not nan)
- void **Normalize** ()

  Normalize the vector length.
- bool **operator!=** (const **Vector4** &_pt) const

Not equal to operator.

- const **Vector4 operator**∗ (const **Vector4** &_pt) const

  Multiplication operator.
- const **Vector4 operator**∗ (const **Matrix4** &_m) const

  Matrix multiplication operator.
- const **Vector4 operator**∗ (double _v) const

  Multiplication operators.
- const **Vector4** & **operator∗=** (const **Vector4** &_pt)

  Multiplication assignment operator.
- const **Vector4** & **operator∗=** (double _v)

  Multiplication assignment operator.
- **Vector4 operator+** (const **Vector4** &_v) const

  Addition operator.
- const **Vector4** & **operator+=** (const **Vector4** &_v)

  Addition operator.
- **Vector4 operator-** (const **Vector4** &_v) const

  Subtraction operator.
- const **Vector4** & **operator-=** (const **Vector4** &_v)

  Subtraction assigment 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.

10.180.1 Detailed Description

double Generic x, y, z, w vector

10.180.2 Constructor & Destructor Documentation

10.180.2.1 gazebo::math::Vector4::Vector4 ( )

Constructor.

10.180.2.2 gazebo::math::Vector4::Vector4 (const double & *_x*, const double & *_y*, const double & *_z*, const double & *_w*)

Constructor with component values.

Parameters

in	*_x*	value along x axis
in	*_y*	value along y axis
in	*_z*	value along z axis
in	*_w*	value along w axis

10.180.2.3 gazebo::math::Vector4::Vector4 (const Vector4 & *_v*)

Copy constructor.

Parameters

in	*_v*	vector

10.180.2.4 virtual gazebo::math::Vector4::~Vector4 ( ) [virtual]

Destructor.

10.180.3 Member Function Documentation

10.180.3.1 double gazebo::math::Vector4::Distance (const Vector4 & *_pt*) const

Calc distance to the given point.

Parameters

in	*_pt*	the point

Returns

the distance

10.180.3.2 double gazebo::math::Vector4::GetLength ( ) const

Returns the length (magnitude) of the vector.

10.180.3.3 double gazebo::math::Vector4::GetSquaredLength ( ) const

Return the square of the length (magnitude) of the vector.

Returns

the length

10.180.3.4 bool gazebo::math::Vector4::IsFinite ( ) const

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.180.3.5 void gazebo::math::Vector4::Normalize ( )

Normalize the vector length.

10.180.3.6 bool gazebo::math::Vector4::operator!= (const Vector4 & *_pt*) const

Not equal to operator.

Parameters

in	*_pt*	the other vector

Returns

true if each component is equal withing a default tolerence (1e-6), false otherwise

10.180.3.7 const Vector4 gazebo::math::Vector4::operator* (const Vector4 & *_pt*) const

Multiplication operator.

Remarks

Performs element wise multiplication, which has limited use.

Parameters

in	*_pt*	another vector

Returns

result vector

10.180.3.8 const Vector4 gazebo::math::Vector4::operator∗ (const Matrix4 & *_m*) const

Matrix multiplication operator.

Parameters

in	*_m*	matrix

Returns

the vector multiplied by _m

10.180.3.9 const Vector4 gazebo::math::Vector4::operator∗ (double *_v*) const

Multiplication operators.

Parameters

in	*_v*	scaling factor

Returns

a scaled vector

10.180.3.10 const Vector4& gazebo::math::Vector4::operator∗= (const Vector4 & *_pt*)

Multiplication assignment operator.

Remarks

Performs element wise multiplication, which has limited use.

Parameters

in	*_pt*	a vector

Returns

this

10.180.3.11 const Vector4& gazebo::math::Vector4::operator*= (double _v)

Multiplication assignment operator.

Parameters

in	_v	scaling factor

Returns

this

10.180.3.12 Vector4 gazebo::math::Vector4::operator+ (const Vector4 & _v) const

Addition operator.

Parameters

in	_v	the vector to add

Returns

a sum vector

10.180.3.13 const Vector4& gazebo::math::Vector4::operator+= (const Vector4 & _v)

Addition operator.

Parameters

in	_v	the vector to add

Returns

this vector

10.180.3.14 Vector4 gazebo::math::Vector4::operator- (const Vector4 & _v) const

Subtraction operator.

Parameters

in	_v	the vector to substract

Returns

a vector

10.180.3.15 const Vector4& gazebo::math::Vector4::operator-= (const Vector4 & _v)

Subtraction assigment operators.

Parameters

in	_v	the vector to substract

Returns

this vector

10.180.3.16 const Vector4 gazebo::math::Vector4::operator/ (const Vector4 & _v) const

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

in	_v	the vector to perform element wise division with

Returns

a result vector

10.180.3.17 const Vector4 gazebo::math::Vector4::operator/ (double _v) const

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

in	_pt	another vector

Returns

a result vector

10.180.3.18 const Vector4& gazebo::math::Vector4::operator/= (const Vector4 & _v)

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

in	_v	the vector to perform element wise division with

Returns

this

10.180.3.19 const Vector4& gazebo::math::Vector4::operator/= (double _v)

Division operator.

Parameters

in	_v	scaling factor

Returns

a vector

10.180.3.20 Vector4& gazebo::math::Vector4::operator= (const Vector4 & _v)

Assignment operator.

Parameters

in	_v	the vector

Returns

a reference to this vector

10.180.3.21 Vector4& gazebo::math::Vector4::operator= (double _value)

Assignment operator.

Parameters

in	_value	

10.180.3.22 bool gazebo::math::Vector4::operator== (const Vector4 & _pt) const

Equal to operator.

Parameters

in	*_pt*	the other vector

Returns

true if each component is equal withing a default tolerence (1e-6), false otherwise

10.180.3.23 double gazebo::math::Vector4::operator[] (unsigned int _index) const

Array subscript operator.

Parameters

in	*_index*	

10.180.3.24 void gazebo::math::Vector4::Set (double _x = 0, double _y = 0, double _z = 0, double _w = 0)

Set the contents of the vector.

Parameters

in	*_x*	value along x axis
in	*_y*	value along y axis
in	*_z*	value along z axis
in	*_w*	value along w axis

10.180.4 Friends And Related Function Documentation

10.180.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector4 & _pt) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_pt*	**Vector4** (p. 869) to output

Returns

The stream

10.180.4.2 std::istream& operator>> (std::istream & _in, gazebo::math::Vector4 & _pt) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_pt*	**Vector4** (p. 869) to read values into

Returns

the stream

10.180.5 Member Data Documentation

10.180.5.1 double gazebo::math::Vector4::w

W value.

10.180.5.2 double gazebo::math::Vector4::x

X value.

10.180.5.3 double gazebo::math::Vector4::y

Y value.

10.180.5.4 double gazebo::math::Vector4::z

Z value.

The documentation for this class was generated from the following file:

- **Vector4.hh**

10.181 gazebo::common::Video Class Reference

Handle video encoding and decoding using libavcodec.

`#include <common/common.hh>`

Public Member Functions

- **Video** ()

  Constructor.
- virtual ∼**Video** ()

  Destructor.
- int **GetHeight** () const

  Get the height of the video in pixels.
- bool **GetNextFrame** (unsigned char **_buffer)

  Get the next frame of the video.
- int **GetWidth** () const

Get the width of the video in pixels.

- bool **Load** (const std::string &_filename)

  Load a video file.

10.181.1 Detailed Description

Handle video encoding and decoding using libavcodec.

10.181.2 Constructor & Destructor Documentation

10.181.2.1 gazebo::common::Video::Video ( )

Constructor.

10.181.2.2 virtual gazebo::common::Video::~Video ( ) [virtual]

Destructor.

10.181.3 Member Function Documentation

10.181.3.1 int gazebo::common::Video::GetHeight ( ) const

Get the height of the video in pixels.

Returns

the height

10.181.3.2 bool gazebo::common::Video::GetNextFrame (unsigned char ** *_buffer*)

Get the next frame of the video.

Parameters

out	*_img*	**Image** (p. 360) in which the frame is stored

Returns

false if HAVE_FFMPEG is not defined, true otherwise

10.181.3.3 int gazebo::common::Video::GetWidth ( ) const

Get the width of the video in pixels.

Returns

the width

10.181.3.4 bool gazebo::common::Video::Load (const std::string & _filename)

Load a video file.

Parameters

in	*_filename*	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:

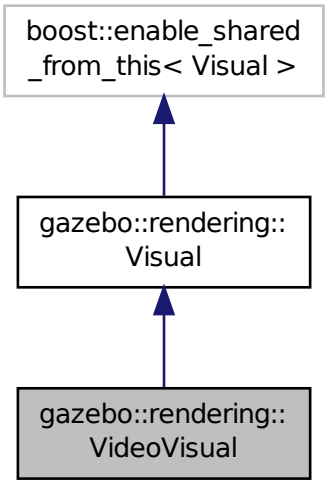
- **Video.hh**

10.182 gazebo::rendering::VideoVisual Class Reference

A (p. 111) 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

10.182.1 Detailed Description

A (p. 111) visual element that displays a video as a texture.

10.182.2 Constructor & Destructor Documentation

10.182.2.1 gazebo::rendering::VideoVisual::VideoVisual (const std::string & *_name,* VisualPtr *_parent*)

Constructor.

Parameters

in	*_name*	Name of the video visual.
in	*_parent*	Parent of the video visual.

10.182.2.2 virtual gazebo::rendering::VideoVisual::~VideoVisual ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

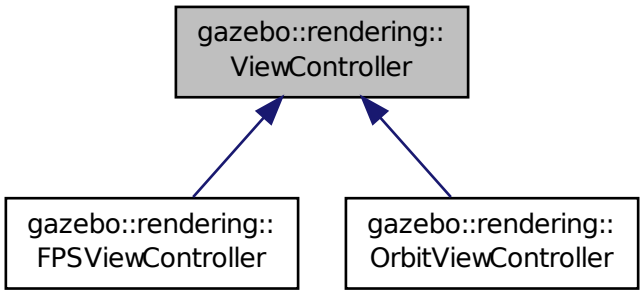
- **VideoVisual.hh**

10.183 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. *162*).

Protected Attributes

- **UserCameraPtr camera**

  Pointer to the camera to control.
- bool **enabled**

  True if enabled.
- std::string **typeString**

  Type of view controller.

10.183.1 Detailed Description

Base class for view controllers.

10.183.2 Constructor & Destructor Documentation

10.183.2.1 gazebo::rendering::ViewController::ViewController (UserCameraPtr *_camera*)

Constructor.

Parameters

in	*_camera*	The user camera to controll.

10.183.2.2 virtual gazebo::rendering::ViewController::~ViewController ( ) [virtual]

Destructor.

10.183.3 Member Function Documentation

10.183.3.1 std::string gazebo::rendering::ViewController::GetTypeString ( ) const

Get the type of view controller.

Returns

The view controller type string.

10.183.3.2 virtual void gazebo::rendering::ViewController::HandleKeyPressEvent (const std::string & _key) [pure virtual]

Handle a key press event.

Parameters

in	*_key*	The key that was pressed.

Implemented in **gazebo::rendering::OrbitViewController** (p. 557), and **gazebo::rendering::FPSViewController** (p. 322).

10.183.3.3 virtual void gazebo::rendering::ViewController::HandleKeyReleaseEvent (const std::string & _key) [pure virtual]

Handle a key release event.

Parameters

in	*_key*	The key that was released.

Implemented in **gazebo::rendering::OrbitViewController** (p. 558), and **gazebo::rendering::FPSViewController** (p. 322).

10.183.3.4 virtual void gazebo::rendering::ViewController::HandleMouseEvent (const common::MouseEvent & _event) [pure virtual]

Handle a mouse event.

Parameters

in	*_event*	The mouse position.

Implemented in **gazebo::rendering::OrbitViewController** (p. 558), and **gazebo::rendering::FPSViewController** (p. 322).

10.183.3.5 virtual void gazebo::rendering::ViewController::Init ( ) [pure virtual]

Initialize the view controller.

Implemented in **gazebo::rendering::OrbitViewController** (p. 558), and **gazebo::rendering::FPSViewController** (p. 323).

10.183.3.6 virtual void gazebo::rendering::ViewController::Init (const math::Vector3 & *_focalPoint*) [virtual]

Initialize with a focus point.

Parameters

in	*_focalPoint*	The point to look at.

Reimplemented in **gazebo::rendering::OrbitViewController** (p. 558).

10.183.3.7 void gazebo::rendering::ViewController::SetEnabled (bool *_value*)

Set whether the controller is enabled.

Parameters

in	*_value*	True if the controller is enabled.

10.183.3.8 virtual void gazebo::rendering::ViewController::Update ( ) [pure virtual]

Update the controller, which should update the position of the **Camera** (p. 162).

Implemented in **gazebo::rendering::OrbitViewController** (p. 559), and **gazebo::rendering::FPSViewController** (p. 323).

10.183.4 Member Data Documentation

10.183.4.1 UserCameraPtr gazebo::rendering::ViewController::camera [protected]

Pointer to the camera to control.

10.183.4.2 bool gazebo::rendering::ViewController::enabled [protected]

True if enabled.

10.183.4.3 std::string gazebo::rendering::ViewController::typeString [protected]

Type of view controller.

The documentation for this class was generated from the following file:

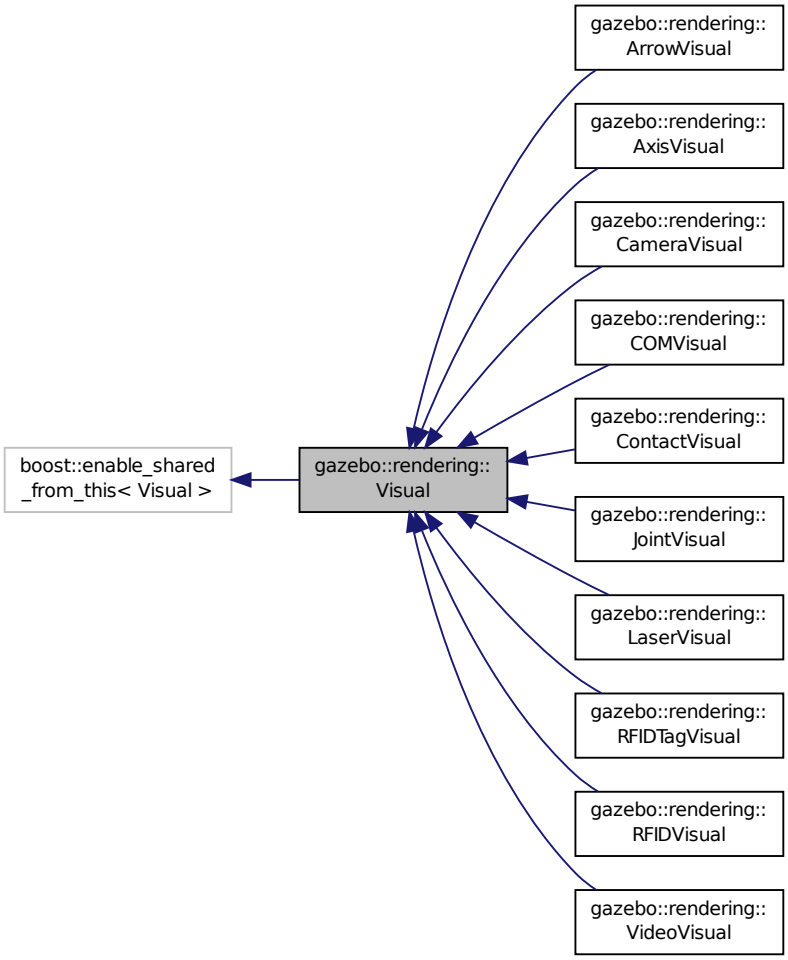
- **ViewController.hh**

10.184 gazebo::rendering::Visual Class Reference

A (p. 111) renderable object.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::Visual:



Public Member Functions

- **Visual** (const std::string &_name, **VisualPtr** _parent, bool _useRTShader=true)

  Constructor.
- **Visual** (const std::string &_name, **ScenePtr** _scene, bool _useRTShader=true)

  Constructor.
- virtual ~**Visual** ()

  Destructor.
- void **AttachAxes** ()

Attach visualization axes.

- void **AttachLineVertex** (**DynamicLines** *_line, unsigned int _index)

  Attach a vertex of a line to the position of the visual.

- Ogre::MovableObject * **AttachMesh** (const std::string &_meshName, const std::string &_subMesh="", bool _centerSubmesh=false, const std::string &_objName="")

  Attach a mesh to this visual by name.

- void **AttachObject** (Ogre::MovableObject *_obj)

  Attach a renerable object to the visual.

- void **AttachVisual** (**VisualPtr** _vis)

  Attach a visual to this visual.

- void **ClearParent** ()

  Clear parents.

- **VisualPtr Clone** (const std::string &_name, **VisualPtr** _newParent)

  Clone the visual with a new name.

- **DynamicLines** * **CreateDynamicLine** (**RenderOpType** _type=**RENDERING_LINE_STRIP**)

  Add a line to the visual.

- void **DeleteDynamicLine** (**DynamicLines** *_line)

  Delete a dynamic line.

- void **DetachObjects** ()

  Detach all objects.

- void **DetachVisual** (**VisualPtr** _vis)

  Detach a visual.

- void **DetachVisual** (const std::string &_name)

  Detach a visual.

- void **DisableTrackVisual** ()

  Disable tracking of a visual.

- void **EnableTrackVisual** (**VisualPtr** _vis)

  Set one visual to track/follow another.

- void **Fini** ()

  Helper for the destructor.

- unsigned int **GetAttachedObjectCount** () const

  Return the number of attached movable objects.

- **math::Box GetBoundingBox** () const

  Get the bounding box for the visual.

- **VisualPtr GetChild** (unsigned int _index)

  Get an attached visual based on an index.

- unsigned int **GetChildCount** ()

  Get the number of attached visuals.

- std::string **GetMaterialName** () const

  Get the name of the material.

- std::string **GetMeshName** () const

  The name of the mesh set in the visual's SDF.

- std::string **GetName** () const

  Get the name of the visual.

- std::string **GetNormalMap** () const

  Get the normal map.

- **VisualPtr GetParent** () const

Get the parent visual, if one exists.

- **math::Pose GetPose** () const

  Get the pose of the visual.
- **math::Vector3 GetPosition** () const

  Get the position of the visual.
- **VisualPtr GetRootVisual** ()

  Get the root visual.
- **math::Quaternion GetRotation** () const

  Get the rotation of the visual.
- **math::Vector3 GetScale** ()

  Get the scale.
- **ScenePtr GetScene** () const

  Get current.
- Ogre::SceneNode * **GetSceneNode** () const

  Return the scene Node of this visual entity.
- std::string **GetShaderType** () const

  Get the shader type.
- std::string **GetSubMeshName** () const

  Get the name of the sub mesh set in the visual's SDF.
- float **GetTransparency** ()

  Get the transparency.
- uint32_t **GetVisibilityFlags** ()

  Get visibility flags for this visual and all children.
- bool **GetVisible** () const

  Get whether the visual is visible.
- **math::Pose GetWorldPose** () const

  Get the global pose of the node.
- bool **HasAttachedObject** (const std::string &_name)

  Returns true if an object with _name is attached.
- void **Init** ()

  Helper for the contructor.
- void **InsertMesh** (const std::string &_meshName, const std::string &_subMesh="", bool _centerSubmesh=false)

  *Insert a mesh into **Ogre** (p. 106).*
- bool **IsPlane** () const

  Return true if the visual is a plane.
- bool **IsStatic** () const

  Return true if the visual is a static geometry.
- void **Load** (**sdf::ElementPtr** _sdf)

  Load the visual with a set of parameters.
- virtual void **Load** ()

  Load the visual with default parameters.
- void **LoadFromMsg** (ConstVisualPtr &_msg)

  Load from a message.
- void **LoadPlugin** (const std::string &_filename, const std::string &_name, **sdf::ElementPtr** _sdf)

  Load a plugin.
- void **MakeStatic** ()

  Make the visual objects static renderables.

- void **MoveToPosition** (const **math::Pose** &_pose, double _time)

  Move to a pose and over a given time.
- void **MoveToPositions** (const std::vector< **math::Pose** > &_pts, double _time, boost::function< void()> _onComplete=**NULL**)

  Move to a series of pose and over a given time.
- void **RemovePlugin** (const std::string &_name)

  Remove a running plugin.
- void **SetAmbient** (const **common::Color** &_color)

  Set the ambient color of the visual.
- void **SetCastShadows** (bool _shadows)

  Set whether the visual should cast shadows.
- void **SetDiffuse** (const **common::Color** &_color)

  Set the diffuse color of the visual.
- virtual void **SetEmissive** (const **common::Color** &_color)

  Set the emissive value.
- void **SetHighlighted** (bool _highlighted)

  Set the visual to be visually highlighted.
- void **SetMaterial** (const std::string &_materialName, bool _unique=true)

  Set the material.
- void **SetName** (const std::string &_name)

  Set the name of the visual.
- void **SetNormalMap** (const std::string &_nmap)

  Set the normal map.
- void **SetPose** (const **math::Pose** &_pose)

  Set the pose of the visual.
- void **SetPosition** (const **math::Vector3** &_pos)

  Set the position of the visual.
- void **SetRibbonTrail** (bool _value, const **common::Color** &_initialColor, const **common::Color** &_changeColor)

  True on or off a ribbon trail.
- void **SetRotation** (const **math::Quaternion** &_rot)

  Set the rotation of the visual.
- void **SetScale** (const **math::Vector3** &_scale)

  Set the scale.
- void **SetScene** (**ScenePtr** _scene)

  Set current scene.
- void **SetShaderType** (const std::string &_type)

  Set the shader type for the visual's material.
- void **SetSkeletonPose** (const msgs::PoseAnimation &_pose)

  Set animation skeleton pose.
- void **SetSpecular** (const **common::Color** &_color)

  Set the specular color of the visual.
- void **SetTransparency** (float _trans)

  Set the transparency.
- void **SetVisibilityFlags** (uint32_t _flags)

  Set visibility flags for this visual and all children.
- void **SetVisible** (bool _visible, bool _cascade=true)

Set whether the visual is visible.

- void **SetWireframe** (bool _show)

  Enable or disable wireframe for this visual.

- void **SetWorldPose** (const **math::Pose** _pose)

  Set the world pose of the visual.

- void **SetWorldPosition** (const **math::Vector3** &_pos)

  Set the world linear position of the visual.

- void **SetWorldRotation** (const **math::Quaternion** &_rot)

  Set the world orientation of the visual.

- void **ShowBoundingBox** ()

  Display the bounding box visual.

- void **ShowCollision** (bool _show)

  Display the collision visuals.

- void **ShowCOM** (bool _show)

  Display Center of Mass visuals.

- void **ShowJoints** (bool _show)

  Display joint visuals.

- void **ShowSkeleton** (bool _show)

  Display the skeleton visuals.

- void **ToggleVisible** ()

  Toggle whether this visual is visible.

- void **Update** ()

  Update the visual.

- void **UpdateFromMsg** (ConstVisualPtr &_msg)

  Update a visual based on a message.

Static Public Member Functions

- static void **InsertMesh** (const **common::Mesh** *_mesh, const std::string &_subMesh="", bool _centerSubmesh=false)

  *Insert a mesh into **Ogre*** (p. *106*).

Protected Attributes

- **VisualPtr parent**

  Parent visual.

- **ScenePtr scene**

  Pointer to the visual's scene.

- Ogre::SceneNode * **sceneNode**

  *Pointer to the visual's scene node in **Ogre*** (p. *106*).

10.184.1 Detailed Description

A (p. 111) renderable object.

10.184.2 Constructor & Destructor Documentation

10.184.2.1 gazebo::rendering::Visual::Visual (const std::string & *_name*, VisualPtr *_parent*, bool *_useRTShader* = true)

Constructor.

Parameters

in	*_name*	Name of the visual.
in	*_parent*	Parent of the visual.
in	*_useRTShader*	True if the visual should use the real-time shader system (RTShader).

10.184.2.2 gazebo::rendering::Visual::Visual (const std::string & *_name*, ScenePtr *_scene*, bool *_useRTShader* = true)

Constructor.

Parameters

in	*_name*	Name of the visual.
in	*_scene*	**Scene** (p. 676) containing the visual.
in	*_useRTShader*	True if the visual should use the real-time shader system (RTShader).

10.184.2.3 virtual gazebo::rendering::Visual::~Visual ( ) [virtual]

Destructor.

10.184.3 Member Function Documentation

10.184.3.1 void gazebo::rendering::Visual::AttachAxes ( )

Attach visualization axes.

10.184.3.2 void gazebo::rendering::Visual::AttachLineVertex (DynamicLines * *_line*, unsigned int *_index*)

Attach a vertex of a line to the position of the visual.

Parameters

in	*_line*	Line to attach to this visual.
in	*_index*	Index of the line vertex to attach.

10.184.3.3 Ogre::MovableObject* gazebo::rendering::Visual::AttachMesh (const std::string & *_meshName*, const std::string & *_subMesh* = " ", bool *_centerSubmesh* = false, const std::string & *_objName* = " ")

Attach a mesh to this visual by name.

Parameters

in	*_meshName*	Name of the mesh.

in	*_subMesh*	Name of the submesh. Empty string to use all submeshes.
in	*_centerSubmesh*	True to center a submesh.
in	*_objName*	Name of the attached Object to put the mesh onto.

10.184.3.4 void gazebo::rendering::Visual::AttachObject (Ogre::MovableObject * *_obj*)

Attach a renerable object to the visual.

Parameters

in	*_obj*	**A** (p. 111) movable object to attach to the visual.

10.184.3.5 void gazebo::rendering::Visual::AttachVisual (VisualPtr *_vis*)

Attach a visual to this visual.

Parameters

in	*_vis*	**Visual** (p. 885) to attach.

10.184.3.6 void gazebo::rendering::Visual::ClearParent ( )

Clear parents.

10.184.3.7 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. 885).
in	*_newParent*	Parent of the cloned **Visual** (p. 885).

Returns

The visual.

10.184.3.8 DynamicLines* gazebo::rendering::Visual::CreateDynamicLine (RenderOpType *_type* = RENDERING_LINE_STRIP)

Add a line to the visual.

Parameters

in	*_type*	The type of line to make.

Returns

A (p. 111) pointer to the new dynamic line.

10.184.3.9 void gazebo::rendering::Visual::DeleteDynamicLine (DynamicLines * _line)

Delete a dynamic line.

Parameters

in	*_line*	Pointer to the line to delete.

10.184.3.10 void gazebo::rendering::Visual::DetachObjects ( )

Detach all objects.

10.184.3.11 void gazebo::rendering::Visual::DetachVisual (VisualPtr _vis)

Detach a visual.

Parameters

in	*_vis*	**Visual** (p. 885) to detach.

10.184.3.12 void gazebo::rendering::Visual::DetachVisual (const std::string & _name)

Detach a visual.

Parameters

in	*_name*	Name of the visual to detach.

10.184.3.13 void gazebo::rendering::Visual::DisableTrackVisual ( )

Disable tracking of a visual.

10.184.3.14 void gazebo::rendering::Visual::EnableTrackVisual (VisualPtr _vis)

Set one visual to track/follow another.

Parameters

in	*_vis*	**Visual** (p. 885) to track.

10.184.3.15 void gazebo::rendering::Visual::Fini ( )

Helper for the destructor.

10.184.3.16 unsigned int gazebo::rendering::Visual::GetAttachedObjectCount ( ) const

Return the number of attached movable objects.

Returns

The number of attached movable objects.

10.184.3.17 math::Box gazebo::rendering::Visual::GetBoundingBox ( ) const

Get the bounding box for the visual.

Returns

The bounding box in world coordinates.

10.184.3.18 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. 893).

Parameters

in	*_index*	Index of the child to retreive.

Returns

Pointer to the child visual, NULL if index is invalid.

10.184.3.19 unsigned int gazebo::rendering::Visual::GetChildCount ( )

Get the number of attached visuals.

Returns

The number of children.

10.184.3.20 std::string gazebo::rendering::Visual::GetMaterialName ( ) const

Get the name of the material.

Returns

The name of the visual applied to this visual.

10.184.3.21 std::string gazebo::rendering::Visual::GetMeshName ( ) const

The name of the mesh set in the visual's SDF.

Returns

Name of the mesh.

10.184.3.22 std::string gazebo::rendering::Visual::GetName ( ) const

Get the name of the visual.

Returns

The name of the visual.

10.184.3.23 std::string gazebo::rendering::Visual::GetNormalMap ( ) const

Get the normal map.

Returns

The name of the normal map material.

10.184.3.24 VisualPtr gazebo::rendering::Visual::GetParent ( ) const

Get the parent visual, if one exists.

Returns

Pointer to the parent visual, NULL if no parent.

10.184.3.25 math::Pose gazebo::rendering::Visual::GetPose ( ) const

Get the pose of the visual.

Returns

The **Visual** (p. 885)'s pose.

10.184.3.26 math::Vector3 gazebo::rendering::Visual::GetPosition ( ) const

Get the position of the visual.

Returns

The visual's position.

10.184.3.27 VisualPtr gazebo::rendering::Visual::GetRootVisual ( )

Get the root visual.

Returns

The root visual, which is one level below the world visual.

10.184.3.28 math::Quaternion gazebo::rendering::Visual::GetRotation ( ) const

Get the rotation of the visual.

Returns

The visual's rotation.

10.184.3.29 math::Vector3 gazebo::rendering::Visual::GetScale ( )

Get the scale.

Returns

The scaling factor.

10.184.3.30 ScenePtr gazebo::rendering::Visual::GetScene ( ) const

Get current.

Returns

Pointer to the scene.

10.184.3.31 Ogre::SceneNode* gazebo::rendering::Visual::GetSceneNode ( ) const

Return the scene Node of this visual entity.

Returns

The **Ogre** (p. 106) scene node.

10.184.3.32 std::string gazebo::rendering::Visual::GetShaderType ( ) const

Get the shader type.

Returns

String of the shader type: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".

10.184.3.33 std::string gazebo::rendering::Visual::GetSubMeshName ( ) const

Get the name of the sub mesh set in the visual's SDF.

Returns

Name of the submesh. Empty string if no submesh is specified.

10.184.3.34 float gazebo::rendering::Visual::GetTransparency ( )

Get the transparency.

Returns

The transparency.

10.184.3.35 uint32_t gazebo::rendering::Visual::GetVisibilityFlags ( )

Get visibility flags for this visual and all children.

Returns

The visiblity flags.

See Also

GZ_VISIBILITY_ALL (p. 1067)
GZ_VISIBILITY_GUI (p. 1067)
GZ_VISIBILITY_NOT_SELECTABLE (p. 1067)

10.184.3.36 bool gazebo::rendering::Visual::GetVisible ( ) const

Get whether the visual is visible.

Returns

True if the visual is visible.

10.184.3.37 math::Pose gazebo::rendering::Visual::GetWorldPose ( ) const

Get the global pose of the node.

Returns

The pose in the world coordinate frame.

10.184.3.38 bool gazebo::rendering::Visual::HasAttachedObject (const std::string & *_name*)

Returns true if an object with _name is attached.

Parameters

in	*_name*	Name of an object to find.

10.184.3.39 void gazebo::rendering::Visual::Init ( )

Helper for the contructor.

10.184.3.40 void gazebo::rendering::Visual::InsertMesh (const std::string & *_meshName,* const std::string & *_subMesh* = " ", bool *_centerSubmesh* = false)

Insert a mesh into **Ogre** (p. 106).

Parameters

in	*_meshName*	Name of the mesh to insert.
in	*_subMesh*	Name of the mesh within _meshName to insert.
in	*_centerSubmesh*	True to center the submesh.

10.184.3.41 static void gazebo::rendering::Visual::InsertMesh (const common::Mesh * *_mesh,* const std::string & *_subMesh* = " ", bool *_centerSubmesh* = false) [static]

Insert a mesh into **Ogre** (p. 106).

Parameters

in	*_mesh*	Pointer to the mesh to insert.
in	*_subMesh*	Name of the mesh within _meshName to insert.
in	*_centerSubmesh*	True to center the submesh.

10.184.3.42 bool gazebo::rendering::Visual::IsPlane ( ) const

Return true if the visual is a plane.

Returns

True if a plane.

10.184.3.43 bool gazebo::rendering::Visual::IsStatic ( ) const

Return true if the visual is a static geometry.

Returns

True if the visual is static.

10.184.3.44 void gazebo::rendering::Visual::Load (sdf::ElementPtr *_sdf*)

Load the visual with a set of parameters.

Parameters

in	*_sdf*	Load from an SDF element.

10.184.3.45 virtual void gazebo::rendering::Visual::Load ( ) [virtual]

Load the visual with default parameters.

Reimplemented in **gazebo::rendering::ArrowVisual** (p. 131), and **gazebo::rendering::AxisVisual** (p. 134).

10.184.3.46 void gazebo::rendering::Visual::LoadFromMsg (ConstVisualPtr & *_msg*)

Load from a message.

Parameters

in	*_msg*	**A** (p. 111) visual message.

10.184.3.47 void gazebo::rendering::Visual::LoadPlugin (const std::string & *_filename*, const std::string & *_name*, sdf::ElementPtr *_sdf*)

Load a plugin.

Parameters

_filename	The filename of the plugin
_name	**A** (p. 111) unique name for the plugin
_sdf	The SDF to pass into the plugin.

10.184.3.48 void gazebo::rendering::Visual::MakeStatic ( )

Make the visual objects static renderables.

10.184.3.49 void gazebo::rendering::Visual::MoveToPosition (const math::Pose & *_pose*, double *_time*)

Move to a pose and over a given time.

Parameters

in	*_pose*	Pose the visual will end at.
in	*_time*	Time it takes the visual to move to the pose.

10.184.3.50 void gazebo::rendering::Visual::MoveToPositions (const std::vector< math::Pose > & *_pts*, double *_time*, boost::function< void() > *_onComplete* = NULL)

Move to a series of pose and over a given time.

Parameters

in	*_poses*	Series of poses the visual will move to.
in	*_time*	Time it takes the visual to move to the pose.
in	*_onComplete*	Callback used when the move is complete.

10.184.3.51 void gazebo::rendering::Visual::RemovePlugin (const std::string & *_name*)

Remove a running plugin.

Parameters

_name	The unique name of the plugin to remove

10.184.3.52 void gazebo::rendering::Visual::SetAmbient (const **common::Color** & *_color*)

Set the ambient color of the visual.

Parameters

in	*_color*	The ambient color.

10.184.3.53 void gazebo::rendering::Visual::SetCastShadows (bool *_shadows*)

Set whether the visual should cast shadows.

Parameters

in	*_shadows*	True to enable shadows.

10.184.3.54 void gazebo::rendering::Visual::SetDiffuse (const **common::Color** & *_color*)

Set the diffuse color of the visual.

Parameters

in	*_color*	Set the diffuse color.

10.184.3.55 virtual void gazebo::rendering::Visual::SetEmissive (const **common::Color** & *_color*) [virtual]

Set the emissive value.

Parameters

in	*_color*	The emissive color.

Reimplemented in **gazebo::rendering::LaserVisual** (p. 412).

10.184.3.56 void gazebo::rendering::Visual::SetHighlighted (bool *_highlighted*)

Set the visual to be visually highlighted.

This is most often used when an object is selected by a user via the GUI.

Parameters

in	*_highlighted*	True to enable the highlighting.

10.184.3.57 void gazebo::rendering::Visual::SetMaterial (const std::string & *_materialName*, bool *_unique* = `true`)

Set the material.

Parameters

in	*_materialName*	The name of the material.
in	*_unique*	True to make the material unique, which allows the material to change without changing materials that originally had the same name.

10.184.3.58 void gazebo::rendering::Visual::SetName (const std::string & *_name*)

Set the name of the visual.

Parameters

in	*_name*	Name of the visual

10.184.3.59 void gazebo::rendering::Visual::SetNormalMap (const std::string & *_nmap*)

Set the normal map.

Parameters

in	*_nmap*	Name of the normal map material.

10.184.3.60 void gazebo::rendering::Visual::SetPose (const **math::Pose** & *_pose*)

Set the pose of the visual.

Parameters

in	*_pose*	The new pose of the visual.

10.184.3.61 void gazebo::rendering::Visual::SetPosition (const **math::Vector3** & *_pos*)

Set the position of the visual.

Parameters

in	*_pos*	The position to set the visual to.

10.184.3.62 void gazebo::rendering::Visual::SetRibbonTrail (bool *_value,* const common::Color & *_initialColor,* const common::Color & *_changeColor*)

True on or off a ribbon trail.

Parameters

in	*_value*	True to enable ribbon trail.
in	*_initialColor*	The initial color of the ribbon trail.
in	*_changeColor*	Color to change too as the trail grows.

10.184.3.63 void gazebo::rendering::Visual::SetRotation (const math::Quaternion & *_rot*)

Set the rotation of the visual.

Parameters

in	*_rot*	The rotation of the visual.

10.184.3.64 void gazebo::rendering::Visual::SetScale (const math::Vector3 & *_scale*)

Set the scale.

Parameters

in	*_scale*	The scaling factor for the visual.

10.184.3.65 void gazebo::rendering::Visual::SetScene (ScenePtr *_scene*)

Set current scene.

Parameters

in	*_scene*	Pointer to the scene.

10.184.3.66 void gazebo::rendering::Visual::SetShaderType (const std::string & *_type*)

Set the shader type for the visual's material.

Parameters

in	*_type*	Shader type string: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".

10.184.3.67 void gazebo::rendering::Visual::SetSkeletonPose (const msgs::PoseAnimation & *_pose*)

Set animation skeleton pose.

Parameters

in	*_pose*	Skelton message

10.184.3.68 void gazebo::rendering::Visual::SetSpecular (const common::Color & *_color*)

Set the specular color of the visual.

Parameters

in	*_color*	Specular color.

10.184.3.69 void gazebo::rendering::Visual::SetTransparency (float *_trans*)

Set the transparency.

Parameters

in	*_trans*	The transparency, between 0 and 1 where 0 is no transparency.

10.184.3.70 void gazebo::rendering::Visual::SetVisibilityFlags (uint32_t *_flags*)

Set visibility flags for this visual and all children.

Parameters

in	*_flags*	The visiblity flags.

See Also

GZ_VISIBILITY_ALL (p. 1067)
GZ_VISIBILITY_GUI (p. 1067)
GZ_VISIBILITY_NOT_SELECTABLE (p. 1067)

10.184.3.71 void gazebo::rendering::Visual::SetVisible (bool *_visible*, bool *_cascade* = `true`)

Set whether the visual is visible.

Parameters

in	*_visible*	set this node visible.
in	*_cascade*	setting this parameter in children too.

10.184.3.72 void gazebo::rendering::Visual::SetWireframe (bool *_show*)

Enable or disable wireframe for this visual.

Parameters

in	*_show*	True to enable wireframe for this visual.

10.184.3.73 void gazebo::rendering::Visual::SetWorldPose (const math::Pose _pose)

Set the world pose of the visual.

Parameters

in	*_pose*	Pose of the visual in the world coordinate frame.

10.184.3.74 void gazebo::rendering::Visual::SetWorldPosition (const math::Vector3 & _pos)

Set the world linear position of the visual.

Parameters

in	*_pose*	Position in the world coordinate frame.

10.184.3.75 void gazebo::rendering::Visual::SetWorldRotation (const math::Quaternion & _rot)

Set the world orientation of the visual.

Parameters

in	*_rot*	Rotation in the world coordinate frame.

10.184.3.76 void gazebo::rendering::Visual::ShowBoundingBox ( )

Display the bounding box visual.

10.184.3.77 void gazebo::rendering::Visual::ShowCollision (bool _show)

Display the collision visuals.

Parameters

in	*_show*	True to show visuals labeled as collision objects.

10.184.3.78 void gazebo::rendering::Visual::ShowCOM (bool _show)

Display Center of Mass visuals.

Parameters

in	*_show*	True to show center of mass visualizations.

10.184.3.79 void gazebo::rendering::Visual::ShowJoints (bool *_show*)

Display joint visuals.

Parameters

in	*_show*	True to show joint visualizations.

10.184.3.80 void gazebo::rendering::Visual::ShowSkeleton (bool *_show*)

Display the skeleton visuals.

Parameters

in	*_show*	True to show skeleton visuals.

10.184.3.81 void gazebo::rendering::Visual::ToggleVisible ( )

Toggle whether this visual is visible.

10.184.3.82 void gazebo::rendering::Visual::Update ( )

Update the visual.

10.184.3.83 void gazebo::rendering::Visual::UpdateFromMsg (ConstVisualPtr & *_msg*)

Update a visual based on a message.

Parameters

in	*_msg*	The visual message.

10.184.4 Member Data Documentation

10.184.4.1 VisualPtr gazebo::rendering::Visual::parent [protected]

Parent visual.

10.184.4.2 ScenePtr gazebo::rendering::Visual::scene [protected]

Pointer to the visual's scene.

10.184.4.3 Ogre::SceneNode* gazebo::rendering::Visual::sceneNode [protected]

Pointer to the visual's scene node in **Ogre** (p. 106).

The documentation for this class was generated from the following file:

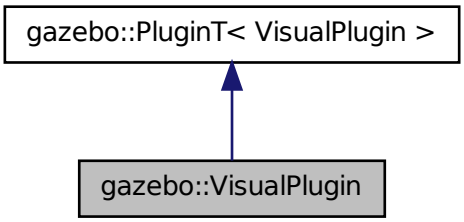
- **Visual.hh**

10.185 gazebo::VisualPlugin Class Reference

A (p. 111) 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

10.185.1 Detailed Description

A (p. 111) plugin loaded within the gzserver on startup.

See `reference`.

10.185.2 Constructor & Destructor Documentation

10.185.2.1 gazebo::VisualPlugin::VisualPlugin ( ) [inline]

References gazebo::PluginT< VisualPlugin >::type, and gazebo::VISUAL_PLUGIN.

10.185.3 Member Function Documentation

10.185.3.1 virtual void gazebo::VisualPlugin::Init ( ) [inline], [virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.185.3.2 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	*_visual*	Pointer the Visual Object.
in	*_sdf*	Pointer the the SDF element of the plugin.

10.185.3.3 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:

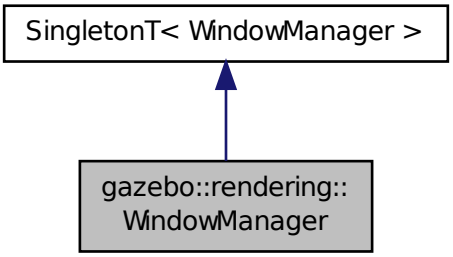
- **common/Plugin.hh**

10.186 gazebo::rendering::WindowManager Class Reference

Class to mangage render windows.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::WindowManager:



Public Member Functions

- int **CreateWindow** (const std::string &_ogreHandle, uint32_t _width, uint32_t _height)

Create a window.

- void **Fini** ()

  Shutdown all the windows.

- float **GetAvgFPS** (uint32_t _id)

  Get the average FPS.

- uint32_t **GetTriangleCount** (uint32_t _id)

  Get the triangle count.

- Ogre::RenderWindow * **GetWindow** (uint32_t _id)

  Get the render window associated with the given id.

- void **Moved** (uint32_t _id)

  *Tells **Ogre** (p. 106) the window has moved, and needs updating.*

- void **Resize** (uint32_t _id, int _width, int _height)

  Resize a window.

- void **SetCamera** (int _windowId, **CameraPtr** _camera)

  Attach a camera to a window.

Additional Inherited Members

10.186.1 Detailed Description

Class to mangage render windows.

10.186.2 Member Function Documentation

10.186.2.1 int gazebo::rendering::WindowManager::CreateWindow (const std::string & *_ogreHandle*, uint32_t *_width*, uint32_t *_height*)

Create a window.

Parameters

in	*_ogreHandle*	String representing the ogre window handle.
in	*_width*	With of the window in pixels.
in	*_height*	Height of the window in pixels.

10.186.2.2 void gazebo::rendering::WindowManager::Fini ( )

Shutdown all the windows.

10.186.2.3 float gazebo::rendering::WindowManager::GetAvgFPS (uint32_t *_id*)

Get the average FPS.

Parameters

in	*_id*	ID of the window.

Returns

The frames per second.

10.186.2.4 uint32_t gazebo::rendering::WindowManager::GetTriangleCount (uint32_t *_id*)

Get the triangle count.

Parameters

in	*_id*	ID of the window.

Returns

The triangle count.

10.186.2.5 Ogre::RenderWindow* gazebo::rendering::WindowManager::GetWindow (uint32_t *_id*)

Get the render window associated with the given id.

Parameters

in	*_id*	ID of the window.

Returns

Pointer to the render window, NULL if the id is invalid.

10.186.2.6 void gazebo::rendering::WindowManager::Moved (uint32_t *_id*)

Tells **Ogre** (p. 106) the window has moved, and needs updating.

Parameters

in	*_id*	ID of the window.

10.186.2.7 void gazebo::rendering::WindowManager::Resize (uint32_t *_id,* int *_width,* int *_height*)

Resize a window.

Parameters

in	*_id*	Id of the window to resize.
in	*_width*	New width of the window.
in	*_height*	New height of the window.

10.186.2.8 void gazebo::rendering::WindowManager::SetCamera (int _windowId, CameraPtr _camera)

Attach a camera to a window.

Parameters

in	*_windowId*	Id of the window to add the camera to.
in	*_camera*	Pointer to the camera to attach.

The documentation for this class was generated from the following file:

- **WindowManager.hh**

10.187 gazebo::rendering::WireBox Class Reference

Draws a wireframe box.

`#include <rendering/rendering.hh>`

Public Member Functions

- **WireBox** (**VisualPtr** _parent, const **math::Box** &_box)
  Constructor.
- ~**WireBox** ()
  Destructor.
- void **Init** (const **math::Box** &_box)
  Builds the wireframe line list.
- void **SetVisible** (bool _visible)
  Set the visibility of the box.

10.187.1 Detailed Description

Draws a wireframe box.

10.187.2 Constructor & Destructor Documentation

10.187.2.1 gazebo::rendering::WireBox::WireBox (VisualPtr _parent, const math::Box & _box) [explicit]

Constructor.

Parameters

in	*_box*	Dimenision of the box to draw.

10.187.2.2 gazebo::rendering::WireBox::~WireBox ( )

Destructor.

10.187.3 Member Function Documentation

10.187.3.1 void gazebo::rendering::WireBox::Init (const math::Box & _box)

Builds the wireframe line list.

Parameters

in	*_box*	Box to build a wireframe from.

10.187.3.2 void gazebo::rendering::WireBox::SetVisible (bool _visible)

Set the visibility of the box.

Parameters

in	*_visible*	True to make the box visible, False to hide.

The documentation for this class was generated from the following file:

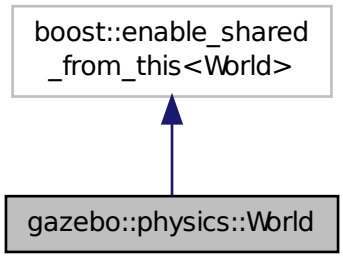
- **WireBox.hh**

10.188 gazebo::physics::World Class Reference

The world provides access to all other object within a simulated environment.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::World:



Public Member Functions

- **World** (const std::string &_name="")

  Constructor.
- ~**World** ()

Destructor.

- void **Clear** ()

  Remove all entities from the world.
- void **DisableAllModels** ()

  Disable all links in all the models.
- void **EnableAllModels** ()

  Enable all links in all the models.
- void **EnablePhysicsEngine** (bool _enable)

  enable/disable physics engine during World::Update.
- void **Fini** ()

  Finalize the world.
- **BasePtr GetByName** (const std::string &_name)

  Get an element by name.
- bool **GetEnablePhysicsEngine** ()

  check if physics engine is enabled/disabled.
- **EntityPtr GetEntity** (const std::string &_name)

  *Get a pointer to an **Entity** (p. 281) based on a name.*
- **EntityPtr GetEntityBelowPoint** (const **math::Vector3** &_pt)

  Get the nearest entity below a point.
- **ModelPtr GetModel** (unsigned int _index) const

  Get a model based on an index.
- **ModelPtr GetModel** (const std::string &_name)

  Get a model by name.
- **ModelPtr GetModelBelowPoint** (const **math::Vector3** &_pt)

  Get the nearest model below a point.
- unsigned int **GetModelCount** () const

  Get the number of models.
- **Model_V GetModels** () const

  Get a list of all the models.
- std::string **GetName** () const

  Get the name of the world.
- **common::Time GetPauseTime** () const

  Get the amount of time simulation has been paused.
- **PhysicsEnginePtr GetPhysicsEngine** () const

  Return the physics engine.
- **common::Time GetRealTime** () const

  Get the real time (elapsed time).
- **EntityPtr GetSelectedEntity** () const

  *Get the selected **Entity** (p. 281).*
- boost::mutex * **GetSetWorldPoseMutex** () const

  Get the set world pose mutex.
- **common::Time GetSimTime** () const

  *Get the world simulation time, note if you want the PC wall clock call **common::Time::GetWallTime** (p. 796).*
- **common::Time GetStartTime** () const

  Get the wall time simulation was started.
- void **Init** ()

  Initialize the world.

- void **InsertModelFile** (const std::string &_sdfFilename)

  Insert a model from an SDF file.
- void **InsertModelSDF** (const **sdf::SDF** &_sdf)

  Insert a model using SDF.
- void **InsertModelString** (const std::string &_sdfString)

  Insert a model from an SDF string.
- bool **IsLoaded** () const

  Return true if the world has been loaded.
- bool **IsPaused** () const

  Returns the state of the simulation true if paused.
- void **Load** (**sdf::ElementPtr** _sdf)

  Load the world using SDF parameters.
- void **LoadPlugin** (const std::string &_filename, const std::string &_name, **sdf::ElementPtr** _sdf)

  Load a plugin.
- void **PrintEntityTree** ()

  *Print **Entity** (*p. *281) tree.*
- void **PublishModelPose** (**physics::ModelPtr** _model)

  Publish pose updates for a model.
- void **RemovePlugin** (const std::string &_name)

  Remove a running plugin.
- void **Reset** ()

  Reset time and model poses, configurations in simulation.
- void **ResetEntities** (**Base::EntityType** _type=**Base::BASE**)

  Reset with options.
- void **ResetTime** ()

  Reset simulation time back to zero.
- void **Run** ()

  Run the world in a thread.
- void **Save** (const std::string &_filename)

  Save a world to a file.
- void **SetPaused** (bool _p)

  Set whether the simulation is paused.
- void **SetSimTime** (const **common::Time** &_t)

  Set the sim time.
- void **SetState** (const **WorldState** &_state)

  Set the current world state.
- void **StepWorld** (int _steps)

  Step callback.
- void **Stop** ()

  Stop the world.
- std::string **StripWorldName** (const std::string &_name) const

  Return a version of the name with "<world_name>::" removed.
- void **UpdateStateSDF** ()

  Update the state SDF value from the current state.

Public Attributes

- std::list< **Entity** * > **dirtyPoses**

  when physics engine makes an update and changes a link pose, this flag is set to trigger ***Entity::SetWorldPose*** *(p. 290) on the* ***physics::Link*** *(p. 418) in World::Update.*

10.188.1 Detailed Description

The world provides access to all other object within a simulated environment.

The **World** (p. 910) is the container for all models and their components (links, joints, sensors, plugins, etc), and **WorldPlugin** (p. 921) instances. Many core function are also handled in the **World** (p. 910), including physics update, model updates, and message processing.

10.188.2 Constructor & Destructor Documentation

10.188.2.1 gazebo::physics::World::World (const std::string & *_name* = "") [explicit]

Constructor.

Constructor for the **World** (p. 910). Must specify a unique name.

Parameters

in	*_name*	Name of the world.

10.188.2.2 gazebo::physics::World::~World ( )

Destructor.

10.188.3 Member Function Documentation

10.188.3.1 void gazebo::physics::World::Clear ( )

Remove all entities from the world.

10.188.3.2 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.

10.188.3.3 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.

10.188.3.4 void gazebo::physics::World::EnablePhysicsEngine (bool *_enable*) [inline]

enable/disable physics engine during World::Update.

Parameters

in	*_enable*	True to enable the physics engine.

10.188.3.5 void gazebo::physics::World::Fini ( )

Finalize the world.

Call this function to tear-down the world.

10.188.3.6 BasePtr gazebo::physics::World::GetByName (const std::string & *_name*)

Get an element by name.

Searches the list of entities, and return a pointer to the model with a matching _name.

Parameters

in	*_name*	The name of the **Model** (p. 489) to find.

Returns

A (p. 111) pointer to the entity, or NULL if no entity was found.

10.188.3.7 bool gazebo::physics::World::GetEnablePhysicsEngine ( ) [inline]

check if physics engine is enabled/disabled.

Parameters

True	if the physics engine is enabled.

10.188.3.8 EntityPtr gazebo::physics::World::GetEntity (const std::string & *_name*)

Get a pointer to an **Entity** (p. 281) based on a name.

This function is the same as GetByName, but limits the search to only Entities.

Parameters

in	*_name*	The name of the **Entity** (p. 281) to find.

Returns

A (p. 111) pointer to the **Entity** (p. 281), or NULL if no **Entity** (p. 281) was found.

10.188.3.9 EntityPtr gazebo::physics::World::GetEntityBelowPoint (const math::Vector3 & _pt)

Get the nearest entity below a point.

Projects a Ray down (-Z axis) starting at the given point. The first entity hit by the Ray is returned.

Parameters

in	*_pt*	The 3D point to search below

Returns

A (p. 111) pointer to nearest **Entity** (p. 281), NULL if none is found.

10.188.3.10 ModelPtr gazebo::physics::World::GetModel (unsigned int _index) const

Get a model based on an index.

Get a **Model** (p. 489) using an index, where index must be greater than zero and less than **World::GetModelCount()** (p. 916)

Parameters

in	*_index*	The index of the model [0..GetModelCount)

Returns

A (p. 111) pointer to the **Model** (p. 489). NULL if _index is invalid.

10.188.3.11 ModelPtr gazebo::physics::World::GetModel (const std::string & _name)

Get a model by name.

This function is the same as GetByName, but limits the search to only models.

Parameters

in	*_name*	The name of the **Model** (p. 489) to find.

Returns

A (p. 111) pointer to the **Model** (p. 489), or NULL if no model was found.

10.188.3.12 ModelPtr gazebo::physics::World::GetModelBelowPoint (const math::Vector3 & _pt)

Get the nearest model below a point.

This function makes use of **World::GetEntityBelowPoint** (p. 915).

Parameters

in	*_pt*	The 3D point to search below.

Returns

A (p. 111) pointer to nearest **Model** (p. 489), NULL if none is found.

10.188.3.13 unsigned int gazebo::physics::World::GetModelCount ( ) const

Get the number of models.

Returns

The number of models in the **World** (p. 910).

10.188.3.14 Model_V gazebo::physics::World::GetModels ( ) const

Get a list of all the models.

Returns

A (p. 111) list of all the Models in the world.

10.188.3.15 std::string gazebo::physics::World::GetName ( ) const

Get the name of the world.

Returns

The name of the world.

10.188.3.16 common::Time gazebo::physics::World::GetPauseTime ( ) const

Get the amount of time simulation has been paused.

Returns

The pause time.

10.188.3.17 PhysicsEnginePtr gazebo::physics::World::GetPhysicsEngine ( ) const

Return the physics engine.

Get a pointer to the physics engine used by the world.

Returns

Pointer to the physics engine.

10.188.3.18 common::Time gazebo::physics::World::GetRealTime ( ) const

Get the real time (elapsed time).

Returns

The real time.

10.188.3.19 EntityPtr gazebo::physics::World::GetSelectedEntity ( ) const

Get the selected **Entity** (p. 281).

The selected entity is set via the GUI.

Returns

A (p. 111) point to the **Entity** (p. 281), NULL if nothing is selected.

10.188.3.20 boost::mutex* gazebo::physics::World::GetSetWorldPoseMutex ( ) const [inline]

Get the set world pose mutex.

Returns

Pointer to the mutex.

10.188.3.21 common::Time gazebo::physics::World::GetSimTime ( ) const

Get the world simulation time, note if you want the PC wall clock call **common::Time::GetWallTime** (p. 796).

Returns

The current simulation time

10.188.3.22 common::Time gazebo::physics::World::GetStartTime ( ) const

Get the wall time simulation was started.

Returns

The start time.

10.188.3.23 void gazebo::physics::World::Init ( )

Initialize the world.

This is called after Load.

10.188.3.24 void gazebo::physics::World::InsertModelFile (const std::string & _sdfFilename)

Insert a model from an SDF file.

Spawns a model into the world base on and SDF file.

Parameters

in	*_sdfFilename*	The name of the SDF file (including path).

10.188.3.25 void gazebo::physics::World::InsertModelSDF (const sdf::SDF & _sdf)

Insert a model using SDF.

Spawns a model into the world base on and SDF object.

Parameters

in	*_sdf*	**A** (p. 111) reference to an SDF object.

10.188.3.26 void gazebo::physics::World::InsertModelString (const std::string & _sdfString)

Insert a model from an SDF string.

Spawns a model into the world base on and SDF string.

Parameters

in	*_sdfString*	**A** (p. 111) string containing valid SDF markup.

10.188.3.27 bool gazebo::physics::World::IsLoaded ( ) const

Return true if the world has been loaded.

Returns

True if **World::Load** (p. 918) has completed.

10.188.3.28 bool gazebo::physics::World::IsPaused ( ) const

Returns the state of the simulation true if paused.

Returns

True if paused.

10.188.3.29 void gazebo::physics::World::Load (sdf::ElementPtr _sdf)

Load the world using SDF parameters.

Load a world from and SDF pointer.

Parameters

in	*_sdf*	SDF parameters.

10.188.3.30 void gazebo::physics::World::LoadPlugin (const std::string & *_filename,* const std::string & *_name,* sdf::ElementPtr *_sdf*)

Load a plugin.

Parameters

in	*_filename*	The filename of the plugin.
in	*_name*	**A** (p. 111) unique name for the plugin.
in	*_sdf*	The SDF to pass into the plugin.

10.188.3.31 void gazebo::physics::World::PrintEntityTree ( )

Print **Entity** (p. 281) tree.

Prints alls the entities to stdout.

10.188.3.32 void gazebo::physics::World::PublishModelPose (physics::ModelPtr *_model*)

Publish pose updates for a model.

This list of models to publish is processed and cleared once every iteration.

Parameters

in	*_model*	Pointer to the model to publish.

10.188.3.33 void gazebo::physics::World::RemovePlugin (const std::string & *_name*)

Remove a running plugin.

Parameters

in	*_name*	The unique name of the plugin to remove.

10.188.3.34 void gazebo::physics::World::Reset ( )

Reset time and model poses, configurations in simulation.

10.188.3.35 void gazebo::physics::World::ResetEntities (Base::EntityType *_type* = Base::BASE)

Reset with options.

The _type parameter specifies which type of eneities to reset. See **Base::EntityType** (p. 140).

Parameters

in	*_type*	The type of reset.

10.188.3.36 void gazebo::physics::World::ResetTime ( )

Reset simulation time back to zero.

10.188.3.37 void gazebo::physics::World::Run ( )

Run the world in a thread.

Run the update loop.

10.188.3.38 void gazebo::physics::World::Save (const std::string & *_filename*)

Save a world to a file.

Save the current world and its state to a file.

Parameters

in	*_filename*	Name of the file to save into.

10.188.3.39 void gazebo::physics::World::SetPaused (bool *_p*)

Set whether the simulation is paused.

Parameters

in	*_p*	True pauses the simulation. False runs the simulation.

10.188.3.40 void gazebo::physics::World::SetSimTime (const common::Time & *_t*)

Set the sim time.

Parameters

in	*_t*	The new simulation time

10.188.3.41 void gazebo::physics::World::SetState (const WorldState & *_state*)

Set the current world state.

Parameters

_state	The state to set the **World** (p. 910) to.

10.188.3.42 void gazebo::physics::World::StepWorld (int _steps)

Step callback.

Parameters

in	*_steps*	The number of steps the **World** (p. 910) should take.

10.188.3.43 void gazebo::physics::World::Stop ( )

Stop the world.

Stop the update loop.

10.188.3.44 std::string gazebo::physics::World::StripWorldName (const std::string & _name) const

Return a version of the name with "<world_name>::" removed.

Parameters

in	*_name*	Usually the name of an entity.

Returns

The stripped world name.

10.188.3.45 void gazebo::physics::World::UpdateStateSDF ( )

Update the state SDF value from the current state.

10.188.4 Member Data Documentation

10.188.4.1 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. 290) on the **physics::Link** (p. 418) in World::Update.

The documentation for this class was generated from the following file:

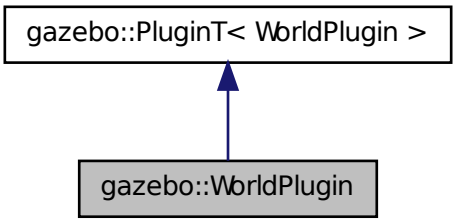
- **World.hh**

10.189 gazebo::WorldPlugin Class Reference

A (p. 111) plugin with access to **physics::World** (p. 910).

`#include <common/common.hh>`

Inheritance diagram for gazebo::WorldPlugin:



Public Member Functions

- **WorldPlugin** ()

  Constructor.
- virtual ~**WorldPlugin** ()

  Destructor.
- virtual void **Init** ()
- virtual void **Load** (**physics::WorldPtr** _world, **sdf::ElementPtr** _sdf)=0

  Load function.
- virtual void **Reset** ()

Additional Inherited Members

10.189.1 Detailed Description

A (p. 111) plugin with access to **physics::World** (p. 910).

See `reference`.

10.189.2 Constructor & Destructor Documentation

10.189.2.1 gazebo::WorldPlugin::WorldPlugin ( ) [inline]

Constructor.

References gazebo::PluginT< WorldPlugin >::type, and gazebo::WORLD_PLUGIN.

10.189.2.2 virtual gazebo::WorldPlugin::~WorldPlugin ( ) [inline], [virtual]

Destructor.

10.189.3 Member Function Documentation

10.189.3.1 virtual void gazebo::WorldPlugin::Init ( ) [inline], [virtual]

10.189.3.2 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	*_world*	Pointer the World
in	*_sdf*	Pointer the the SDF element of the plugin.

10.189.3.3 virtual void gazebo::WorldPlugin::Reset ( ) [inline], [virtual]

The documentation for this class was generated from the following file:

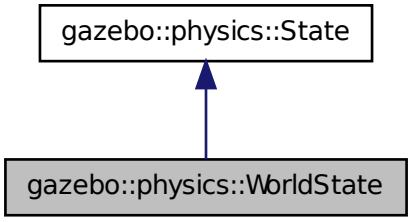
- **common/Plugin.hh**

10.190 gazebo::physics::WorldState Class Reference

Store state information of a **physics::World** (p. 910) object.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::WorldState:



Public Member Functions

- **WorldState** ()

  Default constructor.
- **WorldState** (const **WorldPtr** _world)

  Constructor.

- **WorldState** (const **sdf::ElementPtr** _sdf)

  Constructor.
- virtual ∼**WorldState** ()

  Destructor.
- void **FillSDF** (**sdf::ElementPtr** _sdf)

  Populate a state SDF element with data from the object.
- **ModelState GetModelState** (unsigned int _index) const

  Get a model state.
- **ModelState GetModelState** (const std::string &_modelName) const

  Get a model state by model name.
- unsigned int **GetModelStateCount** () const

  Get the number of model states.
- const std::vector< **ModelState** > & **GetModelStates** () const

  Get the model states.
- bool **HasModelState** (const std::string &_modelName) const

  *Return true if **WorldState** (p. 923) has a **ModelState** (p. 505) with the given name.*
- bool **IsZero** () const

  Return true if the values in the state are zero.
- virtual void **Load** (const **sdf::ElementPtr** _elem)

  Load state from SDF element.
- **WorldState operator+** (const **WorldState** &_state) const

  Addition operator.
- **WorldState operator-** (const **WorldState** &_state) const

  Subtraction operator.
- **WorldState** & **operator=** (const **WorldState** &_state)

  Assignment operator.
- void **SetWorld** (const **WorldPtr** _world)

  Set the world.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::WorldState** &_state)

  Stream insertion operator.

Additional Inherited Members

10.190.1 Detailed Description

Store state information of a **physics::World** (p. 910) object.

Instances of this class contain the state of a **World** (p. 910) at a specific time. **World** (p. 910) state includes the state of all models, and their children.

10.190.2 Constructor & Destructor Documentation

10.190.2.1 gazebo::physics::WorldState::WorldState ( )

Default constructor.

10.190.2.2 gazebo::physics::WorldState::WorldState (const WorldPtr _world) [explicit]

Constructor.

Generate a **WorldState** (p. 923) from an instance of a **World** (p. 910).

Parameters

in	*_world*	Pointer to a world

10.190.2.3 gazebo::physics::WorldState::WorldState (const sdf::ElementPtr _sdf) [explicit]

Constructor.

Build a **WorldState** (p. 923) from SDF data

Parameters

in	*_sdf*	SDF data to load a world state from.

10.190.2.4 virtual gazebo::physics::WorldState::~WorldState ( ) [virtual]

Destructor.

10.190.3 Member Function Documentation

10.190.3.1 void gazebo::physics::WorldState::FillSDF (sdf::ElementPtr _sdf)

Populate a state SDF element with data from the object.

Parameters

out	*_sdf*	SDF element to populate.

10.190.3.2 ModelState gazebo::physics::WorldState::GetModelState (unsigned int _index) const

Get a model state.

Get the state of a **Model** (p. 489) based on an index. The min index is and the max is **WorldState::GetModelStateCount()** (p. 926).

Parameters

in	*_index*	Index of the model.

Returns

State (p. 758) of the requested **Model** (p. 489).

10.190.3.3 ModelState gazebo::physics::WorldState::GetModelState (const std::string & _modelName) const

Get a model state by model name.

Parameters

in	*_modelName*	Name of the model state to get.

Returns

The model state.

Exceptions

common::Exception *(p. 318)*	When the _modelName doesn't exist.

10.190.3.4 unsigned int gazebo::physics::WorldState::GetModelStateCount ( ) const

Get the number of model states.

Returns the number of models in this instance.

Returns

Number of models.

10.190.3.5 const std::vector<ModelState>& gazebo::physics::WorldState::GetModelStates ( ) const

Get the model states.

Returns

A (p. 111) vector of model states.

10.190.3.6 bool gazebo::physics::WorldState::HasModelState (const std::string & _modelName) const

Return true if **WorldState** (p. 923) has a **ModelState** (p. 505) with the given name.

Parameters

in	*_modelName*	Name of the model to search for.

Returns

True if the **ModelState** (p. 505) exists.

10.190.3.7 bool gazebo::physics::WorldState::IsZero ( ) const

Return true if the values in the state are zero.

This will check to see if the all model states are zero.

Returns

True if the values in the state are zero.

10.190.3.8 virtual void gazebo::physics::WorldState::Load (const sdf::ElementPtr _elem) [virtual]

Load state from SDF element.

Set a **WorldState** (p. 923) from an SDF element containing **WorldState** (p. 923) info.

Parameters

in	*_elem*	Pointer to the **WorldState** (p. 923) SDF element.

Reimplemented from **gazebo::physics::State** (p. 761).

10.190.3.9 WorldState gazebo::physics::WorldState::operator+ (const WorldState & _state) const

Addition operator.

Parameters

in	*_pt*	**A** (p. 111) state to add.

Returns

The resulting state.

10.190.3.10 WorldState gazebo::physics::WorldState::operator- (const WorldState & _state) const

Subtraction operator.

Parameters

in	*_pt*	**A** (p. 111) state to substract.

Returns

The resulting state.

10.190.3.11 WorldState& gazebo::physics::WorldState::operator= (const WorldState & _state)

Assignment operator.

Parameters

in	*_state*	**State** (p. 758) value

Returns

Reference to this

10.190.3.12 void gazebo::physics::WorldState::SetWorld (const WorldPtr _world)

Set the world.

Parameters

in	*_world*	Pointer to the world.

10.190.4 Friends And Related Function Documentation

10.190.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::physics::WorldState & _state) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_state*	**World** (p. 910) state to output

Returns

the stream

The documentation for this class was generated from the following file:

- **WorldState.hh**

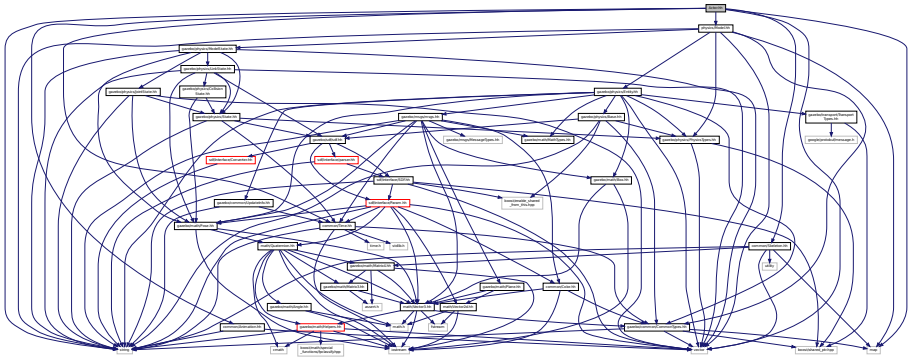
Chapter 11

File Documentation

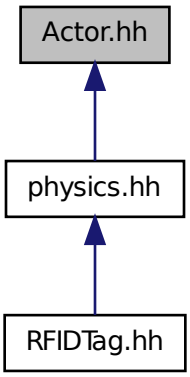
11.1 Actor.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "physics/Model.hh"
#include "common/Time.hh"
#include "common/Skeleton.hh"
#include "common/Animation.hh"
```

Include dependency graph for Actor.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Actor**

  Actor *(*p. *111) 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.

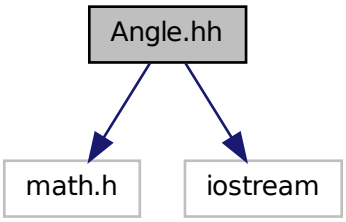
- namespace **gazebo::physics**

  namespace for physics

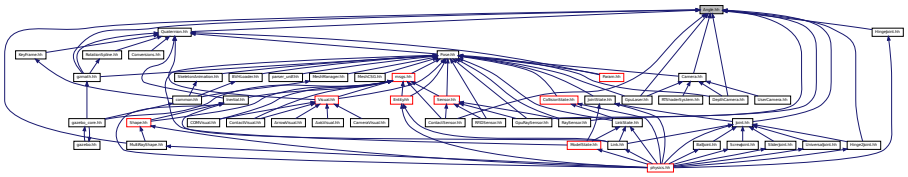
11.2 Angle.hh File Reference

```
#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	*degrees*	

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	*angle*	

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	*radians*	

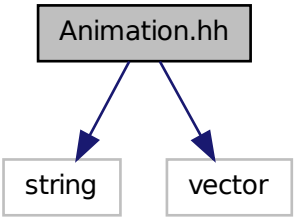
Returns

degrees

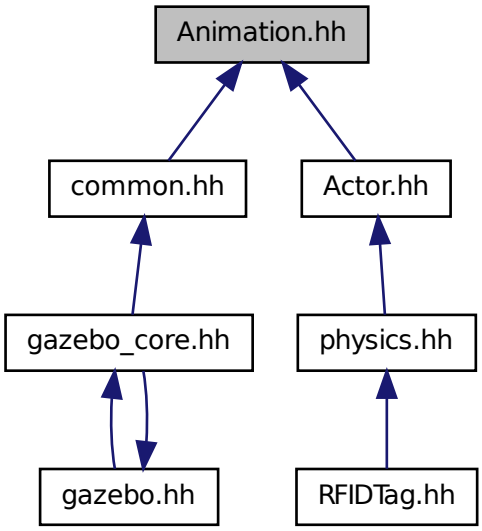
11.3 Animation.hh File Reference

```
#include <string>
#include <vector>
```

Include dependency graph for Animation.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Animation**

  Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.
- class **gazebo::common::NumericAnimation**

  A (p. *111) numeric animation.*
- class **gazebo::common::PoseAnimation**

  A (p. *111) pose animation.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

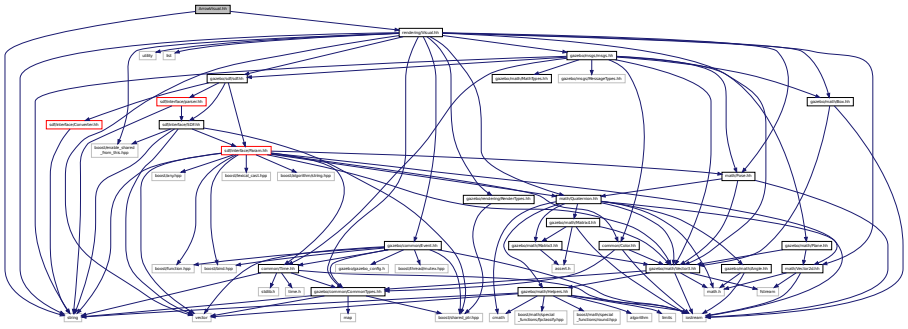
  Common namespace.
- namespace **gazebo::math**

  Math namespace.

11.4 ArrowVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
```

Include dependency graph for ArrowVisual.hh:



Classes

- class **gazebo::rendering::ArrowVisual**

  Basic arrow visualization.

Namespaces

- namespace **gazebo**

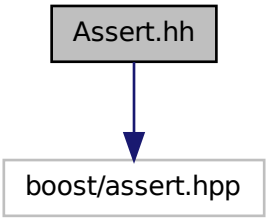
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **ogre**

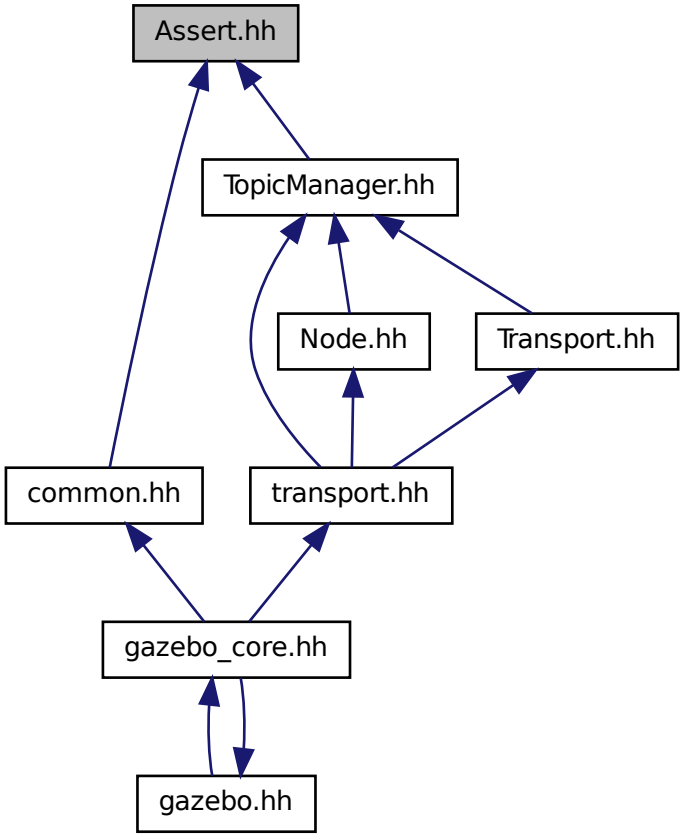
11.5 Assert.hh File Reference

```
#include <boost/assert.hpp>
```

Include dependency graph for Assert.hh:



This graph shows which files directly or indirectly include this file:



Macros

- #define **GZ_ASSERT**(_expr, _msg) BOOST_ASSERT_MSG(_expr, _msg)

  This macro define the standard way of launching an exception inside gazebo.

11.5.1 Macro Definition Documentation

11.5.1.1 #define GZ_ASSERT(*_expr, _msg*) BOOST_ASSERT_MSG(_expr, _msg)

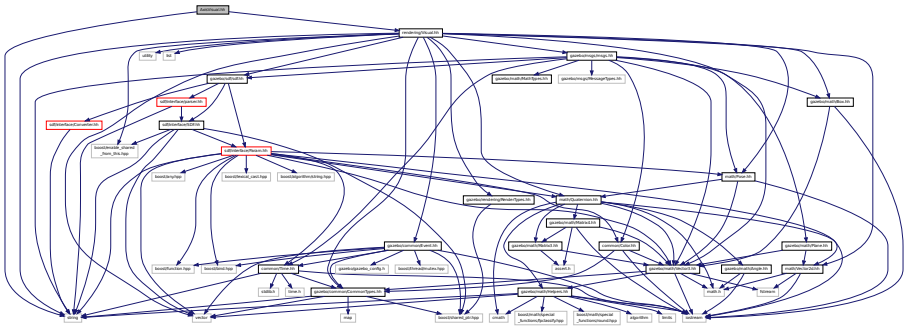
This macro define the standard way of launching an exception inside gazebo.

Referenced by gazebo::transport::TopicManager::Advertise().

11.6 AxisVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
```

Include dependency graph for AxisVisual.hh:



Classes

- class **gazebo::rendering::AxisVisual**

  Basic axis visualization.

Namespaces

- namespace **gazebo**

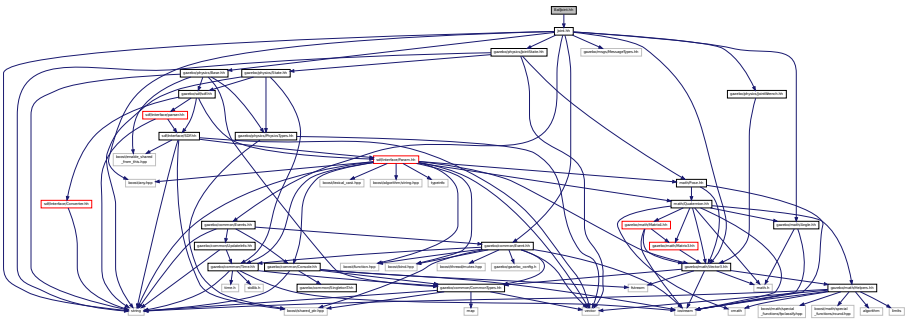
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

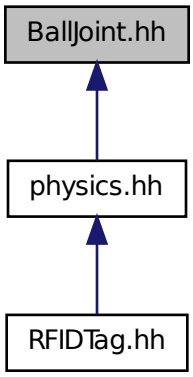
11.7 BallJoint.hh File Reference

```
#include "Joint.hh"
```

Include dependency graph for BallJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::BallJoint**< **T** >

  Base (p. *137*) *class for a ball joint.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

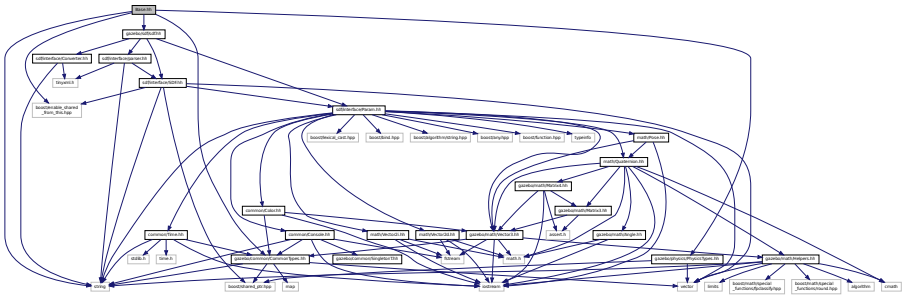
- namespace **gazebo::physics**

  namespace for physics

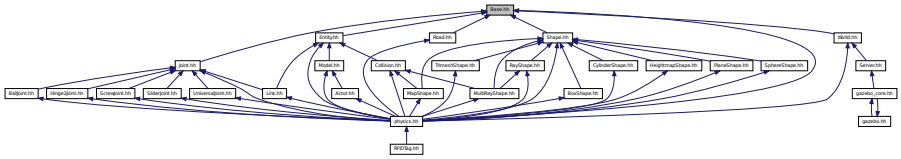
11.8 Base.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include "gazebo/sdf/sdf.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for Base.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Base**

  Base *(*p. *137) class for most physics classes.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Variables

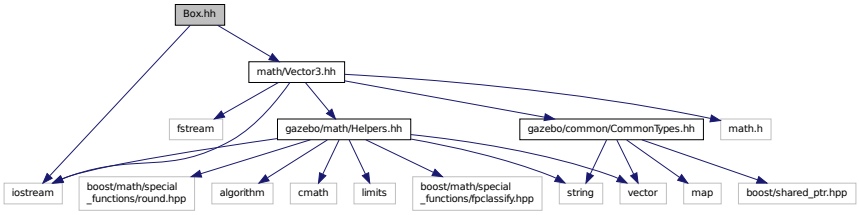
- static std::string **gazebo::physics::EntityTypename** [ ]

  String names for the different entity types.

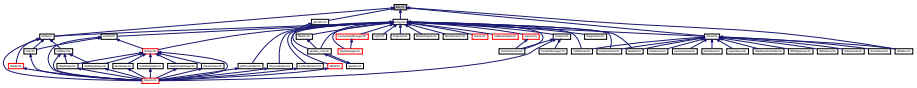
11.9 Box.hh File Reference

```
#include <iostream>
#include "math/Vector3.hh"
```

Include dependency graph for Box.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Box**

  Mathematical representation of a box and related functions.

Namespaces

- namespace **gazebo**

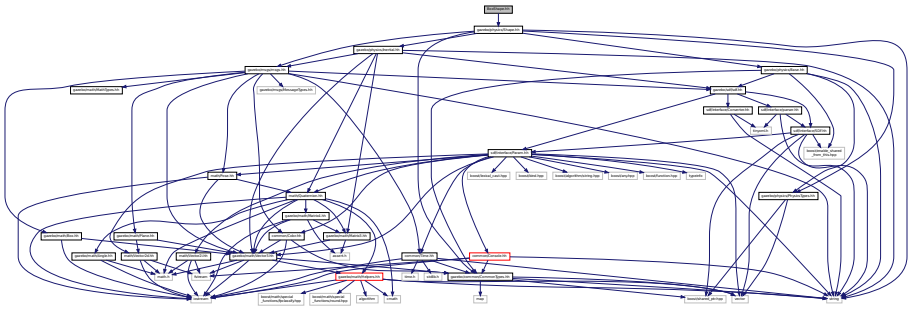
  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

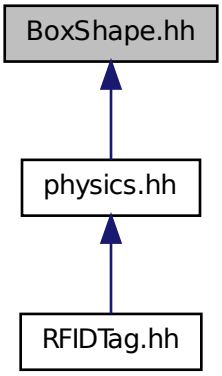
11.10 BoxShape.hh File Reference

```
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for BoxShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::BoxShape**

  Box geometry primitive.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

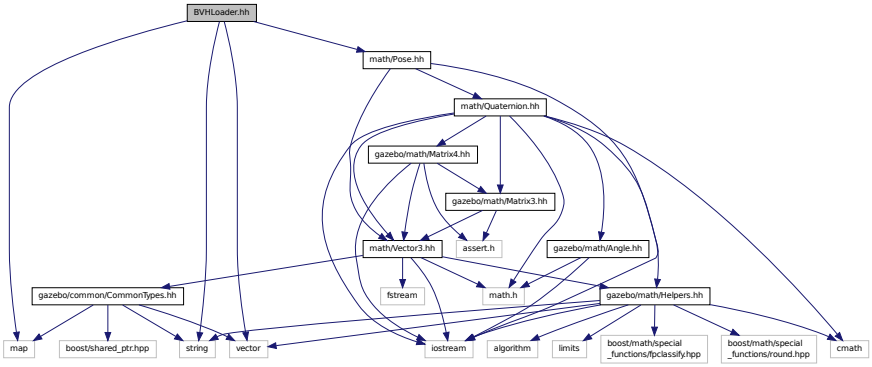
- namespace **gazebo::physics**

  namespace for physics

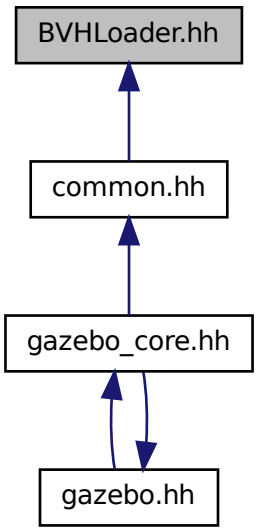
11.11 BVHLoader.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include "math/Pose.hh"
```

Include dependency graph for BVHLoader.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::BVHLoader**

  Handles loading BVH animation files.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

Macros

- #define **X_POSITION** 0
- #define **X_ROTATION** 3
- #define **Y_POSITION** 1
- #define **Y_ROTATION** 4
- #define **Z_POSITION** 2
- #define **Z_ROTATION** 5

11.11.1 Macro Definition Documentation

11.11.1.1 #define X_POSITION 0

11.11.1.2 #define X_ROTATION 3

11.11.1.3 #define Y_POSITION 1

11.11.1.4 #define Y_ROTATION 4

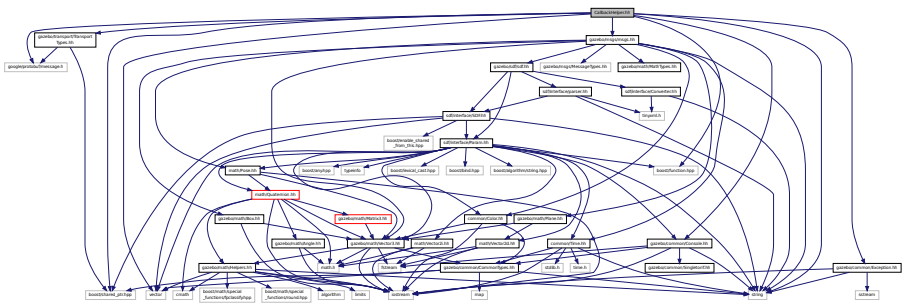
11.11.1.5 #define Z_POSITION 2

11.11.1.6 #define Z_ROTATION 5

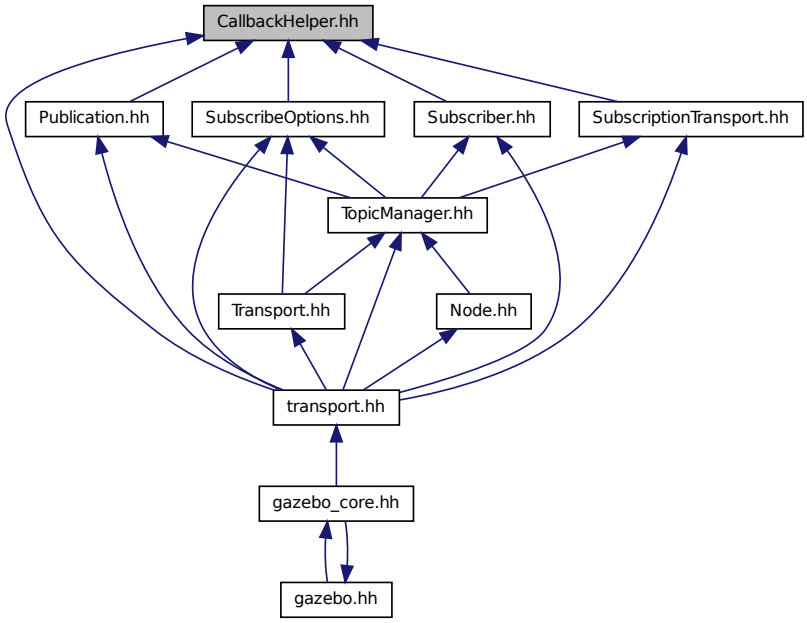
11.12 CallbackHelper.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <vector>
#include <string>
#include "gazebo/common/Console.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Exception.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for CallbackHelper.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::CallbackHelper**

  A *(p. 111) helper class to handle callbacks when messages arrive.*

- class **gazebo::transport::CallbackHelperT< M >**

  Callback helper Template.

- class **gazebo::transport::RawCallbackHelper**

  Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::transport**

Typedefs

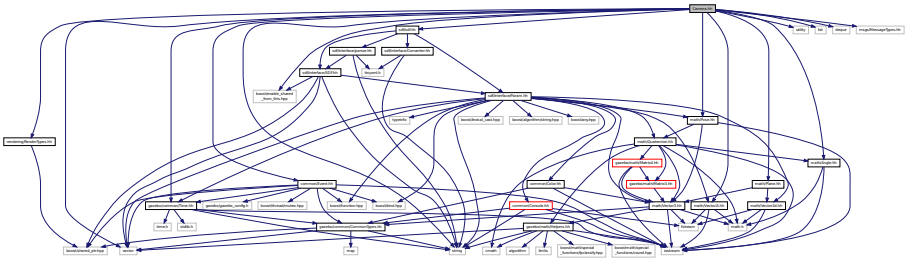
- typedef CallbackHelper * **gazebo::transport::CallbackHelperPtr**

  boost shared pointer to ***transport::CallbackHelper*** *(p. 157)*

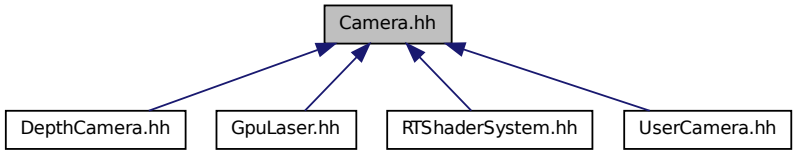
11.13 Camera.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <list>
#include <vector>
#include <deque>
#include "common/Event.hh"
#include "common/Time.hh"
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "math/Plane.hh"
#include "math/Vector2i.hh"
#include "msgs/MessageTypes.hh"
#include "rendering/RenderTypes.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for Camera.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Camera**

  Basic camera sensor.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::rendering**

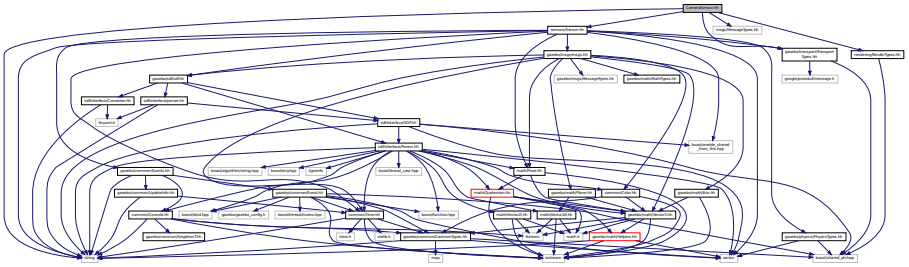
  Rendering namespace.

- namespace **Ogre**

11.14 CameraSensor.hh File Reference

```
#include <string>
#include "sensors/Sensor.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for CameraSensor.hh:



Classes

- class **gazebo::sensors::CameraSensor**

  Basic camera sensor.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

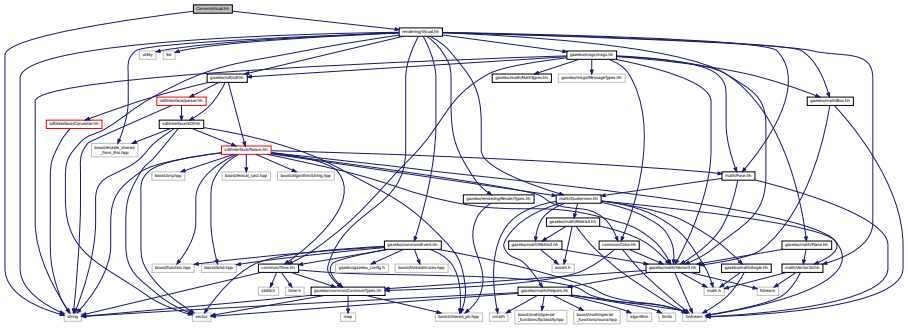
- namespace **gazebo::sensors**

  Sensors namespace.

11.15 CameraVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
```

Include dependency graph for CameraVisual.hh:



Classes

- class **gazebo::rendering::CameraVisual**

  Basic camera visualization.

Namespaces

- namespace **gazebo**

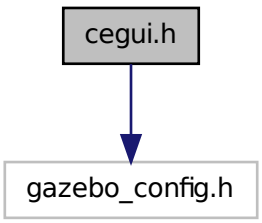
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

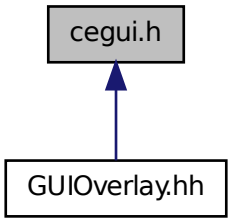
11.16 cegui.h File Reference

```
#include "gazebo_config.h"
```

Include dependency graph for cegui.h:



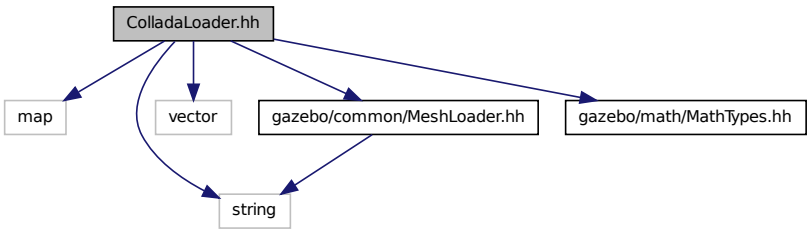
This graph shows which files directly or indirectly include this file:



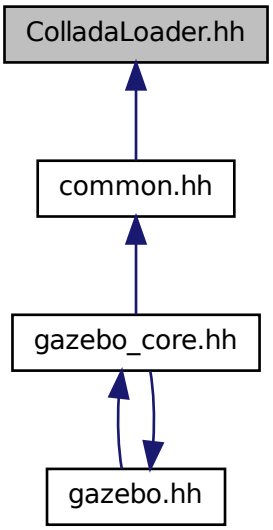
11.17 ColladaLoader.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include "gazebo/common/MeshLoader.hh"
#include "gazebo/math/MathTypes.hh"
```

Include dependency graph for ColladaLoader.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::ColladaLoader**

  Class used to load Collada mesh files.

Namespaces

- namespace **gazebo**

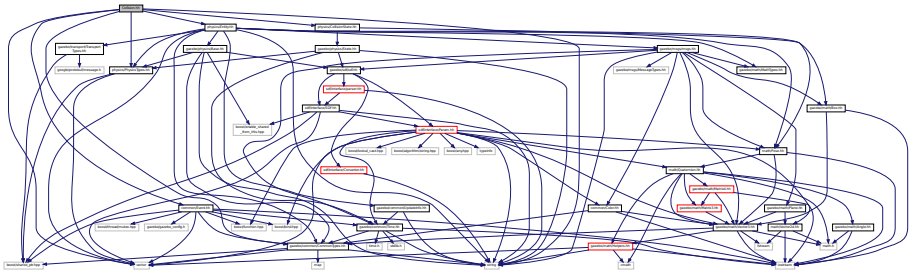
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

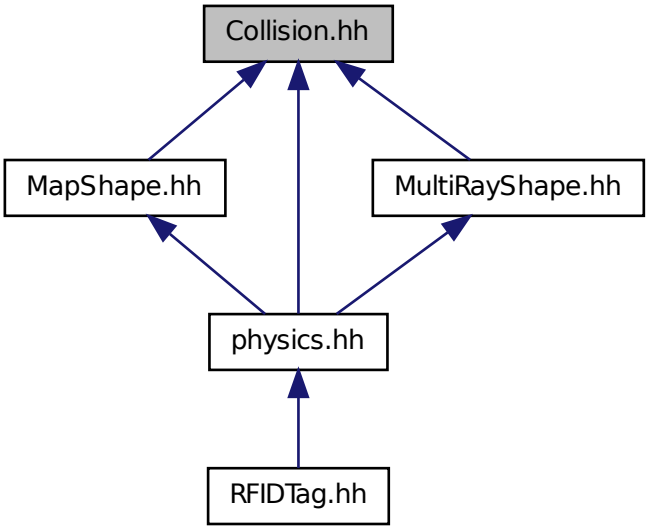
11.18 Collision.hh File Reference

```
#include <string>
#include <vector>
#include "common/Event.hh"
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/CollisionState.hh"
#include "physics/Entity.hh"
```

Include dependency graph for Collision.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Collision**

  Base *(p. 137) class for all collision entities.*

Namespaces

- namespace **gazebo**

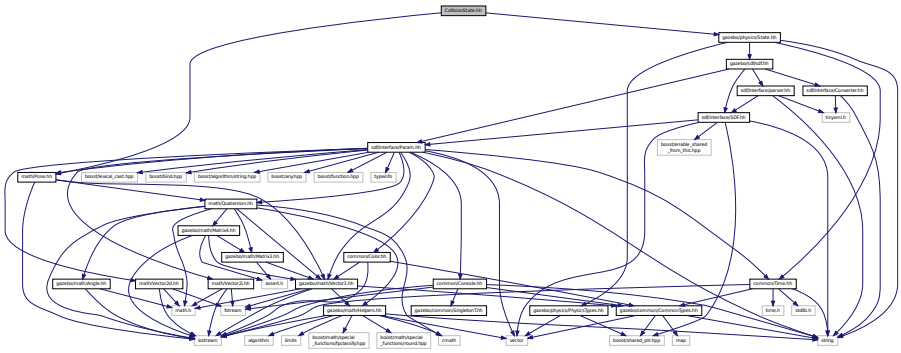
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

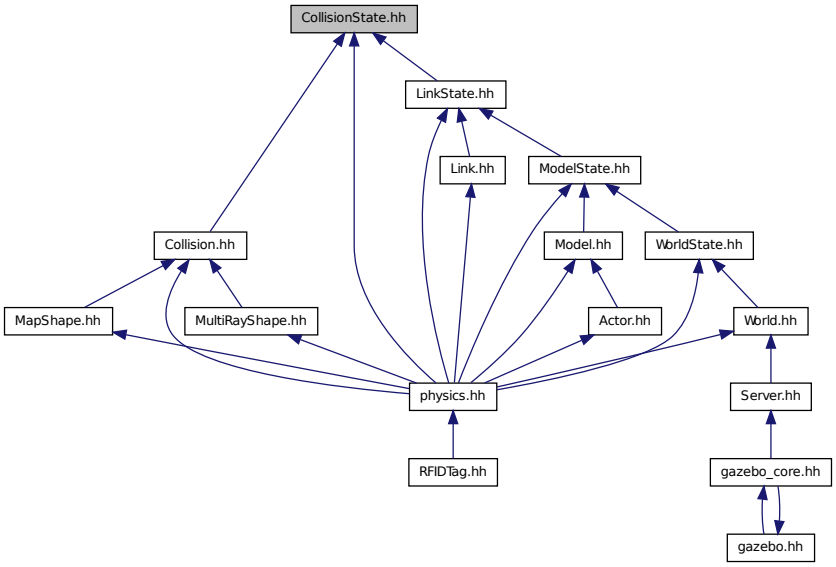
11.19 CollisionState.hh File Reference

```
#include "gazebo/physics/State.hh"
#include "gazebo/math/Pose.hh"
```

Include dependency graph for CollisionState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::CollisionState**

  *Store state information of a **physics::Collision** (p. 195) object.*

Namespaces

- namespace **gazebo**

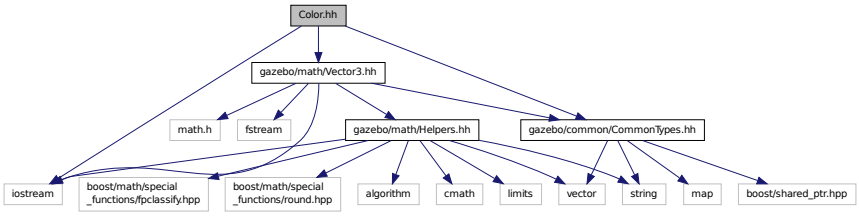
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

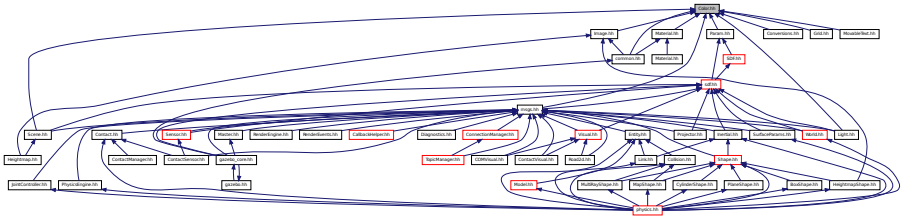
11.20 Color.hh File Reference

```
#include <iostream>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for Color.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Color**

  Defines a color.

Namespaces

- namespace **gazebo**

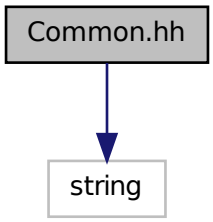
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

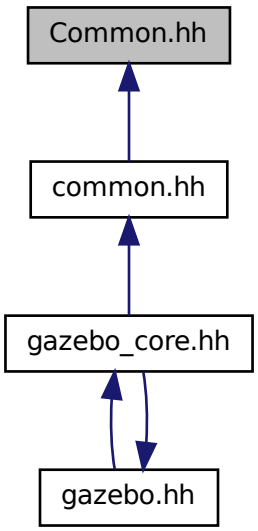
11.21 Common.hh File Reference

```
#include <string>
```

Include dependency graph for Common.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

Common namespace.

Functions

- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)

  *add path prefix to **common::SystemPaths*** (p. *784*)
- std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath=true)

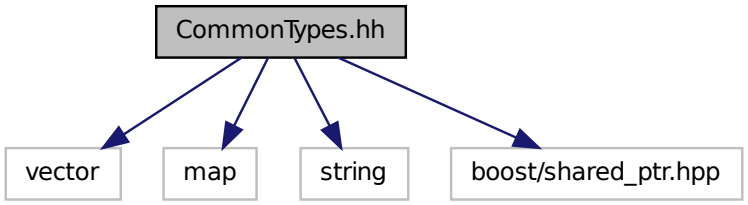
  *search for file in **common::SystemPaths*** (p. *784*)
- std::string **gazebo::common::find_file_path** (const std::string &_file)

  *search for a file in **common::SystemPaths*** (p. *784*)

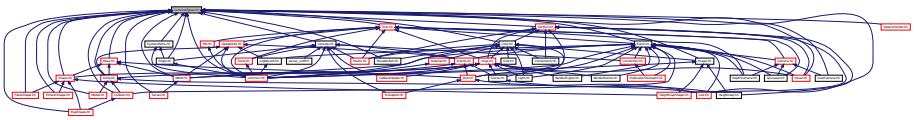
11.22 CommonTypes.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for CommonTypes.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **ParamT**< **T** >

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

- namespace **gazebo::event**

  Event *(p. 292) namespace.*

Macros

- #define **GAZEBO_DEPRECATED**(version) ()
- #define **GAZEBO_FORCEINLINE**
- #define **NULL** 0

Typedefs

- typedef Animation * **gazebo::common::AnimationPtr**
- typedef std::vector < ConnectionPtr > **gazebo::event::Connection_V**
- typedef Connection * **gazebo::event::ConnectionPtr**
- typedef DiagnosticTimer * **gazebo::common::DiagnosticTimerPtr**
- typedef GUIPlugin * **gazebo::GUIPluginPtr**
- typedef ModelPlugin * **gazebo::ModelPluginPtr**
- typedef NumericAnimation * **gazebo::common::NumericAnimationPtr**
- typedef std::vector < common::Param * > **gazebo::common::Param_V**
- typedef PoseAnimation * **gazebo::common::PoseAnimationPtr**
- typedef SensorPlugin * **gazebo::SensorPluginPtr**
- typedef std::map< std::string, std::string > **gazebo::common::StrStr_M**
- typedef SystemPlugin * **gazebo::SystemPluginPtr**
- typedef VisualPlugin * **gazebo::VisualPluginPtr**
- typedef WorldPlugin * **gazebo::WorldPluginPtr**

11.22.1 Macro Definition Documentation

11.22.1.1 #define GAZEBO_DEPRECATED(*version*) ()

11.22.1.2 #define GAZEBO_FORCEINLINE

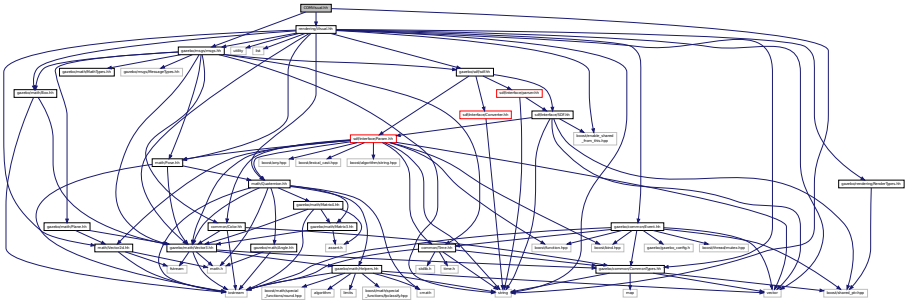
11.22.1.3 #define NULL 0

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::event::EventT< T >::Disconnect(), gazebo::transport::PublishTask::execute(), gazebo::transport::ConnectionReadTask::execute(), gazebo::transport::CallbackHelperT< M >::GetMsgType(), and gazebo::transport::SubscribeOptions::Init().

11.23 COMVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/msgs.hh"
```

Include dependency graph for COMVisual.hh:



Classes

- class **gazebo::rendering::COMVisual**

  Basic Center of Mass visualization.

Namespaces

- namespace **gazebo**

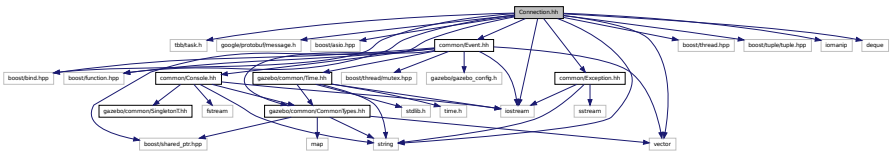
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **ogre**

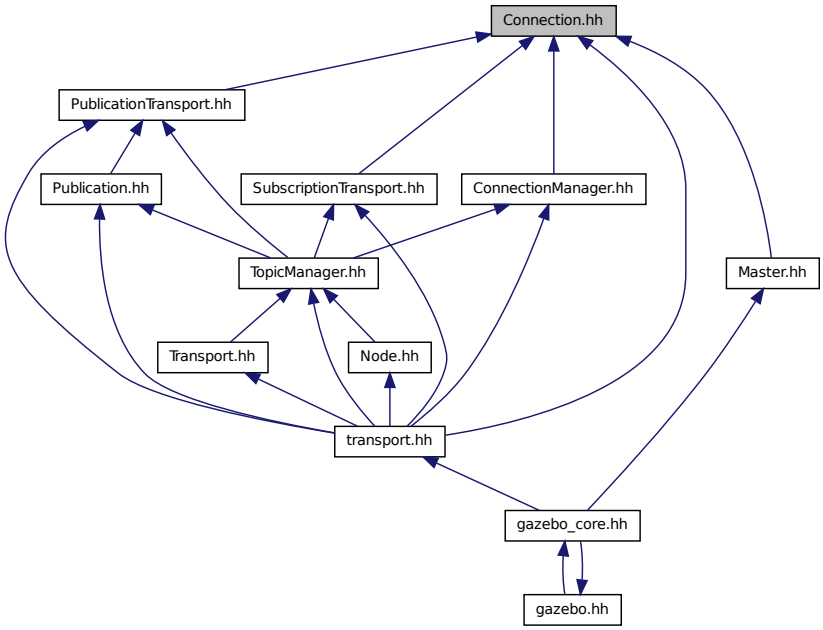
11.24 Connection.hh File Reference

```
#include <tbb/task.h>
#include <google/protobuf/message.h>
#include <boost/asio.hpp>
#include <boost/bind.hpp>
#include <boost/function.hpp>
#include <boost/thread.hpp>
#include <boost/tuple/tuple.hpp>
#include <string>
#include <vector>
#include <iostream>
#include <iomanip>
#include <deque>
#include "common/Event.hh"
#include "common/Console.hh"
#include "common/Exception.hh"
```

Include dependency graph for Connection.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Connection**

  Single TCP/IP connection manager.
- class **gazebo::transport::ConnectionReadTask**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::transport**

Macros

- #define **HEADER_LENGTH** 8

Typedefs

- typedef Connection ∗ **gazebo::transport::ConnectionPtr**

Functions

- bool **gazebo::transport::is_stopped** ()

  Is the transport system stopped?

11.24.1 Macro Definition Documentation

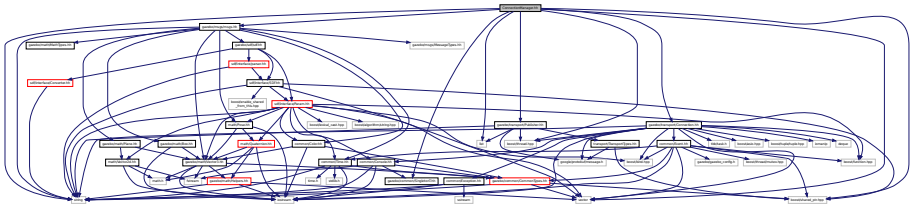
11.24.1.1 #define HEADER_LENGTH 8

Referenced by gazebo::transport::Connection::AsyncRead().

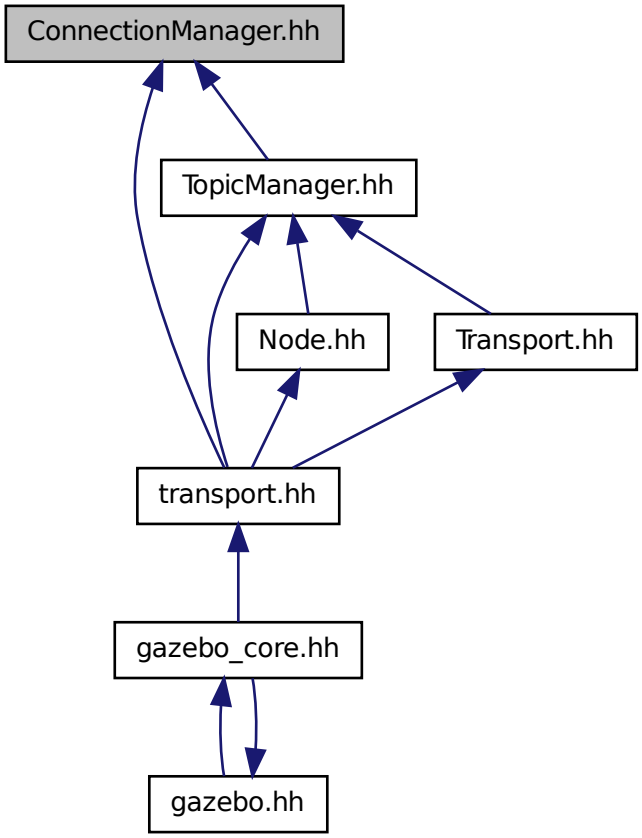
11.25 ConnectionManager.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <boost/thread.hpp>
#include <string>
#include <list>
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/transport/Publisher.hh"
#include "gazebo/transport/Connection.hh"
```

Include dependency graph for ConnectionManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::ConnectionManager**

  Manager of connections.

Namespaces

- namespace **gazebo**

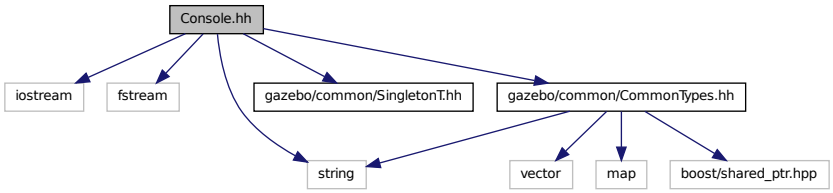
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.26 Console.hh File Reference

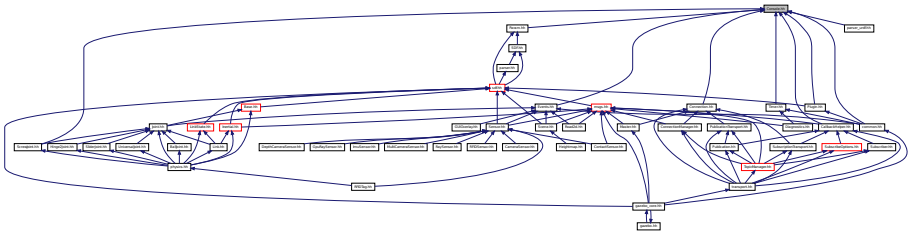
```
#include <iostream>
```

```
#include <fstream>
#include <string>
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for Console.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Console**

  Message, error, warning functionality.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Macros

- #define **gzclr_end** "\033[0m"

  End marker.
- #define **gzclr_start**(clr) "\033[1;33m"

  Start marker.
- #define **gzdbg** (**gazebo::common::Console::Instance**()->ColorMsg("Dbg", 36))

  Output a debug message.

- #define **gzerr**

  Output an error message.

- #define **gzlog** (**gazebo::common::Console::Instance**()->Log())

  Output a message to a log file.

- #define **gzmsg** (**gazebo::common::Console::Instance**()->ColorMsg("Msg", 32))

  Output a message.

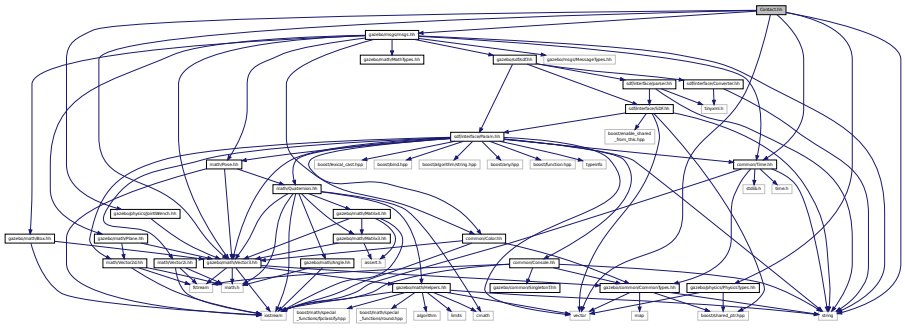
- #define **gzwarn**

  Output a warning message.

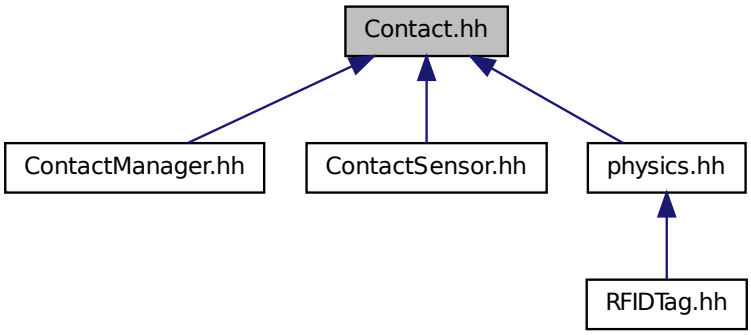
11.27 Contact.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/JointWrench.hh"
```

Include dependency graph for Contact.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Contact**

  A (p. *111) contact between two collisions.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Macros

- #define **MAX_COLLIDE_RETURNS** 250
- #define **MAX_CONTACT_JOINTS** 32

11.27.1 Macro Definition Documentation

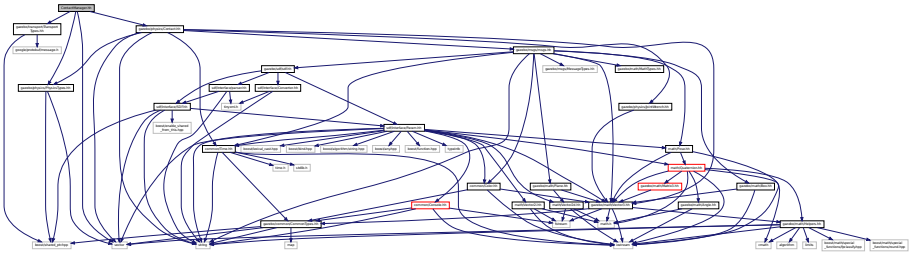
11.27.1.1 #define MAX_COLLIDE_RETURNS 250

11.27.1.2 #define MAX_CONTACT_JOINTS 32

11.28 ContactManager.hh File Reference

```
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Contact.hh"
```

Include dependency graph for ContactManager.hh:



Classes

- class **gazebo::physics::ContactManager**

  Aggregates all the contact information generated by the collision detection engine.

Namespaces

- namespace **gazebo**

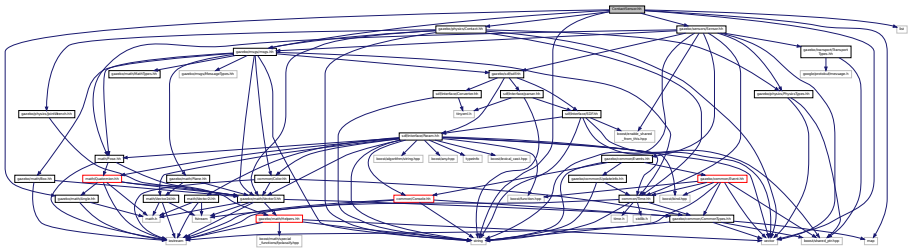
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.29 ContactSensor.hh File Reference

```
#include <vector>
#include <map>
#include <list>
#include <string>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/physics/Contact.hh"
```

Include dependency graph for ContactSensor.hh:



Classes

- class **gazebo::sensors::ContactSensor**

Contact sensor.

Namespaces

- namespace **gazebo**

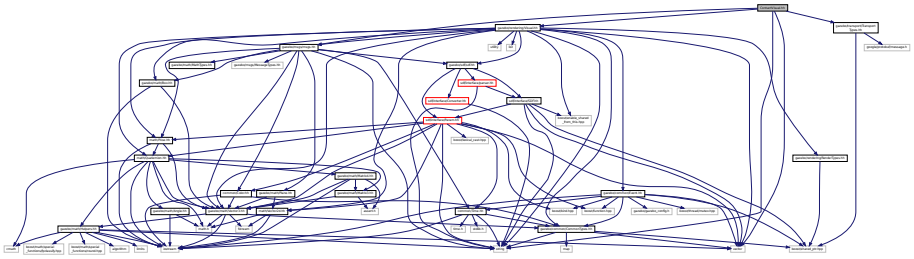
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.30 ContactVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for ContactVisual.hh:



Classes

- class **gazebo::rendering::ContactVisual**

  Contact visualization.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.31 Conversions.hh File Reference

```
#include "rendering/ogre_gazebo.h"
#include "common/Color.hh"
#include "math/Vector3.hh"
#include "math/Quaternion.hh"
```

Include dependency graph for Conversions.hh:

Classes

- class **gazebo::rendering::Conversions**

  Conversions (p. *247*) ***Conversions.hh*** (p. *964*) ***rendering/Conversions.hh*** (p. *964*).

Namespaces

- namespace **gazebo**

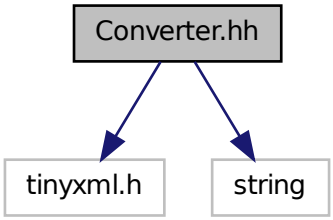
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

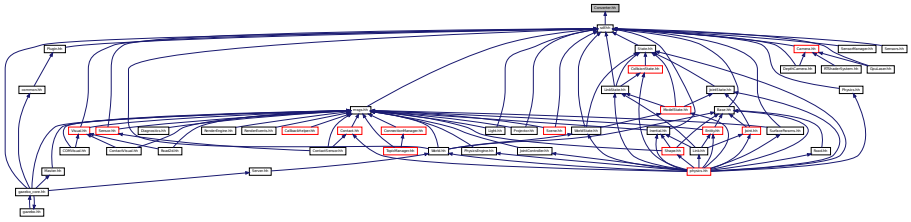
11.32 Converter.hh File Reference

```
#include <tinyxml.h>
#include <string>
```

Include dependency graph for Converter.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Converter**

  *Convert from one version of **SDF** (p. 695) to another.*

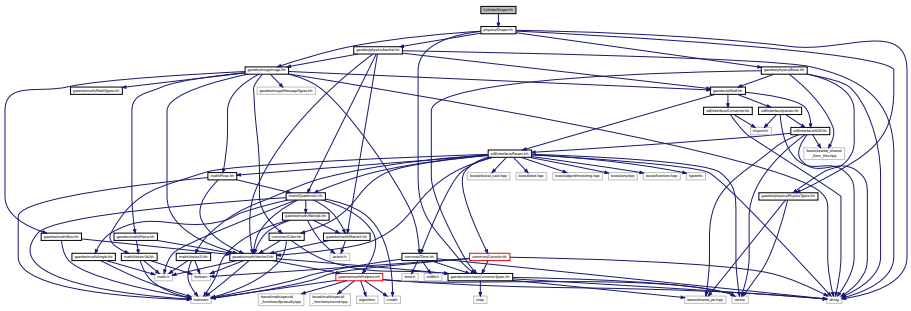
Namespaces

- namespace **sdf**

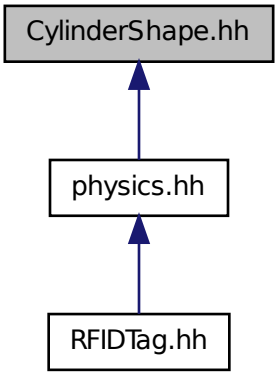
  namespace for Simulation Description Format parser

11.33 CylinderShape.hh File Reference

`#include "physics/Shape.hh"`
Include dependency graph for CylinderShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::CylinderShape**

  Cylinder collision.

Namespaces

- namespace **gazebo**

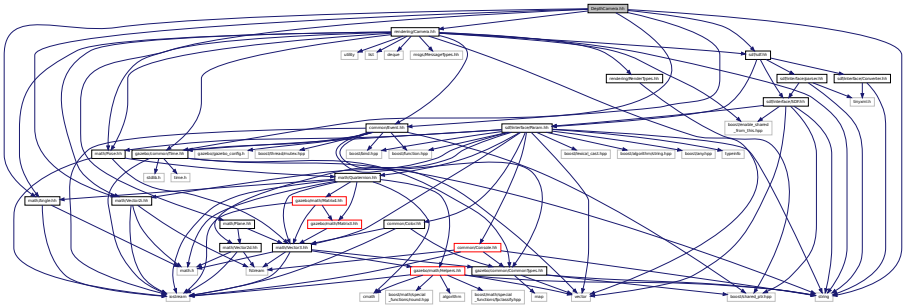
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.34 DepthCamera.hh File Reference

```
#include <string>
#include "common/Event.hh"
#include "common/Time.hh"
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "math/Vector2i.hh"
#include "sdf/sdf.hh"
#include "rendering/Camera.hh"
```

Include dependency graph for DepthCamera.hh:



Classes

- class **gazebo::rendering::DepthCamera**

  Depth camera used to render depth data into an image buffer.

Namespaces

- namespace **gazebo**

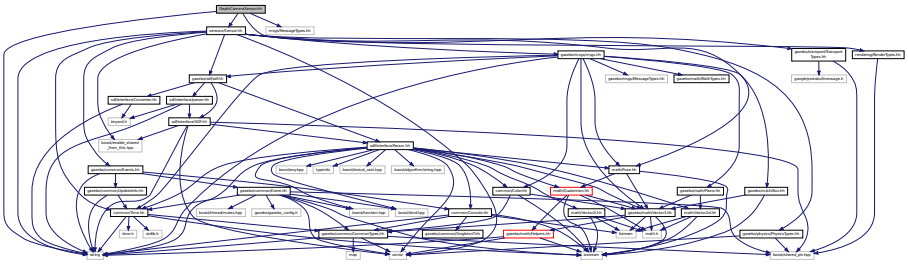
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.35 DepthCameraSensor.hh File Reference

```
#include <string>
#include "sensors/Sensor.hh"
#include "msgs/MessageTypes.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for DepthCameraSensor.hh:



Classes

- class **gazebo::sensors::DepthCameraSensor**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

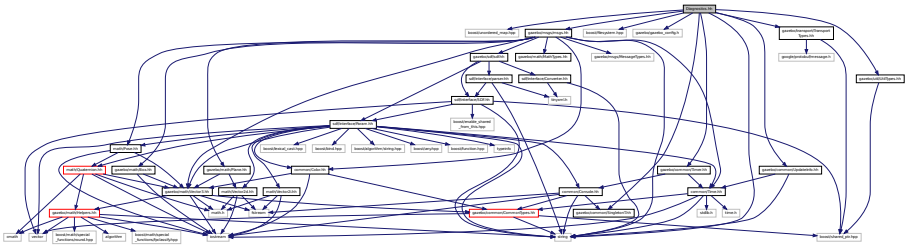
- namespace **gazebo::sensors**

  Sensors namespace.

11.36 Diagnostics.hh File Reference

```
#include <boost/unordered_map.hpp>
#include <string>
#include <boost/filesystem.hpp>
#include "gazebo/gazebo_config.h"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/Timer.hh"
#include "gazebo/util/UtilTypes.hh"
```

Include dependency graph for Diagnostics.hh:



Classes

- class **gazebo::util::DiagnosticManager**

  A (p. *111) diagnostic manager class.*

- class **gazebo::util::DiagnosticTimer**

  A (p. *111) timer designed for diagnostics.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::util**

Macros

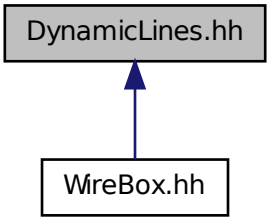
- #define **DIAG_TIMER_LAP**(_name, _prefix) ((void)0)
- #define **DIAG_TIMER_START**(_name) ((void) 0)
- #define **DIAG_TIMER_STOP**(_name) ((void) 0)

11.37 DynamicLines.hh File Reference

```
#include <vector>
#include <string>
#include "math/Vector3.hh"
#include "rendering/DynamicRenderable.hh"
```

Include dependency graph for DynamicLines.hh:

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::DynamicLines**

  Class for drawing lines that can change.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

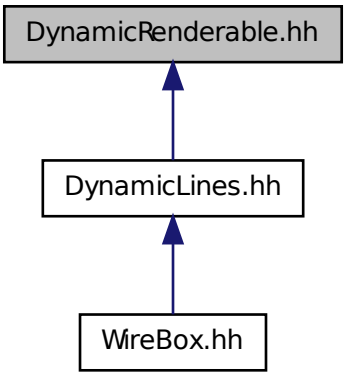
  Rendering namespace.

11.38 DynamicRenderable.hh File Reference

```
#include "rendering/ogre_gazebo.h"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for DynamicRenderable.hh:

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::DynamicRenderable**

  Abstract base class providing mechanisms for dynamically growing hardware buffers.

Namespaces

- namespace **gazebo**

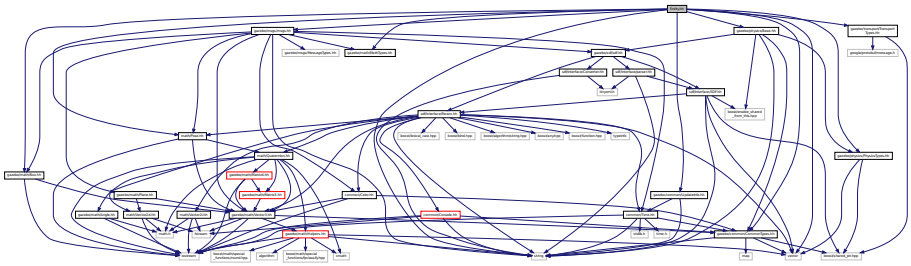
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

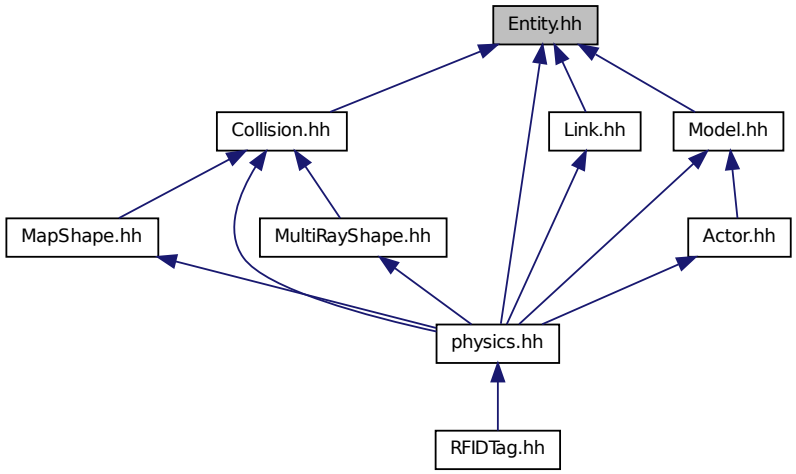
11.39 Entity.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Base.hh"
```

Include dependency graph for Entity.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Entity**

  Base *(p. 137) class for all physics objects in Gazebo.*

Namespaces

- namespace **boost**
- namespace **gazebo**

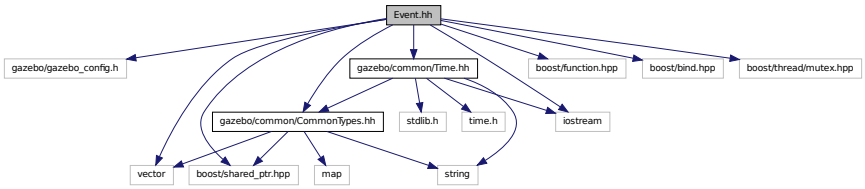
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

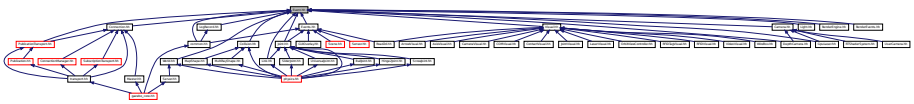
11.40 Event.hh File Reference

```
#include <gazebo/gazebo_config.h>
#include <gazebo/common/Time.hh>
#include <gazebo/common/CommonTypes.hh>
#include <boost/function.hpp>
#include <boost/bind.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <iostream>
#include <vector>
```

Include dependency graph for Event.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::event::Connection**

  A (p. *111) class that encapsulates a connection.*

- class **gazebo::event::Event**

  Base class for all events.

- class **gazebo::event::EventT< T >**

  A (p. *111) class for event processing.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

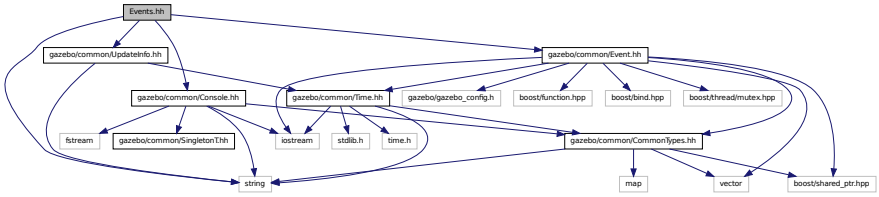
- namespace **gazebo::event**

  Event (p. *292) namespace.*

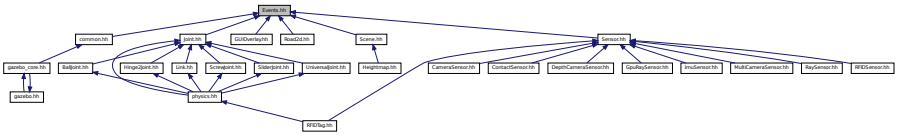
11.41 Events.hh File Reference

```
#include <string>
#include "gazebo/common/Console.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
```

Include dependency graph for Events.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::event::Events**

  *An **Event** (p. 292) class to get notifications for simulator events.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

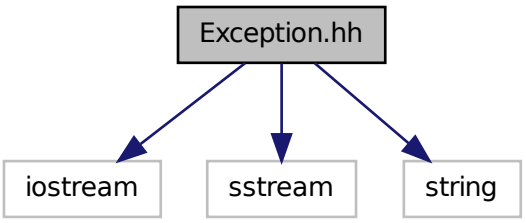
- namespace **gazebo::event**

  ***Event** (p. 292) namespace.*

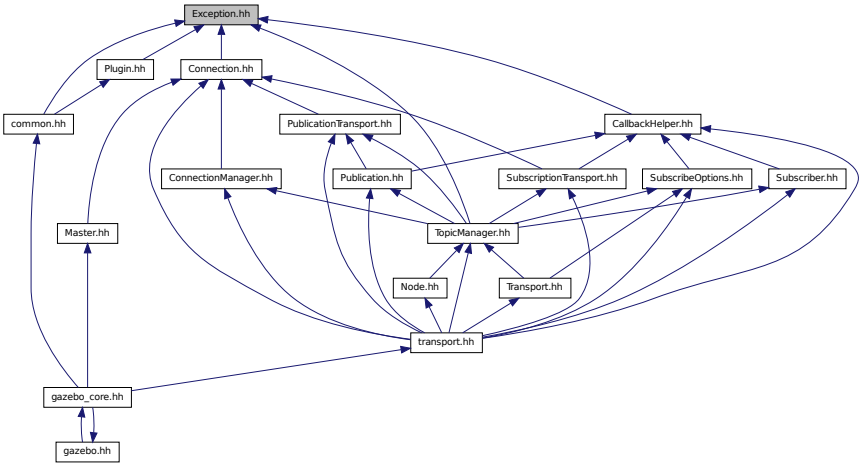
11.42 Exception.hh File Reference

```
#include <iostream>
#include <sstream>
#include <string>
```

Include dependency graph for Exception.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::AssertionInternalError**

  Class for generating Exceptions which come from gazebo assertions.
- class **gazebo::common::Exception**

  Class for generating exceptions.
- class **gazebo::common::InternalError**

  Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

Macros

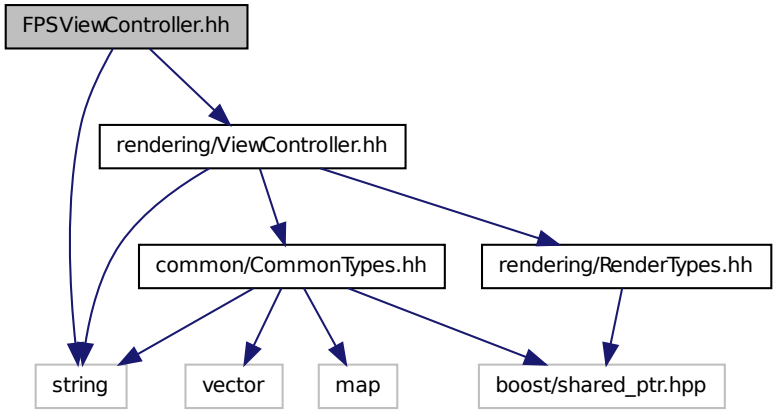
- #define **gzthrow**(msg)

  This macro logs an error to the throw stream and throws an exception that contains the file name and line number.

11.43 FPSViewController.hh File Reference

```
#include <string>
#include "rendering/ViewController.hh"
```

Include dependency graph for FPSViewController.hh:



Classes

- class **gazebo::rendering::FPSViewController**

  First Person Shooter style view controller.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

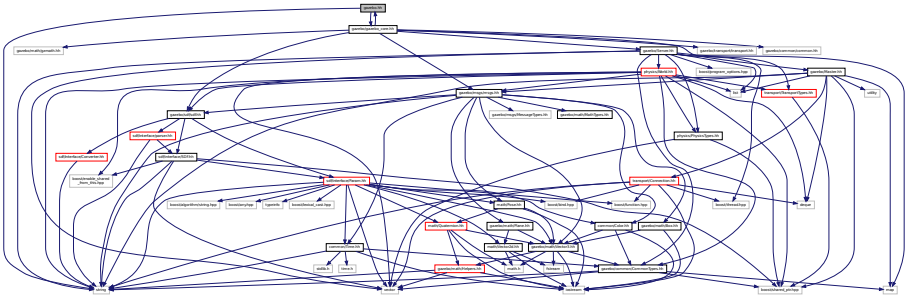
- namespace **gazebo::rendering**

  Rendering namespace.

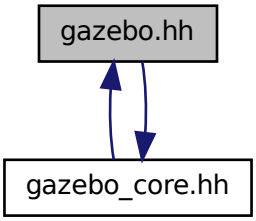
11.44 gazebo.hh File Reference

```
#include <gazebo/gazebo_core.hh>
#include <string>
```

Include dependency graph for gazebo.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

Functions

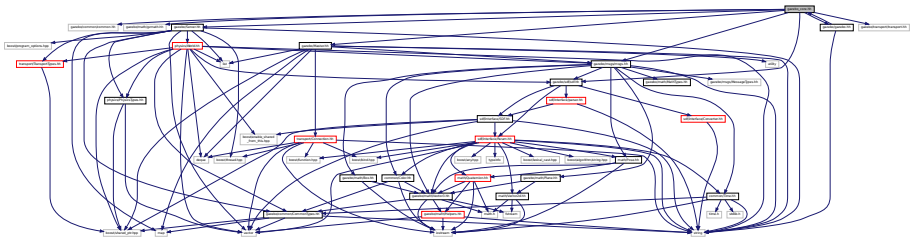
- void **gazebo::add_plugin** (const std::string &_filename)
- std::string **gazebo::find_file** (const std::string &_file)

  Find a file in the gazebo search paths.
- void **gazebo::fini** ()
- bool **gazebo::init** ()
- bool **gazebo::load** (int _argc=0, char ∗∗_argv=0)
- void **gazebo::print_version** ()
- void **gazebo::run** ()
- void **gazebo::stop** ()

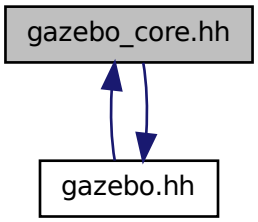
11.45 gazebo_core.hh File Reference

```
#include <gazebo/common/common.hh>
#include <gazebo/math/gzmath.hh>
#include <gazebo/msgs/msgs.hh>
#include <gazebo/sdf/sdf.hh>
#include <gazebo/transport/transport.hh>
#include <gazebo/Server.hh>
#include <gazebo/Master.hh>
#include <gazebo/gazebo.hh>
```

Include dependency graph for gazebo_core.hh:



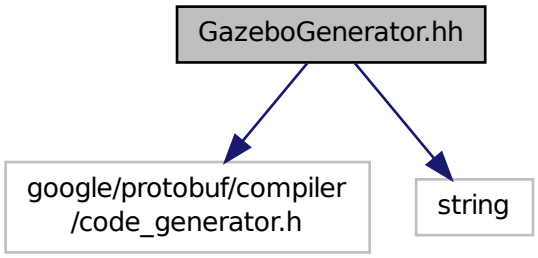
This graph shows which files directly or indirectly include this file:



11.46 GazeboGenerator.hh File Reference

```
#include <google/protobuf/compiler/code_generator.h>
#include <string>
```

Include dependency graph for GazeboGenerator.hh:



Classes

- class **google::protobuf::compiler::cpp::GazeboGenerator**

  *Google protobuf message generator for **gazebo::msgs*** (p. 89).

Namespaces

- namespace **google**
- namespace **google::protobuf**
- namespace **google::protobuf::compiler**
- namespace **google::protobuf::compiler::cpp**

11.47 GpuLaser.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/rendering/Camera.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Vector2i.hh"
#include "gazebo/sdf/sdf.hh"
```

Include dependency graph for GpuLaser.hh:

Classes

- class **gazebo::rendering::GpuLaser**

  GPU based laser distance sensor.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

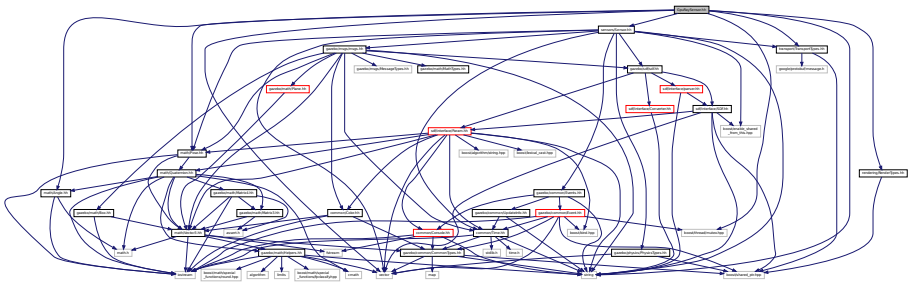
  Common namespace.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.48 GpuRaySensor.hh File Reference

```
#include <vector>
#include <string>
#include <boost/thread/mutex.hpp>
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "transport/TransportTypes.hh"
#include "sensors/Sensor.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for GpuRaySensor.hh:



Classes

- class **gazebo::sensors::GpuRaySensor**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.49 Grid.hh File Reference

```
#include <stdint.h>
#include <vector>
#include <string>
#include "rendering/ogre_gazebo.h"
#include "common/Color.hh"
```

Include dependency graph for Grid.hh:

Classes

- class **gazebo::rendering::Grid**

  Displays a grid of cells, drawn with lines.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

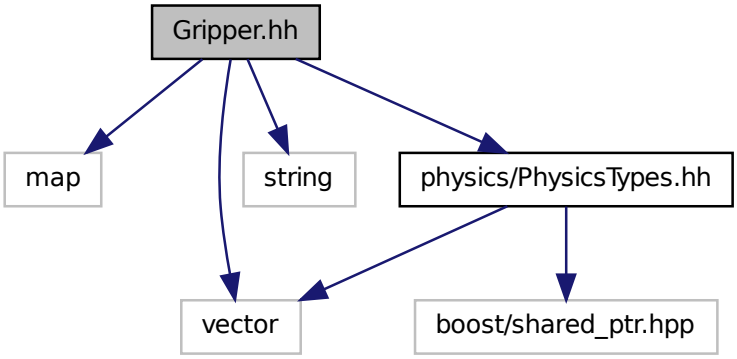
  Rendering namespace.

- namespace **Ogre**

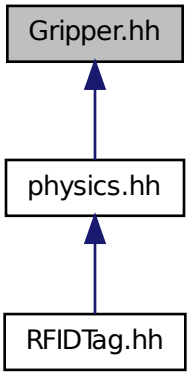
11.50 Gripper.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#include "physics/PhysicsTypes.hh"
```

Include dependency graph for Gripper.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Gripper**

  A *(*p. *111) gripper abstraction.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

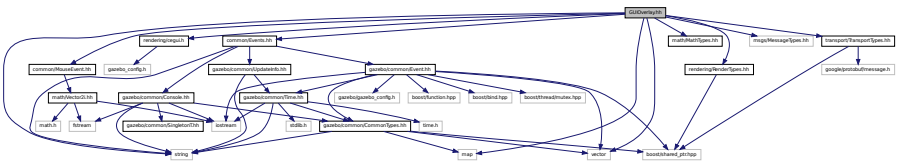
- namespace **gazebo::physics**

  namespace for physics

11.51 GUIOverlay.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "rendering/cegui.h"
#include "common/MouseEvent.hh"
#include "common/Events.hh"
#include "math/MathTypes.hh"
#include "rendering/RenderTypes.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for GUIOverlay.hh:



Classes

- class **gazebo::rendering::GUIOverlay**

  A (p. *111) class that creates a CEGUI overlay on a render window.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

  Rendering namespace.

- namespace **Ogre**

11.52 Heightmap.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/ogre_gazebo.h"
#include "common/Image.hh"
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
#include "rendering/Scene.hh"
```

Include dependency graph for Heightmap.hh:

Classes

- class **gazebo::rendering::GzTerrainMatGen**
- class **gazebo::rendering::Heightmap**

  Rendering a terrain using heightmap information.
- class **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg**

  Keeping the CG shader for reference.
- class **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL**

  Utility class to help with generating shaders for GLSL.
- class **gazebo::rendering::GzTerrainMatGen::SM2Profile**

  Shader model 2 profile target.

Namespaces

- namespace **gazebo**

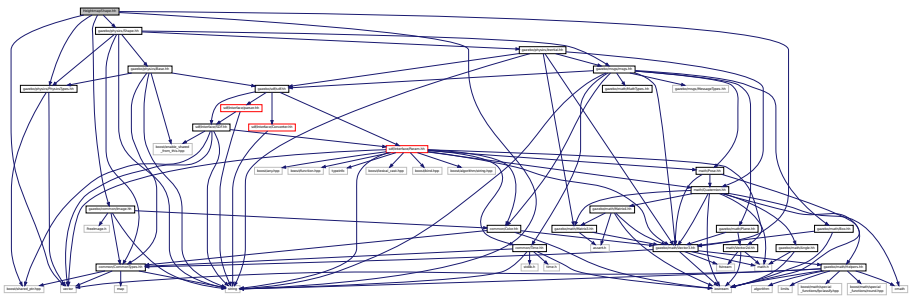
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

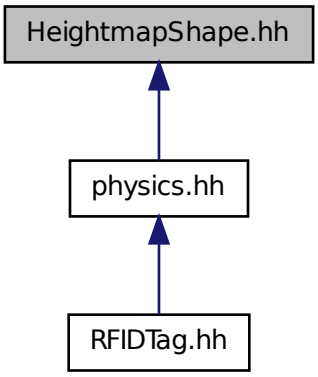
11.53 HeightmapShape.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/common/Image.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for HeightmapShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::HeightmapShape**

  HeightmapShape *(*p. *352) collision shape builds a heightmap from an image.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

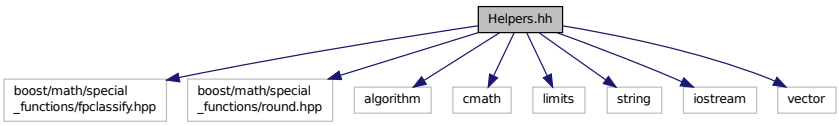
- namespace **gazebo::physics**

  namespace for physics

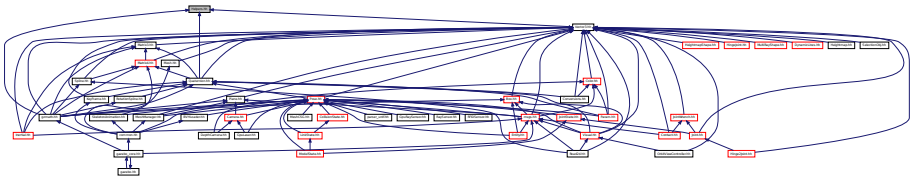
11.54 Helpers.hh File Reference

```
#include <boost/math/special_functions/fpclassify.hpp>
#include <boost/math/special_functions/round.hpp>
#include <algorithm>
#include <cmath>
#include <limits>
#include <string>
#include <iostream>
#include <vector>
```

Include dependency graph for Helpers.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

Macros

- #define **GZ_DBL_MAX** std::numeric_limits<double>::max()
- #define **GZ_DBL_MIN** std::numeric_limits<double>::min()
- #define **GZ_FLT_MAX** std::numeric_limits<float>::max()
- #define **GZ_FLT_MIN** std::numeric_limits<float>::min()

Functions

- template<typename T >
  T **gazebo::math::clamp** (T _v, T _min, T _max)

  Simple clamping function.
- template<typename T >
  bool **gazebo::math::equal** (const T &_a, const T &_b, const T &_epsilon=1e-6)

  check if two values are equal, within a tolerance
- bool **gazebo::math::isnan** (float _v)

  check if a float is NaN
- bool **gazebo::math::isnan** (double _v)

  check if a double is NaN
- bool **gazebo::math::isPowerOfTwo** (unsigned int _x)

  is this a power of 2?

- template<typename T >
  T **gazebo::math::max** (const std::vector< T > &_values)

  get the maximum value of vector of values

- template<typename T >
  T **gazebo::math::mean** (const std::vector< T > &_values)

  get mean of vector of values

- template<typename T >
  T **gazebo::math::min** (const std::vector< T > &_values)

  get the minimum value of vector of values

- double **gazebo::math::parseFloat** (const std::string &_input)

  parse string into float

- int **gazebo::math::parseInt** (const std::string &_input)

  parse string into an integer

- template<typename T >
  T **gazebo::math::precision** (const T &_a, const unsigned int &_precision)

  get value at a specified precision

- template<typename T >
  T **gazebo::math::variance** (const std::vector< T > &_values)

  get variance of vector of values

Variables

- static const double **gazebo::math::NAN_D** = std::numeric_limits<double>::quiet_NaN()

  Returns the representation of a quiet not a number (NAN)

- static const int **gazebo::math::NAN_I** = std::numeric_limits<int>::quiet_NaN()

  Returns the representation of a quiet not a number (NAN)

11.54.1 Macro Definition Documentation

11.54.1.1 #define GZ_DBL_MAX std::numeric_limits<double>::max()

11.54.1.2 #define GZ_DBL_MIN std::numeric_limits<double>::min()

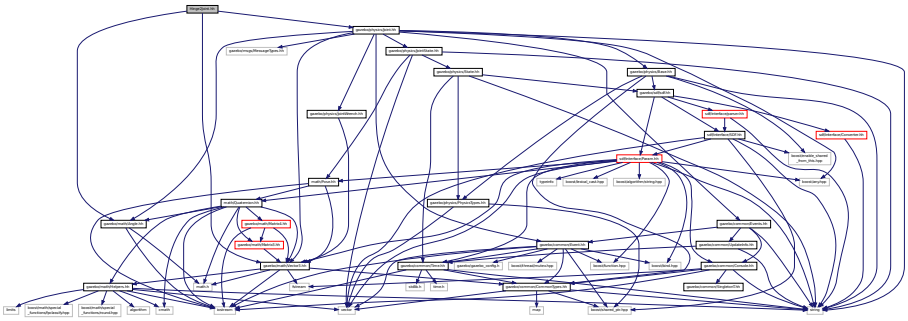
11.54.1.3 #define GZ_FLT_MAX std::numeric_limits<float>::max()

11.54.1.4 #define GZ_FLT_MIN std::numeric_limits<float>::min()

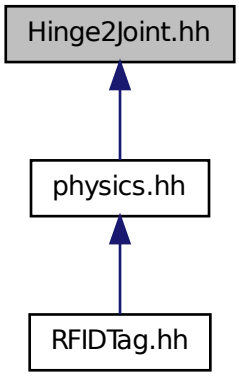
11.55 Hinge2Joint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for Hinge2Joint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Hinge2Joint**< **T** >

  A (p. *111) two axis hinge joint.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

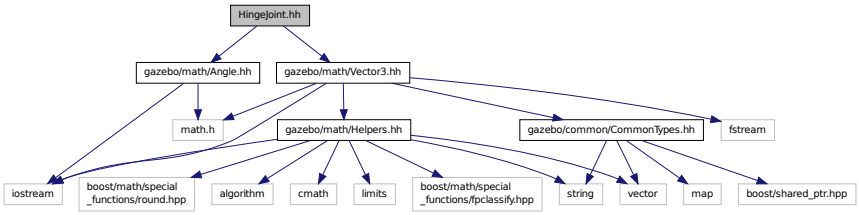
- namespace **gazebo::physics**

  namespace for physics

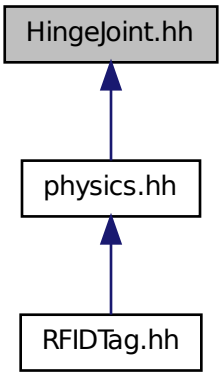
11.56 HingeJoint.hh File Reference

```
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for HingeJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::HingeJoint< T >**

  A (p. *111*) *single axis hinge joint.*

Namespaces

- namespace **gazebo**

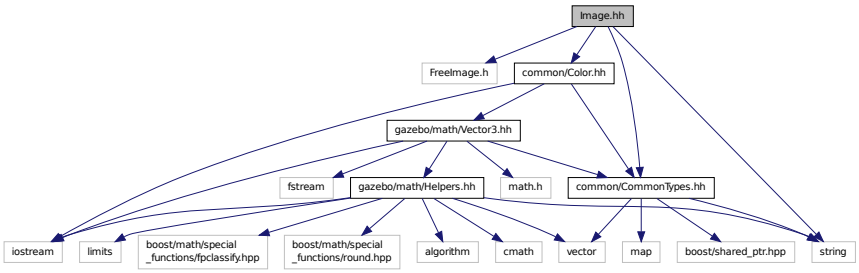
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

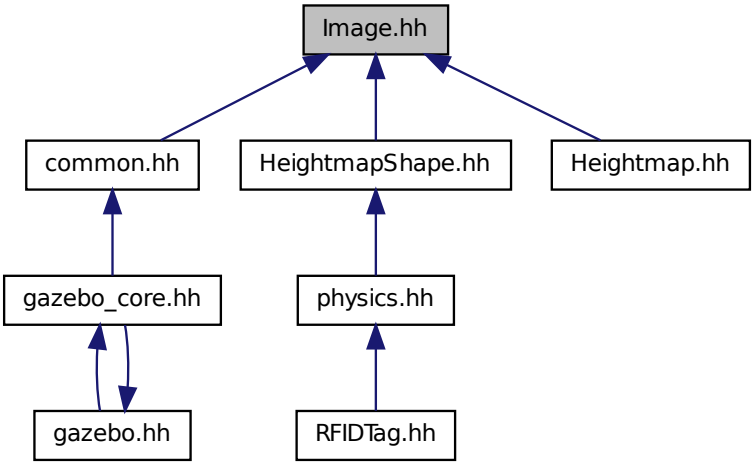
11.57 Image.hh File Reference

```
#include <FreeImage.h>
#include <string>
#include "common/CommonTypes.hh"
#include "common/Color.hh"
```

Include dependency graph for Image.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Image**

  Encapsulates an image.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Variables

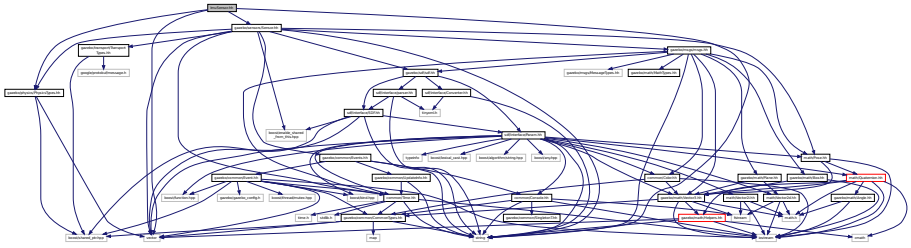
- static std::string **gazebo::common::PixelFormatNames** [ ]

  String names for the pixel formats.

11.58 ImuSensor.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sensors/Sensor.hh"
```

Include dependency graph for ImuSensor.hh:



Classes

- class **gazebo::sensors::ImuSensor**

  An IMU sensor.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

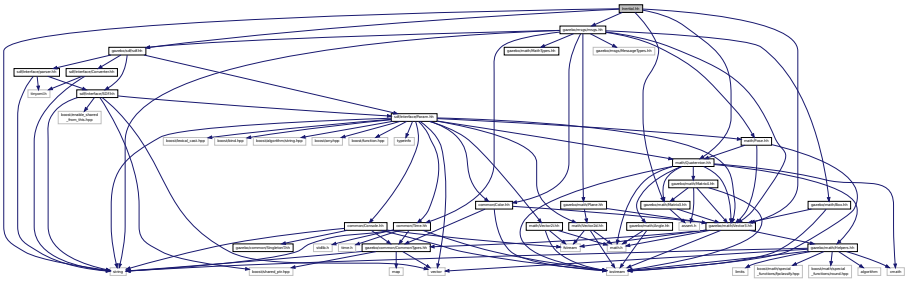
  Sensors namespace.

11.59 Inertial.hh File Reference

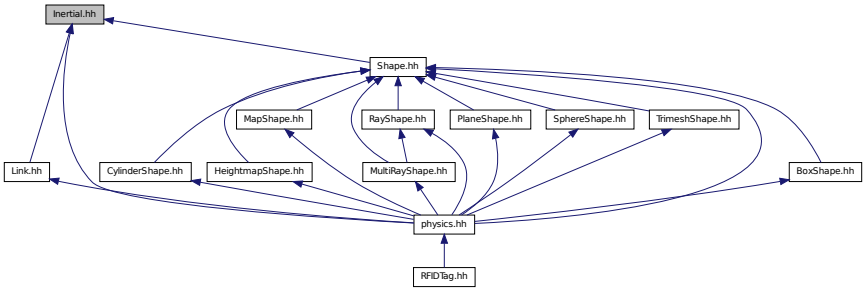
```
#include <string>
```

```
#include "gazebo/msgs/msgs.hh"
#include "gazebo/sdf/sdf.hh"
#include "gazebo/math/Quaternion.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix3.hh"
```

Include dependency graph for Inertial.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Inertial**

  A (p. *111*) *class for inertial information about a link.*

Namespaces

- namespace **gazebo**

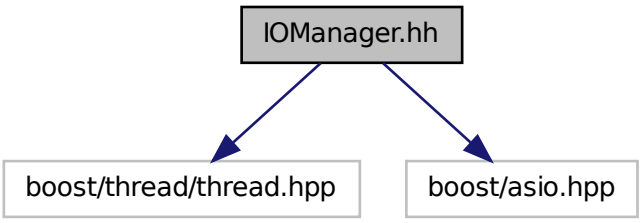
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

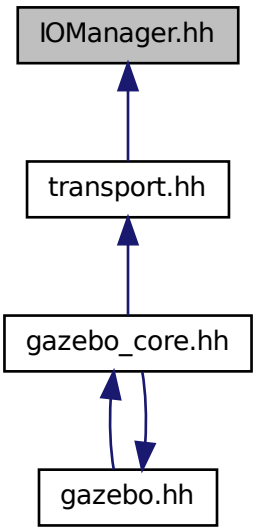
11.60 IOManager.hh File Reference

```
#include <boost/thread/thread.hpp>
#include <boost/asio.hpp>
```

Include dependency graph for IOManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::IOManager**

  Manages boost::asio IO.

Namespaces

- namespace **gazebo**

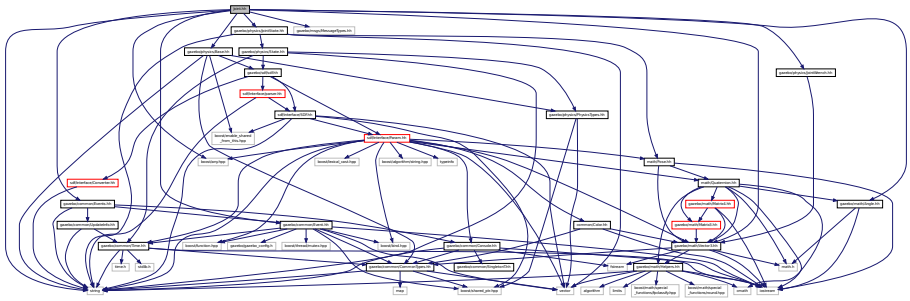
Forward declarations for the common classes.

- namespace **gazebo::transport**

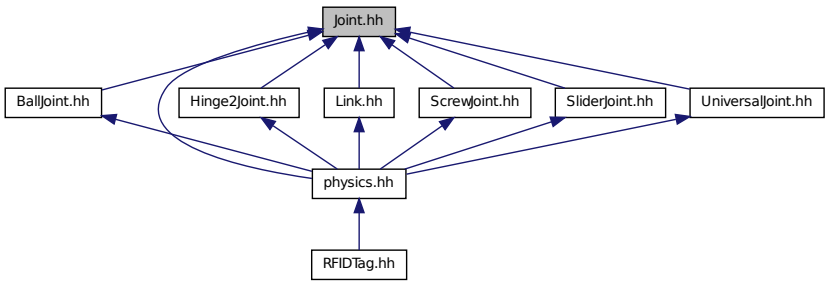
11.61 Joint.hh File Reference

```
#include <string>
#include <boost/any.hpp>
#include "gazebo/common/Event.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/physics/JointState.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/physics/JointWrench.hh"
```

Include dependency graph for Joint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Joint**

  Base *(p. 137) class for all joints.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::physics**

  namespace for physics

Macros

- #define **MAX_JOINT_AXIS** 2

  maximum number of axis per joint anticipated.

11.61.1 Macro Definition Documentation

11.61.1.1 #define MAX_JOINT_AXIS 2

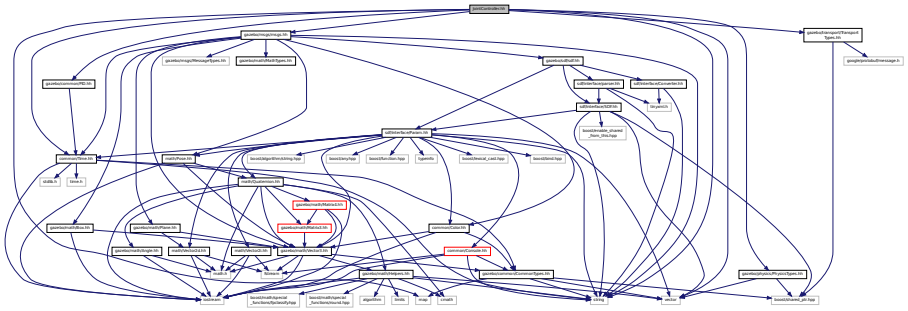
maximum number of axis per joint anticipated.

Currently, this is 2 as 3-axis joints (e.g. ball) actuation, control is not there yet.

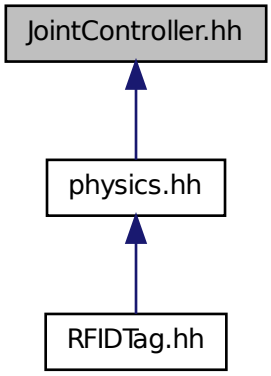
11.62 JointController.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include "gazebo/common/PID.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
```

Include dependency graph for JointController.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointController**

  A (p. *111) class for manipulating* ***physics::Joint*** (p. *381).*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

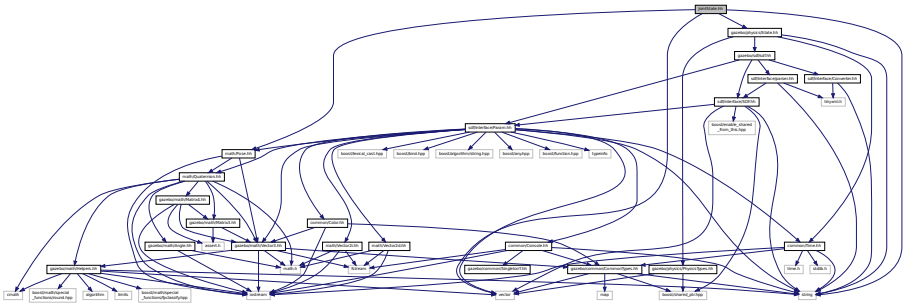
- namespace **gazebo::physics**

  namespace for physics

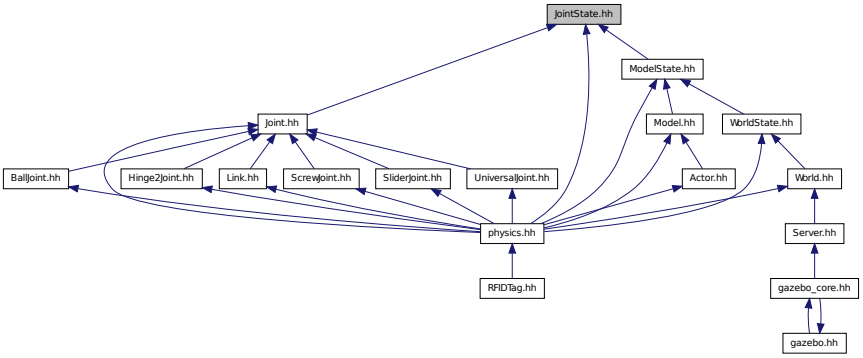
11.63 JointState.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/State.hh"
#include "gazebo/math/Pose.hh"
```

Include dependency graph for JointState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointState**

  *keeps track of state of a **physics::Joint*** (p. *381*)

Namespaces

- namespace **gazebo**

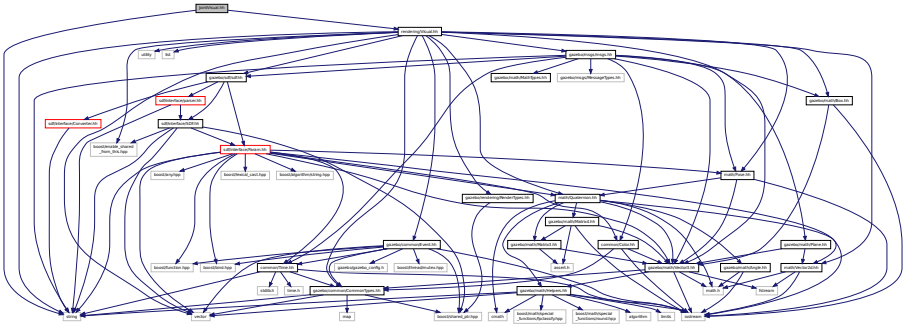
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.64 JointVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
```

Include dependency graph for JointVisual.hh:



Classes

- class **gazebo::rendering::JointVisual**

  Visualization for joints.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

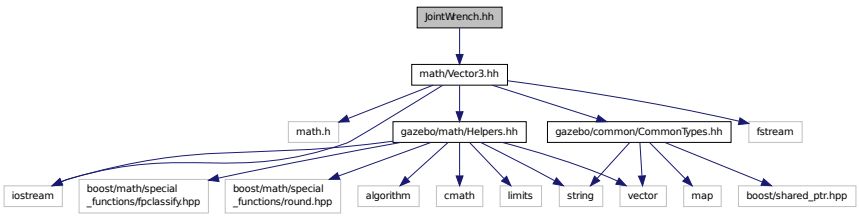
- namespace **gazebo::rendering**

  Rendering namespace.

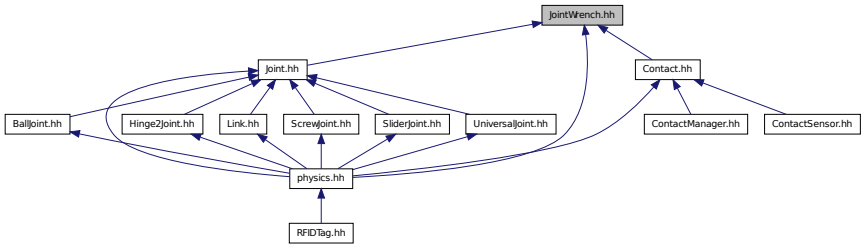
11.65 JointWrench.hh File Reference

```
#include "math/Vector3.hh"
```

Include dependency graph for JointWrench.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointWrench**

  Wrench information from a joint.

Namespaces

- namespace **gazebo**

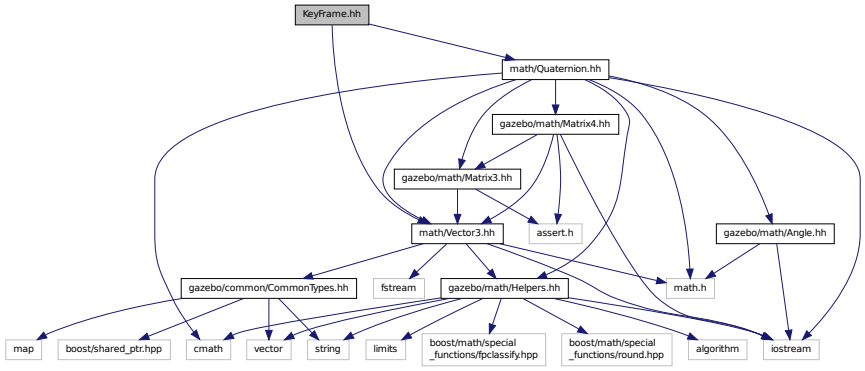
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

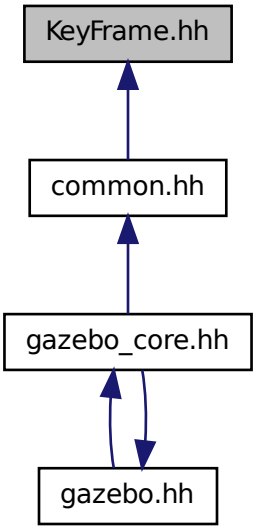
11.66 KeyFrame.hh File Reference

```
#include "math/Vector3.hh"
#include "math/Quaternion.hh"
```

Include dependency graph for KeyFrame.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::KeyFrame**

  A *(p. 111) key frame in an animation.*
- class **gazebo::common::NumericKeyFrame**

  A *(p. 111) keyframe for a* ***NumericAnimation*** *(p. 552).*
- class **gazebo::common::PoseKeyFrame**

  A *(p. 111) keyframe for a* ***PoseAnimation*** *(p. 605).*

Namespaces

- namespace **gazebo**

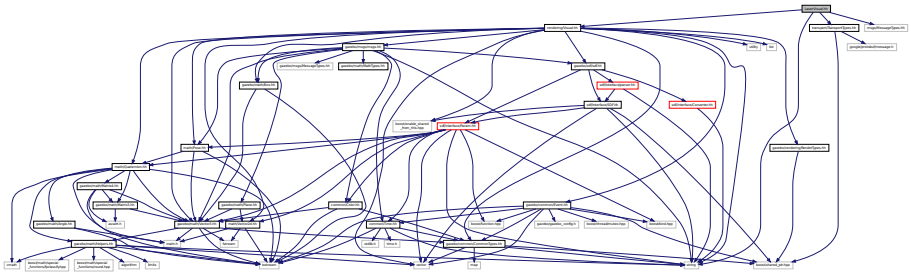
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.67 LaserVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for LaserVisual.hh:



Classes

- class **gazebo::rendering::LaserVisual**

  Visualization for laser data.

Namespaces

- namespace **gazebo**

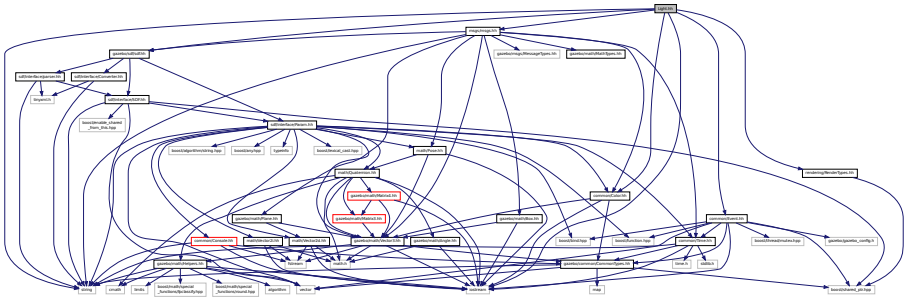
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.68 Light.hh File Reference

```
#include <string>
#include <iostream>
#include "msgs/msgs.hh"
#include "rendering/RenderTypes.hh"
#include "common/Event.hh"
#include "common/Color.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for Light.hh:



Classes

- class **gazebo::rendering::Light**

  A *(p. 111) light source.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

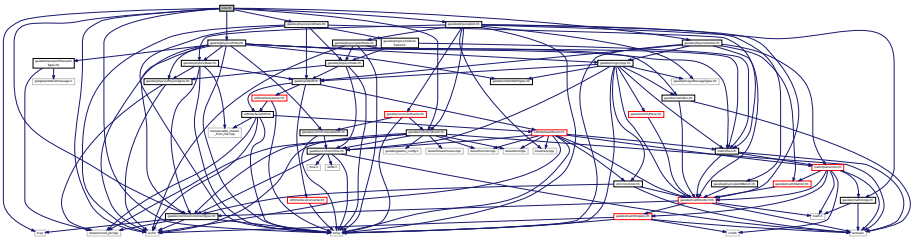
  Rendering namespace.

- namespace **Ogre**

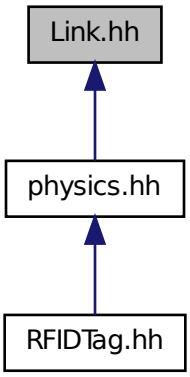
11.69 Link.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#include "gazebo/common/Event.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/LinkState.hh"
#include "gazebo/physics/Entity.hh"
#include "gazebo/physics/Inertial.hh"
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for Link.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Link**

  Link *(*p. *418) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

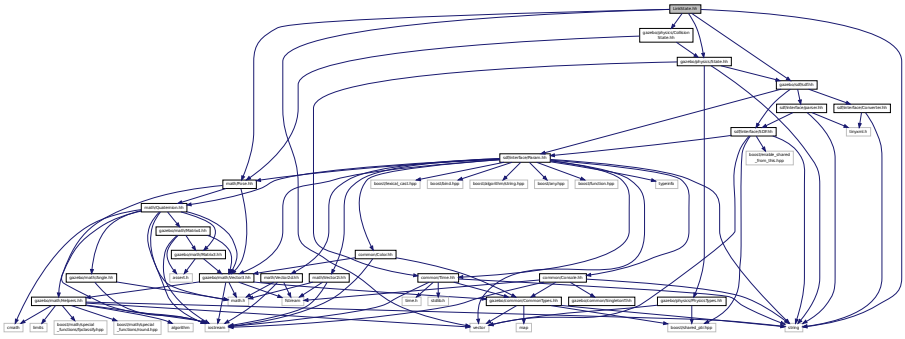
- namespace **gazebo::physics**

  namespace for physics

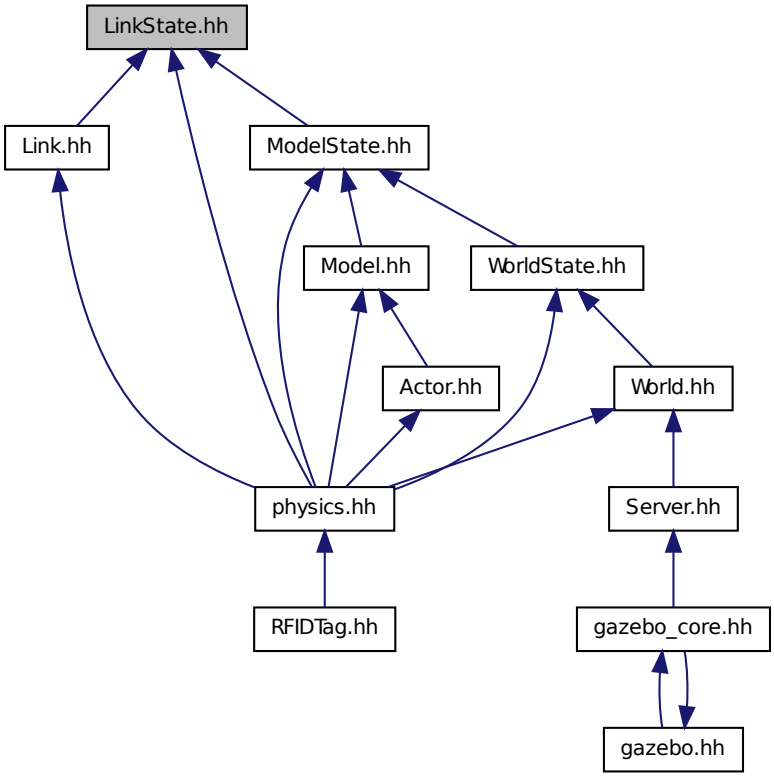
11.70 LinkState.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/sdf/sdf.hh"
#include "gazebo/physics/State.hh"
#include "gazebo/physics/CollisionState.hh"
#include "gazebo/math/Pose.hh"
```

Include dependency graph for LinkState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::LinkState**

  *Store state information of a **physics::Link** (p. 418) object.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

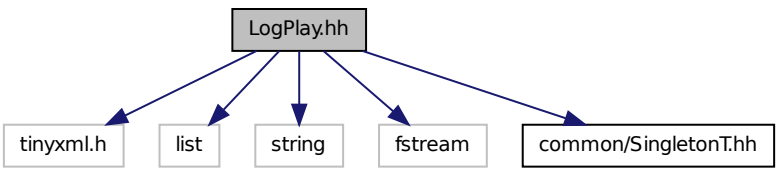
- namespace **gazebo::physics**

  namespace for physics

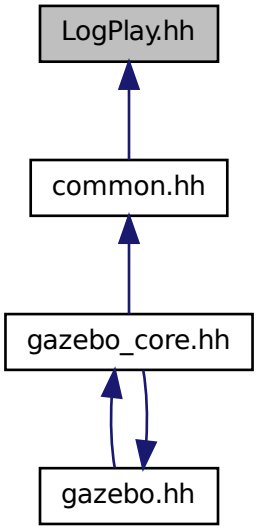
11.71 LogPlay.hh File Reference

```
#include <tinyxml.h>
#include <list>
#include <string>
#include <fstream>
#include "common/SingletonT.hh"
```

Include dependency graph for LogPlay.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::LogPlay**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::common**

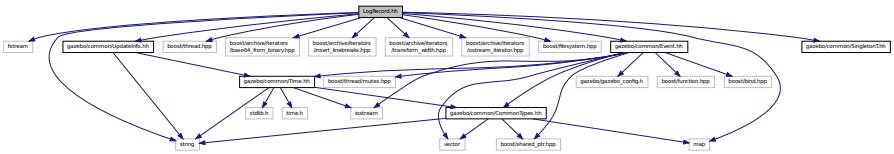
  Common namespace.

11.72 LogRecord.hh File Reference

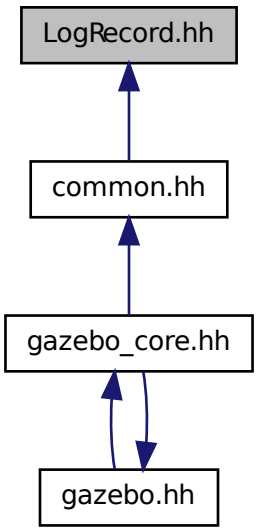
```
#include <fstream>
```

```
#include <string>
#include <map>
#include <boost/thread.hpp>
#include <boost/archive/iterators/base64_from_binary.hpp>
#include <boost/archive/iterators/insert_linebreaks.hpp>
#include <boost/archive/iterators/transform_width.hpp>
#include <boost/archive/iterators/ostream_iterator.hpp>
#include <boost/filesystem.hpp>
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/common/SingletonT.hh"
```

Include dependency graph for LogRecord.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::LogRecord**

  addtogroup gazebo_common

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

Macros

- #define **GZ_LOG_VERSION** "1.0"

11.72.1 Macro Definition Documentation

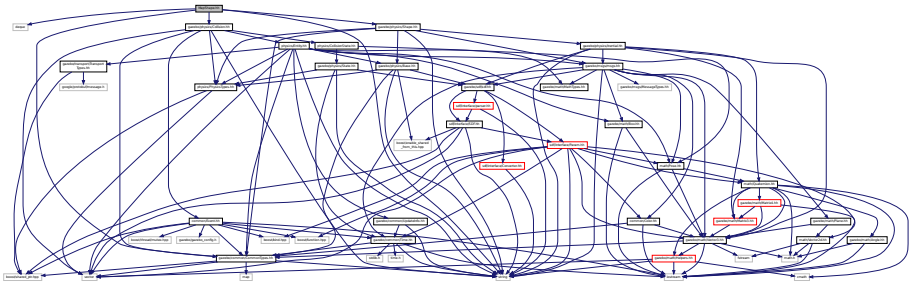
11.72.1.1 #define GZ_LOG_VERSION "1.0"

11.73 mainpage.html File Reference

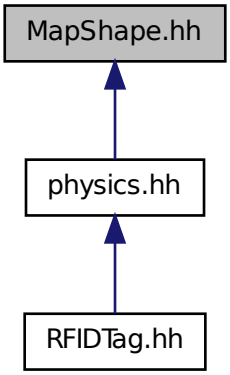
11.74 MapShape.hh File Reference

```
#include <deque>
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for MapShape.hh:



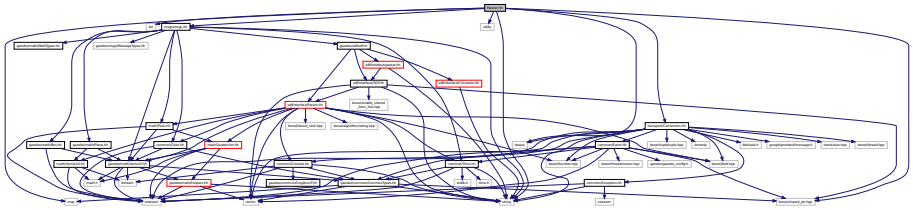
This graph shows which files directly or indirectly include this file:



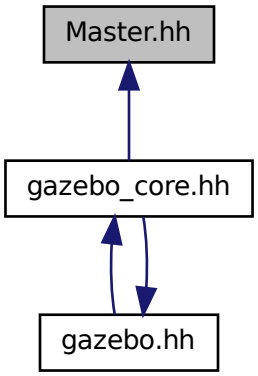
11.75 Master.hh File Reference

```
#include <string>
#include <list>
#include <deque>
#include <utility>
#include <map>
#include <boost/shared_ptr.hpp>
#include "msgs/msgs.hh"
#include "transport/Connection.hh"
```

Include dependency graph for Master.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Master**

  A *(*p. *111) ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.*

Namespaces

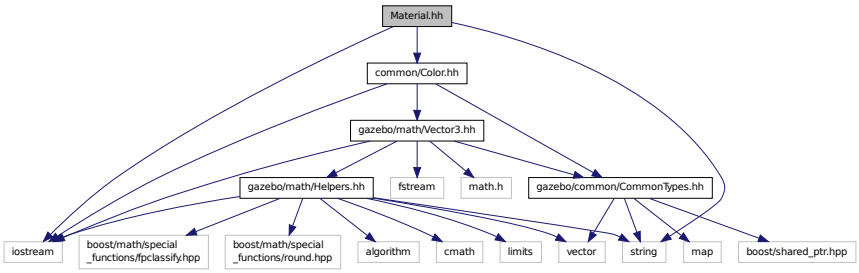
- namespace **gazebo**

  Forward declarations for the common classes.

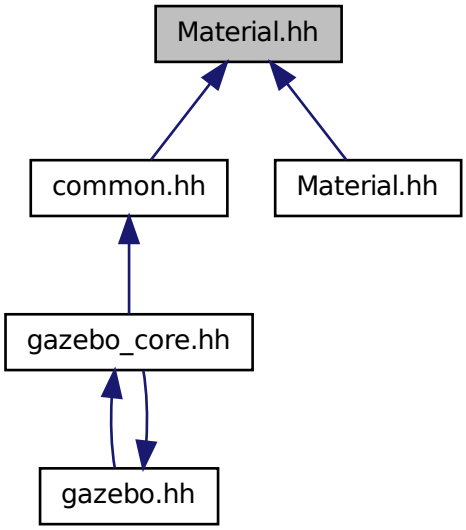
11.76 Material.hh File Reference

```
#include <string>
#include <iostream>
#include "common/Color.hh"
```

Include dependency graph for common/Material.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Material**

  Encapsulates description of a material.

Namespaces

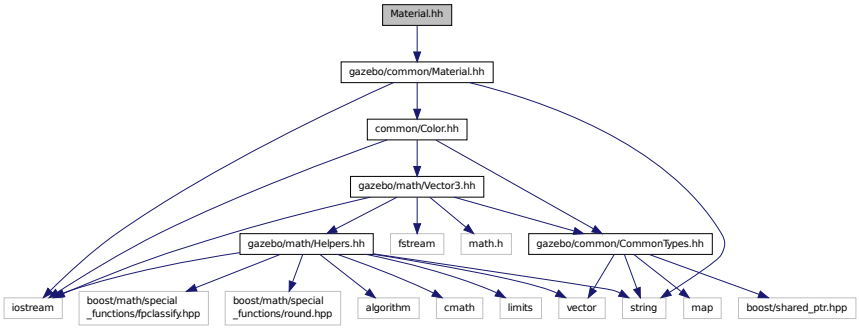
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

Common namespace.

11.77 Material.hh File Reference

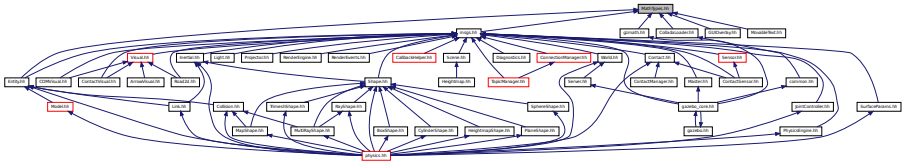
`#include "gazebo/common/Material.hh"`
Include dependency graph for rendering/Material.hh:



11.78 MathTypes.hh File Reference

Forward declarations for the math classes.

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

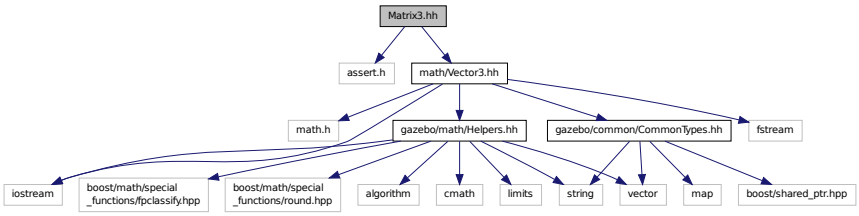
11.78.1 Detailed Description

Forward declarations for the math classes.

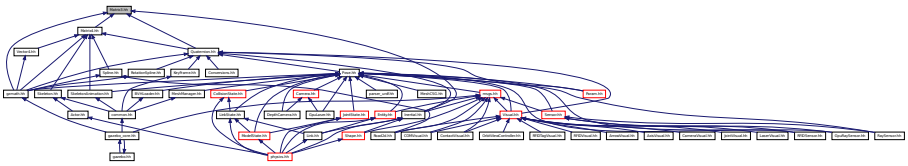
11.79 Matrix3.hh File Reference

```
#include <assert.h>
#include "math/Vector3.hh"
```

Include dependency graph for Matrix3.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Matrix3**

  A (p. *111*) *3x3 matrix class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

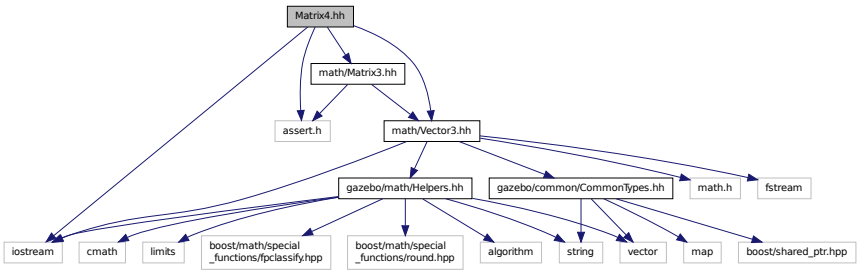
- namespace **gazebo::math**

  Math namespace.

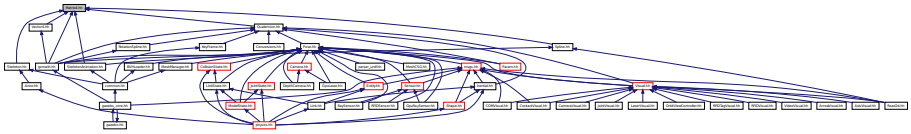
11.80 Matrix4.hh File Reference

```
#include <assert.h>
#include <iostream>
#include "math/Vector3.hh"
#include "math/Matrix3.hh"
```

Include dependency graph for Matrix4.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Matrix4**

  A (p. *111) 3x3 matrix class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

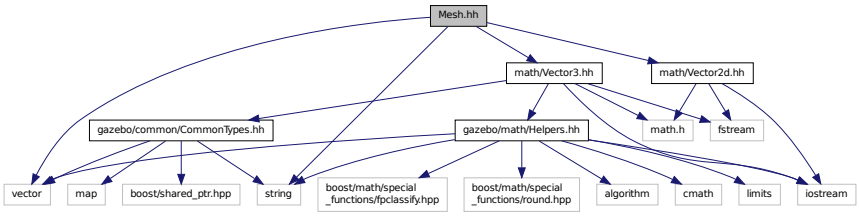
- namespace **gazebo::math**

  Math namespace.

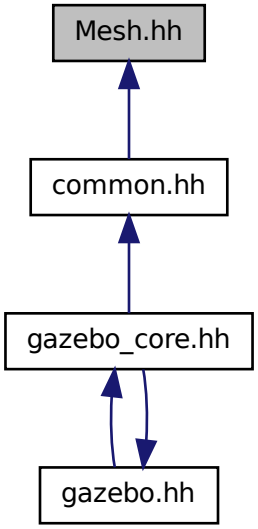
11.81 Mesh.hh File Reference

```
#include <vector>
#include <string>
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
```

Include dependency graph for Mesh.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Mesh**

  A *(*p. *111) 3D mesh.*
- struct **gazebo::common::NodeAssignment**

  Vertex to node weighted assignement for skeleton animation visualization.
- class **gazebo::common::SubMesh**

  A *(*p. *111) child mesh.*

Namespaces

- namespace **gazebo**

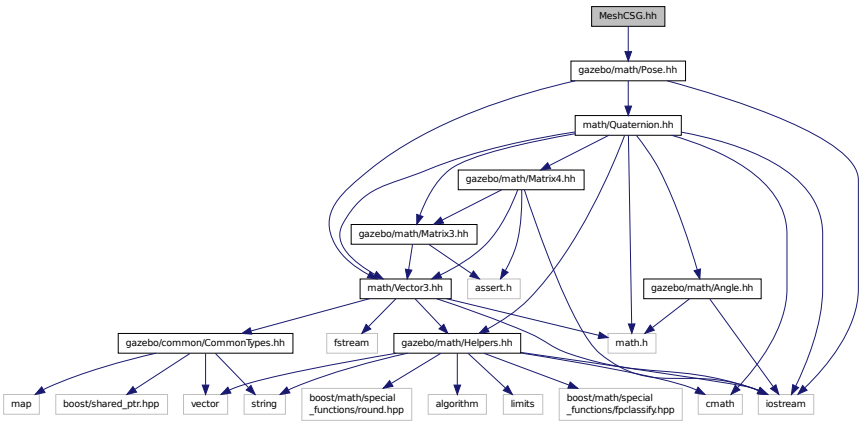
Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

11.82 MeshCSG.hh File Reference

`#include "gazebo/math/Pose.hh"`
Include dependency graph for MeshCSG.hh:



Classes

- class **gazebo::common::MeshCSG**

  Creates CSG meshes.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Typedefs

- typedef _GPtrArray **GPtrArray**
- typedef _GtsSurface **GtsSurface**

11.82.1 Typedef Documentation

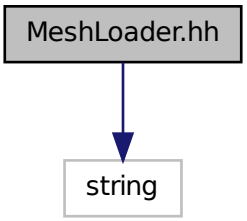
11.82.1.1 typedef _GPtrArray GPtrArray

11.82.1.2 typedef _GtsSurface GtsSurface

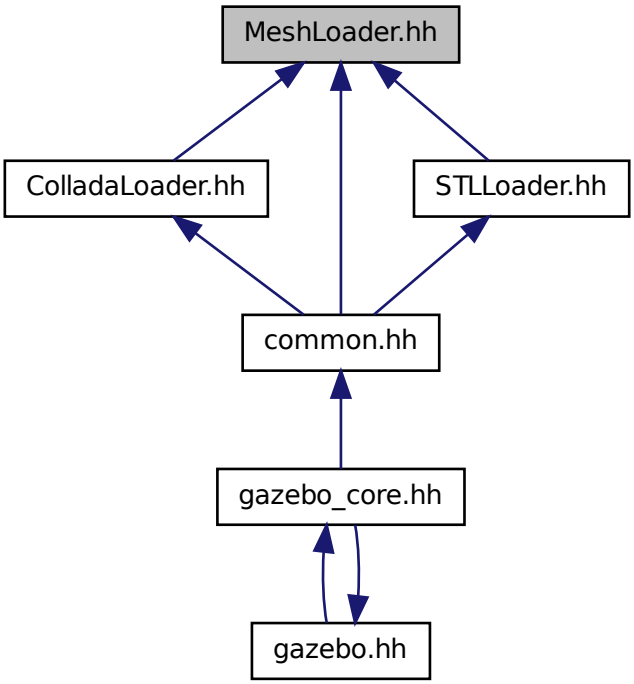
11.83 MeshLoader.hh File Reference

```
#include <string>
```

Include dependency graph for MeshLoader.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MeshLoader**

  Base class for loading meshes.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

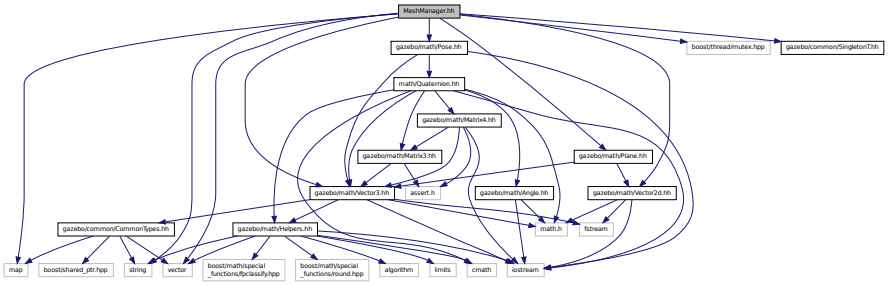
- namespace **gazebo::common**

  Common namespace.

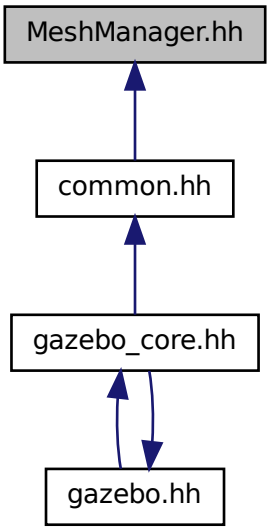
11.84 MeshManager.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include <boost/thread/mutex.hpp>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Plane.hh"
#include "gazebo/common/SingletonT.hh"
```

Include dependency graph for MeshManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MeshManager**

  Maintains and manages all meshes.

Namespaces

- namespace **gazebo**

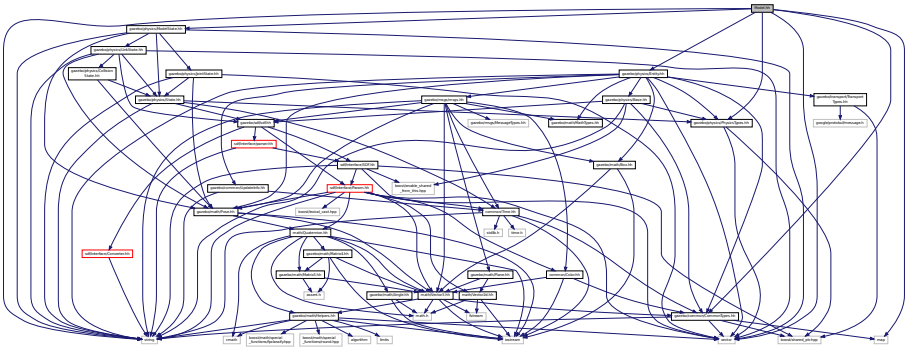
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

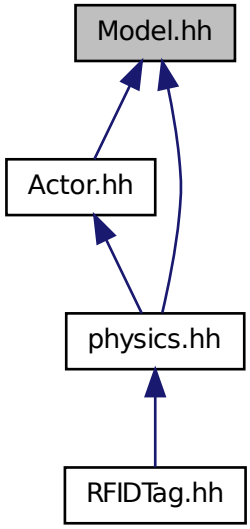
11.85 Model.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/ModelState.hh"
#include "gazebo/physics/Entity.hh"
```

Include dependency graph for Model.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Model**

  A *(*p. *111) model is a collection of links, joints, and plugins.*

Namespaces

- namespace **boost**
- namespace **gazebo**

Forward declarations for the common classes.

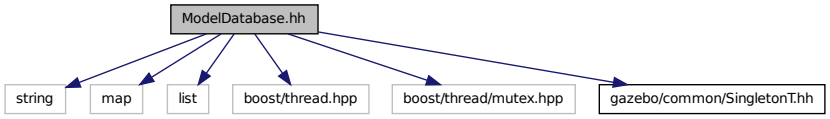
- namespace **gazebo::physics**

  namespace for physics

11.86 ModelDatabase.hh File Reference

```
#include <string>
#include <map>
#include <list>
#include <boost/thread.hpp>
#include <boost/thread/mutex.hpp>
#include "gazebo/common/SingletonT.hh"
```

Include dependency graph for ModelDatabase.hh:



Classes

- class **gazebo::common::ModelDatabase**

  Connects to model database, and has utility functions to find models.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Macros

- #define **GZ_MODEL_DB_MANIFEST_FILENAME** "database.config"

  The file name of model database XML configuration.
- #define **GZ_MODEL_MANIFEST_FILENAME** "model.config"

  The file name of model XML configuration.

11.86.1 Macro Definition Documentation

11.86.1.1 #define GZ_MODEL_DB_MANIFEST_FILENAME "database.config"

The file name of model database XML configuration.

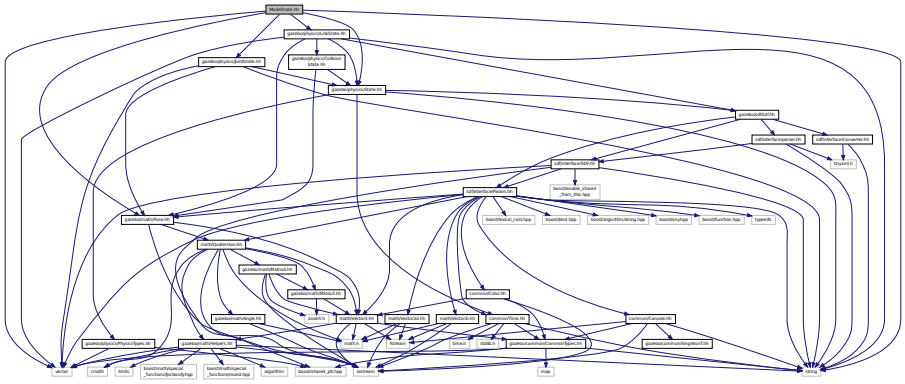
11.86.1.2 #define GZ_MODEL_MANIFEST_FILENAME "model.config"

The file name of model XML configuration.

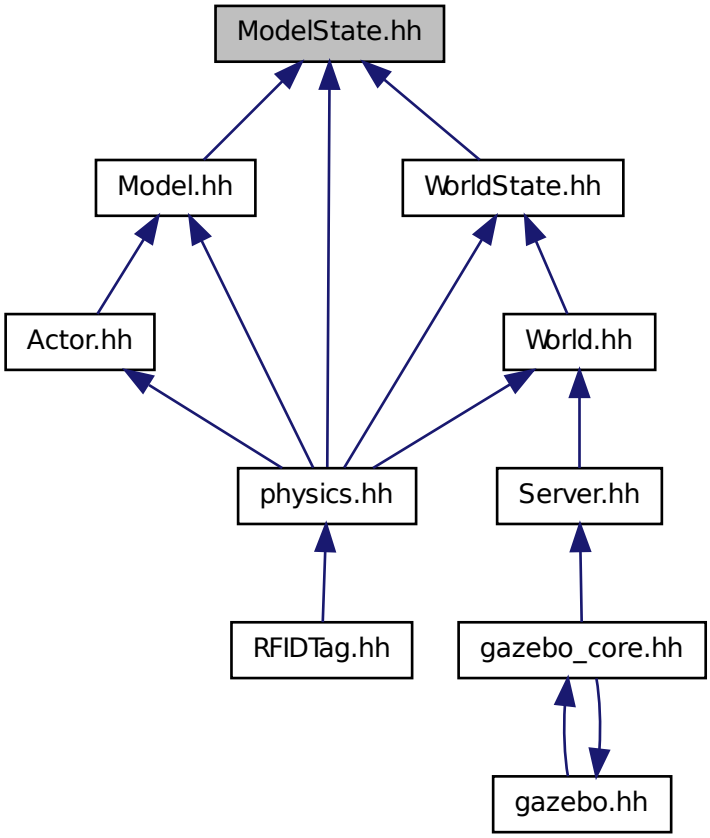
11.87 ModelState.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/State.hh"
#include "gazebo/physics/LinkState.hh"
#include "gazebo/physics/JointState.hh"
```

Include dependency graph for ModelState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ModelState**

  *Store state information of a **physics::Model** (p. 489) object.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

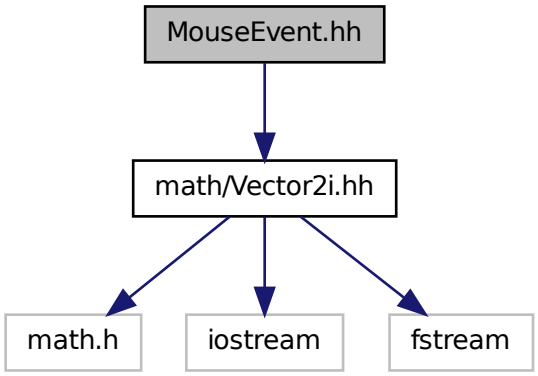
- namespace **gazebo::physics**

  namespace for physics

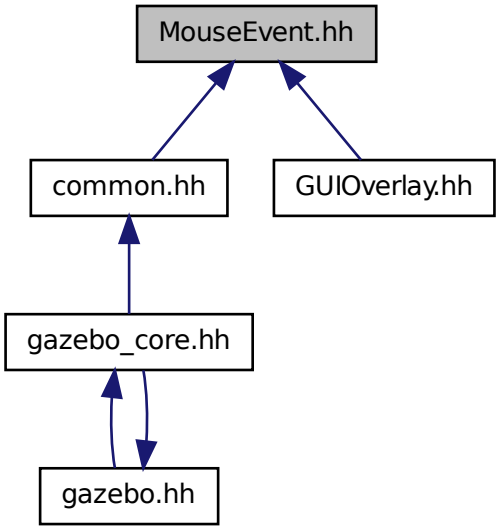
11.88 MouseEvent.hh File Reference

```
#include "math/Vector2i.hh"
```

Include dependency graph for MouseEvent.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MouseEvent**

  Generic description of a mouse event.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.89 MovableText.hh File Reference

```
#include <string>
#include "rendering/ogre_gazebo.h"
#include "common/CommonTypes.hh"
#include "common/Color.hh"
#include "math/MathTypes.hh"
```

Include dependency graph for MovableText.hh:

Classes

- class **gazebo::rendering::MovableText**

  Movable text.

Namespaces

- namespace **boost**
- namespace **gazebo**

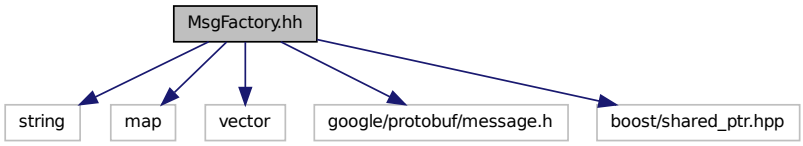
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.90 MsgFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include <google/protobuf/message.h>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for MsgFactory.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::msgs::MsgFactory**

  A *(p. 111) factory that generates protobuf message based on a string type.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::msgs**

  Messages namespace.

Macros

- #define **GZ_REGISTER_STATIC_MSG**(_msgtype, _classname)

  Static message registration macro.

Typedefs

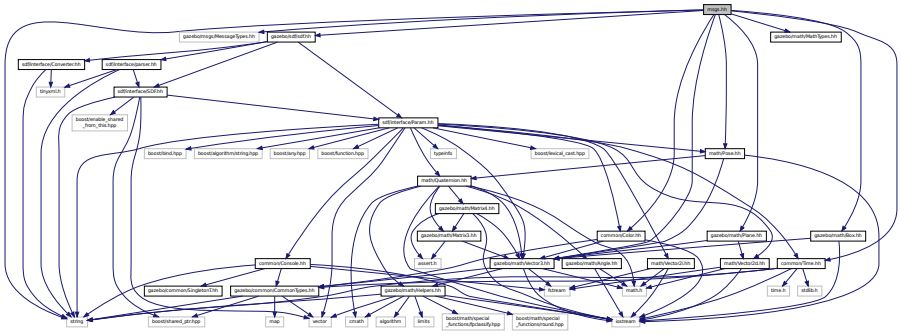
- typedef
  google::protobuf::Message ∗(∗ **gazebo::msgs::MsgFactoryFn**)()

11.91 msgs.hh File Reference

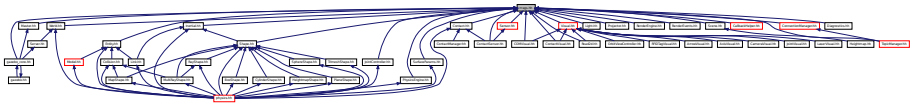
```
#include <string>
```

```
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/sdf/sdf.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Plane.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/common/Color.hh"
#include "gazebo/common/Time.hh"
```

Include dependency graph for msgs.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::msgs**

  Messages namespace.

Functions

- msgs::Vector3d **gazebo::msgs::Convert** (const math::Vector3 &_v)

  *Convert a **math::Vector3** (p. 855) to a msgs::Vector3d.*
- msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)

  *Convert a **math::Quaternion** (p. 623) to a msgs::Quaternion.*
- msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)

  *Convert a **math::Pose** (p. 596) to a msgs::Pose.*
- msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)

  *Convert a **common::Color** (p. 208) to a msgs::Color.*
- msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)

*Convert a **common::Time** (p. 791) to a msgs::Time.*

- msgs::PlaneGeom **gazebo::msgs::Convert** (const math::Plane &_p)

  *Convert a **math::Plane** (p. 587) 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. 623).*
- math::Pose **gazebo::msgs::Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose** (p. 596).*
- common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color** (p. 208).*
- common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time** (p. 791).*
- math::Plane **gazebo::msgs::Convert** (const msgs::PlaneGeom &_p)

  Convert a msgs::PlaneGeom to a common::Plane.
- msgs::Request * **gazebo::msgs::CreateRequest** (const std::string &_request, const std::string &_data="")

  Create a request message.
- msgs::Fog **gazebo::msgs::FogFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Fog from a fog SDF element.
- msgs::Geometry **gazebo::msgs::GeometryFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Geometry from a geometry SDF element.
- msgs::Header * **gazebo::msgs::GetHeader** (google::protobuf::Message &_message)

  Get the header from a protobuf message.
- msgs::GUI **gazebo::msgs::GUIFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::GUI from a GUI SDF element.
- void **gazebo::msgs::Init** (google::protobuf::Message &_message, const std::string &_id="")

  Initialize a message.
- msgs::Light **gazebo::msgs::LightFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Light from a light SDF element.
- msgs::MeshGeom **gazebo::msgs::MeshFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::MeshGeom from a mesh SDF element.
- msgs::Scene **gazebo::msgs::SceneFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Scene from a scene SDF element.
- void **gazebo::msgs::Set** (common::Image &_img, const msgs::Image &_msg)

  *Convert a msgs::Image to a **common::Image** (p. 360).*
- void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)

  *Set a msgs::Image from a **common::Image** (p. 360).*
- void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 855).*
- void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 855).*
- void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 623).*
- void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 596).*
- void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)

  *Set a msgs::Color from a **common::Color** (p. 208).*

- void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)

  *Set a msgs::Time from a **common::Time** (p. 791).*
- void **gazebo::msgs::Set** (msgs::PlaneGeom *_p, const math::Plane &_v)

  *Set a msgs::Plane from a **math::Plane** (p. 587).*
- void **gazebo::msgs::Stamp** (msgs::Header *_header)

  Time stamp a header.
- void **gazebo::msgs::Stamp** (msgs::Time *_time)

  Set the time in a time message.
- msgs::TrackVisual **gazebo::msgs::TrackVisualFromSDF** (**sdf::ElementPtr** _sdf)

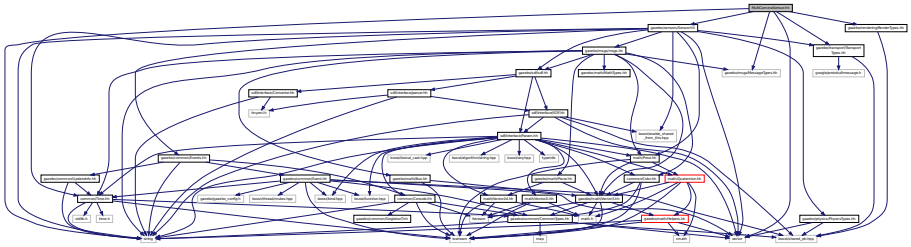
  Create a msgs::TrackVisual from a track visual SDF element.
- msgs::Visual **gazebo::msgs::VisualFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Visual from a visual SDF element.

11.92 MultiCameraSensor.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/RenderTypes.hh"
```

Include dependency graph for MultiCameraSensor.hh:



Classes

- class **gazebo::sensors::MultiCameraSensor**

  Multiple camera sensor.

Namespaces

- namespace **gazebo**

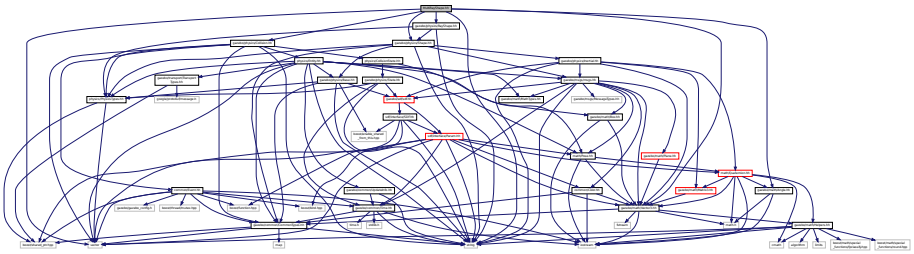
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

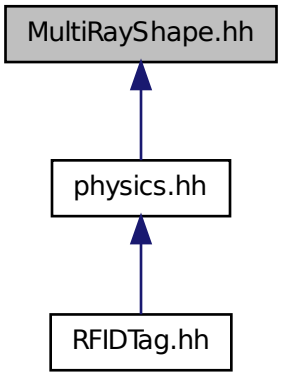
11.93 MultiRayShape.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/physics/Collision.hh"
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/RayShape.hh"
```

Include dependency graph for MultiRayShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::MultiRayShape**

  Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

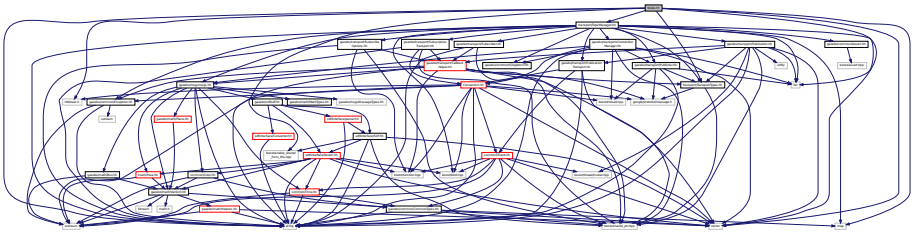
- namespace **gazebo::physics**

  namespace for physics

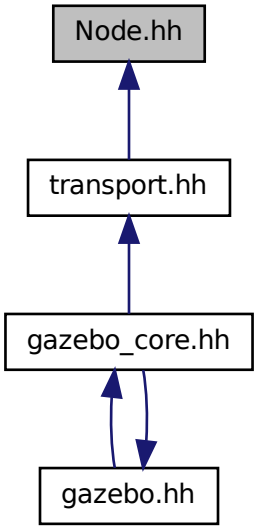
11.94 Node.hh File Reference

```
#include <tbb/task.h>
#include <boost/enable_shared_from_this.hpp>
#include <map>
#include <list>
#include <string>
#include <vector>
#include "transport/TransportTypes.hh"
#include "transport/TopicManager.hh"
```

Include dependency graph for Node.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Node**

  A (p. *111*) *node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.*
- class **gazebo::transport::PublishTask**

Namespaces

- namespace **gazebo**

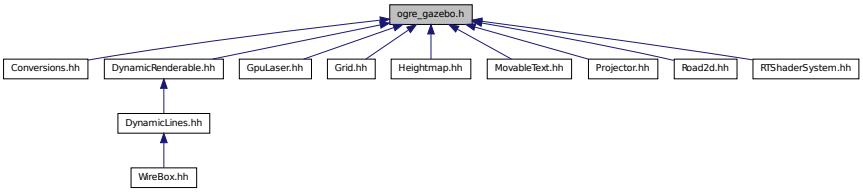
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.95 ogre_gazebo.h File Reference

```
#include <Ogre.h>
#include <OgreImageCodec.h>
#include <OGRE/OgreMovableObject.h>
#include <OGRE/OgreRenderable.h>
#include <OgrePlugin.h>
#include <OgreDataStream.h>
#include <OgreLogManager.h>
#include <OgreWindowEventUtilities.h>
#include <OGRE/OgreSceneQuery.h>
#include <OGRE/OgreRoot.h>
#include <OGRE/OgreSceneManager.h>
#include <OGRE/OgreSceneNode.h>
#include <OGRE/OgreVector3.h>
#include <OGRE/OgreManualObject.h>
#include <OGRE/OgreMaterialManager.h>
#include <OGRE/OgreColourValue.h>
#include <OGRE/OgreQuaternion.h>
#include <OGRE/OgreMesh.h>
#include <OGRE/OgreFontManager.h>
#include <OGRE/OgreHardwareBufferManager.h>
#include <OGRE/OgreCamera.h>
#include <OGRE/OgreNode.h>
#include <OGRE/OgreSimpleRenderable.h>
#include <OGRE/OgreFrameListener.h>
#include <OGRE/OgreTexture.h>
#include <OGRE/OgreRenderObjectListener.h>
#include <OGRE/Terrain/OgreTerrainMaterialGeneratorA.h>
#include <OGRE/Terrain/OgreTerrain.h>
#include <OGRE/Terrain/OgreTerrainGroup.h>
#include <OGRE/OgreTechnique.h>
#include <OGRE/OgrePass.h>
#include <OGRE/OgreTextureUnitState.h>
#include <OGRE/OgreGpuProgramManager.h>
#include <OGRE/OgreHighLevelGpuProgramManager.h>
#include <OGRE/OgreHardwarePixelBuffer.h>
#include <OGRE/OgreShadowCameraSetupPSSM.h>
```

Include dependency graph for ogre_gazebo.h:

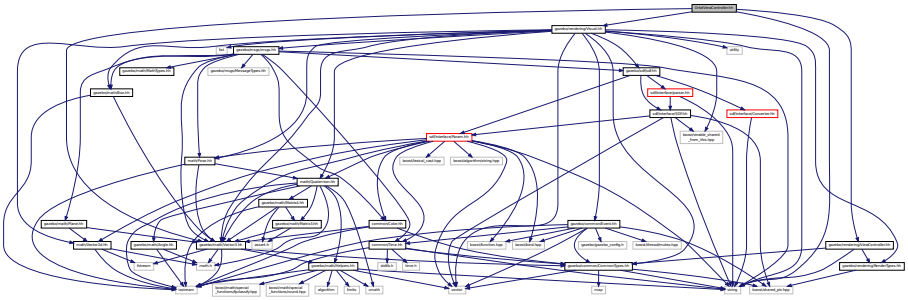
This graph shows which files directly or indirectly include this file:



11.96 OrbitViewController.hh File Reference

```
#include <string>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/rendering/ViewController.hh"
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for OrbitViewController.hh:



Classes

- class **gazebo::rendering::OrbitViewController**

  Orbit view controller.

Namespaces

- namespace **gazebo**

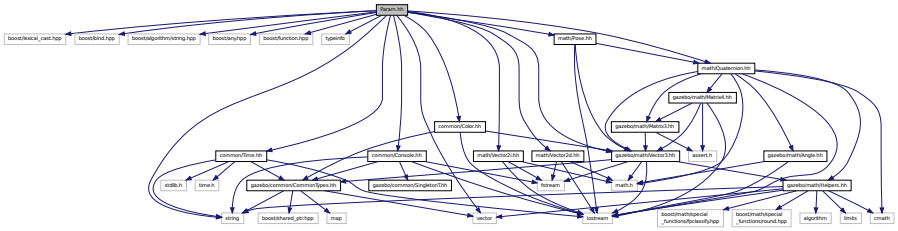
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

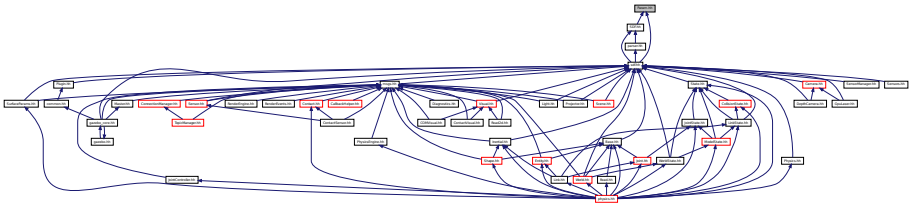
11.97 Param.hh File Reference

```
#include <boost/lexical_cast.hpp>
#include <boost/bind.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/any.hpp>
#include <boost/function.hpp>
#include <typeinfo>
#include <string>
#include <vector>
#include "common/Console.hh"
#include "common/Color.hh"
#include "common/Time.hh"
#include "math/Vector3.hh"
#include "math/Vector2i.hh"
#include "math/Vector2d.hh"
#include "math/Pose.hh"
#include "math/Quaternion.hh"
```

Include dependency graph for Param.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Param**

  A *(p. 111) parameter class.*

- class **sdf::ParamT< T >**

  Templatized parameter class.

Namespaces

- namespace **sdf**

namespace for Simulation Description Format parser

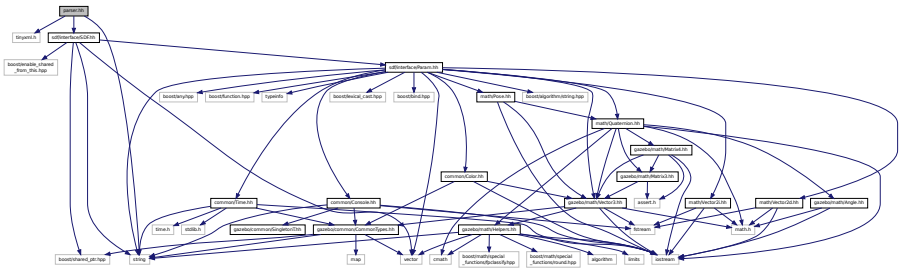
Typedefs

- typedef std::vector< ParamPtr > **sdf::Param_V**
- typedef Param * **sdf::ParamPtr**

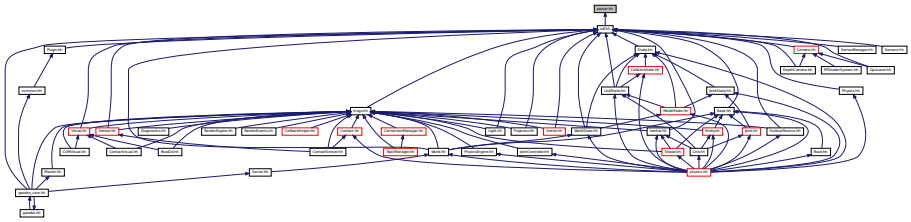
11.98 parser.hh File Reference

```
#include <tinyxml.h>
#include <string>
#include "sdf/interface/SDF.hh"
```

Include dependency graph for parser.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **sdf**

  namespace for Simulation Description Format parser

Functions

- void **sdf::addNestedModel** (ElementPtr _sdf, ElementPtr _includeSDF)
- void **sdf::copyChildren** (ElementPtr _sdf, TiXmlElement *_xml)
- bool **sdf::init** (SDFPtr _sdf)

  Init based on the installed sdf_format.xml file.
- bool **sdf::initDoc** (TiXmlDocument *_xmlDoc, SDFPtr _sdf)

- bool **sdf::initDoc** (TiXmlDocument ∗_xmlDoc, ElementPtr _sdf)
- bool **sdf::initFile** (const std::string &_filename, SDFPtr _sdf)
- bool **sdf::initFile** (const std::string &_filename, ElementPtr _sdf)
- bool **sdf::initString** (const std::string &_xmlString, SDFPtr _sdf)
- bool **sdf::initXml** (TiXmlElement ∗_xml, ElementPtr _sdf)
- bool **sdf::readDoc** (TiXmlDocument ∗_xmlDoc, SDFPtr _sdf, const std::string &_source)

  *Populate the **SDF** (*p. *695)* *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)

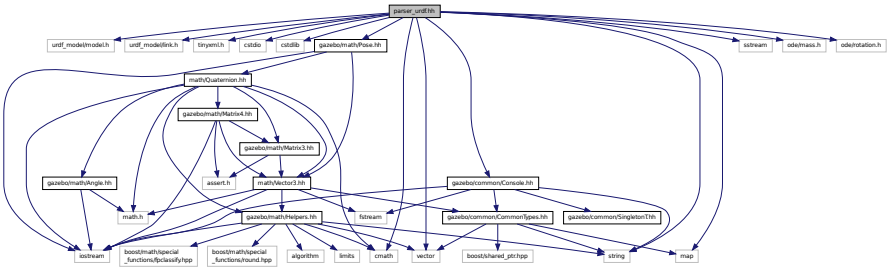
  *Populate the **SDF** (*p. *695)* *values from a file.*
- bool **sdf::readString** (const std::string &_xmlString, SDFPtr _sdf)

  *Populate the **SDF** (*p. *695)* *values from a string.*
- bool **sdf::readString** (const std::string &_xmlString, ElementPtr _sdf)
- bool **sdf::readXml** (TiXmlElement ∗_xml, ElementPtr _sdf)

11.99 parser_urdf.hh File Reference

```
#include <urdf_model/model.h>
#include <urdf_model/link.h>
#include <tinyxml.h>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <string>
#include <sstream>
#include <map>
#include "ode/mass.h"
#include "ode/rotation.h"
#include "gazebo/math/Pose.hh"
#include "gazebo/common/Console.hh"
```

Include dependency graph for parser_urdf.hh:



Classes

- class **urdf2gazebo::GazeboExtension**
- class **urdf2gazebo::URDF2Gazebo**

Namespaces

- namespace **urdf2gazebo**

  namespace for URDF to SDF parser

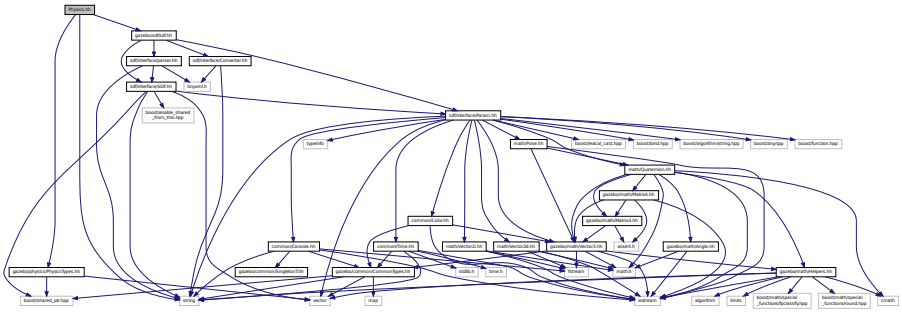
Typedefs

- typedef const urdf::Link * **urdf2gazebo::ConstUrdfLinkPtr**
- typedef urdf::Collision * **urdf2gazebo::UrdfCollisionPtr**
- typedef urdf::Link * **urdf2gazebo::UrdfLinkPtr**
- typedef urdf::Visual * **urdf2gazebo::UrdfVisualPtr**

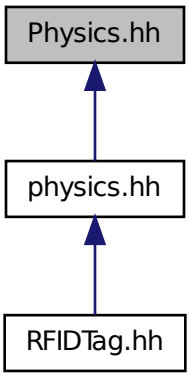
11.100 Physics.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/sdf/sdf.hh"
```

Include dependency graph for Physics.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Functions

- WorldPtr **gazebo::physics::create_world** (const std::string &_name="")

  Create a world given a name.
- bool **gazebo::physics::fini** ()

  *Finalize transport by calling **gazebo::transport::fini** (p. 77).*
- 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. 789)'s and call **gazebo::transport::init** (p. 78).*
- void **gazebo::physics::load_world** (WorldPtr _world, **sdf::ElementPtr** _sdf)

  *Load world from **sdf::Element** (p. 273) pointer.*
- void **gazebo::physics::load_worlds** (**sdf::ElementPtr** _sdf)

  *load multiple worlds from single **sdf::Element** (p. 273) pointer*
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)

  *Pause world by calling **World::SetPaused** (p. 920).*

- 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. 920) 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. 921) given a pointer to it.*

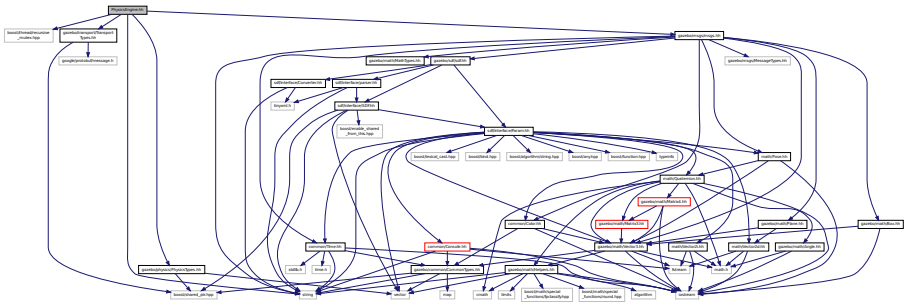
- void **gazebo::physics::stop_worlds** ()

  stop multiple worlds stored in static variable gazebo::g_worlds

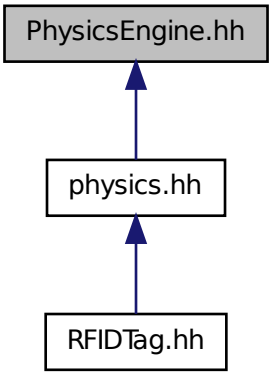
11.101 PhysicsEngine.hh File Reference

```
#include <boost/thread/recursive_mutex.hpp>
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsEngine.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::PhysicsEngine**

  Base *(*p. *137) class for a physics engine.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

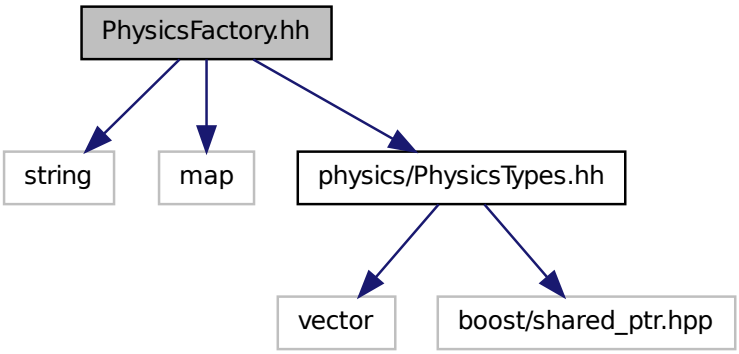
- namespace **gazebo::physics**

  namespace for physics

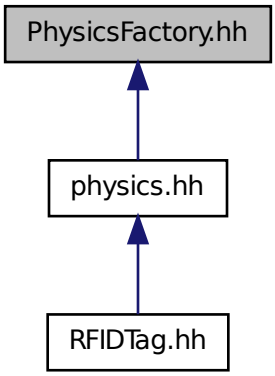
11.102 PhysicsFactory.hh File Reference

```
#include <string>
#include <map>
#include "physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsFactory.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::PhysicsFactory**

  The physics factory instantiates different physics engines.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

  namespace for physics

Macros

- #define **GZ_REGISTER_PHYSICS_ENGINE**(name, classname)

  Static physics registration macro.

Typedefs

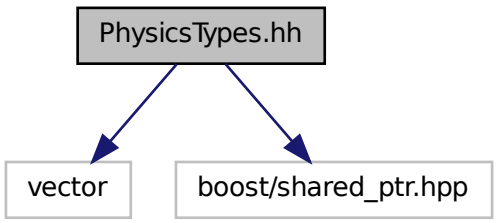
- typedef PhysicsEnginePtr(∗ **gazebo::physics::PhysicsFactoryFn**)(WorldPtr world)

11.103 PhysicsTypes.hh File Reference

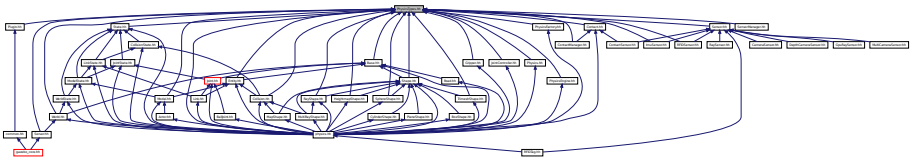
default namespace for gazebo

```
#include <vector>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for PhysicsTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

  namespace for physics

Macros

- #define **GZ_ALL_COLLIDE** 0x0FFFFFFF

  Default collision bitmask.
- #define **GZ_FIXED_COLLIDE** 0x00000001

  Collision object will collide only with fixed objects.
- #define **GZ_GHOST_COLLIDE** 0x10000000

  Collides with everything else but other ghost.
- #define **GZ_NONE_COLLIDE** 0x00000000

  Collision object will collide with nothing.
- #define **GZ_SENSOR_COLLIDE** 0x00000002

  Collision object will collide only with sensors.

Typedefs

- typedef std::vector< ActorPtr > **gazebo::physics::Actor_V**
- typedef Actor * **gazebo::physics::ActorPtr**
- typedef std::vector< BasePtr > **gazebo::physics::Base_V**
- typedef Base * **gazebo::physics::BasePtr**
- typedef BoxShape * **gazebo::physics::BoxShapePtr**
- typedef std::vector< CollisionPtr > **gazebo::physics::Collision_V**
- typedef Collision * **gazebo::physics::CollisionPtr**
- typedef Contact * **gazebo::physics::ContactPtr**
- typedef CylinderShape * **gazebo::physics::CylinderShapePtr**
- typedef Entity * **gazebo::physics::EntityPtr**
- typedef HeightmapShape * **gazebo::physics::HeightmapShapePtr**
- typedef Inertial * **gazebo::physics::InertialPtr**
- typedef std::vector< JointPtr > **gazebo::physics::Joint_V**
- typedef std::vector
  < JointControllerPtr > **gazebo::physics::JointController_V**
- typedef JointController * **gazebo::physics::JointControllerPtr**
- typedef Joint * **gazebo::physics::JointPtr**
- typedef std::vector< LinkPtr > **gazebo::physics::Link_V**
- typedef Link * **gazebo::physics::LinkPtr**
- typedef MeshShape * **gazebo::physics::MeshShapePtr**
- typedef std::vector< ModelPtr > **gazebo::physics::Model_V**
- typedef Model * **gazebo::physics::ModelPtr**
- typedef MultiRayShape * **gazebo::physics::MultiRayShapePtr**
- typedef PhysicsEngine * **gazebo::physics::PhysicsEnginePtr**
- typedef RayShape * **gazebo::physics::RayShapePtr**
- typedef **Road** * **gazebo::physics::RoadPtr**
- typedef Shape * **gazebo::physics::ShapePtr**
- typedef SphereShape * **gazebo::physics::SphereShapePtr**
- typedef SurfaceParams * **gazebo::physics::SurfaceParamsPtr**
- typedef World * **gazebo::physics::WorldPtr**

11.103.1 Detailed Description

default namespace for gazebo

11.103.2 Macro Definition Documentation

11.103.2.1 #define GZ_ALL_COLLIDE 0x0FFFFFFF

Default collision bitmask.

Collision objects will collide with everything.

11.103.2.2 #define GZ_FIXED_COLLIDE 0x00000001

Collision object will collide only with fixed objects.

11.103.2.3 #define GZ_GHOST_COLLIDE 0x10000000

Collides with everything else but other ghost.

11.103.2.4 #define GZ_NONE_COLLIDE 0x00000000

Collision object will collide with nothing.

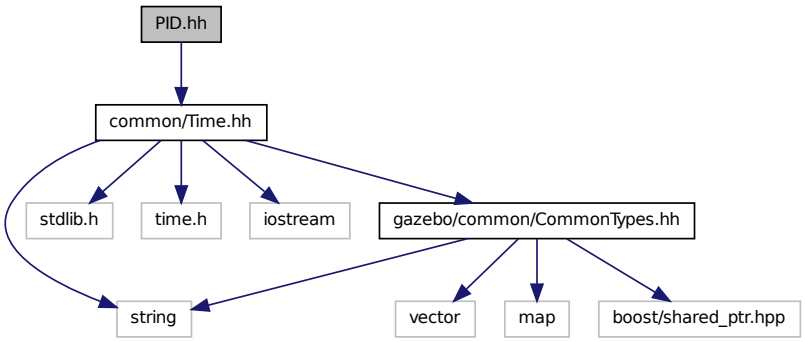
11.103.2.5 #define GZ_SENSOR_COLLIDE 0x00000002

Collision object will collide only with sensors.

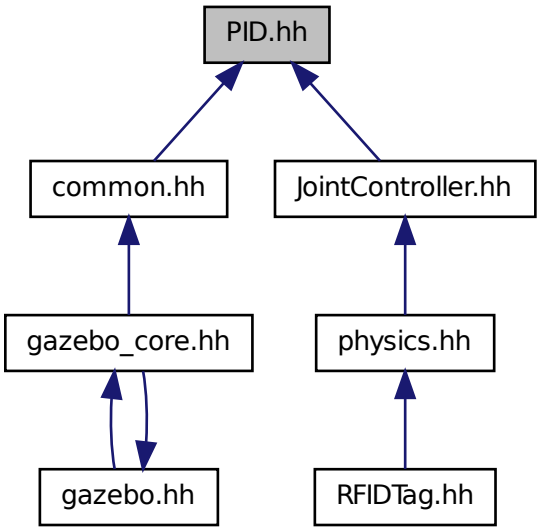
11.104 PID.hh File Reference

```
#include "common/Time.hh"
```

Include dependency graph for PID.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::PID**

  *Generic **PID** (p. 583) controller class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

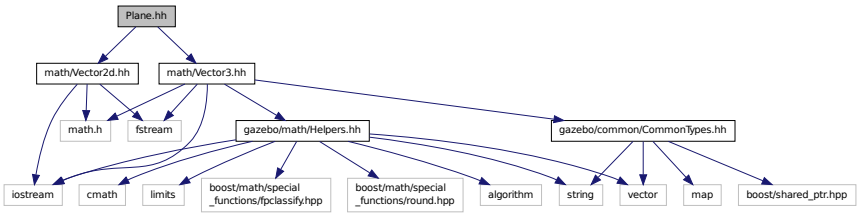
- namespace **gazebo::common**

  Common namespace.

11.105 Plane.hh File Reference

```
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
```

Include dependency graph for Plane.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::math::Plane**

  A *(*p. *111) plane and related functions.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

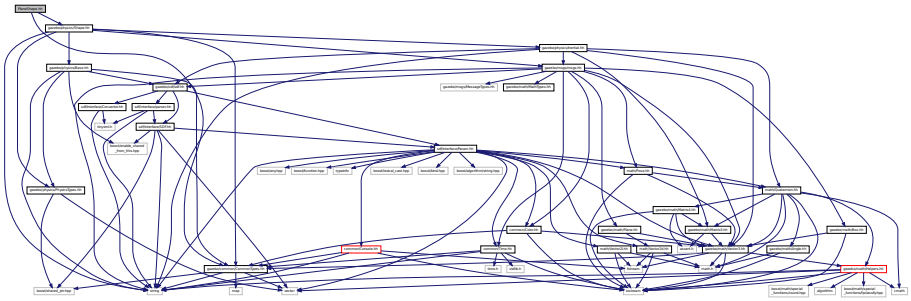
- namespace **gazebo::math**

  Math namespace.

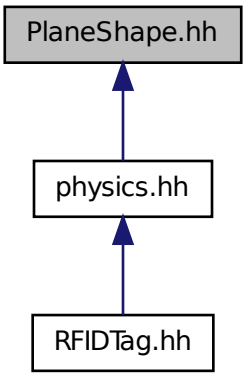
11.106 PlaneShape.hh File Reference

```
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for PlaneShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::PlaneShape**

  Collision *(*p. *195) for an infinite plane.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

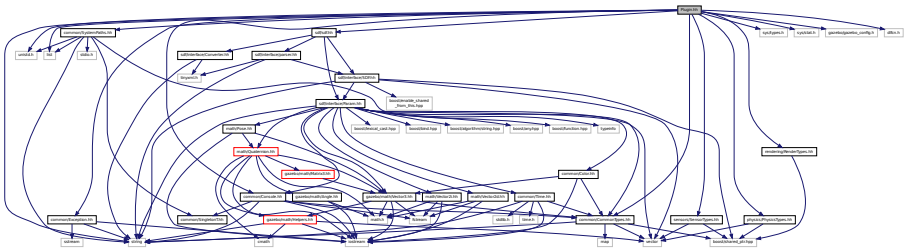
- namespace **gazebo::physics**

  namespace for physics

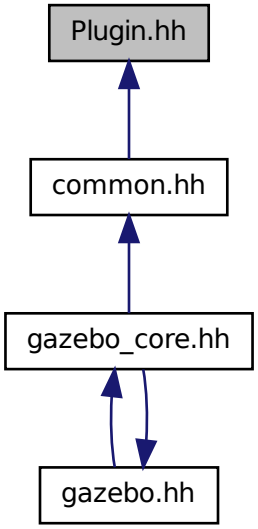
11.107 Plugin.hh File Reference

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <gazebo/gazebo_config.h>
#include <dlfcn.h>
#include <list>
#include <string>
#include "common/CommonTypes.hh"
#include "common/SystemPaths.hh"
#include "common/Console.hh"
#include "common/Exception.hh"
#include "physics/PhysicsTypes.hh"
#include "sensors/SensorTypes.hh"
#include "sdf/sdf.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for common/Plugin.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::ModelPlugin**

  A (p. *111*) *plugin with access to* ***physics::Model*** (p. *489*).
- class **gazebo::PluginT< T >**

  A (p. *111*) *class which all plugins must inherit from.*
- class **gazebo::SensorPlugin**

  A (p. *111*) *plugin with access to physics::Sensor.*
- class **gazebo::SystemPlugin**

  A (p. *111*) *plugin loaded within the gzserver on startup.*
- class **gazebo::VisualPlugin**

  A (p. *111*) *plugin loaded within the gzserver on startup.*
- class **gazebo::WorldPlugin**

  A (p. *111*) *plugin with access to* ***physics::World*** (p. *910*).

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

Macros

- #define **GZ_REGISTER_MODEL_PLUGIN**(classname)

  Plugin registration function for model plugin.
- #define **GZ_REGISTER_SENSOR_PLUGIN**(classname)

  Plugin registration function for sensors.
- #define **GZ_REGISTER_SYSTEM_PLUGIN**(classname)

  Plugin registration function for system plugin.
- #define **GZ_REGISTER_VISUAL_PLUGIN**(classname)

  Plugin registration function for visual plugin.
- #define **GZ_REGISTER_WORLD_PLUGIN**(classname)

  Plugin registration function for world plugin.

Enumerations

- enum **gazebo::PluginType** {
  gazebo::WORLD_PLUGIN, **gazebo::MODEL_PLUGIN**, **gazebo::SENSOR_PLUGIN**, **gazebo::SYSTEM_PLUGIN**,
  gazebo::VISUAL_PLUGIN }

  Used to specify the type of plugin.

11.107.1 Macro Definition Documentation

11.107.1.1 #define GZ_REGISTER_MODEL_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::ModelPlugin *RegisterPlugin();    gazebo::ModelPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for model plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.107.1.2 #define GZ_REGISTER_SENSOR_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::SensorPlugin *RegisterPlugin();    gazebo::SensorPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for sensors.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.107.1.3 #define GZ_REGISTER_SYSTEM_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::SystemPlugin *RegisterPlugin();    gazebo::SystemPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for system plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.107.1.4 #define GZ_REGISTER_VISUAL_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::VisualPlugin *RegisterPlugin();    gazebo::VisualPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for visual plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.107.1.5 #define GZ_REGISTER_WORLD_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::WorldPlugin *RegisterPlugin();    gazebo::WorldPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for world plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

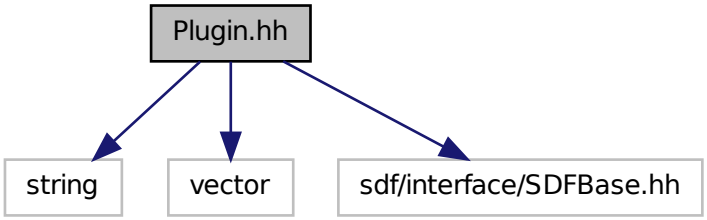
Returns

the name of the registered plugin

11.108 Plugin.hh File Reference

```
#include <string>
#include <vector>
#include "sdf/interface/SDFBase.hh"
```

Include dependency graph for sdf/interface/Plugin.hh:



Classes

- class **sdf::Plugin**

Namespaces

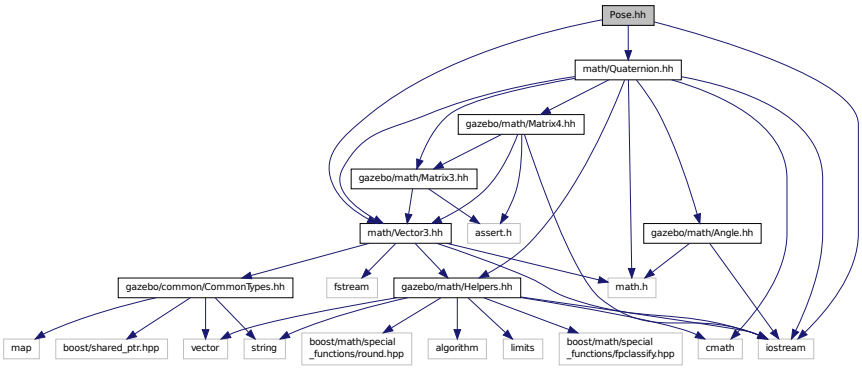
- namespace **sdf**

  namespace for Simulation Description Format parser

11.109 Pose.hh File Reference

```
#include <iostream>
#include "math/Vector3.hh"
#include "math/Quaternion.hh"
```

Include dependency graph for Pose.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::math::Pose**

  Encapsulates a position and rotation in three space.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

11.110 Projector.hh File Reference

```
#include <string>
#include <map>
#include <list>
#include "rendering/ogre_gazebo.h"
#include "msgs/msgs.hh"
#include "sdf/sdf.hh"
#include "transport/transport.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for Projector.hh:

Classes

- class **gazebo::rendering::Projector**

  Projects a material onto surface, light a light projector.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

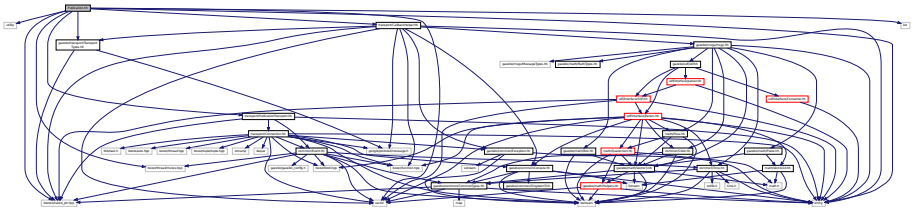
- namespace **gazebo::rendering**

  Rendering namespace.

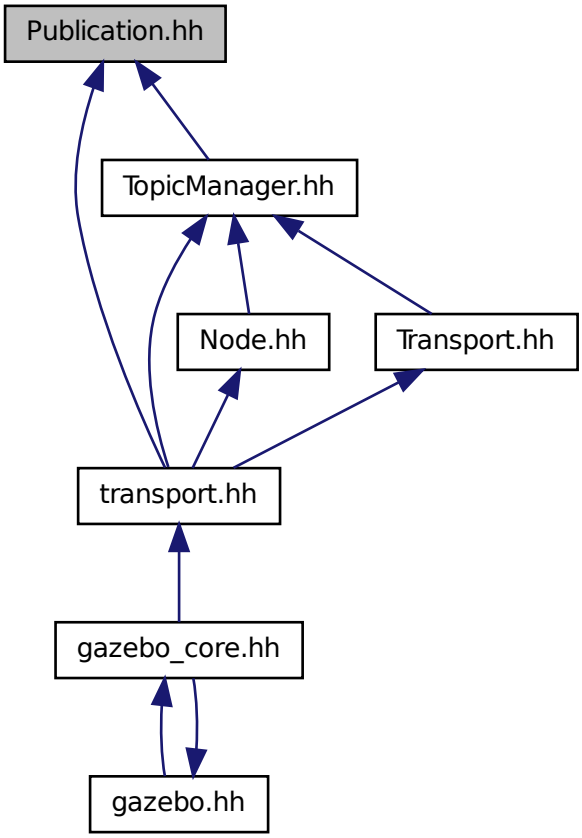
11.111 Publication.hh File Reference

```
#include <utility>
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <list>
#include <string>
#include <vector>
#include "transport/CallbackHelper.hh"
#include "transport/TransportTypes.hh"
#include "transport/PublicationTransport.hh"
```

Include dependency graph for Publication.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Publication**

  A (p. *111*) *publication for a topic.*

Namespaces

- namespace **gazebo**

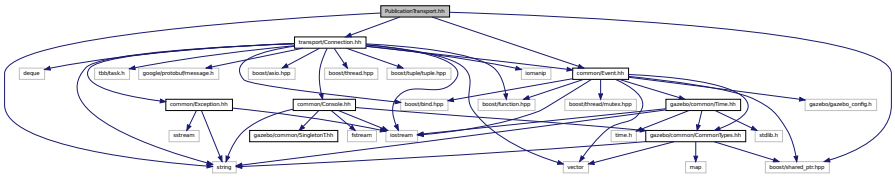
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.112 PublicationTransport.hh File Reference

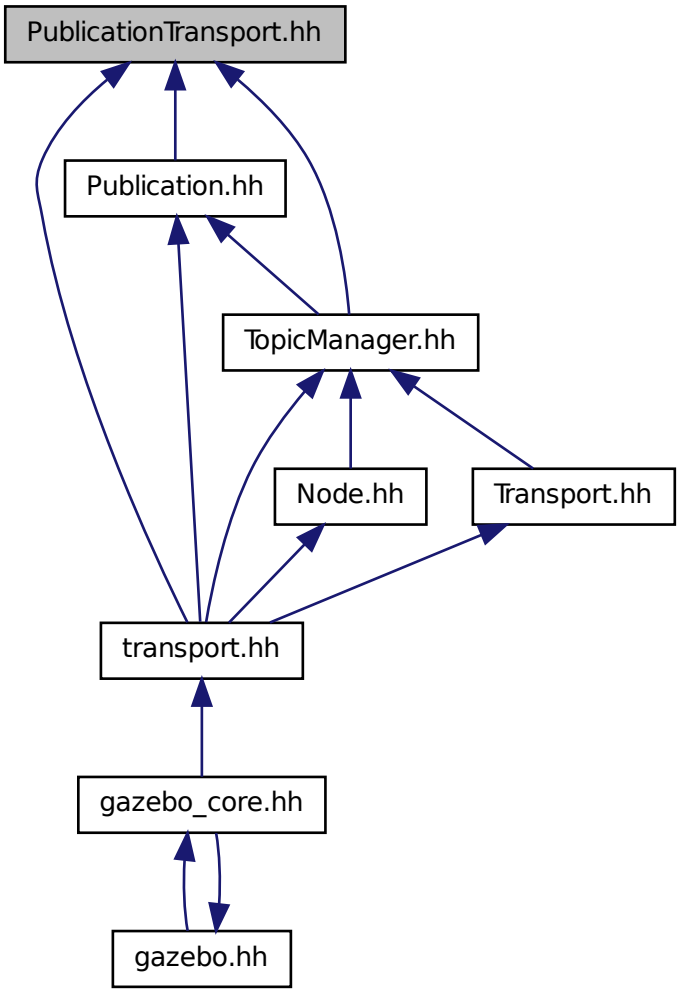
```
#include <boost/shared_ptr.hpp>
```

```
#include <string>
#include "transport/Connection.hh"
#include "common/Event.hh"
```

Include dependency graph for PublicationTransport.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::PublicationTransport**

  transport/transport.hh

Namespaces

- namespace **gazebo**

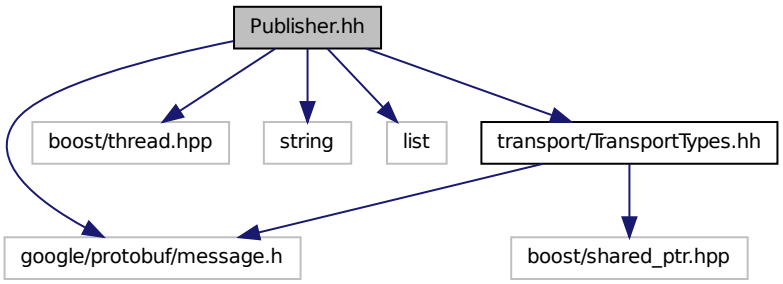
  Forward declarations for the common classes.

- namespace **gazebo::transport**

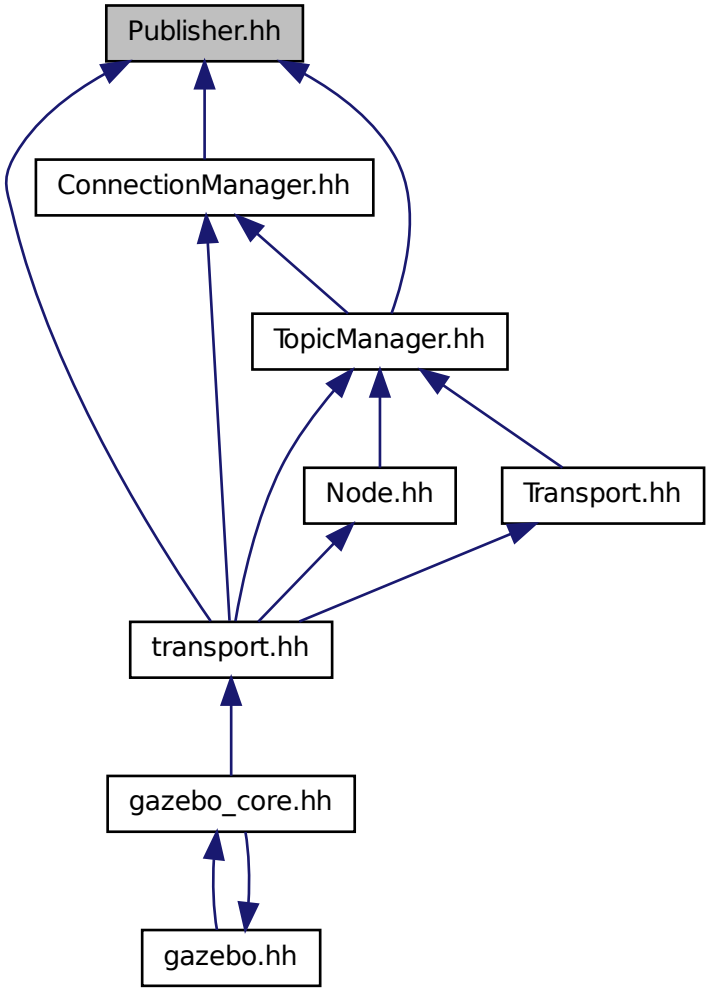
11.113 Publisher.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/thread.hpp>
#include <string>
#include <list>
#include "transport/TransportTypes.hh"
```

Include dependency graph for Publisher.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Publisher**

  A *(*p. *111) publisher of messages on a topic.*

Namespaces

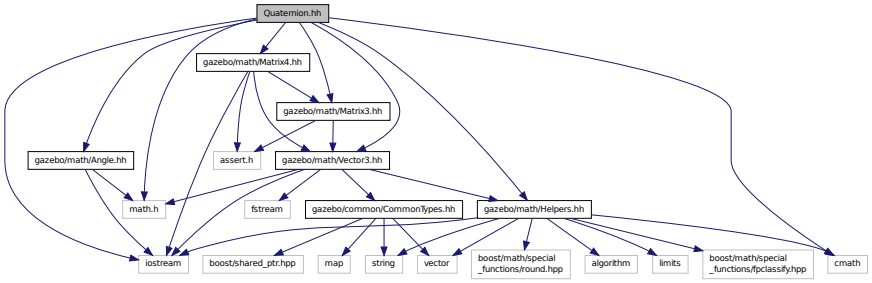
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::transport**

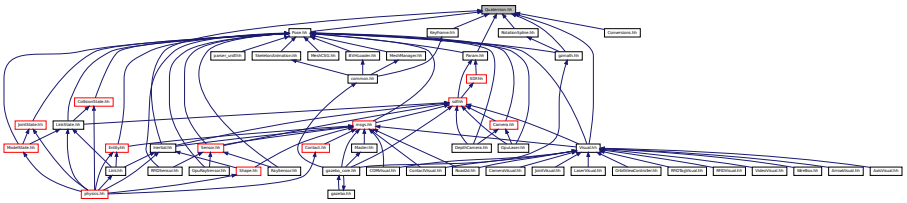
11.114 Quaternion.hh File Reference

```
#include <math.h>
#include <iostream>
#include <cmath>
#include "gazebo/math/Helpers.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Matrix3.hh"
#include "gazebo/math/Matrix4.hh"
```

Include dependency graph for Quaternion.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Quaternion**

  A *(p. 111) quaternion class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

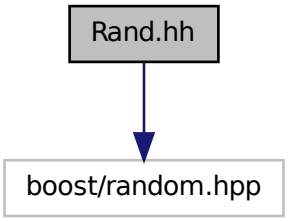
- namespace **gazebo::math**

  Math namespace.

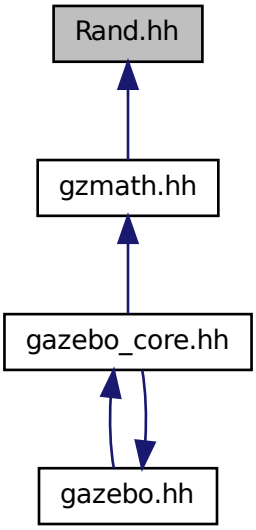
11.115 Rand.hh File Reference

```
#include <boost/random.hpp>
```

Include dependency graph for Rand.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Rand**

  Random number generator class.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

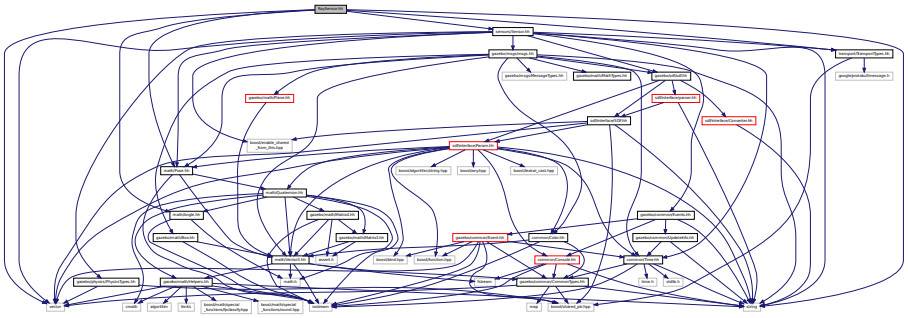
Typedefs

- typedef boost::mt19937 **gazebo::math::GeneratorType**
- typedef boost::normal_distribution < double > **gazebo::math::NormalRealDist**
- typedef boost::variate_generator < GeneratorType &, NormalRealDist > **gazebo::math::NRealGen**
- typedef boost::variate_generator < GeneratorType &, UniformIntDist > **gazebo::math::UIntGen**
- typedef boost::uniform_int< int > **gazebo::math::UniformIntDist**
- typedef boost::uniform_real < double > **gazebo::math::UniformRealDist**
- typedef boost::variate_generator < GeneratorType &, UniformRealDist > **gazebo::math::URealGen**

11.116 RaySensor.hh File Reference

```
#include <vector>
#include <string>
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "transport/TransportTypes.hh"
#include "sensors/Sensor.hh"
```

Include dependency graph for RaySensor.hh:



Classes

- class **gazebo::sensors::RaySensor**

  Sensor (p. *698) with one or more rays.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

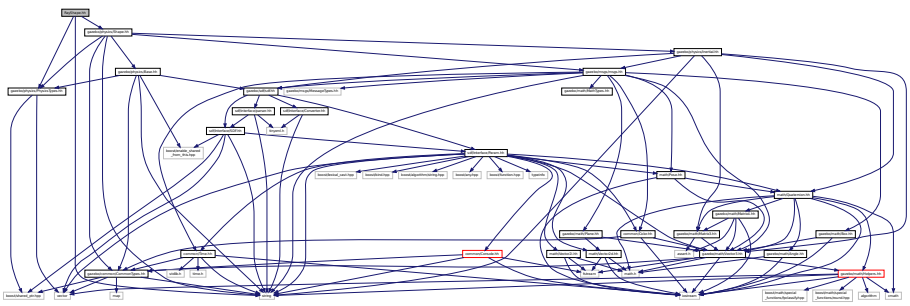
- namespace **gazebo::sensors**

  Sensors namespace.

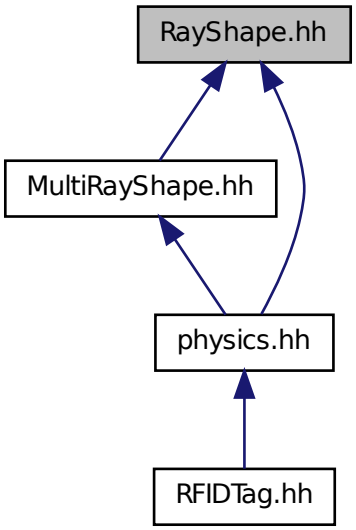
11.117 RayShape.hh File Reference

```
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for RayShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::RayShape**

  Base (p. *137*) *class for Ray collision geometry.*

Namespaces

- namespace **gazebo**

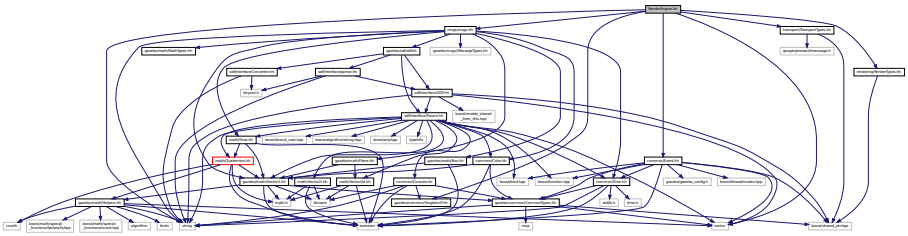
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.118 RenderEngine.hh File Reference

```
#include <vector>
#include <string>
#include "msgs/msgs.hh"
#include "common/SingletonT.hh"
#include "common/Event.hh"
#include "transport/TransportTypes.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for RenderEngine.hh:



Classes

- class **gazebo::rendering::RenderEngine**

  Adaptor to Ogre3d.

Namespaces

- namespace **gazebo**

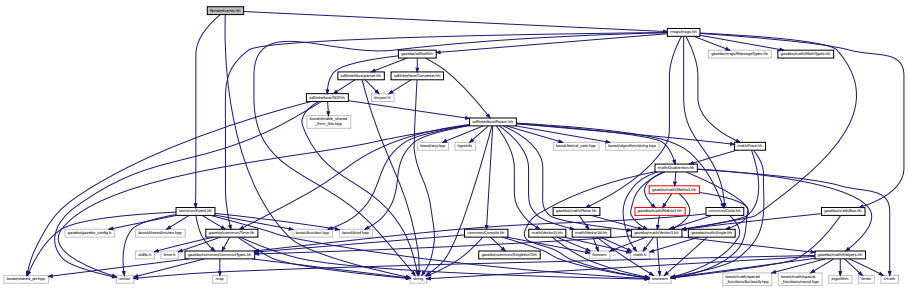
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.119 RenderEvents.hh File Reference

```
#include <string>
#include "common/Event.hh"
#include "msgs/msgs.hh"
```

Include dependency graph for RenderEvents.hh:



Classes

- class **gazebo::rendering::Events**

  Base class for rendering events.

Namespaces

- namespace **gazebo**

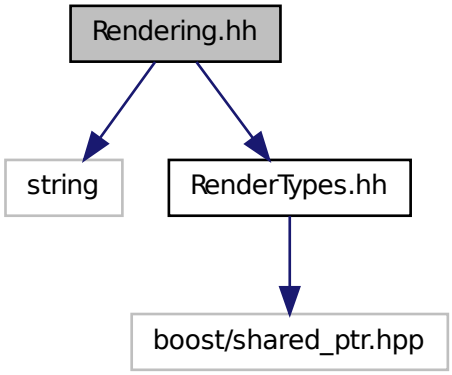
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.120 Rendering.hh File Reference

```
#include <string>
#include "RenderTypes.hh"
```

Include dependency graph for Rendering.hh:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

Functions

- rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations)

  *create **rendering::Scene*** (p. *676*) *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. *676*) *by name.*

- bool **gazebo::rendering::init** ()

  init rendering engine.
- bool **gazebo::rendering::load** ()

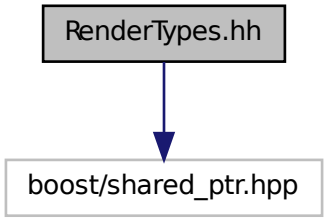
  load rendering engine.
- void **gazebo::rendering::remove_scene** (const std::string &_name)

  *remove a **rendering::Scene*** (p. *676*) *by name*

11.121 RenderTypes.hh File Reference

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for RenderTypes.hh:



This graph shows which files directly or indirectly include this file:

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

Macros

- #define **GZ_VISIBILITY_ALL** 0x0FFFFFFF

  Render everything visibility mask.
- #define **GZ_VISIBILITY_GUI** 0x00000001

  Render GUI visuals mask.

- #define **GZ_VISIBILITY_NOT_SELECTABLE** 0x00000002

  Render visuals that are not selectable mask.
- #define **GZ_VISIBILITY_SELECTION** 0x10000000

  Renders only objects that can be selected.

Typedefs

- typedef ArrowVisual ∗ **gazebo::rendering::ArrowVisualPtr**
- typedef AxisVisual ∗ **gazebo::rendering::AxisVisualPtr**
- typedef Camera ∗ **gazebo::rendering::CameraPtr**
- typedef CameraVisual ∗ **gazebo::rendering::CameraVisualPtr**
- typedef COMVisual ∗ **gazebo::rendering::COMVisualPtr**
- typedef ContactVisual ∗ **gazebo::rendering::ContactVisualPtr**
- typedef DepthCamera ∗ **gazebo::rendering::DepthCameraPtr**
- typedef DynamicLines ∗ **gazebo::rendering::DynamicLinesPtr**
- typedef GpuLaser ∗ **gazebo::rendering::GpuLaserPtr**
- typedef JointVisual ∗ **gazebo::rendering::JointVisualPtr**
- typedef LaserVisual ∗ **gazebo::rendering::LaserVisualPtr**
- typedef Light ∗ **gazebo::rendering::LightPtr**
- typedef RFIDTagVisual ∗ **gazebo::rendering::RFIDTagVisualPtr**
- typedef RFIDVisual ∗ **gazebo::rendering::RFIDVisualPtr**
- typedef Scene ∗ **gazebo::rendering::ScenePtr**
- typedef UserCamera ∗ **gazebo::rendering::UserCameraPtr**
- typedef Visual ∗ **gazebo::rendering::VisualPtr**

Enumerations

- enum **gazebo::rendering::RenderOpType** {
  gazebo::rendering::RENDERING_POINT_LIST = 0, **gazebo::rendering::RENDERING_LINE_LIST** = 1, **gazebo::rendering::RENDERING_LINE_STRIP** = 2, **gazebo::rendering::RENDERING_TRIANGLE_LIST** = 3,
  gazebo::rendering::RENDERING_TRIANGLE_STRIP = 4, **gazebo::rendering::RENDERING_TRIANGLE_FAN** = 5, **gazebo::rendering::RENDERING_MESH_RESOURCE** = 6 }

  Type of render operation for a drawable.

11.121.1 Macro Definition Documentation

11.121.1.1 #define GZ_VISIBILITY_ALL 0x0FFFFFFF

Render everything visibility mask.

11.121.1.2 #define GZ_VISIBILITY_GUI 0x00000001

Render GUI visuals mask.

11.121.1.3 #define GZ_VISIBILITY_NOT_SELECTABLE 0x00000002

Render visuals that are not selectable mask.

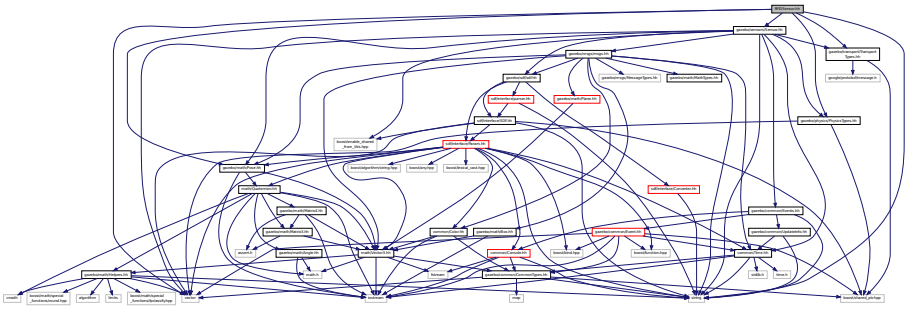
11.121.1.4 #define GZ_VISIBILITY_SELECTION 0x10000000

Renders only objects that can be selected.

11.122 RFIDSensor.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/sensors/Sensor.hh"
```

Include dependency graph for RFIDSensor.hh:



Classes

- class **gazebo::sensors::RFIDSensor**

  Sensor (p. *698*) *class for RFID type of sensor.*

Namespaces

- namespace **gazebo**

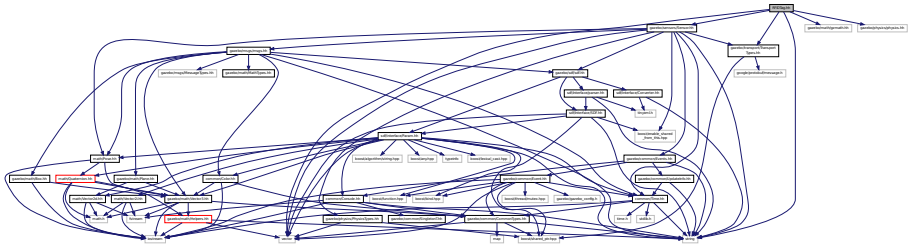
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.123 RFIDTag.hh File Reference

```
#include <vector>
#include <string>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/sensors/Sensor.hh"
#include "gazebo/math/gzmath.hh"
#include "gazebo/physics/physics.hh"
```

Include dependency graph for RFIDTag.hh:



Classes

- class **gazebo::sensors::RFIDTag**

  RFIDTag (p. *660) to interact with RFIDTagSensors.*

Namespaces

- namespace **gazebo**

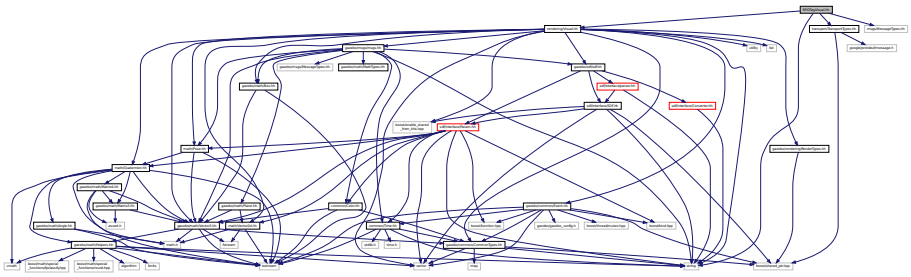
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.124 RFIDTagVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for RFIDTagVisual.hh:



Classes

- class **gazebo::rendering::RFIDTagVisual**

  Visualization for RFID tags sensor.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

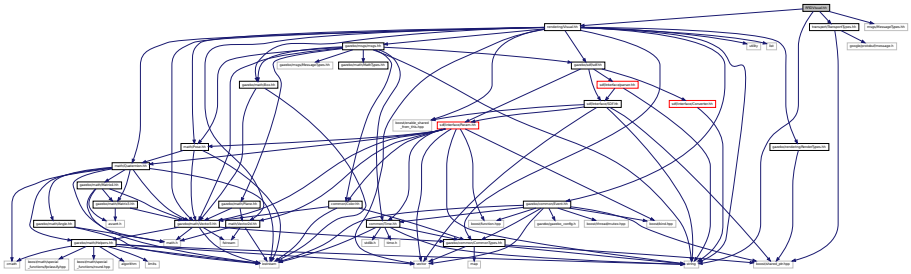
- namespace **gazebo::rendering**

  Rendering namespace.

11.125 RFIDVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for RFIDVisual.hh:



Classes

- class **gazebo::rendering::RFIDVisual**

  Visualization for RFID sensor.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

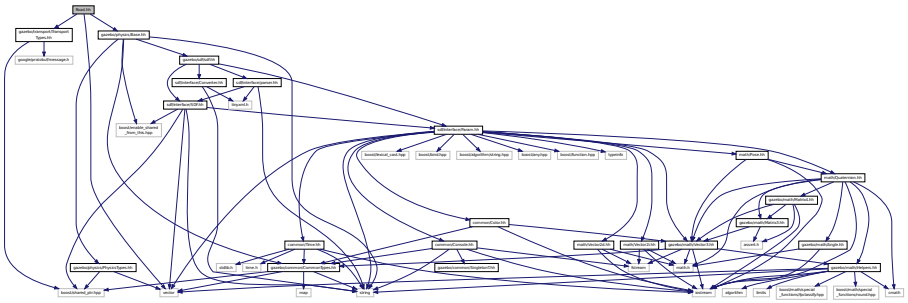
- namespace **gazebo::rendering**

  Rendering namespace.

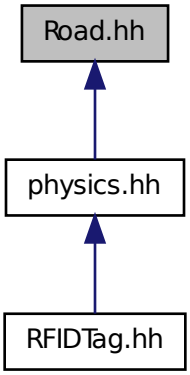
11.126 Road.hh File Reference

```
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/Base.hh"
```

Include dependency graph for Road.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Road**

  *for building a **Road** (p. 665) from SDF*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.127 Road2d.hh File Reference

```
#include <string>
#include <vector>
#include <list>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Spline.hh"
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for Road2d.hh:

Classes

- class **gazebo::rendering::Road2d**

Namespaces

- namespace **gazebo**

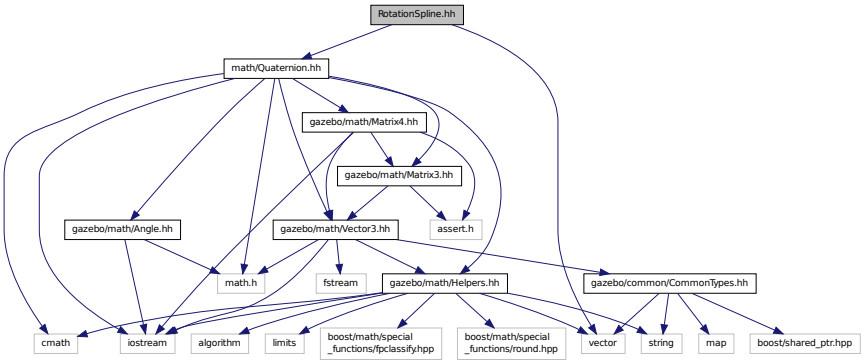
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

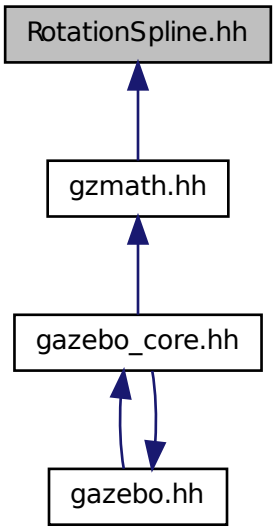
11.128 RotationSpline.hh File Reference

```
#include <vector>
#include "math/Quaternion.hh"
```

Include dependency graph for RotationSpline.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::RotationSpline**

  Spline *(p. 754) for rotations.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

11.129 RTShaderSystem.hh File Reference

```
#include <list>
#include <string>
#include <vector>
#include "rendering/ogre_gazebo.h"
#include "gazebo_config.h"
#include "rendering/Camera.hh"
#include "common/SingletonT.hh"
```

Include dependency graph for RTShaderSystem.hh:

Classes

- class **gazebo::rendering::RTShaderSystem**

  *Implements **Ogre** (p. 106)'s Run-Time Shader system.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

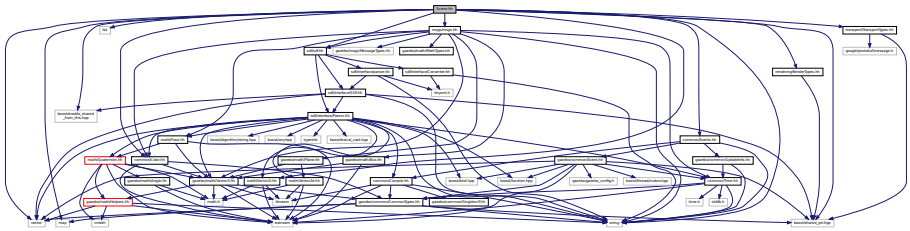
- namespace **gazebo::rendering**

  Rendering namespace.

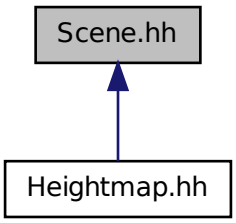
11.130 Scene.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include <list>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include "sdf/sdf.hh"
#include "msgs/msgs.hh"
#include "rendering/RenderTypes.hh"
#include "transport/TransportTypes.hh"
#include "common/Events.hh"
#include "common/Color.hh"
#include "math/Vector2i.hh"
```

Include dependency graph for Scene.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Scene**

  Representation of an entire scene graph.

Namespaces

- namespace **boost**
- namespace **gazebo**

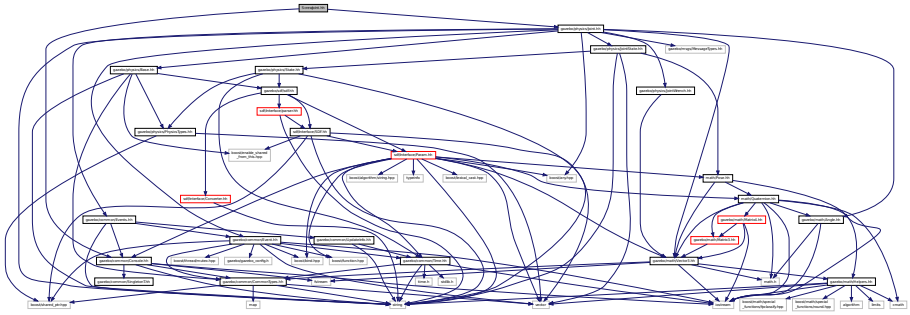
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**
- namespace **SkyX**

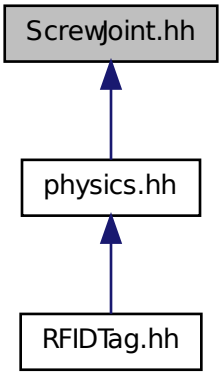
11.131 ScrewJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
#include "gazebo/common/Console.hh"
```

Include dependency graph for ScrewJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ScrewJoint**< **T** >

  A (p. *111) screw joint, which has both prismatic and rotational DOFs.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

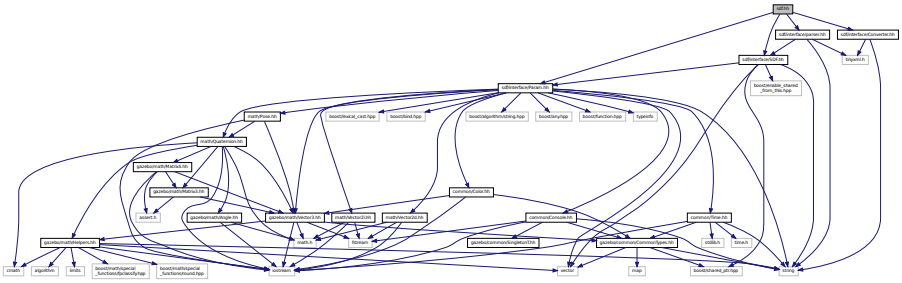
- namespace **gazebo::physics**

  namespace for physics

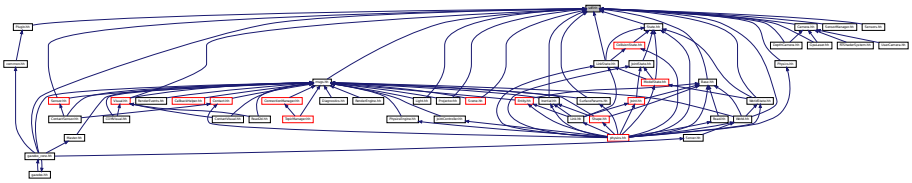
11.132 sdf.hh File Reference

```
#include "sdf/interface/SDF.hh"
#include "sdf/interface/Param.hh"
#include "sdf/interface/parser.hh"
#include "sdf/interface/Converter.hh"
```

Include dependency graph for sdf.hh:



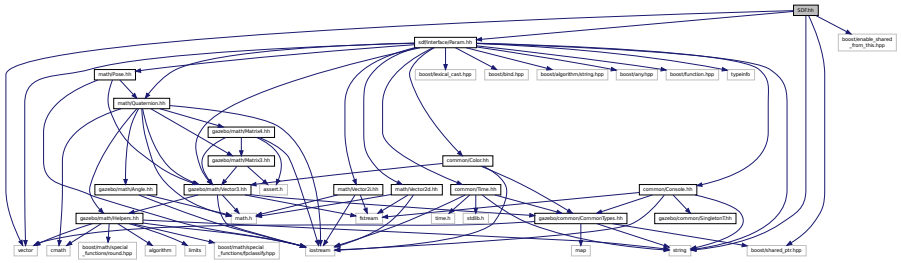
This graph shows which files directly or indirectly include this file:



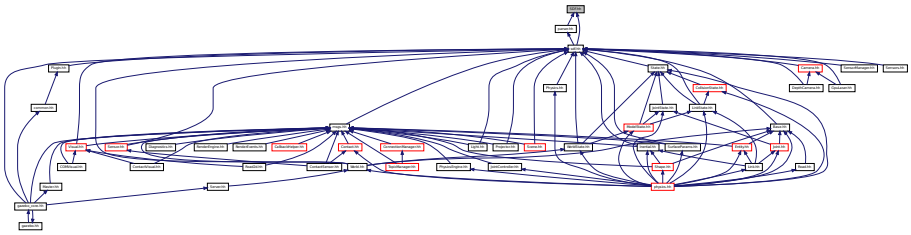
11.133 SDF.hh File Reference

```
#include <vector>
#include <string>
#include <boost/shared_ptr.hpp>
#include <boost/enable_shared_from_this.hpp>
#include "sdf/interface/Param.hh"
```

Include dependency graph for SDF.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Element**

  SDF *(p. 695)* ***Element*** *(p. 273) class.*
- class **sdf::SDF**

  Base ***SDF*** *(p. 695) class.*

Namespaces

- namespace **sdf**

  namespace for Simulation Description Format parser

Macros

- #define **SDF_VERSION** "1.4"

Typedefs

- typedef Element ∗ **sdf::ElementPtr**
- typedef std::vector< ElementPtr > **sdf::ElementPtr_V**
- typedef SDF ∗ **sdf::SDFPtr**

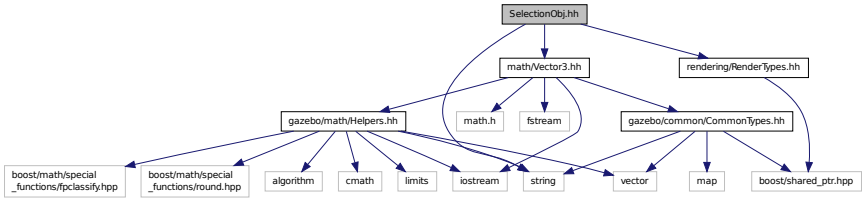
11.133.1 Macro Definition Documentation

11.133.1.1 #define SDF_VERSION "1.4"

11.134 SelectionObj.hh File Reference

```
#include <string>
#include "math/Vector3.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for SelectionObj.hh:



Classes

- class **gazebo::rendering::SelectionObj**

  A (p. *111) graphical selection object.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

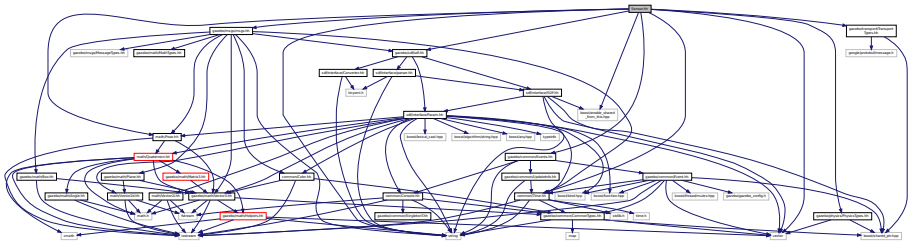
- namespace **gazebo::rendering**

  Rendering namespace.

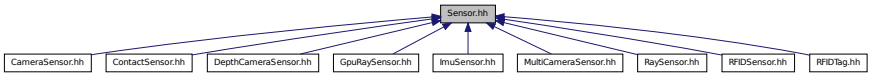
11.135 Sensor.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <vector>
#include <string>
#include "gazebo/sdf/sdf.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for Sensor.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::Sensor**

  Base class for sensors.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::sensors**

  Sensors namespace.

Enumerations

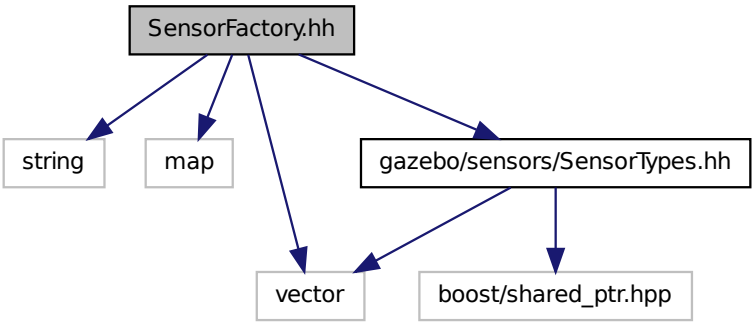
- enum **gazebo::sensors::SensorCategory** { **gazebo::sensors::IMAGE** = 0, **gazebo::sensors::RAY** = 1, **gazebo::sensors::OTHER** = 2, **gazebo::sensors::CATEGORY_COUNT** = 3 }

  SensorClass is used to categorize sensors.

11.136 SensorFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "gazebo/sensors/SensorTypes.hh"
```

Include dependency graph for SensorFactory.hh:



Classes

- class **gazebo::sensors::SensorFactory**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

Macros

- #define **GZ_REGISTER_STATIC_SENSOR**(name, classname)

  Static sensor registration macro.

Typedefs

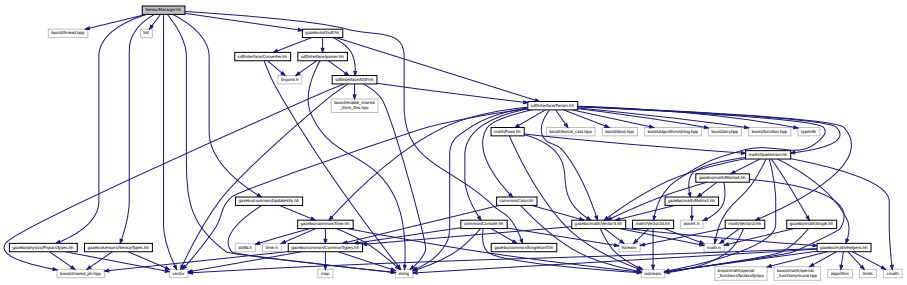
- typedef Sensor ∗(∗ **gazebo::sensors::SensorFactoryFn**)()

11.137 SensorManager.hh File Reference

```
#include <boost/thread.hpp>
```

```
#include <string>
#include <vector>
#include <list>
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/sensors/SensorTypes.hh"
#include "gazebo/sdf/sdf.hh"
```

Include dependency graph for SensorManager.hh:



Classes

- class **gazebo::sensors::SensorManager**

  Class to manage and update all sensors.

- class **gazebo::sensors::SimTimeEvent**

- class **gazebo::sensors::SimTimeEventHandler**

  Monitors simulation time, and notifies conditions when a specified time has been reached.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

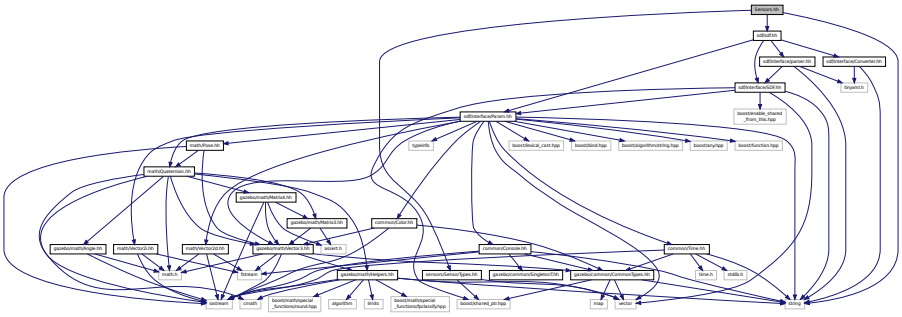
- namespace **gazebo::sensors**

  Sensors namespace.

11.138 Sensors.hh File Reference

```
#include <string>
#include "sdf/sdf.hh"
#include "sensors/SensorTypes.hh"
```

Include dependency graph for Sensors.hh:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

Functions

- std::string **gazebo::sensors::create_sensor** (**sdf::ElementPtr** _elem, const std::string &_worldName, const std::string &_parentName)

  Create a sensor using SDF.
- bool **gazebo::sensors::fini** ()

  shutdown the sensor generation loop.
- SensorPtr **gazebo::sensors::get_sensor** (const std::string &_name)

  Get a sensor using by name.
- bool **gazebo::sensors::init** ()

  initialize the sensor generation loop.
- bool **gazebo::sensors::load** ()

  Load the sensor library.
- void **gazebo::sensors::remove_sensor** (const std::string &_sensorName)

  Remove a sensor by name.
- bool **gazebo::sensors::remove_sensors** ()

  Remove all sensors.
- void **gazebo::sensors::run** () **GAZEBO_DEPRECATED**(1.5)

  Deprecated.
- void **gazebo::sensors::run_once** (bool _force=false)

  Run the sensor generation one step.
- void **gazebo::sensors::run_threads** ()

  Run sensors in a threads. This is a non-blocking call.
- void **gazebo::sensors::stop** ()

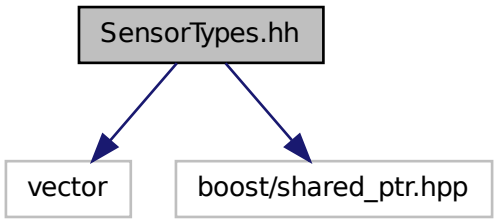
  Stop the sensor generation loop.

11.139 SensorTypes.hh File Reference

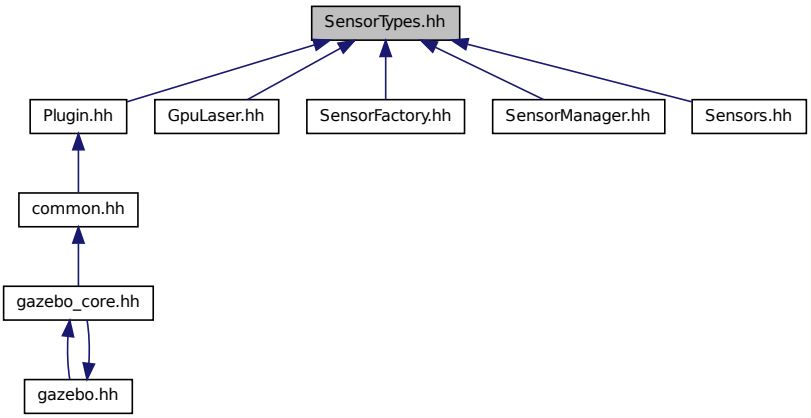
Forward declarations and typedefs for sensors.

```
#include <vector>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for SensorTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::sensors**

  Sensors namespace.

Typedefs

- typedef std::vector < CameraSensorPtr > **gazebo::sensors::CameraSensor_V**
- typedef CameraSensor * **gazebo::sensors::CameraSensorPtr**
- typedef std::vector < ContactSensorPtr > **gazebo::sensors::ContactSensor_V**
- typedef ContactSensor * **gazebo::sensors::ContactSensorPtr**
- typedef std::vector < DepthCameraSensorPtr > **gazebo::sensors::DepthCameraSensor_V**
- typedef DepthCameraSensor * **gazebo::sensors::DepthCameraSensorPtr**
- typedef std::vector < GpuRaySensorPtr > **gazebo::sensors::GpuRaySensor_V**
- typedef GpuRaySensor * **gazebo::sensors::GpuRaySensorPtr**
- typedef std::vector< ImuSensorPtr > **gazebo::sensors::ImuSensor_V**
- typedef ImuSensor * **gazebo::sensors::ImuSensorPtr**
- typedef std::vector< RaySensorPtr > **gazebo::sensors::RaySensor_V**
- typedef RaySensor * **gazebo::sensors::RaySensorPtr**
- typedef std::vector< RFIDSensor > **gazebo::sensors::RFIDSensor_V**
- typedef RFIDSensor * **gazebo::sensors::RFIDSensorPtr**
- typedef std::vector< RFIDTag > **gazebo::sensors::RFIDTag_V**
- typedef RFIDTag * **gazebo::sensors::RFIDTagPtr**
- typedef std::vector< SensorPtr > **gazebo::sensors::Sensor_V**
- typedef Sensor * **gazebo::sensors::SensorPtr**

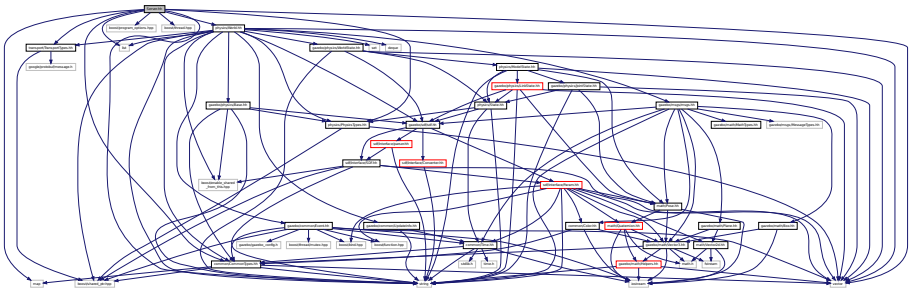
11.139.1 Detailed Description

Forward declarations and typedefs for sensors.

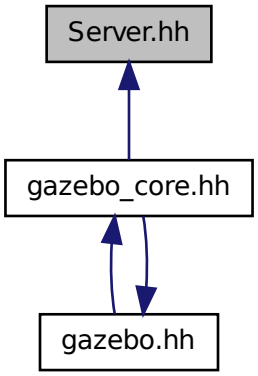
11.140 Server.hh File Reference

```
#include <string>
#include <vector>
#include <list>
#include <map>
#include <boost/program_options.hpp>
#include <boost/thread.hpp>
#include "transport/TransportTypes.hh"
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/World.hh"
```

Include dependency graph for Server.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Server**

Namespaces

- namespace **boost**
- namespace **gazebo**

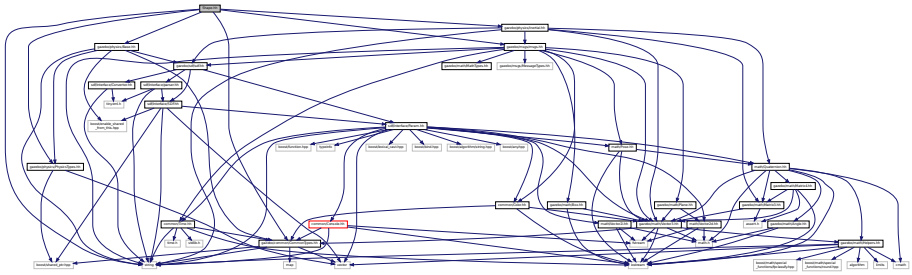
  Forward declarations for the common classes.

11.141 Shape.hh File Reference

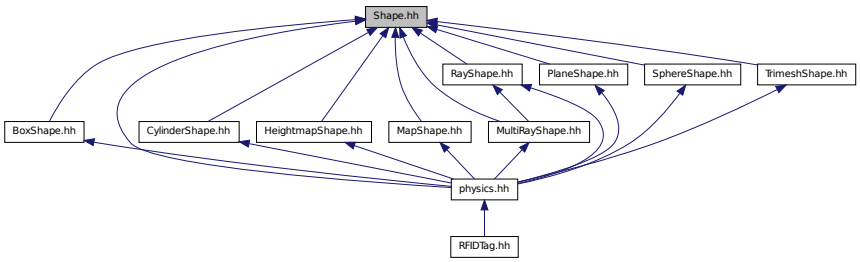
```
#include <string>
```

```
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Inertial.hh"
#include "gazebo/physics/Base.hh"
```

Include dependency graph for Shape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Shape**

  Base *(*p. *137) class for all shapes.*

Namespaces

- namespace **gazebo**

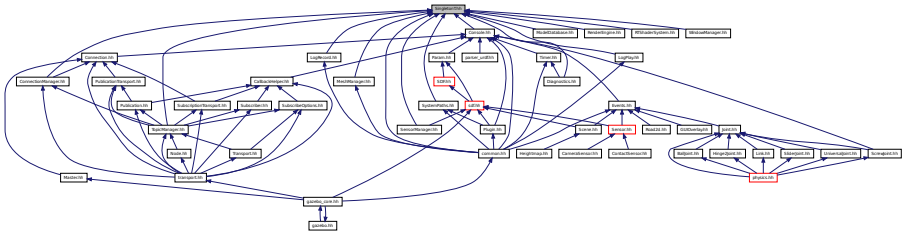
  Forward declarations for the common classes.

- namespace **gazebo::physics**

  namespace for physics

11.142 SingletonT.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

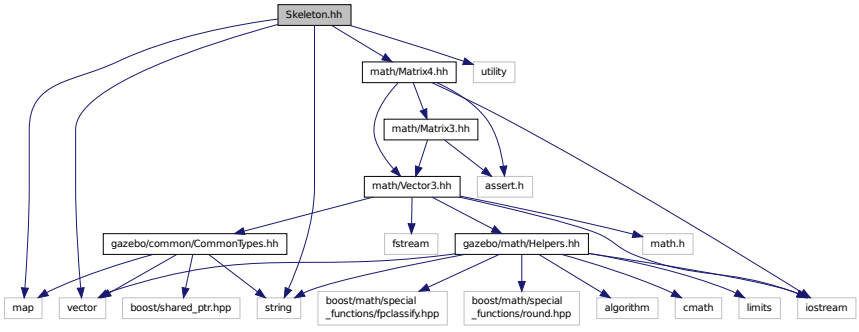
- class **SingletonT**< **T** >

  Singleton template class.

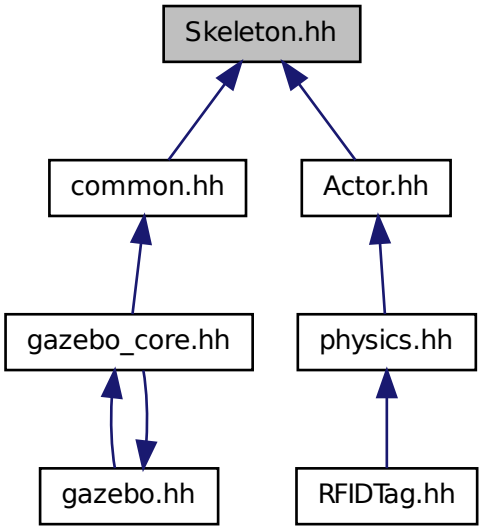
11.143 Skeleton.hh File Reference

```
#include <vector>
#include <string>
#include <map>
#include <utility>
#include "math/Matrix4.hh"
```

Include dependency graph for Skeleton.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::NodeTransform**

  NodeTransform *(*p. *547)* ***Skeleton.hh*** *(*p. *1088) common/common.hh*
- class **gazebo::common::Skeleton**

  A *(*p. *111) skeleton.*
- class **gazebo::common::SkeletonNode**

  A *(*p. *111) skeleton node.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Typedefs

- typedef std::map< unsigned int, SkeletonNode * > **gazebo::common::NodeMap**
- typedef std::map< unsigned int, SkeletonNode * >::iterator **gazebo::common::NodeMapIter**

- typedef std::map< double,
  std::vector< NodeTransform > > **gazebo::common::RawNodeAnim**

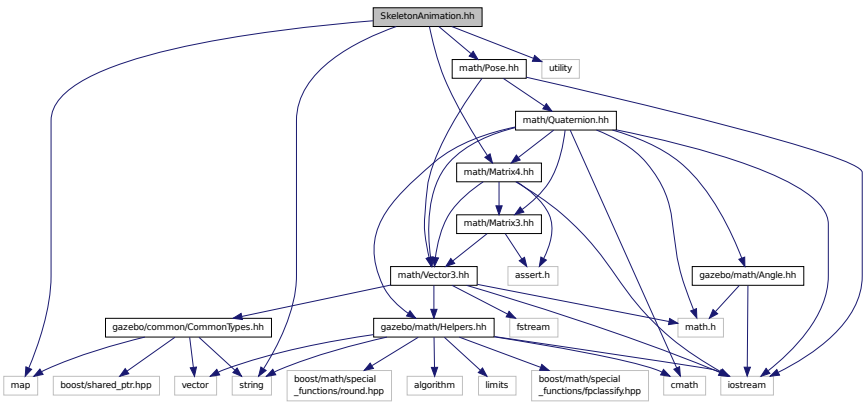
- typedef std::vector
  < std::vector< std::pair
  < std::string, double > > > **gazebo::common::RawNodeWeights**

- typedef std::map< std::string,
  RawNodeAnim > **gazebo::common::RawSkeletonAnim**

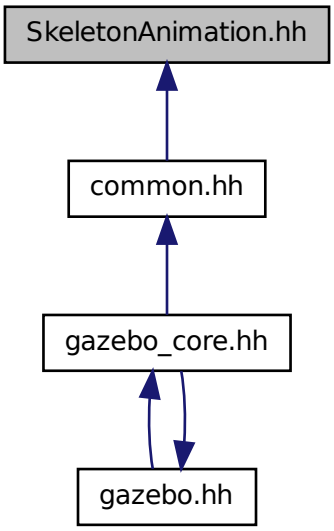
11.144 SkeletonAnimation.hh File Reference

```
#include <math/Matrix4.hh>
#include <math/Pose.hh>
#include <map>
#include <utility>
#include <string>
```

Include dependency graph for SkeletonAnimation.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::NodeAnimation**

  Node animation.

- class **gazebo::common::SkeletonAnimation**

  Skeleton *(p. 727) animation.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

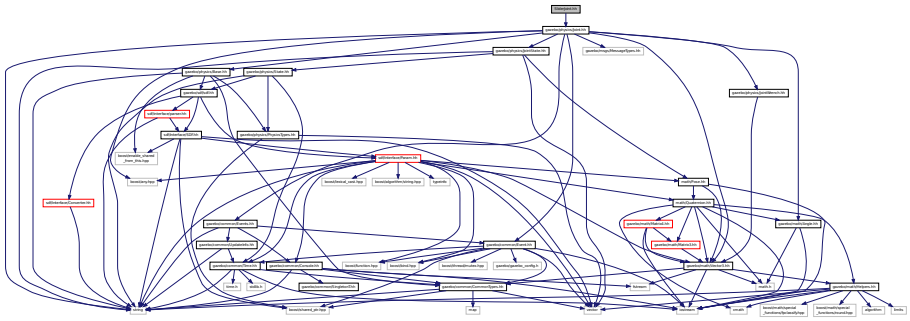
- namespace **gazebo::common**

  Common namespace.

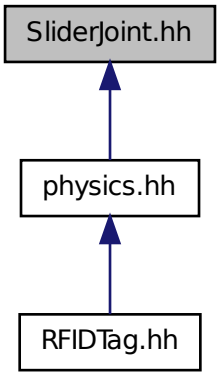
11.145 SliderJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for SliderJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SliderJoint**< **T** >

  A (p. *111) slider joint.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

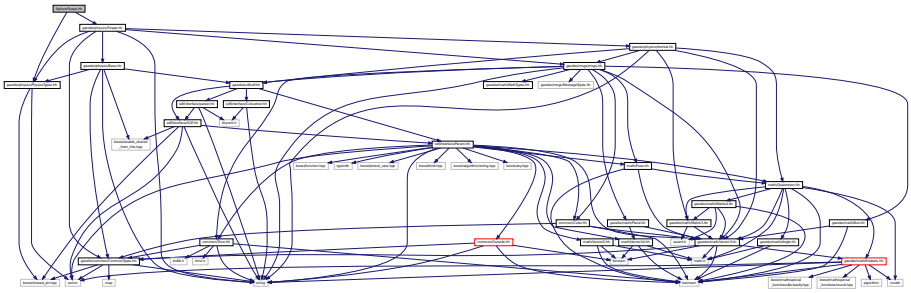
- namespace **gazebo::physics**

  namespace for physics

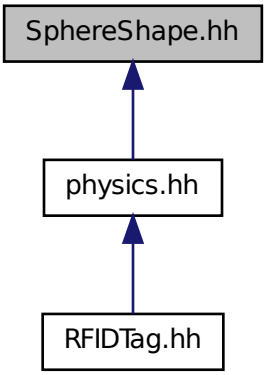
11.146 SphereShape.hh File Reference

```
#include "gazebo/physics/Shape.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for SphereShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SphereShape**

  Sphere collision shape.

Namespaces

- namespace **gazebo**

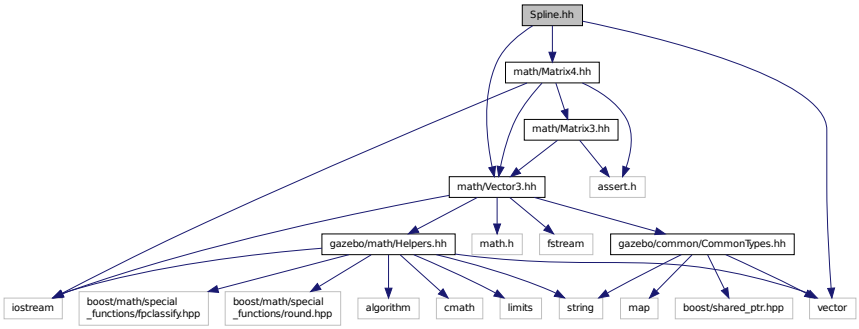
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

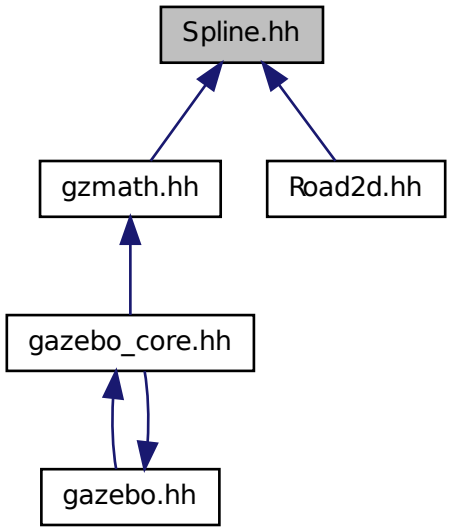
11.147 Spline.hh File Reference

```
#include <vector>
#include "math/Vector3.hh"
#include "math/Matrix4.hh"
```

Include dependency graph for Spline.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Spline**

  Splines.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

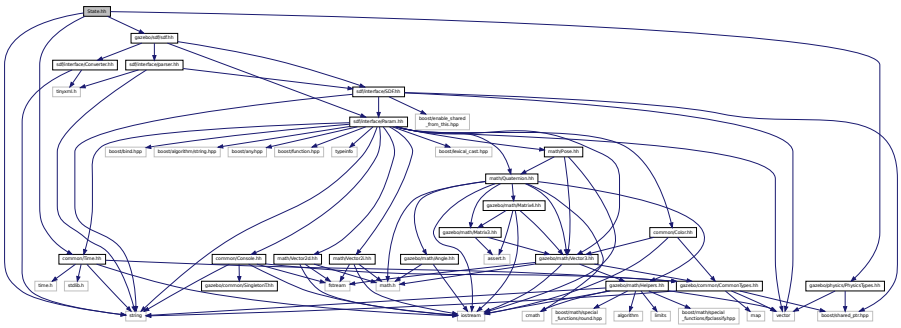
- namespace **gazebo::math**

  Math namespace.

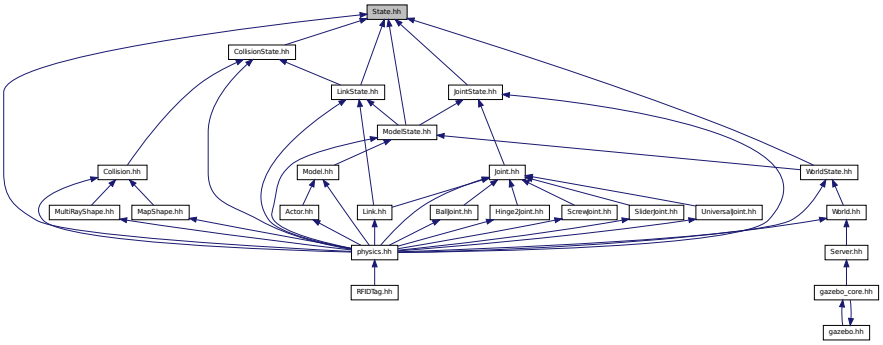
11.148 State.hh File Reference

```
#include <string>
#include "gazebo/sdf/sdf.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/common/Time.hh"
```

Include dependency graph for State.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::State**

  State *(p. 758) of an entity.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

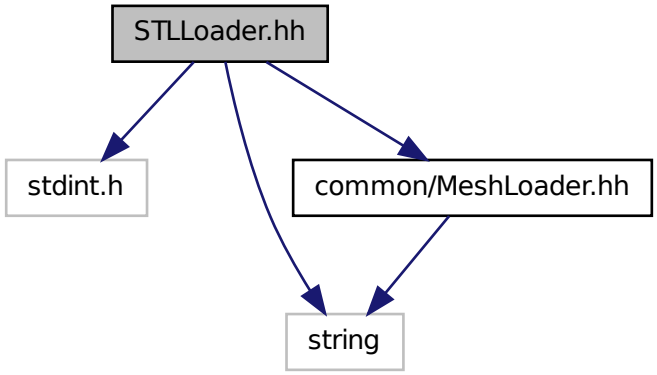
- namespace **gazebo::physics**

  namespace for physics

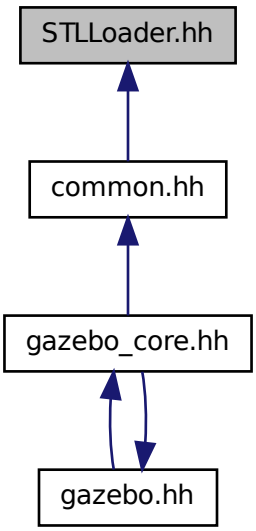
11.149 STLLoader.hh File Reference

```
#include <stdint.h>
#include <string>
#include "common/MeshLoader.hh"
```

Include dependency graph for STLLoader.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::STLLoader**

  Class used to load STL mesh files.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Macros

- #define **COR3_MAX** 200000
- #define **FACE_MAX** 200000
- #define **LINE_MAX_LEN** 256
- #define **ORDER_MAX** 10

11.149.1 Macro Definition Documentation

11.149.1.1 #define COR3_MAX 200000

11.149.1.2 #define FACE_MAX 200000

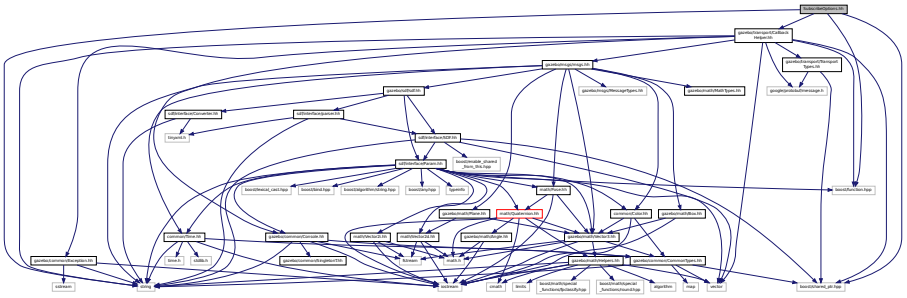
11.149.1.3 #define LINE_MAX_LEN 256

11.149.1.4 #define ORDER_MAX 10

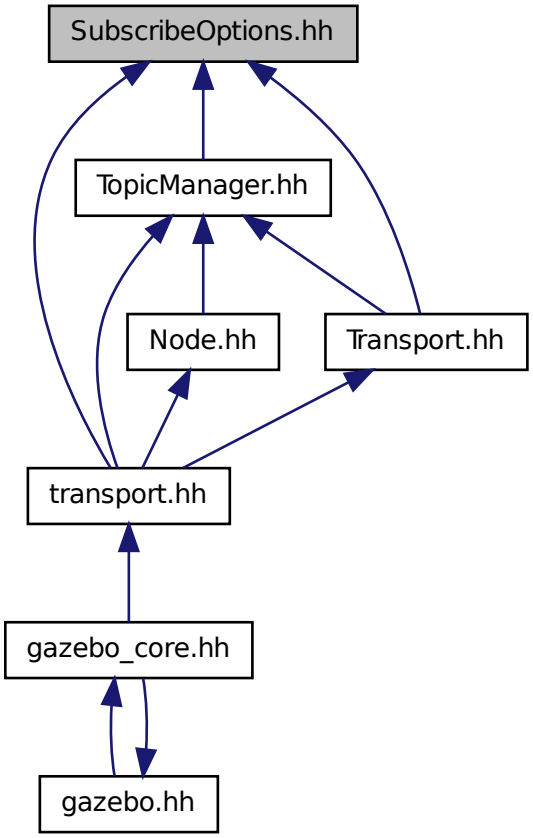
11.150 SubscribeOptions.hh File Reference

```
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <string>
#include "gazebo/transport/CallbackHelper.hh"
```

Include dependency graph for SubscribeOptions.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::SubscribeOptions**

  Options for a subscription.

Namespaces

- namespace **gazebo**

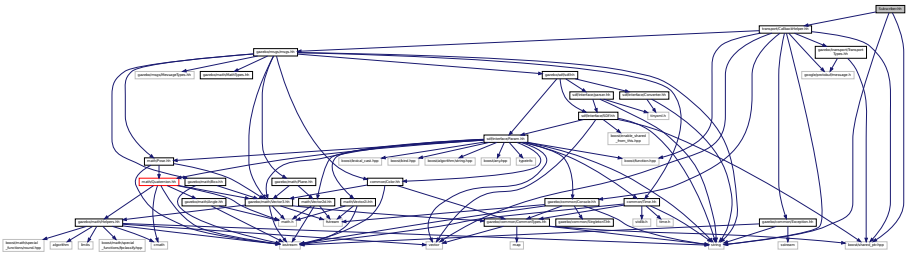
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.151 Subscriber.hh File Reference

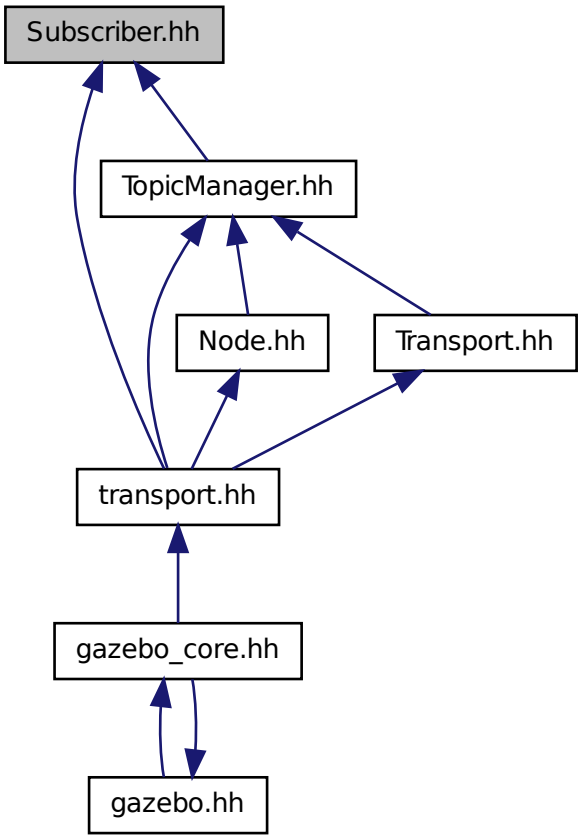
```
#include <string>
```

```
#include <boost/shared_ptr.hpp>
#include "transport/CallbackHelper.hh"
```

Include dependency graph for Subscriber.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Subscriber**

  A *(p. 111) subscriber to a topic.*

Namespaces

- namespace **gazebo**

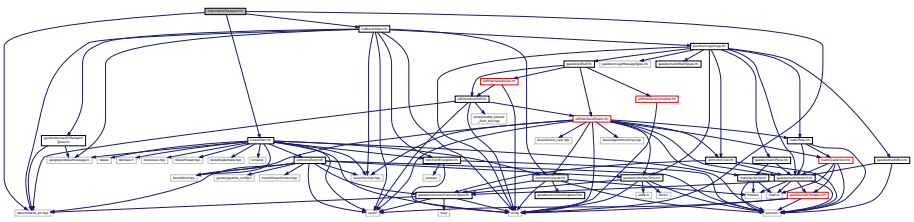
  Forward declarations for the common classes.

- namespace **gazebo::transport**

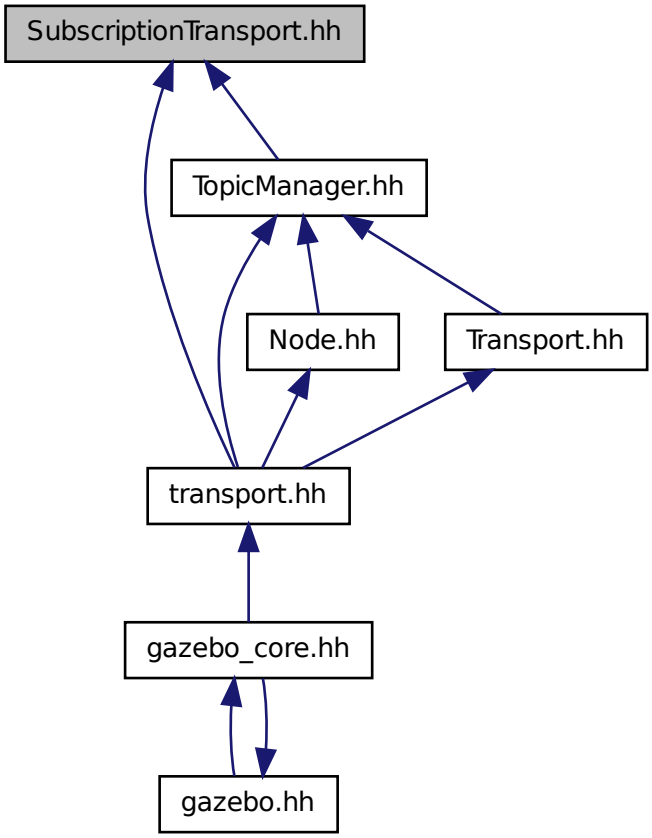
11.152 SubscriptionTransport.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include "Connection.hh"
#include "CallbackHelper.hh"
```

Include dependency graph for SubscriptionTransport.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::SubscriptionTransport**

  transport/transport.hh

Namespaces

- namespace **gazebo**

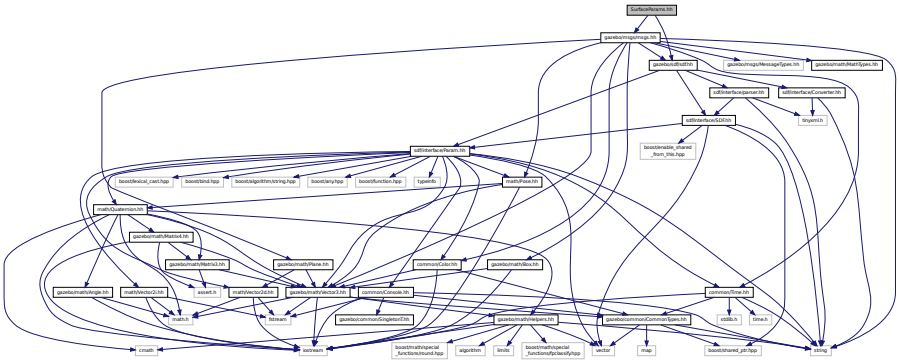
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.153 SurfaceParams.hh File Reference

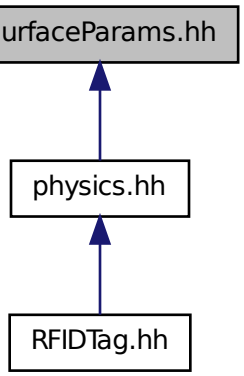
```
#include "gazebo/msgs/msgs.hh"
```

```
#include "gazebo/sdf/sdf.hh"
```

Include dependency graph for SurfaceParams.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SurfaceParams**

  SurfaceParams (p. *780*) *defines various Surface contact parameters.*

Namespaces

- namespace **gazebo**

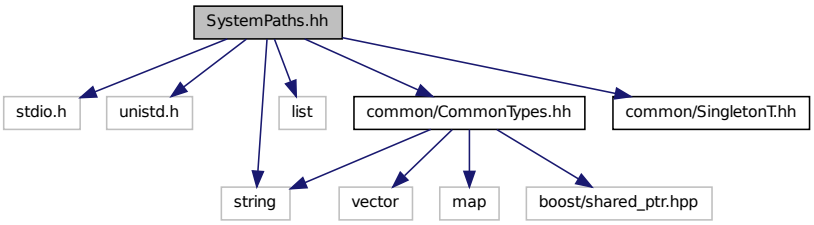
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

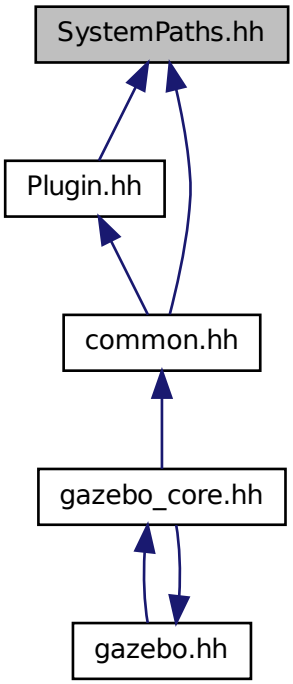
11.154 SystemPaths.hh File Reference

```
#include <stdio.h>
#include <unistd.h>
#include <string>
#include <list>
#include "common/CommonTypes.hh"
#include "common/SingletonT.hh"
```

Include dependency graph for SystemPaths.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::SystemPaths**

  Functions to handle getting system paths, keeps track of:

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Macros

- #define **GetCurrentDir** getcwd
- #define **LINUX**

11.154.1 Macro Definition Documentation

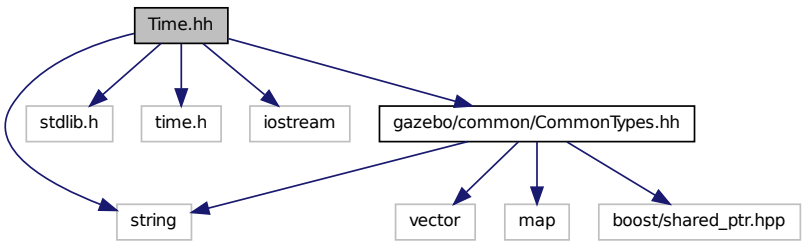
11.154.1.1 #define GetCurrentDir getcwd

11.154.1.2 #define LINUX

11.155 Time.hh File Reference

```
#include <string>
#include <stdlib.h>
#include <time.h>
#include <iostream>
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for Time.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::common::Time**

  A *(p. 111)* ***Time*** *(p. 791) class, can be used to hold wall- or sim-time.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

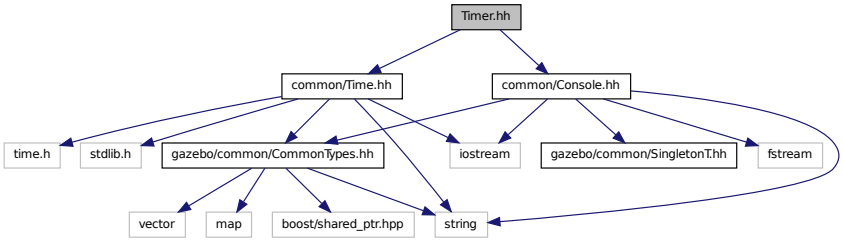
- namespace **gazebo::common**

  Common namespace.

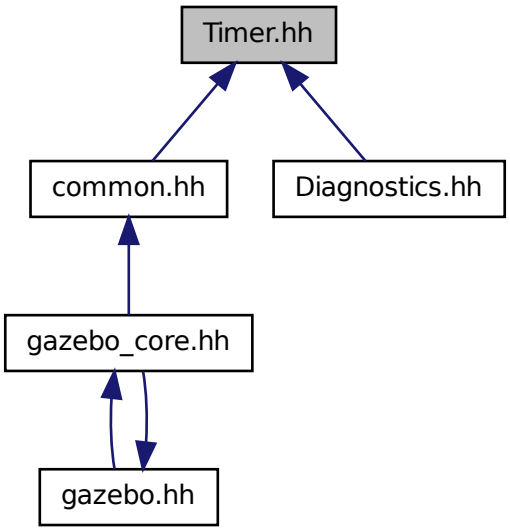
11.156 Timer.hh File Reference

```
#include "common/Console.hh"
#include "common/Time.hh"
```

Include dependency graph for Timer.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Timer**

  A (p. *111) timer class, used to time things in real world walltime.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::common**

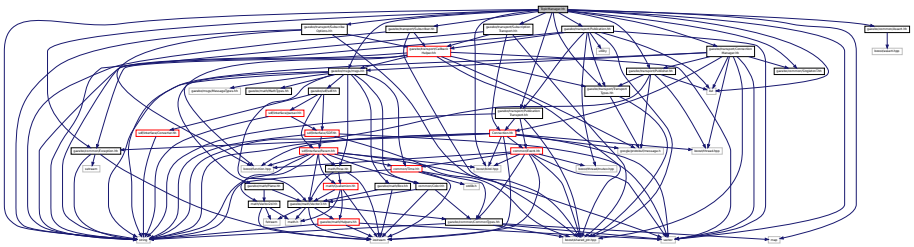
  Common namespace.

11.157 TopicManager.hh File Reference

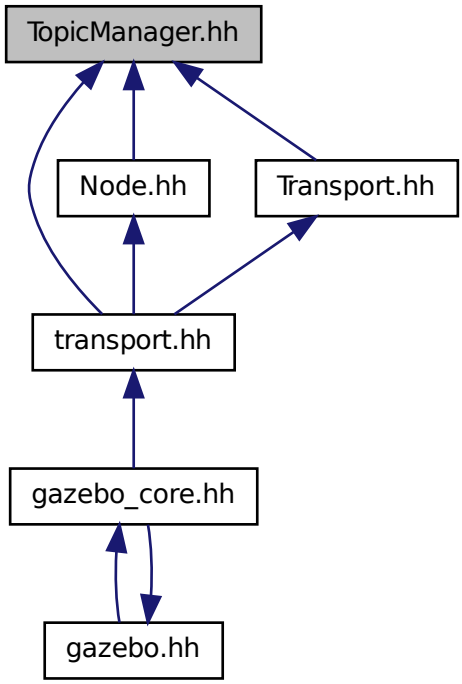
```
#include <boost/bind.hpp>
```

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include "gazebo/common/Assert.hh"
#include "gazebo/common/Exception.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/SingletonT.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/transport/SubscribeOptions.hh"
#include "gazebo/transport/SubscriptionTransport.hh"
#include "gazebo/transport/PublicationTransport.hh"
#include "gazebo/transport/ConnectionManager.hh"
#include "gazebo/transport/Publisher.hh"
#include "gazebo/transport/Publication.hh"
#include "gazebo/transport/Subscriber.hh"
```

Include dependency graph for TopicManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::TopicManager**

  Manages topics and their subscriptions.

Namespaces

- namespace **gazebo**

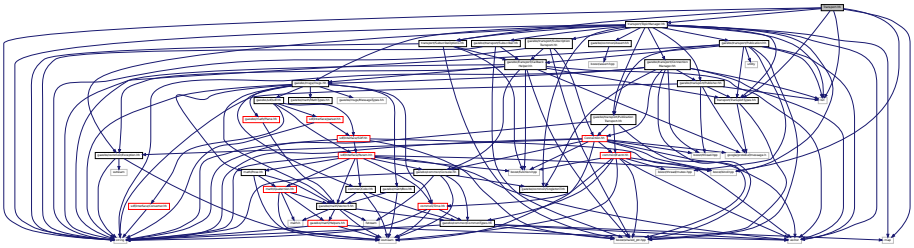
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.158 Transport.hh File Reference

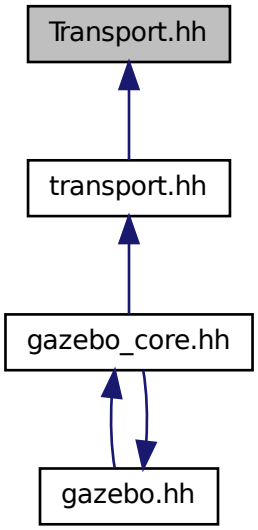
```
#include <boost/bind.hpp>
```

```
#include <string>
#include <list>
#include <map>
#include "transport/TransportTypes.hh"
#include "transport/SubscribeOptions.hh"
#include "transport/TopicManager.hh"
```

Include dependency graph for Transport.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::transport**

Functions

- void **gazebo::transport::clear_buffers** ()

  Clear any remaining communication buffers.
- void **gazebo::transport::fini** ()

  Cleanup the transport component.
- bool **gazebo::transport::get_master_uri** (std::string &_master_host, unsigned int &_master_port)

  Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.
- void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)

  Return all the namespace (world names) on the master.
- std::map< std::string, std::list< std::string > > **gazebo::transport::getAdvertisedTopics** ()

  Get a list of all the topics and their message types.
- std::list< std::string > **gazebo::transport::getAdvertisedTopics** (const std::string &_msgType)

  Get a list of all the unique advertised topic names.
- std::string **gazebo::transport::getTopicMsgType** (const std::string &_topicName)

  Get the message typename that is published on the given topic.
- bool **gazebo::transport::init** (const std::string &_master_host="", unsigned int _master_port=0)

  Initialize the transport system.
- bool **gazebo::transport::is_stopped** ()

  Is the transport system stopped?
- void **gazebo::transport::pause_incoming** (bool _pause)

  Pause or unpause incoming messages.
- msgs::Response * **gazebo::transport::request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  Send a request and receive a response.
- void **gazebo::transport::requestNoReply** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

  Send a request and don't wait for a response.
- void **gazebo::transport::requestNoReply** (NodePtr _node, const std::string &_request, const std::string &_data="")

  Send a request and don't wait for a response.
- void **gazebo::transport::run** ()

  Run the transport component.
- void **gazebo::transport::stop** ()

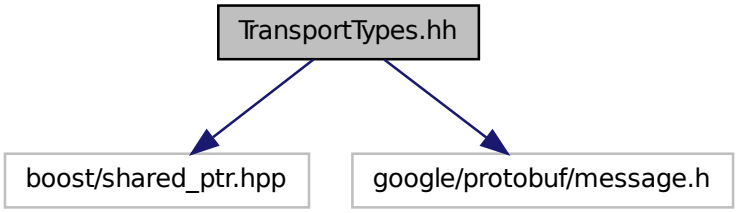
  Stop the transport component from running.

11.159 TransportTypes.hh File Reference

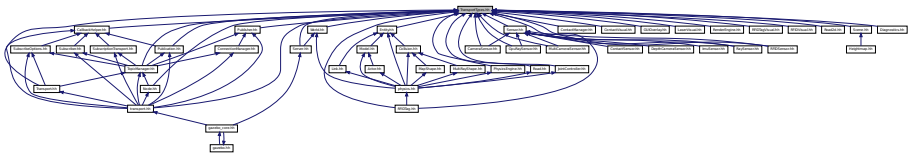
Forward declarations for transport.

```
#include <boost/shared_ptr.hpp>
#include <google/protobuf/message.h>
```

Include dependency graph for TransportTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::transport**

Typedefs

- typedef google::protobuf::Message ∗ **gazebo::transport::MessagePtr**
- typedef Node ∗ **gazebo::transport::NodePtr**
- typedef Publication ∗ **gazebo::transport::PublicationPtr**
- typedef PublicationTransport ∗ **gazebo::transport::PublicationTransportPtr**
- typedef Publisher ∗ **gazebo::transport::PublisherPtr**
- typedef Subscriber ∗ **gazebo::transport::SubscriberPtr**
- typedef SubscriptionTransport ∗ **gazebo::transport::SubscriptionTransportPtr**

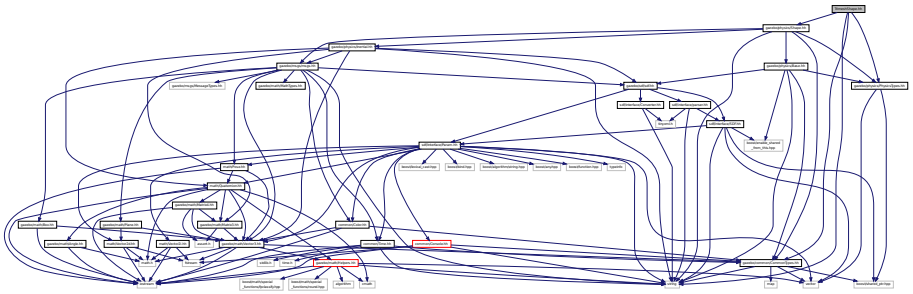
11.159.1 Detailed Description

Forward declarations for transport.

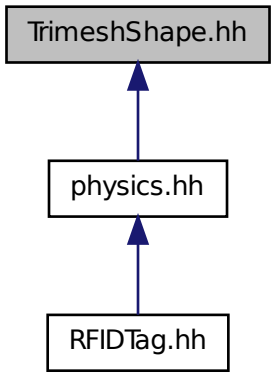
11.160 TrimeshShape.hh File Reference

```
#include <string>
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for TrimeshShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::TrimeshShape**

  Triangle mesh collision shape.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

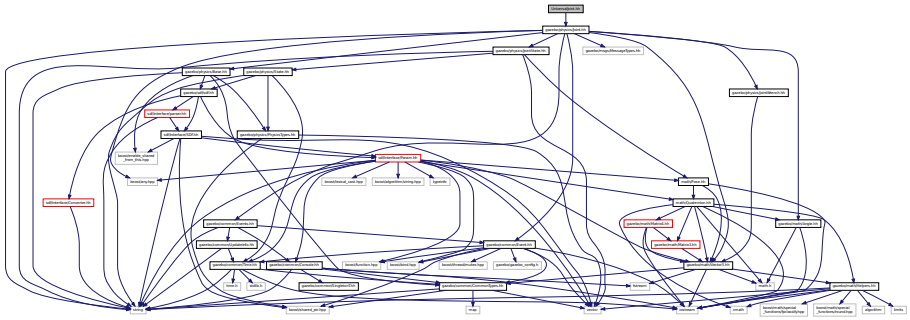
- namespace **gazebo::physics**

  namespace for physics

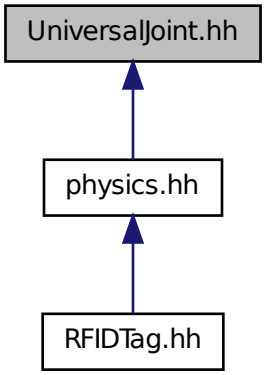
11.161 UniversalJoint.hh File Reference

```
#include "gazebo/physics/Joint.hh"
```

Include dependency graph for UniversalJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::UniversalJoint**< **T** >

  A (p. *111*) *universal joint.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

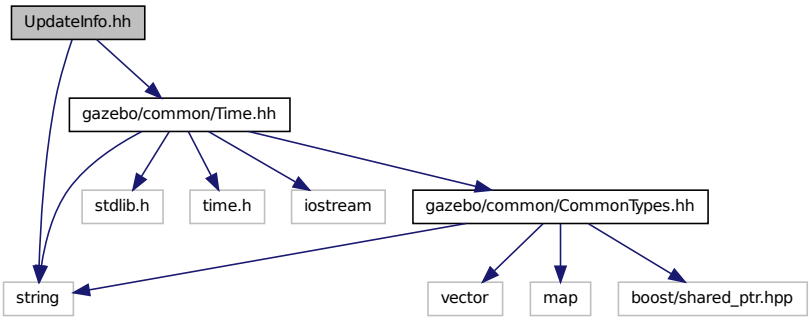
- namespace **gazebo::physics**

  namespace for physics

11.162 UpdateInfo.hh File Reference

```
#include <string>
#include "gazebo/common/Time.hh"
```

Include dependency graph for UpdateInfo.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::common::UpdateInfo**

  Information for use in an update event.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

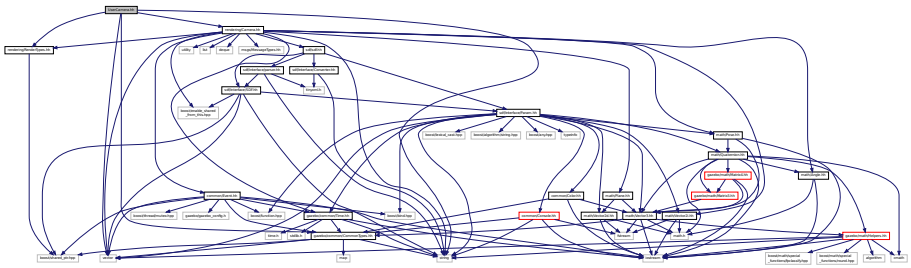
- namespace **gazebo::common**

  Common namespace.

11.163 UserCamera.hh File Reference

```
#include <string>
#include <vector>
#include "rendering/Camera.hh"
#include "rendering/RenderTypes.hh"
#include "common/CommonTypes.hh"
```

Include dependency graph for UserCamera.hh:



Classes

- class **gazebo::rendering::UserCamera**

  A (p. *111*) *camera used for user visualization of a scene.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

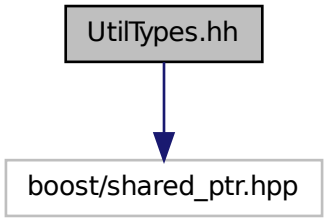
- namespace **gazebo::rendering**

  Rendering namespace.

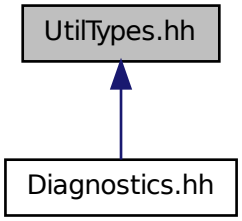
11.164 UtilTypes.hh File Reference

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for UtilTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::util**

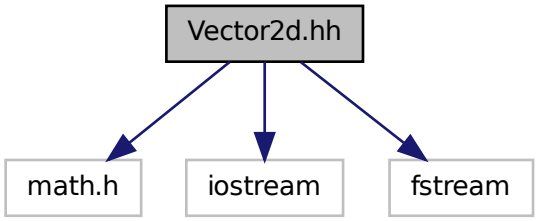
Typedefs

- typedef DiagnosticTimer ∗ **gazebo::util::DiagnosticTimerPtr**

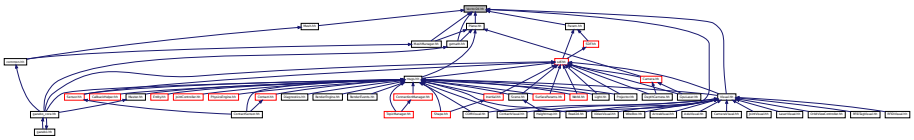
11.165 Vector2d.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
```

Include dependency graph for Vector2d.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector2d**

  Generic double x, y vector.

Namespaces

- namespace **gazebo**

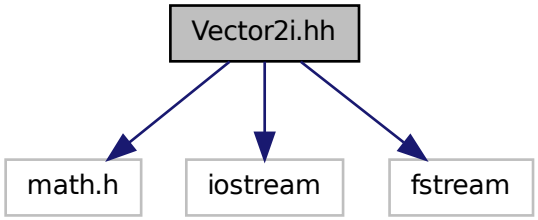
  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

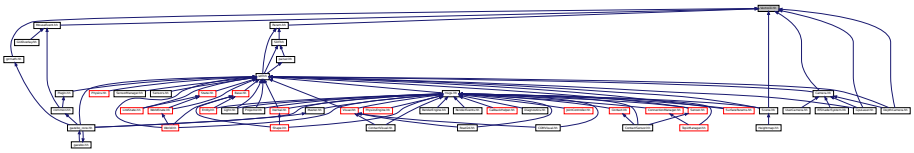
11.166 Vector2i.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
```

Include dependency graph for Vector2i.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector2i**

  Generic integer x, y vector.

Namespaces

- namespace **gazebo**

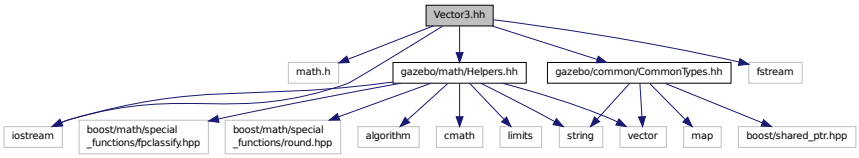
  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

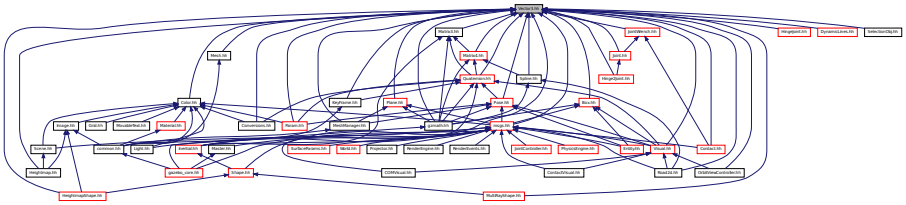
11.167 Vector3.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
#include "gazebo/math/Helpers.hh"
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for Vector3.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector3**

  *The **Vector3** (p. 855) class represents the generic vector containing 3 elements.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

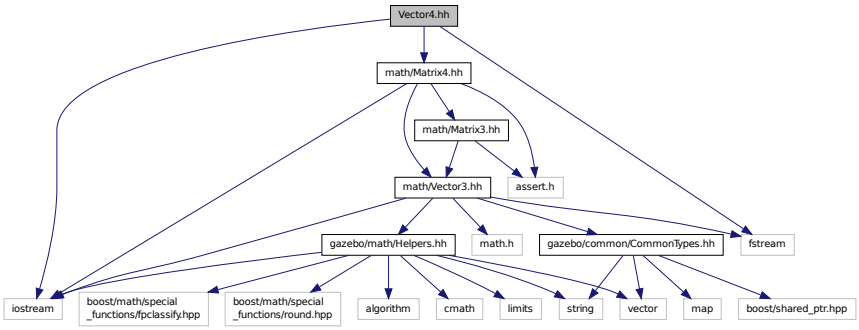
- namespace **gazebo::math**

  Math namespace.

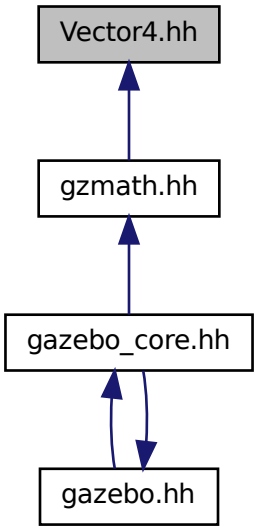
11.168 Vector4.hh File Reference

```
#include <iostream>
#include <fstream>
#include "math/Matrix4.hh"
```

Include dependency graph for Vector4.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector4**

  double Generic x, y, z, w vector

Namespaces

- namespace **gazebo**

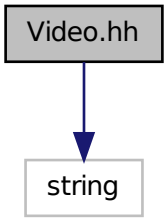
Forward declarations for the common classes.

- namespace **gazebo::math**

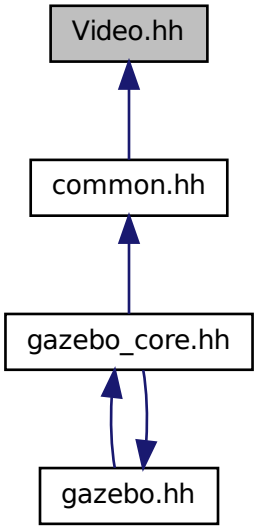
  Math namespace.

11.169 Video.hh File Reference

`#include <string>`
Include dependency graph for Video.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Video**

  Handle video encoding and decoding using libavcodec.

Namespaces

- namespace **gazebo**

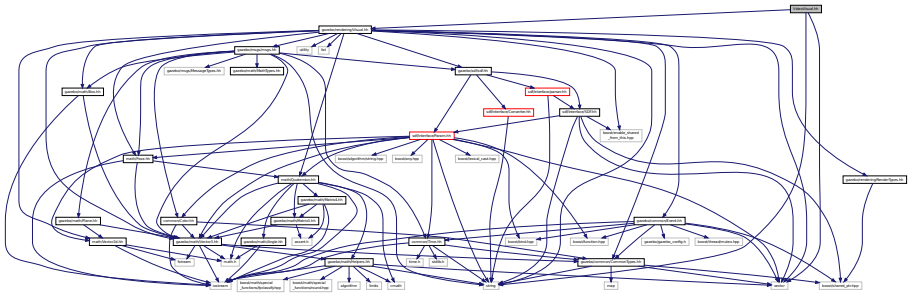
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.170 VideoVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for VideoVisual.hh:



Classes

- class **gazebo::rendering::VideoVisual**

  A (p. *111) visual element that displays a video as a texture.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

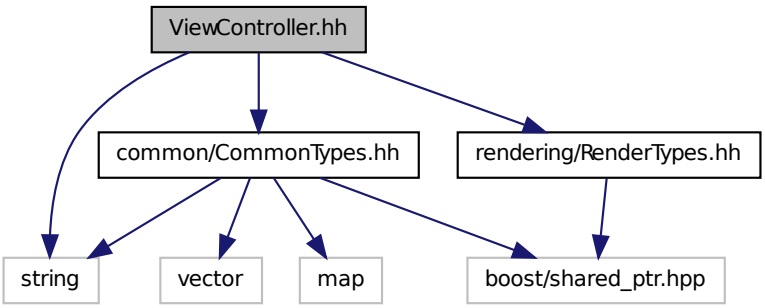
  Common namespace.
- namespace **gazebo::rendering**

  Rendering namespace.

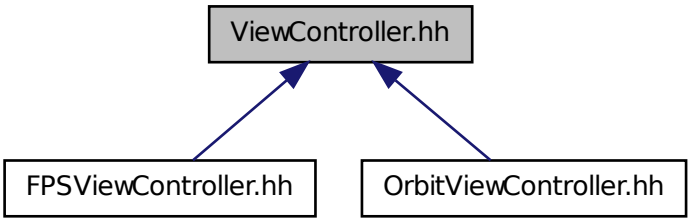
11.171 ViewController.hh File Reference

```
#include <string>
#include "common/CommonTypes.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for ViewController.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::ViewController**

  Base class for view controllers.

Namespaces

- namespace **gazebo**

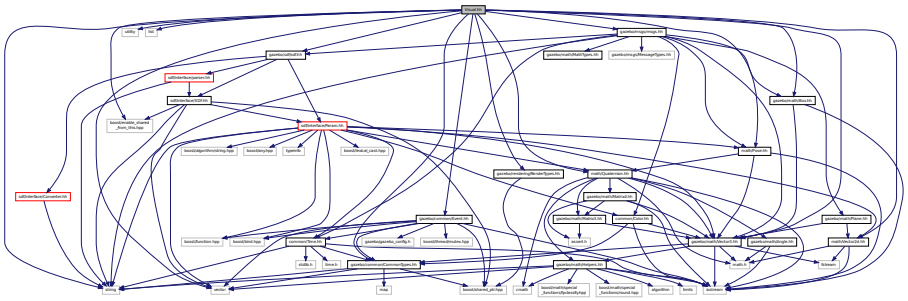
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

Rendering namespace.

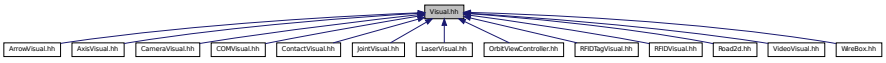
11.172 Visual.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <list>
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/math/Quaternion.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Vector2d.hh"
#include "gazebo/sdf/sdf.hh"
#include "gazebo/rendering/RenderTypes.hh"
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for Visual.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Visual**

  A (p. *111*) *renderable object.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

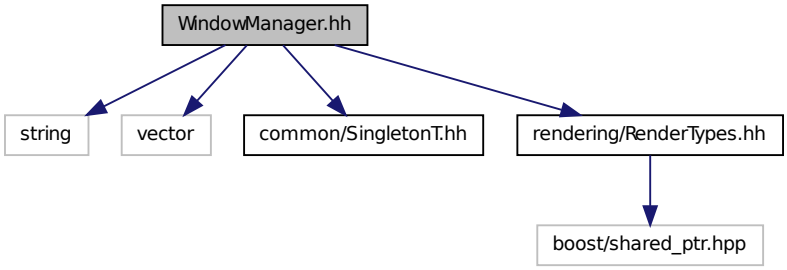
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.173 WindowManager.hh File Reference

```
#include <string>
#include <vector>
#include "common/SingletonT.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for WindowManager.hh:



Classes

- class **gazebo::rendering::WindowManager**

  Class to mangage render windows.

Namespaces

- namespace **gazebo**

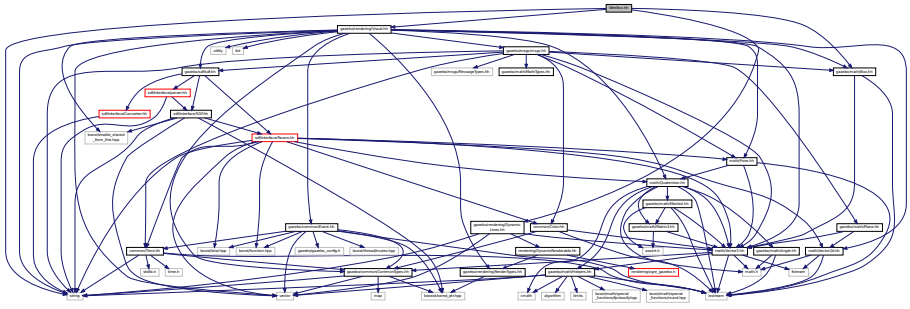
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.174 WireBox.hh File Reference

```
#include <string>
#include "gazebo/math/Box.hh"
#include "gazebo/rendering/Visual.hh"
#include "gazebo/rendering/DynamicLines.hh"
```

Include dependency graph for WireBox.hh:



Classes

- class **gazebo::rendering::WireBox**

  Draws a wireframe box.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

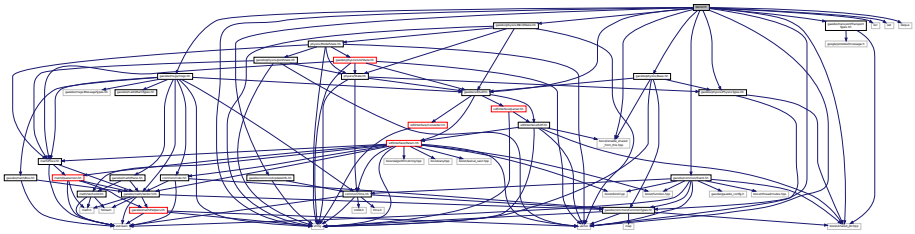
- namespace **gazebo::rendering**

  Rendering namespace.

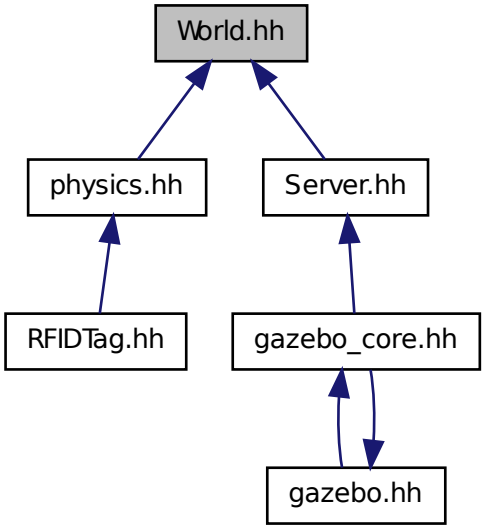
11.175 World.hh File Reference

```
#include <vector>
#include <list>
#include <set>
#include <deque>
#include <string>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/common/UpdateInfo.hh"
#include "gazebo/common/Event.hh"
#include "gazebo/physics/Base.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/WorldState.hh"
#include "gazebo/sdf/sdf.hh"
```

Include dependency graph for World.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::World**

  The world provides access to all other object within a simulated environment.

Namespaces

- namespace **boost**
- namespace **gazebo**

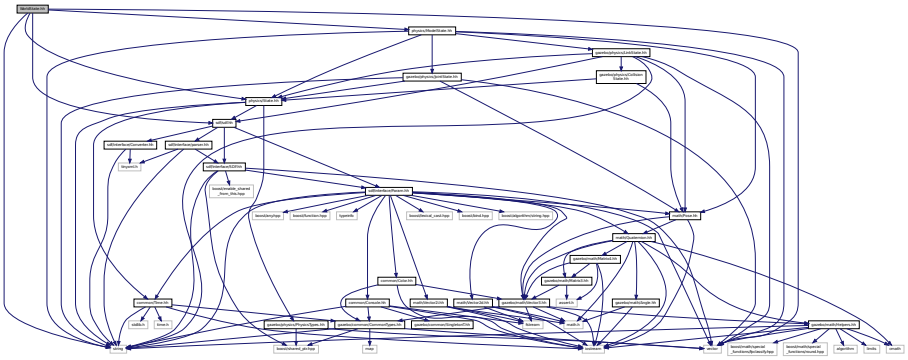
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

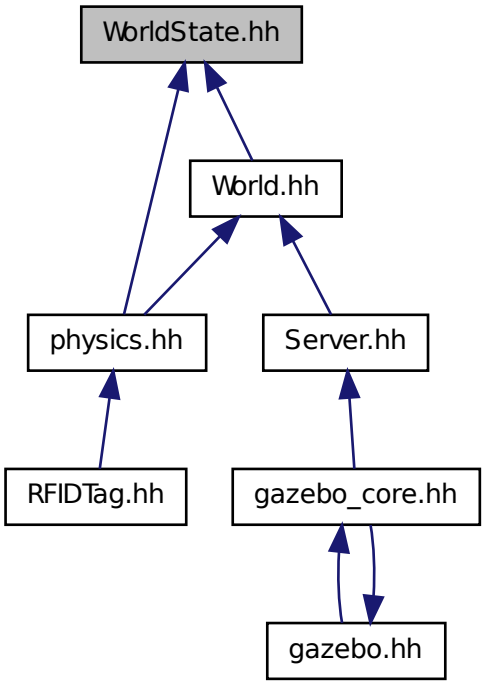
11.176 WorldState.hh File Reference

```
#include <string>
#include <vector>
#include "sdf/sdf.hh"
#include "physics/State.hh"
#include "physics/ModelState.hh"
```

Include dependency graph for WorldState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::WorldState**

  *Store state information of a **physics::World** (p. 910) object.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Index

~Actor
gazebo::physics::Actor, 114
~Angle
gazebo::math::Angle, 120
~Animation
gazebo::common::Animation, 127
~ArrowVisual
gazebo::rendering::ArrowVisual, 131
~AssertionInternalError
gazebo::common::AssertionInternalError, 133
~AxisVisual
gazebo::rendering::AxisVisual, 134
~BVHLoader
gazebo::common::BVHLoader, 156
~BallJoint
gazebo::physics::BallJoint, 137
~Base
gazebo::physics::Base, 141
~Box
gazebo::math::Box, 150
~BoxShape
gazebo::physics::BoxShape, 155
~COMVisual
gazebo::rendering::COMVisual, 221
~CallbackHelper
gazebo::transport::CallbackHelper, 158
~Camera
gazebo::rendering::Camera, 169
~CameraSensor
gazebo::sensors::CameraSensor, 190
~CameraVisual
gazebo::rendering::CameraVisual, 194
~ColladaLoader
gazebo::common::ColladaLoader, 195
~Collision
gazebo::physics::Collision, 198
~CollisionState
gazebo::physics::CollisionState, 206
~Color
gazebo::common::Color, 211
~Connection
gazebo::event::Connection, 222
gazebo::transport::Connection, 225
~Contact
gazebo::physics::Contact, 237
~ContactManager
gazebo::physics::ContactManager, 240
~ContactSensor
gazebo::sensors::ContactSensor, 243
~ContactVisual
gazebo::rendering::ContactVisual, 247
~CylinderShape
gazebo::physics::CylinderShape, 252
~DepthCamera
gazebo::rendering::DepthCamera, 255
~DepthCameraSensor
gazebo::sensors::DepthCameraSensor, 259
~DiagnosticTimer
gazebo::util::DiagnosticTimer, 265
~DynamicLines
gazebo::rendering::DynamicLines, 267
~DynamicRenderable
gazebo::rendering::DynamicRenderable, 271
~Element
sdf::Element, 276
~Entity
gazebo::physics::Entity, 284
~Event
gazebo::event::Event, 294
~EventT
Events, 38
~Exception
gazebo::common::Exception, 319
~FPSViewController
gazebo::rendering::FPSViewController, 322
~GUIOverlay
gazebo::rendering::GUIOverlay, 346
~GazeboGenerator
google::protobuf::compiler::cpp::GazeboGenerator, 324
~GpuLaser
gazebo::rendering::GpuLaser, 326
~GpuRaySensor
gazebo::sensors::GpuRaySensor, 331
~Grid
gazebo::rendering::Grid, 341
~Gripper
gazebo::physics::Gripper, 344
~GzTerrainMatGen
gazebo::rendering::GzTerrainMatGen, 350

~Heightmap
gazebo::rendering::Heightmap, 351
~HeightmapShape
gazebo::physics::HeightmapShape, 354
~Hinge2Joint
gazebo::physics::Hinge2Joint, 357
~HingeJoint
gazebo::physics::HingeJoint, 359
~IOManager
gazebo::transport::IOManager, 380
~Image
gazebo::common::Image, 362
~ImuSensor
gazebo::sensors::ImuSensor, 367
~Inertial
gazebo::physics::Inertial, 371
~InternalError
gazebo::common::InternalError, 380
~Joint
gazebo::physics::Joint, 386
~JointState
gazebo::physics::JointState, 403
~JointVisual
gazebo::rendering::JointVisual, 406
~KeyFrame
gazebo::common::KeyFrame, 409
~LaserVisual
gazebo::rendering::LaserVisual, 412
~Light
gazebo::rendering::Light, 414
~Link
gazebo::physics::Link, 423
~LinkState
gazebo::physics::LinkState, 440
~Master
gazebo::Master, 453
~Material
gazebo::common::Material, 456
~Matrix3
gazebo::math::Matrix3, 464
~Matrix4
gazebo::math::Matrix4, 470
~Mesh
gazebo::common::Mesh, 476
~MeshCSG
gazebo::common::MeshCSG, 482
~MeshLoader
gazebo::common::MeshLoader, 483
~Model
gazebo::physics::Model, 492
~ModelPlugin
gazebo::ModelPlugin, 505
~ModelState
gazebo::physics::ModelState, 508
~MovableText
gazebo::rendering::MovableText, 517
~MultiCameraSensor
gazebo::sensors::MultiCameraSensor, 524
~MultiRayShape
gazebo::physics::MultiRayShape, 530
~Node
gazebo::transport::Node, 537
~NodeAnimation
gazebo::common::NodeAnimation, 543
~NodeTransform
gazebo::common::NodeTransform, 549
~NumericAnimation
gazebo::common::NumericAnimation, 553
~NumericKeyFrame
gazebo::common::NumericKeyFrame, 554
~OrbitViewController
gazebo::rendering::OrbitViewController, 557
~PID
gazebo::common::PID, 584
~Param
sdf::Param, 561
~ParamT
sdf::ParamT, 566
~PhysicsEngine
gazebo::physics::PhysicsEngine, 571
~Plane
gazebo::math::Plane, 588
~PlaneShape
gazebo::physics::PlaneShape, 591
~Pose
gazebo::math::Pose, 599
~PoseAnimation
gazebo::common::PoseAnimation, 606
~PoseKeyFrame
gazebo::common::PoseKeyFrame, 608
~Projector
gazebo::rendering::Projector, 610
~Publication
gazebo::transport::Publication, 613
~PublicationTransport
gazebo::transport::PublicationTransport, 617
~Publisher
gazebo::transport::Publisher, 620
~Quaternion
gazebo::math::Quaternion, 627
~RFIDSensor
gazebo::sensors::RFIDSensor, 659
~RFIDTag
gazebo::sensors::RFIDTag, 661
~RFIDTagVisual
gazebo::rendering::RFIDTagVisual, 664
~RFIDVisual
gazebo::rendering::RFIDVisual, 665

~RaySensor
gazebo::sensors::RaySensor, 643
~RayShape
gazebo::physics::RayShape, 650
~Road
gazebo::physics::Road, 666
~Road2d
gazebo::rendering::Road2d, 668
~RotationSpline
gazebo::math::RotationSpline, 669
~SM2Profile
gazebo::rendering::GzTerrainMatGen::SM2Profile, 751
~STLLoader
gazebo::common::STLLoader, 763
~Scene
gazebo::rendering::Scene, 680
~ScrewJoint
gazebo::physics::ScrewJoint, 693
~SelectionObj
gazebo::rendering::SelectionObj, 697
~Sensor
gazebo::sensors::Sensor, 701
~SensorPlugin
gazebo::SensorPlugin, 713
~Server
gazebo::Server, 715
~Shape
gazebo::physics::Shape, 722
~SimTimeEventHandler
gazebo::sensors::SimTimeEventHandler, 724
~SingletonT
SingletonT, 727
~Skeleton
gazebo::common::Skeleton, 729
~SkeletonAnimation
gazebo::common::SkeletonAnimation, 735
~SkeletonNode
gazebo::common::SkeletonNode, 741
~SliderJoint
gazebo::physics::SliderJoint, 748
~SphereShape
gazebo::physics::SphereShape, 753
~Spline
gazebo::math::Spline, 755
~State
gazebo::physics::State, 760
~SubMesh
gazebo::common::SubMesh, 767
~Subscriber
gazebo::transport::Subscriber, 777
~SubscriptionTransport
gazebo::transport::SubscriptionTransport, 779
~SurfaceParams
gazebo::physics::SurfaceParams, 781
~SystemPlugin
gazebo::SystemPlugin, 790
~Time
gazebo::common::Time, 795
~Timer
gazebo::common::Timer, 813
~TrimeshShape
gazebo::physics::TrimeshShape, 823
~URDF2Gazebo
urdf2gazebo::URDF2Gazebo, 828
~UniversalJoint
gazebo::physics::UniversalJoint, 827
~UserCamera
gazebo::rendering::UserCamera, 832
~Vector2d
gazebo::math::Vector2d, 840
~Vector2i
gazebo::math::Vector2i, 848
~Vector3
gazebo::math::Vector3, 859
~Vector4
gazebo::math::Vector4, 871
~Video
gazebo::common::Video, 879
~VideoVisual
gazebo::rendering::VideoVisual, 881
~ViewController
gazebo::rendering::ViewController, 882
~Visual
gazebo::rendering::Visual, 890
~WireBox
gazebo::rendering::WireBox, 909
~World
gazebo::physics::World, 913
~WorldPlugin
gazebo::WorldPlugin, 922
~WorldState
gazebo::physics::WorldState, 925
_setupGeometry
gazebo::rendering::MovableText, 518
_updateColors
gazebo::rendering::MovableText, 518

A, 111
a
gazebo::common::Color, 219
ABGR
gazebo::common::Color, 211
ACTOR
gazebo::physics::Base, 140
ADD
gazebo::common::Material, 456
ARGB

gazebo::common::Color, 211
AcceptCallback
gazebo::transport::Connection, 224
active
gazebo::physics::Actor, 116
gazebo::sensors::Sensor, 706
Actor
gazebo::physics::Actor, 114
Actor.hh, 929
Actor_V
gazebo::physics, 95
ActorPtr
gazebo::physics, 95
Add
gazebo::common::LogRecord, 448
add_plugin
gazebo, 83
add_search_path_suffix
Common, 32
AddAnimation
gazebo::common::Skeleton, 729
AddAttribute
sdf::Element, 276
AddCallback
gazebo::transport::PublicationTransport, 617
AddChild
gazebo::common::SkeletonNode, 741
gazebo::physics::Base, 141
AddChildJoint
gazebo::physics::Link, 423
AddContact
gazebo::physics::Collision, 198
AddElement
sdf::Element, 276
AddElementDescription
sdf::Element, 276
addEntity
gazebo::event::Events, 307
AddForce
gazebo::physics::Link, 423
AddForceAtRelativePosition
gazebo::physics::Link, 424
AddForceAtWorldPosition
gazebo::physics::Link, 424
AddGazeboPaths
gazebo::common::SystemPaths, 786
AddIndex
gazebo::common::SubMesh, 767
AddJoint
gazebo::physics::JointController, 400
AddKeyFrame
gazebo::common::NodeAnimation, 543
gazebo::common::SkeletonAnimation, 735
AddMaterial
gazebo::common::Mesh, 476
AddMesh
gazebo::common::MeshManager, 485
AddModelPaths
gazebo::common::SystemPaths, 786
addNestedModel
sdf, 107
AddNode
gazebo::transport::TopicManager, 816
AddNodeAssignment
gazebo::common::SubMesh, 767
AddNormal
gazebo::common::SubMesh, 767
AddOgrePaths
gazebo::common::SystemPaths, 786
AddParentJoint
gazebo::physics::Link, 424
AddPluginPaths
gazebo::common::SystemPaths, 786
AddPoint
gazebo::math::RotationSpline, 669
gazebo::math::Spline, 755
gazebo::rendering::DynamicLines, 268
AddPublisher
gazebo::transport::Publication, 613
AddRawTransform
gazebo::common::SkeletonNode, 741
AddRay
gazebo::physics::MultiRayShape, 530
AddRelativeEvent
gazebo::sensors::SimTimeEventHandler, 724
AddRelativeForce
gazebo::physics::Link, 424
AddRelativeTorque
gazebo::physics::Link, 424
AddResourcePath
gazebo::rendering::RenderEngine, 655
AddScene
gazebo::rendering::RTShaderSystem, 674
AddSearchPathSuffix
gazebo::common::SystemPaths, 786
AddSubMesh
gazebo::common::Mesh, 477
AddSubscription
gazebo::transport::Publication, 613
AddTag
gazebo::sensors::RFIDSensor, 659
addTechnique
gazebo::rendering::GzTerrainMatGen::SM2Profile, 751
AddTexCoord
gazebo::common::SubMesh, 767
AddTime
gazebo::common::Animation, 127

AddTorque
gazebo::physics::Link, 425
AddTransport
gazebo::transport::Publication, 613
AddType
gazebo::physics::Base, 141
AddValue
sdf::Element, 276
AddVertNodeWeight
gazebo::common::Skeleton, 729
AddVertex
gazebo::common::SubMesh, 768
AddVisual
gazebo::rendering::Scene, 680
Advertise
gazebo::transport::ConnectionManager, 230
gazebo::transport::Node, 537
gazebo::transport::TopicManager, 816
alt
gazebo::common::MouseEvent, 514
ambient
gazebo::common::Material, 462
anchorLink
gazebo::physics::Joint, 397
anchorPos
gazebo::physics::Joint, 397
anchorPose
gazebo::physics::Joint, 398
Angle
gazebo::math::Angle, 119
Angle.hh, 930
GZ_DTOR, 932
GZ_NORMALIZE, 932
GZ_RTOD, 932
angularAccel
gazebo::physics::Link, 437
animState
gazebo::rendering::Camera, 185
Animation
gazebo::common::Animation, 127
animation
gazebo::physics::Entity, 291
Animation.hh, 933
AnimationComplete
gazebo::rendering::Camera, 169
gazebo::rendering::UserCamera, 832
animationConnection
gazebo::physics::Entity, 291
AnimationPtr
gazebo::common, 86
animationStartPose
gazebo::physics::Entity, 291
animations
gazebo::common::SkeletonAnimation, 737
anims
gazebo::common::Skeleton, 733
ApplyDamping
gazebo::physics::Joint, 386
applyDamping
gazebo::physics::Joint, 398
ApplyShadows
gazebo::rendering::RTShaderSystem, 674
AreConnected
gazebo::physics::Joint, 386
ArrowVisual
gazebo::rendering::ArrowVisual, 130
ArrowVisual.hh, 934
ArrowVisualPtr
gazebo::rendering, 99
Assert.hh, 935
GZ_ASSERT, 936
AssertionInternalError
gazebo::common::AssertionInternalError, 132
AsyncRead
gazebo::transport::Connection, 225
Attach
gazebo::physics::Joint, 386
gazebo::rendering::SelectionObj, 697
AttachAxes
gazebo::rendering::Visual, 890
AttachCameraToImage
gazebo::rendering::GUIOverlay, 346
AttachEntity
gazebo::rendering::RTShaderSystem, 674
AttachLineVertex
gazebo::rendering::Visual, 890
AttachMesh
gazebo::rendering::Visual, 890
AttachObject
gazebo::rendering::Visual, 891
AttachStaticModel
gazebo::physics::Link, 425
gazebo::physics::Model, 492
AttachToVisual
gazebo::rendering::Camera, 169
AttachToVisualImpl
gazebo::rendering::Camera, 169, 170
gazebo::rendering::UserCamera, 832
AttachViewport
gazebo::rendering::RTShaderSystem, 674
AttachVisual
gazebo::rendering::Visual, 891
attachedModels
gazebo::physics::Model, 502
attachedModelsOffset
gazebo::physics::Link, 437
gazebo::physics::Model, 502
Attribute

gazebo::physics::Joint, 385
autoCalc
gazebo::math::RotationSpline, 672
gazebo::math::Spline, 758
autoStart
gazebo::physics::Actor, 116
AxisVisual
gazebo::rendering::AxisVisual, 134
AxisVisual.hh, 937
AxisVisualPtr
gazebo::rendering, 99

b
gazebo::common::Color, 219
BALL_JOINT
gazebo::physics::Base, 141
BASE
gazebo::physics::Base, 140
BAYER_GBRG8
gazebo::common::Image, 362
BAYER_GRBG8
gazebo::common::Image, 362
BAYER_RGGB8
gazebo::common::Image, 362
BAYER_RGGR8
gazebo::common::Image, 362
BGR_INT16
gazebo::common::Image, 361
BGR_INT32
gazebo::common::Image, 361
BGR_INT8
gazebo::common::Image, 361
BGRA
gazebo::common::Color, 211
BGRA_INT8
gazebo::common::Image, 361
BLEND_COUNT
gazebo::common::Material, 456
BLINN
gazebo::common::Material, 456
BOX_SHAPE
gazebo::physics::Base, 141
BVHLoader
gazebo::common::BVHLoader, 156
BVHLoader.hh, 941
X_POSITION, 943
X_ROTATION, 943
Y_POSITION, 943
Y_ROTATION, 943
Z_POSITION, 943
Z_ROTATION, 943
BallJoint
gazebo::physics::BallJoint, 136
BallJoint.hh, 937
Base
gazebo::physics::Base, 141
Base.hh, 938
Base_V
gazebo::physics, 95
BasePtr
gazebo::physics, 95
bayerFrameBuffer
gazebo::rendering::Camera, 186
bindShapeTransform
gazebo::common::Skeleton, 733
Black
gazebo::common::Color, 219
BlendMode
gazebo::common::Material, 456
blendMode
gazebo::common::Material, 462
BlendModeStr
gazebo::common::Material, 462
Blue
gazebo::common::Color, 219
body1Force
gazebo::physics::JointWrench, 408
body1Torque
gazebo::physics::JointWrench, 408
body2Force
gazebo::physics::JointWrench, 408
body2Torque
gazebo::physics::JointWrench, 408
bonePosePub
gazebo::physics::Actor, 116
BooleanOperation
gazebo::common::MeshCSG, 482
boost, 81
bounce
gazebo::physics::SurfaceParams, 782
bounceThreshold
gazebo::physics::SurfaceParams, 782
Box
gazebo::math::Box, 150
Box.hh, 939
BoxShape
gazebo::physics::BoxShape, 155
BoxShape.hh, 940
BoxShapePtr
gazebo::physics, 95
build
gazebo::common::Animation, 128
BuildInterpolationSplines
gazebo::common::PoseAnimation, 606
BuildNodeMap
gazebo::common::Skeleton, 730
button
gazebo::common::MouseEvent, 514

ButtonCallback
gazebo::rendering::GUIOverlay, 347
Buttons
gazebo::common::MouseEvent, 513
buttons
gazebo::common::MouseEvent, 514

CATEGORY_COUNT
gazebo::sensors, 103
CFM
gazebo::physics::Joint, 386
COLLISION
gazebo::physics::Base, 140
COMVisual
gazebo::rendering::COMVisual, 220
COMVisual.hh, 956
COMVisualPtr
gazebo::rendering, 99
COR3_MAX
STLLoader.hh, 1097
CYLINDER_SHAPE
gazebo::physics::Base, 141
CallbackHelper
gazebo::transport::CallbackHelper, 158
CallbackHelper.hh, 943
CallbackHelperPtr
Transport, 76
CallbackHelperT
gazebo::transport::CallbackHelperT, 161
Camera
gazebo::rendering::Camera, 169
camera
gazebo::rendering::Camera, 186
gazebo::rendering::ViewController, 884
Camera.hh, 945
cameraCount
gazebo::sensors::GpuRaySensor, 338
cameraElem
gazebo::sensors::GpuRaySensor, 338
CameraPtr
gazebo::rendering, 99
CameraSensor
gazebo::sensors::CameraSensor, 190
CameraSensor.hh, 946
CameraSensor_V
gazebo::sensors, 102
CameraSensorPtr
gazebo::sensors, 102
CameraVisual
gazebo::rendering::CameraVisual, 193
CameraVisual.hh, 946
CameraVisualPtr
gazebo::rendering, 99
Cancel
gazebo::transport::Connection, 225
captureData
gazebo::rendering::Camera, 186
captureDataOnce
gazebo::rendering::Camera, 186
cegui.h, 947
Center
gazebo::common::Mesh, 477
gazebo::common::SubMesh, 768
cfm
gazebo::physics::SurfaceParams, 782
cgVisuals
gazebo::physics::Link, 437
chfov
gazebo::sensors::GpuRaySensor, 338
childLink
gazebo::physics::Joint, 398
children
gazebo::common::SkeletonNode, 746
gazebo::physics::Base, 148
childrenEnd
gazebo::physics::Base, 148
clamp
Math, 50
Classes for physics and dynamics, 41
create_world, 44
EntityTypename, 46
fini, 44
GZ_REGISTER_PHYSICS_ENGINE, 44
get_world, 44
init_world, 45
init_worlds, 45
load, 45
load_world, 45
load_worlds, 45
pause_world, 45
pause_worlds, 46
PhysicsFactoryFn, 44
remove_worlds, 46
run_world, 46
run_worlds, 46
stop_world, 46
stop_worlds, 46
Clear
gazebo::math::RotationSpline, 669
gazebo::math::Spline, 755
gazebo::physics::ContactManager, 240
gazebo::physics::World, 913
gazebo::rendering::DynamicLines, 268
gazebo::rendering::RTShaderSystem, 675
gazebo::rendering::Scene, 680
gazebo::rendering::SelectionObj, 697
sdf::Plugin, 594
clear_buffers

Transport, 77
ClearBuffers
gazebo::transport::TopicManager, 817
ClearElements
sdf::Element, 276
ClearGazeboPaths
gazebo::common::SystemPaths, 787
ClearModelPaths
gazebo::common::SystemPaths, 787
ClearOgrePaths
gazebo::common::SystemPaths, 787
ClearParent
gazebo::rendering::Visual, 891
ClearPluginPaths
gazebo::common::SystemPaths, 787
Clone
gazebo::rendering::Visual, 891
sdf::Element, 276
sdf::Param, 561
sdf::ParamT, 567
CloneVisual
gazebo::rendering::Scene, 680
coeffs
gazebo::math::Spline, 758
ColladaLoader
gazebo::common::ColladaLoader, 195
ColladaLoader.hh, 948
Collision
gazebo::physics::Collision, 198
Collision.hh, 949
collision1
gazebo::physics::Contact, 238
collision2
gazebo::physics::Contact, 238
Collision_V
gazebo::physics, 95
collisionParent
gazebo::physics::Shape, 723
CollisionPtr
gazebo::physics, 95
collisionPtr1
gazebo::physics::Contact, 238
collisionPtr2
gazebo::physics::Contact, 238
CollisionState
gazebo::physics::CollisionState, 206
CollisionState.hh, 951
Color
gazebo::common::Color, 211
Color.hh, 952
ColorErr
Common, 32
ColorMsg
Common, 33
Common, 27
add_search_path_suffix, 32
ColorErr, 32
ColorMsg, 33
DownloadDependencies, 33
find_file, 33
find_file_path, 33
GetDBConfig, 33
GetManifest, 34
GetModelConfig, 34
GetModelFile, 34
GetModelName, 34
GetModelPath, 35
GetModels, 35
GetURI, 35
gzclr_end, 31
gzclr_start, 31
gzdbg, 31
gzerr, 31
gzlog, 31
gzmsg, 31
gzthrow, 31
gzwarn, 32
HasModel, 35
Init, 36
IsInitialized, 36
Log, 36
MODEL_PLUGIN, 32
NullStream, 32
PixelFormatNames, 36
PluginType, 32
SENSOR_PLUGIN, 32
SYSTEM_PLUGIN, 32
SetQuiet, 36
VISUAL_PLUGIN, 32
WORLD_PLUGIN, 32
Common.hh, 953
common/Plugin.hh
GZ_REGISTER_MODEL_PLUGIN, 1050
GZ_REGISTER_SENSOR_PLUGIN, 1050
GZ_REGISTER_SYSTEM_PLUGIN, 1051
GZ_REGISTER_VISUAL_PLUGIN, 1051
GZ_REGISTER_WORLD_PLUGIN, 1051
CommonTypes.hh, 954
GAZEBO_DEPRECATED, 955
GAZEBO_FORCEINLINE, 955
NULL, 955
condition
gazebo::sensors::SimTimeEvent, 723
Connect
Events, 38
gazebo::transport::Connection, 225
ConnectAddEntity
gazebo::event::Events, 299

ConnectContact
gazebo::physics::Collision, 198
ConnectCreateEntity
gazebo::event::Events, 300
ConnectCreateScene
gazebo::rendering::Events, 295
ConnectDeleteEntity
gazebo::event::Events, 300
ConnectDiagTimerStart
gazebo::event::Events, 300
ConnectDiagTimerStop
gazebo::event::Events, 301
ConnectEnabled
gazebo::physics::Link, 425
ConnectJointUpdate
gazebo::physics::Joint, 387
ConnectNewDepthFrame
gazebo::rendering::DepthCamera, 255
ConnectNewImageFrame
gazebo::rendering::Camera, 170
ConnectNewLaserFrame
gazebo::rendering::GpuLaser, 326
gazebo::sensors::GpuRaySensor, 332
ConnectNewLaserScans
gazebo::physics::MultiRayShape, 530
ConnectNewRGBPointCloud
gazebo::rendering::DepthCamera, 256
ConnectPause
gazebo::event::Events, 301
ConnectPostRender
gazebo::event::Events, 301
ConnectPreRender
gazebo::event::Events, 301
ConnectPubToSub
gazebo::transport::TopicManager, 817
ConnectRemoveScene
gazebo::rendering::Events, 296
ConnectRender
gazebo::event::Events, 302
ConnectSetSelectedEntity
gazebo::event::Events, 302
ConnectStep
gazebo::event::Events, 302
ConnectStop
gazebo::event::Events, 303
ConnectSubToPub
gazebo::transport::TopicManager, 817
ConnectSubscribers
gazebo::transport::TopicManager, 817
ConnectToRemoteHost
gazebo::transport::ConnectionManager, 231
ConnectToShutdown
gazebo::transport::Connection, 225
ConnectUpdated
gazebo::sensors::Sensor, 701
ConnectWorldCreated
gazebo::event::Events, 303
ConnectWorldUpdateBegin
gazebo::event::Events, 303
ConnectWorldUpdateEnd
gazebo::event::Events, 303
ConnectWorldUpdateStart
gazebo::event::Events, 304
Connection
gazebo::event::Connection, 222
gazebo::transport::Connection, 225
Connection.hh, 956
HEADER_LENGTH, 958
Connection_V
gazebo::event, 87
ConnectionCount
Events, 39
ConnectionManager.hh, 958
ConnectionPtr
gazebo::event, 87
gazebo::transport, 105
ConnectionReadTask
gazebo::transport::ConnectionReadTask, 234
connections
gazebo::physics::Entity, 291
gazebo::rendering::Camera, 186
gazebo::sensors::Sensor, 706
Console.hh, 959
ConstUrdfLinkPtr
Gazebo_parser, 69
Contact
gazebo::physics::Contact, 237
Contact.hh, 961
MAX_COLLIDE_RETURNS, 962
MAX_CONTACT_JOINTS, 962
contactFiducial
gazebo::physics::RayShape, 652
contactLen
gazebo::physics::RayShape, 652
ContactManager
gazebo::physics::ContactManager, 240
contactManager
gazebo::physics::PhysicsEngine, 581
ContactManager.hh, 962
ContactPtr
gazebo::physics, 95
contactRetro
gazebo::physics::RayShape, 653
ContactSensor
gazebo::sensors::ContactSensor, 243
ContactSensor.hh, 963
ContactSensor_V
gazebo::sensors, 102

ContactSensorPtr
gazebo::sensors, 102
ContactVisual
gazebo::rendering::ContactVisual, 247
ContactVisual.hh, 964
ContactVisualPtr
gazebo::rendering, 99
control
gazebo::common::MouseEvent, 514
Conversions.hh, 964
Convert
gazebo::rendering::Conversions, 248, 249
Messages, 56–59
sdf::Converter, 250
ConvertPixelFormat
gazebo::common::Image, 362
Converter.hh, 965
CoordPoseSolve
gazebo::math::Pose, 599
CoordPositionAdd
gazebo::math::Pose, 599
CoordPositionSub
gazebo::math::Pose, 600
CoordRotationAdd
gazebo::math::Pose, 600
CoordRotationSub
gazebo::math::Pose, 600
Copy
sdf::Element, 277
copyChildren
sdf, 107
CopyNormals
gazebo::common::SubMesh, 768
CopyVertices
gazebo::common::SubMesh, 768
Correct
gazebo::math::Pose, 601
gazebo::math::Quaternion, 627
gazebo::math::Vector3, 859
count
gazebo::physics::Contact, 238
Create
gazebo::PluginT, 595
create_scene
Rendering, 67
create_sensor
Sensors, 72
create_world
Classes for physics and dynamics, 44
CreateBoolean
gazebo::common::MeshCSG, 482
CreateBox
gazebo::common::MeshManager, 485
CreateCamera
gazebo::common::MeshManager, 486
gazebo::rendering::Scene, 681
CreateCollision
gazebo::physics::PhysicsEngine, 572
CreateCone
gazebo::common::MeshManager, 486
CreateCylinder
gazebo::common::MeshManager, 486
CreateDepthCamera
gazebo::rendering::Scene, 681
CreateDepthTexture
gazebo::rendering::DepthCamera, 256
CreateDynamicLine
gazebo::rendering::Visual, 891
CreateGrid
gazebo::rendering::Scene, 681
CreateJoint
gazebo::physics::PhysicsEngine, 572
CreateKeyFrame
gazebo::common::NumericAnimation, 553
gazebo::common::PoseAnimation, 606
CreateLaserTexture
gazebo::rendering::GpuLaser, 326
CreateLink
gazebo::physics::PhysicsEngine, 572
CreatePlane
gazebo::common::MeshManager, 486, 487
gazebo::physics::PlaneShape, 591
CreateRenderTexture
gazebo::rendering::Camera, 170
CreateRequest
Messages, 59
CreateScene
gazebo::rendering::RenderEngine, 655
createScene
gazebo::rendering::Events, 296
CreateSensor
gazebo::sensors::SensorManager, 710
CreateShape
gazebo::physics::PhysicsEngine, 572
CreateSphere
gazebo::common::MeshManager, 487
CreateTube
gazebo::common::MeshManager, 487
CreateUserCamera
gazebo::rendering::Scene, 681
CreateVertexDeclaration
gazebo::rendering::DynamicLines, 268
gazebo::rendering::DynamicRenderable, 271
CreateWindow
gazebo::rendering::GUIOverlay, 347
gazebo::rendering::WindowManager, 907
Cross
gazebo::math::Vector2d, 840

gazebo::math::Vector2i, 849
gazebo::math::Vector3, 859
cvfov
gazebo::sensors::GpuRaySensor, 338
CylinderShape
gazebo::physics::CylinderShape, 252
CylinderShape.hh, 966
CylinderShapePtr
gazebo::physics, 95

d
gazebo::math::Plane, 589
DEFERRED
gazebo::rendering::RenderEngine, 655
DIAG_TIMER_LAP
Utility, 80
DIAG_TIMER_START
Utility, 80
DIAG_TIMER_STOP
Utility, 80
DIFFERENCE
gazebo::common::MeshCSG, 482
dampingCoefficient
gazebo::physics::Joint, 398
data
sdf::Plugin, 594
DebugPrint
gazebo::physics::PhysicsEngine, 573
DebugString
gazebo::physics::Contact, 237
DecCount
gazebo::transport::IOManager, 381
DecodeTopicName
gazebo::transport::Node, 537
defaultValue
sdf::ParamT, 568
defaultVpParams
gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperCg, 717
gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperGLSL, 719
Degree
gazebo::math::Angle, 120
DeleteDynamicLine
gazebo::rendering::Visual, 892
deleteEntity
gazebo::event::Events, 307
DepthCamera
gazebo::rendering::DepthCamera, 255
DepthCamera.hh, 967
DepthCameraPtr
gazebo::rendering, 99
DepthCameraSensor
gazebo::sensors::DepthCameraSensor, 259
DepthCameraSensor.hh, 968
DepthCameraSensor_V
gazebo::sensors, 102
DepthCameraSensorPtr
gazebo::sensors, 102
depthTarget
gazebo::rendering::DepthCamera, 258
depthTexture
gazebo::rendering::DepthCamera, 258
depthViewport
gazebo::rendering::DepthCamera, 258
depths
gazebo::physics::Contact, 238
description
sdf::Param, 564
Detach
gazebo::physics::Joint, 387
DetachAllStaticModels
gazebo::physics::Link, 425
DetachEntity
gazebo::rendering::RTShaderSystem, 675
DetachObjects
gazebo::rendering::Visual, 892
DetachStaticModel
gazebo::physics::Link, 425
gazebo::physics::Model, 493
DetachViewport
gazebo::rendering::RTShaderSystem, 675
DetachVisual
gazebo::rendering::Visual, 892
diagTimerStart
gazebo::event::Events, 307
diagTimerStop
gazebo::event::Events, 308
DiagnosticTimer
gazebo::util::DiagnosticTimer, 265
DiagnosticTimerPtr
gazebo::common, 86
gazebo::util, 105
Diagnostics.hh, 968
diffuse
gazebo::common::Material, 462
dirtyPose
gazebo::physics::Entity, 292
dirtyPoses
gazebo::physics::World, 921
DisableAllModels
gazebo::physics::World, 913
DisableTrackVisual
gazebo::rendering::Visual, 892
Disconnect
Events, 39, 40
gazebo::event::Event, 294
DisconnectAddEntity

gazebo::event::Events, 304
DisconnectContact
gazebo::physics::Collision, 199
DisconnectCreateEntity
gazebo::event::Events, 304
DisconnectCreateScene
gazebo::rendering::Events, 296
DisconnectDeleteEntity
gazebo::event::Events, 304
DisconnectDiagTimerStart
gazebo::event::Events, 305
DisconnectDiagTimerStop
gazebo::event::Events, 305
DisconnectEnabled
gazebo::physics::Link, 425
DisconnectJointUpdate
gazebo::physics::Joint, 387
DisconnectNewDepthFrame
gazebo::rendering::DepthCamera, 256
DisconnectNewImageFrame
gazebo::rendering::Camera, 171
DisconnectNewLaserFrame
gazebo::rendering::GpuLaser, 327
gazebo::sensors::GpuRaySensor, 332
DisconnectNewLaserScans
gazebo::physics::MultiRayShape, 531
DisconnectNewRGBPointCloud
gazebo::rendering::DepthCamera, 256
DisconnectPause
gazebo::event::Events, 305
DisconnectPostRender
gazebo::event::Events, 305
DisconnectPreRender
gazebo::event::Events, 305
DisconnectPubFromSub
gazebo::transport::TopicManager, 817
DisconnectRemoveScene
gazebo::rendering::Events, 296
DisconnectRender
gazebo::event::Events, 306
DisconnectSetSelectedEntity
gazebo::event::Events, 306
DisconnectShutdown
gazebo::transport::Connection, 226
DisconnectStep
gazebo::event::Events, 306
DisconnectStop
gazebo::event::Events, 306
DisconnectSubFromPub
gazebo::transport::TopicManager, 818
DisconnectUpdated
gazebo::sensors::Sensor, 701
DisconnectWorldCreated
gazebo::event::Events, 306
DisconnectWorldUpdateBegin
gazebo::event::Events, 307
DisconnectWorldUpdateEnd
gazebo::event::Events, 307
DisconnectWorldUpdateStart
gazebo::event::Events, 307
Distance
gazebo::math::Plane, 589
gazebo::math::Vector2d, 840
gazebo::math::Vector2i, 849
gazebo::math::Vector3, 859
gazebo::math::Vector4, 871
Dot
gazebo::math::Quaternion, 627
gazebo::math::Vector3, 860
Double
gazebo::common::Time, 796
DownloadDependencies
Common, 33
dragging
gazebo::common::MouseEvent, 514
DrawLine
gazebo::rendering::Scene, 682
dummyContext
gazebo::rendering::RenderEngine, 657
dummyDisplay
gazebo::rendering::RenderEngine, 657
dummyWindowId
gazebo::rendering::RenderEngine, 657
duration
gazebo::physics::TrajectoryInfo, 821
DynamicLines
gazebo::rendering::DynamicLines, 267
DynamicLines.hh, 969
DynamicLinesPtr
gazebo::rendering, 99
DynamicRenderable
gazebo::rendering::DynamicRenderable, 271
DynamicRenderable.hh, 970

ENTITY
gazebo::physics::Base, 140
ERP
gazebo::physics::Joint, 386
effortLimit
gazebo::physics::Joint, 398
Element
sdf::Element, 276
ElementPtr
sdf, 107
ElementPtr_V
sdf, 107
emissive
gazebo::common::Material, 462

Enable
gazebo::rendering::Grid, 342
EnableAllModels
gazebo::physics::World, 913
EnablePhysicsEngine
gazebo::physics::World, 913
EnableSaveFrame
gazebo::rendering::Camera, 171
EnableTrackVisual
gazebo::rendering::Visual, 892
EnableViewController
gazebo::rendering::UserCamera, 832
enabled
gazebo::rendering::ViewController, 884
EncodeTopicName
gazebo::transport::Node, 537
endTime
gazebo::physics::TrajectoryInfo, 821
EnqueueMsg
gazebo::transport::Connection, 226
Entity
gazebo::physics::Entity, 284
Entity.hh, 971
entityCreated
gazebo::event::Events, 308
EntityPtr
gazebo::physics, 95
EntityType
gazebo::physics::Base, 140
EntityTypename
Classes for physics and dynamics, 46
Equal
gazebo::math::Vector3, 860
equal
Math, 50
erp
gazebo::physics::SurfaceParams, 782
EulerToQuaternion
gazebo::math::Quaternion, 627
Event.hh, 973
eventConnections
gazebo::transport::ConnectionManager, 233
EventType
gazebo::common::MouseEvent, 513
Events, 38
∼EventT, 38
Connect, 38
ConnectionCount, 39
Disconnect, 39, 40
Events.hh, 974
Exception
gazebo::common::Exception, 319
Exception.hh, 974
execute
gazebo::transport::ConnectionReadTask, 234
gazebo::transport::PublishTask, 623

FACE_MAX
STLLoader.hh, 1097
FLAT
gazebo::common::Material, 456
FMAX
gazebo::physics::Joint, 386
FORWARD
gazebo::rendering::RenderEngine, 655
FPSViewController
gazebo::rendering::FPSViewController, 322
FPSViewController.hh, 976
FUDGE_FACTOR
gazebo::physics::Joint, 385
fakeAnchor
gazebo::physics::ScrewJoint, 694
gazebo::physics::SliderJoint, 749
far
gazebo::sensors::GpuRaySensor, 339
fdir1
gazebo::physics::SurfaceParams, 782
filename
gazebo::PluginT, 596
sdf::Plugin, 594
FillArrays
gazebo::common::Mesh, 477
gazebo::common::SubMesh, 769
FillHardwareBuffers
gazebo::rendering::DynamicLines, 268
gazebo::rendering::DynamicRenderable, 271
FillMsg
gazebo::physics::BoxShape, 155
gazebo::physics::Collision, 199
gazebo::physics::Contact, 237
gazebo::physics::CylinderShape, 252
gazebo::physics::HeightmapShape, 354
gazebo::physics::Joint, 387
gazebo::physics::Link, 426
gazebo::physics::Model, 493
gazebo::physics::MultiRayShape, 531
gazebo::physics::PlaneShape, 591
gazebo::physics::RayShape, 650
gazebo::physics::Shape, 722
gazebo::physics::SphereShape, 753
gazebo::physics::SurfaceParams, 781
gazebo::physics::TrimeshShape, 823
gazebo::rendering::Light, 414
gazebo::sensors::Sensor, 701
FillSDF
gazebo::physics::CollisionState, 206
gazebo::physics::JointState, 403
gazebo::physics::LinkState, 440

gazebo::physics::ModelState, 508
gazebo::physics::WorldState, 925
find_file
Common, 33
gazebo, 83
find_file_path
Common, 33
FindFile
gazebo::common::SystemPaths, 787
FindFileURI
gazebo::common::SystemPaths, 787
FindPublication
gazebo::transport::TopicManager, 818
Fini
gazebo::common::LogRecord, 449
gazebo::Master, 453
gazebo::physics::Actor, 114
gazebo::physics::Base, 142
gazebo::physics::Collision, 199
gazebo::physics::Entity, 284
gazebo::physics::Link, 426
gazebo::physics::Model, 493
gazebo::physics::PhysicsEngine, 573
gazebo::physics::World, 914
gazebo::rendering::Camera, 171
gazebo::rendering::DepthCamera, 257
gazebo::rendering::GpuLaser, 327
gazebo::rendering::RenderEngine, 656
gazebo::rendering::RTShaderSystem, 675
gazebo::rendering::UserCamera, 833
gazebo::rendering::Visual, 892
gazebo::rendering::WindowManager, 907
gazebo::sensors::CameraSensor, 190
gazebo::sensors::ContactSensor, 243
gazebo::sensors::DepthCameraSensor, 259
gazebo::sensors::GpuRaySensor, 332
gazebo::sensors::ImuSensor, 367
gazebo::sensors::MultiCameraSensor, 524
gazebo::sensors::RaySensor, 643
gazebo::sensors::RFIDSensor, 659
gazebo::sensors::RFIDTag, 661
gazebo::sensors::Sensor, 702
gazebo::sensors::SensorManager, 710
gazebo::Server, 715
gazebo::transport::ConnectionManager, 231
gazebo::transport::Node, 538
gazebo::transport::PublicationTransport, 617
gazebo::transport::TopicManager, 818
fini
Classes for physics and dynamics, 44
gazebo, 83
Rendering, 67
Sensors, 72
Transport, 77
Float
gazebo::common::Time, 796
FogFromSDF
Messages, 59
forceApplied
gazebo::physics::Joint, 398

g
gazebo::common::Color, 219
GAZEBO_DEPRECATED
CommonTypes.hh, 955
GAZEBO_FORCEINLINE
CommonTypes.hh, 955
GOURAUD
gazebo::common::Material, 456
GPtrArray
MeshCSG.hh, 1016
GUIFromSDF
Messages, 60
GUIOverlay
gazebo::rendering::GUIOverlay, 346
GUIOverlay.hh, 983
GUIPluginPtr
gazebo, 83
GZ_ALL_COLLIDE
PhysicsTypes.hh, 1044
GZ_ASSERT
Assert.hh, 936
GZ_DBL_MAX
Helpers.hh, 987
GZ_DBL_MIN
Helpers.hh, 987
GZ_DTOR
Angle.hh, 932
GZ_FIXED_COLLIDE
PhysicsTypes.hh, 1044
GZ_FLT_MAX
Helpers.hh, 987
GZ_FLT_MIN
Helpers.hh, 987
GZ_GHOST_COLLIDE
PhysicsTypes.hh, 1044
GZ_LOG_VERSION
LogRecord.hh, 1008
GZ_MODEL_DB_MANIFEST_FILENAME
ModelDatabase.hh, 1021
GZ_MODEL_MANIFEST_FILENAME
ModelDatabase.hh, 1021
GZ_NONE_COLLIDE
PhysicsTypes.hh, 1044
GZ_NORMALIZE
Angle.hh, 932
GZ_REGISTER_MODEL_PLUGIN
common/Plugin.hh, 1050

GZ_REGISTER_PHYSICS_ENGINE
Classes for physics and dynamics, 44
GZ_REGISTER_SENSOR_PLUGIN
common/Plugin.hh, 1050
GZ_REGISTER_STATIC_MSG
Messages, 56
GZ_REGISTER_STATIC_SENSOR
Sensors, 71
GZ_REGISTER_SYSTEM_PLUGIN
common/Plugin.hh, 1051
GZ_REGISTER_VISUAL_PLUGIN
common/Plugin.hh, 1051
GZ_REGISTER_WORLD_PLUGIN
common/Plugin.hh, 1051
GZ_RTOD
Angle.hh, 932
GZ_SENSOR_COLLIDE
PhysicsTypes.hh, 1044
GZ_VISIBILITY_ALL
RenderTypes.hh, 1067
GZ_VISIBILITY_GUI
RenderTypes.hh, 1067
GZ_VISIBILITY_NOT_SELECTABLE
RenderTypes.hh, 1067
GZ_VISIBILITY_SELECTION
RenderTypes.hh, 1067
gazebo, 81
add_plugin, 83
find_file, 83
fini, 83
GUIPluginPtr, 83
init, 83
load, 83
ModelPluginPtr, 83
print_version, 83
run, 83
SensorPluginPtr, 83
stop, 83
SystemPluginPtr, 83
VisualPluginPtr, 83
WorldPluginPtr, 83
gazebo.hh, 977
gazebo::Master, 452
∼Master, 453
Fini, 453
Init, 453
Master, 453
Run, 453
RunOnce, 453
RunThread, 453
Stop, 453
gazebo::ModelPlugin, 504
∼ModelPlugin, 505
Init, 505
Load, 505
ModelPlugin, 504
Reset, 505
gazebo::PluginT
Create, 595
filename, 596
GetFilename, 596
GetHandle, 596
GetType, 596
handle, 596
TPtr, 595
type, 596
gazebo::PluginT< T >, 594
gazebo::SensorPlugin, 712
∼SensorPlugin, 713
Init, 714
Load, 714
Reset, 714
SensorPlugin, 713
gazebo::Server, 714
∼Server, 715
Fini, 715
GetInitialized, 715
Init, 715
LoadFile, 715
LoadString, 715
ParseArgs, 715
PrintUsage, 715
Run, 715
Server, 715
SetParams, 715
Stop, 715
systemPluginsArgc, 715
systemPluginsArgv, 715
gazebo::SystemPlugin, 789
∼SystemPlugin, 790
Init, 790
Load, 790
Reset, 790
SystemPlugin, 790
gazebo::VisualPlugin, 905
Init, 905
Load, 906
Reset, 906
VisualPlugin, 905
gazebo::WorldPlugin, 921
∼WorldPlugin, 922
Init, 923
Load, 923
Reset, 923
WorldPlugin, 922
gazebo::common, 83
AnimationPtr, 86
DiagnosticTimerPtr, 86

NodeMap, 86
NodeMapIter, 86
NumericAnimationPtr, 86
Param_V, 86
PoseAnimationPtr, 86
RawNodeAnim, 86
RawNodeWeights, 86
RawSkeletonAnim, 86
StrStr_M, 86
gazebo::common::Animation, 125
~Animation, 127
AddTime, 127
Animation, 127
build, 128
GetKeyFrame, 127
GetKeyFrameCount, 127
GetKeyFramesAtTime, 127
GetLength, 128
GetTime, 128
KeyFrame_V, 126
keyFrames, 128
length, 129
loop, 129
name, 129
SetLength, 128
SetTime, 128
timePos, 129
gazebo::common::AssertionInternalError, 131
~AssertionInternalError, 133
AssertionInternalError, 132
gazebo::common::BVHLoader, 156
~BVHLoader, 156
BVHLoader, 156
Load, 156
gazebo::common::ColladaLoader, 194
~ColladaLoader, 195
ColladaLoader, 195
Load, 195
gazebo::common::Color, 208
~Color, 211
a, 219
ABGR, 211
ARGB, 211
b, 219
BGRA, 211
Black, 219
Blue, 219
Color, 211
g, 219
GetAsABGR, 212
GetAsARGB, 212
GetAsBGRA, 212
GetAsHSV, 212
GetAsRGBA, 212
GetAsYUV, 212
Green, 219
operator<<, 218
operator>>, 218
operator*, 213
operator*=, 213
operator+, 214
operator+=, 214
operator-, 214, 215
operator-=, 215
operator/, 215
operator/=, 216
operator=, 216
operator==, 216
operator[], 216
Purple, 219
r, 219
RGBA, 211
Red, 219
Reset, 217
Set, 217
SetFromABGR, 217
SetFromARGB, 217
SetFromBGRA, 217
SetFromHSV, 217
SetFromRGBA, 218
SetFromYUV, 218
White, 219
Yellow, 219
gazebo::common::Console, 235
gazebo::common::Exception, 318
~Exception, 319
Exception, 319
GetErrorFile, 319
GetErrorStr, 320
operator<<, 320
Print, 320
gazebo::common::Image, 360
~Image, 362
BAYER_GBRG8, 362
BAYER_GRBG8, 362
BAYER_RGGB8, 362
BAYER_RGGR8, 362
BGR_INT16, 361
BGR_INT32, 361
BGR_INT8, 361
BGRA_INT8, 361
ConvertPixelFormat, 362
GetAvgColor, 362
GetBPP, 363
GetData, 363
GetFilename, 363
GetHeight, 363
GetMaxColor, 363

GetPitch, 363
GetPixel, 363
GetPixelFormat, 364
GetRGBData, 364
GetWidth, 364
Image, 362
L_INT16, 361
L_INT8, 361
Load, 364
PIXEL_FORMAT_COUNT, 362
PixelFormat, 361
R_FLOAT16, 362
R_FLOAT32, 362
RGB_FLOAT16, 362
RGB_FLOAT32, 362
RGB_INT16, 361
RGB_INT32, 361
RGB_INT8, 361
RGBA_INT8, 361
Rescale, 364
SavePNG, 365
SetFromData, 365
UNKNOWN_PIXEL_FORMAT, 361
Valid, 365
gazebo::common::InternalError, 378
~InternalError, 380
InternalError, 379
gazebo::common::KeyFrame, 409
~KeyFrame, 409
GetTime, 410
KeyFrame, 409
time, 410
gazebo::common::LogPlay, 444
GetChunk, 445
GetChunkCount, 445
GetEncoding, 445
GetGazeboVersion, 445
GetLogVersion, 445
GetRandSeed, 445
IsOpen, 446
Open, 446
Step, 446
gazebo::common::LogRecord, 447
Add, 448
Fini, 449
GetBasePath, 449
GetEncoding, 449
GetFileSize, 449
GetFilename, 449
GetFirstUpdate, 450
GetPaused, 450
GetRunTime, 450
GetRunning, 450
Init, 450
Remove, 451
SetBasePath, 451
SetPaused, 451
Start, 451
Stop, 452
gazebo::common::Material, 453
~Material, 456
ADD, 456
ambient, 462
BLEND_COUNT, 456
BLINN, 456
BlendMode, 456
blendMode, 462
BlendModeStr, 462
diffuse, 462
emissive, 462
FLAT, 456
GOURAUD, 456
GetAmbient, 457
GetBlendFactors, 457
GetBlendMode, 457
GetDepthWrite, 457
GetDiffuse, 457
GetEmissive, 458
GetLighting, 458
GetName, 458
GetPointSize, 458
GetShadeMode, 458
GetShininess, 458
GetSpecular, 458
GetTextureImage, 459
GetTransparency, 459
MODULATE, 456
Material, 456, 457
name, 462
operator<<, 461
PHONG, 456
pointSize, 462
REPLACE, 456
SHADE_COUNT, 456
SetAmbient, 459
SetBlendFactors, 459
SetBlendMode, 459
SetDepthWrite, 460
SetDiffuse, 460
SetEmissive, 460
SetLighting, 460
SetPointSize, 460
SetShadeMode, 460
SetShininess, 460
SetSpecular, 461
SetTextureImage, 461
SetTransparency, 461
ShadeMode, 456

shadeMode, 462
ShadeModeStr, 462
shininess, 462
specular, 462
texImage, 462
transparency, 462
gazebo::common::Mesh, 475
~Mesh, 476
AddMaterial, 476
AddSubMesh, 477
Center, 477
FillArrays, 477
GenSphericalTexCoord, 477
GetAABB, 477
GetIndexCount, 478
GetMaterial, 478
GetMaterialCount, 478
GetMax, 478
GetMin, 478
GetName, 478
GetNormalCount, 478
GetPath, 479
GetSkeleton, 479
GetSubMesh, 479
GetSubMeshCount, 479
GetTexCoordCount, 480
GetVertexCount, 480
HasSkeleton, 480
Mesh, 476
RecalculateNormals, 480
Scale, 480
SetName, 480
SetPath, 480
SetScale, 481
SetSkeleton, 481
Translate, 481
gazebo::common::MeshCSG, 481
~MeshCSG, 482
BooleanOperation, 482
CreateBoolean, 482
DIFFERENCE, 482
INTERSECTION, 482
MeshCSG, 482
UNION, 482
gazebo::common::MeshLoader, 483
~MeshLoader, 483
Load, 484
MeshLoader, 483
gazebo::common::MeshManager, 484
AddMesh, 485
CreateBox, 485
CreateCamera, 486
CreateCone, 486
CreateCylinder, 486
CreatePlane, 486, 487
CreateSphere, 487
CreateTube, 487
GenSphericalTexCoord, 487
GetMesh, 488
GetMeshAABB, 488
HasMesh, 488
IsValidFilename, 488
Load, 488
gazebo::common::ModelDatabase, 502
gazebo::common::MouseEvent, 512
alt, 514
button, 514
Buttons, 513
buttons, 514
control, 514
dragging, 514
EventType, 513
LEFT, 513
MIDDLE, 513
MOVE, 514
MouseEvent, 514
moveScale, 514
NO_BUTTON, 513
NO_EVENT, 514
PRESS, 514
pos, 514
pressPos, 514
prevPos, 515
RELEASE, 514
RIGHT, 513
SCROLL, 514
scroll, 515
shift, 515
type, 515
gazebo::common::NodeAnimation, 542
~NodeAnimation, 543
AddKeyFrame, 543
GetFrameAt, 543
GetFrameCount, 544
GetKeyFrame, 544
GetLength, 544
GetName, 545
GetTimeAtX, 545
keyFrames, 545
length, 545
name, 545
NodeAnimation, 543
Scale, 545
SetName, 545
gazebo::common::NodeAssignment, 546
nodeIndex, 546
vertexIndex, 546
weight, 546

gazebo::common::NodeTransform, 547
~NodeTransform, 549
Get, 549
GetSID, 549
GetType, 549
MATRIX, 548
NodeTransform, 548
operator*, 549, 550
operator(), 549
PrintSource, 550
ROTATE, 548
RecalculateMatrix, 550
SCALE, 548
Set, 550
SetComponent, 550
SetSID, 550
SetSourceValues, 550, 551
SetType, 551
sid, 551
source, 551
TRANSLATE, 548
transform, 551
TransformType, 548
type, 551
gazebo::common::NumericAnimation, 552
~NumericAnimation, 553
CreateKeyFrame, 553
GetInterpolatedKeyFrame, 553
NumericAnimation, 552
gazebo::common::NumericKeyFrame, 553
~NumericKeyFrame, 554
GetValue, 555
NumericKeyFrame, 554
SetValue, 555
value, 555
gazebo::common::PID, 583
~PID, 584
GetCmd, 584
GetErrors, 585
Init, 585
operator=, 585
PID, 584
Reset, 585
SetCmd, 585
SetCmdMax, 586
SetCmdMin, 586
SetDGain, 586
SetIGain, 586
SetIMax, 586
SetIMin, 586
SetPGain, 587
Update, 587
gazebo::common::PoseAnimation, 605
~PoseAnimation, 606
BuildInterpolationSplines, 606
CreateKeyFrame, 606
GetInterpolatedKeyFrame, 606, 607
PoseAnimation, 606
gazebo::common::PoseKeyFrame, 607
~PoseKeyFrame, 608
GetRotation, 608
GetTranslation, 608
PoseKeyFrame, 608
rotate, 609
SetRotation, 609
SetTranslation, 609
translate, 609
gazebo::common::STLLoader, 762
~STLLoader, 763
Load, 763
STLLoader, 763
gazebo::common::Skeleton, 727
~Skeleton, 729
AddAnimation, 729
AddVertNodeWeight, 729
anims, 733
bindShapeTransform, 733
BuildNodeMap, 730
GetAnimation, 730
GetBindShapeTransform, 730
GetNodeByHandle, 730
GetNodeById, 730
GetNodeByName, 731
GetNodes, 731
GetNumAnimations, 731
GetNumJoints, 731
GetNumNodes, 731
GetNumVertNodeWeights, 731
GetRootNode, 732
GetVertNodeWeight, 732
nodes, 733
PrintTransforms, 732
rawNW, 733
root, 733
Scale, 732
SetBindShapeTransform, 732
SetNumVertAttached, 733
SetRootNode, 733
Skeleton, 729
gazebo::common::SkeletonAnimation, 734
~SkeletonAnimation, 735
AddKeyFrame, 735
animations, 737
GetLength, 735
GetName, 736
GetNodeCount, 736
GetNodePoseAt, 736
GetPoseAt, 736

GetPoseAtX, 737
HasNode, 737
length, 737
name, 738
Scale, 737
SetName, 737
SkeletonAnimation, 735
gazebo::common::SkeletonNode, 738
~SkeletonNode, 741
AddChild, 741
AddRawTransform, 741
children, 746
GetChild, 741
GetChildById, 741
GetChildByName, 742
GetChildCount, 742
GetHandle, 742
GetId, 742
GetInverseBindTransform, 742
GetModelTransform, 742
GetName, 743
GetNumRawTrans, 743
GetParent, 743
GetRawTransform, 743
GetRawTransforms, 743
GetTransform, 744
GetTransforms, 744
handle, 746
id, 746
initialTransform, 746
invBindTransform, 746
IsJoint, 744
IsRootNode, 744
JOINT, 740
modelTransform, 746
NODE, 740
name, 746
parent, 747
rawTransforms, 747
Reset, 744
SetHandle, 744
SetId, 744
SetInitialTransform, 745
SetInverseBindTransform, 745
SetModelTransform, 745
SetName, 745
SetParent, 745
SetTransform, 745
SetType, 746
SkeletonNode, 740, 741
SkeletonNodeType, 740
transform, 747
type, 747
UpdateChildrenTransforms, 746
gazebo::common::SubMesh, 764
~SubMesh, 767
AddIndex, 767
AddNodeAssignment, 767
AddNormal, 767
AddTexCoord, 767
AddVertex, 768
Center, 768
CopyNormals, 768
CopyVertices, 768
FillArrays, 769
GenSphericalTexCoord, 769
GetIndex, 769
GetIndexCount, 769
GetMaterialIndex, 769
GetMax, 769
GetMaxIndex, 769
GetMin, 769
GetName, 770
GetNodeAssignment, 770
GetNodeAssignmentsCount, 770
GetNormal, 770
GetNormalCount, 770
GetPrimitiveType, 770
GetTexCoord, 771
GetTexCoordCount, 771
GetVertex, 771
GetVertexCount, 771
GetVertexIndex, 771
HasVertex, 771
LINES, 766
LINESTRIPS, 766
POINTS, 766
PrimitiveType, 766
RecalculateNormals, 772
Scale, 772
SetIndexCount, 772
SetMaterialIndex, 772
SetName, 772
SetNormal, 772
SetNormalCount, 773
SetPrimitiveType, 773
SetScale, 773
SetSubMeshCenter, 773
SetTexCoord, 773
SetTexCoordCount, 773
SetVertex, 774
SetVertexCount, 774
SubMesh, 766
TRIANGLES, 766
TRIFANS, 766
TRISTRIPS, 766
Translate, 774
gazebo::common::SystemPaths, 784

AddGazeboPaths, 786
AddModelPaths, 786
AddOgrePaths, 786
AddPluginPaths, 786
AddSearchPathSuffix, 786
ClearGazeboPaths, 787
ClearModelPaths, 787
ClearOgrePaths, 787
ClearPluginPaths, 787
FindFile, 787
FindFileURI, 787
gazeboPathsFromEnv, 789
GetGazeboPaths, 787
GetLogPath, 788
GetModelPaths, 788
GetOgrePaths, 788
GetPluginPaths, 788
GetWorldPathExtension, 788
modelPathsFromEnv, 789
ogrePathsFromEnv, 789
pluginPathsFromEnv, 789
gazebo::common::Time, 791
∼Time, 795
Double, 796
Float, 796
GetWallTime, 796
GetWallTimeAsISOString, 796
MSleep, 797
MicToNano, 796
MilToNano, 796
NSleep, 797
nsec, 812
operator<, 804, 805
operator<<, 811
operator<=, 805, 806
operator>, 808, 809
operator>>, 811
operator>=, 809, 810
operator∗, 798, 799
operator∗=, 799, 800
operator+, 800
operator+=, 801
operator-, 801, 802
operator-=, 802, 803
operator/, 803
operator/=, 804
operator=, 806, 807
operator==, 807, 808
sec, 812
SecToNano, 810
Set, 810, 811
SetToWallTime, 811
Sleep, 811
Time, 795
Zero, 812
gazebo::common::Timer, 812
∼Timer, 813
GetElapsed, 814
GetRunning, 814
operator<<, 814
Start, 814
Stop, 814
Timer, 813
gazebo::common::UpdateInfo, 827
realTime, 827
simTime, 827
worldName, 828
gazebo::common::Video, 878
∼Video, 879
GetHeight, 879
GetNextFrame, 879
GetWidth, 879
Load, 879
Video, 879
gazebo::event, 86
Connection_V, 87
ConnectionPtr, 87
gazebo::event::Connection, 221
∼Connection, 222
Connection, 222
GetId, 222
gazebo::event::Event, 292
∼Event, 294
Disconnect, 294
gazebo::event::EventT
operator(), 312–314
Signal, 315–317
gazebo::event::EventT< T >, 309
gazebo::event::Events, 297
addEntity, 307
ConnectAddEntity, 299
ConnectCreateEntity, 300
ConnectDeleteEntity, 300
ConnectDiagTimerStart, 300
ConnectDiagTimerStop, 301
ConnectPause, 301
ConnectPostRender, 301
ConnectPreRender, 301
ConnectRender, 302
ConnectSetSelectedEntity, 302
ConnectStep, 302
ConnectStop, 303
ConnectWorldCreated, 303
ConnectWorldUpdateBegin, 303
ConnectWorldUpdateEnd, 303
ConnectWorldUpdateStart, 304
deleteEntity, 307
diagTimerStart, 307

diagTimerStop, 308
DisconnectAddEntity, 304
DisconnectCreateEntity, 304
DisconnectDeleteEntity, 304
DisconnectDiagTimerStart, 305
DisconnectDiagTimerStop, 305
DisconnectPause, 305
DisconnectPostRender, 305
DisconnectPreRender, 305
DisconnectRender, 306
DisconnectSetSelectedEntity, 306
DisconnectStep, 306
DisconnectStop, 306
DisconnectWorldCreated, 306
DisconnectWorldUpdateBegin, 307
DisconnectWorldUpdateEnd, 307
DisconnectWorldUpdateStart, 307
entityCreated, 308
pause, 308
postRender, 308
preRender, 308
render, 308
setSelectedEntity, 308
step, 308
stop, 308
worldCreated, 309
worldUpdateBegin, 309
worldUpdateEnd, 309
worldUpdateStart, 309
gazebo::math, 87
GeneratorType, 89
NRealGen, 89
NormalRealDist, 89
UIntGen, 89
URealGen, 89
UniformIntDist, 89
UniformRealDist, 89
gazebo::math::Angle, 118
∼Angle, 120
Angle, 119
Degree, 120
Normalize, 120
operator<, 122
operator<<, 124
operator<=, 123
operator>, 123
operator>>, 124
operator>=, 123
operator∗, 120
operator∗=, 121
operator+, 121
operator+=, 121
operator-, 121
operator-=, 122
operator/, 122
operator/=, 122
operator==, 123
Radian, 124
SetFromDegree, 124
SetFromRadian, 124
gazebo::math::Box, 148
∼Box, 150
Box, 150
GetCenter, 150
GetSize, 150
GetXLength, 150
GetYLength, 151
GetZLength, 151
max, 153
Merge, 151
min, 153
operator<<, 152
operator+, 151
operator+=, 151
operator-, 152
operator=, 152
operator==, 152
gazebo::math::Matrix3, 463
∼Matrix3, 464
m, 467
Matrix3, 464
operator<<, 467
operator∗, 465, 467
operator+, 465
operator-, 465
operator==, 465
operator[], 465, 466
SetCol, 466
SetFromAxes, 466
SetFromAxis, 466
gazebo::math::Matrix4, 467
∼Matrix4, 470
GetAsPose, 470
GetEulerRotation, 470
GetRotation, 470
GetTranslation, 470
IDENTITY, 474
Inverse, 470
IsAffine, 471
m, 474
Matrix4, 469
operator<<, 474
operator∗, 471
operator=, 471, 472
operator==, 472
operator[], 472
Set, 473
SetScale, 473

SetTranslate, 473
TransformAffine, 474
ZERO, 474
gazebo::math::Plane, 587
∼Plane, 588
d, 589
Distance, 589
normal, 589
operator=, 589
Plane, 588
Set, 589
size, 590
gazebo::math::Pose, 596
∼Pose, 599
CoordPoseSolve, 599
CoordPositionAdd, 599
CoordPositionSub, 600
CoordRotationAdd, 600
CoordRotationSub, 600
Correct, 601
GetInverse, 601
IsFinite, 601
operator<<, 604
operator>>, 604
operator∗, 601
operator+, 601
operator+=, 602
operator-, 602
operator-=, 602
operator==, 603
pos, 604
Pose, 598, 599
Reset, 603
rot, 605
RotatePositionAboutOrigin, 603
Round, 603
Set, 603, 604
Zero, 605
gazebo::math::Quaternion, 623
∼Quaternion, 627
Correct, 627
Dot, 627
EulerToQuaternion, 627
GetAsAxis, 628
GetAsEuler, 628
GetAsMatrix3, 628
GetAsMatrix4, 628
GetExp, 628
GetInverse, 628
GetLog, 629
GetPitch, 629
GetRoll, 629
GetXAxis, 629
GetYAxis, 629
GetYaw, 629
GetZAxis, 630
Invert, 630
IsFinite, 630
Normalize, 630
operator<<, 635
operator>>, 636
operator∗, 630, 631
operator∗=, 631
operator+, 631
operator+=, 632
operator-, 632
operator-=, 632
operator=, 632
operator==, 633
Quaternion, 626, 627
RotateVector, 633
RotateVectorReverse, 633
Round, 633
Scale, 634
Set, 634
SetFromAxis, 634
SetFromEuler, 634, 635
SetToIdentity, 635
Slerp, 635
Squad, 635
w, 636
x, 636
y, 636
z, 636
gazebo::math::Rand, 637
GetDblNormal, 637
GetDblUniform, 637
GetIntNormal, 637
GetIntUniform, 638
GetSeed, 638
SetSeed, 638
gazebo::math::RotationSpline, 668
∼RotationSpline, 669
AddPoint, 669
autoCalc, 672
Clear, 669
GetNumPoints, 670
GetPoint, 670
Interpolate, 670, 671
points, 672
RecalcTangents, 671
RotationSpline, 669
SetAutoCalculate, 671
tangents, 672
UpdatePoint, 671
gazebo::math::Spline, 754
∼Spline, 755
AddPoint, 755

autoCalc, 758
Clear, 755
coeffs, 758
GetPoint, 755
GetPointCount, 756
GetTangent, 756
GetTension, 756
Interpolate, 756
points, 758
RecalcTangents, 756
SetAutoCalculate, 757
SetTension, 757
Spline, 755
tangents, 758
tension, 758
UpdatePoint, 757
gazebo::math::Vector2d, 838
∼Vector2d, 840
Cross, 840
Distance, 840
IsFinite, 841
Normalize, 841
operator<<, 845
operator>>, 846
operator∗, 841
operator∗=, 842
operator+, 842
operator+=, 842
operator-, 843
operator-=, 843
operator/, 843
operator/=, 844
operator=, 844
operator==, 845
operator[], 845
Set, 845
Vector2d, 840
x, 846
y, 846
gazebo::math::Vector2i, 846
∼Vector2i, 848
Cross, 849
Distance, 849
IsFinite, 849
Normalize, 849
operator<<, 854
operator>>, 854
operator∗, 850
operator∗=, 850
operator+, 851
operator+=, 851
operator-, 851
operator-=, 851
operator/, 852
operator/=, 852, 853
operator=, 853
operator==, 854
operator[], 854
Set, 854
Vector2i, 848
x, 855
y, 855
gazebo::math::Vector3, 855
∼Vector3, 859
Correct, 859
Cross, 859
Distance, 859
Dot, 860
Equal, 860
GetAbs, 860
GetDistToLine, 860
GetLength, 860
GetMax, 861
GetMin, 861
GetNormal, 861
GetPerpendicular, 861
GetRounded, 861
GetSquaredLength, 861
GetSum, 862
IsFinite, 862
Normalize, 862
One, 868
operator<<, 867
operator>>, 868
operator∗, 862, 863, 867
operator∗=, 863
operator+, 863
operator+=, 864
operator-, 864
operator-=, 864
operator/, 864, 865
operator/=, 865
operator=, 865, 866
operator==, 866
operator[], 866
Round, 866
Set, 867
SetToMax, 867
SetToMin, 867
UnitX, 868
UnitY, 868
UnitZ, 868
Vector3, 858
x, 868
y, 868
z, 869
Zero, 869
gazebo::math::Vector4, 869

~Vector4, 871
Distance, 871
GetLength, 872
GetSquaredLength, 872
IsFinite, 872
Normalize, 872
operator<<, 877
operator>>, 877
operator*, 872, 873
operator*=, 873, 874
operator+, 874
operator+=, 874
operator-, 874
operator-=, 875
operator/, 875
operator/=, 875, 876
operator=, 876
operator==, 876
operator[], 877
Set, 877
Vector4, 871
w, 878
x, 878
y, 878
z, 878
gazebo::msgs, 89
MsgFactoryFn, 91
gazebo::msgs::MsgFactory, 521
GetMsgTypes, 521
NewMsg, 522
RegisterMsg, 522
gazebo::physics, 91
Actor_V, 95
ActorPtr, 95
Base_V, 95
BasePtr, 95
BoxShapePtr, 95
Collision_V, 95
CollisionPtr, 95
ContactPtr, 95
CylinderShapePtr, 95
EntityPtr, 95
HeightmapShapePtr, 95
InertialPtr, 95
Joint_V, 95
JointController_V, 95
JointControllerPtr, 95
JointPtr, 95
Link_V, 95
LinkPtr, 96
MeshShapePtr, 96
Model_V, 96
ModelPtr, 96
MultiRayShapePtr, 96
PhysicsEnginePtr, 96
RayShapePtr, 96
RoadPtr, 96
ShapePtr, 96
SphereShapePtr, 96
SurfaceParamsPtr, 96
WorldPtr, 96
gazebo::physics::Actor, 111
~Actor, 114
active, 116
Actor, 114
autoStart, 116
bonePosePub, 116
Fini, 114
GetSDF, 114
Init, 115
interpolateX, 116
IsActive, 115
lastPos, 116
lastScriptTime, 116
lastTraj, 116
Load, 115
loop, 116
mainLink, 116
mesh, 116
oldAction, 116
pathLength, 116
Play, 115
playStartTime, 117
prevFrameTime, 117
scriptLength, 117
skelAnimation, 117
skelNodesMap, 117
skeleton, 117
skinFile, 117
skinScale, 117
startDelay, 117
Stop, 115
trajInfo, 117
trajectories, 117
Update, 115
UpdateParameters, 115
visualName, 118
gazebo::physics::BallJoint
~BallJoint, 137
BallJoint, 136
GetAngleCount, 137
GetHighStop, 137
GetLowStop, 137
Load, 137
SetAxis, 137
SetHighStop, 137
SetLowStop, 137
gazebo::physics::BallJoint< T >, 135

gazebo::physics::Base, 137
~Base, 141
ACTOR, 140
AddChild, 141
AddType, 141
BALL_JOINT, 141
BASE, 140
BOX_SHAPE, 141
Base, 141
COLLISION, 140
CYLINDER_SHAPE, 141
children, 148
childrenEnd, 148
ENTITY, 140
EntityType, 140
Fini, 142
GetById, 142
GetByName, 142
GetChild, 142, 143
GetChildCount, 143
GetId, 143
GetName, 143
GetParent, 143
GetParentId, 143
GetSDF, 144
GetSaveable, 144
GetScopedName, 144
GetType, 144
GetWorld, 144
HEIGHTMAP_SHAPE, 141
HINGE2_JOINT, 141
HINGE_JOINT, 141
HasType, 144
Init, 145
IsSelected, 145
JOINT, 141
LIGHT, 141
LINK, 140
Load, 145
MAP_SHAPE, 141
MODEL, 140
MULTIRAY_SHAPE, 141
operator==, 145
PLANE_SHAPE, 141
parent, 148
Print, 146
RAY_SHAPE, 141
RemoveChild, 146
RemoveChildren, 146
Reset, 146
SCREW_JOINT, 141
SHAPE, 141
SLIDER_JOINT, 141
SPHERE_SHAPE, 141
sdf, 148
SetName, 147
SetParent, 147
SetSaveable, 147
SetSelected, 147
SetWorld, 147
TRIMESH_SHAPE, 141
UNIVERSAL_JOINT, 141
Update, 147
UpdateParameters, 148
VISUAL, 141
world, 148
gazebo::physics::BoxShape, 153
~BoxShape, 155
BoxShape, 155
FillMsg, 155
GetSize, 155
Init, 155
ProcessMsg, 155
SetSize, 155
gazebo::physics::Collision, 195
~Collision, 198
AddContact, 198
Collision, 198
ConnectContact, 198
DisconnectContact, 199
FillMsg, 199
Fini, 199
GetBoundingBox, 199
GetContactsEnabled, 199
GetLaserRetro, 199
GetLink, 199
GetModel, 200
GetRelativeAngularAccel, 200
GetRelativeAngularVel, 200
GetRelativeLinearAccel, 200
GetRelativeLinearVel, 200
GetShape, 200
GetShapeType, 201
GetState, 201
GetSurface, 201
GetWorldAngularAccel, 201
GetWorldAngularVel, 201
GetWorldLinearAccel, 202
GetWorldLinearVel, 202
Init, 202
IsPlaceable, 202
link, 204
Load, 202
placeable, 204
ProcessMsg, 203
SetCategoryBits, 203
SetCollideBits, 203
SetCollision, 203

SetContactsEnabled, 203
SetLaserRetro, 203
SetShape, 204
SetState, 204
shape, 204
UpdateParameters, 204
gazebo::physics::CollisionState, 205
~CollisionState, 206
CollisionState, 206
FillSDF, 206
GetPose, 207
IsZero, 207
Load, 207
operator<<, 208
operator+, 207
operator-, 207
operator=, 208
gazebo::physics::Contact, 236
~Contact, 237
collision1, 238
collision2, 238
collisionPtr1, 238
collisionPtr2, 238
Contact, 237
count, 238
DebugString, 237
depths, 238
FillMsg, 237
normals, 239
operator=, 237, 238
positions, 239
Reset, 238
time, 239
world, 239
wrench, 239
gazebo::physics::ContactManager, 239
~ContactManager, 240
Clear, 240
ContactManager, 240
GetContact, 240
GetContactCount, 240
GetContacts, 240
Init, 241
NewContact, 241
PublishContacts, 241
ResetCount, 241
gazebo::physics::CylinderShape, 250
~CylinderShape, 252
CylinderShape, 252
FillMsg, 252
GetLength, 252
GetRadius, 252
Init, 252
ProcessMsg, 253
SetLength, 253
SetRadius, 253
SetSize, 253
gazebo::physics::Entity, 281
~Entity, 284
animation, 291
animationConnection, 291
animationStartPose, 291
connections, 291
dirtyPose, 292
Entity, 284
Fini, 284
GetBoundingBox, 284
GetChildCollision, 284
GetChildLink, 285
GetCollisionBoundingBox, 285
GetDirtyPose, 285
GetInitialRelativePose, 285
GetNearestEntityBelow, 285
GetParentModel, 286
GetRelativeAngularAccel, 286
GetRelativeAngularVel, 286
GetRelativeLinearAccel, 286
GetRelativeLinearVel, 286
GetRelativePose, 287
GetWorldAngularAccel, 287
GetWorldAngularVel, 287
GetWorldLinearAccel, 287
GetWorldLinearVel, 287
GetWorldPose, 288
IsCanonicalLink, 288
IsStatic, 288
Load, 288
node, 292
OnPoseChange, 288
parentEntity, 292
PlaceOnEntity, 289
PlaceOnNearestEntityBelow, 289
poseMsg, 292
prevAnimationTime, 292
requestPub, 292
Reset, 289
SetAnimation, 289
SetCanonicalLink, 289
SetInitialRelativePose, 290
SetName, 290
SetRelativePose, 290
SetStatic, 290
SetWorldPose, 290
SetWorldTwist, 291
StopAnimation, 291
UpdateParameters, 291
visPub, 292
visualMsg, 292

gazebo::physics::Gripper, 344
~Gripper, 344
Gripper, 344
Init, 345
Load, 345
gazebo::physics::HeightmapShape, 352
~HeightmapShape, 354
FillMsg, 354
GetHeight, 354
GetMaxHeight, 354
GetMinHeight, 354
GetPos, 354
GetSize, 355
GetSubSampling, 355
GetURI, 355
GetVertexCount, 355
HeightmapShape, 353
heights, 356
img, 356
Init, 355
Load, 355
ProcessMsg, 356
scale, 356
subSampling, 356
vertSize, 356
gazebo::physics::Hinge2Joint
~Hinge2Joint, 357
GetAngleCount, 358
Hinge2Joint, 357
Load, 358
gazebo::physics::Hinge2Joint< T >, 356
gazebo::physics::HingeJoint
~HingeJoint, 359
GetAngleCount, 359
HingeJoint, 359
Init, 359
Load, 359
gazebo::physics::HingeJoint< T >, 358
gazebo::physics::Inertial, 369
~Inertial, 371
GetCoG, 371
GetIXX, 372
GetIXY, 372
GetIXZ, 372
GetIYY, 372
GetIYZ, 373
GetIZZ, 373
GetInertial, 372
GetMOI, 373
GetMass, 373
GetPose, 374
GetPrincipalMoments, 374
GetProductsofInertia, 374
Inertial, 371
Load, 374
operator<<, 378
operator+, 374
operator+=, 375
operator=, 375
ProcessMsg, 375
Reset, 375
Rotate, 375
SetCoG, 375, 376
SetIXX, 377
SetIXY, 377
SetIXZ, 377
SetIYY, 377
SetIYZ, 377
SetIZZ, 377
SetInertiaMatrix, 376
SetMOI, 378
SetMass, 378
UpdateParameters, 378
gazebo::physics::Joint, 381
~Joint, 386
anchorLink, 397
anchorPos, 397
anchorPose, 398
ApplyDamping, 386
applyDamping, 398
AreConnected, 386
Attach, 386
Attribute, 385
CFM, 386
childLink, 398
ConnectJointUpdate, 387
dampingCoefficient, 398
Detach, 387
DisconnectJointUpdate, 387
ERP, 386
effortLimit, 398
FMAX, 386
FUDGE_FACTOR, 385
FillMsg, 387
forceApplied, 398
GetAnchor, 387
GetAngle, 388
GetAngleCount, 388
GetAngleImpl, 388
GetAttribute, 388
GetChild, 388
GetEffortLimit, 389
GetForce, 389
GetForceTorque, 389, 390
GetGlobalAxis, 390
GetHighStop, 390
GetInertiaRatio, 391
GetJointLink, 391

GetLinkForce, 391
GetLinkTorque, 391
GetLocalAxis, 392
GetLowStop, 392
GetLowerLimit, 392
GetMaxForce, 392
GetParent, 393
GetUpperLimit, 393
GetVelocity, 393
GetVelocityLimit, 393
HI_STOP, 386
inertiaRatio, 398
Init, 394
Joint, 386
LO_STOP, 386
Load, 394
lowerLimit, 398
model, 398
parentLink, 398
Reset, 394
STOP_CFM, 386
STOP_ERP, 386
SUSPENSION_CFM, 385
SUSPENSION_ERP, 385
SetAnchor, 395
SetAngle, 395
SetAttribute, 395
SetAxis, 395
SetDamping, 395
SetForce, 396
SetHighStop, 396
SetLowStop, 396
SetMaxForce, 396
SetModel, 396
SetState, 397
SetVelocity, 397
Update, 397
UpdateParameters, 397
upperLimit, 399
useCFMDamping, 399
VEL, 386
velocityLimit, 399
gazebo::physics::JointController, 399
AddJoint, 400
JointController, 400
Reset, 400
SetJointPosition, 400
SetJointPositions, 400
Update, 401
gazebo::physics::JointState, 401
∼JointState, 403
FillSDF, 403
GetAngle, 403
GetAngleCount, 403
GetAngles, 403
IsZero, 403
JointState, 402
Load, 404
operator<<, 405
operator+, 404
operator-, 404
operator=, 404
gazebo::physics::JointWrench, 406
body1Force, 408
body1Torque, 408
body2Force, 408
body2Torque, 408
operator+, 407
operator-, 407
operator=, 408
gazebo::physics::Link, 418
∼Link, 423
AddChildJoint, 423
AddForce, 423
AddForceAtRelativePosition, 424
AddForceAtWorldPosition, 424
AddParentJoint, 424
AddRelativeForce, 424
AddRelativeTorque, 424
AddTorque, 425
angularAccel, 437
AttachStaticModel, 425
attachedModelsOffset, 437
cgVisuals, 437
ConnectEnabled, 425
DetachAllStaticModels, 425
DetachStaticModel, 425
DisconnectEnabled, 425
FillMsg, 426
Fini, 426
GetAngularDamping, 426
GetBoundingBox, 426
GetChildJointsLinks, 426
GetCollision, 426, 427
GetCollisionById, 427
GetCollisions, 427
GetEnabled, 427
GetGravityMode, 428
GetInertial, 428
GetKinematic, 428
GetLinearDamping, 428
GetModel, 428
GetParentJointsLinks, 428
GetRelativeAngularAccel, 429
GetRelativeAngularVel, 429
GetRelativeForce, 429
GetRelativeLinearAccel, 429
GetRelativeLinearVel, 429

GetRelativeTorque, 429
GetSelfCollide, 430
GetSensorCount, 430
GetSensorName, 430
GetWorldAngularAccel, 430
GetWorldCoGLinearVel, 431
GetWorldCoGPose, 431
GetWorldForce, 431
GetWorldLinearAccel, 431
GetWorldLinearVel, 431, 432
GetWorldTorque, 432
inertial, 437
Init, 432
linearAccel, 438
Link, 423
Load, 432
OnPoseChange, 432
ProcessMsg, 432
RemoveChildJoint, 433
RemoveParentJoint, 433
Reset, 433
ResetPhysicsStates, 433
SetAngularAccel, 434
SetAngularDamping, 434
SetAngularVel, 434
SetAutoDisable, 434
SetCollideMode, 434
SetEnabled, 434
SetForce, 435
SetGravityMode, 435
SetInertial, 435
SetKinematic, 435
SetLaserRetro, 435
SetLinearAccel, 435
SetLinearDamping, 436
SetLinearVel, 436
SetSelected, 436
SetSelfCollide, 436
SetState, 436
SetTorque, 436
Update, 437
UpdateMass, 437
UpdateParameters, 437
UpdateSurface, 437
visuals, 438
gazebo::physics::LinkState, 438
~LinkState, 440
FillSDF, 440
GetAcceleration, 440
GetCollisionState, 440, 441
GetCollisionStateCount, 441
GetCollisionStates, 441
GetPose, 442
GetVelocity, 442
GetWrench, 442
IsZero, 442
LinkState, 440
Load, 442
operator<<, 443
operator+, 442
operator-, 443
operator=, 443
gazebo::physics::Model, 489
~Model, 492
AttachStaticModel, 492
attachedModels, 502
attachedModelsOffset, 502
DetachStaticModel, 493
FillMsg, 493
Fini, 493
GetAutoDisable, 493
GetBoundingBox, 493
GetJoint, 494
GetJointController, 494
GetJointCount, 494
GetJoints, 494
GetLink, 494
GetLinkById, 495
GetLinks, 495
GetPluginCount, 495
GetRelativeAngularAccel, 495
GetRelativeAngularVel, 495
GetRelativeLinearAccel, 496
GetRelativeLinearVel, 496
GetSDF, 496
GetSensorCount, 496
GetWorldAngularAccel, 496
GetWorldAngularVel, 497
GetWorldLinearAccel, 497
GetWorldLinearVel, 497
Init, 497
Load, 497
LoadJoints, 498
LoadPlugins, 498
Model, 492
OnPoseChange, 498
ProcessMsg, 498
RemoveChild, 498
Reset, 498
SetAngularAccel, 498
SetAngularVel, 499
SetAutoDisable, 499
SetCollideMode, 499
SetEnabled, 499
SetGravityMode, 499
SetJointAnimation, 499
SetJointPosition, 500
SetJointPositions, 500

SetLaserRetro, 500
SetLinearAccel, 500
SetLinearVel, 501
SetLinkWorldPose, 501
SetState, 501
StopAnimation, 501
Update, 501
UpdateParameters, 502
gazebo::physics::ModelState, 505
~ModelState, 508
FillSDF, 508
GetJointState, 508
GetJointStateCount, 509
GetJointStates, 509
GetLinkState, 509
GetLinkStateCount, 510
GetLinkStates, 510
GetPose, 510
HasJointState, 510
HasLinkState, 510
IsZero, 511
Load, 511
ModelState, 507
operator<<, 512
operator+, 511
operator-, 511
operator=, 512
gazebo::physics::MultiRayShape, 527
~MultiRayShape, 530
AddRay, 530
ConnectNewLaserScans, 530
DisconnectNewLaserScans, 531
FillMsg, 531
GetFiducial, 531
GetMaxAngle, 531
GetMaxRange, 531
GetMinAngle, 532
GetMinRange, 532
GetRange, 532
GetResRange, 532
GetRetro, 532
GetSampleCount, 533
GetScanResolution, 533
GetVerticalMaxAngle, 533
GetVerticalMinAngle, 533
GetVerticalSampleCount, 533
GetVerticalScanResolution, 533
horzElem, 534
Init, 533
MultiRayShape, 530
newLaserScans, 534
offset, 534
ProcessMsg, 534
rangeElem, 534
rayElem, 534
rays, 535
scanElem, 535
Update, 534
UpdateRays, 534
vertElem, 535
gazebo::physics::PhysicsEngine, 568
~PhysicsEngine, 571
contactManager, 581
CreateCollision, 572
CreateJoint, 572
CreateLink, 572
CreateShape, 572
DebugPrint, 573
Fini, 573
GetAutoDisableFlag, 573
GetContactManager, 573
GetContactMaxCorrectingVel, 573
GetContactSurfaceLayer, 573
GetGravity, 573
GetMaxContacts, 574
GetMaxStepSize, 574
GetParam, 574
GetPhysicsUpdateMutex, 574
GetRealTimeUpdateRate, 574
GetSORPGSIters, 575
GetSORPGSPreconIters, 575
GetSORPGSW, 575
GetStepTime, 575
GetTargetRealTimeFactor, 575
GetType, 575
GetUpdatePeriod, 576
GetUpdateRate, 576
GetWorldCFM, 576
GetWorldERP, 576
Init, 576
InitForThread, 576
Load, 577
maxStepSize, 581
node, 581
OnPhysicsMsg, 577
OnRequest, 577
PhysicsEngine, 571
physicsSub, 581
physicsUpdateMutex, 581
realTimeUpdateRate, 581
requestSub, 581
Reset, 577
responsePub, 581
sdf, 581
SetAutoDisableFlag, 577
SetContactMaxCorrectingVel, 577
SetContactSurfaceLayer, 578
SetGravity, 578

SetMaxContacts, 578
SetMaxStepSize, 578
SetParam, 578
SetRealTimeUpdateRate, 578
SetSORPGSIters, 579
SetSORPGSPreconIters, 579
SetSORPGSW, 579
SetSeed, 579
SetStepTime, 579
SetTargetRealTimeFactor, 580
SetUpdateRate, 580
SetWorldCFM, 580
SetWorldERP, 580
targetRealTimeFactor, 581
UpdateCollision, 580
UpdatePhysics, 580
world, 581
gazebo::physics::PhysicsFactory, 582
IsRegistered, 582
NewPhysicsEngine, 582
RegisterAll, 583
RegisterPhysicsEngine, 583
gazebo::physics::PlaneShape, 590
∼PlaneShape, 591
CreatePlane, 591
FillMsg, 591
GetNormal, 592
GetSize, 592
Init, 592
PlaneShape, 591
ProcessMsg, 592
SetAltitude, 592
SetNormal, 592
SetSize, 593
gazebo::physics::RayShape, 647
∼RayShape, 650
contactFiducial, 652
contactLen, 652
contactRetro, 653
FillMsg, 650
GetFiducial, 650
GetGlobalPoints, 650
GetIntersection, 650
GetLength, 651
GetRelativePoints, 651
GetRetro, 651
globalEndPos, 653
globalStartPos, 653
Init, 651
ProcessMsg, 651
RayShape, 649, 650
relativeEndPos, 653
relativeStartPos, 653
SetFiducial, 651
SetLength, 652
SetPoints, 652
SetRetro, 652
Update, 652
gazebo::physics::Road, 665
∼Road, 666
Init, 667
Load, 667
Road, 666
gazebo::physics::ScrewJoint
∼ScrewJoint, 693
fakeAnchor, 694
GetAnchor, 693
GetAngleCount, 693
GetThreadPitch, 693
Load, 693
ScrewJoint, 693
SetAnchor, 694
SetThreadPitch, 694
threadPitch, 694
gazebo::physics::ScrewJoint< T >, 691
gazebo::physics::Shape, 720
∼Shape, 722
collisionParent, 723
FillMsg, 722
Init, 722
ProcessMsg, 722
Shape, 721
gazebo::physics::SliderJoint
∼SliderJoint, 748
fakeAnchor, 749
GetAnchor, 748
GetAngleCount, 749
Load, 749
SetAnchor, 749
SliderJoint, 748
gazebo::physics::SliderJoint< T >, 747
gazebo::physics::SphereShape, 751
∼SphereShape, 753
FillMsg, 753
GetRadius, 753
Init, 753
ProcessMsg, 753
SetRadius, 753
SphereShape, 753
gazebo::physics::State, 758
∼State, 760
GetName, 760
GetRealTime, 760
GetSimTime, 761
GetWallTime, 761
Load, 761
name, 762
operator-, 761

operator=, 761
realTime, 762
SetName, 762
simTime, 762
State, 760
wallTime, 762
gazebo::physics::SurfaceParams, 780
~SurfaceParams, 781
bounce, 782
bounceThreshold, 782
cfm, 782
erp, 782
fdir1, 782
FillMsg, 781
kd, 783
kp, 783
Load, 781
maxVel, 783
minDepth, 783
mu1, 783
mu2, 783
ProcessMsg, 782
slip1, 784
slip2, 784
SurfaceParams, 781
gazebo::physics::TrajectoryInfo, 821
duration, 821
endTime, 821
id, 821
startTime, 821
translated, 821
type, 821
gazebo::physics::TrimeshShape, 822
~TrimeshShape, 823
FillMsg, 823
GetFilename, 824
GetMeshURI, 824
GetSize, 824
Init, 824
mesh, 825
ProcessMsg, 824
SetFilename, 824
SetMesh, 825
SetScale, 825
submesh, 825
TrimeshShape, 823
Update, 825
gazebo::physics::UniversalJoint
~UniversalJoint, 827
GetAngleCount, 827
Load, 827
UniversalJoint, 826
gazebo::physics::UniversalJoint< T >, 826
gazebo::physics::World, 910
~World, 913
Clear, 913
dirtyPoses, 921
DisableAllModels, 913
EnableAllModels, 913
EnablePhysicsEngine, 913
Fini, 914
GetByName, 914
GetEnablePhysicsEngine, 914
GetEntity, 914
GetEntityBelowPoint, 914
GetModel, 915
GetModelBelowPoint, 915
GetModelCount, 916
GetModels, 916
GetName, 916
GetPauseTime, 916
GetPhysicsEngine, 916
GetRealTime, 916
GetSelectedEntity, 917
GetSetWorldPoseMutex, 917
GetSimTime, 917
GetStartTime, 917
Init, 917
InsertModelFile, 917
InsertModelSDF, 918
InsertModelString, 918
IsLoaded, 918
IsPaused, 918
Load, 918
LoadPlugin, 919
PrintEntityTree, 919
PublishModelPose, 919
RemovePlugin, 919
Reset, 919
ResetEntities, 919
ResetTime, 920
Run, 920
Save, 920
SetPaused, 920
SetSimTime, 920
SetState, 920
StepWorld, 920
Stop, 921
StripWorldName, 921
UpdateStateSDF, 921
World, 913
gazebo::physics::WorldState, 923
~WorldState, 925
FillSDF, 925
GetModelState, 925
GetModelStateCount, 926
GetModelStates, 926
HasModelState, 926

IsZero, 926
Load, 927
operator<<, 928
operator+, 927
operator-, 927
operator=, 927
SetWorld, 928
WorldState, 924, 925
gazebo::rendering, 96
ArrowVisualPtr, 99
AxisVisualPtr, 99
COMVisualPtr, 99
CameraPtr, 99
CameraVisualPtr, 99
ContactVisualPtr, 99
DepthCameraPtr, 99
DynamicLinesPtr, 99
GpuLaserPtr, 99
JointVisualPtr, 99
LaserVisualPtr, 99
LightPtr, 99
RENDERING_LINE_LIST, 99
RENDERING_LINE_STRIP, 99
RENDERING_MESH_RESOURCE, 100
RENDERING_POINT_LIST, 99
RENDERING_TRIANGLE_FAN, 100
RENDERING_TRIANGLE_LIST, 99
RENDERING_TRIANGLE_STRIP, 100
RFIDTagVisualPtr, 99
RFIDVisualPtr, 99
RenderOpType, 99
ScenePtr, 99
UserCameraPtr, 99
VisualPtr, 99
gazebo::rendering::ArrowVisual, 129
∼ArrowVisual, 131
ArrowVisual, 130
Load, 131
ShowRotation, 131
gazebo::rendering::AxisVisual, 133
∼AxisVisual, 134
AxisVisual, 134
Load, 134
ScaleXAxis, 134
ScaleYAxis, 134
ScaleZAxis, 135
SetAxisMaterial, 135
ShowRotation, 135
gazebo::rendering::COMVisual, 219
∼COMVisual, 221
COMVisual, 220
Load, 221
gazebo::rendering::Camera, 162
∼Camera, 169
animState, 185
AnimationComplete, 169
AttachToVisual, 169
AttachToVisualImpl, 169, 170
bayerFrameBuffer, 186
Camera, 169
camera, 186
captureData, 186
captureDataOnce, 186
ConnectNewImageFrame, 170
connections, 186
CreateRenderTexture, 170
DisconnectNewImageFrame, 171
EnableSaveFrame, 171
Fini, 171
GetAspectRatio, 171
GetAvgFPS, 171
GetCameraToViewportRay, 171
GetDirection, 172
GetFarClip, 172
GetFrameFilename, 172
GetHFOV, 172
GetImageByteSize, 172
GetImageData, 173
GetImageDepth, 173
GetImageFormat, 173
GetImageHeight, 173
GetImageWidth, 174
GetInitialized, 174
GetLastRenderWallTime, 174
GetName, 174
GetNearClip, 174
GetOgreCamera, 174
GetPitchNode, 175
GetRenderRate, 175
GetRenderTexture, 175
GetRight, 175
GetScene, 175
GetSceneNode, 175
GetScreenshotPath, 176
GetTextureHeight, 176
GetTextureWidth, 176
GetTriangleCount, 176
GetUp, 176
GetVFOV, 176
GetViewport, 177
GetViewportHeight, 177
GetViewportWidth, 177
GetWindowId, 177
GetWorldPointOnPlane, 177
GetWorldPose, 178
GetWorldPosition, 178
GetWorldRotation, 178
GetZValue, 178

imageFormat, 186
imageHeight, 186
imageWidth, 186
Init, 178
initialized, 186
IsAnimating, 178
IsInitialized, 179
IsVisible, 179
lastRenderWallTime, 186
Load, 179
MoveToPosition, 180
MoveToPositions, 180
name, 186
newData, 187
newImageFrame, 187
onAnimationComplete, 187
pitchNode, 187
PostRender, 180
prevAnimTime, 187
Render, 180
RenderImpl, 180
renderTarget, 187
renderTexture, 187
requests, 187
RotatePitch, 180
RotateYaw, 181
saveCount, 187
SaveFrame, 181
saveFrameBuffer, 187
scene, 188
sceneNode, 188
screenshotPath, 188
sdf, 188
SetAspectRatio, 181
SetCaptureData, 182
SetCaptureDataOnce, 182
SetClipDist, 182
SetHFOV, 182
SetImageHeight, 182
SetImageSize, 182
SetImageWidth, 183
SetName, 183
SetRenderRate, 183
SetRenderTarget, 183
SetSaveFramePathname, 183
SetScene, 183
SetSceneNode, 183
SetWindowId, 184
SetWorldPose, 184
SetWorldPosition, 184
SetWorldRotation, 184
ShowWireframe, 184
textureHeight, 188
textureWidth, 188
ToggleShowWireframe, 184
TrackVisual, 184
TrackVisualImpl, 185
Translate, 185
Update, 185
viewport, 188
windowId, 188
gazebo::rendering::CameraVisual, 193
~CameraVisual, 194
CameraVisual, 193
Load, 194
gazebo::rendering::ContactVisual, 246
~ContactVisual, 247
ContactVisual, 247
SetEnabled, 247
gazebo::rendering::Conversions, 247
Convert, 248, 249
gazebo::rendering::DepthCamera, 253
~DepthCamera, 255
ConnectNewDepthFrame, 255
ConnectNewRGBPointCloud, 256
CreateDepthTexture, 256
DepthCamera, 255
depthTarget, 258
depthTexture, 258
depthViewport, 258
DisconnectNewDepthFrame, 256
DisconnectNewRGBPointCloud, 256
Fini, 257
GetDepthData, 257
Init, 257
Load, 257
PostRender, 257
SetDepthTarget, 257
gazebo::rendering::DynamicLines, 266
~DynamicLines, 267
AddPoint, 268
Clear, 268
CreateVertexDeclaration, 268
DynamicLines, 267
FillHardwareBuffers, 268
GetMovableType, 268
getMovableType, 268
GetPoint, 269
GetPointCount, 269
SetPoint, 269
Update, 269
gazebo::rendering::DynamicRenderable, 269
~DynamicRenderable, 271
CreateVertexDeclaration, 271
DynamicRenderable, 271
FillHardwareBuffers, 271
getBoundingRadius, 271
GetOperationType, 272

getSquaredViewDepth, 272
indexBufferCapacity, 273
Init, 272
PrepareHardwareBuffers, 272
SetOperationType, 273
vertexBufferCapacity, 273
gazebo::rendering::Events, 295
ConnectCreateScene, 295
ConnectRemoveScene, 296
createScene, 296
DisconnectCreateScene, 296
DisconnectRemoveScene, 296
removeScene, 296
gazebo::rendering::FPSViewController, 320
∼FPSViewController, 322
FPSViewController, 322
GetTypeString, 322
HandleKeyPressEvent, 322
HandleKeyReleaseEvent, 322
HandleMouseEvent, 322
Init, 323
Update, 323
gazebo::rendering::GUIOverlay, 345
∼GUIOverlay, 346
AttachCameraToImage, 346
ButtonCallback, 347
CreateWindow, 347
GUIOverlay, 346
HandleKeyPressEvent, 347
HandleKeyReleaseEvent, 348
HandleMouseEvent, 348
Hide, 348
Init, 348
IsInitialized, 348
LoadLayout, 348
Resize, 349
Show, 349
Update, 349
gazebo::rendering::GpuLaser, 324
∼GpuLaser, 326
ConnectNewLaserFrame, 326
CreateLaserTexture, 326
DisconnectNewLaserFrame, 327
Fini, 327
GetLaserData, 327
GpuLaser, 326
Init, 327
Load, 327
notifyRenderSingleObject, 327
PostRender, 327
SetParentSensor, 327
SetRangeCount, 328
gazebo::rendering::Grid, 340
∼Grid, 341
Enable, 342
GetCellCount, 342
GetCellLength, 342
GetColor, 342
GetHeight, 342
GetLineWidth, 342
GetSceneNode, 342
Grid, 341
Init, 343
SetCellCount, 343
SetCellLength, 343
SetColor, 343
SetHeight, 343
SetLineWidth, 343
SetUserData, 344
gazebo::rendering::GzTerrainMatGen, 349
∼GzTerrainMatGen, 350
GzTerrainMatGen, 350
gazebo::rendering::GzTerrainMatGen::SM2Profile, 749
∼SM2Profile, 751
addTechnique, 751
generate, 751
generateForCompositeMap, 751
SM2Profile, 751
UpdateParams, 751
UpdateParamsForCompositeMap, 751
gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperCg, 715
defaultVpParams, 717
generateFragmentProgram, 717
generateVertexProgram, 717
generateVertexProgramSource, 717
generateVpDynamicShadows, 717
generateVpDynamicShadowsParams, 717
generateVpFooter, 717
generateVpHeader, 717
gazebo::rendering::GzTerrainMatGen::SM2Profile::-
ShaderHelperGLSL, 717
defaultVpParams, 719
generateFpDynamicShadows, 719
generateFpDynamicShadowsHelpers, 719
generateFpDynamicShadowsParams, 719
generateFpFooter, 719
generateFpHeader, 719
generateFpLayer, 719
generateFragmentProgram, 719
generateFragmentProgramSource, 719
generateVertexProgram, 720
generateVertexProgramSource, 720
generateVpDynamicShadows, 720
generateVpDynamicShadowsParams, 720
generateVpFooter, 720
generateVpHeader, 720
updateParams, 720

updateVpParams, 720
gazebo::rendering::Heightmap, 350
~Heightmap, 351
GetHeight, 351
GetOgreTerrain, 351
Heightmap, 351
Load, 351
LoadFromMsg, 351
gazebo::rendering::JointVisual, 405
~JointVisual, 406
JointVisual, 406
Load, 406
gazebo::rendering::LaserVisual, 410
~LaserVisual, 412
LaserVisual, 411
SetEmissive, 412
gazebo::rendering::Light, 412
~Light, 414
FillMsg, 414
GetDiffuseColor, 414
GetDirection, 414
GetName, 414
GetPosition, 414
GetSpecularColor, 415
GetType, 415
Light, 414
Load, 415
LoadFromMsg, 415
OnPoseChange, 415
SetAttenuation, 415
SetCastShadows, 416
SetDiffuseColor, 416
SetDirection, 416
SetLightType, 416
SetName, 416
SetPosition, 416
SetRange, 417
SetSelected, 417
SetSpecularColor, 417
SetSpotFalloff, 417
SetSpotInnerAngle, 417
SetSpotOuterAngle, 417
ShowVisual, 418
ToggleShowVisual, 418
UpdateFromMsg, 418
gazebo::rendering::MovableText, 515
~MovableText, 517
_setupGeometry, 518
_updateColors, 518
GetAABB, 518
GetBaseline, 518
getBoundingRadius, 518
GetCharHeight, 518
GetColor, 518
GetFont, 518
getLights, 518
getMaterial, 518
getRenderOperation, 518
GetShowOnTop, 519
GetSpaceWidth, 519
getSquaredViewDepth, 519
GetText, 519
getWorldTransforms, 519
H_CENTER, 517
H_LEFT, 517
HorizAlign, 517
Load, 519
MovableText, 517
SetBaseline, 519
SetCharHeight, 520
SetColor, 520
SetFontName, 520
SetShowOnTop, 520
SetSpaceWidth, 520
SetText, 520
SetTextAlignment, 521
Update, 521
V_ABOVE, 517
V_BELOW, 517
VertAlign, 517
visitRenderables, 521
gazebo::rendering::OrbitViewController, 555
~OrbitViewController, 557
GetFocalPoint, 557
GetTypeString, 557
HandleKeyPressEvent, 557
HandleKeyReleaseEvent, 557
HandleMouseEvent, 558
Init, 558
OrbitViewController, 557
SetDistance, 558
SetFocalPoint, 558
Update, 559
gazebo::rendering::Projector, 609
~Projector, 610
GetParent, 610
Load, 610, 611
Projector, 610
SetEnabled, 611
SetTexture, 611
Toggle, 611
gazebo::rendering::RFIDTagVisual, 662
~RFIDTagVisual, 664
RFIDTagVisual, 663
gazebo::rendering::RFIDVisual, 664
~RFIDVisual, 665
RFIDVisual, 665
gazebo::rendering::RTShaderSystem, 672

AddScene, 674
ApplyShadows, 674
AttachEntity, 674
AttachViewport, 674
Clear, 675
DetachEntity, 675
DetachViewport, 675
Fini, 675
GenerateShaders, 675
GetPSSMShadowCameraSetup, 675
Init, 675
LightingModel, 674
RemoveScene, 675
RemoveShadows, 676
SSLM_NormalMapLightingObjectSpace, 674
SSLM_NormalMapLightingTangentSpace, 674
SSLM_PerPixelLighting, 674
SSLM_PerVertexLighting, 674
SetPerPixelLighting, 676
UpdateShaders, 676
gazebo::rendering::RenderEngine, 653
AddResourcePath, 655
CreateScene, 655
DEFERRED, 655
dummyContext, 657
dummyDisplay, 657
dummyWindowId, 657
FORWARD, 655
Fini, 656
GetRenderPathType, 656
GetScene, 656
GetSceneCount, 656
Init, 657
Load, 657
NONE, 655
RENDER_PATH_COUNT, 655
RemoveScene, 657
RenderPathType, 655
root, 657
VERTEX, 655
gazebo::rendering::Road2d, 667
∼Road2d, 668
Load, 668
Road2d, 668
gazebo::rendering::Scene, 676
∼Scene, 680
AddVisual, 680
Clear, 680
CloneVisual, 680
CreateCamera, 681
CreateDepthCamera, 681
CreateGrid, 681
CreateUserCamera, 681
DrawLine, 682
GetAmbientColor, 682
GetBackgroundColor, 682
GetCamera, 682
GetCameraCount, 683
GetFirstContact, 683
GetGrid, 683
GetGridCount, 684
GetHeightBelowPoint, 684
GetHeightmap, 684
GetId, 684
GetIdString, 684
GetInitialized, 684
GetLight, 684, 685
GetLightCount, 685
GetManager, 685
GetModelVisualAt, 685
GetName, 686
GetSelectedVisual, 686
GetShadowsEnabled, 686
GetUserCamera, 686
GetUserCameraCount, 686
GetVisual, 686
GetVisualAt, 687
GetVisualBelow, 687
GetVisualsBelowPoint, 687
GetWorldVisual, 688
Init, 688
Load, 688
PreRender, 688
PrintSceneGraph, 688
RemoveVisual, 688
Scene, 680
SelectVisual, 688
SetAmbientColor, 689
SetBackgroundColor, 689
SetFog, 689
SetGrid, 689
SetShadowsEnabled, 689
SetTransparent, 690
SetVisible, 690
SetWireframe, 690
ShowCOMs, 690
ShowCollisions, 690
ShowContacts, 690
ShowJoints, 691
skyx, 691
SnapVisualToNearestBelow, 691
StripSceneName, 691
gazebo::rendering::SelectionObj, 696
∼SelectionObj, 697
Attach, 697
Clear, 697
GetVisualName, 697
Init, 697

IsActive, 697
SelectionObj, 697
SetActive, 697
SetHighlight, 698
gazebo::rendering::UserCamera, 829
~UserCamera, 832
AnimationComplete, 832
AttachToVisualImpl, 832
EnableViewController, 832
Fini, 833
GetAvgFPS, 833
GetGUIOverlay, 833
GetImageHeight, 833
GetImageWidth, 833
GetTriangleCount, 833
GetViewControllerTypeString, 834
GetVisual, 834
HandleKeyPressEvent, 834
HandleKeyReleaseEvent, 834
HandleMouseEvent, 835
Init, 835
Load, 835
MoveToPosition, 835
MoveToVisual, 836
PostRender, 836
Resize, 836
SetFocalPoint, 836
SetRenderTarget, 836
SetViewController, 837
SetViewportDimensions, 837
SetWorldPose, 837
TrackVisualImpl, 837
Update, 838
UserCamera, 832
gazebo::rendering::VideoVisual, 880
~VideoVisual, 881
VideoVisual, 881
gazebo::rendering::ViewController, 881
~ViewController, 882
camera, 884
enabled, 884
GetTypeString, 883
HandleKeyPressEvent, 883
HandleKeyReleaseEvent, 883
HandleMouseEvent, 883
Init, 883, 884
SetEnabled, 884
typeString, 884
Update, 884
ViewController, 882
gazebo::rendering::Visual, 885
~Visual, 890
AttachAxes, 890
AttachLineVertex, 890
AttachMesh, 890
AttachObject, 891
AttachVisual, 891
ClearParent, 891
Clone, 891
CreateDynamicLine, 891
DeleteDynamicLine, 892
DetachObjects, 892
DetachVisual, 892
DisableTrackVisual, 892
EnableTrackVisual, 892
Fini, 892
GetAttachedObjectCount, 892
GetBoundingBox, 893
GetChild, 893
GetChildCount, 893
GetMaterialName, 893
GetMeshName, 893
GetName, 894
GetNormalMap, 894
GetParent, 894
GetPose, 894
GetPosition, 894
GetRootVisual, 894
GetRotation, 894
GetScale, 895
GetScene, 895
GetSceneNode, 895
GetShaderType, 895
GetSubMeshName, 895
GetTransparency, 895
GetVisibilityFlags, 896
GetVisible, 896
GetWorldPose, 896
HasAttachedObject, 896
Init, 896
InsertMesh, 897
IsPlane, 897
IsStatic, 897
Load, 897, 898
LoadFromMsg, 898
LoadPlugin, 898
MakeStatic, 898
MoveToPosition, 898
MoveToPositions, 898
parent, 904
RemovePlugin, 898
scene, 904
sceneNode, 904
SetAmbient, 899
SetCastShadows, 899
SetDiffuse, 899
SetEmissive, 899
SetHighlighted, 899

SetMaterial, 900
SetName, 900
SetNormalMap, 900
SetPose, 900
SetPosition, 900
SetRibbonTrail, 900
SetRotation, 901
SetScale, 901
SetScene, 901
SetShaderType, 901
SetSkeletonPose, 901
SetSpecular, 902
SetTransparency, 902
SetVisibilityFlags, 902
SetVisible, 902
SetWireframe, 902
SetWorldPose, 903
SetWorldPosition, 903
SetWorldRotation, 903
ShowBoundingBox, 903
ShowCOM, 903
ShowCollision, 903
ShowJoints, 903
ShowSkeleton, 904
ToggleVisible, 904
Update, 904
UpdateFromMsg, 904
Visual, 890
gazebo::rendering::WindowManager, 906
CreateWindow, 907
Fini, 907
GetAvgFPS, 907
GetTriangleCount, 908
GetWindow, 908
Moved, 908
Resize, 908
SetCamera, 908
gazebo::rendering::WireBox, 909
∼WireBox, 909
Init, 910
SetVisible, 910
WireBox, 909
gazebo::sensors, 100
CATEGORY_COUNT, 103
CameraSensor_V, 102
CameraSensorPtr, 102
ContactSensor_V, 102
ContactSensorPtr, 102
DepthCameraSensor_V, 102
DepthCameraSensorPtr, 102
GpuRaySensor_V, 102
GpuRaySensorPtr, 102
IMAGE, 103
ImuSensor_V, 102
ImuSensorPtr, 102
OTHER, 103
RAY, 103
RFIDSensor_V, 102
RFIDSensorPtr, 102
RFIDTag_V, 102
RFIDTagPtr, 102
RaySensor_V, 102
RaySensorPtr, 102
Sensor_V, 102
SensorCategory, 102
SensorFactoryFn, 102
SensorPtr, 102
gazebo::sensors::CameraSensor, 188
∼CameraSensor, 190
CameraSensor, 190
Fini, 190
GetCamera, 190
GetImageData, 190
GetImageHeight, 190
GetImageWidth, 191
GetTopic, 191
Init, 191
IsActive, 191
Load, 191, 192
SaveFrame, 192
SetParent, 192
UpdateImpl, 192
gazebo::sensors::ContactSensor, 241
∼ContactSensor, 243
ContactSensor, 243
Fini, 243
GetCollisionContactCount, 243
GetCollisionCount, 243
GetCollisionName, 244
GetContacts, 244
Init, 245
IsActive, 245
Load, 245
UpdateImpl, 245
gazebo::sensors::DepthCameraSensor, 258
∼DepthCameraSensor, 259
DepthCameraSensor, 259
Fini, 259
GetDepthCamera, 259
Init, 260
Load, 260
SaveFrame, 260
SetActive, 260
SetParent, 261
UpdateImpl, 261
gazebo::sensors::GpuRaySensor, 328
∼GpuRaySensor, 331
cameraCount, 338

cameraElem, 338
chfov, 338
ConnectNewLaserFrame, 332
cvfov, 338
DisconnectNewLaserFrame, 332
far, 339
Fini, 332
GetAngleMax, 332
GetAngleMin, 332
GetAngleResolution, 332
GetCameraCount, 332
GetCosHorzFOV, 333
GetCosVertFOV, 333
GetFiducial, 333
GetHorzFOV, 333
GetHorzHalfAngle, 333
GetLaserCamera, 334
GetRange, 334
GetRangeCount, 334
GetRangeCountRatio, 334
GetRangeMax, 334
GetRangeMin, 335
GetRangeResolution, 335
GetRanges, 335
GetRayCount, 335
GetRayCountRatio, 335
GetRetro, 335
GetVertFOV, 336
GetVertHalfAngle, 336
GetVerticalAngleMax, 336
GetVerticalAngleMin, 336
GetVerticalRangeCount, 336
GetVerticalRayCount, 336
GpuRaySensor, 331
hfov, 339
horzElem, 339
horzHalfAngle, 339
horzRangeCount, 339
horzRayCount, 339
Init, 337
IsHorizontal, 337
isHorizontal, 339
Load, 337
near, 339
rangeCountRatio, 339
rangeElem, 339
rayCountRatio, 339
scanElem, 340
SetAngleMax, 337
SetAngleMin, 337
SetVerticalAngleMax, 338
SetVerticalAngleMin, 338
UpdateImpl, 338
vertElem, 340
vertHalfAngle, 340
vertRangeCount, 340
vertRayCount, 340
vfov, 340
gazebo::sensors::ImuSensor, 365
~ImuSensor, 367
Fini, 367
GetAngularVelocity, 367
GetImuMessage, 367
GetLinearAcceleration, 367
GetOrientation, 368
ImuSensor, 367
Init, 368
Load, 368
SetReferencePose, 368
UpdateImpl, 368
gazebo::sensors::MultiCameraSensor, 522
~MultiCameraSensor, 524
Fini, 524
GetCamera, 524
GetCameraCount, 524
GetImageData, 525
GetImageHeight, 525
GetImageWidth, 525
GetTopic, 526
Init, 526
IsActive, 526
Load, 526
MultiCameraSensor, 524
SaveFrame, 526
UpdateImpl, 527
gazebo::sensors::RFIDSensor, 658
~RFIDSensor, 659
AddTag, 659
Fini, 659
Init, 659
Load, 659
RFIDSensor, 659
UpdateImpl, 660
gazebo::sensors::RFIDTag, 660
~RFIDTag, 661
Fini, 661
GetTagPose, 661
Init, 662
Load, 662
RFIDTag, 661
UpdateImpl, 662
gazebo::sensors::RaySensor, 641
~RaySensor, 643
Fini, 643
GetAngleMax, 643
GetAngleMin, 643
GetAngleResolution, 643
GetFiducial, 643

GetLaserShape, 644
GetRange, 644
GetRangeCount, 644
GetRangeMax, 644
GetRangeMin, 644
GetRangeResolution, 645
GetRanges, 645
GetRayCount, 645
GetRetro, 645
GetTopic, 646
GetVerticalAngleMax, 646
GetVerticalAngleMin, 646
GetVerticalRangeCount, 646
GetVerticalRayCount, 646
Init, 646
IsActive, 646
Load, 647
RaySensor, 643
UpdateImpl, 647
gazebo::sensors::Sensor, 698
~Sensor, 701
active, 706
ConnectUpdated, 701
connections, 706
DisconnectUpdated, 701
FillMsg, 701
Fini, 702
GetCategory, 702
GetLastMeasurementTime, 702
GetLastUpdateTime, 702
GetName, 702
GetParentName, 702
GetPose, 703
GetScopedName, 703
GetTopic, 703
GetType, 703
GetUpdateRate, 703
GetVisualize, 703
GetWorldName, 704
Init, 704
IsActive, 704
lastMeasurementTime, 706
lastUpdateTime, 706
Load, 704
node, 706
parentName, 706
plugins, 706
pose, 707
poseSub, 707
ResetLastUpdateTime, 705
sdf, 707
Sensor, 701
SetActive, 705
SetParent, 705
SetUpdateRate, 705
Update, 705
UpdateImpl, 706
updatePeriod, 707
world, 707
gazebo::sensors::SensorFactory, 707
GetSensorTypes, 708
NewSensor, 708
RegisterAll, 708
RegisterSensor, 708
gazebo::sensors::SensorManager, 709
CreateSensor, 710
Fini, 710
GetSensor, 710
GetSensorTypes, 711
GetSensors, 711
Init, 711
RemoveSensor, 711
RemoveSensors, 711
ResetLastUpdateTimes, 711
Run, 711
RunThreads, 712
SensorsInitialized, 712
Stop, 712
Update, 712
gazebo::sensors::SimTimeEvent, 723
condition, 723
time, 723
gazebo::sensors::SimTimeEventHandler, 723
~SimTimeEventHandler, 724
AddRelativeEvent, 724
SimTimeEventHandler, 724
gazebo::transport, 103
ConnectionPtr, 105
MessagePtr, 105
NodePtr, 105
PublicationPtr, 105
PublicationTransportPtr, 105
PublisherPtr, 105
SubscriberPtr, 105
SubscriptionTransportPtr, 105
gazebo::transport::CallbackHelper, 157
~CallbackHelper, 158
CallbackHelper, 158
GetId, 158
GetLatching, 158
GetMsgType, 159
HandleData, 159
HandleMessage, 159
IsLocal, 159
latching, 160
gazebo::transport::CallbackHelperT
CallbackHelperT, 161
GetMsgType, 161

HandleData, 161
HandleMessage, 162
IsLocal, 162
gazebo::transport::CallbackHelperT< M >, 160
gazebo::transport::Connection, 222
~Connection, 225
AcceptCallback, 224
AsyncRead, 225
Cancel, 225
Connect, 225
ConnectToShutdown, 225
Connection, 225
DisconnectShutdown, 226
EnqueueMsg, 226
GetId, 226
GetLocalAddress, 226
GetLocalHostname, 226
GetLocalPort, 226
GetLocalURI, 227
GetRemoteAddress, 227
GetRemoteHostname, 227
GetRemotePort, 227
GetRemoteURI, 227
IsOpen, 227
Listen, 228
ProcessWriteQueue, 228
Read, 228
ReadCallback, 224
Shutdown, 228
StartRead, 228
StopRead, 228
ValidateIP, 229
gazebo::transport::ConnectionManager, 229
Advertise, 230
ConnectToRemoteHost, 231
eventConnections, 233
Fini, 231
GetAllPublishers, 231
GetTopicNamespaces, 231
Init, 231
IsRunning, 231
RegisterTopicNamespace, 232
RemoveConnection, 232
Run, 232
RunUpdate, 232
Stop, 232
Subscribe, 232
Unadvertise, 232
Unsubscribe, 233
gazebo::transport::ConnectionReadTask, 233
ConnectionReadTask, 234
execute, 234
gazebo::transport::IOManager, 380
~IOManager, 380
DecCount, 381
GetCount, 381
GetIO, 381
IOManager, 380
IncCount, 381
Stop, 381
gazebo::transport::Node, 535
~Node, 537
Advertise, 537
DecodeTopicName, 537
EncodeTopicName, 537
Fini, 538
GetId, 538
GetMsgType, 538
GetTopicNamespace, 538
HandleData, 538
HandleMessage, 539
HasLatchedSubscriber, 539
Init, 539
InsertLatchedMsg, 539
Node, 537
ProcessIncoming, 540
ProcessPublishers, 540
Publish, 540
RemoveCallback, 540
Subscribe, 540, 541
gazebo::transport::Publication, 612
~Publication, 613
AddPublisher, 613
AddSubscription, 613
AddTransport, 613
GetCallbackCount, 614
GetLocallyAdvertised, 614
GetMsgType, 614
GetNodeCount, 614
GetRemoteSubscriptionCount, 614
GetTransportCount, 614
HasTransport, 615
LocalPublish, 615
Publication, 613
Publish, 615
RemoveSubscription, 615
RemoveTransport, 616
SetLocallyAdvertised, 616
gazebo::transport::PublicationTransport, 616
~PublicationTransport, 617
AddCallback, 617
Fini, 617
GetConnection, 617
GetMsgType, 618
GetTopic, 618
Init, 618
PublicationTransport, 617
gazebo::transport::PublishTask, 622

execute, 623
PublishTask, 622
gazebo::transport::Publisher, 618
~Publisher, 620
GetLatching, 620
GetMsgType, 620
GetOutgoingCount, 620
GetPrevMsg, 620
GetPrevMsgPtr, 620
GetTopic, 620
HasConnections, 620
Publish, 621
Publisher, 619
SendMessage, 621
SetPublication, 621
WaitForConnection, 621
gazebo::transport::RawCallbackHelper, 638
GetMsgType, 640
HandleData, 640
HandleMessage, 640
IsLocal, 640
RawCallbackHelper, 639
gazebo::transport::SubscribeOptions, 774
GetLatching, 775
GetMsgType, 775
GetNode, 775
GetTopic, 775
Init, 776
SubscribeOptions, 775
gazebo::transport::Subscriber, 776
~Subscriber, 777
GetCallbackId, 777
GetTopic, 777
SetCallbackId, 777
Subscriber, 777
Unsubscribe, 777
gazebo::transport::SubscriptionTransport, 778
~SubscriptionTransport, 779
GetConnection, 779
HandleData, 779
HandleMessage, 779
Init, 779
IsLocal, 780
SubscriptionTransport, 779
gazebo::transport::TopicManager, 814
AddNode, 816
Advertise, 816
ClearBuffers, 817
ConnectPubToSub, 817
ConnectSubToPub, 817
ConnectSubscribers, 817
DisconnectPubFromSub, 817
DisconnectSubFromPub, 818
FindPublication, 818
Fini, 818
GetAdvertisedTopics, 818
GetTopicNamespaces, 818
Init, 819
IsAdvertised, 819
PauseIncoming, 819
ProcessNodes, 819
Publish, 819
RegisterTopicNamespace, 820
RemoveNode, 820
SubNodeMap, 816
Subscribe, 820
Unadvertise, 820
Unsubscribe, 820
UpdatePublications, 821
gazebo::util, 105
DiagnosticTimerPtr, 105
gazebo::util::DiagnosticManager, 261
GetLabel, 262
GetLogPath, 262
GetTime, 262, 263
GetTimerCount, 263
Init, 263
Lap, 263
StartTimer, 264
StopTimer, 264
gazebo::util::DiagnosticTimer, 264
~DiagnosticTimer, 265
DiagnosticTimer, 265
GetName, 265
Lap, 265
Start, 265
Stop, 266
gazebo_core.hh, 978
Gazebo_parser, 69
ConstUrdfLinkPtr, 69
UrdfCollisionPtr, 69
UrdfLinkPtr, 69
UrdfVisualPtr, 69
GazeboGenerator
google::protobuf::compiler::cpp::GazeboGenerator, 324
GazeboGenerator.hh, 978
gazeboPathsFromEnv
gazebo::common::SystemPaths, 789
GenSphericalTexCoord
gazebo::common::Mesh, 477
gazebo::common::MeshManager, 487
gazebo::common::SubMesh, 769
Generate
google::protobuf::compiler::cpp::GazeboGenerator, 324
generate

gazebo::rendering::GzTerrainMatGen::SM2Profile, 751
generateForCompositeMap
gazebo::rendering::GzTerrainMatGen::SM2Profile, 751
generateFpDynamicShadows
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 719
generateFpDynamicShadowsHelpers
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 719
generateFpDynamicShadowsParams
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 719
generateFpFooter
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 719
generateFpHeader
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 719
generateFpLayer
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 719
generateFragmentProgram
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 717
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 719
generateFragmentProgramSource
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 719
GenerateShaders
gazebo::rendering::RTShaderSystem, 675
generateVertexProgram
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 717
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 720
generateVertexProgramSource
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 717
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 720
generateVpDynamicShadows
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 717
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 720
generateVpDynamicShadowsParams
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 717
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 720
generateVpFooter
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 717
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 720
generateVpHeader
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg, 717
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL, 720
GeneratorType
gazebo::math, 89
GeometryFromSDF
Messages, 60
Get
gazebo::common::NodeTransform, 549
sdf::Param, 561, 562
get_master_uri
Transport, 77
get_scene
Rendering, 67
get_sensor
Sensors, 72
get_topic_namespaces
Transport, 77
get_world
Classes for physics and dynamics, 44
GetAABB
gazebo::common::Mesh, 477
gazebo::rendering::MovableText, 518
GetAbs
gazebo::math::Vector3, 860
GetAcceleration
gazebo::physics::LinkState, 440
GetAdvertisedTopics
gazebo::transport::TopicManager, 818
getAdvertisedTopics
Transport, 77
GetAllPublishers
gazebo::transport::ConnectionManager, 231
GetAmbient
gazebo::common::Material, 457
GetAmbientColor
gazebo::rendering::Scene, 682
GetAnchor
gazebo::physics::Joint, 387
gazebo::physics::ScrewJoint, 693
gazebo::physics::SliderJoint, 748
GetAngle
gazebo::physics::Joint, 388
gazebo::physics::JointState, 403
GetAngleCount
gazebo::physics::BallJoint, 137
gazebo::physics::Hinge2Joint, 358
gazebo::physics::HingeJoint, 359

gazebo::physics::Joint, 388
gazebo::physics::JointState, 403
gazebo::physics::ScrewJoint, 693
gazebo::physics::SliderJoint, 749
gazebo::physics::UniversalJoint, 827
GetAngleImpl
gazebo::physics::Joint, 388
GetAngleMax
gazebo::sensors::GpuRaySensor, 332
gazebo::sensors::RaySensor, 643
GetAngleMin
gazebo::sensors::GpuRaySensor, 332
gazebo::sensors::RaySensor, 643
GetAngleResolution
gazebo::sensors::GpuRaySensor, 332
gazebo::sensors::RaySensor, 643
GetAngles
gazebo::physics::JointState, 403
GetAngularDamping
gazebo::physics::Link, 426
GetAngularVelocity
gazebo::sensors::ImuSensor, 367
GetAnimation
gazebo::common::Skeleton, 730
GetAsABGR
gazebo::common::Color, 212
GetAsARGB
gazebo::common::Color, 212
GetAsAxis
gazebo::math::Quaternion, 628
GetAsBGRA
gazebo::common::Color, 212
GetAsEuler
gazebo::math::Quaternion, 628
GetAsHSV
gazebo::common::Color, 212
GetAsMatrix3
gazebo::math::Quaternion, 628
GetAsMatrix4
gazebo::math::Quaternion, 628
GetAsPose
gazebo::math::Matrix4, 470
GetAsRGBA
gazebo::common::Color, 212
GetAsString
sdf::Param, 562
sdf::ParamT, 567
GetAsYUV
gazebo::common::Color, 212
GetAspectRatio
gazebo::rendering::Camera, 171
GetAttachedObjectCount
gazebo::rendering::Visual, 892
GetAttribute
gazebo::physics::Joint, 388
sdf::Element, 277
GetAttributeCount
sdf::Element, 277
GetAttributeSet
sdf::Element, 277
GetAutoDisable
gazebo::physics::Model, 493
GetAutoDisableFlag
gazebo::physics::PhysicsEngine, 573
GetAvgColor
gazebo::common::Image, 362
GetAvgFPS
gazebo::rendering::Camera, 171
gazebo::rendering::UserCamera, 833
gazebo::rendering::WindowManager, 907
GetBPP
gazebo::common::Image, 363
GetBackgroundColor
gazebo::rendering::Scene, 682
GetBasePath
gazebo::common::LogRecord, 449
GetBaseline
gazebo::rendering::MovableText, 518
GetBindShapeTransform
gazebo::common::Skeleton, 730
GetBlendFactors
gazebo::common::Material, 457
GetBlendMode
gazebo::common::Material, 457
GetBoundingBox
gazebo::physics::Collision, 199
gazebo::physics::Entity, 284
gazebo::physics::Link, 426
gazebo::physics::Model, 493
gazebo::rendering::Visual, 893
getBoundingRadius
gazebo::rendering::DynamicRenderable, 271
gazebo::rendering::MovableText, 518
GetById
gazebo::physics::Base, 142
GetByName
gazebo::physics::Base, 142
gazebo::physics::World, 914
GetCallbackCount
gazebo::transport::Publication, 614
GetCallbackId
gazebo::transport::Subscriber, 777
GetCamera
gazebo::rendering::Scene, 682
gazebo::sensors::CameraSensor, 190
gazebo::sensors::MultiCameraSensor, 524
GetCameraCount
gazebo::rendering::Scene, 683

gazebo::sensors::GpuRaySensor, 332
gazebo::sensors::MultiCameraSensor, 524
GetCameraToViewportRay
gazebo::rendering::Camera, 171
GetCategory
gazebo::sensors::Sensor, 702
GetCellCount
gazebo::rendering::Grid, 342
GetCellLength
gazebo::rendering::Grid, 342
GetCenter
gazebo::math::Box, 150
GetCharHeight
gazebo::rendering::MovableText, 518
GetChild
gazebo::common::SkeletonNode, 741
gazebo::physics::Base, 142, 143
gazebo::physics::Joint, 388
gazebo::rendering::Visual, 893
GetChildById
gazebo::common::SkeletonNode, 741
GetChildByName
gazebo::common::SkeletonNode, 742
GetChildCollision
gazebo::physics::Entity, 284
GetChildCount
gazebo::common::SkeletonNode, 742
gazebo::physics::Base, 143
gazebo::rendering::Visual, 893
GetChildJointsLinks
gazebo::physics::Link, 426
GetChildLink
gazebo::physics::Entity, 285
GetChunk
gazebo::common::LogPlay, 445
GetChunkCount
gazebo::common::LogPlay, 445
GetCmd
gazebo::common::PID, 584
GetCoG
gazebo::physics::Inertial, 371
GetCollision
gazebo::physics::Link, 426, 427
GetCollisionBoundingBox
gazebo::physics::Entity, 285
GetCollisionById
gazebo::physics::Link, 427
GetCollisionContactCount
gazebo::sensors::ContactSensor, 243
GetCollisionCount
gazebo::sensors::ContactSensor, 243
GetCollisionName
gazebo::sensors::ContactSensor, 244
GetCollisionState
gazebo::physics::LinkState, 440, 441
GetCollisionStateCount
gazebo::physics::LinkState, 441
GetCollisionStates
gazebo::physics::LinkState, 441
GetCollisions
gazebo::physics::Link, 427
GetColor
gazebo::rendering::Grid, 342
gazebo::rendering::MovableText, 518
GetConnection
gazebo::transport::PublicationTransport, 617
gazebo::transport::SubscriptionTransport, 779
GetContact
gazebo::physics::ContactManager, 240
GetContactCount
gazebo::physics::ContactManager, 240
GetContactManager
gazebo::physics::PhysicsEngine, 573
GetContactMaxCorrectingVel
gazebo::physics::PhysicsEngine, 573
GetContactSurfaceLayer
gazebo::physics::PhysicsEngine, 573
GetContacts
gazebo::physics::ContactManager, 240
gazebo::sensors::ContactSensor, 244
GetContactsEnabled
gazebo::physics::Collision, 199
GetCopyChildren
sdf::Element, 277
GetCosHorzFOV
gazebo::sensors::GpuRaySensor, 333
GetCosVertFOV
gazebo::sensors::GpuRaySensor, 333
GetCount
gazebo::transport::IOManager, 381
GetCurrentDir
SystemPaths.hh, 1105
GetDBConfig
Common, 33
GetData
gazebo::common::Image, 363
GetDblNormal
gazebo::math::Rand, 637
GetDblUniform
gazebo::math::Rand, 637
GetDefaultAsString
sdf::Param, 562
sdf::ParamT, 567
GetDefaultValue
sdf::ParamT, 567
GetDepthCamera
gazebo::sensors::DepthCameraSensor, 259
GetDepthData

gazebo::rendering::DepthCamera, 257
GetDepthWrite
gazebo::common::Material, 457
GetDescription
sdf::Element, 277
sdf::Param, 562
GetDiffuse
gazebo::common::Material, 457
GetDiffuseColor
gazebo::rendering::Light, 414
GetDirection
gazebo::rendering::Camera, 172
gazebo::rendering::Light, 414
GetDirtyPose
gazebo::physics::Entity, 285
GetDistToLine
gazebo::math::Vector3, 860
GetEffortLimit
gazebo::physics::Joint, 389
GetElapsed
gazebo::common::Timer, 814
GetElement
sdf::Element, 277
GetElementDescription
sdf::Element, 277
GetElementDescriptionCount
sdf::Element, 277
GetElementImpl
sdf::Element, 278
GetEmissive
gazebo::common::Material, 458
GetEnablePhysicsEngine
gazebo::physics::World, 914
GetEnabled
gazebo::physics::Link, 427
GetEncoding
gazebo::common::LogPlay, 445
gazebo::common::LogRecord, 449
GetEntity
gazebo::physics::World, 914
GetEntityBelowPoint
gazebo::physics::World, 914
GetErrorFile
gazebo::common::Exception, 319
GetErrorStr
gazebo::common::Exception, 320
GetErrors
gazebo::common::PID, 585
GetEulerRotation
gazebo::math::Matrix4, 470
GetExp
gazebo::math::Quaternion, 628
GetFarClip
gazebo::rendering::Camera, 172
GetFiducial
gazebo::physics::MultiRayShape, 531
gazebo::physics::RayShape, 650
gazebo::sensors::GpuRaySensor, 333
gazebo::sensors::RaySensor, 643
GetFileSize
gazebo::common::LogRecord, 449
GetFilename
gazebo::common::Image, 363
gazebo::common::LogRecord, 449
gazebo::physics::TrimeshShape, 824
gazebo::PluginT, 596
GetFirstContact
gazebo::rendering::Scene, 683
GetFirstElement
sdf::Element, 278
GetFirstUpdate
gazebo::common::LogRecord, 450
GetFocalPoint
gazebo::rendering::OrbitViewController, 557
GetFont
gazebo::rendering::MovableText, 518
GetForce
gazebo::physics::Joint, 389
GetForceTorque
gazebo::physics::Joint, 389, 390
GetFrameAt
gazebo::common::NodeAnimation, 543
GetFrameCount
gazebo::common::NodeAnimation, 544
GetFrameFilename
gazebo::rendering::Camera, 172
GetGUIOverlay
gazebo::rendering::UserCamera, 833
GetGazeboPaths
gazebo::common::SystemPaths, 787
GetGazeboVersion
gazebo::common::LogPlay, 445
GetGlobalAxis
gazebo::physics::Joint, 390
GetGlobalPoints
gazebo::physics::RayShape, 650
GetGravity
gazebo::physics::PhysicsEngine, 573
GetGravityMode
gazebo::physics::Link, 428
GetGrid
gazebo::rendering::Scene, 683
GetGridCount
gazebo::rendering::Scene, 684
GetHFOV
gazebo::rendering::Camera, 172
GetHandle
gazebo::common::SkeletonNode, 742

gazebo::PluginT, 596
GetHeader
Messages, 60
GetHeight
gazebo::common::Image, 363
gazebo::common::Video, 879
gazebo::physics::HeightmapShape, 354
gazebo::rendering::Grid, 342
gazebo::rendering::Heightmap, 351
GetHeightBelowPoint
gazebo::rendering::Scene, 684
GetHeightmap
gazebo::rendering::Scene, 684
GetHighStop
gazebo::physics::BallJoint, 137
gazebo::physics::Joint, 390
GetHorzFOV
gazebo::sensors::GpuRaySensor, 333
GetHorzHalfAngle
gazebo::sensors::GpuRaySensor, 333
GetIO
gazebo::transport::IOManager, 381
GetIXX
gazebo::physics::Inertial, 372
GetIXY
gazebo::physics::Inertial, 372
GetIXZ
gazebo::physics::Inertial, 372
GetIYY
gazebo::physics::Inertial, 372
GetIYZ
gazebo::physics::Inertial, 373
GetIZZ
gazebo::physics::Inertial, 373
GetId
gazebo::common::SkeletonNode, 742
gazebo::event::Connection, 222
gazebo::physics::Base, 143
gazebo::rendering::Scene, 684
gazebo::transport::CallbackHelper, 158
gazebo::transport::Connection, 226
gazebo::transport::Node, 538
GetIdString
gazebo::rendering::Scene, 684
GetImageByteSize
gazebo::rendering::Camera, 172
GetImageData
gazebo::rendering::Camera, 173
gazebo::sensors::CameraSensor, 190
gazebo::sensors::MultiCameraSensor, 525
GetImageDepth
gazebo::rendering::Camera, 173
GetImageFormat
gazebo::rendering::Camera, 173
GetImageHeight
gazebo::rendering::Camera, 173
gazebo::rendering::UserCamera, 833
gazebo::sensors::CameraSensor, 190
gazebo::sensors::MultiCameraSensor, 525
GetImageWidth
gazebo::rendering::Camera, 174
gazebo::rendering::UserCamera, 833
gazebo::sensors::CameraSensor, 191
gazebo::sensors::MultiCameraSensor, 525
GetImuMessage
gazebo::sensors::ImuSensor, 367
GetInclude
sdf::Element, 278
GetIndex
gazebo::common::SubMesh, 769
GetIndexCount
gazebo::common::Mesh, 478
gazebo::common::SubMesh, 769
GetInertiaRatio
gazebo::physics::Joint, 391
GetInertial
gazebo::physics::Inertial, 372
gazebo::physics::Link, 428
GetInitialRelativePose
gazebo::physics::Entity, 285
GetInitialized
gazebo::rendering::Camera, 174
gazebo::rendering::Scene, 684
gazebo::Server, 715
GetIntNormal
gazebo::math::Rand, 637
GetIntUniform
gazebo::math::Rand, 638
GetInterpolatedKeyFrame
gazebo::common::NumericAnimation, 553
gazebo::common::PoseAnimation, 606, 607
GetIntersection
gazebo::physics::RayShape, 650
GetInverse
gazebo::math::Pose, 601
gazebo::math::Quaternion, 628
GetInverseBindTransform
gazebo::common::SkeletonNode, 742
GetJoint
gazebo::physics::Model, 494
GetJointController
gazebo::physics::Model, 494
GetJointCount
gazebo::physics::Model, 494
GetJointLink
gazebo::physics::Joint, 391
GetJointState
gazebo::physics::ModelState, 508

GetJointStateCount
gazebo::physics::ModelState, 509
GetJointStates
gazebo::physics::ModelState, 509
GetJoints
gazebo::physics::Model, 494
GetKey
sdf::Param, 562
GetKeyFrame
gazebo::common::Animation, 127
gazebo::common::NodeAnimation, 544
GetKeyFrameCount
gazebo::common::Animation, 127
GetKeyFramesAtTime
gazebo::common::Animation, 127
GetKinematic
gazebo::physics::Link, 428
GetLabel
gazebo::util::DiagnosticManager, 262
GetLaserCamera
gazebo::sensors::GpuRaySensor, 334
GetLaserData
gazebo::rendering::GpuLaser, 327
GetLaserRetro
gazebo::physics::Collision, 199
GetLaserShape
gazebo::sensors::RaySensor, 644
GetLastMeasurementTime
gazebo::sensors::Sensor, 702
GetLastRenderWallTime
gazebo::rendering::Camera, 174
GetLastUpdateTime
gazebo::sensors::Sensor, 702
GetLatching
gazebo::transport::CallbackHelper, 158
gazebo::transport::Publisher, 620
gazebo::transport::SubscribeOptions, 775
GetLength
gazebo::common::Animation, 128
gazebo::common::NodeAnimation, 544
gazebo::common::SkeletonAnimation, 735
gazebo::math::Vector3, 860
gazebo::math::Vector4, 872
gazebo::physics::CylinderShape, 252
gazebo::physics::RayShape, 651
GetLight
gazebo::rendering::Scene, 684, 685
GetLightCount
gazebo::rendering::Scene, 685
GetLighting
gazebo::common::Material, 458
getLights
gazebo::rendering::MovableText, 518
GetLineWidth
gazebo::rendering::Grid, 342
GetLinearAcceleration
gazebo::sensors::ImuSensor, 367
GetLinearDamping
gazebo::physics::Link, 428
GetLink
gazebo::physics::Collision, 199
gazebo::physics::Model, 494
GetLinkById
gazebo::physics::Model, 495
GetLinkForce
gazebo::physics::Joint, 391
GetLinkState
gazebo::physics::ModelState, 509
GetLinkStateCount
gazebo::physics::ModelState, 510
GetLinkStates
gazebo::physics::ModelState, 510
GetLinkTorque
gazebo::physics::Joint, 391
GetLinks
gazebo::physics::Model, 495
GetLocalAddress
gazebo::transport::Connection, 226
GetLocalAxis
gazebo::physics::Joint, 392
GetLocalHostname
gazebo::transport::Connection, 226
GetLocalPort
gazebo::transport::Connection, 226
GetLocalURI
gazebo::transport::Connection, 227
GetLocallyAdvertised
gazebo::transport::Publication, 614
GetLog
gazebo::math::Quaternion, 629
GetLogPath
gazebo::common::SystemPaths, 788
gazebo::util::DiagnosticManager, 262
GetLogVersion
gazebo::common::LogPlay, 445
GetLowStop
gazebo::physics::BallJoint, 137
gazebo::physics::Joint, 392
GetLowerLimit
gazebo::physics::Joint, 392
GetMOI
gazebo::physics::Inertial, 373
GetManager
gazebo::rendering::Scene, 685
GetManifest
Common, 34
GetMass
gazebo::physics::Inertial, 373

GetMaterial
gazebo::common::Mesh, 478
getMaterial
gazebo::rendering::MovableText, 518
GetMaterialCount
gazebo::common::Mesh, 478
GetMaterialIndex
gazebo::common::SubMesh, 769
GetMaterialName
gazebo::rendering::Visual, 893
GetMax
gazebo::common::Mesh, 478
gazebo::common::SubMesh, 769
gazebo::math::Vector3, 861
GetMaxAngle
gazebo::physics::MultiRayShape, 531
GetMaxColor
gazebo::common::Image, 363
GetMaxContacts
gazebo::physics::PhysicsEngine, 574
GetMaxForce
gazebo::physics::Joint, 392
GetMaxHeight
gazebo::physics::HeightmapShape, 354
GetMaxIndex
gazebo::common::SubMesh, 769
GetMaxRange
gazebo::physics::MultiRayShape, 531
GetMaxStepSize
gazebo::physics::PhysicsEngine, 574
GetMesh
gazebo::common::MeshManager, 488
GetMeshAABB
gazebo::common::MeshManager, 488
GetMeshName
gazebo::rendering::Visual, 893
GetMeshURI
gazebo::physics::TrimeshShape, 824
GetMin
gazebo::common::Mesh, 478
gazebo::common::SubMesh, 769
gazebo::math::Vector3, 861
GetMinAngle
gazebo::physics::MultiRayShape, 532
GetMinHeight
gazebo::physics::HeightmapShape, 354
GetMinRange
gazebo::physics::MultiRayShape, 532
GetModel
gazebo::physics::Collision, 200
gazebo::physics::Link, 428
gazebo::physics::World, 915
GetModelBelowPoint
gazebo::physics::World, 915
GetModelConfig
Common, 34
GetModelCount
gazebo::physics::World, 916
GetModelFile
Common, 34
GetModelName
Common, 34
GetModelPath
Common, 35
GetModelPaths
gazebo::common::SystemPaths, 788
GetModelState
gazebo::physics::WorldState, 925
GetModelStateCount
gazebo::physics::WorldState, 926
GetModelStates
gazebo::physics::WorldState, 926
GetModelTransform
gazebo::common::SkeletonNode, 742
GetModelVisualAt
gazebo::rendering::Scene, 685
GetModels
Common, 35
gazebo::physics::World, 916
GetMovableType
gazebo::rendering::DynamicLines, 268
getMovableType
gazebo::rendering::DynamicLines, 268
GetMsgType
gazebo::transport::CallbackHelper, 159
gazebo::transport::CallbackHelperT, 161
gazebo::transport::Node, 538
gazebo::transport::Publication, 614
gazebo::transport::PublicationTransport, 618
gazebo::transport::Publisher, 620
gazebo::transport::RawCallbackHelper, 640
gazebo::transport::SubscribeOptions, 775
GetMsgTypes
gazebo::msgs::MsgFactory, 521
GetName
gazebo::common::Material, 458
gazebo::common::Mesh, 478
gazebo::common::NodeAnimation, 545
gazebo::common::SkeletonAnimation, 736
gazebo::common::SkeletonNode, 743
gazebo::common::SubMesh, 770
gazebo::physics::Base, 143
gazebo::physics::State, 760
gazebo::physics::World, 916
gazebo::rendering::Camera, 174
gazebo::rendering::Light, 414
gazebo::rendering::Scene, 686
gazebo::rendering::Visual, 894

gazebo::sensors::Sensor, 702
gazebo::util::DiagnosticTimer, 265
sdf::Element, 278
GetNearClip
gazebo::rendering::Camera, 174
GetNearestEntityBelow
gazebo::physics::Entity, 285
GetNextElement
sdf::Element, 278
GetNextFrame
gazebo::common::Video, 879
GetNode
gazebo::transport::SubscribeOptions, 775
GetNodeAssignment
gazebo::common::SubMesh, 770
GetNodeAssignmentsCount
gazebo::common::SubMesh, 770
GetNodeByHandle
gazebo::common::Skeleton, 730
GetNodeById
gazebo::common::Skeleton, 730
GetNodeByName
gazebo::common::Skeleton, 731
GetNodeCount
gazebo::common::SkeletonAnimation, 736
gazebo::transport::Publication, 614
GetNodePoseAt
gazebo::common::SkeletonAnimation, 736
GetNodes
gazebo::common::Skeleton, 731
GetNormal
gazebo::common::SubMesh, 770
gazebo::math::Vector3, 861
gazebo::physics::PlaneShape, 592
GetNormalCount
gazebo::common::Mesh, 478
gazebo::common::SubMesh, 770
GetNormalMap
gazebo::rendering::Visual, 894
GetNumAnimations
gazebo::common::Skeleton, 731
GetNumJoints
gazebo::common::Skeleton, 731
GetNumNodes
gazebo::common::Skeleton, 731
GetNumPoints
gazebo::math::RotationSpline, 670
GetNumRawTrans
gazebo::common::SkeletonNode, 743
GetNumVertNodeWeights
gazebo::common::Skeleton, 731
GetOgreCamera
gazebo::rendering::Camera, 174
GetOgrePaths
gazebo::common::SystemPaths, 788
GetOgreTerrain
gazebo::rendering::Heightmap, 351
GetOperationType
gazebo::rendering::DynamicRenderable, 272
GetOrientation
gazebo::sensors::ImuSensor, 368
GetOutgoingCount
gazebo::transport::Publisher, 620
GetPSSMShadowCameraSetup
gazebo::rendering::RTShaderSystem, 675
GetParam
gazebo::physics::PhysicsEngine, 574
GetParent
gazebo::common::SkeletonNode, 743
gazebo::physics::Base, 143
gazebo::physics::Joint, 393
gazebo::rendering::Projector, 610
gazebo::rendering::Visual, 894
sdf::Element, 278
GetParentId
gazebo::physics::Base, 143
GetParentJointsLinks
gazebo::physics::Link, 428
GetParentModel
gazebo::physics::Entity, 286
GetParentName
gazebo::sensors::Sensor, 702
GetPath
gazebo::common::Mesh, 479
GetPauseTime
gazebo::physics::World, 916
GetPaused
gazebo::common::LogRecord, 450
GetPerpendicular
gazebo::math::Vector3, 861
GetPhysicsEngine
gazebo::physics::World, 916
GetPhysicsUpdateMutex
gazebo::physics::PhysicsEngine, 574
GetPitch
gazebo::common::Image, 363
gazebo::math::Quaternion, 629
GetPitchNode
gazebo::rendering::Camera, 175
GetPixel
gazebo::common::Image, 363
GetPixelFormat
gazebo::common::Image, 364
GetPluginCount
gazebo::physics::Model, 495
GetPluginPaths
gazebo::common::SystemPaths, 788
GetPoint

gazebo::math::RotationSpline, 670
gazebo::math::Spline, 755
gazebo::rendering::DynamicLines, 269
GetPointCount
gazebo::math::Spline, 756
gazebo::rendering::DynamicLines, 269
GetPointSize
gazebo::common::Material, 458
GetPos
gazebo::physics::HeightmapShape, 354
GetPose
gazebo::physics::CollisionState, 207
gazebo::physics::Inertial, 374
gazebo::physics::LinkState, 442
gazebo::physics::ModelState, 510
gazebo::rendering::Visual, 894
gazebo::sensors::Sensor, 703
GetPoseAt
gazebo::common::SkeletonAnimation, 736
GetPoseAtX
gazebo::common::SkeletonAnimation, 737
GetPosition
gazebo::rendering::Light, 414
gazebo::rendering::Visual, 894
GetPrevMsg
gazebo::transport::Publisher, 620
GetPrevMsgPtr
gazebo::transport::Publisher, 620
GetPrimitiveType
gazebo::common::SubMesh, 770
GetPrincipalMoments
gazebo::physics::Inertial, 374
GetProductsofInertia
gazebo::physics::Inertial, 374
GetRGBData
gazebo::common::Image, 364
GetRadius
gazebo::physics::CylinderShape, 252
gazebo::physics::SphereShape, 753
GetRandSeed
gazebo::common::LogPlay, 445
GetRange
gazebo::physics::MultiRayShape, 532
gazebo::sensors::GpuRaySensor, 334
gazebo::sensors::RaySensor, 644
GetRangeCount
gazebo::sensors::GpuRaySensor, 334
gazebo::sensors::RaySensor, 644
GetRangeCountRatio
gazebo::sensors::GpuRaySensor, 334
GetRangeMax
gazebo::sensors::GpuRaySensor, 334
gazebo::sensors::RaySensor, 644
GetRangeMin
gazebo::sensors::GpuRaySensor, 335
gazebo::sensors::RaySensor, 644
GetRangeResolution
gazebo::sensors::GpuRaySensor, 335
gazebo::sensors::RaySensor, 645
GetRanges
gazebo::sensors::GpuRaySensor, 335
gazebo::sensors::RaySensor, 645
GetRawTransform
gazebo::common::SkeletonNode, 743
GetRawTransforms
gazebo::common::SkeletonNode, 743
GetRayCount
gazebo::sensors::GpuRaySensor, 335
gazebo::sensors::RaySensor, 645
GetRayCountRatio
gazebo::sensors::GpuRaySensor, 335
GetRealTime
gazebo::physics::State, 760
gazebo::physics::World, 916
GetRealTimeUpdateRate
gazebo::physics::PhysicsEngine, 574
GetRelativeAngularAccel
gazebo::physics::Collision, 200
gazebo::physics::Entity, 286
gazebo::physics::Link, 429
gazebo::physics::Model, 495
GetRelativeAngularVel
gazebo::physics::Collision, 200
gazebo::physics::Entity, 286
gazebo::physics::Link, 429
gazebo::physics::Model, 495
GetRelativeForce
gazebo::physics::Link, 429
GetRelativeLinearAccel
gazebo::physics::Collision, 200
gazebo::physics::Entity, 286
gazebo::physics::Link, 429
gazebo::physics::Model, 496
GetRelativeLinearVel
gazebo::physics::Collision, 200
gazebo::physics::Entity, 286
gazebo::physics::Link, 429
gazebo::physics::Model, 496
GetRelativePoints
gazebo::physics::RayShape, 651
GetRelativePose
gazebo::physics::Entity, 287
GetRelativeTorque
gazebo::physics::Link, 429
GetRemoteAddress
gazebo::transport::Connection, 227
GetRemoteHostname
gazebo::transport::Connection, 227

GetRemotePort
gazebo::transport::Connection, 227
GetRemoteSubscriptionCount
gazebo::transport::Publication, 614
GetRemoteURI
gazebo::transport::Connection, 227
getRenderOperation
gazebo::rendering::MovableText, 518
GetRenderPathType
gazebo::rendering::RenderEngine, 656
GetRenderRate
gazebo::rendering::Camera, 175
GetRenderTexture
gazebo::rendering::Camera, 175
GetRequired
sdf::Element, 278
sdf::Param, 562
GetResRange
gazebo::physics::MultiRayShape, 532
GetRetro
gazebo::physics::MultiRayShape, 532
gazebo::physics::RayShape, 651
gazebo::sensors::GpuRaySensor, 335
gazebo::sensors::RaySensor, 645
GetRight
gazebo::rendering::Camera, 175
GetRoll
gazebo::math::Quaternion, 629
GetRootNode
gazebo::common::Skeleton, 732
GetRootVisual
gazebo::rendering::Visual, 894
GetRotation
gazebo::common::PoseKeyFrame, 608
gazebo::math::Matrix4, 470
gazebo::rendering::Visual, 894
GetRounded
gazebo::math::Vector3, 861
GetRunTime
gazebo::common::LogRecord, 450
GetRunning
gazebo::common::LogRecord, 450
gazebo::common::Timer, 814
GetSDF
gazebo::physics::Actor, 114
gazebo::physics::Base, 144
gazebo::physics::Model, 496
GetSID
gazebo::common::NodeTransform, 549
GetSORPGSIters
gazebo::physics::PhysicsEngine, 575
GetSORPGSPreconIters
gazebo::physics::PhysicsEngine, 575
GetSORPGSW
gazebo::physics::PhysicsEngine, 575
GetSampleCount
gazebo::physics::MultiRayShape, 533
GetSaveable
gazebo::physics::Base, 144
GetScale
gazebo::rendering::Visual, 895
GetScanResolution
gazebo::physics::MultiRayShape, 533
GetScene
gazebo::rendering::Camera, 175
gazebo::rendering::RenderEngine, 656
gazebo::rendering::Visual, 895
GetSceneCount
gazebo::rendering::RenderEngine, 656
GetSceneNode
gazebo::rendering::Camera, 175
gazebo::rendering::Grid, 342
gazebo::rendering::Visual, 895
GetScopedName
gazebo::physics::Base, 144
gazebo::sensors::Sensor, 703
GetScreenshotPath
gazebo::rendering::Camera, 176
GetSeed
gazebo::math::Rand, 638
GetSelectedEntity
gazebo::physics::World, 917
GetSelectedVisual
gazebo::rendering::Scene, 686
GetSelfCollide
gazebo::physics::Link, 430
GetSensor
gazebo::sensors::SensorManager, 710
GetSensorCount
gazebo::physics::Link, 430
gazebo::physics::Model, 496
GetSensorName
gazebo::physics::Link, 430
GetSensorTypes
gazebo::sensors::SensorFactory, 708
gazebo::sensors::SensorManager, 711
GetSensors
gazebo::sensors::SensorManager, 711
GetSet
sdf::Param, 562
GetSetWorldPoseMutex
gazebo::physics::World, 917
GetShadeMode
gazebo::common::Material, 458
GetShaderType
gazebo::rendering::Visual, 895
GetShadowsEnabled
gazebo::rendering::Scene, 686

GetShape
gazebo::physics::Collision, 200
GetShapeType
gazebo::physics::Collision, 201
GetShininess
gazebo::common::Material, 458
GetShowOnTop
gazebo::rendering::MovableText, 519
GetSimTime
gazebo::physics::State, 761
gazebo::physics::World, 917
GetSize
gazebo::math::Box, 150
gazebo::physics::BoxShape, 155
gazebo::physics::HeightmapShape, 355
gazebo::physics::PlaneShape, 592
gazebo::physics::TrimeshShape, 824
GetSkeleton
gazebo::common::Mesh, 479
GetSpaceWidth
gazebo::rendering::MovableText, 519
GetSpecular
gazebo::common::Material, 458
GetSpecularColor
gazebo::rendering::Light, 415
GetSquaredLength
gazebo::math::Vector3, 861
gazebo::math::Vector4, 872
getSquaredViewDepth
gazebo::rendering::DynamicRenderable, 272
gazebo::rendering::MovableText, 519
GetStartTime
gazebo::physics::World, 917
GetState
gazebo::physics::Collision, 201
GetStepTime
gazebo::physics::PhysicsEngine, 575
GetSubMesh
gazebo::common::Mesh, 479
GetSubMeshCount
gazebo::common::Mesh, 479
GetSubMeshName
gazebo::rendering::Visual, 895
GetSubSampling
gazebo::physics::HeightmapShape, 355
GetSum
gazebo::math::Vector3, 862
GetSurface
gazebo::physics::Collision, 201
GetTagPose
gazebo::sensors::RFIDTag, 661
GetTangent
gazebo::math::Spline, 756
GetTargetRealTimeFactor
gazebo::physics::PhysicsEngine, 575
GetTension
gazebo::math::Spline, 756
GetTexCoord
gazebo::common::SubMesh, 771
GetTexCoordCount
gazebo::common::Mesh, 480
gazebo::common::SubMesh, 771
GetText
gazebo::rendering::MovableText, 519
GetTextureHeight
gazebo::rendering::Camera, 176
GetTextureImage
gazebo::common::Material, 459
GetTextureWidth
gazebo::rendering::Camera, 176
GetThreadPitch
gazebo::physics::ScrewJoint, 693
GetTime
gazebo::common::Animation, 128
gazebo::common::KeyFrame, 410
gazebo::util::DiagnosticManager, 262, 263
GetTimeAtX
gazebo::common::NodeAnimation, 545
GetTimerCount
gazebo::util::DiagnosticManager, 263
GetTopic
gazebo::sensors::CameraSensor, 191
gazebo::sensors::MultiCameraSensor, 526
gazebo::sensors::RaySensor, 646
gazebo::sensors::Sensor, 703
gazebo::transport::PublicationTransport, 618
gazebo::transport::Publisher, 620
gazebo::transport::SubscribeOptions, 775
gazebo::transport::Subscriber, 777
getTopicMsgType
Transport, 78
GetTopicNamespace
gazebo::transport::Node, 538
GetTopicNamespaces
gazebo::transport::ConnectionManager, 231
gazebo::transport::TopicManager, 818
GetTransform
gazebo::common::SkeletonNode, 744
GetTransforms
gazebo::common::SkeletonNode, 744
GetTranslation
gazebo::common::PoseKeyFrame, 608
gazebo::math::Matrix4, 470
GetTransparency
gazebo::common::Material, 459
gazebo::rendering::Visual, 895
GetTransportCount
gazebo::transport::Publication, 614

GetTriangleCount
gazebo::rendering::Camera, 176
gazebo::rendering::UserCamera, 833
gazebo::rendering::WindowManager, 908
GetType
gazebo::common::NodeTransform, 549
gazebo::physics::Base, 144
gazebo::physics::PhysicsEngine, 575
gazebo::PluginT, 596
gazebo::rendering::Light, 415
gazebo::sensors::Sensor, 703
GetTypeName
sdf::Param, 562
GetTypeString
gazebo::rendering::FPSViewController, 322
gazebo::rendering::OrbitViewController, 557
gazebo::rendering::ViewController, 883
GetURI
Common, 35
gazebo::physics::HeightmapShape, 355
GetUp
gazebo::rendering::Camera, 176
GetUpdatePeriod
gazebo::physics::PhysicsEngine, 576
GetUpdateRate
gazebo::physics::PhysicsEngine, 576
gazebo::sensors::Sensor, 703
GetUpperLimit
gazebo::physics::Joint, 393
GetUserCamera
gazebo::rendering::Scene, 686
GetUserCameraCount
gazebo::rendering::Scene, 686
GetVFOV
gazebo::rendering::Camera, 176
GetValue
gazebo::common::NumericKeyFrame, 555
sdf::Element, 278
sdf::ParamT, 567
GetValueBool
sdf::Element, 278
GetValueChar
sdf::Element, 278
GetValueColor
sdf::Element, 278
GetValueDouble
sdf::Element, 278
GetValueFloat
sdf::Element, 278
GetValueInt
sdf::Element, 278
GetValuePose
sdf::Element, 278
GetValueQuaternion
sdf::Element, 278
GetValueString
sdf::Element, 278
GetValueTime
sdf::Element, 278
GetValueUInt
sdf::Element, 278
GetValueVector2d
sdf::Element, 278
GetValueVector3
sdf::Element, 278
GetVelocity
gazebo::physics::Joint, 393
gazebo::physics::LinkState, 442
GetVelocityLimit
gazebo::physics::Joint, 393
GetVertFOV
gazebo::sensors::GpuRaySensor, 336
GetVertHalfAngle
gazebo::sensors::GpuRaySensor, 336
GetVertNodeWeight
gazebo::common::Skeleton, 732
GetVertex
gazebo::common::SubMesh, 771
GetVertexCount
gazebo::common::Mesh, 480
gazebo::common::SubMesh, 771
gazebo::physics::HeightmapShape, 355
GetVertexIndex
gazebo::common::SubMesh, 771
GetVerticalAngleMax
gazebo::sensors::GpuRaySensor, 336
gazebo::sensors::RaySensor, 646
GetVerticalAngleMin
gazebo::sensors::GpuRaySensor, 336
gazebo::sensors::RaySensor, 646
GetVerticalMaxAngle
gazebo::physics::MultiRayShape, 533
GetVerticalMinAngle
gazebo::physics::MultiRayShape, 533
GetVerticalRangeCount
gazebo::sensors::GpuRaySensor, 336
gazebo::sensors::RaySensor, 646
GetVerticalRayCount
gazebo::sensors::GpuRaySensor, 336
gazebo::sensors::RaySensor, 646
GetVerticalSampleCount
gazebo::physics::MultiRayShape, 533
GetVerticalScanResolution
gazebo::physics::MultiRayShape, 533
GetViewControllerTypeString
gazebo::rendering::UserCamera, 834
GetViewport
gazebo::rendering::Camera, 177

GetViewportHeight
gazebo::rendering::Camera, 177
GetViewportWidth
gazebo::rendering::Camera, 177
GetVisibilityFlags
gazebo::rendering::Visual, 896
GetVisible
gazebo::rendering::Visual, 896
GetVisual
gazebo::rendering::Scene, 686
gazebo::rendering::UserCamera, 834
GetVisualAt
gazebo::rendering::Scene, 687
GetVisualBelow
gazebo::rendering::Scene, 687
GetVisualName
gazebo::rendering::SelectionObj, 697
GetVisualize
gazebo::sensors::Sensor, 703
GetVisualsBelowPoint
gazebo::rendering::Scene, 687
GetWallTime
gazebo::common::Time, 796
gazebo::physics::State, 761
GetWallTimeAsISOString
gazebo::common::Time, 796
GetWidth
gazebo::common::Image, 364
gazebo::common::Video, 879
GetWindow
gazebo::rendering::WindowManager, 908
GetWindowId
gazebo::rendering::Camera, 177
GetWorld
gazebo::physics::Base, 144
GetWorldAngularAccel
gazebo::physics::Collision, 201
gazebo::physics::Entity, 287
gazebo::physics::Link, 430
gazebo::physics::Model, 496
GetWorldAngularVel
gazebo::physics::Collision, 201
gazebo::physics::Entity, 287
gazebo::physics::Model, 497
GetWorldCFM
gazebo::physics::PhysicsEngine, 576
GetWorldCoGLinearVel
gazebo::physics::Link, 431
GetWorldCoGPose
gazebo::physics::Link, 431
GetWorldERP
gazebo::physics::PhysicsEngine, 576
GetWorldForce
gazebo::physics::Link, 431
GetWorldLinearAccel
gazebo::physics::Collision, 202
gazebo::physics::Entity, 287
gazebo::physics::Link, 431
gazebo::physics::Model, 497
GetWorldLinearVel
gazebo::physics::Collision, 202
gazebo::physics::Entity, 287
gazebo::physics::Link, 431, 432
gazebo::physics::Model, 497
GetWorldName
gazebo::sensors::Sensor, 704
GetWorldPathExtension
gazebo::common::SystemPaths, 788
GetWorldPointOnPlane
gazebo::rendering::Camera, 177
GetWorldPose
gazebo::physics::Entity, 288
gazebo::rendering::Camera, 178
gazebo::rendering::Visual, 896
GetWorldPosition
gazebo::rendering::Camera, 178
GetWorldRotation
gazebo::rendering::Camera, 178
GetWorldTorque
gazebo::physics::Link, 432
getWorldTransforms
gazebo::rendering::MovableText, 519
GetWorldVisual
gazebo::rendering::Scene, 688
GetWrench
gazebo::physics::LinkState, 442
GetXAxis
gazebo::math::Quaternion, 629
GetXLength
gazebo::math::Box, 150
GetYAxis
gazebo::math::Quaternion, 629
GetYLength
gazebo::math::Box, 151
GetYaw
gazebo::math::Quaternion, 629
GetZAxis
gazebo::math::Quaternion, 630
GetZLength
gazebo::math::Box, 151
GetZValue
gazebo::rendering::Camera, 178
globalEndPos
gazebo::physics::RayShape, 653
globalStartPos
gazebo::physics::RayShape, 653
google, 105
google::protobuf, 105

google::protobuf::compiler, 106
google::protobuf::compiler::cpp, 106
google::protobuf::compiler::cpp::GazeboGenerator, 323
~GazeboGenerator, 324
GazeboGenerator, 324
Generate, 324
GpuLaser
gazebo::rendering::GpuLaser, 326
GpuLaser.hh, 979
GpuLaserPtr
gazebo::rendering, 99
GpuRaySensor
gazebo::sensors::GpuRaySensor, 331
GpuRaySensor.hh, 980
GpuRaySensor_V
gazebo::sensors, 102
GpuRaySensorPtr
gazebo::sensors, 102
Green
gazebo::common::Color, 219
Grid
gazebo::rendering::Grid, 341
Grid.hh, 981
Gripper
gazebo::physics::Gripper, 344
Gripper.hh, 981
GtsSurface
MeshCSG.hh, 1016
GzTerrainMatGen
gazebo::rendering::GzTerrainMatGen, 350
gzclr_end
Common, 31
gzclr_start
Common, 31
gzdbg
Common, 31
gzerr
Common, 31
gzlog
Common, 31
gzmsg
Common, 31
gzthrow
Common, 31
gzwarn
Common, 32

H_CENTER
gazebo::rendering::MovableText, 517
H_LEFT
gazebo::rendering::MovableText, 517
HEADER_LENGTH
Connection.hh, 958
HEIGHTMAP_SHAPE
gazebo::physics::Base, 141
HI_STOP
gazebo::physics::Joint, 386
HINGE2_JOINT
gazebo::physics::Base, 141
HINGE_JOINT
gazebo::physics::Base, 141
handle
gazebo::common::SkeletonNode, 746
gazebo::PluginT, 596
HandleData
gazebo::transport::CallbackHelper, 159
gazebo::transport::CallbackHelperT, 161
gazebo::transport::Node, 538
gazebo::transport::RawCallbackHelper, 640
gazebo::transport::SubscriptionTransport, 779
HandleKeyPressEvent
gazebo::rendering::FPSViewController, 322
gazebo::rendering::GUIOverlay, 347
gazebo::rendering::OrbitViewController, 557
gazebo::rendering::UserCamera, 834
gazebo::rendering::ViewController, 883
HandleKeyReleaseEvent
gazebo::rendering::FPSViewController, 322
gazebo::rendering::GUIOverlay, 348
gazebo::rendering::OrbitViewController, 557
gazebo::rendering::UserCamera, 834
gazebo::rendering::ViewController, 883
HandleMessage
gazebo::transport::CallbackHelper, 159
gazebo::transport::CallbackHelperT, 162
gazebo::transport::Node, 539
gazebo::transport::RawCallbackHelper, 640
gazebo::transport::SubscriptionTransport, 779
HandleMouseEvent
gazebo::rendering::FPSViewController, 322
gazebo::rendering::GUIOverlay, 348
gazebo::rendering::OrbitViewController, 558
gazebo::rendering::UserCamera, 835
gazebo::rendering::ViewController, 883
HasAttachedObject
gazebo::rendering::Visual, 896
HasAttribute
sdf::Element, 278
HasConnections
gazebo::transport::Publisher, 620
HasElement
sdf::Element, 279
HasElementDescription
sdf::Element, 279
HasJointState
gazebo::physics::ModelState, 510
HasLatchedSubscriber
gazebo::transport::Node, 539

HasLinkState
gazebo::physics::ModelState, 510
HasMesh
gazebo::common::MeshManager, 488
HasModel
Common, 35
HasModelState
gazebo::physics::WorldState, 926
HasNode
gazebo::common::SkeletonAnimation, 737
HasSkeleton
gazebo::common::Mesh, 480
HasTransport
gazebo::transport::Publication, 615
HasType
gazebo::physics::Base, 144
HasVertex
gazebo::common::SubMesh, 771
Heightmap
gazebo::rendering::Heightmap, 351
Heightmap.hh, 983
HeightmapShape
gazebo::physics::HeightmapShape, 353
HeightmapShape.hh, 984
HeightmapShapePtr
gazebo::physics, 95
heights
gazebo::physics::HeightmapShape, 356
Helpers.hh, 985
GZ_DBL_MAX, 987
GZ_DBL_MIN, 987
GZ_FLT_MAX, 987
GZ_FLT_MIN, 987
hfov
gazebo::sensors::GpuRaySensor, 339
Hide
gazebo::rendering::GUIOverlay, 348
Hinge2Joint
gazebo::physics::Hinge2Joint, 357
Hinge2Joint.hh, 987
HingeJoint
gazebo::physics::HingeJoint, 359
HingeJoint.hh, 989
HorizAlign
gazebo::rendering::MovableText, 517
horzElem
gazebo::physics::MultiRayShape, 534
gazebo::sensors::GpuRaySensor, 339
horzHalfAngle
gazebo::sensors::GpuRaySensor, 339
horzRangeCount
gazebo::sensors::GpuRaySensor, 339
horzRayCount
gazebo::sensors::GpuRaySensor, 339
IDENTITY
gazebo::math::Matrix4, 474
IMAGE
gazebo::sensors, 103
INTERSECTION
gazebo::common::MeshCSG, 482
IOManager
gazebo::transport::IOManager, 380
IOManager.hh, 992
id
gazebo::common::SkeletonNode, 746
gazebo::physics::TrajectoryInfo, 821
Image
gazebo::common::Image, 362
Image.hh, 990
imageFormat
gazebo::rendering::Camera, 186
imageHeight
gazebo::rendering::Camera, 186
imageWidth
gazebo::rendering::Camera, 186
img
gazebo::physics::HeightmapShape, 356
ImuSensor
gazebo::sensors::ImuSensor, 367
ImuSensor.hh, 991
ImuSensor_V
gazebo::sensors, 102
ImuSensorPtr
gazebo::sensors, 102
IncCount
gazebo::transport::IOManager, 381
indexBufferCapacity
gazebo::rendering::DynamicRenderable, 273
inertiaRatio
gazebo::physics::Joint, 398
Inertial
gazebo::physics::Inertial, 371
inertial
gazebo::physics::Link, 437
Inertial.hh, 991
InertialPtr
gazebo::physics, 95
Init
Common, 36
gazebo::common::LogRecord, 450
gazebo::common::PID, 585
gazebo::Master, 453
gazebo::ModelPlugin, 505
gazebo::physics::Actor, 115
gazebo::physics::Base, 145
gazebo::physics::BoxShape, 155
gazebo::physics::Collision, 202
gazebo::physics::ContactManager, 241

gazebo::physics::CylinderShape, 252
gazebo::physics::Gripper, 345
gazebo::physics::HeightmapShape, 355
gazebo::physics::HingeJoint, 359
gazebo::physics::Joint, 394
gazebo::physics::Link, 432
gazebo::physics::Model, 497
gazebo::physics::MultiRayShape, 533
gazebo::physics::PhysicsEngine, 576
gazebo::physics::PlaneShape, 592
gazebo::physics::RayShape, 651
gazebo::physics::Road, 667
gazebo::physics::Shape, 722
gazebo::physics::SphereShape, 753
gazebo::physics::TrimeshShape, 824
gazebo::physics::World, 917
gazebo::rendering::Camera, 178
gazebo::rendering::DepthCamera, 257
gazebo::rendering::DynamicRenderable, 272
gazebo::rendering::FPSViewController, 323
gazebo::rendering::GpuLaser, 327
gazebo::rendering::Grid, 343
gazebo::rendering::GUIOverlay, 348
gazebo::rendering::OrbitViewController, 558
gazebo::rendering::RenderEngine, 657
gazebo::rendering::RTShaderSystem, 675
gazebo::rendering::Scene, 688
gazebo::rendering::SelectionObj, 697
gazebo::rendering::UserCamera, 835
gazebo::rendering::ViewController, 883, 884
gazebo::rendering::Visual, 896
gazebo::rendering::WireBox, 910
gazebo::SensorPlugin, 714
gazebo::sensors::CameraSensor, 191
gazebo::sensors::ContactSensor, 245
gazebo::sensors::DepthCameraSensor, 260
gazebo::sensors::GpuRaySensor, 337
gazebo::sensors::ImuSensor, 368
gazebo::sensors::MultiCameraSensor, 526
gazebo::sensors::RaySensor, 646
gazebo::sensors::RFIDSensor, 659
gazebo::sensors::RFIDTag, 662
gazebo::sensors::Sensor, 704
gazebo::sensors::SensorManager, 711
gazebo::Server, 715
gazebo::SystemPlugin, 790
gazebo::transport::ConnectionManager, 231
gazebo::transport::Node, 539
gazebo::transport::PublicationTransport, 618
gazebo::transport::SubscribeOptions, 776
gazebo::transport::SubscriptionTransport, 779
gazebo::transport::TopicManager, 819
gazebo::util::DiagnosticManager, 263
gazebo::VisualPlugin, 905
gazebo::WorldPlugin, 923
Messages, 60
init
gazebo, 83
Rendering, 67
sdf, 107
Sensors, 72
Transport, 78
init_world
Classes for physics and dynamics, 45
init_worlds
Classes for physics and dynamics, 45
initDoc
sdf, 107, 108
initFile
sdf, 108
InitForThread
gazebo::physics::PhysicsEngine, 576
InitModelDoc
urdf2gazebo::URDF2Gazebo, 828
InitModelFile
urdf2gazebo::URDF2Gazebo, 829
InitModelString
urdf2gazebo::URDF2Gazebo, 829
initString
sdf, 108
initXml
sdf, 108
initialTransform
gazebo::common::SkeletonNode, 746
initialized
gazebo::rendering::Camera, 186
InsertElement
sdf::Element, 279
InsertLatchedMsg
gazebo::transport::Node, 539
InsertMesh
gazebo::rendering::Visual, 897
InsertModelFile
gazebo::physics::World, 917
InsertModelSDF
gazebo::physics::World, 918
InsertModelString
gazebo::physics::World, 918
Instance
SingletonT, 727
InternalError
gazebo::common::InternalError, 379
Interpolate
gazebo::math::RotationSpline, 670, 671
gazebo::math::Spline, 756
interpolateX
gazebo::physics::Actor, 116
invBindTransform

gazebo::common::SkeletonNode, 746
Inverse
gazebo::math::Matrix4, 470
Invert
gazebo::math::Quaternion, 630
is_stopped
Transport, 78
IsActive
gazebo::physics::Actor, 115
gazebo::rendering::SelectionObj, 697
gazebo::sensors::CameraSensor, 191
gazebo::sensors::ContactSensor, 245
gazebo::sensors::MultiCameraSensor, 526
gazebo::sensors::RaySensor, 646
gazebo::sensors::Sensor, 704
IsAdvertised
gazebo::transport::TopicManager, 819
IsAffine
gazebo::math::Matrix4, 471
IsAnimating
gazebo::rendering::Camera, 178
IsBool
sdf::Param, 562
IsCanonicalLink
gazebo::physics::Entity, 288
IsChar
sdf::Param, 562
IsColor
sdf::Param, 562
IsDouble
sdf::Param, 563
IsFinite
gazebo::math::Pose, 601
gazebo::math::Quaternion, 630
gazebo::math::Vector2d, 841
gazebo::math::Vector2i, 849
gazebo::math::Vector3, 862
gazebo::math::Vector4, 872
IsFloat
sdf::Param, 563
IsHorizontal
gazebo::sensors::GpuRaySensor, 337
isHorizontal
gazebo::sensors::GpuRaySensor, 339
IsInitialized
Common, 36
gazebo::rendering::Camera, 179
gazebo::rendering::GUIOverlay, 348
IsInt
sdf::Param, 563
IsJoint
gazebo::common::SkeletonNode, 744
IsLoaded
gazebo::physics::World, 918
IsLocal
gazebo::transport::CallbackHelper, 159
gazebo::transport::CallbackHelperT, 162
gazebo::transport::RawCallbackHelper, 640
gazebo::transport::SubscriptionTransport, 780
IsOpen
gazebo::common::LogPlay, 446
gazebo::transport::Connection, 227
IsPaused
gazebo::physics::World, 918
IsPlaceable
gazebo::physics::Collision, 202
IsPlane
gazebo::rendering::Visual, 897
IsPose
sdf::Param, 563
isPowerOfTwo
Math, 51
IsQuaternion
sdf::Param, 563
IsRegistered
gazebo::physics::PhysicsFactory, 582
IsRootNode
gazebo::common::SkeletonNode, 744
IsRunning
gazebo::transport::ConnectionManager, 231
IsSelected
gazebo::physics::Base, 145
IsStatic
gazebo::physics::Entity, 288
gazebo::rendering::Visual, 897
IsStr
sdf::Param, 563
IsTime
sdf::Param, 563
IsUInt
sdf::Param, 563
IsValidFilename
gazebo::common::MeshManager, 488
IsVector2d
sdf::Param, 563
IsVector2i
sdf::Param, 563
IsVector3
sdf::Param, 563
IsVisible
gazebo::rendering::Camera, 179
IsZero
gazebo::physics::CollisionState, 207
gazebo::physics::JointState, 403
gazebo::physics::LinkState, 442
gazebo::physics::ModelState, 511
gazebo::physics::WorldState, 926
isnan

Math, 50

JOINT
gazebo::common::SkeletonNode, 740
gazebo::physics::Base, 141
Joint
gazebo::physics::Joint, 386
Joint.hh, 994
MAX_JOINT_AXIS, 995
Joint_V
gazebo::physics, 95
JointController
gazebo::physics::JointController, 400
JointController.hh, 995
JointController_V
gazebo::physics, 95
JointControllerPtr
gazebo::physics, 95
JointPtr
gazebo::physics, 95
JointState
gazebo::physics::JointState, 402
JointState.hh, 996
JointVisual
gazebo::rendering::JointVisual, 406
JointVisual.hh, 997
JointVisualPtr
gazebo::rendering, 99
JointWrench.hh, 998

kd
gazebo::physics::SurfaceParams, 783
key
sdf::Param, 564
KeyFrame
gazebo::common::KeyFrame, 409
KeyFrame.hh, 999
KeyFrame_V
gazebo::common::Animation, 126
keyFrames
gazebo::common::Animation, 128
gazebo::common::NodeAnimation, 545
kp
gazebo::physics::SurfaceParams, 783

L_INT16
gazebo::common::Image, 361
L_INT8
gazebo::common::Image, 361
LEFT
gazebo::common::MouseEvent, 513
LIGHT
gazebo::physics::Base, 141
LINE_MAX_LEN
STLLoader.hh, 1098
LINES
gazebo::common::SubMesh, 766
LINESTRIPS
gazebo::common::SubMesh, 766
LINK
gazebo::physics::Base, 140
LINUX
SystemPaths.hh, 1105
LO_STOP
gazebo::physics::Joint, 386
Lap
gazebo::util::DiagnosticManager, 263
gazebo::util::DiagnosticTimer, 265
LaserVisual
gazebo::rendering::LaserVisual, 411
LaserVisual.hh, 1000
LaserVisualPtr
gazebo::rendering, 99
lastMeasurementTime
gazebo::sensors::Sensor, 706
lastPos
gazebo::physics::Actor, 116
lastRenderWallTime
gazebo::rendering::Camera, 186
lastScriptTime
gazebo::physics::Actor, 116
lastTraj
gazebo::physics::Actor, 116
lastUpdateTime
gazebo::sensors::Sensor, 706
latching
gazebo::transport::CallbackHelper, 160
length
gazebo::common::Animation, 129
gazebo::common::NodeAnimation, 545
gazebo::common::SkeletonAnimation, 737
Light
gazebo::rendering::Light, 414
Light.hh, 1001
LightFromSDF
Messages, 61
LightPtr
gazebo::rendering, 99
LightingModel
gazebo::rendering::RTShaderSystem, 674
linearAccel
gazebo::physics::Link, 438
Link
gazebo::physics::Link, 423
link
gazebo::physics::Collision, 204
Link.hh, 1002
Link_V
gazebo::physics, 95

LinkPtr
gazebo::physics, 96
LinkState
gazebo::physics::LinkState, 440
LinkState.hh, 1003
Listen
gazebo::transport::Connection, 228
Load
gazebo::common::BVHLoader, 156
gazebo::common::ColladaLoader, 195
gazebo::common::Image, 364
gazebo::common::MeshLoader, 484
gazebo::common::MeshManager, 488
gazebo::common::STLLoader, 763
gazebo::common::Video, 879
gazebo::ModelPlugin, 505
gazebo::physics::Actor, 115
gazebo::physics::BallJoint, 137
gazebo::physics::Base, 145
gazebo::physics::Collision, 202
gazebo::physics::CollisionState, 207
gazebo::physics::Entity, 288
gazebo::physics::Gripper, 345
gazebo::physics::HeightmapShape, 355
gazebo::physics::Hinge2Joint, 358
gazebo::physics::HingeJoint, 359
gazebo::physics::Inertial, 374
gazebo::physics::Joint, 394
gazebo::physics::JointState, 404
gazebo::physics::Link, 432
gazebo::physics::LinkState, 442
gazebo::physics::Model, 497
gazebo::physics::ModelState, 511
gazebo::physics::PhysicsEngine, 577
gazebo::physics::Road, 667
gazebo::physics::ScrewJoint, 693
gazebo::physics::SliderJoint, 749
gazebo::physics::State, 761
gazebo::physics::SurfaceParams, 781
gazebo::physics::UniversalJoint, 827
gazebo::physics::World, 918
gazebo::physics::WorldState, 927
gazebo::rendering::ArrowVisual, 131
gazebo::rendering::AxisVisual, 134
gazebo::rendering::Camera, 179
gazebo::rendering::CameraVisual, 194
gazebo::rendering::COMVisual, 221
gazebo::rendering::DepthCamera, 257
gazebo::rendering::GpuLaser, 327
gazebo::rendering::Heightmap, 351
gazebo::rendering::JointVisual, 406
gazebo::rendering::Light, 415
gazebo::rendering::MovableText, 519
gazebo::rendering::Projector, 610, 611
gazebo::rendering::RenderEngine, 657
gazebo::rendering::Road2d, 668
gazebo::rendering::Scene, 688
gazebo::rendering::UserCamera, 835
gazebo::rendering::Visual, 897, 898
gazebo::SensorPlugin, 714
gazebo::sensors::CameraSensor, 191, 192
gazebo::sensors::ContactSensor, 245
gazebo::sensors::DepthCameraSensor, 260
gazebo::sensors::GpuRaySensor, 337
gazebo::sensors::ImuSensor, 368
gazebo::sensors::MultiCameraSensor, 526
gazebo::sensors::RaySensor, 647
gazebo::sensors::RFIDSensor, 659
gazebo::sensors::RFIDTag, 662
gazebo::sensors::Sensor, 704
gazebo::SystemPlugin, 790
gazebo::VisualPlugin, 906
gazebo::WorldPlugin, 923
load
Classes for physics and dynamics, 45
gazebo, 83
Rendering, 67
Sensors, 73
load_world
Classes for physics and dynamics, 45
load_worlds
Classes for physics and dynamics, 45
LoadFile
gazebo::Server, 715
LoadFromMsg
gazebo::rendering::Heightmap, 351
gazebo::rendering::Light, 415
gazebo::rendering::Visual, 898
LoadJoints
gazebo::physics::Model, 498
LoadLayout
gazebo::rendering::GUIOverlay, 348
LoadPlugin
gazebo::physics::World, 919
gazebo::rendering::Visual, 898
LoadPlugins
gazebo::physics::Model, 498
LoadString
gazebo::Server, 715
LocalPublish
gazebo::transport::Publication, 615
Log
Common, 36
LogPlay.hh, 1005
LogRecord.hh, 1006
GZ_LOG_VERSION, 1008
Logplay, 446
loop

gazebo::common::Animation, 129
gazebo::physics::Actor, 116
lowerLimit
gazebo::physics::Joint, 398

m
gazebo::math::Matrix3, 467
gazebo::math::Matrix4, 474
MAP_SHAPE
gazebo::physics::Base, 141
MATRIX
gazebo::common::NodeTransform, 548
MAX_COLLIDE_RETURNS
Contact.hh, 962
MAX_CONTACT_JOINTS
Contact.hh, 962
MAX_JOINT_AXIS
Joint.hh, 995
MIDDLE
gazebo::common::MouseEvent, 513
MODEL
gazebo::physics::Base, 140
MODEL_PLUGIN
Common, 32
MODULATE
gazebo::common::Material, 456
MOVE
gazebo::common::MouseEvent, 514
MSleep
gazebo::common::Time, 797
MULTIRAY_SHAPE
gazebo::physics::Base, 141
mainLink
gazebo::physics::Actor, 116
mainpage.html, 1008
MakeStatic
gazebo::rendering::Visual, 898
MapShape.hh, 1008
Master
gazebo::Master, 453
Master.hh, 1009
Material
gazebo::common::Material, 456, 457
Material.hh, 1010, 1012
Math, 48
clamp, 50
equal, 50
isPowerOfTwo, 51
isnan, 50
max, 51
mean, 51
min, 51
NAN_D, 53
NAN_I, 53
parseFloat, 52
parseInt, 52
precision, 52
variance, 52
MathTypes.hh, 1012
Matrix3
gazebo::math::Matrix3, 464
Matrix3.hh, 1013
Matrix4
gazebo::math::Matrix4, 469
Matrix4.hh, 1013
max
gazebo::math::Box, 153
Math, 51
maxStepSize
gazebo::physics::PhysicsEngine, 581
maxVel
gazebo::physics::SurfaceParams, 783
mean
Math, 51
Merge
gazebo::math::Box, 151
Mesh
gazebo::common::Mesh, 476
mesh
gazebo::physics::Actor, 116
gazebo::physics::TrimeshShape, 825
Mesh.hh, 1014
MeshCSG
gazebo::common::MeshCSG, 482
MeshCSG.hh, 1016
GPtrArray, 1016
GtsSurface, 1016
MeshFromSDF
Messages, 61
MeshLoader
gazebo::common::MeshLoader, 483
MeshLoader.hh, 1017
MeshManager.hh, 1018
MeshShapePtr
gazebo::physics, 96
MessagePtr
gazebo::transport, 105
Messages, 54
Convert, 56–59
CreateRequest, 59
FogFromSDF, 59
GUIFromSDF, 60
GZ_REGISTER_STATIC_MSG, 56
GeometryFromSDF, 60
GetHeader, 60
Init, 60
LightFromSDF, 61
MeshFromSDF, 61

SceneFromSDF, 61
Set, 61–63
Stamp, 63
TrackVisualFromSDF, 63
VisualFromSDF, 64
MicToNano
gazebo::common::Time, 796
MilToNano
gazebo::common::Time, 796
min
gazebo::math::Box, 153
Math, 51
minDepth
gazebo::physics::SurfaceParams, 783
Model
gazebo::physics::Model, 492
model
gazebo::physics::Joint, 398
Model.hh, 1019
Model_V
gazebo::physics, 96
ModelDatabase.hh, 1021
GZ_MODEL_DB_MANIFEST_FILENAME, 1021
GZ_MODEL_MANIFEST_FILENAME, 1021
modelPathsFromEnv
gazebo::common::SystemPaths, 789
ModelPlugin
gazebo::ModelPlugin, 504
ModelPluginPtr
gazebo, 83
ModelPtr
gazebo::physics, 96
ModelState
gazebo::physics::ModelState, 507
ModelState.hh, 1022
modelTransform
gazebo::common::SkeletonNode, 746
MouseEvent
gazebo::common::MouseEvent, 514
MouseEvent.hh, 1024
MovableText
gazebo::rendering::MovableText, 517
MovableText.hh, 1025
moveScale
gazebo::common::MouseEvent, 514
MoveToPosition
gazebo::rendering::Camera, 180
gazebo::rendering::UserCamera, 835
gazebo::rendering::Visual, 898
MoveToPositions
gazebo::rendering::Camera, 180
gazebo::rendering::Visual, 898
MoveToVisual
gazebo::rendering::UserCamera, 836
Moved
gazebo::rendering::WindowManager, 908
MsgFactory.hh, 1025
MsgFactoryFn
gazebo::msgs, 91
msgs.hh, 1026
mu1
gazebo::physics::SurfaceParams, 783
mu2
gazebo::physics::SurfaceParams, 783
MultiCameraSensor
gazebo::sensors::MultiCameraSensor, 524
MultiCameraSensor.hh, 1029
MultiRayShape
gazebo::physics::MultiRayShape, 530
MultiRayShape.hh, 1030
MultiRayShapePtr
gazebo::physics, 96

NAN_D
Math, 53
NAN_I
Math, 53
NO_BUTTON
gazebo::common::MouseEvent, 513
NO_EVENT
gazebo::common::MouseEvent, 514
NODE
gazebo::common::SkeletonNode, 740
NONE
gazebo::rendering::RenderEngine, 655
NRealGen
gazebo::math, 89
NSleep
gazebo::common::Time, 797
NULL
CommonTypes.hh, 955
name
gazebo::common::Animation, 129
gazebo::common::Material, 462
gazebo::common::NodeAnimation, 545
gazebo::common::SkeletonAnimation, 738
gazebo::common::SkeletonNode, 746
gazebo::physics::State, 762
gazebo::rendering::Camera, 186
sdf::Plugin, 594
near
gazebo::sensors::GpuRaySensor, 339
NewContact
gazebo::physics::ContactManager, 241
newData
gazebo::rendering::Camera, 187
newImageFrame
gazebo::rendering::Camera, 187

newLaserScans
gazebo::physics::MultiRayShape, 534
NewMsg
gazebo::msgs::MsgFactory, 522
NewPhysicsEngine
gazebo::physics::PhysicsFactory, 582
NewSensor
gazebo::sensors::SensorFactory, 708
Node
gazebo::transport::Node, 537
node
gazebo::physics::Entity, 292
gazebo::physics::PhysicsEngine, 581
gazebo::sensors::Sensor, 706
Node.hh, 1031
NodeAnimation
gazebo::common::NodeAnimation, 543
nodeIndex
gazebo::common::NodeAssignment, 546
NodeMap
gazebo::common, 86
NodeMapIter
gazebo::common, 86
NodePtr
gazebo::transport, 105
NodeTransform
gazebo::common::NodeTransform, 548
nodes
gazebo::common::Skeleton, 733
normal
gazebo::math::Plane, 589
NormalRealDist
gazebo::math, 89
Normalize
gazebo::math::Angle, 120
gazebo::math::Quaternion, 630
gazebo::math::Vector2d, 841
gazebo::math::Vector2i, 849
gazebo::math::Vector3, 862
gazebo::math::Vector4, 872
normals
gazebo::physics::Contact, 239
notifyRenderSingleObject
gazebo::rendering::GpuLaser, 327
nsec
gazebo::common::Time, 812
NullStream
Common, 32
NumericAnimation
gazebo::common::NumericAnimation, 552
NumericAnimationPtr
gazebo::common, 86
NumericKeyFrame
gazebo::common::NumericKeyFrame, 554
ORDER_MAX
STLLoader.hh, 1098
OTHER
gazebo::sensors, 103
offset
gazebo::physics::MultiRayShape, 534
Ogre, 106
ogre, 106
ogre_gazebo.h, 1032
ogrePathsFromEnv
gazebo::common::SystemPaths, 789
oldAction
gazebo::physics::Actor, 116
onAnimationComplete
gazebo::rendering::Camera, 187
OnPhysicsMsg
gazebo::physics::PhysicsEngine, 577
OnPoseChange
gazebo::physics::Entity, 288
gazebo::physics::Link, 432
gazebo::physics::Model, 498
gazebo::rendering::Light, 415
OnRequest
gazebo::physics::PhysicsEngine, 577
One
gazebo::math::Vector3, 868
Open
gazebo::common::LogPlay, 446
operator<
gazebo::common::Time, 804, 805
gazebo::math::Angle, 122
operator<<
gazebo::common::Color, 218
gazebo::common::Exception, 320
gazebo::common::Material, 461
gazebo::common::Time, 811
gazebo::common::Timer, 814
gazebo::math::Angle, 124
gazebo::math::Box, 152
gazebo::math::Matrix3, 467
gazebo::math::Matrix4, 474
gazebo::math::Pose, 604
gazebo::math::Quaternion, 635
gazebo::math::Vector2d, 845
gazebo::math::Vector2i, 854
gazebo::math::Vector3, 867
gazebo::math::Vector4, 877
gazebo::physics::CollisionState, 208
gazebo::physics::Inertial, 378
gazebo::physics::JointState, 405
gazebo::physics::LinkState, 443
gazebo::physics::ModelState, 512
gazebo::physics::WorldState, 928
sdf::ParamT, 568

operator<=
gazebo::common::Time, 805, 806
gazebo::math::Angle, 123
operator>
gazebo::common::Time, 808, 809
gazebo::math::Angle, 123
operator>>
gazebo::common::Color, 218
gazebo::common::Time, 811
gazebo::math::Angle, 124
gazebo::math::Pose, 604
gazebo::math::Quaternion, 636
gazebo::math::Vector2d, 846
gazebo::math::Vector2i, 854
gazebo::math::Vector3, 868
gazebo::math::Vector4, 877
operator>=
gazebo::common::Time, 809, 810
gazebo::math::Angle, 123
operator*
gazebo::common::Color, 213
gazebo::common::NodeTransform, 549, 550
gazebo::common::Time, 798, 799
gazebo::math::Angle, 120
gazebo::math::Matrix3, 465, 467
gazebo::math::Matrix4, 471
gazebo::math::Pose, 601
gazebo::math::Quaternion, 630, 631
gazebo::math::Vector2d, 841
gazebo::math::Vector2i, 850
gazebo::math::Vector3, 862, 863, 867
gazebo::math::Vector4, 872, 873
sdf::ParamT, 567
operator*=
gazebo::common::Color, 213
gazebo::common::Time, 799, 800
gazebo::math::Angle, 121
gazebo::math::Quaternion, 631
gazebo::math::Vector2d, 842
gazebo::math::Vector2i, 850
gazebo::math::Vector3, 863
gazebo::math::Vector4, 873, 874
operator()
gazebo::common::NodeTransform, 549
gazebo::event::EventT, 312–314
operator+
gazebo::common::Color, 214
gazebo::common::Time, 800
gazebo::math::Angle, 121
gazebo::math::Box, 151
gazebo::math::Matrix3, 465
gazebo::math::Pose, 601
gazebo::math::Quaternion, 631
gazebo::math::Vector2d, 842
gazebo::math::Vector2i, 851
gazebo::math::Vector3, 863
gazebo::math::Vector4, 874
gazebo::physics::CollisionState, 207
gazebo::physics::Inertial, 374
gazebo::physics::JointState, 404
gazebo::physics::JointWrench, 407
gazebo::physics::LinkState, 442
gazebo::physics::ModelState, 511
gazebo::physics::WorldState, 927
operator+=
gazebo::common::Color, 214
gazebo::common::Time, 801
gazebo::math::Angle, 121
gazebo::math::Box, 151
gazebo::math::Pose, 602
gazebo::math::Quaternion, 632
gazebo::math::Vector2d, 842
gazebo::math::Vector2i, 851
gazebo::math::Vector3, 864
gazebo::math::Vector4, 874
gazebo::physics::Inertial, 375
operator-
gazebo::common::Color, 214, 215
gazebo::common::Time, 801, 802
gazebo::math::Angle, 121
gazebo::math::Box, 152
gazebo::math::Matrix3, 465
gazebo::math::Pose, 602
gazebo::math::Quaternion, 632
gazebo::math::Vector2d, 843
gazebo::math::Vector2i, 851
gazebo::math::Vector3, 864
gazebo::math::Vector4, 874
gazebo::physics::CollisionState, 207
gazebo::physics::JointState, 404
gazebo::physics::JointWrench, 407
gazebo::physics::LinkState, 443
gazebo::physics::ModelState, 511
gazebo::physics::State, 761
gazebo::physics::WorldState, 927
operator-=
gazebo::common::Color, 215
gazebo::common::Time, 802, 803
gazebo::math::Angle, 122
gazebo::math::Pose, 602
gazebo::math::Quaternion, 632
gazebo::math::Vector2d, 843
gazebo::math::Vector2i, 851
gazebo::math::Vector3, 864
gazebo::math::Vector4, 875
operator/
gazebo::common::Color, 215
gazebo::common::Time, 803

gazebo::math::Angle, 122
gazebo::math::Vector2d, 843
gazebo::math::Vector2i, 852
gazebo::math::Vector3, 864, 865
gazebo::math::Vector4, 875
operator/=
gazebo::common::Color, 216
gazebo::common::Time, 804
gazebo::math::Angle, 122
gazebo::math::Vector2d, 844
gazebo::math::Vector2i, 852, 853
gazebo::math::Vector3, 865
gazebo::math::Vector4, 875, 876
operator=
gazebo::common::Color, 216
gazebo::common::PID, 585
gazebo::common::Time, 806, 807
gazebo::math::Box, 152
gazebo::math::Matrix4, 471, 472
gazebo::math::Plane, 589
gazebo::math::Quaternion, 632
gazebo::math::Vector2d, 844
gazebo::math::Vector2i, 853
gazebo::math::Vector3, 865, 866
gazebo::math::Vector4, 876
gazebo::physics::CollisionState, 208
gazebo::physics::Contact, 237, 238
gazebo::physics::Inertial, 375
gazebo::physics::JointState, 404
gazebo::physics::JointWrench, 408
gazebo::physics::LinkState, 443
gazebo::physics::ModelState, 512
gazebo::physics::State, 761
gazebo::physics::WorldState, 927
operator==
gazebo::common::Color, 216
gazebo::common::Time, 807, 808
gazebo::math::Angle, 123
gazebo::math::Box, 152
gazebo::math::Matrix3, 465
gazebo::math::Matrix4, 472
gazebo::math::Pose, 603
gazebo::math::Quaternion, 633
gazebo::math::Vector2d, 845
gazebo::math::Vector2i, 854
gazebo::math::Vector3, 866
gazebo::math::Vector4, 876
gazebo::physics::Base, 145
operator[]
gazebo::common::Color, 216
gazebo::math::Matrix3, 465, 466
gazebo::math::Matrix4, 472
gazebo::math::Vector2d, 845
gazebo::math::Vector2i, 854
gazebo::math::Vector3, 866
gazebo::math::Vector4, 877
OrbitViewController
gazebo::rendering::OrbitViewController, 557
OrbitViewController.hh, 1033

PHONG
gazebo::common::Material, 456
PID
gazebo::common::PID, 584
PID.hh, 1044
PIXEL_FORMAT_COUNT
gazebo::common::Image, 362
PLANE_SHAPE
gazebo::physics::Base, 141
POINTS
gazebo::common::SubMesh, 766
PRESS
gazebo::common::MouseEvent, 514
Param
sdf::Param, 561
Param.hh, 1034
Param_V
gazebo::common, 86
sdf, 107
ParamPtr
sdf, 107
ParamT
sdf::ParamT, 566
ParamT< T >, 565
parent
gazebo::common::SkeletonNode, 747
gazebo::physics::Base, 148
gazebo::rendering::Visual, 904
parentEntity
gazebo::physics::Entity, 292
parentLink
gazebo::physics::Joint, 398
parentName
gazebo::sensors::Sensor, 706
ParseArgs
gazebo::Server, 715
parseFloat
Math, 52
parseInt
Math, 52
parser.hh, 1035
parser_urdf.hh, 1036
pathLength
gazebo::physics::Actor, 116
pause
gazebo::event::Events, 308
pause_incoming
Transport, 78

pause_world
Classes for physics and dynamics, 45
pause_worlds
Classes for physics and dynamics, 46
PauseIncoming
gazebo::transport::TopicManager, 819
Physics.hh, 1037
PhysicsEngine
gazebo::physics::PhysicsEngine, 571
PhysicsEngine.hh, 1039
PhysicsEnginePtr
gazebo::physics, 96
PhysicsFactory.hh, 1040
PhysicsFactoryFn
Classes for physics and dynamics, 44
physicsSub
gazebo::physics::PhysicsEngine, 581
PhysicsTypes.hh, 1042
GZ_ALL_COLLIDE, 1044
GZ_FIXED_COLLIDE, 1044
GZ_GHOST_COLLIDE, 1044
GZ_NONE_COLLIDE, 1044
GZ_SENSOR_COLLIDE, 1044
physicsUpdateMutex
gazebo::physics::PhysicsEngine, 581
pitchNode
gazebo::rendering::Camera, 187
PixelFormat
gazebo::common::Image, 361
PixelFormatNames
Common, 36
PlaceOnEntity
gazebo::physics::Entity, 289
PlaceOnNearestEntityBelow
gazebo::physics::Entity, 289
placeable
gazebo::physics::Collision, 204
Plane
gazebo::math::Plane, 588
Plane.hh, 1045
PlaneShape
gazebo::physics::PlaneShape, 591
PlaneShape.hh, 1046
Play
gazebo::physics::Actor, 115
playStartTime
gazebo::physics::Actor, 117
Plugin
sdf::Plugin, 594
Plugin.hh, 1048, 1052
pluginPathsFromEnv
gazebo::common::SystemPaths, 789
PluginType
Common, 32
plugins
gazebo::sensors::Sensor, 706
pointSize
gazebo::common::Material, 462
points
gazebo::math::RotationSpline, 672
gazebo::math::Spline, 758
pos
gazebo::common::MouseEvent, 514
gazebo::math::Pose, 604
Pose
gazebo::math::Pose, 598, 599
pose
gazebo::sensors::Sensor, 707
Pose.hh, 1052
PoseAnimation
gazebo::common::PoseAnimation, 606
PoseAnimationPtr
gazebo::common, 86
PoseKeyFrame
gazebo::common::PoseKeyFrame, 608
poseMsg
gazebo::physics::Entity, 292
poseSub
gazebo::sensors::Sensor, 707
positions
gazebo::physics::Contact, 239
PostRender
gazebo::rendering::Camera, 180
gazebo::rendering::DepthCamera, 257
gazebo::rendering::GpuLaser, 327
gazebo::rendering::UserCamera, 836
postRender
gazebo::event::Events, 308
PreRender
gazebo::rendering::Scene, 688
preRender
gazebo::event::Events, 308
precision
Math, 52
PrepareHardwareBuffers
gazebo::rendering::DynamicRenderable, 272
pressPos
gazebo::common::MouseEvent, 514
prevAnimTime
gazebo::rendering::Camera, 187
prevAnimationTime
gazebo::physics::Entity, 292
prevFrameTime
gazebo::physics::Actor, 117
prevPos
gazebo::common::MouseEvent, 515
PrimitiveType
gazebo::common::SubMesh, 766

Print
gazebo::common::Exception, 320
gazebo::physics::Base, 146
sdf::Plugin, 594
print_version
gazebo, 83
PrintDescription
sdf::Element, 279
sdf::SDF, 695
PrintDoc
sdf::SDF, 695
PrintDocLeftPane
sdf::Element, 279
PrintDocRightPane
sdf::Element, 279
PrintEntityTree
gazebo::physics::World, 919
PrintSceneGraph
gazebo::rendering::Scene, 688
PrintSource
gazebo::common::NodeTransform, 550
PrintTransforms
gazebo::common::Skeleton, 732
PrintUsage
gazebo::Server, 715
PrintValues
sdf::Element, 279
sdf::SDF, 695
PrintWiki
sdf::Element, 279
sdf::SDF, 695
ProcessIncoming
gazebo::transport::Node, 540
ProcessMsg
gazebo::physics::BoxShape, 155
gazebo::physics::Collision, 203
gazebo::physics::CylinderShape, 253
gazebo::physics::HeightmapShape, 356
gazebo::physics::Inertial, 375
gazebo::physics::Link, 432
gazebo::physics::Model, 498
gazebo::physics::MultiRayShape, 534
gazebo::physics::PlaneShape, 592
gazebo::physics::RayShape, 651
gazebo::physics::Shape, 722
gazebo::physics::SphereShape, 753
gazebo::physics::SurfaceParams, 782
gazebo::physics::TrimeshShape, 824
ProcessNodes
gazebo::transport::TopicManager, 819
ProcessPublishers
gazebo::transport::Node, 540
ProcessWriteQueue
gazebo::transport::Connection, 228
Projector
gazebo::rendering::Projector, 610
Projector.hh, 1053
Publication
gazebo::transport::Publication, 613
Publication.hh, 1054
PublicationPtr
gazebo::transport, 105
PublicationTransport
gazebo::transport::PublicationTransport, 617
PublicationTransport.hh, 1055
PublicationTransportPtr
gazebo::transport, 105
Publish
gazebo::transport::Node, 540
gazebo::transport::Publication, 615
gazebo::transport::Publisher, 621
gazebo::transport::TopicManager, 819
PublishContacts
gazebo::physics::ContactManager, 241
PublishModelPose
gazebo::physics::World, 919
PublishTask
gazebo::transport::PublishTask, 622
Publisher
gazebo::transport::Publisher, 619
Publisher.hh, 1057
PublisherPtr
gazebo::transport, 105
Purple
gazebo::common::Color, 219

Quaternion
gazebo::math::Quaternion, 626, 627
Quaternion.hh, 1059

r
gazebo::common::Color, 219
R_FLOAT16
gazebo::common::Image, 362
R_FLOAT32
gazebo::common::Image, 362
RAY
gazebo::sensors, 103
RAY_SHAPE
gazebo::physics::Base, 141
RELEASE
gazebo::common::MouseEvent, 514
RENDER_PATH_COUNT
gazebo::rendering::RenderEngine, 655
RENDERING_LINE_LIST
gazebo::rendering, 99
RENDERING_LINE_STRIP
gazebo::rendering, 99
RENDERING_MESH_RESOURCE

gazebo::rendering, 100
RENDERING_POINT_LIST
gazebo::rendering, 99
RENDERING_TRIANGLE_FAN
gazebo::rendering, 100
RENDERING_TRIANGLE_LIST
gazebo::rendering, 99
RENDERING_TRIANGLE_STRIP
gazebo::rendering, 100
REPLACE
gazebo::common::Material, 456
RFIDSensor
gazebo::sensors::RFIDSensor, 659
RFIDSensor.hh, 1068
RFIDSensor_V
gazebo::sensors, 102
RFIDSensorPtr
gazebo::sensors, 102
RFIDTag
gazebo::sensors::RFIDTag, 661
RFIDTag.hh, 1068
RFIDTag_V
gazebo::sensors, 102
RFIDTagPtr
gazebo::sensors, 102
RFIDTagVisual
gazebo::rendering::RFIDTagVisual, 663
RFIDTagVisual.hh, 1069
RFIDTagVisualPtr
gazebo::rendering, 99
RFIDVisual
gazebo::rendering::RFIDVisual, 665
RFIDVisual.hh, 1070
RFIDVisualPtr
gazebo::rendering, 99
RGB_FLOAT16
gazebo::common::Image, 362
RGB_FLOAT32
gazebo::common::Image, 362
RGB_INT16
gazebo::common::Image, 361
RGB_INT32
gazebo::common::Image, 361
RGB_INT8
gazebo::common::Image, 361
RGBA
gazebo::common::Color, 211
RGBA_INT8
gazebo::common::Image, 361
RIGHT
gazebo::common::MouseEvent, 513
ROTATE
gazebo::common::NodeTransform, 548
RTShaderSystem.hh, 1073
Radian
gazebo::math::Angle, 124
Rand.hh, 1060
rangeCountRatio
gazebo::sensors::GpuRaySensor, 339
rangeElem
gazebo::physics::MultiRayShape, 534
gazebo::sensors::GpuRaySensor, 339
RawCallbackHelper
gazebo::transport::RawCallbackHelper, 639
rawNW
gazebo::common::Skeleton, 733
RawNodeAnim
gazebo::common, 86
RawNodeWeights
gazebo::common, 86
RawSkeletonAnim
gazebo::common, 86
rawTransforms
gazebo::common::SkeletonNode, 747
rayCountRatio
gazebo::sensors::GpuRaySensor, 339
rayElem
gazebo::physics::MultiRayShape, 534
RaySensor
gazebo::sensors::RaySensor, 643
RaySensor.hh, 1061
RaySensor_V
gazebo::sensors, 102
RaySensorPtr
gazebo::sensors, 102
RayShape
gazebo::physics::RayShape, 649, 650
RayShape.hh, 1062
RayShapePtr
gazebo::physics, 96
rays
gazebo::physics::MultiRayShape, 535
Read
gazebo::transport::Connection, 228
ReadCallback
gazebo::transport::Connection, 224
readDoc
sdf, 108
readFile
sdf, 108
readString
sdf, 108
readXml
sdf, 108
realTime
gazebo::common::UpdateInfo, 827
gazebo::physics::State, 762
realTimeUpdateRate

gazebo::physics::PhysicsEngine, 581
RecalcTangents
gazebo::math::RotationSpline, 671
gazebo::math::Spline, 756
RecalculateMatrix
gazebo::common::NodeTransform, 550
RecalculateNormals
gazebo::common::Mesh, 480
gazebo::common::SubMesh, 772
Red
gazebo::common::Color, 219
RegisterAll
gazebo::physics::PhysicsFactory, 583
gazebo::sensors::SensorFactory, 708
RegisterMsg
gazebo::msgs::MsgFactory, 522
RegisterPhysicsEngine
gazebo::physics::PhysicsFactory, 583
RegisterSensor
gazebo::sensors::SensorFactory, 708
RegisterTopicNamespace
gazebo::transport::ConnectionManager, 232
gazebo::transport::TopicManager, 820
relativeEndPos
gazebo::physics::RayShape, 653
relativeStartPos
gazebo::physics::RayShape, 653
Remove
gazebo::common::LogRecord, 451
remove_scene
Rendering, 67
remove_sensor
Sensors, 73
remove_sensors
Sensors, 73
remove_worlds
Classes for physics and dynamics, 46
RemoveCallback
gazebo::transport::Node, 540
RemoveChild
gazebo::physics::Base, 146
gazebo::physics::Model, 498
sdf::Element, 279
RemoveChildJoint
gazebo::physics::Link, 433
RemoveChildren
gazebo::physics::Base, 146
RemoveConnection
gazebo::transport::ConnectionManager, 232
RemoveFromParent
sdf::Element, 279
RemoveNode
gazebo::transport::TopicManager, 820
RemoveParentJoint
gazebo::physics::Link, 433
RemovePlugin
gazebo::physics::World, 919
gazebo::rendering::Visual, 898
RemoveScene
gazebo::rendering::RenderEngine, 657
gazebo::rendering::RTShaderSystem, 675
removeScene
gazebo::rendering::Events, 296
RemoveSensor
gazebo::sensors::SensorManager, 711
RemoveSensors
gazebo::sensors::SensorManager, 711
RemoveShadows
gazebo::rendering::RTShaderSystem, 676
RemoveSubscription
gazebo::transport::Publication, 615
RemoveTransport
gazebo::transport::Publication, 616
RemoveVisual
gazebo::rendering::Scene, 688
Render
gazebo::rendering::Camera, 180
render
gazebo::event::Events, 308
RenderEngine.hh, 1063
RenderEvents.hh, 1064
RenderImpl
gazebo::rendering::Camera, 180
RenderOpType
gazebo::rendering, 99
RenderPathType
gazebo::rendering::RenderEngine, 655
renderTarget
gazebo::rendering::Camera, 187
renderTexture
gazebo::rendering::Camera, 187
RenderTypes.hh, 1066
GZ_VISIBILITY_ALL, 1067
GZ_VISIBILITY_GUI, 1067
GZ_VISIBILITY_NOT_SELECTABLE, 1067
GZ_VISIBILITY_SELECTION, 1067
Rendering, 65
create_scene, 67
fini, 67
get_scene, 67
init, 67
load, 67
remove_scene, 67
Rendering.hh, 1065
request
Transport, 78
requestNoReply
Transport, 79

requestPub
gazebo::physics::Entity, 292
requestSub
gazebo::physics::PhysicsEngine, 581
requests
gazebo::rendering::Camera, 187
required
sdf::Param, 564
Rescale
gazebo::common::Image, 364
Reset
gazebo::common::Color, 217
gazebo::common::PID, 585
gazebo::common::SkeletonNode, 744
gazebo::math::Pose, 603
gazebo::ModelPlugin, 505
gazebo::physics::Base, 146
gazebo::physics::Contact, 238
gazebo::physics::Entity, 289
gazebo::physics::Inertial, 375
gazebo::physics::Joint, 394
gazebo::physics::JointController, 400
gazebo::physics::Link, 433
gazebo::physics::Model, 498
gazebo::physics::PhysicsEngine, 577
gazebo::physics::World, 919
gazebo::SensorPlugin, 714
gazebo::SystemPlugin, 790
gazebo::VisualPlugin, 906
gazebo::WorldPlugin, 923
sdf::Element, 280
sdf::Param, 563
sdf::ParamT, 567
ResetCount
gazebo::physics::ContactManager, 241
ResetEntities
gazebo::physics::World, 919
ResetLastUpdateTime
gazebo::sensors::Sensor, 705
ResetLastUpdateTimes
gazebo::sensors::SensorManager, 711
ResetPhysicsStates
gazebo::physics::Link, 433
ResetTime
gazebo::physics::World, 920
Resize
gazebo::rendering::GUIOverlay, 349
gazebo::rendering::UserCamera, 836
gazebo::rendering::WindowManager, 908
responsePub
gazebo::physics::PhysicsEngine, 581
Road, 667
gazebo::physics::Road, 666
Road.hh, 1070
Road2d
gazebo::rendering::Road2d, 668
Road2d.hh, 1072
RoadPtr
gazebo::physics, 96
root
gazebo::common::Skeleton, 733
gazebo::rendering::RenderEngine, 657
sdf::SDF, 696
rot
gazebo::math::Pose, 605
Rotate
gazebo::physics::Inertial, 375
rotate
gazebo::common::PoseKeyFrame, 609
RotatePitch
gazebo::rendering::Camera, 180
RotatePositionAboutOrigin
gazebo::math::Pose, 603
RotateVector
gazebo::math::Quaternion, 633
RotateVectorReverse
gazebo::math::Quaternion, 633
RotateYaw
gazebo::rendering::Camera, 181
RotationSpline
gazebo::math::RotationSpline, 669
RotationSpline.hh, 1072
Round
gazebo::math::Pose, 603
gazebo::math::Quaternion, 633
gazebo::math::Vector3, 866
Run
gazebo::Master, 453
gazebo::physics::World, 920
gazebo::sensors::SensorManager, 711
gazebo::Server, 715
gazebo::transport::ConnectionManager, 232
run
gazebo, 83
Sensors, 73
Transport, 79
run_once
Sensors, 73
run_threads
Sensors, 73
run_world
Classes for physics and dynamics, 46
run_worlds
Classes for physics and dynamics, 46
RunOnce
gazebo::Master, 453
RunThread
gazebo::Master, 453

RunThreads
gazebo::sensors::SensorManager, 712
RunUpdate
gazebo::transport::ConnectionManager, 232

SCALE
gazebo::common::NodeTransform, 548
SCREW_JOINT
gazebo::physics::Base, 141
SCROLL
gazebo::common::MouseEvent, 514
SDF
sdf::SDF, 695
SDF.hh, 1077
SDF_VERSION, 1078
SDF_VERSION
SDF.hh, 1078
SDFPtr
sdf, 107
SENSOR_PLUGIN
Common, 32
SHADE_COUNT
gazebo::common::Material, 456
SHAPE
gazebo::physics::Base, 141
SLIDER_JOINT
gazebo::physics::Base, 141
SM2Profile
gazebo::rendering::GzTerrainMatGen::SM2Profile, 751
SPHERE_SHAPE
gazebo::physics::Base, 141
SSLM_NormalMapLightingObjectSpace
gazebo::rendering::RTShaderSystem, 674
SSLM_NormalMapLightingTangentSpace
gazebo::rendering::RTShaderSystem, 674
SSLM_PerPixelLighting
gazebo::rendering::RTShaderSystem, 674
SSLM_PerVertexLighting
gazebo::rendering::RTShaderSystem, 674
STLLoader
gazebo::common::STLLoader, 763
STLLoader.hh, 1096
COR3_MAX, 1097
FACE_MAX, 1097
LINE_MAX_LEN, 1098
ORDER_MAX, 1098
STOP_CFM
gazebo::physics::Joint, 386
STOP_ERP
gazebo::physics::Joint, 386
SUSPENSION_CFM
gazebo::physics::Joint, 385
SUSPENSION_ERP
gazebo::physics::Joint, 385
SYSTEM_PLUGIN
Common, 32
Save
gazebo::physics::World, 920
saveCount
gazebo::rendering::Camera, 187
SaveFrame
gazebo::rendering::Camera, 181
gazebo::sensors::CameraSensor, 192
gazebo::sensors::DepthCameraSensor, 260
gazebo::sensors::MultiCameraSensor, 526
saveFrameBuffer
gazebo::rendering::Camera, 187
SavePNG
gazebo::common::Image, 365
Scale
gazebo::common::Mesh, 480
gazebo::common::NodeAnimation, 545
gazebo::common::Skeleton, 732
gazebo::common::SkeletonAnimation, 737
gazebo::common::SubMesh, 772
gazebo::math::Quaternion, 634
scale
gazebo::physics::HeightmapShape, 356
ScaleXAxis
gazebo::rendering::AxisVisual, 134
ScaleYAxis
gazebo::rendering::AxisVisual, 134
ScaleZAxis
gazebo::rendering::AxisVisual, 135
scanElem
gazebo::physics::MultiRayShape, 535
gazebo::sensors::GpuRaySensor, 340
Scene
gazebo::rendering::Scene, 680
scene
gazebo::rendering::Camera, 188
gazebo::rendering::Visual, 904
Scene.hh, 1074
SceneFromSDF
Messages, 61
sceneNode
gazebo::rendering::Camera, 188
gazebo::rendering::Visual, 904
ScenePtr
gazebo::rendering, 99
screenshotPath
gazebo::rendering::Camera, 188
ScrewJoint
gazebo::physics::ScrewJoint, 693
ScrewJoint.hh, 1075
scriptLength
gazebo::physics::Actor, 117

scroll
gazebo::common::MouseEvent, 515
sdf, 106
addNestedModel, 107
copyChildren, 107
ElementPtr, 107
ElementPtr_V, 107
gazebo::physics::Base, 148
gazebo::physics::PhysicsEngine, 581
gazebo::rendering::Camera, 188
gazebo::sensors::Sensor, 707
init, 107
initDoc, 107, 108
initFile, 108
initString, 108
initXml, 108
Param_V, 107
ParamPtr, 107
readDoc, 108
readFile, 108
readString, 108
readXml, 108
SDFPtr, 107
sdf.hh, 1076
sdf::Converter, 249
Convert, 250
sdf::Element, 273
∼Element, 276
AddAttribute, 276
AddElement, 276
AddElementDescription, 276
AddValue, 276
ClearElements, 276
Clone, 276
Copy, 277
Element, 276
GetAttribute, 277
GetAttributeCount, 277
GetAttributeSet, 277
GetCopyChildren, 277
GetDescription, 277
GetElement, 277
GetElementDescription, 277
GetElementDescriptionCount, 277
GetElementImpl, 278
GetFirstElement, 278
GetInclude, 278
GetName, 278
GetNextElement, 278
GetParent, 278
GetRequired, 278
GetValue, 278
GetValueBool, 278
GetValueChar, 278
GetValueColor, 278
GetValueDouble, 278
GetValueFloat, 278
GetValueInt, 278
GetValuePose, 278
GetValueQuaternion, 278
GetValueString, 278
GetValueTime, 278
GetValueUInt, 278
GetValueVector2d, 278
GetValueVector3, 278
HasAttribute, 278
HasElement, 279
HasElementDescription, 279
InsertElement, 279
PrintDescription, 279
PrintDocLeftPane, 279
PrintDocRightPane, 279
PrintValues, 279
PrintWiki, 279
RemoveChild, 279
RemoveFromParent, 279
Reset, 280
Set, 280
SetCopyChildren, 280
SetDescription, 280
SetInclude, 280
SetName, 280
SetParent, 280
SetRequired, 280
ToString, 280
Update, 281
sdf::Param, 559
∼Param, 561
Clone, 561
description, 564
Get, 561, 562
GetAsString, 562
GetDefaultAsString, 562
GetDescription, 562
GetKey, 562
GetRequired, 562
GetSet, 562
GetTypeName, 562
IsBool, 562
IsChar, 562
IsColor, 562
IsDouble, 563
IsFloat, 563
IsInt, 563
IsPose, 563
IsQuaternion, 563
IsStr, 563
IsTime, 563

IsUInt, 563
IsVector2d, 563
IsVector2i, 563
IsVector3, 563
key, 564
Param, 561
required, 564
Reset, 563
Set, 563, 564
set, 564
SetDescription, 564
SetFromString, 564
SetUpdateFunc, 564
typeName, 564
Update, 564
updateFunc, 565
sdf::ParamT
~ParamT, 566
Clone, 567
defaultValue, 568
GetAsString, 567
GetDefaultAsString, 567
GetDefaultValue, 567
GetValue, 567
operator<<, 568
operator*, 567
ParamT, 566
Reset, 567
Set, 567
SetFromString, 567
SetValue, 567
Update, 568
value, 568
sdf::ParamT< T >, 565
sdf::Plugin, 593
Clear, 594
data, 594
filename, 594
name, 594
Plugin, 594
Print, 594
sdf::SDF, 695
PrintDescription, 695
PrintDoc, 695
PrintValues, 695
PrintWiki, 695
root, 696
SDF, 695
SetFromString, 695
ToString, 695
version, 696
Write, 696
sec
gazebo::common::Time, 812
SecToNano
gazebo::common::Time, 810
SelectVisual
gazebo::rendering::Scene, 688
SelectionObj
gazebo::rendering::SelectionObj, 697
SelectionObj.hh, 1078
SendMessage
gazebo::transport::Publisher, 621
Sensor
gazebo::sensors::Sensor, 701
Sensor.hh, 1079
Sensor_V
gazebo::sensors, 102
SensorCategory
gazebo::sensors, 102
SensorFactor, 707
SensorFactory.hh, 1080
SensorFactoryFn
gazebo::sensors, 102
SensorManager.hh, 1081
SensorPlugin
gazebo::SensorPlugin, 713
SensorPluginPtr
gazebo, 83
SensorPtr
gazebo::sensors, 102
SensorTypes.hh, 1084
Sensors, 70
create_sensor, 72
fini, 72
GZ_REGISTER_STATIC_SENSOR, 71
get_sensor, 72
init, 72
load, 73
remove_sensor, 73
remove_sensors, 73
run, 73
run_once, 73
run_threads, 73
stop, 73
Sensors.hh, 1082
SensorsInitialized
gazebo::sensors::SensorManager, 712
Server
gazebo::Server, 715
Server.hh, 1085
Set
gazebo::common::Color, 217
gazebo::common::NodeTransform, 550
gazebo::common::Time, 810, 811
gazebo::math::Matrix4, 473
gazebo::math::Plane, 589
gazebo::math::Pose, 603, 604

gazebo::math::Quaternion, 634
gazebo::math::Vector2d, 845
gazebo::math::Vector2i, 854
gazebo::math::Vector3, 867
gazebo::math::Vector4, 877
Messages, 61–63
sdf::Element, 280
sdf::Param, 563, 564
sdf::ParamT, 567
set
sdf::Param, 564
SetActive
gazebo::rendering::SelectionObj, 697
gazebo::sensors::DepthCameraSensor, 260
gazebo::sensors::Sensor, 705
SetAltitude
gazebo::physics::PlaneShape, 592
SetAmbient
gazebo::common::Material, 459
gazebo::rendering::Visual, 899
SetAmbientColor
gazebo::rendering::Scene, 689
SetAnchor
gazebo::physics::Joint, 395
gazebo::physics::ScrewJoint, 694
gazebo::physics::SliderJoint, 749
SetAngle
gazebo::physics::Joint, 395
SetAngleMax
gazebo::sensors::GpuRaySensor, 337
SetAngleMin
gazebo::sensors::GpuRaySensor, 337
SetAngularAccel
gazebo::physics::Link, 434
gazebo::physics::Model, 498
SetAngularDamping
gazebo::physics::Link, 434
SetAngularVel
gazebo::physics::Link, 434
gazebo::physics::Model, 499
SetAnimation
gazebo::physics::Entity, 289
SetAspectRatio
gazebo::rendering::Camera, 181
SetAttenuation
gazebo::rendering::Light, 415
SetAttribute
gazebo::physics::Joint, 395
SetAutoCalculate
gazebo::math::RotationSpline, 671
gazebo::math::Spline, 757
SetAutoDisable
gazebo::physics::Link, 434
gazebo::physics::Model, 499
SetAutoDisableFlag
gazebo::physics::PhysicsEngine, 577
SetAxis
gazebo::physics::BallJoint, 137
gazebo::physics::Joint, 395
SetAxisMaterial
gazebo::rendering::AxisVisual, 135
SetBackgroundColor
gazebo::rendering::Scene, 689
SetBasePath
gazebo::common::LogRecord, 451
SetBaseline
gazebo::rendering::MovableText, 519
SetBindShapeTransform
gazebo::common::Skeleton, 732
SetBlendFactors
gazebo::common::Material, 459
SetBlendMode
gazebo::common::Material, 459
SetCallbackId
gazebo::transport::Subscriber, 777
SetCamera
gazebo::rendering::WindowManager, 908
SetCanonicalLink
gazebo::physics::Entity, 289
SetCaptureData
gazebo::rendering::Camera, 182
SetCaptureDataOnce
gazebo::rendering::Camera, 182
SetCastShadows
gazebo::rendering::Light, 416
gazebo::rendering::Visual, 899
SetCategoryBits
gazebo::physics::Collision, 203
SetCellCount
gazebo::rendering::Grid, 343
SetCellLength
gazebo::rendering::Grid, 343
SetCharHeight
gazebo::rendering::MovableText, 520
SetClipDist
gazebo::rendering::Camera, 182
SetCmd
gazebo::common::PID, 585
SetCmdMax
gazebo::common::PID, 586
SetCmdMin
gazebo::common::PID, 586
SetCoG
gazebo::physics::Inertial, 375, 376
SetCol
gazebo::math::Matrix3, 466
SetCollideBits
gazebo::physics::Collision, 203

SetCollideMode
gazebo::physics::Link, 434
gazebo::physics::Model, 499
SetCollision
gazebo::physics::Collision, 203
SetColor
gazebo::rendering::Grid, 343
gazebo::rendering::MovableText, 520
SetComponent
gazebo::common::NodeTransform, 550
SetContactMaxCorrectingVel
gazebo::physics::PhysicsEngine, 577
SetContactSurfaceLayer
gazebo::physics::PhysicsEngine, 578
SetContactsEnabled
gazebo::physics::Collision, 203
SetCopyChildren
sdf::Element, 280
SetDGain
gazebo::common::PID, 586
SetDamping
gazebo::physics::Joint, 395
SetDepthTarget
gazebo::rendering::DepthCamera, 257
SetDepthWrite
gazebo::common::Material, 460
SetDescription
sdf::Element, 280
sdf::Param, 564
SetDiffuse
gazebo::common::Material, 460
gazebo::rendering::Visual, 899
SetDiffuseColor
gazebo::rendering::Light, 416
SetDirection
gazebo::rendering::Light, 416
SetDistance
gazebo::rendering::OrbitViewController, 558
SetEmissive
gazebo::common::Material, 460
gazebo::rendering::LaserVisual, 412
gazebo::rendering::Visual, 899
SetEnabled
gazebo::physics::Link, 434
gazebo::physics::Model, 499
gazebo::rendering::ContactVisual, 247
gazebo::rendering::Projector, 611
gazebo::rendering::ViewController, 884
SetFiducial
gazebo::physics::RayShape, 651
SetFilename
gazebo::physics::TrimeshShape, 824
SetFocalPoint
gazebo::rendering::OrbitViewController, 558
gazebo::rendering::UserCamera, 836
SetFog
gazebo::rendering::Scene, 689
SetFontName
gazebo::rendering::MovableText, 520
SetForce
gazebo::physics::Joint, 396
gazebo::physics::Link, 435
SetFromABGR
gazebo::common::Color, 217
SetFromARGB
gazebo::common::Color, 217
SetFromAxes
gazebo::math::Matrix3, 466
SetFromAxis
gazebo::math::Matrix3, 466
gazebo::math::Quaternion, 634
SetFromBGRA
gazebo::common::Color, 217
SetFromData
gazebo::common::Image, 365
SetFromDegree
gazebo::math::Angle, 124
SetFromEuler
gazebo::math::Quaternion, 634, 635
SetFromHSV
gazebo::common::Color, 217
SetFromRGBA
gazebo::common::Color, 218
SetFromRadian
gazebo::math::Angle, 124
SetFromString
sdf::Param, 564
sdf::ParamT, 567
sdf::SDF, 695
SetFromYUV
gazebo::common::Color, 218
SetGravity
gazebo::physics::PhysicsEngine, 578
SetGravityMode
gazebo::physics::Link, 435
gazebo::physics::Model, 499
SetGrid
gazebo::rendering::Scene, 689
SetHFOV
gazebo::rendering::Camera, 182
SetHandle
gazebo::common::SkeletonNode, 744
SetHeight
gazebo::rendering::Grid, 343
SetHighStop
gazebo::physics::BallJoint, 137
gazebo::physics::Joint, 396
SetHighlight

gazebo::rendering::SelectionObj, 698
SetHighlighted
gazebo::rendering::Visual, 899
SetIGain
gazebo::common::PID, 586
SetIMax
gazebo::common::PID, 586
SetIMin
gazebo::common::PID, 586
SetIXX
gazebo::physics::Inertial, 377
SetIXY
gazebo::physics::Inertial, 377
SetIXZ
gazebo::physics::Inertial, 377
SetIYY
gazebo::physics::Inertial, 377
SetIYZ
gazebo::physics::Inertial, 377
SetIZZ
gazebo::physics::Inertial, 377
SetId
gazebo::common::SkeletonNode, 744
SetImageHeight
gazebo::rendering::Camera, 182
SetImageSize
gazebo::rendering::Camera, 182
SetImageWidth
gazebo::rendering::Camera, 183
SetInclude
sdf::Element, 280
SetIndexCount
gazebo::common::SubMesh, 772
SetInertiaMatrix
gazebo::physics::Inertial, 376
SetInertial
gazebo::physics::Link, 435
SetInitialRelativePose
gazebo::physics::Entity, 290
SetInitialTransform
gazebo::common::SkeletonNode, 745
SetInverseBindTransform
gazebo::common::SkeletonNode, 745
SetJointAnimation
gazebo::physics::Model, 499
SetJointPosition
gazebo::physics::JointController, 400
gazebo::physics::Model, 500
SetJointPositions
gazebo::physics::JointController, 400
gazebo::physics::Model, 500
SetKinematic
gazebo::physics::Link, 435
SetLaserRetro
gazebo::physics::Collision, 203
gazebo::physics::Link, 435
gazebo::physics::Model, 500
SetLength
gazebo::common::Animation, 128
gazebo::physics::CylinderShape, 253
gazebo::physics::RayShape, 652
SetLightType
gazebo::rendering::Light, 416
SetLighting
gazebo::common::Material, 460
SetLineWidth
gazebo::rendering::Grid, 343
SetLinearAccel
gazebo::physics::Link, 435
gazebo::physics::Model, 500
SetLinearDamping
gazebo::physics::Link, 436
SetLinearVel
gazebo::physics::Link, 436
gazebo::physics::Model, 501
SetLinkWorldPose
gazebo::physics::Model, 501
SetLocallyAdvertised
gazebo::transport::Publication, 616
SetLowStop
gazebo::physics::BallJoint, 137
gazebo::physics::Joint, 396
SetMOI
gazebo::physics::Inertial, 378
SetMass
gazebo::physics::Inertial, 378
SetMaterial
gazebo::rendering::Visual, 900
SetMaterialIndex
gazebo::common::SubMesh, 772
SetMaxContacts
gazebo::physics::PhysicsEngine, 578
SetMaxForce
gazebo::physics::Joint, 396
SetMaxStepSize
gazebo::physics::PhysicsEngine, 578
SetMesh
gazebo::physics::TrimeshShape, 825
SetModel
gazebo::physics::Joint, 396
SetModelTransform
gazebo::common::SkeletonNode, 745
SetName
gazebo::common::Mesh, 480
gazebo::common::NodeAnimation, 545
gazebo::common::SkeletonAnimation, 737
gazebo::common::SkeletonNode, 745
gazebo::common::SubMesh, 772

gazebo::physics::Base, 147
gazebo::physics::Entity, 290
gazebo::physics::State, 762
gazebo::rendering::Camera, 183
gazebo::rendering::Light, 416
gazebo::rendering::Visual, 900
sdf::Element, 280
SetNormal
gazebo::common::SubMesh, 772
gazebo::physics::PlaneShape, 592
SetNormalCount
gazebo::common::SubMesh, 773
SetNormalMap
gazebo::rendering::Visual, 900
SetNumVertAttached
gazebo::common::Skeleton, 733
SetOperationType
gazebo::rendering::DynamicRenderable, 273
SetPGain
gazebo::common::PID, 587
SetParam
gazebo::physics::PhysicsEngine, 578
SetParams
gazebo::Server, 715
SetParent
gazebo::common::SkeletonNode, 745
gazebo::physics::Base, 147
gazebo::sensors::CameraSensor, 192
gazebo::sensors::DepthCameraSensor, 261
gazebo::sensors::Sensor, 705
sdf::Element, 280
SetParentSensor
gazebo::rendering::GpuLaser, 327
SetPath
gazebo::common::Mesh, 480
SetPaused
gazebo::common::LogRecord, 451
gazebo::physics::World, 920
SetPerPixelLighting
gazebo::rendering::RTShaderSystem, 676
SetPoint
gazebo::rendering::DynamicLines, 269
SetPointSize
gazebo::common::Material, 460
SetPoints
gazebo::physics::RayShape, 652
SetPose
gazebo::rendering::Visual, 900
SetPosition
gazebo::rendering::Light, 416
gazebo::rendering::Visual, 900
SetPrimitiveType
gazebo::common::SubMesh, 773
SetPublication
gazebo::transport::Publisher, 621
SetQuiet
Common, 36
SetRadius
gazebo::physics::CylinderShape, 253
gazebo::physics::SphereShape, 753
SetRange
gazebo::rendering::Light, 417
SetRangeCount
gazebo::rendering::GpuLaser, 328
SetRealTimeUpdateRate
gazebo::physics::PhysicsEngine, 578
SetReferencePose
gazebo::sensors::ImuSensor, 368
SetRelativePose
gazebo::physics::Entity, 290
SetRenderRate
gazebo::rendering::Camera, 183
SetRenderTarget
gazebo::rendering::Camera, 183
gazebo::rendering::UserCamera, 836
SetRequired
sdf::Element, 280
SetRetro
gazebo::physics::RayShape, 652
SetRibbonTrail
gazebo::rendering::Visual, 900
SetRootNode
gazebo::common::Skeleton, 733
SetRotation
gazebo::common::PoseKeyFrame, 609
gazebo::rendering::Visual, 901
SetSID
gazebo::common::NodeTransform, 550
SetSORPGSIters
gazebo::physics::PhysicsEngine, 579
SetSORPGSPreconIters
gazebo::physics::PhysicsEngine, 579
SetSORPGSW
gazebo::physics::PhysicsEngine, 579
SetSaveFramePathname
gazebo::rendering::Camera, 183
SetSaveable
gazebo::physics::Base, 147
SetScale
gazebo::common::Mesh, 481
gazebo::common::SubMesh, 773
gazebo::math::Matrix4, 473
gazebo::physics::TrimeshShape, 825
gazebo::rendering::Visual, 901
SetScene
gazebo::rendering::Camera, 183
gazebo::rendering::Visual, 901
SetSceneNode

gazebo::rendering::Camera, 183
SetSeed
gazebo::math::Rand, 638
gazebo::physics::PhysicsEngine, 579
SetSelected
gazebo::physics::Base, 147
gazebo::physics::Link, 436
gazebo::rendering::Light, 417
setSelectedEntity
gazebo::event::Events, 308
SetSelfCollide
gazebo::physics::Link, 436
SetShadeMode
gazebo::common::Material, 460
SetShaderType
gazebo::rendering::Visual, 901
SetShadowsEnabled
gazebo::rendering::Scene, 689
SetShape
gazebo::physics::Collision, 204
SetShininess
gazebo::common::Material, 460
SetShowOnTop
gazebo::rendering::MovableText, 520
SetSimTime
gazebo::physics::World, 920
SetSize
gazebo::physics::BoxShape, 155
gazebo::physics::CylinderShape, 253
gazebo::physics::PlaneShape, 593
SetSkeleton
gazebo::common::Mesh, 481
SetSkeletonPose
gazebo::rendering::Visual, 901
SetSourceValues
gazebo::common::NodeTransform, 550, 551
SetSpaceWidth
gazebo::rendering::MovableText, 520
SetSpecular
gazebo::common::Material, 461
gazebo::rendering::Visual, 902
SetSpecularColor
gazebo::rendering::Light, 417
SetSpotFalloff
gazebo::rendering::Light, 417
SetSpotInnerAngle
gazebo::rendering::Light, 417
SetSpotOuterAngle
gazebo::rendering::Light, 417
SetState
gazebo::physics::Collision, 204
gazebo::physics::Joint, 397
gazebo::physics::Link, 436
gazebo::physics::Model, 501
gazebo::physics::World, 920
SetStatic
gazebo::physics::Entity, 290
SetStepTime
gazebo::physics::PhysicsEngine, 579
SetSubMeshCenter
gazebo::common::SubMesh, 773
SetTargetRealTimeFactor
gazebo::physics::PhysicsEngine, 580
SetTension
gazebo::math::Spline, 757
SetTexCoord
gazebo::common::SubMesh, 773
SetTexCoordCount
gazebo::common::SubMesh, 773
SetText
gazebo::rendering::MovableText, 520
SetTextAlignment
gazebo::rendering::MovableText, 521
SetTexture
gazebo::rendering::Projector, 611
SetTextureImage
gazebo::common::Material, 461
SetThreadPitch
gazebo::physics::ScrewJoint, 694
SetTime
gazebo::common::Animation, 128
SetToIdentity
gazebo::math::Quaternion, 635
SetToMax
gazebo::math::Vector3, 867
SetToMin
gazebo::math::Vector3, 867
SetToWallTime
gazebo::common::Time, 811
SetTorque
gazebo::physics::Link, 436
SetTransform
gazebo::common::SkeletonNode, 745
SetTranslate
gazebo::math::Matrix4, 473
SetTranslation
gazebo::common::PoseKeyFrame, 609
SetTransparency
gazebo::common::Material, 461
gazebo::rendering::Visual, 902
SetTransparent
gazebo::rendering::Scene, 690
SetType
gazebo::common::NodeTransform, 551
gazebo::common::SkeletonNode, 746
SetUpdateFunc
sdf::Param, 564
SetUpdateRate

gazebo::physics::PhysicsEngine, 580
gazebo::sensors::Sensor, 705
SetUserData
gazebo::rendering::Grid, 344
SetValue
gazebo::common::NumericKeyFrame, 555
sdf::ParamT, 567
SetVelocity
gazebo::physics::Joint, 397
SetVertex
gazebo::common::SubMesh, 774
SetVertexCount
gazebo::common::SubMesh, 774
SetVerticalAngleMax
gazebo::sensors::GpuRaySensor, 338
SetVerticalAngleMin
gazebo::sensors::GpuRaySensor, 338
SetViewController
gazebo::rendering::UserCamera, 837
SetViewportDimensions
gazebo::rendering::UserCamera, 837
SetVisibilityFlags
gazebo::rendering::Visual, 902
SetVisible
gazebo::rendering::Scene, 690
gazebo::rendering::Visual, 902
gazebo::rendering::WireBox, 910
SetWindowId
gazebo::rendering::Camera, 184
SetWireframe
gazebo::rendering::Scene, 690
gazebo::rendering::Visual, 902
SetWorld
gazebo::physics::Base, 147
gazebo::physics::WorldState, 928
SetWorldCFM
gazebo::physics::PhysicsEngine, 580
SetWorldERP
gazebo::physics::PhysicsEngine, 580
SetWorldPose
gazebo::physics::Entity, 290
gazebo::rendering::Camera, 184
gazebo::rendering::UserCamera, 837
gazebo::rendering::Visual, 903
SetWorldPosition
gazebo::rendering::Camera, 184
gazebo::rendering::Visual, 903
SetWorldRotation
gazebo::rendering::Camera, 184
gazebo::rendering::Visual, 903
SetWorldTwist
gazebo::physics::Entity, 291
ShadeMode
gazebo::common::Material, 456
shadeMode
gazebo::common::Material, 462
ShadeModeStr
gazebo::common::Material, 462
Shape
gazebo::physics::Shape, 721
shape
gazebo::physics::Collision, 204
Shape.hh, 1086
ShapePtr
gazebo::physics, 96
shift
gazebo::common::MouseEvent, 515
shininess
gazebo::common::Material, 462
Show
gazebo::rendering::GUIOverlay, 349
ShowBoundingBox
gazebo::rendering::Visual, 903
ShowCOM
gazebo::rendering::Visual, 903
ShowCOMs
gazebo::rendering::Scene, 690
ShowCollision
gazebo::rendering::Visual, 903
ShowCollisions
gazebo::rendering::Scene, 690
ShowContacts
gazebo::rendering::Scene, 690
ShowJoints
gazebo::rendering::Scene, 691
gazebo::rendering::Visual, 903
ShowRotation
gazebo::rendering::ArrowVisual, 131
gazebo::rendering::AxisVisual, 135
ShowSkeleton
gazebo::rendering::Visual, 904
ShowVisual
gazebo::rendering::Light, 418
ShowWireframe
gazebo::rendering::Camera, 184
Shutdown
gazebo::transport::Connection, 228
sid
gazebo::common::NodeTransform, 551
Signal
gazebo::event::EventT, 315–317
simTime
gazebo::common::UpdateInfo, 827
gazebo::physics::State, 762
SimTimeEventHandler
gazebo::sensors::SimTimeEventHandler, 724
SingletonT
∼SingletonT, 727

Instance, 727
SingletonT, 727
SingletonT, 727
SingletonT< T >, 725
SingletonT.hh, 1088
size
gazebo::math::Plane, 590
skelAnimation
gazebo::physics::Actor, 117
skelNodesMap
gazebo::physics::Actor, 117
Skeleton
gazebo::common::Skeleton, 729
skeleton
gazebo::physics::Actor, 117
Skeleton.hh, 1088
SkeletonAnimation
gazebo::common::SkeletonAnimation, 735
SkeletonAnimation.hh, 1090
SkeletonNode
gazebo::common::SkeletonNode, 740, 741
SkeletonNodeType
gazebo::common::SkeletonNode, 740
skinFile
gazebo::physics::Actor, 117
skinScale
gazebo::physics::Actor, 117
SkyX, 108
skyx
gazebo::rendering::Scene, 691
Sleep
gazebo::common::Time, 811
Slerp
gazebo::math::Quaternion, 635
SliderJoint
gazebo::physics::SliderJoint, 748
SliderJoint.hh, 1091
slip1
gazebo::physics::SurfaceParams, 784
slip2
gazebo::physics::SurfaceParams, 784
SnapVisualToNearestBelow
gazebo::rendering::Scene, 691
source
gazebo::common::NodeTransform, 551
specular
gazebo::common::Material, 462
SphereShape
gazebo::physics::SphereShape, 753
SphereShape.hh, 1093
SphereShapePtr
gazebo::physics, 96
Spline
gazebo::math::Spline, 755
Spline.hh, 1094
Squad
gazebo::math::Quaternion, 635
Stamp
Messages, 63
Start
gazebo::common::LogRecord, 451
gazebo::common::Timer, 814
gazebo::util::DiagnosticTimer, 265
startDelay
gazebo::physics::Actor, 117
StartRead
gazebo::transport::Connection, 228
startTime
gazebo::physics::TrajectoryInfo, 821
StartTimer
gazebo::util::DiagnosticManager, 264
State
gazebo::physics::State, 760
State.hh, 1095
Step
gazebo::common::LogPlay, 446
step
gazebo::event::Events, 308
StepWorld
gazebo::physics::World, 920
Stop
gazebo::common::LogRecord, 452
gazebo::common::Timer, 814
gazebo::Master, 453
gazebo::physics::Actor, 115
gazebo::physics::World, 921
gazebo::sensors::SensorManager, 712
gazebo::Server, 715
gazebo::transport::ConnectionManager, 232
gazebo::transport::IOManager, 381
gazebo::util::DiagnosticTimer, 266
stop
gazebo, 83
gazebo::event::Events, 308
Sensors, 73
Transport, 79
stop_world
Classes for physics and dynamics, 46
stop_worlds
Classes for physics and dynamics, 46
StopAnimation
gazebo::physics::Entity, 291
gazebo::physics::Model, 501
StopRead
gazebo::transport::Connection, 228
StopTimer
gazebo::util::DiagnosticManager, 264
StrStr_M

gazebo::common, 86
StripSceneName
gazebo::rendering::Scene, 691
StripWorldName
gazebo::physics::World, 921
SubMesh
gazebo::common::SubMesh, 766
SubNodeMap
gazebo::transport::TopicManager, 816
subSampling
gazebo::physics::HeightmapShape, 356
submesh
gazebo::physics::TrimeshShape, 825
Subscribe
gazebo::transport::ConnectionManager, 232
gazebo::transport::Node, 540, 541
gazebo::transport::TopicManager, 820
SubscribeOptions
gazebo::transport::SubscribeOptions, 775
SubscribeOptions.hh, 1098
Subscriber
gazebo::transport::Subscriber, 777
Subscriber.hh, 1099
SubscriberPtr
gazebo::transport, 105
SubscriptionTransport
gazebo::transport::SubscriptionTransport, 779
SubscriptionTransport.hh, 1101
SubscriptionTransportPtr
gazebo::transport, 105
SurfaceParams
gazebo::physics::SurfaceParams, 781
SurfaceParams.hh, 1102
SurfaceParamsPtr
gazebo::physics, 96
SystemPaths.hh, 1104
GetCurrentDir, 1105
LINUX, 1105
SystemPlugin
gazebo::SystemPlugin, 790
SystemPluginPtr
gazebo, 83
systemPluginsArgc
gazebo::Server, 715
systemPluginsArgv
gazebo::Server, 715

TPtr
gazebo::PluginT, 595
TRANSLATE
gazebo::common::NodeTransform, 548
TRIANGLES
gazebo::common::SubMesh, 766
TRIFANS
gazebo::common::SubMesh, 766
TRIMESH_SHAPE
gazebo::physics::Base, 141
TRISTRIPS
gazebo::common::SubMesh, 766
tangents
gazebo::math::RotationSpline, 672
gazebo::math::Spline, 758
targetRealTimeFactor
gazebo::physics::PhysicsEngine, 581
tension
gazebo::math::Spline, 758
texImage
gazebo::common::Material, 462
textureHeight
gazebo::rendering::Camera, 188
textureWidth
gazebo::rendering::Camera, 188
threadPitch
gazebo::physics::ScrewJoint, 694
Time
gazebo::common::Time, 795
time
gazebo::common::KeyFrame, 410
gazebo::physics::Contact, 239
gazebo::sensors::SimTimeEvent, 723
Time.hh, 1105
timePos
gazebo::common::Animation, 129
Timer
gazebo::common::Timer, 813
Timer.hh, 1106
ToString
sdf::Element, 280
sdf::SDF, 695
Toggle
gazebo::rendering::Projector, 611
ToggleShowVisual
gazebo::rendering::Light, 418
ToggleShowWireframe
gazebo::rendering::Camera, 184
ToggleVisible
gazebo::rendering::Visual, 904
TopicManager.hh, 1107
TrackVisual
gazebo::rendering::Camera, 184
TrackVisualFromSDF
Messages, 63
TrackVisualImpl
gazebo::rendering::Camera, 185
gazebo::rendering::UserCamera, 837
trajInfo
gazebo::physics::Actor, 117
trajectories

gazebo::physics::Actor, 117
transform
gazebo::common::NodeTransform, 551
gazebo::common::SkeletonNode, 747
TransformAffine
gazebo::math::Matrix4, 474
TransformType
gazebo::common::NodeTransform, 548
Translate
gazebo::common::Mesh, 481
gazebo::common::SubMesh, 774
gazebo::rendering::Camera, 185
translate
gazebo::common::PoseKeyFrame, 609
translated
gazebo::physics::TrajectoryInfo, 821
transparency
gazebo::common::Material, 462
Transport, 75
CallbackHelperPtr, 76
clear_buffers, 77
fini, 77
get_master_uri, 77
get_topic_namespaces, 77
getAdvertisedTopics, 77
getTopicMsgType, 78
init, 78
is_stopped, 78
pause_incoming, 78
request, 78
requestNoReply, 79
run, 79
stop, 79
Transport.hh, 1109
TransportTypes.hh, 1111
TrimeshShape
gazebo::physics::TrimeshShape, 823
TrimeshShape.hh, 1113
type
gazebo::common::MouseEvent, 515
gazebo::common::NodeTransform, 551
gazebo::common::SkeletonNode, 747
gazebo::physics::TrajectoryInfo, 821
gazebo::PluginT, 596
typeName
sdf::Param, 564
typeString
gazebo::rendering::ViewController, 884

UIntGen
gazebo::math, 89
UNION
gazebo::common::MeshCSG, 482
UNIVERSAL_JOINT
gazebo::physics::Base, 141
UNKNOWN_PIXEL_FORMAT
gazebo::common::Image, 361
URDF2Gazebo
urdf2gazebo::URDF2Gazebo, 828
URealGen
gazebo::math, 89
Unadvertise
gazebo::transport::ConnectionManager, 232
gazebo::transport::TopicManager, 820
UniformIntDist
gazebo::math, 89
UniformRealDist
gazebo::math, 89
UnitX
gazebo::math::Vector3, 868
UnitY
gazebo::math::Vector3, 868
UnitZ
gazebo::math::Vector3, 868
UniversalJoint
gazebo::physics::UniversalJoint, 826
UniversalJoint.hh, 1114
Unsubscribe
gazebo::transport::ConnectionManager, 233
gazebo::transport::Subscriber, 777
gazebo::transport::TopicManager, 820
Update
gazebo::common::PID, 587
gazebo::physics::Actor, 115
gazebo::physics::Base, 147
gazebo::physics::Joint, 397
gazebo::physics::JointController, 401
gazebo::physics::Link, 437
gazebo::physics::Model, 501
gazebo::physics::MultiRayShape, 534
gazebo::physics::RayShape, 652
gazebo::physics::TrimeshShape, 825
gazebo::rendering::Camera, 185
gazebo::rendering::DynamicLines, 269
gazebo::rendering::FPSViewController, 323
gazebo::rendering::GUIOverlay, 349
gazebo::rendering::MovableText, 521
gazebo::rendering::OrbitViewController, 559
gazebo::rendering::UserCamera, 838
gazebo::rendering::ViewController, 884
gazebo::rendering::Visual, 904
gazebo::sensors::Sensor, 705
gazebo::sensors::SensorManager, 712
sdf::Element, 281
sdf::Param, 564
sdf::ParamT, 568
UpdateChildrenTransforms
gazebo::common::SkeletonNode, 746

UpdateCollision
gazebo::physics::PhysicsEngine, 580
UpdateFromMsg
gazebo::rendering::Light, 418
gazebo::rendering::Visual, 904
updateFunc
sdf::Param, 565
UpdateImpl
gazebo::sensors::CameraSensor, 192
gazebo::sensors::ContactSensor, 245
gazebo::sensors::DepthCameraSensor, 261
gazebo::sensors::GpuRaySensor, 338
gazebo::sensors::ImuSensor, 368
gazebo::sensors::MultiCameraSensor, 527
gazebo::sensors::RaySensor, 647
gazebo::sensors::RFIDSensor, 660
gazebo::sensors::RFIDTag, 662
gazebo::sensors::Sensor, 706
UpdateInfo.hh, 1115
UpdateMass
gazebo::physics::Link, 437
UpdateParameters
gazebo::physics::Actor, 115
gazebo::physics::Base, 148
gazebo::physics::Collision, 204
gazebo::physics::Entity, 291
gazebo::physics::Inertial, 378
gazebo::physics::Joint, 397
gazebo::physics::Link, 437
gazebo::physics::Model, 502
UpdateParams
gazebo::rendering::GzTerrainMatGen::SM2Profile, 751
updateParams
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 720
UpdateParamsForCompositeMap
gazebo::rendering::GzTerrainMatGen::SM2Profile, 751
updatePeriod
gazebo::sensors::Sensor, 707
UpdatePhysics
gazebo::physics::PhysicsEngine, 580
UpdatePoint
gazebo::math::RotationSpline, 671
gazebo::math::Spline, 757
UpdatePublications
gazebo::transport::TopicManager, 821
UpdateRays
gazebo::physics::MultiRayShape, 534
UpdateShaders
gazebo::rendering::RTShaderSystem, 676
UpdateStateSDF
gazebo::physics::World, 921
UpdateSurface
gazebo::physics::Link, 437
updateVpParams
gazebo::rendering::GzTerrainMatGen::SM2Profile::-ShaderHelperGLSL, 720
upperLimit
gazebo::physics::Joint, 399
urdf2gazebo, 108
urdf2gazebo::GazeboExtension, 323
urdf2gazebo::URDF2Gazebo, 828
∼URDF2Gazebo, 828
InitModelDoc, 828
InitModelFile, 829
InitModelString, 829
URDF2Gazebo, 828
UrdfCollisionPtr
Gazebo_parser, 69
UrdfLinkPtr
Gazebo_parser, 69
UrdfVisualPtr
Gazebo_parser, 69
useCFMDamping
gazebo::physics::Joint, 399
UserCamera
gazebo::rendering::UserCamera, 832
UserCamera.hh, 1116
UserCameraPtr
gazebo::rendering, 99
UtilTypes.hh, 1116
Utility, 80
DIAG_TIMER_LAP, 80
DIAG_TIMER_START, 80
DIAG_TIMER_STOP, 80

V_ABOVE
gazebo::rendering::MovableText, 517
V_BELOW
gazebo::rendering::MovableText, 517
VEL
gazebo::physics::Joint, 386
VERTEX
gazebo::rendering::RenderEngine, 655
VISUAL
gazebo::physics::Base, 141
VISUAL_PLUGIN
Common, 32
Valid
gazebo::common::Image, 365
ValidateIP
gazebo::transport::Connection, 229
value
gazebo::common::NumericKeyFrame, 555
sdf::ParamT, 568
variance

Math, 52
Vector2d
gazebo::math::Vector2d, 840
Vector2d.hh, 1117
Vector2i
gazebo::math::Vector2i, 848
Vector2i.hh, 1118
Vector3
gazebo::math::Vector3, 858
Vector3.hh, 1119
Vector4
gazebo::math::Vector4, 871
Vector4.hh, 1120
velocityLimit
gazebo::physics::Joint, 399
version
sdf::SDF, 696
VertAlign
gazebo::rendering::MovableText, 517
vertElem
gazebo::physics::MultiRayShape, 535
gazebo::sensors::GpuRaySensor, 340
vertHalfAngle
gazebo::sensors::GpuRaySensor, 340
vertRangeCount
gazebo::sensors::GpuRaySensor, 340
vertRayCount
gazebo::sensors::GpuRaySensor, 340
vertSize
gazebo::physics::HeightmapShape, 356
vertexBufferCapacity
gazebo::rendering::DynamicRenderable, 273
vertexIndex
gazebo::common::NodeAssignment, 546
vfov
gazebo::sensors::GpuRaySensor, 340
Video
gazebo::common::Video, 879
Video.hh, 1122
VideoVisual
gazebo::rendering::VideoVisual, 881
VideoVisual.hh, 1123
ViewController
gazebo::rendering::ViewController, 882
ViewController.hh, 1124
viewport
gazebo::rendering::Camera, 188
visPub
gazebo::physics::Entity, 292
visitRenderables
gazebo::rendering::MovableText, 521
Visual
gazebo::rendering::Visual, 890
Visual.hh, 1125
VisualFromSDF
Messages, 64
visualMsg
gazebo::physics::Entity, 292
visualName
gazebo::physics::Actor, 118
VisualPlugin
gazebo::VisualPlugin, 905
VisualPluginPtr
gazebo, 83
VisualPtr
gazebo::rendering, 99
visuals
gazebo::physics::Link, 438

w
gazebo::math::Quaternion, 636
gazebo::math::Vector4, 878
WORLD_PLUGIN
Common, 32
WaitForConnection
gazebo::transport::Publisher, 621
wallTime
gazebo::physics::State, 762
weight
gazebo::common::NodeAssignment, 546
White
gazebo::common::Color, 219
windowId
gazebo::rendering::Camera, 188
WindowManager.hh, 1126
WireBox
gazebo::rendering::WireBox, 909
WireBox.hh, 1126
World
gazebo::physics::World, 913
world
gazebo::physics::Base, 148
gazebo::physics::Contact, 239
gazebo::physics::PhysicsEngine, 581
gazebo::sensors::Sensor, 707
World.hh, 1127
worldCreated
gazebo::event::Events, 309
worldName
gazebo::common::UpdateInfo, 828
WorldPlugin
gazebo::WorldPlugin, 922
WorldPluginPtr
gazebo, 83
WorldPtr
gazebo::physics, 96
WorldState
gazebo::physics::WorldState, 924, 925

WorldState.hh, 1129
worldUpdateBegin
gazebo::event::Events, 309
worldUpdateEnd
gazebo::event::Events, 309
worldUpdateStart
gazebo::event::Events, 309
wrench
gazebo::physics::Contact, 239
Write
sdf::SDF, 696

x
gazebo::math::Quaternion, 636
gazebo::math::Vector2d, 846
gazebo::math::Vector2i, 855
gazebo::math::Vector3, 868
gazebo::math::Vector4, 878
X_POSITION
BVHLoader.hh, 943
X_ROTATION
BVHLoader.hh, 943

y
gazebo::math::Quaternion, 636
gazebo::math::Vector2d, 846
gazebo::math::Vector2i, 855
gazebo::math::Vector3, 868
gazebo::math::Vector4, 878
Y_POSITION
BVHLoader.hh, 943
Y_ROTATION
BVHLoader.hh, 943
Yellow
gazebo::common::Color, 219

z
gazebo::math::Quaternion, 636
gazebo::math::Vector3, 869
gazebo::math::Vector4, 878
Z_POSITION
BVHLoader.hh, 943
Z_ROTATION
BVHLoader.hh, 943
ZERO
gazebo::math::Matrix4, 474
Zero
gazebo::common::Time, 812
gazebo::math::Pose, 605
gazebo::math::Vector3, 869