Gazebo

4.0.0

Generated by Doxygen 1.7.6.1

Tue Jul 8 2014 02:47:53

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 Class Index **9**

5.1 Class Hierarchy 9

6 Class Index **17**

6.1 Class List 17

7 File Index **31**

7.1 File List 31

8 Module Documentation **39**

8.1 Common 39

8.1.1 Detailed Description 45

8.1.2 Define Documentation 45

8.1.2.1 gzdbg 45

8.1.2.2 gzerr 45

8.1.2.3 gzlog . . . 45
8.1.2.4 gzLogInit . . . 45
8.1.2.5 gzmsg . . . 46
8.1.2.6 gzthrow . . . 46
8.1.2.7 gzwarn . . . 46
8.1.3 Enumeration Type Documentation . . . 46
8.1.3.1 PluginType . . . 46
8.1.4 Function Documentation . . . 46
8.1.4.1 MovingWindowFilter . . . 47
8.1.4.2 MovingWindowFilterPrivate . . . 47
8.1.4.3 ∼MovingWindowFilter . . . 47
8.1.4.4 add_search_path_suffix . . . 47
8.1.4.5 DownloadDependencies . . . 47
8.1.4.6 find_file . . . 47
8.1.4.7 find_file . . . 48
8.1.4.8 find_file_path . . . 48
8.1.4.9 Fini . . . 48
8.1.4.10 Get . . . 49
8.1.4.11 get_sha1 . . . 49
8.1.4.12 GetDBConfig . . . 49
8.1.4.13 GetModelConfig . . . 49
8.1.4.14 GetModelFile . . . 50
8.1.4.15 GetModelName . . . 50
8.1.4.16 GetModelPath . . . 50
8.1.4.17 GetModels . . . 51
8.1.4.18 GetModels . . . 51
8.1.4.19 GetURI . . . 51
8.1.4.20 GetWindowFilled . . . 51
8.1.4.21 GetWindowSize . . . 52
8.1.4.22 HasModel . . . 52
8.1.4.23 load . . . 52

8.1.4.24 SetWindowSize . . . 52
8.1.4.25 Start . . . 53
8.1.4.26 Update . . . 53
8.1.5 Variable Documentation . . . 53
8.1.5.1 PixelFormatNames . . . 53
8.2 Events . . . 55
8.2.1 Function Documentation . . . 56
8.2.1.1 EventT . . . 56
8.2.1.2 ∼EventT . . . 56
8.2.1.3 Connect . . . 56
8.2.1.4 ConnectionCount . . . 56
8.2.1.5 Disconnect . . . 57
8.2.1.6 Disconnect . . . 57
8.3 Math . . . 58
8.3.1 Detailed Description . . . 60
8.3.2 Function Documentation . . . 60
8.3.2.1 clamp . . . 60
8.3.2.2 equal . . . 61
8.3.2.3 fixnan . . . 61
8.3.2.4 fixnan . . . 61
8.3.2.5 isnan . . . 62
8.3.2.6 isnan . . . 62
8.3.2.7 isPowerOfTwo . . . 62
8.3.2.8 max . . . 63
8.3.2.9 mean . . . 63
8.3.2.10 min . . . 63
8.3.2.11 parseFloat . . . 64
8.3.2.12 parseInt . . . 64
8.3.2.13 precision . . . 64
8.3.2.14 roundUpPowerOfTwo . . . 65
8.3.2.15 variance . . . 65

8.3.3 Variable Documentation . . . 65
8.3.3.1 NAN_D . . . 65
8.3.3.2 NAN_I . . . 66
8.4 Messages . . . 67
8.4.1 Detailed Description . . . 70
8.4.2 Define Documentation . . . 70
8.4.2.1 GZ_REGISTER_STATIC_MSG . . . 70
8.4.3 Function Documentation . . . 70
8.4.3.1 Convert . . . 71
8.4.3.2 Convert . . . 71
8.4.3.3 Convert . . . 71
8.4.3.4 Convert . . . 72
8.4.3.5 Convert . . . 72
8.4.3.6 Convert . . . 72
8.4.3.7 Convert . . . 73
8.4.3.8 Convert . . . 73
8.4.3.9 Convert . . . 73
8.4.3.10 Convert . . . 74
8.4.3.11 Convert . . . 74
8.4.3.12 Convert . . . 74
8.4.3.13 CreateRequest . . . 75
8.4.3.14 FogFromSDF . . . 75
8.4.3.15 GeometryFromSDF . . . 75
8.4.3.16 GetHeader . . . 76
8.4.3.17 GUIFromSDF . . . 76
8.4.3.18 Init . . . 76
8.4.3.19 LightFromSDF . . . 77
8.4.3.20 LightToSDF . . . 77
8.4.3.21 MeshFromSDF . . . 77
8.4.3.22 SceneFromSDF . . . 78
8.4.3.23 Set . . . 78

8.4.3.24 Set . . . 78
8.4.3.25 Set . . . 78
8.4.3.26 Set . . . 79
8.4.3.27 Set . . . 79
8.4.3.28 Set . . . 79
8.4.3.29 Set . . . 79
8.4.3.30 Set . . . 80
8.4.3.31 Set . . . 80
8.4.3.32 Set . . . 80
8.4.3.33 Stamp . . . 80
8.4.3.34 Stamp . . . 81
8.4.3.35 TrackVisualFromSDF . . . 81
8.4.3.36 VisualFromSDF . . . 81
8.5 Classes for physics and dynamics . . . 82
8.5.1 Define Documentation . . . 86
8.5.1.1 GZ_REGISTER_PHYSICS_ENGINE . . . 86
8.5.2 Typedef Documentation . . . 87
8.5.2.1 PhysicsFactoryFn . . . 87
8.5.3 Function Documentation . . . 87
8.5.3.1 create_world . . . 87
8.5.3.2 fini . . . 87
8.5.3.3 get_world . . . 87
8.5.3.4 getUniqueId . . . 88
8.5.3.5 init_world . . . 88
8.5.3.6 init_worlds . . . 88
8.5.3.7 load . . . 88
8.5.3.8 load_world . . . 88
8.5.3.9 load_worlds . . . 89
8.5.3.10 pause_world . . . 89
8.5.3.11 pause_worlds . . . 89
8.5.3.12 remove_worlds . . . 89

8.5.3.13 run_world . . . 89
8.5.3.14 run_worlds . . . 90
8.5.3.15 stop_world . . . 90
8.5.3.16 stop_worlds . . . 90
8.5.3.17 worlds_running . . . 90
8.5.4 Variable Documentation . . . 90
8.5.4.1 EntityTypename . . . 90
8.6 DART Physics . . . 92
8.6.1 Detailed Description . . . 93
8.6.2 Function Documentation . . . 93
8.6.2.1 DARTRayShape . . . 93
8.6.2.2 DARTRayShape . . . 94
8.6.2.3 ∼DARTRayShape . . . 94
8.6.2.4 ConvPose . . . 94
8.6.2.5 ConvPose . . . 94
8.6.2.6 ConvQuat . . . 94
8.6.2.7 ConvQuat . . . 94
8.6.2.8 ConvVec3 . . . 94
8.6.2.9 ConvVec3 . . . 95
8.6.2.10 GetIntersection . . . 95
8.6.2.11 SetPoints . . . 95
8.6.2.12 Update . . . 95
8.7 Bullet Physics . . . 96
8.7.1 Function Documentation . . . 96
8.7.1.1 DARTMultiRayShape . . . 96
8.7.1.2 ∼DARTMultiRayShape . . . 97
8.7.1.3 AddRay . . . 97
8.7.1.4 UpdateRays . . . 97
8.8 Simbody Physics . . . 98
8.8.1 Detailed Description . . . 99
8.9 Rendering . . . 100

8.9.1 Detailed Description . . . 102
8.9.2 Function Documentation . . . 102
8.9.2.1 create_scene . . . 102
8.9.2.2 fini . . . 103
8.9.2.3 get_scene . . . 103
8.9.2.4 init . . . 103
8.9.2.5 load . . . 103
8.9.2.6 remove_scene . . . 103
8.10 Sensors . . . 104
8.10.1 Detailed Description . . . 106
8.10.2 Define Documentation . . . 106
8.10.2.1 GZ_REGISTER_STATIC_SENSOR . . . 106
8.10.3 Function Documentation . . . 107
8.10.3.1 create_sensor . . . 107
8.10.3.2 disable . . . 107
8.10.3.3 enable . . . 107
8.10.3.4 fini . . . 107
8.10.3.5 get_sensor . . . 107
8.10.3.6 init . . . 108
8.10.3.7 load . . . 108
8.10.3.8 remove_sensor . . . 108
8.10.3.9 remove_sensors . . . 108
8.10.3.10 run_once . . . 109
8.10.3.11 run_threads . . . 109
8.10.3.12 stop . . . 109
8.11 Transport . . . 110
8.11.1 Detailed Description . . . 112
8.11.2 Typedef Documentation . . . 113
8.11.2.1 CallbackHelperPtr . . . 113
8.11.3 Function Documentation . . . 113
8.11.3.1 clear_buffers . . . 113

8.11.3.2 connectToMaster . . . 113
8.11.3.3 fini . . . 113
8.11.3.4 get_master_uri . . . 113
8.11.3.5 get_topic_namespaces . . . 114
8.11.3.6 getAdvertisedTopics . . . 114
8.11.3.7 getAdvertisedTopics . . . 114
8.11.3.8 getMinimalComms . . . 115
8.11.3.9 getTopicMsgType . . . 115
8.11.3.10 init . . . 115
8.11.3.11 is_stopped . . . 115
8.11.3.12 pause_incoming . . . 116
8.11.3.13 publish . . . 116
8.11.3.14 request . . . 116
8.11.3.15 requestNoReply . . . 117
8.11.3.16 requestNoReply . . . 117
8.11.3.17 run . . . 117
8.11.3.18 setMinimalComms . . . 117
8.11.3.19 stop . . . 118
8.11.3.20 waitForNamespaces . . . 118
8.12 Utility . . . 119
8.12.1 Define Documentation . . . 119
8.12.1.1 DIAG_TIMER_LAP . . . 119
8.12.1.2 DIAG_TIMER_START . . . 119
8.12.1.3 DIAG_TIMER_STOP . . . 119

9 Namespace Documentation 121

9.1 boost Namespace Reference . . . 121
9.2 gazebo Namespace Reference . . . 121
9.2.1 Detailed Description . . . 123
9.2.2 Typedef Documentation . . . 123
9.2.2.1 GUIPluginPtr . . . 123

9.2.2.2 ModelPluginPtr . . . 123
9.2.2.3 SensorPluginPtr . . . 123
9.2.2.4 SystemPluginPtr . . . 123
9.2.2.5 VisualPluginPtr . . . 123
9.2.2.6 WorldPluginPtr . . . 123
9.2.3 Function Documentation . . . 123
9.2.3.1 addPlugin . . . 123
9.2.3.2 loadWorld . . . 124
9.2.3.3 printVersion . . . 124
9.2.3.4 runWorld . . . 124
9.2.3.5 setupClient . . . 124
9.2.3.6 setupClient . . . 125
9.2.3.7 setupServer . . . 125
9.2.3.8 setupServer . . . 126
9.2.3.9 shutdown . . . 126
9.3 gazebo::common Namespace Reference . . . 126
9.3.1 Detailed Description . . . 130
9.3.2 Typedef Documentation . . . 130
9.3.2.1 AnimationPtr . . . 130
9.3.2.2 DiagnosticTimerPtr . . . 130
9.3.2.3 NodeMap . . . 130
9.3.2.4 NodeMapIter . . . 130
9.3.2.5 NumericAnimationPtr . . . 131
9.3.2.6 Param_V . . . 131
9.3.2.7 PoseAnimationPtr . . . 131
9.3.2.8 RawNodeAnim . . . 131
9.3.2.9 RawNodeWeights . . . 131
9.3.2.10 RawSkeletonAnim . . . 131
9.3.2.11 SphericalCoordinatesPtr . . . 131
9.3.2.12 StrStr_M . . . 131
9.3.3 Variable Documentation . . . 131

9.3.3.1 SpeedOfLight . . . 131
9.4 gazebo::event Namespace Reference . . . 131
9.4.1 Detailed Description . . . 132
9.4.2 Typedef Documentation . . . 132
9.4.2.1 Connection_V . . . 132
9.4.2.2 ConnectionPtr . . . 132
9.5 gazebo::math Namespace Reference . . . 132
9.5.1 Detailed Description . . . 135
9.5.2 Typedef Documentation . . . 135
9.5.2.1 GeneratorType . . . 135
9.5.2.2 NormalRealDist . . . 135
9.5.2.3 NRealGen . . . 135
9.5.2.4 UIntGen . . . 135
9.5.2.5 UniformIntDist . . . 135
9.5.2.6 UniformRealDist . . . 135
9.5.2.7 URealGen . . . 135
9.6 gazebo::msgs Namespace Reference . . . 135
9.6.1 Detailed Description . . . 138
9.6.2 Typedef Documentation . . . 138
9.6.2.1 MsgFactoryFn . . . 138
9.7 gazebo::physics Namespace Reference . . . 138
9.7.1 Detailed Description . . . 145
9.7.2 Typedef Documentation . . . 145
9.7.2.1 Actor_V . . . 145
9.7.2.2 ActorPtr . . . 145
9.7.2.3 Base_V . . . 145
9.7.2.4 BasePtr . . . 145
9.7.2.5 BoxShapePtr . . . 145
9.7.2.6 Collision_V . . . 145
9.7.2.7 CollisionPtr . . . 145
9.7.2.8 ContactPtr . . . 145

9.7.2.9 CylinderShapePtr . . . 145
9.7.2.10 DARTCollisionPtr . . . 145
9.7.2.11 DARTJointPtr . . . 145
9.7.2.12 DARTLinkPtr . . . 145
9.7.2.13 DARTModelPtr . . . 145
9.7.2.14 DARTPhysicsPtr . . . 145
9.7.2.15 DARTRayShapePtr . . . 146
9.7.2.16 EntityPtr . . . 146
9.7.2.17 GripperPtr . . . 146
9.7.2.18 HeightmapShapePtr . . . 146
9.7.2.19 InertialPtr . . . 146
9.7.2.20 Joint_V . . . 146
9.7.2.21 JointController_V . . . 146
9.7.2.22 JointControllerPtr . . . 146
9.7.2.23 JointPtr . . . 146
9.7.2.24 JointState_M . . . 146
9.7.2.25 Link_V . . . 146
9.7.2.26 LinkPtr . . . 146
9.7.2.27 LinkState_M . . . 146
9.7.2.28 MeshShapePtr . . . 146
9.7.2.29 Model_V . . . 146
9.7.2.30 ModelPtr . . . 146
9.7.2.31 ModelState_M . . . 146
9.7.2.32 MultiRayShapePtr . . . 146
9.7.2.33 PhysicsEnginePtr . . . 146
9.7.2.34 RayShapePtr . . . 147
9.7.2.35 RoadPtr . . . 147
9.7.2.36 ShapePtr . . . 147
9.7.2.37 SimbodyCollisionPtr . . . 147
9.7.2.38 SimbodyLinkPtr . . . 147
9.7.2.39 SimbodyModelPtr . . . 147

9.7.2.40 SimbodyPhysicsPtr . . . 147
9.7.2.41 SimbodyRayShapePtr . . . 147
9.7.2.42 SphereShapePtr . . . 147
9.7.2.43 SurfaceParamsPtr . . . 147
9.7.2.44 WorldPtr . . . 147
9.8 gazebo::rendering Namespace Reference . . . 147
9.8.1 Detailed Description . . . 152
9.8.2 Typedef Documentation . . . 152
9.8.2.1 ArrowVisualPtr . . . 152
9.8.2.2 AxisVisualPtr . . . 152
9.8.2.3 CameraPtr . . . 152
9.8.2.4 CameraVisualPtr . . . 152
9.8.2.5 COMVisualPtr . . . 152
9.8.2.6 ContactVisualPtr . . . 152
9.8.2.7 DepthCameraPtr . . . 152
9.8.2.8 DynamicLinesPtr . . . 152
9.8.2.9 GpuLaserPtr . . . 152
9.8.2.10 JointVisualPtr . . . 152
9.8.2.11 LaserVisualPtr . . . 152
9.8.2.12 LightPtr . . . 152
9.8.2.13 RFIDTagVisualPtr . . . 152
9.8.2.14 RFIDVisualPtr . . . 152
9.8.2.15 ScenePtr . . . 152
9.8.2.16 SelectionObjPtr . . . 152
9.8.2.17 SonarVisualPtr . . . 153
9.8.2.18 UserCameraPtr . . . 153
9.8.2.19 VisualPtr . . . 153
9.8.2.20 WindowManagerPtr . . . 153
9.8.2.21 WrenchVisualPtr . . . 153
9.8.3 Enumeration Type Documentation . . . 153
9.8.3.1 RenderOpType . . . 153

9.9 gazebo::sensors Namespace Reference . . . 153
9.9.1 Detailed Description . . . 156
9.9.2 Typedef Documentation . . . 156
9.9.2.1 CameraSensor_V . . . 156
9.9.2.2 CameraSensorPtr . . . 156
9.9.2.3 ContactSensor_V . . . 156
9.9.2.4 ContactSensorPtr . . . 157
9.9.2.5 DepthCameraSensor_V . . . 157
9.9.2.6 DepthCameraSensorPtr . . . 157
9.9.2.7 ForceTorqueSensorPtr . . . 157
9.9.2.8 GaussianNoiseModelPtr . . . 157
9.9.2.9 GpsSensorPtr . . . 157
9.9.2.10 GpuRaySensor_V . . . 157
9.9.2.11 GpuRaySensorPtr . . . 157
9.9.2.12 ImageGaussianNoiseModelPtr . . . 157
9.9.2.13 ImuSensor_V . . . 157
9.9.2.14 ImuSensorPtr . . . 157
9.9.2.15 MultiCameraSensor_V . . . 157
9.9.2.16 MultiCameraSensorPtr . . . 157
9.9.2.17 NoisePtr . . . 157
9.9.2.18 RaySensor_V . . . 157
9.9.2.19 RaySensorPtr . . . 157
9.9.2.20 RFIDSensor_V . . . 158
9.9.2.21 RFIDSensorPtr . . . 158
9.9.2.22 RFIDTag_V . . . 158
9.9.2.23 RFIDTagPtr . . . 158
9.9.2.24 Sensor_V . . . 158
9.9.2.25 SensorFactoryFn . . . 158
9.9.2.26 SensorPtr . . . 158
9.9.2.27 SonarSensorPtr . . . 158
9.9.2.28 WirelessReceiver_V . . . 158

9.9.2.29 WirelessReceiverPtr . . . 158
9.9.2.30 WirelessTransceiver_V . . . 158
9.9.2.31 WirelessTransceiverPtr . . . 158
9.9.2.32 WirelessTransmitter_V . . . 158
9.9.2.33 WirelessTransmitterPtr . . . 158
9.9.3 Enumeration Type Documentation . . . 158
9.9.3.1 SensorCategory . . . 158
9.10 gazebo::transport Namespace Reference . . . 159
9.10.1 Typedef Documentation . . . 161
9.10.1.1 ConnectionPtr . . . 161
9.10.1.2 MessagePtr . . . 161
9.10.1.3 NodePtr . . . 162
9.10.1.4 PublicationPtr . . . 162
9.10.1.5 PublicationTransportPtr . . . 162
9.10.1.6 PublisherPtr . . . 162
9.10.1.7 SubscriberPtr . . . 162
9.10.1.8 SubscriptionTransportPtr . . . 162
9.11 gazebo::util Namespace Reference . . . 162
9.11.1 Typedef Documentation . . . 162
9.11.1.1 DiagnosticTimerPtr . . . 162
9.11.1.2 OpenALSinkPtr . . . 162
9.11.1.3 OpenALSourcePtr . . . 162
9.12 google Namespace Reference . . . 163
9.13 google::protobuf Namespace Reference . . . 163
9.14 google::protobuf::compiler Namespace Reference . . . 163
9.15 google::protobuf::compiler::cpp Namespace Reference . . . 163
9.16 Ogre Namespace Reference . . . 163
9.17 ogre Namespace Reference . . . 163
9.18 OVR Namespace Reference . . . 163
9.19 OVR::Util Namespace Reference . . . 164
9.20 OVR::Util::Render Namespace Reference . . . 164

9.21 SimTK Namespace Reference 164
9.22 SkyX Namespace Reference 164

10 Class Documentation **165**

10.1 gazebo::physics::Actor Class Reference 165
10.1.1 Detailed Description 168
10.1.2 Constructor & Destructor Documentation 168
10.1.2.1 Actor 168
10.1.2.2 ∼Actor 169
10.1.3 Member Function Documentation 169
10.1.3.1 Fini 169
10.1.3.2 GetSDF 169
10.1.3.3 Init 169
10.1.3.4 IsActive 169
10.1.3.5 Load 169
10.1.3.6 Play 170
10.1.3.7 Stop 170
10.1.3.8 Update 170
10.1.3.9 UpdateParameters 170
10.1.4 Member Data Documentation 170
10.1.4.1 active 170
10.1.4.2 autoStart 170
10.1.4.3 bonePosePub 170
10.1.4.4 interpolateX 171
10.1.4.5 lastPos 171
10.1.4.6 lastScriptTime 171
10.1.4.7 lastTraj 171
10.1.4.8 loop 171
10.1.4.9 mainLink 171
10.1.4.10 mesh 171
10.1.4.11 oldAction 171

10.1.4.12 pathLength . . . 171
10.1.4.13 playStartTime . . . 172
10.1.4.14 prevFrameTime . . . 172
10.1.4.15 scriptLength . . . 172
10.1.4.16 skelAnimation . . . 172
10.1.4.17 skeleton . . . 172
10.1.4.18 skelNodesMap . . . 172
10.1.4.19 skinFile . . . 172
10.1.4.20 skinScale . . . 172
10.1.4.21 startDelay . . . 172
10.1.4.22 trajectories . . . 173
10.1.4.23 trajInfo . . . 173
10.1.4.24 visualId . . . 173
10.1.4.25 visualName . . . 173
10.2 gazebo::math::Angle Class Reference . . . 173
10.2.1 Detailed Description . . . 175
10.2.2 Constructor & Destructor Documentation . . . 175
10.2.2.1 Angle . . . 175
10.2.2.2 Angle . . . 175
10.2.2.3 Angle . . . 176
10.2.2.4 ∼Angle . . . 176
10.2.3 Member Function Documentation . . . 176
10.2.3.1 Degree . . . 176
10.2.3.2 Normalize . . . 176
10.2.3.3 operator!= . . . 176
10.2.3.4 operator∗ . . . 176
10.2.3.5 operator∗ . . . 177
10.2.3.6 operator∗= . . . 177
10.2.3.7 operator+ . . . 177
10.2.3.8 operator+= . . . 178
10.2.3.9 operator- . . . 178

10.2.3.10 operator-= . . . 178
10.2.3.11 operator/ . . . 178
10.2.3.12 operator/= . . . 179
10.2.3.13 operator< . . . 179
10.2.3.14 operator<= . . . 179
10.2.3.15 operator== . . . 180
10.2.3.16 operator> . . . 180
10.2.3.17 operator>= . . . 180
10.2.3.18 Radian . . . 181
10.2.3.19 SetFromDegree . . . 181
10.2.3.20 SetFromRadian . . . 181
10.2.4 Friends And Related Function Documentation . . . 181
10.2.4.1 operator<< . . . 181
10.2.4.2 operator>> . . . 182
10.2.5 Member Data Documentation . . . 182
10.2.5.1 HalfPi . . . 182
10.2.5.2 Pi . . . 182
10.2.5.3 TwoPi . . . 182
10.2.5.4 Zero . . . 182
10.3 gazebo::common::Animation Class Reference . . . 182
10.3.1 Detailed Description . . . 184
10.3.2 Member Typedef Documentation . . . 184
10.3.2.1 KeyFrame_V . . . 184
10.3.3 Constructor & Destructor Documentation . . . 184
10.3.3.1 Animation . . . 184
10.3.3.2 ~Animation . . . 185
10.3.4 Member Function Documentation . . . 185
10.3.4.1 AddTime . . . 185
10.3.4.2 GetKeyFrame . . . 185
10.3.4.3 GetKeyFrameCount . . . 185
10.3.4.4 GetKeyFramesAtTime . . . 186

10.3.4.5 GetLength . . . 186
10.3.4.6 GetTime . . . 186
10.3.4.7 SetLength . . . 186
10.3.4.8 SetTime . . . 187
10.3.5 Member Data Documentation . . . 187
10.3.5.1 build . . . 187
10.3.5.2 keyFrames . . . 187
10.3.5.3 length . . . 187
10.3.5.4 loop . . . 187
10.3.5.5 name . . . 187
10.3.5.6 timePos . . . 187
10.4 gazebo::rendering::ArrowVisual Class Reference . . . 188
10.4.1 Detailed Description . . . 188
10.4.2 Constructor & Destructor Documentation . . . 188
10.4.2.1 ArrowVisual . . . 189
10.4.2.2 ∼ArrowVisual . . . 189
10.4.3 Member Function Documentation . . . 189
10.4.3.1 Load . . . 189
10.4.3.2 ShowRotation . . . 189
10.5 gazebo::rendering::ArrowVisualPrivate Class Reference . . . 189
10.5.1 Detailed Description . . . 190
10.5.2 Member Data Documentation . . . 190
10.5.2.1 headNode . . . 190
10.5.2.2 rotationNode . . . 190
10.5.2.3 shaftNode . . . 190
10.6 gazebo::common::AssertionInternalError Class Reference . . . 191
10.6.1 Detailed Description . . . 191
10.6.2 Constructor & Destructor Documentation . . . 192
10.6.2.1 AssertionInternalError . . . 192
10.6.2.2 ∼AssertionInternalError . . . 192
10.7 gazebo::common::AudioDecoder Class Reference . . . 192

10.7.1 Detailed Description . . . 193
10.7.2 Constructor & Destructor Documentation . . . 193
10.7.2.1 AudioDecoder . . . 193
10.7.2.2 ∼AudioDecoder . . . 193
10.7.3 Member Function Documentation . . . 193
10.7.3.1 Decode . . . 193
10.7.3.2 GetFile . . . 194
10.7.3.3 GetSampleRate . . . 194
10.7.3.4 SetFile . . . 194
10.8 gazebo::rendering::AxisVisual Class Reference . . . 194
10.8.1 Detailed Description . . . 195
10.8.2 Constructor & Destructor Documentation . . . 196
10.8.2.1 AxisVisual . . . 196
10.8.2.2 ∼AxisVisual . . . 196
10.8.3 Member Function Documentation . . . 196
10.8.3.1 Load . . . 196
10.8.3.2 ScaleXAxis . . . 196
10.8.3.3 ScaleYAxis . . . 196
10.8.3.4 ScaleZAxis . . . 197
10.8.3.5 SetAxisMaterial . . . 197
10.8.3.6 ShowRotation . . . 197
10.9 gazebo::rendering::AxisVisualPrivate Class Reference . . . 197
10.9.1 Detailed Description . . . 198
10.9.2 Member Data Documentation . . . 198
10.9.2.1 xAxis . . . 198
10.9.2.2 yAxis . . . 198
10.9.2.3 zAxis . . . 199
10.10 gazebo::physics::BallJoint< T > Class Template Reference . . . 199
10.10.1 Detailed Description . . . 199
10.10.2 Constructor & Destructor Documentation . . . 199
10.10.2.1 BallJoint . . . 200

10.10.2.2 ~BallJoint 200
10.10.3 Member Function Documentation 200
10.10.3.1 GetAngleCount 200
10.10.3.2 Init 200
10.10.3.3 Load 200
10.11 gazebo::physics::Base Class Reference 201
10.11.1 Detailed Description 205
10.11.2 Member Enumeration Documentation 205
10.11.2.1 EntityType 205
10.11.3 Constructor & Destructor Documentation 206
10.11.3.1 Base 206
10.11.3.2 ~Base 207
10.11.4 Member Function Documentation 207
10.11.4.1 AddChild 207
10.11.4.2 AddType 207
10.11.4.3 ComputeScopedName 207
10.11.4.4 Fini 207
10.11.4.5 GetById 208
10.11.4.6 GetByName 208
10.11.4.7 GetChild 208
10.11.4.8 GetChild 209
10.11.4.9 GetChildCount 209
10.11.4.10 GetId 209
10.11.4.11 GetName 209
10.11.4.12 GetParent 209
10.11.4.13 GetParentId 210
10.11.4.14 GetSaveable 210
10.11.4.15 GetScopedName 210
10.11.4.16 GetSDF 210
10.11.4.17 GetType 211
10.11.4.18 GetWorld 211

10.11.4.19HasType . . . 211
10.11.4.20Init . . . 211
10.11.4.21IsSelected . . . 212
10.11.4.22Load . . . 212
10.11.4.23operator== . . . 213
10.11.4.24Print . . . 213
10.11.4.25RemoveChild . . . 214
10.11.4.26RemoveChild . . . 214
10.11.4.27RemoveChildren . . . 214
10.11.4.28Reset . . . 214
10.11.4.29Reset . . . 214
10.11.4.30SetName . . . 215
10.11.4.31SetParent . . . 215
10.11.4.32SetSaveable . . . 215
10.11.4.33SetSelected . . . 215
10.11.4.34SetWorld . . . 215
10.11.4.35Update . . . 216
10.11.4.36UpdateParameters . . . 216
10.11.5 Member Data Documentation . . . 216
10.11.5.1 children . . . 216
10.11.5.2 parent . . . 216
10.11.5.3 sdf . . . 217
10.11.5.4 world . . . 217
10.12gazebo::math::BiQuad< T > Class Template Reference . . . 217
10.12.1 Detailed Description . . . 218
10.12.2 Constructor & Destructor Documentation . . . 218
10.12.2.1 BiQuad . . . 218
10.12.2.2 BiQuad . . . 219
10.12.3 Member Function Documentation . . . 219
10.12.3.1 process . . . 219
10.12.3.2 SetFc . . . 219

10.12.3.3 SetFc . . . 219
10.12.3.4 SetValue . . . 220
10.12.4 Member Data Documentation . . . 220
10.12.4.1 a0 . . . 220
10.12.4.2 a1 . . . 220
10.12.4.3 a2 . . . 220
10.12.4.4 b0 . . . 220
10.12.4.5 b1 . . . 220
10.12.4.6 b2 . . . 220
10.12.4.7 x1 . . . 220
10.12.4.8 x2 . . . 221
10.12.4.9 y1 . . . 221
10.12.4.10 y2 . . . 221
10.13 gazebo::math::BiQuadVector3 Class Reference . . . 221
10.13.1 Detailed Description . . . 222
10.13.2 Constructor & Destructor Documentation . . . 222
10.13.2.1 BiQuadVector3 . . . 222
10.13.2.2 BiQuadVector3 . . . 222
10.14 gazebo::math::Box Class Reference . . . 222
10.14.1 Detailed Description . . . 223
10.14.2 Constructor & Destructor Documentation . . . 224
10.14.2.1 Box . . . 224
10.14.2.2 Box . . . 224
10.14.2.3 Box . . . 224
10.14.2.4 ∼Box . . . 224
10.14.3 Member Function Documentation . . . 224
10.14.3.1 GetCenter . . . 224
10.14.3.2 GetSize . . . 224
10.14.3.3 GetXLength . . . 225
10.14.3.4 GetYLength . . . 225
10.14.3.5 GetZLength . . . 225

10.14.3.6 Merge . . . 225
10.14.3.7 operator+ . . . 225
10.14.3.8 operator+= . . . 226
10.14.3.9 operator- . . . 226
10.14.3.10operator= . . . 226
10.14.3.11operator== . . . 227
10.14.4 Friends And Related Function Documentation . . . 227
10.14.4.1 operator<< . . . 227
10.14.5 Member Data Documentation . . . 227
10.14.5.1 max . . . 227
10.14.5.2 min . . . 227
10.15gazebo::physics::BoxShape Class Reference . . . 228
10.15.1 Detailed Description . . . 229
10.15.2 Constructor & Destructor Documentation . . . 229
10.15.2.1 BoxShape . . . 229
10.15.2.2 ∼BoxShape . . . 229
10.15.3 Member Function Documentation . . . 229
10.15.3.1 FillMsg . . . 229
10.15.3.2 GetSize . . . 230
10.15.3.3 Init . . . 230
10.15.3.4 ProcessMsg . . . 230
10.15.3.5 SetScale . . . 230
10.15.3.6 SetSize . . . 230
10.16gazebo::common::FileLogger::Buffer Class Reference . . . 231
10.16.1 Detailed Description . . . 231
10.16.2 Constructor & Destructor Documentation . . . 231
10.16.2.1 Buffer . . . 232
10.16.2.2 ∼Buffer . . . 232
10.16.3 Member Function Documentation . . . 232
10.16.3.1 sync . . . 232
10.16.4 Member Data Documentation . . . 232

10.16.4.1 stream 232
10.17gazebo::common::Logger::Buffer Class Reference 232
10.17.1 Detailed Description 233
10.17.2 Constructor & Destructor Documentation 233
10.17.2.1 Buffer 233
10.17.2.2 ∼Buffer 233
10.17.3 Member Function Documentation 233
10.17.3.1 sync 233
10.17.4 Member Data Documentation 234
10.17.4.1 color 234
10.17.4.2 type 234
10.18gazebo::common::BVHLoader Class Reference 234
10.18.1 Detailed Description 234
10.18.2 Constructor & Destructor Documentation 235
10.18.2.1 BVHLoader 235
10.18.2.2 ∼BVHLoader 235
10.18.3 Member Function Documentation 235
10.18.3.1 Load 235
10.19gazebo::transport::CallbackHelper Class Reference 235
10.19.1 Detailed Description 237
10.19.2 Constructor & Destructor Documentation 237
10.19.2.1 CallbackHelper 237
10.19.2.2 ∼CallbackHelper 237
10.19.3 Member Function Documentation 237
10.19.3.1 GetId 237
10.19.3.2 GetLatching 237
10.19.3.3 GetMsgType 237
10.19.3.4 HandleData 238
10.19.3.5 HandleMessage 238
10.19.3.6 IsLocal 239
10.19.3.7 SetLatching 239

10.19.4 Member Data Documentation 239
10.19.4.1 latching 239
10.20 gazebo::transport::CallbackHelperT< M > Class Template Reference . 240
10.20.1 Detailed Description 240
10.20.2 Constructor & Destructor Documentation 241
10.20.2.1 CallbackHelperT 241
10.20.3 Member Function Documentation 241
10.20.3.1 GetMsgType 241
10.20.3.2 HandleData 241
10.20.3.3 HandleMessage 242
10.20.3.4 IsLocal 242
10.21 gazebo::rendering::Camera Class Reference 242
10.21.1 Detailed Description 250
10.21.2 Constructor & Destructor Documentation 250
10.21.2.1 Camera 250
10.21.2.2 ∼Camera 250
10.21.3 Member Function Documentation 250
10.21.3.1 AnimationComplete 251
10.21.3.2 AttachToVisual 251
10.21.3.3 AttachToVisual 251
10.21.3.4 AttachToVisualImpl 251
10.21.3.5 AttachToVisualImpl 252
10.21.3.6 AttachToVisualImpl 252
10.21.3.7 ConnectNewImageFrame 253
10.21.3.8 CreateRenderTexture 253
10.21.3.9 DisconnectNewImageFrame 253
10.21.3.10 EnableSaveFrame 254
10.21.3.11 Fini 254
10.21.3.12 GetAspectRatio 254
10.21.3.13 GetAvgFPS 254
10.21.3.14 GetCameraToViewportRay 255

10.21.3.15GetCaptureData 255
10.21.3.16GetDirection 255
10.21.3.17GetFarClip 255
10.21.3.18GetFrameFilename 255
10.21.3.19GetHFOV 256
10.21.3.20GetImageByteSize 256
10.21.3.21GetImageByteSize 256
10.21.3.22GetImageData 256
10.21.3.23GetImageDepth 257
10.21.3.24GetImageFormat 257
10.21.3.25GetImageHeight 257
10.21.3.26GetImageWidth 257
10.21.3.27GetInitialized 258
10.21.3.28GetLastRenderWallTime 258
10.21.3.29GetName 258
10.21.3.30GetNearClip 258
10.21.3.31GetOgreCamera 258
10.21.3.32GetRenderRate 259
10.21.3.33GetRenderTexture 259
10.21.3.34GetRight 259
10.21.3.35GetScene 259
10.21.3.36GetSceneNode 259
10.21.3.37GetScopedName 260
10.21.3.38GetScreenshotPath 260
10.21.3.39GetTextureHeight 260
10.21.3.40GetTextureWidth 260
10.21.3.41GetTriangleCount 260
10.21.3.42GetUp 261
10.21.3.43GetVFOV 261
10.21.3.44GetViewport 261
10.21.3.45GetViewportHeight 261

10.21.3.46GetViewportWidth 261
10.21.3.47GetWindowId 262
10.21.3.48GetWorldPointOnPlane 262
10.21.3.49GetWorldPose 262
10.21.3.50GetWorldPosition 262
10.21.3.51GetWorldRotation 263
10.21.3.52GetZValue 263
10.21.3.53Init 263
10.21.3.54IsAnimating 263
10.21.3.55IsVisible 263
10.21.3.56IsVisible 264
10.21.3.57Load 264
10.21.3.58Load 264
10.21.3.59MoveToPosition 264
10.21.3.60MoveToPositions 265
10.21.3.61PostRender 265
10.21.3.62ReadPixelBuffer 265
10.21.3.63Render 266
10.21.3.64RenderImpl 266
10.21.3.65RotatePitch 266
10.21.3.66RotateYaw 266
10.21.3.67SaveFrame 266
10.21.3.68SaveFrame 267
10.21.3.69SetAspectRatio 267
10.21.3.70SetCaptureData 267
10.21.3.71SetCaptureDataOnce 268
10.21.3.72SetClipDist 268
10.21.3.73SetHFOV 268
10.21.3.74SetImageHeight 268
10.21.3.75SetImageSize 268
10.21.3.76SetImageWidth 269

10.21.3.77 SetName . . . 269
10.21.3.78 SetRenderRate . . . 269
10.21.3.79 SetRenderTarget . . . 269
10.21.3.80 SetSaveFramePathname . . . 269
10.21.3.81 SetScene . . . 270
10.21.3.82 SetSceneNode . . . 270
10.21.3.83 SetWindowId . . . 270
10.21.3.84 SetWorldPose . . . 270
10.21.3.85 SetWorldPosition . . . 270
10.21.3.86 SetWorldRotation . . . 271
10.21.3.87 ShowWireframe . . . 271
10.21.3.88 ToggleShowWireframe . . . 271
10.21.3.89 TrackVisual . . . 271
10.21.3.90 TrackVisualImpl . . . 271
10.21.3.91 TrackVisualImpl . . . 272
10.21.3.92 Translate . . . 272
10.21.3.93 Update . . . 272
10.21.4 Member Data Documentation . . . 272
10.21.4.1 animState . . . 272
10.21.4.2 bayerFrameBuffer . . . 273
10.21.4.3 camera . . . 273
10.21.4.4 captureData . . . 273
10.21.4.5 captureDataOnce . . . 273
10.21.4.6 connections . . . 273
10.21.4.7 imageFormat . . . 273
10.21.4.8 imageHeight . . . 273
10.21.4.9 imageWidth . . . 273
10.21.4.10 initialized . . . 273
10.21.4.11 lastRenderWallTime . . . 274
10.21.4.12 name . . . 274
10.21.4.13 newData . . . 274

10.21.4.14 newImageFrame . . . 274
10.21.4.15 onAnimationComplete . . . 274
10.21.4.16 prevAnimTime . . . 274
10.21.4.17 renderTarget . . . 274
10.21.4.18 renderTexture . . . 274
10.21.4.19 requests . . . 275
10.21.4.20 saveCount . . . 275
10.21.4.21 saveFrameBuffer . . . 275
10.21.4.22 scene . . . 275
10.21.4.23 sceneNode . . . 275
10.21.4.24 scopedName . . . 275
10.21.4.25 scopedUniqueName . . . 275
10.21.4.26 screenshotPath . . . 275
10.21.4.27 sdf . . . 275
10.21.4.28 textureHeight . . . 276
10.21.4.29 textureWidth . . . 276
10.21.4.30 viewport . . . 276
10.21.4.31 windowId . . . 276
10.22 gazebo::rendering::CameraPrivate Class Reference . . . 276
10.22.1 Detailed Description . . . 277
10.22.2 Member Typedef Documentation . . . 278
10.22.2.1 CameraCmdMsgs_L . . . 278
10.22.3 Member Data Documentation . . . 278
10.22.3.1 cameraCounter . . . 278
10.22.3.2 cmdSub . . . 278
10.22.3.3 commandMsgs . . . 278
10.22.3.4 dlGBufferInstance . . . 278
10.22.3.5 dlMergeInstance . . . 278
10.22.3.6 dsGBufferInstance . . . 278
10.22.3.7 dsMergeInstance . . . 278
10.22.3.8 moveToPositionQueue . . . 279

10.22.3.9 node . . . 279
10.22.3.10 receiveMutex . . . 279
10.22.3.11 renderPeriod . . . 279
10.22.3.12 ssaoInstance . . . 279
10.22.3.13 trackedVisual . . . 279
10.22.3.14 trackVisualPID . . . 279
10.22.3.15 trackVisualPitchPID . . . 279
10.22.3.16 trackVisualYawPID . . . 279
10.23 gazebo::sensors::CameraSensor Class Reference . . . 280
10.23.1 Detailed Description . . . 281
10.23.2 Constructor & Destructor Documentation . . . 281
10.23.2.1 CameraSensor . . . 281
10.23.2.2 ∼CameraSensor . . . 281
10.23.3 Member Function Documentation . . . 281
10.23.3.1 Fini . . . 281
10.23.3.2 GetCamera . . . 282
10.23.3.3 GetImageData . . . 282
10.23.3.4 GetImageHeight . . . 282
10.23.3.5 GetImageWidth . . . 282
10.23.3.6 GetTopic . . . 282
10.23.3.7 Init . . . 283
10.23.3.8 IsActive . . . 283
10.23.3.9 Load . . . 283
10.23.3.10 Load . . . 283
10.23.3.11 SaveFrame . . . 284
10.23.3.12 UpdateImpl . . . 284
10.24 gazebo::rendering::CameraVisual Class Reference . . . 284
10.24.1 Detailed Description . . . 285
10.24.2 Constructor & Destructor Documentation . . . 285
10.24.2.1 CameraVisual . . . 285
10.24.2.2 ∼CameraVisual . . . 286

10.24.3 Member Function Documentation 286
10.24.3.1 Load 286
10.25gazebo::rendering::CameraVisualPrivate Class Reference 286
10.25.1 Member Data Documentation 287
10.25.1.1 camera 287
10.25.1.2 connections 287
10.26gazebo::common::ColladaExporter Class Reference 287
10.26.1 Detailed Description 288
10.26.2 Member Enumeration Documentation 288
10.26.2.1 GeometryType 289
10.26.3 Constructor & Destructor Documentation 289
10.26.3.1 ColladaExporter 289
10.26.3.2 ∼ColladaExporter 289
10.26.4 Member Function Documentation 289
10.26.4.1 Export 289
10.27gazebo::common::ColladaExporterPrivate Class Reference 290
10.27.1 Detailed Description 290
10.27.2 Member Data Documentation 290
10.27.2.1 exportTextures 290
10.27.2.2 filename 290
10.27.2.3 materialCount 290
10.27.2.4 mesh 291
10.27.2.5 path 291
10.27.2.6 subMeshCount 291
10.28gazebo::common::ColladaLoader Class Reference 291
10.28.1 Detailed Description 292
10.28.2 Constructor & Destructor Documentation 292
10.28.2.1 ColladaLoader 292
10.28.2.2 ∼ColladaLoader 292
10.28.3 Member Function Documentation 292
10.28.3.1 Load 292

10.29 gazebo::common::ColladaLoaderPrivate Class Reference 293
10.29.1 Detailed Description 293
10.29.2 Member Data Documentation 294
10.29.2.1 colladaXml 294
10.29.2.2 currentNodeName 294
10.29.2.3 filename 294
10.29.2.4 materialIds 294
10.29.2.5 materialMap 294
10.29.2.6 meter 294
10.29.2.7 normalDuplicateMap 294
10.29.2.8 normalIds 294
10.29.2.9 path 295
10.29.2.10 positionDuplicateMap 295
10.29.2.11 positionIds 295
10.29.2.12 texcoordDuplicateMap 295
10.29.2.13 texcoordIds 295
10.30 gazebo::physics::Collision Class Reference 295
10.30.1 Detailed Description 298
10.30.2 Constructor & Destructor Documentation 298
10.30.2.1 Collision 298
10.30.2.2 ∼Collision 299
10.30.3 Member Function Documentation 299
10.30.3.1 FillMsg 299
10.30.3.2 Fini 299
10.30.3.3 GetBoundingBox 299
10.30.3.4 GetLaserRetro 299
10.30.3.5 GetLink 300
10.30.3.6 GetMaxContacts 300
10.30.3.7 GetModel 300
10.30.3.8 GetRelativeAngularAccel 300
10.30.3.9 GetRelativeAngularVel 300

10.30.3.10GetRelativeLinearAccel . . . 301
10.30.3.11GetRelativeLinearVel . . . 301
10.30.3.12GetShape . . . 301
10.30.3.13GetShapeType . . . 301
10.30.3.14GetState . . . 302
10.30.3.15GetSurface . . . 302
10.30.3.16GetWorldAngularAccel . . . 302
10.30.3.17GetWorldAngularVel . . . 302
10.30.3.18GetWorldLinearAccel . . . 303
10.30.3.19GetWorldLinearVel . . . 303
10.30.3.20GetWorldPose . . . 303
10.30.3.21Init . . . 303
10.30.3.22IsPlaceable . . . 303
10.30.3.23Load . . . 304
10.30.3.24ProcessMsg . . . 304
10.30.3.25SetCategoryBits . . . 304
10.30.3.26SetCollideBits . . . 304
10.30.3.27SetCollision . . . 305
10.30.3.28SetLaserRetro . . . 305
10.30.3.29SetMaxContacts . . . 305
10.30.3.30SetScale . . . 305
10.30.3.31SetShape . . . 306
10.30.3.32SetState . . . 306
10.30.3.33SetWorldPoseDirty . . . 306
10.30.3.34UpdateParameters . . . 306
10.30.4 Member Data Documentation . . . 306
10.30.4.1 link . . . 306
10.30.4.2 placeable . . . 307
10.30.4.3 shape . . . 307
10.30.4.4 surface . . . 307
10.31gazebo::physics::CollisionState Class Reference . . . 307

10.31.1 Detailed Description 308
10.31.2 Constructor & Destructor Documentation 308
10.31.2.1 CollisionState 308
10.31.2.2 CollisionState 309
10.31.2.3 CollisionState 309
10.31.2.4 ∼CollisionState 309
10.31.3 Member Function Documentation 309
10.31.3.1 FillSDF 309
10.31.3.2 GetPose 309
10.31.3.3 IsZero 310
10.31.3.4 Load 310
10.31.3.5 operator+ 310
10.31.3.6 operator- 310
10.31.3.7 operator= 311
10.31.4 Friends And Related Function Documentation 311
10.31.4.1 operator<< 311
10.32gazebo::common::Color Class Reference 312
10.32.1 Detailed Description 314
10.32.2 Member Typedef Documentation 315
10.32.2.1 ABGR 315
10.32.2.2 ARGB 315
10.32.2.3 BGRA 315
10.32.2.4 RGBA 315
10.32.3 Constructor & Destructor Documentation 315
10.32.3.1 Color 315
10.32.3.2 Color 315
10.32.3.3 Color 315
10.32.3.4 ∼Color 315
10.32.4 Member Function Documentation 315
10.32.4.1 GetAsABGR 316
10.32.4.2 GetAsARGB 316

10.32.4.3 GetAsBGRA 316
10.32.4.4 GetAsHSV 316
10.32.4.5 GetAsRGBA 316
10.32.4.6 GetAsYUV 317
10.32.4.7 operator!= 317
10.32.4.8 operator∗ 317
10.32.4.9 operator∗ 317
10.32.4.10operator∗= 318
10.32.4.11operator+ 318
10.32.4.12operator+ 318
10.32.4.13operator+= 319
10.32.4.14operator- 319
10.32.4.15operator- 319
10.32.4.16operator-= 319
10.32.4.17operator/ 320
10.32.4.18operator/ 320
10.32.4.19operator/= 320
10.32.4.20operator= 321
10.32.4.21operator== 321
10.32.4.22operator[] 321
10.32.4.23Reset 322
10.32.4.24Set 322
10.32.4.25SetFromABGR 322
10.32.4.26SetFromARGB 322
10.32.4.27SetFromBGRA 322
10.32.4.28SetFromHSV 323
10.32.4.29SetFromRGBA 323
10.32.4.30SetFromYUV 323
10.32.5 Friends And Related Function Documentation 323
10.32.5.1 operator<< 323
10.32.5.2 operator>> 324

10.32.6 Member Data Documentation . . . 324
10.32.6.1 a . . . 324
10.32.6.2 b . . . 324
10.32.6.3 Black . . . 324
10.32.6.4 Blue . . . 324
10.32.6.5 g . . . 324
10.32.6.6 Green . . . 324
10.32.6.7 Purple . . . 324
10.32.6.8 r . . . 324
10.32.6.9 Red . . . 325
10.32.6.10 White . . . 325
10.32.6.11 Yellow . . . 325
10.33 gazebo::rendering::COMVisual Class Reference . . . 325
10.33.1 Detailed Description . . . 326
10.33.2 Constructor & Destructor Documentation . . . 326
10.33.2.1 COMVisual . . . 326
10.33.2.2 ∼COMVisual . . . 326
10.33.3 Member Function Documentation . . . 326
10.33.3.1 Load . . . 326
10.33.3.2 Load . . . 327
10.34 gazebo::rendering::COMVisualPrivate Class Reference . . . 327
10.34.1 Detailed Description . . . 328
10.34.2 Member Data Documentation . . . 328
10.34.2.1 boxNode . . . 328
10.34.2.2 crossLines . . . 328
10.35 gazebo::transport::Connection Class Reference . . . 328
10.35.1 Detailed Description . . . 330
10.35.2 Member Typedef Documentation . . . 330
10.35.2.1 AcceptCallback . . . 330
10.35.2.2 ReadCallback . . . 330
10.35.3 Constructor & Destructor Documentation . . . 330

10.35.3.1 Connection . . . 330
10.35.3.2 ∼Connection . . . 330
10.35.4 Member Function Documentation . . . 331
10.35.4.1 AsyncRead . . . 331
10.35.4.2 Cancel . . . 331
10.35.4.3 Connect . . . 331
10.35.4.4 ConnectToShutdown . . . 331
10.35.4.5 DisconnectShutdown . . . 332
10.35.4.6 EnqueueMsg . . . 332
10.35.4.7 EnqueueMsg . . . 332
10.35.4.8 GetId . . . 332
10.35.4.9 GetIPWhiteList . . . 333
10.35.4.10 GetLocalAddress . . . 333
10.35.4.11 GetLocalHostname . . . 333
10.35.4.12 GetLocalPort . . . 333
10.35.4.13 GetLocalURI . . . 333
10.35.4.14 GetRemoteAddress . . . 334
10.35.4.15 GetRemoteHostname . . . 334
10.35.4.16 GetRemotePort . . . 334
10.35.4.17 GetRemoteURI . . . 334
10.35.4.18 IsOpen . . . 334
10.35.4.19 Listen . . . 335
10.35.4.20 ProcessWriteQueue . . . 335
10.35.4.21 Read . . . 335
10.35.4.22 Shutdown . . . 335
10.35.4.23 StartRead . . . 335
10.35.4.24 StopRead . . . 336
10.35.4.25 ValidateIP . . . 336
10.36 gazebo::event::Connection Class Reference . . . 336
10.36.1 Detailed Description . . . 336
10.36.2 Constructor & Destructor Documentation . . . 337

10.36.2.1 Connection . . . 337
10.36.2.2 Connection . . . 337
10.36.2.3 ∼Connection . . . 337
10.36.3 Member Function Documentation . . . 337
10.36.3.1 GetId . . . 337
10.37 gazebo::transport::ConnectionManager Class Reference . . . 337
10.37.1 Detailed Description . . . 339
10.37.2 Member Function Documentation . . . 339
10.37.2.1 Advertise . . . 339
10.37.2.2 ConnectToRemoteHost . . . 339
10.37.2.3 Fini . . . 340
10.37.2.4 GetAllPublishers . . . 340
10.37.2.5 GetTopicNamespaces . . . 340
10.37.2.6 Init . . . 340
10.37.2.7 IsRunning . . . 341
10.37.2.8 RegisterTopicNamespace . . . 341
10.37.2.9 RemoveConnection . . . 341
10.37.2.10 Run . . . 341
10.37.2.11 Stop . . . 341
10.37.2.12 Subscribe . . . 342
10.37.2.13 TriggerUpdate . . . 342
10.37.2.14 Unadvertise . . . 342
10.37.2.15 Unsubscribe . . . 342
10.37.2.16 Unsubscribe . . . 342
10.37.3 Member Data Documentation . . . 343
10.37.3.1 eventConnections . . . 343
10.38 gazebo::event::ConnectionPrivate Class Reference . . . 343
10.38.1 Constructor & Destructor Documentation . . . 343
10.38.1.1 ConnectionPrivate . . . 343
10.38.1.2 ConnectionPrivate . . . 343
10.38.2 Member Data Documentation . . . 344

10.38.2.1 creationTime 344
10.38.2.2 event 344
10.38.2.3 id 344
10.39gazebo::transport::ConnectionReadTask Class Reference 344
10.39.1 Detailed Description 344
10.39.2 Constructor & Destructor Documentation 345
10.39.2.1 ConnectionReadTask 345
10.39.3 Member Function Documentation 345
10.39.3.1 execute 345
10.40gazebo::common::Console Class Reference 345
10.40.1 Detailed Description 346
10.40.2 Member Function Documentation 346
10.40.2.1 GetQuiet 346
10.40.2.2 SetQuiet 346
10.40.3 Member Data Documentation 346
10.40.3.1 dbg 346
10.40.3.2 err 347
10.40.3.3 log 347
10.40.3.4 msg 347
10.40.3.5 warn 347
10.41gazebo::physics::Contact Class Reference 347
10.41.1 Detailed Description 348
10.41.2 Constructor & Destructor Documentation 348
10.41.2.1 Contact 348
10.41.2.2 Contact 349
10.41.2.3 ∼Contact 349
10.41.3 Member Function Documentation 349
10.41.3.1 DebugString 349
10.41.3.2 FillMsg 349
10.41.3.3 operator= 349
10.41.3.4 operator= 350

10.41.3.5 Reset . . . 350
10.41.4 Member Data Documentation . . . 350
10.41.4.1 collision1 . . . 350
10.41.4.2 collision2 . . . 350
10.41.4.3 count . . . 350
10.41.4.4 depths . . . 350
10.41.4.5 normals . . . 350
10.41.4.6 positions . . . 351
10.41.4.7 time . . . 351
10.41.4.8 world . . . 351
10.41.4.9 wrench . . . 351
10.42 gazebo::physics::ContactManager Class Reference . . . 351
10.42.1 Detailed Description . . . 352
10.42.2 Constructor & Destructor Documentation . . . 352
10.42.2.1 ContactManager . . . 352
10.42.2.2 ∼ContactManager . . . 352
10.42.3 Member Function Documentation . . . 353
10.42.3.1 Clear . . . 353
10.42.3.2 CreateFilter . . . 353
10.42.3.3 CreateFilter . . . 353
10.42.3.4 CreateFilter . . . 353
10.42.3.5 GetContact . . . 354
10.42.3.6 GetContactCount . . . 354
10.42.3.7 GetContacts . . . 354
10.42.3.8 GetFilterCount . . . 354
10.42.3.9 HasFilter . . . 354
10.42.3.10 Init . . . 355
10.42.3.11 NewContact . . . 355
10.42.3.12 PublishContacts . . . 355
10.42.3.13 RemoveFilter . . . 355
10.42.3.14 ResetCount . . . 355

10.43gazebo::rendering::ContactVisualPrivate::ContactPoint Class Reference 356
10.43.1 Detailed Description . . . 356
10.43.2 Member Data Documentation . . . 356
10.43.2.1 depth . . . 356
10.43.2.2 normal . . . 356
10.43.2.3 sceneNode . . . 356
10.44gazebo::physics::ContactPublisher Class Reference . . . 357
10.44.1 Detailed Description . . . 357
10.44.2 Member Data Documentation . . . 357
10.44.2.1 collisionNames . . . 357
10.44.2.2 collisions . . . 357
10.44.2.3 contacts . . . 357
10.44.2.4 publisher . . . 357
10.45gazebo::sensors::ContactSensor Class Reference . . . 358
10.45.1 Detailed Description . . . 359
10.45.2 Constructor & Destructor Documentation . . . 359
10.45.2.1 ContactSensor . . . 359
10.45.2.2 ∼ContactSensor . . . 359
10.45.3 Member Function Documentation . . . 359
10.45.3.1 Fini . . . 359
10.45.3.2 GetCollisionContactCount . . . 360
10.45.3.3 GetCollisionCount . . . 360
10.45.3.4 GetCollisionName . . . 360
10.45.3.5 GetContacts . . . 360
10.45.3.6 GetContacts . . . 361
10.45.3.7 Init . . . 362
10.45.3.8 IsActive . . . 362
10.45.3.9 Load . . . 362
10.45.3.10Load . . . 362
10.45.3.11UpdateImpl . . . 363
10.46gazebo::rendering::ContactVisual Class Reference . . . 363

10.46.1 Detailed Description . . . 364
10.46.2 Constructor & Destructor Documentation . . . 364
10.46.2.1 ContactVisual . . . 364
10.46.2.2 ∼ContactVisual . . . 364
10.46.3 Member Function Documentation . . . 364
10.46.3.1 SetEnabled . . . 364
10.47 gazebo::rendering::ContactVisualPrivate Class Reference . . . 365
10.47.1 Detailed Description . . . 366
10.47.2 Member Data Documentation . . . 366
10.47.2.1 connections . . . 366
10.47.2.2 contactsMsg . . . 366
10.47.2.3 contactsSub . . . 366
10.47.2.4 enabled . . . 366
10.47.2.5 mutex . . . 367
10.47.2.6 node . . . 367
10.47.2.7 points . . . 367
10.47.2.8 receivedMsg . . . 367
10.47.2.9 topicName . . . 367
10.48 gazebo::rendering::Conversions Class Reference . . . 367
10.48.1 Detailed Description . . . 368
10.48.2 Member Function Documentation . . . 368
10.48.2.1 Convert . . . 368
10.48.2.2 Convert . . . 368
10.48.2.3 Convert . . . 369
10.48.2.4 Convert . . . 369
10.48.2.5 Convert . . . 369
10.48.2.6 Convert . . . 370
10.49 gazebo::physics::CylinderShape Class Reference . . . 370
10.49.1 Detailed Description . . . 371
10.49.2 Constructor & Destructor Documentation . . . 371
10.49.2.1 CylinderShape . . . 371

10.49.2.2 ~CylinderShape . . . 372
10.49.3 Member Function Documentation . . . 372
10.49.3.1 FillMsg . . . 372
10.49.3.2 GetLength . . . 372
10.49.3.3 GetRadius . . . 372
10.49.3.4 Init . . . 372
10.49.3.5 ProcessMsg . . . 373
10.49.3.6 SetLength . . . 373
10.49.3.7 SetRadius . . . 373
10.49.3.8 SetScale . . . 373
10.49.3.9 SetSize . . . 373
10.50 gazebo::physics::DARTBallJoint Class Reference . . . 374
10.50.1 Detailed Description . . . 376
10.50.2 Constructor & Destructor Documentation . . . 377
10.50.2.1 DARTBallJoint . . . 377
10.50.2.2 ~DARTBallJoint . . . 377
10.50.3 Member Function Documentation . . . 377
10.50.3.1 GetAnchor . . . 377
10.50.3.2 GetAngleImpl . . . 377
10.50.3.3 GetGlobalAxis . . . 378
10.50.3.4 GetHighStop . . . 378
10.50.3.5 GetLowStop . . . 378
10.50.3.6 GetMaxForce . . . 379
10.50.3.7 GetVelocity . . . 379
10.50.3.8 Init . . . 379
10.50.3.9 Load . . . 380
10.50.3.10 SetAxis . . . 380
10.50.3.11 SetForceImpl . . . 380
10.50.3.12 SetHighStop . . . 381
10.50.3.13 SetLowStop . . . 381
10.50.3.14 SetMaxForce . . . 381

10.50.3.15 SetVelocity 382
10.50.4 Member Data Documentation 382
10.50.4.1 dtBallJoint 382
10.51 gazebo::physics::DARTBoxShape Class Reference 382
10.51.1 Detailed Description 383
10.51.2 Constructor & Destructor Documentation 383
10.51.2.1 DARTBoxShape 384
10.51.2.2 ∼DARTBoxShape 384
10.51.3 Member Function Documentation 384
10.51.3.1 SetSize 384
10.52 gazebo::physics::DARTCollision Class Reference 384
10.52.1 Detailed Description 386
10.52.2 Constructor & Destructor Documentation 386
10.52.2.1 DARTCollision 386
10.52.2.2 ∼DARTCollision 386
10.52.3 Member Function Documentation 386
10.52.3.1 Fini 387
10.52.3.2 GetBoundingBox 387
10.52.3.3 GetCategoryBits 387
10.52.3.4 GetCollideBits 387
10.52.3.5 GetDARTBodyNode 387
10.52.3.6 GetDARTCollisionShape 388
10.52.3.7 Init 388
10.52.3.8 Load 388
10.52.3.9 OnPoseChange 388
10.52.3.10 SetCategoryBits 388
10.52.3.11 SetCollideBits 389
10.52.3.12 SetDARTCollisionShape 389
10.53 gazebo::physics::DARTCylinderShape Class Reference 389
10.53.1 Detailed Description 390
10.53.2 Constructor & Destructor Documentation 390

10.53.2.1 DARTCylinderShape . . . 391
10.53.2.2 ∼DARTCylinderShape . . . 391
10.53.3 Member Function Documentation . . . 391
10.53.3.1 SetSize . . . 391
10.54gazebo::physics::DARTHeightmapShape Class Reference . . . 391
10.54.1 Detailed Description . . . 392
10.54.2 Constructor & Destructor Documentation . . . 392
10.54.2.1 DARTHeightmapShape . . . 393
10.54.2.2 ∼DARTHeightmapShape . . . 393
10.54.3 Member Function Documentation . . . 393
10.54.3.1 Init . . . 393
10.55gazebo::physics::DARTHinge2Joint Class Reference . . . 393
10.55.1 Detailed Description . . . 395
10.55.2 Constructor & Destructor Documentation . . . 395
10.55.2.1 DARTHinge2Joint . . . 395
10.55.2.2 ∼DARTHinge2Joint . . . 396
10.55.3 Member Function Documentation . . . 396
10.55.3.1 GetAnchor . . . 396
10.55.3.2 GetAngleImpl . . . 396
10.55.3.3 GetGlobalAxis . . . 396
10.55.3.4 GetMaxForce . . . 397
10.55.3.5 GetVelocity . . . 397
10.55.3.6 Init . . . 397
10.55.3.7 Load . . . 398
10.55.3.8 SetAxis . . . 398
10.55.3.9 SetForceImpl . . . 398
10.55.3.10 SetMaxForce . . . 399
10.55.3.11 SetVelocity . . . 399
10.55.4 Member Data Documentation . . . 399
10.55.4.1 dtUniveralJoint . . . 399
10.56gazebo::physics::DARTHingeJoint Class Reference . . . 399

10.56.1 Detailed Description . . . 401
10.56.2 Constructor & Destructor Documentation . . . 401
10.56.2.1 DARTHingeJoint . . . 401
10.56.2.2 ∼DARTHingeJoint . . . 402
10.56.3 Member Function Documentation . . . 402
10.56.3.1 GetAnchor . . . 402
10.56.3.2 GetAngleImpl . . . 402
10.56.3.3 GetGlobalAxis . . . 402
10.56.3.4 GetMaxForce . . . 403
10.56.3.5 GetVelocity . . . 403
10.56.3.6 Init . . . 403
10.56.3.7 Load . . . 404
10.56.3.8 SetAxis . . . 404
10.56.3.9 SetForceImpl . . . 404
10.56.3.10 SetMaxForce . . . 405
10.56.3.11 SetVelocity . . . 405
10.56.4 Member Data Documentation . . . 405
10.56.4.1 dtRevoluteJoint . . . 405
10.57 gazebo::physics::DARTJoint Class Reference . . . 405
10.57.1 Detailed Description . . . 408
10.57.2 Constructor & Destructor Documentation . . . 408
10.57.2.1 DARTJoint . . . 408
10.57.2.2 ∼DARTJoint . . . 408
10.57.3 Member Function Documentation . . . 408
10.57.3.1 ApplyDamping . . . 408
10.57.3.2 AreConnected . . . 408
10.57.3.3 Attach . . . 409
10.57.3.4 Detach . . . 409
10.57.3.5 GetAngleCount . . . 409
10.57.3.6 GetDARTJoint . . . 409
10.57.3.7 GetDARTModel . . . 410

10.57.3.8 GetForce . . . 410
10.57.3.9 GetForceTorque . . . 410
10.57.3.10 GetHighStop . . . 411
10.57.3.11 GetJointLink . . . 411
10.57.3.12 GetLinkForce . . . 412
10.57.3.13 GetLinkTorque . . . 412
10.57.3.14 GetLowStop . . . 412
10.57.3.15 GetParam . . . 413
10.57.3.16 Init . . . 413
10.57.3.17 Load . . . 414
10.57.3.18 Reset . . . 414
10.57.3.19 SetAnchor . . . 414
10.57.3.20 SetDamping . . . 414
10.57.3.21 SetForce . . . 415
10.57.3.22 SetForceImpl . . . 415
10.57.3.23 SetHighStop . . . 415
10.57.3.24 SetLowStop . . . 416
10.57.3.25 SetParam . . . 416
10.57.3.26 SetStiffness . . . 416
10.57.3.27 SetStiffnessDamping . . . 417
10.57.4 Member Data Documentation . . . 417
10.57.4.1 dartPhysicsEngine . . . 417
10.57.4.2 dtChildBodyNode . . . 417
10.57.4.3 dtJoint . . . 417
10.58 gazebo::physics::DARTLink Class Reference . . . 418
10.58.1 Detailed Description . . . 421
10.58.2 Constructor & Destructor Documentation . . . 421
10.58.2.1 DARTLink . . . 421
10.58.2.2 ∼DARTLink . . . 421
10.58.3 Member Function Documentation . . . 421
10.58.3.1 AddDARTChildJoint . . . 421

10.58.3.2 AddForce . . . 421
10.58.3.3 AddForceAtRelativePosition . . . 422
10.58.3.4 AddForceAtWorldPosition . . . 422
10.58.3.5 AddRelativeForce . . . 422
10.58.3.6 AddRelativeTorque . . . 422
10.58.3.7 AddTorque . . . 423
10.58.3.8 Fini . . . 423
10.58.3.9 GetDARTBodyNode . . . 423
10.58.3.10 GetDARTModel . . . 423
10.58.3.11 GetDARTPhysics . . . 423
10.58.3.12 GetDARTWorld . . . 424
10.58.3.13 GetEnabled . . . 424
10.58.3.14 GetGravityMode . . . 424
10.58.3.15 GetKinematic . . . 424
10.58.3.16 GetWorldAngularVel . . . 425
10.58.3.17 GetWorldCoGLinearVel . . . 425
10.58.3.18 GetWorldForce . . . 425
10.58.3.19 GetWorldLinearVel . . . 425
10.58.3.20 GetWorldLinearVel . . . 426
10.58.3.21 GetWorldTorque . . . 426
10.58.3.22 Init . . . 426
10.58.3.23 Load . . . 427
10.58.3.24 OnPoseChange . . . 427
10.58.3.25 SetAngularDamping . . . 427
10.58.3.26 SetAngularVel . . . 427
10.58.3.27 SetAutoDisable . . . 428
10.58.3.28 SetDARTParentJoint . . . 428
10.58.3.29 SetEnabled . . . 428
10.58.3.30 SetForce . . . 428
10.58.3.31 SetGravityMode . . . 429
10.58.3.32 SetKinematic . . . 429

10.58.3.33 SetLinearDamping . . . 429
10.58.3.34 SetLinearVel . . . 429
10.58.3.35 SetLinkStatic . . . 430
10.58.3.36 SetSelfCollide . . . 430
10.58.3.37 SetTorque . . . 430
10.58.3.38 updateDirtyPoseFromDARTTransformation . . . 430
10.59 gazebo::physics::DARTMeshShape Class Reference . . . 431
10.59.1 Detailed Description . . . 432
10.59.2 Constructor & Destructor Documentation . . . 432
10.59.2.1 DARTMeshShape . . . 432
10.59.2.2 ∼DARTMeshShape . . . 432
10.59.3 Member Function Documentation . . . 432
10.59.3.1 Init . . . 432
10.59.3.2 Load . . . 432
10.59.3.3 Update . . . 433
10.60 gazebo::physics::DARTModel Class Reference . . . 433
10.60.1 Detailed Description . . . 434
10.60.2 Constructor & Destructor Documentation . . . 434
10.60.2.1 DARTModel . . . 434
10.60.2.2 ∼DARTModel . . . 435
10.60.3 Member Function Documentation . . . 435
10.60.3.1 BackupState . . . 435
10.60.3.2 Fini . . . 435
10.60.3.3 GetDARTPhysics . . . 435
10.60.3.4 GetDARTSkeleton . . . 435
10.60.3.5 GetDARTWorld . . . 435
10.60.3.6 Init . . . 435
10.60.3.7 Load . . . 435
10.60.3.8 RestoreState . . . 435
10.60.3.9 Update . . . 436
10.60.4 Member Data Documentation . . . 436

10.60.4.1 dtConfig . . . 436
10.60.4.2 dtSkeleton . . . 436
10.60.4.3 dtVelocity . . . 436
10.61 gazebo::physics::DARTMultiRayShape Class Reference . . . 436
10.61.1 Detailed Description . . . 438
10.62 gazebo::physics::DARTPhysics Class Reference . . . 438
10.62.1 Detailed Description . . . 440
10.62.2 Member Enumeration Documentation . . . 440
10.62.2.1 DARTParam . . . 440
10.62.3 Constructor & Destructor Documentation . . . 440
10.62.3.1 DARTPhysics . . . 440
10.62.3.2 ∼DARTPhysics . . . 440
10.62.4 Member Function Documentation . . . 440
10.62.4.1 CreateCollision . . . 440
10.62.4.2 CreateJoint . . . 441
10.62.4.3 CreateLink . . . 441
10.62.4.4 CreateModel . . . 441
10.62.4.5 CreateShape . . . 442
10.62.4.6 DebugPrint . . . 442
10.62.4.7 Fini . . . 442
10.62.4.8 GetDARTWorld . . . 442
10.62.4.9 GetParam . . . 442
10.62.4.10 GetType . . . 443
10.62.4.11 Init . . . 443
10.62.4.12 InitForThread . . . 443
10.62.4.13 Load . . . 443
10.62.4.14 OnPhysicsMsg . . . 444
10.62.4.15 OnRequest . . . 444
10.62.4.16 Reset . . . 444
10.62.4.17 SetGravity . . . 444
10.62.4.18 SetParam . . . 444

10.62.4.19 SetSeed . . . 445
10.62.4.20 UpdateCollision . . . 446
10.62.4.21 UpdatePhysics . . . 446
10.63 gazebo::physics::DARTPlaneShape Class Reference . . . 446
10.63.1 Detailed Description . . . 447
10.63.2 Constructor & Destructor Documentation . . . 448
10.63.2.1 DARTPlaneShape . . . 448
10.63.2.2 ~DARTPlaneShape . . . 448
10.63.3 Member Function Documentation . . . 448
10.63.3.1 CreatePlane . . . 448
10.63.3.2 SetAltitude . . . 448
10.64 gazebo::physics::DARTRayShape Class Reference . . . 449
10.64.1 Detailed Description . . . 450
10.65 gazebo::physics::DARTScrewJoint Class Reference . . . 450
10.65.1 Detailed Description . . . 452
10.65.2 Constructor & Destructor Documentation . . . 453
10.65.2.1 DARTScrewJoint . . . 453
10.65.2.2 ~DARTScrewJoint . . . 453
10.65.3 Member Function Documentation . . . 453
10.65.3.1 GetAnchor . . . 453
10.65.3.2 GetAngleImpl . . . 453
10.65.3.3 GetGlobalAxis . . . 454
10.65.3.4 GetHighStop . . . 454
10.65.3.5 GetLowStop . . . 454
10.65.3.6 GetMaxForce . . . 455
10.65.3.7 GetParam . . . 455
10.65.3.8 GetThreadPitch . . . 455
10.65.3.9 GetThreadPitch . . . 456
10.65.3.10 GetVelocity . . . 456
10.65.3.11 Init . . . 456
10.65.3.12 Load . . . 457

10.65.3.13SetAnchor . . . 457
10.65.3.14SetAxis . . . 457
10.65.3.15SetForceImpl . . . 457
10.65.3.16SetMaxForce . . . 458
10.65.3.17SetThreadPitch . . . 458
10.65.3.18SetThreadPitch . . . 458
10.65.3.19SetVelocity . . . 459
10.65.4 Member Data Documentation . . . 459
10.65.4.1 dartScrewJoint . . . 459
10.66gazebo::physics::DARTSliderJoint Class Reference . . . 459
10.66.1 Detailed Description . . . 461
10.66.2 Constructor & Destructor Documentation . . . 461
10.66.2.1 DARTSliderJoint . . . 461
10.66.2.2 ∼DARTSliderJoint . . . 462
10.66.3 Member Function Documentation . . . 462
10.66.3.1 GetAnchor . . . 462
10.66.3.2 GetAngleImpl . . . 462
10.66.3.3 GetGlobalAxis . . . 462
10.66.3.4 GetMaxForce . . . 463
10.66.3.5 GetVelocity . . . 463
10.66.3.6 Init . . . 463
10.66.3.7 Load . . . 464
10.66.3.8 SetAxis . . . 464
10.66.3.9 SetForceImpl . . . 464
10.66.3.10SetMaxForce . . . 465
10.66.3.11SetVelocity . . . 465
10.66.4 Member Data Documentation . . . 465
10.66.4.1 dtPrismaticJoint . . . 465
10.67gazebo::physics::DARTSphereShape Class Reference . . . 465
10.67.1 Detailed Description . . . 466
10.67.2 Constructor & Destructor Documentation . . . 467

10.67.2.1 DARTSphereShape 467
10.67.2.2 ∼DARTSphereShape 467
10.67.3 Member Function Documentation 467
10.67.3.1 SetRadius 467
10.68gazebo::physics::DARTTypes Class Reference 467
10.68.1 Detailed Description 468
10.69gazebo::physics::DARTUniversalJoint Class Reference 468
10.69.1 Detailed Description 470
10.69.2 Constructor & Destructor Documentation 470
10.69.2.1 DARTUniversalJoint 470
10.69.2.2 ∼DARTUniversalJoint 471
10.69.3 Member Function Documentation 471
10.69.3.1 GetAnchor 471
10.69.3.2 GetAngleImpl 471
10.69.3.3 GetGlobalAxis 471
10.69.3.4 GetMaxForce 472
10.69.3.5 GetVelocity 472
10.69.3.6 Init 472
10.69.3.7 Load 473
10.69.3.8 SetAxis 473
10.69.3.9 SetForceImpl 473
10.69.3.10SetMaxForce 474
10.69.3.11SetVelocity 474
10.69.4 Member Data Documentation 474
10.69.4.1 dtUniveralJoint 474
10.70gazebo::rendering::DepthCamera Class Reference 474
10.70.1 Detailed Description 476
10.70.2 Constructor & Destructor Documentation 476
10.70.2.1 DepthCamera 476
10.70.2.2 ∼DepthCamera 477
10.70.3 Member Function Documentation 477

10.70.3.1 ConnectNewDepthFrame . . . 477
10.70.3.2 ConnectNewRGBPointCloud . . . 477
10.70.3.3 CreateDepthTexture . . . 477
10.70.3.4 DisconnectNewDepthFrame . . . 478
10.70.3.5 DisconnectNewRGBPointCloud . . . 478
10.70.3.6 Fini . . . 478
10.70.3.7 GetDepthData . . . 478
10.70.3.8 Init . . . 478
10.70.3.9 Load . . . 479
10.70.3.10 Load . . . 479
10.70.3.11 PostRender . . . 479
10.70.3.12 SetDepthTarget . . . 479
10.70.4 Member Data Documentation . . . 479
10.70.4.1 depthTarget . . . 479
10.70.4.2 depthTexture . . . 480
10.70.4.3 depthViewport . . . 480
10.71 gazebo::sensors::DepthCameraSensor Class Reference . . . 480
10.71.1 Constructor & Destructor Documentation . . . 481
10.71.1.1 DepthCameraSensor . . . 481
10.71.1.2 ~DepthCameraSensor . . . 481
10.71.2 Member Function Documentation . . . 481
10.71.2.1 Fini . . . 481
10.71.2.2 GetDepthCamera . . . 482
10.71.2.3 Init . . . 482
10.71.2.4 Load . . . 482
10.71.2.5 Load . . . 482
10.71.2.6 SaveFrame . . . 483
10.71.2.7 SetActive . . . 483
10.71.2.8 UpdateImpl . . . 483
10.72 gazebo::util::DiagnosticManager Class Reference . . . 484
10.72.1 Detailed Description . . . 485

10.72.2 Member Function Documentation 485
10.72.2.1 GetLabel 485
10.72.2.2 GetLogPath 485
10.72.2.3 GetTime 485
10.72.2.4 GetTime 486
10.72.2.5 GetTimerCount 486
10.72.2.6 Init 486
10.72.2.7 Lap 486
10.72.2.8 StartTimer 487
10.72.2.9 StopTimer 487
10.73gazebo::util::DiagnosticTimer Class Reference 487
10.73.1 Detailed Description 488
10.73.2 Constructor & Destructor Documentation 488
10.73.2.1 DiagnosticTimer 488
10.73.2.2 ∼DiagnosticTimer 489
10.73.3 Member Function Documentation 489
10.73.3.1 GetName 489
10.73.3.2 Lap 489
10.73.3.3 Start 489
10.73.3.4 Stop 489
10.74gazebo::rendering::DummyPageProvider Class Reference 490
10.74.1 Detailed Description 490
10.74.2 Member Function Documentation 490
10.74.2.1 loadProceduralPage 490
10.74.2.2 prepareProceduralPage 490
10.74.2.3 unloadProceduralPage 490
10.74.2.4 unprepareProceduralPage 491
10.75gazebo::rendering::DynamicLines Class Reference 491
10.75.1 Detailed Description 492
10.75.2 Constructor & Destructor Documentation 492
10.75.2.1 DynamicLines 492

10.75.2.2 ~DynamicLines . . . 493
10.75.3 Member Function Documentation . . . 493
10.75.3.1 AddPoint . . . 493
10.75.3.2 AddPoint . . . 493
10.75.3.3 Clear . . . 493
10.75.3.4 GetMovableType . . . 493
10.75.3.5 getMovableType . . . 494
10.75.3.6 GetPoint . . . 494
10.75.3.7 GetPointCount . . . 494
10.75.3.8 SetColor . . . 494
10.75.3.9 SetPoint . . . 495
10.75.3.10 Update . . . 495
10.76 gazebo::rendering::DynamicRenderable Class Reference . . . 495
10.76.1 Detailed Description . . . 496
10.76.2 Constructor & Destructor Documentation . . . 497
10.76.2.1 DynamicRenderable . . . 497
10.76.2.2 ~DynamicRenderable . . . 497
10.76.3 Member Function Documentation . . . 497
10.76.3.1 CreateVertexDeclaration . . . 497
10.76.3.2 FillHardwareBuffers . . . 497
10.76.3.3 getBoundingRadius . . . 497
10.76.3.4 GetMovableType . . . 498
10.76.3.5 GetOperationType . . . 498
10.76.3.6 getSquaredViewDepth . . . 498
10.76.3.7 Init . . . 498
10.76.3.8 PrepareHardwareBuffers . . . 499
10.76.3.9 SetOperationType . . . 499
10.76.4 Member Data Documentation . . . 499
10.76.4.1 indexBufferCapacity . . . 500
10.76.4.2 vertexBufferCapacity . . . 500
10.77 gazebo::physics::Entity Class Reference . . . 500

10.77.1 Detailed Description 503
10.77.2 Constructor & Destructor Documentation 504
10.77.2.1 Entity 504
10.77.2.2 ∼Entity 504
10.77.3 Member Function Documentation 504
10.77.3.1 Fini 504
10.77.3.2 GetBoundingBox 504
10.77.3.3 GetChildCollision 504
10.77.3.4 GetChildLink 505
10.77.3.5 GetCollisionBoundingBox 505
10.77.3.6 GetDirtyPose 505
10.77.3.7 GetInitialRelativePose 505
10.77.3.8 GetNearestEntityBelow 506
10.77.3.9 GetParentModel 506
10.77.3.10 GetRelativeAngularAccel 506
10.77.3.11 GetRelativeAngularVel 506
10.77.3.12 GetRelativeLinearAccel 507
10.77.3.13 GetRelativeLinearVel 507
10.77.3.14 GetRelativePose 507
10.77.3.15 GetWorldAngularAccel 507
10.77.3.16 GetWorldAngularVel 508
10.77.3.17 GetWorldLinearAccel 508
10.77.3.18 GetWorldLinearVel 508
10.77.3.19 GetWorldPose 508
10.77.3.20 IsCanonicalLink 509
10.77.3.21 IsStatic 509
10.77.3.22 Load 509
10.77.3.23 OnPoseChange 509
10.77.3.24 PlaceOnEntity 510
10.77.3.25 PlaceOnNearestEntityBelow 510
10.77.3.26 Reset 510

10.77.3.27SetAnimation 510
10.77.3.28SetAnimation 510
10.77.3.29SetCanonicalLink 511
10.77.3.30SetInitialRelativePose 511
10.77.3.31SetName 511
10.77.3.32SetRelativePose 511
10.77.3.33SetStatic 512
10.77.3.34SetWorldPose 512
10.77.3.35SetWorldTwist 512
10.77.3.36StopAnimation 512
10.77.3.37UpdateParameters 512
10.77.4 Member Data Documentation 513
10.77.4.1 animation 513
10.77.4.2 animationConnection 513
10.77.4.3 animationStartPose 513
10.77.4.4 connections 513
10.77.4.5 dirtyPose 513
10.77.4.6 node 513
10.77.4.7 parentEntity 513
10.77.4.8 prevAnimationTime 514
10.77.4.9 requestPub 514
10.77.4.10scale 514
10.77.4.11visPub 514
10.77.4.12visualMsg 514
10.77.4.13worldPose 514
10.78gazebo::event::Event Class Reference 514
10.78.1 Detailed Description 516
10.78.2 Constructor & Destructor Documentation 516
10.78.2.1 Event 516
10.78.2.2 ∼Event 516
10.78.2.3 Event 516

10.78.3 Member Function Documentation 516
10.78.3.1 Disconnect 516
10.78.3.2 Disconnect 517
10.78.3.3 GetSignaled 517
10.78.4 Member Data Documentation 517
10.78.4.1 dataPtr 517
10.79 gazebo::event::EventPrivate Class Reference 518
10.79.1 Constructor & Destructor Documentation 518
10.79.1.1 EventPrivate 518
10.79.2 Member Data Documentation 519
10.79.2.1 signaled 519
10.80 gazebo::event::Events Class Reference 519
10.80.1 Detailed Description 522
10.80.2 Member Function Documentation 522
10.80.2.1 ConnectAddEntity 522
10.80.2.2 ConnectCreateEntity 522
10.80.2.3 ConnectDeleteEntity 523
10.80.2.4 ConnectDiagTimerStart 523
10.80.2.5 ConnectDiagTimerStop 523
10.80.2.6 ConnectPause 524
10.80.2.7 ConnectPostRender 524
10.80.2.8 ConnectPreRender 524
10.80.2.9 ConnectRender 525
10.80.2.10 ConnectSetSelectedEntity 525
10.80.2.11 ConnectSigInt 525
10.80.2.12 ConnectStep 526
10.80.2.13 ConnectStop 526
10.80.2.14 ConnectWorldCreated 526
10.80.2.15 ConnectWorldUpdateBegin 527
10.80.2.16 ConnectWorldUpdateEnd 527
10.80.2.17 DisconnectAddEntity 527

10.80.2.18 DisconnectCreateEntity . . . 528
10.80.2.19 DisconnectDeleteEntity . . . 528
10.80.2.20 DisconnectDiagTimerStart . . . 528
10.80.2.21 DisconnectDiagTimerStop . . . 528
10.80.2.22 DisconnectPause . . . 528
10.80.2.23 DisconnectPostRender . . . 529
10.80.2.24 DisconnectPreRender . . . 529
10.80.2.25 DisconnectRender . . . 529
10.80.2.26 DisconnectSetSelectedEntity . . . 529
10.80.2.27 DisconnectSigInt . . . 530
10.80.2.28 DisconnectStep . . . 530
10.80.2.29 DisconnectStop . . . 530
10.80.2.30 DisconnectWorldCreated . . . 530
10.80.2.31 DisconnectWorldUpdateBegin . . . 530
10.80.2.32 DisconnectWorldUpdateEnd . . . 531
10.80.3 Member Data Documentation . . . 531
10.80.3.1 addEntity . . . 531
10.80.3.2 deleteEntity . . . 531
10.80.3.3 diagTimerStart . . . 531
10.80.3.4 diagTimerStop . . . 531
10.80.3.5 entityCreated . . . 531
10.80.3.6 pause . . . 531
10.80.3.7 postRender . . . 532
10.80.3.8 preRender . . . 532
10.80.3.9 render . . . 532
10.80.3.10 setSelectedEntity . . . 532
10.80.3.11 sigInt . . . 532
10.80.3.12 step . . . 532
10.80.3.13 stop . . . 532
10.80.3.14 worldCreated . . . 532
10.80.3.15 worldUpdateBegin . . . 532

10.80.3.16 worldUpdateEnd . . . 533
10.81 gazebo::rendering::Events Class Reference . . . 533
10.81.1 Detailed Description . . . 533
10.81.2 Member Function Documentation . . . 534
10.81.2.1 ConnectCreateScene . . . 534
10.81.2.2 ConnectRemoveScene . . . 534
10.81.2.3 DisconnectCreateScene . . . 534
10.81.2.4 DisconnectRemoveScene . . . 535
10.81.3 Member Data Documentation . . . 535
10.81.3.1 createScene . . . 535
10.81.3.2 removeScene . . . 535
10.82 gazebo::event::EventT< T > Class Template Reference . . . 535
10.82.1 Detailed Description . . . 538
10.82.2 Member Function Documentation . . . 539
10.82.2.1 operator() . . . 539
10.82.2.2 operator() . . . 539
10.82.2.3 operator() . . . 539
10.82.2.4 operator() . . . 539
10.82.2.5 operator() . . . 540
10.82.2.6 operator() . . . 540
10.82.2.7 operator() . . . 540
10.82.2.8 operator() . . . 541
10.82.2.9 operator() . . . 541
10.82.2.10 operator() . . . 542
10.82.2.11 operator() . . . 542
10.82.2.12 Signal . . . 543
10.82.2.13 Signal . . . 543
10.82.2.14 Signal . . . 543
10.82.2.15 Signal . . . 543
10.82.2.16 Signal . . . 544
10.82.2.17 Signal . . . 544

10.82.2.18Signal 544
10.82.2.19Signal 545
10.82.2.20Signal 545
10.82.2.21Signal 546
10.82.2.22Signal 546
10.83gazebo::event::EventTPrivate< T > Class Template Reference 547
10.83.1 Member Data Documentation 547
10.83.1.1 connections 547
10.83.1.2 connectionsEraseMutex 548
10.83.1.3 connectionsToErase 548
10.84gazebo::common::Exception Class Reference 548
10.84.1 Detailed Description 549
10.84.2 Constructor & Destructor Documentation 549
10.84.2.1 Exception 549
10.84.2.2 Exception 549
10.84.2.3 ∼Exception 550
10.84.3 Member Function Documentation 550
10.84.3.1 GetErrorFile 550
10.84.3.2 GetErrorStr 550
10.84.3.3 Print 550
10.84.4 Friends And Related Function Documentation 550
10.84.4.1 operator<< 550
10.85gazebo::common::FileLogger Class Reference 551
10.85.1 Detailed Description 552
10.85.2 Constructor & Destructor Documentation 552
10.85.2.1 FileLogger 552
10.85.2.2 ∼FileLogger 552
10.85.3 Member Function Documentation 552
10.85.3.1 Init 552
10.85.3.2 operator() 553
10.85.3.3 operator() 553

10.86gazebo::math::Filter< T > Class Template Reference 553
10.86.1 Detailed Description 554
10.86.2 Constructor & Destructor Documentation 554
10.86.2.1 ∼Filter 555
10.86.3 Member Function Documentation 555
10.86.3.1 GetValue 555
10.86.3.2 SetFc 555
10.86.3.3 SetValue 555
10.86.4 Member Data Documentation 556
10.86.4.1 y0 556
10.87gazebo::sensors::ForceTorqueSensor Class Reference 556
10.87.1 Detailed Description 557
10.87.2 Constructor & Destructor Documentation 557
10.87.2.1 ForceTorqueSensor 557
10.87.2.2 ∼ForceTorqueSensor 558
10.87.3 Member Function Documentation 558
10.87.3.1 ConnectUpdate 558
10.87.3.2 DisconnectUpdate 558
10.87.3.3 Fini 558
10.87.3.4 GetForce 558
10.87.3.5 GetJoint 559
10.87.3.6 GetTopic 559
10.87.3.7 GetTorque 559
10.87.3.8 Init 559
10.87.3.9 IsActive 560
10.87.3.10Load 560
10.87.3.11Load 560
10.87.3.12UpdateImpl 560
10.87.4 Member Data Documentation 561
10.87.4.1 update 561
10.88gazebo::rendering::FPSViewController Class Reference 561

10.88.1 Detailed Description 562
10.88.2 Constructor & Destructor Documentation 562
10.88.2.1 FPSViewController 562
10.88.2.2 ∼FPSViewController 563
10.88.3 Member Function Documentation 563
10.88.3.1 GetTypeString 563
10.88.3.2 HandleKeyPressEvent 563
10.88.3.3 HandleKeyReleaseEvent 563
10.88.3.4 HandleMouseEvent 563
10.88.3.5 Init 564
10.88.3.6 Update 564
10.89 gazebo::physics::FrictionPyramid Class Reference 564
10.89.1 Detailed Description 565
10.89.2 Constructor & Destructor Documentation 565
10.89.2.1 FrictionPyramid 565
10.89.2.2 ∼FrictionPyramid 565
10.89.3 Member Function Documentation 565
10.89.3.1 GetMuPrimary 565
10.89.3.2 GetMuSecondary 565
10.89.3.3 SetMuPrimary 566
10.89.3.4 SetMuSecondary 566
10.89.4 Member Data Documentation 566
10.89.4.1 direction1 566
10.90 gazebo::sensors::GaussianNoiseModel Class Reference 566
10.90.1 Detailed Description 568
10.90.2 Constructor & Destructor Documentation 568
10.90.2.1 GaussianNoiseModel 568
10.90.2.2 ∼GaussianNoiseModel 568
10.90.3 Member Function Documentation 568
10.90.3.1 ApplyImpl 568
10.90.3.2 Fini 569

10.90.3.3 GetBias . . . 569
10.90.3.4 GetMean . . . 569
10.90.3.5 GetStdDev . . . 569
10.90.3.6 Load . . . 570
10.90.4 Member Data Documentation . . . 570
10.90.4.1 bias . . . 570
10.90.4.2 mean . . . 570
10.90.4.3 precision . . . 570
10.90.4.4 quantized . . . 570
10.90.4.5 stdDev . . . 570
10.91 google::protobuf::compiler::cpp::GazeboGenerator Class Reference . . . 571
10.91.1 Detailed Description . . . 571
10.91.2 Constructor & Destructor Documentation . . . 571
10.91.2.1 GazeboGenerator . . . 571
10.91.2.2 ∼GazeboGenerator . . . 571
10.91.3 Member Function Documentation . . . 571
10.91.3.1 Generate . . . 571
10.92 gazebo::physics::GearboxJoint< T > Class Template Reference . . . 571
10.92.1 Detailed Description . . . 572
10.92.2 Constructor & Destructor Documentation . . . 572
10.92.2.1 GearboxJoint . . . 572
10.92.2.2 ∼GearboxJoint . . . 573
10.92.3 Member Function Documentation . . . 573
10.92.3.1 GetAngleCount . . . 573
10.92.3.2 GetGearboxRatio . . . 573
10.92.3.3 Init . . . 573
10.92.3.4 Load . . . 573
10.92.3.5 SetGearboxRatio . . . 574
10.92.4 Member Data Documentation . . . 574
10.92.4.1 gearRatio . . . 574
10.92.4.2 referenceBody . . . 574

10.93 gazebo::common::GeometryIndices Class Reference 574
10.93.1 Detailed Description 575
10.93.2 Member Data Documentation 575
10.93.2.1 mappedIndex 575
10.93.2.2 normalIndex 575
10.93.2.3 texcoordIndex 575
10.93.2.4 vertexIndex 575
10.94 gazebo::sensors::GpsSensor Class Reference 575
10.94.1 Detailed Description 577
10.94.2 Constructor & Destructor Documentation 577
10.94.2.1 GpsSensor 577
10.94.2.2 ∼GpsSensor 577
10.94.3 Member Function Documentation 577
10.94.3.1 Fini 577
10.94.3.2 GetAltitude 577
10.94.3.3 GetLatitude 577
10.94.3.4 GetLongitude 578
10.94.3.5 Init 578
10.94.3.6 Load 578
10.94.3.7 Load 578
10.94.3.8 UpdateImpl 578
10.95 gazebo::rendering::GpuLaser Class Reference 579
10.95.1 Detailed Description 582
10.95.2 Constructor & Destructor Documentation 582
10.95.2.1 GpuLaser 582
10.95.2.2 ∼GpuLaser 582
10.95.3 Member Function Documentation 582
10.95.3.1 ConnectNewLaserFrame 583
10.95.3.2 CreateLaserTexture 583
10.95.3.3 DisconnectNewLaserFrame 583
10.95.3.4 Fini 583

10.95.3.5 GetCameraCount 584
10.95.3.6 GetCosHorzFOV 584
10.95.3.7 GetCosVertFOV 584
10.95.3.8 GetFarClip 584
10.95.3.9 GetHorzFOV 584
10.95.3.10 GetHorzHalfAngle 585
10.95.3.11 GetLaserData 585
10.95.3.12 GetNearClip 585
10.95.3.13 GetRayCountRatio 585
10.95.3.14 GetVertFOV 585
10.95.3.15 GetVertHalfAngle 586
10.95.3.16 Init 586
10.95.3.17 IsHorizontal 586
10.95.3.18 Load 586
10.95.3.19 Load 586
10.95.3.20 notifyRenderSingleObject 587
10.95.3.21 PostRender 587
10.95.3.22 SetCameraCount 587
10.95.3.23 SetCosHorzFOV 587
10.95.3.24 SetCosVertFOV 587
10.95.3.25 SetFarClip 587
10.95.3.26 SetHorzFOV 588
10.95.3.27 SetHorzHalfAngle 588
10.95.3.28 SetIsHorizontal 588
10.95.3.29 SetNearClip 588
10.95.3.30 SetRangeCount 588
10.95.3.31 SetRayCountRatio 589
10.95.3.32 SetVertFOV 589
10.95.3.33 SetVertHalfAngle 589
10.95.4 Member Data Documentation 589
10.95.4.1 cameraCount 589

10.95.4.2 chfov . . . 589
10.95.4.3 cvfov . . . 590
10.95.4.4 far . . . 590
10.95.4.5 hfov . . . 590
10.95.4.6 horzHalfAngle . . . 590
10.95.4.7 isHorizontal . . . 590
10.95.4.8 near . . . 590
10.95.4.9 rayCountRatio . . . 590
10.95.4.10 vertHalfAngle . . . 590
10.95.4.11 vfov . . . 590
10.96 gazebo::sensors::GpuRaySensor Class Reference . . . 591
10.96.1 Constructor & Destructor Documentation . . . 594
10.96.1.1 GpuRaySensor . . . 594
10.96.1.2 ∼GpuRaySensor . . . 594
10.96.2 Member Function Documentation . . . 594
10.96.2.1 ConnectNewLaserFrame . . . 594
10.96.2.2 DisconnectNewLaserFrame . . . 595
10.96.2.3 Fini . . . 595
10.96.2.4 GetAngleMax . . . 595
10.96.2.5 GetAngleMin . . . 595
10.96.2.6 GetAngleResolution . . . 595
10.96.2.7 GetCameraCount . . . 596
10.96.2.8 GetCosHorzFOV . . . 596
10.96.2.9 GetCosVertFOV . . . 596
10.96.2.10 GetFiducial . . . 596
10.96.2.11 GetHorzFOV . . . 597
10.96.2.12 GetHorzHalfAngle . . . 597
10.96.2.13 GetLaserCamera . . . 597
10.96.2.14 GetRange . . . 597
10.96.2.15 GetRangeCount . . . 598
10.96.2.16 GetRangeCountRatio . . . 598

10.96.2.17GetRangeMax 598
10.96.2.18GetRangeMin 598
10.96.2.19GetRangeResolution 598
10.96.2.20GetRanges 599
10.96.2.21GetRayCount 599
10.96.2.22GetRayCountRatio 599
10.96.2.23GetRetro 599
10.96.2.24GetTopic 600
10.96.2.25GetVertFOV 600
10.96.2.26GetVertHalfAngle 600
10.96.2.27GetVerticalAngleMax 600
10.96.2.28GetVerticalAngleMin 600
10.96.2.29GetVerticalAngleResolution 601
10.96.2.30GetVerticalRangeCount 601
10.96.2.31GetVerticalRayCount 601
10.96.2.32Init 601
10.96.2.33IsActive 601
10.96.2.34IsHorizontal 602
10.96.2.35Load 602
10.96.2.36Load 602
10.96.2.37SetAngleMax 602
10.96.2.38SetAngleMin 602
10.96.2.39SetVerticalAngleMax 603
10.96.2.40SetVerticalAngleMin 603
10.96.2.41UpdateImpl 603
10.96.3 Member Data Documentation 603
10.96.3.1 cameraElem 604
10.96.3.2 horzElem 604
10.96.3.3 horzRangeCount 604
10.96.3.4 horzRayCount 604
10.96.3.5 rangeCountRatio 604

10.96.3.6 rangeElem 604
10.96.3.7 scanElem 604
10.96.3.8 vertElem 604
10.96.3.9 vertRangeCount 605
10.96.3.10 vertRayCount 605
10.97 gazebo::rendering::Grid Class Reference 605
10.97.1 Detailed Description 606
10.97.2 Constructor & Destructor Documentation 606
10.97.2.1 Grid 606
10.97.2.2 ∼Grid 606
10.97.3 Member Function Documentation 607
10.97.3.1 Enable 607
10.97.3.2 GetCellCount 607
10.97.3.3 GetCellLength 607
10.97.3.4 GetColor 607
10.97.3.5 GetHeight 607
10.97.3.6 GetLineWidth 608
10.97.3.7 GetSceneNode 608
10.97.3.8 Init 608
10.97.3.9 SetCellCount 608
10.97.3.10 SetCellLength 608
10.97.3.11 SetColor 608
10.97.3.12 SetHeight 609
10.97.3.13 SetLineWidth 609
10.97.3.14 SetUserData 609
10.98 gazebo::physics::Gripper Class Reference 609
10.98.1 Detailed Description 610
10.98.2 Constructor & Destructor Documentation 610
10.98.2.1 Gripper 610
10.98.2.2 ∼Gripper 610
10.98.3 Member Function Documentation 611

10.98.3.1 GetName . . . 611
10.98.3.2 Init . . . 611
10.98.3.3 IsAttached . . . 611
10.98.3.4 Load . . . 611
10.98.4 Member Data Documentation . . . 611
10.98.4.1 node . . . 611
10.99 gazebo::rendering::GUIOverlay Class Reference . . . 612
10.99.1 Detailed Description . . . 613
10.99.2 Constructor & Destructor Documentation . . . 613
10.99.2.1 GUIOverlay . . . 613
10.99.2.2 ∼GUIOverlay . . . 613
10.99.3 Member Function Documentation . . . 613
10.99.3.1 AttachCameraToImage . . . 613
10.99.3.2 AttachCameraToImage . . . 614
10.99.3.3 ButtonCallback . . . 614
10.99.3.4 CreateWindow . . . 614
10.99.3.5 HandleKeyPressEvent . . . 615
10.99.3.6 HandleKeyReleaseEvent . . . 615
10.99.3.7 HandleMouseEvent . . . 615
10.99.3.8 Hide . . . 616
10.99.3.9 Init . . . 616
10.99.3.10 IsInitialized . . . 616
10.99.3.11 LoadLayout . . . 616
10.99.3.12 Resize . . . 617
10.99.3.13 Show . . . 617
10.99.3.14 Update . . . 617
10.99.4 Member Data Documentation . . . 617
10.99.4.1 callbacks . . . 617
10.100 gazebo::rendering::GUIOverlayPrivate Class Reference . . . 617
10.100.1 Detailed Description . . . 618
10.100.2 Member Data Documentation . . . 618

10.100.2.1 connections . . . 618
10.100.2.2 initialized . . . 618
10.100.2.3 layoutFilename . . . 618
10.100.2.4 rttImageSetCount . . . 618
10.101 gazebo::rendering::GzTerrainMatGen Class Reference . . . 618
10.101.1 Constructor & Destructor Documentation . . . 619
10.101.1.1 GzTerrainMatGen . . . 619
10.101.1.2 ~GzTerrainMatGen . . . 619
10.102 gazebo::rendering::Heightmap Class Reference . . . 619
10.102.1 Detailed Description . . . 620
10.102.2 Constructor & Destructor Documentation . . . 620
10.102.2.1 Heightmap . . . 620
10.102.2.2 ~Heightmap . . . 621
10.102.3 Member Function Documentation . . . 621
10.102.3.1 Flatten . . . 621
10.102.3.2 GetAvgHeight . . . 621
10.102.3.3 GetHeight . . . 621
10.102.3.4 GetImage . . . 622
10.102.3.5 GetMouseHit . . . 622
10.102.3.6 GetOgreTerrain . . . 622
10.102.3.7 GetTerrainSubdivisionCount . . . 623
10.102.3.8 Load . . . 623
10.102.3.9 LoadFromMsg . . . 623
10.102.3.10 Lower . . . 623
10.102.3.11 Raise . . . 624
10.102.3.12 SetWireframe . . . 624
10.102.3.13 Smooth . . . 624
10.102.3.14 SplitHeights . . . 625
10.103 gazebo::common::HeightmapData Class Reference . . . 625
10.103.1 Detailed Description . . . 626
10.103.2 Constructor & Destructor Documentation . . . 626

10.103.2.1 ~HeightmapData . . . 626
10.103.3 Member Function Documentation . . . 626
10.103.3.1 FillHeightMap . . . 626
10.103.3.2 GetHeight . . . 627
10.103.3.3 GetMaxElevation . . . 627
10.103.3.4 GetWidth . . . 627
10.104 gazebo::physics::HeightmapShape Class Reference . . . 628
10.104.1 Detailed Description . . . 629
10.104.2 Constructor & Destructor Documentation . . . 630
10.104.2.1 HeightmapShape . . . 630
10.104.2.2 ~HeightmapShape . . . 630
10.104.3 Member Function Documentation . . . 630
10.104.3.1 FillMsg . . . 630
10.104.3.2 GetHeight . . . 630
10.104.3.3 GetImage . . . 631
10.104.3.4 GetMaxHeight . . . 631
10.104.3.5 GetMinHeight . . . 631
10.104.3.6 GetPos . . . 631
10.104.3.7 GetSize . . . 631
10.104.3.8 GetSubSampling . . . 632
10.104.3.9 GetURI . . . 632
10.104.3.10 GetVertexCount . . . 632
10.104.3.11 Init . . . 632
10.104.3.12 Load . . . 632
10.104.3.13 ProcessMsg . . . 633
10.104.3.14 SetScale . . . 633
10.104.4 Member Data Documentation . . . 633
10.104.4.1 flipY . . . 633
10.104.4.2 heightmapData . . . 633
10.104.4.3 heights . . . 633
10.104.4.4 img . . . 634

10.104.4.5 subSampling 634
10.104.4.6 vertSize 634
10.105 gazebo::physics::Hinge2Joint< T > Class Template Reference 634
10.105.1 Detailed Description 634
10.105.2 Constructor & Destructor Documentation 635
10.105.2.1 Hinge2Joint 635
10.105.2.2 ~Hinge2Joint 635
10.105.3 Member Function Documentation 635
10.105.3.1 GetAngleCount 635
10.105.3.2 Load 635
10.106 gazebo::physics::HingeJoint< T > Class Template Reference 635
10.106.1 Detailed Description 636
10.106.2 Constructor & Destructor Documentation 636
10.106.2.1 HingeJoint 636
10.106.2.2 ~HingeJoint 636
10.106.3 Member Function Documentation 636
10.106.3.1 GetAngleCount 637
10.106.3.2 Init 637
10.106.3.3 Load 637
10.107 gazebo::common::Image Class Reference 637
10.107.1 Detailed Description 639
10.107.2 Member Enumeration Documentation 639
10.107.2.1 PixelFormat 639
10.107.3 Constructor & Destructor Documentation 639
10.107.3.1 Image 639
10.107.3.2 ~Image 640
10.107.4 Member Function Documentation 640
10.107.4.1 ConvertPixelFormat 640
10.107.4.2 GetAvgColor 640
10.107.4.3 GetBPP 640
10.107.4.4 GetData 641

10.107.4.5 GetFilename . . . 641
10.107.4.6 GetHeight . . . 641
10.107.4.7 GetMaxColor . . . 641
10.107.4.8 GetPitch . . . 641
10.107.4.9 GetPixel . . . 642
10.107.4.10 GetPixelFormat . . . 642
10.107.4.11 GetRGBData . . . 642
10.107.4.12 GetWidth . . . 642
10.107.4.13 Load . . . 643
10.107.4.14 Rescale . . . 643
10.107.4.15 SavePNG . . . 643
10.107.4.16 SetFromData . . . 643
10.107.4.17 Valid . . . 644
10.108 gazebo::sensors::ImageGaussianNoiseModel Class Reference . . . 644
10.108.1 Constructor & Destructor Documentation . . . 645
10.108.1.1 ImageGaussianNoiseModel . . . 645
10.108.1.2 ~ImageGaussianNoiseModel . . . 645
10.108.2 Member Function Documentation . . . 645
10.108.2.1 Fini . . . 645
10.108.2.2 Load . . . 646
10.108.2.3 SetCamera . . . 646
10.108.3 Member Data Documentation . . . 646
10.108.3.1 gaussianNoiseCompositorListener . . . 646
10.108.3.2 gaussianNoiseInstance . . . 646
10.109 gazebo::common::ImageHeightmap Class Reference . . . 647
10.109.1 Detailed Description . . . 648
10.109.2 Constructor & Destructor Documentation . . . 648
10.109.2.1 ImageHeightmap . . . 648
10.109.3 Member Function Documentation . . . 648
10.109.3.1 FillHeightMap . . . 648
10.109.3.2 GetFilename . . . 648

10.109.3.3 GetHeight . . . 649
10.109.3.4 GetMaxElevation . . . 649
10.109.3.5 GetWidth . . . 649
10.109.3.6 Load . . . 649
10.110 gazebo::sensors::ImuSensor Class Reference . . . 650
10.110.1 Detailed Description . . . 651
10.110.2 Constructor & Destructor Documentation . . . 651
10.110.2.1 ImuSensor . . . 651
10.110.2.2 ~ImuSensor . . . 651
10.110.3 Member Function Documentation . . . 651
10.110.3.1 Fini . . . 652
10.110.3.2 GetAngularVelocity . . . 652
10.110.3.3 GetImuMessage . . . 652
10.110.3.4 GetLinearAcceleration . . . 652
10.110.3.5 GetOrientation . . . 652
10.110.3.6 Init . . . 653
10.110.3.7 IsActive . . . 653
10.110.3.8 Load . . . 653
10.110.3.9 Load . . . 653
10.110.3.10 SetReferencePose . . . 653
10.110.3.11 UpdateImpl . . . 654
10.111 gazebo::physics::Inertial Class Reference . . . 654
10.111.1 Detailed Description . . . 657
10.111.2 Constructor & Destructor Documentation . . . 657
10.111.2.1 Inertial . . . 657
10.111.2.2 Inertial . . . 657
10.111.2.3 Inertial . . . 657
10.111.2.4 ~Inertial . . . 657
10.111.3 Member Function Documentation . . . 657
10.111.3.1 GetCoG . . . 657
10.111.3.2 GetInertial . . . 658

10.111.3.3 GetIXX . . . 658
10.111.3.4 GetIXY . . . 658
10.111.3.5 GetIXZ . . . 658
10.111.3.6 GetIYY . . . 659
10.111.3.7 GetIYZ . . . 659
10.111.3.8 GetIZZ . . . 659
10.111.3.9 GetMass . . . 659
10.111.3.10 GetMOI . . . 659
10.111.3.11 GetMOI . . . 660
10.111.3.12 GetPose . . . 660
10.111.3.13 GetPrincipalMoments . . . 660
10.111.3.14 GetProductsofInertia . . . 660
10.111.3.15 Load . . . 661
10.111.3.16 operator+ . . . 661
10.111.3.17 operator+= . . . 661
10.111.3.18 operator= . . . 661
10.111.3.19 ProcessMsg . . . 662
10.111.3.20 Reset . . . 662
10.111.3.21 Rotate . . . 662
10.111.3.22 SetCoG . . . 662
10.111.3.23 SetCoG . . . 662
10.111.3.24 SetCoG . . . 663
10.111.3.25 SetCoG . . . 663
10.111.3.26 SetInertiaMatrix . . . 663
10.111.3.27 SetIXX . . . 664
10.111.3.28 SetIXY . . . 664
10.111.3.29 SetIXZ . . . 664
10.111.3.30 SetIYY . . . 664
10.111.3.31 SetIYZ . . . 664
10.111.3.32 SetIZZ . . . 665
10.111.3.33 SetMass . . . 665

10.111.3.3 GetMOI 665
10.111.3.4 UpdateParameters 665
10.111.4 Friends And Related Function Documentation 665
10.111.4.1 operator<< 665
10.112 gazebo::common::InternalError Class Reference 666
10.112.1 Detailed Description 667
10.112.2 Constructor & Destructor Documentation 667
10.112.2.1 InternalError 667
10.112.2.2 InternalError 667
10.112.2.3 ~InternalError 667
10.113 gazebo::transport::IOManager Class Reference 667
10.113.1 Detailed Description 668
10.113.2 Constructor & Destructor Documentation 668
10.113.2.1 IOManager 668
10.113.2.2 ~IOManager 668
10.113.3 Member Function Documentation 668
10.113.3.1 DecCount 668
10.113.3.2 GetCount 668
10.113.3.3 GetIO 669
10.113.3.4 IncCount 669
10.113.3.5 Stop 669
10.114 gazebo::physics::Joint Class Reference 669
10.114.1 Detailed Description 675
10.114.2 Member Enumeration Documentation 675
10.114.2.1 Attribute 675
10.114.3 Constructor & Destructor Documentation 676
10.114.3.1 Joint 676
10.114.3.2 ~Joint 676
10.114.4 Member Function Documentation 676
10.114.4.1 ApplyStiffnessDamping 676
10.114.4.2 AreConnected 676

10.114.4.3 Attach 677
10.114.4.4 CacheForceTorque 677
10.114.4.5 CheckAndTruncateForce 677
10.114.4.6 ComputeChildLinkPose 677
10.114.4.7 ConnectJointUpdate 678
10.114.4.8 Detach 678
10.114.4.9 DisconnectJointUpdate 678
10.114.4.10 FillMsg 678
10.114.4.11 FindAllConnectedLinks 679
10.114.4.12 Fini 679
10.114.4.13 GetAnchor 679
10.114.4.14 GetAnchorErrorPose 680
10.114.4.15 GetAngle 680
10.114.4.16 GetAngleCount 680
10.114.4.17 GetAngleImpl 681
10.114.4.18 GetAxisFrame 681
10.114.4.19 GetAxisFrameOffset 682
10.114.4.20 GetChild 682
10.114.4.21 GetDamping 682
10.114.4.22 GetEffortLimit 683
10.114.4.23 GetForce 683
10.114.4.24 GetForceTorque 683
10.114.4.25 GetGlobalAxis 684
10.114.4.26 GetHighStop 684
10.114.4.27 GetInertiaRatio 685
10.114.4.28 GetInertiaRatio 685
10.114.4.29 GetInitialAnchorPose 686
10.114.4.30 GetJointLink 686
10.114.4.31 GetLinkForce 686
10.114.4.32 GetLinkTorque 687
10.114.4.33 GetLocalAxis 687

10.114.4.34 GetLowerLimit 687
10.114.4.35 GetLowStop 688
10.114.4.36 GetMaxForce 688
10.114.4.37 GetParam 689
10.114.4.38 GetParent 689
10.114.4.39 GetParentWorldPose 689
10.114.4.40 GetSpringReferencePosition 690
10.114.4.41 GetStiffness 690
10.114.4.42 GetStopDissipation 690
10.114.4.43 GetStopStiffness 691
10.114.4.44 GetUpperLimit 691
10.114.4.45 GetVelocity 691
10.114.4.46 GetVelocityLimit 692
10.114.4.47 GetWorldEnergyPotentialSpring 692
10.114.4.48 GetWorldPose 692
10.114.4.49 Init 693
10.114.4.50 Load 693
10.114.4.51 Load 693
10.114.4.52 Reset 694
10.114.4.53 SetAnchor 694
10.114.4.54 SetAngle 694
10.114.4.55 SetAxis 695
10.114.4.56 SetDamping 695
10.114.4.57 SetEffortLimit 695
10.114.4.58 SetForce 696
10.114.4.59 SetHighStop 696
10.114.4.60 SetLowerLimit 696
10.114.4.61 SetLowStop 697
10.114.4.62 SetMaxForce 697
10.114.4.63 SetModel 698
10.114.4.64 SetParam 698

10.114.4.65 SetPosition 698
10.114.4.66 SetPositionMaximal 699
10.114.4.67 SetProvideFeedback 699
10.114.4.68 SetState 699
10.114.4.69 SetStiffness 699
10.114.4.70 SetStiffnessDamping 700
10.114.4.71 SetStopDissipation 700
10.114.4.72 SetStopStiffness 700
10.114.4.73 SetUpperLimit 701
10.114.4.74 SetVelocity 701
10.114.4.75 Update 701
10.114.4.76 UpdateParameters 702
10.114.5 Member Data Documentation 702
10.114.5.1 anchorLink 702
10.114.5.2 anchorPos 702
10.114.5.3 anchorPose 702
10.114.5.4 applyDamping 702
10.114.5.5 axisParentModelFrame 702
10.114.5.6 childLink 703
10.114.5.7 dissipationCoefficient 703
10.114.5.8 effortLimit 703
10.114.5.9 lowerLimit 703
10.114.5.10 model 703
10.114.5.11 parentAnchorPose 703
10.114.5.12 parentLink 703
10.114.5.13 provideFeedback 703
10.114.5.14 springReferencePosition 703
10.114.5.15 stiffnessCoefficient 704
10.114.5.16 upperLimit 704
10.114.5.17 velocityLimit 704
10.114.5.18 wrench 704

10.115 gazebo::physics::JointController Class Reference 704
10.115.1 Detailed Description 705
10.115.2 Constructor & Destructor Documentation 706
10.115.2.1 JointController 706
10.115.2.2 ~JointController 706
10.115.3 Member Function Documentation 706
10.115.3.1 AddJoint 706
10.115.3.2 GetForces 706
10.115.3.3 GetJoints 706
10.115.3.4 GetLastUpdateTime 707
10.115.3.5 GetPositionPIDs 707
10.115.3.6 GetPositions 707
10.115.3.7 GetVelocities 707
10.115.3.8 GetVelocityPIDs 708
10.115.3.9 Reset 708
10.115.3.10 SetJointPosition 708
10.115.3.11 SetJointPosition 708
10.115.3.12 SetJointPositions 709
10.115.3.13 SetPositionPID 709
10.115.3.14 SetPositionTarget 709
10.115.3.15 SetVelocityPID 709
10.115.3.16 SetVelocityTarget 710
10.115.3.17 Update 710
10.116 gazebo::physics::JointControllerPrivate Class Reference 710
10.116.1 Member Data Documentation 711
10.116.1.1 forces 711
10.116.1.2 jointCmdSub 711
10.116.1.3 joints 711
10.116.1.4 model 711
10.116.1.5 node 711
10.116.1.6 positions 711

10.116.1.7 posPids . . . 712
10.116.1.8 prevUpdateTime . . . 712
10.116.1.9 updatedLinks . . . 712
10.116.1.10 velocities . . . 712
10.116.1.11 velPids . . . 712
10.117 gazebo::physics::JointState Class Reference . . . 712
10.117.1 Detailed Description . . . 714
10.117.2 Constructor & Destructor Documentation . . . 714
10.117.2.1 JointState . . . 714
10.117.2.2 JointState . . . 714
10.117.2.3 JointState . . . 715
10.117.2.4 JointState . . . 715
10.117.2.5 ~JointState . . . 715
10.117.3 Member Function Documentation . . . 715
10.117.3.1 FillSDF . . . 715
10.117.3.2 GetAngle . . . 715
10.117.3.3 GetAngleCount . . . 716
10.117.3.4 GetAngles . . . 716
10.117.3.5 IsZero . . . 716
10.117.3.6 Load . . . 717
10.117.3.7 Load . . . 717
10.117.3.8 operator+ . . . 717
10.117.3.9 operator- . . . 717
10.117.3.10 operator= . . . 718
10.117.4 Friends And Related Function Documentation . . . 718
10.117.4.1 operator<< . . . 718
10.118 gazebo::rendering::JointVisual Class Reference . . . 718
10.118.1 Detailed Description . . . 719
10.118.2 Constructor & Destructor Documentation . . . 719
10.118.2.1 JointVisual . . . 719
10.118.2.2 ~JointVisual . . . 720

10.118.3 Member Function Documentation 720
10.118.3.1 Load 720
10.119 gazebo::rendering::JointVisualPrivate Class Reference 720
10.119.1 Detailed Description 721
10.119.2 Member Data Documentation 721
10.119.2.1 axisVisual 721
10.120 gazebo::physics::JointWrench Class Reference 721
10.120.1 Detailed Description 722
10.120.2 Member Function Documentation 722
10.120.2.1 operator+ 722
10.120.2.2 operator- 722
10.120.2.3 operator= 722
10.120.3 Member Data Documentation 723
10.120.3.1 body1Force 723
10.120.3.2 body1Torque 723
10.120.3.3 body2Force 723
10.120.3.4 body2Torque 723
10.121 gazebo::common::KeyEvent Class Reference 723
10.121.1 Detailed Description 724
10.121.2 Member Enumeration Documentation 724
10.121.2.1 EventType 724
10.121.3 Constructor & Destructor Documentation 724
10.121.3.1 KeyEvent 724
10.121.4 Member Data Documentation 725
10.121.4.1 key 725
10.121.4.2 type 725
10.122 gazebo::common::KeyFrame Class Reference 725
10.122.1 Detailed Description 726
10.122.2 Constructor & Destructor Documentation 726
10.122.2.1 KeyFrame 726
10.122.2.2 ~KeyFrame 726

10.122.3 Member Function Documentation . . . 726
10.122.3.1 GetTime . . . 726
10.122.4 Member Data Documentation . . . 726
10.122.4.1 time . . . 726
10.123 gazebo::rendering::LaserVisual Class Reference . . . 727
10.123.1 Detailed Description . . . 727
10.123.2 Constructor & Destructor Documentation . . . 727
10.123.2.1 LaserVisual . . . 727
10.123.2.2 ~LaserVisual . . . 728
10.123.3 Member Function Documentation . . . 728
10.123.3.1 SetEmissive . . . 728
10.124 gazebo::rendering::LaserVisualPrivate Class Reference . . . 728
10.124.1 Detailed Description . . . 729
10.124.2 Member Data Documentation . . . 729
10.124.2.1 connection . . . 729
10.124.2.2 laserMsg . . . 729
10.124.2.3 laserScanSub . . . 729
10.124.2.4 mutex . . . 730
10.124.2.5 node . . . 730
10.124.2.6 rayFans . . . 730
10.124.2.7 receivedMsg . . . 730
10.125 gazebo::rendering::Light Class Reference . . . 730
10.125.1 Detailed Description . . . 732
10.125.2 Constructor & Destructor Documentation . . . 732
10.125.2.1 Light . . . 732
10.125.2.2 ~Light . . . 733
10.125.3 Member Function Documentation . . . 733
10.125.3.1 Clone . . . 733
10.125.3.2 FillMsg . . . 733
10.125.3.3 GetDiffuseColor . . . 733
10.125.3.4 GetDirection . . . 733

10.125.3.5 GetName . . . 734
10.125.3.6 GetPosition . . . 734
10.125.3.7 GetRotation . . . 734
10.125.3.8 GetSpecularColor . . . 734
10.125.3.9 GetType . . . 734
10.125.3.10 GetVisible . . . 735
10.125.3.11 Load . . . 735
10.125.3.12 Load . . . 735
10.125.3.13 LoadFromMsg . . . 735
10.125.3.14 LoadFromMsg . . . 735
10.125.3.15 OnPoseChange . . . 735
10.125.3.16 SetAttenuation . . . 736
10.125.3.17 SetCastShadows . . . 736
10.125.3.18 SetDiffuseColor . . . 736
10.125.3.19 SetDirection . . . 736
10.125.3.20 SetLightType . . . 736
10.125.3.21 SetName . . . 737
10.125.3.22 SetPosition . . . 737
10.125.3.23 SetRange . . . 737
10.125.3.24 SetRotation . . . 737
10.125.3.25 SetSelected . . . 738
10.125.3.26 SetSpecularColor . . . 738
10.125.3.27 SetSpotFalloff . . . 738
10.125.3.28 SetSpotInnerAngle . . . 738
10.125.3.29 SetSpotOuterAngle . . . 738
10.125.3.30 ShowVisual . . . 739
10.125.3.31 ToggleShowVisual . . . 739
10.125.3.32 UpdateFromMsg . . . 739
10.126 gazebo::physics::Link Class Reference . . . 739
10.126.1 Detailed Description . . . 746
10.126.2 Member Typedef Documentation . . . 746

10.126.2.1 Visuals_M . . . 746
10.126.3 Constructor & Destructor Documentation . . . 746
10.126.3.1 Link . . . 746
10.126.3.2 ~Link . . . 746
10.126.4 Member Function Documentation . . . 746
10.126.4.1 AddChildJoint . . . 746
10.126.4.2 AddForce . . . 747
10.126.4.3 AddForceAtRelativePosition . . . 747
10.126.4.4 AddForceAtWorldPosition . . . 747
10.126.4.5 AddParentJoint . . . 747
10.126.4.6 AddRelativeForce . . . 748
10.126.4.7 AddRelativeTorque . . . 748
10.126.4.8 AddTorque . . . 748
10.126.4.9 AttachStaticModel . . . 749
10.126.4.10 ConnectEnabled . . . 749
10.126.4.11 DetachAllStaticModels . . . 749
10.126.4.12 DetachStaticModel . . . 749
10.126.4.13 DisconnectEnabled . . . 749
10.126.4.14 FillMsg . . . 750
10.126.4.15 FindAllConnectedLinksHelper . . . 750
10.126.4.16 Fini . . . 750
10.126.4.17 GetAngularDamping . . . 751
10.126.4.18 GetBoundingBox . . . 751
10.126.4.19 GetChildJoints . . . 751
10.126.4.20 GetChildJointsLinks . . . 751
10.126.4.21 GetCollision . . . 751
10.126.4.22 GetCollision . . . 752
10.126.4.23 GetCollisionById . . . 752
10.126.4.24 GetCollisions . . . 752
10.126.4.25 GetEnabled . . . 753
10.126.4.26 GetGravityMode . . . 753

10.126.4.27 GetInertial 753
10.126.4.28 GetKinematic 753
10.126.4.29 GetLinearDamping 754
10.126.4.30 GetModel 754
10.126.4.31 GetParentJoints 754
10.126.4.32 GetParentJointsLinks 754
10.126.4.33 GetRelativeAngularAccel 754
10.126.4.34 GetRelativeAngularVel 755
10.126.4.35 GetRelativeForce 755
10.126.4.36 GetRelativeLinearAccel 755
10.126.4.37 GetRelativeLinearVel 755
10.126.4.38 GetRelativeTorque 756
10.126.4.39 GetSelfCollide 756
10.126.4.40 GetSensorCount 756
10.126.4.41 GetSensorName 756
10.126.4.42 GetWorldAngularAccel 757
10.126.4.43 GetWorldCoGLinearVel 757
10.126.4.44 GetWorldCoGPose 757
10.126.4.45 GetWorldEnergy 757
10.126.4.46 GetWorldEnergyKinetic 758
10.126.4.47 GetWorldEnergyPotential 758
10.126.4.48 GetWorldForce 758
10.126.4.49 GetWorldInertialPose 758
10.126.4.50 GetWorldInertiaMatrix 759
10.126.4.51 GetWorldLinearAccel 759
10.126.4.52 GetWorldLinearVel 759
10.126.4.53 GetWorldLinearVel 759
10.126.4.54 GetWorldLinearVel 760
10.126.4.55 GetWorldTorque 760
10.126.4.56 Init 760
10.126.4.57 Load 761

10.126.4.58 MoveFrame 761
10.126.4.59 OnPoseChange 761
10.126.4.60 ProcessMsg 762
10.126.4.61 RemoveChild 762
10.126.4.62 RemoveChildJoint 762
10.126.4.63 RemoveCollision 762
10.126.4.64 RemoveParentJoint 762
10.126.4.65 Reset 762
10.126.4.66 ResetPhysicsStates 763
10.126.4.67 SetAngularAccel 763
10.126.4.68 SetAngularDamping 763
10.126.4.69 SetAngularVel 763
10.126.4.70 SetAutoDisable 763
10.126.4.71 SetCollideMode 764
10.126.4.72 SetEnabled 764
10.126.4.73 SetForce 764
10.126.4.74 SetGravityMode 765
10.126.4.75 SetInertial 765
10.126.4.76 SetKinematic 765
10.126.4.77 SetLaserRetro 765
10.126.4.78 SetLinearAccel 765
10.126.4.79 SetLinearDamping 766
10.126.4.80 SetLinearVel 766
10.126.4.81 SetLinkStatic 766
10.126.4.82 SetPublishData 766
10.126.4.83 SetScale 767
10.126.4.84 SetSelected 767
10.126.4.85 SetSelfCollide 767
10.126.4.86 SetState 767
10.126.4.87 SetTorque 768
10.126.4.88 Update 768

10.126.4.8 UpdateMass . . . 768
10.126.4.9 UpdateParameters . . . 768
10.126.4.10 UpdateSurface . . . 768
10.126.5 Member Data Documentation . . . 769
10.126.5.1 angularAccel . . . 769
10.126.5.2 attachedModelsOffset . . . 769
10.126.5.3 cgVisuals . . . 769
10.126.5.4 inertial . . . 769
10.126.5.5 initialized . . . 769
10.126.5.6 linearAccel . . . 769
10.126.5.7 visuals . . . 769
10.127 gazebo::physics::LinkState Class Reference . . . 769
10.127.1 Detailed Description . . . 771
10.127.2 Constructor & Destructor Documentation . . . 772
10.127.2.1 LinkState . . . 772
10.127.2.2 LinkState . . . 772
10.127.2.3 LinkState . . . 772
10.127.2.4 LinkState . . . 772
10.127.2.5 ~LinkState . . . 773
10.127.3 Member Function Documentation . . . 773
10.127.3.1 FillSDF . . . 773
10.127.3.2 GetAcceleration . . . 773
10.127.3.3 GetCollisionState . . . 773
10.127.3.4 GetCollisionState . . . 774
10.127.3.5 GetCollisionStateCount . . . 774
10.127.3.6 GetCollisionStates . . . 774
10.127.3.7 GetPose . . . 775
10.127.3.8 GetVelocity . . . 775
10.127.3.9 GetWrench . . . 775
10.127.3.10 SetZero . . . 775
10.127.3.11 Load . . . 775

10.127.3.12 load . . . 776
10.127.3.13 operator+ . . . 776
10.127.3.14 operator- . . . 776
10.127.3.15 operator= . . . 776
10.127.3.16 SetRealTime . . . 777
10.127.3.17 SetSimTime . . . 777
10.127.3.18 SetWallTime . . . 777
10.127.4 Friends And Related Function Documentation . . . 778
10.127.4.1 operator<< . . . 778
10.128 gazebo::common::Logger Class Reference . . . 778
10.128.1 Detailed Description . . . 780
10.128.2 Member Enumeration Documentation . . . 780
10.128.2.1 LogType . . . 780
10.128.3 Constructor & Destructor Documentation . . . 780
10.128.3.1 Logger . . . 780
10.128.3.2 ~Logger . . . 781
10.128.4 Member Function Documentation . . . 781
10.128.4.1 operator() . . . 781
10.128.4.2 operator() . . . 781
10.128.5 Member Data Documentation . . . 781
10.128.5.1 color . . . 781
10.129 gazebo::util::LogPlay Class Reference . . . 781
10.129.1 Member Function Documentation . . . 783
10.129.1.1 GetChunk . . . 783
10.129.1.2 GetChunkCount . . . 783
10.129.1.3 GetEncoding . . . 783
10.129.1.4 GetGazeboVersion . . . 783
10.129.1.5 GetHeader . . . 784
10.129.1.6 GetLogVersion . . . 784
10.129.1.7 GetRandSeed . . . 784
10.129.1.8 IsOpen . . . 784

10.129.1.9 Open . . . 784
10.129.1.10 Step . . . 785
10.130 Logplay Class Reference . . . 785
10.130.1 Detailed Description . . . 785
10.131 gazebo::util::LogRecord Class Reference . . . 786
10.131.1 Detailed Description . . . 787
10.131.2 Member Function Documentation . . . 788
10.131.2.1 Add . . . 788
10.131.2.2 Fini . . . 788
10.131.2.3 GetBasePath . . . 788
10.131.2.4 GetBufferSize . . . 788
10.131.2.5 GetEncoding . . . 789
10.131.2.6 GetFilename . . . 789
10.131.2.7 GetFileSize . . . 789
10.131.2.8 GetFirstUpdate . . . 789
10.131.2.9 GetPaused . . . 790
10.131.2.10 GetRunning . . . 790
10.131.2.11 GetRunTime . . . 790
10.131.2.12 Init . . . 790
10.131.2.13 IsReadyToStart . . . 791
10.131.2.14 Notify . . . 791
10.131.2.15 Remove . . . 791
10.131.2.16 SetBasePath . . . 791
10.131.2.17 SetPaused . . . 791
10.131.2.18 Start . . . 792
10.131.2.19 Stop . . . 792
10.131.2.20 Write . . . 792
10.132 gazebo::Master Class Reference . . . 792
10.132.1 Detailed Description . . . 793
10.132.2 Constructor & Destructor Documentation . . . 793
10.132.2.1 Master . . . 793

10.132.2.2 ~Master . . . 793
10.132.3 Member Function Documentation . . . 794
10.132.3.1 Fini . . . 794
10.132.3.2 Init . . . 794
10.132.3.3 Run . . . 794
10.132.3.4 RunOnce . . . 794
10.132.3.5 RunThread . . . 794
10.132.3.6 Stop . . . 794
10.133 gazebo::common::Material Class Reference . . . 794
10.133.1 Detailed Description . . . 797
10.133.2 Member Enumeration Documentation . . . 798
10.133.2.1 BlendMode . . . 798
10.133.2.2 ShadeMode . . . 798
10.133.3 Constructor & Destructor Documentation . . . 798
10.133.3.1 Material . . . 798
10.133.3.2 ~Material . . . 798
10.133.3.3 Material . . . 798
10.133.4 Member Function Documentation . . . 799
10.133.4.1 GetAmbient . . . 799
10.133.4.2 GetBlendFactors . . . 799
10.133.4.3 GetBlendMode . . . 799
10.133.4.4 GetDepthWrite . . . 799
10.133.4.5 GetDiffuse . . . 799
10.133.4.6 GetEmissive . . . 800
10.133.4.7 GetLighting . . . 800
10.133.4.8 GetName . . . 800
10.133.4.9 GetPointSize . . . 800
10.133.4.10 GetShadeMode . . . 800
10.133.4.11 GetShininess . . . 801
10.133.4.12 GetSpecular . . . 801
10.133.4.13 GetTextureImage . . . 801

10.133.4.14 GetTransparency . . . 801
10.133.4.15 SetAmbient . . . 801
10.133.4.16 SetBlendFactors . . . 802
10.133.4.17 SetBlendMode . . . 802
10.133.4.18 SetDepthWrite . . . 802
10.133.4.19 SetDiffuse . . . 802
10.133.4.20 SetEmissive . . . 802
10.133.4.21 SetLighting . . . 803
10.133.4.22 SetPointSize . . . 803
10.133.4.23 SetShadeMode . . . 803
10.133.4.24 SetShininess . . . 803
10.133.4.25 SetSpecular . . . 803
10.133.4.26 SetTextureImage . . . 804
10.133.4.27 SetTextureImage . . . 804
10.133.4.28 SetTransparency . . . 804
10.133.5 Friends And Related Function Documentation . . . 804
10.133.5.1 operator<< . . . 804
10.133.6 Member Data Documentation . . . 804
10.133.6.1 ambient . . . 805
10.133.6.2 blendMode . . . 805
10.133.6.3 BlendModeStr . . . 805
10.133.6.4 diffuse . . . 805
10.133.6.5 emissive . . . 805
10.133.6.6 name . . . 805
10.133.6.7 pointSize . . . 805
10.133.6.8 shadeMode . . . 805
10.133.6.9 ShadeModeStr . . . 805
10.133.6.10 shininess . . . 805
10.133.6.11 specular . . . 806
10.133.6.12 texImage . . . 806
10.133.6.13 transparency . . . 806

10.134 gazebo::math::Matrix3 Class Reference 806
10.134.1 Detailed Description 807
10.134.2 Constructor & Destructor Documentation 808
10.134.2.1 Matrix3 808
10.134.2.2 Matrix3 808
10.134.2.3 Matrix3 808
10.134.2.4 ~Matrix3 808
10.134.3 Member Function Documentation 808
10.134.3.1 operator* 808
10.134.3.2 operator* 809
10.134.3.3 operator* 809
10.134.3.4 operator+ 809
10.134.3.5 operator- 809
10.134.3.6 operator== 809
10.134.3.7 operator[] 810
10.134.3.8 operator[] 810
10.134.3.9 SetCol 810
10.134.3.10 SetFromAxes 811
10.134.3.11 SetFromAxis 811
10.134.4 Friends And Related Function Documentation 811
10.134.4.1 operator* 811
10.134.4.2 operator<< 811
10.134.5 Member Data Documentation 812
10.134.5.1 IDENTITY 812
10.134.5.2 m 812
10.134.5.3 ZERO 812
10.135 gazebo::math::Matrix4 Class Reference 812
10.135.1 Detailed Description 814
10.135.2 Constructor & Destructor Documentation 814
10.135.2.1 Matrix4 814
10.135.2.2 Matrix4 814

10.135.2.3 Matrix4 . . . 815
10.135.2.4 ~Matrix4 . . . 815
10.135.3 Member Function Documentation . . . 815
10.135.3.1 GetAsPose . . . 815
10.135.3.2 GetEulerRotation . . . 816
10.135.3.3 GetRotation . . . 816
10.135.3.4 GetTranslation . . . 816
10.135.3.5 Inverse . . . 816
10.135.3.6 IsAffine . . . 816
10.135.3.7 operator* . . . 816
10.135.3.8 operator* . . . 817
10.135.3.9 operator* . . . 817
10.135.3.10 operator= . . . 817
10.135.3.11 operator= . . . 818
10.135.3.12 operator== . . . 818
10.135.3.13 operator[] . . . 818
10.135.3.14 operator[] . . . 819
10.135.3.15 Set . . . 819
10.135.3.16 SetScale . . . 819
10.135.3.17 SetTranslate . . . 820
10.135.3.18 TransformAffine . . . 820
10.135.4 Friends And Related Function Documentation . . . 820
10.135.4.1 operator<< . . . 820
10.135.5 Member Data Documentation . . . 820
10.135.5.1 IDENTITY . . . 821
10.135.5.2 m . . . 821
10.135.5.3 ZERO . . . 821
10.136 gazebo::common::Mesh Class Reference . . . 821
10.136.1 Detailed Description . . . 823
10.136.2 Constructor & Destructor Documentation . . . 823
10.136.2.1 Mesh . . . 823

10.136.2.2 ~Mesh . . . 823
10.136.3 Member Function Documentation . . . 823
10.136.3.1 AddMaterial . . . 823
10.136.3.2 AddSubMesh . . . 823
10.136.3.3 Center . . . 824
10.136.3.4 FillArrays . . . 824
10.136.3.5 GenSphericalTexCoord . . . 824
10.136.3.6 GetAABB . . . 824
10.136.3.7 GetIndexCount . . . 825
10.136.3.8 GetMaterial . . . 825
10.136.3.9 GetMaterialCount . . . 825
10.136.3.10 GetMaterialIndex . . . 825
10.136.3.11 GetMax . . . 826
10.136.3.12 GetMin . . . 826
10.136.3.13 GetName . . . 826
10.136.3.14 GetNormalCount . . . 826
10.136.3.15 GetPath . . . 826
10.136.3.16 GetSkeleton . . . 827
10.136.3.17 GetSubMesh . . . 827
10.136.3.18 GetSubMesh . . . 827
10.136.3.19 GetSubMeshCount . . . 827
10.136.3.20 GetTexCoordCount . . . 828
10.136.3.21 GetVertexCount . . . 828
10.136.3.22 HasSkeleton . . . 828
10.136.3.23 RecalculateNormals . . . 828
10.136.3.24 Scale . . . 828
10.136.3.25 SetName . . . 828
10.136.3.26 SetPath . . . 829
10.136.3.27 SetScale . . . 829
10.136.3.28 SetSkeleton . . . 829
10.136.3.29 Translate . . . 829

10.137 gazebo::common::MeshCSG Class Reference 829
10.137.1 Detailed Description 830
10.137.2 Member Enumeration Documentation 830
10.137.2.1 BooleanOperation 830
10.137.3 Constructor & Destructor Documentation 830
10.137.3.1 MeshCSG 830
10.137.3.2 ~MeshCSG 830
10.137.4 Member Function Documentation 831
10.137.4.1 CreateBoolean 831
10.138 gazebo::common::MeshExporter Class Reference 831
10.138.1 Detailed Description 832
10.138.2 Constructor & Destructor Documentation 832
10.138.2.1 MeshExporter 832
10.138.2.2 ~MeshExporter 832
10.138.3 Member Function Documentation 832
10.138.3.1 Export 832
10.139 gazebo::common::MeshLoader Class Reference 833
10.139.1 Detailed Description 833
10.139.2 Constructor & Destructor Documentation 833
10.139.2.1 MeshLoader 834
10.139.2.2 ~MeshLoader 834
10.139.3 Member Function Documentation 834
10.139.3.1 Load 834
10.140 gazebo::common::MeshManager Class Reference 834
10.140.1 Detailed Description 836
10.140.2 Member Function Documentation 836
10.140.2.1 AddMesh 836
10.140.2.2 CreateBox 836
10.140.2.3 CreateCamera 837
10.140.2.4 CreateCone 837
10.140.2.5 CreateCylinder 837

10.140.2.6 CreatePlane . . . 838
10.140.2.7 CreatePlane . . . 838
10.140.2.8 CreateSphere . . . 838
10.140.2.9 CreateTube . . . 839
10.140.2.10 Export . . . 839
10.140.2.11 GenSphericalTexCoord . . . 839
10.140.2.12 GetMesh . . . 839
10.140.2.13 GetMeshAABB . . . 840
10.140.2.14 HasMesh . . . 840
10.140.2.15 IsValidFilename . . . 840
10.140.2.16 Load . . . 841
10.141 gazebo::physics::MeshShape Class Reference . . . 841
10.141.1 Detailed Description . . . 843
10.141.2 Constructor & Destructor Documentation . . . 843
10.141.2.1 MeshShape . . . 843
10.141.2.2 ~MeshShape . . . 843
10.141.3 Member Function Documentation . . . 843
10.141.3.1 FillMsg . . . 843
10.141.3.2 GetMeshURI . . . 844
10.141.3.3 GetSize . . . 844
10.141.3.4 Init . . . 844
10.141.3.5 ProcessMsg . . . 844
10.141.3.6 SetMesh . . . 844
10.141.3.7 SetScale . . . 845
10.141.3.8 Update . . . 845
10.141.4 Member Data Documentation . . . 845
10.141.4.1 mesh . . . 845
10.141.4.2 submesh . . . 845
10.142 gazebo::physics::Model Class Reference . . . 846
10.142.1 Detailed Description . . . 850
10.142.2 Constructor & Destructor Documentation . . . 850

10.142.2.1 Model . . . 850
10.142.2.2 ~Model . . . 850
10.142.3 Member Function Documentation . . . 850
10.142.3.1 AttachStaticModel . . . 850
10.142.3.2 DetachStaticModel . . . 850
10.142.3.3 FillMsg . . . 851
10.142.3.4 Fini . . . 851
10.142.3.5 GetAutoDisable . . . 851
10.142.3.6 GetBoundingBox . . . 851
10.142.3.7 GetGripper . . . 852
10.142.3.8 GetGripperCount . . . 852
10.142.3.9 GetJoint . . . 852
10.142.3.10 GetJointController . . . 852
10.142.3.11 GetJointCount . . . 853
10.142.3.12 GetJoints . . . 853
10.142.3.13 GetLink . . . 853
10.142.3.14 GetLinkById . . . 853
10.142.3.15 GetLinks . . . 853
10.142.3.16 GetPluginCount . . . 854
10.142.3.17 GetRelativeAngularAccel . . . 854
10.142.3.18 GetRelativeAngularVel . . . 854
10.142.3.19 GetRelativeLinearAccel . . . 854
10.142.3.20 GetRelativeLinearVel . . . 855
10.142.3.21 GetSDF . . . 855
10.142.3.22 GetSensorCount . . . 855
10.142.3.23 GetWorldAngularAccel . . . 855
10.142.3.24 GetWorldAngularVel . . . 856
10.142.3.25 GetWorldEnergy . . . 856
10.142.3.26 GetWorldEnergyKinetic . . . 856
10.142.3.27 GetWorldEnergyPotential . . . 856
10.142.3.28 GetWorldLinearAccel . . . 856

10.142.3.29 SetWorldLinearVel . . . 857
10.142.3.30 Init . . . 857
10.142.3.31 Load . . . 857
10.142.3.32 LoadJoints . . . 857
10.142.3.33 LoadPlugins . . . 858
10.142.3.34 OnPoseChange . . . 858
10.142.3.35 ProcessMsg . . . 858
10.142.3.36 RemoveChild . . . 858
10.142.3.37 Reset . . . 858
10.142.3.38 SetAngularAccel . . . 858
10.142.3.39 SetAngularVel . . . 859
10.142.3.40 SetAutoDisable . . . 859
10.142.3.41 SetCollideMode . . . 859
10.142.3.42 SetEnabled . . . 859
10.142.3.43 SetGravityMode . . . 860
10.142.3.44 SetJointAnimation . . . 860
10.142.3.45 SetJointPosition . . . 860
10.142.3.46 SetJointPositions . . . 860
10.142.3.47 SetLaserRetro . . . 861
10.142.3.48 SetLinearAccel . . . 861
10.142.3.49 SetLinearVel . . . 861
10.142.3.50 SetLinkWorldPose . . . 861
10.142.3.51 SetLinkWorldPose . . . 862
10.142.3.52 SetScale . . . 862
10.142.3.53 SetState . . . 862
10.142.3.54 StopAnimation . . . 862
10.142.3.55 Update . . . 863
10.142.3.56 UpdateParameters . . . 863
10.142.4 Member Data Documentation . . . 863
10.142.4.1 attachedModels . . . 863
10.142.4.2 attachedModelsOffset . . . 863

10.142.4.3 jointPub 863
10.143 gazebo::common::ModelDatabase Class Reference 864
10.143.1 Detailed Description 865
10.144 gazebo::common::ModelDatabasePrivate Class Reference 865
10.144.1 Detailed Description 866
10.144.2 Member Typedef Documentation 866
10.144.2.1 CallbackFunc 866
10.144.3 Member Data Documentation 866
10.144.3.1 callbacksMutex 866
10.144.3.2 modelCache 866
10.144.3.3 modelDBUpdated 866
10.144.3.4 startCacheMutex 866
10.144.3.5 stop 867
10.144.3.6 updateCacheCompleteCondition 867
10.144.3.7 updateCacheCondition 867
10.144.3.8 updateCacheThread 867
10.144.3.9 updateMutex 867
10.145 gazebo::ModelPlugin Class Reference 867
10.145.1 Detailed Description 868
10.145.2 Constructor & Destructor Documentation 868
10.145.2.1 ModelPlugin 868
10.145.2.2 ~ModelPlugin 869
10.145.3 Member Function Documentation 869
10.145.3.1 Init 869
10.145.3.2 Load 869
10.145.3.3 Reset 869
10.146 gazebo::physics::ModelState Class Reference 869
10.146.1 Detailed Description 872
10.146.2 Constructor & Destructor Documentation 872
10.146.2.1 ModelState 872
10.146.2.2 ModelState 872

10.146.2.3 ModelState . . . 872
10.146.2.4 ModelState . . . 872
10.146.2.5 ~ModelState . . . 873
10.146.3 Member Function Documentation . . . 873
10.146.3.1 FillSDF . . . 873
10.146.3.2 GetJointState . . . 873
10.146.3.3 GetJointState . . . 874
10.146.3.4 GetJointStateCount . . . 874
10.146.3.5 GetJointStates . . . 874
10.146.3.6 GetJointStates . . . 875
10.146.3.7 GetLinkState . . . 875
10.146.3.8 GetLinkStateCount . . . 875
10.146.3.9 GetLinkStates . . . 876
10.146.3.10 GetLinkStates . . . 876
10.146.3.11 GetPose . . . 876
10.146.3.12 HasJointState . . . 876
10.146.3.13 HasLinkState . . . 877
10.146.3.14 IsZero . . . 877
10.146.3.15 Load . . . 877
10.146.3.16 Load . . . 877
10.146.3.17 operator+ . . . 878
10.146.3.18 operator- . . . 878
10.146.3.19 operator= . . . 878
10.146.3.20 SetRealTime . . . 879
10.146.3.21 SetSimTime . . . 879
10.146.3.22 SetWallTime . . . 879
10.146.4 Friends And Related Function Documentation . . . 879
10.146.4.1 operator<< . . . 880
10.147 gazebo::common::MouseEvent Class Reference . . . 880
10.147.1 Detailed Description . . . 881
10.147.2 Member Enumeration Documentation . . . 881

10.147.2.1 Buttons . . . 881
10.147.2.2 EventType . . . 882
10.147.3 Constructor & Destructor Documentation . . . 882
10.147.3.1 MouseEvent . . . 882
10.147.4 Member Data Documentation . . . 882
10.147.4.1 alt . . . 882
10.147.4.2 button . . . 882
10.147.4.3 buttons . . . 882
10.147.4.4 control . . . 882
10.147.4.5 dragging . . . 882
10.147.4.6 moveScale . . . 883
10.147.4.7 pos . . . 883
10.147.4.8 pressPos . . . 883
10.147.4.9 prevPos . . . 883
10.147.4.10 scroll . . . 883
10.147.4.11 shift . . . 883
10.147.4.12 type . . . 883
10.148 gazebo::rendering::MovableText Class Reference . . . 883
10.148.1 Detailed Description . . . 885
10.148.2 Member Enumeration Documentation . . . 885
10.148.2.1 HorizAlign . . . 885
10.148.2.2 VertAlign . . . 886
10.148.3 Constructor & Destructor Documentation . . . 886
10.148.3.1 MovableText . . . 886
10.148.3.2 ~MovableText . . . 886
10.148.4 Member Function Documentation . . . 886
10.148.4.1 _setupGeometry . . . 886
10.148.4.2 _updateColors . . . 886
10.148.4.3 GetAABB . . . 886
10.148.4.4 GetBaseline . . . 886
10.148.4.5 getBoundingRadius . . . 887

10.148.4.6 GetCharHeight . . . 887
10.148.4.7 GetColor . . . 887
10.148.4.8 GetFont . . . 887
10.148.4.9 getLights . . . 887
10.148.4.10 getMaterial . . . 887
10.148.4.11 getRenderOperation . . . 887
10.148.4.12 GetShowOnTop . . . 887
10.148.4.13 GetSpaceWidth . . . 888
10.148.4.14 getSquaredViewDepth . . . 888
10.148.4.15 GetText . . . 888
10.148.4.16 getWorldTransforms . . . 888
10.148.4.17 Load . . . 888
10.148.4.18 SetBaseline . . . 888
10.148.4.19 SetCharHeight . . . 889
10.148.4.20 SetColor . . . 889
10.148.4.21 SetFontName . . . 889
10.148.4.22 SetShowOnTop . . . 889
10.148.4.23 SetSpaceWidth . . . 889
10.148.4.24 SetText . . . 890
10.148.4.25 SetTextAlignment . . . 890
10.148.4.26 Update . . . 890
10.148.4.27 VisitRenderables . . . 890
10.149 gazebo::common::MovingWindowFilter< T > Class Template Reference 890
10.149.1 Detailed Description . . . 891
10.149.2 Constructor & Destructor Documentation . . . 891
10.149.2.1 MovingWindowFilter . . . 891
10.149.3 Member Data Documentation . . . 892
10.149.3.1 dataPtr . . . 892
10.150 gazebo::common::MovingWindowFilterPrivate< T > Class Template Reference . . . 892
10.150.1 Member Data Documentation . . . 893

10.150.1.1samples . . . 893
10.150.1.2sum . . . 893
10.150.1.3valHistory . . . 893
10.150.1.4valIter . . . 893
10.150.1.5valWindowSize . . . 893
10.151gazebo::msgs::MsgFactory Class Reference . . . 893
10.151.1Detailed Description . . . 894
10.151.2Member Function Documentation . . . 894
10.151.2.1GetMsgTypes . . . 894
10.151.2.2NewMsg . . . 894
10.151.2.3RegisterMsg . . . 895
10.152gazebo::sensors::MultiCameraSensor Class Reference . . . 895
10.152.1Detailed Description . . . 896
10.152.2Constructor & Destructor Documentation . . . 896
10.152.2.1MultiCameraSensor . . . 896
10.152.2.2∼MultiCameraSensor . . . 897
10.152.3Member Function Documentation . . . 897
10.152.3.1Fini . . . 897
10.152.3.2GetCamera . . . 897
10.152.3.3GetCameraCount . . . 897
10.152.3.4GetImageData . . . 898
10.152.3.5GetImageHeight . . . 898
10.152.3.6GetImageWidth . . . 898
10.152.3.7GetTopic . . . 899
10.152.3.8Init . . . 899
10.152.3.9IsActive . . . 899
10.152.3.10Load . . . 899
10.152.3.11SaveFrame . . . 900
10.152.3.12UpdateImpl . . . 900
10.153gazebo::physics::MultiRayShape Class Reference . . . 901
10.153.1Detailed Description . . . 903

10.153.2 Constructor & Destructor Documentation 903
10.153.2.1 MultiRayShape 903
10.153.2.2 ~MultiRayShape 903
10.153.3 Member Function Documentation 904
10.153.3.1 AddRay 904
10.153.3.2 ConnectNewLaserScans 904
10.153.3.3 DisconnectNewLaserScans 904
10.153.3.4 FillMsg 905
10.153.3.5 GetFiducial 905
10.153.3.6 GetMaxAngle 905
10.153.3.7 GetMaxRange 905
10.153.3.8 GetMinAngle 906
10.153.3.9 GetMinRange 906
10.153.3.10 GetRange 906
10.153.3.11 GetResRange 906
10.153.3.12 GetRetro 906
10.153.3.13 GetSampleCount 907
10.153.3.14 GetScanResolution 907
10.153.3.15 GetVerticalMaxAngle 907
10.153.3.16 GetVerticalMinAngle 907
10.153.3.17 GetVerticalSampleCount 908
10.153.3.18 GetVerticalScanResolution 908
10.153.3.19 Init 908
10.153.3.20 ProcessMsg 908
10.153.3.21 SetScale 908
10.153.3.22 Update 909
10.153.3.23 UpdateRays 909
10.153.4 Member Data Documentation 909
10.153.4.1 horzElem 909
10.153.4.2 newLaserScans 909
10.153.4.3 offset 909

10.153.4.4 rangeElem . . . 909
10.153.4.5 rayElem . . . 910
10.153.4.6 rays . . . 910
10.153.4.7 scanElem . . . 910
10.153.4.8 vertElem . . . 910
10.154 gazebo::transport::Node Class Reference . . . 910
10.154.1 Detailed Description . . . 912
10.154.2 Constructor & Destructor Documentation . . . 912
10.154.2.1 Node . . . 912
10.154.2.2 ~Node . . . 912
10.154.3 Member Function Documentation . . . 912
10.154.3.1 Advertise . . . 912
10.154.3.2 DecodeTopicName . . . 913
10.154.3.3 EncodeTopicName . . . 913
10.154.3.4 Fini . . . 913
10.154.3.5 GetId . . . 913
10.154.3.6 GetMsgType . . . 914
10.154.3.7 GetTopicNamespace . . . 914
10.154.3.8 HandleData . . . 914
10.154.3.9 HandleMessage . . . 914
10.154.3.10 HasLatchedSubscriber . . . 915
10.154.3.11 Init . . . 915
10.154.3.12 InsertLatchedMsg . . . 915
10.154.3.13 InsertLatchedMsg . . . 916
10.154.3.14 ProcessIncoming . . . 916
10.154.3.15 ProcessPublishers . . . 916
10.154.3.16 Publish . . . 916
10.154.3.17 RemoveCallback . . . 916
10.154.3.18 Subscribe . . . 917
10.154.3.19 Subscribe . . . 917
10.154.3.20 Subscribe . . . 917

10.154.3.23 Subscribe 918
10.155 Gazebo::common::NodeAnimation Class Reference 918
10.155.1 Detailed Description 920
10.155.2 Constructor & Destructor Documentation 920
10.155.2.1 NodeAnimation 920
10.155.2.2 ~NodeAnimation 920
10.155.3 Member Function Documentation 920
10.155.3.1 AddKeyFrame 920
10.155.3.2 AddKeyFrame 920
10.155.3.3 GetFrameAt 921
10.155.3.4 GetFrameCount 921
10.155.3.5 GetKeyFrame 921
10.155.3.6 GetKeyFrame 921
10.155.3.7 GetLength 922
10.155.3.8 GetName 922
10.155.3.9 GetTimeAtX 922
10.155.3.10 Scale 922
10.155.3.11 SetName 923
10.155.4 Member Data Documentation 923
10.155.4.1 keyFrames 923
10.155.4.2 length 923
10.155.4.3 name 923
10.156 Gazebo::common::NodeAssignment Class Reference 923
10.156.1 Detailed Description 924
10.156.2 Constructor & Destructor Documentation 924
10.156.2.1 NodeAssignment 924
10.156.3 Member Data Documentation 924
10.156.3.1 nodeIndex 924
10.156.3.2 vertexIndex 924
10.156.3.3 weight 924
10.157 Gazebo::common::NodeTransform Class Reference 925

10.157.1 Detailed Description . . . 926
10.157.2 Member Enumeration Documentation . . . 926
10.157.2.1 TransformType . . . 926
10.157.3 Constructor & Destructor Documentation . . . 927
10.157.3.1 NodeTransform . . . 927
10.157.3.2 NodeTransform . . . 927
10.157.3.3 ~NodeTransform . . . 927
10.157.4 Member Function Documentation . . . 927
10.157.4.1 Get . . . 927
10.157.4.2 GetSID . . . 927
10.157.4.3 GetType . . . 928
10.157.4.4 operator() . . . 928
10.157.4.5 operator* . . . 928
10.157.4.6 operator* . . . 928
10.157.4.7 PrintSource . . . 929
10.157.4.8 RecalculateMatrix . . . 929
10.157.4.9 Set . . . 929
10.157.4.10 SetComponent . . . 929
10.157.4.11 SetSID . . . 929
10.157.4.12 SetSourceValues . . . 930
10.157.4.13 SetSourceValues . . . 930
10.157.4.14 SetSourceValues . . . 930
10.157.4.15 SetType . . . 930
10.157.5 Member Data Documentation . . . 930
10.157.5.1 sid . . . 930
10.157.5.2 source . . . 930
10.157.5.3 transform . . . 931
10.157.5.4 type . . . 931
10.158 gazebo::sensors::Noise Class Reference . . . 931
10.158.1 Detailed Description . . . 932
10.158.2 Member Enumeration Documentation . . . 932

10.158.2.1 NoiseType . . . 932
10.158.3 Constructor & Destructor Documentation . . . 933
10.158.3.1 Noise . . . 933
10.158.3.2 ~Noise . . . 933
10.158.4 Member Function Documentation . . . 933
10.158.4.1 Apply . . . 933
10.158.4.2 ApplyImpl . . . 933
10.158.4.3 Fini . . . 934
10.158.4.4 GetNoiseType . . . 934
10.158.4.5 Load . . . 934
10.158.4.6 SetCamera . . . 934
10.158.4.7 SetCustomNoiseCallback . . . 935
10.159 gazebo::sensors::NoiseFactory Class Reference . . . 935
10.159.1 Detailed Description . . . 935
10.159.2 Member Function Documentation . . . 935
10.159.2.1 NewNoiseModel . . . 936
10.160 gazebo::common::NumericAnimation Class Reference . . . 936
10.160.1 Detailed Description . . . 937
10.160.2 Constructor & Destructor Documentation . . . 937
10.160.2.1 NumericAnimation . . . 937
10.160.2.2 ~NumericAnimation . . . 937
10.160.3 Member Function Documentation . . . 937
10.160.3.1 CreateKeyFrame . . . 937
10.160.3.2 GetInterpolatedKeyFrame . . . 938
10.161 gazebo::common::NumericKeyFrame Class Reference . . . 938
10.161.1 Detailed Description . . . 939
10.161.2 Constructor & Destructor Documentation . . . 939
10.161.2.1 NumericKeyFrame . . . 939
10.161.2.2 ~NumericKeyFrame . . . 939
10.161.3 Member Function Documentation . . . 939
10.161.3.1 GetValue . . . 940

10.161.3.2 SetValue . . . 940
10.161.4 Member Data Documentation . . . 940
10.161.4.1 value . . . 940
10.162 gazebo::rendering::OculusCamera Class Reference . . . 940
10.162.1 Detailed Description . . . 942
10.162.2 Constructor & Destructor Documentation . . . 943
10.162.2.1 OculusCamera . . . 943
10.162.2.2 ~OculusCamera . . . 943
10.162.3 Member Function Documentation . . . 943
10.162.3.1 AdjustAspect . . . 943
10.162.3.2 AttachToVisualImpl . . . 943
10.162.3.3 Fini . . . 944
10.162.3.4 GetAvgFPS . . . 944
10.162.3.5 GetImageHeight . . . 944
10.162.3.6 GetImageWidth . . . 944
10.162.3.7 GetTriangleCount . . . 945
10.162.3.8 Init . . . 945
10.162.3.9 Load . . . 945
10.162.3.10 Load . . . 945
10.162.3.11 MoveToPosition . . . 945
10.162.3.12 MoveToVisual . . . 946
10.162.3.13 MoveToVisual . . . 946
10.162.3.14 PostRender . . . 946
10.162.3.15 Ready . . . 946
10.162.3.16 ResetSensor . . . 947
10.162.3.17 Resize . . . 947
10.162.3.18 SetRenderTarget . . . 947
10.162.3.19 TrackVisualImpl . . . 947
10.162.3.20 Update . . . 948
10.162.4 Member Data Documentation . . . 948
10.162.4.1 rightCamera . . . 948

10.162.4.2 rightViewport 948
10.163 gazebo::math::OnePole< T > Class Template Reference 948
10.163.1 Detailed Description 949
10.163.2 Constructor & Destructor Documentation 950
10.163.2.1 OnePole 950
10.163.2.2 OnePole 950
10.163.3 Member Function Documentation 950
10.163.3.1 Process 950
10.163.3.2 SetFc 950
10.163.4 Member Data Documentation 951
10.163.4.1 a0 951
10.163.4.2 b1 951
10.164 gazebo::math::OnePoleQuaternion Class Reference 951
10.164.1 Detailed Description 952
10.164.2 Constructor & Destructor Documentation 952
10.164.2.1 OnePoleQuaternion 952
10.164.2.2 OnePoleQuaternion 953
10.164.3 Member Function Documentation 953
10.164.3.1 Process 953
10.165 gazebo::math::OnePoleVector3 Class Reference 953
10.165.1 Detailed Description 954
10.165.2 Constructor & Destructor Documentation 954
10.165.2.1 OnePoleVector3 954
10.165.2.2 OnePoleVector3 955
10.166 gazebo::rendering::OrbitViewController Class Reference 955
10.166.1 Detailed Description 956
10.166.2 Constructor & Destructor Documentation 956
10.166.2.1 OrbitViewController 956
10.166.2.2 ~OrbitViewController 957
10.166.3 Member Function Documentation 957
10.166.3.1 GetFocalPoint 957

10.166.3.2 GetTypeString . . . 957
10.166.3.3 HandleKeyPressEvent . . . 957
10.166.3.4 HandleKeyReleaseEvent . . . 957
10.166.3.5 HandleMouseEvent . . . 958
10.166.3.6 Init . . . 958
10.166.3.7 Init . . . 958
10.166.3.8 SetDistance . . . 958
10.166.3.9 SetFocalPoint . . . 959
10.166.3.10 Update . . . 959
10.167 gazebo::physics::PhysicsEngine Class Reference . . . 959
10.167.1 Detailed Description . . . 963
10.167.2 Constructor & Destructor Documentation . . . 963
10.167.2.1 PhysicsEngine . . . 963
10.167.2.2 ~PhysicsEngine . . . 963
10.167.3 Member Function Documentation . . . 963
10.167.3.1 CreateCollision . . . 963
10.167.3.2 CreateCollision . . . 963
10.167.3.3 CreateJoint . . . 964
10.167.3.4 CreateLink . . . 964
10.167.3.5 CreateModel . . . 964
10.167.3.6 CreateShape . . . 965
10.167.3.7 DebugPrint . . . 965
10.167.3.8 Fini . . . 965
10.167.3.9 GetAutoDisableFlag . . . 965
10.167.3.10 GetContactManager . . . 965
10.167.3.11 GetContactMaxCorrectingVel . . . 966
10.167.3.12 GetContactSurfaceLayer . . . 966
10.167.3.13 GetGravity . . . 966
10.167.3.14 GetMaxContacts . . . 966
10.167.3.15 GetMaxStepSize . . . 967
10.167.3.16 GetParam . . . 967

10.167.3.17GetPhysicsUpdateMutex 967
10.167.3.18GetRealTimeUpdateRate 967
10.167.3.19GetTargetRealTimeFactor 968
10.167.3.20GetType 968
10.167.3.21GetUpdatePeriod 968
10.167.3.22GetWorldCFM 968
10.167.3.23GetWorldERP 969
10.167.3.24Init 969
10.167.3.25InitForThread 969
10.167.3.26Load 969
10.167.3.27OnPhysicsMsg 969
10.167.3.28OnRequest 970
10.167.3.29Reset 970
10.167.3.30SetAutoDisableFlag 970
10.167.3.31SetContactMaxCorrectingVel 970
10.167.3.32SetContactSurfaceLayer 971
10.167.3.33SetGravity 971
10.167.3.34SetMaxContacts 971
10.167.3.35SetMaxStepSize 971
10.167.3.36SetParam 972
10.167.3.37SetRealTimeUpdateRate 973
10.167.3.38SetSeed 973
10.167.3.39SetTargetRealTimeFactor 973
10.167.3.40SetWorldCFM 973
10.167.3.41SetWorldERP 974
10.167.3.42UpdateCollision 974
10.167.3.43UpdatePhysics 974
10.167.4Member Data Documentation 974
10.167.4.1contactManager 974
10.167.4.2maxStepSize 974
10.167.4.3node 975

10.167.4.4 physicsSub . . . 975
10.167.4.5 physicsUpdateMutex . . . 975
10.167.4.6 realTimeUpdateRate . . . 975
10.167.4.7 requestSub . . . 975
10.167.4.8 responsePub . . . 975
10.167.4.9 sdf . . . 975
10.167.4.10 targetRealTimeFactor . . . 975
10.167.4.11 world . . . 976
10.168 gazebo::physics::PhysicsFactory Class Reference . . . 976
10.168.1 Detailed Description . . . 976
10.168.2 Member Function Documentation . . . 976
10.168.2.1 IsRegistered . . . 976
10.168.2.2 NewPhysicsEngine . . . 977
10.168.2.3 RegisterAll . . . 977
10.168.2.4 RegisterPhysicsEngine . . . 977
10.169 gazebo::common::PID Class Reference . . . 977
10.169.1 Detailed Description . . . 979
10.169.2 Constructor & Destructor Documentation . . . 979
10.169.2.1 PID . . . 979
10.169.2.2 ~PID . . . 979
10.169.3 Member Function Documentation . . . 980
10.169.3.1 GetCmd . . . 980
10.169.3.2 GetCmdMax . . . 980
10.169.3.3 GetCmdMin . . . 980
10.169.3.4 GetDGain . . . 980
10.169.3.5 GetErrors . . . 980
10.169.3.6 GetIGain . . . 981
10.169.3.7 GetIMax . . . 981
10.169.3.8 GetIMin . . . 981
10.169.3.9 GetPGain . . . 981
10.169.3.10 Init . . . 981

10.169.3.10 operator= . . . 982
10.169.3.11 Reset . . . 982
10.169.3.12 SetCmd . . . 982
10.169.3.13 SetCmdMax . . . 982
10.169.3.14 SetCmdMin . . . 983
10.169.3.15 SetDGain . . . 983
10.169.3.16 SetIGain . . . 983
10.169.3.17 SetIMax . . . 983
10.169.3.18 SetIMin . . . 983
10.169.3.19 SetPGain . . . 984
10.169.3.20 Update . . . 984
10.170 gazebo::math::Plane Class Reference . . . 984
10.170.1 Detailed Description . . . 985
10.170.2 Constructor & Destructor Documentation . . . 985
10.170.2.1 Plane . . . 985
10.170.2.2 Plane . . . 985
10.170.2.3 Plane . . . 986
10.170.2.4 ~Plane . . . 986
10.170.3 Member Function Documentation . . . 986
10.170.3.1 Distance . . . 986
10.170.3.2 operator= . . . 986
10.170.3.3 Set . . . 987
10.170.4 Member Data Documentation . . . 987
10.170.4.1 d . . . 987
10.170.4.2 normal . . . 987
10.170.4.3 size . . . 987
10.171 gazebo::physics::PlaneShape Class Reference . . . 987
10.171.1 Detailed Description . . . 989
10.171.2 Constructor & Destructor Documentation . . . 989
10.171.2.1 PlaneShape . . . 989
10.171.2.2 ~PlaneShape . . . 989

10.171.3 Member Function Documentation . . . 989
10.171.3.1 CreatePlane . . . 989
10.171.3.2 FillMsg . . . 990
10.171.3.3 GetNormal . . . 990
10.171.3.4 GetSize . . . 990
10.171.3.5 Init . . . 990
10.171.3.6 ProcessMsg . . . 990
10.171.3.7 SetAltitude . . . 991
10.171.3.8 SetNormal . . . 991
10.171.3.9 SetScale . . . 991
10.171.3.10 SetSize . . . 991
10.172 gazebo::PluginT< T > Class Template Reference . . . 992
10.172.1 Detailed Description . . . 993
10.172.2 Member Typedef Documentation . . . 993
10.172.2.1 TPtr . . . 993
10.172.3 Constructor & Destructor Documentation . . . 993
10.172.3.1 PluginT . . . 993
10.172.3.2 ~PluginT . . . 993
10.172.4 Member Function Documentation . . . 993
10.172.4.1 Create . . . 993
10.172.4.2 GetFilename . . . 994
10.172.4.3 GetHandle . . . 994
10.172.4.4 GetType . . . 994
10.172.5 Member Data Documentation . . . 994
10.172.5.1 filename . . . 994
10.172.5.2 handle . . . 994
10.172.5.3 type . . . 995
10.173 gazebo::math::Pose Class Reference . . . 995
10.173.1 Detailed Description . . . 997
10.173.2 Constructor & Destructor Documentation . . . 997
10.173.2.1 Pose . . . 997

10.173.2.2 Pose . . . 997
10.173.2.3 Pose . . . 998
10.173.2.4 Pose . . . 998
10.173.2.5 ~Pose . . . 998
10.173.3 Member Function Documentation . . . 998
10.173.3.1 CoordPoseSolve . . . 998
10.173.3.2 CoordPositionAdd . . . 998
10.173.3.3 CoordPositionAdd . . . 999
10.173.3.4 CoordPositionSub . . . 999
10.173.3.5 CoordRotationAdd . . . 999
10.173.3.6 CoordRotationSub . . . 1000
10.173.3.7 Correct . . . 1000
10.173.3.8 GetInverse . . . 1000
10.173.3.9 IsFinite . . . 1000
10.173.3.10 operator!= . . . 1001
10.173.3.11 operator* . . . 1001
10.173.3.12 operator+ . . . 1001
10.173.3.13 operator+= . . . 1001
10.173.3.14 operator- . . . 1002
10.173.3.15 operator- . . . 1002
10.173.3.16 operator-= . . . 1002
10.173.3.17 operator= . . . 1003
10.173.3.18 operator== . . . 1003
10.173.3.19 Reset . . . 1003
10.173.3.20 RotatePositionAboutOrigin . . . 1003
10.173.3.21 Round . . . 1004
10.173.3.22 Set . . . 1004
10.173.3.23 Set . . . 1004
10.173.3.24 Set . . . 1004
10.173.4 Friends And Related Function Documentation . . . 1005
10.173.4.1 operator<< . . . 1005

10.173.4.2 operator>> . . . 1005
10.173.5 Member Data Documentation . . . 1005
10.173.5.1 pos . . . 1005
10.173.5.2 rot . . . 1005
10.173.5.3 Zero . . . 1006
10.174 gazebo::common::PoseAnimation Class Reference . . . 1006
10.174.1 Detailed Description . . . 1007
10.174.2 Constructor & Destructor Documentation . . . 1007
10.174.2.1 PoseAnimation . . . 1007
10.174.2.2 ~PoseAnimation . . . 1007
10.174.3 Member Function Documentation . . . 1007
10.174.3.1 BuildInterpolationSplines . . . 1007
10.174.3.2 CreateKeyFrame . . . 1007
10.174.3.3 GetInterpolatedKeyFrame . . . 1008
10.174.3.4 GetInterpolatedKeyFrame . . . 1008
10.175 gazebo::common::PoseKeyFrame Class Reference . . . 1008
10.175.1 Detailed Description . . . 1009
10.175.2 Constructor & Destructor Documentation . . . 1010
10.175.2.1 PoseKeyFrame . . . 1010
10.175.2.2 ~PoseKeyFrame . . . 1010
10.175.3 Member Function Documentation . . . 1010
10.175.3.1 GetRotation . . . 1010
10.175.3.2 GetTranslation . . . 1010
10.175.3.3 SetRotation . . . 1010
10.175.3.4 SetTranslation . . . 1011
10.175.4 Member Data Documentation . . . 1011
10.175.4.1 rotate . . . 1011
10.175.4.2 translate . . . 1011
10.176 gazebo::rendering::Projector Class Reference . . . 1011
10.176.1 Detailed Description . . . 1012
10.176.2 Constructor & Destructor Documentation . . . 1012

10.176.2.1 Projector 1012
10.176.2.2 ~Projector 1012
10.176.3 Member Function Documentation 1013
10.176.3.1 GetParent 1013
10.176.3.2 Load 1013
10.176.3.3 Load 1013
10.176.3.4 Load 1013
10.176.3.5 SetEnabled 1014
10.176.3.6 SetTexture 1014
10.176.3.7 Toggle 1014
10.177 gazebo::transport::Publication Class Reference 1014
10.177.1 Detailed Description 1016
10.177.2 Constructor & Destructor Documentation 1016
10.177.2.1 Publication 1016
10.177.2.2 ~Publication 1016
10.177.3 Member Function Documentation 1016
10.177.3.1 AddPublisher 1016
10.177.3.2 AddSubscription 1016
10.177.3.3 AddSubscription 1017
10.177.3.4 AddTransport 1017
10.177.3.5 GetCallbackCount 1017
10.177.3.6 GetLocallyAdvertised 1017
10.177.3.7 GetMsgType 1017
10.177.3.8 GetNodeCount 1018
10.177.3.9 GetPrevMsg 1018
10.177.3.10 GetRemoteSubscriptionCount 1018
10.177.3.11 GetTransportCount 1018
10.177.3.12 HasTransport 1019
10.177.3.13 LocalPublish 1019
10.177.3.14 Publish 1019
10.177.3.15 RemovePublisher 1019

10.177.3.16 RemoveSubscription 1020
10.177.3.17 RemoveSubscription 1020
10.177.3.18 RemoveTransport 1020
10.177.3.19 SetLocallyAdvertised 1020
10.177.3.20 SetPrevMsg 1021
10.178 gazebo::transport::PublicationTransport Class Reference 1021
10.178.1 Detailed Description 1022
10.178.2 Constructor & Destructor Documentation 1022
10.178.2.1 PublicationTransport 1022
10.178.2.2 ~PublicationTransport 1022
10.178.3 Member Function Documentation 1022
10.178.3.1 AddCallback 1022
10.178.3.2 Fini 1022
10.178.3.3 GetConnection 1023
10.178.3.4 GetMsgType 1023
10.178.3.5 GetTopic 1023
10.178.3.6 Init 1023
10.179 gazebo::transport::Publisher Class Reference 1024
10.179.1 Detailed Description 1025
10.179.2 Constructor & Destructor Documentation 1025
10.179.2.1 Publisher 1025
10.179.2.2 ~Publisher 1025
10.179.3 Member Function Documentation 1025
10.179.3.1 Fini 1025
10.179.3.2 GetMsgType 1025
10.179.3.3 GetOutgoingCount 1025
10.179.3.4 GetPrevMsg 1026
10.179.3.5 GetPrevMsgPtr 1026
10.179.3.6 GetTopic 1026
10.179.3.7 HasConnections 1026
10.179.3.8 Publish 1026

10.179.3.9 Publish 1027
10.179.3.10 SendMessage 1027
10.179.3.11 SetNode 1027
10.179.3.12 SetPublication 1027
10.179.3.13 WaitForConnection 1027
10.179.3.14 WaitForConnection 1028
10.180 gazebo::transport::PublishTask Class Reference 1028
10.180.1 Detailed Description 1028
10.180.2 Constructor & Destructor Documentation 1028
10.180.2.1 PublishTask 1028
10.180.3 Member Function Documentation 1029
10.180.3.1 execute 1029
10.181 gazebo::math::Quaternion Class Reference 1029
10.181.1 Detailed Description 1032
10.181.2 Constructor & Destructor Documentation 1033
10.181.2.1 Quaternion 1033
10.181.2.2 Quaternion 1033
10.181.2.3 Quaternion 1033
10.181.2.4 Quaternion 1033
10.181.2.5 Quaternion 1034
10.181.2.6 Quaternion 1034
10.181.2.7 ~Quaternion 1034
10.181.3 Member Function Documentation 1034
10.181.3.1 Correct 1034
10.181.3.2 Dot 1034
10.181.3.3 EulerToQuaternion 1035
10.181.3.4 EulerToQuaternion 1035
10.181.3.5 GetAsAxis 1035
10.181.3.6 GetAsEuler 1035
10.181.3.7 GetAsMatrix3 1035
10.181.3.8 GetAsMatrix4 1036

10.181.3.9 GetExp . . . 1036
10.181.3.10 GetInverse . . . 1036
10.181.3.11 GetLog . . . 1036
10.181.3.12 GetPitch . . . 1036
10.181.3.13 GetRoll . . . 1037
10.181.3.14 GetXAxis . . . 1037
10.181.3.15 GetYaw . . . 1037
10.181.3.16 GetYAxis . . . 1037
10.181.3.17 GetZAxis . . . 1037
10.181.3.18 Invert . . . 1038
10.181.3.19 IsFinite . . . 1038
10.181.3.20 Normalize . . . 1038
10.181.3.21 operator!= . . . 1038
10.181.3.22 operator* . . . 1038
10.181.3.23 operator* . . . 1039
10.181.3.24 operator* . . . 1039
10.181.3.25 operator*= . . . 1039
10.181.3.26 operator+ . . . 1039
10.181.3.27 operator+= . . . 1040
10.181.3.28 operator- . . . 1040
10.181.3.29 operator- . . . 1040
10.181.3.30 operator-= . . . 1041
10.181.3.31 operator= . . . 1041
10.181.3.32 operator== . . . 1041
10.181.3.33 RotateVector . . . 1041
10.181.3.34 RotateVectorReverse . . . 1042
10.181.3.35 Round . . . 1042
10.181.3.36 Scale . . . 1042
10.181.3.37 Set . . . 1042
10.181.3.38 SetFromAxis . . . 1043
10.181.3.39 SetFromAxis . . . 1043

10.181.3.49 SetFromEuler 1043
10.181.3.50 SetFromEuler 1044
10.181.3.51 SetToIdentity 1044
10.181.3.52 Slerp 1044
10.181.3.53 Squad 1044
10.181.4 Friends And Related Function Documentation 1045
10.181.4.1 operator<< 1045
10.181.4.2 operator>> 1045
10.181.5 Member Data Documentation 1045
10.181.5.1 w 1045
10.181.5.2 x 1046
10.181.5.3 y 1046
10.181.5.4 z 1046
10.182 gazebo::math::Rand Class Reference 1046
10.182.1 Detailed Description 1047
10.182.2 Member Function Documentation 1047
10.182.2.1 GetDblNormal 1047
10.182.2.2 GetDblUniform 1047
10.182.2.3 GetIntNormal 1047
10.182.2.4 GetIntUniform 1048
10.182.2.5 GetSeed 1048
10.182.2.6 SetSeed 1048
10.183 gazebo::transport::RawCallbackHelper Class Reference 1048
10.183.1 Detailed Description 1049
10.183.2 Constructor & Destructor Documentation 1049
10.183.2.1 RawCallbackHelper 1050
10.183.3 Member Function Documentation 1050
10.183.3.1 GetMsgType 1050
10.183.3.2 HandleData 1050
10.183.3.3 HandleMessage 1051
10.183.3.4 IsLocal 1051

10.184 gazebo::sensors::RaySensor Class Reference 1051
10.184.1 Detailed Description 1053
10.184.2 Constructor & Destructor Documentation 1054
10.184.2.1 RaySensor 1054
10.184.2.2 ~RaySensor 1054
10.184.3 Member Function Documentation 1054
10.184.3.1 Fini 1054
10.184.3.2 GetAngleMax 1054
10.184.3.3 GetAngleMin 1054
10.184.3.4 GetAngleResolution 1054
10.184.3.5 GetFiducial 1055
10.184.3.6 GetLaserShape 1055
10.184.3.7 GetRange 1055
10.184.3.8 GetRangeCount 1056
10.184.3.9 GetRangeMax 1056
10.184.3.10 GetRangeMin 1056
10.184.3.11 GetRangeResolution 1056
10.184.3.12 GetRanges 1056
10.184.3.13 GetRayCount 1057
10.184.3.14 GetRetro 1057
10.184.3.15 GetTopic 1057
10.184.3.16 GetVerticalAngleMax 1057
10.184.3.17 GetVerticalAngleMin 1058
10.184.3.18 GetVerticalAngleResolution 1058
10.184.3.19 GetVerticalRangeCount 1058
10.184.3.20 GetVerticalRayCount 1058
10.184.3.21 Init 1058
10.184.3.22 IsActive 1059
10.184.3.23 Load 1059
10.184.3.24 UpdateImpl 1059
10.185 gazebo::physics::RayShape Class Reference 1060

10.185.1 Detailed Description 1062
10.185.2 Constructor & Destructor Documentation 1062
10.185.2.1 RayShape 1062
10.185.2.2 RayShape 1062
10.185.2.3 ~RayShape 1062
10.185.3 Member Function Documentation 1062
10.185.3.1 FillMsg 1062
10.185.3.2 GetFiducial 1063
10.185.3.3 GetGlobalPoints 1063
10.185.3.4 GetIntersection 1063
10.185.3.5 GetLength 1063
10.185.3.6 GetRelativePoints 1063
10.185.3.7 GetRetro 1064
10.185.3.8 Init 1064
10.185.3.9 ProcessMsg 1064
10.185.3.10 SetFiducial 1064
10.185.3.11 SetLength 1064
10.185.3.12 SetPoints 1065
10.185.3.13 SetRetro 1065
10.185.3.14 SetScale 1065
10.185.3.15 Update 1065
10.185.4 Member Data Documentation 1066
10.185.4.1 contactFiducial 1066
10.185.4.2 contactLen 1066
10.185.4.3 contactRetro 1066
10.185.4.4 globalEndPos 1066
10.185.4.5 globalStartPos 1066
10.185.4.6 relativeEndPos 1066
10.185.4.7 relativeStartPos 1066
10.186 gazebo::rendering::RenderEngine Class Reference 1067
10.186.1 Detailed Description 1068

10.186.2 Member Enumeration Documentation . . . 1068
10.186.2.1 RenderPathType . . . 1068
10.186.3 Member Function Documentation . . . 1069
10.186.3.1 AddResourcePath . . . 1069
10.186.3.2 CreateScene . . . 1069
10.186.3.3 Fini . . . 1069
10.186.3.4 GetRenderPathType . . . 1069
10.186.3.5 GetScene . . . 1070
10.186.3.6 GetScene . . . 1070
10.186.3.7 GetSceneCount . . . 1070
10.186.3.8 GetWindowManager . . . 1071
10.186.3.9 Init . . . 1071
10.186.3.10 Load . . . 1071
10.186.3.11 RemoveScene . . . 1071
10.186.4 Member Data Documentation . . . 1071
10.186.4.1 dummyContext . . . 1071
10.186.4.2 dummyDisplay . . . 1071
10.186.4.3 dummyWindowId . . . 1072
10.186.4.4 root . . . 1072
10.187 gazebo::sensors::RFIDSensor Class Reference . . . 1072
10.187.1 Detailed Description . . . 1073
10.187.2 Constructor & Destructor Documentation . . . 1073
10.187.2.1 RFIDSensor . . . 1073
10.187.2.2 ~RFIDSensor . . . 1073
10.187.3 Member Function Documentation . . . 1073
10.187.3.1 AddTag . . . 1073
10.187.3.2 Fini . . . 1073
10.187.3.3 Init . . . 1074
10.187.3.4 Load . . . 1074
10.187.3.5 Load . . . 1074
10.187.3.6 UpdateImpl . . . 1074

10.188 gazebo::sensors::RFIDTag Class Reference 1075
10.188.1 Detailed Description 1076
10.188.2 Constructor & Destructor Documentation 1076
10.188.2.1 RFIDTag 1076
10.188.2.2 ~RFIDTag 1076
10.188.3 Member Function Documentation 1076
10.188.3.1 Fini 1076
10.188.3.2 GetTagPose 1076
10.188.3.3 Init 1077
10.188.3.4 Load 1077
10.188.3.5 Load 1077
10.188.3.6 UpdateImpl 1077
10.189 gazebo::rendering::RFIDTagVisual Class Reference 1078
10.189.1 Detailed Description 1078
10.189.2 Constructor & Destructor Documentation 1079
10.189.2.1 RFIDTagVisual 1079
10.189.2.2 ~RFIDTagVisual 1079
10.190 gazebo::rendering::RFIDTagVisualPrivate Class Reference 1079
10.190.1 Detailed Description 1080
10.190.2 Member Data Documentation 1080
10.190.2.1 node 1080
10.190.2.2 rfidSub 1080
10.191 gazebo::rendering::RFIDVisual Class Reference 1081
10.191.1 Detailed Description 1081
10.191.2 Constructor & Destructor Documentation 1081
10.191.2.1 RFIDVisual 1081
10.191.2.2 ~RFIDVisual 1082
10.192 gazebo::rendering::RFIDVisualPrivate Class Reference 1082
10.192.1 Detailed Description 1083
10.192.2 Member Data Documentation 1083
10.192.2.1 node 1083

10.192.2.2fidSub 1083
10.193Road Class Reference 1083
10.193.1Detailed Description 1083
10.194gazebo::physics::Road Class Reference 1083
10.194.1Detailed Description 1084
10.194.2Constructor & Destructor Documentation 1084
10.194.2.1Road 1084
10.194.2.2~Road 1085
10.194.3Member Function Documentation 1085
10.194.3.1GetPoints 1085
10.194.3.2GetWidth 1085
10.194.3.3Init 1085
10.194.3.4Load 1085
10.195gazebo::rendering::Road2d Class Reference 1086
10.195.1Constructor & Destructor Documentation 1086
10.195.1.1Road2d 1086
10.195.1.2~Road2d 1086
10.195.2Member Function Documentation 1086
10.195.2.1Load 1086
10.196gazebo::math::RotationSpline Class Reference 1087
10.196.1Detailed Description 1088
10.196.2Constructor & Destructor Documentation 1088
10.196.2.1RotationSpline 1088
10.196.2.2~RotationSpline 1088
10.196.3Member Function Documentation 1088
10.196.3.1AddPoint 1088
10.196.3.2Clear 1088
10.196.3.3GetNumPoints 1089
10.196.3.4GetPoint 1089
10.196.3.5Interpolate 1089
10.196.3.6Interpolate 1090

10.196.3.7 RecalcTangents . . . 1090
10.196.3.8 SetAutoCalculate . . . 1090
10.196.3.9 UpdatePoint . . . 1091
10.196.4 Member Data Documentation . . . 1091
10.196.4.1 autoCalc . . . 1091
10.196.4.2 points . . . 1091
10.196.4.3 tangents . . . 1091
10.197 gazebo::rendering::RTShaderSystem Class Reference . . . 1092
10.197.1 Detailed Description . . . 1093
10.197.2 Member Enumeration Documentation . . . 1093
10.197.2.1 LightingModel . . . 1093
10.197.3 Member Function Documentation . . . 1094
10.197.3.1 AddScene . . . 1094
10.197.3.2 ApplyShadows . . . 1094
10.197.3.3 AttachEntity . . . 1094
10.197.3.4 AttachViewport . . . 1094
10.197.3.5 Clear . . . 1095
10.197.3.6 DetachEntity . . . 1095
10.197.3.7 DetachViewport . . . 1095
10.197.3.8 Fini . . . 1095
10.197.3.9 GenerateShaders . . . 1095
10.197.3.10 GetPSSMShadowCameraSetup . . . 1095
10.197.3.11 Init . . . 1096
10.197.3.12 RemoveScene . . . 1096
10.197.3.13 RemoveShadows . . . 1096
10.197.3.14 SetPerPixelLighting . . . 1096
10.197.3.15 UpdateShaders . . . 1096
10.198 gazebo::rendering::Scene Class Reference . . . 1097
10.198.1 Detailed Description . . . 1101
10.198.2 Member Enumeration Documentation . . . 1101
10.198.2.1 SkyXMode . . . 1101

10.198.3 Constructor & Destructor Documentation 1101
10.198.3.1 Scene 1101
10.198.3.2 ~Scene 1102
10.198.4 Member Function Documentation 1102
10.198.4.1 AddLight 1102
10.198.4.2 AddVisual 1102
10.198.4.3 Clear 1102
10.198.4.4 CreateCamera 1102
10.198.4.5 CreateDepthCamera 1103
10.198.4.6 CreateGpuLaser 1103
10.198.4.7 CreateGrid 1103
10.198.4.8 CreateUserCamera 1104
10.198.4.9 DrawLine 1104
10.198.4.10 GetAmbientColor 1104
10.198.4.11 GetBackgroundColor 1104
10.198.4.12 GetCamera 1105
10.198.4.13 GetCamera 1105
10.198.4.14 GetCameraCount 1105
10.198.4.15 GetFirstContact 1105
10.198.4.16 GetGrid 1106
10.198.4.17 GetGridCount 1106
10.198.4.18 GetHeightBelowPoint 1106
10.198.4.19 GetHeightmap 1107
10.198.4.20 GetId 1107
10.198.4.21 GetIdString 1107
10.198.4.22 GetInitialized 1107
10.198.4.23 GetLight 1107
10.198.4.24 GetLight 1108
10.198.4.25 GetLightCount 1108
10.198.4.26 GetManager 1108
10.198.4.27 GetModelVisualAt 1108

10.198.4.28 GetName 1109
10.198.4.29 GetSelectedVisual 1109
10.198.4.30 GetShadowsEnabled 1109
10.198.4.31 GetShowClouds 1109
10.198.4.32 GetSimTime 1109
10.198.4.33 GetUserCamera 1110
10.198.4.34 GetUserCameraCount 1110
10.198.4.35 GetVisual 1110
10.198.4.36 GetVisual 1111
10.198.4.37 GetVisualAt 1111
10.198.4.38 GetVisualAt 1111
10.198.4.39 GetVisualBelow 1112
10.198.4.40 GetVisualCount 1112
10.198.4.41 GetVisualsBelowPoint 1112
10.198.4.42 GetWorldVisual 1112
10.198.4.43 Init 1113
10.198.4.44 Load 1113
10.198.4.45 Load 1113
10.198.4.46 PreRender 1113
10.198.4.47 PrintSceneGraph 1113
10.198.4.48 RemoveCamera 1113
10.198.4.49 RemoveLight 1113
10.198.4.50 RemoveProjectors 1114
10.198.4.51 RemoveVisual 1114
10.198.4.52 SelectVisual 1114
10.198.4.53 SetAmbientColor 1114
10.198.4.54 SetBackgroundColor 1114
10.198.4.55 SetFog 1115
10.198.4.56 SetGrid 1115
10.198.4.57 SetShadowsEnabled 1115
10.198.4.58 SetSkyXMode 1115

10.198.4.59 SetTransparent 1116
10.198.4.60 SetVisible 1116
10.198.4.61 SetWireframe 1116
10.198.4.62 ShowClouds 1116
10.198.4.63 ShowCollisions 1117
10.198.4.64 ShowCOMs 1117
10.198.4.65 ShowContacts 1117
10.198.4.66 ShowJoints 1117
10.198.4.67 SnapVisualToNearestBelow 1117
10.198.4.68 StripSceneName 1118
10.198.5 Member Data Documentation 1118
10.198.5.1 skyx 1118
10.199 gazebo::physics::ScrewJoint< T > Class Template Reference 1118
10.199.1 Detailed Description 1119
10.199.2 Constructor & Destructor Documentation 1119
10.199.2.1 ScrewJoint 1119
10.199.2.2 ~ScrewJoint 1119
10.199.3 Member Function Documentation 1119
10.199.3.1 GetAngleCount 1119
10.199.3.2 GetThreadPitch 1119
10.199.3.3 Init 1120
10.199.3.4 Load 1120
10.199.3.5 SetThreadPitch 1120
10.199.4 Member Data Documentation 1121
10.199.4.1 threadPitch 1121
10.200 gazebo::rendering::SelectionObj Class Reference 1121
10.200.1 Detailed Description 1122
10.200.2 Member Enumeration Documentation 1122
10.200.2.1 SelectionMode 1122
10.200.3 Constructor & Destructor Documentation 1123
10.200.3.1 SelectionObj 1123

10.200.3.2 ~SelectionObj 1123
10.200.4 Member Function Documentation 1123
10.200.4.1 Attach 1123
10.200.4.2 Detach 1124
10.200.4.3 GetMode 1124
10.200.4.4 GetState 1124
10.200.4.5 Load 1124
10.200.4.6 SetGlobal 1124
10.200.4.7 SetMode 1124
10.200.4.8 SetMode 1124
10.200.4.9 SetState 1125
10.200.4.10 SetState 1125
10.200.4.11 UpdateSize 1125
10.201 gazebo::rendering::SelectionObj Private Class Reference 1125
10.201.1 Detailed Description 1127
10.201.2 Member Data Documentation 1127
10.201.2.1 maxScale 1127
10.201.2.2 minScale 1127
10.201.2.3 mode 1128
10.201.2.4 rotVisual 1128
10.201.2.5 rotXVisual 1128
10.201.2.6 rotYVisual 1128
10.201.2.7 rotZVisual 1128
10.201.2.8 scaleVisual 1128
10.201.2.9 scaleXVisual 1128
10.201.2.10 scaleYVisual 1128
10.201.2.11 scaleZVisual 1128
10.201.2.12 selectedVis 1129
10.201.2.13 state 1129
10.201.2.14 transVisual 1129
10.201.2.15 transXVisual 1129

10.201.2.16 TransYVisual 1129
10.201.2.17 TransZVisual 1129
10.201.2.18 AxisMat 1129
10.201.2.19 AxisMatOverlay 1129
10.201.2.20 AxisMat 1129
10.201.2.21 AxisMatOverlay 1130
10.201.2.22 AxisMat 1130
10.201.2.23 AxisMatOverlay 1130
10.202 gazebo::sensors::Sensor Class Reference 1130
10.202.1 Detailed Description 1134
10.202.2 Constructor & Destructor Documentation 1134
10.202.2.1 Sensor 1134
10.202.2.2 ~Sensor 1134
10.202.3 Member Function Documentation 1134
10.202.3.1 ConnectUpdated 1134
10.202.3.2 DisconnectUpdated 1135
10.202.3.3 FillMsg 1135
10.202.3.4 Fini 1135
10.202.3.5 GetCategory 1136
10.202.3.6 GetId 1136
10.202.3.7 GetLastMeasurementTime 1136
10.202.3.8 GetLastUpdateTime 1136
10.202.3.9 GetName 1137
10.202.3.10 GetNoise 1137
10.202.3.11 GetParentId 1137
10.202.3.12 GetParentName 1137
10.202.3.13 GetPose 1138
10.202.3.14 GetScopedName 1138
10.202.3.15 GetTopic 1138
10.202.3.16 GetType 1138
10.202.3.17 GetUpdateRate 1139

10.202.3.18 GetVisualize . . . 1139
10.202.3.19 GetWorldName . . . 1139
10.202.3.20 Init . . . 1139
10.202.3.21 IsActive . . . 1139
10.202.3.22 Load . . . 1140
10.202.3.23 Load . . . 1140
10.202.3.24 NeedsUpdate . . . 1141
10.202.3.25 ResetLastUpdateTime . . . 1141
10.202.3.26 SetActive . . . 1141
10.202.3.27 SetParent . . . 1141
10.202.3.28 SetUpdateRate . . . 1141
10.202.3.29 Update . . . 1142
10.202.3.30 UpdateImpl . . . 1142
10.202.4 Member Data Documentation . . . 1142
10.202.4.1 active . . . 1142
10.202.4.2 connections . . . 1142
10.202.4.3 lastMeasurementTime . . . 1143
10.202.4.4 lastUpdateTime . . . 1143
10.202.4.5 node . . . 1143
10.202.4.6 noises . . . 1143
10.202.4.7 parentId . . . 1143
10.202.4.8 parentName . . . 1143
10.202.4.9 plugins . . . 1143
10.202.4.10 pose . . . 1143
10.202.4.11 poseSub . . . 1144
10.202.4.12 scene . . . 1144
10.202.4.13 sdf . . . 1144
10.202.4.14 updatePeriod . . . 1144
10.202.4.15 world . . . 1144
10.203 SensorFactor Class Reference . . . 1144
10.203.1 Detailed Description . . . 1144

10.204 gazebo::sensors::SensorFactory Class Reference 1145
10.204.1 Member Function Documentation 1145
10.204.1.1 GetSensorTypes 1145
10.204.1.2 NewSensor 1145
10.204.1.3 RegisterAll 1146
10.204.1.4 RegisterSensor 1146
10.205 gazebo::sensors::SensorManager Class Reference 1146
10.205.1 Detailed Description 1148
10.205.2 Member Function Documentation 1148
10.205.2.1 CreateSensor 1148
10.205.2.2 Fini 1148
10.205.2.3 GetSensor 1149
10.205.2.4 GetSensors 1149
10.205.2.5 GetSensorTypes 1149
10.205.2.6 Init 1149
10.205.2.7 RemoveSensor 1149
10.205.2.8 RemoveSensors 1150
10.205.2.9 ResetLastUpdateTimes 1150
10.205.2.10 RunThreads 1150
10.205.2.11 SensorsInitialized 1150
10.205.2.12 Stop 1150
10.205.2.13 Update 1150
10.206 gazebo::SensorPlugin Class Reference 1151
10.206.1 Detailed Description 1151
10.206.2 Constructor & Destructor Documentation 1152
10.206.2.1 SensorPlugin 1152
10.206.2.2 ~SensorPlugin 1152
10.206.3 Member Function Documentation 1152
10.206.3.1 Init 1152
10.206.3.2 Load 1152
10.206.3.3 Reset 1152

10.207 gazebo::Server Class Reference 1153
10.207.1 Constructor & Destructor Documentation 1153
10.207.1.1 Server 1153
10.207.1.2 ~Server 1154
10.207.2 Member Function Documentation 1154
10.207.2.1 Fini 1154
10.207.2.2 GetInitialized 1154
10.207.2.3 LoadFile 1154
10.207.2.4 LoadString 1154
10.207.2.5 ParseArgs 1155
10.207.2.6 PreLoad 1155
10.207.2.7 PrintUsage 1155
10.207.2.8 Run 1155
10.207.2.9 SetParams 1155
10.207.2.10 Stop 1156
10.208 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference 1156
10.208.1 Detailed Description 1157
10.208.2 Member Function Documentation 1157
10.208.2.1 defaultVpParams 1157
10.208.2.2 generateFragmentProgram 1157
10.208.2.3 generateVertexProgram 1157
10.208.2.4 generateVertexProgramSource 1157
10.208.2.5 generateVpDynamicShadows 1157
10.208.2.6 generateVpDynamicShadowsParams 1157
10.208.2.7 generateVpFooter 1157
10.208.2.8 generateVpHeader 1158
10.209 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference 1158
10.209.1 Detailed Description 1159
10.209.2 Member Function Documentation 1159
10.209.2.1 defaultVpParams 1159

10.209.2.2 generateFpDynamicShadows . . . 1159
10.209.2.3 generateFpDynamicShadowsHelpers . . . 1159
10.209.2.4 generateFpDynamicShadowsParams . . . 1160
10.209.2.5 generateFpFooter . . . 1160
10.209.2.6 generateFpHeader . . . 1160
10.209.2.7 generateFpLayer . . . 1160
10.209.2.8 generateFragmentProgram . . . 1160
10.209.2.9 generateFragmentProgramSource . . . 1160
10.209.2.10 generateVertexProgram . . . 1160
10.209.2.11 generateVertexProgramSource . . . 1160
10.209.2.12 generateVpDynamicShadows . . . 1161
10.209.2.13 generateVpDynamicShadowsParams . . . 1161
10.209.2.14 generateVpFooter . . . 1161
10.209.2.15 generateVpHeader . . . 1161
10.209.2.16 updateParams . . . 1161
10.209.2.17 updateVpParams . . . 1161
10.210 gazebo::physics::Shape Class Reference . . . 1161
10.210.1 Detailed Description . . . 1163
10.210.2 Constructor & Destructor Documentation . . . 1163
10.210.2.1 Shape . . . 1163
10.210.2.2 ~Shape . . . 1163
10.210.3 Member Function Documentation . . . 1163
10.210.3.1 FillMsg . . . 1163
10.210.3.2 GetScale . . . 1164
10.210.3.3 Init . . . 1164
10.210.3.4 ProcessMsg . . . 1164
10.210.3.5 SetScale . . . 1165
10.210.4 Member Data Documentation . . . 1165
10.210.4.1 collisionParent . . . 1165
10.210.4.2 scale . . . 1165
10.211 gazebo::physics::SimbodyBallJoint Class Reference . . . 1165

10.211.1 Detailed Description . . . 1167
10.211.2 Constructor & Destructor Documentation . . . 1167
10.211.2.1 SimbodyBallJoint . . . 1167
10.211.2.2 ~SimbodyBallJoint . . . 1168
10.211.3 Member Function Documentation . . . 1168
10.211.3.1 GetAnchor . . . 1168
10.211.3.2 GetAngleImpl . . . 1168
10.211.3.3 GetAxis . . . 1168
10.211.3.4 GetGlobalAxis . . . 1169
10.211.3.5 GetHighStop . . . 1169
10.211.3.6 GetLowStop . . . 1169
10.211.3.7 GetMaxForce . . . 1170
10.211.3.8 GetVelocity . . . 1170
10.211.3.9 Load . . . 1170
10.211.3.10 SetAxis . . . 1171
10.211.3.11 SetForceImpl . . . 1171
10.211.3.12 SetHighStop . . . 1171
10.211.3.13 SetLowStop . . . 1172
10.211.3.14 SetMaxForce . . . 1172
10.211.3.15 SetVelocity . . . 1172
10.212 gazebo::physics::SimbodyBoxShape Class Reference . . . 1173
10.212.1 Detailed Description . . . 1174
10.212.2 Constructor & Destructor Documentation . . . 1174
10.212.2.1 SimbodyBoxShape . . . 1174
10.212.2.2 ~SimbodyBoxShape . . . 1174
10.212.3 Member Function Documentation . . . 1174
10.212.3.1 SetSize . . . 1174
10.213 gazebo::physics::SimbodyCollision Class Reference . . . 1174
10.213.1 Detailed Description . . . 1176
10.213.2 Constructor & Destructor Documentation . . . 1176
10.213.2.1 SimbodyCollision . . . 1176

10.213.2.2 ~SimbodyCollision 1176
10.213.3 Member Function Documentation 1176
10.213.3.1 GetBoundingBox 1176
10.213.3.2 GetCollisionShape 1176
10.213.3.3 Load 1177
10.213.3.4 OnPoseChange 1177
10.213.3.5 SetCategoryBits 1177
10.213.3.6 SetCollideBits 1177
10.213.3.7 SetCollisionShape 1178
10.214 gazebo::physics::SimbodyCylinderShape Class Reference 1178
10.214.1 Detailed Description 1179
10.214.2 Constructor & Destructor Documentation 1179
10.214.2.1 SimbodyCylinderShape 1179
10.214.2.2 ~SimbodyCylinderShape 1179
10.214.3 Member Function Documentation 1179
10.214.3.1 SetSize 1179
10.215 gazebo::physics::SimbodyHeightmapShape Class Reference 1180
10.215.1 Detailed Description 1181
10.215.2 Constructor & Destructor Documentation 1181
10.215.2.1 SimbodyHeightmapShape 1181
10.215.2.2 ~SimbodyHeightmapShape 1181
10.215.3 Member Function Documentation 1181
10.215.3.1 Init 1181
10.216 gazebo::physics::SimbodyHinge2Joint Class Reference 1181
10.216.1 Detailed Description 1183
10.216.2 Constructor & Destructor Documentation 1183
10.216.2.1 SimbodyHinge2Joint 1183
10.216.2.2 ~SimbodyHinge2Joint 1183
10.216.3 Member Function Documentation 1184
10.216.3.1 GetAnchor 1184
10.216.3.2 GetAngleImpl 1184

10.216.3.3 GetAxis . . . 1184
10.216.3.4 GetGlobalAxis . . . 1184
10.216.3.5 GetMaxForce . . . 1185
10.216.3.6 GetVelocity . . . 1185
10.216.3.7 Load . . . 1186
10.216.3.8 SetAxis . . . 1186
10.216.3.9 SetForceImpl . . . 1186
10.216.3.10 SetMaxForce . . . 1186
10.216.3.11 SetVelocity . . . 1187
10.217 gazebo::physics::SimbodyHingeJoint Class Reference . . . 1187
10.217.1 Detailed Description . . . 1189
10.217.2 Constructor & Destructor Documentation . . . 1189
10.217.2.1 SimbodyHingeJoint . . . 1189
10.217.2.2 ~SimbodyHingeJoint . . . 1189
10.217.3 Member Function Documentation . . . 1190
10.217.3.1 GetAngleImpl . . . 1190
10.217.3.2 GetGlobalAxis . . . 1190
10.217.3.3 GetMaxForce . . . 1190
10.217.3.4 GetVelocity . . . 1191
10.217.3.5 Load . . . 1191
10.217.3.6 RestoreSimbodyState . . . 1191
10.217.3.7 SaveSimbodyState . . . 1191
10.217.3.8 SetAxis . . . 1192
10.217.3.9 SetForceImpl . . . 1192
10.217.3.10 SetMaxForce . . . 1192
10.217.3.11 SetVelocity . . . 1193
10.218 gazebo::physics::SimbodyJoint Class Reference . . . 1193
10.218.1 Detailed Description . . . 1196
10.218.2 Constructor & Destructor Documentation . . . 1196
10.218.2.1 SimbodyJoint . . . 1196
10.218.2.2 ~SimbodyJoint . . . 1196

10.218.3 Member Function Documentation 1196
10.218.3.1 AreConnected 1196
10.218.3.2 CacheForceTorque 1197
10.218.3.3 Detach 1197
10.218.3.4 GetAnchor 1197
10.218.3.5 GetForce 1197
10.218.3.6 GetForceTorque 1198
10.218.3.7 GetHighStop 1198
10.218.3.8 GetJointLink 1199
10.218.3.9 GetLinkForce 1199
10.218.3.10 GetLinkTorque 1200
10.218.3.11 GetLowStop 1200
10.218.3.12 GetParam 1200
10.218.3.13 Load 1201
10.218.3.14 Reset 1201
10.218.3.15 RestoreSimbodyState 1201
10.218.3.16 SaveSimbodyState 1201
10.218.3.17 SetAnchor 1202
10.218.3.18 SetAttribute 1202
10.218.3.19 SetAxis 1202
10.218.3.20 SetDamping 1202
10.218.3.21 SetForce 1203
10.218.3.22 SetForceImpl 1203
10.218.3.23 SetHighStop 1204
10.218.3.24 SetLowStop 1204
10.218.3.25 SetParam 1204
10.218.3.26 SetStiffness 1205
10.218.3.27 SetStiffnessDamping 1205
10.218.4 Member Data Documentation 1205
10.218.4.1 constraint 1205
10.218.4.2 damper 1205

10.218.4.3 defxAB . . . 1206
10.218.4.4 isReversed . . . 1206
10.218.4.5 limitForce . . . 1206
10.218.4.6 mobod . . . 1206
10.218.4.7 mustBreakLoopHere . . . 1206
10.218.4.8 physicsInitialized . . . 1206
10.218.4.9 simbodyPhysics . . . 1206
10.218.4.10 spring . . . 1207
10.218.4.11 world . . . 1207
10.218.4.12 xCB . . . 1207
10.218.4.13 xPA . . . 1207
10.219 gazebo::physics::SimbodyLink Class Reference . . . 1207
10.219.1 Detailed Description . . . 1210
10.219.2 Constructor & Destructor Documentation . . . 1210
10.219.2.1 SimbodyLink . . . 1210
10.219.2.2 ~SimbodyLink . . . 1211
10.219.3 Member Function Documentation . . . 1211
10.219.3.1 AddForce . . . 1211
10.219.3.2 AddForceAtRelativePosition . . . 1211
10.219.3.3 AddForceAtWorldPosition . . . 1211
10.219.3.4 AddRelativeForce . . . 1212
10.219.3.5 AddRelativeTorque . . . 1212
10.219.3.6 AddTorque . . . 1212
10.219.3.7 Fini . . . 1212
10.219.3.8 GetEffectiveMassProps . . . 1212
10.219.3.9 GetEnabled . . . 1213
10.219.3.10 GetGravityMode . . . 1213
10.219.3.11 GetMassProperties . . . 1213
10.219.3.12 GetWorldAngularVel . . . 1213
10.219.3.13 GetWorldCoGLinearVel . . . 1213
10.219.3.14 GetWorldForce . . . 1214

10.219.3.16 GetWorldLinearVel 1214
10.219.3.16 GetWorldLinearVel 1214
10.219.3.17 GetWorldTorque 1215
10.219.3.18 Init 1215
10.219.3.19 Load 1215
10.219.3.20 OnPoseChange 1216
10.219.3.21 RestoreSimbodyState 1216
10.219.3.22 SaveSimbodyState 1216
10.219.3.23 SetAngularDamping 1216
10.219.3.24 SetAngularVel 1216
10.219.3.25 SetAutoDisable 1216
10.219.3.26 SetDirtyPose 1217
10.219.3.27 SetEnabled 1217
10.219.3.28 SetForce 1217
10.219.3.29 SetGravityMode 1217
10.219.3.30 SetLinearDamping 1217
10.219.3.31 SetLinearVel 1218
10.219.3.32 SetLinkStatic 1218
10.219.3.33 SetSelfCollide 1218
10.219.3.34 SetTorque 1219
10.219.4 Member Data Documentation 1219
10.219.4.1 masterMobod 1219
10.219.4.2 mustBeBaseLink 1219
10.219.4.3 physicsInitialized 1219
10.219.4.4 slaveMobods 1219
10.219.4.5 slaveWelds 1219
10.220 gazebo::physics::SimbodyMeshShape Class Reference 1219
10.220.1 Detailed Description 1220
10.220.2 Constructor & Destructor Documentation 1221
10.220.2.1 SimbodyMeshShape 1221
10.220.2.2 ~SimbodyMeshShape 1221

10.220.3 Member Function Documentation 1221
10.220.3.1 Init 1221
10.220.3.2 Load 1221
10.221 gazebo::physics::SimbodyModel Class Reference 1221
10.221.1 Detailed Description 1222
10.221.2 Constructor & Destructor Documentation 1223
10.221.2.1 SimbodyModel 1223
10.221.2.2 ~SimbodyModel 1223
10.221.3 Member Function Documentation 1223
10.221.3.1 Init 1223
10.221.3.2 Load 1223
10.222 gazebo::physics::SimbodyMultiRayShape Class Reference 1223
10.222.1 Detailed Description 1225
10.222.2 Constructor & Destructor Documentation 1225
10.222.2.1 SimbodyMultiRayShape 1225
10.222.2.2 ~SimbodyMultiRayShape 1225
10.222.3 Member Function Documentation 1225
10.222.3.1 AddRay 1225
10.222.3.2 UpdateRays 1225
10.223 gazebo::physics::SimbodyPhysics Class Reference 1226
10.223.1 Detailed Description 1228
10.223.2 Constructor & Destructor Documentation 1229
10.223.2.1 SimbodyPhysics 1229
10.223.2.2 ~SimbodyPhysics 1229
10.223.3 Member Function Documentation 1229
10.223.3.1 CreateCollision 1229
10.223.3.2 CreateJoint 1229
10.223.3.3 CreateLink 1229
10.223.3.4 CreateModel 1230
10.223.3.5 CreateShape 1230
10.223.3.6 DebugPrint 1230

10.223.3.7 Fini . . . 1230
10.223.3.8 GetDynamicsWorld . . . 1231
10.223.3.9 GetParam . . . 1231
10.223.3.10 GetPose . . . 1231
10.223.3.11 GetType . . . 1231
10.223.3.12 GetTypeString . . . 1232
10.223.3.13 GetTypeString . . . 1232
10.223.3.14 Init . . . 1232
10.223.3.15 InitForThread . . . 1232
10.223.3.16 InitModel . . . 1232
10.223.3.17 Load . . . 1233
10.223.3.18 OnPhysicsMsg . . . 1233
10.223.3.19 OnRequest . . . 1233
10.223.3.20 Pose2Transform . . . 1233
10.223.3.21 QuadToQuad . . . 1234
10.223.3.22 QuadToQuad . . . 1234
10.223.3.23 Reset . . . 1234
10.223.3.24 SetGravity . . . 1235
10.223.3.25 SetParam . . . 1235
10.223.3.26 SetSeed . . . 1236
10.223.3.27 Transform2Pose . . . 1237
10.223.3.28 UpdateCollision . . . 1237
10.223.3.29 UpdatePhysics . . . 1237
10.223.3.30 Vec3ToVector3 . . . 1237
10.223.3.31 Vector3ToVec3 . . . 1238
10.223.4 Member Data Documentation . . . 1238
10.223.4.1 contact . . . 1238
10.223.4.2 discreteForces . . . 1238
10.223.4.3 forces . . . 1238
10.223.4.4 gravity . . . 1238
10.223.4.5 inteq . . . 1238

10.223.4.6matter 1238
10.223.4.7simbodyPhysicsInitialized 1238
10.223.4.8simbodyPhysicsStepped 1238
10.223.4.9system 1238
10.223.4.10tracker 1238
10.224gazebo::physics::SimbodyPlaneShape Class Reference 1239
10.224.1Detailed Description 1240
10.224.2Constructor & Destructor Documentation 1240
10.224.2.1SimbodyPlaneShape 1240
10.224.2.2∼SimbodyPlaneShape 1240
10.224.3Member Function Documentation 1240
10.224.3.1CreatePlane 1240
10.224.3.2SetAltitude 1240
10.225gazebo::physics::SimbodyRayShape Class Reference 1241
10.225.1Detailed Description 1242
10.225.2Constructor & Destructor Documentation 1242
10.225.2.1SimbodyRayShape 1242
10.225.2.2SimbodyRayShape 1242
10.225.2.3∼SimbodyRayShape 1242
10.225.3Member Function Documentation 1242
10.225.3.1GetIntersection 1242
10.225.3.2SetPoints 1243
10.225.3.3Update 1243
10.226gazebo::physics::SimbodyScrewJoint Class Reference 1243
10.226.1Detailed Description 1245
10.226.2Constructor & Destructor Documentation 1246
10.226.2.1SimbodyScrewJoint 1246
10.226.2.2∼SimbodyScrewJoint 1246
10.226.3Member Function Documentation 1246
10.226.3.1GetAngleImpl 1246
10.226.3.2GetGlobalAxis 1246

10.226.3.3 GetHighStop . . . 1247
10.226.3.4 GetLowStop . . . 1247
10.226.3.5 GetMaxForce . . . 1248
10.226.3.6 GetParam . . . 1248
10.226.3.7 GetThreadPitch . . . 1248
10.226.3.8 GetThreadPitch . . . 1248
10.226.3.9 GetVelocity . . . 1249
10.226.3.10 Load . . . 1249
10.226.3.11 SetAxis . . . 1249
10.226.3.12 SetForceImpl . . . 1249
10.226.3.13 SetHighStop . . . 1250
10.226.3.14 SetLowStop . . . 1250
10.226.3.15 SetMaxForce . . . 1250
10.226.3.16 SetParam . . . 1251
10.226.3.17 SetThreadPitch . . . 1251
10.226.3.18 SetThreadPitch . . . 1251
10.226.3.19 SetVelocity . . . 1252
10.227 gazebo::physics::SimbodySliderJoint Class Reference . . . 1252
10.227.1 Detailed Description . . . 1254
10.227.2 Constructor & Destructor Documentation . . . 1254
10.227.2.1 SimbodySliderJoint . . . 1254
10.227.2.2 ~SimbodySliderJoint . . . 1254
10.227.3 Member Function Documentation . . . 1254
10.227.3.1 GetAngleImpl . . . 1255
10.227.3.2 GetGlobalAxis . . . 1255
10.227.3.3 GetMaxForce . . . 1255
10.227.3.4 GetVelocity . . . 1256
10.227.3.5 Load . . . 1256
10.227.3.6 SetAxis . . . 1256
10.227.3.7 SetForceImpl . . . 1257
10.227.3.8 SetMaxForce . . . 1257

10.227.3.9 SetVelocity 1257
10.228 gazebo::physics::SimbodySphereShape Class Reference 1258
10.228.1 Detailed Description 1259
10.228.2 Constructor & Destructor Documentation 1259
10.228.2.1 SimbodySphereShape 1259
10.228.2.2 ~SimbodySphereShape 1259
10.228.3 Member Function Documentation 1259
10.228.3.1 SetRadius 1259
10.229 gazebo::physics::SimbodyUniversalJoint Class Reference 1260
10.229.1 Detailed Description 1261
10.229.2 Constructor & Destructor Documentation 1261
10.229.2.1 SimbodyUniversalJoint 1261
10.229.2.2 ~SimbodyUniversalJoint 1262
10.229.3 Member Function Documentation 1262
10.229.3.1 GetAnchor 1262
10.229.3.2 GetAngleImpl 1262
10.229.3.3 GetAxis 1262
10.229.3.4 GetGlobalAxis 1263
10.229.3.5 GetMaxForce 1263
10.229.3.6 GetVelocity 1263
10.229.3.7 Load 1264
10.229.3.8 SetAxis 1264
10.229.3.9 SetForceImpl 1264
10.229.3.10 SetMaxForce 1265
10.229.3.11 SetVelocity 1265
10.230 gazebo::sensors::SimTimeEvent Class Reference 1265
10.230.1 Detailed Description 1266
10.230.2 Member Data Documentation 1266
10.230.2.1 condition 1266
10.230.2.2 time 1266
10.231 gazebo::sensors::SimTimeEventHandler Class Reference 1266

10.231.1 Detailed Description 1267
10.231.2 Constructor & Destructor Documentation 1267
10.231.2.1 SimTimeEventHandler 1267
10.231.2.2 ~SimTimeEventHandler 1267
10.231.3 Member Function Documentation 1267
10.231.3.1 AddRelativeEvent 1267
10.232 SingletonT< T > Class Template Reference 1267
10.232.1 Detailed Description 1269
10.232.2 Constructor & Destructor Documentation 1269
10.232.2.1 SingletonT 1269
10.232.2.2 ~SingletonT 1269
10.232.3 Member Function Documentation 1269
10.232.3.1 Instance 1269
10.233 gazebo::common::Skeleton Class Reference 1269
10.233.1 Detailed Description 1271
10.233.2 Constructor & Destructor Documentation 1271
10.233.2.1 Skeleton 1271
10.233.2.2 Skeleton 1271
10.233.2.3 ~Skeleton 1272
10.233.3 Member Function Documentation 1272
10.233.3.1 AddAnimation 1272
10.233.3.2 AddVertNodeWeight 1272
10.233.3.3 BuildNodeMap 1272
10.233.3.4 GetAnimation 1272
10.233.3.5 GetBindShapeTransform 1273
10.233.3.6 GetNodeByHandle 1273
10.233.3.7 GetNodeById 1273
10.233.3.8 GetNodeByName 1274
10.233.3.9 GetNodes 1274
10.233.3.10 GetNumAnimations 1274
10.233.3.11 GetNumJoints 1274

10.233.3.12GetNumNodes 1274
10.233.3.13GetNumVertNodeWeights 1275
10.233.3.14GetRootNode 1275
10.233.3.15GetVertNodeWeight 1275
10.233.3.16PrintTransforms 1275
10.233.3.17Scale 1276
10.233.3.18SetBindShapeTransform 1276
10.233.3.19SetNumVertAttached 1276
10.233.3.20SetRootNode 1276
10.233.4Member Data Documentation 1276
10.233.4.1anims 1276
10.233.4.2bindShapeTransform 1277
10.233.4.3nodes 1277
10.233.4.4rawNW 1277
10.233.4.5root 1277
10.234gazebo::common::SkeletonAnimation Class Reference 1277
10.234.1Detailed Description 1278
10.234.2Constructor & Destructor Documentation 1279
10.234.2.1SkeletonAnimation 1279
10.234.2.2~SkeletonAnimation 1279
10.234.3Member Function Documentation 1279
10.234.3.1AddKeyFrame 1279
10.234.3.2AddKeyFrame 1279
10.234.3.3GetLength 1280
10.234.3.4GetName 1280
10.234.3.5GetNodeCount 1280
10.234.3.6GetNodePoseAt 1280
10.234.3.7GetPoseAt 1281
10.234.3.8GetPoseAtX 1281
10.234.3.9HasNode 1281
10.234.3.10Scale 1282

10.234.3.1 SetName . . . 1282
10.234.4 Member Data Documentation . . . 1282
10.234.4.1 animations . . . 1282
10.234.4.2 length . . . 1282
10.234.4.3 name . . . 1282
10.235 gazebo::common::SkeletonNode Class Reference . . . 1283
10.235.1 Detailed Description . . . 1285
10.235.2 Member Enumeration Documentation . . . 1285
10.235.2.1 SkeletonNodeType . . . 1285
10.235.3 Constructor & Destructor Documentation . . . 1286
10.235.3.1 SkeletonNode . . . 1286
10.235.3.2 SkeletonNode . . . 1286
10.235.3.3 ~SkeletonNode . . . 1286
10.235.4 Member Function Documentation . . . 1286
10.235.4.1 AddChild . . . 1286
10.235.4.2 AddRawTransform . . . 1287
10.235.4.3 GetChild . . . 1287
10.235.4.4 GetChildById . . . 1287
10.235.4.5 GetChildByName . . . 1287
10.235.4.6 GetChildCount . . . 1288
10.235.4.7 GetHandle . . . 1288
10.235.4.8 GetId . . . 1288
10.235.4.9 GetInverseBindTransform . . . 1288
10.235.4.10 GetModelTransform . . . 1289
10.235.4.11 GetName . . . 1289
10.235.4.12 GetNumRawTrans . . . 1289
10.235.4.13 GetParent . . . 1289
10.235.4.14 GetRawTransform . . . 1289
10.235.4.15 GetRawTransforms . . . 1290
10.235.4.16 GetTransform . . . 1290
10.235.4.17 GetTransforms . . . 1290

10.235.4.18 IsJoint . . . 1290
10.235.4.19 IsRootNode . . . 1290
10.235.4.20 Reset . . . 1291
10.235.4.21 SetHandle . . . 1291
10.235.4.22 SetId . . . 1291
10.235.4.23 SetInitialTransform . . . 1291
10.235.4.24 SetInverseBindTransform . . . 1291
10.235.4.25 SetModelTransform . . . 1292
10.235.4.26 SetName . . . 1292
10.235.4.27 SetParent . . . 1292
10.235.4.28 SetTransform . . . 1292
10.235.4.29 SetType . . . 1293
10.235.4.30 UpdateChildrenTransforms . . . 1293
10.235.5 Member Data Documentation . . . 1293
10.235.5.1 children . . . 1293
10.235.5.2 handle . . . 1293
10.235.5.3 id . . . 1293
10.235.5.4 initialTransform . . . 1293
10.235.5.5 invBindTransform . . . 1293
10.235.5.6 modelTransform . . . 1294
10.235.5.7 name . . . 1294
10.235.5.8 parent . . . 1294
10.235.5.9 rawTransforms . . . 1294
10.235.5.10 transform . . . 1294
10.235.5.11 type . . . 1294
10.236 gazebo::physics::SliderJoint< T > Class Template Reference . . . 1294
10.236.1 Detailed Description . . . 1295
10.236.2 Constructor & Destructor Documentation . . . 1295
10.236.2.1 SliderJoint . . . 1295
10.236.2.2 ~SliderJoint . . . 1295
10.236.3 Member Function Documentation . . . 1295

10.236.3.1 GetAngleCount 1295
10.236.3.2 Load 1295
10.237 gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference . . . 1296
10.237.1 Detailed Description 1297
10.237.2 Constructor & Destructor Documentation 1297
10.237.2.1 SM2Profile 1297
10.237.2.2 ~SM2Profile 1297
10.237.3 Member Function Documentation 1297
10.237.3.1 addTechnique 1297
10.237.3.2 generate 1297
10.237.3.3 generateForCompositeMap 1297
10.237.3.4 UpdateParams 1297
10.237.3.5 UpdateParamsForCompositeMap 1297
10.238 gazebo::sensors::SonarSensor Class Reference 1298
10.238.1 Detailed Description 1299
10.238.2 Constructor & Destructor Documentation 1299
10.238.2.1 SonarSensor 1299
10.238.2.2 ~SonarSensor 1299
10.238.3 Member Function Documentation 1300
10.238.3.1 ConnectUpdate 1300
10.238.3.2 DisconnectUpdate 1300
10.238.3.3 Fini 1300
10.238.3.4 GetRadius 1300
10.238.3.5 GetRange 1300
10.238.3.6 GetRangeMax 1301
10.238.3.7 GetRangeMin 1301
10.238.3.8 GetTopic 1301
10.238.3.9 Init 1301
10.238.3.10 IsActive 1302
10.238.3.11 Load 1302
10.238.3.12 UpdateImpl 1302

10.238.4 Member Data Documentation . . . 1302
10.238.4.1 update . . . 1302
10.239 gazebo::rendering::SonarVisual Class Reference . . . 1303
10.239.1 Detailed Description . . . 1303
10.239.2 Constructor & Destructor Documentation . . . 1304
10.239.2.1 SonarVisual . . . 1304
10.239.2.2 ~SonarVisual . . . 1304
10.239.3 Member Function Documentation . . . 1304
10.239.3.1 Load . . . 1304
10.240 gazebo::rendering::SonarVisualPrivate Class Reference . . . 1304
10.240.1 Detailed Description . . . 1305
10.240.2 Member Data Documentation . . . 1306
10.240.2.1 coneNode . . . 1306
10.240.2.2 connections . . . 1306
10.240.2.3 mutex . . . 1306
10.240.2.4 node . . . 1306
10.240.2.5 receivedMsg . . . 1306
10.240.2.6 sonarMsg . . . 1306
10.240.2.7 sonarRay . . . 1306
10.240.2.8 sonarSub . . . 1306
10.241 gazebo::physics::SphereShape Class Reference . . . 1307
10.241.1 Detailed Description . . . 1308
10.241.2 Constructor & Destructor Documentation . . . 1308
10.241.2.1 SphereShape . . . 1308
10.241.2.2 ~SphereShape . . . 1308
10.241.3 Member Function Documentation . . . 1308
10.241.3.1 FillMsg . . . 1308
10.241.3.2 GetRadius . . . 1308
10.241.3.3 Init . . . 1309
10.241.3.4 ProcessMsg . . . 1309
10.241.3.5 SetRadius . . . 1309

10.241.3.6 SetScale . . . 1309
10.242 gazebo::common::SphericalCoordinates Class Reference . . . 1310
10.242.1 Detailed Description . . . 1311
10.242.2 Member Enumeration Documentation . . . 1311
10.242.2.1 SurfaceType . . . 1311
10.242.3 Constructor & Destructor Documentation . . . 1311
10.242.3.1 SphericalCoordinates . . . 1311
10.242.3.2 SphericalCoordinates . . . 1312
10.242.3.3 SphericalCoordinates . . . 1312
10.242.3.4 ~SphericalCoordinates . . . 1312
10.242.4 Member Function Documentation . . . 1312
10.242.4.1 Convert . . . 1312
10.242.4.2 Distance . . . 1313
10.242.4.3 GetElevationReference . . . 1313
10.242.4.4 GetHeadingOffset . . . 1313
10.242.4.5 GetLatitudeReference . . . 1313
10.242.4.6 GetLongitudeReference . . . 1314
10.242.4.7 GetSurfaceType . . . 1314
10.242.4.8 GlobalFromLocal . . . 1314
10.242.4.9 SetElevationReference . . . 1314
10.242.4.10 SetHeadingOffset . . . 1315
10.242.4.11 SetLatitudeReference . . . 1315
10.242.4.12 SetLongitudeReference . . . 1315
10.242.4.13 SetSurfaceType . . . 1315
10.242.4.14 SphericalFromLocal . . . 1316
10.243 gazebo::common::SphericalCoordinatesPrivate Class Reference . . . 1316
10.243.1 Detailed Description . . . 1317
10.243.2 Member Data Documentation . . . 1317
10.243.2.1 elevationReference . . . 1317
10.243.2.2 headingOffset . . . 1317
10.243.2.3 latitudeReference . . . 1317

10.243.2.4 longitudeReference . . . 1317
10.243.2.5 surfaceType . . . 1317
10.244 gazebo::math::Spline Class Reference . . . 1317
10.244.1 Detailed Description . . . 1319
10.244.2 Constructor & Destructor Documentation . . . 1319
10.244.2.1 Spline . . . 1319
10.244.2.2 ~Spline . . . 1319
10.244.3 Member Function Documentation . . . 1319
10.244.3.1 AddPoint . . . 1319
10.244.3.2 Clear . . . 1319
10.244.3.3 GetPoint . . . 1319
10.244.3.4 GetPointCount . . . 1320
10.244.3.5 GetTangent . . . 1320
10.244.3.6 GetTension . . . 1320
10.244.3.7 Interpolate . . . 1320
10.244.3.8 Interpolate . . . 1321
10.244.3.9 RecalcTangents . . . 1321
10.244.3.10 SetAutoCalculate . . . 1321
10.244.3.11 SetTension . . . 1322
10.244.3.12 UpdatePoint . . . 1322
10.244.4 Member Data Documentation . . . 1322
10.244.4.1 autoCalc . . . 1322
10.244.4.2 coeffs . . . 1322
10.244.4.3 points . . . 1322
10.244.4.4 tangents . . . 1322
10.244.4.5 tension . . . 1323
10.245 gazebo::physics::State Class Reference . . . 1323
10.245.1 Detailed Description . . . 1324
10.245.2 Constructor & Destructor Documentation . . . 1325
10.245.2.1 State . . . 1325
10.245.2.2 State . . . 1325

10.245.2.3 ~State . . . 1325
10.245.3 Member Function Documentation . . . 1325
10.245.3.1 GetName . . . 1325
10.245.3.2 GetRealTime . . . 1325
10.245.3.3 GetSimTime . . . 1326
10.245.3.4 GetWallTime . . . 1326
10.245.3.5 Load . . . 1326
10.245.3.6 operator- . . . 1326
10.245.3.7 operator= . . . 1327
10.245.3.8 SetName . . . 1327
10.245.3.9 SetRealTime . . . 1327
10.245.3.10 SetSimTime . . . 1327
10.245.3.11 SetWallTime . . . 1328
10.245.4 Member Data Documentation . . . 1328
10.245.4.1 name . . . 1328
10.245.4.2 realTime . . . 1328
10.245.4.3 simTime . . . 1328
10.245.4.4 wallTime . . . 1328
10.246 gazebo::common::STLLoader Class Reference . . . 1328
10.246.1 Detailed Description . . . 1329
10.246.2 Constructor & Destructor Documentation . . . 1329
10.246.2.1 STLLoader . . . 1329
10.246.2.2 ~STLLoader . . . 1329
10.246.3 Member Function Documentation . . . 1330
10.246.3.1 Load . . . 1330
10.247 gazebo::common::SubMesh Class Reference . . . 1330
10.247.1 Detailed Description . . . 1333
10.247.2 Member Enumeration Documentation . . . 1333
10.247.2.1 PrimitiveType . . . 1333
10.247.3 Constructor & Destructor Documentation . . . 1333
10.247.3.1 SubMesh . . . 1333

10.247.3.2 SubMesh 1333
10.247.3.3 ~SubMesh 1333
10.247.4 Member Function Documentation 1334
10.247.4.1 AddIndex 1334
10.247.4.2 AddNodeAssignment 1334
10.247.4.3 AddNormal 1334
10.247.4.4 AddNormal 1334
10.247.4.5 AddTexCoord 1335
10.247.4.6 AddVertex 1335
10.247.4.7 AddVertex 1335
10.247.4.8 Center 1335
10.247.4.9 CopyNormals 1336
10.247.4.10 CopyVertices 1336
10.247.4.11 FillArrays 1336
10.247.4.12 GenSphericalTexCoord 1336
10.247.4.13 GetIndex 1336
10.247.4.14 GetIndexCount 1337
10.247.4.15 GetMaterialIndex 1337
10.247.4.16 GetMax 1337
10.247.4.17 GetMaxIndex 1337
10.247.4.18 GetMin 1337
10.247.4.19 GetName 1337
10.247.4.20 GetNodeAssignment 1338
10.247.4.21 GetNodeAssignmentsCount 1338
10.247.4.22 GetNormal 1338
10.247.4.23 GetNormalCount 1338
10.247.4.24 GetPrimitiveType 1338
10.247.4.25 GetTexCoord 1339
10.247.4.26 GetTexCoordCount 1339
10.247.4.27 GetVertex 1339
10.247.4.28 GetVertexCount 1339

10.247.4.29 GetVertexIndex 1339
10.247.4.30 HasVertex 1340
10.247.4.31 RecalculateNormals 1340
10.247.4.32 Scale 1340
10.247.4.33 SetIndexCount 1340
10.247.4.34 SetMaterialIndex 1340
10.247.4.35 SetName 1341
10.247.4.36 SetNormal 1341
10.247.4.37 SetNormalCount 1341
10.247.4.38 SetPrimitiveType 1341
10.247.4.39 SetScale 1341
10.247.4.40 SetSubMeshCenter 1342
10.247.4.41 SetTexCoord 1342
10.247.4.42 SetTexCoordCount 1342
10.247.4.43 SetVertex 1342
10.247.4.44 SetVertexCount 1343
10.247.4.45 Translate 1343
10.248 gazebo::transport::SubscribeOptions Class Reference 1343
10.248.1 Detailed Description 1344
10.248.2 Constructor & Destructor Documentation 1344
10.248.2.1 SubscribeOptions 1344
10.248.3 Member Function Documentation 1344
10.248.3.1 GetLatching 1344
10.248.3.2 GetMsgType 1344
10.248.3.3 GetNode 1344
10.248.3.4 GetTopic 1345
10.248.3.5 Init 1345
10.248.3.6 Init 1345
10.249 gazebo::transport::Subscriber Class Reference 1346
10.249.1 Detailed Description 1346
10.249.2 Constructor & Destructor Documentation 1346

10.249.2.1 Subscriber 1346
10.249.2.2 ~Subscriber 1346
10.249.3 Member Function Documentation 1347
10.249.3.1 GetCallbackId 1347
10.249.3.2 GetTopic 1347
10.249.3.3 SetCallbackId 1347
10.249.3.4 Unsubscribe 1347
10.250 gazebo::transport::SubscriptionTransport Class Reference 1347
10.250.1 Detailed Description 1348
10.250.2 Constructor & Destructor Documentation 1348
10.250.2.1 SubscriptionTransport 1348
10.250.2.2 ~SubscriptionTransport 1348
10.250.3 Member Function Documentation 1348
10.250.3.1 GetConnection 1348
10.250.3.2 HandleData 1349
10.250.3.3 HandleMessage 1349
10.250.3.4 Init 1350
10.250.3.5 IsLocal 1350
10.251 gazebo::physics::SurfaceParams Class Reference 1350
10.251.1 Detailed Description 1351
10.251.2 Constructor & Destructor Documentation 1351
10.251.2.1 SurfaceParams 1351
10.251.2.2 ~SurfaceParams 1351
10.251.3 Member Function Documentation 1351
10.251.3.1 FillMsg 1351
10.251.3.2 Load 1351
10.251.3.3 ProcessMsg 1352
10.251.4 Member Data Documentation 1352
10.251.4.1 collideWithoutContact 1352
10.251.4.2 collideWithoutContactBitmask 1352
10.252 gazebo::common::SystemPaths Class Reference 1352

10.252.1 Detailed Description 1354
10.252.2 Member Function Documentation 1355
10.252.2.1 AddGazeboPaths 1355
10.252.2.2 AddModelPaths 1355
10.252.2.3 AddOgrePaths 1355
10.252.2.4 AddPluginPaths 1355
10.252.2.5 AddSearchPathSuffix 1356
10.252.2.6 ClearGazeboPaths 1356
10.252.2.7 ClearModelPaths 1356
10.252.2.8 ClearOgrePaths 1356
10.252.2.9 ClearPluginPaths 1356
10.252.2.10 FindFile 1356
10.252.2.11 FindFileURI 1357
10.252.2.12 GetDefaultTestPath 1357
10.252.2.13 GetGazeboPaths 1357
10.252.2.14 GetLogPath 1357
10.252.2.15 GetModelPaths 1358
10.252.2.16 GetOgrePaths 1358
10.252.2.17 GetPluginPaths 1358
10.252.2.18 GetTmpInstancePath 1358
10.252.2.19 GetTmpPath 1358
10.252.2.20 GetWorldPathExtension 1359
10.252.3 Member Data Documentation 1359
10.252.3.1 gazeboPathsFromEnv 1359
10.252.3.2 modelPathsFromEnv 1359
10.252.3.3 ogrePathsFromEnv 1359
10.252.3.4 pluginPathsFromEnv 1359
10.253 gazebo::SystemPlugin Class Reference 1359
10.253.1 Detailed Description 1360
10.253.2 Constructor & Destructor Documentation 1360
10.253.2.1 SystemPlugin 1361

10.253.2.2 ~SystemPlugin 1361
10.253.3 Member Function Documentation 1361
10.253.3.1 Init 1361
10.253.3.2 Load 1361
10.253.3.3 Reset 1361
10.254 gazebo::common::Time Class Reference 1361
10.254.1 Detailed Description 1366
10.254.2 Constructor & Destructor Documentation 1366
10.254.2.1 Time 1366
10.254.2.2 Time 1366
10.254.2.3 Time 1366
10.254.2.4 Time 1367
10.254.2.5 Time 1367
10.254.2.6 Time 1367
10.254.2.7 ~Time 1367
10.254.3 Member Function Documentation 1367
10.254.3.1 Double 1367
10.254.3.2 Float 1367
10.254.3.3 GetWallTime 1368
10.254.3.4 GetWallTimeAsISOString 1368
10.254.3.5 MicToNano 1368
10.254.3.6 MilToNano 1368
10.254.3.7 MSleep 1369
10.254.3.8 NSleep 1369
10.254.3.9 operator!= 1369
10.254.3.10 operator!= 1370
10.254.3.11 operator!= 1370
10.254.3.12 operator!= 1370
10.254.3.13 operator* 1371
10.254.3.14 operator* 1371
10.254.3.15 operator* 1371

10.254.3.16 operator*= 1371
10.254.3.17 operator*= 1372
10.254.3.18 operator*= 1372
10.254.3.19 operator+ 1372
10.254.3.20 operator+ 1373
10.254.3.21 operator+ 1373
10.254.3.22 operator+= 1373
10.254.3.23 operator+= 1374
10.254.3.24 operator+= 1374
10.254.3.25 operator- 1374
10.254.3.26 operator- 1374
10.254.3.27 operator- 1375
10.254.3.28 operator-= 1375
10.254.3.29 operator-= 1375
10.254.3.30 operator-= 1376
10.254.3.31 operator/ 1376
10.254.3.32 operator/ 1376
10.254.3.33 operator/ 1377
10.254.3.34 operator/= 1377
10.254.3.35 operator/= 1377
10.254.3.36 operator/= 1377
10.254.3.37 operator< 1378
10.254.3.38 operator< 1378
10.254.3.39 operator< 1378
10.254.3.40 operator< 1379
10.254.3.41 operator<= 1379
10.254.3.42 operator<= 1379
10.254.3.43 operator<= 1380
10.254.3.44 operator<= 1380
10.254.3.45 operator= 1380
10.254.3.46 operator= 1380

10.254.3.47 operator= . . . 1381
10.254.3.48 operator== . . . 1381
10.254.3.49 operator== . . . 1381
10.254.3.50 operator== . . . 1382
10.254.3.51 operator== . . . 1382
10.254.3.52 operator> . . . 1382
10.254.3.53 operator> . . . 1383
10.254.3.54 operator> . . . 1383
10.254.3.55 operator> . . . 1383
10.254.3.56 operator>= . . . 1383
10.254.3.57 operator>= . . . 1384
10.254.3.58 operator>= . . . 1384
10.254.3.59 operator>= . . . 1384
10.254.3.60 SecToNano . . . 1385
10.254.3.61 Set . . . 1385
10.254.3.62 Set . . . 1385
10.254.3.63 SetToWallTime . . . 1385
10.254.3.64 Sleep . . . 1386
10.254.4 Friends And Related Function Documentation . . . 1386
10.254.4.1 operator<< . . . 1386
10.254.4.2 operator>> . . . 1386
10.254.5 Member Data Documentation . . . 1387
10.254.5.1 nsec . . . 1387
10.254.5.2 sec . . . 1387
10.254.5.3 Zero . . . 1387
10.255 Gazebo::common::Timer Class Reference . . . 1387
10.255.1 Detailed Description . . . 1388
10.255.2 Constructor & Destructor Documentation . . . 1388
10.255.2.1 Timer . . . 1388
10.255.2.2 ~Timer . . . 1388
10.255.3 Member Function Documentation . . . 1388

10.255.3.1 GetElapsed . . . 1388
10.255.3.2 GetRunning . . . 1389
10.255.3.3 Start . . . 1389
10.255.3.4 Stop . . . 1389
10.255.4 Friends And Related Function Documentation . . . 1389
10.255.4.1 operator<< . . . 1389
10.256 gazebo::transport::TopicManager Class Reference . . . 1389
10.256.1 Detailed Description . . . 1392
10.256.2 Member Typedef Documentation . . . 1392
10.256.2.1 SubNodeMap . . . 1392
10.256.3 Member Function Documentation . . . 1392
10.256.3.1 AddNode . . . 1392
10.256.3.2 AddNodeToProcess . . . 1392
10.256.3.3 Advertise . . . 1392
10.256.3.4 ClearBuffers . . . 1393
10.256.3.5 ConnectPubToSub . . . 1393
10.256.3.6 ConnectSubscribers . . . 1393
10.256.3.7 ConnectSubToPub . . . 1393
10.256.3.8 DisconnectPubFromSub . . . 1394
10.256.3.9 DisconnectSubFromPub . . . 1394
10.256.3.10 FindPublication . . . 1394
10.256.3.11 Fini . . . 1394
10.256.3.12 GetTopicNamespaces . . . 1395
10.256.3.13 Init . . . 1395
10.256.3.14 IsAdvertised . . . 1395
10.256.3.15 PauseIncoming . . . 1395
10.256.3.16 ProcessNodes . . . 1395
10.256.3.17 Publish . . . 1396
10.256.3.18 RegisterTopicNamespace . . . 1396
10.256.3.19 RemoveNode . . . 1396
10.256.3.20 Subscribe . . . 1396

10.256.3.21 Unadvertise 1397
10.256.3.22 Unadvertise 1397
10.256.3.23 Unsubscribe 1397
10.256.3.24 UpdatePublications 1397
10.257 gazebo::physics::TrajectoryInfo Class Reference 1398
10.257.1 Detailed Description 1399
10.257.2 Constructor & Destructor Documentation 1399
10.257.2.1 TrajectoryInfo 1399
10.257.3 Member Data Documentation 1399
10.257.3.1 duration 1399
10.257.3.2 endTime 1399
10.257.3.3 id 1399
10.257.3.4 startTime 1399
10.257.3.5 translated 1399
10.257.3.6 type 1399
10.258 gazebo::rendering::TransmitterVisual Class Reference 1400
10.258.1 Detailed Description 1400
10.258.2 Constructor & Destructor Documentation 1400
10.258.2.1 TransmitterVisual 1401
10.258.2.2 ~TransmitterVisual 1401
10.258.3 Member Function Documentation 1401
10.258.3.1 Load 1401
10.258.3.2 Update 1401
10.259 gazebo::rendering::TransmitterVisualPrivate Class Reference 1401
10.259.1 Detailed Description 1402
10.259.2 Member Data Documentation 1403
10.259.2.1 connections 1403
10.259.2.2 gridMsg 1403
10.259.2.3 isFirst 1403
10.259.2.4 mutex 1403
10.259.2.5 node 1403

10.259.2.6 points . . . 1403
10.259.2.7 receivedMsg . . . 1403
10.259.2.8 signalPropagationSub . . . 1403
10.259.2.9 vectorLink . . . 1404
10.260 gazebo::physics::UniversalJoint< T > Class Template Reference . . . 1404
10.260.1 Detailed Description . . . 1405
10.260.2 Member Enumeration Documentation . . . 1405
10.260.2.1 AxisIndex . . . 1405
10.260.3 Constructor & Destructor Documentation . . . 1405
10.260.3.1 UniversalJoint . . . 1405
10.260.3.2 ~UniversalJoint . . . 1405
10.260.4 Member Function Documentation . . . 1405
10.260.4.1 GetAngleCount . . . 1405
10.260.4.2 Init . . . 1406
10.260.4.3 Load . . . 1406
10.261 gazebo::common::UpdateInfo Class Reference . . . 1406
10.261.1 Detailed Description . . . 1406
10.261.2 Member Data Documentation . . . 1407
10.261.2.1 realTime . . . 1407
10.261.2.2 simTime . . . 1407
10.261.2.3 worldName . . . 1407
10.262 gazebo::rendering::UserCamera Class Reference . . . 1407
10.262.1 Detailed Description . . . 1409
10.262.2 Constructor & Destructor Documentation . . . 1410
10.262.2.1 UserCamera . . . 1410
10.262.2.2 ~UserCamera . . . 1410
10.262.3 Member Function Documentation . . . 1410
10.262.3.1 AnimationComplete . . . 1410
10.262.3.2 AttachToVisualImpl . . . 1410
10.262.3.3 EnableViewController . . . 1411
10.262.3.4 Fini . . . 1411

10.262.3.5 GetAvgFPS . . . 1411
10.262.3.6 GetGUIOverlay . . . 1411
10.262.3.7 GetImageHeight . . . 1412
10.262.3.8 GetImageWidth . . . 1412
10.262.3.9 GetTriangleCount . . . 1412
10.262.3.10 GetViewControllerTypeString . . . 1412
10.262.3.11 GetVisual . . . 1413
10.262.3.12 GetVisual . . . 1413
10.262.3.13 HandleKeyPressEvent . . . 1413
10.262.3.14 HandleKeyReleaseEvent . . . 1413
10.262.3.15 HandleMouseEvent . . . 1414
10.262.3.16 Init . . . 1414
10.262.3.17 InitCameraSetInWorldFile . . . 1414
10.262.3.18 Load . . . 1414
10.262.3.19 Load . . . 1414
10.262.3.20 MoveToPosition . . . 1414
10.262.3.21 MoveToVisual . . . 1415
10.262.3.22 MoveToVisual . . . 1415
10.262.3.23 PostRender . . . 1415
10.262.3.24 Resize . . . 1415
10.262.3.25 SetFocalPoint . . . 1416
10.262.3.26 SetRenderTarget . . . 1416
10.262.3.27 SetUseSDFPose . . . 1416
10.262.3.28 SetViewController . . . 1416
10.262.3.29 SetViewController . . . 1417
10.262.3.30 SetViewportDimensions . . . 1417
10.262.3.31 SetWorldPose . . . 1417
10.262.3.32 TrackVisualImpl . . . 1417
10.262.3.33 Update . . . 1418
10.263 gazebo::rendering::UserCameraPrivate Class Reference . . . 1418
10.263.1 Detailed Description . . . 1419

10.263.2 Member Data Documentation 1419
10.263.2.1 fpsViewController 1419
10.263.2.2 gui 1419
10.263.2.3 isCameraSetInWorldFile 1419
10.263.2.4 orbitViewController 1419
10.263.2.5 selectionBuffer 1419
10.263.2.6 viewController 1419
10.264 gazebo::math::Vector2d Class Reference 1420
10.264.1 Detailed Description 1421
10.264.2 Constructor & Destructor Documentation 1422
10.264.2.1 Vector2d 1422
10.264.2.2 Vector2d 1422
10.264.2.3 Vector2d 1422
10.264.2.4 ~Vector2d 1422
10.264.3 Member Function Documentation 1422
10.264.3.1 Distance 1422
10.264.3.2 Dot 1423
10.264.3.3 IsFinite 1423
10.264.3.4 Normalize 1423
10.264.3.5 operator!= 1423
10.264.3.6 operator* 1423
10.264.3.7 operator* 1424
10.264.3.8 operator*= 1424
10.264.3.9 operator*= 1424
10.264.3.10 operator+ 1425
10.264.3.11 operator+= 1425
10.264.3.12 operator- 1425
10.264.3.13 operator-= 1425
10.264.3.14 operator/ 1426
10.264.3.15 operator/ 1426
10.264.3.16 operator/= 1426

10.264.3.17operator/= 1427
10.264.3.18operator= 1427
10.264.3.19operator= 1427
10.264.3.20operator== 1428
10.264.3.21operator[] 1428
10.264.3.22set 1428
10.264.4Friends And Related Function Documentation 1429
10.264.4.1operator<< 1429
10.264.4.2operator>> 1429
10.264.5Member Data Documentation 1429
10.264.5.1x 1429
10.264.5.2y 1429
10.265gazebo::math::Vector2i Class Reference 1430
10.265.1Detailed Description 1431
10.265.2Constructor & Destructor Documentation 1432
10.265.2.1Vector2i 1432
10.265.2.2Vector2i 1432
10.265.2.3Vector2i 1432
10.265.2.4∼Vector2i 1432
10.265.3Member Function Documentation 1432
10.265.3.1Cross 1432
10.265.3.2Distance 1433
10.265.3.3IsFinite 1433
10.265.3.4Normalize 1433
10.265.3.5operator!= 1433
10.265.3.6operator∗ 1433
10.265.3.7operator∗ 1434
10.265.3.8operator∗= 1434
10.265.3.9operator∗= 1435
10.265.3.10operator+ 1435
10.265.3.11operator+= 1435

10.265.3.12operator- . . . 1435
10.265.3.13operator-= . . . 1436
10.265.3.14operator/ . . . 1436
10.265.3.15operator/ . . . 1436
10.265.3.16operator/= . . . 1437
10.265.3.17operator/= . . . 1437
10.265.3.18operator= . . . 1438
10.265.3.19operator= . . . 1438
10.265.3.20operator== . . . 1438
10.265.3.21operator[] . . . 1439
10.265.3.22set . . . 1439
10.265.4Friends And Related Function Documentation . . . 1439
10.265.4.1operator<< . . . 1439
10.265.4.2operator>> . . . 1439
10.265.5Member Data Documentation . . . 1440
10.265.5.1x . . . 1440
10.265.5.2y . . . 1440
10.266gazebo::math::Vector3 Class Reference . . . 1440
10.266.1Detailed Description . . . 1443
10.266.2Constructor & Destructor Documentation . . . 1444
10.266.2.1Vector3 . . . 1444
10.266.2.2Vector3 . . . 1444
10.266.2.3Vector3 . . . 1444
10.266.2.4~Vector3 . . . 1444
10.266.3Member Function Documentation . . . 1444
10.266.3.1Correct . . . 1444
10.266.3.2Cross . . . 1444
10.266.3.3Distance . . . 1445
10.266.3.4Distance . . . 1445
10.266.3.5Dot . . . 1445
10.266.3.6Equal . . . 1445

10.266.3.7 GetAbs 1446
10.266.3.8 GetDistToLine 1446
10.266.3.9 GetLength 1446
10.266.3.10 GetMax 1446
10.266.3.11 GetMin 1447
10.266.3.12 GetNormal 1447
10.266.3.13 GetPerpendicular 1447
10.266.3.14 GetRounded 1447
10.266.3.15 GetSquaredLength 1448
10.266.3.16 GetSum 1448
10.266.3.17 IsFinite 1448
10.266.3.18 Normalize 1448
10.266.3.19 operator!= 1448
10.266.3.20 operator* 1449
10.266.3.21 operator* 1449
10.266.3.22 operator*= 1449
10.266.3.23 operator*= 1450
10.266.3.24 operator+ 1450
10.266.3.25 operator+= 1450
10.266.3.26 operator- 1450
10.266.3.27 operator- 1451
10.266.3.28 operator-= 1451
10.266.3.29 operator/ 1451
10.266.3.30 operator/ 1451
10.266.3.31 operator/= 1452
10.266.3.32 operator/= 1452
10.266.3.33 operator= 1452
10.266.3.34 operator= 1453
10.266.3.35 operator== 1453
10.266.3.36 operator[] 1453
10.266.3.37 Round 1453

10.266.3.3 Round 1454
10.266.3.3 Set 1454
10.266.3.4 SetToMax 1454
10.266.3.4 SetToMin 1454
10.266.4 Friends And Related Function Documentation 1454
10.266.4.1 operator* 1454
10.266.4.2 operator<< 1455
10.266.4.3 operator>> 1455
10.266.5 Member Data Documentation 1455
10.266.5.1 One 1455
10.266.5.2 UnitX 1456
10.266.5.3 UnitY 1456
10.266.5.4 UnitZ 1456
10.266.5.5 x 1456
10.266.5.6 y 1456
10.266.5.7 z 1456
10.266.5.8 Zero 1456
10.267 gazebo::math::Vector4 Class Reference 1457
10.267.1 Detailed Description 1459
10.267.2 Constructor & Destructor Documentation 1459
10.267.2.1 Vector4 1459
10.267.2.2 Vector4 1459
10.267.2.3 Vector4 1459
10.267.2.4 ~Vector4 1460
10.267.3 Member Function Documentation 1460
10.267.3.1 Distance 1460
10.267.3.2 GetLength 1460
10.267.3.3 GetSquaredLength 1460
10.267.3.4 IsFinite 1460
10.267.3.5 Normalize 1460
10.267.3.6 operator!= 1461

10.267.3.7 operator* . . . 1461
10.267.3.8 operator* . . . 1461
10.267.3.9 operator* . . . 1462
10.267.3.10 operator*= . . . 1462
10.267.3.11 operator*= . . . 1462
10.267.3.12 operator+ . . . 1463
10.267.3.13 operator+= . . . 1463
10.267.3.14 operator- . . . 1463
10.267.3.15 operator-= . . . 1463
10.267.3.16 operator/ . . . 1464
10.267.3.17 operator/ . . . 1464
10.267.3.18 operator/= . . . 1465
10.267.3.19 operator/= . . . 1465
10.267.3.20 operator= . . . 1465
10.267.3.21 operator= . . . 1466
10.267.3.22 operator== . . . 1466
10.267.3.23 operator[] . . . 1466
10.267.3.24 set . . . 1466
10.267.4 Friends And Related Function Documentation . . . 1466
10.267.4.1 operator<< . . . 1467
10.267.4.2 operator>> . . . 1467
10.267.5 Member Data Documentation . . . 1467
10.267.5.1 w . . . 1467
10.267.5.2 x . . . 1467
10.267.5.3 y . . . 1467
10.267.5.4 z . . . 1468
10.268 gazebo::common::Video Class Reference . . . 1468
10.268.1 Detailed Description . . . 1468
10.268.2 Constructor & Destructor Documentation . . . 1468
10.268.2.1 Video . . . 1468
10.268.2.2 ~Video . . . 1469

10.268.3 Member Function Documentation 1469
10.268.3.1 GetHeight 1469
10.268.3.2 GetNextFrame 1469
10.268.3.3 GetWidth 1469
10.268.3.4 Load 1469
10.269 gazebo::rendering::VideoVisual Class Reference 1470
10.269.1 Detailed Description 1470
10.269.2 Constructor & Destructor Documentation 1471
10.269.2.1 VideoVisual 1471
10.269.2.2 ~VideoVisual 1471
10.270 gazebo::rendering::VideoVisualPrivate Class Reference 1471
10.270.1 Detailed Description 1472
10.270.2 Member Data Documentation 1472
10.270.2.1 connections 1472
10.270.2.2 height 1472
10.270.2.3 imageBuffer 1472
10.270.2.4 texture 1472
10.270.2.5 video 1473
10.270.2.6 width 1473
10.271 gazebo::rendering::ViewController Class Reference 1473
10.271.1 Detailed Description 1474
10.271.2 Constructor & Destructor Documentation 1474
10.271.2.1 ViewController 1474
10.271.2.2 ~ViewController 1475
10.271.3 Member Function Documentation 1475
10.271.3.1 GetTypeString 1475
10.271.3.2 HandleKeyPressEvent 1475
10.271.3.3 HandleKeyReleaseEvent 1475
10.271.3.4 HandleMouseEvent 1476
10.271.3.5 Init 1476
10.271.3.6 Init 1476

10.271.3.7 SetEnabled . . . 1476
10.271.3.8 Update . . . 1476
10.271.4 Member Data Documentation . . . 1477
10.271.4.1 camera . . . 1477
10.271.4.2 enabled . . . 1477
10.271.4.3 typeString . . . 1477
10.272 gazebo::rendering::Visual Class Reference . . . 1477
10.272.1 Detailed Description . . . 1484
10.272.2 Constructor & Destructor Documentation . . . 1484
10.272.2.1 Visual . . . 1484
10.272.2.2 Visual . . . 1484
10.272.2.3 ~Visual . . . 1485
10.272.2.4 Visual . . . 1485
10.272.2.5 Visual . . . 1485
10.272.3 Member Function Documentation . . . 1485
10.272.3.1 AttachAxes . . . 1485
10.272.3.2 AttachLineVertex . . . 1485
10.272.3.3 AttachMesh . . . 1485
10.272.3.4 AttachObject . . . 1486
10.272.3.5 AttachVisual . . . 1486
10.272.3.6 ClearParent . . . 1486
10.272.3.7 Clone . . . 1486
10.272.3.8 CreateDynamicLine . . . 1486
10.272.3.9 DeleteDynamicLine . . . 1487
10.272.3.10 DetachObjects . . . 1487
10.272.3.11 DetachVisual . . . 1487
10.272.3.12 DetachVisual . . . 1487
10.272.3.13 DisableTrackVisual . . . 1487
10.272.3.14 EnableTrackVisual . . . 1488
10.272.3.15 Fini . . . 1488
10.272.3.16 GetAttachedObjectCount . . . 1488

10.272.3.17 GetBoundingBox 1488
10.272.3.18 GetChild 1488
10.272.3.19 GetChildCount 1489
10.272.3.20 GetHighlighted 1489
10.272.3.21 GetId 1489
10.272.3.22 GetMaterialName 1489
10.272.3.23 GetMeshName 1489
10.272.3.24 GetName 1490
10.272.3.25 GetNormalMap 1490
10.272.3.26 GetParent 1490
10.272.3.27 GetPose 1490
10.272.3.28 GetPosition 1490
10.272.3.29 GetRootVisual 1491
10.272.3.30 GetRotation 1491
10.272.3.31 GetScale 1491
10.272.3.32 GetScene 1491
10.272.3.33 GetSceneNode 1491
10.272.3.34 GetShaderType 1492
10.272.3.35 GetSubMeshName 1492
10.272.3.36 GetTransparency 1492
10.272.3.37 GetVisibilityFlags 1492
10.272.3.38 GetVisible 1493
10.272.3.39 GetWorldPose 1493
10.272.3.40 HasAttachedObject 1493
10.272.3.41 Init 1493
10.272.3.42 InsertMesh 1493
10.272.3.43 InsertMesh 1494
10.272.3.44 IsPlane 1494
10.272.3.45 IsStatic 1494
10.272.3.46 Load 1494
10.272.3.47 Load 1495

10.272.3.48 LoadFromMsg 1495
10.272.3.49 LoadPlugin 1495
10.272.3.50 MakeStatic 1495
10.272.3.51 MoveToPosition 1495
10.272.3.52 MoveToPositions 1496
10.272.3.53 RemovePlugin 1496
10.272.3.54 SetAmbient 1496
10.272.3.55 SetCastShadows 1496
10.272.3.56 SetDiffuse 1497
10.272.3.57 SetEmissive 1497
10.272.3.58 SetHighlighted 1497
10.272.3.59 SetId 1497
10.272.3.60 SetLighting 1497
10.272.3.61 SetMaterial 1498
10.272.3.62 SetName 1498
10.272.3.63 SetNormalMap 1498
10.272.3.64 SetPose 1498
10.272.3.65 SetPosition 1499
10.272.3.66 SetRibbonTrail 1499
10.272.3.67 SetRotation 1499
10.272.3.68 SetScale 1499
10.272.3.69 SetScene 1500
10.272.3.70 SetShaderType 1500
10.272.3.71 SetSkeletonPose 1500
10.272.3.72 SetSpecular 1500
10.272.3.73 SetTransparency 1500
10.272.3.74 SetVisibilityFlags 1501
10.272.3.75 SetVisible 1501
10.272.3.76 SetWireframe 1501
10.272.3.77 SetWorldPose 1501
10.272.3.78 SetWorldPosition 1502

10.272.3.79 SetWorldRotation 1502
10.272.3.80 ShowBoundingBox 1502
10.272.3.81 ShowCollision 1502
10.272.3.82 ShowCOM 1502
10.272.3.83 ShowJoints 1503
10.272.3.84 ShowSkeleton 1503
10.272.3.85 ToggleVisible 1503
10.272.3.86 Update 1503
10.272.3.87 UpdateFromMsg 1503
10.272.4 Member Data Documentation 1503
10.272.4.1 dataPtr 1504
10.273 gazebo::VisualPlugin Class Reference 1504
10.273.1 Detailed Description 1504
10.273.2 Constructor & Destructor Documentation 1505
10.273.2.1 VisualPlugin 1505
10.273.3 Member Function Documentation 1505
10.273.3.1 Init 1505
10.273.3.2 Load 1505
10.273.3.3 Reset 1505
10.274 gazebo::rendering::VisualPrivate Class Reference 1505
10.274.1 Detailed Description 1508
10.274.2 Member Data Documentation 1508
10.274.2.1 animState 1508
10.274.2.2 boundingBox 1508
10.274.2.3 children 1508
10.274.2.4 id 1508
10.274.2.5 initialized 1509
10.274.2.6 isStatic 1509
10.274.2.7 lighting 1509
10.274.2.8 lines 1509
10.274.2.9 lineVertices 1509

10.274.2.10 myMaterialName . . . 1509
10.274.2.11 name . . . 1509
10.274.2.12 onAnimationComplete . . . 1509
10.274.2.13 origMaterialName . . . 1509
10.274.2.14 parent . . . 1510
10.274.2.15 plugins . . . 1510
10.274.2.16 preRenderConnection . . . 1510
10.274.2.17 prevAnimTime . . . 1510
10.274.2.18 ribbonTrail . . . 1510
10.274.2.19 scale . . . 1510
10.274.2.20 scene . . . 1510
10.274.2.21 sceneNode . . . 1510
10.274.2.22 sdf . . . 1510
10.274.2.23 skeleton . . . 1511
10.274.2.24 staticGeom . . . 1511
10.274.2.25 transparency . . . 1511
10.274.2.26 useRTShader . . . 1511
10.274.2.27 visible . . . 1511
10.274.2.28 visualIdCount . . . 1511
10.275 gazebo::rendering::WindowManager Class Reference . . . 1511
10.275.1 Detailed Description . . . 1512
10.275.2 Constructor & Destructor Documentation . . . 1512
10.275.2.1 WindowManager . . . 1512
10.275.2.2 ~WindowManager . . . 1512
10.275.3 Member Function Documentation . . . 1512
10.275.3.1 CreateWindow . . . 1512
10.275.3.2 Fini . . . 1513
10.275.3.3 GetAvgFPS . . . 1513
10.275.3.4 GetTriangleCount . . . 1513
10.275.3.5 GetWindow . . . 1513
10.275.3.6 Moved . . . 1514

10.275.3.7 Resize . . . 1514
10.275.3.8 SetCamera . . . 1514
10.276 gazebo::rendering::WireBox Class Reference . . . 1515
10.276.1 Detailed Description . . . 1515
10.276.2 Constructor & Destructor Documentation . . . 1515
10.276.2.1 WireBox . . . 1515
10.276.2.2 ~WireBox . . . 1515
10.276.3 Member Function Documentation . . . 1516
10.276.3.1 GetBox . . . 1516
10.276.3.2 GetVisible . . . 1516
10.276.3.3 Init . . . 1516
10.276.3.4 SetVisible . . . 1516
10.277 gazebo::rendering::WireBoxPrivate Class Reference . . . 1516
10.277.1 Detailed Description . . . 1517
10.277.2 Member Data Documentation . . . 1517
10.277.2.1 lines . . . 1517
10.277.2.2 parent . . . 1517
10.278 gazebo::sensors::WirelessReceiver Class Reference . . . 1517
10.278.1 Detailed Description . . . 1519
10.278.2 Constructor & Destructor Documentation . . . 1519
10.278.2.1 WirelessReceiver . . . 1519
10.278.2.2 ~WirelessReceiver . . . 1519
10.278.3 Member Function Documentation . . . 1519
10.278.3.1 Fini . . . 1519
10.278.3.2 GetMaxFreqFiltered . . . 1519
10.278.3.3 GetMinFreqFiltered . . . 1519
10.278.3.4 GetSensitivity . . . 1520
10.278.3.5 Init . . . 1520
10.278.3.6 Load . . . 1520
10.279 gazebo::sensors::WirelessTransceiver Class Reference . . . 1520
10.279.1 Detailed Description . . . 1522

10.279.2 Constructor & Destructor Documentation 1522
10.279.2.1 WirelessTransceiver 1522
10.279.2.2 ~WirelessTransceiver 1522
10.279.3 Member Function Documentation 1522
10.279.3.1 Fini 1522
10.279.3.2 GetGain 1522
10.279.3.3 GetPower 1523
10.279.3.4 GetTopic 1523
10.279.3.5 Init 1523
10.279.3.6 Load 1523
10.279.4 Member Data Documentation 1524
10.279.4.1 gain 1524
10.279.4.2 parentEntity 1524
10.279.4.3 power 1524
10.279.4.4 pub 1524
10.279.4.5 referencePose 1524
10.280 gazebo::sensors::WirelessTransmitter Class Reference 1524
10.280.1 Detailed Description 1526
10.280.2 Constructor & Destructor Documentation 1526
10.280.2.1 WirelessTransmitter 1526
10.280.2.2 ~WirelessTransmitter 1526
10.280.3 Member Function Documentation 1527
10.280.3.1 GetESSID 1527
10.280.3.2 GetFreq 1527
10.280.3.3 GetSignalStrength 1527
10.280.3.4 Init 1527
10.280.3.5 Load 1527
10.280.3.6 UpdateImpl 1528
10.280.4 Member Data Documentation 1528
10.280.4.1 freq 1528
10.280.4.2 ModelStdDesv 1528

10.280.4.3 NEmpty . . . 1528
10.280.4.4 NObstacle . . . 1528
10.281 gazebo::physics::World Class Reference . . . 1529
10.281.1 Detailed Description . . . 1532
10.281.2 Constructor & Destructor Documentation . . . 1532
10.281.2.1 World . . . 1532
10.281.2.2 ~World . . . 1532
10.281.3 Member Function Documentation . . . 1532
10.281.3.1 Clear . . . 1532
10.281.3.2 ClearModels . . . 1533
10.281.3.3 DisableAllModels . . . 1533
10.281.3.4 EnableAllModels . . . 1533
10.281.3.5 EnablePhysicsEngine . . . 1533
10.281.3.6 Fini . . . 1533
10.281.3.7 GetByName . . . 1533
10.281.3.8 GetEnablePhysicsEngine . . . 1534
10.281.3.9 GetEntity . . . 1534
10.281.3.10 GetEntityBelowPoint . . . 1534
10.281.3.11 GetIterations . . . 1535
10.281.3.12 GetModel . . . 1535
10.281.3.13 GetModel . . . 1535
10.281.3.14 GetModelBelowPoint . . . 1535
10.281.3.15 GetModelCount . . . 1536
10.281.3.16 GetModels . . . 1536
10.281.3.17 GetName . . . 1536
10.281.3.18 GetPauseTime . . . 1536
10.281.3.19 GetPhysicsEngine . . . 1537
10.281.3.20 GetRealTime . . . 1537
10.281.3.21 GetRunning . . . 1537
10.281.3.22 GetSceneMsg . . . 1537
10.281.3.23 GetSelectedEntity . . . 1537

10.281.3.24 GetSetWorldPoseMutex 1538
10.281.3.25 GetSimTime 1538
10.281.3.26 GetSphericalCoordinates 1538
10.281.3.27 GetStartTime 1538
10.281.3.28 Init 1538
10.281.3.29 InsertModelFile 1539
10.281.3.30 InsertModelSDF 1539
10.281.3.31 InsertModelString 1539
10.281.3.32 IsLoaded 1539
10.281.3.33 IsPaused 1539
10.281.3.34 Load 1540
10.281.3.35 LoadPlugin 1540
10.281.3.36 PrintEntityTree 1540
10.281.3.37 PublishModelPose 1540
10.281.3.38 RemovePlugin 1541
10.281.3.39 Reset 1541
10.281.3.40 ResetEntities 1541
10.281.3.41 ResetTime 1541
10.281.3.42 Run 1541
10.281.3.43 RunBlocking 1541
10.281.3.44 Save 1542
10.281.3.45 SetPaused 1542
10.281.3.46 SetSimTime 1542
10.281.3.47 SetState 1542
10.281.3.48 Step 1543
10.281.3.49 Stop 1543
10.281.3.50 StripWorldName 1543
10.281.3.51 UpdateStateSDF 1543
10.281.4 Member Data Documentation 1543
10.281.4.1 dirtyPoses 1543
10.282 gazebo::WorldPlugin Class Reference 1544

10.282.1 Detailed Description 1544
10.282.2 Constructor & Destructor Documentation 1544
10.282.2.1 WorldPlugin 1545
10.282.2.2 ~WorldPlugin 1545
10.282.3 Member Function Documentation 1545
10.282.3.1 Init 1545
10.282.3.2 Load 1545
10.282.3.3 Reset 1545
10.283 gazebo::physics::WorldState Class Reference 1545
10.283.1 Detailed Description 1547
10.283.2 Constructor & Destructor Documentation 1547
10.283.2.1 WorldState 1547
10.283.2.2 WorldState 1548
10.283.2.3 WorldState 1548
10.283.2.4 ~WorldState 1548
10.283.3 Member Function Documentation 1548
10.283.3.1 FillSDF 1548
10.283.3.2 GetModelState 1548
10.283.3.3 GetModelStateCount 1549
10.283.3.4 GetModelStates 1549
10.283.3.5 GetModelStates 1549
10.283.3.6 HasModelState 1550
10.283.3.7 IsZero 1550
10.283.3.8 Load 1550
10.283.3.9 Load 1550
10.283.3.10 operator+ 1551
10.283.3.11 operator- 1551
10.283.3.12 operator= 1551
10.283.3.13 SetRealTime 1552
10.283.3.14 SetSimTime 1552
10.283.3.15 SetWallTime 1552

10.283.3.15 SetWorld . . . 1552
10.283.4 Friends And Related Function Documentation . . . 1553
10.283.4.1 operator<< . . . 1553
10.284 gazebo::rendering::WrenchVisual Class Reference . . . 1553
10.284.1 Detailed Description . . . 1554
10.284.2 Constructor & Destructor Documentation . . . 1554
10.284.2.1 WrenchVisual . . . 1554
10.284.2.2 ~WrenchVisual . . . 1554
10.284.3 Member Function Documentation . . . 1554
10.284.3.1 Load . . . 1554
10.284.3.2 SetEnabled . . . 1555
10.285 gazebo::rendering::WrenchVisualPrivate Class Reference . . . 1555
10.285.1 Detailed Description . . . 1556
10.285.2 Member Data Documentation . . . 1556
10.285.2.1 coneXNode . . . 1556
10.285.2.2 coneYNode . . . 1556
10.285.2.3 coneZNode . . . 1556
10.285.2.4 connections . . . 1557
10.285.2.5 enabled . . . 1557
10.285.2.6 forceLine . . . 1557
10.285.2.7 forceNode . . . 1557
10.285.2.8 mutex . . . 1557
10.285.2.9 node . . . 1557
10.285.2.10 receivedMsg . . . 1557
10.285.2.11 wrenchMsg . . . 1557
10.285.2.12 wrenchSub . . . 1557

11 File Documentation **1559**

11.1 Actor.hh File Reference . . . 1559
11.2 Angle.hh File Reference . . . 1561
11.2.1 Define Documentation . . . 1562

11.2.1.1 GZ_DTOR 1562
11.2.1.2 GZ_NORMALIZE 1562
11.2.1.3 GZ_RTOD 1562
11.3 Animation.hh File Reference 1563
11.4 ArrowVisual.hh File Reference 1565
11.5 ArrowVisualPrivate.hh File Reference 1565
11.6 Assert.hh File Reference 1566
11.6.1 Define Documentation 1567
11.6.1.1 GZ_ASSERT 1567
11.7 AudioDecoder.hh File Reference 1567
11.8 AxisVisual.hh File Reference 1569
11.9 AxisVisualPrivate.hh File Reference 1570
11.10 BallJoint.hh File Reference 1570
11.11 Base.hh File Reference 1571
11.12 Base64.hh File Reference 1573
11.12.1 Function Documentation 1574
11.12.1.1 Base64Decode 1574
11.12.1.2 Base64Encode 1575
11.13 Box.hh File Reference 1575
11.14 BoxShape.hh File Reference 1576
11.15 BVHLoader.hh File Reference 1577
11.15.1 Define Documentation 1579
11.15.1.1 X_POSITION 1579
11.15.1.2 X_ROTATION 1579
11.15.1.3 Y_POSITION 1579
11.15.1.4 Y_ROTATION 1579
11.15.1.5 Z_POSITION 1579
11.15.1.6 Z_ROTATION 1579
11.16 CallbackHelper.hh File Reference 1579
11.17 Camera.hh File Reference 1581
11.18 CameraPrivate.hh File Reference 1582

11.19CameraSensor.hh File Reference 1583
11.20CameraVisual.hh File Reference 1584
11.21CameraVisualPrivate.hh File Reference 1584
11.22cegui.h File Reference 1585
11.23ColladaExporter.hh File Reference 1586
11.24ColladaExporterPrivate.hh File Reference 1586
11.25ColladaLoader.hh File Reference 1587
11.26ColladaLoaderPrivate.hh File Reference 1589
11.27Collision.hh File Reference 1590
11.28CollisionState.hh File Reference 1591
11.29Color.hh File Reference 1591
11.30CommonIface.hh File Reference 1592
11.31CommonTypes.hh File Reference 1594
11.31.1 Detailed Description 1596
11.31.2 Define Documentation 1596
11.31.2.1 GAZEBO_DEPRECATED 1596
11.31.2.2 GAZEBO_FORCEINLINE 1596
11.31.2.3 NULL 1596
11.32COMVisual.hh File Reference 1596
11.33COMVisualPrivate.hh File Reference 1597
11.34Connection.hh File Reference 1598
11.34.1 Define Documentation 1600
11.34.1.1 HEADER_LENGTH 1600
11.35ConnectionManager.hh File Reference 1600
11.36Console.hh File Reference 1601
11.37Contact.hh File Reference 1603
11.37.1 Define Documentation 1605
11.37.1.1 MAX_COLLIDE_RETURNS 1605
11.37.1.2 MAX_CONTACT_JOINTS 1605
11.38ContactManager.hh File Reference 1605
11.39ContactSensor.hh File Reference 1606

11.40 ContactVisual.hh File Reference 1607
11.41 ContactVisualPrivate.hh File Reference 1607
11.42 Conversions.hh File Reference 1608
11.43 CylinderShape.hh File Reference 1609
11.44 dart_inc.h File Reference 1611
11.45 DARTBallJoint.hh File Reference 1612
11.46 DARTBoxShape.hh File Reference 1612
11.47 DARTCollision.hh File Reference 1613
11.48 DARTCylinderShape.hh File Reference 1614
11.49 DARTHeightmapShape.hh File Reference 1614
11.50 DARTHinge2Joint.hh File Reference 1615
11.51 DARTHingeJoint.hh File Reference 1616
11.52 DARTJoint.hh File Reference 1616
11.53 DARTLink.hh File Reference 1617
11.54 DARTMeshShape.hh File Reference 1618
11.55 DARTModel.hh File Reference 1618
11.56 DARTMultiRayShape.hh File Reference 1619
11.57 DARTPhysics.hh File Reference 1620
11.58 DARTPlaneShape.hh File Reference 1620
11.59 DARTRayShape.hh File Reference 1621
11.60 DARTScrewJoint.hh File Reference 1622
11.61 DARTSliderJoint.hh File Reference 1622
11.62 DARTSphereShape.hh File Reference 1623
11.63 DARTTypes.hh File Reference 1623
11.63.1 Detailed Description 1625
11.64 DARTUniversalJoint.hh File Reference 1625
11.65 Dem.hh File Reference 1626
11.66 DemPrivate.hh File Reference 1626
11.67 DepthCamera.hh File Reference 1628
11.68 DepthCameraSensor.hh File Reference 1628
11.69 Diagnostics.hh File Reference 1629

11.70 DynamicLines.hh File Reference 1630
11.71 DynamicRenderable.hh File Reference 1631
11.72 Entity.hh File Reference 1632
11.73 Event.hh File Reference 1633
11.74 Events.hh File Reference 1634
11.75 Exception.hh File Reference 1635
11.76 ffmpeg_inc.h File Reference 1636
11.77 Filter.hh File Reference 1637
11.78 ForceTorqueSensor.hh File Reference 1639
11.79 FPSViewController.hh File Reference 1639
11.80 GaussianNoiseModel.hh File Reference 1640
11.81 gazebo.hh File Reference 1641
11.82 gazebo_core.hh File Reference 1643
11.83 GazeboGenerator.hh File Reference 1644
11.84 GearboxJoint.hh File Reference 1645
11.85 GpsSensor.hh File Reference 1646
11.86 GpuLaser.hh File Reference 1647
11.87 GpuRaySensor.hh File Reference 1648
11.88 Grid.hh File Reference 1649
11.89 Gripper.hh File Reference 1649
11.90 GUIOverlay.hh File Reference 1650
11.91 GUIOverlayPrivate.hh File Reference 1651
11.92 Heightmap.hh File Reference 1652
11.93 HeightmapData.hh File Reference 1653
11.94 HeightmapShape.hh File Reference 1654
11.95 Helpers.hh File Reference 1655
11.95.1 Define Documentation 1657
11.95.1.1 GZ_DBL_MAX 1657
11.95.1.2 GZ_DBL_MIN 1657
11.95.1.3 GZ_FLT_MAX 1658
11.95.1.4 GZ_FLT_MIN 1658

11.95.1.5 GZ_INT32_MAX . . . 1658
11.95.1.6 GZ_INT32_MIN . . . 1658
11.95.1.7 GZ_UINT32_MAX . . . 1658
11.95.1.8 GZ_UINT32_MIN . . . 1658
11.96 Hinge2Joint.hh File Reference . . . 1658
11.97 HingeJoint.hh File Reference . . . 1659
11.98 Image.hh File Reference . . . 1661
11.99 ImageHeightmap.hh File Reference . . . 1662
11.100 ImuSensor.hh File Reference . . . 1663
11.101 Inertial.hh File Reference . . . 1663
11.102 IOManager.hh File Reference . . . 1664
11.103 Joint.hh File Reference . . . 1666
11.103.1 Define Documentation . . . 1667
11.103.1.1 MAX_JOINT_AXIS . . . 1667
11.104 JointController.hh File Reference . . . 1667
11.105 JointControllerPrivate.hh File Reference . . . 1668
11.106 JointState.hh File Reference . . . 1669
11.107 JointVisual.hh File Reference . . . 1670
11.108 JointVisualPrivate.hh File Reference . . . 1671
11.109 JointWrench.hh File Reference . . . 1671
11.110 KeyEvent.hh File Reference . . . 1672
11.111 KeyFrame.hh File Reference . . . 1674
11.112 LaserVisual.hh File Reference . . . 1676
11.113 LaserVisualPrivate.hh File Reference . . . 1676
11.114 Light.hh File Reference . . . 1677
11.115 Link.hh File Reference . . . 1678
11.116 LinkState.hh File Reference . . . 1679
11.117 LogPlay.hh File Reference . . . 1680
11.118 LogRecord.hh File Reference . . . 1681
11.118.1 Define Documentation . . . 1681
11.118.1.1 GZ_LOG_VERSION . . . 1681

11.119 mainpage.html File Reference 1682
11.120 MapShape.hh File Reference 1682
11.121 Master.hh File Reference 1682
11.122 Material.hh File Reference 1684
11.123 Material.hh File Reference 1685
11.124 MathTypes.hh File Reference 1685
11.124.1 Detailed Description 1686
11.125 Matrix3.hh File Reference 1686
11.126 Matrix4.hh File Reference 1687
11.127 Mesh.hh File Reference 1688
11.128 MeshCSG.hh File Reference 1689
11.128.1 Typedef Documentation 1690
11.128.1.1 GPtrArray 1690
11.128.1.2 GtsSurface 1690
11.129 MeshExporter.hh File Reference 1690
11.130 MeshLoader.hh File Reference 1691
11.131 MeshManager.hh File Reference 1694
11.132 MeshShape.hh File Reference 1695
11.133 Model.hh File Reference 1697
11.134 ModelDatabase.hh File Reference 1698
11.134.1 Define Documentation 1700
11.134.1.1 GZ_MODEL_DB_MANIFEST_FILENAME 1700
11.134.1.2 GZ_MODEL_MANIFEST_FILENAME 1700
11.135 ModelDatabasePrivate.hh File Reference 1700
11.136 ModelState.hh File Reference 1701
11.137 MouseEvent.hh File Reference 1702
11.138 MovableText.hh File Reference 1703
11.139 MovingWindowFilter.hh File Reference 1704
11.140 MsgFactory.hh File Reference 1706
11.141 msgs.hh File Reference 1707
11.142 MultiCameraSensor.hh File Reference 1710

11.143MultiRayShape.hh File Reference 1711
11.144Node.hh File Reference 1712
11.145Noise.hh File Reference 1713
11.146OculusCamera.hh File Reference 1714
11.147ogre_gazebo.h File Reference 1715
11.148OpenAL.hh File Reference 1716
11.149OrbitViewController.hh File Reference 1717
11.150PhysicsEngine.hh File Reference 1718
11.151PhysicsFactory.hh File Reference 1718
11.152PhysicsIface.hh File Reference 1720
11.153PhysicsTypes.hh File Reference 1722
11.153.1Detailed Description 1725
11.153.2Define Documentation 1725
11.153.2.1GZ_ALL_COLLIDE 1725
11.153.2.2GZ_FIXED_COLLIDE 1725
11.153.2.3GZ_GHOST_COLLIDE 1725
11.153.2.4GZ_NONE_COLLIDE 1725
11.153.2.5GZ_SENSOR_COLLIDE 1725
11.154PID.hh File Reference 1726
11.155Plane.hh File Reference 1727
11.156PlaneShape.hh File Reference 1728
11.157Plugin.hh File Reference 1729
11.157.1Define Documentation 1731
11.157.1.1GZ_REGISTER_MODEL_PLUGIN 1731
11.157.1.2GZ_REGISTER_SENSOR_PLUGIN 1732
11.157.1.3GZ_REGISTER_SYSTEM_PLUGIN 1732
11.157.1.4GZ_REGISTER_VISUAL_PLUGIN 1733
11.157.1.5GZ_REGISTER_WORLD_PLUGIN 1733
11.158Pose.hh File Reference 1733
11.159Projector.hh File Reference 1734
11.160Publication.hh File Reference 1735

11.161 PublicationTransport.hh File Reference 1736
11.162 Publisher.hh File Reference 1738
11.163 Quaternion.hh File Reference 1739
11.164 Rand.hh File Reference 1740
11.165 RaySensor.hh File Reference 1742
11.166 RayShape.hh File Reference 1743
11.167 RenderEngine.hh File Reference 1744
11.168 RenderEvents.hh File Reference 1744
11.169 RenderingIface.hh File Reference 1745
11.170 RenderTypes.hh File Reference 1746
11.170.1 Define Documentation 1748
11.170.1.1 GZ_VISIBILITY_ALL 1748
11.170.1.2 GZ_VISIBILITY_GUI 1748
11.170.1.3 GZ_VISIBILITY_SELECTABLE 1748
11.170.1.4 GZ_VISIBILITY_SELECTION 1749
11.171 RFIDSensor.hh File Reference 1749
11.172 RFIDTag.hh File Reference 1749
11.173 RFIDTagVisual.hh File Reference 1750
11.174 RFIDTagVisualPrivate.hh File Reference 1751
11.175 RFIDVisual.hh File Reference 1751
11.176 RFIDVisualPrivate.hh File Reference 1752
11.177 Road.hh File Reference 1753
11.178 Road2d.hh File Reference 1754
11.179 RotationSpline.hh File Reference 1755
11.180 RTShaderSystem.hh File Reference 1756
11.181 Scene.hh File Reference 1757
11.182 ScrewJoint.hh File Reference 1758
11.183 SelectionObj.hh File Reference 1760
11.184 SelectionObjPrivate.hh File Reference 1761
11.185 Sensor.hh File Reference 1761
11.186 SensorFactory.hh File Reference 1763

11.187SensorManager.hh File Reference 1764
11.188SensorsIface.hh File Reference 1765
11.189SensorTypes.hh File Reference 1766
11.189.1Detailed Description 1768
11.190Server.hh File Reference 1768
11.191Shape.hh File Reference 1770
11.192simbody_inc.h File Reference 1771
11.193SimbodyBallJoint.hh File Reference 1771
11.194SimbodyBoxShape.hh File Reference 1772
11.195SimbodyCollision.hh File Reference 1773
11.196SimbodyCylinderShape.hh File Reference 1773
11.197SimbodyHeightmapShape.hh File Reference 1774
11.198SimbodyHinge2Joint.hh File Reference 1775
11.199SimbodyHingeJoint.hh File Reference 1775
11.200SimbodyJoint.hh File Reference 1776
11.201SimbodyLink.hh File Reference 1777
11.202SimbodyMeshShape.hh File Reference 1777
11.203SimbodyModel.hh File Reference 1778
11.204SimbodyMultiRayShape.hh File Reference 1779
11.205SimbodyPhysics.hh File Reference 1779
11.206SimbodyPlaneShape.hh File Reference 1780
11.207SimbodyRayShape.hh File Reference 1781
11.208SimbodyScrewJoint.hh File Reference 1782
11.209SimbodySliderJoint.hh File Reference 1783
11.210SimbodySphereShape.hh File Reference 1783
11.211SimbodyTypes.hh File Reference 1784
11.211.1Detailed Description 1785
11.212SimbodyUniversalJoint.hh File Reference 1785
11.213SingletonT.hh File Reference 1786
11.214Skeleton.hh File Reference 1787
11.215SkeletonAnimation.hh File Reference 1789

11.216SliderJoint.hh File Reference 1791
11.217SonarSensor.hh File Reference 1792
11.218SonarVisual.hh File Reference 1792
11.219SonarVisualPrivate.hh File Reference 1793
11.220SphereShape.hh File Reference 1794
11.221SphericalCoordinates.hh File Reference 1795
11.222SphericalCoordinatesPrivate.hh File Reference 1796
11.223Spline.hh File Reference 1797
11.224State.hh File Reference 1799
11.225STLLoader.hh File Reference 1800
11.225.1Define Documentation 1802
11.225.1.1COR3_MAX 1802
11.225.1.2FACE_MAX 1802
11.225.1.3LINE_MAX_LEN 1802
11.225.1.4ORDER_MAX 1802
11.226SubscribeOptions.hh File Reference 1802
11.227Subscriber.hh File Reference 1803
11.228SubscriptionTransport.hh File Reference 1805
11.229SurfaceParams.hh File Reference 1806
11.230System.hh File Reference 1807
11.230.1Define Documentation 1807
11.230.1.1GAZEBO_HIDDEN 1807
11.230.1.2GAZEBO_VISIBLE 1807
11.231SystemPaths.hh File Reference 1807
11.231.1Define Documentation 1809
11.231.1.1GetCurrentDir 1809
11.231.1.2LINUX 1809
11.232Time.hh File Reference 1809
11.233Timer.hh File Reference 1810
11.234TopicManager.hh File Reference 1811
11.235TransmitterVisual.hh File Reference 1813

11.236TransmitterVisualPrivate.hh File Reference 1813
11.237TransportIface.hh File Reference 1814
11.238TransportTypes.hh File Reference 1817
11.238.1Detailed Description 1818
11.239UniversalJoint.hh File Reference 1818
11.240UpdateInfo.hh File Reference 1819
11.241UserCamera.hh File Reference 1820
11.242UserCameraPrivate.hh File Reference 1821
11.243UtilTypes.hh File Reference 1822
11.243.1Detailed Description 1823
11.244Vector2d.hh File Reference 1823
11.245Vector2i.hh File Reference 1824
11.246Vector3.hh File Reference 1825
11.247Vector4.hh File Reference 1826
11.248Video.hh File Reference 1827
11.249VideoVisual.hh File Reference 1829
11.250VideoVisualPrivate.hh File Reference 1830
11.251ViewController.hh File Reference 1830
11.252Visual.hh File Reference 1832
11.253VisualPrivate.hh File Reference 1833
11.254WindowManager.hh File Reference 1834
11.255WireBox.hh File Reference 1834
11.256WireBoxPrivate.hh File Reference 1835
11.257WirelessReceiver.hh File Reference 1836
11.258WirelessTransceiver.hh File Reference 1836
11.259WirelessTransmitter.hh File Reference 1838
11.260World.hh File Reference 1838
11.261WorldState.hh File Reference 1840
11.262WrenchVisual.hh File Reference 1841
11.263WrenchVisualPrivate.hh File Reference 1842

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 set of classes that encapsulate math related properties and functions.

`Messages`: All messages and helper functions.

`Physics`: Classes for physics and dynamics

`Rendering`: A set of rendering related class, functions, and definitions.

`Sensors`: A 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 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. **683) (unsigned int _index)**

: not yet implemented. Get external forces applied at this Joint. Note that the unit of force should be consistent with the rest of the simulation scales.

Member gazebo::physics::World::RunBlocking (p. **1541) (unsigned int _-iterations=0)**

In gazebo 3.0 this should be move to the proper section.

Member gazebo::sensors::CameraSensor::GetTopic (p. **282) () const**

to be implemented

Class gazebo::SystemPlugin (p. **1359)**

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 39
Events 55
Math 58
Messages 67
Classes for physics and dynamics 82
    DART Physics 92
    Bullet Physics 96
    Simbody Physics 98
Rendering 100
Sensors 104
Transport 110
Utility 119

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

boost . . . 121
gazebo
Forward declarations for the common classes . . . 121
gazebo::common
Common namespace . . . 126
gazebo::event
Event (p. 514) namespace . . . 131
gazebo::math
Math namespace . . . 132
gazebo::msgs
Messages namespace . . . 135
gazebo::physics
Namespace for physics . . . 138
gazebo::rendering
Rendering namespace . . . 147
gazebo::sensors
Sensors namespace . . . 153
gazebo::transport . . . 159
gazebo::util . . . 162
google . . . 163
google::protobuf . . . 163
google::protobuf::compiler . . . 163
google::protobuf::compiler::cpp . . . 163
Ogre . . . 163
ogre . . . 163

OVR 163
OVR::Util 164
OVR::Util::Render 164
SimTK 164
SkyX 164

Chapter 5

Class Index

5.1 Class Hierarchy

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

gazebo::math::Angle . . . 173
gazebo::common::Animation . . . 182
gazebo::common::NumericAnimation . . . 936
gazebo::common::PoseAnimation . . . 1006
gazebo::common::AudioDecoder . . . 192
gazebo::physics::BallJoint< T > . . . 199
gazebo::physics::Base . . . 201
gazebo::physics::Entity . . . 500
gazebo::physics::Collision . . . 295
gazebo::physics::DARTCollision . . . 384
gazebo::physics::SimbodyCollision . . . 1174
gazebo::physics::Link . . . 739
gazebo::physics::DARTLink . . . 418
gazebo::physics::SimbodyLink . . . 1207
gazebo::physics::Model . . . 846
gazebo::physics::Actor . . . 165
gazebo::physics::DARTModel . . . 433
gazebo::physics::SimbodyModel . . . 1221
gazebo::physics::Joint . . . 669
gazebo::physics::DARTJoint . . . 405
gazebo::physics::BallJoint< DARTJoint > . . . 199
gazebo::physics::DARTBallJoint . . . 374
gazebo::physics::Hinge2Joint< DARTJoint > . . . 634
gazebo::physics::DARTHinge2Joint . . . 393

gazebo::physics::HingeJoint< DARTJoint > 635
gazebo::physics::DARTHingeJoint 399
gazebo::physics::ScrewJoint< DARTJoint > 1118
gazebo::physics::DARTScrewJoint 450
gazebo::physics::SliderJoint< DARTJoint > 1294
gazebo::physics::DARTSliderJoint 459
gazebo::physics::UniversalJoint< DARTJoint > 1404
gazebo::physics::DARTUniversalJoint 468
gazebo::physics::SimbodyJoint 1193
gazebo::physics::BallJoint< SimbodyJoint > 199
gazebo::physics::SimbodyBallJoint 1165
gazebo::physics::Hinge2Joint< SimbodyJoint > 634
gazebo::physics::SimbodyHinge2Joint 1181
gazebo::physics::HingeJoint< SimbodyJoint > 635
gazebo::physics::SimbodyHingeJoint 1187
gazebo::physics::ScrewJoint< SimbodyJoint > 1118
gazebo::physics::SimbodyScrewJoint 1243
gazebo::physics::SliderJoint< SimbodyJoint > 1294
gazebo::physics::SimbodySliderJoint 1252
gazebo::physics::UniversalJoint< SimbodyJoint > 1404
gazebo::physics::SimbodyUniversalJoint 1260
gazebo::physics::Road 1083
gazebo::physics::Shape 1161
gazebo::physics::BoxShape 228
gazebo::physics::DARTBoxShape 382
gazebo::physics::SimbodyBoxShape 1173
gazebo::physics::CylinderShape 370
gazebo::physics::DARTCylinderShape 389
gazebo::physics::SimbodyCylinderShape 1178
gazebo::physics::HeightmapShape 628
gazebo::physics::DARTHeightmapShape 391
gazebo::physics::SimbodyHeightmapShape 1180
gazebo::physics::MeshShape 841
gazebo::physics::DARTMeshShape 431
gazebo::physics::SimbodyMeshShape 1219
gazebo::physics::MultiRayShape 901
gazebo::physics::DARTMultiRayShape 436
gazebo::physics::SimbodyMultiRayShape 1223
gazebo::physics::PlaneShape 987
gazebo::physics::DARTPlaneShape 446
gazebo::physics::SimbodyPlaneShape 1239
gazebo::physics::RayShape 1060
gazebo::physics::DARTRayShape 449

gazebo::physics::SimbodyRayShape 1241
gazebo::physics::SphereShape 1307
gazebo::physics::DARTSphereShape 465
gazebo::physics::SimbodySphereShape 1258
gazebo::math::Box 222
gazebo::common::FileLogger::Buffer 231
gazebo::common::Logger::Buffer 232
gazebo::common::BVHLoader 234
gazebo::transport::CallbackHelper 235
gazebo::transport::CallbackHelperT< M > 240
gazebo::transport::RawCallbackHelper 1048
gazebo::transport::SubscriptionTransport 1347
gazebo::rendering::Camera 242
gazebo::rendering::DepthCamera 474
gazebo::rendering::GpuLaser 579
gazebo::rendering::OculusCamera 940
gazebo::rendering::UserCamera 1407
gazebo::rendering::CameraPrivate 276
gazebo::common::ColladaExporterPrivate 290
gazebo::common::ColladaLoaderPrivate 293
gazebo::common::Color 312
gazebo::transport::Connection 328
gazebo::event::Connection 336
gazebo::event::ConnectionPrivate 343
gazebo::transport::ConnectionReadTask 344
gazebo::common::Console 345
gazebo::physics::Contact 347
gazebo::physics::ContactManager 351
gazebo::rendering::ContactVisualPrivate::ContactPoint 356
gazebo::physics::ContactPublisher 357
gazebo::rendering::Conversions 367
gazebo::physics::DARTTypes 467
gazebo::rendering::DummyPageProvider 490
gazebo::rendering::DynamicRenderable 495
gazebo::rendering::DynamicLines 491
gazebo::event::Event 514
gazebo::event::EventT< T > 535
gazebo::event::EventPrivate 518
gazebo::event::EventTPrivate< T > 547
gazebo::event::Events 519
gazebo::rendering::Events 533
gazebo::common::Exception 548
gazebo::common::InternalError 666
gazebo::common::AssertionInternalError 191

gazebo::common::FileLogger 551
gazebo::math::Filter< T > 553
gazebo::math::BiQuad< T > 217
gazebo::math::OnePole< T > 948
gazebo::math::Filter< math::Quaternion > 553
gazebo::math::OnePole< math::Quaternion > 948
gazebo::math::OnePoleQuaternion 951
gazebo::math::Filter< math::Vector3 > 553
gazebo::math::BiQuad< math::Vector3 > 217
gazebo::math::BiQuadVector3 221
gazebo::math::OnePole< math::Vector3 > 948
gazebo::math::OnePoleVector3 953
gazebo::physics::FrictionPyramid 564
google::protobuf::compiler::cpp::GazeboGenerator 571
gazebo::physics::GearboxJoint< T > 571
gazebo::common::GeometryIndices 574
gazebo::rendering::Grid 605
gazebo::physics::Gripper 609
gazebo::rendering::GUIOverlay 612
gazebo::rendering::GUIOverlayPrivate 617
gazebo::rendering::GzTerrainMatGen 618
gazebo::rendering::Heightmap 619
gazebo::common::HeightmapData 625
gazebo::common::ImageHeightmap 647
gazebo::physics::Hinge2Joint< T > 634
gazebo::physics::HingeJoint< T > 635
gazebo::common::Image 637
gazebo::physics::Inertial 654
gazebo::transport::IOManager 667
gazebo::physics::JointController 704
gazebo::physics::JointControllerPrivate 710
gazebo::physics::JointWrench 721
gazebo::common::KeyEvent 723
gazebo::common::KeyFrame 725
gazebo::common::NumericKeyFrame 938
gazebo::common::PoseKeyFrame 1008
gazebo::rendering::Light 730
gazebo::common::Logger 778
Logplay 785
gazebo::Master 792
gazebo::common::Material 794
gazebo::math::Matrix3 806
gazebo::math::Matrix4 812
gazebo::common::Mesh 821

gazebo::common::MeshCSG 829
gazebo::common::MeshExporter 831
gazebo::common::ColladaExporter 287
gazebo::common::MeshLoader 833
gazebo::common::ColladaLoader 291
gazebo::common::STLLoader 1328
gazebo::common::ModelDatabasePrivate 865
gazebo::common::MouseEvent 880
gazebo::rendering::MovableText 883
gazebo::common::MovingWindowFilter< T > 890
gazebo::common::MovingWindowFilterPrivate< T > 892
gazebo::msgs::MsgFactory 893
gazebo::transport::Node 910
gazebo::common::NodeAnimation 918
gazebo::common::NodeAssignment 923
gazebo::common::NodeTransform 925
gazebo::sensors::Noise 931
gazebo::sensors::GaussianNoiseModel 566
gazebo::sensors::ImageGaussianNoiseModel 644
gazebo::sensors::NoiseFactory 935
gazebo::physics::PhysicsEngine 959
gazebo::physics::DARTPhysics 438
gazebo::physics::SimbodyPhysics 1226
gazebo::physics::PhysicsFactory 976
gazebo::common::PID 977
gazebo::math::Plane 984
gazebo::PluginT< T > 992
gazebo::PluginT< ModelPlugin > 992
gazebo::ModelPlugin 867
gazebo::PluginT< SensorPlugin > 992
gazebo::SensorPlugin 1151
gazebo::PluginT< SystemPlugin > 992
gazebo::SystemPlugin 1359
gazebo::PluginT< VisualPlugin > 992
gazebo::VisualPlugin 1504
gazebo::PluginT< WorldPlugin > 992
gazebo::WorldPlugin 1544
gazebo::math::Pose 995
gazebo::rendering::Projector 1011
gazebo::transport::Publication 1014
gazebo::transport::PublicationTransport 1021
gazebo::transport::Publisher 1024
gazebo::transport::PublishTask 1028

gazebo::math::Quaternion 1029
gazebo::math::Rand 1046
Road 1083
gazebo::rendering::Road2d 1086
gazebo::math::RotationSpline 1087
gazebo::rendering::Scene 1097
gazebo::physics::ScrewJoint< T > 1118
gazebo::sensors::Sensor 1130
gazebo::sensors::CameraSensor 280
gazebo::sensors::ContactSensor 358
gazebo::sensors::DepthCameraSensor 480
gazebo::sensors::ForceTorqueSensor 556
gazebo::sensors::GpsSensor 575
gazebo::sensors::GpuRaySensor 591
gazebo::sensors::ImuSensor 650
gazebo::sensors::MultiCameraSensor 895
gazebo::sensors::RaySensor 1051
gazebo::sensors::RFIDSensor 1072
gazebo::sensors::RFIDTag 1075
gazebo::sensors::SonarSensor 1298
gazebo::sensors::WirelessTransceiver 1520
gazebo::sensors::WirelessReceiver 1517
gazebo::sensors::WirelessTransmitter 1524
SensorFactor 1144
gazebo::sensors::SensorFactory 1145
gazebo::Server 1153
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg 1156
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL 1158
gazebo::sensors::SimTimeEvent 1265
gazebo::sensors::SimTimeEventHandler 1266
SingletonT< T > 1267
gazebo::common::MeshManager 834
gazebo::common::ModelDatabase 864
gazebo::common::SystemPaths 1352
gazebo::rendering::RenderEngine 1067
gazebo::rendering::RTShaderSystem 1092
gazebo::sensors::SensorManager 1146
gazebo::transport::ConnectionManager 337
gazebo::transport::TopicManager 1389
gazebo::util::DiagnosticManager 484
gazebo::util::LogPlay 781
gazebo::util::LogRecord 786
SingletonT< ConnectionManager > 1267
SingletonT< DiagnosticManager > 1267
SingletonT< LogPlay > 1267

SingletonT< LogRecord > 1267
SingletonT< MeshManager > 1267
SingletonT< ModelDatabase > 1267
SingletonT< RenderEngine > 1267
SingletonT< RTShaderSystem > 1267
SingletonT< SensorManager > 1267
SingletonT< SystemPaths > 1267
SingletonT< TopicManager > 1267
gazebo::common::Skeleton 1269
gazebo::common::SkeletonAnimation 1277
gazebo::common::SkeletonNode 1283
gazebo::physics::SliderJoint< T > 1294
gazebo::rendering::GzTerrainMatGen::SM2Profile 1296
gazebo::common::SphericalCoordinates 1310
gazebo::common::SphericalCoordinatesPrivate 1316
gazebo::math::Spline 1317
gazebo::physics::State 1323
gazebo::physics::CollisionState 307
gazebo::physics::JointState 712
gazebo::physics::LinkState 769
gazebo::physics::ModelState 869
gazebo::physics::WorldState 1545
gazebo::common::SubMesh 1330
gazebo::transport::SubscribeOptions 1343
gazebo::transport::Subscriber 1346
gazebo::physics::SurfaceParams 1350
gazebo::common::Time 1361
gazebo::common::Timer 1387
gazebo::util::DiagnosticTimer 487
gazebo::physics::TrajectoryInfo 1398
gazebo::physics::UniversalJoint< T > 1404
gazebo::common::UpdateInfo 1406
gazebo::rendering::UserCameraPrivate 1418
gazebo::math::Vector2d 1420
gazebo::math::Vector2i 1430
gazebo::math::Vector3 1440
gazebo::math::Vector4 1457
gazebo::common::Video 1468
gazebo::rendering::ViewController 1473
gazebo::rendering::FPSViewController 561
gazebo::rendering::OrbitViewController 955
gazebo::rendering::Visual 1477
gazebo::rendering::ArrowVisual 188
gazebo::rendering::AxisVisual 194
gazebo::rendering::CameraVisual 284

gazebo::rendering::COMVisual 325
gazebo::rendering::ContactVisual 363
gazebo::rendering::JointVisual 718
gazebo::rendering::LaserVisual 727
gazebo::rendering::RFIDTagVisual 1078
gazebo::rendering::RFIDVisual 1081
gazebo::rendering::SelectionObj 1121
gazebo::rendering::SonarVisual 1303
gazebo::rendering::TransmitterVisual 1400
gazebo::rendering::VideoVisual 1470
gazebo::rendering::WrenchVisual 1553
gazebo::rendering::VisualPrivate 1505
gazebo::rendering::ArrowVisualPrivate 189
gazebo::rendering::AxisVisualPrivate 197
gazebo::rendering::CameraVisualPrivate 286
gazebo::rendering::COMVisualPrivate 327
gazebo::rendering::ContactVisualPrivate 365
gazebo::rendering::JointVisualPrivate 720
gazebo::rendering::LaserVisualPrivate 728
gazebo::rendering::RFIDTagVisualPrivate 1079
gazebo::rendering::RFIDVisualPrivate 1082
gazebo::rendering::SelectionObjPrivate 1125
gazebo::rendering::SonarVisualPrivate 1304
gazebo::rendering::TransmitterVisualPrivate 1401
gazebo::rendering::VideoVisualPrivate 1471
gazebo::rendering::WrenchVisualPrivate 1555
gazebo::rendering::WindowManager 1511
gazebo::rendering::WireBox 1515
gazebo::rendering::WireBoxPrivate 1516
gazebo::physics::World 1529

Chapter 6

Class Index

6.1 Class List

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

gazebo::physics::Actor
Actor (p. 165) class enables GPU based mesh model / skeleton scriptable animation 165

gazebo::math::Angle
An angle and related functions 173

gazebo::common::Animation
Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes 182

gazebo::rendering::ArrowVisual
Basic arrow visualization 188

gazebo::rendering::ArrowVisualPrivate
Private data for the Arrow **Visual** (p. 1477) class 189

gazebo::common::AssertionInternalError
Class for generating Exceptions which come from gazebo assertions 191

gazebo::common::AudioDecoder
An audio decoder based on FFMPEG 192

gazebo::rendering::AxisVisual
Basic axis visualization 194

gazebo::rendering::AxisVisualPrivate
Private data for the Axis **Visual** (p. 1477) class 197

gazebo::physics::BallJoint< **T** >
Base (p. 201) class for a ball joint 199

gazebo::physics::Base
Base (p. 201) class for most physics classes 201

gazebo::math::BiQuad< T >
Bi-quad filter base class 217
gazebo::math::BiQuadVector3
BiQuad (p. 217) vector3 filter 221
gazebo::math::Box
Mathematical representation of a box and related functions 222
gazebo::physics::BoxShape
Box geometry primitive 228
gazebo::common::FileLogger::Buffer
String buffer for the file logger 231
gazebo::common::Logger::Buffer
String buffer for the base logger 232
gazebo::common::BVHLoader
Handles loading BVH animation files 234
gazebo::transport::CallbackHelper
A helper class to handle callbacks when messages arrive 235
gazebo::transport::CallbackHelperT< M >
Callback helper Template 240
gazebo::rendering::Camera
Basic camera sensor 242
gazebo::rendering::CameraPrivate
Private data for the **Camera** (p. 242) class 276
gazebo::sensors::CameraSensor
Basic camera sensor 280
gazebo::rendering::CameraVisual
Basic camera visualization 284
gazebo::rendering::CameraVisualPrivate 286
gazebo::common::ColladaExporter
Class used to export Collada mesh files 287
gazebo::common::ColladaExporterPrivate
Private data for the **ColladaExporter** (p. 287) class 290
gazebo::common::ColladaLoader
Class used to load Collada mesh files 291
gazebo::common::ColladaLoaderPrivate
Private data for the **ColladaLoader** (p. 291) class 293
gazebo::physics::Collision
Base (p. 201) class for all collision entities 295
gazebo::physics::CollisionState
Store state information of a **physics::Collision** (p. 295) object . . . 307
gazebo::common::Color
Defines a color 312
gazebo::rendering::COMVisual
Basic Center of Mass visualization 325
gazebo::rendering::COMVisualPrivate
Private data for the COM **Visual** (p. 1477) class 327

gazebo::transport::Connection
Single TCP/IP connection manager 328
gazebo::event::Connection
A class that encapsulates a connection 336
gazebo::transport::ConnectionManager
Manager of connections 337
gazebo::event::ConnectionPrivate 343
gazebo::transport::ConnectionReadTask
344
gazebo::common::Console
Container for loggers, and global logging options (such as verbose vs 345
gazebo::physics::Contact
A contact between two collisions 347
gazebo::physics::ContactManager
Aggregates all the contact information generated by the collision detection engine 351
gazebo::rendering::ContactVisualPrivate::ContactPoint
A contact point visualization 356
gazebo::physics::ContactPublisher
A custom contact publisher created for each contact filter in the **-Contact** (p. 347) Manager 357
gazebo::sensors::ContactSensor
Contact sensor 358
gazebo::rendering::ContactVisual
Contact visualization 363
gazebo::rendering::ContactVisualPrivate
Private data for the Arrow **Visual** (p. 1477) class 365
gazebo::rendering::Conversions
Conversions (p. 367) **Conversions.hh** (p. 1608) **rendering/-Conversions.hh** (p. 1608) 367
gazebo::physics::CylinderShape
Cylinder collision 370
gazebo::physics::DARTBallJoint
An **DARTBallJoint** (p. 374) 374
gazebo::physics::DARTBoxShape
DART Box shape 382
gazebo::physics::DARTCollision
Base (p. 201) class for all DART collisions 384
gazebo::physics::DARTCylinderShape
DART cylinder shape 389
gazebo::physics::DARTHeightmapShape
DART Height map collision 391
gazebo::physics::DARTHinge2Joint
A two axis hinge joint 393
gazebo::physics::DARTHingeJoint
A single axis hinge joint 399

gazebo::physics::DARTJoint
DART joint interface 405
gazebo::physics::DARTLink
DART **Link** (p. 739) class 418
gazebo::physics::DARTMeshShape
Triangle mesh collision 431
gazebo::physics::DARTModel
DART model class 433
gazebo::physics::DARTMultiRayShape
DART specific version of **MultiRayShape** (p. 901) 436
gazebo::physics::DARTPhysics
DART physics engine 438
gazebo::physics::DARTPlaneShape
An DART Plane shape 446
gazebo::physics::DARTRayShape
Ray collision 449
gazebo::physics::DARTScrewJoint
A screw joint 450
gazebo::physics::DARTSliderJoint
A slider joint 459
gazebo::physics::DARTSphereShape
A DART sphere shape 465
gazebo::physics::DARTTypes
A set of functions for converting between the math types used by gazebo and dart 467
gazebo::physics::DARTUniversalJoint
A universal joint 468
gazebo::rendering::DepthCamera
Depth camera used to render depth data into an image buffer 474
gazebo::sensors::DepthCameraSensor 480
gazebo::util::DiagnosticManager
A diagnostic manager class 484
gazebo::util::DiagnosticTimer
A timer designed for diagnostics 487
gazebo::rendering::DummyPageProvider
Pretends to provide procedural page content to avoid page loading . 490
gazebo::rendering::DynamicLines
Class for drawing lines that can change 491
gazebo::rendering::DynamicRenderable
Abstract base class providing mechanisms for dynamically growing hardware buffers 495
gazebo::physics::Entity
Base (p. 201) class for all physics objects in Gazebo 500
gazebo::event::Event
Base class for all events 514
gazebo::event::EventPrivate 518

gazebo::event::Events
An **Event** (p. 514) class to get notifications for simulator events . . . 519
gazebo::rendering::Events
Base class for rendering events . . . 533
gazebo::event::EventT< T >
A class for event processing . . . 535
gazebo::event::EventTPrivate< T > . . . 547
gazebo::common::Exception
Class for generating exceptions . . . 548
gazebo::common::FileLogger
A logger that outputs messages to a file . . . 551
gazebo::math::Filter< T >
Filter (p. 553) base class . . . 553
gazebo::sensors::ForceTorqueSensor
Sensor (p. 1130) for measure force and torque on a joint . . . 556
gazebo::rendering::FPSViewController
First Person Shooter style view controller . . . 561
gazebo::physics::FrictionPyramid
Parameters used for friction pyramid model . . . 564
gazebo::sensors::GaussianNoiseModel
Gaussian noise class . . . 566
google::protobuf::compiler::cpp::GazeboGenerator
Google protobuf message generator for **gazebo::msgs** (p. 135) . . 571
gazebo::physics::GearboxJoint< T >
A double axis gearbox joint . . . 571
gazebo::common::GeometryIndices
Helper data structure for loading collada geometries . . . 574
gazebo::sensors::GpsSensor
GpsSensor (p. 575) to provide position measurement . . . 575
gazebo::rendering::GpuLaser
GPU based laser distance sensor . . . 579
gazebo::sensors::GpuRaySensor . . . 591
gazebo::rendering::Grid
Displays a grid of cells, drawn with lines . . . 605
gazebo::physics::Gripper
A gripper abstraction . . . 609
gazebo::rendering::GUIOverlay
A class that creates a CEGUI overlay on a render window . . . 612
gazebo::rendering::GUIOverlayPrivate
Private data for the **GUIOverlay** (p. 612) class . . . 617
gazebo::rendering::GzTerrainMatGen . . . 618
gazebo::rendering::Heightmap
Rendering a terrain using heightmap information . . . 619
gazebo::common::HeightmapData
Encapsulates a generic heightmap data file . . . 625

gazebo::physics::HeightmapShape
HeightmapShape (p. 628) collision shape builds a heightmap from an image 628
gazebo::physics::Hinge2Joint< T >
A two axis hinge joint 634
gazebo::physics::HingeJoint< T >
A single axis hinge joint 635
gazebo::common::Image
Encapsulates an image 637
gazebo::sensors::ImageGaussianNoiseModel 644
gazebo::common::ImageHeightmap
Encapsulates an image that will be interpreted as a heightmap . . . 647
gazebo::sensors::ImuSensor
An IMU sensor 650
gazebo::physics::Inertial
A class for inertial information about a link 654
gazebo::common::InternalError
Class for generating Internal Gazebo Errors: those errors which should never happend and represent programming bugs 666
gazebo::transport::IOManager
Manages boost::asio IO 667
gazebo::physics::Joint
Base (p. 201) class for all joints 669
gazebo::physics::JointController
A class for manipulating **physics::Joint** (p. 669) 704
gazebo::physics::JointControllerPrivate 710
gazebo::physics::JointState
Keeps track of state of a **physics::Joint** (p. 669) 712
gazebo::rendering::JointVisual
Visualization for joints 718
gazebo::rendering::JointVisualPrivate
Private data for the Joint **Visual** (p. 1477) class 720
gazebo::physics::JointWrench
Wrench information from a joint 721
gazebo::common::KeyEvent
Generic description of a keyboard event 723
gazebo::common::KeyFrame
A key frame in an animation 725
gazebo::rendering::LaserVisual
Visualization for laser data 727
gazebo::rendering::LaserVisualPrivate
Private data for the Laser **Visual** (p. 1477) class 728
gazebo::rendering::Light
A light source 730

gazebo::physics::Link
Link (p. 739) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body 739
gazebo::physics::LinkState
Store state information of a **physics::Link** (p. 739) object 769
gazebo::common::Logger
Terminal logger 778
gazebo::util::LogPlay 781
Logplay
Open and playback log files that were recorded using LogRecord . . 785
gazebo::util::LogRecord
Addtogroup gazebo_util 786
gazebo::Master
A manager that directs topic connections, enables each gazebo network client to locate one another for peer-to-peer communication . . 792
gazebo::common::Material
Encapsulates description of a material 794
gazebo::math::Matrix3
A 3x3 matrix class 806
gazebo::math::Matrix4
A 3x3 matrix class 812
gazebo::common::Mesh
A 3D mesh 821
gazebo::common::MeshCSG
Creates CSG meshes 829
gazebo::common::MeshExporter
Base class for exporting meshes 831
gazebo::common::MeshLoader
Base class for loading meshes 833
gazebo::common::MeshManager
Maintains and manages all meshes 834
gazebo::physics::MeshShape
Triangle mesh collision shape 841
gazebo::physics::Model
A model is a collection of links, joints, and plugins 846
gazebo::common::ModelDatabase
Connects to model database, and has utility functions to find models 864
gazebo::common::ModelDatabasePrivate
Private class attributes for **ModelDatabase** (p. 864) 865
gazebo::ModelPlugin
A plugin with access to **physics::Model** (p. 846) 867
gazebo::physics::ModelState
Store state information of a **physics::Model** (p. 846) object 869
gazebo::common::MouseEvent
Generic description of a mouse event 880

gazebo::rendering::MovableText
Movable text 883
gazebo::common::MovingWindowFilter< T >
Base class for **MovingWindowFilter** (p. 890) 890
gazebo::common::MovingWindowFilterPrivate< T > 892
gazebo::msgs::MsgFactory
A factory that generates protobuf message based on a string type . 893
gazebo::sensors::MultiCameraSensor
Multiple camera sensor 895
gazebo::physics::MultiRayShape
Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner 901
gazebo::transport::Node
A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics 910
gazebo::common::NodeAnimation
Node animation 918
gazebo::common::NodeAssignment
Vertex to node weighted assignement for skeleton animation visualization 923
gazebo::common::NodeTransform
NodeTransform (p. 925) **Skeleton.hh** (p. 1787) common/common.hh 925
gazebo::sensors::Noise
Noise (p. 931) models for sensor output signals 931
gazebo::sensors::NoiseFactory
Use this noise manager for creating and loading noise models . . . 935
gazebo::common::NumericAnimation
A numeric animation 936
gazebo::common::NumericKeyFrame
A keyframe for a **NumericAnimation** (p. 936) 938
gazebo::rendering::OculusCamera
A camera used for user visualization of a scene 940
gazebo::math::OnePole< T >
A one-pole DSP filter 948
gazebo::math::OnePoleQuaternion
One-pole quaternion filter 951
gazebo::math::OnePoleVector3
One-pole vector3 filter 953
gazebo::rendering::OrbitViewController
Orbit view controller 955
gazebo::physics::PhysicsEngine
Base (p. 201) class for a physics engine 959
gazebo::physics::PhysicsFactory
The physics factory instantiates different physics engines 976

gazebo::common::PID
Generic **PID** (p. 977) controller class 977
gazebo::math::Plane
A plane and related functions 984
gazebo::physics::PlaneShape
Collision (p. 295) for an infinite plane 987
gazebo::PluginT< T >
A class which all plugins must inherit from 992
gazebo::math::Pose
Encapsulates a position and rotation in three space 995
gazebo::common::PoseAnimation
A pose animation 1006
gazebo::common::PoseKeyFrame
A keyframe for a **PoseAnimation** (p. 1006) 1008
gazebo::rendering::Projector
Projects a material onto surface, light a light projector 1011
gazebo::transport::Publication
A publication for a topic 1014
gazebo::transport::PublicationTransport
Transport/transport.hh 1021
gazebo::transport::Publisher
A publisher of messages on a topic 1024
gazebo::transport::PublishTask
1028
gazebo::math::Quaternion
A quaternion class 1029
gazebo::math::Rand
Random number generator class 1046
gazebo::transport::RawCallbackHelper
Used to connect publishers to subscribers, where the subscriber wants the raw data from the publisher 1048
gazebo::sensors::RaySensor
Sensor (p. 1130) with one or more rays 1051
gazebo::physics::RayShape
Base (p. 201) class for Ray collision geometry 1060
gazebo::rendering::RenderEngine
Adaptor to Ogre3d 1067
gazebo::sensors::RFIDSensor
Sensor (p. 1130) class for RFID type of sensor 1072
gazebo::sensors::RFIDTag
RFIDTag (p. 1075) to interact with RFIDTagSensors 1075
gazebo::rendering::RFIDTagVisual
Visualization for RFID tags sensor 1078
gazebo::rendering::RFIDTagVisualPrivate
Private data for the RFID Tag **Visual** (p. 1477) class 1079

gazebo::rendering::RFIDVisual
Visualization for RFID sensor 1081
gazebo::rendering::RFIDVisualPrivate
Private data for the RFID **Visual** (p. 1477) class 1082
Road
Used to render a strip of road 1083
gazebo::physics::Road
For building a **Road** (p. 1083) from SDF 1083
gazebo::rendering::Road2d 1086
gazebo::math::RotationSpline
Spline (p. 1317) for rotations 1087
gazebo::rendering::RTShaderSystem
Implements **Ogre** (p. 163)'s Run-Time Shader system 1092
gazebo::rendering::Scene
Representation of an entire scene graph 1097
gazebo::physics::ScrewJoint< T >
A screw joint, which has both prismatic and rotational DOFs 1118
gazebo::rendering::SelectionObj
Interactive selection object for models and links 1121
gazebo::rendering::SelectionObjPrivate
Private data for the Selection Obj class 1125
gazebo::sensors::Sensor
Base class for sensors 1130
SensorFactor
The sensor factory; the class is just for namespacing purposes . . . 1144
gazebo::sensors::SensorFactory 1145
gazebo::sensors::SensorManager
Class to manage and update all sensors 1146
gazebo::SensorPlugin
A plugin with access to physics::Sensor 1151
gazebo::Server 1153
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg
Keeping the CG shader for reference 1156
gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL
Utility class to help with generating shaders for GLSL 1158
gazebo::physics::Shape
Base (p. 201) class for all shapes 1161
gazebo::physics::SimbodyBallJoint
SimbodyBallJoint (p. 1165) class models a ball joint in Simbody . . 1165
gazebo::physics::SimbodyBoxShape
Simbody box collision 1173
gazebo::physics::SimbodyCollision
Simbody collisions 1174
gazebo::physics::SimbodyCylinderShape
Cylinder collision 1178

gazebo::physics::SimbodyHeightmapShape
Height map collision 1180
gazebo::physics::SimbodyHinge2Joint
A two axis hinge joint 1181
gazebo::physics::SimbodyHingeJoint
A single axis hinge joint 1187
gazebo::physics::SimbodyJoint
Base (p. 201) class for all joints 1193
gazebo::physics::SimbodyLink
Simbody **Link** (p. 739) class 1207
gazebo::physics::SimbodyMeshShape
Triangle mesh collision 1219
gazebo::physics::SimbodyModel
A model is a collection of links, joints, and plugins 1221
gazebo::physics::SimbodyMultiRayShape
Simbody specific version of **MultiRayShape** (p. 901) 1223
gazebo::physics::SimbodyPhysics
Simbody physics engine 1226
gazebo::physics::SimbodyPlaneShape
Simbody collision for an infinite plane 1239
gazebo::physics::SimbodyRayShape
Ray shape for simbody 1241
gazebo::physics::SimbodyScrewJoint
A screw joint 1243
gazebo::physics::SimbodySliderJoint
A slider joint 1252
gazebo::physics::SimbodySphereShape
Simbody sphere collision 1258
gazebo::physics::SimbodyUniversalJoint
A simbody universal joint class 1260
gazebo::sensors::SimTimeEvent
1265
gazebo::sensors::SimTimeEventHandler
Monitors simulation time, and notifies conditions when a specified time has been reached 1266
SingletonT< T >
Singleton template class 1267
gazebo::common::Skeleton
A skeleton 1269
gazebo::common::SkeletonAnimation
Skeleton (p. 1269) animation 1277
gazebo::common::SkeletonNode
A skeleton node 1283
gazebo::physics::SliderJoint< T >
A slider joint 1294

gazebo::rendering::GzTerrainMatGen::SM2Profile
Shader model 2 profile target 1296
gazebo::sensors::SonarSensor
Sensor (p. 1130) with sonar cone 1298
gazebo::rendering::SonarVisual
Visualization for sonar data 1303
gazebo::rendering::SonarVisualPrivate
Private data for the Sonar **Visual** (p. 1477) class 1304
gazebo::physics::SphereShape
Sphere collision shape 1307
gazebo::common::SphericalCoordinates
Convert spherical coordinates for planetary surfaces 1310
gazebo::common::SphericalCoordinatesPrivate
Commmon/common.hh 1316
gazebo::math::Spline
Splines 1317
gazebo::physics::State
State (p. 1323) of an entity 1323
gazebo::common::STLLoader
Class used to load STL mesh files 1328
gazebo::common::SubMesh
A child mesh 1330
gazebo::transport::SubscribeOptions
Options for a subscription 1343
gazebo::transport::Subscriber
A subscriber to a topic 1346
gazebo::transport::SubscriptionTransport
Transport/transport.hh 1347
gazebo::physics::SurfaceParams
SurfaceParams (p. 1350) defines various Surface contact parameters 1350
gazebo::common::SystemPaths
Functions to handle getting system paths, keeps track of: 1352
gazebo::SystemPlugin
A plugin loaded within the gzserver on startup 1359
gazebo::common::Time
A **Time** (p. 1361) class, can be used to hold wall- or sim-time 1361
gazebo::common::Timer
A timer class, used to time things in real world walltime 1387
gazebo::transport::TopicManager
Manages topics and their subscriptions 1389
gazebo::physics::TrajectoryInfo
Information about a trajectory for an **Actor** (p. 165) 1398
gazebo::rendering::TransmitterVisual
Visualization for the wireless propagation data 1400

gazebo::rendering::TransmitterVisualPrivate
Private data for the Transmitter **Visual** (p. 1477) class 1401
gazebo::physics::UniversalJoint< T >
A universal joint 1404
gazebo::common::UpdateInfo
Information for use in an update event 1406
gazebo::rendering::UserCamera
A camera used for user visualization of a scene 1407
gazebo::rendering::UserCameraPrivate
Private data for the **UserCamera** (p. 1407) class 1418
gazebo::math::Vector2d
Generic double x, y vector 1420
gazebo::math::Vector2i
Generic integer x, y vector 1430
gazebo::math::Vector3
Generic vector containing 3 elements 1440
gazebo::math::Vector4
Double Generic x, y, z, w vector 1457
gazebo::common::Video
Handle video encoding and decoding using libavcodec 1468
gazebo::rendering::VideoVisual
A visual element that displays a video as a texture 1470
gazebo::rendering::VideoVisualPrivate
Private data for the Video **Visual** (p. 1477) class 1471
gazebo::rendering::ViewController
Base class for view controllers 1473
gazebo::rendering::Visual
A renderable object 1477
gazebo::VisualPlugin
A plugin loaded within the gzserver on startup 1504
gazebo::rendering::VisualPrivate
Private data for the **Visual** (p. 1477) class 1505
gazebo::rendering::WindowManager
Class to mangage render windows 1511
gazebo::rendering::WireBox
Draws a wireframe box 1515
gazebo::rendering::WireBoxPrivate
Private data for the **WireBox** (p. 1515) class 1516
gazebo::sensors::WirelessReceiver
Sensor (p. 1130) class for receiving wireless signals 1517
gazebo::sensors::WirelessTransceiver
Sensor (p. 1130) class for receiving wireless signals 1520
gazebo::sensors::WirelessTransmitter
Transmitter to send wireless signals 1524

gazebo::physics::World
The world provides access to all other object within a simulated environment 1529
gazebo::WorldPlugin
A plugin with access to **physics::World** (p. 1529) 1544
gazebo::physics::WorldState
Store state information of a **physics::World** (p. 1529) object 1545
gazebo::rendering::WrenchVisual
Visualization for sonar data 1553
gazebo::rendering::WrenchVisualPrivate
Private data for the Wrench **Visual** (p. 1477) class 1555

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Actor.hh 1559
Angle.hh 1561
Animation.hh 1563
ArrowVisual.hh 1565
ArrowVisualPrivate.hh 1565
Assert.hh 1566
AudioDecoder.hh 1567
AxisVisual.hh 1569
AxisVisualPrivate.hh 1570
BallJoint.hh 1570
Base.hh 1571
Base64.hh 1573
Box.hh 1575
BoxShape.hh 1576
BVHLoader.hh 1577
CallbackHelper.hh 1579
Camera.hh 1581
CameraPrivate.hh 1582
CameraSensor.hh 1583
CameraVisual.hh 1584
CameraVisualPrivate.hh 1584
cegui.h 1585
ColladaExporter.hh 1586
ColladaExporterPrivate.hh 1586
ColladaLoader.hh 1587

ColladaLoaderPrivate.hh . . . 1589
Collision.hh . . . 1590
CollisionState.hh . . . 1591
Color.hh . . . 1591
CommonIface.hh . . . 1592
CommonTypes.hh . . . 1594
COMVisual.hh . . . 1596
COMVisualPrivate.hh . . . 1597
Connection.hh . . . 1598
ConnectionManager.hh . . . 1600
Console.hh . . . 1601
Contact.hh . . . 1603
ContactManager.hh . . . 1605
ContactSensor.hh . . . 1606
ContactVisual.hh . . . 1607
ContactVisualPrivate.hh . . . 1607
Conversions.hh . . . 1608
CylinderShape.hh . . . 1609
dart_inc.h . . . 1611
DARTBallJoint.hh . . . 1612
DARTBoxShape.hh . . . 1612
DARTCollision.hh . . . 1613
DARTCylinderShape.hh . . . 1614
DARTHeightmapShape.hh . . . 1614
DARTHinge2Joint.hh . . . 1615
DARTHingeJoint.hh . . . 1616
DARTJoint.hh . . . 1616
DARTLink.hh . . . 1617
DARTMeshShape.hh . . . 1618
DARTModel.hh . . . 1618
DARTMultiRayShape.hh . . . 1619
DARTPhysics.hh . . . 1620
DARTPlaneShape.hh . . . 1620
DARTRayShape.hh . . . 1621
DARTScrewJoint.hh . . . 1622
DARTSliderJoint.hh . . . 1622
DARTSphereShape.hh . . . 1623
DARTTypes.hh
DART wrapper forward declarations and typedefs . . . 1623
DARTUniversalJoint.hh . . . 1625
Dem.hh . . . 1626
DemPrivate.hh . . . 1626
DepthCamera.hh . . . 1628
DepthCameraSensor.hh . . . 1628
Diagnostics.hh . . . 1629
DynamicLines.hh . . . 1630

DynamicRenderable.hh . . . 1631
Entity.hh . . . 1632
Event.hh . . . 1633
Events.hh . . . 1634
Exception.hh . . . 1635
ffmpeg_inc.h . . . 1636
Filter.hh . . . 1637
ForceTorqueSensor.hh . . . 1639
FPSViewController.hh . . . 1639
GaussianNoiseModel.hh . . . 1640
gazebo.hh . . . 1641
gazebo_core.hh . . . 1643
GazeboGenerator.hh . . . 1644
GearboxJoint.hh . . . 1645
GpsSensor.hh . . . 1646
GpuLaser.hh . . . 1647
GpuRaySensor.hh . . . 1648
Grid.hh . . . 1649
Gripper.hh . . . 1649
GUIOverlay.hh . . . 1650
GUIOverlayPrivate.hh . . . 1651
Heightmap.hh . . . 1652
HeightmapData.hh . . . 1653
HeightmapShape.hh . . . 1654
Helpers.hh . . . 1655
Hinge2Joint.hh . . . 1658
HingeJoint.hh . . . 1659
Image.hh . . . 1661
ImageHeightmap.hh . . . 1662
ImuSensor.hh . . . 1663
Inertial.hh . . . 1663
IOManager.hh . . . 1664
Joint.hh . . . 1666
JointController.hh . . . 1667
JointControllerPrivate.hh . . . 1668
JointState.hh . . . 1669
JointVisual.hh . . . 1670
JointVisualPrivate.hh . . . 1671
JointWrench.hh . . . 1671
KeyEvent.hh . . . 1672
KeyFrame.hh . . . 1674
LaserVisual.hh . . . 1676
LaserVisualPrivate.hh . . . 1676
Light.hh . . . 1677
Link.hh . . . 1678
LinkState.hh . . . 1679

LogPlay.hh . . . 1680
LogRecord.hh . . . 1681
mainpage.html . . . 1682
MapShape.hh . . . 1682
Master.hh . . . 1682
common/Material.hh . . . 1684
rendering/Material.hh . . . 1685
MathTypes.hh
Forward declarations for the math classes . . . 1685
Matrix3.hh . . . 1686
Matrix4.hh . . . 1687
Mesh.hh . . . 1688
MeshCSG.hh . . . 1689
MeshExporter.hh . . . 1690
MeshLoader.hh . . . 1691
MeshManager.hh . . . 1694
MeshShape.hh . . . 1695
Model.hh . . . 1697
ModelDatabase.hh . . . 1698
ModelDatabasePrivate.hh . . . 1700
ModelState.hh . . . 1701
MouseEvent.hh . . . 1702
MovableText.hh . . . 1703
MovingWindowFilter.hh . . . 1704
MsgFactory.hh . . . 1706
msgs.hh . . . 1707
MultiCameraSensor.hh . . . 1710
MultiRayShape.hh . . . 1711
Node.hh . . . 1712
Noise.hh . . . 1713
OculusCamera.hh . . . 1714
ogre_gazebo.h . . . 1715
OpenAL.hh . . . 1716
OrbitViewController.hh . . . 1717
PhysicsEngine.hh . . . 1718
PhysicsFactory.hh . . . 1718
PhysicsIface.hh . . . 1720
PhysicsTypes.hh
Default namespace for gazebo . . . 1722
PID.hh . . . 1726
Plane.hh . . . 1727
PlaneShape.hh . . . 1728
Plugin.hh . . . 1729
Pose.hh . . . 1733
Projector.hh . . . 1734
Publication.hh . . . 1735

PublicationTransport.hh . . . 1736
Publisher.hh . . . 1738
Quaternion.hh . . . 1739
Rand.hh . . . 1740
RaySensor.hh . . . 1742
RayShape.hh . . . 1743
RenderEngine.hh . . . 1744
RenderEvents.hh . . . 1744
RenderingIface.hh . . . 1745
RenderTypes.hh . . . 1746
RFIDSensor.hh . . . 1749
RFIDTag.hh . . . 1749
RFIDTagVisual.hh . . . 1750
RFIDTagVisualPrivate.hh . . . 1751
RFIDVisual.hh . . . 1751
RFIDVisualPrivate.hh . . . 1752
Road.hh . . . 1753
Road2d.hh . . . 1754
RotationSpline.hh . . . 1755
RTShaderSystem.hh . . . 1756
Scene.hh . . . 1757
ScrewJoint.hh . . . 1758
SelectionObj.hh . . . 1760
SelectionObjPrivate.hh . . . 1761
Sensor.hh . . . 1761
SensorFactory.hh . . . 1763
SensorManager.hh . . . 1764
SensorsIface.hh . . . 1765
SensorTypes.hh
Forward declarations and typedefs for sensors . . . 1766
Server.hh . . . 1768
Shape.hh . . . 1770
simbody_inc.h . . . 1771
SimbodyBallJoint.hh . . . 1771
SimbodyBoxShape.hh . . . 1772
SimbodyCollision.hh . . . 1773
SimbodyCylinderShape.hh . . . 1773
SimbodyHeightmapShape.hh . . . 1774
SimbodyHinge2Joint.hh . . . 1775
SimbodyHingeJoint.hh . . . 1775
SimbodyJoint.hh . . . 1776
SimbodyLink.hh . . . 1777
SimbodyMeshShape.hh . . . 1777
SimbodyModel.hh . . . 1778
SimbodyMultiRayShape.hh . . . 1779
SimbodyPhysics.hh . . . 1779

SimbodyPlaneShape.hh 1780
SimbodyRayShape.hh 1781
SimbodyScrewJoint.hh 1782
SimbodySliderJoint.hh 1783
SimbodySphereShape.hh 1783
SimbodyTypes.hh
Simbody wrapper forward declarations and typedefs 1784
SimbodyUniversalJoint.hh 1785
SingletonT.hh 1786
Skeleton.hh 1787
SkeletonAnimation.hh 1789
SliderJoint.hh 1791
SonarSensor.hh 1792
SonarVisual.hh 1792
SonarVisualPrivate.hh 1793
SphereShape.hh 1794
SphericalCoordinates.hh 1795
SphericalCoordinatesPrivate.hh 1796
Spline.hh 1797
State.hh 1799
STLLoader.hh 1800
SubscribeOptions.hh 1802
Subscriber.hh 1803
SubscriptionTransport.hh 1805
SurfaceParams.hh 1806
system.hh 1807
SystemPaths.hh 1807
Time.hh 1809
Timer.hh 1810
TopicManager.hh 1811
TransmitterVisual.hh 1813
TransmitterVisualPrivate.hh 1813
TransportIface.hh 1814
TransportTypes.hh
Forward declarations for transport 1817
UniversalJoint.hh 1818
UpdateInfo.hh 1819
UserCamera.hh 1820
UserCameraPrivate.hh 1821
UtilTypes.hh 1822
Vector2d.hh 1823
Vector2i.hh 1824
Vector3.hh 1825
Vector4.hh 1826
Video.hh 1827
VideoVisual.hh 1829

VideoVisualPrivate.hh 1830
ViewController.hh 1830
Visual.hh 1832
VisualPrivate.hh 1833
WindowManager.hh 1834
WireBox.hh 1834
WireBoxPrivate.hh 1835
WirelessReceiver.hh 1836
WirelessTransceiver.hh 1836
WirelessTransmitter.hh 1838
World.hh 1838
WorldState.hh 1840
WrenchVisual.hh 1841
WrenchVisualPrivate.hh 1842

Chapter 8

Module Documentation

8.1 Common

Output a message.

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::AudioDecoder**

  An audio decoder based on FFMPEG.
- class **gazebo::common::BVHLoader**

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

  Class used to export Collada mesh files.
- class **gazebo::common::ColladaLoader**

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

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

  Container for loggers, and global logging options (such as verbose vs.
- class **gazebo::common::Exception**

  Class for generating exceptions.

- class **gazebo::common::FileLogger**

  A logger that outputs messages to a file.
- class **gazebo::common::HeightmapData**

  Encapsulates a generic heightmap data file.
- class **gazebo::common::Image**

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

  Encapsulates an image that will be interpreted as a heightmap.
- class **gazebo::common::InternalError**

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

  Generic description of a keyboard event.
- class **gazebo::common::KeyFrame**

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

  Terminal logger.
- class **gazebo::common::Material**

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

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

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

  Base class for exporting 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 plugin with access to **physics::Model** (p. 846).*
- class **gazebo::common::MouseEvent**

  Generic description of a mouse event.
- class **gazebo::common::MovingWindowFilter**< **T** >

  *Base class for **MovingWindowFilter** (p. 890).*
- class **gazebo::common::MovingWindowFilterPrivate**< **T** >
- class **gazebo::common::NodeAnimation**

  Node animation.

- class **gazebo::common::NodeAssignment**

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

  NodeTransform *(*p. *925)* ***Skeleton.hh*** *(*p. *1787) common/common.hh*
- class **gazebo::common::NumericAnimation**

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

  A keyframe for a ***NumericAnimation*** *(*p. *936).*
- class **gazebo::common::PID**

  Generic ***PID*** *(*p. *977) controller class.*
- class **gazebo::PluginT< T >**

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

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

  A keyframe for a ***PoseAnimation*** *(*p. *1006).*
- class **gazebo::SensorPlugin**

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

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

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

  Skeleton *(*p. *1269) animation.*
- class **gazebo::common::SkeletonNode**

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

  Convert spherical coordinates for planetary surfaces.
- class **gazebo::common::SphericalCoordinatesPrivate**

  commmon/common.hh
- class **gazebo::common::STLLoader**

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

  A child mesh.
- class **gazebo::common::SystemPaths**

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

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

*A **Time** (p. 1361) class, can be used to hold wall- or sim-time.*

- class **gazebo::common::Timer**

  A timer class, used to time things in real world walltime.

- class **gazebo::common::Video**

  Handle video encoding and decoding using libavcodec.

- class **gazebo::VisualPlugin**

  A plugin loaded within the gzserver on startup.

- class **gazebo::WorldPlugin**

  *A plugin with access to **physics::World** (p. 1529).*

Namespaces

- namespace **gazebo::common**

  Common namespace.

Files

- file **CommonTypes.hh**

Defines

- #define **gzdbg** (**gazebo::common::Console::dbg**(__FILE__, __LINE__))

  Output a debug message.

- #define **gzerr** (**gazebo::common::Console::err**(__FILE__, __LINE__))

  Output an error message.

- #define **gzlog** (**gazebo::common::Console::log**())

  Output a message to a log file.

- #define **gzLogInit**(_str) (gazebo::common::Console::log.Init(_str))

  Initialize log file with filename given by _str.

- #define **gzmsg** (**gazebo::common::Console::msg**())

- #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** (**gazebo::common::Console::warn**(__FILE__, __LINE__))

  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::MovingWindowFilter< T >::MovingWindowFilter** ()

  Constructor.
- **gazebo::common::MovingWindowFilterPrivate< T >::MovingWindowFilterPrivate** ()
- virtual **gazebo::common::MovingWindowFilter< T >::~MovingWindowFilter** ()

  Destructor.
- **GAZEBO_VISIBLE** void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)

  *add path sufix to **common::SystemPaths*** (p. *1352*)
- void **gazebo::common::ModelDatabase::DownloadDependencies** (const std::string &_path)

  Download all dependencies for a give model path.
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file** (const std::string &_file)

  *search for file in **common::SystemPaths*** (p. *1352*)
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath)

  *search for file in **common::SystemPaths*** (p. *1352*)
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file_path** (const std::string &_file)

  *search for a file in **common::SystemPaths*** (p. *1352*)
- void **gazebo::common::ModelDatabase::Fini** ()

  Finalize the model database.
- T **gazebo::common::MovingWindowFilter< T >::Get** ()

  Get filtered result.
- template<typename T >
  GAZEBO_VISIBLE std::string **gazebo::common::get_sha1** (const T &_buffer)

  Compute the SHA1 hash of an array of bytes.
- std::string **gazebo::common::ModelDatabase::GetDBConfig** (const std::string &_uri)

Return the database.config file as a string.

- 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.

- event::ConnectionPtr **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::MovingWindowFilter< T >::GetWindowFilled** () const

  Get whether the window has been filled.

- unsigned int **gazebo::common::MovingWindowFilter< T >::GetWindowSize** () const

  Get the window size.

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

  Returns true if the model exists on the database.

- **GAZEBO_VISIBLE** void **gazebo::common::load** ()

  Load the common library.

- void **gazebo::common::MovingWindowFilter< T >::SetWindowSize** (unsigned int _n)

  Set window size.

- void **gazebo::common::ModelDatabase::Start** (bool _fetchImmediately=false)

  Start the model database.

- void **gazebo::common::MovingWindowFilter< T >::Update** (T _val)

  Update value of filter.

Variables

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

  String names for the pixel formats.

8.1.1 Detailed Description

Output a message.

8.1.2 Define Documentation

8.1.2.1 #define gzdbg (gazebo::common::Console::dbg(__FILE__, __LINE__))

Output a debug message.

8.1.2.2 #define gzerr (gazebo::common::Console::err(__FILE__, __LINE__))

Output an error message.

Referenced by gazebo::transport::Connection::AsyncRead(), gazebo::rendering::GUIOverlay::ButtonCallback(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::physics::GearboxJoint< T >::Load(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::SimbodyCylinderShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

8.1.2.3 #define gzlog (gazebo::common::Console::log())

Output a message to a log file.

8.1.2.4 #define gzLogInit(*_str*) (gazebo::common::Console::logInit(_str))

Initialize log file with filename given by _str.

If called twice, it will close currently in use and open a new log file.

Parameters

in	*_str*	Name of log file for gzlog messages.

8.1.2.5 #define gzmsg (gazebo::common::Console::msg())

8.1.2.6 #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::transport::-CallbackHelperT< M >::GetMsgType(), and gazebo::transport::SubscribeOptions::-Init().

8.1.2.7 #define gzwarn (gazebo::common::Console::warn(__FILE__, __LINE__))

Output a warning message.

Referenced by gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::-SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::SimbodyCylinder-Shape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

8.1.3 Enumeration Type Documentation

8.1.3.1 enum gazebo::PluginType

Used to specify the type of plugin.

Enumerator:

WORLD_PLUGIN A World plugin.

MODEL_PLUGIN A Model plugin.

SENSOR_PLUGIN A Sensor plugin.

SYSTEM_PLUGIN A System plugin.

VISUAL_PLUGIN A Visual plugin.

8.1.4 Function Documentation

8.1.4.1 template<typename T > gazebo::common::MovingWindowFilter< T >::MovingWindowFilter ( )

Constructor.

8.1.4.2 template<typename T > gazebo::common::MovingWindowFilterPrivate< T >::MovingWindowFilterPrivate ( )

FIXME hardcoded initial value for now

8.1.4.3 template<typename T > gazebo::common::MovingWindowFilter< T >::~MovingWindowFilter ( ) [virtual]

Destructor.

References NULL.

8.1.4.4 GAZEBO_VISIBLE void gazebo::common::add_search_path_suffix (const std::string & _suffix)

add path sufix to **common::SystemPaths** (p. 1352)

Parameters

in	*_suffix*	The suffix to add.

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 GAZEBO_VISIBLE std::string gazebo::common::find_file (const std::string & _file)

search for file in **common::SystemPaths** (p. 1352)

Parameters

in	*_file*	Name of the file to find.

Returns

The path containing the file.

8.1.4.7 GAZEBO_VISIBLE std::string gazebo::common::find_file (const std::string & *_file*, bool *_searchLocalPath*)

search for file in **common::SystemPaths** (p. 1352)

Parameters

in	*_file*	Name of the file to find.
in	*_searchLocalPath*	True to search in the current working directory.

Returns

The path containing the file.

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

search for a file in **common::SystemPaths** (p. 1352)

Parameters

in	*_file*	the file name to look for.

Returns

The path containing the file.

8.1.4.9 void gazebo::common::ModelDatabase::Fini ( )

Finalize the model database.

8.1.4.10 template<typename T > T gazebo::common::MovingWindowFilter< T >::Get ( )

Get filtered result.

Returns

latest filtered value

8.1.4.11 template<typename T > GAZEBO_VISIBLE std::string gazebo::common::get_sha1 (const T & _buffer)

Compute the SHA1 hash of an array of bytes.

Parameters

in	*_buffer*	Input sequence. The permitted data types for this function are std::string and any STL container.

Returns

The string representation (40 character) of the SHA1 hash.

References NULL.

8.1.4.12 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.

8.1.4.13 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.

8.1.4.14 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

8.1.4.15 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.

8.1.4.16 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	*_force-Download*	True to skip searching local paths.

Returns

path to a model directory

8.1.4.17 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. 864) to download the model list.

Returns

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

8.1.4.18 event::ConnectionPtr 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. 51)

Parameters

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

Returns

A boost shared pointer. This pointer must remain valid in order to receive the callback.

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

Returns the the global model database URI.

Returns

the URI.

8.1.4.20 template<typename T > bool gazebo::common::MovingWindowFilter< T >::GetWindowFilled ( ) const

Get whether the window has been filled.

Returns

True if the window has been filled.

8.1.4.21 template<typename T > unsigned int gazebo::common::MovingWindowFilter< T >::GetWindowSize ( ) const

Get the window size.

Returns

The size of the moving window.

8.1.4.22 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.

8.1.4.23 GAZEBO_VISIBLE void gazebo::common::load ( )

Load the common library.

8.1.4.24 template<typename T > void gazebo::common::MovingWindowFilter< T >::SetWindowSize (unsigned int _n)

Set window size.

Parameters

in	*_n*	new desired window size

8.1.4.25 void gazebo::common::ModelDatabase::Start (bool *_fetchImmediately* = false)

Start the model database.

Parameters

in	*_fetchImmediately*	True to fetch the models without waiting.

8.1.4.26 template<typename T > void gazebo::common::MovingWindowFilter< T >::Update (T *_val*)

Update value of filter.

Parameters

in	*_val*	new raw value

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. 639).

8.2 Events

Classes

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

  A class that encapsulates a connection.
- class **gazebo::event::ConnectionPrivate**
- class **gazebo::event::Event**

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

  *An **Event** (p. 514) class to get notifications for simulator events.*
- class **gazebo::event::EventT< T >**

  A class for event processing.
- class **gazebo::event::EventTPrivate< T >**

Namespaces

- namespace **gazebo::event**

  ***Event** (p. 514) namespace.*

Functions

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

  Constructor.
- 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 Function Documentation

8.2.1.1 template<typename T > gazebo::event::EventT< T >::EventT ( )

Constructor.

References gazebo::event::Event::dataPtr.

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

Destructor.

Destructor. Deletes all the associated connections.

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

Connect a callback to this event.

Adds a connection.

Parameters

in	*_subscriber*	Pointer to a callback function.

Returns

A **Connection** (p. 336) object, which will automatically call Disconnect when it goes out of scope.

Parameters

in	*_subscriber*	the subscriber to connect.

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

Get the number of connections.

Returns

Number of connection to this **Event** (p. 514).
Number of connections.

8.2.1.5 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. 516).

References NULL.

8.2.1.6 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. 517).

8.3 Math

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

Classes

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

  An angle and related functions.
- class **gazebo::math::BiQuad< T >**

  Bi-quad filter base class.
- class **gazebo::math::BiQuadVector3**

  BiQuad *(p. 217) vector3 filter.*
- class **gazebo::math::Box**

  Mathematical representation of a box and related functions.
- class **gazebo::math::Filter< T >**

  Filter *(p. 553) base class.*
- class **gazebo::math::Matrix3**

  A 3x3 matrix class.
- class **gazebo::math::Matrix4**

  A 3x3 matrix class.
- class **gazebo::math::OnePole< T >**

  A one-pole DSP filter.
- class **gazebo::math::OnePoleQuaternion**

  One-pole quaternion filter.
- class **gazebo::math::OnePoleVector3**

  One-pole vector3 filter.
- class **gazebo::math::Plane**

  A plane and related functions.
- class **gazebo::math::Pose**

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

  A quaternion class.
- class **gazebo::math::Rand**

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

  Spline *(p. 1317) 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. *1440) class represents the generic vector containing 3 elements.*

- class **gazebo::math::Vector4**

  double Generic x, y, z, w vector

Namespaces

- namespace **gazebo::math**

  Math namespace.

Files

- file **MathTypes.hh**

  Forward declarations for the math classes.

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

- float **gazebo::math::fixnan** (float _v)

  Fix a nan value.

- double **gazebo::math::fixnan** (double _v)

  Fix a nan value.

- 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
- unsigned int **gazebo::math::roundUpPowerOfTwo** (unsigned int _x)

  Get the smallest power of two that is greater or equal to a given value.
- 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)

8.3.1 Detailed Description

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

8.3.2 Function Documentation

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

Simple clamping function.

Parameters

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

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

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

check if two values are equal, within a tolerance

Parameters

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

Referenced by gazebo::math::Quaternion::Correct(), gazebo::math::Quaternion::GetInverse(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::SimbodySphereShape::SetRadius(), gazebo::physics::SimbodyBoxShape::SetSize(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::SimbodyCylinderShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

8.3.2.3 float gazebo::math::fixnan (float _v) [inline]

Fix a nan value.

Parameters

in	*_v*	Value to correct.

Returns

0 if _v is NaN, _v otherwise.

References gazebo::math::isnan().

8.3.2.4 double gazebo::math::fixnan (double _v) [inline]

Fix a nan value.

Parameters

in	*_v*	Value to correct.

Returns

0 if _v is NaN, _v otherwise.

References gazebo::math::isnan().

8.3.2.5 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::fixnan(), and gazebo::math::isnan().

8.3.2.6 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().

8.3.2.7 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

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

8.3.2.8 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().

8.3.2.9 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

8.3.2.10 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().

8.3.2.11 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.

8.3.2.12 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.

8.3.2.13 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

8.3.2.14 unsigned int gazebo::math::roundUpPowerOfTwo (unsigned int _x) [inline]

Get the smallest power of two that is greater or equal to a given value.

Parameters

in	_x	the number

Returns

the same value if _x is already a power of two. Otherwise, it returns the smallest power of two that is greater than _x

References gazebo::math::isPowerOfTwo().

8.3.2.15 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

8.3.3 Variable Documentation

8.3.3.1 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().

8.3.3.2 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().

8.4 Messages

All messages and helper functions.

Classes

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

  *Google protobuf message generator for **gazebo::msgs*** (p. *135*).
- class **gazebo::msgs::MsgFactory**

  A factory that generates protobuf message based on a string type.

Namespaces

- namespace **gazebo::msgs**

  Messages namespace.

Defines

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

  Static message registration macro.

Functions

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

  *Convert a **math::Vector3*** (p. *1440*) *to a msgs::Vector3d.*
- **GAZEBO_VISIBLE** msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)

  *Convert a **math::Quaternion*** (p. *1029*) *to a msgs::Quaternion.*
- **GAZEBO_VISIBLE** msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)

  *Convert a **math::Pose*** (p. *995*) *to a msgs::Pose.*
- **GAZEBO_VISIBLE** msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)

  *Convert a **common::Color*** (p. *312*) *to a msgs::Color.*
- **GAZEBO_VISIBLE** msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)

  *Convert a **common::Time*** (p. *1361*) *to a msgs::Time.*
- **GAZEBO_VISIBLE** msgs::PlaneGeom **gazebo::msgs::Convert** (const math::Plane &_p)

*Convert a **math::Plane** (p. 984) to a msgs::PlaneGeom.*

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

  Convert a msgs::Vector3d to a math::Vector.
- **GAZEBO_VISIBLE** math::Quaternion **gazebo::msgs::Convert** (const msgs::Quaternion &_q)

  *Convert a msgs::Quaternion to a **math::Quaternion** (p. 1029).*
- **GAZEBO_VISIBLE** math::Pose **gazebo::msgs::Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose** (p. 995).*
- **GAZEBO_VISIBLE** common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color** (p. 312).*
- **GAZEBO_VISIBLE** common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time** (p. 1361).*
- **GAZEBO_VISIBLE** math::Plane **gazebo::msgs::Convert** (const msgs::PlaneGeom &_p)

  Convert a msgs::PlaneGeom to a common::Plane.
- **GAZEBO_VISIBLE** msgs::Request * **gazebo::msgs::CreateRequest** (const std::string &_request, const std::string &_data="")

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

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

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

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

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

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

  Create a msgs::Light from a light SDF element.
- **GAZEBO_VISIBLE** sdf::ElementPtr **gazebo::msgs::LightToSDF** (const msgs::Light &_msg, sdf::ElementPtr _sdf=sdf::ElementPtr())

Create an SDF element from a msgs::Scene.

- **GAZEBO_VISIBLE** msgs::MeshGeom **gazebo::msgs::MeshFromSDF** (sdf::ElementPtr _sdf)

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

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

  *Convert a msgs::Image to a **common::Image** (p. 637).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)

  *Set a msgs::Image from a **common::Image** (p. 637).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 1440).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 1440).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 1029).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 995).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)

  *Set a msgs::Color from a **common::Color** (p. 312).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)

  *Set a msgs::Time from a **common::Time** (p. 1361).*
- void **gazebo::msgs::Set** (msgs::SphericalCoordinates *_s, const common::SphericalCoordinates &_v)

  *Set a msgs::SphericalCoordinates from a **common::SphericalCoordinates** (p. 1310) object.*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::PlaneGeom *_p, const math::Plane &_v)

  *Set a msgs::Plane from a **math::Plane** (p. 984).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Stamp** (msgs::Header *_header)

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

Set the time in a time message.

- **GAZEBO_VISIBLE** msgs::TrackVisual **gazebo::msgs::TrackVisualFromSDF** (sdf::ElementPtr _sdf)

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

- **GAZEBO_VISIBLE** msgs::Visual **gazebo::msgs::VisualFromSDF** (sdf::ElementPtr _sdf)

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

8.4.1 Detailed Description

All messages and helper functions.

8.4.2 Define Documentation

8.4.2.1 #define GZ_REGISTER_STATIC_MSG(*_msgtype, _classname*)

Value:

```
GAZEBO_VISIBLE \
    boost::shared_ptr<google::protobuf::Message> New##_classname() \
    { \
      return boost::shared_ptr<gazebo::msgs::_classname>(\
          new gazebo::msgs::_classname); \
    } \
    class GAZEBO_VISIBLE 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.

8.4.3 Function Documentation

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

Convert a **math::Vector3** (p. 1440) to a msgs::Vector3d.

Parameters

in	*_v*	The vector to convert

Returns

A msgs::Vector3d object

8.4.3.2 GAZEBO_VISIBLE msgs::Quaternion gazebo::msgs::Convert (const math::Quaternion & _q)

Convert a **math::Quaternion** (p. 1029) to a msgs::Quaternion.

Parameters

in	*_q*	The quaternion to convert

Returns

A msgs::Quaternion object

8.4.3.3 GAZEBO_VISIBLE msgs::Pose gazebo::msgs::Convert (const math::Pose & _p)

Convert a **math::Pose** (p. 995) to a msgs::Pose.

Parameters

in	*_p*	The pose to convert

Returns

A msgs::Pose object

8.4.3.4 GAZEBO_VISIBLE msgs::Color gazebo::msgs::Convert (const common::Color & _c)

Convert a **common::Color** (p. 312) to a msgs::Color.

Parameters

in	*_c*	The color to convert

Returns

A msgs::Color object

8.4.3.5 GAZEBO_VISIBLE msgs::Time gazebo::msgs::Convert (const common::Time & _t)

Convert a **common::Time** (p. 1361) to a msgs::Time.

Parameters

in	*_t*	The time to convert

Returns

A msgs::Time object

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

Convert a **math::Plane** (p. 984) to a msgs::PlaneGeom.

Parameters

in	*_p*	The plane to convert

Returns

A msgs::PlaneGeom object

8.4.3.7 GAZEBO_VISIBLE 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 **math::Vector3** (p. 1440) object

8.4.3.8 GAZEBO_VISIBLE math::Quaternion gazebo::msgs::Convert (const msgs::Quaternion & _q)

Convert a msgs::Quaternion to a **math::Quaternion** (p. 1029).

Parameters

in	_q	The quaternion to convert

Returns

A **math::Quaternion** (p. 1029) object

8.4.3.9 GAZEBO_VISIBLE math::Pose gazebo::msgs::Convert (const msgs::Pose & _p)

Convert a msgs::Pose to a **math::Pose** (p. 995).

Parameters

in	_q	The pose to convert

Returns

A **math::Pose** (p. 995) object

8.4.3.10 GAZEBO_VISIBLE common::Color gazebo::msgs::Convert (const msgs::Color & _c)

Convert a msgs::Color to a **common::Color** (p. 312).

Parameters

in	_c	The color to convert

Returns

A **common::Color** (p. 312) object

8.4.3.11 GAZEBO_VISIBLE common::Time gazebo::msgs::Convert (const msgs::Time & _t)

Convert a msgs::Time to a **common::Time** (p. 1361).

Parameters

in	_t	The time to convert

Returns

A **common::Time** (p. 1361) object

8.4.3.12 GAZEBO_VISIBLE 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 common::Plane object

8.4.3.13 GAZEBO_VISIBLE 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 Request message

8.4.3.14 GAZEBO_VISIBLE 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

8.4.3.15 GAZEBO_VISIBLE 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

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

Get the header from a protobuf message.

Parameters

in	*_message*	A google protobuf message

Returns

A pointer to the message's header

8.4.3.17 GAZEBO_VISIBLE 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

8.4.3.18 GAZEBO_VISIBLE 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< SimbodyJoint >::Init(), gazebo::physics::BallJoint< SimbodyJoint >::Init(), gazebo::physics::UniversalJoint< SimbodyJoint >::Init(), gazebo::physics::ScrewJoint< SimbodyJoint >::Init(), and gazebo::physics::GearboxJoint< T >::Init().

8.4.3.19 GAZEBO_VISIBLE 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

8.4.3.20 GAZEBO_VISIBLE sdf::ElementPtr gazebo::msgs::LightToSDF (const msgs::Light & *_msg*, sdf::ElementPtr *_sdf* = `sdf::ElementPtr()`)

Create an SDF element from a msgs::Scene.

Parameters

in	*_msg*	Light messsage
in	*_sdf*	if supplied, performs an update from _msg intead of creating a new sdf element.

Returns

The new SDF element

8.4.3.21 GAZEBO_VISIBLE 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

8.4.3.22 GAZEBO_VISIBLE 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

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

Convert a msgs::Image to a **common::Image** (p. 637).

Parameters

out	*_img*	The **common::Image** (p. 637) container
in	*_msg*	The Image message to convert

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

Set a msgs::Image from a **common::Image** (p. 637).

Parameters

out	*_msg*	A msgs::Image pointer
in	*_i*	A **common::Image** (p. 637) reference

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

Set a msgs::Vector3d from a **math::Vector3** (p. 1440).

Parameters

out	*_pt*	A msgs::Vector3d pointer
in	*_v*	A **math::Vector3** (p. 1440) reference

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

Set a msgs::Vector2d from a **math::Vector3** (p. 1440).

Parameters

out	*_pt*	A msgs::Vector2d pointer
in	*_v*	A **math::Vector2d** (p. 1420) reference

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

Set a msgs::Quaternion from a **math::Quaternion** (p. 1029).

Parameters

out	*_q*	A msgs::Quaternion pointer
in	*_v*	A **math::Quaternion** (p. 1029) reference

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

Set a msgs::Pose from a **math::Pose** (p. 995).

Parameters

out	*_p*	A msgs::Pose pointer
in	*_v*	A **math::Pose** (p. 995) reference

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

Set a msgs::Color from a **common::Color** (p. 312).

Parameters

out	*_p*	A msgs::Color pointer
in	*_v*	A **common::Color** (p. 312) reference

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

Set a msgs::Time from a **common::Time** (p. 1361).

Parameters

out	*_p*	A msgs::Time pointer
in	*_v*	A **common::Time** (p. 1361) reference

8.4.3.31 void gazebo::msgs::Set (msgs::SphericalCoordinates * _s, const common::SphericalCoordinates & _v)

Set a msgs::SphericalCoordinates from a **common::SphericalCoordinates** (p. 1310) object.

Parameters

out	*_p*	A msgs::SphericalCoordinates pointer.
in	*_v*	A **common::SphericalCoordinates** (p. 1310) reference

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

Set a msgs::Plane from a **math::Plane** (p. 984).

Parameters

out	*_p*	A msgs::Plane pointer
in	*_v*	A **math::Plane** (p. 984) reference

8.4.3.33 GAZEBO_VISIBLE void gazebo::msgs::Stamp (msgs::Header * _header)

Time stamp a header.

Parameters

in	*_header*	Header to stamp

8.4.3.34 GAZEBO_VISIBLE void gazebo::msgs::Stamp (msgs::Time * _time)

Set the time in a time message.

Parameters

in	*_time*	A Time message

8.4.3.35 GAZEBO_VISIBLE 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

8.4.3.36 GAZEBO_VISIBLE msgs::Visual gazebo::msgs::VisualFromSDF (sdf::ElementPtr _sdf)

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

Parameters

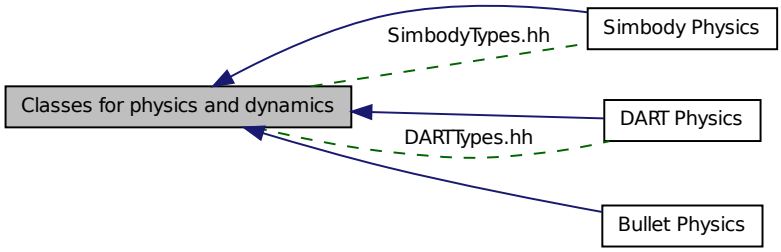
in	*_sdf*	The sdf element

Returns

The new msgs::Visual object

8.5 Classes for physics and dynamics

Collaboration diagram for Classes for physics and dynamics:



Classes

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

  Actor (p. *165) class enables GPU based mesh model / skeleton scriptable animation.*
- class **gazebo::physics::BallJoint**< **T** >

  Base (p. *201) class for a ball joint.*
- class **gazebo::physics::Base**

  Base (p. *201) class for most physics classes.*
- class **gazebo::physics::BoxShape**

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

  Base (p. *201) class for all collision entities.*
- class **gazebo::physics::CollisionState**

  Store state information of a ***physics::Collision*** (p. *295) object.*
- class **gazebo::physics::Contact**

  A contact between two collisions.
- class **gazebo::physics::ContactManager**

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

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

  Base (p. *201) class for all physics objects in Gazebo.*
- class **gazebo::physics::FrictionPyramid**

Parameters used for friction pyramid model.

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

  A double axis gearbox joint.

- class **gazebo::physics::Gripper**

  A gripper abstraction.

- class **gazebo::physics::HeightmapShape**

  HeightmapShape *(*p. *628) collision shape builds a heightmap from an image.*

- class **gazebo::physics::Hinge2Joint< T >**

  A two axis hinge joint.

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

  A single axis hinge joint.

- class **gazebo::physics::Inertial**

  A class for inertial information about a link.

- class **gazebo::physics::Joint**

  Base *(*p. *201) class for all joints.*

- class **gazebo::physics::JointController**

  A class for manipulating ***physics::Joint*** *(*p. *669).*

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

  keeps track of state of a ***physics::Joint*** *(*p. *669)*

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

  Wrench information from a joint.

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

  Link *(*p. *739) 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. *739) object.*

- class **Logplay**

  Open and playback log files that were recorded using LogRecord.

- class **gazebo::util::LogPlay**

- class **gazebo::physics::MeshShape**

  Triangle mesh collision shape.

- class **gazebo::physics::Model**

  A model is a collection of links, joints, and plugins.

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

  Store state information of a ***physics::Model*** *(*p. *846) 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. *201) class for a physics engine.*

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

  The physics factory instantiates different physics engines.
- class **gazebo::physics::PlaneShape**

  Collision *(*p. *295) for an infinite plane.*
- class **gazebo::physics::RayShape**

  Base (p. *201) class for Ray collision geometry.*
- class **gazebo::physics::Road**

  for building a ***Road*** (p. *1083) from SDF*
- class **gazebo::physics::ScrewJoint< T >**

  A screw joint, which has both prismatic and rotational DOFs.
- class **gazebo::physics::Shape**

  Base (p. *201) class for all shapes.*
- class **gazebo::physics::SimbodyModel**

  A model is a collection of links, joints, and plugins.
- class **gazebo::physics::SliderJoint< T >**

  A slider joint.
- class **gazebo::physics::SphereShape**

  Sphere collision shape.
- class **gazebo::physics::State**

  State (p. *1323) of an entity.*
- class **gazebo::physics::SurfaceParams**

  SurfaceParams (p. *1350) defines various Surface contact parameters.*
- class **gazebo::physics::UniversalJoint< T >**

  A universal joint.
- class **gazebo::physics::World**

  The world provides access to all other object within a simulated environment.
- class **gazebo::physics::WorldState**

  Store state information of a ***physics::World*** (p. *1529) object.*

Namespaces

- namespace **gazebo::physics**

  namespace for physics

Modules

- **DART Physics**

  dart physics engine wrapper
- **Bullet Physics**
- **Simbody Physics**

  simbody physics engine wrapper

Files

- file **DARTTypes.hh**

  DART wrapper forward declarations and typedefs.
- file **PhysicsTypes.hh**

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

  Simbody wrapper forward declarations and typedefs.

Defines

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

  Static physics registration macro.

Typedefs

- typedef PhysicsEnginePtr(∗ **gazebo::physics::PhysicsFactoryFn**)(WorldPtr world)

Functions

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

  Create a world given a name.
- **GAZEBO_VISIBLE** bool **gazebo::physics::fini** ()

  *Finalize transport by calling **gazebo::transport::fini** (p. 113).*
- **GAZEBO_VISIBLE** WorldPtr **gazebo::physics::get_world** (const std::string &-_name="")

  Returns a pointer to a world by name.
- **GAZEBO_VISIBLE** uint32_t **gazebo::physics::getUniqueId** ()

  Get a unique ID.
- **GAZEBO_VISIBLE** void **gazebo::physics::init_world** (WorldPtr _world)

  Init world given a pointer to it.
- **GAZEBO_VISIBLE** void **gazebo::physics::init_worlds** ()

  initialize multiple worlds stored in static variable gazebo::g_worlds
- **GAZEBO_VISIBLE** bool **gazebo::physics::load** ()

  *Setup **gazebo::SystemPlugin** (p. 1359)'s and call **gazebo::transport::init** (p. 115).*
- **GAZEBO_VISIBLE** void **gazebo::physics::load_world** (WorldPtr _world, sdf::-ElementPtr _sdf)

  Load world from sdf::Element pointer.

- **GAZEBO_VISIBLE** void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)

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

  *Pause world by calling **World::SetPaused*** (p. *1542).*
- **GAZEBO_VISIBLE** void **gazebo::physics::pause_worlds** (bool pause)

  pause multiple worlds stored in static variable gazebo::g_worlds
- **GAZEBO_VISIBLE** void **gazebo::physics::remove_worlds** ()

  remove multiple worlds stored in static variable gazebo::g_worlds
- **GAZEBO_VISIBLE** void **gazebo::physics::run_world** (WorldPtr _world, unsigned int _iterations=0)

  *Run world by calling **World::Run()*** (p. *1541) given a pointer to it.*
- **GAZEBO_VISIBLE** void **gazebo::physics::run_worlds** (unsigned int _iterations=0)

  Run multiple worlds stored in static variable gazebo::g_worlds.
- **GAZEBO_VISIBLE** void **gazebo::physics::stop_world** (WorldPtr _world)

  *Stop world by calling **World::Stop()*** (p. *1543) given a pointer to it.*
- **GAZEBO_VISIBLE** void **gazebo::physics::stop_worlds** ()

  stop multiple worlds stored in static variable gazebo::g_worlds
- **GAZEBO_VISIBLE** bool **gazebo::physics::worlds_running** ()

  Return true if any world is running.

Variables

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

  String names for the different entity types.

8.5.1 Define Documentation

8.5.1.1 #define GZ_REGISTER_PHYSICS_ENGINE(*name, classname*)

Value:

```
GAZEBO_VISIBLE PhysicsEnginePtr New##classname(WorldPtr _world) \
    { \
      return PhysicsEnginePtr(new gazebo::physics::classname(_world)); \
    } \
    GAZEBO_VISIBLE \
    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.

8.5.2 Typedef Documentation

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

8.5.3 Function Documentation

8.5.3.1 GAZEBO_VISIBLE 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.

8.5.3.2 GAZEBO_VISIBLE bool gazebo::physics::fini ( )

Finalize transport by calling **gazebo::transport::fini** (p. 113).

8.5.3.3 GAZEBO_VISIBLE 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.

8.5.3.4 GAZEBO_VISIBLE uint32_t gazebo::physics::getUniqueId ( )

Get a unique ID.

Returns

A unique integer

8.5.3.5 GAZEBO_VISIBLE void gazebo::physics::init_world (WorldPtr *_world*)

Init world given a pointer to it.

Parameters

in	*_world*	**World** (p. 1529) to initialize.

8.5.3.6 GAZEBO_VISIBLE void gazebo::physics::init_worlds ( )

initialize multiple worlds stored in static variable gazebo::g_worlds

8.5.3.7 GAZEBO_VISIBLE bool gazebo::physics::load ( )

Setup **gazebo::SystemPlugin** (p. 1359)'s and call **gazebo::transport::init** (p. 115).

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

Load world from sdf::Element pointer.

Parameters

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

8.5.3.9 GAZEBO_VISIBLE void gazebo::physics::load_worlds (sdf::ElementPtr *_sdf*)

load multiple worlds from single sdf::Element pointer

Parameters

in	*_sdf*	SDF values used to create worlds.

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

Pause world by calling **World::SetPaused** (p. 1542).

Parameters

in	*_world*	**World** (p. 1529) to pause or unpause.
in	*_pause*	True to pause, False to unpause.

8.5.3.11 GAZEBO_VISIBLE 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.

8.5.3.12 GAZEBO_VISIBLE void gazebo::physics::remove_worlds ( )

remove multiple worlds stored in static variable gazebo::g_worlds

8.5.3.13 GAZEBO_VISIBLE void gazebo::physics::run_world (WorldPtr *_world*, unsigned int *_iterations* = 0)

Run world by calling **World::Run()** (p. 1541) given a pointer to it.

Parameters

in	*_world*	**World** (p. 1529) to run.
in	*_iterations*	Number of iterations for each world to take. Zero indicates that each world should continue forever.

8.5.3.14 GAZEBO_VISIBLE void gazebo::physics::run_worlds (unsigned int _iterations = 0)

Run multiple worlds stored in static variable gazebo::g_worlds.

Parameters

in	*_iterations*	Number of iterations for each world to take. Zero indicates that each world should continue forever.

8.5.3.15 GAZEBO_VISIBLE void gazebo::physics::stop_world (WorldPtr _world)

Stop world by calling **World::Stop()** (p. 1543) given a pointer to it.

Parameters

in	*_world*	**World** (p. 1529) to stop.

8.5.3.16 GAZEBO_VISIBLE void gazebo::physics::stop_worlds ( )

stop multiple worlds stored in static variable gazebo::g_worlds

8.5.3.17 GAZEBO_VISIBLE bool gazebo::physics::worlds_running ( )

Return true if any world is running.

Returns

True if any world is running.

8.5.4 Variable Documentation

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

Initial value:

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

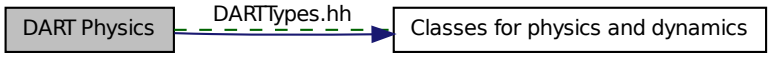
```
    "visual",
    "joint",
    "ball",
    "hinge2",
    "hinge",
    "slider",
    "universal",
    "shape",
    "box",
    "cylinder",
    "heightmap",
    "map",
    "multiray",
    "ray",
    "plane",
    "sphere",
    "trimesh"
  }
```

String names for the different entity types.

8.6 DART Physics

dart physics engine wrapper

Collaboration diagram for DART Physics:



Classes

- class **gazebo::physics::DARTLink**

  *DART **Link** (p. 739) class.*
- class **gazebo::physics::DARTModel**

  DART model class.
- class **gazebo::physics::DARTPhysics**

  DART physics engine.
- class **gazebo::physics::DARTRayShape**

  Ray collision.
- class **gazebo::physics::DARTTypes**

  A set of functions for converting between the math types used by gazebo and dart.

Files

- file **DARTTypes.hh**

  DART wrapper forward declarations and typedefs.

Functions

- **gazebo::physics::DARTRayShape::DARTRayShape** (PhysicsEnginePtr _physicsEngine)

  Constructor for a global ray.
- **gazebo::physics::DARTRayShape::DARTRayShape** (CollisionPtr _collision)

  Constructor.
- virtual **gazebo::physics::DARTRayShape::∼DARTRayShape** ()

Destructor.

- static Eigen::Isometry3d **gazebo::physics::DARTTypes::ConvPose** (const math::Pose &_pose)
- static math::Pose **gazebo::physics::DARTTypes::ConvPose** (const Eigen::-Isometry3d &_T)
- static Eigen::Quaterniond **gazebo::physics::DARTTypes::ConvQuat** (const math::Quaternion &_quat)
- static math::Quaternion **gazebo::physics::DARTTypes::ConvQuat** (const -Eigen::Quaterniond &_quat)
- static Eigen::Vector3d **gazebo::physics::DARTTypes::ConvVec3** (const math-::Vector3 &_vec3)
- static math::Vector3 **gazebo::physics::DARTTypes::ConvVec3** (const Eigen::-Vector3d &_vec3)
- virtual void **gazebo::physics::DARTRayShape::GetIntersection** (double &_-dist, std::string &_entity)

  Get the nearest intersection.
- virtual void **gazebo::physics::DARTRayShape::SetPoints** (const math::Vector3 &_posStart, const math::Vector3 &_posEnd)

  Set the ray based on starting and ending points relative to the body.
- virtual void **gazebo::physics::DARTRayShape::Update** ()

  Update the ray collision.

8.6.1 Detailed Description

dart physics engine wrapper

8.6.2 Function Documentation

8.6.2.1 gazebo::physics::DARTRayShape::DARTRayShape (PhysicsEnginePtr *_physicsEngine*) [explicit]

Constructor for a global ray.

Parameters

in	*_physics-Engine*	Pointer to the physics engine.

8.6.2.2 gazebo::physics::DARTRayShape::DARTRayShape (CollisionPtr *_collision*) [explicit]

Constructor.

Parameters

in	*_collision*	**Collision** (p. 295) object this ray is attached to.

8.6.2.3 virtual gazebo::physics::DARTRayShape::~DARTRayShape ( ) [virtual]

Destructor.

8.6.2.4 static Eigen::Isometry3d gazebo::physics::DARTTypes::ConvPose (const math::Pose & *_pose*) [inline, static]

References gazebo::math::Pose::pos, and gazebo::math::Pose::rot.

8.6.2.5 static math::Pose gazebo::physics::DARTTypes::ConvPose (const Eigen::Isometry3d & *_T*) [inline, static]

References gazebo::math::Pose::pos, and gazebo::math::Pose::rot.

8.6.2.6 static Eigen::Quaterniond gazebo::physics::DARTTypes::ConvQuat (const math::Quaternion & *_quat*) [inline, static]

References gazebo::math::Quaternion::w, gazebo::math::Quaternion::x, gazebo::math::Quaternion::y, and gazebo::math::Quaternion::z.

8.6.2.7 static math::Quaternion gazebo::physics::DARTTypes::ConvQuat (const Eigen::Quaterniond & *_quat*) [inline, static]

8.6.2.8 static Eigen::Vector3d gazebo::physics::DARTTypes::ConvVec3 (const math::Vector3 & *_vec3*) [inline, static]

References gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

Referenced by gazebo::physics::DARTBoxShape::SetSize().

8.6.2.9 static math::Vector3 gazebo::physics::DARTTypes::ConvVec3 (const Eigen::Vector3d & _vec3) [inline, static]

8.6.2.10 virtual void gazebo::physics::DARTRayShape::GetIntersection (double & _dist, std::string & _entity) [virtual]

Get the nearest intersection.

Parameters

out	*_dist*	Distance to the intersection.
out	*_entity*	Name of the entity that was hit.

Implements **gazebo::physics::RayShape** (p. 1063).

8.6.2.11 virtual void gazebo::physics::DARTRayShape::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

Reimplemented from **gazebo::physics::RayShape** (p. 1065).

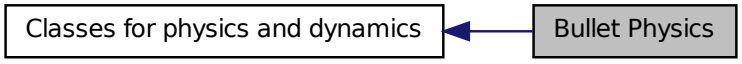
8.6.2.12 virtual void gazebo::physics::DARTRayShape::Update ( ) [virtual]

Update the ray collision.

Implements **gazebo::physics::RayShape** (p. 1065).

8.7 Bullet Physics

Collaboration diagram for Bullet Physics:



Classes

- class **gazebo::physics::DARTMultiRayShape**

  *DART specific version of **MultiRayShape*** (p. *901*).

Functions

- **gazebo::physics::DARTMultiRayShape::DARTMultiRayShape** (CollisionPtr _parent)

  Constructor.
- virtual **gazebo::physics::DARTMultiRayShape::~DARTMultiRayShape** ()

  Destructor.
- void **gazebo::physics::DARTMultiRayShape::AddRay** (const math::Vector3 &_start, const math::Vector3 &_end)

  Add a ray to the collision.
- virtual void **gazebo::physics::DARTMultiRayShape::UpdateRays** ()

  Physics engine specific method for updating the rays.

8.7.1 Function Documentation

8.7.1.1 gazebo::physics::DARTMultiRayShape::DARTMultiRayShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent **Collision** (p. 295).

8.7.1.2 virtual gazebo::physics::DARTMultiRayShape::~DARTMultiRayShape ( ) [virtual]

Destructor.

8.7.1.3 void gazebo::physics::DARTMultiRayShape::AddRay (const math::Vector3 & *_start*, const math::Vector3 & *_end*) [protected, virtual]

Add a ray to the collision.

Parameters

in	*_start*	Start location of the ray.
in	*_end*	End location of the ray.

Reimplemented from **gazebo::physics::MultiRayShape** (p. 904).

8.7.1.4 virtual void gazebo::physics::DARTMultiRayShape::UpdateRays ( ) [virtual]

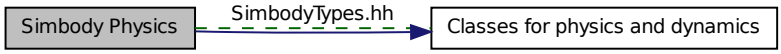
Physics engine specific method for updating the rays.

Implements **gazebo::physics::MultiRayShape** (p. 909).

8.8 Simbody Physics

simbody physics engine wrapper

Collaboration diagram for Simbody Physics:



Classes

- class **gazebo::physics::SimbodyBallJoint**

  SimbodyBallJoint *(*p. *1165) class models a ball joint in Simbody.*
- class **gazebo::physics::SimbodyBoxShape**

  Simbody box collision.
- class **gazebo::physics::SimbodyCollision**

  Simbody collisions.
- class **gazebo::physics::SimbodyCylinderShape**

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

  Height map collision.
- class **gazebo::physics::SimbodyHinge2Joint**

  A two axis hinge joint.
- class **gazebo::physics::SimbodyHingeJoint**

  A single axis hinge joint.
- class **gazebo::physics::SimbodyJoint**

  Base *(*p. *201) class for all joints.*
- class **gazebo::physics::SimbodyLink**

  Simbody ***Link*** *(*p. *739) class.*
- class **gazebo::physics::SimbodyMeshShape**

  Triangle mesh collision.
- class **gazebo::physics::SimbodyMultiRayShape**

  Simbody specific version of ***MultiRayShape*** *(*p. *901).*
- class **gazebo::physics::SimbodyPhysics**

  Simbody physics engine.
- class **gazebo::physics::SimbodyPlaneShape**

Simbody collision for an infinite plane.

- class **gazebo::physics::SimbodyRayShape**

  Ray shape for simbody.
- class **gazebo::physics::SimbodyScrewJoint**

  A screw joint.
- class **gazebo::physics::SimbodySliderJoint**

  A slider joint.
- class **gazebo::physics::SimbodySphereShape**

  Simbody sphere collision.
- class **gazebo::physics::SimbodyUniversalJoint**

  A simbody universal joint class.

Files

- file **SimbodyTypes.hh**

  Simbody wrapper forward declarations and typedefs.

8.8.1 Detailed Description

simbody physics engine wrapper

8.9 Rendering

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

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. *367)* ***Conversions.hh*** *(*p. *1608)* ***rendering/Conversions.hh*** *(*p. *1608).*
- class **gazebo::rendering::DepthCamera**

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

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

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

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

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

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

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

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

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

Visualization for joints.

- class **gazebo::rendering::LaserVisual**

  Visualization for laser data.
- class **gazebo::rendering::Light**

  A light source.
- class **gazebo::rendering::MovableText**

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

  A camera used for user visualization of a scene.
- 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. 163)'s Run-Time Shader system.*
- class **gazebo::rendering::Scene**

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

  Interactive selection object for models and links.
- class **gazebo::rendering::SonarVisual**

  Visualization for sonar data.
- class **gazebo::rendering::TransmitterVisual**

  Visualization for the wireless propagation data.
- class **gazebo::rendering::UserCamera**

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

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

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

  A renderable object.

- class **gazebo::rendering::WindowManager**

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

  Draws a wireframe box.
- class **gazebo::rendering::WrenchVisual**

  Visualization for sonar data.

Namespaces

- namespace **gazebo::rendering**

  Rendering namespace.

Functions

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

  *create **rendering::Scene*** (p. *1097) by name.*
- **GAZEBO_VISIBLE** bool **gazebo::rendering::fini** ()

  teardown rendering engine.
- **GAZEBO_VISIBLE** rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name="")

  *get pointer to **rendering::Scene*** (p. *1097) by name.*
- **GAZEBO_VISIBLE** bool **gazebo::rendering::init** ()

  init rendering engine.
- **GAZEBO_VISIBLE** bool **gazebo::rendering::load** ()

  load rendering engine.
- **GAZEBO_VISIBLE** void **gazebo::rendering::remove_scene** (const std::string &_name)

  *remove a **rendering::Scene*** (p. *1097) by name*

8.9.1 Detailed Description

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

8.9.2 Function Documentation

8.9.2.1 GAZEBO_VISIBLE rendering::ScenePtr gazebo::rendering::create_scene (const std::string & *_name,* bool *_enableVisualizations,* bool *_isServer* = `false`)

create **rendering::Scene** (p. 1097) by name.

Parameters

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

8.9.2.2 GAZEBO_VISIBLE bool gazebo::rendering::fini ( )

teardown rendering engine.

8.9.2.3 GAZEBO_VISIBLE rendering::ScenePtr gazebo::rendering::get_scene (const std::string & *_name* = " ")

get pointer to **rendering::Scene** (p. 1097) by name.

Parameters

in	*_name*	Name of the scene to retreive.

8.9.2.4 GAZEBO_VISIBLE bool gazebo::rendering::init ( )

init rendering engine.

8.9.2.5 GAZEBO_VISIBLE bool gazebo::rendering::load ( )

load rendering engine.

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

remove a **rendering::Scene** (p. 1097) by name

Parameters

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

8.10 Sensors

A set of sensor classes, functions, and definitions.

Classes

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

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

  Contact sensor.
- class **gazebo::sensors::DepthCameraSensor**
- class **gazebo::sensors::ForceTorqueSensor**

  Sensor *(*p. *1130) for measure force and torque on a joint.*
- class **gazebo::sensors::GpsSensor**

  GpsSensor *(*p. *575) to provide position measurement.*
- class **gazebo::sensors::GpuRaySensor**
- class **gazebo::sensors::ImuSensor**

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

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

  Noise *(*p. *931) models for sensor output signals.*
- class **gazebo::sensors::NoiseFactory**

  Use this noise manager for creating and loading noise models.
- class **gazebo::sensors::RaySensor**

  Sensor *(*p. *1130) with one or more rays.*
- class **gazebo::sensors::RFIDSensor**

  Sensor *(*p. *1130) class for RFID type of sensor.*
- class **gazebo::sensors::RFIDTag**

  RFIDTag *(*p. *1075) 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.
- class **gazebo::sensors::SonarSensor**

  Sensor *(*p. *1130) with sonar cone.*
- class **gazebo::sensors::WirelessReceiver**

Sensor *(*p. *1130) class for receiving wireless signals.*

- class **gazebo::sensors::WirelessTransceiver**

  Sensor *(*p. *1130) class for receiving wireless signals.*

- class **gazebo::sensors::WirelessTransmitter**

  Transmitter to send wireless signals.

Namespaces

- namespace **gazebo::sensors**

  Sensors namespace.

Files

- file **SensorTypes.hh**

  Forward declarations and typedefs for sensors.

Defines

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

  Static sensor registration macro.

Functions

- **GAZEBO_VISIBLE** std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)

  Create a sensor using SDF.

- **GAZEBO_VISIBLE** void **gazebo::sensors::disable** ()

  Disable sensors.

- **GAZEBO_VISIBLE** void **gazebo::sensors::enable** ()

  Enable sensors.

- **GAZEBO_VISIBLE** bool **gazebo::sensors::fini** ()

  shutdown the sensor generation loop.

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

  Get a sensor using by name.

- **GAZEBO_VISIBLE** bool **gazebo::sensors::init** ()

  initialize the sensor generation loop.

- **GAZEBO_VISIBLE** bool **gazebo::sensors::load** ()

Load the sensor library.

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

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

  Remove all sensors.
- **GAZEBO_VISIBLE** void **gazebo::sensors::run_once** (bool _force=false)

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

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

  Stop the sensor generation loop.

8.10.1 Detailed Description

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

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

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

8.10.2 Define Documentation

8.10.2.1 #define GZ_REGISTER_STATIC_SENSOR(*name, classname*)

Value:

```
GAZEBO_VISIBLE Sensor *New##classname() \
  { \
    return new gazebo::sensors::classname(); \
  } \
  GAZEBO_VISIBLE \
  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.

8.10.3 Function Documentation

8.10.3.1 GAZEBO_VISIBLE std::string gazebo::sensors::create_sensor (sdf::ElementPtr _elem, const std::string & _worldName, const std::string & _parentName, uint32_t _parentId)

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.

8.10.3.2 GAZEBO_VISIBLE void gazebo::sensors::disable ( )

Disable sensors.

8.10.3.3 GAZEBO_VISIBLE void gazebo::sensors::enable ( )

Enable sensors.

8.10.3.4 GAZEBO_VISIBLE bool gazebo::sensors::fini ( )

shutdown the sensor generation loop.

Returns

True if successfully finalized, false if not

8.10.3.5 GAZEBO_VISIBLE 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.

8.10.3.6 GAZEBO_VISIBLE bool gazebo::sensors::init ( )

initialize the sensor generation loop.

Returns

True if successfully initialized, false if not

8.10.3.7 GAZEBO_VISIBLE bool gazebo::sensors::load ( )

Load the sensor library.

Returns

True if successfully loaded, false if not.

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

Remove a sensor by name.

Parameters

in	*_sensorName*	Name of sensor to remove

8.10.3.9 GAZEBO_VISIBLE bool gazebo::sensors::remove_sensors ( )

Remove all sensors.

Returns

True if all successfully removed, false if not

8.10.3.10 GAZEBO_VISIBLE 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.

8.10.3.11 GAZEBO_VISIBLE void gazebo::sensors::run_threads ( )

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

8.10.3.12 GAZEBO_VISIBLE void gazebo::sensors::stop ( )

Stop the sensor generation loop.

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

8.11 Transport

Handles transportation of messages.

Classes

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

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

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

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

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

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

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

  A publication for a topic.
- class **gazebo::transport::PublicationTransport**

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

  A publisher of messages on a topic.
- class **gazebo::transport::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 subscriber to a topic.
- class **gazebo::transport::SubscriptionTransport**

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

  Manages topics and their subscriptions.

Files

- file **TransportTypes.hh**

  Forward declarations for transport.

Typedefs

- typedef boost::shared_ptr < CallbackHelper > **gazebo::transport::CallbackHelperPtr**

  *boost shared pointer to **transport::CallbackHelper** (p. 235)*

Functions

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

  Clear any remaining communication buffers.
- **GAZEBO_VISIBLE** transport::ConnectionPtr **gazebo::transport::connectToMaster** ()

  Create a connection to master.
- **GAZEBO_VISIBLE** void **gazebo::transport::fini** ()

  Cleanup the transport component.
- **GAZEBO_VISIBLE** 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.
- **GAZEBO_VISIBLE** void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)

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

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

  Get a list of all the unique advertised topic names.
- **GAZEBO_VISIBLE** bool **gazebo::transport::getMinimalComms** ()

  Get whether minimal comms has been enabled.
- **GAZEBO_VISIBLE** std::string **gazebo::transport::getTopicMsgType** (const std::string &_topicName)

  Get the message typename that is published on the given topic.
- **GAZEBO_VISIBLE** bool **gazebo::transport::init** (const std::string &_masterHost="", unsigned int _masterPort=0, uint32_t _timeoutIterations=30)

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

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

  Pause or unpause incoming messages.

- template<typename M >
  GAZEBO_VISIBLE void **gazebo::transport::publish** (const std::string &_topic, const google::protobuf::Message &_message)

  A convenience function for a one-time publication of a message.
- **GAZEBO_VISIBLE** boost::shared_ptr < msgs::Response > **gazebo::transport::request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

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

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

  Run the transport component.
- **GAZEBO_VISIBLE** void **gazebo::transport::setMinimalComms** (bool _enabled)

  Set whether minimal comms should be used.
- **GAZEBO_VISIBLE** void **gazebo::transport::stop** ()

  Stop the transport component from running.
- **GAZEBO_VISIBLE** bool **gazebo::transport::waitForNamespaces** (const **gazebo::common::Time** &_maxWait)

  *Blocks while waiting for topic namespaces from the **Master** (p. 792).*

8.11.1 Detailed Description

Handles transportation of messages.

Remarks

Environment Variables:

- GAZEBO_IP_WHITE_LIST: Comma separated list of valid IPs. Leave this empty to accept connections from all addresses.
- GAZEBO_IP: IP address to export. This will override the default IP lookup.
- GAZEBO_HOSTNAME: Hostame to export. Setting this will override both GAZEBO_IP and the default IP lookup.

8.11.2 Typedef Documentation

8.11.2.1 typedef boost::shared_ptr<CallbackHelper> gazebo::transport::CallbackHelperPtr

boost shared pointer to **transport::CallbackHelper** (p. 235)

8.11.3 Function Documentation

8.11.3.1 GAZEBO_VISIBLE void gazebo::transport::clear_buffers ( )

Clear any remaining communication buffers.

8.11.3.2 GAZEBO_VISIBLE transport::ConnectionPtr gazebo::transport::connectToMaster ( )

Create a connection to master.

Returns

Connection (p. 328) to the master, NULL on error.

8.11.3.3 GAZEBO_VISIBLE void gazebo::transport::fini ( )

Cleanup the transport component.

8.11.3.4 GAZEBO_VISIBLE 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)

8.11.3.5 GAZEBO_VISIBLE 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

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

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

Returns

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

8.11.3.7 GAZEBO_VISIBLE 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 list of the advertised topics that publish messages of the type specified by _msgType.

8.11.3.8 GAZEBO_VISIBLE bool gazebo::transport::getMinimalComms ( )

Get whether minimal comms has been enabled.

Returns

True if minimal comms is enabled.

8.11.3.9 GAZEBO_VISIBLE 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.

8.11.3.10 GAZEBO_VISIBLE bool gazebo::transport::init (const std::string & *_masterHost* = " ", unsigned int *_masterPort* = 0, uint32_t *_timeoutIterations* = 30)

Initialize the transport system.

Parameters

in	*_masterHost*	The hostname or IP of the master. Leave empty to use pull address from the GAZEBO_MASTER_URI env var.
in	*_masterPort*	The port of the master. Leave empty to use pull address from the GAZEBO_MASTER_URI env var.
in	*_timeout-Iterations*	Number of times to wait for a connection to master.

Returns

true if initialization succeeded; false otherwise

8.11.3.11 GAZEBO_VISIBLE bool gazebo::transport::is_stopped ( )

Is the transport system stopped?

Returns

true if the transport system is stopped; false otherwise

8.11.3.12 GAZEBO_VISIBLE 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

8.11.3.13 template<typename M > GAZEBO_VISIBLE void gazebo::transport::publish (const std::string & *_topic,* const google::protobuf::Message & *_message*)

A convenience function for a one-time publication of a message.

This is inefficient, compared to **Node::Advertise** (p. 912) followed by **Publisher::-Publish** (p. 1026). 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

8.11.3.14 GAZEBO_VISIBLE boost::shared_ptr<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.

8.11.3.15 GAZEBO_VISIBLE 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.

8.11.3.16 GAZEBO_VISIBLE 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.

8.11.3.17 GAZEBO_VISIBLE 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

8.11.3.18 GAZEBO_VISIBLE void gazebo::transport::setMinimalComms (bool _enabled)

Set whether minimal comms should be used.

This will be used to reduce network traffic.

8.11.3.19 GAZEBO_VISIBLE void gazebo::transport::stop ( )

Stop the transport component from running.

8.11.3.20 GAZEBO_VISIBLE bool gazebo::transport::waitForNamespaces (const gazebo::common::Time & *_maxWait*)

Blocks while waiting for topic namespaces from the **Master** (p. 792).

This function will wait a maximum of _maxWait.

Returns

True if namespaces were found before _maxWait time.

8.12 Utility

Classes

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

  A diagnostic manager class.
- class **gazebo::util::DiagnosticTimer**

  A timer designed for diagnostics.

Files

- file **UtilTypes.hh**

Defines

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

8.12.1 Define Documentation

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

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

8.12.1.3 #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. *514) 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 manager that directs topic connections, enables each gazebo network client to locate one another for peer-to-peer communication.
- class **ModelPlugin**

  A plugin with access to ***physics::Model*** *(*p. *846).*
- class **PluginT**

  A class which all plugins must inherit from.
- class **SensorPlugin**

  A plugin with access to physics::Sensor.
- class **Server**
- class **SystemPlugin**

  A plugin loaded within the gzserver on startup.
- class **VisualPlugin**

  A plugin loaded within the gzserver on startup.
- class **WorldPlugin**

  A plugin with access to ***physics::World*** *(*p. *1529).*

Typedefs

- typedef boost::shared_ptr < GUIPlugin > **GUIPluginPtr**
- typedef boost::shared_ptr < **ModelPlugin** > **ModelPluginPtr**
- typedef boost::shared_ptr < **SensorPlugin** > **SensorPluginPtr**
- typedef boost::shared_ptr < **SystemPlugin** > **SystemPluginPtr**
- typedef boost::shared_ptr < **VisualPlugin** > **VisualPluginPtr**
- typedef boost::shared_ptr < **WorldPlugin** > **WorldPluginPtr**

Enumerations

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

  Used to specify the type of plugin.

Functions

- **GAZEBO_VISIBLE** void **addPlugin** (const std::string &_filename)

  Add a system plugin.
- **GAZEBO_VISIBLE gazebo::physics::WorldPtr loadWorld** (const std::string &_worldFile)

Create and load a new world from an SDF world file.

- **GAZEBO_VISIBLE** void **printVersion** ()

  Output version information to the terminal.
- **GAZEBO_VISIBLE** void **runWorld** (**gazebo::physics::WorldPtr** _world, unsigned int _iterations)

  Run a world for a specific number of iterations.
- **GAZEBO_VISIBLE** bool **setupClient** (int _argc=0, char **_argv=0)

  Start a gazebo client.
- **GAZEBO_VISIBLE** bool **setupClient** (const std::vector< std::string > &_args)

  Start a gazebo client.
- **GAZEBO_VISIBLE** bool **setupServer** (int _argc=0, char **_argv=0)

  Start a gazebo server.
- **GAZEBO_VISIBLE** bool **setupServer** (const std::vector< std::string > &_args)

  Start a gazebo server.
- **GAZEBO_VISIBLE** bool **shutdown** ()

  Stop and cleanup simulation.

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 boost::shared_ptr<GUIPlugin> gazebo::GUIPluginPtr

9.2.2.2 typedef boost::shared_ptr<ModelPlugin> gazebo::ModelPluginPtr

9.2.2.3 typedef boost::shared_ptr<SensorPlugin> gazebo::SensorPluginPtr

9.2.2.4 typedef boost::shared_ptr<SystemPlugin> gazebo::SystemPluginPtr

9.2.2.5 typedef boost::shared_ptr<VisualPlugin> gazebo::VisualPluginPtr

9.2.2.6 typedef boost::shared_ptr<WorldPlugin> gazebo::WorldPluginPtr

9.2.3 Function Documentation

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

Add a system plugin.

Parameters

in	*_filename*	Path to the plugin.

9.2.3.2 GAZEBO_VISIBLE gazebo::physics::WorldPtr gazebo::loadWorld (const std::string & *_worldFile*)

Create and load a new world from an SDF world file.

Parameters

in	*_worldFile*	The world file to load from.

Returns

Pointer to the created world. NULL on error.

9.2.3.3 GAZEBO_VISIBLE void gazebo::printVersion ( )

Output version information to the terminal.

9.2.3.4 GAZEBO_VISIBLE void gazebo::runWorld (gazebo::physics::WorldPtr *_world,* unsigned int *_iterations*)

Run a world for a specific number of iterations.

Parameters

in	*_world*	Pointer to a world.
in	*_iterations*	Number of iterations to execute.

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

Start a gazebo client.

This starts transportation, and makes it possible to connect to a running simulation.

Parameters

in	*_argc*	Number of commandline arguments.
in	*_argv*	The commandline arguments.

Returns

True on success.

9.2.3.6 GAZEBO_VISIBLE bool gazebo::setupClient (const std::vector< std::string > & *_args*)

Start a gazebo client.

This starts transportation, and makes it possible to connect to a running simulation.

Parameters

in	*_args*	Vector of arguments only parsed by the system plugins. - Note that when you run gazebo/gzserver, all the options (--version, --server-plugin, etc.) are parsed but when using Gazebo as a library, the arguments are only parsed by the system plugins.

See also

gazebo::SystemPlugin::Load() (p. 1361)

Returns

True on success.

9.2.3.7 GAZEBO_VISIBLE bool gazebo::setupServer (int *_argc* = 0, char ** *_argv* = 0)

Start a gazebo server.

This starts transportation, and makes it possible to create worlds.

Parameters

in	*_argc*	Number of commandline arguments.
in	*_argv*	The commandline arguments.

Returns

True on success.

9.2.3.8 GAZEBO_VISIBLE bool gazebo::setupServer (const std::vector< std::string > & _args)

Start a gazebo server.

This starts transportation, and makes it possible to create worlds.

Parameters

in	*_args*	Vector of arguments only parsed by the system plugins. - Note that when you run gazebo/gzserver, all the options (--version, --server-plugin, etc.) are parsed but when using Gazebo as a library, the arguments are only parsed by the system plugins.

See also

gazebo::SystemPlugin::Load() (p. 1361)

Returns

True on success.

9.2.3.9 GAZEBO_VISIBLE bool gazebo::shutdown ( )

Stop and cleanup simulation.

Returns

True if the simulation is shutdown; false otherwise.

Referenced by gazebo::transport::Connection::ConnectToShutdown(), and gazebo::transport::Connection::DisconnectShutdown().

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 **AudioDecoder**

  An audio decoder based on FFMPEG.
- class **BVHLoader**

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

  Class used to export Collada mesh files.
- class **ColladaExporterPrivate**

  *Private data for the **ColladaExporter** (p. 287) class.*
- class **ColladaLoader**

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

  *Private data for the **ColladaLoader** (p. 291) class.*
- class **Color**

  Defines a color.
- class **Console**

  Container for loggers, and global logging options (such as verbose vs.
- class **Exception**

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

  A logger that outputs messages to a file.
- class **GeometryIndices**

  Helper data structure for loading collada geometries.
- class **HeightmapData**

  Encapsulates a generic heightmap data file.
- class **Image**

  Encapsulates an image.
- class **ImageHeightmap**

  Encapsulates an image that will be interpreted as a heightmap.
- class **InternalError**

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

  Generic description of a keyboard event.
- class **KeyFrame**

  A key frame in an animation.
- class **Logger**

Terminal logger.

- class **Material**

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

  A 3D mesh.
- class **MeshCSG**

  Creates CSG meshes.
- class **MeshExporter**

  Base class for exporting 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 **ModelDatabasePrivate**

  *Private class attributes for **ModelDatabase** (p. 864).*
- class **MouseEvent**

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

  *Base class for **MovingWindowFilter** (p. 890).*
- class **MovingWindowFilterPrivate**
- class **NodeAnimation**

  Node animation.
- class **NodeAssignment**

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

  ***NodeTransform** (p. 925) **Skeleton.hh** (p. 1787) common/common.hh*
- class **NumericAnimation**

  A numeric animation.
- class **NumericKeyFrame**

  *A keyframe for a **NumericAnimation** (p. 936).*
- class **PID**

  *Generic **PID** (p. 977) controller class.*
- class **PoseAnimation**

  A pose animation.
- class **PoseKeyFrame**

  *A keyframe for a **PoseAnimation** (p. 1006).*
- class **Skeleton**

  A skeleton.

- class **SkeletonAnimation**

  Skeleton *(p. 1269) animation.*
- class **SkeletonNode**

  A skeleton node.
- class **SphericalCoordinates**

  Convert spherical coordinates for planetary surfaces.
- class **SphericalCoordinatesPrivate**

  commmon/common.hh
- class **STLLoader**

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

  A child mesh.
- class **SystemPaths**

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

  *A **Time** (p. 1361) class, can be used to hold wall- or sim-time.*
- class **Timer**

  A 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 boost::shared_ptr < **Animation** > **AnimationPtr**
- typedef boost::shared_ptr < DiagnosticTimer > **DiagnosticTimerPtr**
- typedef std::map< unsigned int, **SkeletonNode** * > **NodeMap**
- typedef std::map< unsigned int, **SkeletonNode** * >::iterator **NodeMapIter**
- typedef boost::shared_ptr < **NumericAnimation** > **NumericAnimationPtr**
- typedef std::vector < common::Param * > **Param_V**
- typedef boost::shared_ptr < **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 boost::shared_ptr < **SphericalCoordinates** > **SphericalCoordinatesPtr**
- typedef std::map< std::string, std::string > **StrStr_M**

Functions

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

  *add path sufix to **common::SystemPaths*** (p. *1352*)
- **GAZEBO_VISIBLE** std::string **find_file** (const std::string &_file)

  *search for file in **common::SystemPaths*** (p. *1352*)
- **GAZEBO_VISIBLE** std::string **find_file** (const std::string &_file, bool _search-LocalPath)

  *search for file in **common::SystemPaths*** (p. *1352*)
- **GAZEBO_VISIBLE** std::string **find_file_path** (const std::string &_file)

  *search for a file in **common::SystemPaths*** (p. *1352*)
- template<typename T >
  GAZEBO_VISIBLE std::string **get_sha1** (const T &_buffer)

  Compute the SHA1 hash of an array of bytes.
- **GAZEBO_VISIBLE** void **load** ()

  Load the common library.

Variables

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

  String names for the pixel formats.
- static const double **SpeedOfLight** = 299792458

  Speed of light.

9.3.1 Detailed Description

Common namespace.

9.3.2 Typedef Documentation

9.3.2.1 typedef boost::shared_ptr<Animation> gazebo::common::AnimationPtr

9.3.2.2 typedef boost::shared_ptr<DiagnosticTimer> gazebo::common::Diagnostic-TimerPtr

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 boost::shared_ptr<NumericAnimation> gazebo::common::NumericAnimationPtr

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

9.3.2.7 typedef boost::shared_ptr<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 boost::shared_ptr<SphericalCoordinates> gazebo::common::SphericalCoordinatesPtr

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

9.3.3 Variable Documentation

9.3.3.1 const double gazebo::common::SpeedOfLight = 299792458 [static]

Speed of light.

9.4 gazebo::event Namespace Reference

Event (p. 514) namespace.

Classes

- class **Connection**

  A class that encapsulates a connection.
- class **ConnectionPrivate**
- class **Event**

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

*An **Event** (p. 514) class to get notifications for simulator events.*

- class **EventT**

  A class for event processing.

- class **EventTPrivate**

Typedefs

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

9.4.1 Detailed Description

Event (p. 514) namespace.

9.4.2 Typedef Documentation

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

9.4.2.2 typedef boost::shared_ptr<Connection> gazebo::event::ConnectionPtr

9.5 gazebo::math Namespace Reference

Math namespace.

Classes

- class **Angle**

  An angle and related functions.

- class **BiQuad**

  Bi-quad filter base class.

- class **BiQuadVector3**

  ***BiQuad** (p. 217) vector3 filter.*

- class **Box**

  Mathematical representation of a box and related functions.

- class **Filter**

  ***Filter** (p. 553) base class.*

- class **Matrix3**

  A 3x3 matrix class.

- class **Matrix4**

A 3x3 matrix class.

- class **OnePole**

  A one-pole DSP filter.
- class **OnePoleQuaternion**

  One-pole quaternion filter.
- class **OnePoleVector3**

  One-pole vector3 filter.
- class **Plane**

  A plane and related functions.
- class **Pose**

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

  A quaternion class.
- class **Rand**

  Random number generator class.
- class **RotationSpline**

  Spline (p. *1317) for rotations.*
- class **Spline**

  Splines.
- class **Vector2d**

  Generic double x, y vector.
- class **Vector2i**

  Generic integer x, y vector.
- class **Vector3**

  *The **Vector3** (p. 1440) 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
- float **fixnan** (float _v)

  Fix a nan value.
- double **fixnan** (double _v)

  Fix a nan value.
- 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
- unsigned int **roundUpPowerOfTwo** (unsigned int _x)

  Get the smallest power of two that is greater or equal to a given value.
- 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 factory that generates protobuf message based on a string type.

Typedefs

- typedef boost::shared_ptr < google::protobuf::Message >(∗ **MsgFactoryFn**)()

Functions

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

  *Convert a **math::Vector3*** (p. *1440) to a msgs::Vector3d.*
- **GAZEBO_VISIBLE** msgs::Quaternion **Convert** (const **math::Quaternion** &_q)

  *Convert a **math::Quaternion*** (p. *1029) to a msgs::Quaternion.*
- **GAZEBO_VISIBLE** msgs::Pose **Convert** (const **math::Pose** &_p)

  *Convert a **math::Pose*** (p. *995) to a msgs::Pose.*
- **GAZEBO_VISIBLE** msgs::Color **Convert** (const **common::Color** &_c)

  *Convert a **common::Color*** (p. *312) to a msgs::Color.*
- **GAZEBO_VISIBLE** msgs::Time **Convert** (const **common::Time** &_t)

  *Convert a **common::Time*** (p. *1361) to a msgs::Time.*
- **GAZEBO_VISIBLE** msgs::PlaneGeom **Convert** (const **math::Plane** &_p)

  *Convert a **math::Plane*** (p. *984) to a msgs::PlaneGeom.*
- **GAZEBO_VISIBLE math::Vector3 Convert** (const msgs::Vector3d &_v)

  Convert a msgs::Vector3d to a math::Vector.
- **GAZEBO_VISIBLE math::Quaternion Convert** (const msgs::Quaternion &_q)

  *Convert a msgs::Quaternion to a **math::Quaternion*** (p. *1029).*
- **GAZEBO_VISIBLE math::Pose Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose*** (p. *995).*
- **GAZEBO_VISIBLE common::Color Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color*** (p. *312).*
- **GAZEBO_VISIBLE common::Time Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time*** (p. *1361).*
- **GAZEBO_VISIBLE math::Plane Convert** (const msgs::PlaneGeom &_p)

  Convert a msgs::PlaneGeom to a common::Plane.
- **GAZEBO_VISIBLE** msgs::Request ∗ **CreateRequest** (const std::string &_request, const std::string &_data="")

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

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

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

Get the header from a protobuf message.

- **GAZEBO_VISIBLE** msgs::GUI **GUIFromSDF** (sdf::ElementPtr _sdf)

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

- **GAZEBO_VISIBLE** void **Init** (google::protobuf::Message &_message, const std::string &_id="")

  Initialize a message.

- **GAZEBO_VISIBLE** msgs::Light **LightFromSDF** (sdf::ElementPtr _sdf)

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

- **GAZEBO_VISIBLE** sdf::ElementPtr **LightToSDF** (const msgs::Light &_msg, sdf::ElementPtr _sdf=sdf::ElementPtr())

  Create an SDF element from a msgs::Scene.

- **GAZEBO_VISIBLE** msgs::MeshGeom **MeshFromSDF** (sdf::ElementPtr _sdf)

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

- **GAZEBO_VISIBLE** msgs::Scene **SceneFromSDF** (sdf::ElementPtr _sdf)

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

- **GAZEBO_VISIBLE** void **Set** (**common::Image** &_img, const msgs::Image &_msg)

  *Convert a msgs::Image to a **common::Image** (p. 637).*

- **GAZEBO_VISIBLE** void **Set** (msgs::Image *_msg, const **common::Image** &_i)

  *Set a msgs::Image from a **common::Image** (p. 637).*

- **GAZEBO_VISIBLE** void **Set** (msgs::Vector3d *_pt, const **math::Vector3** &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 1440).*

- **GAZEBO_VISIBLE** void **Set** (msgs::Vector2d *_pt, const **math::Vector2d** &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 1440).*

- **GAZEBO_VISIBLE** void **Set** (msgs::Quaternion *_q, const **math::Quaternion** &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 1029).*

- **GAZEBO_VISIBLE** void **Set** (msgs::Pose *_p, const **math::Pose** &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 995).*

- **GAZEBO_VISIBLE** void **Set** (msgs::Color *_c, const **common::Color** &_v)

  *Set a msgs::Color from a **common::Color** (p. 312).*

- **GAZEBO_VISIBLE** void **Set** (msgs::Time *_t, const **common::Time** &_v)

  *Set a msgs::Time from a **common::Time** (p. 1361).*

- void **Set** (msgs::SphericalCoordinates *_s, const **common::SphericalCoordinates** &_v)

  *Set a msgs::SphericalCoordinates from a **common::SphericalCoordinates** (p. 1310) object.*

- **GAZEBO_VISIBLE** void **Set** (msgs::PlaneGeom *_p, const **math::Plane** &_v)

  *Set a msgs::Plane from a **math::Plane** (p. 984).*

- **GAZEBO_VISIBLE** void **Stamp** (msgs::Header *_header)

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

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

  Create a msgs::TrackVisual from a track visual SDF element.
- **GAZEBO_VISIBLE** 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 boost::shared_ptr<google::protobuf::Message>(* gazebo::msgs::MsgFactoryFn)()

9.7 gazebo::physics Namespace Reference

namespace for physics

Classes

- class **Actor**

  Actor (p. *165) class enables GPU based mesh model / skeleton scriptable animation.*
- class **BallJoint**

  Base (p. *201) class for a ball joint.*
- class **Base**

  Base (p. *201) class for most physics classes.*
- class **BoxShape**

  Box geometry primitive.
- class **Collision**

  Base (p. *201) class for all collision entities.*
- class **CollisionState**

  Store state information of a ***physics::Collision*** (p. *295) object.*
- class **Contact**

  A contact between two collisions.

- class **ContactManager**

  Aggregates all the contact information generated by the collision detection engine.
- class **ContactPublisher**

  *A custom contact publisher created for each contact filter in the **Contact** (p. 347) - Manager.*
- class **CylinderShape**

  Cylinder collision.
- class **DARTBallJoint**

  *An **DARTBallJoint** (p. 374).*
- class **DARTBoxShape**

  DART Box shape.
- class **DARTCollision**

  ***Base** (p. 201) class for all DART collisions.*
- class **DARTCylinderShape**

  DART cylinder shape.
- class **DARTHeightmapShape**

  DART Height map collision.
- class **DARTHinge2Joint**

  A two axis hinge joint.
- class **DARTHingeJoint**

  A single axis hinge joint.
- class **DARTJoint**

  DART joint interface.
- class **DARTLink**

  *DART **Link** (p. 739) class.*
- class **DARTMeshShape**

  Triangle mesh collision.
- class **DARTModel**

  DART model class.
- class **DARTMultiRayShape**

  *DART specific version of **MultiRayShape** (p. 901).*
- class **DARTPhysics**

  DART physics engine.
- class **DARTPlaneShape**

  An DART Plane shape.
- class **DARTRayShape**

  Ray collision.
- class **DARTScrewJoint**

  A screw joint.
- class **DARTSliderJoint**

A slider joint.

- class **DARTSphereShape**

  A DART sphere shape.

- class **DARTTypes**

  A set of functions for converting between the math types used by gazebo and dart.

- class **DARTUniversalJoint**

  A universal joint.

- class **Entity**

  Base (p. *201) class for all physics objects in Gazebo.*

- class **FrictionPyramid**

  Parameters used for friction pyramid model.

- class **GearboxJoint**

  A double axis gearbox joint.

- class **Gripper**

  A gripper abstraction.

- class **HeightmapShape**

  HeightmapShape (p. *628) collision shape builds a heightmap from an image.*

- class **Hinge2Joint**

  A two axis hinge joint.

- class **HingeJoint**

  A single axis hinge joint.

- class **Inertial**

  A class for inertial information about a link.

- class **Joint**

  Base (p. *201) class for all joints.*

- class **JointController**

  *A class for manipulating **physics::Joint*** (p. *669).*

- class **JointControllerPrivate**

- class **JointState**

  *keeps track of state of a **physics::Joint*** (p. *669)*

- class **JointWrench**

  Wrench information from a joint.

- class **Link**

  Link (p. *739) 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. *739) object.*

- class **MeshShape**

  Triangle mesh collision shape.

- class **Model**

A model is a collection of links, joints, and plugins.

- class **ModelState**

  *Store state information of a **physics::Model** (p. 846) object.*

- class **MultiRayShape**

  Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

- class **PhysicsEngine**

  ***Base** (p. 201) class for a physics engine.*

- class **PhysicsFactory**

  The physics factory instantiates different physics engines.

- class **PlaneShape**

  ***Collision** (p. 295) for an infinite plane.*

- class **RayShape**

  ***Base** (p. 201) class for Ray collision geometry.*

- class **Road**

  *for building a **Road** (p. 1083) from SDF*

- class **ScrewJoint**

  A screw joint, which has both prismatic and rotational DOFs.

- class **Shape**

  ***Base** (p. 201) class for all shapes.*

- class **SimbodyBallJoint**

  ***SimbodyBallJoint** (p. 1165) class models a ball joint in Simbody.*

- class **SimbodyBoxShape**

  Simbody box collision.

- class **SimbodyCollision**

  Simbody collisions.

- class **SimbodyCylinderShape**

  Cylinder collision.

- class **SimbodyHeightmapShape**

  Height map collision.

- class **SimbodyHinge2Joint**

  A two axis hinge joint.

- class **SimbodyHingeJoint**

  A single axis hinge joint.

- class **SimbodyJoint**

  ***Base** (p. 201) class for all joints.*

- class **SimbodyLink**

  *Simbody **Link** (p. 739) class.*

- class **SimbodyMeshShape**

  Triangle mesh collision.

- class **SimbodyModel**

  A model is a collection of links, joints, and plugins.
- class **SimbodyMultiRayShape**

  Simbody specific version of ***MultiRayShape*** *(*p. *901).*
- class **SimbodyPhysics**

  Simbody physics engine.
- class **SimbodyPlaneShape**

  Simbody collision for an infinite plane.
- class **SimbodyRayShape**

  Ray shape for simbody.
- class **SimbodyScrewJoint**

  A screw joint.
- class **SimbodySliderJoint**

  A slider joint.
- class **SimbodySphereShape**

  Simbody sphere collision.
- class **SimbodyUniversalJoint**

  A simbody universal joint class.
- class **SliderJoint**

  A slider joint.
- class **SphereShape**

  Sphere collision shape.
- class **State**

  State *(*p. *1323) of an entity.*
- class **SurfaceParams**

  SurfaceParams *(*p. *1350) defines various Surface contact parameters.*
- class **TrajectoryInfo**

  Information about a trajectory for an ***Actor*** *(*p. *165).*
- class **UniversalJoint**

  A universal joint.
- class **World**

  The world provides access to all other object within a simulated environment.
- class **WorldState**

  Store state information of a ***physics::World*** *(*p. *1529) object.*

Typedefs

- typedef std::vector< **ActorPtr** > **Actor_V**
- typedef boost::shared_ptr< **Actor** > **ActorPtr**
- typedef std::vector< **BasePtr** > **Base_V**
- typedef boost::shared_ptr< **Base** > **BasePtr**
- typedef boost::shared_ptr < **BoxShape** > **BoxShapePtr**
- typedef std::vector< **CollisionPtr** > **Collision_V**
- typedef boost::shared_ptr < **Collision** > **CollisionPtr**
- typedef boost::shared_ptr < **Contact** > **ContactPtr**
- typedef boost::shared_ptr < **CylinderShape** > **CylinderShapePtr**
- typedef boost::shared_ptr < **DARTCollision** > **DARTCollisionPtr**
- typedef boost::shared_ptr < **DARTJoint** > **DARTJointPtr**
- typedef boost::shared_ptr < **DARTLink** > **DARTLinkPtr**
- typedef boost::shared_ptr < **DARTModel** > **DARTModelPtr**
- typedef boost::shared_ptr < **DARTPhysics** > **DARTPhysicsPtr**
- typedef boost::shared_ptr < **DARTRayShape** > **DARTRayShapePtr**
- typedef boost::shared_ptr< **Entity** > **EntityPtr**
- typedef boost::shared_ptr < **Gripper** > **GripperPtr**
- typedef boost::shared_ptr < **HeightmapShape** > **HeightmapShapePtr**
- typedef boost::shared_ptr < **Inertial** > **InertialPtr**
- typedef std::vector< **JointPtr** > **Joint_V**
- typedef std::vector < **JointControllerPtr** > **JointController_V**
- typedef boost::shared_ptr < **JointController** > **JointControllerPtr**
- typedef boost::shared_ptr< **Joint** > **JointPtr**
- typedef std::map< std::string, **JointState** > **JointState_M**
- typedef std::vector< **LinkPtr** > **Link_V**
- typedef boost::shared_ptr< **Link** > **LinkPtr**
- typedef std::map< std::string, **LinkState** > **LinkState_M**
- typedef boost::shared_ptr < **MeshShape** > **MeshShapePtr**
- typedef std::vector< **ModelPtr** > **Model_V**
- typedef boost::shared_ptr< **Model** > **ModelPtr**
- typedef std::map< std::string, **ModelState** > **ModelState_M**
- typedef boost::shared_ptr < **MultiRayShape** > **MultiRayShapePtr**
- typedef boost::shared_ptr < **PhysicsEngine** > **PhysicsEnginePtr**
- typedef **PhysicsEnginePtr**(∗ **PhysicsFactoryFn**)(**WorldPtr** world)
- typedef boost::shared_ptr < **RayShape** > **RayShapePtr**
- typedef boost::shared_ptr< **Road** > **RoadPtr**
- typedef boost::shared_ptr< **Shape** > **ShapePtr**
- typedef boost::shared_ptr < **SimbodyCollision** > **SimbodyCollisionPtr**
- typedef boost::shared_ptr < **SimbodyLink** > **SimbodyLinkPtr**
- typedef boost::shared_ptr < **SimbodyModel** > **SimbodyModelPtr**
- typedef boost::shared_ptr < **SimbodyPhysics** > **SimbodyPhysicsPtr**

- typedef boost::shared_ptr < **SimbodyRayShape** > **SimbodyRayShapePtr**
- typedef boost::shared_ptr < **SphereShape** > **SphereShapePtr**
- typedef boost::shared_ptr < **SurfaceParams** > **SurfaceParamsPtr**
- typedef boost::shared_ptr< **World** > **WorldPtr**

Functions

- **GAZEBO_VISIBLE WorldPtr create_world** (const std::string &_name="")

  Create a world given a name.
- **GAZEBO_VISIBLE** bool **fini** ()

  *Finalize transport by calling **gazebo::transport::fini** (p. 113).*
- **GAZEBO_VISIBLE WorldPtr get_world** (const std::string &_name="")

  Returns a pointer to a world by name.
- **GAZEBO_VISIBLE** uint32_t **getUniqueId** ()

  Get a unique ID.
- **GAZEBO_VISIBLE** void **init_world** (**WorldPtr** _world)

  Init world given a pointer to it.
- **GAZEBO_VISIBLE** void **init_worlds** ()

  initialize multiple worlds stored in static variable gazebo::g_worlds
- **GAZEBO_VISIBLE** bool **load** ()

  *Setup **gazebo::SystemPlugin** (p. 1359)'s and call **gazebo::transport::init** (p. 115).*
- **GAZEBO_VISIBLE** void **load_world** (**WorldPtr** _world, sdf::ElementPtr _sdf)

  Load world from sdf::Element pointer.
- **GAZEBO_VISIBLE** void **load_worlds** (sdf::ElementPtr _sdf)

  load multiple worlds from single sdf::Element pointer
- **GAZEBO_VISIBLE** void **pause_world** (**WorldPtr** _world, bool _pause)

  *Pause world by calling **World::SetPaused** (p. 1542).*
- **GAZEBO_VISIBLE** void **pause_worlds** (bool pause)

  pause multiple worlds stored in static variable gazebo::g_worlds
- **GAZEBO_VISIBLE** void **remove_worlds** ()

  remove multiple worlds stored in static variable gazebo::g_worlds
- **GAZEBO_VISIBLE** void **run_world** (**WorldPtr** _world, unsigned int _iterations=0)

  *Run world by calling **World::Run()** (p. 1541) given a pointer to it.*
- **GAZEBO_VISIBLE** void **run_worlds** (unsigned int _iterations=0)

  Run multiple worlds stored in static variable gazebo::g_worlds.
- **GAZEBO_VISIBLE** void **stop_world** (**WorldPtr** _world)

  *Stop world by calling **World::Stop()** (p. 1543) given a pointer to it.*
- **GAZEBO_VISIBLE** void **stop_worlds** ()

  stop multiple worlds stored in static variable gazebo::g_worlds
- **GAZEBO_VISIBLE** bool **worlds_running** ()

  Return true if any world is running.

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 boost::shared_ptr<Actor> gazebo::physics::ActorPtr

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

9.7.2.4 typedef boost::shared_ptr<Base> gazebo::physics::BasePtr

9.7.2.5 typedef boost::shared_ptr<BoxShape> gazebo::physics::BoxShapePtr

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

9.7.2.7 typedef boost::shared_ptr<Collision> gazebo::physics::CollisionPtr

9.7.2.8 typedef boost::shared_ptr<Contact> gazebo::physics::ContactPtr

9.7.2.9 typedef boost::shared_ptr<CylinderShape> gazebo::physics::CylinderShape-Ptr

9.7.2.10 typedef boost::shared_ptr<DARTCollision> gazebo::physics::DART-CollisionPtr

9.7.2.11 typedef boost::shared_ptr<DARTJoint> gazebo::physics::DARTJointPtr

9.7.2.12 typedef boost::shared_ptr<DARTLink> gazebo::physics::DARTLinkPtr

9.7.2.13 typedef boost::shared_ptr<DARTModel> gazebo::physics::DARTModelPtr

9.7.2.14 typedef boost::shared_ptr<DARTPhysics> gazebo::physics::DARTPhysics-Ptr

9.7.2.15 typedef boost::shared_ptr<DARTRayShape> gazebo::physics::DARTRayShapePtr

9.7.2.16 typedef boost::shared_ptr<Entity> gazebo::physics::EntityPtr

9.7.2.17 typedef boost::shared_ptr<Gripper> gazebo::physics::GripperPtr

9.7.2.18 typedef boost::shared_ptr<HeightmapShape> gazebo::physics::HeightmapShapePtr

9.7.2.19 typedef boost::shared_ptr<Inertial> gazebo::physics::InertialPtr

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

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

9.7.2.22 typedef boost::shared_ptr<JointController> gazebo::physics::JointControllerPtr

9.7.2.23 typedef boost::shared_ptr<Joint> gazebo::physics::JointPtr

9.7.2.24 typedef std::map<std::string, JointState> gazebo::physics::JointState_M

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

9.7.2.26 typedef boost::shared_ptr<Link> gazebo::physics::LinkPtr

9.7.2.27 typedef std::map<std::string, LinkState> gazebo::physics::LinkState_M

9.7.2.28 typedef boost::shared_ptr<MeshShape> gazebo::physics::MeshShapePtr

9.7.2.29 typedef std::vector<ModelPtr> gazebo::physics::Model_V

9.7.2.30 typedef boost::shared_ptr<Model> gazebo::physics::ModelPtr

9.7.2.31 typedef std::map<std::string, ModelState> gazebo::physics::ModelState_M

9.7.2.32 typedef boost::shared_ptr<MultiRayShape> gazebo::physics::MultiRayShapePtr

9.7.2.33 typedef boost::shared_ptr<PhysicsEngine> gazebo::physics::PhysicsEnginePtr

9.7.2.34 typedef boost::shared_ptr<RayShape> gazebo::physics::RayShapePtr

9.7.2.35 typedef boost::shared_ptr<Road> gazebo::physics::RoadPtr

9.7.2.36 typedef boost::shared_ptr<Shape> gazebo::physics::ShapePtr

9.7.2.37 typedef boost::shared_ptr<SimbodyCollision> gazebo::physics::SimbodyCollisionPtr

9.7.2.38 typedef boost::shared_ptr<SimbodyLink> gazebo::physics::SimbodyLinkPtr

9.7.2.39 typedef boost::shared_ptr<SimbodyModel> gazebo::physics::SimbodyModelPtr

9.7.2.40 typedef boost::shared_ptr<SimbodyPhysics> gazebo::physics::SimbodyPhysicsPtr

9.7.2.41 typedef boost::shared_ptr<SimbodyRayShape> gazebo::physics::SimbodyRayShapePtr

9.7.2.42 typedef boost::shared_ptr<SphereShape> gazebo::physics::SphereShapePtr

9.7.2.43 typedef boost::shared_ptr<SurfaceParams> gazebo::physics::SurfaceParamsPtr

9.7.2.44 typedef boost::shared_ptr<World> gazebo::physics::WorldPtr

9.8 gazebo::rendering Namespace Reference

Rendering namespace.

Classes

- class **ArrowVisual**

  Basic arrow visualization.
- class **ArrowVisualPrivate**

  *Private data for the Arrow **Visual*** (p. *1477*) *class.*
- class **AxisVisual**

  Basic axis visualization.
- class **AxisVisualPrivate**

  *Private data for the Axis **Visual*** (p. *1477*) *class.*

- class **Camera**

  Basic camera sensor.
- class **CameraPrivate**

  *Private data for the **Camera** (p. 242) class.*
- class **CameraVisual**

  Basic camera visualization.
- class **CameraVisualPrivate**
- class **COMVisual**

  Basic Center of Mass visualization.
- class **COMVisualPrivate**

  *Private data for the COM **Visual** (p. 1477) class.*
- class **ContactVisual**

  Contact visualization.
- class **ContactVisualPrivate**

  *Private data for the Arrow **Visual** (p. 1477) class.*
- class **Conversions**

  ***Conversions** (p. 367) **Conversions.hh** (p. 1608) **rendering/Conversions.hh** (p. 1608).*
- class **DepthCamera**

  Depth camera used to render depth data into an image buffer.
- class **DummyPageProvider**

  Pretends to provide procedural page content to avoid page loading.
- class **DynamicLines**

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

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

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

  First Person Shooter style view controller.
- class **GpuLaser**

  GPU based laser distance sensor.
- class **Grid**

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

  A class that creates a CEGUI overlay on a render window.
- class **GUIOverlayPrivate**

  *Private data for the **GUIOverlay** (p. 612) class.*
- class **GzTerrainMatGen**
- class **Heightmap**

Rendering a terrain using heightmap information.

- class **JointVisual**

  Visualization for joints.
- class **JointVisualPrivate**

  *Private data for the Joint **Visual*** (p. *1477) class.*
- class **LaserVisual**

  Visualization for laser data.
- class **LaserVisualPrivate**

  *Private data for the Laser **Visual*** (p. *1477) class.*
- class **Light**

  A light source.
- class **MovableText**

  Movable text.
- class **OculusCamera**

  A camera used for user visualization of a scene.
- 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 **RFIDTagVisualPrivate**

  *Private data for the RFID Tag **Visual*** (p. *1477) class.*
- class **RFIDVisual**

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

  *Private data for the RFID **Visual*** (p. *1477) class.*
- class **Road2d**
- class **RTShaderSystem**

  *Implements **Ogre*** (p. *163)'s Run-Time Shader system.*
- class **Scene**

  Representation of an entire scene graph.
- class **SelectionObj**

  Interactive selection object for models and links.
- class **SelectionObjPrivate**

  Private data for the Selection Obj class.
- class **SonarVisual**

  Visualization for sonar data.

- class **SonarVisualPrivate**

  *Private data for the Sonar **Visual** (p. 1477) class.*
- class **TransmitterVisual**

  Visualization for the wireless propagation data.
- class **TransmitterVisualPrivate**

  *Private data for the Transmitter **Visual** (p. 1477) class.*
- class **UserCamera**

  A camera used for user visualization of a scene.
- class **UserCameraPrivate**

  *Private data for the **UserCamera** (p. 1407) class.*
- class **VideoVisual**

  A visual element that displays a video as a texture.
- class **VideoVisualPrivate**

  *Private data for the Video **Visual** (p. 1477) class.*
- class **ViewController**

  Base class for view controllers.
- class **Visual**

  A renderable object.
- class **VisualPrivate**

  *Private data for the **Visual** (p. 1477) class.*
- class **WindowManager**

  Class to mangage render windows.
- class **WireBox**

  Draws a wireframe box.
- class **WireBoxPrivate**

  *Private data for the **WireBox** (p. 1515) class.*
- class **WrenchVisual**

  Visualization for sonar data.
- class **WrenchVisualPrivate**

  *Private data for the Wrench **Visual** (p. 1477) class.*

Typedefs

- typedef boost::shared_ptr < **ArrowVisual** > **ArrowVisualPtr**
- typedef boost::shared_ptr < **AxisVisual** > **AxisVisualPtr**
- typedef boost::shared_ptr< **Camera** > **CameraPtr**
- typedef boost::shared_ptr < **CameraVisual** > **CameraVisualPtr**
- typedef boost::shared_ptr < **COMVisual** > **COMVisualPtr**
- typedef boost::shared_ptr < **ContactVisual** > **ContactVisualPtr**
- typedef boost::shared_ptr < **DepthCamera** > **DepthCameraPtr**

- typedef boost::shared_ptr < **DynamicLines** > **DynamicLinesPtr**
- typedef boost::shared_ptr < **GpuLaser** > **GpuLaserPtr**
- typedef boost::shared_ptr < **JointVisual** > **JointVisualPtr**
- typedef boost::shared_ptr < **LaserVisual** > **LaserVisualPtr**
- typedef boost::shared_ptr< **Light** > **LightPtr**
- typedef boost::shared_ptr < **RFIDTagVisual** > **RFIDTagVisualPtr**
- typedef boost::shared_ptr < **RFIDVisual** > **RFIDVisualPtr**
- typedef boost::shared_ptr< **Scene** > **ScenePtr**
- typedef boost::shared_ptr < **SelectionObj** > **SelectionObjPtr**
- typedef boost::shared_ptr < **SonarVisual** > **SonarVisualPtr**
- typedef boost::shared_ptr < **UserCamera** > **UserCameraPtr**
- typedef boost::shared_ptr< **Visual** > **VisualPtr**
- typedef boost::shared_ptr < **WindowManager** > **WindowManagerPtr**
- typedef boost::shared_ptr < **WrenchVisual** > **WrenchVisualPtr**

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

- **GAZEBO_VISIBLE rendering::ScenePtr create_scene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)

  *create **rendering::Scene*** (p. *1097*) *by name.*
- **GAZEBO_VISIBLE** bool **fini** ()

  teardown rendering engine.
- **GAZEBO_VISIBLE rendering::ScenePtr get_scene** (const std::string &_name="")

  *get pointer to **rendering::Scene*** (p. *1097*) *by name.*
- **GAZEBO_VISIBLE** bool **init** ()

  init rendering engine.
- **GAZEBO_VISIBLE** bool **load** ()

  load rendering engine.
- **GAZEBO_VISIBLE** void **remove_scene** (const std::string &_name)

  *remove a **rendering::Scene*** (p. *1097*) *by name*

9.8.1 Detailed Description

Rendering namespace.

9.8.2 Typedef Documentation

9.8.2.1 typedef boost::shared_ptr<ArrowVisual> gazebo::rendering::ArrowVisualPtr

9.8.2.2 typedef boost::shared_ptr<AxisVisual> gazebo::rendering::AxisVisualPtr

9.8.2.3 typedef boost::shared_ptr<Camera> gazebo::rendering::CameraPtr

9.8.2.4 typedef boost::shared_ptr<CameraVisual> gazebo::rendering::CameraVisualPtr

9.8.2.5 typedef boost::shared_ptr<COMVisual> gazebo::rendering::COMVisualPtr

9.8.2.6 typedef boost::shared_ptr<ContactVisual> gazebo::rendering::ContactVisualPtr

9.8.2.7 typedef boost::shared_ptr<DepthCamera> gazebo::rendering::DepthCameraPtr

9.8.2.8 typedef boost::shared_ptr<DynamicLines> gazebo::rendering::DynamicLinesPtr

9.8.2.9 typedef boost::shared_ptr<GpuLaser> gazebo::rendering::GpuLaserPtr

9.8.2.10 typedef boost::shared_ptr<JointVisual> gazebo::rendering::JointVisualPtr

9.8.2.11 typedef boost::shared_ptr<LaserVisual> gazebo::rendering::LaserVisualPtr

9.8.2.12 typedef boost::shared_ptr<Light> gazebo::rendering::LightPtr

9.8.2.13 typedef boost::shared_ptr<RFIDTagVisual> gazebo::rendering::RFIDTagVisualPtr

9.8.2.14 typedef boost::shared_ptr<RFIDVisual> gazebo::rendering::RFIDVisualPtr

9.8.2.15 typedef boost::shared_ptr<Scene> gazebo::rendering::ScenePtr

9.8.2.16 typedef boost::shared_ptr<SelectionObj> gazebo::rendering::SelectionObjPtr

9.8.2.17 typedef boost::shared_ptr<SonarVisual> gazebo::rendering::SonarVisualPtr

9.8.2.18 typedef boost::shared_ptr<UserCamera> gazebo::rendering::UserCameraPtr

9.8.2.19 typedef boost::shared_ptr<Visual> gazebo::rendering::VisualPtr

9.8.2.20 typedef boost::shared_ptr<WindowManager> gazebo::rendering::WindowManagerPtr

9.8.2.21 typedef boost::shared_ptr<WrenchVisual> gazebo::rendering::WrenchVisualPtr

9.8.3 Enumeration Type Documentation

9.8.3.1 enum gazebo::rendering::RenderOpType

Type of render operation for a drawable.

Enumerator:

RENDERING_POINT_LIST A list of points, 1 vertex per point.

RENDERING_LINE_LIST A list of lines, 2 vertices per line.

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

RENDERING_TRIANGLE_LIST A list of triangles, 3 vertices per triangle.

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

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

RENDERING_MESH_RESOURCE N/A.

9.9 gazebo::sensors Namespace Reference

Sensors namespace.

Classes

- class **CameraSensor**

  Basic camera sensor.
- class **ContactSensor**

Contact sensor.

- class **DepthCameraSensor**
- class **ForceTorqueSensor**

  Sensor *(*p. *1130) for measure force and torque on a joint.*
- class **GaussianNoiseModel**

  Gaussian noise class.
- class **GpsSensor**

  GpsSensor *(*p. *575) to provide position measurement.*
- class **GpuRaySensor**
- class **ImageGaussianNoiseModel**
- class **ImuSensor**

  An IMU sensor.
- class **MultiCameraSensor**

  Multiple camera sensor.
- class **Noise**

  Noise *(*p. *931) models for sensor output signals.*
- class **NoiseFactory**

  Use this noise manager for creating and loading noise models.
- class **RaySensor**

  Sensor *(*p. *1130) with one or more rays.*
- class **RFIDSensor**

  Sensor *(*p. *1130) class for RFID type of sensor.*
- class **RFIDTag**

  RFIDTag *(*p. *1075) 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.
- class **SonarSensor**

  Sensor *(*p. *1130) with sonar cone.*
- class **WirelessReceiver**

  Sensor *(*p. *1130) class for receiving wireless signals.*
- class **WirelessTransceiver**

  Sensor *(*p. *1130) class for receiving wireless signals.*
- class **WirelessTransmitter**

  Transmitter to send wireless signals.

Typedefs

- typedef std::vector < **CameraSensorPtr** > **CameraSensor_V**
- typedef boost::shared_ptr < **CameraSensor** > **CameraSensorPtr**
- typedef std::vector < **ContactSensorPtr** > **ContactSensor_V**
- typedef boost::shared_ptr < **ContactSensor** > **ContactSensorPtr**
- typedef std::vector < **DepthCameraSensorPtr** > **DepthCameraSensor_V**
- typedef boost::shared_ptr < **DepthCameraSensor** > **DepthCameraSensorPtr**
- typedef boost::shared_ptr < **ForceTorqueSensor** > **ForceTorqueSensorPtr**
- typedef boost::shared_ptr < **GaussianNoiseModel** > **GaussianNoiseModelPtr**
- typedef boost::shared_ptr < **GpsSensor** > **GpsSensorPtr**
- typedef std::vector < **GpuRaySensorPtr** > **GpuRaySensor_V**
- typedef boost::shared_ptr < **GpuRaySensor** > **GpuRaySensorPtr**
- typedef boost::shared_ptr < **ImageGaussianNoiseModel** > **ImageGaussianNoiseModelPtr**

  Shared pointer to ***Noise*** *(*p. 931*).*
- typedef std::vector< **ImuSensorPtr** > **ImuSensor_V**
- typedef boost::shared_ptr < **ImuSensor** > **ImuSensorPtr**
- typedef std::vector < **MultiCameraSensorPtr** > **MultiCameraSensor_V**
- typedef boost::shared_ptr < **MultiCameraSensor** > **MultiCameraSensorPtr**
- typedef boost::shared_ptr< **Noise** > **NoisePtr**
- typedef std::vector< **RaySensorPtr** > **RaySensor_V**
- typedef boost::shared_ptr < **RaySensor** > **RaySensorPtr**
- typedef std::vector< **RFIDSensor** > **RFIDSensor_V**
- typedef boost::shared_ptr < **RFIDSensor** > **RFIDSensorPtr**
- typedef std::vector< **RFIDTag** > **RFIDTag_V**
- typedef boost::shared_ptr < **RFIDTag** > **RFIDTagPtr**
- typedef std::vector< **SensorPtr** > **Sensor_V**
- typedef **Sensor** ∗(∗ **SensorFactoryFn**)()
- typedef boost::shared_ptr< **Sensor** > **SensorPtr**
- typedef boost::shared_ptr < **SonarSensor** > **SonarSensorPtr**
- typedef std::vector < **WirelessReceiver** > **WirelessReceiver_V**
- typedef boost::shared_ptr < **WirelessReceiver** > **WirelessReceiverPtr**
- typedef std::vector < **WirelessTransceiver** > **WirelessTransceiver_V**
- typedef boost::shared_ptr < **WirelessTransceiver** > **WirelessTransceiverPtr**
- typedef std::vector < **WirelessTransmitter** > **WirelessTransmitter_V**
- typedef boost::shared_ptr < **WirelessTransmitter** > **WirelessTransmitterPtr**

Enumerations

- enum **SensorCategory** { **IMAGE** = 0, **RAY** = 1, **OTHER** = 2, **CATEGORY_COUNT** = 3 }

  SensorClass is used to categorize sensors.

Functions

- **GAZEBO_VISIBLE** std::string **create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)

  Create a sensor using SDF.
- **GAZEBO_VISIBLE** void **disable** ()

  Disable sensors.
- **GAZEBO_VISIBLE** void **enable** ()

  Enable sensors.
- **GAZEBO_VISIBLE** bool **fini** ()

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

  Get a sensor using by name.
- **GAZEBO_VISIBLE** bool **init** ()

  initialize the sensor generation loop.
- **GAZEBO_VISIBLE** bool **load** ()

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

  Remove a sensor by name.
- **GAZEBO_VISIBLE** bool **remove_sensors** ()

  Remove all sensors.
- **GAZEBO_VISIBLE** void **run_once** (bool _force=false)

  Run the sensor generation one step.
- **GAZEBO_VISIBLE** void **run_threads** ()

  Run sensors in a threads. This is a non-blocking call.
- **GAZEBO_VISIBLE** 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 boost::shared_ptr<CameraSensor> gazebo::sensors::CameraSensorPtr

9.9.2.3 typedef std::vector<ContactSensorPtr> gazebo::sensors::ContactSensor_V

9.9.2.4 typedef boost::shared_ptr<ContactSensor> gazebo::sensors::ContactSensorPtr

9.9.2.5 typedef std::vector<DepthCameraSensorPtr> gazebo::sensors::DepthCameraSensor_V

9.9.2.6 typedef boost::shared_ptr<DepthCameraSensor> gazebo::sensors::DepthCameraSensorPtr

9.9.2.7 typedef boost::shared_ptr<ForceTorqueSensor> gazebo::sensors::ForceTorqueSensorPtr

9.9.2.8 typedef boost::shared_ptr<GaussianNoiseModel> gazebo::sensors::GaussianNoiseModelPtr

9.9.2.9 typedef boost::shared_ptr<GpsSensor> gazebo::sensors::GpsSensorPtr

9.9.2.10 typedef std::vector<GpuRaySensorPtr> gazebo::sensors::GpuRaySensor_V

9.9.2.11 typedef boost::shared_ptr<GpuRaySensor> gazebo::sensors::GpuRaySensorPtr

9.9.2.12 typedef boost::shared_ptr<ImageGaussianNoiseModel> gazebo::sensors::ImageGaussianNoiseModelPtr

Shared pointer to **Noise** (p. 931).

9.9.2.13 typedef std::vector<ImuSensorPtr> gazebo::sensors::ImuSensor_V

9.9.2.14 typedef boost::shared_ptr<ImuSensor> gazebo::sensors::ImuSensorPtr

9.9.2.15 typedef std::vector<MultiCameraSensorPtr> gazebo::sensors::MultiCameraSensor_V

9.9.2.16 typedef boost::shared_ptr<MultiCameraSensor> gazebo::sensors::MultiCameraSensorPtr

9.9.2.17 typedef boost::shared_ptr<Noise> gazebo::sensors::NoisePtr

9.9.2.18 typedef std::vector<RaySensorPtr> gazebo::sensors::RaySensor_V

9.9.2.19 typedef boost::shared_ptr<RaySensor> gazebo::sensors::RaySensorPtr

9.9.2.20 typedef std::vector<RFIDSensor> gazebo::sensors::RFIDSensor_V

9.9.2.21 typedef boost::shared_ptr<RFIDSensor> gazebo::sensors::RFIDSensorPtr

9.9.2.22 typedef std::vector<RFIDTag> gazebo::sensors::RFIDTag_V

9.9.2.23 typedef boost::shared_ptr<RFIDTag> gazebo::sensors::RFIDTagPtr

9.9.2.24 typedef std::vector<SensorPtr> gazebo::sensors::Sensor_V

9.9.2.25 typedef Sensor*(* gazebo::sensors::SensorFactoryFn)()

9.9.2.26 typedef boost::shared_ptr<Sensor> gazebo::sensors::SensorPtr

9.9.2.27 typedef boost::shared_ptr<SonarSensor> gazebo::sensors::SonarSensorPtr

9.9.2.28 typedef std::vector<WirelessReceiver> gazebo::sensors::Wireless-Receiver_V

9.9.2.29 typedef boost::shared_ptr<WirelessReceiver> gazebo::sensors::WirelessReceiverPtr

9.9.2.30 typedef std::vector<WirelessTransceiver> gazebo::sensors::Wireless-Transceiver_V

9.9.2.31 typedef boost::shared_ptr<WirelessTransceiver> gazebo::sensors::WirelessTransceiverPtr

9.9.2.32 typedef std::vector<WirelessTransmitter> gazebo::sensors::Wireless-Transmitter_V

9.9.2.33 typedef boost::shared_ptr<WirelessTransmitter> gazebo::sensors::WirelessTransmitterPtr

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 type of sensor is not a RAY or IMAGE sensor.

CATEGORY_COUNT Number of **Sensor** (p. 1130) Categories.

9.10 gazebo::transport Namespace Reference

Classes

- class **CallbackHelper**

  A helper class to handle callbacks when messages arrive.
- class **CallbackHelperT**

  Callback helper Template.
- class **Connection**

  Single TCP/IP connection manager.
- class **ConnectionManager**

  Manager of connections.
- class **ConnectionReadTask**
- class **IOManager**

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

  A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.
- class **Publication**

  A publication for a topic.
- class **PublicationTransport**

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

  A publisher of messages on a topic.
- class **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 subscriber to a topic.
- class **SubscriptionTransport**

transport/transport.hh

- class **TopicManager**

  Manages topics and their subscriptions.

Typedefs

- typedef boost::shared_ptr < **CallbackHelper** > **CallbackHelperPtr**

  boost shared pointer to ***transport::CallbackHelper*** *(*p. *235)*
- typedef boost::shared_ptr < **Connection** > **ConnectionPtr**
- typedef boost::shared_ptr < google::protobuf::Message > **MessagePtr**
- typedef boost::shared_ptr< **Node** > **NodePtr**
- typedef boost::shared_ptr < **Publication** > **PublicationPtr**
- typedef boost::shared_ptr < **PublicationTransport** > **PublicationTransportPtr**
- typedef boost::shared_ptr < **Publisher** > **PublisherPtr**
- typedef boost::shared_ptr < **Subscriber** > **SubscriberPtr**
- typedef boost::shared_ptr < **SubscriptionTransport** > **SubscriptionTransportPtr**

Functions

- **GAZEBO_VISIBLE** void **clear_buffers** ()

  Clear any remaining communication buffers.
- **GAZEBO_VISIBLE transport::ConnectionPtr connectToMaster** ()

  Create a connection to master.
- **GAZEBO_VISIBLE** void **fini** ()

  Cleanup the transport component.
- **GAZEBO_VISIBLE** 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.
- **GAZEBO_VISIBLE** void **get_topic_namespaces** (std::list< std::string > &_namespaces)

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

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

  Get a list of all the unique advertised topic names.
- **GAZEBO_VISIBLE** bool **getMinimalComms** ()

  Get whether minimal comms has been enabled.

- **GAZEBO_VISIBLE** std::string **getTopicMsgType** (const std::string &_topicName)

  Get the message typename that is published on the given topic.
- **GAZEBO_VISIBLE** bool **init** (const std::string &_masterHost="", unsigned int _masterPort=0, uint32_t _timeoutIterations=30)

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

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

  Pause or unpause incoming messages.
- template<typename M >
  GAZEBO_VISIBLE void **publish** (const std::string &_topic, const google::protobuf::Message &_message)

  A convenience function for a one-time publication of a message.
- **GAZEBO_VISIBLE** boost::shared_ptr < msgs::Response > **request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

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

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

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

  Run the transport component.
- **GAZEBO_VISIBLE** void **setMinimalComms** (bool _enabled)

  Set whether minimal comms should be used.
- **GAZEBO_VISIBLE** void **stop** ()

  Stop the transport component from running.
- **GAZEBO_VISIBLE** bool **waitForNamespaces** (const **gazebo::common::Time** &_maxWait)

  *Blocks while waiting for topic namespaces from the **Master** (p. 792).*

9.10.1 Typedef Documentation

9.10.1.1 typedef boost::shared_ptr<Connection> gazebo::transport::ConnectionPtr

9.10.1.2 typedef boost::shared_ptr<google::protobuf::Message> gazebo::transport::MessagePtr

9.10.1.3 typedef boost::shared_ptr<Node> gazebo::transport::NodePtr

9.10.1.4 typedef boost::shared_ptr<Publication> gazebo::transport::PublicationPtr

9.10.1.5 typedef boost::shared_ptr<PublicationTransport> gazebo::transport::PublicationTransportPtr

9.10.1.6 typedef boost::shared_ptr<Publisher> gazebo::transport::PublisherPtr

9.10.1.7 typedef boost::shared_ptr<Subscriber> gazebo::transport::SubscriberPtr

9.10.1.8 typedef boost::shared_ptr<SubscriptionTransport> gazebo::transport::SubscriptionTransportPtr

9.11 gazebo::util Namespace Reference

Classes

- class **DiagnosticManager**

  A diagnostic manager class.
- class **DiagnosticTimer**

  A timer designed for diagnostics.
- class **LogPlay**
- class **LogRecord**

  addtogroup gazebo_util

Typedefs

- typedef boost::shared_ptr < **DiagnosticTimer** > **DiagnosticTimerPtr**
- typedef boost::shared_ptr < OpenALSink > **OpenALSinkPtr**
- typedef boost::shared_ptr < OpenALSource > **OpenALSourcePtr**

9.11.1 Typedef Documentation

9.11.1.1 typedef boost::shared_ptr<DiagnosticTimer> gazebo::util::DiagnosticTimerPtr

9.11.1.2 typedef boost::shared_ptr<OpenALSink> gazebo::util::OpenALSinkPtr

9.11.1.3 typedef boost::shared_ptr<OpenALSource> gazebo::util::OpenALSourcePtr

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. *135*).

9.16 Ogre Namespace Reference

9.17 ogre Namespace Reference

9.18 OVR Namespace Reference

Namespaces

- namespace **Util**

9.19 OVR::Util Namespace Reference

Namespaces

- namespace **Render**

9.20 OVR::Util::Render Namespace Reference

9.21 SimTK Namespace Reference

9.22 SkyX Namespace Reference

Chapter 10

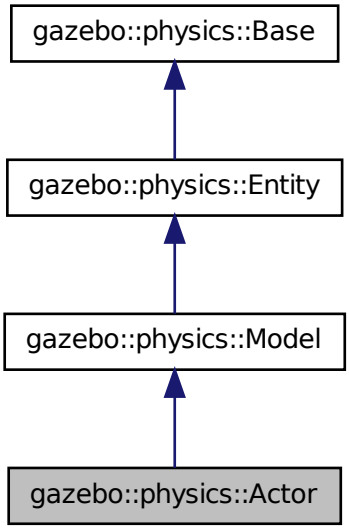
Class Documentation

10.1 gazebo::physics::Actor Class Reference

Actor (p. 165) 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. *201)* *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.
- uint32_t **visualId**

  ID for this visual.
- std::string **visualName**

  Name of the visual.

10.1.1 Detailed Description

Actor (p. 165) class enables GPU based mesh model / skeleton scriptable animation.

10.1.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_parent*	Parent object

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

Destructor.

10.1.3 Member Function Documentation

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

Finalize the actor.

Reimplemented from **gazebo::physics::Model** (p. 851).

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

Get the SDF values for the actor.

Returns

Pointer to the SDF values.

Reimplemented from **gazebo::physics::Model** (p. 855).

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

Initialize the actor.

Reimplemented from **gazebo::physics::Model** (p. 857).

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

Returns true when actor is playing animation.

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

Load the actor.

Parameters

in	*_sdf*	SDF parameters

Reimplemented from **gazebo::physics::Model** (p. 857).

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

Start playing the script.

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

Stop playing the script.

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

Update the actor.

Reimplemented from **gazebo::physics::Model** (p. 863).

10.1.3.9 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. 863).

10.1.4 Member Data Documentation

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

True if the actor is being updated.

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

True if the actor should start running automatically.

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

Where to send bone info.

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

True to interpolate along x direction.

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

Last position of the actor.

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

Time the scipt was last updated.

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

THe last trajectory.

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

True if the animation should loop.

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

Base (p. 201) link.

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

Pointer to the actor's mesh.

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

THe old action.

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

Length of the actor's path.

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

Time when the animation was started.

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

Time of the previous frame.

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

Time length of a scipt.

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

Skeleton animations.

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

The actor's skeleton.

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

Skeleton to naode map.

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

Filename for the skin.

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

Scaling factor to apply to the skin.

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

Amount of time to delay start by.

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

All the trajectories.

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

Trajectory information.

10.1.4.24 uint32_t gazebo::physics::Actor::visualId [protected]

ID for this visual.

10.1.4.25 std::string gazebo::physics::Actor::visualName [protected]

Name of the visual.

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

- **Actor.hh**

10.2 gazebo::math::Angle Class Reference

An angle and related functions.

`#include <math/gzmath.hh>`

Public Member Functions

- **Angle** ()

  Constructor.
- **Angle** (double _radian)

  Copy Constructor.
- **Angle** (const **Angle** &_angle)

  Copy constructor.
- virtual ~**Angle** ()

  Destructor.
- double **Degree** () const

  Get the angle in degrees.

- 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.

Static Public Attributes

- static const **Angle HalfPi**

  math::Angle *(*p. *173)(M_PI ∗ 0.5)*
- static const **Angle Pi**

  math::Angle(M_PI)
- static const **Angle TwoPi**

  math::Angle(M_PI ∗ 2)
- static const **Angle Zero**

  math::Angle(0)

Friends

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

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

  Stream extraction operator.

10.2.1 Detailed Description

An angle and related functions.

10.2.2 Constructor & Destructor Documentation

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

Constructor.

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

Copy Constructor.

Parameters

in	*_radian*	Radians

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

Copy constructor.

Parameters

in	*_angle*	**Angle** (p. 173) to copy

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

Destructor.

10.2.3 Member Function Documentation

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

Get the angle in degrees.

Returns

double containing the angle's degree value

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

Normalize the angle in the range -Pi to Pi.

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

Inequality.

Parameters

in	*_angle*	**Angle** (p. 173) to check for inequality

Returns

true if this != _angle

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

Dereference operator.

Returns

Double containing the angle's radian value

10.2.3.5 Angle gazebo::math::Angle::operator∗ (const Angle & _angle) const

Multiplication operator, result = this ∗ _angle.

Parameters

in	*_angle*	**Angle** (p. 173) for multiplication

Returns

the new angle

10.2.3.6 Angle gazebo::math::Angle::operator∗= (const Angle & _angle)

Multiplication set, this = this ∗ _angle.

Parameters

in	*_angle*	**Angle** (p. 173) for multiplication

Returns

angle

10.2.3.7 Angle gazebo::math::Angle::operator+ (const Angle & _angle) const

Addition operator, result = this + _angle.

Parameters

in	*_angle*	**Angle** (p. 173) for addition

Returns

the new angle

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

Addition set, this = this + _angle.

Parameters

in	*_angle*	**Angle** (p. 173) for addition

Returns

angle

10.2.3.9 Angle gazebo::math::Angle::operator- (const Angle & _angle) const

Substraction, result = this - _angle.

Parameters

in	*_angle*	**Angle** (p. 173) for substraction

Returns

the new angle

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

Subtraction set, this = this - _angle.

Parameters

in	*_angle*	**Angle** (p. 173) for subtraction

Returns

angle

10.2.3.11 Angle gazebo::math::Angle::operator/ (const Angle & _angle) const

Division, result = this / _angle.

Parameters

in	*_angle*	**Angle** (p. 173) for division

Returns

the new angle

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

Division set, this = this / _angle.

Parameters

in	*_angle*	**Angle** (p. 173) for division

Returns

angle

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

Less than operator.

Parameters

in	*_angle*	**Angle** (p. 173) to check

Returns

true if this < _angle

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

Less or equal operator.

Parameters

in	*_angle*	**Angle** (p. 173) to check

Returns

true if this <= _angle

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

Equality operator, result = this == _angle.

Parameters

in	*_angle*	**Angle** (p. 173) to check for equality

Returns

true if this == _angle

10.2.3.16 bool gazebo::math::Angle::operator> (const Angle & *_angle*) const

Greater than operator.

Parameters

in	*_angle*	**Angle** (p. 173) to check

Returns

true if this > _angle

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

Greater or equal operator.

Parameters

in	*_angle*	**Angle** (p. 173) to check

Returns

true if this >= _angle

10.2.3.18 double gazebo::math::Angle::Radian ( ) const

Get the angle in radians.

Returns

double containing the angle's radian value

10.2.3.19 void gazebo::math::Angle::SetFromDegree (double _degree)

Set the value from an angle in degrees.

Parameters

in	*_degree*	Degree value

10.2.3.20 void gazebo::math::Angle::SetFromRadian (double _radian)

Set the value from an angle in radians.

Parameters

in	*_radian*	Radian value

10.2.4 Friends And Related Function Documentation

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

Stream insertion operator.

Outputs in degrees

Parameters

in	*_out*	output stream
in	*_a*	angle to output

Returns

The output stream

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

Stream extraction operator.

Assumes input is in degrees

Parameters

in	input stream
pt	angle to read value into

Returns

The input stream

10.2.5 Member Data Documentation

10.2.5.1 const Angle gazebo::math::Angle::HalfPi [static]

math::Angle (p. 173)(M_PI ∗ 0.5)

10.2.5.2 const Angle gazebo::math::Angle::Pi [static]

math::Angle(M_PI)

10.2.5.3 const Angle gazebo::math::Angle::TwoPi [static]

math::Angle(M_PI ∗ 2)

10.2.5.4 const Angle gazebo::math::Angle::Zero [static]

math::Angle(0)

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

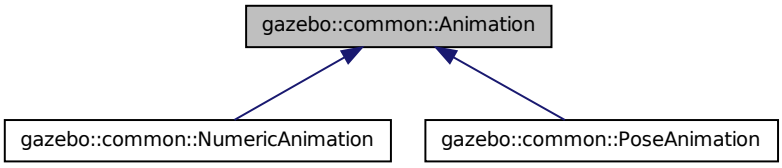
- **Angle.hh**

10.3 gazebo::common::Animation Class Reference

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

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::Animation:



Public Member Functions

- **Animation** (const std::string &_name, double _length, bool _loop)

  Constructor.
- virtual ~**Animation** ()

  Destructor.
- void **AddTime** (double _time)

  Add time to the animation.
- **KeyFrame** * **GetKeyFrame** (unsigned int _index) const

  Get a key frame using an index value.
- unsigned int **GetKeyFrameCount** () const

  Return the number of key frames in the animation.
- double **GetLength** () const

  Return the duration of the animation.
- double **GetTime** () const

  Return the current time position.
- void **SetLength** (double _len)

  Set the duration of the animation.
- void **SetTime** (double _time)

  Set the current time position of the animation.

Protected Types

- typedef std::vector< **KeyFrame** * > **KeyFrame_V**

  array of keyframe type alias

Protected Member Functions

- double **GetKeyFramesAtTime** (double _time, **KeyFrame** **_kf1, **KeyFrame** **-_kf2, unsigned int &_firstKeyIndex) const

  Get the two key frames that bound a time value.

Protected Attributes

- bool **build**

  determines if the interpolation splines need building
- **KeyFrame_V keyFrames**

  array of key frames
- double **length**

  animation duration
- bool **loop**

  true if animation repeats
- std::string **name**

  animation name
- double **timePos**

  current time position

10.3.1 Detailed Description

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

10.3.2 Member Typedef Documentation

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

array of keyframe type alias

10.3.3 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

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

Destructor.

10.3.4 Member Function Documentation

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

Add time to the animation.

Parameters

in	*_time*	The amount of time to add in seconds

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

Get a key frame using an index value.

Parameters

in	*_index*	The index of the key frame

Returns

A pointer the keyframe, NULL if the _index is invalid

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

Return the number of key frames in the animation.

Returns

The number of keyframes

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

Get the two key frames that bound a time value.

Parameters

in	*_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

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

Return the duration of the animation.

Returns

Duration of the animation in seconds

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

Return the current time position.

Returns

The time position in seconds

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

Set the duration of the animation.

Parameters

in	*_len*	The length of the animation in seconds

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

Set the current time position of the animation.

Parameters

in	*_time*	The time position in seconds

10.3.5 Member Data Documentation

10.3.5.1 bool gazebo::common::Animation::build [mutable, protected]

determines if the interpolation splines need building

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

array of key frames

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

animation duration

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

true if animation repeats

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

animation name

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

current time position

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

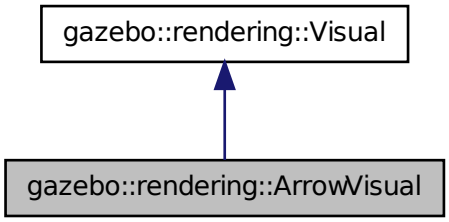
- **Animation.hh**

10.4 gazebo::rendering::ArrowVisual Class Reference

Basic arrow visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::ArrowVisual:



Public Member Functions

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

  Constructor.
- virtual ~**ArrowVisual** ()

  Destructor.
- virtual void **Load** ()

  Load the visual with default parameters.
- void **ShowRotation** ()

  Show the rotation of the visual.

10.4.1 Detailed Description

Basic arrow visualization.

10.4.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_name*	Name of the arrow visual
in	*_vis*	Pointer to the parent visual

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

Destructor.

10.4.3 Member Function Documentation

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

Load the visual with default parameters.

Reimplemented from **gazebo::rendering::Visual** (p. 1495).

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

Show the rotation of the visual.

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

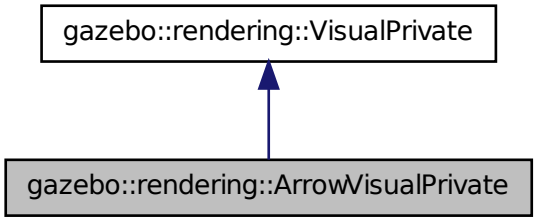
- **ArrowVisual.hh**

10.5 gazebo::rendering::ArrowVisualPrivate Class Reference

Private data for the Arrow **Visual** (p. 1477) class.

`#include <ArrowVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::ArrowVisualPrivate:



Public Attributes

- Ogre::SceneNode ∗ **headNode**
- Ogre::SceneNode ∗ **rotationNode**
- Ogre::SceneNode ∗ **shaftNode**

10.5.1 Detailed Description

Private data for the Arrow **Visual** (p. 1477) class.

10.5.2 Member Data Documentation

10.5.2.1 Ogre::SceneNode∗ gazebo::rendering::ArrowVisualPrivate::headNode

10.5.2.2 Ogre::SceneNode∗ gazebo::rendering::ArrowVisualPrivate::rotationNode

10.5.2.3 Ogre::SceneNode∗ gazebo::rendering::ArrowVisualPrivate::shaftNode

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

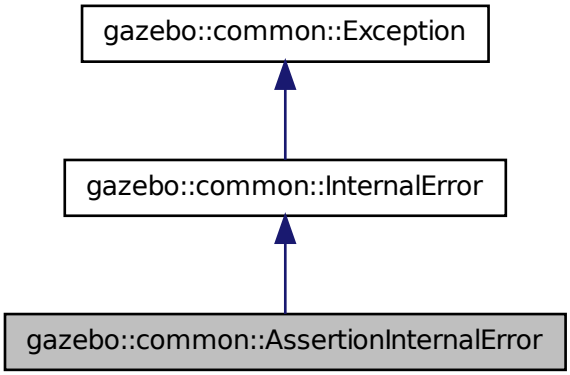
- **ArrowVisualPrivate.hh**

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::common::AudioDecoder Class Reference

An audio decoder based on FFMPEG.

`#include <common/common.hh>`

Public Member Functions

- **AudioDecoder** ()

  Constructor.
- virtual ~**AudioDecoder** ()

  Destructor.
- bool **Decode** (uint8_t **_outBuffer, unsigned int *_outBufferSize)

  Decode the loaded audio file.
- std::string **GetFile** () const

  Get the audio filename that was set.
- int **GetSampleRate** ()

Get the sample rate from the latest decoded file.

- bool **SetFile** (const std::string &_filename)

  Set the file to decode.

10.7.1 Detailed Description

An audio decoder based on FFMPEG.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 gazebo::common::AudioDecoder::AudioDecoder ( )

Constructor.

10.7.2.2 virtual gazebo::common::AudioDecoder::~AudioDecoder ( ) [virtual]

Destructor.

10.7.3 Member Function Documentation

10.7.3.1 bool gazebo::common::AudioDecoder::Decode (uint8_t ** *_outBuffer*, unsigned int * *_outBufferSize*)

Decode the loaded audio file.

See also

AudioDecoder::SetFile (p. 194)

Parameters

out	*_outBuffer*	Buffer that holds the decoded audio data.
out	*_outBuffer-Size*	Size of the _outBuffer.

Returns

True if decoding was succesful.

10.7.3.2 std::string gazebo::common::AudioDecoder::GetFile ( ) const

Get the audio filename that was set.

Returns

The name of the set audio file.

See also

AudioDecoder::SetFile (p. 194)

10.7.3.3 int gazebo::common::AudioDecoder::GetSampleRate ( )

Get the sample rate from the latest decoded file.

Returns

Integer sample rate, such as 44100.

10.7.3.4 bool gazebo::common::AudioDecoder::SetFile (const std::string & *_filename*)

Set the file to decode.

Parameters

in	*_filename*	Path to an audio file.

Returns

True if the file was successfull opened.

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

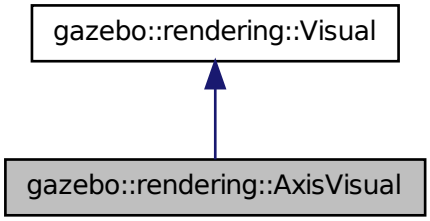
- **AudioDecoder.hh**

10.8 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.

10.8.1 Detailed Description

Basic axis visualization.

10.8.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_name*	Name of the **AxisVisual** (p. 194)
in	*_vis*	Parent visual

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

Destructor.

10.8.3 Member Function Documentation

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

Load the axis visual.

Reimplemented from **gazebo::rendering::Visual** (p. 1495).

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

Scale the X axis.

Parameters

in	*_scale*	Scaling factor

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

Scale the Y axis.

Parameters

in	*_scale*	Scaling factor

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

Scale the Z axis.

Parameters

in	*_scale*	Scaling factor

10.8.3.5 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

10.8.3.6 void gazebo::rendering::AxisVisual::ShowRotation (unsigned int _axis)

Load the rotation tube.

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

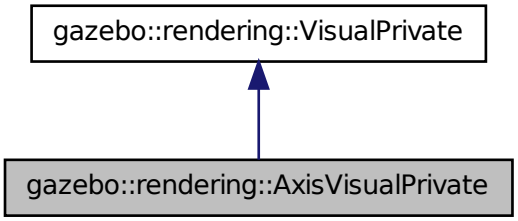
- **AxisVisual.hh**

10.9 gazebo::rendering::AxisVisualPrivate Class Reference

Private data for the Axis **Visual** (p. 1477) class.

`#include <AxisVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::AxisVisualPrivate:



Public Attributes

- **ArrowVisualPtr xAxis**

  Pointer to the x-axis visual.
- **ArrowVisualPtr yAxis**

  Pointer to the y-axis visual.
- **ArrowVisualPtr zAxis**

  Pointer to the z-axis visual.

10.9.1 Detailed Description

Private data for the Axis **Visual** (p. 1477) class.

10.9.2 Member Data Documentation

10.9.2.1 ArrowVisualPtr gazebo::rendering::AxisVisualPrivate::xAxis

Pointer to the x-axis visual.

10.9.2.2 ArrowVisualPtr gazebo::rendering::AxisVisualPrivate::yAxis

Pointer to the y-axis visual.

10.9.2.3 ArrowVisualPtr gazebo::rendering::AxisVisualPrivate::zAxis

Pointer to the z-axis visual.

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

- **AxisVisualPrivate.hh**

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

Base (p. 201) class for a ball joint.

`#include <physics/physics.hh>`

Public Member Functions

- **BallJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**BallJoint** ()

  Destructor.
- virtual unsigned int **GetAngleCount** () const
- void **Load** (sdf::ElementPtr _sdf)

  *Template to ::Load the **BallJoint** (p. 199).*

Protected Member Functions

- virtual void **Init** ()

  Initialize joint.

10.10.1 Detailed Description

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

Base (p. 201) class for a ball joint.

Each physics engine should implement this class.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 template<class T> gazebo::physics::BallJoint< T >::BallJoint (BasePtr _parent) [inline, explicit]

Constructor.

Parameters

in	*_parent*	Pointer to the parent link.

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

Destructor.

10.10.3 Member Function Documentation

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

10.10.3.2 template<class T> virtual void gazebo::physics::BallJoint< T >::Init ( ) [inline, protected, virtual]

Initialize joint.

Reimplemented in **gazebo::physics::DARTBallJoint** (p. 379).

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

Template to ::Load the **BallJoint** (p. 199).

Parameters

in	*_sdf*	SDF to load the joint from.

Reimplemented in **gazebo::physics::SimbodyBallJoint** (p. 1170), and **gazebo::physics::DARTBallJoint** (p. 380).

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

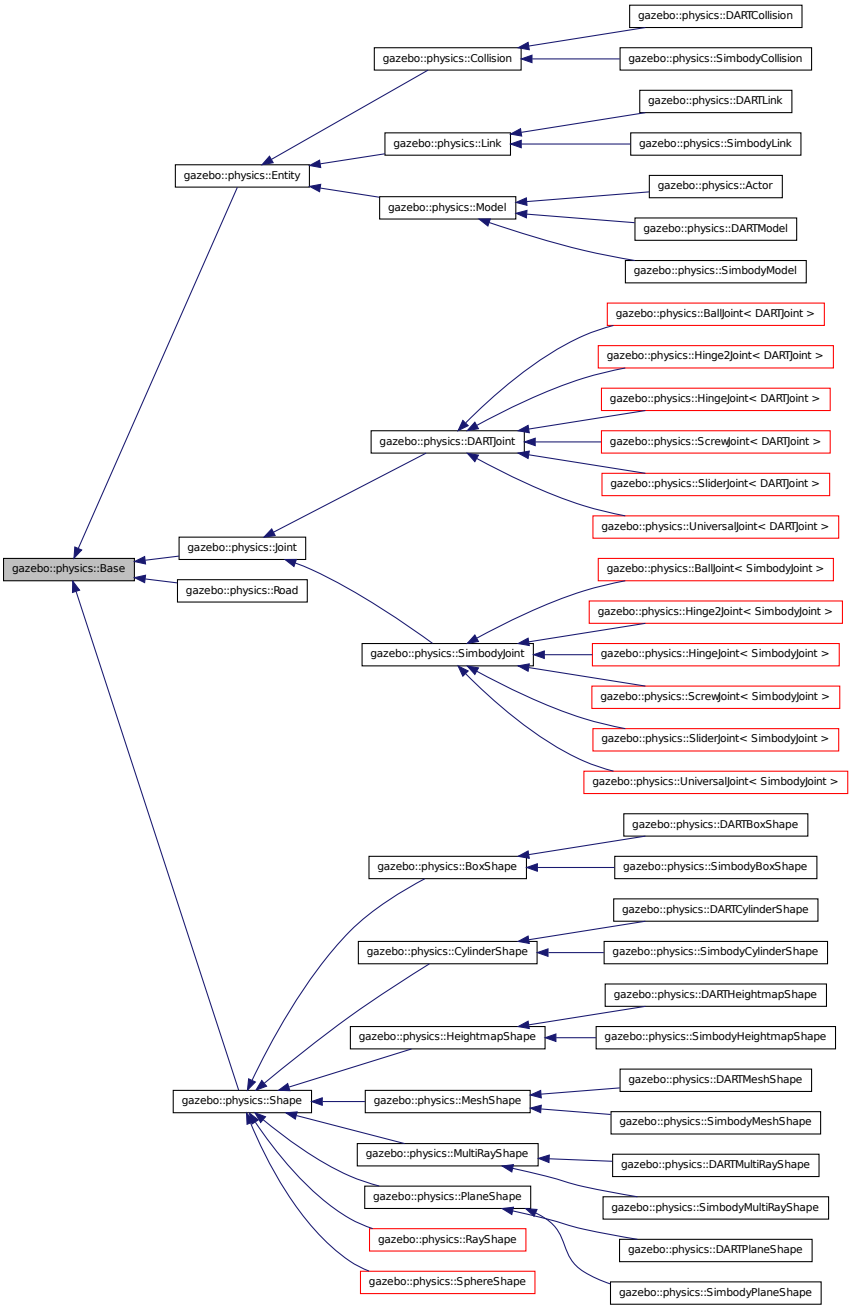
- **BallJoint.hh**

10.11 gazebo::physics::Base Class Reference

Base (p. 201) 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, **GEARBOX_JOINT** = 0x00002000, **SHAPE** = 0x00010000, **BOX_SHAPE** = 0x00020000, **CYLINDER_SHAPE** = 0x00040000, **HEIGHTMAP_SHAPE** = 0x00080000, **MAP_SHAPE** = 0x00100000, **MULTIRAY_SHAPE** = 0x00200000, **RAY_SHAPE** = 0x00400000, **PLANE_SHAPE** = 0x00800000, **SPHERE_SHAPE** = 0x01000000, **MESH_SHAPE** = 0x02000000, **SENSOR_COLLISION** = 0x10000000 }

  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.
- uint32_t **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** (bool _prependWorldName=false) const

  Return the name of this entity with the model scope 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. 1529) 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. 205)'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 Member Functions

- void **ComputeScopedName** ()

  Compute the scoped name of this object based on its parents.

Protected Attributes

- **Base_V children**

  Children of this entity.

- **BasePtr parent**

  Parent of this entity.

- sdf::ElementPtr **sdf**

  The SDF values for this object.

- **WorldPtr world**

  Pointer to the world.

10.11.1 Detailed Description

Base (p. 201) class for most physics classes.

10.11.2 Member Enumeration Documentation

10.11.2.1 enum gazebo::physics::Base::EntityType

Unique identifiers for all entity types.

Enumerator:

BASE **Base** (p. 201) type.

ENTITY **Entity** (p. 500) type.

MODEL **Model** (p. 846) type.

LINK **Link** (p. 739) type.

COLLISION **Collision** (p. 295) type.

ACTOR **Actor** (p. 165) type.

LIGHT Light type.

VISUAL Visual type.

JOINT **Joint** (p. 669) type.

BALL_JOINT **BallJoint** (p. 199) type.

HINGE2_JOINT Hing2Joint type.

HINGE_JOINT **HingeJoint** (p. 635) type.

SLIDER_JOINT **SliderJoint** (p. 1294) type.

SCREW_JOINT **ScrewJoint** (p. 1118) type.

UNIVERSAL_JOINT **UniversalJoint** (p. 1404) type.

GEARBOX_JOINT **GearboxJoint** (p. 571) type.

SHAPE **Shape** (p. 1161) type.

BOX_SHAPE **BoxShape** (p. 228) type.

CYLINDER_SHAPE **CylinderShape** (p. 370) type.

HEIGHTMAP_SHAPE **HeightmapShape** (p. 628) type.

MAP_SHAPE MapShape type.

MULTIRAY_SHAPE **MultiRayShape** (p. 901) type.

RAY_SHAPE **RayShape** (p. 1060) type.

PLANE_SHAPE **PlaneShape** (p. 987) type.

SPHERE_SHAPE **SphereShape** (p. 1307) type.

MESH_SHAPE **MeshShape** (p. 841) type.

SENSOR_COLLISION Indicates a collision shape used for sensing.

10.11.3 Constructor & Destructor Documentation

10.11.3.1 gazebo::physics::Base::Base (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of this object

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

Destructor.

10.11.4 Member Function Documentation

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

Add a child to this entity.

Parameters

in	*_child*	Child entity.

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

Add a type specifier.

Parameters

in	*_type*	New type to append to this objects type definition.

10.11.4.3 void gazebo::physics::Base::ComputeScopedName ( ) [protected]

Compute the scoped name of this object based on its parents.

See also

Base::GetScopedName (p. 210)

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

Finialize the object.

Reimplemented in **gazebo::physics::Joint** (p. 679), **gazebo::physics::Actor** (p. 169), **gazebo::physics::Link** (p. 750), **gazebo::physics::Model** (p. 851), **gazebo::physics::Entity** (p. 504), **gazebo::physics::DARTModel** (p. 435), **gazebo-**

::physics::Collision (p. 299), **gazebo::physics::DARTLink** (p. 423), **gazebo-::physics::SimbodyLink** (p. 1212), and **gazebo::physics::DARTCollision** (p. 387).

10.11.4.5 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 pointer to the object, NULL if not found

10.11.4.6 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 pointer to the object, NULL if not found

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

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

Get a child by name.

Parameters

in	*_name*	Name of the child.

Returns

A pointer to the object, NULL if not found

10.11.4.9 unsigned int gazebo::physics::Base::GetChildCount ( ) const

Get the number of children.

Returns

The number of children.

10.11.4.10 uint32_t gazebo::physics::Base::GetId ( ) const

Return the ID of this entity.

This id is unique.

Returns

Integer ID.

10.11.4.11 std::string gazebo::physics::Base::GetName ( ) const

Return the name of the entity.

Returns

Name of the entity.

10.11.4.12 BasePtr gazebo::physics::Base::GetParent ( ) const

Get the parent.

Returns

Pointer to the parent entity.

Reimplemented in **gazebo::physics::Joint** (p. 689).

10.11.4.13 int gazebo::physics::Base::GetParentId ( ) const

Return the ID of the parent.

Returns

Integer ID.

10.11.4.14 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.

10.11.4.15 std::string gazebo::physics::Base::GetScopedName (bool *_prependWorldName* = `false`) const

Return the name of this entity with the model scope model1::...::modelN::entityName.

Parameters

`in`	*_prependWorldName*	True to prended the returned string with the world name. - The result will be world::model1::...::modelN::entityName.

Returns

The scoped name.

10.11.4.16 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. 169), and **gazebo::physics::Model** (p. 855).

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

Get the full type definition.

Returns

The full type definition.

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

Get the **World** (p. 1529) this object is in.

Returns

The **World** (p. 1529) this object is part of.

10.11.4.19 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.

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

Initialize the object.

Reimplemented in **gazebo::physics::RayShape** (p. 1064), **gazebo::physics::Joint** (p. 693), **gazebo::physics::Actor** (p. 169), **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1120), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 1120),

gazebo::physics::Link (p. 760), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1406), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1406), **gazebo::physics::BallJoint< DARTJoint >** (p. 200), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 200), **gazebo::physics::Model** (p. 857), **gazebo::physics::HeightmapShape** (p. 632), **gazebo::physics::HingeJoint< DARTJoint >** (p. 637), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 637), **gazebo::physics::Collision** (p. 303), **gazebo::physics::MeshShape** (p. 844), **gazebo::physics::Road** (p. 1085), **gazebo::physics::DARTLink** (p. 426), **gazebo::physics::MultiRayShape** (p. 908), **gazebo::physics::PlaneShape** (p. 990), **gazebo::physics::SimbodyLink** (p. 1215), **gazebo::physics::DARTModel** (p. 435), **gazebo::physics::DARTScrewJoint** (p. 456), **gazebo::physics::Shape** (p. 1164), **gazebo::physics::DARTJoint** (p. 413), **gazebo::physics::SphereShape** (p. 1309), **gazebo::physics::BoxShape** (p. 230), **gazebo::physics::CylinderShape** (p. 372), **gazebo::physics::DARTCollision** (p. 388), **gazebo::physics::DARTHinge2Joint** (p. 397), **gazebo::physics::DARTHingeJoint** (p. 403), **gazebo::physics::SimbodyHeightmapShape** (p. 1181), **gazebo::physics::SimbodyModel** (p. 1223), **gazebo::physics::SimbodyMeshShape** (p. 1221), **gazebo::physics::DARTBallJoint** (p. 379), **gazebo::physics::DARTHeightmapShape** (p. 393), **gazebo::physics::DARTSliderJoint** (p. 463), **gazebo::physics::DARTUniversalJoint** (p. 472), and **gazebo::physics::DARTMeshShape** (p. 432).

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

True if the entity is selected by the user.

Returns

True if the entity is selected.

10.11.4.22 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. 693), **gazebo::physics::Actor** (p. 169), **gazebo::physics::Link** (p. 761), **gazebo::physics::SimbodySliderJoint** (p. 1256), **gazebo::physics::Entity** (p. 509), **gazebo::physics::Model** (p. 857), **gazebo::physics::BallJoint< DARTJoint >** (p. 200), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 200), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1406), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1406),

gazebo::physics::Hinge2Joint< DARTJoint > (p. 635), **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 635), **gazebo::physics::HeightmapShape** (p. 632), **gazebo::physics::Collision** (p. 304), **gazebo::physics::HingeJoint< DARTJoint >** (p. 637), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 637), **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1120), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 1120), **gazebo::physics::SliderJoint< DARTJoint >** (p. 1295), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 1295), **gazebo::physics::Road** (p. 1085), **gazebo::physics::SimbodyCollision** (p. 1177), **gazebo::physics::DARTLink** (p. 427), **gazebo::physics::SimbodyHingeJoint** (p. 1191), **gazebo::physics::SimbodyLink** (p. 1215), **gazebo::physics::SimbodyUniversalJoint** (p. 1264), **gazebo::physics::DARTModel** (p. 435), **gazebo::physics::SimbodyHinge2Joint** (p. 1186), **gazebo::physics::SimbodyScrewJoint** (p. 1249), **gazebo::physics::SimbodyJoint** (p. 1201), **gazebo::physics::DARTJoint** (p. 414), **gazebo::physics::DARTCollision** (p. 388), **gazebo::physics::SimbodyBallJoint** (p. 1170), **gazebo::physics::DARTHinge2Joint** (p. 398), **gazebo::physics::DARTHingeJoint** (p. 404), **gazebo::physics::SimbodyModel** (p. 1223), **gazebo::physics::SimbodyMeshShape** (p. 1221), **gazebo::physics::DARTBallJoint** (p. 380), **gazebo::physics::DARTScrewJoint** (p. 457), **gazebo::physics::DARTSliderJoint** (p. 464), **gazebo::physics::DARTUniversalJoint** (p. 473), and **gazebo::physics::DARTMeshShape** (p. 432).

10.11.4.23 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. 201) object to compare with.

Returns

True if the entities are the same.

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

Print this object to screen via gzmsg.

Parameters

in	*_prefix*	Usually a set of spaces.

10.11.4.25 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.

10.11.4.26 void gazebo::physics::Base::RemoveChild (const std::string & *_name*)

Remove a child by name.

Parameters

in	*_name*	Name of the child.

10.11.4.27 void gazebo::physics::Base::RemoveChildren ( )

Remove all children.

10.11.4.28 virtual void gazebo::physics::Base::Reset ( ) [virtual]

Reset the object.

Reimplemented in **gazebo::physics::Joint** (p. 694), **gazebo::physics::Model** (p. 858), **gazebo::physics::Link** (p. 762), **gazebo::physics::Entity** (p. 510), **gazebo::physics::DARTJoint** (p. 414), and **gazebo::physics::SimbodyJoint** (p. 1201).

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

Calls recursive Reset on one of the **Base::EntityType** (p. 205)'s.

Parameters

in	*_resetType*	The type of reset operation

10.11.4.30 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. 511).

10.11.4.31 void gazebo::physics::Base::SetParent (BasePtr _parent)

Set the parent.

Parameters

in	*_parent*	Parent object.

10.11.4.32 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.

10.11.4.33 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. 767).

10.11.4.34 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. 1529) this object is part of.

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

Update the object.

Reimplemented in **gazebo::physics::MultiRayShape** (p. 909), **gazebo::physics::Joint** (p. 701), **gazebo::physics::Actor** (p. 170), **gazebo::physics::RayShape** (p. 1065), **gazebo::physics::Model** (p. 863), **gazebo::physics::DARTModel** (p. 436), **gazebo::physics::DARTRayShape** (p. 95), **gazebo::physics::MeshShape** (p. 845), **gazebo::physics::SimbodyRayShape** (p. 1243), and **gazebo::physics::DARTMeshShape** (p. 433).

10.11.4.36 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. 702), **gazebo::physics::Actor** (p. 170), **gazebo::physics::Link** (p. 768), **gazebo::physics::Model** (p. 863), **gazebo::physics::Entity** (p. 512), and **gazebo::physics::Collision** (p. 306).

10.11.5 Member Data Documentation

10.11.5.1 Base_V gazebo::physics::Base::children [protected]

Children of this entity.

10.11.5.2 BasePtr gazebo::physics::Base::parent [protected]

Parent of this entity.

10.11.5.3 sdf::ElementPtr gazebo::physics::Base::sdf [protected]

The SDF values for this object.

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

Pointer to the world.

Reimplemented in **gazebo::physics::SimbodyJoint** (p. 1207).

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

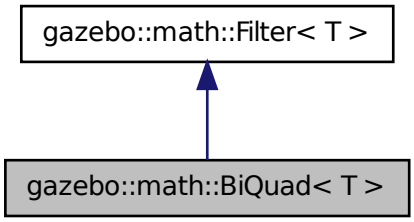
- **Base.hh**

10.12 gazebo::math::BiQuad< T > Class Template Reference

Bi-quad filter base class.

`#include <math/gzmath.hh>`

Inheritance diagram for gazebo::math::BiQuad< T >:



Public Member Functions

- **BiQuad** ()

  Constructor.
- **BiQuad** (double _fc, double _fs)

  Constructor.

- virtual const T & **process** (const T &_x)

  Update the filter's output.
- void **SetFc** (double _fc, double _fs)

  Set the cutoff frequency and sample rate.
- void **SetFc** (double _fc, double _fs, double _q)

  Set the cutoff frequency, sample rate and Q coefficient.
- virtual void **SetValue** (const T &_val)

  Set the current filter's output.

Protected Attributes

- double **a0**

  Input gain control coefficients.
- double **a1**
- double **a2**
- double **b0**
- double **b1**
- double **b2**
- T **x1**

  Gain of the feedback coefficients.
- T **x2**
- T **y1**
- T **y2**

10.12.1 Detailed Description

template<class T>class gazebo::math::BiQuad< T >

Bi-quad filter base class.

See also

`http://www.earlevel.com/main/2003/03/02/the-bilinear-z-transform/`

10.12.2 Constructor & Destructor Documentation

10.12.2.1 template<class T> gazebo::math::BiQuad< T >::BiQuad ( ) [inline]

Constructor.

10.12.2.2 template<class T> gazebo::math::BiQuad< T >::BiQuad (double _fc, double _fs) [inline]

Constructor.

Parameters

in	*_fc*	Cutoff frequency.
in	*_fs*	Sample rate.

10.12.3 Member Function Documentation

10.12.3.1 template<class T> virtual const T& gazebo::math::BiQuad< T >::process (const T & _x) [inline, virtual]

Update the filter's output.

Parameters

in	*_x*	Input value.

Returns

The filter's current output.

10.12.3.2 template<class T> void gazebo::math::BiQuad< T >::SetFc (double _fc, double _fs) [inline, virtual]

Set the cutoff frequency and sample rate.

Parameters

in	*_fc*	Cutoff frequency.
in	*_fs*	Sample rate.

Implements **gazebo::math::Filter< T >** (p. 555).

10.12.3.3 template<class T> void gazebo::math::BiQuad< T >::SetFc (double _fc, double _fs, double _q) [inline]

Set the cutoff frequency, sample rate and Q coefficient.

Parameters

in	*_fc*	Cutoff frequency.
in	*_fs*	Sample rate.
in	*_q*	Q coefficient.

10.12.3.4 template<class T> virtual void gazebo::math::BiQuad< T >::SetValue (const T & *_val*) [inline, virtual]

Set the current filter's output.

Parameters

in	*_val*	New filter's output.

Reimplemented from **gazebo::math::Filter< T >** (p. 555).

10.12.4 Member Data Documentation

10.12.4.1 template<class T> double gazebo::math::BiQuad< T >::a0 [protected]

Input gain control coefficients.

10.12.4.2 template<class T> double gazebo::math::BiQuad< T >::a1 [protected]

10.12.4.3 template<class T> double gazebo::math::BiQuad< T >::a2 [protected]

10.12.4.4 template<class T> double gazebo::math::BiQuad< T >::b0 [protected]

10.12.4.5 template<class T> double gazebo::math::BiQuad< T >::b1 [protected]

10.12.4.6 template<class T> double gazebo::math::BiQuad< T >::b2 [protected]

10.12.4.7 template<class T> T gazebo::math::BiQuad< T >::x1 [protected]

Gain of the feedback coefficients.

10.12.4.8 template<class T> T gazebo::math::BiQuad< T >::x2 [protected]

10.12.4.9 template<class T> T gazebo::math::BiQuad< T >::y1 [protected]

10.12.4.10 template<class T> T gazebo::math::BiQuad< T >::y2 [protected]

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

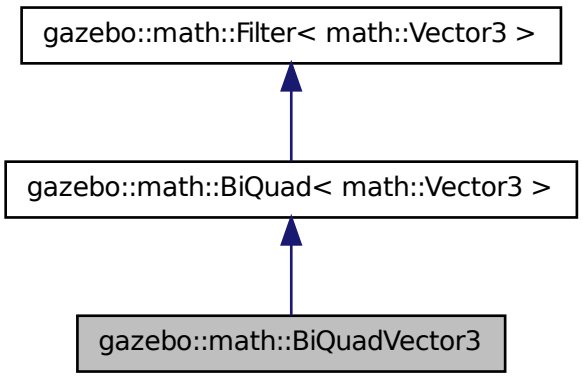
- **Filter.hh**

10.13 gazebo::math::BiQuadVector3 Class Reference

BiQuad (p. 217) vector3 filter.

`#include <math/gzmath.hh>`

Inheritance diagram for gazebo::math::BiQuadVector3:



Public Member Functions

- **BiQuadVector3** ()
  Constructor.
- **BiQuadVector3** (double _fc, double _fs)
  Constructor.

10.13.1 Detailed Description

BiQuad (p. 217) vector3 filter.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 gazebo::math::BiQuadVector3::BiQuadVector3 ( ) [inline]

Constructor.

10.13.2.2 gazebo::math::BiQuadVector3::BiQuadVector3 (double *_fc*, double *_fs*) [inline]

Constructor.

Parameters

in	*_fc*	Cutoff frequency.
in	*_fs*	Sample rate.

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

- **Filter.hh**

10.14 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) const

  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.

10.14.1 Detailed Description

Mathematical representation of a box and related functions.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 gazebo::math::Box::Box ( )

Default constructor.

10.14.2.2 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

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

Copy Constructor.

Parameters

in	*_b*	**Box** (p. 222) to copy

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

Destructor.

10.14.3 Member Function Documentation

10.14.3.1 math::Vector3 gazebo::math::Box::GetCenter ( ) const

Get the box center.

Returns

The center position of the box

10.14.3.2 math::Vector3 gazebo::math::Box::GetSize ( ) const

Get the size of the box.

Returns

Size of the box

10.14.3.3 double gazebo::math::Box::GetXLength ( ) const

Get the length along the x dimension.

Returns

Double value of the length in the x dimension

10.14.3.4 double gazebo::math::Box::GetYLength ( ) const

Get the length along the y dimension.

Returns

Double value of the length in the y dimension

10.14.3.5 double gazebo::math::Box::GetZLength ( ) const

Get the length along the z dimension.

Returns

Double value of the length in the z dimension

10.14.3.6 void gazebo::math::Box::Merge (const Box & *_box*)

Merge a box with this box.

Parameters

in	*_box*	**Box** (p. 222) to add to this box

10.14.3.7 Box gazebo::math::Box::operator+ (const Box & *_b*) const

Addition operator.

result = this + _b

Parameters

in	*_b*	**Box** (p. 222) to add

Returns

The new box

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

Addition set operator.

this = this + _b

Parameters

in	*_b*	**Box** (p. 222) to add

Returns

This new box

10.14.3.9 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

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

Assignment operator.

Set this box to the parameter

Parameters

in	*_b*	**Box** (p. 222) to copy

Returns

The new box.

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

Equality test operatoer.

Parameters

in	*_b*	**Box** (p. 222) to test

Returns

True if equal

10.14.4 Friends And Related Function Documentation

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

Output operator.

Parameters

in	*_out*	Output stream
in	*_b*	**Box** (p. 222) to output to the stream

Returns

The stream

10.14.5 Member Data Documentation

10.14.5.1 Vector3 gazebo::math::Box::max

Maximum corner of the box.

10.14.5.2 Vector3 gazebo::math::Box::min

Minimum corner of the box.

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

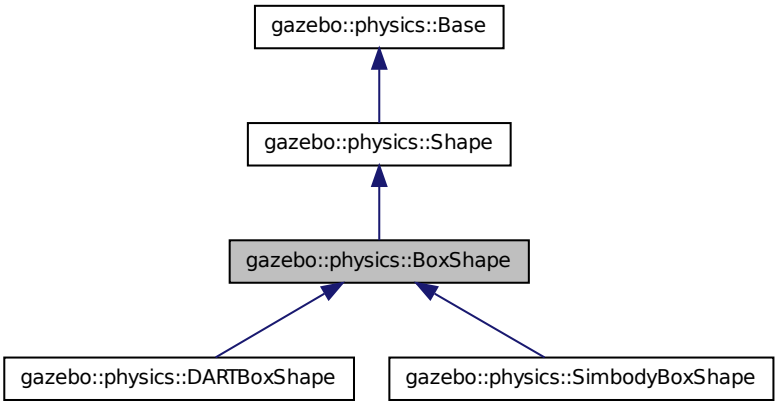
- **Box.hh**

10.15 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 **SetScale** (const **math::Vector3** &_scale)

  Set the scale of the box.

- virtual void **SetSize** (const **math::Vector3** &_size)

  Set the size of the box.

10.15.1 Detailed Description

Box geometry primitive.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 gazebo::physics::BoxShape::BoxShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent **Collision** (p. 295).

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

Destructor.

10.15.3 Member Function Documentation

10.15.3.1 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. 1163).

10.15.3.2 math::Vector3 gazebo::physics::BoxShape::GetSize ( ) const

Get the size of the box.

Returns

The size of each side of the box.

10.15.3.3 virtual void gazebo::physics::BoxShape::Init ( ) [virtual]

Initialize the box.

Implements **gazebo::physics::Shape** (p. 1164).

10.15.3.4 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. 1164).

10.15.3.5 virtual void gazebo::physics::BoxShape::SetScale (const math::Vector3 & *_scale*) [virtual]

Set the scale of the box.

Parameters

in	*_scale*	Scale of the box.

Implements **gazebo::physics::Shape** (p. 1165).

10.15.3.6 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.

Reimplemented in **gazebo::physics::DARTBoxShape** (p. 384), and **gazebo::physics::SimbodyBoxShape** (p. 1174).

Referenced by gazebo::physics::SimbodyBoxShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

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

- **BoxShape.hh**

10.16 gazebo::common::FileLogger::Buffer Class Reference

String buffer for the file logger.

`#include <Console.hh>`

Public Member Functions

- **Buffer** (const std::string &_filename)

  Constructor.
- virtual ∼**Buffer** ()

  Destructor.
- virtual int **sync** ()

  Sync the stream (output the string buffer contents).

Public Attributes

- std::ofstream ∗ **stream**

  Stream to output information into.

10.16.1 Detailed Description

String buffer for the file logger.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 gazebo::common::FileLogger::Buffer::Buffer (const std::string & *_filename*)

Constructor.

Parameters

in	*_filename*	Filename to write into.

10.16.2.2 virtual gazebo::common::FileLogger::Buffer::~Buffer ( ) [virtual]

Destructor.

10.16.3 Member Function Documentation

10.16.3.1 virtual int gazebo::common::FileLogger::Buffer::sync ( ) [virtual]

Sync the stream (output the string buffer contents).

Returns

Return 0 on success.

10.16.4 Member Data Documentation

10.16.4.1 std::ofstream* gazebo::common::FileLogger::Buffer::stream

Stream to output information into.

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

- **Console.hh**

10.17 gazebo::common::Logger::Buffer Class Reference

String buffer for the base logger.

`#include <Console.hh>`

Public Member Functions

- **Buffer** (**LogType** _type, int _color)

Constructor.

- virtual ~**Buffer** ()

  Destructor.

- virtual int **sync** ()

  Sync the stream (output the string buffer contents).

Public Attributes

- int **color**

  ANSI color code using Select Graphic Rendition parameters (SGR).

- **LogType type**

  Destination type for the messages.

10.17.1 Detailed Description

String buffer for the base logger.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 gazebo::common::Logger::Buffer::Buffer (LogType *_type,* int *_color*)

Constructor.

Parameters

in	*_type*	Output destination type (STDOUT, or STDERR)
in	*_color*	**Color** (p. 312) of the output stream.

10.17.2.2 virtual gazebo::common::Logger::Buffer::~Buffer ( ) [virtual]

Destructor.

10.17.3 Member Function Documentation

10.17.3.1 virtual int gazebo::common::Logger::Buffer::sync ( ) [virtual]

Sync the stream (output the string buffer contents).

Returns

Return 0 on success.

10.17.4 Member Data Documentation

10.17.4.1 int gazebo::common::Logger::Buffer::color

ANSI color code using Select Graphic Rendition parameters (SGR).

See `http://en.wikipedia.org/wiki/ANSI_escape_code#Colors`

10.17.4.2 LogType gazebo::common::Logger::Buffer::type

Destination type for the messages.

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

- **Console.hh**

10.18 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.

10.18.1 Detailed Description

Handles loading BVH animation files.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 gazebo::common::BVHLoader::BVHLoader ( )

Constructor.

10.18.2.2 gazebo::common::BVHLoader::~BVHLoader ( )

Desutrctor.

10.18.3 Member Function Documentation

10.18.3.1 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 pointer to a new **Skeleton** (p. 1269)

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

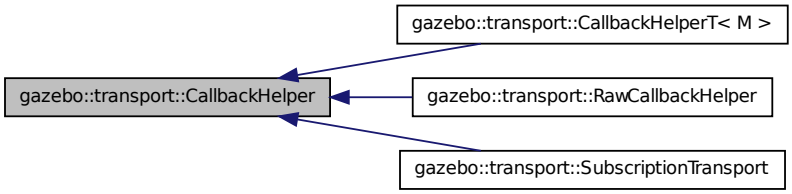
- **BVHLoader.hh**

10.19 gazebo::transport::CallbackHelper Class Reference

A helper class to handle callbacks when messages arrive.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::CallbackHelper:



Public Member Functions

- **CallbackHelper** (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, boost::function< void(uint32_t)> _cb, uint32_t _id)=0

  Process new incoming data.
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)=0

  Process new incoming message.
- virtual bool **IsLocal** () const =0

  Is the callback local?
- void **SetLatching** (bool _latch)

  Set whether this callback is latching.

Protected Attributes

- bool **latching**

  True means that the callback helper will get the last published message on the topic.

10.19.1 Detailed Description

A helper class to handle callbacks when messages arrive.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 gazebo::transport::CallbackHelper::CallbackHelper (bool *_latching* = false)

Constructor.

Parameters

in	*_latching*	Set to true to make the callback helper latching.

10.19.2.2 virtual gazebo::transport::CallbackHelper::~CallbackHelper ( ) [virtual]

Destructor.

10.19.3 Member Function Documentation

10.19.3.1 unsigned int gazebo::transport::CallbackHelper::GetId ( ) const

Get the unique ID of this callback.

Returns

The unique ID of this callback.

10.19.3.2 bool gazebo::transport::CallbackHelper::GetLatching ( ) const

Is the callback latching?

Returns

true if the callback is latching, false otherwise

10.19.3.3 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. 1050), and **gazebo::transport::CallbackHelperT< M >** (p. 241).

10.19.3.4 virtual bool gazebo::transport::CallbackHelper::HandleData (const std::string & *_newdata*, boost::function< void(uint32_t)> *_cb*, uint32_t *_id*) [pure virtual]

Process new incoming data.

Parameters

in	*_newdata*	Incoming data to be processed

Returns

true if successfully processed; false otherwise

Parameters

in	*_cb*	If non-null, callback to be invoked which signals that transmission is complete.
in	*_id*	ID associated with the message data.

Implemented in **gazebo::transport::RawCallbackHelper** (p. 1050), **gazebo::transport::CallbackHelperT< M >** (p. 241), and **gazebo::transport::SubscriptionTransport** (p. 1349).

10.19.3.5 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. 1051), **gazebo::transport::CallbackHelperT< M >** (p. 242), and **gazebo::transport::SubscriptionTransport** (p. 1349).

10.19.3.6 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. 1051), **gazebo::transport::CallbackHelperT< M >** (p. 242), and **gazebo::transport::SubscriptionTransport** (p. 1350).

10.19.3.7 void gazebo::transport::CallbackHelper::SetLatching (bool *_latch*)

Set whether this callback is latching.

This function should only be used by the Transport library.

Parameters

in	*_latch*	False to turn off latching.

10.19.4 Member Data Documentation

10.19.4.1 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:

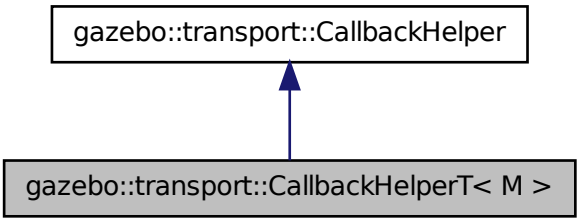
- **CallbackHelper.hh**

10.20 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 boost::shared_ptr< 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, boost::function< void(uint32_t)> _cb, uint32_t _id)

  Process new incoming data.
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)

  Process new incoming message.
- virtual bool **IsLocal** () const

  Is the callback local?

10.20.1 Detailed Description

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

Callback helper Template.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 template<class M > gazebo::transport::CallbackHelperT< M >::CallbackHelperT (const boost::function< void(const boost::shared_ptr< M const > &)> & _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.

10.20.3 Member Function Documentation

10.20.3.1 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. 237).

References gzthrow, and NULL.

10.20.3.2 template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::HandleData (const std::string & _newdata, boost::function< void(uint32_t)> _cb, uint32_t _id) [inline, virtual]

Process new incoming data.

Parameters

in	*_newdata*	Incoming data to be processed

Returns

true if successfully processed; false otherwise

Parameters

in	*_cb*	If non-null, callback to be invoked which signals that transmission is complete.
in	*_id*	ID associated with the message data.

Implements **gazebo::transport::CallbackHelper** (p. 238).

10.20.3.3 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. 238).

10.20.3.4 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. 239).

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

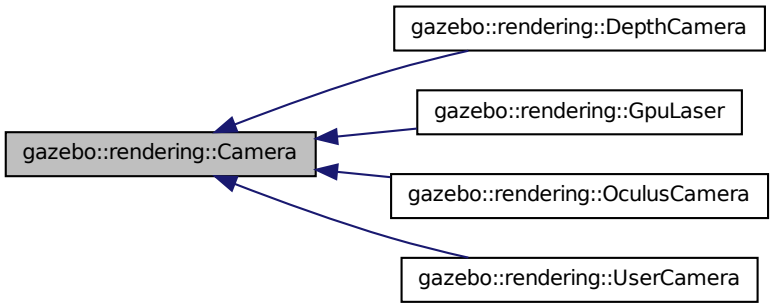
- **CallbackHelper.hh**

10.21 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.
- void **AttachToVisual** (uint32_t _id, 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** () const

  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.
- bool **GetCaptureData** () const

  Return the value of this->captureData.
- **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 unscoped name.
- double **GetNearClip** ()

  Get the near clip distance.
- Ogre::Camera * **GetOgreCamera** () const

  Get a pointer to the ogre camera.
- double **GetRenderRate** () const

Get the render Hz rate.

- Ogre::Texture * **GetRenderTexture** () const

  Get the render texture.

- **math::Vector3 GetRight** ()

  Get the viewport right vector.

- **ScenePtr GetScene** () const

  Get the scene this camera is in.

- Ogre::SceneNode * **GetSceneNode** () const

  Get the camera's scene node.

- std::string **GetScopedName** () const

  Get the camera's scoped name (scene_name::camera_name)

- 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** () const

  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** () const

  Get the world pose.

- **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 **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** (bool _force=false)

  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** (uint32_t _id, 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.
- void **ReadPixelBuffer** ()

  Read image data from pixel buffer.
- virtual void **RenderImpl** ()

  Implementation of the render call.
- bool **TrackVisualImpl** (const std::string &_visualName)

  *Implementation of the **Camera::TrackVisual*** (p. *271) 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.
- **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. 1097) node that controls camera position and orientation.*
- std::string **scopedName**

  Scene *(p. 1097) scoped name of the camera.*
- std::string **scopedUniqueName**

Scene *(p. 1097) scoped name of the camera with a unique ID.*

- std::string **screenshotPath**

  Path to saved screenshots.

- sdf::ElementPtr **sdf**

  Camera *(p. 242)'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.

10.21.1 Detailed Description

Basic camera sensor.

This is the base class for all cameras.

10.21.2 Constructor & Destructor Documentation

10.21.2.1 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. 1097) that will contain the camera
`in`	*_auto-Render*	Almost everyone should leave this as true.

10.21.2.2 virtual gazebo::rendering::Camera::∼Camera ( ) `[virtual]`

Destructor.

10.21.3 Member Function Documentation

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

Internal function used to indicate that an animation has completed.

Reimplemented in **gazebo::rendering::UserCamera** (p. 1410).

10.21.3.2 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	*_inheritOrientation*	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

10.21.3.3 void gazebo::rendering::Camera::AttachToVisual (uint32_t *_id*, bool *_inheritOrientation*, double *_minDist* = 0.0, double *_maxDist* = 0.0)

Attach the camera to a scene node.

Parameters

in	*_id*	ID of the visual to attach the camera to
in	*_inheritOrientation*	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

10.21.3.4 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	*_visual-Name*	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

10.21.3.5 virtual bool gazebo::rendering::Camera::AttachToVisualImpl (uint32_t *_id*, bool *_inheritOrientation*, double *_minDist* = 0, double *_maxDist* = 0) [protected, virtual]

Attach the camera to a scene node.

Parameters

in	*_id*	ID 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

10.21.3.6 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. 1410), and **gazebo::rendering::OculusCamera** (p. 943).

10.21.3.7 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 pointer to the connection. This must be kept in scope.

10.21.3.8 void gazebo::rendering::Camera::CreateRenderTexture (const std::string & *_textureName*)

Set the render target.

Parameters

in	*_textureName*	Name of the new render texture

10.21.3.9 void gazebo::rendering::Camera::DisconnectNewImageFrame (event::ConnectionPtr & *_c*) [inline]

Disconnect from an image frame.

Parameters

in	*_c*	The connection to disconnect

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

Enable or disable saving.

Parameters

in	*_enable*	Set to True to enable saving of frames

10.21.3.11 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. 583), **gazebo::rendering::OculusCamera** (p. 944), **gazebo::rendering::DepthCamera** (p. 478), and **gazebo::rendering::UserCamera** (p. 1411).

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

Get the apect ratio.

Returns

The aspect ratio (width / height) in pixels

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

Get the average FPS.

Returns

The average frames per second

Reimplemented in **gazebo::rendering::UserCamera** (p. 1411), and **gazebo::rendering::OculusCamera** (p. 944).

10.21.3.14 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

10.21.3.15 bool gazebo::rendering::Camera::GetCaptureData ( ) const

Return the value of this->captureData.

Returns

True if the camera is set to capture data.

10.21.3.16 math::Vector3 gazebo::rendering::Camera::GetDirection ( ) const

Get the camera's direction vector.

Returns

Direction the camera is facing

10.21.3.17 double gazebo::rendering::Camera::GetFarClip ( )

Get the far clip distance.

Returns

Far clip distance

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

Get the next frame filename based on SDF parameters.

Returns

The frame's filename

10.21.3.19 math::Angle gazebo::rendering::Camera::GetHFOV ( ) const

Get the camera FOV (horizontal)

Returns

The horizontal field of view

10.21.3.20 size_t gazebo::rendering::Camera::GetImageByteSize ( ) const

Get the image size in bytes.

Returns

Size in bytes

10.21.3.21 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

10.21.3.22 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.

10.21.3.23 unsigned int gazebo::rendering::Camera::GetImageDepth ( ) const

Get the depth of the image.

Returns

Depth of the image

10.21.3.24 std::string gazebo::rendering::Camera::GetImageFormat ( ) const

Get the string representation of the image format.

Returns

String representation of the image format.

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

Get the height of the image.

Returns

Image height

Reimplemented in **gazebo::rendering::UserCamera** (p. 1412), and **gazebo::rendering::OculusCamera** (p. 944).

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

Get the width of the image.

Returns

Image width

Reimplemented in **gazebo::rendering::UserCamera** (p. 1412), and **gazebo::rendering::OculusCamera** (p. 944).

10.21.3.27 bool gazebo::rendering::Camera::GetInitialized ( ) const

Return true if the camera has been initialized.

Returns

True if initialized was successful

10.21.3.28 common::Time gazebo::rendering::Camera::GetLastRenderWallTime ( )

Get the last time the camera was rendered.

Returns

Time the camera was last rendered

10.21.3.29 std::string gazebo::rendering::Camera::GetName ( ) const

Get the camera's unscoped name.

Returns

The name of the camera

10.21.3.30 double gazebo::rendering::Camera::GetNearClip ( )

Get the near clip distance.

Returns

Near clip distance

10.21.3.31 Ogre::Camera* gazebo::rendering::Camera::GetOgreCamera ( ) const

Get a pointer to the ogre camera.

Returns

Pointer to the OGRE camera

10.21.3.32 double gazebo::rendering::Camera::GetRenderRate ( ) const

Get the render Hz rate.

Returns

The Hz rate

10.21.3.33 Ogre::Texture* gazebo::rendering::Camera::GetRenderTexture ( ) const

Get the render texture.

Returns

Pointer to the render texture

10.21.3.34 math::Vector3 gazebo::rendering::Camera::GetRight ( )

Get the viewport right vector.

Returns

The viewport right vector

10.21.3.35 ScenePtr gazebo::rendering::Camera::GetScene ( ) const

Get the scene this camera is in.

Returns

Pointer to scene containing this camera

10.21.3.36 Ogre::SceneNode* gazebo::rendering::Camera::GetSceneNode ( ) const

Get the camera's scene node.

Returns

The scene node the camera is attached to

10.21.3.37 std::string gazebo::rendering::Camera::GetScopedName ( ) const

Get the camera's scoped name (scene_name::camera_name)

Returns

The name of the camera

10.21.3.38 std::string gazebo::rendering::Camera::GetScreenshotPath ( ) const

Get the path to saved screenshots.

Returns

Path to saved screenshots.

10.21.3.39 unsigned int gazebo::rendering::Camera::GetTextureHeight ( ) const

Get the height of the off-screen render texture.

Returns

Render texture height

10.21.3.40 unsigned int gazebo::rendering::Camera::GetTextureWidth ( ) const

Get the width of the off-screen render texture.

Returns

Render texture width

10.21.3.41 virtual unsigned int gazebo::rendering::Camera::GetTriangleCount ( ) const [inline, virtual]

Get the triangle count.

Returns

The current triangle count

Reimplemented in **gazebo::rendering::UserCamera** (p. 1412), and **gazebo::rendering::OculusCamera** (p. 945).

10.21.3.42 math::Vector3 gazebo::rendering::Camera::GetUp ( )

Get the viewport up vector.

Returns

The viewport up vector

10.21.3.43 math::Angle gazebo::rendering::Camera::GetVFOV ( ) const

Get the camera FOV (vertical)

Returns

The vertical field of view

10.21.3.44 Ogre::Viewport* gazebo::rendering::Camera::GetViewport ( ) const

Get a pointer to the Ogre::Viewport.

Returns

Pointer to the Ogre::Viewport

10.21.3.45 unsigned int gazebo::rendering::Camera::GetViewportHeight ( ) const

Get the viewport height in pixels.

Returns

The viewport height

10.21.3.46 unsigned int gazebo::rendering::Camera::GetViewportWidth ( ) const

Get the viewport width in pixels.

Returns

The viewport width

10.21.3.47 unsigned int gazebo::rendering::Camera::GetWindowId ( ) const

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

Returns

The ID of the window.

10.21.3.48 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

10.21.3.49 math::Pose gazebo::rendering::Camera::GetWorldPose ( ) const

Get the world pose.

Returns

The pose of the camera in the world coordinate frame.

10.21.3.50 math::Vector3 gazebo::rendering::Camera::GetWorldPosition ( ) const

Get the camera position in the world.

Returns

The world position of the camera

10.21.3.51 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. 1029)

10.21.3.52 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.

10.21.3.53 virtual void gazebo::rendering::Camera::Init ( ) [virtual]

Initialize the camera.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 586), **gazebo::rendering::OculusCamera** (p. 945), **gazebo::rendering::DepthCamera** (p. 478), and **gazebo::rendering::UserCamera** (p. 1414).

10.21.3.54 bool gazebo::rendering::Camera::IsAnimating ( ) const

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

10.21.3.55 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

10.21.3.56 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

10.21.3.57 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::GpuLaser** (p. 586), **gazebo::rendering::OculusCamera** (p. 945), **gazebo::rendering::DepthCamera** (p. 479), and **gazebo::rendering::UserCamera** (p. 1414).

10.21.3.58 virtual void gazebo::rendering::Camera::Load ( ) [virtual]

Load the camera with default parmeters.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 586), **gazebo::rendering::OculusCamera** (p. 945), **gazebo::rendering::DepthCamera** (p. 479), and **gazebo::rendering::UserCamera** (p. 1414).

10.21.3.59 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. 265)

Parameters

in	*_pose*	End position of the camera
in	*_time*	Duration of the camera's movement

Reimplemented in **gazebo::rendering::UserCamera** (p. 1414), and **gazebo::rendering::OculusCamera** (p. 945).

10.21.3.60 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. 264)

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

10.21.3.61 virtual void gazebo::rendering::Camera::PostRender ( ) [virtual]

Post render.

Called afer the render signal.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 587), **gazebo::rendering::OculusCamera** (p. 946), **gazebo::rendering::DepthCamera** (p. 479), and **gazebo::rendering::UserCamera** (p. 1415).

10.21.3.62 void gazebo::rendering::Camera::ReadPixelBuffer ( ) [protected]

Read image data from pixel buffer.

10.21.3.63 void gazebo::rendering::Camera::Render (bool *_force* = `false`)

Render the camera.

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

Parameters

in	*_force*	Force camera to render. Ignore camera update rate.

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

Implementation of the render call.

10.21.3.65 void gazebo::rendering::Camera::RotatePitch (math::Angle *_angle*)

Rotate the camera around the pitch axis.

Parameters

in	*_angle*	Pitch amount

10.21.3.66 void gazebo::rendering::Camera::RotateYaw (math::Angle *_angle*)

Rotate the camera around the yaw axis.

Parameters

in	*_angle*	Rotation amount

10.21.3.67 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

10.21.3.68 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

10.21.3.69 void gazebo::rendering::Camera::SetAspectRatio (float _ratio)

Set the aspect ratio.

Parameters

in	*_ratio*	The aspect ratio (width / height) in pixels

10.21.3.70 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.

10.21.3.71 void gazebo::rendering::Camera::SetCaptureDataOnce ( )

Capture data once and save to disk.

10.21.3.72 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

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

Set the camera FOV (horizontal)

Parameters

in	*_radians*	Horizontal field of view

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

Set the image height.

Parameters

in	*_h*	Image height

10.21.3.75 void gazebo::rendering::Camera::SetImageSize (unsigned int *_w,* unsigned int *_h*)

Set the image size.

Parameters

in	*_w*	Image width
in	*_h*	Image height

10.21.3.76 void gazebo::rendering::Camera::SetImageWidth (unsigned int _w)

Set the image height.

Parameters

in	*_w*	Image width

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

Set the camera's name.

Parameters

in	*_name*	New name for the camera

10.21.3.78 void gazebo::rendering::Camera::SetRenderRate (double _hz)

Set the render Hz rate.

Parameters

in	*_hz*	The Hz rate

10.21.3.79 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. 1416), and **gazebo::rendering::OculusCamera** (p. 947).

10.21.3.80 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

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

Set the scene this camera is viewing.

Parameters

in	*_scene*	Pointer to the scene

10.21.3.82 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

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

10.21.3.84 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. 995) of the camera

Reimplemented in **gazebo::rendering::UserCamera** (p. 1417).

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

Set the world position.

Parameters

in	*_pos*	The new position of the camera

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

Set the world orientation.

Parameters

in	*_quat*	The new orientation of the camera

10.21.3.87 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

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

Toggle whether to view the world in wireframe.

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

Set the camera to track a scene node.

Parameters

in	*_visual-Name*	Name of the visual to track

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

Implementation of the **Camera::TrackVisual** (p. 271) call.

Parameters

in	*_visual-Name*	Name of the visual to track

Returns

True if able to track the visual

10.21.3.91 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. 1417), and **gazebo::rendering::OculusCamera** (p. 947).

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

Translate the camera.

Parameters

in	*_direction*	The translation vector

10.21.3.93 virtual void gazebo::rendering::Camera::Update ( ) [virtual]

Reimplemented in **gazebo::rendering::OculusCamera** (p. 948), and **gazebo::rendering::UserCamera** (p. 1418).

10.21.4 Member Data Documentation

10.21.4.1 Ogre::AnimationState* gazebo::rendering::Camera::animState [protected]

Animation state, used to animate the camera.

10.21.4.2 unsigned char* gazebo::rendering::Camera::bayerFrameBuffer [protected]

Buffer for a bayer image frame.

10.21.4.3 Ogre::Camera* gazebo::rendering::Camera::camera [protected]

The OGRE camera.

10.21.4.4 bool gazebo::rendering::Camera::captureData [protected]

True to capture frames into an image buffer.

10.21.4.5 bool gazebo::rendering::Camera::captureDataOnce [protected]

True to capture a frame once and save to disk.

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

The camera's event connections.

10.21.4.7 int gazebo::rendering::Camera::imageFormat [protected]

Format for saving images.

10.21.4.8 int gazebo::rendering::Camera::imageHeight [protected]

Save image height.

10.21.4.9 int gazebo::rendering::Camera::imageWidth [protected]

Save image width.

10.21.4.10 bool gazebo::rendering::Camera::initialized [protected]

True if initialized.

10.21.4.11 common::Time gazebo::rendering::Camera::lastRenderWallTime [protected]

Time the last frame was rendered.

10.21.4.12 std::string gazebo::rendering::Camera::name [protected]

Name of the camera.

10.21.4.13 bool gazebo::rendering::Camera::newData [protected]

True if new data is available.

10.21.4.14 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.

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

User callback for when an animation completes.

10.21.4.16 common::Time gazebo::rendering::Camera::prevAnimTime [protected]

Previous time the camera animation was updated.

10.21.4.17 Ogre::RenderTarget* gazebo::rendering::Camera::renderTarget [protected]

Target that renders frames.

10.21.4.18 Ogre::Texture* gazebo::rendering::Camera::renderTexture [protected]

Texture that receives results from rendering.

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

List of requests.

10.21.4.20 unsigned int gazebo::rendering::Camera::saveCount [protected]

Number of saved frames.

10.21.4.21 unsigned char* gazebo::rendering::Camera::saveFrameBuffer [protected]

10.21.4.22 ScenePtr gazebo::rendering::Camera::scene [protected]

Pointer to the scene.

10.21.4.23 Ogre::SceneNode* gazebo::rendering::Camera::sceneNode [protected]

Scene (p. 1097) node that controls camera position and orientation.

10.21.4.24 std::string gazebo::rendering::Camera::scopedName [protected]

Scene (p. 1097) scoped name of the camera.

10.21.4.25 std::string gazebo::rendering::Camera::scopedUniqueName [protected]

Scene (p. 1097) scoped name of the camera with a unique ID.

10.21.4.26 std::string gazebo::rendering::Camera::screenshotPath [protected]

Path to saved screenshots.

10.21.4.27 sdf::ElementPtr gazebo::rendering::Camera::sdf [protected]

Camera (p. 242)'s SDF values.

10.21.4.28 unsigned int gazebo::rendering::Camera::textureHeight [protected]

Height of the render texture.

10.21.4.29 unsigned int gazebo::rendering::Camera::textureWidth [protected]

Width of the render texture.

10.21.4.30 Ogre::Viewport* gazebo::rendering::Camera::viewport [protected]

Viewport the ogre camera uses.

10.21.4.31 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**

10.22 gazebo::rendering::CameraPrivate Class Reference

Private data for the **Camera** (p. 242) class.

`#include <CameraPrivate.hh>`

Public Types

- typedef std::list < boost::shared_ptr < msgs::CameraCmd const > > **CameraCmdMsgs_L**

Public Attributes

- **transport::SubscriberPtr cmdSub**

  Subscribe to camera command topic.
- **CameraCmdMsgs_L commandMsgs**

  List of camera cmd messages.
- Ogre::CompositorInstance * **dlGBufferInstance**

  Deferred lighting geometry buffer.

- Ogre::CompositorInstance * **dlMergeInstance**

  Deferred lighting merge compositor.

- Ogre::CompositorInstance * **dsGBufferInstance**

  Deferred shading geometry buffer.

- Ogre::CompositorInstance * **dsMergeInstance**

  Deferred shading merge compositor.

- std::deque< std::pair < **math::Pose**, double > > **moveToPositionQueue**

  Queue of move positions.

- **transport::NodePtr node**

  Communication Node.

- boost::mutex **receiveMutex**

  Mutex to lock the various message buffers.

- **common::Time renderPeriod**

  Render period.

- Ogre::CompositorInstance * **ssaoInstance**

  Screen space ambient occlusion compositor.

- **VisualPtr trackedVisual**

  Visual (p. *1477) that the camera is tracking.*

- **common::PID trackVisualPID**

  Position PID used to track a visual smoothly.

- **common::PID trackVisualPitchPID**

  Pitch PID used to track a visual smoothly.

- **common::PID trackVisualYawPID**

  Yaw PID used to track a visual smoothly.

Static Public Attributes

- static unsigned int **cameraCounter**

  Counter used to create unique camera names.

10.22.1 Detailed Description

Private data for the **Camera** (p. 242) class.

10.22.2 Member Typedef Documentation

10.22.2.1 typedef std::list<boost::shared_ptr<msgs::CameraCmd const> > gazebo::rendering::CameraPrivate::CameraCmdMsgs_L

10.22.3 Member Data Documentation

10.22.3.1 unsigned int gazebo::rendering::CameraPrivate::cameraCounter [static]

Counter used to create unique camera names.

10.22.3.2 transport::SubscriberPtr gazebo::rendering::CameraPrivate::cmdSub

Subscribe to camera command topic.

10.22.3.3 CameraCmdMsgs_L gazebo::rendering::CameraPrivate::commandMsgs

List of camera cmd messages.

10.22.3.4 Ogre::CompositorInstance* gazebo::rendering::CameraPrivate::dlGBufferInstance

Deferred lighting geometry buffer.

10.22.3.5 Ogre::CompositorInstance* gazebo::rendering::CameraPrivate::dlMergeInstance

Deferred lighting merge compositor.

10.22.3.6 Ogre::CompositorInstance* gazebo::rendering::CameraPrivate::dsGBufferInstance

Deferred shading geometry buffer.

10.22.3.7 Ogre::CompositorInstance* gazebo::rendering::CameraPrivate::dsMergeInstance

Deferred shading merge compositor.

10.22.3.8 std::deque<std::pair<math::Pose, double> > gazebo::rendering::CameraPrivate::moveToPositionQueue

Queue of move positions.

10.22.3.9 transport::NodePtr gazebo::rendering::CameraPrivate::node

Communication Node.

10.22.3.10 boost::mutex gazebo::rendering::CameraPrivate::receiveMutex

Mutex to lock the various message buffers.

10.22.3.11 common::Time gazebo::rendering::CameraPrivate::renderPeriod

Render period.

10.22.3.12 Ogre::CompositorInstance* gazebo::rendering::CameraPrivate::ssaoInstance

Screen space ambient occlusion compositor.

10.22.3.13 VisualPtr gazebo::rendering::CameraPrivate::trackedVisual

Visual (p. 1477) that the camera is tracking.

10.22.3.14 common::PID gazebo::rendering::CameraPrivate::trackVisualPID

Position PID used to track a visual smoothly.

10.22.3.15 common::PID gazebo::rendering::CameraPrivate::trackVisualPitchPID

Pitch PID used to track a visual smoothly.

10.22.3.16 common::PID gazebo::rendering::CameraPrivate::trackVisualYawPID

Yaw PID used to track a visual smoothly.

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

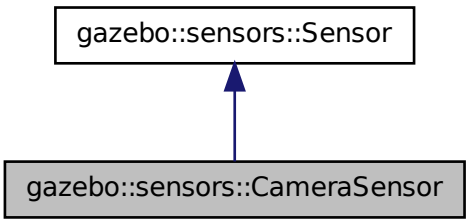
- **CameraPrivate.hh**

10.23 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. *242*).
- 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.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the camera.
- virtual bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

10.23.1 Detailed Description

Basic camera sensor.

This sensor is used for simulating standard monocular cameras

10.23.2 Constructor & Destructor Documentation

10.23.2.1 gazebo::sensors::CameraSensor::CameraSensor ( )

Constructor.

10.23.2.2 virtual gazebo::sensors::CameraSensor::~CameraSensor ( ) [virtual]

Destructor.

10.23.3 Member Function Documentation

10.23.3.1 virtual void gazebo::sensors::CameraSensor::Fini ( ) [protected, virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

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

Returns a pointer to the **rendering::Camera** (p. 242).

Returns

The Pointer to the camera sensor.

10.23.3.3 const unsigned char* gazebo::sensors::CameraSensor::GetImageData ( )

Gets the raw image data from the sensor.

Returns

The pointer to the image data array.

10.23.3.4 unsigned int gazebo::sensors::CameraSensor::GetImageHeight ( ) const

Gets the height of the image in pixels.

Returns

The image height in pixels.

10.23.3.5 unsigned int gazebo::sensors::CameraSensor::GetImageWidth ( ) const

Gets the width of the image in pixels.

Returns

The image width in pixels.

10.23.3.6 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. 1138).

10.23.3.7 virtual void gazebo::sensors::CameraSensor::Init ( ) [virtual]

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.23.3.8 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. 1139).

10.23.3.9 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. 1130) parameters
in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 1140).

10.23.3.10 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. 1140).

10.23.3.11 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.

10.23.3.12 virtual bool gazebo::sensors::CameraSensor::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

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

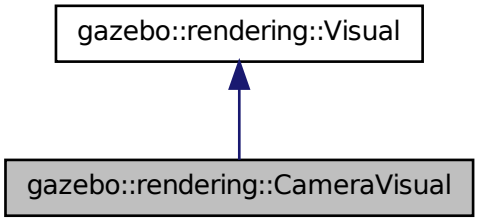
- **CameraSensor.hh**

10.24 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. 1477).*

10.24.1 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.

10.24.2 Constructor & Destructor Documentation

10.24.2.1 gazebo::rendering::CameraVisual::CameraVisual (const std::string & *_name*, VisualPtr *_vis*)

Constructor.

Parameters

in	*_name*	Name of the **Visual** (p. 1477)
in	*_vis*	Pointer to the parent **Visual** (p. 1477)

10.24.2.2 virtual gazebo::rendering::CameraVisual::~CameraVisual ( ) [virtual]

Destructor.

10.24.3 Member Function Documentation

10.24.3.1 void gazebo::rendering::CameraVisual::Load (unsigned int *_width,* unsigned int *_height*)

Load the **Visual** (p. 1477).

Parameters

in	*_width*	Width of the **Camera** (p. 242) image
in	*_height*	Height of the **Camera** (p. 242) image

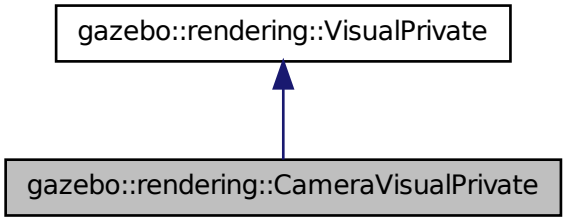
The documentation for this class was generated from the following file:

- **CameraVisual.hh**

10.25 gazebo::rendering::CameraVisualPrivate Class Reference

`#include <CameraVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::CameraVisualPrivate:



Public Attributes

- **CameraPtr camera**

  Pointer to the camera.
- std::vector< **event::ConnectionPtr** > **connections**

  Event connections.

10.25.1 Member Data Documentation

10.25.1.1 CameraPtr gazebo::rendering::CameraVisualPrivate::camera

Pointer to the camera.

10.25.1.2 std::vector<event::ConnectionPtr> gazebo::rendering::CameraVisualPrivate::connections

Event connections.

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

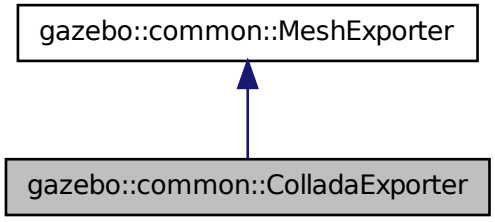
- **CameraVisualPrivate.hh**

10.26 gazebo::common::ColladaExporter Class Reference

Class used to export Collada mesh files.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::ColladaExporter:



Public Types

- enum **GeometryType** { **POSITION**, **NORMAL**, **UVMAP** }

  Geometry types.

Public Member Functions

- **ColladaExporter** ()

  Constructor.
- virtual ~**ColladaExporter** ()

  Destructor.
- virtual void **Export** (const **Mesh** *_mesh, const std::string &_filename, bool _exportTextures)

  Export a mesh to a file.

10.26.1 Detailed Description

Class used to export Collada mesh files.

10.26.2 Member Enumeration Documentation

10.26.2.1 enum gazebo::common::ColladaExporter::GeometryType

Geometry types.

Enumerator:

POSITION

NORMAL

UVMAP

10.26.3 Constructor & Destructor Documentation

10.26.3.1 gazebo::common::ColladaExporter::ColladaExporter ( )

Constructor.

10.26.3.2 virtual gazebo::common::ColladaExporter::~ColladaExporter ( ) [virtual]

Destructor.

10.26.4 Member Function Documentation

10.26.4.1 virtual void gazebo::common::ColladaExporter::Export (const Mesh * *_mesh*, const std::string & *_filename*, bool *_exportTextures*) [virtual]

Export a mesh to a file.

Parameters

in	*_mesh*	Pointer to the mesh to be exported
in	*_filename*	Exported file's path and name
in	*_exportTextures*	True to export texture images to '../materials/textures' folder

Implements **gazebo::common::MeshExporter** (p. 832).

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

- **ColladaExporter.hh**

10.27 gazebo::common::ColladaExporterPrivate Class Reference

Private data for the **ColladaExporter** (p. 287) class.

`#include <ColladaExporterPrivate.hh>`

Public Attributes

- bool **exportTextures**

  True to export texture images to '../materials/textures' folder.
- std::string **filename**

  File name.
- unsigned int **materialCount**

  Material *(p. 794) count.*
- const **Mesh** * **mesh**

  Gazebo mesh.
- std::string **path**

  File path.
- unsigned int **subMeshCount**

  SubMesh *(p. 1330) count.*

10.27.1 Detailed Description

Private data for the **ColladaExporter** (p. 287) class.

10.27.2 Member Data Documentation

10.27.2.1 bool gazebo::common::ColladaExporterPrivate::exportTextures

True to export texture images to '../materials/textures' folder.

10.27.2.2 std::string gazebo::common::ColladaExporterPrivate::filename

File name.

10.27.2.3 unsigned int gazebo::common::ColladaExporterPrivate::materialCount

Material (p. 794) count.

10.27.2.4 const Mesh* gazebo::common::ColladaExporterPrivate::mesh

Gazebo mesh.

10.27.2.5 std::string gazebo::common::ColladaExporterPrivate::path

File path.

10.27.2.6 unsigned int gazebo::common::ColladaExporterPrivate::subMeshCount

SubMesh (p. 1330) count.

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

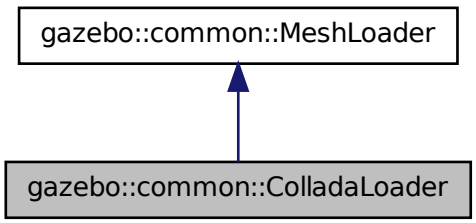
- **ColladaExporterPrivate.hh**

10.28 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.

10.28.1 Detailed Description

Class used to load Collada mesh files.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 gazebo::common::ColladaLoader::ColladaLoader ( )

Constructor.

10.28.2.2 virtual gazebo::common::ColladaLoader::∼ColladaLoader ( ) [virtual]

Destructor.

10.28.3 Member Function Documentation

10.28.3.1 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. 821)

Implements **gazebo::common::MeshLoader** (p. 834).

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

- **ColladaLoader.hh**

10.29 gazebo::common::ColladaLoaderPrivate Class Reference

Private data for the **ColladaLoader** (p. 291) class.

`#include <ColladaLoaderPrivate.hh>`

Public Attributes

- TiXmlElement * **colladaXml**

  root xml element of COLLADA data
- std::string **currentNodeName**

  Name of the current node.
- std::string **filename**

  COLLADA file name.
- std::map< std::string, **Material** * > **materialIds**

  *Map of collada **Material** (p. 794) ids to Gazebo materials.*
- std::map< std::string, std::string > **materialMap**

  material dictionary indexed by name
- double **meter**

  scaling factor
- std::map< std::string, std::map< unsigned int, unsigned int > > **normalDuplicateMap**

  Map of collada NORMAL ids to a map of duplicate normals.
- std::map< std::string, std::vector< **math::Vector3** > > **normalIds**

  Map of collada NORMAL ids to list of normals.
- std::string **path**

  directory of COLLADA file name
- std::map< std::string, std::map< unsigned int, unsigned int > > **positionDuplicateMap**

  Map of collada POSITION ids to a map of duplicate positions.
- std::map< std::string, std::vector< **math::Vector3** > > **positionIds**

  Map of collada POSITION ids to list of vectors.
- std::map< std::string, std::map< unsigned int, unsigned int > > **texcoordDuplicateMap**

  Map of collada TEXCOORD ids to a map of duplicate texture coordinates.
- std::map< std::string, std::vector< **math::Vector2d** > > **texcoordIds**

  Map of collada TEXCOORD ids to list of texture coordinates.

10.29.1 Detailed Description

Private data for the **ColladaLoader** (p. 291) class.

10.29.2 Member Data Documentation

10.29.2.1 TiXmlElement* gazebo::common::ColladaLoaderPrivate::colladaXml

root xml element of COLLADA data

10.29.2.2 std::string gazebo::common::ColladaLoaderPrivate::currentNodeName

Name of the current node.

10.29.2.3 std::string gazebo::common::ColladaLoaderPrivate::filename

COLLADA file name.

10.29.2.4 std::map<std::string, Material *> gazebo::common::ColladaLoaderPrivate::materialIds

Map of collada **Material** (p. 794) ids to Gazebo materials.

10.29.2.5 std::map<std::string, std::string> gazebo::common::ColladaLoaderPrivate::materialMap

material dictionary indexed by name

10.29.2.6 double gazebo::common::ColladaLoaderPrivate::meter

scaling factor

10.29.2.7 std::map<std::string, std::map<unsigned int, unsigned int> > gazebo::common::ColladaLoaderPrivate::normalDuplicateMap

Map of collada NORMAL ids to a map of duplicate normals.

10.29.2.8 std::map<std::string, std::vector<math::Vector3> > gazebo::common::ColladaLoaderPrivate::normalIds

Map of collada NORMAL ids to list of normals.

10.29.2.9 std::string gazebo::common::ColladaLoaderPrivate::path

directory of COLLADA file name

10.29.2.10 std::map<std::string, std::map<unsigned int, unsigned int> > gazebo::common::ColladaLoaderPrivate::positionDuplicateMap

Map of collada POSITION ids to a map of duplicate positions.

10.29.2.11 std::map<std::string, std::vector<math::Vector3> > gazebo::common::ColladaLoaderPrivate::positionIds

Map of collada POSITION ids to list of vectors.

10.29.2.12 std::map<std::string, std::map<unsigned int, unsigned int> > gazebo::common::ColladaLoaderPrivate::texcoordDuplicateMap

Map of collada TEXCOORD ids to a map of duplicate texture coordinates.

10.29.2.13 std::map<std::string, std::vector<math::Vector2d> > gazebo::common::ColladaLoaderPrivate::texcoordIds

Map of collada TEXCOORD ids to list of texture coordinates.

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

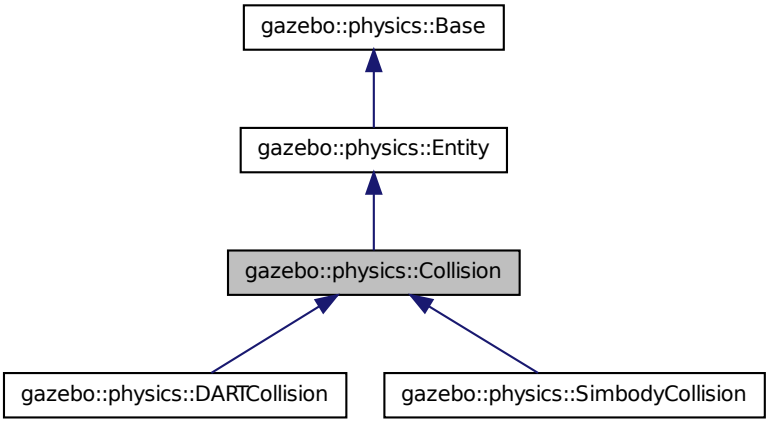
- **ColladaLoaderPrivate.hh**

10.30 gazebo::physics::Collision Class Reference

Base (p. 201) 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 **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.
- float **GetLaserRetro** () const

  Get the laser retro reflectiveness.
- **LinkPtr GetLink** () const

  Get the link this collision belongs to.
- virtual unsigned int **GetMaxContacts** ()

  returns number of contacts allowed for this collision.
- **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 const **math::Pose** & **GetWorldPose** () const

  Get the absolute pose of the entity.
- 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 **SetLaserRetro** (float _retro)

  Set the laser retro reflectiveness.

- virtual void **SetMaxContacts** (unsigned int _maxContacts)

  Number of contacts allowed for this collision.

- void **SetScale** (const **math::Vector3** &_scale)

  Set the scale of the collision.

- void **SetShape** (**ShapePtr** _shape)

  Set the shape for this collision.

- void **SetState** (const **CollisionState** &_state)

  Set the current collision state.

- void **SetWorldPoseDirty** ()

  Indicate that the world pose should be recalculated.

- 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. 1161).*

- **SurfaceParamsPtr surface**

  The surface parameters.

10.30.1 Detailed Description

Base (p. 201) class for all collision entities.

10.30.2 Constructor & Destructor Documentation

10.30.2.1 gazebo::physics::Collision::Collision (LinkPtr *_link*) [explicit]

Constructor.

Parameters

in	*_link*	**Link** (p. 739) that contains this collision object.

10.30.2.2 virtual gazebo::physics::Collision::~Collision ( ) [virtual]

Destructor.

10.30.3 Member Function Documentation

10.30.3.1 void gazebo::physics::Collision::FillMsg (msgs::Collision & *_msg*)

Fill a collision message.

Parameters

out	*_msg*	The message to fill with this collision's data.

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

Finalize the collision.

Reimplemented from **gazebo::physics::Entity** (p. 504).

Reimplemented in **gazebo::physics::DARTCollision** (p. 387).

10.30.3.3 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. 504).

Implemented in **gazebo::physics::DARTCollision** (p. 387), and **gazebo::physics::SimbodyCollision** (p. 1176).

10.30.3.4 float gazebo::physics::Collision::GetLaserRetro ( ) const

Get the laser retro reflectiveness.

Returns

The laser retro value.

10.30.3.5 LinkPtr gazebo::physics::Collision::GetLink ( ) const

Get the link this collision belongs to.

Returns

The parent **Link** (p. 739).

10.30.3.6 virtual unsigned int gazebo::physics::Collision::GetMaxContacts ( ) [virtual]

returns number of contacts allowed for this collision.

This overrides global value (in **PhysicsEngine** (p. 959)) if specified.

Returns

max num contacts allowed for this collision.

10.30.3.7 ModelPtr gazebo::physics::Collision::GetModel ( ) const

Get the model this collision belongs to.

Returns

The parent model.

10.30.3.8 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. 506).

10.30.3.9 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. 506).

10.30.3.10 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. 507).

10.30.3.11 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. 507).

10.30.3.12 ShapePtr gazebo::physics::Collision::GetShape ( ) const

Get the collision shape.

Returns

The collision shape.

10.30.3.13 unsigned int gazebo::physics::Collision::GetShapeType ( )

Get the shape type.

Returns

The shape type.

See also

EntityType (p. 205)

10.30.3.14 CollisionState gazebo::physics::Collision::GetState ( )

Get the collision state.

Returns

The collision state.

10.30.3.15 SurfaceParamsPtr gazebo::physics::Collision::GetSurface ( ) const [inline]

Get the surface parameters.

Returns

The surface parameters.

10.30.3.16 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. 507).

10.30.3.17 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. 508).

10.30.3.18 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. 508).

10.30.3.19 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. 508).

10.30.3.20 virtual const math::Pose& gazebo::physics::Collision::GetWorldPose ( ) const [virtual]

Get the absolute pose of the entity.

Returns

The absolute pose of the entity.

Reimplemented from **gazebo::physics::Entity** (p. 508).

10.30.3.21 virtual void gazebo::physics::Collision::Init ( ) [virtual]

Initialize the collision.

Reimplemented from **gazebo::physics::Base** (p. 211).

Reimplemented in **gazebo::physics::DARTCollision** (p. 388).

10.30.3.22 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.

10.30.3.23 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. 509).

Reimplemented in **gazebo::physics::SimbodyCollision** (p. 1177), and **gazebo::physics::DARTCollision** (p. 388).

10.30.3.24 void gazebo::physics::Collision::ProcessMsg (const msgs::Collision & _msg)

Update parameters from a message.

Parameters

in	*_msg*	Message to update from.

10.30.3.25 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.

Implemented in **gazebo::physics::DARTCollision** (p. 388), and **gazebo::physics::SimbodyCollision** (p. 1177).

10.30.3.26 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.

Implemented in **gazebo::physics::DARTCollision** (p. 389), and **gazebo::physics::SimbodyCollision** (p. 1177).

10.30.3.27 void gazebo::physics::Collision::SetCollision (bool *_placeable*)

Set the encapsulated collsion object.

Parameters

in	*_placeable*	True to make the object movable.

10.30.3.28 void gazebo::physics::Collision::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	*_retro*	The laser retro value.

10.30.3.29 virtual void gazebo::physics::Collision::SetMaxContacts (unsigned int *_maxContacts*) [virtual]

Number of contacts allowed for this collision.

This overrides global value (in **PhysicsEngine** (p. 959)) if specified.

Parameters

in	*_maxContacts*	max num contacts allowed for this collision.

10.30.3.30 void gazebo::physics::Collision::SetScale (const math::Vector3 & *_scale*)

Set the scale of the collision.

Parameters

in	*_scale*	Scale to set the collision to.

10.30.3.31 void gazebo::physics::Collision::SetShape (ShapePtr *_shape*)

Set the shape for this collision.

Parameters

in	*_shape*	The shape for this collision object.

10.30.3.32 void gazebo::physics::Collision::SetState (const CollisionState & *_state*)

Set the current collision state.

Parameters

in	*The*	collision state.

10.30.3.33 void gazebo::physics::Collision::SetWorldPoseDirty ( )

Indicate that the world pose should be recalculated.

The recalculation will be done when **Collision::GetWorldPose** (p. 303) is called.

10.30.3.34 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. 512).

10.30.4 Member Data Documentation

10.30.4.1 LinkPtr gazebo::physics::Collision::link [protected]

The link this collision belongs to.

10.30.4.2 bool gazebo::physics::Collision::placeable [protected]

Flag for placeable.

10.30.4.3 ShapePtr gazebo::physics::Collision::shape [protected]

Pointer to **physics::Shape** (p. 1161).

10.30.4.4 SurfaceParamsPtr gazebo::physics::Collision::surface [protected]

The surface parameters.

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

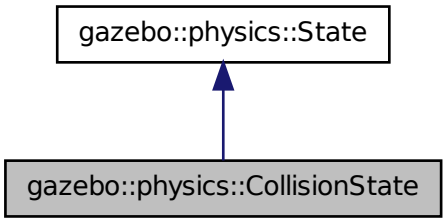
- **Collision.hh**

10.31 gazebo::physics::CollisionState Class Reference

Store state information of a **physics::Collision** (p. 295) 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. 295) 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.

10.31.1 Detailed Description

Store state information of a **physics::Collision** (p. 295) object.

This class captures the entire state of a **Collision** (p. 295) at one specific time during a simulation run.

State (p. 1323) of a **Collision** (p. 295) is its Pose.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 gazebo::physics::CollisionState::CollisionState ( )

Default constructor.

10.31.2.2 gazebo::physics::CollisionState::CollisionState (const CollisionPtr _collision) [explicit]

Constructor.

Build a **CollisionState** (p. 307) from an existing **Collision** (p. 295).

Parameters

in	*_model*	Pointer to the **Link** (p. 739) from which to gather state info.

10.31.2.3 gazebo::physics::CollisionState::CollisionState (const sdf::ElementPtr _sdf) [explicit]

Constructor.

Build a **CollisionState** (p. 307) from SDF data

Parameters

in	*_sdf*	SDF data to load a collision state from.

10.31.2.4 virtual gazebo::physics::CollisionState::~CollisionState ( ) [virtual]

Destructor.

10.31.3 Member Function Documentation

10.31.3.1 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.

10.31.3.2 const math::Pose& gazebo::physics::CollisionState::GetPose ( ) const

Get the **Collision** (p. 295) pose.

Returns

The pose of the **CollisionState** (p. 307)

10.31.3.3 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.

10.31.3.4 virtual void gazebo::physics::CollisionState::Load (const sdf::ElementPtr _*elem*) [virtual]

Load state from SDF element.

Load **CollisionState** (p. 307) 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. 1326).

10.31.3.5 CollisionState gazebo::physics::CollisionState::operator+ (const CollisionState & _*state*) const

Addition operator.

Parameters

in	*_pt*	A state to add.

Returns

The resulting state.

10.31.3.6 CollisionState gazebo::physics::CollisionState::operator- (const CollisionState & _*state*) const

Subtraction operator.

Parameters

in	*_pt*	A state to substract.

Returns

The resulting state.

10.31.3.7 CollisionState& gazebo::physics::CollisionState::operator= (const CollisionState & *_state*)

Assignment operator.

Parameters

in	*_state*	**State** (p. 1323) value

Returns

Reference to this

10.31.4 Friends And Related Function Documentation

10.31.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::physics::CollisionState & *_state*) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_state*	**Collision** (p. 295) state to output

Returns

the stream

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

- **CollisionState.hh**

10.32 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.

10.32.1 Detailed Description

Defines a color.

10.32.2 Member Typedef Documentation

10.32.2.1 typedef unsigned int gazebo::common::Color::ABGR

10.32.2.2 typedef unsigned int gazebo::common::Color::ARGB

10.32.2.3 typedef unsigned int gazebo::common::Color::BGRA

10.32.2.4 typedef unsigned int gazebo::common::Color::RGBA

10.32.3 Constructor & Destructor Documentation

10.32.3.1 gazebo::common::Color::Color ( )

Constructor.

10.32.3.2 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)

10.32.3.3 gazebo::common::Color::Color (const Color & *_clr*)

Copy Constructor.

Parameters

in	*_clr*	**Color** (p. 312) to copy

10.32.3.4 virtual gazebo::common::Color::~Color ( ) [virtual]

Destructor.

10.32.4 Member Function Documentation

10.32.4.1 ABGR gazebo::common::Color::GetAsABGR ( ) const

Get as uint32 ABGR packed value.

Returns

the color

10.32.4.2 ARGB gazebo::common::Color::GetAsARGB ( ) const

Get as uint32 ARGB packed value.

Returns

the color

10.32.4.3 BGRA gazebo::common::Color::GetAsBGRA ( ) const

Get as uint32 BGRA packed value.

Returns

the color

10.32.4.4 math::Vector3 gazebo::common::Color::GetAsHSV ( ) const

Get the color in HSV colorspace.

Returns

HSV values in a **math::Vector3** (p. 1440) format

10.32.4.5 RGBA gazebo::common::Color::GetAsRGBA ( ) const

Get as uint32 RGBA packed value.

Returns

the color

10.32.4.6 math::Vector3 gazebo::common::Color::GetAsYUV ( ) const

Get the color in YUV colorspace.

Returns

the YUV color

10.32.4.7 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

10.32.4.8 const Color gazebo::common::Color::operator* (const Color & _pt) const

Multiplication operator.

Parameters

in	*_pt*	The color to muliply by

Returns

The resulting color

10.32.4.9 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

10.32.4.10 const Color& gazebo::common::Color::operator*= (const Color & _pt)

Multiplication equal operator.

Parameters

in	*_pt*	The color to muliply by

Returns

The resulting color

10.32.4.11 Color gazebo::common::Color::operator+ (const Color & _pt) const

Addition operator (this + _pt)

Parameters

in	*_pt*	**Color** (p. 312) to add

Returns

The resulting color

10.32.4.12 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

10.32.4.13 const Color& gazebo::common::Color::operator+= (const Color & _pt)

Addition equal operator.

Parameters

in	*_pt*	**Color** (p. 312) to add

Returns

The resulting color

10.32.4.14 Color gazebo::common::Color::operator- (const Color & _pt) const

Subtraction operator.

Parameters

in	*_pt*	The color to substract

Returns

The resulting color

10.32.4.15 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

10.32.4.16 const Color& gazebo::common::Color::operator-= (const Color & _pt)

Subtraction equal operator.

Parameters

in	*_pt*	**Color** (p. 312) to subtract

Returns

The resulting color

10.32.4.17 const Color gazebo::common::Color::operator/ (const Color & _pt) const

Division operator.

Parameters

in	*_pt*	**Color** (p. 312) to divide by

Returns

The resulting color

10.32.4.18 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

10.32.4.19 const Color& gazebo::common::Color::operator/= (const Color & _pt)

Division equal operator.

Parameters

in	*_pt*	**Color** (p. 312) to divide by

Returns

The resulting color

10.32.4.20 Color& gazebo::common::Color::operator= (const Color & _pt)

Equal operator.

Parameters

in	*_pt*	**Color** (p. 312) to copy

Returns

Reference to this color

10.32.4.21 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

10.32.4.22 float gazebo::common::Color::operator[] (unsigned int _index)

Array index operator.

Parameters

in	*_index*	**Color** (p. 312) component index(0=red, 1=green, 2=blue)

Returns

r, g, b, or a when _index is 0, 1, 2 or 3

10.32.4.23 void gazebo::common::Color::Reset ( )

Reset the color to default values.

10.32.4.24 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)

10.32.4.25 void gazebo::common::Color::SetFromABGR (const ABGR _v)

Set from uint32 ABGR packed value.

Parameters

in	_v	the new color

10.32.4.26 void gazebo::common::Color::SetFromARGB (const ARGB _v)

Set from uint32 ARGB packed value.

Parameters

in	_v	the new color

10.32.4.27 void gazebo::common::Color::SetFromBGRA (const BGRA _v)

Set from uint32 BGRA packed value.

Parameters

in	_v	the new color

10.32.4.28 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)

10.32.4.29 void gazebo::common::Color::SetFromRGBA (const RGBA _v)

Set from uint32 RGBA packed value.

Parameters

in	_v	the new color

10.32.4.30 void gazebo::common::Color::SetFromYUV (float _y, float _u, float _v)

Set from yuv.

Parameters

in	_y	value
in	_u	value
in	_v	value

10.32.5 Friends And Related Function Documentation

10.32.5.1 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

10.32.5.2 std::istream& operator>> (std::istream & *_in,* Color & *_pt*) [friend]

Stream insertion operator.

Parameters

in	*_in*	the input stream
in	*_pt*	

10.32.6 Member Data Documentation

10.32.6.1 float gazebo::common::Color::a

10.32.6.2 float gazebo::common::Color::b

10.32.6.3 const Color gazebo::common::Color::Black [static]

(0, 0, 0)

10.32.6.4 const Color gazebo::common::Color::Blue [static]

(0, 0, 1)

10.32.6.5 float gazebo::common::Color::g

10.32.6.6 const Color gazebo::common::Color::Green [static]

(0, 1, 0)

10.32.6.7 const Color gazebo::common::Color::Purple [static]

(1, 0, 1)

10.32.6.8 float gazebo::common::Color::r

10.32.6.9 const Color gazebo::common::Color::Red [static]

(1, 0, 0)

10.32.6.10 const Color gazebo::common::Color::White [static]

(1, 1, 1)

10.32.6.11 const Color gazebo::common::Color::Yellow [static]

(1, 1, 0)

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

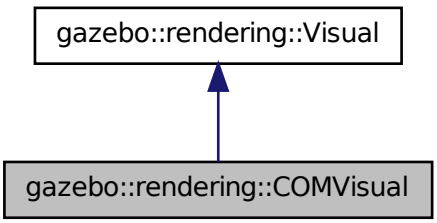
- **Color.hh**

10.33 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. 1477) from an SDF pointer.*

- virtual void **Load** (ConstLinkPtr &_msg)

  Load from a message.

10.33.1 Detailed Description

Basic Center of Mass visualization.

10.33.2 Constructor & Destructor Documentation

10.33.2.1 gazebo::rendering::COMVisual::COMVisual (const std::string & *_name*, VisualPtr *_vis*)

Constructor.

Parameters

in	*_name*	Name of the **Visual** (p. 1477)
in	*_vis*	Parent **Visual** (p. 1477)

10.33.2.2 virtual gazebo::rendering::COMVisual::∼COMVisual ( ) [virtual]

Destructor.

10.33.3 Member Function Documentation

10.33.3.1 virtual void gazebo::rendering::COMVisual::Load (sdf::ElementPtr *_elem*) [virtual]

Load the **Visual** (p. 1477) from an SDF pointer.

Parameters

in	*_elem*	SDF Element pointer

Reimplemented from **gazebo::rendering::Visual** (p. 1494).

10.33.3.2 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:

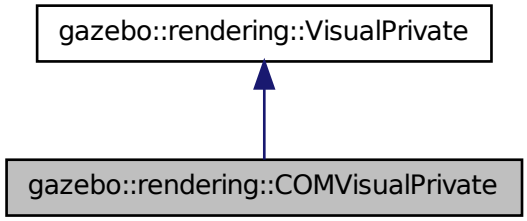
- **COMVisual.hh**

10.34 gazebo::rendering::COMVisualPrivate Class Reference

Private data for the COM **Visual** (p. 1477) class.

`#include <COMVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::COMVisualPrivate:



Public Attributes

- Ogre::SceneNode ∗ **boxNode**

  Box that make the cross marking the center of mass.

- **DynamicLines** ∗ **crossLines**

  Lines that make the cross marking the center of mass.

10.34.1 Detailed Description

Private data for the COM **Visual** (p. 1477) class.

10.34.2 Member Data Documentation

10.34.2.1 Ogre::SceneNode* gazebo::rendering::COMVisualPrivate::boxNode

Box that make the cross marking the center of mass.

10.34.2.2 DynamicLines* gazebo::rendering::COMVisualPrivate::crossLines

Lines that make the cross marking the center of mass.

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

- **COMVisualPrivate.hh**

10.35 gazebo::transport::Connection Class Reference

Single TCP/IP connection manager.

`#include <transport/transport.hh>`

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, boost::function< void(uint32_t)> _cb, uint32_t _id, bool _force=false)

  Write data to the socket.
- 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 **GetIPWhiteList** () const

  Get the IP white list, from GAZEBO_IP_WHITE_LIST environment variable.
- 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.

10.35.1 Detailed Description

Single TCP/IP connection manager.

10.35.2 Member Typedef Documentation

10.35.2.1 typedef boost::function<void(const ConnectionPtr&)> gazebo::transport::Connection::AcceptCallback

The signature of a connection accept callback.

10.35.2.2 typedef boost::function<void(const std::string &_data)> gazebo::transport::Connection::ReadCallback

The signature of a connection read callback.

10.35.3 Constructor & Destructor Documentation

10.35.3.1 gazebo::transport::Connection::Connection ( )

Constructor.

10.35.3.2 virtual gazebo::transport::Connection::~Connection ( ) [virtual]

Destructor.

10.35.4 Member Function Documentation

10.35.4.1 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, and HEADER_LENGTH.

10.35.4.2 void gazebo::transport::Connection::Cancel ( )

Cancel all async operations on an open socket.

10.35.4.3 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

10.35.4.4 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::shutdown().

10.35.4.5 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. 331)

References gazebo::shutdown().

10.35.4.6 void gazebo::transport::Connection::EnqueueMsg (const std::string & _buffer, boost::function< void(uint32_t)> _cb, uint32_t _id, 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
in	*_cb*	If non-null, callback to be invoked after transmission is complete.
in	*_id*	ID associated with the message data.

10.35.4.7 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

10.35.4.8 unsigned int gazebo::transport::Connection::GetId ( ) const

Get the ID of the connection.

Returns

The connection's unique ID.

10.35.4.9 std::string gazebo::transport::Connection::GetIPWhiteList ( ) const

Get the IP white list, from GAZEBO_IP_WHITE_LIST environment variable.

Returns

GAZEBO_IP_WHITE_LIST

10.35.4.10 std::string gazebo::transport::Connection::GetLocalAddress ( ) const

Get the local address of this connection.

Returns

The local address

10.35.4.11 static std::string gazebo::transport::Connection::GetLocalHostname ( ) [static]

Get the local hostname.

Returns

The local hostname

10.35.4.12 unsigned int gazebo::transport::Connection::GetLocalPort ( ) const

Get the port of this connection.

Returns

The local port

10.35.4.13 std::string gazebo::transport::Connection::GetLocalURI ( ) const

Get the local URI.

Returns

The local URI

10.35.4.14 std::string gazebo::transport::Connection::GetRemoteAddress ( ) const

Get the remote address.

Returns

The remote address

10.35.4.15 std::string gazebo::transport::Connection::GetRemoteHostname ( ) const

Get the remote hostname.

Returns

The remote hostname

10.35.4.16 unsigned int gazebo::transport::Connection::GetRemotePort ( ) const

Get the remote port number.

Returns

The remote port

10.35.4.17 std::string gazebo::transport::Connection::GetRemoteURI ( ) const

Get the remote URI.

Returns

The remote URI

10.35.4.18 bool gazebo::transport::Connection::IsOpen ( ) const

Is the connection open?

Returns

true if the connection is open; false otherwise

10.35.4.19 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

10.35.4.20 void gazebo::transport::Connection::ProcessWriteQueue (bool *_blocking* = false)

Handle on-write callbacks.

10.35.4.21 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

10.35.4.22 void gazebo::transport::Connection::Shutdown ( )

Shutdown the socket.

10.35.4.23 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

10.35.4.24 void gazebo::transport::Connection::StopRead ( )

Stop the read loop.

10.35.4.25 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**

10.36 gazebo::event::Connection Class Reference

A class that encapsulates a connection.

`#include <Event.hh>`

Public Member Functions

- **Connection** ()
  Constructor.
- **Connection** (**Event** *_e, int _i)
  Constructor.
- ~**Connection** ()
  Destructor.
- int **GetId** () const
  Get the id of this connection.

10.36.1 Detailed Description

A class that encapsulates a connection.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 gazebo::event::Connection::Connection ( )

Constructor.

10.36.2.2 gazebo::event::Connection::Connection (Event * _e, int _i)

Constructor.

Parameters

in	*_e*	**Event** (p. 514) pointer to connect with.
in	*_i*	Unique id.

10.36.2.3 gazebo::event::Connection::~Connection ( )

Destructor.

10.36.3 Member Function Documentation

10.36.3.1 int gazebo::event::Connection::GetId ( ) const

Get the id of this connection.

Returns

The id of this connection.

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

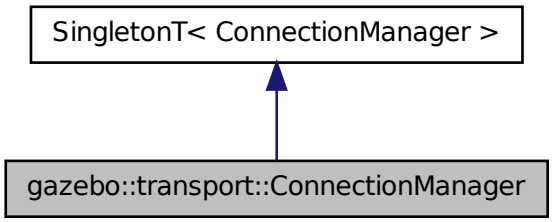
- **Event.hh**

10.37 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, uint32_t _timeoutIterations=30)

  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 **Stop** ()

  Stop the conneciton manager.
- void **Subscribe** (const std::string &_topic, const std::string &_msgType, bool _latching)

  Subscribe to a topic.
- void **TriggerUpdate** ()

  Inform the connection manager that it needs an update.
- 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**

10.37.1 Detailed Description

Manager of connections.

10.37.2 Member Function Documentation

10.37.2.1 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

10.37.2.2 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)

10.37.2.3 void gazebo::transport::ConnectionManager::Fini ( )

Finalize the connection manager.

10.37.2.4 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

10.37.2.5 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

10.37.2.6 bool gazebo::transport::ConnectionManager::Init (const std::string & *_masterHost*, unsigned int *_masterPort*, uint32_t *_timeoutIterations* = 30)

Initialize the connection manager.

Parameters

in	*_masterHost*	Host where the master is running.
in	*_masterPort*	Port where the master is running.
in	*_timeoutIterations*	Number of times to wait for a connection to master.

Returns

true if initialization succeeded, false otherwise

10.37.2.7 bool gazebo::transport::ConnectionManager::IsRunning ( ) const

Is the manager running?

Returns

true if running, false otherwise

10.37.2.8 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

10.37.2.9 void gazebo::transport::ConnectionManager::RemoveConnection (ConnectionPtr & _conn)

Remove a connection from the manager.

Parameters

in	*_conn*	The connection to be removed

10.37.2.10 void gazebo::transport::ConnectionManager::Run ( )

Run the connection manager loop.

Does not return until stopped.

10.37.2.11 void gazebo::transport::ConnectionManager::Stop ( )

Stop the conneciton manager.

10.37.2.12 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.

10.37.2.13 void gazebo::transport::ConnectionManager::TriggerUpdate ( )

Inform the connection manager that it needs an update.

10.37.2.14 void gazebo::transport::ConnectionManager::Unadvertise (const std::string & _topic)

Unadvertise a topic.

Parameters

in	*_topic*	The topic to unadvertise

10.37.2.15 void gazebo::transport::ConnectionManager::Unsubscribe (const msgs::Subscribe & _sub)

Unsubscribe from a topic.

Parameters

in	*_sub*	A subscription object

10.37.2.16 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

10.37.3 Member Data Documentation

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

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

- **ConnectionManager.hh**

10.38 gazebo::event::ConnectionPrivate Class Reference

`#include <Event.hh>`

Public Member Functions

- **ConnectionPrivate** ()

  Constructor.
- **ConnectionPrivate** (**Event** *_e, int _i)

  Constructor.

Public Attributes

- **common::Time creationTime**

  set during the constructor
- **Event** * **event**

  the event for this connection
- int **id**

  the id set in the constructor

10.38.1 Constructor & Destructor Documentation

10.38.1.1 gazebo::event::ConnectionPrivate::ConnectionPrivate ( )

Constructor.

10.38.1.2 gazebo::event::ConnectionPrivate::ConnectionPrivate (Event * *_e*, int *_i*)

Constructor.

Parameters

in	*_e*	**Event** (p. 514) pointer to connect with
in	*_i*	Unique id

10.38.2 Member Data Documentation

10.38.2.1 common::Time gazebo::event::ConnectionPrivate::creationTime

set during the constructor

10.38.2.2 Event* gazebo::event::ConnectionPrivate::event

the event for this connection

10.38.2.3 int gazebo::event::ConnectionPrivate::id

the id set in the constructor

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

- **Event.hh**

10.39 gazebo::transport::ConnectionReadTask Class Reference

`#include <Connection.hh>`

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.

10.39.1 Detailed Description

A task instance that is created when data is read from a socket and used by TBB

10.39.2 Constructor & Destructor Documentation

10.39.2.1 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.

10.39.3 Member Function Documentation

10.39.3.1 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**

10.40 gazebo::common::Console Class Reference

Container for loggers, and global logging options (such as verbose vs.

`#include <common/common.hh>`

Static Public Member Functions

- static bool **GetQuiet** ()

  Get whether quiet output is set.
- static void **SetQuiet** (bool _q)

  Set quiet output.

Static Public Attributes

- static **Logger dbg**

Global instance of the debug logger.

- static **Logger err**

  Global instance of the error logger.

- static **FileLogger log**

  Global instance of the file logger.

- static **Logger msg**

  Global instance of the message logger.

- static **Logger warn**

  Global instance of the warning logger.

10.40.1 Detailed Description

Container for loggers, and global logging options (such as verbose vs.

quiet output).

10.40.2 Member Function Documentation

10.40.2.1 static bool gazebo::common::Console::GetQuiet ( ) [static]

Get whether quiet output is set.

Returns

True to if quiet output is set.

10.40.2.2 static void gazebo::common::Console::SetQuiet (bool _q) [static]

Set quiet output.

Parameters

in	*q*	True to prevent warning.

10.40.3 Member Data Documentation

10.40.3.1 Logger gazebo::common::Console::dbg [static]

Global instance of the debug logger.

10.40.3.2 Logger gazebo::common::Console::err [static]

Global instance of the error logger.

10.40.3.3 FileLogger gazebo::common::Console::log [static]

Global instance of the file logger.

10.40.3.4 Logger gazebo::common::Console::msg [static]

Global instance of the message logger.

10.40.3.5 Logger gazebo::common::Console::warn [static]

Global instance of the warning logger.

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

- **Console.hh**

10.41 gazebo::physics::Contact Class Reference

A contact between two collisions.

`#include <physics/physics.hh>`

Public Member Functions

- **Contact** ()

  Constructor.
- **Contact** (const **Contact** &_contact)

  Copy constructor.
- virtual ~**Contact** ()

  Destructor.
- 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

- **Collision** ∗ **collision1**

  Pointer to the first collision object.
- **Collision** ∗ **collision2**

  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. *1529) in which the contact occurred.*
- **JointWrench wrench** [32]

  Array of forces for the contact.

10.41.1 Detailed Description

A contact between two collisions.

Each contact can consist of a number of contact points

10.41.2 Constructor & Destructor Documentation

10.41.2.1 gazebo::physics::Contact::Contact ( )

Constructor.

10.41.2.2 gazebo::physics::Contact::Contact (const Contact & _contact)

Copy constructor.

Parameters

in	*_contact*	**Contact** (p. 347) to copy.

10.41.2.3 virtual gazebo::physics::Contact::~Contact ( ) [virtual]

Destructor.

10.41.3 Member Function Documentation

10.41.3.1 std::string gazebo::physics::Contact::DebugString ( ) const

Produce a debug string.

Returns

A string that contains the values of the contact.

10.41.3.2 void gazebo::physics::Contact::FillMsg (msgs::Contact & _msg) const

Populate a msgs::Contact with data from this.

Parameters

out	*_msg*	**Contact** (p. 347) message the will hold the data.

10.41.3.3 Contact& gazebo::physics::Contact::operator= (const Contact & _contact)

Operator =.

Parameters

in	*_contact*	**Contact** (p. 347) to copy.

Returns

Reference to this contact

10.41.3.4 Contact& gazebo::physics::Contact::operator= (const msgs::Contact & *_contact*)

Operator =.

Parameters

in	*_contact*	msgs::Contact to copy.

Returns

Reference to this contact

10.41.3.5 void gazebo::physics::Contact::Reset ( )

Reset to default values.

10.41.4 Member Data Documentation

10.41.4.1 Collision* gazebo::physics::Contact::collision1

Pointer to the first collision object.

10.41.4.2 Collision* gazebo::physics::Contact::collision2

Pointer to the second collision object.

10.41.4.3 int gazebo::physics::Contact::count

Length of all the arrays.

10.41.4.4 double gazebo::physics::Contact::depths[32]

Array of contact depths.

10.41.4.5 math::Vector3 gazebo::physics::Contact::normals[32]

Array of force normals.

10.41.4.6 math::Vector3 gazebo::physics::Contact::positions[32]

Array of force positions.

10.41.4.7 common::Time gazebo::physics::Contact::time

Time at which the contact occurred.

10.41.4.8 WorldPtr gazebo::physics::Contact::world

World (p. 1529) in which the contact occurred.

10.41.4.9 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**

10.42 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.
- std::string **CreateFilter** (const std::string &_topic, const std::vector< std::string > &_collisions)

  Create a filter for contacts.
- std::string **CreateFilter** (const std::string &_topic, const std::string &_collision)

Create a filter for contacts.

- std::string **CreateFilter** (const std::string &_name, const std::map< std::string, **physics::CollisionPtr** > &_collisions)

  Create a filter for 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.
- unsigned int **GetFilterCount** ()

  Get the number of filters in the contact manager.
- bool **HasFilter** (const std::string &_name)

  Check if a filter with the specified name exists.
- void **Init** (**WorldPtr** _world)

  *Initialize the **ContactManager** (p. 351).*
- **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 **RemoveFilter** (const std::string &_name)

  Remove a contacts filter and the associated custom publisher param[in] _name Filter name.
- void **ResetCount** ()

  Set the contact count to zero.

10.42.1 Detailed Description

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

10.42.2 Constructor & Destructor Documentation

10.42.2.1 gazebo::physics::ContactManager::ContactManager ( )

Constructor.

10.42.2.2 virtual gazebo::physics::ContactManager::~ContactManager ( ) [virtual]

Destructor.

10.42.3 Member Function Documentation

10.42.3.1 void gazebo::physics::ContactManager::Clear ( )

Clear all stored contacts.

10.42.3.2 std::string gazebo::physics::ContactManager::CreateFilter (const std::string & *_topic,* const std::vector< std::string > & *_collisions*)

Create a filter for contacts.

A new publisher will be created that publishes contacts associated to the input collisions. param[in] _name Filter name. param[in] _collisions A list of collision names used for filtering.

Returns

New topic where filtered messages will be published to.

10.42.3.3 std::string gazebo::physics::ContactManager::CreateFilter (const std::string & *_topic,* const std::string & *_collision*)

Create a filter for contacts.

A new publisher will be created that publishes contacts associated to the input collision. param[in] _name Filter name. param[in] _collision A collision name used for filtering.

Returns

New topic where filtered messages will be published to.

10.42.3.4 std::string gazebo::physics::ContactManager::CreateFilter (const std::string & *_name,* const std::map< std::string, physics::CollisionPtr > & *_collisions*)

Create a filter for contacts.

A new publisher will be created that publishes contacts associated to the input collision. param[in] _name Filter name. param[in] _collisions A map of collision name to collision object.

Returns

New topic where filtered messages will be published to.

10.42.3.5 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. 354).

Parameters

in	*_index*	Index of the **Contact** (p. 347) to return.

Returns

Pointer to a contact, NULL If index is invalid.

10.42.3.6 unsigned int gazebo::physics::ContactManager::GetContactCount ( ) const

Return the number of valid contacts.

10.42.3.7 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. 354)

Returns

Vector of contact pointers.

10.42.3.8 unsigned int gazebo::physics::ContactManager::GetFilterCount ( )

Get the number of filters in the contact manager.

return Number of filters

10.42.3.9 bool gazebo::physics::ContactManager::HasFilter (const std::string & _name)

Check if a filter with the specified name exists.

param[in] _name Name of filter. return True if the filter exists.

10.42.3.10 void gazebo::physics::ContactManager::Init (WorldPtr _world)

Initialize the **ContactManager** (p. 351).

This is required in order to publish contact messages via the **ContactManager::-PublishContacts** (p. 355) method.

Parameters

in	*_world*	Pointer to the world that is initializing the contact manager.

10.42.3.11 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.

10.42.3.12 void gazebo::physics::ContactManager::PublishContacts ( )

Publish all contacts in a msgs::Contacts message.

10.42.3.13 void gazebo::physics::ContactManager::RemoveFilter (const std::string & _name)

Remove a contacts filter and the associated custom publisher param[in] _name Filter name.

10.42.3.14 void gazebo::physics::ContactManager::ResetCount ( )

Set the contact count to zero.

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

- **ContactManager.hh**

10.43 gazebo::rendering::ContactVisualPrivate::ContactPoint - Class Reference

A contact point visualization.

`#include <ContactVisualPrivate.hh>`

Public Attributes

- **DynamicLines** * **depth**
- **DynamicLines** * **normal**

  Normal and depth for the contact point.
- Ogre::SceneNode * **sceneNode**

  The scene node for the contact visualization.

10.43.1 Detailed Description

A contact point visualization.

10.43.2 Member Data Documentation

10.43.2.1 DynamicLines * gazebo::rendering::ContactVisualPrivate::ContactPoint::depth

10.43.2.2 DynamicLines* gazebo::rendering::ContactVisualPrivate::ContactPoint::normal

Normal and depth for the contact point.

10.43.2.3 Ogre::SceneNode* gazebo::rendering::ContactVisualPrivate::ContactPoint::sceneNode

The scene node for the contact visualization.

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

- **ContactVisualPrivate.hh**

10.44 gazebo::physics::ContactPublisher Class Reference

A custom contact publisher created for each contact filter in the **Contact** (p. 347) - Manager.

`#include <ContactManager.hh>`

Public Attributes

- std::vector< std::string > **collisionNames**
- boost::unordered_set< **Collision** * > **collisions**

  Pointers of collisions monitored by contact manager for contacts.
- std::vector< **Contact** * > **contacts**

  A list of contacts associated to the collisions.
- **transport::PublisherPtr publisher**

  Contact *(p. 347) message publisher.*

10.44.1 Detailed Description

A custom contact publisher created for each contact filter in the **Contact** (p. 347) - Manager.

10.44.2 Member Data Documentation

10.44.2.1 std::vector<std::string> gazebo::physics::ContactPublisher::collisionNames

10.44.2.2 boost::unordered_set<Collision *> gazebo::physics::ContactPublisher::collisions

Pointers of collisions monitored by contact manager for contacts.

10.44.2.3 std::vector<Contact *> gazebo::physics::ContactPublisher::contacts

A list of contacts associated to the collisions.

10.44.2.4 transport::PublisherPtr gazebo::physics::ContactPublisher::publisher

Contact (p. 347) message publisher.

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

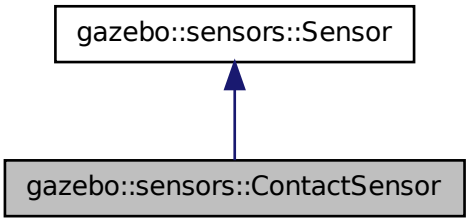
- **ContactManager.hh**

10.45 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. 358).*
- 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 bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

10.45.1 Detailed Description

Contact sensor.

This sensor detects and reports contacts between objects

10.45.2 Constructor & Destructor Documentation

10.45.2.1 gazebo::sensors::ContactSensor::ContactSensor ( )

Constructor.

10.45.2.2 virtual gazebo::sensors::ContactSensor::~ContactSensor ( ) [virtual]

Destructor.

10.45.3 Member Function Documentation

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

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.45.3.2 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.

10.45.3.3 unsigned int gazebo::sensors::ContactSensor::GetCollisionCount ( ) const

Get the number of collisions that the sensor is observing.

Returns

Number of collisions.

10.45.3.4 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.

10.45.3.5 msgs::Contacts gazebo::sensors::ContactSensor::GetContacts ( ) const

Get all the contacts for the **ContactSensor** (p. 358).

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. 358) 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. 358) SDF. All collision pairs between **ContactSensor** (p. 358) <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 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.

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

Gets contacts of a collision.

Parameters

in	*_collision-Name*	Name of collision

Returns

Container of contacts

10.45.3.7 virtual void gazebo::sensors::ContactSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.45.3.8 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. 1139).

10.45.3.9 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. 1130) parameters
in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 1140).

10.45.3.10 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. 1140).

10.45.3.11 virtual bool gazebo::sensors::ContactSensor::UpdateImpl (bool)

`[protected, virtual]`

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

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

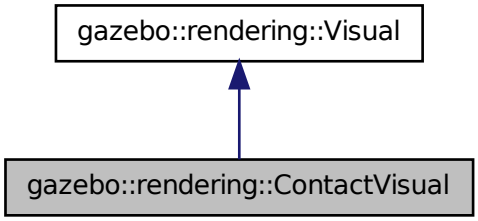
- **ContactSensor.hh**

10.46 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.

10.46.1 Detailed Description

Contact visualization.

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

10.46.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_name*	Name of the **ContactVisual** (p. 363)
in	*_vis*	Pointer the parent **Visual** (p. 1477)
in	*_topicName*	Name of the topic which publishes the contact information.

10.46.2.2 virtual gazebo::rendering::ContactVisual::~ContactVisual ( ) [virtual]

Destructor.

10.46.3 Member Function Documentation

10.46.3.1 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:

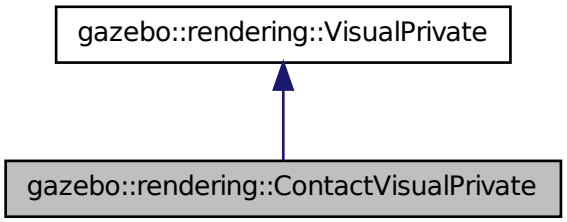
- **ContactVisual.hh**

10.47 gazebo::rendering::ContactVisualPrivate Class Reference

Private data for the Arrow **Visual** (p. 1477) class.

`#include <ContactVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::ContactVisualPrivate:



Classes

- class **ContactPoint**

  A contact point visualization.

Public Attributes

- std::vector< **event::ConnectionPtr** > **connections**

  All the event connections.
- boost::shared_ptr < msgs::Contacts const > **contactsMsg**

  The current contact message.
- **transport::SubscriberPtr contactsSub**

Subscription to the contact data.

- bool **enabled**

  True if this visualization is enabled.

- boost::mutex **mutex**

  Mutex to protect the contact message.

- **transport::NodePtr node**

  Node for communication.

- std::vector < **ContactVisualPrivate::ContactPoint** * > **points**

  All the contact points.

- bool **receivedMsg**

  True if we have received a message.

- std::string **topicName**

  Name of the topic contact information is published on.

10.47.1 Detailed Description

Private data for the Arrow **Visual** (p. 1477) class.

10.47.2 Member Data Documentation

10.47.2.1 std::vector<event::ConnectionPtr> gazebo::rendering::ContactVisualPrivate::connections

All the event connections.

10.47.2.2 boost::shared_ptr<msgs::Contacts const> gazebo::rendering::ContactVisualPrivate::contactsMsg

The current contact message.

10.47.2.3 transport::SubscriberPtr gazebo::rendering::ContactVisualPrivate::contactsSub

Subscription to the contact data.

10.47.2.4 bool gazebo::rendering::ContactVisualPrivate::enabled

True if this visualization is enabled.

10.47.2.5 boost::mutex gazebo::rendering::ContactVisualPrivate::mutex

Mutex to protect the contact message.

10.47.2.6 transport::NodePtr gazebo::rendering::ContactVisualPrivate::node

Node for communication.

10.47.2.7 std::vector<ContactVisualPrivate::ContactPoint *> gazebo::rendering::ContactVisualPrivate::points

All the contact points.

10.47.2.8 bool gazebo::rendering::ContactVisualPrivate::receivedMsg

True if we have received a message.

10.47.2.9 std::string gazebo::rendering::ContactVisualPrivate::topicName

Name of the topic contact information is published on.

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

- **ContactVisualPrivate.hh**

10.48 gazebo::rendering::Conversions Class Reference

Conversions (p. 367) **Conversions.hh** (p. 1608) **rendering/Conversions.hh** (p. 1608).

```
#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. 163) 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. *163) quaternion.*

- static **math::Quaternion Convert** (const Ogre::Quaternion &_v)

  Ogre *(*p. *163) quaternion to Gazebo quaternion.*

10.48.1 Detailed Description

Conversions (p. 367) **Conversions.hh** (p. 1608) **rendering/Conversions.hh** (p. 1608).

A set of utility function to convert between Gazebo and **Ogre** (p. 163) data types

10.48.2 Member Function Documentation

10.48.2.1 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. 163) color value

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

Return the equivalent gazebo color.

Parameters

in	*_clr*	**Ogre** (p. 163) color to convert

Returns

Gazebo color value

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

return **Ogre** (p. 163) Vector from Gazebo Vector3

Parameters

in	_v	Gazebo vector

Returns

Ogre (p. 163) vector

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

return gazebo Vector from ogre Vector3

Parameters

in	_v	**Ogre** (p. 163) vector

Returns

Gazebo vector

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

Gazebo quaternion to **Ogre** (p. 163) quaternion.

Parameters

in	_v	Gazebo quaternion

Returns

Ogre (p. 163) quaternion

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

Ogre (p. 163) quaternion to Gazebo quaternion.

Parameters

in	_v	**Ogre** (p. 163) quaternion return Gazebo quaternion

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

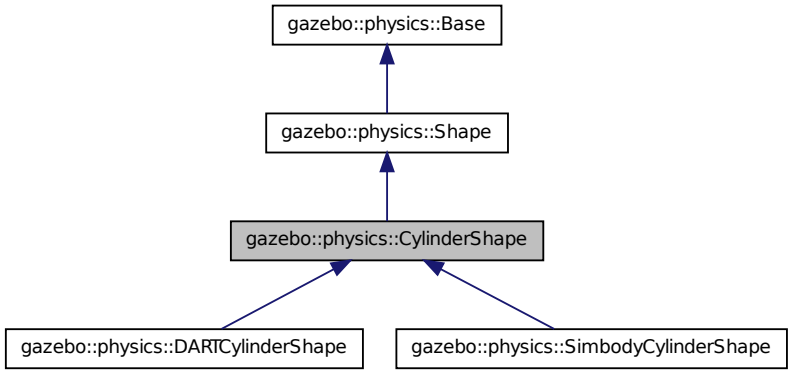
- **Conversions.hh**

10.49 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 **SetScale** (const **math::Vector3** &_scale)

  Set scale of cylinder.
- virtual void **SetSize** (double _radius, double _length)

  Set the size of the cylinder.

10.49.1 Detailed Description

Cylinder collision.

10.49.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_parent*	Parent of the shape.

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

Destructor.

10.49.3 Member Function Documentation

10.49.3.1 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. 1163).

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

Get length.

Returns

The cylinder length.

10.49.3.3 double gazebo::physics::CylinderShape::GetRadius ( ) const

Get radius.

Returns

The cylinder radius.

10.49.3.4 void gazebo::physics::CylinderShape::Init ( ) [virtual]

Initialize the cylinder.

Implements **gazebo::physics::Shape** (p. 1164).

10.49.3.5 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. 1164).

10.49.3.6 void gazebo::physics::CylinderShape::SetLength (double _length)

Set length.

Parameters

in	*_length*	New length of the cylinder.

10.49.3.7 void gazebo::physics::CylinderShape::SetRadius (double _radius)

Set radius.

Parameters

in}	_radius New radius of the cylinder.

10.49.3.8 virtual void gazebo::physics::CylinderShape::SetScale (const math::Vector3 & _scale) [virtual]

Set scale of cylinder.

Parameters

in	*_scale*	Scale to set the cylinder to.

Implements **gazebo::physics::Shape** (p. 1165).

10.49.3.9 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.

Reimplemented in **gazebo::physics::SimbodyCylinderShape** (p. 1179), and **gazebo::physics::DARTCylinderShape** (p. 391).

Referenced by gazebo::physics::DARTCylinderShape::SetSize(), and gazebo::physics-::SimbodyCylinderShape::SetSize().

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

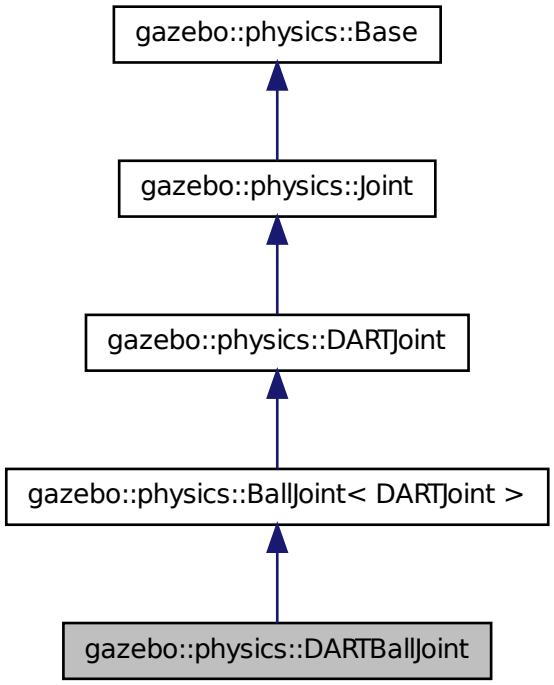
- **CylinderShape.hh**

10.50 gazebo::physics::DARTBallJoint Class Reference

An **DARTBallJoint** (p. 374).

`#include <DARTBallJoint.hh>`

Inheritance diagram for gazebo::physics::DARTBallJoint:



Public Member Functions

- **DARTBallJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**DARTBallJoint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const

  Get the anchor point.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.

- virtual **math::Angle GetHighStop** (unsigned int _index)

  Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (unsigned int _index)

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)
- virtual void **Init** ()

  Initialize joint.
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Template to ::Load the **BallJoint** (p. 199).*
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual void **SetMaxForce** (unsigned int _index, double _t)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual void **SetVelocity** (unsigned int _index, double _angle)

  Set the velocity of an axis(index).

Protected Member Functions

- void **SetForceImpl** (unsigned int _index, double _torque)

  *Set the force applied to this **physics::Joint** (p. 669).*

Protected Attributes

- dart::dynamics::BallJoint * **dtBallJoint**

10.50.1 Detailed Description

An **DARTBallJoint** (p. 374).

10.50.2 Constructor & Destructor Documentation

10.50.2.1 gazebo::physics::DARTBallJoint::DARTBallJoint (BasePtr *_parent*)

Constructor.

Parameters

in	*_parent*	Parent of the **Joint** (p. 669)

10.50.2.2 virtual gazebo::physics::DARTBallJoint::~DARTBallJoint ( ) [virtual]

Destructor.

10.50.3 Member Function Documentation

10.50.3.1 virtual math::Vector3 gazebo::physics::DARTBallJoint::GetAnchor (unsigned int *_index*) const [virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 679).

10.50.3.2 virtual math::Angle gazebo::physics::DARTBallJoint::GetAngleImpl (unsigned int *_index*) const [virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.50.3.3 virtual math::Vector3 gazebo::physics::DARTBallJoint::GetGlobalAxis (unsigned int *_index*) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.50.3.4 virtual math::Angle gazebo::physics::DARTBallJoint::GetHighStop (unsigned int *_index*) [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.

Reimplemented from **gazebo::physics::DARTJoint** (p. 411).

10.50.3.5 virtual math::Angle gazebo::physics::DARTBallJoint::GetLowStop (unsigned int *_index*) [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.

Reimplemented from **gazebo::physics::DARTJoint** (p. 412).

10.50.3.6 virtual double gazebo::physics::DARTBallJoint::GetMaxForce (unsigned int *_index*) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.50.3.7 virtual double gazebo::physics::DARTBallJoint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.50.3.8 virtual void gazebo::physics::DARTBallJoint::Init ( ) [virtual]

Initialize joint.

Reimplemented from **gazebo::physics::BallJoint< DARTJoint >** (p. 200).

10.50.3.9 virtual void gazebo::physics::DARTBallJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Template to ::Load the **BallJoint** (p. 199).

Parameters

in	*_sdf*	SDF to load the joint from.

Reimplemented from **gazebo::physics::BallJoint< DARTJoint >** (p. 200).

10.50.3.10 virtual void gazebo::physics::DARTBallJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 695).

10.50.3.11 void gazebo::physics::DARTBallJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value.

Implements **gazebo::physics::DARTJoint** (p. 415).

10.50.3.12 virtual bool gazebo::physics::DARTBallJoint::SetHighStop (unsigned 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.

Reimplemented from **gazebo::physics::DARTJoint** (p. 415).

10.50.3.13 virtual bool gazebo::physics::DARTBallJoint::SetLowStop (unsigned 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.

Reimplemented from **gazebo::physics::DARTJoint** (p. 416).

10.50.3.14 virtual void gazebo::physics::DARTBallJoint::SetMaxForce (unsigned int _index, double _force) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.50.3.15 virtual void gazebo::physics::DARTBallJoint::SetVelocity (unsigned int _index, double _vel) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

10.50.4 Member Data Documentation

10.50.4.1 dart::dynamics::BallJoint* gazebo::physics::DARTBallJoint::dtBallJoint [protected]

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

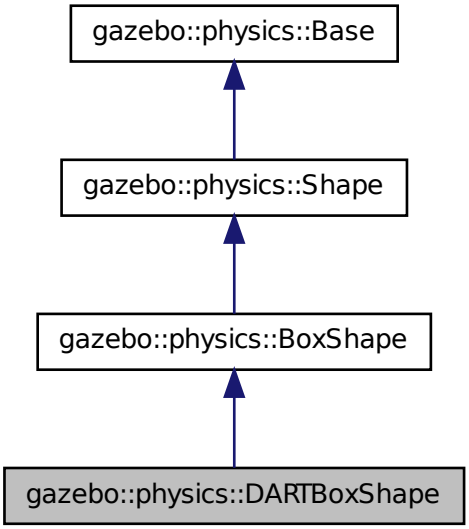
- **DARTBallJoint.hh**

10.51 gazebo::physics::DARTBoxShape Class Reference

DART Box shape.

`#include <DARTBoxShape.hh>`

Inheritance diagram for gazebo::physics::DARTBoxShape:



Public Member Functions

- **DARTBoxShape** (**DARTCollisionPtr** _parent)

  Constructor.
- virtual ~**DARTBoxShape** ()

  Destructor.
- virtual void **SetSize** (const **math::Vector3** &_size)

  Set the size of the box.

10.51.1 Detailed Description

DART Box shape.

10.51.2 Constructor & Destructor Documentation

10.51.2.1 gazebo::physics::DARTBoxShape::DARTBoxShape (DARTCollisionPtr *_parent*) [inline, explicit]

Constructor.

Parameters

in	*_parent*	Parent **Collision** (p. 295).

10.51.2.2 virtual gazebo::physics::DARTBoxShape::~DARTBoxShape ( ) [inline, virtual]

Destructor.

10.51.3 Member Function Documentation

10.51.3.1 virtual void gazebo::physics::DARTBoxShape::SetSize (const math::Vector3 & *_size*) [inline, virtual]

Set the size of the box.

Parameters

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

Reimplemented from **gazebo::physics::BoxShape** (p. 230).

References gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::equal(), gazebo::physics::DARTCollision::GetDARTBodyNode(), gzerr, gzwarn, NULL, gazebo::physics::BoxShape::SetSize(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

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

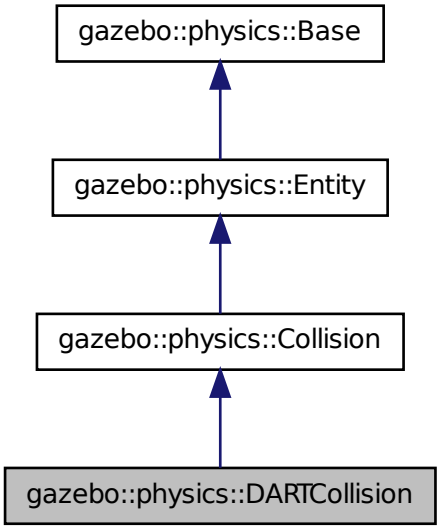
- **DARTBoxShape.hh**

10.52 gazebo::physics::DARTCollision Class Reference

Base (p. 201) class for all DART collisions.

`#include <DARTCollision.hh>`

Inheritance diagram for gazebo::physics::DARTCollision:



Public Member Functions

- **DARTCollision** (**LinkPtr** _parent)

  Constructor.
- virtual ~**DARTCollision** ()

  Destructor.
- virtual void **Fini** ()

  Finalize the collision.
- virtual **math::Box GetBoundingBox** () const

  Get the bounding box for this collision.
- virtual unsigned int **GetCategoryBits** () const

  Get the category bits, used during collision detection.
- virtual unsigned int **GetCollideBits** () const

  Get the collide bits, used during collision detection.
- dart::dynamics::BodyNode * **GetDARTBodyNode** () const

  Get DART body node.

- dart::dynamics::Shape * **GetDARTCollisionShape** () const

  Get DART collision shape.
- virtual void **Init** ()

  Initialize the collision.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the collision.
- virtual void **OnPoseChange** ()

  This function is called when the entity's (or one of its parents) pose of the parent has changed.
- virtual void **SetCategoryBits** (unsigned int _bits)

  Set the category bits, used during collision detection.
- virtual void **SetCollideBits** (unsigned int _bits)

  Set the collide bits, used during collision detection.
- void **SetDARTCollisionShape** (dart::dynamics::Shape *_shape, bool _placeable=true)

  Set DART collision shape.

10.52.1 Detailed Description

Base (p. 201) class for all DART collisions.

10.52.2 Constructor & Destructor Documentation

10.52.2.1 gazebo::physics::DARTCollision::DARTCollision (LinkPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_link*	Parent **Link** (p. 739)

10.52.2.2 virtual gazebo::physics::DARTCollision::~DARTCollision ( ) [virtual]

Destructor.

10.52.3 Member Function Documentation

10.52.3.1 virtual void gazebo::physics::DARTCollision::Fini ( ) [virtual]

Finalize the collision.

Reimplemented from **gazebo::physics::Collision** (p. 299).

10.52.3.2 virtual math::Box gazebo::physics::DARTCollision::GetBoundingBox ( ) const [virtual]

Get the bounding box for this collision.

Returns

The bounding box.

Implements **gazebo::physics::Collision** (p. 299).

10.52.3.3 virtual unsigned int gazebo::physics::DARTCollision::GetCategoryBits ( ) const [virtual]

Get the category bits, used during collision detection.

Returns

The bits

10.52.3.4 virtual unsigned int gazebo::physics::DARTCollision::GetCollideBits ( ) const [virtual]

Get the collide bits, used during collision detection.

Returns

The bits

10.52.3.5 dart::dynamics::BodyNode* gazebo::physics::DARTCollision::GetDARTBodyNode ( ) const

Get DART body node.

Returns

Pointer to the dart BodyNode.

Referenced by gazebo::physics::DARTPlaneShape::CreatePlane(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::DARTCylinderShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

10.52.3.6 dart::dynamics::Shape* gazebo::physics::DARTCollision::GetDARTCollisionShape ( ) const

Get DART collision shape.

10.52.3.7 virtual void gazebo::physics::DARTCollision::Init ( ) [virtual]

Initialize the collision.

Reimplemented from **gazebo::physics::Collision** (p. 303).

10.52.3.8 virtual void gazebo::physics::DARTCollision::Load (sdf::ElementPtr _sdf) [virtual]

Load the collision.

Parameters

in	*_sdf*	SDF to load from.

Reimplemented from **gazebo::physics::Collision** (p. 304).

10.52.3.9 virtual void gazebo::physics::DARTCollision::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. 509).

10.52.3.10 virtual void gazebo::physics::DARTCollision::SetCategoryBits (unsigned int _bits) [virtual]

Set the category bits, used during collision detection.

Parameters

in	*_bits*	The bits to set.

Implements **gazebo::physics::Collision** (p. 304).

10.52.3.11 virtual void gazebo::physics::DARTCollision::SetCollideBits (unsigned int *_bits*) [virtual]

Set the collide bits, used during collision detection.

Parameters

in	*_bits*	The bits to set.

Implements **gazebo::physics::Collision** (p. 304).

10.52.3.12 void gazebo::physics::DARTCollision::SetDARTCollisionShape (dart::dynamics::Shape * *_shape*, bool *_placeable* = true)

Set DART collision shape.

Parameters

in	*_shape*	DART **Collision** (p. 295) shape
in	*_placeable*	True to make the object movable.

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

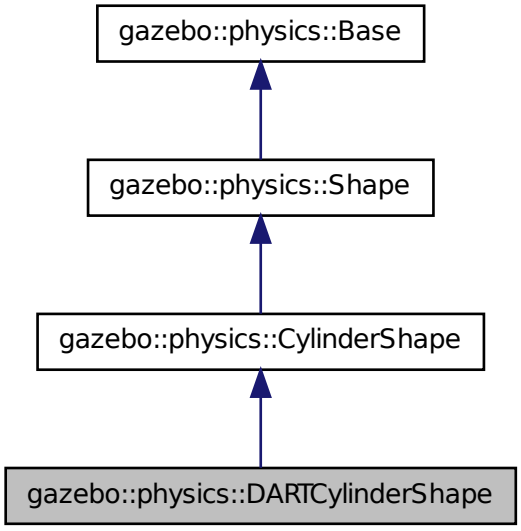
- **DARTCollision.hh**

10.53 gazebo::physics::DARTCylinderShape Class Reference

DART cylinder shape.

`#include <DARTCylinderShape.hh>`

Inheritance diagram for gazebo::physics::DARTCylinderShape:



Public Member Functions

- **DARTCylinderShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**DARTCylinderShape** ()

  Destructor.
- void **SetSize** (double _radius, double _length)

  Set the size of the cylinder.

10.53.1 Detailed Description

DART cylinder shape.

10.53.2 Constructor & Destructor Documentation

10.53.2.1 gazebo::physics::DARTCylinderShape::DARTCylinderShape (CollisionPtr *_parent*) [inline, explicit]

Constructor.

Parameters

in	*_parent*	**Collision** (p. 295) parent.

10.53.2.2 virtual gazebo::physics::DARTCylinderShape::~DARTCylinderShape ( ) [inline, virtual]

Destructor.

10.53.3 Member Function Documentation

10.53.3.1 void gazebo::physics::DARTCylinderShape::SetSize (double *_radius*, double *_length*) [inline, virtual]

Set the size of the cylinder.

Parameters

in	*_radius*	New radius.
in	*_lenght*	New length.

Reimplemented from **gazebo::physics::CylinderShape** (p. 373).

References gazebo::math::equal(), gazebo::physics::DARTCollision::GetDARTBodyNode(), gzerr, gzwarn, NULL, and gazebo::physics::CylinderShape::SetSize().

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

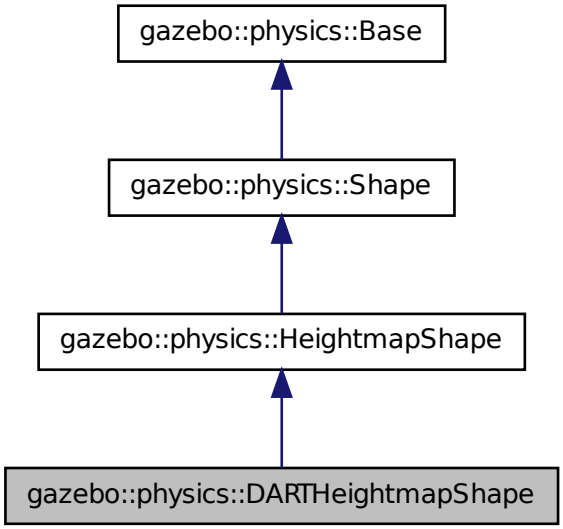
- **DARTCylinderShape.hh**

10.54 gazebo::physics::DARTHeightmapShape Class Reference

DART Height map collision.

`#include <DARTHeightmapShape.hh>`

Inheritance diagram for gazebo::physics::DARTHeightmapShape:



Public Member Functions

- **DARTHeightmapShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**DARTHeightmapShape** ()

  Destructor.
- virtual void **Init** ()

  Initialize the heightmap.

10.54.1 Detailed Description

DART Height map collision.

10.54.2 Constructor & Destructor Documentation

10.54.2.1 gazebo::physics::DARTHeightmapShape::DARTHeightmapShape (CollisionPtr _parent)

Constructor.

Parameters

in	*_parent*	**Collision** (p. 295) parent.

10.54.2.2 virtual gazebo::physics::DARTHeightmapShape::~DARTHeightmapShape ( ) [virtual]

Destructor.

10.54.3 Member Function Documentation

10.54.3.1 virtual void gazebo::physics::DARTHeightmapShape::Init ( ) [virtual]

Initialize the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 632).

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

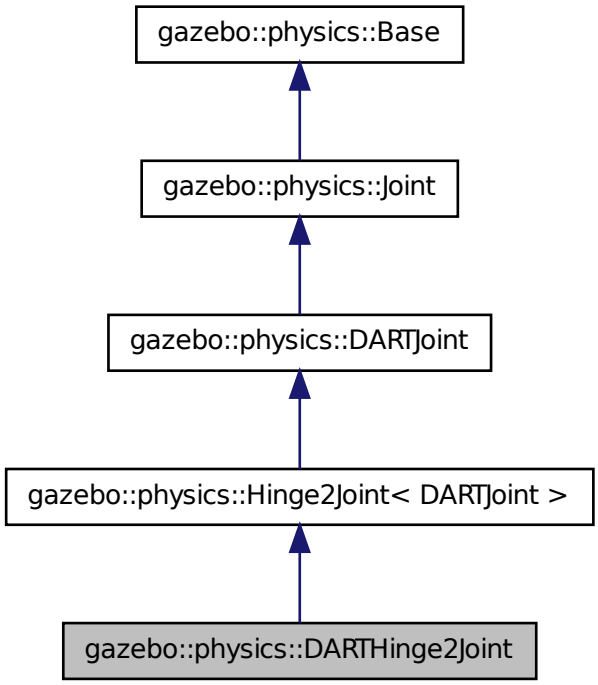
- **DARTHeightmapShape.hh**

10.55 gazebo::physics::DARTHinge2Joint Class Reference

A two axis hinge joint.

`#include <DARTHinge2Joint.hh>`

Inheritance diagram for gazebo::physics::DARTHinge2Joint:



Public Member Functions

- **DARTHinge2Joint** (**BasePtr** _parent)

  Constructor.
- virtual ~**DARTHinge2Joint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const

  Get the anchor point.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.

- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)
- virtual void **Init** ()

  Initialize a joint.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the joint.
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _force)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual void **SetVelocity** (unsigned int _index, double _vel)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _effort)

  *Set the force applied to this **physics::Joint** (p. 669).*

Protected Attributes

- dart::dynamics::UniversalJoint * **dtUniveralJoint**

  Universal joint of DART.

10.55.1 Detailed Description

A two axis hinge joint.

10.55.2 Constructor & Destructor Documentation

10.55.2.1 gazebo::physics::DARTHinge2Joint::DARTHinge2Joint (BasePtr *_parent*)

Constructor.

Parameters

in	*_parent*	Parent of the **Joint** (p. 669)

10.55.2.2 virtual gazebo::physics::DARTHinge2Joint::~DARTHinge2Joint ( ) [virtual]

Destructor.

10.55.3 Member Function Documentation

10.55.3.1 virtual math::Vector3 gazebo::physics::DARTHinge2Joint::GetAnchor (unsigned int *_index*) const [virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 679).

10.55.3.2 virtual math::Angle gazebo::physics::DARTHinge2Joint::GetAngleImpl (unsigned int *_index*) const [virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.55.3.3 virtual math::Vector3 gazebo::physics::DARTHinge2Joint::GetGlobalAxis (unsigned int *_index*) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.55.3.4 virtual double gazebo::physics::DARTHinge2Joint::GetMaxForce (unsigned int *_index*) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.55.3.5 virtual double gazebo::physics::DARTHinge2Joint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.55.3.6 virtual void gazebo::physics::DARTHinge2Joint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::DARTJoint** (p. 413).

10.55.3.7 virtual void gazebo::physics::DARTHinge2Joint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the joint.

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::Hinge2Joint< DARTJoint >** (p. 635).

10.55.3.8 virtual void gazebo::physics::DARTHinge2Joint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 695).

10.55.3.9 virtual void gazebo::physics::DARTHinge2Joint::SetForceImpl (unsigned int *_index*, double *_force*) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value.

Implements **gazebo::physics::DARTJoint** (p. 415).

10.55.3.10 virtual void gazebo::physics::DARTHinge2Joint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.55.3.11 virtual void gazebo::physics::DARTHinge2Joint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

10.55.4 Member Data Documentation

10.55.4.1 dart::dynamics::UniversalJoint* gazebo::physics::DARTHinge2Joint::dtUniveralJoint [protected]

Universal joint of DART.

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

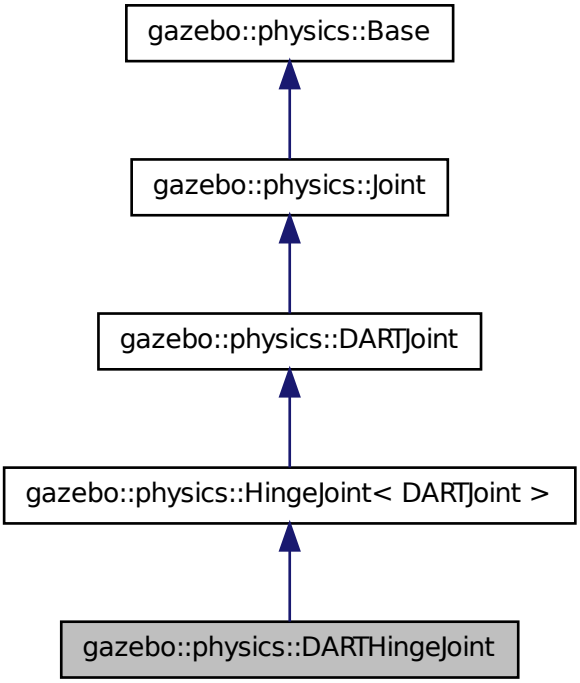
- **DARTHinge2Joint.hh**

10.56 gazebo::physics::DARTHingeJoint Class Reference

A single axis hinge joint.

`#include <DARTHingeJoint.hh>`

Inheritance diagram for gazebo::physics::DARTHingeJoint:



Public Member Functions

- **DARTHingeJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**DARTHingeJoint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const

  Get the anchor point.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.

- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.
- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity*** (p. *701).*
- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)
- virtual void **Init** ()

  Initialize joint.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load joint.
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _force)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity*** (p. *701).*
- virtual void **SetVelocity** (unsigned int _index, double _vel)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _effort)

  *Set the force applied to this **physics::Joint*** (p. *669).*

Protected Attributes

- dart::dynamics::RevoluteJoint * **dtRevoluteJoint**

  Revolute joint of DART.

10.56.1 Detailed Description

A single axis hinge joint.

10.56.2 Constructor & Destructor Documentation

10.56.2.1 gazebo::physics::DARTHingeJoint::DARTHingeJoint (BasePtr *_parent*)

Constructor.

Parameters

in	*_parent*	Parent of the **Joint** (p. 669)

10.56.2.2 virtual gazebo::physics::DARTHingeJoint::~DARTHingeJoint ( ) [virtual]

Destructor.

10.56.3 Member Function Documentation

10.56.3.1 virtual math::Vector3 gazebo::physics::DARTHingeJoint::GetAnchor (unsigned int _index) const [virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 679).

10.56.3.2 virtual math::Angle gazebo::physics::DARTHingeJoint::GetAngleImpl (unsigned int _index) const [virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.56.3.3 virtual math::Vector3 gazebo::physics::DARTHingeJoint::GetGlobalAxis (unsigned int _index) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.56.3.4 virtual double gazebo::physics::DARTHingeJoint::GetMaxForce (unsigned int *_index*) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.56.3.5 virtual double gazebo::physics::DARTHingeJoint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.56.3.6 virtual void gazebo::physics::DARTHingeJoint::Init ( ) [virtual]

Initialize joint.

Reimplemented from **gazebo::physics::HingeJoint< DARTJoint >** (p. 637).

10.56.3.7 virtual void gazebo::physics::DARTHingeJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load joint.

Parameters

in	*_sdf*	Pointer to SDF element

Reimplemented from **gazebo::physics::HingeJoint< DARTJoint >** (p. 637).

10.56.3.8 virtual void gazebo::physics::DARTHingeJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 695).

10.56.3.9 virtual void gazebo::physics::DARTHingeJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value.

Implements **gazebo::physics::DARTJoint** (p. 415).

10.56.3.10 virtual void gazebo::physics::DARTHingeJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.56.3.11 virtual void gazebo::physics::DARTHingeJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

10.56.4 Member Data Documentation

10.56.4.1 dart::dynamics::RevoluteJoint* gazebo::physics::DARTHingeJoint::dtRevoluteJoint [protected]

Revolute joint of DART.

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

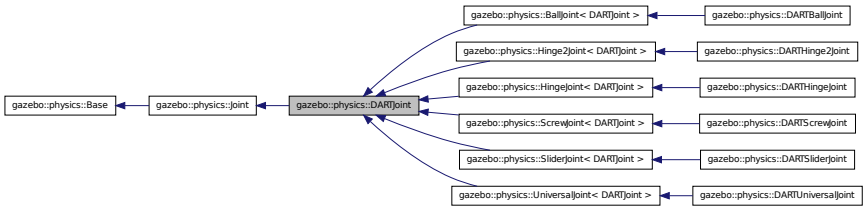
- **DARTHingeJoint.hh**

10.57 gazebo::physics::DARTJoint Class Reference

DART joint interface.

`#include <DARTJoint.hh>`

Inheritance diagram for gazebo::physics::DARTJoint:



Public Member Functions

- **DARTJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**DARTJoint** ()

  Destructor.
- virtual void **ApplyDamping** ()
- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const

  Determines of the two bodies are connected by a joint.
- virtual void **Attach** (**LinkPtr** _parent, **LinkPtr** _child)

  Attach the two bodies with this joint.
- virtual void **Detach** ()

  Detach this joint from all links.
- virtual unsigned int **GetAngleCount** () const

  Get the angle count.
- dart::dynamics::Joint * **GetDARTJoint** ()

  Get DART joint pointer.
- **DARTModelPtr GetDARTModel** () const

  Get DART model pointer.
- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench GetForceTorque** (unsigned int _index)

  get internal force and torque values at a joint.
- virtual **math::Angle GetHighStop** (unsigned int _index)

  Get the high stop of an axis(index).
- virtual **LinkPtr GetJointLink** (unsigned int _index) const

  Get the link to which the joint is attached according the _index.
- virtual **math::Vector3 GetLinkForce** (unsigned int _index) const

*Get the forces applied to the center of mass of a **physics::Link** (p. 739) due to the existence of this **Joint** (p. 669).*

- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const

  *Get the torque applied to the center of mass of a **physics::Link** (p. 739) due to the existence of this **Joint** (p. 669).*

- virtual **math::Angle GetLowStop** (unsigned int _index)

  Get the low stop of an axis(index).

- virtual double **GetParam** (const std::string &_key, unsigned int _index)

  Get a non-generic parameter for the joint.

- virtual void **Init** ()

  Initialize a joint.

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load **physics::Joint** (p. 669) from a SDF sdf::Element.*

- virtual void **Reset** ()

  Reset the joint.

- virtual void **SetAnchor** (unsigned int, const **gazebo::math::Vector3** &)

  Set the anchor point.

- virtual void **SetDamping** (unsigned int _index, double _damping)

  Set the joint damping.

- virtual void **SetForce** (unsigned int _index, double _force)

  *Set the force applied to this **physics::Joint** (p. 669).*

- virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).

- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).

- virtual bool **SetParam** (const std::string &_key, unsigned int _index, const boost::any &_value)

  Set a non-generic parameter for the joint.

- virtual void **SetStiffness** (unsigned int _index, const double _stiffness)

  Set the joint spring stiffness.

- virtual void **SetStiffnessDamping** (unsigned int _index, double _stiffness, double _damping, double _reference=0)

  Set the joint spring stiffness.

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _force)=0

  *Set the force applied to this **physics::Joint** (p. 669).*

Protected Attributes

- **DARTPhysicsPtr dartPhysicsEngine**

  DARTPhysics (p. *438*) *engine pointer.*
- dart::dynamics::BodyNode * **dtChildBodyNode**

  DART child body node pointer.
- dart::dynamics::Joint * **dtJoint**

  DART joint pointer.

10.57.1 Detailed Description

DART joint interface.

10.57.2 Constructor & Destructor Documentation

10.57.2.1 gazebo::physics::DARTJoint::DARTJoint (BasePtr *_parent*)

Constructor.

Parameters

in	*_parent*	Parent of the **Joint** (p. 669).

10.57.2.2 virtual gazebo::physics::DARTJoint::~DARTJoint ( ) [virtual]

Destructor.

10.57.3 Member Function Documentation

10.57.3.1 virtual void gazebo::physics::DARTJoint::ApplyDamping ( ) [virtual]

10.57.3.2 virtual bool gazebo::physics::DARTJoint::AreConnected (LinkPtr *_one*, LinkPtr *_two*) const [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.

Implements **gazebo::physics::Joint** (p. 676).

10.57.3.3 virtual void gazebo::physics::DARTJoint::Attach (LinkPtr *_parent,* LinkPtr *_child*) [virtual]

Attach the two bodies with this joint.

Parameters

in	*_parent*	Parent link.
in	*_child*	Child link.

Reimplemented from **gazebo::physics::Joint** (p. 677).

10.57.3.4 virtual void gazebo::physics::DARTJoint::Detach ( ) [virtual]

Detach this joint from all links.

Reimplemented from **gazebo::physics::Joint** (p. 678).

10.57.3.5 virtual unsigned int gazebo::physics::DARTJoint::GetAngleCount ( ) const [virtual]

Get the angle count.

Returns

The number of DOF for the joint.

Implements **gazebo::physics::Joint** (p. 680).

Reimplemented in **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1405), **gazebo::physics::BallJoint< DARTJoint >** (p. 200), **gazebo::physics::SliderJoint< DARTJoint >** (p. 1295), **gazebo::physics::Hinge2Joint< DARTJoint >** (p. 635), **gazebo::physics::HingeJoint< DARTJoint >** (p. 637), and **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1119).

10.57.3.6 dart::dynamics::Joint* gazebo::physics::DARTJoint::GetDARTJoint ( )

Get DART joint pointer.

Returns

A pointer to the DART joint.

10.57.3.7 DARTModelPtr gazebo::physics::DARTJoint::GetDARTModel ( ) const

Get DART model pointer.

Returns

A pointer to the DART model.

10.57.3.8 virtual double gazebo::physics::DARTJoint::GetForce (unsigned int *_index*) [virtual]

Todo : not yet implemented. Get external forces applied at this **Joint** (p. 669). Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The force applied to an axis.

Reimplemented from **gazebo::physics::Joint** (p. 683).

10.57.3.9 virtual JointWrench gazebo::physics::DARTJoint::GetForceTorque (unsigned int *_index*) [virtual]

get internal force and torque values at a joint.

The force and torque values are returned in a **JointWrench** (p. 721) data structure. - Where **JointWrench.body1Force** (p. 723) contains the force applied by the parent **Link** (p. 739) on the **Joint** (p. 669) specified in the parent **Link** (p. 739) frame, and **JointWrench.body2Force** (p. 723) contains the force applied by the child **Link** (p. 739) on the **Joint** (p. 669) specified in the child **Link** (p. 739) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 669) on the Links.

FIXME TODO: change name of this function to something like: GetNegatedForceTorqueInLinkFrame and make GetForceTorque call return non-negated reaction forces in perspective **Link** (p. 739) frames.

Note that for ODE you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

Parameters

in	*_index*	Not used right now

Returns

The force and torque at the joint, see above for details on conventions.

Implements **gazebo::physics::Joint** (p. 683).

10.57.3.10 virtual math::Angle gazebo::physics::DARTJoint::GetHighStop (unsigned int *_index*) [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.

Implements **gazebo::physics::Joint** (p. 684).

Reimplemented in **gazebo::physics::DARTScrewJoint** (p. 454), and **gazebo::physics::DARTBallJoint** (p. 378).

10.57.3.11 virtual LinkPtr gazebo::physics::DARTJoint::GetJointLink (unsigned int *_index*) const [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.

Implements **gazebo::physics::Joint** (p. 686).

10.57.3.12 virtual math::Vector3 gazebo::physics::DARTJoint::GetLinkForce (unsigned int _index) const [virtual]

Get the forces applied to the center of mass of a **physics::Link** (p. 739) due to the existence of this **Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*index*	The index of the link(0 or 1).

Returns

Force applied to the link.

Implements **gazebo::physics::Joint** (p. 686).

10.57.3.13 virtual math::Vector3 gazebo::physics::DARTJoint::GetLinkTorque (unsigned int _index) const [virtual]

Get the torque applied to the center of mass of a **physics::Link** (p. 739) due to the existence of this **Joint** (p. 669).

Note that the unit of torque should be consistent with the rest of the simulation scales.

Parameters

in	*index*	The index of the link(0 or 1)

Returns

Torque applied to the link.

Implements **gazebo::physics::Joint** (p. 687).

10.57.3.14 virtual math::Angle gazebo::physics::DARTJoint::GetLowStop (unsigned int _index) [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.

Implements **gazebo::physics::Joint** (p. 688).

Reimplemented in **gazebo::physics::DARTScrewJoint** (p. 454), and **gazebo::physics::DARTBallJoint** (p. 378).

10.57.3.15 virtual double gazebo::physics::DARTJoint::GetParam (const std::string & *_key*, unsigned int *_index*) [virtual]

Get a non-generic parameter for the joint.

Parameters

in	*_key*	String key.
in	*_index*	Index of the axis.

Implements **gazebo::physics::Joint** (p. 689).

Reimplemented in **gazebo::physics::DARTScrewJoint** (p. 455).

10.57.3.16 virtual void gazebo::physics::DARTJoint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Joint** (p. 693).

Reimplemented in **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1120), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1406), **gazebo::physics::BallJoint< DARTJoint >** (p. 200), **gazebo::physics::HingeJoint< DARTJoint >** (p. 637), **gazebo::physics::DARTScrewJoint** (p. 456), **gazebo::physics::DARTHinge2Joint** (p. 397), **gazebo::physics::DARTHingeJoint** (p. 403), **gazebo::physics::DARTBallJoint** (p. 379), **gazebo::physics::DARTSliderJoint** (p. 463), and **gazebo::physics::DARTUniversalJoint** (p. 472).

10.57.3.17 virtual void gazebo::physics::DARTJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load **physics::Joint** (p. 669) from a SDF sdf::Element.

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::Joint** (p. 693).

Reimplemented in **gazebo::physics::BallJoint< DARTJoint >** (p. 200), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1406), **gazebo::physics::Hinge2Joint< DARTJoint >** (p. 635), **gazebo::physics::HingeJoint< DARTJoint >** (p. 637), **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1120), **gazebo::physics::SliderJoint< DARTJoint >** (p. 1295), **gazebo::physics::DARTHinge2Joint** (p. 398), **gazebo::physics::DARTHingeJoint** (p. 404), **gazebo::physics::DARTBallJoint** (p. 380), **gazebo::physics::DARTScrewJoint** (p. 457), **gazebo::physics::DARTSliderJoint** (p. 464), and **gazebo::physics::DARTUniversalJoint** (p. 473).

10.57.3.18 virtual void gazebo::physics::DARTJoint::Reset ( ) [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Joint** (p. 694).

10.57.3.19 virtual void gazebo::physics::DARTJoint::SetAnchor (unsigned *int,* const gazebo::math::Vector3 &) [virtual]

Set the anchor point.

Implements **gazebo::physics::Joint** (p. 694).

Reimplemented in **gazebo::physics::DARTScrewJoint** (p. 457).

10.57.3.20 virtual void gazebo::physics::DARTJoint::SetDamping (unsigned int *_index,* double *_damping*) [virtual]

Set the joint damping.

Parameters

in	*_index*	Index of the axis to set, currently ignored, to be implemented.
in	*_damping*	Damping value for the axis.

Implements **gazebo::physics::Joint** (p. 695).

10.57.3.21 virtual void gazebo::physics::DARTJoint::SetForce (unsigned int *_index*, double *_effort*) [virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForce to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)). Forces are truncated by effortLimit before applied.

Parameters

in	*_index*	Index of the axis.
in	*_effort*	Force value.

Implements **gazebo::physics::Joint** (p. 696).

10.57.3.22 virtual void gazebo::physics::DARTJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected, pure virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value.

Implemented in **gazebo::physics::DARTScrewJoint** (p. 457), **gazebo::physics::DARTHinge2Joint** (p. 398), **gazebo::physics::DARTHingeJoint** (p. 404), **gazebo::physics::DARTSliderJoint** (p. 464), **gazebo::physics::DARTUniversalJoint** (p. 473), and **gazebo::physics::DARTBallJoint** (p. 380).

10.57.3.23 virtual bool gazebo::physics::DARTJoint::SetHighStop (unsigned 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.

Reimplemented from **gazebo::physics::Joint** (p. 696).

Reimplemented in **gazebo::physics::DARTBallJoint** (p. 381).

10.57.3.24 virtual bool gazebo::physics::DARTJoint::SetLowStop (unsigned 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.

Reimplemented from **gazebo::physics::Joint** (p. 697).

Reimplemented in **gazebo::physics::DARTBallJoint** (p. 381).

10.57.3.25 virtual bool gazebo::physics::DARTJoint::SetParam (const std::string & *_key*, unsigned int *_index*, const boost::any & *_value*) [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.

Implements **gazebo::physics::Joint** (p. 698).

10.57.3.26 virtual void gazebo::physics::DARTJoint::SetStiffness (unsigned int *_index*, const double *_stiffness*) [virtual]

Set the joint spring stiffness.

Parameters

in	*_index*	Index of the axis to set, currently ignored, to be implemented.
in	*_stiffness*	Spring stiffness value for the axis. : rename to SetSpringStiffness()

Implements **gazebo::physics::Joint** (p. 699).

10.57.3.27 virtual void gazebo::physics::DARTJoint::SetStiffnessDamping (unsigned int *_index*, double *_stiffness*, double *_damping*, double *_reference* = 0) [virtual]

Set the joint spring stiffness.

Parameters

in	*_index*	Index of the axis to set, currently ignored, to be implemented.
in	*_stiffness*	Stiffness value for the axis.
in	*_reference*	Spring zero load reference position. : rename to SetSpringStiffnessDamping()

Implements **gazebo::physics::Joint** (p. 700).

10.57.4 Member Data Documentation

10.57.4.1 DARTPhysicsPtr gazebo::physics::DARTJoint::dartPhysicsEngine [protected]

DARTPhysics (p. 438) engine pointer.

10.57.4.2 dart::dynamics::BodyNode* gazebo::physics::DARTJoint::dtChildBodyNode [protected]

DART child body node pointer.

10.57.4.3 dart::dynamics::Joint* gazebo::physics::DARTJoint::dtJoint [protected]

DART joint pointer.

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

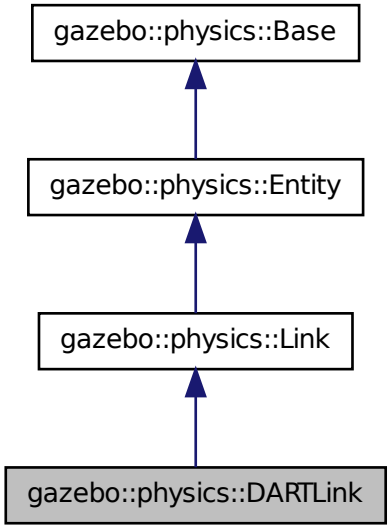
- **DARTJoint.hh**

10.58 gazebo::physics::DARTLink Class Reference

DART **Link** (p. 739) class.

`#include <DARTLink.hh>`

Inheritance diagram for gazebo::physics::DARTLink:



Public Member Functions

- **DARTLink** (**EntityPtr** _parent)

  Constructor.
- virtual ~**DARTLink** ()

  Destructor.
- void **AddDARTChildJoint** (**DARTJointPtr** _dartChildJoint)

  Set child joint of this link.

- virtual void **AddForce** (const **math::Vector3** &_force)

  Add a force to the body.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)

  Add a force to the body at position expressed to the body's own frame of reference.
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)

  Add a force to the body using a global position.
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)

  Add a force to the body, components are relative to the body's own frame of reference.
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)

  Add a torque to the body, components are relative to the body's own frame of reference.
- virtual void **AddTorque** (const **math::Vector3** &_torque)

  Add a torque to the body.
- virtual void **Fini** ()

  Finalize the body.
- dart::dynamics::BodyNode * **GetDARTBodyNode** () const

  Get pointer to DART BodyNode associated with this link.
- **DARTModelPtr GetDARTModel** () const

  *Get pointer to DART **Model** (*p. *846) associated with this link.*
- **DARTPhysicsPtr GetDARTPhysics** (void) const

  Get pointer to DART Physics engine associated with this link.
- dart::simulation::World * **GetDARTWorld** (void) const

  *Get pointer to DART **World** (*p. *1529) associated with this link.*
- virtual bool **GetEnabled** () const

  Get whether this body is enabled in the physics engine.
- virtual bool **GetGravityMode** () const

  Get the gravity mode.
- virtual bool **GetKinematic** () const

  Implement this function.
- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the entity in the world frame.
- virtual **math::Vector3 GetWorldCoGLinearVel** () const

  Get the linear velocity at the body's center of gravity in the world frame.
- virtual **math::Vector3 GetWorldForce** () const

  Get the force applied to the body in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset=**math::Vector3**(0, 0, 0)) const

  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

  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

  Get the torque applied to the body in the world frame.
- virtual void **Init** ()

  Initialize the body.
- virtual void **Load** (sdf::ElementPtr _ptr)

  Load the body based on an SDF element.
- virtual void **OnPoseChange** ()

  This function is called when the entity's (or one of its parents) pose of the parent has changed.
- virtual void **SetAngularDamping** (double _damping)

  Set the angular damping factor.
- virtual void **SetAngularVel** (const **math::Vector3** &_vel)

  Set the angular velocity of the body.
- virtual void **SetAutoDisable** (bool _disable)

  Allow the link to auto disable.
- void **SetDARTParentJoint** (**DARTJointPtr** _dartParentJoint)

  Set parent joint of this link.
- virtual void **SetEnabled** (bool _enable) const

  Set whether this body is enabled.
- virtual void **SetForce** (const **math::Vector3** &_force)

  Set the force applied to the body.
- virtual void **SetGravityMode** (bool _mode)

  Set whether gravity affects this body.
- virtual void **SetKinematic** (const bool &_state)

  Implement this function.
- virtual void **SetLinearDamping** (double _damping)

  Set the linear damping factor.
- virtual void **SetLinearVel** (const **math::Vector3** &_vel)

  Set the linear velocity of the body.
- virtual void **SetLinkStatic** (bool _static)

  Freeze link to ground (inertial frame).
- virtual void **SetSelfCollide** (bool _collide)

  Set whether this body will collide with others in the model.
- virtual void **SetTorque** (const **math::Vector3** &_torque)

  Set the torque applied to the body.
- void **updateDirtyPoseFromDARTTransformation** ()

  *Store DART Transformation to **Entity::dirtyPose** (p. 513) and add this link to **World::dirtyPoses** (p. 1543) so that World::Update() trigger **Entity::SetWorldPose()** (p. 512) for this link.*

10.58.1 Detailed Description

DART **Link** (p. 739) class.

10.58.2 Constructor & Destructor Documentation

10.58.2.1 gazebo::physics::DARTLink::DARTLink (EntityPtr *_parent*) [explicit]

Constructor.

10.58.2.2 virtual gazebo::physics::DARTLink::~DARTLink ( ) [virtual]

Destructor.

10.58.3 Member Function Documentation

10.58.3.1 void gazebo::physics::DARTLink::AddDARTChildJoint (DARTJointPtr *_dartChildJoint*)

Set child joint of this link.

Parameters

in	*_dartChildJoint*	Pointer to the child joint.

10.58.3.2 virtual void gazebo::physics::DARTLink::AddForce (const math::Vector3 & *_force*) [virtual]

Add a force to the body.

Parameters

in	*_force*	Force to add.

Implements **gazebo::physics::Link** (p. 747).

10.58.3.3 virtual void gazebo::physics::DARTLink::AddForceAtRelativePosition (const math::Vector3 & _force, const math::Vector3 & _relPos) [virtual]

Add a force to the body at position expressed to the body's own frame of reference.

Parameters

in	*_force*	Force to add.
in	*_relPos*	Position on the link to add the force.

Implements **gazebo::physics::Link** (p. 747).

10.58.3.4 virtual void gazebo::physics::DARTLink::AddForceAtWorldPosition (const math::Vector3 & _force, const math::Vector3 & _pos) [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.

Implements **gazebo::physics::Link** (p. 747).

10.58.3.5 virtual void gazebo::physics::DARTLink::AddRelativeForce (const math::Vector3 & _force) [virtual]

Add a force to the body, components are relative to the body's own frame of reference.

Parameters

in	*_force*	Force to add.

Implements **gazebo::physics::Link** (p. 748).

10.58.3.6 virtual void gazebo::physics::DARTLink::AddRelativeTorque (const math::Vector3 & _torque) [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.

Implements **gazebo::physics::Link** (p. 748).

10.58.3.7 virtual void gazebo::physics::DARTLink::AddTorque (const math::Vector3 & *_torque*) [virtual]

Add a torque to the body.

Parameters

in	*_torque*	Torque value to add to the link.

Implements **gazebo::physics::Link** (p. 748).

10.58.3.8 virtual void gazebo::physics::DARTLink::Fini ( ) [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Link** (p. 750).

10.58.3.9 dart::dynamics::BodyNode* gazebo::physics::DARTLink::GetDARTBodyNode ( ) const

Get pointer to DART BodyNode associated with this link.

Returns

Pointer to DART BodyNode.

10.58.3.10 DARTModelPtr gazebo::physics::DARTLink::GetDARTModel ( ) const

Get pointer to DART **Model** (p. 846) associated with this link.

Returns

Pointer to the DART **Model** (p. 846).

10.58.3.11 DARTPhysicsPtr gazebo::physics::DARTLink::GetDARTPhysics (void) const

Get pointer to DART Physics engine associated with this link.

Returns

Pointer to the DART Physics engine.

10.58.3.12 dart::simulation::World* gazebo::physics::DARTLink::GetDARTWorld (void) const

Get pointer to DART **World** (p. 1529) associated with this link.

Returns

Pointer to the DART **World** (p. 1529).

10.58.3.13 virtual bool gazebo::physics::DARTLink::GetEnabled ( ) const [virtual]

Get whether this body is enabled in the physics engine.

Returns

True if the link is enabled.

Implements **gazebo::physics::Link** (p. 753).

10.58.3.14 virtual bool gazebo::physics::DARTLink::GetGravityMode ( ) const [virtual]

Get the gravity mode.

Returns

True if gravity is enabled.

Implements **gazebo::physics::Link** (p. 753).

10.58.3.15 virtual bool gazebo::physics::DARTLink::GetKinematic ( ) const [virtual]

Implement this function.

Get whether this body is in the kinematic state.

Returns

True if the link is kinematic only.

Reimplemented from **gazebo::physics::Link** (p. 753).

10.58.3.16 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the entity in the world frame.

Returns

A **math::Vector3** (p. 1440) for the velocity.

Reimplemented from **gazebo::physics::Entity** (p. 508).

10.58.3.17 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldCoGLinearVel ( ) const [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.

Implements **gazebo::physics::Link** (p. 757).

10.58.3.18 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldForce ( ) const [virtual]

Get the force applied to the body in the world frame.

Returns

Force applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 758).

10.58.3.19 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldLinearVel (const math::Vector3 & *_offset* = math::Vector3(0, 0, 0)) const [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. 739) frame will be returned.

Parameters

in	*_offset*	Offset of the point from the origin of the **Link** (p. 739) frame, expressed in the body-fixed frame.

Returns

Linear velocity of the point on the body

Implements **gazebo::physics::Link** (p. 759).

10.58.3.20 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldLinearVel (const math::Vector3 & *_offset*, const math::Quaternion & *_q*) const [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.

Implements **gazebo::physics::Link** (p. 760).

10.58.3.21 virtual math::Vector3 gazebo::physics::DARTLink::GetWorldTorque ( ) const [virtual]

Get the torque applied to the body in the world frame.

Returns

Torque applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 760).

10.58.3.22 virtual void gazebo::physics::DARTLink::Init ( ) [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Link** (p. 760).

10.58.3.23 virtual void gazebo::physics::DARTLink::Load (sdf::ElementPtr _sdf) [virtual]

Load the body based on an SDF element.

Parameters

in	*_sdf*	SDF parameters.

Reimplemented from **gazebo::physics::Link** (p. 761).

10.58.3.24 virtual void gazebo::physics::DARTLink::OnPoseChange ( ) [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Reimplemented from **gazebo::physics::Link** (p. 761).

10.58.3.25 virtual void gazebo::physics::DARTLink::SetAngularDamping (double _damping) [virtual]

Set the angular damping factor.

Parameters

in	*_damping*	Angular damping factor.

Implements **gazebo::physics::Link** (p. 763).

10.58.3.26 virtual void gazebo::physics::DARTLink::SetAngularVel (const math::Vector3 & _vel) [virtual]

Set the angular velocity of the body.

Parameters

in	*_vel*	Angular velocity.

Implements **gazebo::physics::Link** (p. 763).

10.58.3.27 virtual void gazebo::physics::DARTLink::SetAutoDisable (bool *_disable*) [virtual]

Allow the link to auto disable.

Parameters

in	*_disable*	If true, the link is allowed to auto disable.

Implements **gazebo::physics::Link** (p. 763).

10.58.3.28 void gazebo::physics::DARTLink::SetDARTParentJoint (DARTJointPtr *_dartParentJoint*)

Set parent joint of this link.

Parameters

in	*_dartParentJoint*	Pointer to the parent joint.

10.58.3.29 virtual void gazebo::physics::DARTLink::SetEnabled (bool *_enable*) const [virtual]

Set whether this body is enabled.

Parameters

in	*_enable*	True to enable the link in the physics engine.

Implements **gazebo::physics::Link** (p. 764).

10.58.3.30 virtual void gazebo::physics::DARTLink::SetForce (const math::Vector3 & *_force*) [virtual]

Set the force applied to the body.

Parameters

in	*_force*	Force value.

Implements **gazebo::physics::Link** (p. 764).

10.58.3.31 virtual void gazebo::physics::DARTLink::SetGravityMode (bool *_mode*) [virtual]

Set whether gravity affects this body.

Parameters

in	*_mode*	True to enable gravity.

Implements **gazebo::physics::Link** (p. 765).

10.58.3.32 virtual void gazebo::physics::DARTLink::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.

Reimplemented from **gazebo::physics::Link** (p. 765).

10.58.3.33 virtual void gazebo::physics::DARTLink::SetLinearDamping (double *_damping*) [virtual]

Set the linear damping factor.

Parameters

in	*_damping*	Linear damping factor.

Implements **gazebo::physics::Link** (p. 766).

10.58.3.34 virtual void gazebo::physics::DARTLink::SetLinearVel (const math::Vector3 & *_vel*) [virtual]

Set the linear velocity of the body.

Parameters

in	*_vel*	Linear velocity.

Implements **gazebo::physics::Link** (p. 766).

10.58.3.35 virtual void gazebo::physics::DARTLink::SetLinkStatic (bool *_static*) [virtual]

Freeze link to ground (inertial frame).

Parameters

in	*_static*	if true, freeze link to ground. Otherwise unfreeze link.

Implements **gazebo::physics::Link** (p. 766).

10.58.3.36 virtual void gazebo::physics::DARTLink::SetSelfCollide (bool *_collide*) [virtual]

Set whether this body will collide with others in the model.

Parameters

in	*_collid*	True to enable collisions.

Implements **gazebo::physics::Link** (p. 767).

10.58.3.37 virtual void gazebo::physics::DARTLink::SetTorque (const math::Vector3 & *_torque*) [virtual]

Set the torque applied to the body.

Parameters

in	*_torque*	Torque value.

Implements **gazebo::physics::Link** (p. 768).

10.58.3.38 void gazebo::physics::DARTLink::updateDirtyPoseFromDARTTransformation ( )

Store DART Transformation to **Entity::dirtyPose** (p. 513) and add this link to **World::dirtyPoses** (p. 1543) so that World::Update() trigger **Entity::SetWorldPose()** (p. 512) for this link.

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

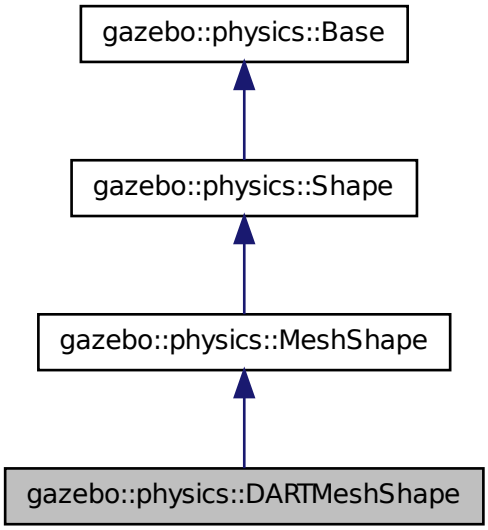
- **DARTLink.hh**

10.59 gazebo::physics::DARTMeshShape Class Reference

Triangle mesh collision.

`#include <DARTMeshShape.hh>`

Inheritance diagram for gazebo::physics::DARTMeshShape:



Public Member Functions

- **DARTMeshShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**DARTMeshShape** ()

  Destructor.
- virtual void **Init** ()
- virtual void **Load** (sdf::ElementPtr _sdf)

Load.

- virtual void **Update** ()

  Update the tri mesh.

10.59.1 Detailed Description

Triangle mesh collision.

10.59.2 Constructor & Destructor Documentation

10.59.2.1 gazebo::physics::DARTMeshShape::DARTMeshShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent collision object.

10.59.2.2 virtual gazebo::physics::DARTMeshShape::~DARTMeshShape ( ) [virtual]

Destructor.

10.59.3 Member Function Documentation

10.59.3.1 virtual void gazebo::physics::DARTMeshShape::Init ( ) [virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::MeshShape** (p. 844).

10.59.3.2 virtual void gazebo::physics::DARTMeshShape::Load (sdf::ElementPtr *_sdf*) [virtual]

Load.

Parameters

in	*node*	Pointer to an SDF parameters

Reimplemented from **gazebo::physics::Base** (p. 212).

10.59.3.3 virtual void gazebo::physics::DARTMeshShape::Update ( ) [virtual]

Update the tri mesh.

Reimplemented from **gazebo::physics::MeshShape** (p. 845).

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

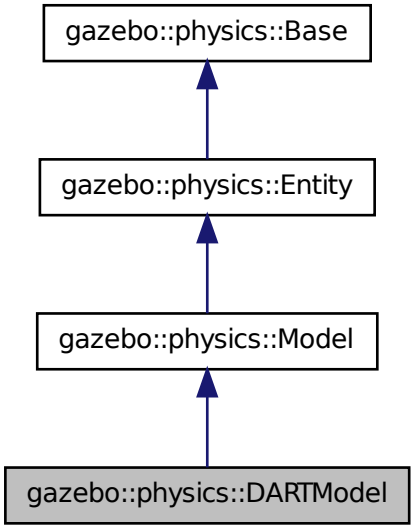
- **DARTMeshShape.hh**

10.60 gazebo::physics::DARTModel Class Reference

DART model class.

`#include <DARTModel.hh>`

Inheritance diagram for gazebo::physics::DARTModel:



Public Member Functions

- **DARTModel** (**BasePtr** _parent)

  Constructor.
- virtual ∼**DARTModel** ()

  Destructor.
- void **BackupState** ()
- virtual void **Fini** ()

  Finalize the model.
- **DARTPhysicsPtr GetDARTPhysics** (void) const
- dart::dynamics::Skeleton ∗ **GetDARTSkeleton** ()
- dart::simulation::World ∗ **GetDARTWorld** (void) const
- virtual void **Init** ()

  Initialize the model.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the model.
- void **RestoreState** ()
- virtual void **Update** ()

  Update the model.

Protected Attributes

- Eigen::VectorXd **dtConfig**
- dart::dynamics::Skeleton ∗ **dtSkeleton**
- Eigen::VectorXd **dtVelocity**

10.60.1 Detailed Description

DART model class.

10.60.2 Constructor & Destructor Documentation

10.60.2.1 gazebo::physics::DARTModel::DARTModel (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent object.

10.60.2.2 virtual gazebo::physics::DARTModel::~DARTModel ( ) [virtual]

Destructor.

10.60.3 Member Function Documentation

10.60.3.1 void gazebo::physics::DARTModel::BackupState ( )

10.60.3.2 virtual void gazebo::physics::DARTModel::Fini ( ) [virtual]

Finalize the model.

Reimplemented from **gazebo::physics::Model** (p. 851).

10.60.3.3 DARTPhysicsPtr gazebo::physics::DARTModel::GetDARTPhysics (void) const

10.60.3.4 dart::dynamics::Skeleton* gazebo::physics::DARTModel::GetDART-Skeleton ( )

10.60.3.5 dart::simulation::World* gazebo::physics::DARTModel::GetDARTWorld (void) const

10.60.3.6 virtual void gazebo::physics::DARTModel::Init ( ) [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Model** (p. 857).

10.60.3.7 virtual void gazebo::physics::DARTModel::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the model.

Parameters

in	*_sdf*	SDF parameters to load from.

Reimplemented from **gazebo::physics::Model** (p. 857).

10.60.3.8 void gazebo::physics::DARTModel::RestoreState ( )

10.60.3.9 virtual void gazebo::physics::DARTModel::Update ( ) [virtual]

Update the model.

Reimplemented from **gazebo::physics::Model** (p. 863).

10.60.4 Member Data Documentation

10.60.4.1 Eigen::VectorXd gazebo::physics::DARTModel::dtConfig [protected]

10.60.4.2 dart::dynamics::Skeleton* gazebo::physics::DARTModel::dtSkeleton [protected]

10.60.4.3 Eigen::VectorXd gazebo::physics::DARTModel::dtVelocity [protected]

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

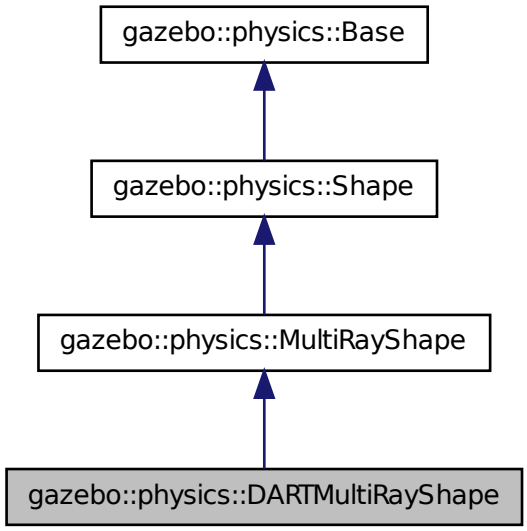
- **DARTModel.hh**

10.61 gazebo::physics::DARTMultiRayShape Class Reference

DART specific version of **MultiRayShape** (p. 901).

`#include <DARTMultiRayShape.hh>`

Inheritance diagram for gazebo::physics::DARTMultiRayShape:



Public Member Functions

- **DARTMultiRayShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**DARTMultiRayShape** ()

  Destructor.
- virtual void **UpdateRays** ()

  Physics engine specific method for updating the rays.

Protected Member Functions

- void **AddRay** (const **math::Vector3** &_start, const **math::Vector3** &_end)

  Add a ray to the collision.

10.61.1 Detailed Description

DART specific version of **MultiRayShape** (p. 901).

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

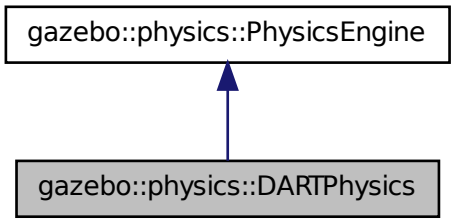
- **DARTMultiRayShape.hh**

10.62 gazebo::physics::DARTPhysics Class Reference

DART physics engine.

`#include <DARTPhysics.hh>`

Inheritance diagram for gazebo::physics::DARTPhysics:



Public Types

- enum **DARTParam** { **MAX_CONTACTS**, **MIN_STEP_SIZE** }

  DART physics parameter types.

Public Member Functions

- **DARTPhysics** (**WorldPtr** _world)

  Constructor.
- virtual ∼**DARTPhysics** ()

  Destructor.
- virtual **CollisionPtr CreateCollision** (const std::string &_type, **LinkPtr** _body)

Create a collision.

- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent)

  Create a new joint.
- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)

  Create a new body.
- virtual **ModelPtr CreateModel** (**BasePtr** _parent)

  Create a new model.
- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)

  *Create a **physics::Shape*** (p. *1161) object.*
- virtual void **DebugPrint** () const

  Debug print out of the physic engine state.
- virtual void **Fini** ()

  Finilize the physics engine.
- dart::simulation::World * **GetDARTWorld** ()

  *Get pointer to DART **World*** (p. *1529) associated with this DART Physics.*
- virtual boost::any **GetParam** (const std::string &_key) const

  Get an parameter of the physics engine.
- virtual std::string **GetType** () const

  Return the physics engine type (ode|bullet|dart|simbody).
- virtual void **Init** ()

  Initialize the physics engine.
- virtual void **InitForThread** ()

  Init the engine for threads.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the physics engine.
- virtual void **Reset** ()

  Rest the physics engine.
- virtual void **SetGravity** (const **gazebo::math::Vector3** &_gravity)

  Set the gavity vector.
- virtual bool **SetParam** (const std::string &_key, const boost::any &_value)

  Set a parameter of the physics engine.
- virtual void **SetSeed** (uint32_t _seed)

  Set the random number seed for the physics engine.
- virtual void **UpdateCollision** ()

  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".

10.62.1 Detailed Description

DART physics engine.

10.62.2 Member Enumeration Documentation

10.62.2.1 enum gazebo::physics::DARTPhysics::DARTParam

DART physics parameter types.

Enumerator:

MAX_CONTACTS Maximum number of contacts.

MIN_STEP_SIZE Minimum step size.

10.62.3 Constructor & Destructor Documentation

10.62.3.1 gazebo::physics::DARTPhysics::DARTPhysics (WorldPtr *_world*)

Constructor.

10.62.3.2 virtual gazebo::physics::DARTPhysics::~DARTPhysics ( ) [virtual]

Destructor.

10.62.4 Member Function Documentation

10.62.4.1 virtual CollisionPtr gazebo::physics::DARTPhysics::CreateCollision (const std::string & *_shapeType*, LinkPtr *_link*) [virtual]

Create a collision.

Parameters

in	*_shapeType*	Type of collision to create.
in	*_link*	Parent link.

Implements **gazebo::physics::PhysicsEngine** (p. 963).

10.62.4.2 virtual JointPtr gazebo::physics::DARTPhysics::CreateJoint (const std::string & *_type*, ModelPtr *_parent*) [virtual]

Create a new joint.

Parameters

in	*_type*	Type of joint to create.
in	*_parent*	**Model** (p. 846) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 964).

10.62.4.3 virtual LinkPtr gazebo::physics::DARTPhysics::CreateLink (ModelPtr *_parent*) [virtual]

Create a new body.

Parameters

in	*_parent*	Parent model for the link.

Implements **gazebo::physics::PhysicsEngine** (p. 964).

10.62.4.4 virtual ModelPtr gazebo::physics::DARTPhysics::CreateModel (BasePtr *_base*) [virtual]

Create a new model.

Parameters

in	*_base*	Boost shared pointer to a new model.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 964).

10.62.4.5 virtual ShapePtr gazebo::physics::DARTPhysics::CreateShape (const std::string & _shapeType, CollisionPtr _collision) [virtual]

Create a **physics::Shape** (p. 1161) object.

Parameters

in	*_shapeType*	Type of shape to create.
in	*_collision*	**Collision** (p. 295) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 965).

10.62.4.6 virtual void gazebo::physics::DARTPhysics::DebugPrint ( ) const [virtual]

Debug print out of the physic engine state.

Implements **gazebo::physics::PhysicsEngine** (p. 965).

10.62.4.7 virtual void gazebo::physics::DARTPhysics::Fini ( ) [virtual]

Finilize the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 965).

10.62.4.8 dart::simulation::World* gazebo::physics::DARTPhysics::GetDARTWorld ( )

Get pointer to DART **World** (p. 1529) associated with this DART Physics.

Returns

The pointer to DART **World** (p. 1529).

10.62.4.9 virtual boost::any gazebo::physics::DARTPhysics::GetParam (const std::string & _key) const [virtual]

Get an parameter of the physics engine.

Parameters

in	*_attr*	String key

See also

SetParam (p. 444)

Returns

The value of the parameter

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 967).

10.62.4.10 virtual std::string gazebo::physics::DARTPhysics::GetType ( ) const [virtual]

Return the physics engine type (ode|bullet|dart|simbody).

Returns

Type of the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 968).

10.62.4.11 virtual void gazebo::physics::DARTPhysics::Init ( ) [virtual]

Initialize the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 969).

10.62.4.12 virtual void gazebo::physics::DARTPhysics::InitForThread ( ) [virtual]

Init the engine for threads.

Implements **gazebo::physics::PhysicsEngine** (p. 969).

10.62.4.13 virtual void gazebo::physics::DARTPhysics::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the physics engine.

Parameters

in	*_sdf*	Pointer to the SDF parameters.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 969).

10.62.4.14 virtual void gazebo::physics::DARTPhysics::OnPhysicsMsg (ConstPhysicsPtr & _msg) [protected, virtual]

virtual callback for gztopic "~/physics".

Parameters

in	*_msg*	Physics message.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 969).

10.62.4.15 virtual void gazebo::physics::DARTPhysics::OnRequest (ConstRequestPtr & _msg) [protected, virtual]

virtual callback for gztopic "~/request".

Parameters

in	*_msg*	Request message.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 970).

10.62.4.16 virtual void gazebo::physics::DARTPhysics::Reset ( ) [virtual]

Rest the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 970).

10.62.4.17 virtual void gazebo::physics::DARTPhysics::SetGravity (const gazebo::math::Vector3 & _gravity) [virtual]

Set the gavity vector.

Parameters

in	*_gravity*	New gravity vector.

Implements **gazebo::physics::PhysicsEngine** (p. 971).

10.62.4.18 virtual bool gazebo::physics::DARTPhysics::SetParam (const std::string & _key, const boost::any & _value) [virtual]

Set a parameter of the physics engine.

See SetParam documentation for descriptions of duplicate parameters.

Parameters

in	*_key*	String key Below is a list of _key parameter definitions: 1. "solver_type" (string) - returns solver used by engine, e.g. "sequential_impulse' for Bullet, "quick" for ODE "Featherstone and Lemkes" for DART and "Spatial - Algebra and Elastic Foundation" for Simbody. -# "cfm" (double) - global CFM -# "erp" (double) - global ERP -# "precon_iters" (bool) - precondition iterations (experimental). 2. "iters" (int) - number of LCP PGS iterations. If sor_lcp_tolerance is negative, full iteration count is executed. Otherwise, PGS may stop iteration early if sor_lcp_tolerance is satisfied by the total RMS residual. 3. "sor" (double) - relaxation parameter for Projected - Gauss-Seidel (PGS) updates. 4. "contact_max_correcting_vel" (double) - truncates correction impulses from ERP by this value. 5. "contact_surface_layer" (double) - ERP is 0 for interpenetration depths below this value. 6. "max_contacts" (int) - max number of contact constraints between any pair of collision bodies. 7. "min_step_size" (double) - minimum internal step size. (defined but not used in ode). 8. "max_step_size" (double) - maximum physics step size when physics update step must return.
in	*_value*	The value to set to

Returns

true if SetParam is successful, false if operation fails.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 972).

10.62.4.19 virtual void gazebo::physics::DARTPhysics::SetSeed (uint32_t *_seed*) [virtual]

Set the random number seed for the physics engine.

Parameters

in	*_seed*	The random number seed.

Implements **gazebo::physics::PhysicsEngine** (p. 973).

10.62.4.20 virtual void gazebo::physics::DARTPhysics::UpdateCollision ( ) [virtual]

Update the physics engine collision.

Implements **gazebo::physics::PhysicsEngine** (p. 974).

10.62.4.21 virtual void gazebo::physics::DARTPhysics::UpdatePhysics ( ) [virtual]

Update the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 974).

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

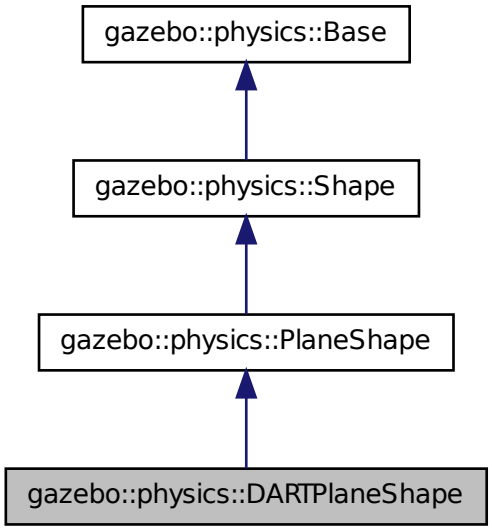
- **DARTPhysics.hh**

10.63 gazebo::physics::DARTPlaneShape Class Reference

An DART Plane shape.

`#include <DARTPlaneShape.hh>`

Inheritance diagram for gazebo::physics::DARTPlaneShape:



Public Member Functions

- **DARTPlaneShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**DARTPlaneShape** ()

  Destructor.
- virtual void **CreatePlane** ()

  Create the plane.
- virtual void **SetAltitude** (const **math::Vector3** &_pos)

  Set the altitude of the plane.

10.63.1 Detailed Description

An DART Plane shape.

10.63.2 Constructor & Destructor Documentation

10.63.2.1 gazebo::physics::DARTPlaneShape::DARTPlaneShape (CollisionPtr *_parent*) [inline, explicit]

Constructor.

Parameters

in	*_parent*	Parent **Collision** (p. 295).

10.63.2.2 virtual gazebo::physics::DARTPlaneShape::~DARTPlaneShape ( ) [inline, virtual]

Destructor.

10.63.3 Member Function Documentation

10.63.3.1 virtual void gazebo::physics::DARTPlaneShape::CreatePlane ( ) [inline, virtual]

Create the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 989).

References gazebo::physics::PlaneShape::CreatePlane(), and gazebo::physics::DARTCollision::GetDARTBodyNode().

10.63.3.2 virtual void gazebo::physics::DARTPlaneShape::SetAltitude (const math::Vector3 & *_pos*) [inline, virtual]

Set the altitude of the plane.

Parameters

in	*_pos*	Position of the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 991).

References gazebo::physics::PlaneShape::SetAltitude().

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

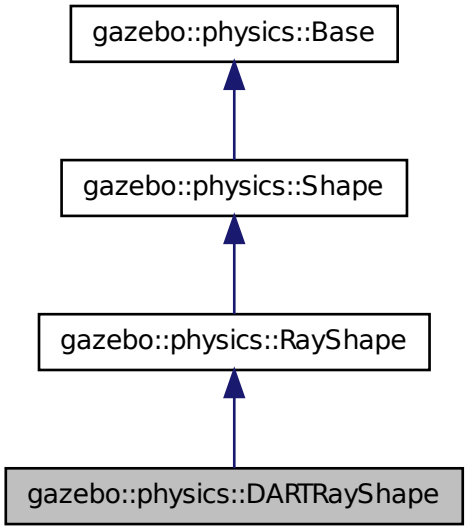
- **DARTPlaneShape.hh**

10.64 gazebo::physics::DARTRayShape Class Reference

Ray collision.

`#include <DARTRayShape.hh>`

Inheritance diagram for gazebo::physics::DARTRayShape:



Public Member Functions

- **DARTRayShape** (**PhysicsEnginePtr** _physicsEngine)

  Constructor for a global ray.
- **DARTRayShape** (**CollisionPtr** _collision)

  Constructor.
- virtual ~**DARTRayShape** ()

  Destructor.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)

  Get the nearest intersection.
- virtual void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)

Set the ray based on starting and ending points relative to the body.

- virtual void **Update** ()

Update the ray collision.

10.64.1 Detailed Description

Ray collision.

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

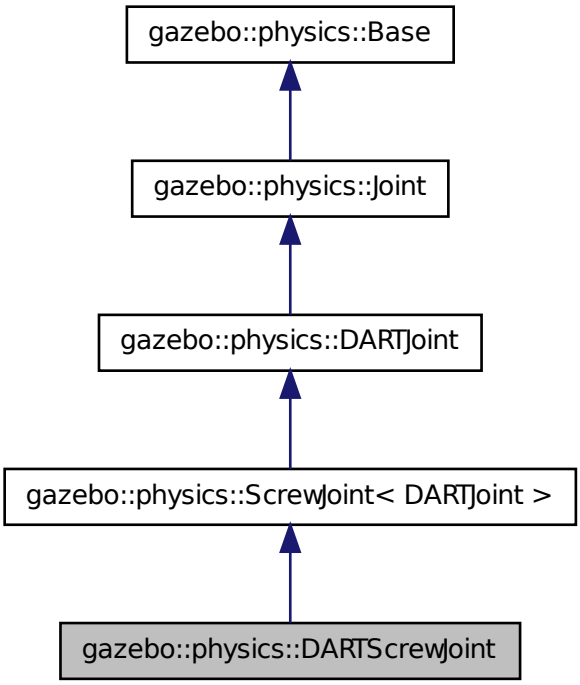
- **DARTRayShape.hh**

10.65 gazebo::physics::DARTScrewJoint Class Reference

A screw joint.

`#include <DARTScrewJoint.hh>`

Inheritance diagram for gazebo::physics::DARTScrewJoint:



Public Member Functions

- **DARTScrewJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**DARTScrewJoint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const

  Get the anchor point.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.

- virtual **math::Angle GetHighStop** (unsigned int _index)

  Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (unsigned int _index)

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual double **GetParam** (const std::string &_key, unsigned int _index)

  Get a non-generic parameter for the joint.
- virtual double **GetThreadPitch** (unsigned int _index)
- virtual double **GetThreadPitch** ()
- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)
- virtual void **Init** ()

  Initialize joint.
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **ScrewJoint** (p. 1118).*
- virtual void **SetAnchor** (unsigned int _index, const **math::Vector3** &_anchor)

  Set the anchor point.
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _force)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual void **SetThreadPitch** (unsigned int _index, double _threadPitch)
- virtual void **SetThreadPitch** (double _threadPitch)
- virtual void **SetVelocity** (unsigned int _index, double _vel)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _effort)

  *Set the force applied to this **physics::Joint** (p. 669).*

Protected Attributes

- dart::dynamics::ScrewJoint * **dartScrewJoint**

  Universal joint of DART.

10.65.1 Detailed Description

A screw joint.

10.65.2 Constructor & Destructor Documentation

10.65.2.1 gazebo::physics::DARTScrewJoint::DARTScrewJoint (BasePtr _parent)

Constructor.

Parameters

in	*_parent*	Pointer to the **Link** (p. 739) that is the joint' parent

10.65.2.2 virtual gazebo::physics::DARTScrewJoint::~DARTScrewJoint ( ) [virtual]

Destructor.

10.65.3 Member Function Documentation

10.65.3.1 virtual math::Vector3 gazebo::physics::DARTScrewJoint::GetAnchor (unsigned int _index) const [virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 679).

10.65.3.2 virtual math::Angle gazebo::physics::DARTScrewJoint::GetAngleImpl (unsigned int _index) const [virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.65.3.3 virtual math::Vector3 gazebo::physics::DARTScrewJoint::GetGlobalAxis (unsigned int _index) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.65.3.4 virtual math::Angle gazebo::physics::DARTScrewJoint::GetHighStop (unsigned int _index) [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.

Reimplemented from **gazebo::physics::DARTJoint** (p. 411).

10.65.3.5 virtual math::Angle gazebo::physics::DARTScrewJoint::GetLowStop (unsigned int _index) [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.

Reimplemented from **gazebo::physics::DARTJoint** (p. 412).

10.65.3.6 virtual double gazebo::physics::DARTScrewJoint::GetMaxForce (unsigned int *_index*) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.65.3.7 virtual double gazebo::physics::DARTScrewJoint::GetParam (const std::string & *_key,* unsigned int *_index*) [virtual]

Get a non-generic parameter for the joint.

Parameters

in	*_key*	String key.
in	*_index*	Index of the axis.

Reimplemented from **gazebo::physics::DARTJoint** (p. 413).

10.65.3.8 virtual double gazebo::physics::DARTScrewJoint::GetThreadPitch (unsigned int *_index*) [virtual]

Get screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This

must be implemented in a child class

Returns

_threadPitch Thread pitch value.

10.65.3.9 virtual double gazebo::physics::DARTScrewJoint::GetThreadPitch ( ) [virtual]

Get screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class

Returns

_threadPitch Thread pitch value.

Implements **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1119).

10.65.3.10 virtual double gazebo::physics::DARTScrewJoint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.65.3.11 virtual void gazebo::physics::DARTScrewJoint::Init ( ) [virtual]

Initialize joint.

Reimplemented from **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1120).

10.65.3.12 virtual void gazebo::physics::DARTScrewJoint::Load (sdf::ElementPtr _sdf) [virtual]

Load a **ScrewJoint** (p. 1118).

Parameters

in	*_sdf*	SDF value to load from

Reimplemented from **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1120).

10.65.3.13 virtual void gazebo::physics::DARTScrewJoint::SetAnchor (unsigned int *int,* const math::Vector3 &) [virtual]

Set the anchor point.

Reimplemented from **gazebo::physics::DARTJoint** (p. 414).

10.65.3.14 virtual void gazebo::physics::DARTScrewJoint::SetAxis (unsigned int *_index,* const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 695).

10.65.3.15 virtual void gazebo::physics::DARTScrewJoint::SetForceImpl (unsigned int *_index,* double *_force*) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value.

Implements **gazebo::physics::DARTJoint** (p. 415).

10.65.3.16 virtual void gazebo::physics::DARTScrewJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.65.3.17 virtual void gazebo::physics::DARTScrewJoint::SetThreadPitch (unsigned int *_index*, double *_threadPitch*) [virtual]

Set screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. - This must be implemented in a child class To clarify direction, these are modeling right handed threads with positive thread_pitch, i.e. the child **Link** (p. 739) is the nut (interior threads) while the parent **Link** (p. 739) is the bolt/screw (exterior threads).

Parameters

in	*_threadPitch*	Thread pitch value.

10.65.3.18 virtual void gazebo::physics::DARTScrewJoint::SetThreadPitch (double *_threadPitch*) [virtual]

Set screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. - This must be implemented in a child class To clarify direction, these are modeling right handed threads with positive thread_pitch, i.e. the child **Link** (p. 739) is the nut (interior threads) while the parent **Link** (p. 739) is the bolt/screw (exterior threads).

Parameters

in	*_threadPitch*	Thread pitch value.

Implements **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1120).

10.65.3.19 virtual void gazebo::physics::DARTScrewJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

10.65.4 Member Data Documentation

10.65.4.1 dart::dynamics::ScrewJoint* gazebo::physics::DARTScrewJoint::dartScrewJoint [protected]

Universal joint of DART.

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

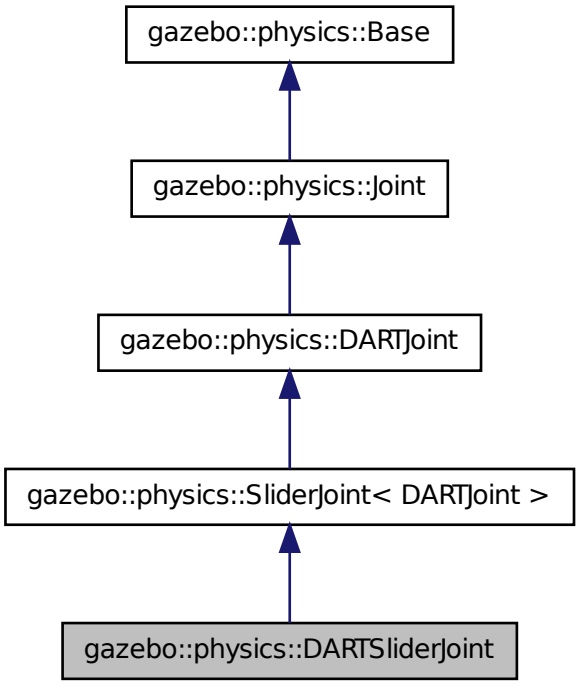
- **DARTScrewJoint.hh**

10.66 gazebo::physics::DARTSliderJoint Class Reference

A slider joint.

`#include <DARTSliderJoint.hh>`

Inheritance diagram for gazebo::physics::DARTSliderJoint:



Public Member Functions

- **DARTSliderJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**DARTSliderJoint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const

  Get the anchor point.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.

- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)
- virtual void **Init** ()

  Initialize a joint.
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **SliderJoint** (p. 1294).*
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _force)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual void **SetVelocity** (unsigned int _index, double _vel)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _effort)

  *Set the force applied to this **physics::Joint** (p. 669).*

Protected Attributes

- dart::dynamics::PrismaticJoint * **dtPrismaticJoint**

  Prismatic joint of DART.

10.66.1 Detailed Description

A slider joint.

10.66.2 Constructor & Destructor Documentation

10.66.2.1 gazebo::physics::DARTSliderJoint::DARTSliderJoint (BasePtr *_parent*)

Constructor.

Parameters

in	*_parent*	Pointer to the **Link** (p. 739) that is the joint' parent

10.66.2.2 virtual gazebo::physics::DARTSliderJoint::~DARTSliderJoint ( ) [virtual]

Destructor.

10.66.3 Member Function Documentation

10.66.3.1 virtual math::Vector3 gazebo::physics::DARTSliderJoint::GetAnchor (unsigned int _index) const [virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 679).

10.66.3.2 virtual math::Angle gazebo::physics::DARTSliderJoint::GetAngleImpl (unsigned int _index) const [virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.66.3.3 virtual math::Vector3 gazebo::physics::DARTSliderJoint::GetGlobalAxis (unsigned int _index) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.66.3.4 virtual double gazebo::physics::DARTSliderJoint::GetMaxForce (unsigned int *_index*) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.66.3.5 virtual double gazebo::physics::DARTSliderJoint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.66.3.6 virtual void gazebo::physics::DARTSliderJoint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::DARTJoint** (p. 413).

10.66.3.7 virtual void gazebo::physics::DARTSliderJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load a **SliderJoint** (p. 1294).

Parameters

in	*_sdf*	SDF values to load from

Reimplemented from **gazebo::physics::SliderJoint**< **DARTJoint** > (p. 1295).

10.66.3.8 virtual void gazebo::physics::DARTSliderJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 695).

10.66.3.9 virtual void gazebo::physics::DARTSliderJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value.

Implements **gazebo::physics::DARTJoint** (p. 415).

10.66.3.10 virtual void gazebo::physics::DARTSliderJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.66.3.11 virtual void gazebo::physics::DARTSliderJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

10.66.4 Member Data Documentation

10.66.4.1 dart::dynamics::PrismaticJoint* gazebo::physics::DARTSliderJoint::dtPrismaticJoint [protected]

Prismatic joint of DART.

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

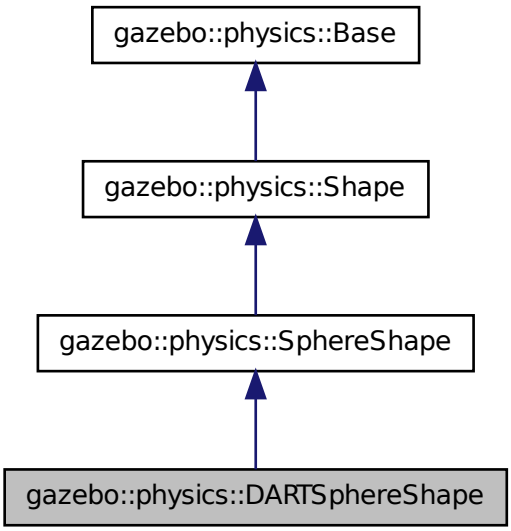
- **DARTSliderJoint.hh**

10.67 gazebo::physics::DARTSphereShape Class Reference

A DART sphere shape.

`#include <DARTSphereShape.hh>`

Inheritance diagram for gazebo::physics::DARTSphereShape:



Public Member Functions

- **DARTSphereShape** (**DARTCollisionPtr** _parent)

  Constructor.
- virtual ~**DARTSphereShape** ()

  Destructor.
- virtual void **SetRadius** (double _radius)

  Set the size.

10.67.1 Detailed Description

A DART sphere shape.

10.67.2 Constructor & Destructor Documentation

10.67.2.1 gazebo::physics::DARTSphereShape::DARTSphereShape (DARTCollisionPtr *_parent*) [inline, explicit]

Constructor.

Parameters

in	*_parent*	Parent **Collision** (p. 295).

10.67.2.2 virtual gazebo::physics::DARTSphereShape::~DARTSphereShape ( ) [inline, virtual]

Destructor.

10.67.3 Member Function Documentation

10.67.3.1 virtual void gazebo::physics::DARTSphereShape::SetRadius (double *_radius*) [inline, virtual]

Set the size.

Parameters

in	*_radius*	Radius of the sphere.

Reimplemented from **gazebo::physics::SphereShape** (p. 1309).

References gazebo::math::equal(), gazebo::physics::DARTCollision::GetDARTBodyNode(), gzerr, gzwarn, NULL, and gazebo::physics::SphereShape::SetRadius().

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

- **DARTSphereShape.hh**

10.68 gazebo::physics::DARTTypes Class Reference

A set of functions for converting between the math types used by gazebo and dart.

`#include <DARTTypes.hh>`

Static Public Member Functions

- static Eigen::Isometry3d **ConvPose** (const **math::Pose** &_pose)
- static **math::Pose ConvPose** (const Eigen::Isometry3d &_T)
- static Eigen::Quaterniond **ConvQuat** (const **math::Quaternion** &_quat)
- static **math::Quaternion ConvQuat** (const Eigen::Quaterniond &_quat)
- static Eigen::Vector3d **ConvVec3** (const **math::Vector3** &_vec3)
- static **math::Vector3 ConvVec3** (const Eigen::Vector3d &_vec3)

10.68.1 Detailed Description

A set of functions for converting between the math types used by gazebo and dart.

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

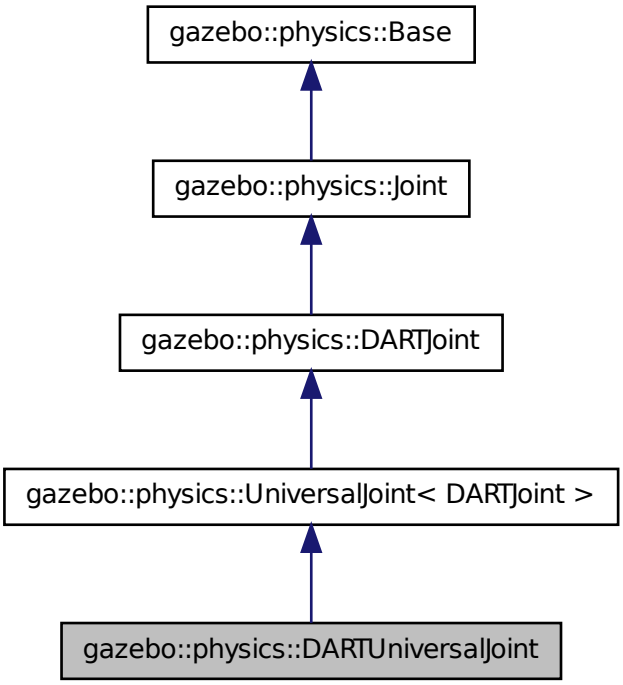
- **DARTTypes.hh**

10.69 gazebo::physics::DARTUniversalJoint Class Reference

A universal joint.

`#include <DARTUniversalJoint.hh>`

Inheritance diagram for gazebo::physics::DARTUniversalJoint:



Public Member Functions

- **DARTUniversalJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**DARTUniversalJoint** ()

  Destuctor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const

  Get the anchor point.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.

- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)
- virtual void **Init** ()

  Initialize joint.
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **UniversalJoint** (p. 1404).*
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _force)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual void **SetVelocity** (unsigned int _index, double _vel)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _effort)

  *Set the force applied to this **physics::Joint** (p. 669).*

Protected Attributes

- dart::dynamics::UniversalJoint * **dtUniveralJoint**

  Universal joint of DART.

10.69.1 Detailed Description

A universal joint.

10.69.2 Constructor & Destructor Documentation

10.69.2.1 gazebo::physics::DARTUniversalJoint::DARTUniversalJoint (BasePtr *_parent*)

Constructor.

Parameters

in	*_parent*	Pointer to the **Link** (p. 739) that is the joint' parent

10.69.2.2 virtual gazebo::physics::DARTUniversalJoint::~DARTUniversalJoint ( ) [virtual]

Destuctor.

10.69.3 Member Function Documentation

10.69.3.1 virtual math::Vector3 gazebo::physics::DARTUniversalJoint::GetAnchor (unsigned int *_index*) const [virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 679).

10.69.3.2 virtual math::Angle gazebo::physics::DARTUniversalJoint::GetAngleImpl (unsigned int *_index*) const [virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.69.3.3 virtual math::Vector3 gazebo::physics::DARTUniversalJoint::GetGlobalAxis (unsigned int *_index*) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.69.3.4 virtual double gazebo::physics::DARTUniversalJoint::GetMaxForce (unsigned int *_index*) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.69.3.5 virtual double gazebo::physics::DARTUniversalJoint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.69.3.6 virtual void gazebo::physics::DARTUniversalJoint::Init ( ) [virtual]

Initialize joint.

Reimplemented from **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1406).

10.69.3.7 virtual void gazebo::physics::DARTUniversalJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load a **UniversalJoint** (p. 1404).

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1406).

10.69.3.8 virtual void gazebo::physics::DARTUniversalJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 695).

10.69.3.9 virtual void gazebo::physics::DARTUniversalJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value.

Implements **gazebo::physics::DARTJoint** (p. 415).

10.69.3.10 virtual void gazebo::physics::DARTUniversalJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.69.3.11 virtual void gazebo::physics::DARTUniversalJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

10.69.4 Member Data Documentation

10.69.4.1 dart::dynamics::UniversalJoint* gazebo::physics::DARTUniversalJoint::dtUniveralJoint [protected]

Universal joint of DART.

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

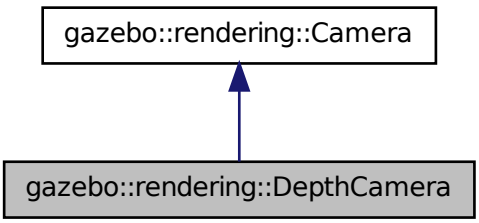
- **DARTUniversalJoint.hh**

10.70 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.

10.70.1 Detailed Description

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

10.70.2 Constructor & Destructor Documentation

10.70.2.1 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. 1097) that will contain the camera
in	*_auto-Render*	Almost everyone should leave this as true.

10.70.2.2 virtual gazebo::rendering::DepthCamera::~DepthCamera ( )

[virtual]

Destructor.

10.70.3 Member Function Documentation

10.70.3.1 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

10.70.3.2 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

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

Create a texture which will hold the depth data.

Parameters

in	*_texture-Name*	Name of the texture to create

10.70.3.4 void gazebo::rendering::DepthCamera::DisconnectNewDepthFrame (event::ConnectionPtr & *_c*) [inline]

Disconnect from an depth image singal.

Parameters

in	*_c*	The connection to disconnect

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

Disconnect from an rgb point cloud singal.

Parameters

in	*_c*	The connection to disconnect

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

Finalize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 254).

10.70.3.7 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

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

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 263).

10.70.3.9 void gazebo::rendering::DepthCamera::Load (sdf::ElementPtr _sdf) [virtual]

Load the camera with a set of parmeters.

Parameters

in	*_sdf*	The SDF camera info

Reimplemented from **gazebo::rendering::Camera** (p. 264).

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

Load the camera with default parmeters.

Reimplemented from **gazebo::rendering::Camera** (p. 264).

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

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 265).

10.70.3.12 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

10.70.4 Member Data Documentation

10.70.4.1 Ogre::RenderTarget* gazebo::rendering::DepthCamera::depthTarget [protected]

Pointer to the depth target.

10.70.4.2 Ogre::Texture* gazebo::rendering::DepthCamera::depthTexture [protected]

Pointer to the depth texture.

10.70.4.3 Ogre::Viewport* gazebo::rendering::DepthCamera::depthViewport [protected]

Pointer to the depth viewport.

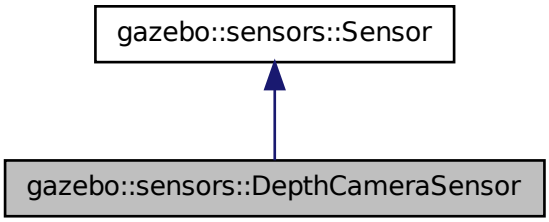
The documentation for this class was generated from the following file:

- **DepthCamera.hh**

10.71 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. *474*).

- 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.

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 bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

10.71.1 Constructor & Destructor Documentation

10.71.1.1 gazebo::sensors::DepthCameraSensor::DepthCameraSensor ( )

Constructor.

10.71.1.2 virtual gazebo::sensors::DepthCameraSensor::~DepthCameraSensor ( ) [virtual]

Destructor.

10.71.2 Member Function Documentation

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

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.71.2.2 rendering::DepthCameraPtr gazebo::sensors::-DepthCameraSensor::GetDepthCamera ( ) const [inline]

Returns a pointer to the **rendering::DepthCamera** (p. 474).

Returns

Depth Camera pointer

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

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.71.2.4 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. 1130) parameters
in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 1140).

10.71.2.5 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. 1140).

10.71.2.6 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

10.71.2.7 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. 1141).

10.71.2.8 virtual bool gazebo::sensors::DepthCameraSensor::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

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

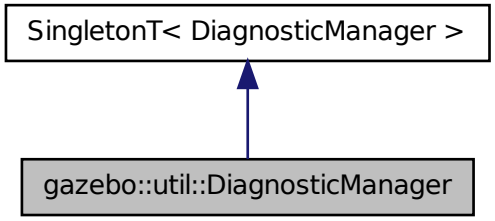
- **DepthCameraSensor.hh**

10.72 gazebo::util::DiagnosticManager Class Reference

A 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.

10.72.1 Detailed Description

A diagnostic manager class.

10.72.2 Member Function Documentation

10.72.2.1 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

10.72.2.2 boost::filesystem::path gazebo::util::DiagnosticManager::GetLogPath ( ) const

Get the path in which logs are stored.

Returns

The path in which logs are stored.

10.72.2.3 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

10.72.2.4 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

10.72.2.5 int gazebo::util::DiagnosticManager::GetTimerCount ( ) const

Get the number of timers.

Returns

The number of timers

10.72.2.6 void gazebo::util::DiagnosticManager::Init (const std::string & _worldName)

Initialize to report diagnostics about a world.

Parameters

in	*_worldName*	Name of the world.

10.72.2.7 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.

10.72.2.8 void gazebo::util::DiagnosticManager::StartTimer (const std::string & _name)

Start a new timer instance.

Parameters

in	*_name*	Name of the timer.

Returns

A pointer to the new diagnostic timer

10.72.2.9 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:

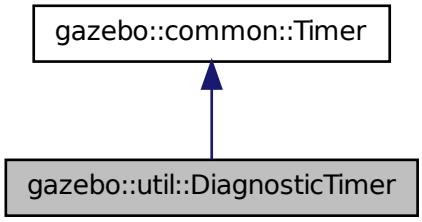
- **Diagnostics.hh**

10.73 gazebo::util::DiagnosticTimer Class Reference

A 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.

10.73.1 Detailed Description

A timer designed for diagnostics.

10.73.2 Constructor & Destructor Documentation

10.73.2.1 gazebo::util::DiagnosticTimer::DiagnosticTimer (const std::string & *_name*)

Constructor.

Parameters

in	*_name*	Name of the timer

10.73.2.2 virtual gazebo::util::DiagnosticTimer::~DiagnosticTimer ( ) [virtual]

Destructor.

10.73.3 Member Function Documentation

10.73.3.1 const std::string gazebo::util::DiagnosticTimer::GetName ( ) const [inline]

Get the name of the timer.

Returns

The name of timer

10.73.3.2 void gazebo::util::DiagnosticTimer::Lap (const std::string & *_prefix*)

Output a lap time.

Parameters

in	*_prefix*	Annotation to output with the elapsed time.

10.73.3.3 virtual void gazebo::util::DiagnosticTimer::Start ( ) [virtual]

Start the timer.

Reimplemented from **gazebo::common::Timer** (p. 1389).

10.73.3.4 virtual void gazebo::util::DiagnosticTimer::Stop ( ) [virtual]

Stop the timer.

Reimplemented from **gazebo::common::Timer** (p. 1389).

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

- **Diagnostics.hh**

10.74 gazebo::rendering::DummyPageProvider Class Reference

Pretends to provide procedural page content to avoid page loading.

`#include <rendering/rendering.hh>`

Public Member Functions

- bool **loadProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)
  Give a provider the opportunity to load page content procedurally.
- bool **prepareProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)
  Give a provider the opportunity to prepare page content procedurally.
- bool **unloadProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)
  Give a provider the opportunity to unload page content procedurally.
- bool **unprepareProceduralPage** (Ogre::Page *, Ogre::PagedWorldSection *)
  Give a provider the opportunity to unprepare page content procedurally.

10.74.1 Detailed Description

Pretends to provide procedural page content to avoid page loading.

10.74.2 Member Function Documentation

10.74.2.1 bool gazebo::rendering::DummyPageProvider::loadProceduralPage (Ogre::Page *, Ogre::PagedWorldSection *) [inline]

Give a provider the opportunity to load page content procedurally.

The parameters are not used.

10.74.2.2 bool gazebo::rendering::DummyPageProvider::prepareProceduralPage (Ogre::Page *, Ogre::PagedWorldSection *) [inline]

Give a provider the opportunity to prepare page content procedurally.

The parameters are not used.

10.74.2.3 bool gazebo::rendering::DummyPageProvider::unloadProceduralPage (Ogre::Page *, Ogre::PagedWorldSection *) [inline]

Give a provider the opportunity to unload page content procedurally.

The parameters are not used.

10.74.2.4 bool gazebo::rendering::DummyPageProvider::unprepareProceduralPage (Ogre::Page * , Ogre::PagedWorldSection *) [inline]

Give a provider the opportunity to unprepare page content procedurally.

The parameters are not used.

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

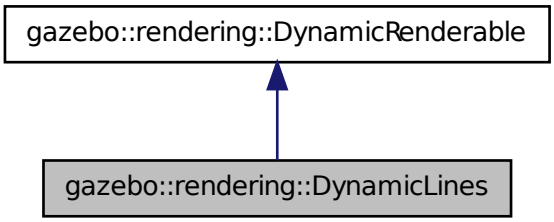
- **Heightmap.hh**

10.75 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, const **common::Color** &_color=**common::Color::White**)

  Add a point to the point list.

- void **AddPoint** (double _x, double _y, double _z, const **common::Color** &_color=**common::Color::White**)

  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. *163)'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 **SetColor** (unsigned int _index, const **common::Color** &_color)

  Change the color of an existing point 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.

10.75.1 Detailed Description

Class for drawing lines that can change.

10.75.2 Constructor & Destructor Documentation

10.75.2.1 gazebo::rendering::DynamicLines::DynamicLines (RenderOpType _opType = RENDERING_LINE_STRIP)

Constructor.

Parameters

in	*_opType*	The type of Line

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

Destructor.

10.75.3 Member Function Documentation

10.75.3.1 void gazebo::rendering::DynamicLines::AddPoint (const math::Vector3 & _pt, const common::Color & _color = common::Color::White)

Add a point to the point list.

Parameters

in	*_pt*	**math::Vector3** (p. 1440) point
in	*_color*	**common::Color** (p. 312) Point color

10.75.3.2 void gazebo::rendering::DynamicLines::AddPoint (double _x, double _y, double _z, const common::Color & _color = common::Color::White)

Add a point to the point list.

Parameters

in	*_x*	X position
in	*_y*	Y position
in	*_z*	Z position
in	*_color*	**common::Color** (p. 312) Point color

10.75.3.3 void gazebo::rendering::DynamicLines::Clear ( )

Remove all points from the point list.

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

Get type of movable.

Returns

This returns "gazebo::dynamiclines"

10.75.3.5 virtual const Ogre::String& gazebo::rendering::DynamicLines::getMovableType ( ) const [virtual]

Overridden function from **Ogre** (p. 163)'s base class.

Returns

Returns "gazebo::ogredynamicslines"

10.75.3.6 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. 1440) value of the point

10.75.3.7 unsigned int gazebo::rendering::DynamicLines::GetPointCount ( ) const

Return the total number of points in the point list.

Returns

Number of points

10.75.3.8 void gazebo::rendering::DynamicLines::SetColor (unsigned int *_index*, const common::Color & *_color*)

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

Parameters

in	*_index*	Index of the point to set
in	*_color*	**common::Color** (p. 312) Pixelcolor color to set the point to

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

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

Parameters

in	*_index*	Index of the point to set
in	*_value*	**math::Vector3** (p. 1440) value to set the point to

10.75.3.10 void gazebo::rendering::DynamicLines::Update ( )

Call this to update the hardware buffer after making changes.

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

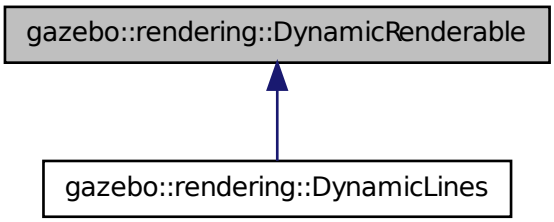
- **DynamicLines.hh**

10.76 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.

- std::string **GetMovableType** () const

  Get type of movable.

- **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.

10.76.1 Detailed Description

Abstract base class providing mechanisms for dynamically growing hardware buffers.

10.76.2 Constructor & Destructor Documentation

10.76.2.1 gazebo::rendering::DynamicRenderable::DynamicRenderable ( )

Constructor.

10.76.2.2 virtual gazebo::rendering::DynamicRenderable::~DynamicRenderable ( ) [virtual]

Virtual destructor.

10.76.3 Member Function Documentation

10.76.3.1 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.

10.76.3.2 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.

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

Implementation of Ogre::SimpleRenderable.

Returns

The bounding radius

10.76.3.4 std::string gazebo::rendering::DynamicRenderable::GetMovableType ( ) const

Get type of movable.

Returns

This returns "gazebo::DynamicRenderable"

10.76.3.5 RenderOpType gazebo::rendering::DynamicRenderable::GetOperationType ( ) const

Get the render operation type.

Returns

The render operation type.

10.76.3.6 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. 163) camera that views the renderable.

Returns

The squared depth in the **Camera** (p. 242)'s view

10.76.3.7 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. 497).

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.

10.76.3.8 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.

10.76.3.9 void gazebo::rendering::DynamicRenderable::SetOperationType (RenderOpType *_opType*)

Set the render operation type.

Parameters

in	*_opType*	The type of render operation to perform.

10.76.4 Member Data Documentation

10.76.4.1 size_t gazebo::rendering::DynamicRenderable::indexBufferCapacity [protected]

Maximum capacity of the currently allocated index buffer.

10.76.4.2 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:

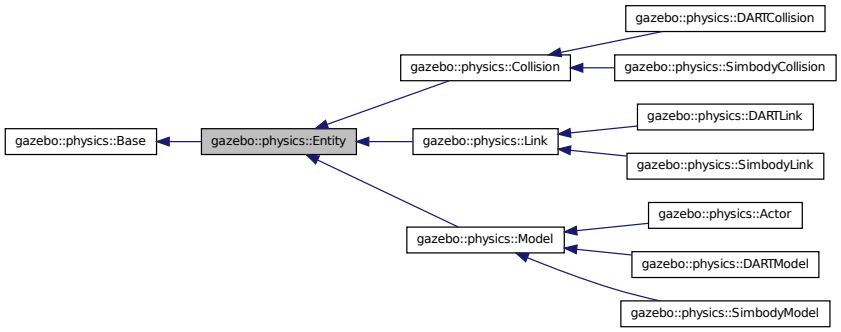
- **DynamicRenderable.hh**

10.77 gazebo::physics::Entity Class Reference

Base (p. 201) 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. *513)*.*
- **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.
- virtual const **math::Pose** & **GetWorldPose** () const

  Get the absolute pose of the entity.
- bool **IsCanonicalLink** () const

A helper function that checks if this is a canonical body.

- bool **IsStatic** () const

  Return whether this entity is static.

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the entity.

- void **PlaceOnEntity** (const std::string &_entityName)

  Move this entity to be ontop of another entity by name.

- void **PlaceOnNearestEntityBelow** ()

  Move this entity to be ontop of the nearest entity below.

- virtual void **Reset** ()

  Reset the entity.

- void **SetAnimation** (const **common::PoseAnimationPtr** &_anim, boost::function< void()> _onComplete)

  Set an animation for this entity.

- void **SetAnimation** (**common::PoseAnimationPtr** _anim)

  Set an animation for this entity.

- void **SetCanonicalLink** (bool _value)

  Set to true if this entity is a canonical link for a model.

- void **SetInitialRelativePose** (const **math::Pose** &_pose)

  Set the initial pose.

- virtual void **SetName** (const std::string &_name)

  Set the name of the entity.

- void **SetRelativePose** (const **math::Pose** &_pose, bool _notify=true, bool _publish=true)

  Set the pose of the entity relative to its parent.

- void **SetStatic** (const bool &_static)

  Set whether this entity is static: immovable.

- void **SetWorldPose** (const **math::Pose** &_pose, bool _notify=true, bool _publish=true)

  Set the world pose of the entity.

- void **SetWorldTwist** (const **math::Vector3** &_linear, const **math::Vector3** &_angular, bool _updateChildren=true)

  *Set angular and linear rates of an **physics::Entity*** (p. *500)*.

- virtual void **StopAnimation** ()

  Stop the current animation, if any.

- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  Update the parameters using new sdf values.

Protected Member Functions

- virtual void **OnPoseChange** ()=0

  This function is called when the entity's (or one of its parents) pose of the parent has changed.

Protected Attributes

- **common::PoseAnimationPtr animation**

  Current pose animation.
- **event::ConnectionPtr animationConnection**

  Connection used to update an animation.
- **math::Pose animationStartPose**

  Start pose of an animation.
- std::vector< **event::ConnectionPtr** > **connections**

  All our event connections.
- **math::Pose dirtyPose**

  The pose set by a physics engine.
- **transport::NodePtr node**

  Communication node.
- **EntityPtr parentEntity**

  A helper that prevents numerous dynamic_casts.
- **common::Time prevAnimationTime**

  Previous time an animation was updated.
- **transport::PublisherPtr requestPub**

  Request publisher.
- **math::Vector3 scale**

  Scale of the entity.
- **transport::PublisherPtr visPub**

  Visual publisher.
- msgs::Visual * **visualMsg**

  Visual message container.
- **math::Pose worldPose**

  World (p. *1529) pose of the entity.*

10.77.1 Detailed Description

Base (p. 201) class for all physics objects in Gazebo.

10.77.2 Constructor & Destructor Documentation

10.77.2.1 gazebo::physics::Entity::Entity (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of the entity.

10.77.2.2 virtual gazebo::physics::Entity::~Entity ( ) [virtual]

Destructor.

10.77.3 Member Function Documentation

10.77.3.1 virtual void gazebo::physics::Entity::Fini ( ) [virtual]

Finalize the entity.

Reimplemented from **gazebo::physics::Base** (p. 207).

Reimplemented in **gazebo::physics::Actor** (p. 169), **gazebo::physics::Link** (p. 750), **gazebo::physics::Model** (p. 851), **gazebo::physics::DARTModel** (p. 435), **gazebo::physics::Collision** (p. 299), **gazebo::physics::DARTLink** (p. 423), **gazebo::physics::SimbodyLink** (p. 1212), and **gazebo::physics::DARTCollision** (p. 387).

10.77.3.2 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. 751), **gazebo::physics::Model** (p. 851), **gazebo::physics::Collision** (p. 299), **gazebo::physics::DARTCollision** (p. 387), and **gazebo::physics::SimbodyCollision** (p. 1176).

10.77.3.3 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. 295) object, or NULL if not found.

10.77.3.4 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. 739) object.

Returns

Pointer to the **Link** (p. 739) object, or NULL if not found.

10.77.3.5 math::Box gazebo::physics::Entity::GetCollisionBoundingBox ( ) const

Returns collision bounding box.

Returns

Collsiion boundin box.

10.77.3.6 const math::Pose& gazebo::physics::Entity::GetDirtyPose ( ) const

Returns **Entity::dirtyPose** (p. 513).

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.

10.77.3.7 math::Pose gazebo::physics::Entity::GetInitialRelativePose ( ) const

Get the initial relative pose.

Returns

The initial relative pose.

10.77.3.8 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.

10.77.3.9 ModelPtr gazebo::physics::Entity::GetParentModel ( )

Get the parent model, if one exists.

Returns

Pointer to a model, or NULL if no parent model exists.

10.77.3.10 virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularAccel ( ) const [inline, virtual]

Get the angular acceleration of the entity.

Returns

A **math::Vector3** (p. 1440) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 754), **gazebo::physics::Collision** (p. 300), and **gazebo::physics::Model** (p. 854).

10.77.3.11 virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularVel ( ) const [inline, virtual]

Get the angular velocity of the entity.

Returns

A **math::Vector3** (p. 1440) for the velocity.

Reimplemented in **gazebo::physics::Link** (p. 755), **gazebo::physics::Collision** (p. 300), and **gazebo::physics::Model** (p. 854).

10.77.3.12 virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearAccel ( ) const [inline, virtual]

Get the linear acceleration of the entity.

Returns

A **math::Vector3** (p. 1440) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 755), **gazebo::physics::Collision** (p. 301), and **gazebo::physics::Model** (p. 854).

10.77.3.13 virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearVel ( ) const [inline, virtual]

Get the linear velocity of the entity.

Returns

A **math::Vector3** (p. 1440) for the linear velocity.

Reimplemented in **gazebo::physics::Link** (p. 755), **gazebo::physics::Collision** (p. 301), and **gazebo::physics::Model** (p. 855).

10.77.3.14 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.

10.77.3.15 virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularAccel ( ) const [inline, virtual]

Get the angular acceleration of the entity in the world frame.

Returns

A **math::Vector3** (p. 1440) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 757), **gazebo::physics::Collision** (p. 302), and **gazebo::physics::Model** (p. 855).

10.77.3.16 virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularVel ( ) const [inline, virtual]

Get the angular velocity of the entity in the world frame.

Returns

A **math::Vector3** (p. 1440) for the velocity.

Reimplemented in **gazebo::physics::Collision** (p. 302), **gazebo::physics::Model** (p. 856), **gazebo::physics::DARTLink** (p. 425), and **gazebo::physics::SimbodyLink** (p. 1213).

10.77.3.17 virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearAccel ( ) const [inline, virtual]

Get the linear acceleration of the entity in the world frame.

Returns

A **math::Vector3** (p. 1440) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 759), **gazebo::physics::Collision** (p. 303), and **gazebo::physics::Model** (p. 856).

10.77.3.18 virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearVel ( ) const [inline, virtual]

Get the linear velocity of the entity in the world frame.

Returns

A **math::Vector3** (p. 1440) for the linear velocity.

Reimplemented in **gazebo::physics::Link** (p. 759), **gazebo::physics::Collision** (p. 303), and **gazebo::physics::Model** (p. 857).

10.77.3.19 virtual const math::Pose& gazebo::physics::Entity::GetWorldPose ( ) const [inline, virtual]

Get the absolute pose of the entity.

Returns

The absolute pose of the entity.

Reimplemented in **gazebo::physics::Collision** (p. 303).

10.77.3.20 bool gazebo::physics::Entity::IsCanonicalLink ( ) const [inline]

A helper function that checks if this is a canonical body.

Returns

True if the link is canonical.

10.77.3.21 bool gazebo::physics::Entity::IsStatic ( ) const

Return whether this entity is static.

Returns

True if static.

10.77.3.22 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. 212).

Reimplemented in **gazebo::physics::Actor** (p. 169), **gazebo::physics::Link** (p. 761), **gazebo::physics::Model** (p. 857), **gazebo::physics::Collision** (p. 304), **gazebo::physics::SimbodyCollision** (p. 1177), **gazebo::physics::DARTLink** (p. 427), **gazebo::physics::SimbodyLink** (p. 1215), **gazebo::physics::DARTModel** (p. 435), **gazebo::physics::DARTCollision** (p. 388), and **gazebo::physics::SimbodyModel** (p. 1223).

10.77.3.23 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. 761), **gazebo::physics::Model** (p. 858), **gazebo::physics::DARTLink** (p. 427), **gazebo::physics::SimbodyLink** (p. 1216), **gazebo::physics::DARTCollision** (p. 388), and **gazebo::physics::SimbodyCollision** (p. 1177).

10.77.3.24 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. 500) this **Entity** (p. 500) should be ontop of.

10.77.3.25 void gazebo::physics::Entity::PlaceOnNearestEntityBelow ( )

Move this entity to be ontop of the nearest entity below.

10.77.3.26 virtual void gazebo::physics::Entity::Reset ( ) [virtual]

Reset the entity.

Reimplemented from **gazebo::physics::Base** (p. 214).

Reimplemented in **gazebo::physics::Model** (p. 858), and **gazebo::physics::Link** (p. 762).

10.77.3.27 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.

10.77.3.28 void gazebo::physics::Entity::SetAnimation (common::PoseAnimationPtr _anim)

Set an animation for this entity.

Parameters

in	*_anim*	Pose animation.

10.77.3.29 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.

10.77.3.30 void gazebo::physics::Entity::SetInitialRelativePose (const math::Pose & *_pose*)

Set the initial pose.

Parameters

in	*_pose*	The initial pose.

10.77.3.31 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. 215).

10.77.3.32 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.

10.77.3.33 void gazebo::physics::Entity::SetStatic (const bool & _static)

Set whether this entity is static: immovable.

Parameters

in	*_static*	True = static.

10.77.3.34 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.

10.77.3.35 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. 500).

Parameters

in	*_linear*	Linear twist.
in	*_angular*	Angular twist.
in	*_update-Children*	True to pass this update to child entities.

10.77.3.36 virtual void gazebo::physics::Entity::StopAnimation ( ) [virtual]

Stop the current animation, if any.

Reimplemented in **gazebo::physics::Model** (p. 862).

10.77.3.37 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. 216).

Reimplemented in **gazebo::physics::Actor** (p. 170), **gazebo::physics::Link** (p. 768), **gazebo::physics::Model** (p. 863), and **gazebo::physics::Collision** (p. 306).

10.77.4 Member Data Documentation

10.77.4.1 common::PoseAnimationPtr gazebo::physics::Entity::animation [protected]

Current pose animation.

10.77.4.2 event::ConnectionPtr gazebo::physics::Entity::animationConnection [protected]

Connection used to update an animation.

10.77.4.3 math::Pose gazebo::physics::Entity::animationStartPose [protected]

Start pose of an animation.

10.77.4.4 std::vector<event::ConnectionPtr> gazebo::physics::Entity::connections [protected]

All our event connections.

10.77.4.5 math::Pose gazebo::physics::Entity::dirtyPose [protected]

The pose set by a physics engine.

10.77.4.6 transport::NodePtr gazebo::physics::Entity::node [protected]

Communication node.

10.77.4.7 EntityPtr gazebo::physics::Entity::parentEntity [protected]

A helper that prevents numerous dynamic_casts.

10.77.4.8 common::Time gazebo::physics::Entity::prevAnimationTime [protected]

Previous time an animation was updated.

10.77.4.9 transport::PublisherPtr gazebo::physics::Entity::requestPub [protected]

Request publisher.

10.77.4.10 math::Vector3 gazebo::physics::Entity::scale [protected]

Scale of the entity.

10.77.4.11 transport::PublisherPtr gazebo::physics::Entity::visPub [protected]

Visual publisher.

10.77.4.12 msgs::Visual* gazebo::physics::Entity::visualMsg [protected]

Visual message container.

10.77.4.13 math::Pose gazebo::physics::Entity::worldPose [mutable, protected]

World (p. 1529) pose of the entity.

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

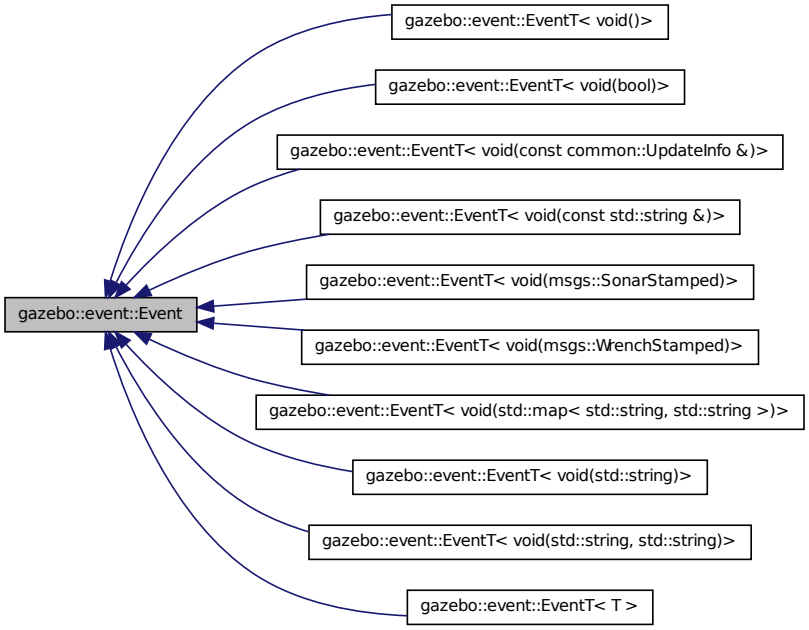
- **Entity.hh**

10.78 gazebo::event::Event Class Reference

Base class for all events.

`#include <common/common.hh>`

Inheritance diagram for gazebo::event::Event:



Public Member Functions

- **Event** ()

  Constructor.
- virtual ~**Event** ()

  Destructor.
- virtual void **Disconnect** (**ConnectionPtr** _c)=0

  Disconnect.
- virtual void **Disconnect** (int _id)=0

  Disconnect.
- bool **GetSignaled** () const

  Get whether this event has been signaled.

Protected Member Functions

- **Event** (**EventPrivate** &_d)

Allow subclasses to initialize their own data pointer.

Protected Attributes

- **EventPrivate * dataPtr**

  Data pointer.

10.78.1 Detailed Description

Base class for all events.

10.78.2 Constructor & Destructor Documentation

10.78.2.1 gazebo::event::Event::Event ( )

Constructor.

10.78.2.2 virtual gazebo::event::Event::~Event ( ) [virtual]

Destructor.

10.78.2.3 gazebo::event::Event::Event (EventPrivate & _d) [protected]

Allow subclasses to initialize their own data pointer.

Parameters

in	*_d*	Reference to data pointer.

10.78.3 Member Function Documentation

10.78.3.1 virtual void gazebo::event::Event::Disconnect (ConnectionPtr _c) [pure virtual]

Disconnect.

Parameters

in	*_c*	A pointer to a connection

Implemented in **gazebo::event::EventT< T >** (p. 57), **gazebo::event::EventT< void(msgs::SonarStamped)>** (p. 57), **gazebo::event::EventT< void(std::string)>** (p. 57), **gazebo::event::EventT< void(msgs::WrenchStamped)>** (p. 57), **gazebo::event::EventT< void(const std::string &)>** (p. 57), **gazebo::event::EventT< void(std::map< std::string, std::string >)>** (p. 57), **gazebo::event::EventT< void()>** (p. 57), **gazebo::event::EventT< void(const common::UpdateInfo &)>** (p. 57), **gazebo::event::EventT< void(std::string, std::string)>** (p. 57), and **gazebo::event::EventT< void(bool)>** (p. 57).

10.78.3.2 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. 57), **gazebo::event::EventT< void(msgs::SonarStamped)>** (p. 57), **gazebo::event::EventT< void(std::string)>** (p. 57), **gazebo::event::EventT< void(msgs::WrenchStamped)>** (p. 57), **gazebo::event::EventT< void(const std::string &)>** (p. 57), **gazebo::event::EventT< void(std::map< std::string, std::string >)>** (p. 57), **gazebo::event::EventT< void()>** (p. 57), **gazebo::event::EventT< void(const common::UpdateInfo &)>** (p. 57), **gazebo::event::EventT< void(std::string, std::string)>** (p. 57), and **gazebo::event::EventT< void(bool)>** (p. 57).

10.78.3.3 bool gazebo::event::Event::GetSignaled ( ) const

Get whether this event has been signaled.

Returns

True if the event has been signaled.

10.78.4 Member Data Documentation

10.78.4.1 EventPrivate* gazebo::event::Event::dataPtr [protected]

Data pointer.

Referenced by gazebo::event::EventT< T >::EventT().

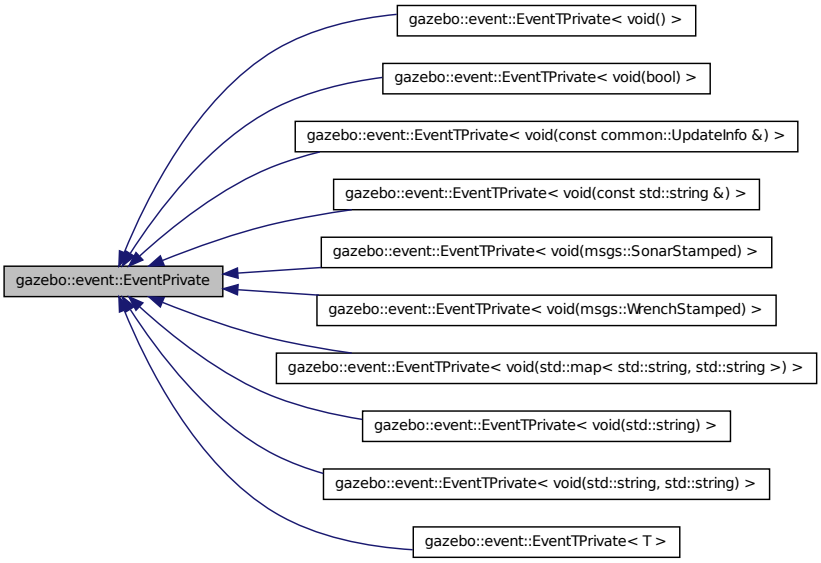
The documentation for this class was generated from the following file:

- **Event.hh**

10.79 gazebo::event::EventPrivate Class Reference

`#include <Event.hh>`

Inheritance diagram for gazebo::event::EventPrivate:



Public Member Functions

- **EventPrivate** ()

Public Attributes

- bool **signaled**

  True if the event has been signaled.

10.79.1 Constructor & Destructor Documentation

10.79.1.1 gazebo::event::EventPrivate::EventPrivate ( )

10.79.2 Member Data Documentation

10.79.2.1 bool gazebo::event::EventPrivate::signaled

True if the event has been signaled.

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

- **Event.hh**

10.80 gazebo::event::Events Class Reference

An **Event** (p. 514) 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 ConnectSigInt** (T _subscriber)

  Connect a boost::slot to the sigint event.
- 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.
- 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 **DisconnectSigInt** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot to the sigint event.
- 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 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()> **sigInt**

Simulation stop signal.

- static **EventT**< void()> **step**

  Step the simulation once signal.

- static **EventT**< void()> **stop**

  Simulation stop signal.

- static **EventT**< void(std::string)> **worldCreated**

  A world has been created.

- static **EventT**< void(const **common::UpdateInfo** &)> **worldUpdateBegin**

  World update has started.

- static **EventT**< void()> **worldUpdateEnd**

  World update has ended.

10.80.1 Detailed Description

An **Event** (p. 514) class to get notifications for simulator events.

10.80.2 Member Function Documentation

10.80.2.1 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

10.80.2.2 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

10.80.2.3 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

10.80.2.4 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

10.80.2.5 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

10.80.2.6 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

10.80.2.7 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

10.80.2.8 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

10.80.2.9 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

10.80.2.10 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

10.80.2.11 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSigInt (T _subscriber) [inline, static]

Connect a boost::slot to the sigint event.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

10.80.2.12 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

10.80.2.13 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::sensors::stop().

10.80.2.14 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

10.80.2.15 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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

10.80.2.17 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

10.80.2.18 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

10.80.2.19 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

10.80.2.20 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

10.80.2.21 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

10.80.2.22 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

10.80.2.23 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

10.80.2.24 static void gazebo::event::Events::DisconnectPreRender (ConnectionPtr *_subscriber*) [inline, static]

Disconnect a render start signal.

Parameters

in	*_subscriber*	the subscriber to this event

10.80.2.25 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

10.80.2.26 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

10.80.2.27 static void gazebo::event::Events::DisconnectSigInt (ConnectionPtr _subscriber) [inline, static]

Disconnect a boost::slot to the sigint event.

Parameters

in	*_subscriber*	the subscriber to this event

10.80.2.28 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

10.80.2.29 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::sensors::stop().

10.80.2.30 static void gazebo::event::Events::DisconnectWorldCreated (ConnectionPtr _subscriber) [inline, static]

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

10.80.2.31 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

10.80.2.32 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

10.80.3 Member Data Documentation

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

An entity has been added.

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

An entity has been deleted.

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

Diagnostic timer start.

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

Diagnostic timer stop.

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

An entity has been created.

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

Pause signal.

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

Post-Render.

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

Pre-render.

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

Render.

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

An entity has been selected.

10.80.3.11 EventT<void ()> gazebo::event::Events::sigInt [static]

Simulation stop signal.

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

Step the simulation once signal.

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

Simulation stop signal.

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

A world has been created.

10.80.3.15 EventT<void (const common::UpdateInfo &)> gazebo::event::Events::worldUpdateBegin [static]

World update has started.

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

World update has ended.

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

- **Events.hh**

10.81 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.

10.81.1 Detailed Description

Base class for rendering events.

10.81.2 Member Function Documentation

10.81.2.1 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.

10.81.2.2 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.

10.81.2.3 static void gazebo::rendering::Events::DisconnectCreateScene (event::ConnectionPtr _connection) [inline, static]

Disconnect from a scene created event.

Parameters

in	*_connection*	The connection to disconnect.

10.81.2.4 static void gazebo::rendering::Events::DisconnectRemoveScene (event::ConnectionPtr _connection) [inline, static]

Disconnect from a scene removed event.

Parameters

in	*_connection*	The connection to disconnect.

10.81.3 Member Data Documentation

10.81.3.1 event::EventT<void (const std::string &)> gazebo::rendering::Events::createScene [static]

The event used to trigger a create scene event.

10.81.3.2 event::EventT<void (const std::string &)> gazebo::rendering::Events::removeScene [static]

The event used to trigger a remve scene event.

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

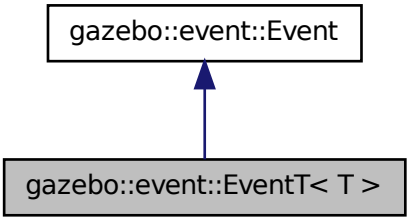
- **RenderEvents.hh**

10.82 gazebo::event::EventT< T > Class Template Reference

A class for event processing.

`#include <common/common.hh>`

Inheritance diagram for gazebo::event::EventT< T >:



Public Member Functions

- **EventT** ()

  Constructor.
- 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.

10.82.1 Detailed Description

template<typename T>class gazebo::event::EventT< T >

A class for event processing.

10.82.2 Member Function Documentation

10.82.2.1 template<typename T> void gazebo::event::EventT< T >::operator() ( ) [inline]

Access the signal.

10.82.2.2 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.

10.82.2.3 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.

10.82.2.4 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.

10.82.2.5 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.

10.82.2.6 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.

10.82.2.7 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.

10.82.2.8 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.

10.82.2.9 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.

10.82.2.10 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.

10.82.2.11 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.

10.82.2.12 template<typename T> void gazebo::event::EventT< T >::Signal ( ) [inline]

Signal the event for all subscribers.

10.82.2.13 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.

10.82.2.14 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.

10.82.2.15 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.

10.82.2.16 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.

10.82.2.17 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.

10.82.2.18 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.

10.82.2.19 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.

10.82.2.20 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.

10.82.2.21 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.

10.82.2.22 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.

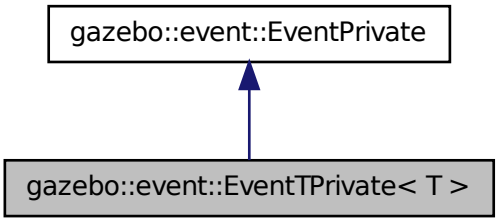
The documentation for this class was generated from the following file:

- **Event.hh**

10.83 gazebo::event::EventTPrivate< T > Class Template Reference

`#include <Event.hh>`

Inheritance diagram for gazebo::event::EventTPrivate< T >:



Public Attributes

- EvtConnectionMap **connections**

  Array of connection callbacks.
- boost::mutex **connectionsEraseMutex**

  A thread lock.
- std::vector< int > **connectionsToErase**

  Set of connections to erased.

template<typename T> class gazebo::event::EventTPrivate< T >

10.83.1 Member Data Documentation

10.83.1.1 template<typename T> EvtConnectionMap gazebo::event::EventTPrivate< T >::connections

Array of connection callbacks.

10.83.1.2 template<typename T> boost::mutex gazebo::event::EventTPrivate< T >::connectionsEraseMutex

A thread lock.

10.83.1.3 template<typename T> std::vector<int> gazebo::event::EventTPrivate< T >::connectionsToErase

Set of connections to erased.

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

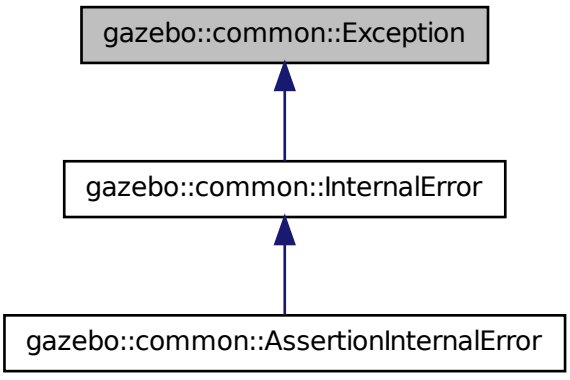
- **Event.hh**

10.84 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

10.84.1 Detailed Description

Class for generating exceptions.

10.84.2 Constructor & Destructor Documentation

10.84.2.1 gazebo::common::Exception::Exception ( )

Constructor.

10.84.2.2 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

10.84.2.3 virtual gazebo::common::Exception::~Exception ( ) [virtual]

Destructor.

10.84.3 Member Function Documentation

10.84.3.1 std::string gazebo::common::Exception::GetErrorFile ( ) const

Return the error function.

Returns

The error function name

10.84.3.2 std::string gazebo::common::Exception::GetErrorStr ( ) const

Return the error string.

Returns

The error string

10.84.3.3 void gazebo::common::Exception::Print ( ) const

Print the exception to std out.

10.84.4 Friends And Related Function Documentation

10.84.4.1 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:

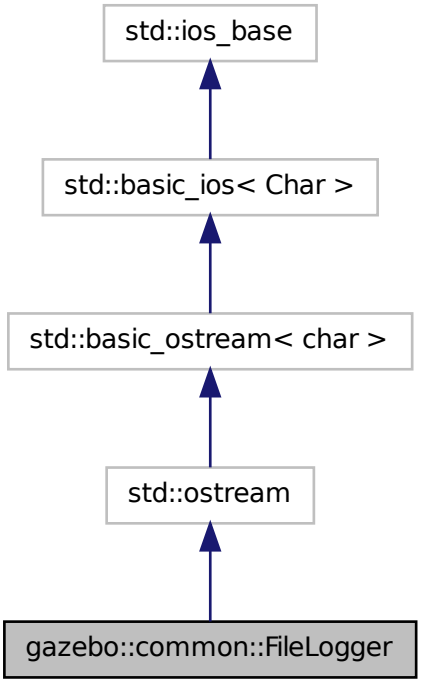
- **Exception.hh**

10.85 gazebo::common::FileLogger Class Reference

A logger that outputs messages to a file.

`#include <Console.hh>`

Inheritance diagram for gazebo::common::FileLogger:



Classes

- class **Buffer**

  String buffer for the file logger.

Public Member Functions

- **FileLogger** (const std::string &_filename="")

  Constructor.
- virtual ~**FileLogger** ()

  Destructor.
- void **Init** (const std::string &_filename)

  Initialize the file logger.
- virtual **FileLogger** & **operator()** ()

  Output a filename and line number, then return a reference to the logger.
- virtual **FileLogger** & **operator()** (const std::string &_file, int _line)

  Output a filename and line number, then return a reference to the logger.

10.85.1 Detailed Description

A logger that outputs messages to a file.

10.85.2 Constructor & Destructor Documentation

10.85.2.1 gazebo::common::FileLogger::FileLogger (const std::string & *_filename* = "")

Constructor.

Parameters

in	*_filename*	Filename to write into. If empty, **FileLogger::Init** (p. 552) must be called separately.

10.85.2.2 virtual gazebo::common::FileLogger::~FileLogger ( ) [virtual]

Destructor.

10.85.3 Member Function Documentation

10.85.3.1 void gazebo::common::FileLogger::Init (const std::string & *_filename*)

Initialize the file logger.

Parameters

in	*_filename*	Name and path of the log file to write output into.

10.85.3.2 virtual FileLogger& gazebo::common::FileLogger::operator() ( ) [virtual]

Output a filename and line number, then return a reference to the logger.

Returns

Reference to this logger.

10.85.3.3 virtual FileLogger& gazebo::common::FileLogger::operator() (const std::string & *_file,* int *_line*) [virtual]

Output a filename and line number, then return a reference to the logger.

Parameters

in	*_file*	Filename to output.
in	*_line*	Line number in the _file.

Returns

Reference to this logger.

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

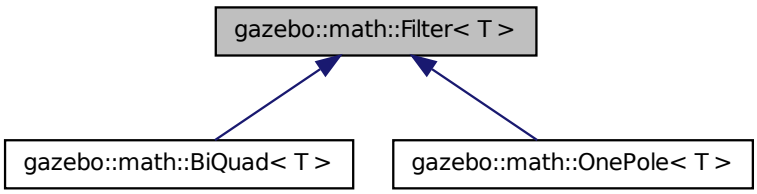
- **Console.hh**

10.86 gazebo::math::Filter< T > Class Template Reference

Filter (p. 553) base class.

`#include <math/gzmath.hh>`

Inheritance diagram for gazebo::math::Filter< T >:



Public Member Functions

- virtual ~**Filter** ()

  Destructor.
- virtual const T & **GetValue** ()

  Get the output of the filter.
- virtual void **SetFc** (double _fc, double _fs)=0

  Set the cutoff frequency and sample rate.
- virtual void **SetValue** (const T &_val)

  Set the output of the filter.

Protected Attributes

- T **y0**

  Output.

10.86.1 Detailed Description

template<class T>class gazebo::math::Filter< T >

Filter (p. 553) base class.

10.86.2 Constructor & Destructor Documentation

10.86.2.1 template<class T> virtual gazebo::math::Filter< T >::~Filter ( ) [inline, virtual]

Destructor.

10.86.3 Member Function Documentation

10.86.3.1 template<class T> virtual const T& gazebo::math::Filter< T >::GetValue ( ) [inline, virtual]

Get the output of the filter.

Returns

Filter (p. 553)'s output.

10.86.3.2 template<class T> virtual void gazebo::math::Filter< T >::SetFc (double *_fc*, double *_fs*) [pure virtual]

Set the cutoff frequency and sample rate.

Parameters

in	*_fc*	Cutoff frequency.
in	*_fs*	Sample rate.

Implemented in **gazebo::math::BiQuad< T >** (p. 219), **gazebo::math::BiQuad< math::Vector3 >** (p. 219), **gazebo::math::OnePole< T >** (p. 950), **gazebo::math::OnePole< math::Quaternion >** (p. 950), and **gazebo::math::OnePole< math::Vector3 >** (p. 950).

10.86.3.3 template<class T> virtual void gazebo::math::Filter< T >::SetValue (const T & *_val*) [inline, virtual]

Set the output of the filter.

Parameters

in	*_val*	New value.

Reimplemented in **gazebo::math::BiQuad< T >** (p. 220), and **gazebo::math::BiQuad< math::Vector3 >** (p. 220).

10.86.4 Member Data Documentation

10.86.4.1 template<class T> T gazebo::math::Filter< T >::y0 [protected]

Output.

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

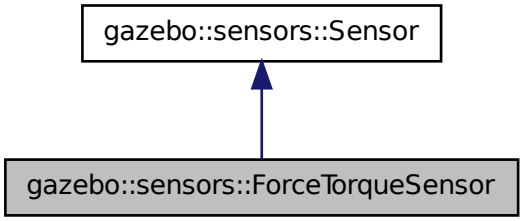
- **Filter.hh**

10.87 gazebo::sensors::ForceTorqueSensor Class Reference

Sensor (p. 1130) for measure force and torque on a joint.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::ForceTorqueSensor:



Public Member Functions

- **ForceTorqueSensor** ()

  Constructor.
- virtual ~**ForceTorqueSensor** ()

  Destructor.
- template<typename T >
  event::ConnectionPtr ConnectUpdate (T _subscriber)

  Connect a to the update signal.
- void **DisconnectUpdate** (**event::ConnectionPtr** &_conn)

Disconnect from the update signal.

- **math::Vector3 GetForce** () const

  Get the current joint force.

- **physics::JointPtr GetJoint** () const

  Get Parent Joint.

- virtual std::string **GetTopic** () const

  Returns the topic name as set in SDF.

- **math::Vector3 GetTorque** () const

  Get the current joint torque.

- 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.

- void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)

  Load the sensor with SDF parameters.

- virtual bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

Protected Attributes

- **event::EventT**< void(msgs::WrenchStamped)> **update**

  Update event.

10.87.1 Detailed Description

Sensor (p. 1130) for measure force and torque on a joint.

10.87.2 Constructor & Destructor Documentation

10.87.2.1 gazebo::sensors::ForceTorqueSensor::ForceTorqueSensor ( )

Constructor.

10.87.2.2 virtual gazebo::sensors::ForceTorqueSensor::~ForceTorqueSensor ( ) [virtual]

Destructor.

10.87.3 Member Function Documentation

10.87.3.1 template<typename T > event::ConnectionPtr gazebo::sensors::ForceTorqueSensor::ConnectUpdate (T *_subscriber*) [inline]

Connect a to the update signal.

Parameters

in	*_subscriber*	Callback function.

Returns

The connection, which must be kept in scope.

10.87.3.2 void gazebo::sensors::ForceTorqueSensor::DisconnectUpdate (event::ConnectionPtr & *_conn*) [inline]

Disconnect from the update signal.

Parameters

in	*_conn*	Connection to remove.

10.87.3.3 virtual void gazebo::sensors::ForceTorqueSensor::Fini ( ) [protected, virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.87.3.4 math::Vector3 gazebo::sensors::ForceTorqueSensor::GetForce ( ) const

Get the current joint force.

Returns

The latested measured force.

10.87.3.5 physics::JointPtr gazebo::sensors::ForceTorqueSensor::GetJoint ( ) const

Get Parent Joint.

Returns

Pointer to the joint containing this sensor

10.87.3.6 virtual std::string gazebo::sensors::ForceTorqueSensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 1138).

10.87.3.7 math::Vector3 gazebo::sensors::ForceTorqueSensor::GetTorque ( ) const

Get the current joint torque.

Returns

The latested measured torque.

10.87.3.8 virtual void gazebo::sensors::ForceTorqueSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.87.3.9 virtual bool gazebo::sensors::ForceTorqueSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.87.3.10 void gazebo::sensors::ForceTorqueSensor::Load (const std::string & *_worldName,* sdf::ElementPtr *_sdf*) [protected, virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 1130) parameters.
in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 1140).

10.87.3.11 virtual void gazebo::sensors::ForceTorqueSensor::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. 1140).

10.87.3.12 virtual bool gazebo::sensors::ForceTorqueSensor::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

10.87.4 Member Data Documentation

10.87.4.1 event::EventT<void(msgs::WrenchStamped)> gazebo::sensors::ForceTorqueSensor::update [protected]

Update event.

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

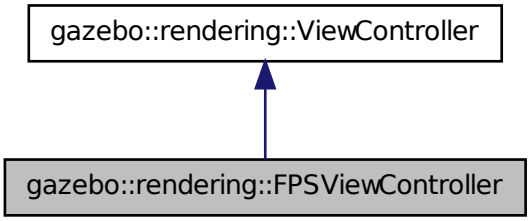
- **ForceTorqueSensor.hh**

10.88 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.

10.88.1 Detailed Description

First Person Shooter style view controller.

10.88.2 Constructor & Destructor Documentation

10.88.2.1 gazebo::rendering::FPSViewController::FPSViewController (UserCameraPtr _camera)

Constructor.

Parameters

in	***Camera*** *(p. 242)*	to controll

10.88.2.2 virtual gazebo::rendering::FPSViewController::~FPSViewController ( ) [virtual]

Destructor.

10.88.3 Member Function Documentation

10.88.3.1 static std::string gazebo::rendering::FPSViewController::GetTypeString ( ) [static]

Get the type name of this view controller.

Returns

The name of the controller type: "fps"

10.88.3.2 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. 1475).

10.88.3.3 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. 1475).

10.88.3.4 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. 1476).

10.88.3.5 virtual void gazebo::rendering::FPSViewController::Init ( ) [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 1476).

10.88.3.6 virtual void gazebo::rendering::FPSViewController::Update ( ) [virtual]

Update the camera position.

Implements **gazebo::rendering::ViewController** (p. 1476).

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

- **FPSViewController.hh**

10.89 gazebo::physics::FrictionPyramid Class Reference

Parameters used for friction pyramid model.

`#include <physics/physics.hh>`

Public Member Functions

- **FrictionPyramid** ()
  Constructor.
- virtual ~**FrictionPyramid** ()
  Destructor.
- double **GetMuPrimary** ()
  Get the friction coefficient in the primary direction.
- double **GetMuSecondary** ()
  Get the friction coefficient in the secondary direction.
- void **SetMuPrimary** (double _mu)
  Set the friction coefficient in the primary direction.
- void **SetMuSecondary** (double _mu)
  Set the friction coefficient in the secondary direction.

Public Attributes

- **math::Vector3 direction1**

  Vector for specifying the primary friction direction, relative to the parent collision frame.

10.89.1 Detailed Description

Parameters used for friction pyramid model.

10.89.2 Constructor & Destructor Documentation

10.89.2.1 gazebo::physics::FrictionPyramid::FrictionPyramid ( )

Constructor.

10.89.2.2 virtual gazebo::physics::FrictionPyramid::~FrictionPyramid ( ) [virtual]

Destructor.

10.89.3 Member Function Documentation

10.89.3.1 double gazebo::physics::FrictionPyramid::GetMuPrimary ( )

Get the friction coefficient in the primary direction.

Returns

Friction coefficient in primary direction.

10.89.3.2 double gazebo::physics::FrictionPyramid::GetMuSecondary ( )

Get the friction coefficient in the secondary direction.

Returns

Friction coefficient in secondary direction.

10.89.3.3 void gazebo::physics::FrictionPyramid::SetMuPrimary (double _mu)

Set the friction coefficient in the primary direction.

Parameters

in	*_mu*	Friction coefficient.

10.89.3.4 void gazebo::physics::FrictionPyramid::SetMuSecondary (double _mu)

Set the friction coefficient in the secondary direction.

Parameters

in	*_mu*	Friction coefficient.

10.89.4 Member Data Documentation

10.89.4.1 math::Vector3 gazebo::physics::FrictionPyramid::direction1

Vector for specifying the primary friction direction, relative to the parent collision frame.

The component of this vector that is orthogonal to the surface normal will be set as the primary friction direction. 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

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

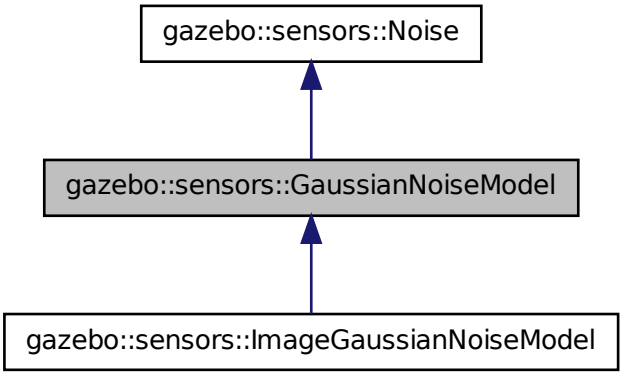
- **SurfaceParams.hh**

10.90 gazebo::sensors::GaussianNoiseModel Class Reference

Gaussian noise class.

#include <GaussianNoiseModel.hh>

Inheritance diagram for gazebo::sensors::GaussianNoiseModel:



Public Member Functions

- **GaussianNoiseModel** ()

  Constructor.
- virtual ~**GaussianNoiseModel** ()

  Destructor.
- double **ApplyImpl** (double _in)

  Apply noise to input data value.
- virtual void **Fini** ()

  Finalize the noise model.
- double **GetBias** () const

  Accessor for bias.
- double **GetMean** () const

  Accessor for mean.
- double **GetStdDev** () const

  Accessor for stddev.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load noise parameters from sdf.

Protected Attributes

- double **bias**

  If type starts with GAUSSIAN, the bias we'll add.
- double **mean**

  If type starts with GAUSSIAN, the mean of the distribution from which we sample when adding noise.
- double **precision**

  If type==GAUSSIAN_QUANTIZED, the precision to which the output signal is rounded.
- bool **quantized**

  True if the type is GAUSSIAN_QUANTIZED.
- double **stdDev**

  If type starts with GAUSSIAN, the standard deviation of the distribution from which we sample when adding noise.

10.90.1 Detailed Description

Gaussian noise class.

Gaussian noise class for image sensors.

10.90.2 Constructor & Destructor Documentation

10.90.2.1 gazebo::sensors::GaussianNoiseModel::GaussianNoiseModel ( )

Constructor.

10.90.2.2 virtual gazebo::sensors::GaussianNoiseModel::~GaussianNoiseModel ( ) [virtual]

Destructor.

10.90.3 Member Function Documentation

10.90.3.1 double gazebo::sensors::GaussianNoiseModel::ApplyImpl (double *_in*) [virtual]

Apply noise to input data value.

This gets overriden by derived classes, and called by Apply.

Parameters

in	*_in*	Input data value.

Returns

Data with noise applied.

Reimplemented from **gazebo::sensors::Noise** (p. 933).

10.90.3.2 virtual void gazebo::sensors::GaussianNoiseModel::Fini ( ) [virtual]

Finalize the noise model.

Reimplemented from **gazebo::sensors::Noise** (p. 934).

Reimplemented in **gazebo::sensors::ImageGaussianNoiseModel** (p. 645).

10.90.3.3 double gazebo::sensors::GaussianNoiseModel::GetBias ( ) const

Accessor for bias.

Returns

Bias on output.

10.90.3.4 double gazebo::sensors::GaussianNoiseModel::GetMean ( ) const

Accessor for mean.

Returns

Mean of Gaussian noise.

10.90.3.5 double gazebo::sensors::GaussianNoiseModel::GetStdDev ( ) const

Accessor for stddev.

Returns

Standard deviation of Gaussian noise.

10.90.3.6 virtual void gazebo::sensors::GaussianNoiseModel::Load (sdf::ElementPtr _sdf) [virtual]

Load noise parameters from sdf.

Parameters

in	*_sdf*	SDF parameters.
in	*_sensor*	Type of sensor.

Reimplemented from **gazebo::sensors::Noise** (p. 934).

Reimplemented in **gazebo::sensors::ImageGaussianNoiseModel** (p. 646).

10.90.4 Member Data Documentation

10.90.4.1 double gazebo::sensors::GaussianNoiseModel::bias [protected]

If type starts with GAUSSIAN, the bias we'll add.

10.90.4.2 double gazebo::sensors::GaussianNoiseModel::mean [protected]

If type starts with GAUSSIAN, the mean of the distribution from which we sample when adding noise.

10.90.4.3 double gazebo::sensors::GaussianNoiseModel::precision [protected]

If type==GAUSSIAN_QUANTIZED, the precision to which the output signal is rounded.

10.90.4.4 bool gazebo::sensors::GaussianNoiseModel::quantized [protected]

True if the type is GAUSSIAN_QUANTIZED.

10.90.4.5 double gazebo::sensors::GaussianNoiseModel::stdDev [protected]

If type starts with GAUSSIAN, the standard deviation of the distribution from which we sample when adding noise.

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

- **GaussianNoiseModel.hh**

10.91 google::protobuf::compiler::cpp::GazeboGenerator Class Reference

Google protobuf message generator for **gazebo::msgs** (p. 135).

`#include <GazeboGenerator.hh>`

Public Member Functions

- **GazeboGenerator** (const std::string &_name)
- virtual ~**GazeboGenerator** ()
- virtual bool **Generate** (const FileDescriptor *file, const string &parameter, OutputDirectory *directory, string *error) const

10.91.1 Detailed Description

Google protobuf message generator for **gazebo::msgs** (p. 135).

10.91.2 Constructor & Destructor Documentation

10.91.2.1 google::protobuf::compiler::cpp::GazeboGenerator::GazeboGenerator (const std::string & *_name*)

10.91.2.2 virtual google::protobuf::compiler::cpp::GazeboGenerator::~GazeboGenerator ( ) [virtual]

10.91.3 Member Function Documentation

10.91.3.1 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**

10.92 gazebo::physics::GearboxJoint< T > Class Template - Reference

A double axis gearbox joint.

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

Public Member Functions

- **GearboxJoint** (**BasePtr** _parent)
  Constructor.
- virtual ~**GearboxJoint** ()
  Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual double **GetGearboxRatio** () const
  Get gearbox joint gear ratio.
- virtual void **Load** (sdf::ElementPtr _sdf)
  Load joint.
- virtual void **SetGearboxRatio** (double _gearRatio)=0
  Set gearbox joint gear ratio.

Protected Member Functions

- virtual void **Init** ()
  Initialize joint.

Protected Attributes

- double **gearRatio**
  Gearbox gearRatio.
- std::string **referenceBody**
  reference link/body for computing joint angles

10.92.1 Detailed Description

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

A double axis gearbox joint.

10.92.2 Constructor & Destructor Documentation

10.92.2.1 template<class T > gazebo::physics::GearboxJoint< T >::GearboxJoint (BasePtr *_parent*) [inline]

Constructor.

Parameters

in	*_parent*	Parent link

References gazebo::physics::Base::GEARBOX_JOINT.

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

Destructor.

10.92.3 Member Function Documentation

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

10.92.3.2 template<class T > virtual double gazebo::physics::GearboxJoint< T >::GetGearboxRatio ( ) const [inline, virtual]

Get gearbox joint gear ratio.

Returns

Gear ratio value.

10.92.3.3 template<class T > virtual void gazebo::physics::GearboxJoint< T >::Init ( ) [inline, protected, virtual]

Initialize joint.

References gazebo::msgs::Init().

10.92.3.4 template<class T > virtual void gazebo::physics::GearboxJoint< T >::Load (sdf::ElementPtr *_sdf*) [inline, virtual]

Load joint.

Parameters

in	*_sdf*	Pointer to SDF element

References gzerr.

10.92.3.5 template<class T > virtual void gazebo::physics::GearboxJoint< T >::SetGearboxRatio (double *_gearRatio*) [pure virtual]

Set gearbox joint gear ratio.

This must be implemented in a child class

Parameters

in	*_index*	Index of the axis.
in	*_gearRatio*	Gear ratio value.

10.92.4 Member Data Documentation

10.92.4.1 template<class T > double gazebo::physics::GearboxJoint< T >::gearRatio [protected]

Gearbox gearRatio.

10.92.4.2 template<class T > std::string gazebo::physics::GearboxJoint< T >::referenceBody [protected]

reference link/body for computing joint angles

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

- **GearboxJoint.hh**

10.93 gazebo::common::GeometryIndices Class Reference

Helper data structure for loading collada geometries.

`#include <ColladaLoaderPrivate.hh>`

Public Attributes

- unsigned int **mappedIndex**

  Index of a vertex in the Gazebo mesh.
- unsigned int **normalIndex**

  Index of a normal in the collada.
- unsigned int **texcoordIndex**

  Index of a texture coordinate in the collada.

- unsigned int **vertexIndex**

  Index of a vertex in the collada.

10.93.1 Detailed Description

Helper data structure for loading collada geometries.

10.93.2 Member Data Documentation

10.93.2.1 unsigned int gazebo::common::GeometryIndices::mappedIndex

Index of a vertex in the Gazebo mesh.

10.93.2.2 unsigned int gazebo::common::GeometryIndices::normalIndex

Index of a normal in the collada.

element

10.93.2.3 unsigned int gazebo::common::GeometryIndices::texcoordIndex

Index of a texture coordinate in the collada.

element

10.93.2.4 unsigned int gazebo::common::GeometryIndices::vertexIndex

Index of a vertex in the collada.

element

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

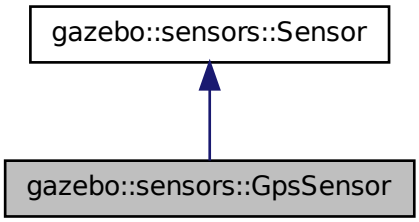
- **ColladaLoaderPrivate.hh**

10.94 gazebo::sensors::GpsSensor Class Reference

GpsSensor (p. 575) to provide position measurement.

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

Inheritance diagram for gazebo::sensors::GpsSensor:



Public Member Functions

- **GpsSensor** ()
  Constructor.
- virtual ~**GpsSensor** ()
  Destructor.
- virtual void **Fini** ()
  Finalize the sensor.
- double **GetAltitude** () const
  Accessor for current altitude.
- **math::Angle GetLatitude** () const
  Accessor for current latitude angle.
- **math::Angle GetLongitude** () const
  Accessor for current longitude angle.
- 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 bool **UpdateImpl** (bool _force)
  This gets overwritten by derived sensor types.

10.94.1 Detailed Description

GpsSensor (p. 575) to provide position measurement.

10.94.2 Constructor & Destructor Documentation

10.94.2.1 gazebo::sensors::GpsSensor::GpsSensor ( )

Constructor.

10.94.2.2 virtual gazebo::sensors::GpsSensor::~GpsSensor ( ) [virtual]

Destructor.

10.94.3 Member Function Documentation

10.94.3.1 virtual void gazebo::sensors::GpsSensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.94.3.2 double gazebo::sensors::GpsSensor::GetAltitude ( ) const

Accessor for current altitude.

Returns

Current altitude above sea level.

10.94.3.3 math::Angle gazebo::sensors::GpsSensor::GetLatitude ( ) const

Accessor for current latitude angle.

Returns

Current latitude angle.

10.94.3.4 math::Angle gazebo::sensors::GpsSensor::GetLongitude ( ) const

Accessor for current longitude angle.

Returns

Current longitude angle.

10.94.3.5 virtual void gazebo::sensors::GpsSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.94.3.6 virtual void gazebo::sensors::GpsSensor::Load (const std::string & _worldName, sdf::ElementPtr _sdf) [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 1130) parameters.
in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 1140).

10.94.3.7 virtual void gazebo::sensors::GpsSensor::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. 1140).

10.94.3.8 virtual bool gazebo::sensors::GpsSensor::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

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

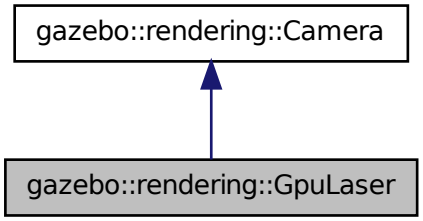
- **GpsSensor.hh**

10.95 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, **ScenePtr** _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.
- double **GetCameraCount** () const

  Get the number of cameras required.
- double **GetCosHorzFOV** () const

  Get Cos Horz field-of-view.
- double **GetCosVertFOV** () const

  Get Cos Vert field-of-view.
- double **GetFarClip** () const

  Get far clip.
- 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.
- const float ∗ **GetLaserData** ()

  All things needed to get back z buffer for laser data.
- double **GetNearClip** () const

  Get near clip.
- double **GetRayCountRatio** () const

  Get the ray count ratio (equivalent to aspect ratio)
- double **GetVertFOV** () const

  Get the vertical field-of-view.
- double **GetVertHalfAngle** () const

  Get (vertical_max_angle + vertical_min_angle) ∗ 0.5.
- virtual void **Init** ()

  Initialize the camera.
- bool **IsHorizontal** () const

  Gets if sensor is horizontal.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the camera with a set of parmeters.
- 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 **SetCameraCount** (double _cameraCount)

  Set the number of cameras required.
- void **SetCosHorzFOV** (double _chfov)

  Set the Cos Horz FOV.
- void **SetCosVertFOV** (double _cvfov)

  Set the Cos Horz FOV.
- void **SetFarClip** (double _far)

  Set the far clip distance.
- void **SetHorzFOV** (double _hfov)

  Set the horizontal fov.
- void **SetHorzHalfAngle** (double _angle)

  Set the horizontal half angle.
- void **SetIsHorizontal** (bool _horizontal)

  Set sensor horizontal or vertical.
- void **SetNearClip** (double _near)

  Set the near clip distance.
- void **SetRangeCount** (unsigned int _w, unsigned int _h=1)

  Set the number of laser samples in the width and height.
- void **SetRayCountRatio** (double _rayCountRatio)

  Sets the ray count ratio (equivalen to aspect ratio)
- void **SetVertFOV** (double _vfov)

  Set the vertical fov.
- void **SetVertHalfAngle** (double _angle)

  Set the vertical half angle.

Protected Attributes

- unsigned int **cameraCount**

  Number of cameras needed to generate the rays.
- 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.
- double **horzHalfAngle**

  Horizontal half angle.
- bool **isHorizontal**

  True if the sensor is horizontal only.
- double **near**

  Near clip plane.
- double **rayCountRatio**

  Ray count ratio.
- double **vertHalfAngle**

  Vertical half angle.
- double **vfov**

  Vertical field-of-view.

10.95.1 Detailed Description

GPU based laser distance sensor.

10.95.2 Constructor & Destructor Documentation

10.95.2.1 gazebo::rendering::GpuLaser::GpuLaser (const std::string & *_namePrefix,* ScenePtr *_scene,* bool *_autoRender* = `true`)

Constructor.

Parameters

`in`	*_namePrefix*	Unique prefix name for the camera.
`in`	*_scene*	**Scene** (p. 1097) that will contain the camera
`in`	*_auto-Render*	Almost everyone should leave this as true.

10.95.2.2 virtual gazebo::rendering::GpuLaser::~GpuLaser ( ) `[virtual]`

Destructor.

10.95.3 Member Function Documentation

10.95.3.1 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 pointer to the connection. This must be kept in scope.

10.95.3.2 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.

10.95.3.3 void gazebo::rendering::GpuLaser::DisconnectNewLaserFrame (event::ConnectionPtr & _c) [inline]

Disconnect from a laser frame signal.

Parameters

in	*_c*	The connection to disconnect

10.95.3.4 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. 254).

10.95.3.5 double gazebo::rendering::GpuLaser::GetCameraCount ( ) const

Get the number of cameras required.

Returns

Number of cameras needed to generate the rays

10.95.3.6 double gazebo::rendering::GpuLaser::GetCosHorzFOV ( ) const

Get Cos Horz field-of-view.

Returns

2 ∗ atan(tan(this->hfov/2) / cos(this->vfov/2))

10.95.3.7 double gazebo::rendering::GpuLaser::GetCosVertFOV ( ) const

Get Cos Vert field-of-view.

Returns

2 ∗ atan(tan(this->vfov/2) / cos(this->hfov/2))

10.95.3.8 double gazebo::rendering::GpuLaser::GetFarClip ( ) const

Get far clip.

Returns

far clip distance

10.95.3.9 double gazebo::rendering::GpuLaser::GetHorzFOV ( ) const

Get the horizontal field of view of the laser sensor.

Returns

The horizontal field of view of the laser sensor.

10.95.3.10 double gazebo::rendering::GpuLaser::GetHorzHalfAngle ( ) const

Get (horizontal_max_angle + horizontal_min_angle) ∗ 0.5.

Returns

(horizontal_max_angle + horizontal_min_angle) ∗ 0.5

10.95.3.11 const float∗ gazebo::rendering::GpuLaser::GetLaserData ( )

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

Returns

Array of laser data.

10.95.3.12 double gazebo::rendering::GpuLaser::GetNearClip ( ) const

Get near clip.

Returns

near clip distance

10.95.3.13 double gazebo::rendering::GpuLaser::GetRayCountRatio ( ) const

Get the ray count ratio (equivalent to aspect ratio)

Returns

The ray count ratio (equivalent to aspect ratio)

10.95.3.14 double gazebo::rendering::GpuLaser::GetVertFOV ( ) const

Get the vertical field-of-view.

Returns

The vertical field of view of the laser sensor.

10.95.3.15 double gazebo::rendering::GpuLaser::GetVertHalfAngle ( ) const

Get (vertical_max_angle + vertical_min_angle) ∗ 0.5.

Returns

(vertical_max_angle + vertical_min_angle) ∗ 0.5

10.95.3.16 virtual void gazebo::rendering::GpuLaser::Init ( ) [virtual]

Initialize the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 263).

10.95.3.17 bool gazebo::rendering::GpuLaser::IsHorizontal ( ) const

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

10.95.3.18 virtual void gazebo::rendering::GpuLaser::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the camera with a set of parmeters.

Parameters

in	*_sdf*	The SDF camera info

Reimplemented from **gazebo::rendering::Camera** (p. 264).

10.95.3.19 virtual void gazebo::rendering::GpuLaser::Load ( ) [virtual]

Load the camera with default parmeters.

Reimplemented from **gazebo::rendering::Camera** (p. 264).

10.95.3.20 virtual void gazebo::rendering::GpuLaser::notifyRenderSingleObject (Ogre::Renderable * _rend, const Ogre::Pass * _p, const Ogre::AutoParamDataSource * _s, const Ogre::LightList * _ll, bool _supp) [virtual]

10.95.3.21 virtual void gazebo::rendering::GpuLaser::PostRender ( ) [virtual]

Post render.

Called afer the render signal.

Reimplemented from **gazebo::rendering::Camera** (p. 265).

10.95.3.22 void gazebo::rendering::GpuLaser::SetCameraCount (double _cameraCount)

Set the number of cameras required.

Parameters

in	*_camera-Count*	The number of cameras required to generate the rays

10.95.3.23 void gazebo::rendering::GpuLaser::SetCosHorzFOV (double _chfov)

Set the Cos Horz FOV.

Parameters

in	*_chfov*	Cos Horz FOV

10.95.3.24 void gazebo::rendering::GpuLaser::SetCosVertFOV (double _cvfov)

Set the Cos Horz FOV.

Parameters

in	*_cvfov*	Cos Horz FOV

10.95.3.25 void gazebo::rendering::GpuLaser::SetFarClip (double _far)

Set the far clip distance.

Parameters

in	*_far*	far clip distance

10.95.3.26 void gazebo::rendering::GpuLaser::SetHorzFOV (double *_hfov*)

Set the horizontal fov.

Parameters

in	*_hfov*	horizontal fov

10.95.3.27 void gazebo::rendering::GpuLaser::SetHorzHalfAngle (double *_angle*)

Set the horizontal half angle.

Parameters

in	*_angle*	horizontal half angle

10.95.3.28 void gazebo::rendering::GpuLaser::SetIsHorizontal (bool *_horizontal*)

Set sensor horizontal or vertical.

Parameters

in	*_horizontal*	True if horizontal, false if not

10.95.3.29 void gazebo::rendering::GpuLaser::SetNearClip (double *_near*)

Set the near clip distance.

Parameters

in	*_near*	near clip distance

10.95.3.30 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

10.95.3.31 void gazebo::rendering::GpuLaser::SetRayCountRatio (double *_rayCountRatio*)

Sets the ray count ratio (equivalen to aspect ratio)

Parameters

in	*_rayCount-Ratio*	ray count ratio (equivalent to aspect ratio)

10.95.3.32 void gazebo::rendering::GpuLaser::SetVertFOV (double *_vfov*)

Set the vertical fov.

Parameters

in	*_vfov*	vertical fov

10.95.3.33 void gazebo::rendering::GpuLaser::SetVertHalfAngle (double *_angle*)

Set the vertical half angle.

Parameters

in	*_angle*	vertical half angle

10.95.4 Member Data Documentation

10.95.4.1 unsigned int gazebo::rendering::GpuLaser::cameraCount [protected]

Number of cameras needed to generate the rays.

10.95.4.2 double gazebo::rendering::GpuLaser::chfov [protected]

Cos horizontal field-of-view.

10.95.4.3 double gazebo::rendering::GpuLaser::cvfov [protected]

Cos vertical field-of-view.

10.95.4.4 double gazebo::rendering::GpuLaser::far [protected]

Far clip plane.

10.95.4.5 double gazebo::rendering::GpuLaser::hfov [protected]

Horizontal field-of-view.

10.95.4.6 double gazebo::rendering::GpuLaser::horzHalfAngle [protected]

Horizontal half angle.

10.95.4.7 bool gazebo::rendering::GpuLaser::isHorizontal [protected]

True if the sensor is horizontal only.

10.95.4.8 double gazebo::rendering::GpuLaser::near [protected]

Near clip plane.

10.95.4.9 double gazebo::rendering::GpuLaser::rayCountRatio [protected]

Ray count ratio.

10.95.4.10 double gazebo::rendering::GpuLaser::vertHalfAngle [protected]

Vertical half angle.

10.95.4.11 double gazebo::rendering::GpuLaser::vfov [protected]

Vertical field-of-view.

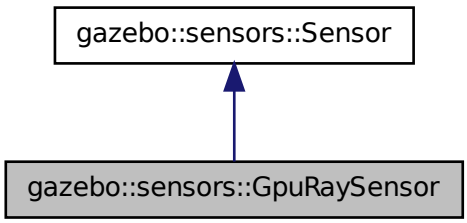
The documentation for this class was generated from the following file:

- **GpuLaser.hh**

10.96 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. *579).*

- 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)

  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.

- virtual std::string **GetTopic** () const

  Returns the topic name as set in SDF.

- 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.

- double **GetVerticalAngleResolution** () const

  Get the vertical angle in radians between each range.

- int **GetVerticalRangeCount** () const

  Get the vertical scan line count.

- int **GetVerticalRayCount** () const

  Get the vertical scan line count.

- virtual void **Init** ()

  Initialize the ray.

- virtual bool **IsActive** ()

  Returns true if sensor generation is active.

- 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 bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

Protected Attributes

- sdf::ElementPtr **cameraElem**

  Camera SDF element.
- sdf::ElementPtr **horzElem**

  Horizontal SDF element.
- unsigned int **horzRangeCount**

  Horizontal range count.
- unsigned int **horzRayCount**

  Horizontal ray count.
- double **rangeCountRatio**

  Range count ratio.
- sdf::ElementPtr **rangeElem**

  Range SDF element.
- sdf::ElementPtr **scanElem**

  Scan SDF elementz.
- sdf::ElementPtr **vertElem**

  Vertical SDF element.
- unsigned int **vertRangeCount**

  Vertical range count.
- unsigned int **vertRayCount**

  Vertical ray count.

10.96.1 Constructor & Destructor Documentation

10.96.1.1 gazebo::sensors::GpuRaySensor::GpuRaySensor ( )

Constructor.

10.96.1.2 virtual gazebo::sensors::GpuRaySensor::~GpuRaySensor ( ) [virtual]

Destructor.

10.96.2 Member Function Documentation

10.96.2.1 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.

10.96.2.2 void gazebo::sensors::GpuRaySensor::DisconnectNewLaserFrame (event::ConnectionPtr & *_conn*)

Disconnect Laser Frame.

Parameters

in,out	*_conn*	Connection pointer to disconnect.

10.96.2.3 virtual void gazebo::sensors::GpuRaySensor::Fini ( ) [protected, virtual]

Finalize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.96.2.4 math::Angle gazebo::sensors::GpuRaySensor::GetAngleMax ( ) const

Get the maximum angle.

Returns

the maximum angle

10.96.2.5 math::Angle gazebo::sensors::GpuRaySensor::GetAngleMin ( ) const

Get the minimum angle.

Returns

The minimum angle

10.96.2.6 double gazebo::sensors::GpuRaySensor::GetAngleResolution ( ) const

Get radians between each range.

10.96.2.7 unsigned int gazebo::sensors::GpuRaySensor::GetCameraCount ( ) const

Gets the camera count.

Returns

Number of cameras

10.96.2.8 double gazebo::sensors::GpuRaySensor::GetCosHorzFOV ( ) const

Get Cos Horz field-of-view.

Returns

2 ∗ atan(tan(this->hfov/2) / cos(this->vfov/2))

10.96.2.9 double gazebo::sensors::GpuRaySensor::GetCosVertFOV ( ) const

Get Cos Vert field-of-view.

Returns

2 ∗ atan(tan(this->vfov/2) / cos(this->hfov/2))

10.96.2.10 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

10.96.2.11 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.

10.96.2.12 double gazebo::sensors::GpuRaySensor::GetHorzHalfAngle ( ) const

Get (horizontal_max_angle + horizontal_min_angle) $*$ 0.5.

Returns

(horizontal_max_angle + horizontal_min_angle) $*$ 0.5

10.96.2.13 rendering::GpuLaserPtr gazebo::sensors::GpuRaySensor::GetLaserCamera ( ) const [inline]

Returns a pointer to the internally kept **rendering::GpuLaser** (p. 579).

Returns

Pointer to GpuLaser

10.96.2.14 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.

10.96.2.15 int gazebo::sensors::GpuRaySensor::GetRangeCount ( ) const

Get the range count.

Returns

The number of ranges

10.96.2.16 double gazebo::sensors::GpuRaySensor::GetRangeCountRatio ( ) const

Return the ratio of horizontal range count to vertical range count.

A ray count is the number of simulated rays. Whereas a range count is the total number of data points returned. When range count != ray count, then values are interpolated between rays.

10.96.2.17 double gazebo::sensors::GpuRaySensor::GetRangeMax ( ) const

Get the maximum range.

Returns

The maximum range

10.96.2.18 double gazebo::sensors::GpuRaySensor::GetRangeMin ( ) const

Get the minimum range.

Returns

The minimum range

10.96.2.19 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

10.96.2.20 void gazebo::sensors::GpuRaySensor::GetRanges (std::vector< double > & _ranges)

Get all the ranges.

Parameters

out	_range	A vector that will contain all the range data

10.96.2.21 int gazebo::sensors::GpuRaySensor::GetRayCount ( ) const

Get the ray count.

Returns

The number of rays

10.96.2.22 double gazebo::sensors::GpuRaySensor::GetRayCountRatio ( ) const

Return the ratio of horizontal ray count to vertical ray count.

A ray count is the number of simulated rays. Whereas a range count is the total number of data points returned. When range count != ray count, then values are interpolated between rays.

10.96.2.23 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

10.96.2.24 virtual std::string gazebo::sensors::GpuRaySensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 1138).

10.96.2.25 double gazebo::sensors::GpuRaySensor::GetVertFOV ( ) const

Get the vertical field-of-view.

10.96.2.26 double gazebo::sensors::GpuRaySensor::GetVertHalfAngle ( ) const

Get (vertical_max_angle + vertical_min_angle) $*$ 0.5.

Returns

(vertical_max_angle + vertical_min_angle) $*$ 0.5

10.96.2.27 math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMax ( ) const

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.96.2.28 math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMin ( ) const

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.96.2.29 double gazebo::sensors::GpuRaySensor::GetVerticalAngleResolution ( ) const

Get the vertical angle in radians between each range.

Returns

Resolution of the angle

10.96.2.30 int gazebo::sensors::GpuRaySensor::GetVerticalRangeCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.96.2.31 int gazebo::sensors::GpuRaySensor::GetVerticalRayCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.96.2.32 virtual void gazebo::sensors::GpuRaySensor::Init ( ) [virtual]

Initialize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.96.2.33 virtual bool gazebo::sensors::GpuRaySensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.96.2.34 bool gazebo::sensors::GpuRaySensor::IsHorizontal ( ) const

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

10.96.2.35 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. 1130) parameters
in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 1140).

10.96.2.36 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. 1140).

10.96.2.37 void gazebo::sensors::GpuRaySensor::SetAngleMax (double _angle)

Set the scan maximum angle.

Parameters

in	*_angle*	The maximum angle

10.96.2.38 void gazebo::sensors::GpuRaySensor::SetAngleMin (double _angle)

Set the scan minimum angle.

Parameters

in	*_angle*	The minimum angle

10.96.2.39 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMax (double _angle)

Set the vertical scan line top angle.

Parameters

in	*_angle*	The Maximum angle of the scan block

10.96.2.40 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMin (double _angle)

Set the vertical scan bottom angle.

Parameters

in	*_angle*	The minimum angle of the scan block

10.96.2.41 virtual bool gazebo::sensors::GpuRaySensor::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

10.96.3 Member Data Documentation

10.96.3.1 sdf::ElementPtr gazebo::sensors::GpuRaySensor::cameraElem [protected]

Camera SDF element.

10.96.3.2 sdf::ElementPtr gazebo::sensors::GpuRaySensor::horzElem [protected]

Horizontal SDF element.

10.96.3.3 unsigned int gazebo::sensors::GpuRaySensor::horzRangeCount [protected]

Horizontal range count.

10.96.3.4 unsigned int gazebo::sensors::GpuRaySensor::horzRayCount [protected]

Horizontal ray count.

10.96.3.5 double gazebo::sensors::GpuRaySensor::rangeCountRatio [protected]

Range count ratio.

10.96.3.6 sdf::ElementPtr gazebo::sensors::GpuRaySensor::rangeElem [protected]

Range SDF element.

10.96.3.7 sdf::ElementPtr gazebo::sensors::GpuRaySensor::scanElem [protected]

Scan SDF elementz.

10.96.3.8 sdf::ElementPtr gazebo::sensors::GpuRaySensor::vertElem [protected]

Vertical SDF element.

10.96.3.9 unsigned int gazebo::sensors::GpuRaySensor::vertRangeCount [protected]

Vertical range count.

10.96.3.10 unsigned int gazebo::sensors::GpuRaySensor::vertRayCount [protected]

Vertical ray count.

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

- **GpuRaySensor.hh**

10.97 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. 163) 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.

10.97.1 Detailed Description

Displays a grid of cells, drawn with lines.

Displays a grid of cells, drawn with lines. A grid with an identity orientation is drawn along the XY plane.

10.97.2 Constructor & Destructor Documentation

10.97.2.1 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

10.97.2.2 gazebo::rendering::Grid::~Grid ( )

Destructor.

10.97.3 Member Function Documentation

10.97.3.1 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.

10.97.3.2 uint32_t gazebo::rendering::Grid::GetCellCount ( ) const [inline]

Get the number of cells.

10.97.3.3 float gazebo::rendering::Grid::GetCellLength ( ) const [inline]

Get the cell length.

Returns

The cell length

10.97.3.4 common::Color gazebo::rendering::Grid::GetColor ( ) const [inline]

Return the grid color.

Returns

The grid color

10.97.3.5 uint32_t gazebo::rendering::Grid::GetHeight ( ) const [inline]

Get the height of the grid.

Returns

The height

10.97.3.6 float gazebo::rendering::Grid::GetLineWidth ( ) const [inline]

Get the width of the grid line.

Returns

The line width

10.97.3.7 Ogre::SceneNode* gazebo::rendering::Grid::GetSceneNode ( ) [inline]

Get the **Ogre** (p. 163) scene node associated with this grid.

Returns

The **Ogre** (p. 163) scene node associated with this grid

10.97.3.8 void gazebo::rendering::Grid::Init ( )

Initialize the grid.

10.97.3.9 void gazebo::rendering::Grid::SetCellCount (uint32_t *_count*)

Set the number of cells.

Parameters

in	*_count*	The number of cells

10.97.3.10 void gazebo::rendering::Grid::SetCellLength (float *_len*)

Set the cell length.

Parameters

in	*_len*	The cell length

10.97.3.11 void gazebo::rendering::Grid::SetColor (const common::Color & *_color*)

Sets the color of the grid.

Parameters

in	*_color*	The grid color

10.97.3.12 void gazebo::rendering::Grid::SetHeight (uint32_t *_count*)

Set the height of the grid.

Parameters

in	*_count*	**Grid** (p. 605) height

10.97.3.13 void gazebo::rendering::Grid::SetLineWidth (float *_width*)

Set the line width.

Parameters

in	*_width*	The width of the grid

10.97.3.14 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**

10.98 gazebo::physics::Gripper Class Reference

A gripper abstraction.

`#include <physics/physics.hh>`

Public Member Functions

- **Gripper** (**ModelPtr** _model)

Constructor.

- virtual ~**Gripper** ()

  Destructor.

- std::string **GetName** () const

  Return the name of the gripper.

- virtual void **Init** ()

  Initialize.

- bool **IsAttached** () const

  True if the gripper is attached to another model.

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the gripper.

Protected Attributes

- **transport::NodePtr node**

  Node for communication.

10.98.1 Detailed Description

A gripper abstraction.

A gripper is a collection of links that act as a gripper. This class will intelligently generate fixed joints between the gripper and an object within the gripper. This allows the object to be manipulated without falling or behaving poorly.

10.98.2 Constructor & Destructor Documentation

10.98.2.1 gazebo::physics::Gripper::Gripper (ModelPtr *_model*) [explicit]

Constructor.

Parameters

in	*_model*	The model which contains the **Gripper** (p. 609).

10.98.2.2 virtual gazebo::physics::Gripper::~Gripper ( ) [virtual]

Destructor.

10.98.3 Member Function Documentation

10.98.3.1 std::string gazebo::physics::Gripper::GetName ( ) const

Return the name of the gripper.

10.98.3.2 virtual void gazebo::physics::Gripper::Init ( ) [virtual]

Initialize.

10.98.3.3 bool gazebo::physics::Gripper::IsAttached ( ) const

True if the gripper is attached to another model.

Returns

True if the gripper is active and a joint has been created between the gripper and another model.

10.98.3.4 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.

10.98.4 Member Data Documentation

10.98.4.1 transport::NodePtr gazebo::physics::Gripper::node [protected]

Node for communication.

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

- **Gripper.hh**

10.99 gazebo::rendering::GUIOverlay Class Reference

A class that creates a CEGUI overlay on a render window.

`#include <rendering/rendering.hh>`

Public Member Functions

- **GUIOverlay** ()

  Constructor.
- virtual ∼**GUIOverlay** ()

  Destructor.
- bool **AttachCameraToImage** (**CameraPtr** &_camera, const std::string &_windowName)

  *Use this function to draw the output from a **rendering::Camera*** (p. *242) to and overlay window.*
- bool **AttachCameraToImage** (**DepthCameraPtr** &_camera, const std::string &_windowName)

  *Use this function to draw the output from a **rendering::DepthCamera*** (p. *474) 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.

Public Attributes

- std::map< std::string, boost::function< void()> > **callbacks**

  Map of callback functions to names.

10.99.1 Detailed Description

A class that creates a CEGUI overlay on a render window.

10.99.2 Constructor & Destructor Documentation

10.99.2.1 gazebo::rendering::GUIOverlay::GUIOverlay ( )

Constructor.

10.99.2.2 virtual gazebo::rendering::GUIOverlay::~GUIOverlay ( ) [virtual]

Destructor.

10.99.3 Member Function Documentation

10.99.3.1 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (CameraPtr & *_camera*, const std::string & *_windowName*)

Use this function to draw the output from a **rendering::Camera** (p. 242) 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

10.99.3.2 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (DepthCameraPtr & _camera, const std::string & _windowName)

Use this function to draw the output from a **rendering::DepthCamera** (p. 474) 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

10.99.3.3 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.

References callbacks, and gzerr.

10.99.3.4 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.

10.99.3.5 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.

10.99.3.6 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.

10.99.3.7 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.

10.99.3.8 void gazebo::rendering::GUIOverlay::Hide ( )

Make the overlay invisible.

10.99.3.9 void gazebo::rendering::GUIOverlay::Init (Ogre::RenderTarget * *_renderTarget*)

Initialize the overlay.

Parameters

in	*_renderTarget*	The render target which will have the overlay.

10.99.3.10 bool gazebo::rendering::GUIOverlay::IsInitialized ( )

Return true if the overlay has been initialized.

Returns

True if initialized

10.99.3.11 void gazebo::rendering::GUIOverlay::LoadLayout (const std::string & *_filename*)

Load a CEGUI layout file.

Parameters

in	*_filename*	Name of the layout file.

10.99.3.12 void gazebo::rendering::GUIOverlay::Resize (unsigned int _width, unsigned int _height)

Resize the window.

10.99.3.13 void gazebo::rendering::GUIOverlay::Show ( )

Make the overlay visible.

10.99.3.14 void gazebo::rendering::GUIOverlay::Update ( )

Update the overlay's objects.

10.99.4 Member Data Documentation

10.99.4.1 std::map<std::string, boost::function<void()> > gazebo::rendering::GUIOverlay::callbacks

Map of callback functions to names.

Referenced by ButtonCallback().

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

- **GUIOverlay.hh**

10.100 gazebo::rendering::GUIOverlayPrivate Class Reference

Private data for the **GUIOverlay** (p. 612) class.

`#include <GUIOverlayPrivate.hh>`

Public Attributes

- std::vector< **event::ConnectionPtr** > **connections**

  All the connections.
- bool **initialized**

  True if initialized.
- std::string **layoutFilename**

  The layout file used to create gui elements.
- unsigned int **rttImageSetCount**

  Used in the AttachCameraToImage function to create unique names.

10.100.1 Detailed Description

Private data for the **GUIOverlay** (p. 612) class.

10.100.2 Member Data Documentation

10.100.2.1 std::vector<event::ConnectionPtr> gazebo::rendering::GUIOverlayPrivate::connections

All the connections.

10.100.2.2 bool gazebo::rendering::GUIOverlayPrivate::initialized

True if initialized.

10.100.2.3 std::string gazebo::rendering::GUIOverlayPrivate::layoutFilename

The layout file used to create gui elements.

10.100.2.4 unsigned int gazebo::rendering::GUIOverlayPrivate::rttImageSetCount

Used in the AttachCameraToImage function to create unique names.

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

- **GUIOverlayPrivate.hh**

10.101 gazebo::rendering::GzTerrainMatGen Class Reference

`#include <Heightmap.hh>`

Classes

- class **SM2Profile**

  Shader model 2 profile target.

Public Member Functions

- **GzTerrainMatGen** ()

Constructor.

- virtual ∼**GzTerrainMatGen** ()

  Destructor.

10.101.1 Constructor & Destructor Documentation

10.101.1.1 gazebo::rendering::GzTerrainMatGen::GzTerrainMatGen ( )

Constructor.

10.101.1.2 virtual gazebo::rendering::GzTerrainMatGen::∼GzTerrainMatGen ( ) [virtual]

Destructor.

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

- **Heightmap.hh**

10.102 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.
- bool **Flatten** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)

  Flatten the terrain based on a mouse press.
- double **GetAvgHeight** (Ogre::Vector3 _pos, double _brushSize)

  Get the average height around a point.
- double **GetHeight** (double _x, double _y, double _z=1000)

  Get the height at a location.
- **common::Image GetImage** () const

  Get the heightmap as an image.

- Ogre::TerrainGroup::RayResult **GetMouseHit** (**CameraPtr** _camera, **math::Vector2i** _mousePos)

  Calculate a mouse ray hit on the terrain.
- Ogre::TerrainGroup ∗ **GetOgreTerrain** () const

  Get a pointer to the OGRE terrain group object.
- unsigned int **GetTerrainSubdivisionCount** () const

  Get the number of subdivision the terrain will be split into.
- void **Load** ()

  Load the heightmap.
- void **LoadFromMsg** (ConstVisualPtr &_msg)

  Load the heightmap from a visual message.
- bool **Lower** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)

  Lower the terrain based on a mouse press.
- bool **Raise** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)

  Raise the terrain based on a mouse press.
- void **SetWireframe** (bool _show)

  Set the heightmap to render in wireframe mode.
- bool **Smooth** (**CameraPtr** _camera, **math::Vector2i** _mousePos, double _outsideRadius, double _insideRadius, double _weight=0.1)

  Smooth the terrain based on a mouse press.
- void **SplitHeights** (const std::vector< float > &_heightmap, int _n, std::vector< std::vector< float > > &_v)

  Split a terrain into subterrains.

10.102.1 Detailed Description

Rendering a terrain using heightmap information.

10.102.2 Constructor & Destructor Documentation

10.102.2.1 gazebo::rendering::Heightmap::Heightmap (ScenePtr *_scene*)

Constructor.

Parameters

in	*_scene*	Pointer to the scene that will contain the heightmap

10.102.2.2 virtual gazebo::rendering::Heightmap::~Heightmap ( ) [virtual]

Destructor.

10.102.3 Member Function Documentation

10.102.3.1 bool gazebo::rendering::Heightmap::Flatten (CameraPtr *_camera*, math::Vector2i *_mousePos*, double *_outsideRadius*, double *_insideRadius*, double *_weight* = 0.1)

Flatten the terrain based on a mouse press.

Parameters

in	*_camera*	**Camera** (p. 242) associated with the mouse press.
in	*_mousePos*	Position of the mouse in viewport coordinates.
in	*_outside-Radius*	Controls the radius of effect.
in	*_inside-Radius*	Controls the size of the radius with the maximum effect (value between 0 and 1).
in	*_weight*	Controls modification magnitude.

Returns

True if the terrain was modified

10.102.3.2 double gazebo::rendering::Heightmap::GetAvgHeight (Ogre::Vector3 *_pos*, double *_brushSize*)

Get the average height around a point.

Parameters

in	*_pos*	Position in world coordinates.
in	*_brushSize*	Controls the radius of effect.

10.102.3.3 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

10.102.3.4 common::Image gazebo::rendering::Heightmap::GetImage ( ) const

Get the heightmap as an image.

Returns

An image that contains the terrain data.

10.102.3.5 Ogre::TerrainGroup::RayResult gazebo::rendering::Heightmap::GetMouseHit (CameraPtr *_camera,* math::Vector2i *_mousePos*)

Calculate a mouse ray hit on the terrain.

Parameters

in	*_camera*	**Camera** (p. 242) associated with the mouse press.
in	*_mousePos*	Position of the mouse in viewport coordinates.

Returns

The result of the mouse ray hit.

10.102.3.6 Ogre::TerrainGroup* gazebo::rendering::Heightmap::GetOgreTerrain ( ) const

Get a pointer to the OGRE terrain group object.

Returns

Pointer to the OGRE terrain.

10.102.3.7 unsigned int gazebo::rendering::Heightmap::GetTerrainSubdivisionCount ( ) const

Get the number of subdivision the terrain will be split into.

Returns

Number of terrain subdivisions

10.102.3.8 void gazebo::rendering::Heightmap::Load ( )

Load the heightmap.

10.102.3.9 void gazebo::rendering::Heightmap::LoadFromMsg (ConstVisualPtr & *_msg*)

Load the heightmap from a visual message.

Parameters

in	*_msg*	The visual message containing heightmap info

10.102.3.10 bool gazebo::rendering::Heightmap::Lower (CameraPtr *_camera*, math::Vector2i *_mousePos*, double *_outsideRadius*, double *_insideRadius*, double *_weight* = 0.1)

Lower the terrain based on a mouse press.

Parameters

in	*_camera*	**Camera** (p. 242) associated with the mouse press.
in	*_mousePos*	Position of the mouse in viewport coordinates.
in	*_outside-Radius*	Controls the radius of effect.
in	*_inside-Radius*	Controls the size of the radius with the maximum effect (value between 0 and 1).
in	*_weight*	Controls modification magnitude.

Returns

True if the terrain was modified

10.102.3.11 bool gazebo::rendering::Heightmap::Raise (CameraPtr _camera, math::Vector2i _mousePos, double _outsideRadius, double _insideRadius, double _weight = 0.1)

Raise the terrain based on a mouse press.

Parameters

in	*_camera*	**Camera** (p. 242) associated with the mouse press.
in	*_mousePos*	Position of the mouse in viewport coordinates.
in	*_outside-Radius*	Controls the radius of effect.
in	*_inside-Radius*	Controls the size of the radius with the maximum effect (value between 0 and 1).
in	*_weight*	Controls modification magnitude.

Returns

True if the terrain was modified

10.102.3.12 void gazebo::rendering::Heightmap::SetWireframe (bool _show)

Set the heightmap to render in wireframe mode.

Parameters

in	*_show*	True to render wireframe, false to render solid.

10.102.3.13 bool gazebo::rendering::Heightmap::Smooth (CameraPtr _camera, math::Vector2i _mousePos, double _outsideRadius, double _insideRadius, double _weight = 0.1)

Smooth the terrain based on a mouse press.

Parameters

in	*_camera*	**Camera** (p. 242) associated with the mouse press.
in	*_mousePos*	Position of the mouse in viewport coordinates.
in	*_outside-Radius*	Controls the radius of effect.
in	*_inside-Radius*	Controls the size of the radius with the maximum effect (value between 0 and 1).
in	*_weight*	Controls modification magnitude.

Returns

True if the terrain was modified

10.102.3.14 void gazebo::rendering::Heightmap::SplitHeights (const std::vector< float > & *_heightmap*, int *_n*, std::vector< std::vector< float > > & *_v*)

Split a terrain into subterrains.

Parameters

in	*_heightmap*	Source vector of floats with the heights.
in	*_n*	Number of subterrains.
out	*_v*	Destination vector with the subterrains.

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

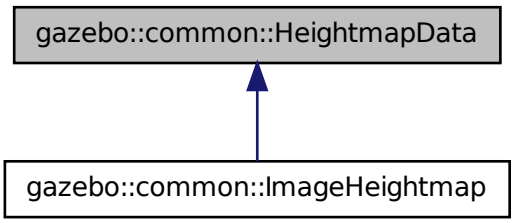
- **Heightmap.hh**

10.103 gazebo::common::HeightmapData Class Reference

Encapsulates a generic heightmap data file.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::HeightmapData:



Public Member Functions

- virtual ∼**HeightmapData** ()

  Destructor.
- virtual void **FillHeightMap** (int _subSampling, unsigned int _vertSize, const **math::Vector3** &_size, const **math::Vector3** &_scale, bool _flipY, std::vector< float > &_heights)=0

  Create a lookup table of the terrain's height.
- virtual unsigned int **GetHeight** () const =0

  Get the terrain's height.
- virtual float **GetMaxElevation** () const =0

  Get the maximum terrain's elevation.
- virtual unsigned int **GetWidth** () const =0

  Get the terrain's width.

10.103.1 Detailed Description

Encapsulates a generic heightmap data file.

10.103.2 Constructor & Destructor Documentation

10.103.2.1 virtual gazebo::common::HeightmapData::∼HeightmapData ( ) [inline, virtual]

Destructor.

10.103.3 Member Function Documentation

10.103.3.1 virtual void gazebo::common::HeightmapData::FillHeightMap (int *_subSampling,* unsigned int *_vertSize,* const math::Vector3 & *_size,* const math::Vector3 & *_scale,* bool *_flipY,* std::vector< float > & *_heights*) [pure virtual]

Create a lookup table of the terrain's height.

Parameters

in	*_subsampling*	Multiplier used to increase the resolution. Ex: A subsampling of 2 in a terrain of 129x129 means that the height vector will be 257 ∗ 257.
in	*_vertSize*	Number of points per row.
in	*_size*	Real dimmensions of the terrain.

in	*_scale*	Vector3 used to scale the height.
in	*_flipY*	If true, it inverts the order in which the vector is filled.
out	*_heights*	Vector containing the terrain heights.

Implemented in **gazebo::common::ImageHeightmap** (p. 648).

10.103.3.2 virtual unsigned int gazebo::common::HeightmapData::GetHeight ( ) const [pure virtual]

Get the terrain's height.

Returns

The terrain's height.

Implemented in **gazebo::common::ImageHeightmap** (p. 649).

10.103.3.3 virtual float gazebo::common::HeightmapData::GetMaxElevation ( ) const [pure virtual]

Get the maximum terrain's elevation.

Returns

The maximum terrain's elevation.

Implemented in **gazebo::common::ImageHeightmap** (p. 649).

10.103.3.4 virtual unsigned int gazebo::common::HeightmapData::GetWidth ( ) const [pure virtual]

Get the terrain's width.

Returns

The terrain's width.

Implemented in **gazebo::common::ImageHeightmap** (p. 649).

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

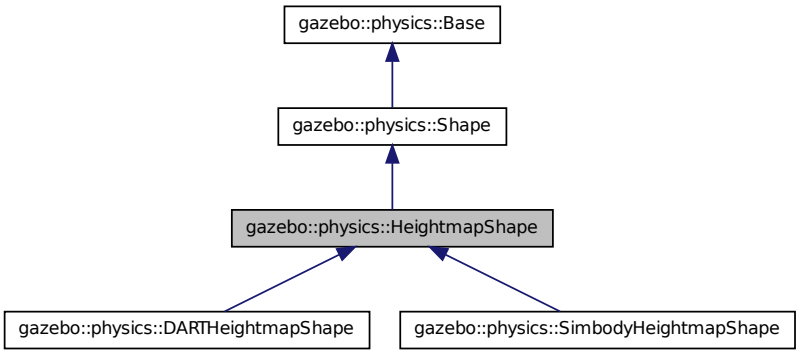
- **HeightmapData.hh**

10.104 gazebo::physics::HeightmapShape Class Reference

HeightmapShape (p. 628) 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) const

  Get a height at a position.
- **common::Image GetImage** () const

  Return an image representation of the heightmap.
- 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.
- virtual void **SetScale** (const **math::Vector3** &_scale)

  Set the scale of the heightmap shape.

Protected Attributes

- bool **flipY**

  True to flip the heights along the y direction.
- **common::HeightmapData** * **heightmapData**

  HeightmapData used to generate the heights.
- std::vector< float > **heights**

  Lookup table of heights.
- **common::ImageHeightmap img**

  Image used to generate the heights.
- int **subSampling**

  The amount of subsampling. Default is 2.
- unsigned int **vertSize**

  Size of the height lookup table.

10.104.1 Detailed Description

HeightmapShape (p. 628) 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.

10.104.2 Constructor & Destructor Documentation

10.104.2.1 gazebo::physics::HeightmapShape::HeightmapShape (CollisionPtr _parent) [explicit]

Constructor.

Parameters

in	*_parent*	Parent **Collision** (p. 295) object.

10.104.2.2 virtual gazebo::physics::HeightmapShape::~HeightmapShape ( ) [virtual]

Destructor.

10.104.3 Member Function Documentation

10.104.3.1 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. 1163).

10.104.3.2 float gazebo::physics::HeightmapShape::GetHeight (int _x, int _y) const

Get a height at a position.

Parameters

in	*_x*	X position.
in	*_y*	Y position.

Returns

The height at a the specified location.

10.104.3.3 common::Image gazebo::physics::HeightmapShape::GetImage ( ) const

Return an image representation of the heightmap.

Returns

Image where white pixels represents the highest locations, and black pixels the lowest.

10.104.3.4 float gazebo::physics::HeightmapShape::GetMaxHeight ( ) const

Get the maximum height.

Returns

The maximum height.

10.104.3.5 float gazebo::physics::HeightmapShape::GetMinHeight ( ) const

Get the minimum height.

Returns

The minimum height.

10.104.3.6 math::Vector3 gazebo::physics::HeightmapShape::GetPos ( ) const

Get the origin in world coordinate frame.

Returns

The origin in world coordinate frame.

10.104.3.7 math::Vector3 gazebo::physics::HeightmapShape::GetSize ( ) const

Get the size in meters.

Returns

The size in meters.

10.104.3.8 int gazebo::physics::HeightmapShape::GetSubSampling ( ) const

Get the amount of subsampling.

Returns

Amount of subsampling.

10.104.3.9 std::string gazebo::physics::HeightmapShape::GetURI ( ) const

Get the URI of the heightmap image.

Returns

The heightmap image URI.

10.104.3.10 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. 1430), result.x = width, result.y = length/height.

10.104.3.11 virtual void gazebo::physics::HeightmapShape::Init ( ) [virtual]

Initialize the heightmap.

Implements **gazebo::physics::Shape** (p. 1164).

Reimplemented in **gazebo::physics::SimbodyHeightmapShape** (p. 1181), and **gazebo::physics::DARTHeightmapShape** (p. 393).

10.104.3.12 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. 212).

10.104.3.13 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. 1164).

10.104.3.14 virtual void gazebo::physics::HeightmapShape::SetScale (const math::Vector3 & *_scale*) [virtual]

Set the scale of the heightmap shape.

Parameters

in	*_scale*	Scale to set the heightmap shape to.

Implements **gazebo::physics::Shape** (p. 1165).

10.104.4 Member Data Documentation

10.104.4.1 bool gazebo::physics::HeightmapShape::flipY [protected]

True to flip the heights along the y direction.

10.104.4.2 common::HeightmapData* gazebo::physics::HeightmapShape::heightmapData [protected]

HeightmapData used to generate the heights.

10.104.4.3 std::vector<float> gazebo::physics::HeightmapShape::heights [protected]

Lookup table of heights.

10.104.4.4 common::ImageHeightmap gazebo::physics::HeightmapShape::img [protected]

Image used to generate the heights.

10.104.4.5 int gazebo::physics::HeightmapShape::subSampling [protected]

The amount of subsampling. Default is 2.

10.104.4.6 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**

10.105 gazebo::physics::Hinge2Joint< T > Class Template - Reference

A two axis hinge joint.

`#include <physics/physics.hh>`

Public Member Functions

- **Hinge2Joint** (**BasePtr** _parent)
  Constructor.
- virtual ~**Hinge2Joint** ()
  Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)
  Load the joint.

10.105.1 Detailed Description

template<class T>class gazebo::physics::Hinge2Joint< T >

A two axis hinge joint.

10.105.2 Constructor & Destructor Documentation

10.105.2.1 template<class T> gazebo::physics::Hinge2Joint< T >::Hinge2Joint (BasePtr *_parent*) [inline, explicit]

Constructor.

Parameters

in	*_parent*	Parent link.

10.105.2.2 template<class T> virtual gazebo::physics::Hinge2Joint< T >::~Hinge2Joint ( ) [inline, virtual]

Destructor.

10.105.3 Member Function Documentation

10.105.3.1 template<class T> virtual unsigned int gazebo::physics::Hinge2Joint< T >::GetAngleCount ( ) const [inline, virtual]

10.105.3.2 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.

Reimplemented in **gazebo::physics::SimbodyHinge2Joint** (p. 1186), and **gazebo::physics::DARTHinge2Joint** (p. 398).

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

- **Hinge2Joint.hh**

10.106 gazebo::physics::HingeJoint< T > Class Template Reference

A single axis hinge joint.

`#include <physics/physics.hh>`

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.

10.106.1 Detailed Description

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

A single axis hinge joint.

10.106.2 Constructor & Destructor Documentation

10.106.2.1 template<class T> gazebo::physics::HingeJoint< T >::HingeJoint (BasePtr *_parent*) [inline]

Constructor.

Parameters

in	*_parent*	Parent link

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

Destructor.

10.106.3 Member Function Documentation

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

10.106.3.2 template<class T> virtual void gazebo::physics::HingeJoint< T >::Init ( ) [inline, protected, virtual]

Initialize joint.

Reimplemented in **gazebo::physics::DARTHingeJoint** (p. 403).

10.106.3.3 template<class T> virtual void gazebo::physics::HingeJoint< T >::Load (sdf::ElementPtr *_sdf*) [inline, virtual]

Load joint.

Parameters

in	*_sdf*	Pointer to SDF element

Reimplemented in **gazebo::physics::SimbodyHingeJoint** (p. 1191), and **gazebo::physics::DARTHingeJoint** (p. 404).

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

- **HingeJoint.hh**

10.107 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** () const

  Get the max color.
- int **GetPitch** () const
- **Color GetPixel** (unsigned int _x, unsigned int _y) const

  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. 639).*

10.107.1 Detailed Description

Encapsulates an image.

10.107.2 Member Enumeration Documentation

10.107.2.1 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

10.107.3 Constructor & Destructor Documentation

10.107.3.1 gazebo::common::Image::Image (const std::string & *_filename* = " ") [explicit]

Constructor.

Parameters

in	*_filename*	the path to the image

10.107.3.2 virtual gazebo::common::Image::~Image ( ) [virtual]

Destructor.

10.107.4 Member Function Documentation

10.107.4.1 static Image::PixelFormat gazebo::common::Image::ConvertPixelFormat (const std::string & *_format*) [static]

Convert a string to a **Image::PixelFormat** (p. 639).

Parameters

in	*_format*	Pixel format string.

See also

Image::PixelFormatNames

Returns

Image::PixelFormat (p. 639)

10.107.4.2 Color gazebo::common::Image::GetAvgColor ( )

Get the average color.

Returns

The average color

10.107.4.3 unsigned int gazebo::common::Image::GetBPP ( ) const

Get the size of one pixel in bits.

Returns

The BPP of the image

10.107.4.4 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

10.107.4.5 std::string gazebo::common::Image::GetFilename ( ) const

Get the full filename of the image.

Returns

The filename used to load the image

10.107.4.6 unsigned int gazebo::common::Image::GetHeight ( ) const

Get the height.

Returns

The image height

10.107.4.7 Color gazebo::common::Image::GetMaxColor ( ) const

Get the max color.

Returns

The max color

10.107.4.8 int gazebo::common::Image::GetPitch ( ) const

Returns

The pitch of the image

10.107.4.9 Color gazebo::common::Image::GetPixel (unsigned int _x, unsigned int _y) const

Get a pixel color value.

Parameters

in	_x	Column location in the image
in	_y	Row location in the image

Returns

The color of the given pixel

10.107.4.10 PixelFormat gazebo::common::Image::GetPixelFormat ( ) const

Get the pixel format.

Returns

PixelFormat

10.107.4.11 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

10.107.4.12 unsigned int gazebo::common::Image::GetWidth ( ) const

Get the width.

Returns

The image width

10.107.4.13 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

Returns

0 when the operation succeeds to open a file or -1 when fails.

10.107.4.14 void gazebo::common::Image::Rescale (int _width, int _height)

Rescale the image.

Parameters

in	*_width*	New image width
in	*_height*	New image height

10.107.4.15 void gazebo::common::Image::SavePNG (const std::string & _filename)

Save the image in PNG format.

Parameters

in	*_filename*	The name of the saved image

10.107.4.16 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

10.107.4.17 bool gazebo::common::Image::Valid ( ) const

Returns whether this is a valid image.

Returns

true if image has a bitmap

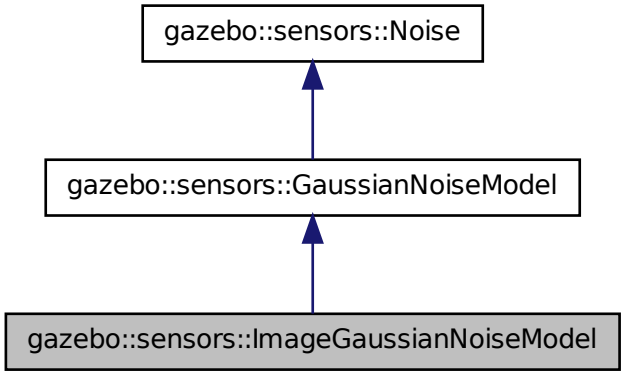
The documentation for this class was generated from the following file:

- **Image.hh**

10.108 gazebo::sensors::ImageGaussianNoiseModel Class Reference

`#include <GaussianNoiseModel.hh>`

Inheritance diagram for gazebo::sensors::ImageGaussianNoiseModel:



Public Member Functions

- **ImageGaussianNoiseModel** ()

  Constructor.

- virtual ∼**ImageGaussianNoiseModel** ()

  Destructor.
- virtual void **Fini** ()

  Finalize the noise model.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load noise parameters from sdf.
- virtual void **SetCamera** (**rendering::CameraPtr** _camera)

  Set camera needed to create image noise.

Public Attributes

- boost::shared_ptr < GaussianNoiseCompositorListener > **gaussianNoiseCompositorListener**

  Gaussian noise compositor listener.
- Ogre::CompositorInstance ∗ **gaussianNoiseInstance**

  Gaussian noise compositor.

10.108.1 Constructor & Destructor Documentation

10.108.1.1 gazebo::sensors::ImageGaussianNoiseModel::ImageGaussianNoiseModel ( )

Constructor.

10.108.1.2 virtual gazebo::sensors::ImageGaussianNoiseModel::∼ImageGaussianNoiseModel ( ) [virtual]

Destructor.

10.108.2 Member Function Documentation

10.108.2.1 virtual void gazebo::sensors::ImageGaussianNoiseModel::Fini ( ) [virtual]

Finalize the noise model.

Reimplemented from **gazebo::sensors::GaussianNoiseModel** (p. 569).

10.108.2.2 virtual void gazebo::sensors::ImageGaussianNoiseModel::Load (sdf::ElementPtr _sdf) [virtual]

Load noise parameters from sdf.

Parameters

in	*_sdf*	SDF parameters.
in	*_sensor*	Type of sensor.

Reimplemented from **gazebo::sensors::GaussianNoiseModel** (p. 570).

10.108.2.3 virtual void gazebo::sensors::ImageGaussianNoiseModel::SetCamera (rendering::CameraPtr _camera) [virtual]

Set camera needed to create image noise.

This is only needed for image sensors, i.e. camera/multicamera/depth sensors, which use shaders for more efficient noise generation.

Parameters

in	*_camera*	Camera associated to an image sensor

Reimplemented from **gazebo::sensors::Noise** (p. 934).

10.108.3 Member Data Documentation

10.108.3.1 boost::shared_ptr<GaussianNoiseCompositorListener> gazebo::sensors::ImageGaussianNoiseModel::gaussianNoiseCompositorListener

Gaussian noise compositor listener.

10.108.3.2 Ogre::CompositorInstance* gazebo::sensors::ImageGaussianNoiseModel::gaussianNoiseInstance

Gaussian noise compositor.

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

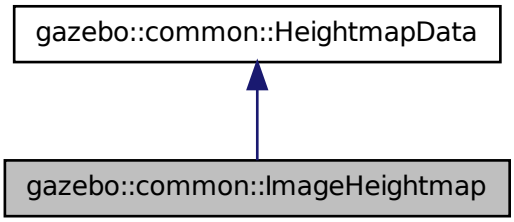
- **GaussianNoiseModel.hh**

10.109 gazebo::common::ImageHeightmap Class Reference

Encapsulates an image that will be interpreted as a heightmap.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::ImageHeightmap:



Public Member Functions

- **ImageHeightmap** ()

  Constructor.
- void **FillHeightMap** (int _subSampling, unsigned int _vertSize, const **math::-Vector3** &_size, const **math::Vector3** &_scale, bool _flipY, std::vector< float > &_heights)

  Create a lookup table of the terrain's height.
- std::string **GetFilename** () const

  Get the full filename of the image.
- unsigned int **GetHeight** () const

  Get the terrain's height.
- float **GetMaxElevation** () const

  Get the maximum terrain's elevation.
- unsigned int **GetWidth** () const

  Get the terrain's width.
- int **Load** (const std::string &_filename="")

  Load an image file as a heightmap.

10.109.1 Detailed Description

Encapsulates an image that will be interpreted as a heightmap.

10.109.2 Constructor & Destructor Documentation

10.109.2.1 gazebo::common::ImageHeightmap::ImageHeightmap ( )

Constructor.

Parameters

in	*_filename*	the path to the image

10.109.3 Member Function Documentation

10.109.3.1 void gazebo::common::ImageHeightmap::FillHeightMap (int *_subSampling*, unsigned int *_vertSize*, const math::Vector3 & *_size*, const math::Vector3 & *_scale*, bool *_flipY*, std::vector< float > & *_heights*) [virtual]

Create a lookup table of the terrain's height.

Parameters

in	*_subsampling*	Multiplier used to increase the resolution. Ex: A subsampling of 2 in a terrain of 129x129 means that the height vector will be $257 * 257$.
in	*_vertSize*	Number of points per row.
in	*_size*	Real dimmensions of the terrain.
in	*_scale*	Vector3 used to scale the height.
in	*_flipY*	If true, it inverts the order in which the vector is filled.
out	*_heights*	Vector containing the terrain heights.

Implements **gazebo::common::HeightmapData** (p. 626).

10.109.3.2 std::string gazebo::common::ImageHeightmap::GetFilename ( ) const

Get the full filename of the image.

Returns

The filename used to load the image

10.109.3.3 unsigned int gazebo::common::ImageHeightmap::GetHeight ( ) const [virtual]

Get the terrain's height.

Returns

The terrain's height.

Implements **gazebo::common::HeightmapData** (p. 627).

10.109.3.4 float gazebo::common::ImageHeightmap::GetMaxElevation ( ) const [virtual]

Get the maximum terrain's elevation.

Returns

The maximum terrain's elevation.

Implements **gazebo::common::HeightmapData** (p. 627).

10.109.3.5 unsigned int gazebo::common::ImageHeightmap::GetWidth ( ) const [virtual]

Get the terrain's width.

Returns

The terrain's width.

Implements **gazebo::common::HeightmapData** (p. 627).

10.109.3.6 int gazebo::common::ImageHeightmap::Load (const std::string & *_filename* = " ")

Load an image file as a heightmap.

Parameters

in	*_filename*	the path to the image file.

Returns

True when the operation succeeds to open a file.

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

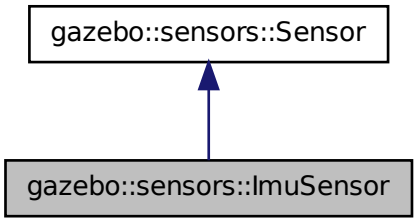
- **ImageHeightmap.hh**

10.110 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.
- virtual bool **IsActive** ()

  Returns true if sensor generation is active.
- 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 bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

10.110.1 Detailed Description

An IMU sensor.

10.110.2 Constructor & Destructor Documentation

10.110.2.1 gazebo::sensors::ImuSensor::ImuSensor ( )

Constructor.

10.110.2.2 virtual gazebo::sensors::ImuSensor::~ImuSensor ( ) [virtual]

Destructor.

10.110.3 Member Function Documentation

10.110.3.1 virtual void gazebo::sensors::ImuSensor::Fini ( ) [protected, virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.110.3.2 math::Vector3 gazebo::sensors::ImuSensor::GetAngularVelocity ( ) const

Returns the angular velocity.

Returns

Angular velocity.

10.110.3.3 msgs::IMU gazebo::sensors::ImuSensor::GetImuMessage ( ) const

Returns the imu message.

Returns

Imu message.

10.110.3.4 math::Vector3 gazebo::sensors::ImuSensor::GetLinearAcceleration ( ) const

Returns the imu linear acceleration.

Returns

Linear acceleration.

10.110.3.5 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.

10.110.3.6 virtual void gazebo::sensors::ImuSensor::Init ( ) [virtual]

Initialize the IMU.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.110.3.7 virtual bool gazebo::sensors::ImuSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.110.3.8 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. 1130) parameters.
in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 1140).

10.110.3.9 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. 1140).

10.110.3.10 void gazebo::sensors::ImuSensor::SetReferencePose ( )

Sets the current pose as the IMU reference pose.

10.110.3.11 virtual bool gazebo::sensors::ImuSensor::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

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

- **ImuSensor.hh**

10.111 gazebo::physics::Inertial Class Reference

A class for inertial information about a link.

`#include <physics/physics.hh>`

Public Member Functions

- **Inertial** ()

  Default Constructor.
- **Inertial** (double _mass)

  Constructor.
- **Inertial** (const **Inertial** &_inertial)

  Copy constructor.
- virtual ~**Inertial** ()

  Destructor.
- const **math::Vector3** & **GetCoG** () const

  Get the center of gravity.
- **Inertial GetInertial** (const **math::Pose** &_frameOffset) const

  *Get equivalent Inertia values with the **Link** (p. 739) 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. 739) frame If you specify GetMOI(this->**GetPose()** (p. 660)), 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. 739) 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. 739) 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.

10.111.1 Detailed Description

A class for inertial information about a link.

10.111.2 Constructor & Destructor Documentation

10.111.2.1 gazebo::physics::Inertial::Inertial ( )

Default Constructor.

10.111.2.2 gazebo::physics::Inertial::Inertial (double *_mass*) [explicit]

Constructor.

Parameters

in	*_mass*	Mass value in kg if using metric.

10.111.2.3 gazebo::physics::Inertial::Inertial (const Inertial & *_inertial*)

Copy constructor.

Parameters

in	*_inertial*	**Inertial** (p. 654) element to copy

10.111.2.4 virtual gazebo::physics::Inertial::~Inertial ( ) [virtual]

Destructor.

10.111.3 Member Function Documentation

10.111.3.1 const math::Vector3& gazebo::physics::Inertial::GetCoG ( ) const [inline]

Get the center of gravity.

Returns

The center of gravity.

10.111.3.2 Inertial gazebo::physics::Inertial::GetInertial (const math::Pose & _frameOffset) const

Get equivalent Inertia values with the **Link** (p. 739) frame offset, while holding the Pose of CoG constant in the world frame.

Parameters

in	*_frameOffset*	amount to offset the **Link** (p. 739) frame by, this is a transform defined in the **Link** (p. 739) frame.

Returns

Inertial (p. 654) parameters with the shifted frame.

10.111.3.3 double gazebo::physics::Inertial::GetIXX ( ) const

Get IXX.

Returns

IXX value

10.111.3.4 double gazebo::physics::Inertial::GetIXY ( ) const

Get IXY.

Returns

IXY value

10.111.3.5 double gazebo::physics::Inertial::GetIXZ ( ) const

Get IXZ.

Returns

IXZ value

10.111.3.6 double gazebo::physics::Inertial::GetIYY ( ) const

Get IYY.

Returns

IYY value

10.111.3.7 double gazebo::physics::Inertial::GetIYZ ( ) const

Get IXZ.

Returns

IYZ value

10.111.3.8 double gazebo::physics::Inertial::GetIZZ ( ) const

Get IZZ.

Returns

IZZ value

10.111.3.9 double gazebo::physics::Inertial::GetMass ( ) const

Get the mass.

10.111.3.10 math::Matrix3 gazebo::physics::Inertial::GetMOI (const math::Pose & *_pose*) const

Get the equivalent inertia from a point in local **Link** (p. 739) frame If you specify GetMOI(this->**GetPose()** (p. 660)), 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. 654) 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. 739) local frame

Returns

equivalent inertia at _pose

10.111.3.11 math::Matrix3 gazebo::physics::Inertial::GetMOI ( ) const

returns Moments of Inertia as a Matrix3

Returns

Moments of Inertia as a Matrix3

10.111.3.12 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. 739) frame.

Returns

The inertial pose.

10.111.3.13 math::Vector3 gazebo::physics::Inertial::GetPrincipalMoments ( ) const

Get the principal moments of inertia (Ixx, Iyy, Izz).

Returns

The principal moments.

10.111.3.14 math::Vector3 gazebo::physics::Inertial::GetProductsofInertia ( ) const

Get the products of inertia (Ixy, Ixz, Iyz).

Returns

The products of inertia.

10.111.3.15 void gazebo::physics::Inertial::Load (sdf::ElementPtr _sdf)

Load from SDF values.

Parameters

in	*_sdf*	SDF value to load from.

10.111.3.16 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. 739) frame. New CG is computed from masses and perspective offsets, and both MOI contributions relocated to the new cog.

Parameters

in	*_inertial*	**Inertial** (p. 654) to add.

Returns

The result of the addition.

10.111.3.17 const Inertial& gazebo::physics::Inertial::operator+= (const Inertial & _inertial)

Addition equal operator.

Parameters

in	*_inertial*	**Inertial** (p. 654) to add.

Returns

Reference to this object.

10.111.3.18 Inertial& gazebo::physics::Inertial::operator= (const Inertial & _inertial)

Equal operator.

Parameters

in	*_inertial*	**Inertial** (p. 654) to copy.

Returns

Reference to this object.

10.111.3.19 void gazebo::physics::Inertial::ProcessMsg (const msgs::Inertial & *_msg*)

Update parameters from a message.

Parameters

in	*_msg*	Message to read

10.111.3.20 void gazebo::physics::Inertial::Reset ( )

Reset all the mass properties.

10.111.3.21 void gazebo::physics::Inertial::Rotate (const math::Quaternion & *_rot*)

Rotate this mass.

Parameters

in	*_rot*	Rotation amount.

10.111.3.22 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.

10.111.3.23 void gazebo::physics::Inertial::SetCoG (const math::Vector3 & *_center*)

Set the center of gravity.

Parameters

in	*_center*	Center of the gravity.

10.111.3.24 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. 739) frame.

Parameters

in	*_cx*	Center offset in x-direction in **Link** (p. 739) frame
in	*_cy*	Center offset in y-direction in **Link** (p. 739) frame
in	*_cz*	Center offset in z-direction in **Link** (p. 739) 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.

10.111.3.25 void gazebo::physics::Inertial::SetCoG (const math::Pose & *_c*)

Set the center of gravity.

Parameters

in	*_c*	Transform to center of gravity.

10.111.3.26 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.

10.111.3.27 void gazebo::physics::Inertial::SetIXX (double _v)

Set IXX.

Parameters

in	_v	IXX value

10.111.3.28 void gazebo::physics::Inertial::SetIXY (double _v)

Set IXY.

Parameters

in	_v	IXY value

10.111.3.29 void gazebo::physics::Inertial::SetIXZ (double _v)

Set IXZ.

Parameters

in	_v	IXZ value

10.111.3.30 void gazebo::physics::Inertial::SetIYY (double _v)

Set IYY.

Parameters

in	_v	IYY value

10.111.3.31 void gazebo::physics::Inertial::SetIYZ (double _v)

Set IYZ.

Parameters

in	_v	IXX value

10.111.3.32 void gazebo::physics::Inertial::SetIZZ (double _v)

Set IZZ.

Parameters

in	*_v*	IZZ value

10.111.3.33 void gazebo::physics::Inertial::SetMass (double *m*)

Set the mass.

10.111.3.34 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

10.111.3.35 void gazebo::physics::Inertial::UpdateParameters (sdf::ElementPtr _sdf)

update the parameters using new sdf values.

Parameters

in	*_sdf*	Update values from.

10.111.4 Friends And Related Function Documentation

10.111.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::physics::Inertial & _inertial) [friend]

Output operator.

Parameters

in	*_out*	Output stream.
in	*_inertial*	**Inertial** (p. 654) object to output.

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

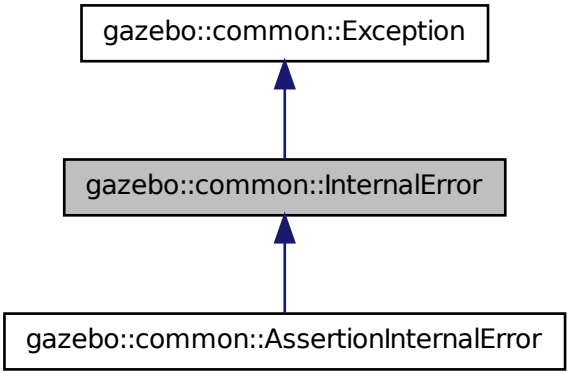
- **Inertial.hh**

10.112 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.

10.112.1 Detailed Description

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

10.112.2 Constructor & Destructor Documentation

10.112.2.1 gazebo::common::InternalError::InternalError ( )

Constructor.

10.112.2.2 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

10.112.2.3 virtual gazebo::common::InternalError::~InternalError ( ) [virtual]

Destructor.

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

- **Exception.hh**

10.113 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.

10.113.1 Detailed Description

Manages boost::asio IO.

10.113.2 Constructor & Destructor Documentation

10.113.2.1 gazebo::transport::IOManager::IOManager ( )

Constructor.

10.113.2.2 gazebo::transport::IOManager::∼IOManager ( )

Destructor.

10.113.3 Member Function Documentation

10.113.3.1 void gazebo::transport::IOManager::DecCount ( )

Decrement the event count by 1.

10.113.3.2 unsigned int gazebo::transport::IOManager::GetCount ( ) const

Get the event count.

Returns

The event count

10.113.3.3 boost::asio::io_service& gazebo::transport::IOManager::GetIO ( )

Get handle to boost::asio IO service.

Returns

Handle to boost::asio IO service

10.113.3.4 void gazebo::transport::IOManager::IncCount ( )

Increment the event count by 1.

10.113.3.5 void gazebo::transport::IOManager::Stop ( )

Stop the IO service.

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

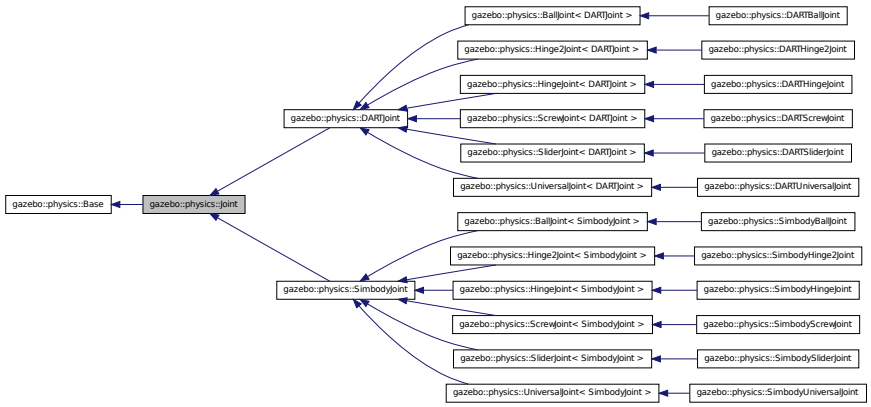
- **IOManager.hh**

10.114 gazebo::physics::Joint Class Reference

Base (p. 201) 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. *669) attribute types.*

Public Member Functions

- **Joint** (**BasePtr** _parent)

  Constructor.
- virtual ∼**Joint** ()

  Destructor.
- virtual void **ApplyStiffnessDamping** ()

  Callback to apply spring stiffness and viscous damping effects 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.
- virtual void **CacheForceTorque** ()

  Cache ***Joint*** *(*p. *669) Force Torque Values if necessary for physics engine.*
- double **CheckAndTruncateForce** (unsigned int _index, double _effort)

  check if the force against velocityLimit and effortLimit, truncate if necessary.
- 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 void **Fini** ()

  Finialize the object.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const =0

  Get the anchor point.
- **math::Pose GetAnchorErrorPose** () const

  Get pose offset between anchor pose on child and parent, expressed in the parent link frame.
- **math::Angle GetAngle** (unsigned int _index) const

Get the angle of rotation of an axis(index)

- virtual unsigned int **GetAngleCount** () const =0

  Get the angle count.

- **math::Quaternion GetAxisFrame** (unsigned int _index) const

  Get orientation of reference frame for specified axis, relative to world frame.

- **math::Quaternion GetAxisFrameOffset** (unsigned int _index) const

  Get orientation of joint axis reference frame relative to joint frame.

- **LinkPtr GetChild** () const

  Get the child link.

- double **GetDamping** (unsigned int _index)

  Returns the current joint damping coefficient.

- virtual double **GetEffortLimit** (unsigned int _index)

  Get the effort limit on axis(index).

- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench GetForceTorque** (unsigned int _index)=0

  get internal force and torque values at a joint.

- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const =0

  Get the axis of rotation in global cooridnate frame.

- virtual **math::Angle GetHighStop** (unsigned int _index)=0

  Get the high stop of an axis(index).

- double **GetInertiaRatio** (const unsigned int _index) const

  Computes moment of inertia (MOI) across a specified joint axis.

- double **GetInertiaRatio** (const **math::Vector3** &_axis) const

  Computes moment of inertia (MOI) across an arbitrary axis specified in the world frame.

- **math::Pose GetInitialAnchorPose** () const

  Get initial Anchor Pose specified by model <joint><pose>...</pose></joint>

- virtual **LinkPtr GetJointLink** (unsigned 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. 739) due to the existence of this **Joint** (p. 669).*

- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const =0

  *Get the torque applied to the center of mass of a **physics::Link** (p. 739) due to the existence of this **Joint** (p. 669).*

- **math::Vector3 GetLocalAxis** (unsigned 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** (unsigned int _index)=0

Get the low stop of an axis(index).

- virtual double **GetMaxForce** (unsigned int _index)=0

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*

- virtual double **GetParam** (const std::string &_key, unsigned int _index)=0

  Get a non-generic parameter for the joint.

- **LinkPtr GetParent** () const

  Get the parent link.

- **math::Pose GetParentWorldPose** () const

  Get anchor pose on parent link relative to world frame.

- double **GetSpringReferencePosition** (unsigned int _index) const

  Get joint spring reference position.

- double **GetStiffness** (unsigned int _index)

  Returns the current joint spring stiffness coefficient.

- double **GetStopDissipation** (unsigned int _index) const

  Get joint stop dissipation.

- double **GetStopStiffness** (unsigned int _index) const

  Get joint stop stiffness.

- **math::Angle GetUpperLimit** (unsigned int _index) const

  : get the joint lower limit (replacee GetLowStop and GetHighStop)

- virtual double **GetVelocity** (unsigned int _index) const =0

  Get the rotation rate of an axis(index)

- virtual double **GetVelocityLimit** (unsigned int _index)

  Get the velocity limit on axis(index).

- double **GetWorldEnergyPotentialSpring** (unsigned int _index) const

  Returns this joint's spring potential energy, based on the reference position of the spring.

- **math::Pose GetWorldPose** () const

  Get pose of joint frame relative to world frame.

- 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. 669).*

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load **physics::Joint** (p. 669) from a SDF sdf::Element.*

- virtual void **Reset** ()

  Reset the joint.

- virtual void **SetAnchor** (unsigned int _index, const **math::Vector3** &_anchor)=0

  Set the anchor point.

- void **SetAngle** (unsigned int _index, **math::Angle** _angle) **GAZEBO_DEPRECATED**(4.0)

  *If the **Joint** (p. 669) is static, Gazebo stores the state of this **Joint** (p. 669) as a scalar inside the **Joint** (p. 669) class, so this call will NOT move the joint dynamically for a static **Model** (p. 846).*
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)=0

  Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetDamping** (unsigned int _index, double _damping)=0

  Set the joint damping.
- virtual void **SetEffortLimit** (unsigned int _index, double _effort)

  Set the effort limit on a joint axis.
- virtual void **SetForce** (unsigned int _index, double _effort)=0

  *Set the force applied to this **physics::Joint** (p. 669).*
- virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- void **SetLowerLimit** (unsigned int _index, **math::Angle** _limit)

  : set the joint upper limit (replaces SetLowStop and SetHighStop)
- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual void **SetMaxForce** (unsigned int _index, double _force)=0

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- void **SetModel** (**ModelPtr** _model)

  Set the model this joint belongs too.
- virtual bool **SetParam** (const std::string &_key, unsigned int _index, const boost::any &_value)=0

  Set a non-generic parameter for the joint.
- virtual bool **SetPosition** (unsigned int _index, double _position)

  The child links of this joint are updated based on desired position.
- virtual void **SetProvideFeedback** (bool _enable)

  Set whether the joint should generate feedback.
- void **SetState** (const **JointState** &_state)

  Set the joint state.
- virtual void **SetStiffness** (unsigned int _index, double _stiffness)=0

  Set the joint spring stiffness.
- virtual void **SetStiffnessDamping** (unsigned int _index, double _stiffness, double _damping, double _reference=0)=0

  Set the joint spring stiffness.
- void **SetStopDissipation** (unsigned int _index, double _dissipation)

  Set joint stop dissipation.
- void **SetStopStiffness** (unsigned int _index, double _stiffness)

Set joint stop stiffness.

- void **SetUpperLimit** (unsigned int _index, **math::Angle** _limit)

  : set the joint lower limit (replacee GetLowStop and GetHighStop)

- virtual void **SetVelocity** (unsigned 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

- **math::Pose ComputeChildLinkPose** (unsigned int _index, double _position)

  internal function to help us compute child link pose if a joint position change is applied.

- bool **FindAllConnectedLinks** (const **LinkPtr** &_originalParentLink, **Link_V** &_connectedLinks)

  internal helper to find all links connected to the child link branching out from the children of the child link and any parent of the child link other than the parent link through this joint.

- virtual **math::Angle GetAngleImpl** (unsigned int _index) const =0

  Get the angle of an axis helper function.

- bool **SetPositionMaximal** (unsigned int _index, double _position)

  Helper function for maximal coordinate solver SetPosition.

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

- bool **axisParentModelFrame** [2]

  Flags that are set to true if an axis value is expressed in the parent model frame.

- **LinkPtr childLink**

  The first link this joint connects to.

- double **dissipationCoefficient** [2]

  joint viscous damping coefficient

- double **effortLimit** [2]

  *Store **Joint** (p. 669) effort limit as specified in SDF.*
- **math::Angle lowerLimit** [2]

  *Store **Joint** (p. 669) position lower limit as specified in SDF.*
- **ModelPtr model**

  Pointer to the parent model.
- **math::Pose parentAnchorPose**

  Anchor pose relative to parent link frame.
- **LinkPtr parentLink**

  The second link this joint connects to.
- bool **provideFeedback**

  Provide Feedback data for contact forces.
- double **springReferencePosition** [2]

  joint spring reference (zero load) position
- double **stiffnessCoefficient** [2]

  joint stiffnessCoefficient
- **math::Angle upperLimit** [2]

  *Store **Joint** (p. 669) position upper limit as specified in SDF.*
- double **velocityLimit** [2]

  *Store **Joint** (p. 669) velocity limit as specified in SDF.*
- **JointWrench wrench**

  *Cache **Joint** (p. 669) force torque values in case physics engine clears them at the end of update step.*

10.114.1 Detailed Description

Base (p. 201) class for all joints.

10.114.2 Member Enumeration Documentation

10.114.2.1 enum gazebo::physics::Joint::Attribute

Joint (p. 669) 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.

10.114.3 Constructor & Destructor Documentation

10.114.3.1 gazebo::physics::Joint::Joint (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	***Joint*** *(p. 669)*	parent

10.114.3.2 virtual gazebo::physics::Joint::~Joint ( ) [virtual]

Destructor.

10.114.4 Member Function Documentation

10.114.4.1 virtual void gazebo::physics::Joint::ApplyStiffnessDamping ( ) [virtual]

Callback to apply spring stiffness and viscous damping effects to joint.

: rename to ApplySpringStiffnessDamping()

10.114.4.2 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.

Implemented in **gazebo::physics::DARTJoint** (p. 408), and **gazebo::physics::SimbodyJoint** (p. 1196).

10.114.4.3 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.

Reimplemented in **gazebo::physics::DARTJoint** (p. 409).

10.114.4.4 virtual void gazebo::physics::Joint::CacheForceTorque ( ) [virtual]

Cache **Joint** (p. 669) Force Torque Values if necessary for physics engine.

Reimplemented in **gazebo::physics::SimbodyJoint** (p. 1197).

10.114.4.5 double gazebo::physics::Joint::CheckAndTruncateForce (unsigned int *_index,* double *_effort*)

check if the force against velocityLimit and effortLimit, truncate if necessary.

Parameters

in	*_index*	Index of the axis.
in	*_effort*	Force value.

Returns

truncated effort

10.114.4.6 math::Pose gazebo::physics::Joint::ComputeChildLinkPose (unsigned int *_index,* double *_position*) [protected]

internal function to help us compute child link pose if a joint position change is applied.

Parameters

in	*_index*	axis index
in	*_position*	new joint position

Returns

new child link pose at new joint position.

10.114.4.7 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.

10.114.4.8 virtual void gazebo::physics::Joint::Detach ( ) [virtual]

Detach this joint from all links.

Reimplemented in **gazebo::physics::DARTJoint** (p. 409), and **gazebo::physics::SimbodyJoint** (p. 1197).

10.114.4.9 void gazebo::physics::Joint::DisconnectJointUpdate (event::ConnectionPtr & *_conn*) [inline]

Disconnect a boost::slot the the joint update signal.

Parameters

in	*_conn*	Connection to disconnect.

10.114.4.10 void gazebo::physics::Joint::FillMsg (msgs::Joint & *_msg*)

Fill a joint message.

Parameters

out	*_msg*	Message to fill with this joint's properties.

10.114.4.11 bool gazebo::physics::Joint::FindAllConnectedLinks (const LinkPtr & *_originalParentLink*, Link_V & *_connectedLinks*) [protected]

internal helper to find all links connected to the child link branching out from the children of the child link and any parent of the child link other than the parent link through this joint.

Parameters

in	*_originalParentLink*	parent link of this joint, this is used to check for loops connecting back to the parent link.
	in/out]	_connectedLinks list of aggregated links that contains all links connected to the child link of this joint. Empty if a loop is found that connects back to the parent link.

Returns

true if successful, false if a loop is found that connects back to the parent link.

10.114.4.12 virtual void gazebo::physics::Joint::Fini ( ) [virtual]

Finialize the object.

Reimplemented from **gazebo::physics::Base** (p. 207).

10.114.4.13 virtual math::Vector3 gazebo::physics::Joint::GetAnchor (unsigned int *_index*) const [pure virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Implemented in **gazebo::physics::SimbodyJoint** (p. 1197), **gazebo::physics::SimbodyUniversalJoint** (p. 1262), **gazebo::physics::SimbodyHinge2Joint**

(p. 1184), **gazebo::physics::DARTHinge2Joint** (p. 396), **gazebo::physics::DARTHingeJoint** (p. 402), **gazebo::physics::SimbodyBallJoint** (p. 1168), **gazebo::physics::DARTBallJoint** (p. 377), **gazebo::physics::DARTSliderJoint** (p. 462), **gazebo::physics::DARTUniversalJoint** (p. 471), and **gazebo::physics::DARTScrewJoint** (p. 453).

10.114.4.14 math::Pose gazebo::physics::Joint::GetAnchorErrorPose ( ) const

Get pose offset between anchor pose on child and parent, expressed in the parent link frame.

This can be used to compute the bilateral constraint error.

Returns

Pose offset between anchor pose on child and parent, in parent link frame.

10.114.4.15 math::Angle gazebo::physics::Joint::GetAngle (unsigned int *_index*) const

Get the angle of rotation of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

10.114.4.16 virtual unsigned int gazebo::physics::Joint::GetAngleCount ( ) const [pure virtual]

Get the angle count.

Returns

The number of DOF for the joint.

Implemented in **gazebo::physics::DARTJoint** (p. 409), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1405), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1405), **gazebo::physics::BallJoint< DARTJoint >** (p. 200), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 200), **gazebo::physics::SliderJoint< DARTJoint >** (p. 1295), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 1295),

gazebo::physics::Hinge2Joint< DARTJoint > (p. 635), **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 635), **gazebo::physics::HingeJoint< DARTJoint >** (p. 637), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 637), **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1119), and **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 1119).

10.114.4.17 virtual math::Angle gazebo::physics::Joint::GetAngleImpl (unsigned 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.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1246), **gazebo::physics::DARTScrewJoint** (p. 453), **gazebo::physics::SimbodyUniversalJoint** (p. 1262), **gazebo::physics::SimbodyHinge2Joint** (p. 1184), **gazebo::physics::SimbodyHingeJoint** (p. 1190), **gazebo::physics::SimbodySliderJoint** (p. 1255), **gazebo::physics::SimbodyBallJoint** (p. 1168), **gazebo::physics::DARTBallJoint** (p. 377), **gazebo::physics::DARTHinge2Joint** (p. 396), **gazebo::physics::DARTHingeJoint** (p. 402), **gazebo::physics::DARTSliderJoint** (p. 462), and **gazebo::physics::DARTUniversalJoint** (p. 471).

10.114.4.18 math::Quaternion gazebo::physics::Joint::GetAxisFrame (unsigned int *_index*) const

Get orientation of reference frame for specified axis, relative to world frame.

The value of axisParentModelFrame is used to determine the appropriate frame.

Parameters

in	*_index*	joint axis index.

Returns

Orientation of axis frame relative to world frame.

10.114.4.19 math::Quaternion gazebo::physics::Joint::GetAxisFrameOffset (unsigned int _index) const

Get orientation of joint axis reference frame relative to joint frame.

This should always return identity unless flag use_parent_model_frame is true in sdf 1.5 or using sdf 1.4-, i.e. bug described in issue #494 is present. In addition, if use_parent-_model_frame is true, and the parent link of the joint is world, the axis is defined in the world frame. The value of axisParentModelFrame is used to determine the appropriate frame internally.

Parameters

in	*_index*	joint axis index.

Returns

Orientation of axis frame relative to joint frame. If supplied _index is out of range, or use_parent_model_frame is not true, this function returns identity rotation quaternion.

10.114.4.20 LinkPtr gazebo::physics::Joint::GetChild ( ) const

Get the child link.

Returns

Pointer to the child link.

10.114.4.21 double gazebo::physics::Joint::GetDamping (unsigned int _index)

Returns the current joint damping coefficient.

Parameters

in	*_index*	Index of the axis to get, currently ignored, to be implemented.

Returns

Joint (p. 669) viscous damping coefficient for this joint.

10.114.4.22 virtual double gazebo::physics::Joint::GetEffortLimit (unsigned 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

10.114.4.23 virtual double gazebo::physics::Joint::GetForce (unsigned int _index) [virtual]

Todo : not yet implemented. Get external forces applied at this **Joint** (p. 669). Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The force applied to an axis.

Reimplemented in **gazebo::physics::DARTJoint** (p. 410), and **gazebo::physics::SimbodyJoint** (p. 1197).

10.114.4.24 virtual JointWrench gazebo::physics::Joint::GetForceTorque (unsigned int _index) [pure virtual]

get internal force and torque values at a joint.

The force and torque values are returned in a **JointWrench** (p. 721) data structure. - Where **JointWrench.body1Force** (p. 723) contains the force applied by the parent **Link** (p. 739) on the **Joint** (p. 669) specified in the parent **Link** (p. 739) frame, and **JointWrench.body2Force** (p. 723) contains the force applied by the child **Link** (p. 739) on the **Joint** (p. 669) specified in the child **Link** (p. 739) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 669) on the Links.

FIXME TODO: change name of this function to something like: GetNegatedForceTorqueInLinkFrame and make GetForceTorque call return non-negated reaction forces in perspective **Link** (p. 739) frames.

Note that for ODE you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

Parameters

in	*_index*	Not used right now

Returns

The force and torque at the joint, see above for details on conventions.

Implemented in **gazebo::physics::SimbodyJoint** (p. 1198), and **gazebo::physics::DARTJoint** (p. 410).

10.114.4.25 virtual math::Vector3 gazebo::physics::Joint::GetGlobalAxis (unsigned 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.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1246), **gazebo::physics::SimbodyUniversalJoint** (p. 1263), **gazebo::physics::SimbodyHinge2Joint** (p. 1184), **gazebo::physics::SimbodyBallJoint** (p. 1169), **gazebo::physics::SimbodyHingeJoint** (p. 1190), **gazebo::physics::SimbodySliderJoint** (p. 1255), **gazebo::physics::DARTScrewJoint** (p. 454), **gazebo::physics::DARTHinge2Joint** (p. 396), **gazebo::physics::DARTHingeJoint** (p. 402), **gazebo::physics::DARTBallJoint** (p. 378), **gazebo::physics::DARTSliderJoint** (p. 462), and **gazebo::physics::DARTUniversalJoint** (p. 471).

10.114.4.26 virtual math::Angle gazebo::physics::Joint::GetHighStop (unsigned 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.

Implemented in **gazebo::physics::SimbodyJoint** (p. 1198), **gazebo::physics::DARTScrewJoint** (p. 454), **gazebo::physics::DARTJoint** (p. 411), **gazebo::physics::SimbodyBallJoint** (p. 1169), **gazebo::physics::DARTBallJoint** (p. 378), and **gazebo::physics::SimbodyScrewJoint** (p. 1247).

10.114.4.27 double gazebo::physics::Joint::GetInertiaRatio (const unsigned int *_index*) const

Computes moment of inertia (MOI) across a specified joint axis.

The ratio is given in the form of MOI_chidl / MOI_parent. If MOI_parent is zero, this funciton will return 0. The inertia ratio for each joint axis indicates the sensitivity of the joint to actuation torques.

Parameters

in	*_index*	axis number about which MOI ratio is computed.

Returns

ratio of child MOI to parent MOI.

10.114.4.28 double gazebo::physics::Joint::GetInertiaRatio (const math::Vector3 & *_axis*) const

Computes moment of inertia (MOI) across an arbitrary axis specified in the world frame.

The ratio is given in the form of MOI_chidl / MOI_parent. If MOI_parent is zero, this funciton will return 0. The moment of inertia ratio along constrained directions of a joint has an impact on the performance of Projected Gauss Seidel (PGS) iterative LCP methods.

Parameters

in	*_axis*	axis in world frame for which MOI ratio is computed.

Returns

ratio of child MOI to parent MOI.

10.114.4.29 math::Pose gazebo::physics::Joint::GetInitialAnchorPose ( ) const

Get initial Anchor Pose specified by model <joint><pose>...</pose></joint>

Returns

Joint::anchorPose (p. 702), initial joint anchor pose.

10.114.4.30 virtual LinkPtr gazebo::physics::Joint::GetJointLink (unsigned 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.

Implemented in **gazebo::physics::DARTJoint** (p. 411), and **gazebo::physics::SimbodyJoint** (p. 1199).

10.114.4.31 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. 739) due to the existence of this **Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*index*	The index of the link(0 or 1).

Returns

Force applied to the link.

Implemented in **gazebo::physics::DARTJoint** (p. 412), and **gazebo::physics::SimbodyJoint** (p. 1199).

10.114.4.32 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. 739) due to the existence of this **Joint** (p. 669).

Note that the unit of torque should be consistent with the rest of the simulation scales.

Parameters

in	*index*	The index of the link(0 or 1)

Returns

Torque applied to the link.

Implemented in **gazebo::physics::DARTJoint** (p. 412), and **gazebo::physics::SimbodyJoint** (p. 1200).

10.114.4.33 math::Vector3 gazebo::physics::Joint::GetLocalAxis (unsigned int *_index*) const

Get the axis of rotation.

Parameters

in	*_index*	Index of the axis to get.

Returns

Axis value for the provided index.

10.114.4.34 math::Angle gazebo::physics::Joint::GetLowerLimit (unsigned int *_index*) const

: get the joint upper limit (replaces GetLowStop and GetHighStop)

Parameters

in	*_index*	Index of the axis.

Returns

Lower limit of the axis.

10.114.4.35 virtual math::Angle gazebo::physics::Joint::GetLowStop (unsigned 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.

Implemented in **gazebo::physics::SimbodyJoint** (p. 1200), **gazebo::physics::DARTScrewJoint** (p. 454), **gazebo::physics::DARTJoint** (p. 412), **gazebo::physics::SimbodyBallJoint** (p. 1169), **gazebo::physics::DARTBallJoint** (p. 378), and **gazebo::physics::SimbodyScrewJoint** (p. 1247).

10.114.4.36 virtual double gazebo::physics::Joint::GetMaxForce (unsigned int *_index*) [pure virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1248), **gazebo::physics::DARTScrewJoint** (p. 455), **gazebo::physics::DARTHinge2Joint** (p. 397), **gazebo::physics::DARTHingeJoint** (p. 403), **gazebo::physics::SimbodyUniversalJoint** (p. 1263), **gazebo::physics::SimbodyHinge2Joint** (p. 1185), **gazebo::physics::DARTSliderJoint** (p. 463), **gazebo::physics::DARTUniversalJoint** (p. 472), **gazebo::physics::SimbodyHingeJoint** (p. 1190), **gazebo::physics::SimbodySliderJoint** (p. 1255), **gazebo::physics::SimbodyBallJoint** (p. 1170), and **gazebo::physics::DARTBallJoint** (p. 379).

10.114.4.37 virtual double gazebo::physics::Joint::GetParam (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.

Implemented in **gazebo::physics::DARTJoint** (p. 413), **gazebo::physics::SimbodyScrewJoint** (p. 1248), **gazebo::physics::SimbodyJoint** (p. 1200), and **gazebo::physics::DARTScrewJoint** (p. 455).

10.114.4.38 LinkPtr gazebo::physics::Joint::GetParent ( ) const

Get the parent link.

Returns

Pointer to the parent link.

Reimplemented from **gazebo::physics::Base** (p. 209).

10.114.4.39 math::Pose gazebo::physics::Joint::GetParentWorldPose ( ) const

Get anchor pose on parent link relative to world frame.

When there is zero joint error, this should match the value returned by **Joint::GetWorldPose()** (p. 692) for the constrained degrees of freedom.

Returns

Anchor pose on parent link in world frame.

10.114.4.40 double gazebo::physics::Joint::GetSpringReferencePosition (unsigned int _index) const

Get joint spring reference position.

Parameters

in	*_index*	Index of the axis to get.

Returns

Joint (p. 669) spring reference position (in radians for angular joints).

10.114.4.41 double gazebo::physics::Joint::GetStiffness (unsigned int _index)

Returns the current joint spring stiffness coefficient.

Parameters

in	*_index*	Index of the axis to get, currently ignored, to be implemented.

Returns

Joint (p. 669) spring stiffness coefficient for this joint. : rename to GetSpringStiffness()

10.114.4.42 double gazebo::physics::Joint::GetStopDissipation (unsigned int _index) const

Get joint stop dissipation.

Parameters

in	*_index*	joint axis index.

Returns

joint stop dissipation coefficient.

10.114.4.43 double gazebo::physics::Joint::GetStopStiffness (unsigned int _index) const

Get joint stop stiffness.

Parameters

in	*_index*	joint axis index.

Returns

joint stop stiffness coefficient.

10.114.4.44 math::Angle gazebo::physics::Joint::GetUpperLimit (unsigned int _index) const

: get the joint lower limit (replacee GetLowStop and GetHighStop)

Parameters

in	*_index*	Index of the axis.

Returns

Upper limit of the axis.

10.114.4.45 virtual double gazebo::physics::Joint::GetVelocity (unsigned 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.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1249), **gazebo::physics::DARTScrewJoint** (p. 456), **gazebo::physics::DARTHingeJoint** (p. 403), **gazebo::physics::SimbodyUniversalJoint** (p. 1263), **gazebo::physics::DARTHinge2Joint** (p. 397), **gazebo::physics::DARTSliderJoint** (p. 463), **gazebo::physics::SimbodyHinge2Joint** (p. 1185), **gazebo::physics::DARTUniversalJoint** (p. 472), **gazebo::physics::SimbodyBallJoint** (p. 1170), **gazebo::physics::SimbodyHingeJoint** (p. 1191), **gazebo::physics::SimbodySliderJoint** (p. 1256), and **gazebo::physics::DARTBallJoint** (p. 379).

10.114.4.46 virtual double gazebo::physics::Joint::GetVelocityLimit (unsigned 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

10.114.4.47 double gazebo::physics::Joint::GetWorldEnergyPotentialSpring (unsigned int *_index*) const

Returns this joint's spring potential energy, based on the reference position of the spring.

If using metric system, the unit of energy will be Joules.

Returns

this joint's spring potential energy,

10.114.4.48 math::Pose gazebo::physics::Joint::GetWorldPose ( ) const

Get pose of joint frame relative to world frame.

Note that the joint frame is defined with a fixed offset from the child link frame.

Returns

Pose of joint frame relative to world frame.

10.114.4.49 virtual void gazebo::physics::Joint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Base** (p. 211).

Reimplemented in **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1120), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 1120), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1406), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1406), **gazebo::physics::BallJoint< DARTJoint >** (p. 200), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 200), **gazebo::physics::HingeJoint< DARTJoint >** (p. 637), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 637), **gazebo::physics::DARTScrewJoint** (p. 456), **gazebo::physics::DARTJoint** (p. 413), **gazebo::physics::DARTHinge2Joint** (p. 397), **gazebo::physics::DARTHingeJoint** (p. 403), **gazebo::physics::DARTBallJoint** (p. 379), **gazebo::physics::DARTSliderJoint** (p. 463), and **gazebo::physics::DARTUniversalJoint** (p. 472).

10.114.4.50 void gazebo::physics::Joint::Load (LinkPtr *_parent,* LinkPtr *_child,* const math::Pose & *_pose*)

Set pose, parent and child links of a **physics::Joint** (p. 669).

Parameters

in	*_parent*	Parent link.
in	*_child*	Child link.
in	*_pose*	Pose containing **Joint** (p. 669) Anchor offset from child link.

10.114.4.51 virtual void gazebo::physics::Joint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load **physics::Joint** (p. 669) from a SDF sdf::Element.

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::Base** (p. 212).

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 1256), **gazebo::physics::BallJoint< DARTJoint >** (p. 200), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 200), **gazebo::physics::UniversalJoint< DARTJoint >** (p. 1406), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1406), **gazebo::physics::Hinge2Joint< DARTJoint >** (p. 635), **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 635), **gazebo::physics::HingeJoint< DARTJoint >** (p. 637),

gazebo::physics::HingeJoint< SimbodyJoint > (p. 637), **gazebo::physics::ScrewJoint< DARTJoint >** (p. 1120), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 1120), **gazebo::physics::SliderJoint< DARTJoint >** (p. 1295), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 1295), **gazebo::physics::SimbodyHingeJoint** (p. 1191), **gazebo::physics::SimbodyUniversalJoint** (p. 1264), **gazebo::physics::SimbodyHinge2Joint** (p. 1186), **gazebo::physics::SimbodyScrewJoint** (p. 1249), **gazebo::physics::SimbodyJoint** (p. 1201), **gazebo::physics::DARTJoint** (p. 414), **gazebo::physics::SimbodyBallJoint** (p. 1170), **gazebo::physics::DARTHinge2Joint** (p. 398), **gazebo::physics::DARTHingeJoint** (p. 404), **gazebo::physics::DARTBallJoint** (p. 380), **gazebo::physics::DARTScrewJoint** (p. 457), **gazebo::physics::DARTSliderJoint** (p. 464), and **gazebo::physics::DARTUniversalJoint** (p. 473).

10.114.4.52 virtual void gazebo::physics::Joint::Reset ( ) [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Base** (p. 214).

Reimplemented in **gazebo::physics::DARTJoint** (p. 414), and **gazebo::physics::SimbodyJoint** (p. 1201).

10.114.4.53 virtual void gazebo::physics::Joint::SetAnchor (unsigned int *_index*, const math::Vector3 & *_anchor*) [pure virtual]

Set the anchor point.

Parameters

in	*_index*	Indx of the axis.
in	*_anchor*	Anchor value.

Implemented in **gazebo::physics::DARTJoint** (p. 414), **gazebo::physics::SimbodyJoint** (p. 1202), and **gazebo::physics::DARTScrewJoint** (p. 457).

10.114.4.54 void gazebo::physics::Joint::SetAngle (unsigned int *_index*, math::Angle *_angle*)

If the **Joint** (p. 669) is static, Gazebo stores the state of this **Joint** (p. 669) as a scalar inside the **Joint** (p. 669) class, so this call will NOT move the joint dynamically for a static **Model** (p. 846).

But if this **Model** (p. 846) is not static, then it is updated dynamically. The child links of this joint are updated based on position change. And all the links connected to the child link of this joint except through the parent link of this joint moves with the child link.

Parameters

in	*_index*	Index of the axis.
in	*_angle*	Angle to set the joint to.

10.114.4.55 virtual void gazebo::physics::Joint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [pure virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Implemented in **gazebo::physics::SimbodyJoint** (p. 1202), **gazebo::physics::SimbodyBallJoint** (p. 1171), **gazebo::physics::DARTBallJoint** (p. 380), **gazebo::physics::DARTScrewJoint** (p. 457), **gazebo::physics::DARTHinge2Joint** (p. 398), **gazebo::physics::DARTHingeJoint** (p. 404), **gazebo::physics::SimbodyUniversalJoint** (p. 1264), **gazebo::physics::SimbodyHinge2Joint** (p. 1186), **gazebo::physics::DARTSliderJoint** (p. 464), **gazebo::physics::DARTUniversalJoint** (p. 473), **gazebo::physics::SimbodyHingeJoint** (p. 1192), **gazebo::physics::SimbodyScrewJoint** (p. 1249), and **gazebo::physics::SimbodySliderJoint** (p. 1256).

10.114.4.56 virtual void gazebo::physics::Joint::SetDamping (unsigned int *_index*, double *_damping*) [pure virtual]

Set the joint damping.

Parameters

in	*_index*	Index of the axis to set, currently ignored, to be implemented.
in	*_damping*	Damping value for the axis.

Implemented in **gazebo::physics::DARTJoint** (p. 414), and **gazebo::physics::SimbodyJoint** (p. 1202).

10.114.4.57 virtual void gazebo::physics::Joint::SetEffortLimit (unsigned int *_index*, double *_effort*) [virtual]

Set the effort limit on a joint axis.

Parameters

in	*_index*	Index of the axis to set.
in	*_effort*	Effort limit for the axis.

10.114.4.58 virtual void gazebo::physics::Joint::SetForce (unsigned int *_index*, double *_effort*) [pure virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForce to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)). Forces are truncated by effortLimit before applied.

Parameters

in	*_index*	Index of the axis.
in	*_effort*	Force value.

Implemented in **gazebo::physics::DARTJoint** (p. 415), and **gazebo::physics::SimbodyJoint** (p. 1203).

10.114.4.59 virtual bool gazebo::physics::Joint::SetHighStop (unsigned 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.

Reimplemented in **gazebo::physics::SimbodyJoint** (p. 1204), **gazebo::physics::SimbodyBallJoint** (p. 1171), **gazebo::physics::DARTJoint** (p. 415), **gazebo::physics::DARTBallJoint** (p. 381), and **gazebo::physics::SimbodyScrewJoint** (p. 1250).

10.114.4.60 void gazebo::physics::Joint::SetLowerLimit (unsigned int *_index*, math::Angle *_limit*)

: set the joint upper limit (replaces SetLowStop and SetHighStop)

Parameters

in	*_index*	Index of the axis.
in	*_limit*	Lower limit of the axis.

10.114.4.61 virtual bool gazebo::physics::Joint::SetLowStop (unsigned 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.

Reimplemented in **gazebo::physics::SimbodyJoint** (p. 1204), **gazebo::physics::SimbodyBallJoint** (p. 1172), **gazebo::physics::DARTJoint** (p. 416), **gazebo::physics::DARTBallJoint** (p. 381), and **gazebo::physics::SimbodyScrewJoint** (p. 1250).

10.114.4.62 virtual void gazebo::physics::Joint::SetMaxForce (unsigned int *_index*, double *_force*) [pure virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1250), **gazebo::physics::DARTScrewJoint** (p. 458), **gazebo::physics::DARTHinge2Joint** (p. 399), **gazebo::physics::DARTHingeJoint** (p. 405), **gazebo::physics::SimbodyUniversalJoint** (p. 1265), **gazebo::physics::SimbodyHinge2Joint** (p. 1186), **gazebo::physics::DARTSliderJoint** (p. 465), **gazebo::physics::DARTUniversalJoint** (p. 474), **gazebo::physics::SimbodyBallJoint** (p. 1172), **gazebo::physics::DARTBallJoint** (p. 381), **gazebo::physics::SimbodyHingeJoint** (p. 1192), and **gazebo::physics::SimbodySliderJoint** (p. 1257).

10.114.4.63 void gazebo::physics::Joint::SetModel (ModelPtr _model)

Set the model this joint belongs too.

Parameters

in	*_model*	Pointer to a model.

10.114.4.64 virtual bool gazebo::physics::Joint::SetParam (const std::string & _key, unsigned 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.

Implemented in **gazebo::physics::DARTJoint** (p. 416), **gazebo::physics::SimbodyScrewJoint** (p. 1251), and **gazebo::physics::SimbodyJoint** (p. 1204).

10.114.4.65 virtual bool gazebo::physics::Joint::SetPosition (unsigned int _index, double _position) [virtual]

The child links of this joint are updated based on desired position.

And all the links connected to the child link of this joint except through the parent link of this joint moves with the child link.

Parameters

in	*_index*	Index of the joint axis (degree of freedom).
in	*_position*	Position to set the joint to. unspecified, pure kinematic teleportation.

Returns

returns true if operation succeeds, 0 if it fails.

10.114.4.66 bool gazebo::physics::Joint::SetPositionMaximal (unsigned int *_index*, double *_position*) [protected]

Helper function for maximal coordinate solver SetPosition.

The child links of this joint are updated based on position change. And all the links connected to the child link of this joint except through the parent link of this joint moves with the child link.

Parameters

in	*_index*	Index of the joint axis (degree of freedom).
in	*_position*	Position to set the joint to. unspecified, pure kinematic teleportation.

Returns

returns true if operation succeeds, 0 if it fails.

10.114.4.67 virtual void gazebo::physics::Joint::SetProvideFeedback (bool *_enable*) [virtual]

Set whether the joint should generate feedback.

Parameters

in	*_enable*	True to enable joint feedback.

10.114.4.68 void gazebo::physics::Joint::SetState (const JointState & *_state*)

Set the joint state.

Parameters

in	*_state*	**Joint** (p. 669) state

10.114.4.69 virtual void gazebo::physics::Joint::SetStiffness (unsigned int *_index*, double *_stiffness*) [pure virtual]

Set the joint spring stiffness.

Parameters

in	*_index*	Index of the axis to set, currently ignored, to be implemented.
in	*_stiffness*	Spring stiffness value for the axis. : rename to SetSpringStiffness()

Implemented in **gazebo::physics::DARTJoint** (p. 416), and **gazebo::physics::SimbodyJoint** (p. 1205).

10.114.4.70 virtual void gazebo::physics::Joint::SetStiffnessDamping (unsigned int *_index*, double *_stiffness*, double *_damping*, double *_reference* = 0) [pure virtual]

Set the joint spring stiffness.

Parameters

in	*_index*	Index of the axis to set, currently ignored, to be implemented.
in	*_stiffness*	Stiffness value for the axis.
in	*_reference*	Spring zero load reference position. : rename to SetSpringStiffnessDamping()

Implemented in **gazebo::physics::DARTJoint** (p. 417), and **gazebo::physics::SimbodyJoint** (p. 1205).

10.114.4.71 void gazebo::physics::Joint::SetStopDissipation (unsigned int *_index*, double *_dissipation*)

Set joint stop dissipation.

Parameters

in	*_index*	joint axis index.
in	*_dissipation*	joint stop dissipation coefficient.

10.114.4.72 void gazebo::physics::Joint::SetStopStiffness (unsigned int *_index*, double *_stiffness*)

Set joint stop stiffness.

Parameters

in	*_index*	joint axis index.
in	*_stiffness*	joint stop stiffness coefficient.

10.114.4.73 void gazebo::physics::Joint::SetUpperLimit (unsigned int *_index*, math::Angle *_limit*)

: set the joint lower limit (replacee GetLowStop and GetHighStop)

Parameters

in	*_index*	Index of the axis.
in	*_limit*	Upper limit of the axis.

10.114.4.74 virtual void gazebo::physics::Joint::SetVelocity (unsigned int *_index*, double *_vel*) [pure virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1252), **gazebo::physics::DARTScrewJoint** (p. 459), **gazebo::physics::DARTHinge2Joint** (p. 399), **gazebo::physics::SimbodyHinge2Joint** (p. 1187), **gazebo::physics::DARTHingeJoint** (p. 405), **gazebo::physics::DARTUniversalJoint** (p. 474), **gazebo::physics::SimbodyUniversalJoint** (p. 1265), **gazebo::physics::DARTSliderJoint** (p. 465), **gazebo::physics::SimbodyBallJoint** (p. 1172), **gazebo::physics::SimbodyHingeJoint** (p. 1193), **gazebo::physics::SimbodySliderJoint** (p. 1257), and **gazebo::physics::DARTBallJoint** (p. 382).

10.114.4.75 void gazebo::physics::Joint::Update ( ) [virtual]

Update the joint.

Reimplemented from **gazebo::physics::Base** (p. 216).

10.114.4.76 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. 216).

10.114.5 Member Data Documentation

10.114.5.1 LinkPtr gazebo::physics::Joint::anchorLink [protected]

Anchor link.

10.114.5.2 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

10.114.5.3 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).

10.114.5.4 gazebo::event::ConnectionPtr gazebo::physics::Joint::applyDamping [protected]

apply damping for adding viscous damping forces on updates

10.114.5.5 bool gazebo::physics::Joint::axisParentModelFrame[2] [protected]

Flags that are set to true if an axis value is expressed in the parent model frame.

Otherwise use the joint frame. See issue #494.

10.114.5.6 LinkPtr gazebo::physics::Joint::childLink [protected]

The first link this joint connects to.

10.114.5.7 double gazebo::physics::Joint::dissipationCoefficient[2] [protected]

joint viscous damping coefficient

10.114.5.8 double gazebo::physics::Joint::effortLimit[2] [protected]

Store **Joint** (p. 669) effort limit as specified in SDF.

10.114.5.9 math::Angle gazebo::physics::Joint::lowerLimit[2] [protected]

Store **Joint** (p. 669) position lower limit as specified in SDF.

10.114.5.10 ModelPtr gazebo::physics::Joint::model [protected]

Pointer to the parent model.

10.114.5.11 math::Pose gazebo::physics::Joint::parentAnchorPose [protected]

Anchor pose relative to parent link frame.

10.114.5.12 LinkPtr gazebo::physics::Joint::parentLink [protected]

The second link this joint connects to.

10.114.5.13 bool gazebo::physics::Joint::provideFeedback [protected]

Provide Feedback data for contact forces.

10.114.5.14 double gazebo::physics::Joint::springReferencePosition[2] [protected]

joint spring reference (zero load) position

10.114.5.15 double gazebo::physics::Joint::stiffnessCoefficient[2] [protected]

joint stiffnessCoefficient

10.114.5.16 math::Angle gazebo::physics::Joint::upperLimit[2] [protected]

Store **Joint** (p. 669) position upper limit as specified in SDF.

10.114.5.17 double gazebo::physics::Joint::velocityLimit[2] [protected]

Store **Joint** (p. 669) velocity limit as specified in SDF.

10.114.5.18 JointWrench gazebo::physics::Joint::wrench [protected]

Cache **Joint** (p. 669) force torque values in case physics engine clears them at the end of update step.

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

- **Joint.hh**

10.115 gazebo::physics::JointController Class Reference

A class for manipulating **physics::Joint** (p. 669).

`#include <physics/physics.hh>`

Public Member Functions

- **JointController** (**ModelPtr** _model)
  Constructor.
- virtual ∼**JointController** ()
  Destructor.
- void **AddJoint** (**JointPtr** _joint)
  Add a joint to control.
- std::map< std::string, double > **GetForces** () const
  Get all the applied forces.
- std::map< std::string, **JointPtr** > **GetJoints** () const
  Get all the joints.
- **common::Time GetLastUpdateTime** () const

Get the last time the controller was updated.

- std::map< std::string, **common::PID** > **GetPositionPIDs** () const

  Get all the position PID controllers.

- std::map< std::string, double > **GetPositions** () const

  Get all the position PID set points.

- std::map< std::string, double > **GetVelocities** () const

  Get all the velocity PID set points.

- std::map< std::string, **common::PID** > **GetVelocityPIDs** () const

  Get all the velocity PID controllers.

- void **Reset** ()

  Reset all commands.

- void **SetJointPosition** (const std::string &_name, double _position, int _index=0)

  *Set the positions of a **Joint** (p. 669) by name.*

- void **SetJointPosition** (**JointPtr** _joint, double _position, int _index=0)

  *Set the positions of a **Joint** (p. 669) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 1294) and radians for **HingeJoint** (p. 635), etc.*

- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)

  *Set the positions of a set of **Joint** (p. 669)'s.*

- void **SetPositionPID** (const std::string &_jointName, const **common::PID** &_pid)

  Set the position PID values for a joint.

- bool **SetPositionTarget** (const std::string &_jointName, double _target)

  Set the target position for the position PID controller.

- void **SetVelocityPID** (const std::string &_jointName, const **common::PID** &_pid)

  Set the velocity PID values for a joint.

- bool **SetVelocityTarget** (const std::string &_jointName, double _target)

  Set the target velocity for the velocity PID controller.

- void **Update** ()

  Update the joint control.

10.115.1 Detailed Description

A class for manipulating **physics::Joint** (p. 669).

10.115.2 Constructor & Destructor Documentation

10.115.2.1 gazebo::physics::JointController::JointController (ModelPtr _model) [explicit]

Constructor.

Parameters

in	*_model*	**Model** (p. 846) that uses this joint controller.

10.115.2.2 virtual gazebo::physics::JointController::~JointController ( ) [virtual]

Destructor.

10.115.3 Member Function Documentation

10.115.3.1 void gazebo::physics::JointController::AddJoint (JointPtr _joint)

Add a joint to control.

Parameters

in	*_joint*	**Joint** (p. 669) to control.

10.115.3.2 std::map<std::string, double> gazebo::physics::JointController::GetForces ( ) const

Get all the applied forces.

Returns

A map<joint_name, force> that contains force values set by the user of the **JointController** (p. 704).

10.115.3.3 std::map<std::string, JointPtr> gazebo::physics::JointController::GetJoints ( ) const

Get all the joints.

Returns

A map<joint_name, joint_ptr> to all the joints that can be controlled.

10.115.3.4 common::Time gazebo::physics::JointController::GetLastUpdateTime ( ) const

Get the last time the controller was updated.

Returns

Last time the controller was updated.

10.115.3.5 std::map<std::string, common::PID> gazebo::physics::JointController::GetPositionPIDs ( ) const

Get all the position PID controllers.

Returns

A map<joint_name, PID> for all the position PID controllers.

10.115.3.6 std::map<std::string, double> gazebo::physics::JointController::GetPositions ( ) const

Get all the position PID set points.

Returns

A map<joint_name, position> that contains position values set by the user of the **JointController** (p. 704).

10.115.3.7 std::map<std::string, double> gazebo::physics::JointController::GetVelocities ( ) const

Get all the velocity PID set points.

Returns

A map<joint_name, position> that contains velocity values set by the user of the **JointController** (p. 704).

10.115.3.8 std::map<std::string, common::PID> gazebo::physics::JointController::GetVelocityPIDs ( ) const

Get all the velocity PID controllers.

Returns

A map<joint_name, PID> for all the velocity PID controllers.

10.115.3.9 void gazebo::physics::JointController::Reset ( )

Reset all commands.

10.115.3.10 void gazebo::physics::JointController::SetJointPosition (const std::string & *_name,* double *_position,* int *_index* = 0)

Set the positions of a **Joint** (p. 669) by name.

See also

JointController::SetJointPosition(JointPtr, double)

10.115.3.11 void gazebo::physics::JointController::SetJointPosition (JointPtr *_joint,* double *_position,* int *_index* = 0)

Set the positions of a **Joint** (p. 669) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 1294) and radians for **HingeJoint** (p. 635), etc.

Implementation: In order to change the position of a **Joint** (p. 669) inside a **Model** (p. 846), this call must recursively crawl through all the connected children **Link** (p. 739)'s in this **Model** (p. 846), and update each **Link** (p. 739) Pose affected by this **Joint** (p. 669) 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. Warning:

Parameters

in	*_joint*	**Joint** (p. 669) to set.
in	*_position*	Position of the joint.

10.115.3.12 void gazebo::physics::JointController::SetJointPositions (const std::map< std::string, double > & _jointPositions)

Set the positions of a set of **Joint** (p. 669)'s.

See also

JointController::SetJointPosition(JointPtr, double)

10.115.3.13 void gazebo::physics::JointController::SetPositionPID (const std::string & _jointName, const common::PID & _pid)

Set the position PID values for a joint.

Parameters

in	*_jointName*	Scoped name of the joint.
in	*_pid*	New position PID controller.

10.115.3.14 bool gazebo::physics::JointController::SetPositionTarget (const std::string & _jointName, double _target)

Set the target position for the position PID controller.

Parameters

in	*_jointName*	Scoped name of the joint.
in	*_target*	Position target.

Returns

False if the joint was not found.

10.115.3.15 void gazebo::physics::JointController::SetVelocityPID (const std::string & _jointName, const common::PID & _pid)

Set the velocity PID values for a joint.

Parameters

in	*_jointName*	Scoped name of the joint.
in	*_pid*	New velocity PID controller.

10.115.3.16 bool gazebo::physics::JointController::SetVelocityTarget (const std::string & _jointName, double _target)

Set the target velocity for the velocity PID controller.

Parameters

in	*_jointName*	Scoped name of the joint.
in	*_target*	Velocity target.

Returns

False if the joint was not found.

10.115.3.17 void gazebo::physics::JointController::Update ( )

Update the joint control.

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

- **JointController.hh**

10.116 gazebo::physics::JointControllerPrivate Class Reference

`#include <JointControllerPrivate.hh>`

Public Attributes

- std::map< std::string, double > **forces**

  Forces applied to joints.
- **transport::SubscriberPtr jointCmdSub**

  Subscribe to joint command.
- std::map< std::string, **JointPtr** > **joints**

  Map of joint names to the joint pointer.
- **ModelPtr model**

  Model *(p. 846) to control.*
- **transport::NodePtr node**

  Node for communication.
- std::map< std::string, double > **positions**

  Joint *(p. 669) positions.*

- std::map< std::string, **common::PID** > **posPids**

  Position PID controllers.
- **common::Time prevUpdateTime**

  Last time the controller was updated.
- **Link_V updatedLinks**

  List of links that have been updated.
- std::map< std::string, double > **velocities**

  Joint *(p. 669) velocities.*
- std::map< std::string, **common::PID** > **velPids**

  Velocity PID controllers.

10.116.1 Member Data Documentation

10.116.1.1 std::map<std::string, double> gazebo::physics::JointControllerPrivate::forces

Forces applied to joints.

10.116.1.2 transport::SubscriberPtr gazebo::physics::JointControllerPrivate::jointCmdSub

Subscribe to joint command.

10.116.1.3 std::map<std::string, JointPtr> gazebo::physics::JointControllerPrivate::joints

Map of joint names to the joint pointer.

10.116.1.4 ModelPtr gazebo::physics::JointControllerPrivate::model

Model (p. 846) to control.

10.116.1.5 transport::NodePtr gazebo::physics::JointControllerPrivate::node

Node for communication.

10.116.1.6 std::map<std::string, double> gazebo::physics::JointControllerPrivate::positions

Joint (p. 669) positions.

10.116.1.7 std::map<std::string, common::PID> gazebo::physics::JointControllerPrivate::posPids

Position PID controllers.

10.116.1.8 common::Time gazebo::physics::JointControllerPrivate::prevUpdateTime

Last time the controller was updated.

10.116.1.9 Link_V gazebo::physics::JointControllerPrivate::updatedLinks

List of links that have been updated.

10.116.1.10 std::map<std::string, double> gazebo::physics::JointControllerPrivate::velocities

Joint (p. 669) velocities.

10.116.1.11 std::map<std::string, common::PID> gazebo::physics::JointControllerPrivate::velPids

Velocity PID controllers.

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

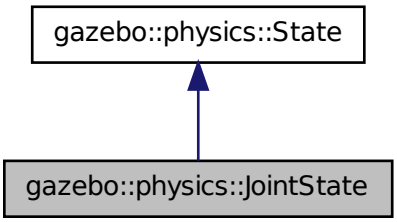
- **JointControllerPrivate.hh**

10.117 gazebo::physics::JointState Class Reference

keeps track of state of a **physics::Joint** (p. 669)

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::JointState:



Public Member Functions

- **JointState** ()

  Default constructor.
- **JointState** (**JointPtr** _joint, const **common::Time** &_realTime, const **common::Time** &_simTime)

  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.
- void **Load** (**JointPtr** _joint, const **common::Time** &_realTime, const **common::Time** &_simTime)

Load.

- 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.

10.117.1 Detailed Description

keeps track of state of a **physics::Joint** (p. 669)

10.117.2 Constructor & Destructor Documentation

10.117.2.1 gazebo::physics::JointState::JointState ( )

Default constructor.

10.117.2.2 gazebo::physics::JointState::JointState (JointPtr *_joint,* const common::Time & *_realTime,* const common::Time & *_simTime*)

Constructor.

Parameters

in	*_joint*	**Joint** (p. 669) to get the state of.
in	*_realTime*	Real time stamp.
in	*_simTime*	Sim time stamp.

10.117.2.3 gazebo::physics::JointState::JointState (JointPtr _joint) [explicit]

Constructor.

Parameters

in	*_joint*	**Joint** (p. 669) to get the state of.

10.117.2.4 gazebo::physics::JointState::JointState (const sdf::ElementPtr _sdf) [explicit]

Constructor.

Build a **JointState** (p. 712) from SDF data

Parameters

in	*_sdf*	SDF data to load a joint state from.

10.117.2.5 virtual gazebo::physics::JointState::~JointState ( ) [virtual]

Destructor.

10.117.3 Member Function Documentation

10.117.3.1 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.

10.117.3.2 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. 548)*	When _axis is invalid.

10.117.3.3 unsigned int gazebo::physics::JointState::GetAngleCount ( ) const

Get the number of angles.

Returns

The number of angles.

10.117.3.4 const std::vector<math::Angle>& gazebo::physics::JointState::GetAngles ( ) const

Get the angles.

Returns

Vector of angles.

10.117.3.5 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.

10.117.3.6 void gazebo::physics::JointState::Load (JointPtr _joint, const common::Time & _realTime, const common::Time & _simTime)

Load.

Parameters

in	*_joint*	**Joint** (p. 669) to get the state of.
in	*_realTime*	Real time stamp.
in	*_simTime*	Sim time stamp.

10.117.3.7 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. 1326).

10.117.3.8 JointState gazebo::physics::JointState::operator+ (const JointState & _state) const

Addition operator.

Parameters

in	*_pt*	A state to add.

Returns

The resulting state.

10.117.3.9 JointState gazebo::physics::JointState::operator- (const JointState & _state) const

Subtraction operator.

Parameters

in	*_pt*	A state to substract.

Returns

The resulting state.

10.117.3.10 JointState& gazebo::physics::JointState::operator= (const JointState & _state)

Assignment operator.

Parameters

in	*_state*	**State** (p. 1323) value

Returns

this

10.117.4 Friends And Related Function Documentation

10.117.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::physics::JointState & _state) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream.
in	*_state*	**Joint** (p. 669) state to output.

Returns

The stream.

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

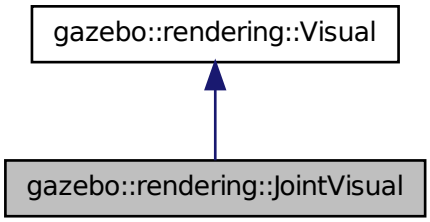
- **JointState.hh**

10.118 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.

10.118.1 Detailed Description

Visualization for joints.

10.118.2 Constructor & Destructor Documentation

10.118.2.1 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

10.118.2.2 virtual gazebo::rendering::JointVisual::~JointVisual ( ) [virtual]

Destructor.

10.118.3 Member Function Documentation

10.118.3.1 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:

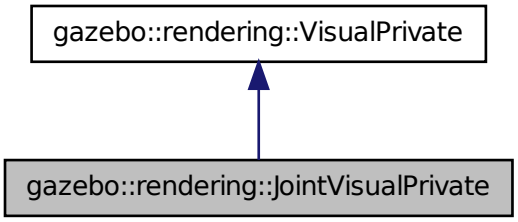
- **JointVisual.hh**

10.119 gazebo::rendering::JointVisualPrivate Class Reference

Private data for the Joint **Visual** (p. 1477) class.

`#include <JointVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::JointVisualPrivate:



Public Attributes

- **AxisVisualPtr axisVisual**

The visual used to draw the joint.

10.119.1 Detailed Description

Private data for the Joint **Visual** (p. 1477) class.

10.119.2 Member Data Documentation

10.119.2.1 AxisVisualPtr gazebo::rendering::JointVisualPrivate::axisVisual

The visual used to draw the joint.

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

- **JointVisualPrivate.hh**

10.120 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.

10.120.1 Detailed Description

Wrench information from a joint.

These are forces and torques on parent and child Links, relative to the **Joint** (p. 669) frame immediately after rotation.

10.120.2 Member Function Documentation

10.120.2.1 JointWrench& gazebo::physics::JointWrench::operator+ (const JointWrench & _wrench) [inline]

Operator +.

Parameters

in	*_wrench*	**Joint** (p. 669) wrench to add

Returns

∗this

References body1Force, body1Torque, body2Force, and body2Torque.

10.120.2.2 JointWrench& gazebo::physics::JointWrench::operator- (const JointWrench & _wrench) [inline]

Operator -.

Parameters

in	*_wrench*	**Joint** (p. 669) wrench to subtract

Returns

∗this

References body1Force, body1Torque, body2Force, and body2Torque.

10.120.2.3 JointWrench& gazebo::physics::JointWrench::operator= (const JointWrench & _wrench) [inline]

Operator =.

Parameters

in	*_wrench*	**Joint** (p. 669) wrench to set from.

Returns

*this

References body1Force, body1Torque, body2Force, and body2Torque.

10.120.3 Member Data Documentation

10.120.3.1 math::Vector3 gazebo::physics::JointWrench::body1Force

Force on the first link.

Referenced by operator+(), operator-(), and operator=().

10.120.3.2 math::Vector3 gazebo::physics::JointWrench::body1Torque

Torque on the first link.

Referenced by operator+(), operator-(), and operator=().

10.120.3.3 math::Vector3 gazebo::physics::JointWrench::body2Force

Force on the second link.

Referenced by operator+(), operator-(), and operator=().

10.120.3.4 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**

10.121 gazebo::common::KeyEvent Class Reference

Generic description of a keyboard event.

`#include <common/common.hh>`

Public Types

- enum **EventType** { **NO_EVENT**, **PRESS**, **RELEASE** }

  Key event types enumeration.

Public Member Functions

- **KeyEvent** ()

  Constructor.

Public Attributes

- int **key**
- **EventType type**

  Event type.

10.121.1 Detailed Description

Generic description of a keyboard event.

10.121.2 Member Enumeration Documentation

10.121.2.1 enum gazebo::common::KeyEvent::EventType

Key event types enumeration.

Enumerator:

NO_EVENT

PRESS

RELEASE

10.121.3 Constructor & Destructor Documentation

10.121.3.1 gazebo::common::KeyEvent::KeyEvent ( ) [inline]

Constructor.

10.121.4 Member Data Documentation

10.121.4.1 int gazebo::common::KeyEvent::key

10.121.4.2 EventType gazebo::common::KeyEvent::type

Event type.

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

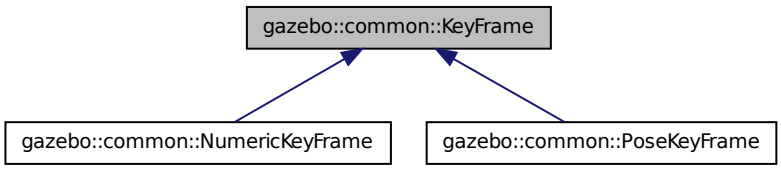
- **KeyEvent.hh**

10.122 gazebo::common::KeyFrame Class Reference

A key frame in an animation.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::KeyFrame:



Public Member Functions

- **KeyFrame** (double _time)

  Constructor.
- virtual ~**KeyFrame** ()

  Destructor.
- double **GetTime** () const

  Get the time of the keyframe.

Protected Attributes

- double **time**

  time of key frame

10.122.1 Detailed Description

A key frame in an animation.

10.122.2 Constructor & Destructor Documentation

10.122.2.1 gazebo::common::KeyFrame::KeyFrame (double *_time*)

Constructor.

Parameters

in	*_time*	**Time** (p. 1361) of the keyframe in seconds

10.122.2.2 virtual gazebo::common::KeyFrame::~KeyFrame ( ) [virtual]

Destructor.

10.122.3 Member Function Documentation

10.122.3.1 double gazebo::common::KeyFrame::GetTime ( ) const

Get the time of the keyframe.

Returns

the time

10.122.4 Member Data Documentation

10.122.4.1 double gazebo::common::KeyFrame::time [protected]

time of key frame

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

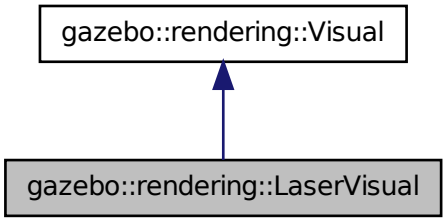
- **KeyFrame.hh**

10.123 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.

10.123.1 Detailed Description

Visualization for laser data.

10.123.2 Constructor & Destructor Documentation

10.123.2.1 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. 1477).
in	*_topicName*	Name of the topic that has laser data.

10.123.2.2 virtual gazebo::rendering::LaserVisual::~LaserVisual ( ) [virtual]

Destructor.

10.123.3 Member Function Documentation

10.123.3.1 virtual void gazebo::rendering::LaserVisual::SetEmissive (const common::Color & *_color*) [virtual]

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 1497).

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

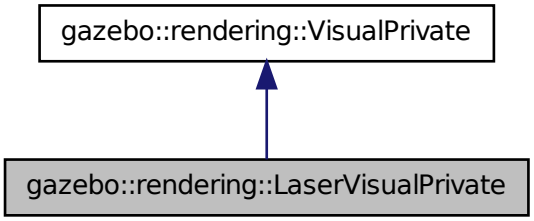
- **LaserVisual.hh**

10.124 gazebo::rendering::LaserVisualPrivate Class Reference

Private data for the Laser **Visual** (p. 1477) class.

`#include <LaserVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::LaserVisualPrivate:



Public Attributes

- **event::ConnectionPtr connection**

  Pre render connection.
- boost::shared_ptr < msgs::LaserScanStamped const > **laserMsg**

  The current contact message.
- **transport::SubscriberPtr laserScanSub**

  Subscription to the laser data.
- boost::mutex **mutex**

  Mutex to protect the contact message.
- **transport::NodePtr node**

  Pointer to a node that handles communication.
- std::vector< **DynamicLines** * > **rayFans**

  Renders the laser data.
- bool **receivedMsg**

  True if we have received a message.

10.124.1 Detailed Description

Private data for the Laser **Visual** (p. 1477) class.

10.124.2 Member Data Documentation

10.124.2.1 event::ConnectionPtr gazebo::rendering::LaserVisualPrivate::connection

Pre render connection.

10.124.2.2 boost::shared_ptr<msgs::LaserScanStamped const> gazebo::rendering::LaserVisualPrivate::laserMsg

The current contact message.

10.124.2.3 transport::SubscriberPtr gazebo::rendering::LaserVisualPrivate::laserScanSub

Subscription to the laser data.

10.124.2.4 boost::mutex gazebo::rendering::LaserVisualPrivate::mutex

Mutex to protect the contact message.

10.124.2.5 transport::NodePtr gazebo::rendering::LaserVisualPrivate::node

Pointer to a node that handles communication.

10.124.2.6 std::vector<DynamicLines *> gazebo::rendering::LaserVisualPrivate::rayFans

Renders the laser data.

10.124.2.7 bool gazebo::rendering::LaserVisualPrivate::receivedMsg

True if we have received a message.

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

- **LaserVisualPrivate.hh**

10.125 gazebo::rendering::Light Class Reference

A light source.

`#include <rendering/rendering.hh>`

Public Member Functions

- **Light** (**ScenePtr** _scene)
  Constructor.
- virtual ~**Light** ()
  Destructor.
- **LightPtr Clone** (const std::string &_name, **ScenePtr** _scene)
  Clone the light with a new name.
- 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.

- **math::Quaternion GetRotation** () const

  Get the rotation of the light.

- **common::Color GetSpecularColor** () const

  Get the specular color.

- std::string **GetType** () const

  Get the type of the light.

- bool **GetVisible** () const

  Get whether the light is visible.

- 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 **LoadFromMsg** (const msgs::Light &_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.

- void **SetRotation** (const **math::Quaternion** &_q)

  Set the rotation of the light.

- 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.

10.125.1 Detailed Description

A light source.

There are three types of lights: Point, Spot, and Directional. This class encapsulates all three. Point lights are light light bulbs, spot lights project a cone of light, and directional lights are light sun light.

10.125.2 Constructor & Destructor Documentation

10.125.2.1 gazebo::rendering::Light::Light (ScenePtr *_scene*)

Constructor.

Parameters

in	*_scene*	Pointer to the scene that contains the **Light** (p. 730).

10.125.2.2 virtual gazebo::rendering::Light::~Light ( ) [virtual]

Destructor.

10.125.3 Member Function Documentation

10.125.3.1 LightPtr gazebo::rendering::Light::Clone (const std::string & _name, ScenePtr _scene)

Clone the light with a new name.

Parameters

in	*_name*	Name of the cloned light.
in	*_scene*	**Scene** (p. 1097) to contain the light.

Returns

a clone of the light

10.125.3.2 void gazebo::rendering::Light::FillMsg (msgs::Light & _msg) const

Fill the contents of a light message.

Parameters

out	*_msg*	Message to fill.

10.125.3.3 common::Color gazebo::rendering::Light::GetDiffuseColor ( ) const

Get the diffuse color.

Returns

The light's diffuse color.

10.125.3.4 math::Vector3 gazebo::rendering::Light::GetDirection ( ) const

Get the direction.

Returns

The light's direction.

10.125.3.5 std::string gazebo::rendering::Light::GetName ( ) const

Get the name of the visual.

Returns

The light's name.

10.125.3.6 math::Vector3 gazebo::rendering::Light::GetPosition ( ) const

Get the position of the light.

Returns

The position of the light

10.125.3.7 math::Quaternion gazebo::rendering::Light::GetRotation ( ) const

Get the rotation of the light.

Returns

The rotation of the light

10.125.3.8 common::Color gazebo::rendering::Light::GetSpecularColor ( ) const

Get the specular color.

Returns

The specular color

10.125.3.9 std::string gazebo::rendering::Light::GetType ( ) const

Get the type of the light.

Returns

The light type: "point", "spot", "directional".

10.125.3.10 bool gazebo::rendering::Light::GetVisible ( ) const

Get whether the light is visible.

Returns

True if the light is visible.

10.125.3.11 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. 730) description.

10.125.3.12 void gazebo::rendering::Light::Load ( )

Load the light using default parameters.

10.125.3.13 void gazebo::rendering::Light::LoadFromMsg (ConstLightPtr & _msg)

Load from a light message.

Parameters

in	*_msg*	Containing the light information.

10.125.3.14 void gazebo::rendering::Light::LoadFromMsg (const msgs::Light & _msg)

Load from a light message.

Parameters

in	*_msg*	Message containing the light information.

10.125.3.15 virtual void gazebo::rendering::Light::OnPoseChange ( ) [inline, protected, virtual]

On pose change callback.

10.125.3.16 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

10.125.3.17 void gazebo::rendering::Light::SetCastShadows (const bool & *_cast*)

Set cast shadows.

Parameters

in	*_cast*	Set to true to cast shadows.

10.125.3.18 void gazebo::rendering::Light::SetDiffuseColor (const common::Color & *_color*)

Set the diffuse color.

Parameters

in	*_color*	**Light** (p. 730) diffuse color.

10.125.3.19 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.

10.125.3.20 void gazebo::rendering::Light::SetLightType (const std::string & *_type*)

Set the light type.

Parameters

in	*_type*	The light type: "point", "spot", "directional"

10.125.3.21 void gazebo::rendering::Light::SetName (const std::string & *_name*)

Set the name of the visual.

Parameters

in	*_name*	Name of the light source.

10.125.3.22 void gazebo::rendering::Light::SetPosition (const math::Vector3 & *_p*)

Set the position of the light.

Parameters

in	*_p*	New position for the light

10.125.3.23 void gazebo::rendering::Light::SetRange (const double & *_range*)

Set the range.

Parameters

in	*_range*	Rage of the light in meters.

10.125.3.24 void gazebo::rendering::Light::SetRotation (const math::Quaternion & *_q*)

Set the rotation of the light.

Parameters

in	*_q*	New rotation for the light

10.125.3.25 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.

10.125.3.26 void gazebo::rendering::Light::SetSpecularColor (const common::Color & _color)

Set the specular color.

Parameters

in	*_color*	The specular color

10.125.3.27 void gazebo::rendering::Light::SetSpotFalloff (const double & _value)

Set the spot light falloff.

Parameters

in	*_value*	Falloff value

10.125.3.28 void gazebo::rendering::Light::SetSpotInnerAngle (const double & _angle)

Set the spot light inner angle.

Parameters

in	*_angle*	Inner angle in radians

10.125.3.29 void gazebo::rendering::Light::SetSpotOuterAngle (const double & _angle)

Set the spot light outer angle.

Parameters

in	*_angle*	Outer angle in radians

10.125.3.30 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.

10.125.3.31 void gazebo::rendering::Light::ToggleShowVisual ( )

10.125.3.32 void gazebo::rendering::Light::UpdateFromMsg (ConstLightPtr & *_msg*)

Update a light source from a message.

Parameters

in	*_msg*	**Light** (p. 730) message to update from

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

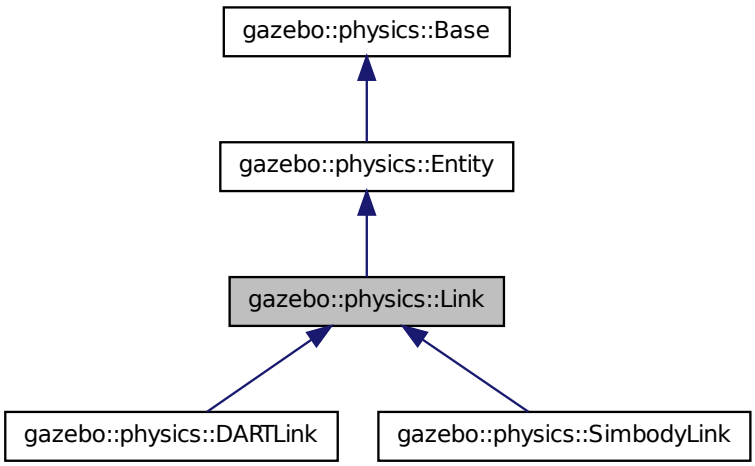
- **Light.hh**

10.126 gazebo::physics::Link Class Reference

Link (p. 739) 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. *739) as a parent **Link*** (p. *739).*
- 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. *739) as a child **Link*** (p. *739).*

- 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.
- bool **FindAllConnectedLinksHelper** (const **LinkPtr** &_originalParentLink, **Link_V** &_connectedLinks, bool _fistLink=false)

  Helper function to find all connected links of a link based on parent/child relations of joints.
- 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.
- **Joint_V GetChildJoints** () const

  Get the child joints.
- **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.
- **Joint_V GetParentJoints** () const

  Get the parent joints.
- **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** () const

  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.
- double **GetWorldEnergy** () const

  *Returns this link's total energy, or sum of **Link::GetWorldEnergyPotential()** (p. 758) and **Link::GetWorldEnergyKinetic()** (p. 758).*
- double **GetWorldEnergyKinetic** () const

  Returns this link's kinetic energy computed using link's CoG velocity in the inertial (world) frame.
- double **GetWorldEnergyPotential** () const

  Returns this link's potential energy, based on position in world frame and gravity.
- virtual **math::Vector3 GetWorldForce** () const =0

  Get the force applied to the body in the world frame.
- **math::Pose GetWorldInertialPose** () const

  Get the world pose of the link inertia (cog position and Moment of Inertia frame).
- **math::Matrix3 GetWorldInertiaMatrix** () const

  Get the inertia matrix in the world frame.
- **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the body in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the origin of the link frame, expressed in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_offset) 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.
- void **MoveFrame** (const **math::Pose** &_worldReferenceFrameSrc, const **math::Pose** &_worldReferenceFrameDst)

  *Move **Link** (p. 739) given source and target frames specified in world coordinates.*
- 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.

- virtual void **RemoveChild** (**EntityPtr** _child)
- void **RemoveChildJoint** (const std::string &_jointName)

  *Remove Joints that have this **Link*** (p. *739) as a parent **Link*** (p. *739).*
- void **RemoveCollision** (const std::string &_name)

  Remove a collision from the link.
- void **RemoveParentJoint** (const std::string &_jointName)

  *Remove Joints that have this **Link*** (p. *739) as a child **Link*** (p. *739).*
- 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 void **SetLinkStatic** (bool _static)=0

  Freeze link to ground (inertial frame).
- void **SetPublishData** (bool _enable)

  Enable/Disable link data publishing.
- void **SetScale** (const **math::Vector3** &_scale)

  Set the scale of the link.
- 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.
- void **Update** (const **common::UpdateInfo** &_info)

  Update the collision.
- 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 Types

- typedef std::map< uint32_t, msgs::Visual > **Visuals_M**

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. 654) properties.*
- bool **initialized**

  This flag is set to true when the link is initialized.

- **math::Vector3 linearAccel**

  Linear acceleration.
- **Visuals_M visuals**

  Link *(*p. *739) visual elements.*

10.126.1 Detailed Description

Link (p. 739) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

10.126.2 Member Typedef Documentation

10.126.2.1 typedef std::map<uint32_t, msgs::Visual> gazebo::physics::Link::Visuals_M [protected]

10.126.3 Constructor & Destructor Documentation

10.126.3.1 gazebo::physics::Link::Link (EntityPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of this link.

10.126.3.2 virtual gazebo::physics::Link::~Link ( ) [virtual]

Destructor.

10.126.4 Member Function Documentation

10.126.4.1 void gazebo::physics::Link::AddChildJoint (JointPtr *_joint*)

Joints that have this **Link** (p. 739) as a parent **Link** (p. 739).

Parameters

in	*_joint*	**Joint** (p. 669) that is a child of this link.

10.126.4.2 virtual void gazebo::physics::Link::AddForce (const math::Vector3 & _force) [pure virtual]

Add a force to the body.

Parameters

in	*_force*	Force to add.

Implemented in **gazebo::physics::SimbodyLink** (p. 1211), and **gazebo::physics::DARTLink** (p. 421).

10.126.4.3 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.

Implemented in **gazebo::physics::SimbodyLink** (p. 1211), and **gazebo::physics::DARTLink** (p. 422).

10.126.4.4 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.

Implemented in **gazebo::physics::SimbodyLink** (p. 1211), and **gazebo::physics::DARTLink** (p. 422).

10.126.4.5 void gazebo::physics::Link::AddParentJoint (JointPtr _joint)

Joints that have this **Link** (p. 739) as a child **Link** (p. 739).

Parameters

in	*_joint*	**Joint** (p. 669) that is a parent of this link.

10.126.4.6 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.

Implemented in **gazebo::physics::SimbodyLink** (p. 1212), and **gazebo::physics::DARTLink** (p. 422).

10.126.4.7 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.

Implemented in **gazebo::physics::SimbodyLink** (p. 1212), and **gazebo::physics::DARTLink** (p. 422).

10.126.4.8 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.

Implemented in **gazebo::physics::SimbodyLink** (p. 1212), and **gazebo::physics::DARTLink** (p. 423).

10.126.4.9 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.

10.126.4.10 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.

10.126.4.11 void gazebo::physics::Link::DetachAllStaticModels ( )

Detach all static models from this link.

10.126.4.12 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.

10.126.4.13 void gazebo::physics::Link::DisconnectEnabled (event::ConnectionPtr & *_conn*) [inline]

Disconnect to the add entity signal.

Parameters

in	*_conn*	Connection pointer to disconnect.

10.126.4.14 void gazebo::physics::Link::FillMsg (msgs::Link & *_msg*)

Fill a link message.

Parameters

out	*_msg*	Message to fill

10.126.4.15 bool gazebo::physics::Link::FindAllConnectedLinksHelper (const LinkPtr & *_originalParentLink,* Link_V & *_connectedLinks,* bool *_fistLink =* `false`)

Helper function to find all connected links of a link based on parent/child relations of joints.

For example, if Link0 --> Link1 --> ... --> LinkN is a kinematic chain with Link0 being the base link. Then, call by Link1: Link1->FindAllConnectedLinksHelper(Link0, _list, true); should return true with _list containing Link1 through LinkN. In the case the _originalParentLink is part of a loop, _connectedLinks is cleared and the function returns false.

Parameters

in	*_originParentLink*	if this link is a child link of the search, we've found a loop.
	in/out]	_connectedLinks aggregate list of connected links.
in	*_fistLink*	this is the first **Link** (p. 739), skip over the parent link that matches the _originalParentLink.

Returns

true if successfully found a subset of connected links

10.126.4.16 void gazebo::physics::Link::Fini ( ) [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Entity** (p. 504).

Reimplemented in **gazebo::physics::DARTLink** (p. 423), and **gazebo::physics::SimbodyLink** (p. 1212).

10.126.4.17 double gazebo::physics::Link::GetAngularDamping ( ) const

Get the angular damping factor.

Returns

Angular damping.

10.126.4.18 virtual math::Box gazebo::physics::Link::GetBoundingBox ( ) const [virtual]

Get the bounding box for the link and all the child elements.

Returns

The link's bounding box.

Reimplemented from **gazebo::physics::Entity** (p. 504).

10.126.4.19 Joint_V gazebo::physics::Link::GetChildJoints ( ) const

Get the child joints.

10.126.4.20 Link_V gazebo::physics::Link::GetChildJointsLinks ( ) const

Returns a vector of children Links connected by joints.

Returns

A vector of children Links connected by joints.

10.126.4.21 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.

10.126.4.22 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.

10.126.4.23 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.

10.126.4.24 Collision_V gazebo::physics::Link::GetCollisions ( ) const

Get all the child collisions.

Returns

A std::vector of all the child collisions.

10.126.4.25 virtual bool gazebo::physics::Link::GetEnabled ( ) const [pure virtual]

Get whether this body is enabled in the physics engine.

Returns

True if the link is enabled.

Implemented in **gazebo::physics::DARTLink** (p. 424), and **gazebo::physics::SimbodyLink** (p. 1213).

10.126.4.26 virtual bool gazebo::physics::Link::GetGravityMode ( ) const [pure virtual]

Get the gravity mode.

Returns

True if gravity is enabled.

Implemented in **gazebo::physics::DARTLink** (p. 424), and **gazebo::physics::SimbodyLink** (p. 1213).

10.126.4.27 InertialPtr gazebo::physics::Link::GetInertial ( ) const [inline]

Get the inertia of the link.

Returns

Inertia of the link.

10.126.4.28 virtual bool gazebo::physics::Link::GetKinematic ( ) const [inline, virtual]

Implement this function.

Get whether this body is in the kinematic state.

Returns

True if the link is kinematic only.

Reimplemented in **gazebo::physics::DARTLink** (p. 424).

10.126.4.29 double gazebo::physics::Link::GetLinearDamping ( ) const

Get the linear damping factor.

Returns

Linear damping.

10.126.4.30 ModelPtr gazebo::physics::Link::GetModel ( ) const

Get the model that this body belongs to.

Returns

Model (p. 846) that this body belongs to.

10.126.4.31 Joint_V gazebo::physics::Link::GetParentJoints ( ) const

Get the parent joints.

10.126.4.32 Link_V gazebo::physics::Link::GetParentJointsLinks ( ) const

Returns a vector of parent Links connected by joints.

Returns

Vector of parent Links connected by joints.

10.126.4.33 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. 506).

10.126.4.34 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. 506).

10.126.4.35 math::Vector3 gazebo::physics::Link::GetRelativeForce ( ) const

Get the force applied to the body.

Returns

Force applied to the body.

10.126.4.36 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. 507).

10.126.4.37 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. 507).

10.126.4.38 math::Vector3 gazebo::physics::Link::GetRelativeTorque ( ) const

Get the torque applied to the body.

Returns

Torque applied to the body.

10.126.4.39 bool gazebo::physics::Link::GetSelfCollide ( ) const

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.

10.126.4.40 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. 756).

Returns

The number of sensors created by the link.

10.126.4.41 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. 756).

Note

A **Link** (p. 739) does not manage or maintain a pointer to a **sensors::Sensor** (p. 1130). Access to a Sensor object is accomplished through the **sensors::SensorManager** (p. 1146). 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.

10.126.4.42 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. 507).

10.126.4.43 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.

Implemented in **gazebo::physics::DARTLink** (p. 425), and **gazebo::physics::SimbodyLink** (p. 1213).

10.126.4.44 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.

10.126.4.45 double gazebo::physics::Link::GetWorldEnergy ( ) const

Returns this link's total energy, or sum of **Link::GetWorldEnergyPotential()** (p. 758) and **Link::GetWorldEnergyKinetic()** (p. 758).

Returns

this link's total energy

10.126.4.46 double gazebo::physics::Link::GetWorldEnergyKinetic ( ) const

Returns this link's kinetic energy computed using link's CoG velocity in the inertial (world) frame.

Returns

this link's kinetic energy

10.126.4.47 double gazebo::physics::Link::GetWorldEnergyPotential ( ) const

Returns this link's potential energy, based on position in world frame and gravity.

Returns

this link's potential energy,

10.126.4.48 virtual math::Vector3 gazebo::physics::Link::GetWorldForce ( ) const [pure virtual]

Get the force applied to the body in the world frame.

Returns

Force applied to the body in the world frame.

Implemented in **gazebo::physics::DARTLink** (p. 425), and **gazebo::physics::SimbodyLink** (p. 1214).

10.126.4.49 math::Pose gazebo::physics::Link::GetWorldInertialPose ( ) const

Get the world pose of the link inertia (cog position and Moment of Inertia frame).

This differs from **GetWorldCoGPose()** (p. 757), which returns the cog position in the link frame (not the Moment of Inertia frame).

Returns

Inertial (p. 654) pose in world frame.

10.126.4.50 math::Matrix3 gazebo::physics::Link::GetWorldInertiaMatrix ( ) const

Get the inertia matrix in the world frame.

Returns

Inertia matrix in world frame, returns matrix of zeros if link has no inertia.

10.126.4.51 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. 508).

10.126.4.52 virtual math::Vector3 gazebo::physics::Link::GetWorldLinearVel ( ) const [inline, virtual]

Get the linear velocity of the origin of the link frame, expressed in the world frame.

Returns

Linear velocity of the link frame.

Reimplemented from **gazebo::physics::Entity** (p. 508).

References GetWorldLinearVel(), and gazebo::math::Vector3::Zero.

Referenced by GetWorldLinearVel().

10.126.4.53 virtual math::Vector3 gazebo::physics::Link::GetWorldLinearVel (const math::Vector3 & *_offset*) 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. 739) frame will be returned.

Parameters

in	*_offset*	Offset of the point from the origin of the **Link** (p. 739) frame, expressed in the body-fixed frame.

Returns

Linear velocity of the point on the body

Implemented in **gazebo::physics::DARTLink** (p. 425), and **gazebo::physics::SimbodyLink** (p. 1214).

10.126.4.54 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.

Implemented in **gazebo::physics::DARTLink** (p. 426), and **gazebo::physics::SimbodyLink** (p. 1214).

10.126.4.55 virtual math::Vector3 gazebo::physics::Link::GetWorldTorque ( ) const [pure virtual]

Get the torque applied to the body in the world frame.

Returns

Torque applied to the body in the world frame.

Implemented in **gazebo::physics::DARTLink** (p. 426), and **gazebo::physics::SimbodyLink** (p. 1215).

10.126.4.56 virtual void gazebo::physics::Link::Init ( ) [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Base** (p. 211).

Reimplemented in **gazebo::physics::DARTLink** (p. 426), and **gazebo::physics::SimbodyLink** (p. 1215).

10.126.4.57 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. 509).

Reimplemented in **gazebo::physics::DARTLink** (p. 427), and **gazebo::physics::SimbodyLink** (p. 1215).

10.126.4.58 void gazebo::physics::Link::MoveFrame (const math::Pose & _worldReferenceFrameSrc, const math::Pose & _worldReferenceFrameDst)

Move **Link** (p. 739) given source and target frames specified in world coordinates.

Assuming link's relative pose to source frame (_worldReferenceFrameSrc) remains unchanged relative to destination frame (_worldReferenceFrameDst).

Parameters

in	*_worldReferenceFrameSrc*	initial reference frame to which this link is attached.
in	*_worldReferenceFrameDst*	final location of the reference frame specified in world coordinates.

10.126.4.59 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. 509).

Reimplemented in **gazebo::physics::DARTLink** (p. 427), and **gazebo::physics::SimbodyLink** (p. 1216).

10.126.4.60 void gazebo::physics::Link::ProcessMsg (const msgs::Link & _msg)

Update parameters from a message.

Parameters

in	*_msg*	Message to read.

10.126.4.61 virtual void gazebo::physics::Link::RemoveChild (EntityPtr _child) [virtual]

10.126.4.62 void gazebo::physics::Link::RemoveChildJoint (const std::string & _jointName)

Remove Joints that have this **Link** (p. 739) as a parent **Link** (p. 739).

Parameters

in	*_jointName*	Child **Joint** (p. 669) name.

10.126.4.63 void gazebo::physics::Link::RemoveCollision (const std::string & _name)

Remove a collision from the link.

Parameters

int]	_name Name of the collision to remove.

10.126.4.64 void gazebo::physics::Link::RemoveParentJoint (const std::string & _jointName)

Remove Joints that have this **Link** (p. 739) as a child **Link** (p. 739).

Parameters

in	*_jointName*	Parent **Joint** (p. 669) name.

10.126.4.65 void gazebo::physics::Link::Reset ( ) [virtual]

Reset the link.

Reimplemented from **gazebo::physics::Entity** (p. 510).

10.126.4.66 void gazebo::physics::Link::ResetPhysicsStates ( )

Reset the link.

10.126.4.67 void gazebo::physics::Link::SetAngularAccel (const math::Vector3 & _accel)

Set the angular acceleration of the body.

Parameters

in	*_accel*	Angular acceleration.

10.126.4.68 virtual void gazebo::physics::Link::SetAngularDamping (double _damping) [pure virtual]

Set the angular damping factor.

Parameters

in	*_damping*	Angular damping factor.

Implemented in **gazebo::physics::DARTLink** (p. 427), and **gazebo::physics::SimbodyLink** (p. 1216).

10.126.4.69 virtual void gazebo::physics::Link::SetAngularVel (const math::Vector3 & _vel) [pure virtual]

Set the angular velocity of the body.

Parameters

in	*_vel*	Angular velocity.

Implemented in **gazebo::physics::DARTLink** (p. 427), and **gazebo::physics::SimbodyLink** (p. 1216).

10.126.4.70 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.

Implemented in **gazebo::physics::DARTLink** (p. 428), and **gazebo::physics::SimbodyLink** (p. 1216).

10.126.4.71 void gazebo::physics::Link::SetCollideMode (const std::string & *_mode*)

Set the collide mode of the body.

Parameters

in	*_mode*	**Collision** (p. 295) 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

10.126.4.72 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.

Implemented in **gazebo::physics::DARTLink** (p. 428), and **gazebo::physics::SimbodyLink** (p. 1217).

10.126.4.73 virtual void gazebo::physics::Link::SetForce (const math::Vector3 & *_force*) [pure virtual]

Set the force applied to the body.

Parameters

in	*_force*	Force value.

Implemented in **gazebo::physics::DARTLink** (p. 428), and **gazebo::physics::SimbodyLink** (p. 1217).

10.126.4.74 virtual void gazebo::physics::Link::SetGravityMode (bool _mode) [pure virtual]

Set whether gravity affects this body.

Parameters

in	*_mode*	True to enable gravity.

Implemented in **gazebo::physics::DARTLink** (p. 429), and **gazebo::physics::SimbodyLink** (p. 1217).

10.126.4.75 void gazebo::physics::Link::SetInertial (const InertialPtr & _inertial)

Set the mass of the link.

[in] _inertial **Inertial** (p. 654) value for the link.

10.126.4.76 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.

Reimplemented in **gazebo::physics::DARTLink** (p. 429).

10.126.4.77 void gazebo::physics::Link::SetLaserRetro (float _retro)

Set the laser retro reflectiveness.

Parameters

in	*_retro*	Retro value for all child collisions.

10.126.4.78 void gazebo::physics::Link::SetLinearAccel (const math::Vector3 & _accel)

Set the linear acceleration of the body.

Parameters

in	*_accel*	Linear acceleration.

10.126.4.79 virtual void gazebo::physics::Link::SetLinearDamping (double *_damping*) [pure virtual]

Set the linear damping factor.

Parameters

in	*_damping*	Linear damping factor.

Implemented in **gazebo::physics::DARTLink** (p. 429), and **gazebo::physics::SimbodyLink** (p. 1217).

10.126.4.80 virtual void gazebo::physics::Link::SetLinearVel (const math::Vector3 & *_vel*) [pure virtual]

Set the linear velocity of the body.

Parameters

in	*_vel*	Linear velocity.

Implemented in **gazebo::physics::DARTLink** (p. 429), and **gazebo::physics::SimbodyLink** (p. 1218).

10.126.4.81 virtual void gazebo::physics::Link::SetLinkStatic (bool *_static*) [pure virtual]

Freeze link to ground (inertial frame).

Parameters

in	*_static*	if true, freeze link to ground. Otherwise unfreeze link.

Implemented in **gazebo::physics::SimbodyLink** (p. 1218), and **gazebo::physics::DARTLink** (p. 430).

10.126.4.82 void gazebo::physics::Link::SetPublishData (bool *_enable*)

Enable/Disable link data publishing.

Parameters

in	*_enable*	True to enable publishing, false to stop publishing

10.126.4.83 void gazebo::physics::Link::SetScale (const math::Vector3 & *_scale*)

Set the scale of the link.

Parameters

in	*_scale*	Scale to set the link to.

10.126.4.84 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. 215).

10.126.4.85 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.

Implemented in **gazebo::physics::DARTLink** (p. 430), and **gazebo::physics::SimbodyLink** (p. 1218).

10.126.4.86 void gazebo::physics::Link::SetState (const LinkState & *_state*)

Set the current link state.

Parameters

in	*_state*	The state to set the link to.

10.126.4.87 virtual void gazebo::physics::Link::SetTorque (const math::Vector3 & _torque) [pure virtual]

Set the torque applied to the body.

Parameters

in	*_torque*	Torque value.

Implemented in **gazebo::physics::DARTLink** (p. 430), and **gazebo::physics::SimbodyLink** (p. 1219).

10.126.4.88 void gazebo::physics::Link::Update (const common::UpdateInfo & _info)

Update the collision.

Parameters

in	*_info*	Update information.

10.126.4.89 virtual void gazebo::physics::Link::UpdateMass ( ) [inline, virtual]

Update the mass matrix.

10.126.4.90 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. 512).

10.126.4.91 virtual void gazebo::physics::Link::UpdateSurface ( ) [inline, virtual]

Update surface parameters.

10.126.5 Member Data Documentation

10.126.5.1 math::Vector3 gazebo::physics::Link::angularAccel [protected]

Angular acceleration.

10.126.5.2 std::vector<math::Pose> gazebo::physics::Link::attachedModelsOffset [protected]

Offsets for the attached models.

10.126.5.3 std::vector<std::string> gazebo::physics::Link::cgVisuals [protected]

Center of gravity visual elements.

10.126.5.4 InertialPtr gazebo::physics::Link::inertial [protected]

Inertial (p. 654) properties.

10.126.5.5 bool gazebo::physics::Link::initialized [protected]

This flag is set to true when the link is initialized.

10.126.5.6 math::Vector3 gazebo::physics::Link::linearAccel [protected]

Linear acceleration.

10.126.5.7 Visuals_M gazebo::physics::Link::visuals [protected]

Link (p. 739) visual elements.

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

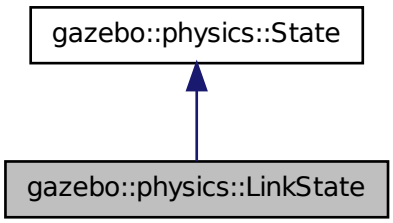
- **Link.hh**

10.127 gazebo::physics::LinkState Class Reference

Store state information of a **physics::Link** (p. 739) object.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::LinkState:



Public Member Functions

- **LinkState** ()

  Default constructor.
- **LinkState** (const **LinkPtr** _link, const **common::Time** &_realTime, const **common::Time** &_simTime)

  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. 739).*
- bool **IsZero** () const

  Return true if the values in the state are zero.
- void **Load** (const **LinkPtr** _link, const **common::Time** &_realTime, const **common::Time** &_simTime)

  *Load a **LinkState** (p. 769) from a **Link** (p. 739) pointer.*
- 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.
- virtual void **SetRealTime** (const **common::Time** &_time)

  Set the real time when this state was generated.
- virtual void **SetSimTime** (const **common::Time** &_time)

  Set the sim time when this state was generated.
- virtual void **SetWallTime** (const **common::Time** &_time)

  Set the wall time when this state was generated.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::LinkState** &_state)

  Stream insertion operator.

10.127.1 Detailed Description

Store state information of a **physics::Link** (p. 739) object.

This class captures the entire state of a **Link** (p. 739) at one specific time during a simulation run.

State (p. 1323) of a **Link** (p. 739) includes the state of itself all its child **Collision** (p. 295) entities.

10.127.2 Constructor & Destructor Documentation

10.127.2.1 gazebo::physics::LinkState::LinkState ( )

Default constructor.

10.127.2.2 gazebo::physics::LinkState::LinkState (const LinkPtr *_link,* const common::Time & *_realTime,* const common::Time & *_simTime*)

Constructor.

Build a **LinkState** (p. 769) from an existing **Link** (p. 739).

Parameters

in	*_model*	Pointer to the **Link** (p. 739) from which to gather state info.
in	*_realTime*	Real time stamp.
in	*_simTime*	Sim time stamp

10.127.2.3 gazebo::physics::LinkState::LinkState (const LinkPtr *_link*) [explicit]

Constructor.

Build a **LinkState** (p. 769) from an existing **Link** (p. 739).

Parameters

in	*_model*	Pointer to the **Link** (p. 739) from which to gather state info.

10.127.2.4 gazebo::physics::LinkState::LinkState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a **LinkState** (p. 769) from SDF data

Parameters

in	*_sdf*	SDF data to load a link state from.

10.127.2.5 virtual gazebo::physics::LinkState::~LinkState ( ) [virtual]

Destructor.

10.127.3 Member Function Documentation

10.127.3.1 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.

10.127.3.2 const math::Pose& gazebo::physics::LinkState::GetAcceleration ( ) const

Get the link acceleration.

Returns

The acceleration represented as a **math::Pose** (p. 995).

10.127.3.3 CollisionState gazebo::physics::LinkState::GetCollisionState (unsigned int _index) const

Get a collision state.

Get a **Collision** (p. 295) **State** (p. 1323) based on an index, where index is in the range of 0...**LinkState::GetCollisionStateCount** (p. 774).

Parameters

in	*_index*	Index of the **CollisionState** (p. 307).

Returns

State (p. 1323) of the **Collision** (p. 295).

Exceptions

common::-Exception *(p. 548)*	When _index is invalid.

10.127.3.4 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. 307) with the matching name, if any.

Parameters

in	*_collisionName*	Name of the **CollisionState** (p. 307)

Returns

State (p. 1323) of the **Collision** (p. 295).

Exceptions

common::Exception *(p. 548)*	When _collisionName is invalid

10.127.3.5 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. 307) recorded.

10.127.3.6 const std::vector<CollisionState>& gazebo::physics::LinkState::GetCollisionStates ( ) const

Get the collision states.

Returns

A vector of collision states.

10.127.3.7 const math::Pose& gazebo::physics::LinkState::GetPose ( ) const

Get the link pose.

Returns

The **math::Pose** (p. 995) of the **Link** (p. 739).

10.127.3.8 const math::Pose& gazebo::physics::LinkState::GetVelocity ( ) const

Get the link velocity.

Returns

The velocity represented as a **math::Pose** (p. 995).

10.127.3.9 const math::Pose& gazebo::physics::LinkState::GetWrench ( ) const

Get the force applied to the **Link** (p. 739).

Returns

Magnitude of the force.

10.127.3.10 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.

10.127.3.11 void gazebo::physics::LinkState::Load (const LinkPtr *_link*, const common::Time & *_realTime*, const common::Time & *_simTime*)

Load a **LinkState** (p. 769) from a **Link** (p. 739) pointer.

Build a **LinkState** (p. 769) from an existing **Link** (p. 739).

Parameters

in	*_model*	Pointer to the **Link** (p. 739) from which to gather state info.
in	*_realTime*	Real time stamp.
in	*_simTime*	Sim time stamp

10.127.3.12 virtual void gazebo::physics::LinkState::Load (const sdf::ElementPtr _elem) [virtual]

Load state from SDF element.

Load **LinkState** (p. 769) 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. 1326).

10.127.3.13 LinkState gazebo::physics::LinkState::operator+ (const LinkState & _state) const

Addition operator.

Parameters

in	*_pt*	A state to add.

Returns

The resulting state.

10.127.3.14 LinkState gazebo::physics::LinkState::operator- (const LinkState & _state) const

Subtraction operator.

Parameters

in	*_pt*	A state to substract.

Returns

The resulting state.

10.127.3.15 LinkState& gazebo::physics::LinkState::operator= (const LinkState & _state)

Assignment operator.

Parameters

in	*_state*	**State** (p. 1323) value

Returns

this

10.127.3.16 virtual void gazebo::physics::LinkState::SetRealTime (const common::Time & *_time*) [virtual]

Set the real time when this state was generated.

Parameters

in	*_time*	Clock time since simulation was stated.

Reimplemented from **gazebo::physics::State** (p. 1327).

10.127.3.17 virtual void gazebo::physics::LinkState::SetSimTime (const common::Time & *_time*) [virtual]

Set the sim time when this state was generated.

Parameters

in	*_time*	Simulation time when the data was recorded.

Reimplemented from **gazebo::physics::State** (p. 1327).

10.127.3.18 virtual void gazebo::physics::LinkState::SetWallTime (const common::Time & *_time*) [virtual]

Set the wall time when this state was generated.

Parameters

in	*_time*	The absolute clock time when the **State** (p. 1323) data was recorded.

Reimplemented from **gazebo::physics::State** (p. 1328).

10.127.4 Friends And Related Function Documentation

10.127.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::physics::LinkState & *_state*) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_state*	**Link** (p. 739) state to output

Returns

the stream

Disabling this for efficiency.

Disabling this for efficiency.

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

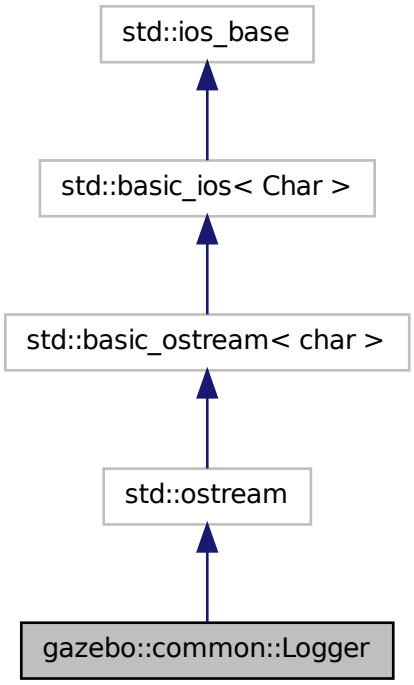
- **LinkState.hh**

10.128 gazebo::common::Logger Class Reference

Terminal logger.

`#include <Console.hh>`

Inheritance diagram for gazebo::common::Logger:



Classes

- class **Buffer**

  String buffer for the base logger.

Public Types

- enum **LogType** { **STDOUT**, **STDERR** }

Public Member Functions

- **Logger** (const std::string &_prefix, int _color, **LogType** _type)

  Constructor.
- virtual ~**Logger** ()

  Destructor.
- virtual **Logger** & **operator()** ()

  Access operator.
- virtual **Logger** & **operator()** (const std::string &_file, int _line)

  Output a filename and line number, then return a reference to the logger.

Public Attributes

- int **color**

  Color *(*p. *312) for the output.*

10.128.1 Detailed Description

Terminal logger.

10.128.2 Member Enumeration Documentation

10.128.2.1 enum gazebo::common::Logger::LogType

Output destination type.

Enumerator:

STDOUT Output to stdout.

STDERR Output to stderr.

10.128.3 Constructor & Destructor Documentation

10.128.3.1 gazebo::common::Logger::Logger (const std::string & *_prefix*, int *_color*, LogType *_type*)

Constructor.

Parameters

in	*_prefix*	String to use as prefix when logging to file.
in	*_color*	**Color** (p. 312) of the output stream.
in	*_type*	Output destination type (STDOUT, or STDERR)

10.128.3.2 virtual gazebo::common::Logger::~Logger ( ) [virtual]

Destructor.

10.128.4 Member Function Documentation

10.128.4.1 virtual Logger& gazebo::common::Logger::operator() ( ) [virtual]

Access operator.

Returns

Reference to this logger.

10.128.4.2 virtual Logger& gazebo::common::Logger::operator() (const std::string & *_file,* int *_line*) [virtual]

Output a filename and line number, then return a reference to the logger.

Parameters

in	*_file*	Filename to output.
in	*_line*	Line number in the _file.

Returns

Reference to this logger.

10.128.5 Member Data Documentation

10.128.5.1 int gazebo::common::Logger::color

Color (p. 312) for the output.

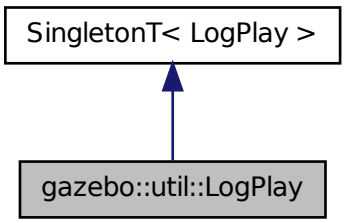
The documentation for this class was generated from the following file:

- **Console.hh**

10.129 gazebo::util::LogPlay Class Reference

```
#include <LogPlay.hh>
```

Inheritance diagram for gazebo::util::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 **GetHeader** () const

  Get the header that was read from a 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.

10.129.1 Member Function Documentation

10.129.1.1 bool gazebo::util::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.

10.129.1.2 unsigned int gazebo::util::LogPlay::GetChunkCount ( ) const

Get the number of chunks (steps) in the open log file.

Returns

The number of recorded states in the log file.

10.129.1.3 std::string gazebo::util::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. 785) has not been called at least once.

10.129.1.4 std::string gazebo::util::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.

10.129.1.5 std::string gazebo::util::LogPlay::GetHeader ( ) const

Get the header that was read from a log file.

Should call **LogPlay::Open** (p. 784) first.

Returns

Header of the open log file.

10.129.1.6 std::string gazebo::util::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.

10.129.1.7 uint32_t gazebo::util::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. 1048).

10.129.1.8 bool gazebo::util::LogPlay::IsOpen ( ) const

Return true if a file is open.

Returns

True if a log file is open.

10.129.1.9 void gazebo::util::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	

10.129.1.10 bool gazebo::util::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**

10.130 Logplay Class Reference

Open and playback log files that were recorded using LogRecord.

10.130.1 Detailed Description

Open and playback log files that were recorded using LogRecord.

Use **Logplay** (p. 785) 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:

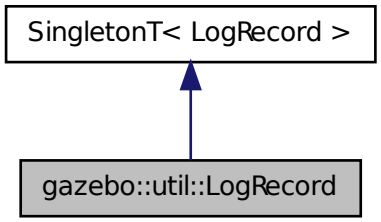
- **LogPlay.hh**

10.131 gazebo::util::LogRecord Class Reference

addtogroup gazebo_util

`#include <util/util.hh>`

Inheritance diagram for gazebo::util::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.
- unsigned int **GetBufferSize** () const

  Get the size of the buffer.
- 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 **IsReadyToStart** () const

  Get whether the logger is ready to start, which implies that any previous runs have finished.

- void **Notify** ()

  Tell the recorder that an update should occur.

- 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="zlib", const std::string &_path="")

  Start the logger.

- void **Stop** ()

  Stop the logger.

- void **Write** (bool _force=false)

  Write all logs.

10.131.1 Detailed Description

addtogroup gazebo_util

Handles logging of data to disk

The **LogRecord** (p. 786) 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. 781) for playback functionality.

State information for an entity may be logged through the **LogRecord::Add** (p. 788) function, and stopped through the **LogRecord::Remove** (p. 791) function. Data may be logged into a single file, or split into many separate files by specifying different filenames for the **LogRecord::Add** (p. 788) function.

The **LogRecord** (p. 786) is updated at the start of each simulation step. This guarantees that all data is stored.

See also

Logplay (p. 785), State

10.131.2 Member Function Documentation

10.131.2.1 void gazebo::util::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	

10.131.2.2 void gazebo::util::LogRecord::Fini ( )

Finialize, and shutdown.

10.131.2.3 std::string gazebo::util::LogRecord::GetBasePath ( ) const

Get the base path for a log recording.

Returns

Path for log recording.

10.131.2.4 unsigned int gazebo::util::LogRecord::GetBufferSize ( ) const

Get the size of the buffer.

Returns

Size of the buffer, in bytes.

10.131.2.5 const std::string& gazebo::util::LogRecord::GetEncoding ( ) const

Get the encoding used.

Returns

Either [txt, zlib, or bz2], where txt is plain txt and bz2 and zlib are compressed data with Base64 encoding.

10.131.2.6 std::string gazebo::util::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.

10.131.2.7 unsigned int gazebo::util::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.

10.131.2.8 bool gazebo::util::LogRecord::GetFirstUpdate ( ) const

Return true if an Update has not yet been completed.

Returns

True if an Update has not yet been completed.

10.131.2.9 bool gazebo::util::LogRecord::GetPaused ( ) const

Get whether logging is paused.

Returns

True if logging is paused.

See also

LogRecord::SetPaused (p. 791)

10.131.2.10 bool gazebo::util::LogRecord::GetRunning ( ) const

Get whether logging is running.

Returns

True if logging has been started.

10.131.2.11 common::Time gazebo::util::LogRecord::GetRunTime ( ) const

Get the run time in sim time.

Returns

Run sim time.

10.131.2.12 bool gazebo::util::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.

10.131.2.13 bool gazebo::util::LogRecord::IsReadyToStart ( ) const

Get whether the logger is ready to start, which implies that any previous runs have finished.

10.131.2.14 void gazebo::util::LogRecord::Notify ( )

Tell the recorder that an update should occur.

10.131.2.15 bool gazebo::util::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.

10.131.2.16 void gazebo::util::LogRecord::SetBasePath (const std::string & *_path*)

Set the base path.

Parameters

in	*_path*	Path to the new logging location.

10.131.2.17 void gazebo::util::LogRecord::SetPaused (bool *_paused*)

Set whether logging should pause.

A 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. 790)

10.131.2.18 bool gazebo::util::LogRecord::Start (const std::string & *_encoding* = "zlib", const std::string & *_path* = "")

Start the logger.

Parameters

in	*_encoding*	The type of encoding (txt, zlib, or bz2).
in	*_path*	Path in which to store log files.

10.131.2.19 void gazebo::util::LogRecord::Stop ( )

Stop the logger.

10.131.2.20 void gazebo::util::LogRecord::Write (bool *_force* = false)

Write all logs.

Parameters

in	*_force*	True to skip waiting on dataAvailableCondition.

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

- **LogRecord.hh**

10.132 gazebo::Master Class Reference

A manager that directs topic 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.

10.132.1 Detailed Description

A manager that directs topic 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. 792), runs World update and sensor generation loops.

10.132.2 Constructor & Destructor Documentation

10.132.2.1 gazebo::Master::Master ( )

Constructor.

10.132.2.2 virtual gazebo::Master::~Master ( ) [virtual]

Destructor.

10.132.3 Member Function Documentation

10.132.3.1 void gazebo::Master::Fini ( )

Finalize the master.

10.132.3.2 void gazebo::Master::Init (uint16_t *_port*)

Initialize.

Parameters

in	*_port*	The master's port

10.132.3.3 void gazebo::Master::Run ( )

Run the master.

10.132.3.4 void gazebo::Master::RunOnce ( )

Run the master one iteration.

10.132.3.5 void gazebo::Master::RunThread ( )

Run the master in a new thread.

10.132.3.6 void gazebo::Master::Stop ( )

Stop the master.

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

- **Master.hh**

10.133 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.

10.133.1 Detailed Description

Encapsulates description of a material.

10.133.2 Member Enumeration Documentation

10.133.2.1 enum gazebo::common::Material::BlendMode

Enumerator:

ADD

MODULATE

REPLACE

BLEND_COUNT

10.133.2.2 enum gazebo::common::Material::ShadeMode

Enumerator:

FLAT

GOURAUD

PHONG

BLINN

SHADE_COUNT

10.133.3 Constructor & Destructor Documentation

10.133.3.1 gazebo::common::Material::Material ( )

Constructor.

10.133.3.2 virtual gazebo::common::Material::~Material ( ) [virtual]

Destructor.

10.133.3.3 gazebo::common::Material::Material (const Color & *_clr*)

Create a material with a default color.

Parameters

in	*_clr*	**Color** (p. 312) of the material

10.133.4 Member Function Documentation

10.133.4.1 Color gazebo::common::Material::GetAmbient ( ) const

Get the ambient color.

Returns

The ambient color

10.133.4.2 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

10.133.4.3 BlendMode gazebo::common::Material::GetBlendMode ( ) const

Get the blending mode.

Returns

the blend mode

10.133.4.4 bool gazebo::common::Material::GetDepthWrite ( ) const

Get depth write.

Returns

the depth write enabled state

10.133.4.5 Color gazebo::common::Material::GetDiffuse ( ) const

Get the diffuse color.

Returns

The diffuse color

10.133.4.6 Color gazebo::common::Material::GetEmissive ( ) const

Get the emissive color.

Returns

The emissive color

10.133.4.7 bool gazebo::common::Material::GetLighting ( ) const

Get lighting enabled.

Returns

the lighting enabled state

10.133.4.8 std::string gazebo::common::Material::GetName ( ) const

Get the name of the material.

Returns

The name of the material

10.133.4.9 double gazebo::common::Material::GetPointSize ( ) const

Get the point size.

Returns

the point size

10.133.4.10 ShadeMode gazebo::common::Material::GetShadeMode ( ) const

Get the shading mode.

Returns

the shading mode

10.133.4.11 double gazebo::common::Material::GetShininess ( ) const

Get the shininess.

Returns

The shininess value

10.133.4.12 Color gazebo::common::Material::GetSpecular ( ) const

Get the specular color.

Returns

The specular color

10.133.4.13 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

10.133.4.14 double gazebo::common::Material::GetTransparency ( ) const

Get the transparency percentage (0..1)

Returns

The transparency percentage

10.133.4.15 void gazebo::common::Material::SetAmbient (const Color & _clr)

Set the ambient color.

Parameters

in	*_clr*	The ambient color

10.133.4.16 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

10.133.4.17 void gazebo::common::Material::SetBlendMode (BlendMode _b)

Set the blending mode.

Parameters

in	*_b*	the blend mode

10.133.4.18 void gazebo::common::Material::SetDepthWrite (bool _value)

Set depth write.

Parameters

in	*_value*	the depth write enabled state

10.133.4.19 void gazebo::common::Material::SetDiffuse (const Color & _clr)

Set the diffuse color.

Parameters

in	*_clr*	The diffuse color

10.133.4.20 void gazebo::common::Material::SetEmissive (const Color & _clr)

Set the emissive color.

Parameters

in	*_clr*	The emissive color

10.133.4.21 void gazebo::common::Material::SetLighting (bool *_value*)

Set lighting enabled.

Parameters

in	*_value*	the lighting enabled state

10.133.4.22 void gazebo::common::Material::SetPointSize (double *_size*)

Set the point size.

Parameters

in	*_size*	the size

10.133.4.23 void gazebo::common::Material::SetShadeMode (ShadeMode *_b*)

Set the shading mode param[in] the shading mode.

10.133.4.24 void gazebo::common::Material::SetShininess (double *_t*)

Set the shininess.

Parameters

in	*_t*	The shininess value

10.133.4.25 void gazebo::common::Material::SetSpecular (const Color & *_clr*)

Set the specular color.

Parameters

in	*_clr*	The specular color

10.133.4.26 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

10.133.4.27 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

10.133.4.28 void gazebo::common::Material::SetTransparency (double _t)

Set the transparency percentage (0..1)

Parameters

in	*_t*	The amount of transparency (0..1)

10.133.5 Friends And Related Function Documentation

10.133.5.1 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.

10.133.6 Member Data Documentation

10.133.6.1 Color gazebo::common::Material::ambient [protected]

the ambient light color

10.133.6.2 BlendMode gazebo::common::Material::blendMode [protected]

blend mode

10.133.6.3 std::string gazebo::common::Material::BlendModeStr[BLEND_COUNT] [static]

10.133.6.4 Color gazebo::common::Material::diffuse [protected]

the diffuse ligth color

10.133.6.5 Color gazebo::common::Material::emissive [protected]

the emissive light color

10.133.6.6 std::string gazebo::common::Material::name [protected]

the name of the material

10.133.6.7 double gazebo::common::Material::pointSize [protected]

point size

10.133.6.8 ShadeMode gazebo::common::Material::shadeMode [protected]

the shade mode

10.133.6.9 std::string gazebo::common::Material::ShadeModeStr[SHADE_COUNT] [static]

10.133.6.10 double gazebo::common::Material::shininess [protected]

shininess value (0 to 1)

10.133.6.11 Color gazebo::common::Material::specular [protected]

the specular light color

10.133.6.12 std::string gazebo::common::Material::texImage [protected]

the texture image file name

10.133.6.13 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**

10.134 gazebo::math::Matrix3 Class Reference

A 3x3 matrix class.

`#include <Matrix3.hh>`

Public Member Functions

- **Matrix3** ()

  Constructor.
- **Matrix3** (const **Matrix3** &_m)

  Copy constructor.
- **Matrix3** (double _v00, double _v01, double _v02, double _v10, double _v11, double _v12, double _v20, double _v21, double _v22)

  Constructor.
- virtual ∼**Matrix3** ()

  Desctructor.
- **Matrix3 operator**∗ (const double &_s) const

  returns the element wise scalar multiplication
- **Matrix3 operator**∗ (const **Matrix3** &_m) const

  Matrix multiplication operator.
- **math::Vector3 operator**∗ (const **math::Vector3** &_v) const

  *Matrix times **Vector3*** (p. *1440) 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.

Static Public Attributes

- static const **Matrix3 IDENTITY**

  Identity matrix.

- static const **Matrix3 ZERO**

  Zero matrix.

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.

10.134.1 Detailed Description

A 3x3 matrix class.

10.134.2 Constructor & Destructor Documentation

10.134.2.1 gazebo::math::Matrix3::Matrix3 ( )

Constructor.

10.134.2.2 gazebo::math::Matrix3::Matrix3 (const Matrix3 & _m)

Copy constructor.

Parameters

_m	Matrix to copy

10.134.2.3 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

10.134.2.4 virtual gazebo::math::Matrix3::~Matrix3 ( ) [virtual]

Desctructor.

10.134.3 Member Function Documentation

10.134.3.1 Matrix3 gazebo::math::Matrix3::operator* (const double & _s) const [inline]

returns the element wise scalar multiplication

10.134.3.2 Matrix3 gazebo::math::Matrix3::operator∗ (const Matrix3 & _m) const [inline]

Matrix multiplication operator.

Parameters

in	_m	**Matrix3** (p. 806) to multiply

Returns

product of this ∗ _m

10.134.3.3 math::Vector3 gazebo::math::Matrix3::operator∗ (const math::Vector3 & _v) const [inline]

Matrix times **Vector3** (p. 1440) operator.

Parameters

in	_v	a **Vector3** (p. 1440)

Returns

this ∗ _v

References gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::-Vector3::z.

10.134.3.4 Matrix3 gazebo::math::Matrix3::operator+ (const Matrix3 & _m) const [inline]

returns the element wise sum of two matrices

10.134.3.5 Matrix3 gazebo::math::Matrix3::operator- (const Matrix3 & _m) const [inline]

returns the element wise difference of two matrices

10.134.3.6 bool gazebo::math::Matrix3::operator== (const Matrix3 & _m) const

Equality test operator.

Parameters

in	*_m*	**Matrix3** (p. 806) to test

Returns

True if equal (using the default tolerance of 1e-6)

10.134.3.7 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

10.134.3.8 double* gazebo::math::Matrix3::operator[] (size_t *_row*) [inline]

Array subscript operator.

Parameters

in	*_row*	row index

Returns

a pointer to the row

10.134.3.9 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

10.134.3.10 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

10.134.3.11 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

10.134.4 Friends And Related Function Documentation

10.134.4.1 Matrix3 operator∗ (double _s, const Matrix3 & _m) [friend]

Multiplication operators.

Parameters

in	*_s*	the scaling factor
in	*_m*	input matrix

Returns

a scaled matrix

10.134.4.2 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

10.134.5 Member Data Documentation

10.134.5.1 const Matrix3 gazebo::math::Matrix3::IDENTITY [static]

Identity matrix.

10.134.5.2 double gazebo::math::Matrix3::m[3][3] [protected]

the 3x3 matrix

10.134.5.3 const Matrix3 gazebo::math::Matrix3::ZERO [static]

Zero matrix.

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

- **Matrix3.hh**

10.135 gazebo::math::Matrix4 Class Reference

A 3x3 matrix class.

`#include <math/gzmath.hh>`

Public Member Functions

- **Matrix4** ()

  Constructor.
- **Matrix4** (const **Matrix4** &_m)

  Copy constructor.
- **Matrix4** (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

Constructor.

- virtual ∼**Matrix4** ()

  Destructor.

- **math::Pose GetAsPose** () const

  *Get the transformation as **math::Pose** (*p. *995).*

- **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. *1440).*

- **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.

10.135.1 Detailed Description

A 3x3 matrix class.

10.135.2 Constructor & Destructor Documentation

10.135.2.1 gazebo::math::Matrix4::Matrix4 ( )

Constructor.

10.135.2.2 gazebo::math::Matrix4::Matrix4 (const Matrix4 & *_m*)

Copy constructor.

Parameters

_m	Matrix to copy

10.135.2.3 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

10.135.2.4 virtual gazebo::math::Matrix4::~Matrix4 ( ) [virtual]

Destructor.

10.135.3 Member Function Documentation

10.135.3.1 math::Pose gazebo::math::Matrix4::GetAsPose ( ) const

Get the transformation as **math::Pose** (p. 995).

Returns

the pose

10.135.3.2 Vector3 gazebo::math::Matrix4::GetEulerRotation (unsigned int *solution_number* = `1`) const

Get the rotation as a Euler angles.

Returns

the rotation

10.135.3.3 Quaternion gazebo::math::Matrix4::GetRotation ( ) const

Get the rotation as a quaternion.

Returns

the rotation

10.135.3.4 Vector3 gazebo::math::Matrix4::GetTranslation ( ) const

Get the translational values as a **Vector3** (p. 1440).

Returns

x,y,z

10.135.3.5 Matrix4 gazebo::math::Matrix4::Inverse ( ) const

Return the inverse matrix.

10.135.3.6 bool gazebo::math::Matrix4::IsAffine ( ) const

Return true if the matrix is affine.

Returns

true if the matrix is affine, false otherwise

10.135.3.7 Matrix4 gazebo::math::Matrix4::operator* (const Matrix4 & *_mat*) const

Multiplication operator.

Parameters

_mat	Incoming matrix

Returns

This matrix * _mat

10.135.3.8 Matrix4 gazebo::math::Matrix4::operator* (const Matrix3 & *_mat*) const

Multiplication operator.

Parameters

_mat	Incoming matrix

Returns

This matrix * _mat

10.135.3.9 Vector3 gazebo::math::Matrix4::operator* (const Vector3 & *_vec*) const

Multiplication operator.

Parameters

_vec	**Vector3** (p. 1440)

Returns

Resulting vector from multiplication

10.135.3.10 Matrix4& gazebo::math::Matrix4::operator= (const Matrix4 & *_mat*)

Equal operator.

this = _mat

Parameters

_mat	Incoming matrix

Returns

itself

10.135.3.11 const Matrix4& gazebo::math::Matrix4::operator= (const Matrix3 & *_mat*)

Equal operator for 3x3 matrix.

Parameters

_mat	Incoming matrix

Returns

itself

10.135.3.12 bool gazebo::math::Matrix4::operator== (const Matrix4 & *_m*) const

Equality operator.

Parameters

in	*_m*	**Matrix3** (p. 806) to test

Returns

true if the 2 matrices are equal (using the tolerance 1e-6), false otherwise

10.135.3.13 double* gazebo::math::Matrix4::operator[] (size_t *_row*) [inline]

Array subscript operator.

Parameters

in	*_row*	the row index

Returns

the row

10.135.3.14 const double* gazebo::math::Matrix4::operator[] (size_t _row) const [inline]

Parameters

in	*_row*	the row index

Returns

the row

10.135.3.15 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

10.135.3.16 void gazebo::math::Matrix4::SetScale (const Vector3 & _s)

Set the scale.

Parameters

in	*_s*	scale

10.135.3.17 void gazebo::math::Matrix4::SetTranslate (const Vector3 & _t)

Set the translational values [(0, 3) (1, 3) (2, 3)].

Parameters

in	*_t*	Values to set

10.135.3.18 Vector3 gazebo::math::Matrix4::TransformAffine (const Vector3 & _v) const

Perform an affine transformation.

Parameters

_v	**Vector3** (p. 1440) value for the transformation

Returns

The result of the transformation

10.135.4 Friends And Related Function Documentation

10.135.4.1 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

10.135.5 Member Data Documentation

10.135.5.1 const Matrix4 gazebo::math::Matrix4::IDENTITY [static]

Identity matrix.

10.135.5.2 double gazebo::math::Matrix4::m[4][4] [protected]

The 4x4 matrix.

10.135.5.3 const Matrix4 gazebo::math::Matrix4::ZERO [static]

Zero matrix.

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

- **Matrix4.hh**

10.136 gazebo::common::Mesh Class Reference

A 3D mesh.

`#include <common/common.hh>`

Public Member Functions

- **Mesh** ()

  Constructor.
- virtual ∼**Mesh** ()

  Destructor.
- int **AddMaterial** (**Material** ∗_mat)

  Add a material to the mesh.
- void **AddSubMesh** (**SubMesh** ∗_child)

  Add a submesh mesh.
- void **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.

- int **GetMaterialIndex** (const **Material** *_mat) const

  Get the index of material.

- **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.

10.136.1 Detailed Description

A 3D mesh.

10.136.2 Constructor & Destructor Documentation

10.136.2.1 gazebo::common::Mesh::Mesh ( )

Constructor.

10.136.2.2 virtual gazebo::common::Mesh::∼Mesh ( ) [virtual]

Destructor.

10.136.3 Member Function Documentation

10.136.3.1 int gazebo::common::Mesh::AddMaterial (Material ∗ *_mat*)

Add a material to the mesh.

Parameters

in	*_mat*	the material

Returns

Index of this material

10.136.3.2 void gazebo::common::Mesh::AddSubMesh (SubMesh ∗ *_child*)

Add a submesh mesh.

The **Mesh** (p. 821) object takes ownership of the submesh.

Parameters

in	*_child*	the submesh

10.136.3.3 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.

10.136.3.4 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

10.136.3.5 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

10.136.3.6 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

10.136.3.7 unsigned int gazebo::common::Mesh::GetIndexCount ( ) const

Return the number of indices.

Returns

the count

10.136.3.8 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

10.136.3.9 unsigned int gazebo::common::Mesh::GetMaterialCount ( ) const

Get the number of materials.

Returns

the count

10.136.3.10 int gazebo::common::Mesh::GetMaterialIndex (const Material * *_mat*) const

Get the index of material.

Parameters

in	*_mat*	the material

Returns

the index of the material or -1 if not found.

10.136.3.11 math::Vector3 gazebo::common::Mesh::GetMax ( ) const

Get the maximun X, Y, Z values.

Returns

the upper bounds of the bounding box

10.136.3.12 math::Vector3 gazebo::common::Mesh::GetMin ( ) const

Get the minimum X, Y, Z values.

Returns

the lower bounds of the bounding box

10.136.3.13 std::string gazebo::common::Mesh::GetName ( ) const

Get the name of this mesh.

Returns

the name

10.136.3.14 unsigned int gazebo::common::Mesh::GetNormalCount ( ) const

Return the number of normals.

Returns

the count

10.136.3.15 std::string gazebo::common::Mesh::GetPath ( ) const

Get the path which contains the mesh resource.

Returns

the path to the mesh resource

10.136.3.16 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.

10.136.3.17 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

10.136.3.18 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.

10.136.3.19 unsigned int gazebo::common::Mesh::GetSubMeshCount ( ) const

Get the number of children.

Returns

the count

10.136.3.20 unsigned int gazebo::common::Mesh::GetTexCoordCount ( ) const

Return the number of texture coordinates.

Returns

the count

10.136.3.21 unsigned int gazebo::common::Mesh::GetVertexCount ( ) const

Return the number of vertices.

Returns

the count

10.136.3.22 bool gazebo::common::Mesh::HasSkeleton ( ) const

Return true if mesh is attached to a skeleton.

10.136.3.23 void gazebo::common::Mesh::RecalculateNormals ( )

Recalculate all the normals of each face defined by three indices.

10.136.3.24 void gazebo::common::Mesh::Scale (double *_factor*)

Scale all vertices by _factor.

Parameters

_factor	Scaling factor

10.136.3.25 void gazebo::common::Mesh::SetName (const std::string & *_n*)

Set the name of this mesh.

Parameters

in	*_n*	the name to set

10.136.3.26 void gazebo::common::Mesh::SetPath (const std::string & _path)

Set the path which contains the mesh resource.

Parameters

in	*_path*	the file path

10.136.3.27 void gazebo::common::Mesh::SetScale (const math::Vector3 & _factor)

Scale all vertices by the _factor vector.

Parameters

in	*_factor*	Scaling vector

10.136.3.28 void gazebo::common::Mesh::SetSkeleton (Skeleton * _skel)

Set the mesh skeleton.

10.136.3.29 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**

10.137 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.

10.137.1 Detailed Description

Creates CSG meshes.

10.137.2 Member Enumeration Documentation

10.137.2.1 enum gazebo::common::MeshCSG::BooleanOperation

An enumeration of the boolean operations.

Enumerator:

UNION

INTERSECTION

DIFFERENCE

10.137.3 Constructor & Destructor Documentation

10.137.3.1 gazebo::common::MeshCSG::MeshCSG ( )

Constructor.

10.137.3.2 virtual gazebo::common::MeshCSG::∼MeshCSG ( ) [virtual]

Destructor.

10.137.4 Member Function Documentation

10.137.4.1 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:

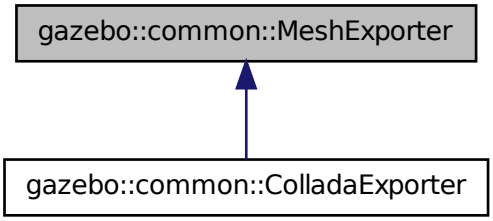
- **MeshCSG.hh**

10.138 gazebo::common::MeshExporter Class Reference

Base class for exporting meshes.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::MeshExporter:



Public Member Functions

- **MeshExporter** ()

  Constructor.
- virtual ∼**MeshExporter** ()

  Destructor.
- virtual void **Export** (const **Mesh** ∗_mesh, const std::string &_filename, bool _exportTextures=false)=0

  Export a mesh to a file.

10.138.1 Detailed Description

Base class for exporting meshes.

10.138.2 Constructor & Destructor Documentation

10.138.2.1 gazebo::common::MeshExporter::MeshExporter ( )

Constructor.

10.138.2.2 virtual gazebo::common::MeshExporter::∼MeshExporter ( ) [virtual]

Destructor.

10.138.3 Member Function Documentation

10.138.3.1 virtual void gazebo::common::MeshExporter::Export (const Mesh ∗ *_mesh,* const std::string & *_filename,* bool *_exportTextures* = false) [pure virtual]

Export a mesh to a file.

Parameters

in	*_mesh*	Pointer to the mesh to be exported
in	*_filename*	Exported file's path and name
in	*_exportTextures*	True to export texture images to '../materials/textures' folder

Implemented in **gazebo::common::ColladaExporter** (p. 289).

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

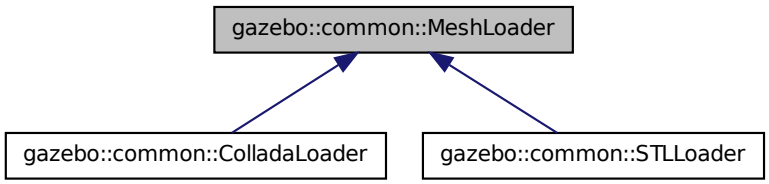
- **MeshExporter.hh**

10.139 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.

10.139.1 Detailed Description

Base class for loading meshes.

10.139.2 Constructor & Destructor Documentation

10.139.2.1 gazebo::common::MeshLoader::MeshLoader ( )

Constructor.

10.139.2.2 virtual gazebo::common::MeshLoader::~MeshLoader ( ) [virtual]

Destructor.

10.139.3 Member Function Documentation

10.139.3.1 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. 292), and **gazebo::common::STLLoader** (p. 1330).

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

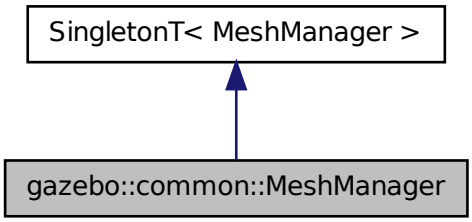
- **MeshLoader.hh**

10.140 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 **Export** (const **Mesh** ∗_mesh, const std::string &_filename, const std::string &_extension, bool _exportTextures=false)

  Export a mesh to a file.
- 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.

10.140.1 Detailed Description

Maintains and manages all meshes.

10.140.2 Member Function Documentation

10.140.2.1 void gazebo::common::MeshManager::AddMesh (Mesh ∗ *_mesh*)

Add a mesh to the manager.

This **MeshManager** (p. 834) takes ownership of the mesh and will destroy it. See ∼-MeshManager.

Parameters

in	*the*	mesh to add.

10.140.2.2 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

10.140.2.3 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

10.140.2.4 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

10.140.2.5 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

10.140.2.6 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

10.140.2.7 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

10.140.2.8 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

10.140.2.9 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	*_inner-Radius*	the inner radius of the tube in the x y plane
in	*_outter-Radius*	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

10.140.2.10 void gazebo::common::MeshManager::Export (const Mesh * _mesh, const std::string & _filename, const std::string & _extension, bool _exportTextures = false)

Export a mesh to a file.

Parameters

in	*_mesh*	Pointer to the mesh to be exported
in	*_filename*	Exported file's path and name
in	*_extension*	Exported file's format ("dae" for Collada)
in	*_export-Textures*	True to export texture images to '../materials/textures' folder

10.140.2.11 void gazebo::common::MeshManager::GenSphericalTexCoord (const Mesh * _mesh, math::Vector3 _center)

generate spherical texture coordinates

10.140.2.12 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

10.140.2.13 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

10.140.2.14 bool gazebo::common::MeshManager::HasMesh (const std::string & *_name*) const

Return true if the mesh exists.

Parameters

in	*_name*	the name of the mesh

10.140.2.15 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

10.140.2.16 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:

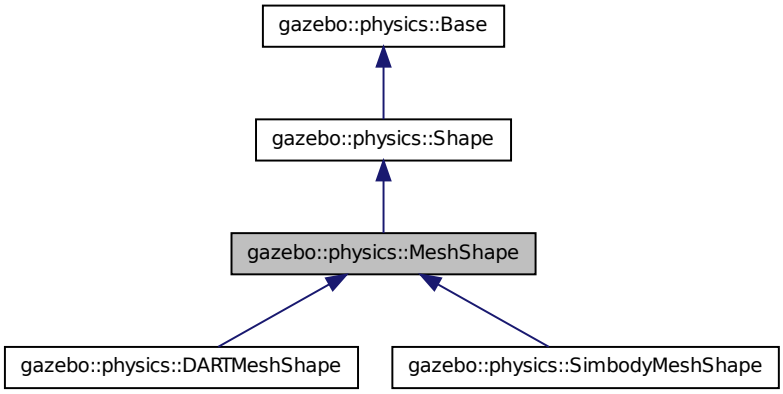
- **MeshManager.hh**

10.141 gazebo::physics::MeshShape Class Reference

Triangle mesh collision shape.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::MeshShape:



Public Member Functions

- **MeshShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**MeshShape** ()

  Destructor.
- void **FillMsg** (msgs::Geometry &_msg)

  Populate a msgs::Geometry message with data from this shape.
- 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** ()
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  Update this shape from a message.
- 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.

10.141.1 Detailed Description

Triangle mesh collision shape.

10.141.2 Constructor & Destructor Documentation

10.141.2.1 gazebo::physics::MeshShape::MeshShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent collision.

10.141.2.2 virtual gazebo::physics::MeshShape::∼MeshShape ( ) [virtual]

Destructor.

10.141.3 Member Function Documentation

10.141.3.1 void gazebo::physics::MeshShape::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. 1163).

10.141.3.2 std::string gazebo::physics::MeshShape::GetMeshURI ( ) const

Get the URI of the mesh data.

Returns

The URI of the mesh data.

10.141.3.3 virtual math::Vector3 gazebo::physics::MeshShape::GetSize ( ) const [virtual]

Get the size of the triangle mesh.

Returns

The size of the triangle mesh.

10.141.3.4 virtual void gazebo::physics::MeshShape::Init ( ) [virtual]

Initialize the shape.

Implements **gazebo::physics::Shape** (p. 1164).

Reimplemented in **gazebo::physics::SimbodyMeshShape** (p. 1221), and **gazebo::physics::DARTMeshShape** (p. 432).

10.141.3.5 virtual void gazebo::physics::MeshShape::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. 1164).

10.141.3.6 void gazebo::physics::MeshShape::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.

10.141.3.7 void gazebo::physics::MeshShape::SetScale (const math::Vector3 & *_scale*) [virtual]

Set the scaling factor.

Parameters

in	*_scale*	Scaling factor.

Implements **gazebo::physics::Shape** (p. 1165).

10.141.3.8 virtual void gazebo::physics::MeshShape::Update ( ) [inline, virtual]

Update the tri mesh.

Reimplemented from **gazebo::physics::Base** (p. 216).

Reimplemented in **gazebo::physics::DARTMeshShape** (p. 433).

10.141.4 Member Data Documentation

10.141.4.1 const common::Mesh* gazebo::physics::MeshShape::mesh [protected]

Pointer to the mesh data.

10.141.4.2 common::SubMesh* gazebo::physics::MeshShape::submesh [protected]

The submesh to use from within the parent mesh.

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

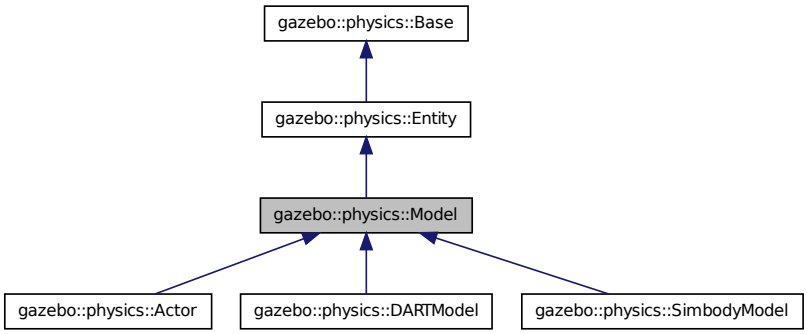
- **MeshShape.hh**

10.142 gazebo::physics::Model Class Reference

A model is a collection of links, joints, and plugins.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Model:



Public Member Functions

- **Model** (**BasePtr** _parent)

  Constructor.
- virtual ~**Model** ()

  Destructor.
- void **AttachStaticModel** (**ModelPtr** &_model, **math::Pose** _offset)

  Attach a static model to this model.
- void **DetachStaticModel** (const std::string &_model)

  Detach a static model from this model.
- virtual 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.
- **GripperPtr GetGripper** (size_t _index) const

Get a gripper based on an index.

- size_t **GetGripperCount** () const

  Get the number of grippers in this model.

- **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

- const **Link_V** & **GetLinks** () const

  *Construct and return a vector of **Link** (p. 739)'s in this model Note this constructs the vector of **Link** (p. 739)'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.

- double **GetWorldEnergy** () const

  *Returns this model's total energy, or sum of **Model::GetWorldEnergyPotential()** (p. 856) and **Model::GetWorldEnergyKinetic()** (p. 856).*

- double **GetWorldEnergyKinetic** () const

Returns sum of the kinetic energies of all links in this model.

- double **GetWorldEnergyPotential** () const

  Returns the potential energy of all links and joint springs in the model.

- 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. 739), 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** > &_anims, boost::function< void()> _onComplete=**NULL**)

  ***Joint** (p. 669) Animation.*

- void **SetJointPosition** (const std::string &_jointName, double _position, int _index=0)

  *Set the positions of a **Joint** (p. 669) 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. 846) by specifying desired Pose of a **Link** (p. 739) within the **Model** (p. 846).*
- void **SetLinkWorldPose** (const **math::Pose** &_pose, const **LinkPtr** &_link)

  *Set the Pose of the entire **Model** (p. 846) by specifying desired Pose of a **Link** (p. 739) within the **Model** (p. 846).*
- void **SetScale** (const **math::Vector3** &_scale)

  Set the scale of model.
- 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. 850)*
- std::vector< **math::Pose** > **attachedModelsOffset**

  *used by **Model::AttachStaticModel** (p. 850)*
- **transport::PublisherPtr jointPub**

  Publisher for joint info.

10.142.1 Detailed Description

A model is a collection of links, joints, and plugins.

10.142.2 Constructor & Destructor Documentation

10.142.2.1 gazebo::physics::Model::Model (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent object.

10.142.2.2 virtual gazebo::physics::Model::~Model ( ) [virtual]

Destructor.

10.142.3 Member Function Documentation

10.142.3.1 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. 846), which is a **Model** (p. 846) that has been marked as static (no physics simulation), and attaches it to this **Model** (p. 846) 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. 846), to place _model.

10.142.3.2 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. 850).

10.142.3.3 virtual void gazebo::physics::Model::FillMsg (msgs::Model & *_msg*) [virtual]

Fill a model message.

Parameters

in	*_msg*	Message to fill using this model's data.

10.142.3.4 virtual void gazebo::physics::Model::Fini ( ) [virtual]

Finalize the model.

Reimplemented from **gazebo::physics::Entity** (p. 504).

Reimplemented in **gazebo::physics::Actor** (p. 169), and **gazebo::physics::DARTModel** (p. 435).

10.142.3.5 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.

10.142.3.6 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. 504).

10.142.3.7 GripperPtr gazebo::physics::Model::GetGripper (size_t *_index*) const

Get a gripper based on an index.

Returns

A pointer to a **Gripper** (p. 609). Null if the _index is invalid.

10.142.3.8 size_t gazebo::physics::Model::GetGripperCount ( ) const

Get the number of grippers in this model.

Returns

Size of this->grippers array.

See also

Model::GetGripper() (p. 852)

10.142.3.9 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

10.142.3.10 JointControllerPtr gazebo::physics::Model::GetJointController ( )

Get a handle to the Controller for the joints in this model.

Returns

A handle to the Controller for the joints in this model.

10.142.3.11 unsigned int gazebo::physics::Model::GetJointCount ( ) const

Get the number of joints.

Returns

Get the number of joints.

10.142.3.12 const Joint_V& gazebo::physics::Model::GetJoints ( ) const

Get the joints.

Returns

Vector of joints.

10.142.3.13 LinkPtr gazebo::physics::Model::GetLink (const std::string & *_name* = "canonical") const

Get a link by name.

Parameters

in	*_name*	Name of the link to get.

Returns

Pointer to the link, NULL if the name is invalid.

10.142.3.14 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.

10.142.3.15 const Link_V& gazebo::physics::Model::GetLinks ( ) const

Construct and return a vector of **Link** (p. 739)'s in this model Note this constructs the vector of **Link** (p. 739)'s on the fly, could be costly.

Returns

a vector of **Link** (p. 739)'s in this model

10.142.3.16 unsigned int gazebo::physics::Model::GetPluginCount ( ) const

Get the number of plugins this model has.

Returns

Number of plugins associated with this model.

10.142.3.17 virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity.

Returns

math::Vector3 (p. 1440), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 506).

10.142.3.18 virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the entity.

Returns

math::Vector3 (p. 1440), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 506).

10.142.3.19 virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity.

Returns

math::Vector3 (p. 1440), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 507).

10.142.3.20 virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the entity.

Returns

math::Vector3 (p. 1440), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 507).

10.142.3.21 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. 210).

Reimplemented in **gazebo::physics::Actor** (p. 169).

10.142.3.22 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.

10.142.3.23 virtual math::Vector3 gazebo::physics::Model::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 1440), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 507).

10.142.3.24 virtual math::Vector3 gazebo::physics::Model::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the entity in the world frame.

Returns

math::Vector3 (p. 1440), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 508).

10.142.3.25 double gazebo::physics::Model::GetWorldEnergy ( ) const

Returns this model's total energy, or sum of **Model::GetWorldEnergyPotential()** (p. 856) and **Model::GetWorldEnergyKinetic()** (p. 856).

Returns

this link's total energy

10.142.3.26 double gazebo::physics::Model::GetWorldEnergyKinetic ( ) const

Returns sum of the kinetic energies of all links in this model.

Computed using link's CoG velocity in the inertial (world) frame.

Returns

this link's kinetic energy

10.142.3.27 double gazebo::physics::Model::GetWorldEnergyPotential ( ) const

Returns the potential energy of all links and joint springs in the model.

Returns

this link's potential energy,

10.142.3.28 virtual math::Vector3 gazebo::physics::Model::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 1440), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 508).

10.142.3.29 virtual math::Vector3 gazebo::physics::Model::GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the entity in the world frame.

Returns

math::Vector3 (p. 1440), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 508).

10.142.3.30 virtual void gazebo::physics::Model::Init ( ) [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Base** (p. 211).

Reimplemented in **gazebo::physics::Actor** (p. 169), **gazebo::physics::DARTModel** (p. 435), and **gazebo::physics::SimbodyModel** (p. 1223).

10.142.3.31 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. 509).

Reimplemented in **gazebo::physics::Actor** (p. 169), **gazebo::physics::DARTModel** (p. 435), and **gazebo::physics::SimbodyModel** (p. 1223).

10.142.3.32 void gazebo::physics::Model::LoadJoints ( )

Load all the joints.

10.142.3.33 void gazebo::physics::Model::LoadPlugins ( )

Load all plugins.

Load all plugins specified in the SDF for the model.

10.142.3.34 virtual void gazebo::physics::Model::OnPoseChange ( ) [protected, virtual]

Callback when the pose of the model has been changed.

Implements **gazebo::physics::Entity** (p. 509).

10.142.3.35 void gazebo::physics::Model::ProcessMsg (const msgs::Model & *_msg*)

Update parameters from a model message.

Parameters

in	*_msg*	Message to process.

10.142.3.36 virtual void gazebo::physics::Model::RemoveChild (EntityPtr *_child*) [virtual]

Remove a child.

Parameters

in	*_child*	Remove a child entity.

10.142.3.37 void gazebo::physics::Model::Reset ( ) [virtual]

Reset the model.

Reimplemented from **gazebo::physics::Entity** (p. 510).

10.142.3.38 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

10.142.3.39 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.

10.142.3.40 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.

10.142.3.41 void gazebo::physics::Model::SetCollideMode (const std::string & *_mode*)

This is not implemented in **Link** (p. 739), which means this function doesn't do anything.

Set the collide mode of the model.

Parameters

in	*_mode*	The collision mode

10.142.3.42 void gazebo::physics::Model::SetEnabled (bool *_enabled*)

Enable all the links in all the models.

Parameters

in	*_enabled*	True to enable all the links.

10.142.3.43 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.

10.142.3.44 void gazebo::physics::Model::SetJointAnimation (const std::map< std::string, common::NumericAnimationPtr > & _anims, boost::function< void() > _onComplete = NULL)

Joint (p. 669) Animation.

Parameters

in	*_anim*	Map of joint names to their position animation.
in	*_onComplete*	Callback function for when the animation completes.

10.142.3.45 void gazebo::physics::Model::SetJointPosition (const std::string & _jointName, double _position, int _index = 0)

Set the positions of a **Joint** (p. 669) by name.

See also

JointController::SetJointPosition (p. 708)

Parameters

in	*_jointName*	Name of the joint to set.
in	*_position*	Position to set the joint to.

10.142.3.46 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. 709).

Parameters

in	*_joint-Positions*	Map of joint names to their positions.

10.142.3.47 void gazebo::physics::Model::SetLaserRetro (const float *_retro*)

Set the laser retro reflectiveness of the model.

Parameters

in	*_retro*	Retro reflectance value.

10.142.3.48 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.

10.142.3.49 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.

10.142.3.50 void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & *_pose,* std::string *_linkName*)

Set the Pose of the entire **Model** (p. 846) by specifying desired Pose of a **Link** (p. 739) within the **Model** (p. 846).

Doing so, keeps the configuration of the **Model** (p. 846) unchanged, i.e. all **Joint** (p. 669) angles are unchanged.

Parameters

in	*_pose*	Pose to set the link to.
in	*_linkName*	Name of the link to set.

10.142.3.51 void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & *_pose*, const LinkPtr & *_link*)

Set the Pose of the entire **Model** (p. 846) by specifying desired Pose of a **Link** (p. 739) within the **Model** (p. 846).

Doing so, keeps the configuration of the **Model** (p. 846) unchanged, i.e. all **Joint** (p. 669) angles are unchanged.

Parameters

in	*_pose*	Pose to set the link to.
in	*_link*	Pointer to the link to set.

10.142.3.52 void gazebo::physics::Model::SetScale (const math::Vector3 & *_scale*)

Set the scale of model.

Parameters

in	*_scale*	Scale to set the model to.

10.142.3.53 void gazebo::physics::Model::SetState (const ModelState & *_state*)

Set the current model state.

Parameters

in	*_state*	**State** (p. 1323) to set the model to.

10.142.3.54 virtual void gazebo::physics::Model::StopAnimation ( ) [virtual]

Stop the current animations.

Reimplemented from **gazebo::physics::Entity** (p. 512).

10.142.3.55 void gazebo::physics::Model::Update ( ) [virtual]

Update the model.

Reimplemented from **gazebo::physics::Base** (p. 216).

Reimplemented in **gazebo::physics::Actor** (p. 170), and **gazebo::physics::DARTModel** (p. 436).

10.142.3.56 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. 512).

Reimplemented in **gazebo::physics::Actor** (p. 170).

10.142.4 Member Data Documentation

10.142.4.1 std::vector<ModelPtr> gazebo::physics::Model::attachedModels [protected]

used by **Model::AttachStaticModel** (p. 850)

10.142.4.2 std::vector<math::Pose> gazebo::physics::Model::attachedModelsOffset [protected]

used by **Model::AttachStaticModel** (p. 850)

10.142.4.3 transport::PublisherPtr gazebo::physics::Model::jointPub [protected]

Publisher for joint info.

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

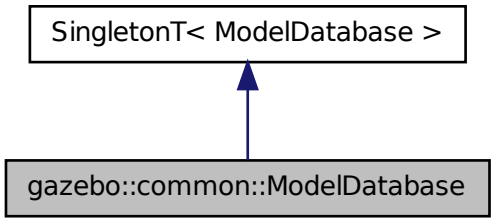
- **Model.hh**

10.143 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.
- void **Fini** ()

  Finalize the model database.
- std::string **GetDBConfig** (const std::string &_uri)

  Return the database.config file as a string.
- 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.
- **event::ConnectionPtr 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.

- void **Start** (bool _fetchImmediately=false)

  Start the model database.

10.143.1 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**

10.144 gazebo::common::ModelDatabasePrivate Class Reference

Private class attributes for **ModelDatabase** (p. 864).

`#include <ModelDatabasePrivate.hh>`

Public Types

- typedef boost::function< void(const std::map< std::string, std::string > &)> **CallbackFunc**

Public Attributes

- boost::mutex **callbacksMutex**

  Protects callback list.

- std::map< std::string, std::string > **modelCache**

  A dictionary of all model names indexed by their uri.

- **event::EventT**< void(std::map < std::string, std::string >)> **modelDBUpdated**

  *Triggered when the model data has been updated after calling **ModelDatabase::GetModels()** (p. 51)*

- boost::recursive_mutex **startCacheMutex**

  Mutex to protect cache thread status checks.

- bool **stop**

True to stop the background thread.

- boost::condition_variable **updateCacheCompleteCondition**

  Condition variable for completion of one cache update.

- boost::condition_variable **updateCacheCondition**

  Condition variable for the updateCacheThread.

- boost::thread ∗ **updateCacheThread**

  Thread to update the model cache.

- boost::mutex **updateMutex**

  Cache update mutex.

10.144.1 Detailed Description

Private class attributes for **ModelDatabase** (p. 864).

10.144.2 Member Typedef Documentation

10.144.2.1 typedef boost::function< void (const std::map<std::string, std::string> &)> gazebo::common::ModelDatabasePrivate::CallbackFunc

10.144.3 Member Data Documentation

10.144.3.1 boost::mutex gazebo::common::ModelDatabasePrivate::callbacksMutex

Protects callback list.

10.144.3.2 std::map<std::string, std::string> gazebo::common::ModelDatabasePrivate::modelCache

A dictionary of all model names indexed by their uri.

10.144.3.3 event::EventT< void (std::map<std::string, std::string>)> gazebo::common::ModelDatabasePrivate::modelDBUpdated

Triggered when the model data has been updated after calling **ModelDatabase::GetModels()** (p. 51)

10.144.3.4 boost::recursive_mutex gazebo::common::ModelDatabasePrivate::startCacheMutex

Mutex to protect cache thread status checks.

10.144.3.5 bool gazebo::common::ModelDatabasePrivate::stop

True to stop the background thread.

10.144.3.6 boost::condition_variable gazebo::common::ModelDatabasePrivate::updateCacheCompleteCondition

Condition variable for completion of one cache update.

10.144.3.7 boost::condition_variable gazebo::common::ModelDatabasePrivate::updateCacheCondition

Condition variable for the updateCacheThread.

10.144.3.8 boost::thread* gazebo::common::ModelDatabasePrivate::updateCacheThread

Thread to update the model cache.

10.144.3.9 boost::mutex gazebo::common::ModelDatabasePrivate::updateMutex

Cache update mutex.

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

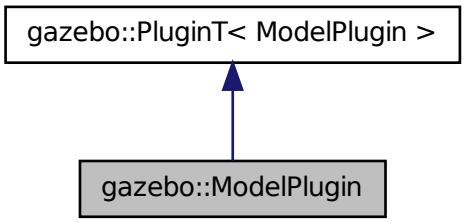
- **ModelDatabasePrivate.hh**

10.145 gazebo::ModelPlugin Class Reference

A plugin with access to **physics::Model** (p. 846).

```
#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.

10.145.1 Detailed Description

A plugin with access to **physics::Model** (p. 846).

See `reference`.

10.145.2 Constructor & Destructor Documentation

10.145.2.1 gazebo::ModelPlugin::ModelPlugin ( ) [inline]

Constructor.

References gazebo::MODEL_PLUGIN.

10.145.2.2 virtual gazebo::ModelPlugin::~ModelPlugin ( ) [inline, virtual]

Destructor.

10.145.3 Member Function Documentation

10.145.3.1 virtual void gazebo::ModelPlugin::Init ( ) [inline, virtual]

Override this method for custom plugin initialization behavior.

10.145.3.2 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.

10.145.3.3 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:

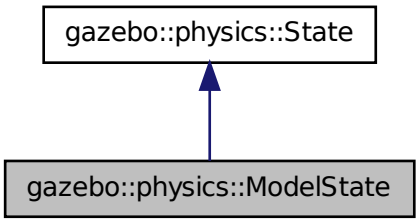
- **Plugin.hh**

10.146 gazebo::physics::ModelState Class Reference

Store state information of a **physics::Model** (p. 846) object.

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

Inheritance diagram for gazebo::physics::ModelState:



Public Member Functions

- **ModelState** ()

  Default constructor.
- **ModelState** (const **ModelPtr** _model, const **common::Time** &_realTime, const **common::Time** &_simTime)

  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. 669) state.*
- **JointState GetJointState** (const std::string &_jointName) const

  *Get a **Joint** (p. 669) state by **Joint** (p. 669) name.*
- unsigned int **GetJointStateCount** () const

  Get the number of joint states.
- **JointState_M GetJointStates** (const boost::regex &_regex) const

  Get joint states based on a regular expression.
- const **JointState_M** & **GetJointStates** () const

Get the joint states.

- **LinkState GetLinkState** (const std::string &_linkName) const

  *Get a link state by **Link** (p. 739) name.*
- unsigned int **GetLinkStateCount** () const

  Get the number of link states.
- **LinkState_M GetLinkStates** (const boost::regex &_regex) const

  Get link states based on a regular expression.
- const **LinkState_M** & **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.
- void **Load** (const **ModelPtr** _model, const **common::Time** &_realTime, const **common::Time** &_simTime)

  *Load state from **Model** (p. 846) pointer.*
- 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.
- virtual void **SetRealTime** (const **common::Time** &_time)

  Set the real time when this state was generated.
- virtual void **SetSimTime** (const **common::Time** &_time)

  Set the sim time when this state was generated.
- virtual void **SetWallTime** (const **common::Time** &_time)

  Set the wall time when this state was generated.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::-ModelState** &_state)

  Stream insertion operator.

10.146.1 Detailed Description

Store state information of a **physics::Model** (p. 846) object.

This class captures the entire state of a **Model** (p. 846) at one specific time during a simulation run.

State (p. 1323) of a **Model** (p. 846) includes the state of all its child Links and Joints.

10.146.2 Constructor & Destructor Documentation

10.146.2.1 gazebo::physics::ModelState::ModelState ( )

Default constructor.

10.146.2.2 gazebo::physics::ModelState::ModelState (const ModelPtr *_model,* const common::Time & *_realTime,* const common::Time & *_simTime*)

Constructor.

Build a **ModelState** (p. 869) from an existing **Model** (p. 846).

Parameters

in	*_model*	Pointer to the model from which to gather state info.
in	*_realTime*	Real time stamp.
in	*_simTime*	Sim time stamp.

10.146.2.3 gazebo::physics::ModelState::ModelState (const ModelPtr *_model*) [explicit]

Constructor.

Build a **ModelState** (p. 869) from an existing **Model** (p. 846).

Parameters

in	*_model*	Pointer to the model from which to gather state info.

10.146.2.4 gazebo::physics::ModelState::ModelState (const sdf::ElementPtr *_sdf*) [explicit]

Constructor.

Build a **ModelState** (p. 869) from SDF data

Parameters

in	*_sdf*	SDF data to load a model state from.

10.146.2.5 virtual gazebo::physics::ModelState::~ModelState ( ) [virtual]

Destructor.

10.146.3 Member Function Documentation

10.146.3.1 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.

10.146.3.2 JointState gazebo::physics::ModelState::GetJointState (unsigned int *_index*) const

Get a **Joint** (p. 669) state.

Return a **JointState** (p. 712) based on a index, where index is between 0...**ModelState::GetJointStateCount()** (p. 874).

Parameters

in	*_index*	Index of a **JointState** (p. 712).

Returns

State (p. 1323) of a **Joint** (p. 669).

Exceptions

common::Exception *(p. 548)*	When _index is out of range.

10.146.3.3 JointState gazebo::physics::ModelState::GetJointState (const std::string & _jointName) const

Get a **Joint** (p. 669) state by **Joint** (p. 669) name.

Searches through all JointStates. Returns the **JointState** (p. 712) with the matching name, if any.

Parameters

in	*_jointName*	Name of the **JointState** (p. 712).

Returns

State (p. 1323) of the **Joint** (p. 669).

Exceptions

common::-Exception *(p. 548)*	When _jointName is invalid.

10.146.3.4 unsigned int gazebo::physics::ModelState::GetJointStateCount ( ) const

Get the number of joint states.

Returns the number of JointStates recorded.

Returns

Number of JointStates.

10.146.3.5 JointState_M gazebo::physics::ModelState::GetJointStates (const boost::regex & _regex) const

Get joint states based on a regular expression.

Parameters

in	*_regex*	The regular expression.

Returns

List of joint states whose names match the regular expression.

10.146.3.6 const JointState_M& gazebo::physics::ModelState::GetJointStates ( ) const

Get the joint states.

Returns

A map of joint states.

10.146.3.7 LinkState gazebo::physics::ModelState::GetLinkState (const std::string & *_linkName*) const

Get a link state by **Link** (p. 739) name.

Searches through all LinkStates. Returns the **LinkState** (p. 769) with the matching name, if any.

Parameters

in	*_linkName*	Name of the **LinkState** (p. 769)

Returns

State (p. 1323) of the **Link** (p. 739).

Exceptions

common::-Exception *(p. 548)*	When _linkName is invalid.

10.146.3.8 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. 769) recorded.

10.146.3.9 LinkState_M gazebo::physics::ModelState::GetLinkStates (const boost::regex & _regex) const

Get link states based on a regular expression.

Parameters

in	*_regex*	The regular expression.

Returns

List of link states whose names match the regular expression.

10.146.3.10 const LinkState_M& gazebo::physics::ModelState::GetLinkStates ( ) const

Get the link states.

Returns

A map of link states.

10.146.3.11 const math::Pose& gazebo::physics::ModelState::GetPose ( ) const

Get the stored model pose.

Returns

The **math::Pose** (p. 995) of the **Model** (p. 846).

10.146.3.12 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.

10.146.3.13 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. 769).

Returns

True if the link exists in the model.

10.146.3.14 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.

10.146.3.15 void gazebo::physics::ModelState::Load (const ModelPtr _model, const common::Time & _realTime, const common::Time & _simTime)

Load state from **Model** (p. 846) pointer.

Build a **ModelState** (p. 869) from an existing **Model** (p. 846).

Parameters

in	*_model*	Pointer to the model from which to gather state info.
in	*_realTime*	Real time stamp.
in	*_simTime*	Sim time stamp.

10.146.3.16 virtual void gazebo::physics::ModelState::Load (const sdf::ElementPtr _elem) [virtual]

Load state from SDF element.

Load **ModelState** (p. 869) 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. 1326).

10.146.3.17 ModelState gazebo::physics::ModelState::operator+ (const ModelState & _state) const

Addition operator.

Parameters

in	*_pt*	A state to substract.

Returns

The resulting state.

10.146.3.18 ModelState gazebo::physics::ModelState::operator- (const ModelState & _state) const

Subtraction operator.

Parameters

in	*_pt*	A state to substract.

Returns

The resulting state.

10.146.3.19 ModelState& gazebo::physics::ModelState::operator= (const ModelState & _state)

Assignment operator.

Parameters

in	*_state*	**State** (p. 1323) value

Returns

this

10.146.3.20 virtual void gazebo::physics::ModelState::SetRealTime (const common::Time & *_time*) [virtual]

Set the real time when this state was generated.

Parameters

in	*_time*	Clock time since simulation was stated.

Reimplemented from **gazebo::physics::State** (p. 1327).

10.146.3.21 virtual void gazebo::physics::ModelState::SetSimTime (const common::Time & *_time*) [virtual]

Set the sim time when this state was generated.

Parameters

in	*_time*	Simulation time when the data was recorded.

Reimplemented from **gazebo::physics::State** (p. 1327).

10.146.3.22 virtual void gazebo::physics::ModelState::SetWallTime (const common::Time & *_time*) [virtual]

Set the wall time when this state was generated.

Parameters

in	*_time*	The absolute clock time when the **State** (p. 1323) data was recorded.

Reimplemented from **gazebo::physics::State** (p. 1328).

10.146.4 Friends And Related Function Documentation

10.146.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::physics::ModelState & *_state*) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream.
in	*_state*	**Model** (p. 846) state to output.

Returns

The stream.

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

- **ModelState.hh**

10.147 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.

10.147.1 Detailed Description

Generic description of a mouse event.

10.147.2 Member Enumeration Documentation

10.147.2.1 enum gazebo::common::MouseEvent::Buttons

Standard mouse buttons enumeration.

Enumerator:

NO_BUTTON

LEFT

MIDDLE

RIGHT

10.147.2.2 enum gazebo::common::MouseEvent::EventType

Mouse event types enumeration.

Enumerator:

NO_EVENT

MOVE

PRESS

RELEASE

SCROLL

10.147.3 Constructor & Destructor Documentation

10.147.3.1 gazebo::common::MouseEvent::MouseEvent ( ) [inline]

Constructor.

10.147.4 Member Data Documentation

10.147.4.1 bool gazebo::common::MouseEvent::alt

Alt key press flag.

10.147.4.2 unsigned int gazebo::common::MouseEvent::button

The button which caused the event.

10.147.4.3 unsigned int gazebo::common::MouseEvent::buttons

State of the buttons when the event was generated.

10.147.4.4 bool gazebo::common::MouseEvent::control

Control key press flag.

10.147.4.5 bool gazebo::common::MouseEvent::dragging

Flag for mouse drag motion.

10.147.4.6 float gazebo::common::MouseEvent::moveScale

Scaling factor.

10.147.4.7 math::Vector2i gazebo::common::MouseEvent::pos

Mouse pointer position on the screen.

10.147.4.8 math::Vector2i gazebo::common::MouseEvent::pressPos

Position of button press.

10.147.4.9 math::Vector2i gazebo::common::MouseEvent::prevPos

Previous position.

10.147.4.10 math::Vector2i gazebo::common::MouseEvent::scroll

Scroll position.

10.147.4.11 bool gazebo::common::MouseEvent::shift

Shift key press flag.

10.147.4.12 EventType gazebo::common::MouseEvent::type

Event type.

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

- **MouseEvent.hh**

10.148 gazebo::rendering::MovableText Class Reference

Movable text.

`#include <rendering/rendering.hh>`

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

10.148.1 Detailed Description

Movable text.

10.148.2 Member Enumeration Documentation

10.148.2.1 enum gazebo::rendering::MovableText::HorizAlign

Horizontal alignment.

Enumerator:

H_LEFT Left alignment.

H_CENTER Center alignment.

10.148.2.2 enum gazebo::rendering::MovableText::VertAlign

vertical alignment

Enumerator:

V_BELOW Align below.

V_ABOVE Align above.

10.148.3 Constructor & Destructor Documentation

10.148.3.1 gazebo::rendering::MovableText::MovableText ( )

Constructor.

10.148.3.2 virtual gazebo::rendering::MovableText::~MovableText ( ) [virtual]

Destructor.

10.148.4 Member Function Documentation

10.148.4.1 void gazebo::rendering::MovableText::_setupGeometry ( ) [protected]

10.148.4.2 void gazebo::rendering::MovableText::_updateColors ( ) [protected]

10.148.4.3 math::Box gazebo::rendering::MovableText::GetAABB ( )

Get the axis aligned bounding box of the text.

Returns

The axis aligned bounding box.

10.148.4.4 float gazebo::rendering::MovableText::GetBaseline ( ) const

Get the baseline height.

Returns

Baseline height

10.148.4.5 float gazebo::rendering::MovableText::getBoundingRadius ( ) const [protected]

10.148.4.6 float gazebo::rendering::MovableText::GetCharHeight ( ) const

Set the height of a characters return Height of the characters.

10.148.4.7 const common::Color& gazebo::rendering::MovableText::GetColor ( ) const

Get the text color.

Returns

Texture color.

10.148.4.8 const std::string& gazebo::rendering::MovableText::GetFont ( ) const

Get the font.

Returns

The font name

10.148.4.9 const Ogre::LightList& gazebo::rendering::MovableText::getLights (void) const [protected]

10.148.4.10 const Ogre::MaterialPtr& gazebo::rendering::MovableText::getMaterial (void) const [protected]

10.148.4.11 void gazebo::rendering::MovableText::getRenderOperation (Ogre::RenderOperation & *op*) [protected]

10.148.4.12 bool gazebo::rendering::MovableText::GetShowOnTop ( ) const

True = text is displayed on top.

Returns

True if MovableText::SetShownOnTop(true) was called.

10.148.4.13 float gazebo::rendering::MovableText::GetSpaceWidth ( ) const

Get the width of a space.

Returns

Space width

10.148.4.14 float gazebo::rendering::MovableText::getSquaredViewDepth (const Ogre::Camera * *cam*) const [protected]

10.148.4.15 const std::string& gazebo::rendering::MovableText::GetText ( ) const

Get the displayed text.

Returns

The displayed text.

10.148.4.16 void gazebo::rendering::MovableText::getWorldTransforms (Ogre::Matrix4 * *xform*) const [protected]

10.148.4.17 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

10.148.4.18 void gazebo::rendering::MovableText::SetBaseline (float *_height*)

Set the baseline height of the text.

Parameters

in	*_height*	Baseline height

10.148.4.19 void gazebo::rendering::MovableText::SetCharHeight (float *_height*)

Set the height of a character.

Parameters

in	*_height*	Height of the characters.

10.148.4.20 void gazebo::rendering::MovableText::SetColor (const common::Color & *_color*)

Set the text color.

Parameters

in	*_color*	Text color.

10.148.4.21 void gazebo::rendering::MovableText::SetFontName (const std::string & *_font*)

Set the font.

Parameters

in	*_font*	Name of the font

10.148.4.22 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.

10.148.4.23 void gazebo::rendering::MovableText::SetSpaceWidth (float *_width*)

Set the width of a space.

Parameters

in	*_width*	space width

10.148.4.24 void gazebo::rendering::MovableText::SetText (const std::string & *_text*)

Set the text to display.

Parameters

in	*_text*	The text to display.

10.148.4.25 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

10.148.4.26 void gazebo::rendering::MovableText::Update ( )

Update the text.

10.148.4.27 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**

10.149 gazebo::common::MovingWindowFilter< T > Class - Template Reference

Base class for **MovingWindowFilter** (p. 890).

`#include <common/common.hh>`

Public Member Functions

- **MovingWindowFilter** ()
  Constructor.
- virtual ~**MovingWindowFilter** ()
  Destructor.
- T **Get** ()
  Get filtered result.
- bool **GetWindowFilled** () const
  Get whether the window has been filled.
- unsigned int **GetWindowSize** () const
  Get the window size.
- void **SetWindowSize** (unsigned int _n)
  Set window size.
- void **Update** (T _val)
  Update value of filter.

Protected Member Functions

- **MovingWindowFilter** (**MovingWindowFilterPrivate**< T > &_d)
  Allow subclasses to initialize their own data pointer.

Protected Attributes

- **MovingWindowFilterPrivate**< T > * **dataPtr**
  Data pointer.

10.149.1 Detailed Description

template<typename T>class gazebo::common::MovingWindowFilter< T >

Base class for **MovingWindowFilter** (p. 890).

10.149.2 Constructor & Destructor Documentation

10.149.2.1 template<typename T > gazebo::common::MovingWindowFilter< T >::MovingWindowFilter (MovingWindowFilterPrivate< T > & _d) [protected]

Allow subclasses to initialize their own data pointer.

Parameters

in	*_d*	Reference to data pointer.

10.149.3 Member Data Documentation

10.149.3.1 template<typename T > MovingWindowFilterPrivate<T>* gazebo::common::MovingWindowFilter< T >::dataPtr [protected]

Data pointer.

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

- **MovingWindowFilter.hh**

10.150 gazebo::common::MovingWindowFilterPrivate< T > Class Template Reference

`#include <MovingWindowFilter.hh>`

Public Member Functions

- **MovingWindowFilterPrivate** ()

Public Attributes

- unsigned int **samples**

  keep track of number of elements
- T **sum**

  keep track of running sum
- std::vector< T > **valHistory**

  buffer history of raw values
- std::vector< T >::iterator **valIter**

  iterator pointing to current value in buffer
- unsigned int **valWindowSize**

  For moving window smoothed value.

template<typename T> class gazebo::common::MovingWindowFilterPrivate< T >

10.150.1 Member Data Documentation

10.150.1.1 template<typename T > unsigned int gazebo::common::MovingWindowFilterPrivate< T >::samples

keep track of number of elements

10.150.1.2 template<typename T > T gazebo::common::MovingWindowFilterPrivate< T >::sum

keep track of running sum

10.150.1.3 template<typename T > std::vector<T> gazebo::common::MovingWindowFilterPrivate< T >::valHistory

buffer history of raw values

10.150.1.4 template<typename T > std::vector<T>::iterator gazebo::common::MovingWindowFilterPrivate< T >::valIter

iterator pointing to current value in buffer

10.150.1.5 template<typename T > unsigned int gazebo::common::MovingWindowFilterPrivate< T >::valWindowSize

For moving window smoothed value.

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

- **MovingWindowFilter.hh**

10.151 gazebo::msgs::MsgFactory Class Reference

A 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 boost::shared_ptr < 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.

10.151.1 Detailed Description

A factory that generates protobuf message based on a string type.

10.151.2 Member Function Documentation

10.151.2.1 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.

10.151.2.2 static boost::shared_ptr<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.

10.151.2.3 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:

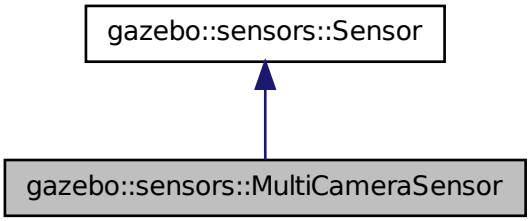
- **MsgFactory.hh**

10.152 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. 242).*
- 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 bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

10.152.1 Detailed Description

Multiple camera sensor.

This sensor type can create one or more synchronized cameras.

10.152.2 Constructor & Destructor Documentation

10.152.2.1 gazebo::sensors::MultiCameraSensor::MultiCameraSensor ( )

Constructor.

10.152.2.2 virtual gazebo::sensors::MultiCameraSensor::~MultiCameraSensor ( ) [virtual]

Destructor.

10.152.3 Member Function Documentation

10.152.3.1 virtual void gazebo::sensors::MultiCameraSensor::Fini ( ) [protected, virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.152.3.2 rendering::CameraPtr gazebo::sensors::MultiCameraSensor::GetCamera (unsigned int *_index*) const

Returns a pointer to a **rendering::Camera** (p. 242).

Parameters

in	*_index*	Index of the camera to get

Returns

The Pointer to the camera sensor.

See also

MultiCameraSensor::GetCameraCount (p. 897)

10.152.3.3 unsigned int gazebo::sensors::MultiCameraSensor::GetCameraCount ( ) const

Get the number of cameras.

Returns

The number of cameras.

10.152.3.4 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. 897)

10.152.3.5 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. 897)

10.152.3.6 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. 897)

10.152.3.7 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. 1138).

10.152.3.8 virtual void gazebo::sensors::MultiCameraSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.152.3.9 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. 1139).

10.152.3.10 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. 1140).

10.152.3.11 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. 897)

10.152.3.12 virtual bool gazebo::sensors::MultiCameraSensor::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

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

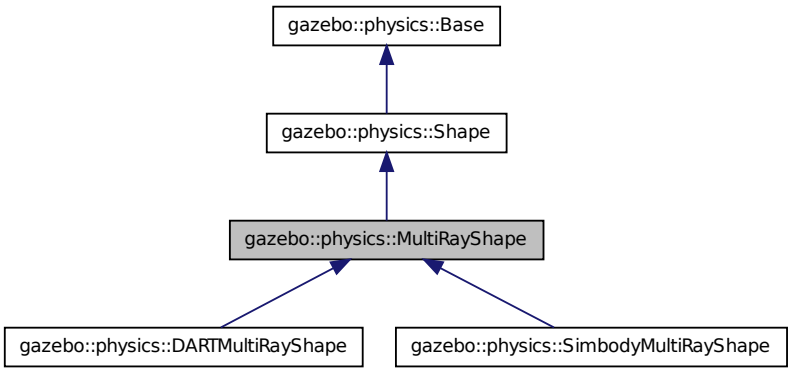
- **MultiCameraSensor.hh**

10.153 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** (unsigned 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** (unsigned int _index)

  Get detected range for a ray.

- double **GetResRange** () const

  Get the range resolution.

- double **GetRetro** (unsigned 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.

- virtual void **SetScale** (const **math::Vector3** &_scale)

  Set the scale of the multi ray shape.

- 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.

10.153.1 Detailed Description

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

10.153.2 Constructor & Destructor Documentation

10.153.2.1 gazebo::physics::MultiRayShape::MultiRayShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent collision shape.

10.153.2.2 virtual gazebo::physics::MultiRayShape::~MultiRayShape ( ) [virtual]

Destructor.

10.153.3 Member Function Documentation

10.153.3.1 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.

Reimplemented in **gazebo::physics::DARTMultiRayShape** (p. 97), and **gazebo::physics::SimbodyMultiRayShape** (p. 1225).

10.153.3.2 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.

10.153.3.3 void gazebo::physics::MultiRayShape::DisconnectNewLaserScans (event::ConnectionPtr & *_conn*) [inline]

Disconnect from the new laser scans signal.

Parameters

in	*_conn*	Connection to remove.

10.153.3.4 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. 1163).

10.153.3.5 int gazebo::physics::MultiRayShape::GetFiducial (unsigned int *_index*)

Get detected fiducial value for a ray.

Parameters

in	*_index*	Index of the ray.

Returns

Fiducial value for the ray.

10.153.3.6 math::Angle gazebo::physics::MultiRayShape::GetMaxAngle ( ) const

Get the maximum angle.

Returns

Maximum angle of ray scan.

10.153.3.7 double gazebo::physics::MultiRayShape::GetMaxRange ( ) const

Get the maximum range.

Returns

Maximum range of all the rays.

10.153.3.8 math::Angle gazebo::physics::MultiRayShape::GetMinAngle ( ) const

Get the minimum angle.

Returns

Minimum angle of ray scan.

10.153.3.9 double gazebo::physics::MultiRayShape::GetMinRange ( ) const

Get the minimum range.

Returns

Minimum range of all the rays.

10.153.3.10 double gazebo::physics::MultiRayShape::GetRange (unsigned int *_index*)

Get detected range for a ray.

Parameters

in	*_index*	Index of the ray.

Returns

Returns DBL_MAX for no detection.

10.153.3.11 double gazebo::physics::MultiRayShape::GetResRange ( ) const

Get the range resolution.

Returns

Range resolution of all the rays.

10.153.3.12 double gazebo::physics::MultiRayShape::GetRetro (unsigned int *_index*)

Get detected retro (intensity) value for a ray.

Parameters

in	*_index*	Index of the ray.

Returns

Retro value for the ray.

10.153.3.13 int gazebo::physics::MultiRayShape::GetSampleCount ( ) const

Get the horizontal sample count.

Returns

Horizontal sample count.

10.153.3.14 double gazebo::physics::MultiRayShape::GetScanResolution ( ) const

Get the horizontal resolution.

Returns

Horizontal resolution.

10.153.3.15 math::Angle gazebo::physics::MultiRayShape::GetVerticalMaxAngle ( ) const

Get the vertical max angle.

Returns

Vertical max angle.

10.153.3.16 math::Angle gazebo::physics::MultiRayShape::GetVerticalMinAngle ( ) const

Get the vertical min angle.

Returns

Vertical min angle.

10.153.3.17 int gazebo::physics::MultiRayShape::GetVerticalSampleCount ( ) const

Get the vertical sample count.

Returns

Verical sample count.

10.153.3.18 double gazebo::physics::MultiRayShape::GetVerticalScanResolution ( ) const

Get the vertical range resolution.

Returns

Vertical range resolution.

10.153.3.19 virtual void gazebo::physics::MultiRayShape::Init ( ) [virtual]

Init the shape.

Implements **gazebo::physics::Shape** (p. 1164).

10.153.3.20 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. 1164).

10.153.3.21 virtual void gazebo::physics::MultiRayShape::SetScale (const math::Vector3 & *_scale*) [virtual]

Set the scale of the multi ray shape.

Returns

_scale Scale to set the multi ray shape to.

Implements **gazebo::physics::Shape** (p. 1165).

10.153.3.22 void gazebo::physics::MultiRayShape::Update ( ) [virtual]

Update the ray collisions.

Reimplemented from **gazebo::physics::Base** (p. 216).

10.153.3.23 virtual void gazebo::physics::MultiRayShape::UpdateRays ( ) [protected, pure virtual]

Physics engine specific method for updating the rays.

Implemented in **gazebo::physics::DARTMultiRayShape** (p. 97), and **gazebo::physics::SimbodyMultiRayShape** (p. 1225).

10.153.4 Member Data Documentation

10.153.4.1 sdf::ElementPtr gazebo::physics::MultiRayShape::horzElem [protected]

Horizontal SDF element pointer.

10.153.4.2 event::EventT<void()> gazebo::physics::MultiRayShape::newLaserScans [protected]

New laser scans event.

10.153.4.3 math::Pose gazebo::physics::MultiRayShape::offset [protected]

Pose offset of all the rays.

10.153.4.4 sdf::ElementPtr gazebo::physics::MultiRayShape::rangeElem [protected]

Range SDF element pointer.

10.153.4.5 sdf::ElementPtr gazebo::physics::MultiRayShape::rayElem [protected]

Ray SDF element pointer.

10.153.4.6 std::vector<RayShapePtr> gazebo::physics::MultiRayShape::rays [protected]

Ray data.

10.153.4.7 sdf::ElementPtr gazebo::physics::MultiRayShape::scanElem [protected]

Scan SDF element pointer.

10.153.4.8 sdf::ElementPtr gazebo::physics::MultiRayShape::vertElem [protected]

Vertical SDF element pointer.

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

- **MultiRayShape.hh**

10.154 gazebo::transport::Node Class Reference

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

`#include <transport/transport.hh>`

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 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 boost::shared_ptr< 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 boost::shared_ptr< 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.

10.154.1 Detailed Description

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

10.154.2 Constructor & Destructor Documentation

10.154.2.1 gazebo::transport::Node::Node ( )

Constructor.

10.154.2.2 virtual gazebo::transport::Node::~Node ( ) [virtual]

Destructor.

10.154.3 Member Function Documentation

10.154.3.1 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 SingletonT< T >::Instance().

10.154.3.2 std::string gazebo::transport::Node::DecodeTopicName (const std::string & _topic)

Decode a topic name.

Parameters

in	*The*	encoded name

Returns

The decoded name

10.154.3.3 std::string gazebo::transport::Node::EncodeTopicName (const std::string & _topic)

Encode a topic name.

Parameters

in	*The*	decoded name

Returns

The encoded name

10.154.3.4 void gazebo::transport::Node::Fini ( )

Finalize the node.

10.154.3.5 unsigned int gazebo::transport::Node::GetId ( ) const

Get the unique ID of the node.

Returns

The unique ID of the node

10.154.3.6 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

10.154.3.7 std::string gazebo::transport::Node::GetTopicNamespace ( ) const

Get the topic namespace for this node.

Returns

The namespace

10.154.3.8 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

10.154.3.9 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

10.154.3.10 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.

10.154.3.11 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. 792)

10.154.3.12 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.

10.154.3.13 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.

10.154.3.14 void gazebo::transport::Node::ProcessIncoming ( )

Process incoming messages.

10.154.3.15 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

10.154.3.16 template<typename M > void gazebo::transport::Node::Publish (const std::string & _topic, const google::protobuf::Message & _message) [inline]

A convenience function for a one-time publication of a message.

This is inefficient, compared to **Node::Advertise** (p. 912) followed by **Publisher::-Publish** (p. 1026). 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

10.154.3.17 void gazebo::transport::Node::RemoveCallback (const std::string & _topic, unsigned int _id)

10.154.3.18 template<typename M , typename T > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & _topic, void(T::*)(const boost::shared_ptr< 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. 1346) object

References SingletonT< T >::Instance().

10.154.3.19 template<typename M > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & _topic, void(*)(const boost::shared_ptr< 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. 1346) object

References SingletonT< T >::Instance().

10.154.3.20 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. 1346) object

References gazebo::transport::SubscribeOptions::Init(), and SingletonT< T >::Instance().

10.154.3.21 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. 1346) object

References gazebo::transport::SubscribeOptions::Init(), and SingletonT< T >::Instance().

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

- **Node.hh**

10.155 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

10.155.1 Detailed Description

Node animation.

10.155.2 Constructor & Destructor Documentation

10.155.2.1 gazebo::common::NodeAnimation::NodeAnimation (const std::string & *_name*)

constructor

Parameters

in	*_name*	the name of the node

10.155.2.2 gazebo::common::NodeAnimation::~NodeAnimation ( )

Destructor. It empties the key frames list.

10.155.3 Member Function Documentation

10.155.3.1 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

10.155.3.2 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

10.155.3.3 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)

10.155.3.4 unsigned int gazebo::common::NodeAnimation::GetFrameCount ( ) const

Returns the number of key frames.

Returns

the count

10.155.3.5 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

10.155.3.6 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. 1361) is -1 if the index is out of bounds

10.155.3.7 double gazebo::common::NodeAnimation::GetLength ( ) const

Returns the duration of the animations.

Returns

the time of the last animation

10.155.3.8 std::string gazebo::common::NodeAnimation::GetName ( ) const

Returns the name.

Returns

the name

10.155.3.9 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.

10.155.3.10 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

10.155.3.11 void gazebo::common::NodeAnimation::SetName (const std::string & *_name*)

Changes the name of the animation.

Parameters

in	*the*	new name

10.155.4 Member Data Documentation

10.155.4.1 std::map<double, math::Matrix4> gazebo::common::NodeAnimation::keyFrames [protected]

the dictionary of key frames, indexed by time

10.155.4.2 double gazebo::common::NodeAnimation::length [protected]

the duration of the animations (time of last key frame)

10.155.4.3 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**

10.156 gazebo::common::NodeAssignment Class Reference

Vertex to node weighted assignement for skeleton animation visualization.

`#include <Mesh.hh>`

Public Member Functions

- **NodeAssignment** ()

Constructor.

Public Attributes

- unsigned int **nodeIndex**

  node (or bone) index
- unsigned int **vertexIndex**

  index of the vertex
- float **weight**

  the weight (between 0 and 1)

10.156.1 Detailed Description

Vertex to node weighted assignement for skeleton animation visualization.

10.156.2 Constructor & Destructor Documentation

10.156.2.1 gazebo::common::NodeAssignment::NodeAssignment ( )

Constructor.

10.156.3 Member Data Documentation

10.156.3.1 unsigned int gazebo::common::NodeAssignment::nodeIndex

node (or bone) index

10.156.3.2 unsigned int gazebo::common::NodeAssignment::vertexIndex

index of the vertex

10.156.3.3 float gazebo::common::NodeAssignment::weight

the weight (between 0 and 1)

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

- **Mesh.hh**

10.157 gazebo::common::NodeTransform Class Reference

NodeTransform (p. 925) **Skeleton.hh** (p. 1787) 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

10.157.1 Detailed Description

NodeTransform (p. 925) **Skeleton.hh** (p. 1787) common/common.hh

A transformation node

10.157.2 Member Enumeration Documentation

10.157.2.1 enum gazebo::common::NodeTransform::TransformType

Enumeration of the transform types.

Enumerator:

TRANSLATE

ROTATE

SCALE

MATRIX

10.157.3 Constructor & Destructor Documentation

10.157.3.1 gazebo::common::NodeTransform::NodeTransform (TransformType _type = MATRIX)

Constructor.

Parameters

in	*_type*	the type of transform

10.157.3.2 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

10.157.3.3 gazebo::common::NodeTransform::~NodeTransform ( )

Destructor. It does nothing.

10.157.4 Member Function Documentation

10.157.4.1 math::Matrix4 gazebo::common::NodeTransform::Get ( )

Returns the transformation matrix.

Returns

the matrix

10.157.4.2 std::string gazebo::common::NodeTransform::GetSID ( )

Returns thr SID.

Returns

the SID

10.157.4.3 TransformType gazebo::common::NodeTransform::GetType ( )

Returns the transformation type.

Returns

the type

10.157.4.4 math::Matrix4 gazebo::common::NodeTransform::operator() ( )

Matrix cast operator.

Returns

the transform

10.157.4.5 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

10.157.4.6 math::Matrix4 gazebo::common::NodeTransform::operator* (math::Matrix4 _m)

Matrix multiplication operator.

Parameters

in	*_m*	a matrix

Returns

transform matrix multiplied by _m

10.157.4.7 void gazebo::common::NodeTransform::PrintSource ( )

Prints the transform matrix to std::err stream.

10.157.4.8 void gazebo::common::NodeTransform::RecalculateMatrix ( )

Sets the transform matrix from the source according to the type.

10.157.4.9 void gazebo::common::NodeTransform::Set (math::Matrix4 *_mat*)

Assign a transformation.

Parameters

in	*_mat*	the transform

10.157.4.10 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

10.157.4.11 void gazebo::common::NodeTransform::SetSID (std::string *_sid*)

Set the SID.

Parameters

in	*_sid*	the sid

10.157.4.12 void gazebo::common::NodeTransform::SetSourceValues (math::Matrix4 _mat)

Set source data values _ param[in] _mat the values.

10.157.4.13 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 _vec)

Set source data values.

10.157.4.14 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

10.157.4.15 void gazebo::common::NodeTransform::SetType (TransformType _type)

Set transform type.

Parameters

in	*_type*	the type

10.157.5 Member Data Documentation

10.157.5.1 std::string gazebo::common::NodeTransform::sid [protected]

the sid

10.157.5.2 std::vector<double> gazebo::common::NodeTransform::source [protected]

source data values (can be a matrix, a position or rotation)

10.157.5.3 math::Matrix4 gazebo::common::NodeTransform::transform [protected]

transform

10.157.5.4 TransformType gazebo::common::NodeTransform::type [protected]

transform type

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

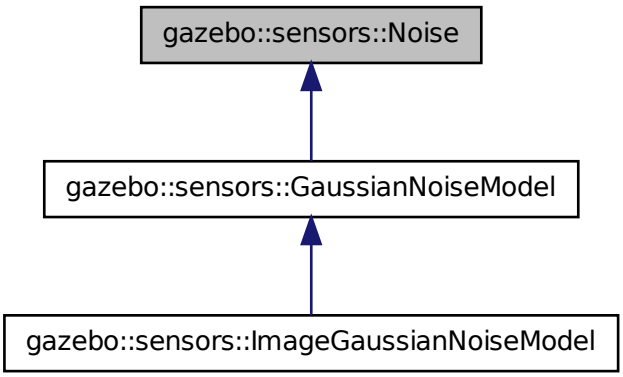
- **Skeleton.hh**

10.158 gazebo::sensors::Noise Class Reference

Noise (p. 931) models for sensor output signals.

`#include <Noise.hh>`

Inheritance diagram for gazebo::sensors::Noise:



Public Types

- enum **NoiseType** { **NONE**, **CUSTOM**, **GAUSSIAN** }

  Which noise types we support.

Public Member Functions

- **Noise** (**NoiseType** _type)

  Constructor.
- virtual ∼**Noise** ()

  Destructor.
- double **Apply** (double _in)

  Apply noise to input data value.
- virtual double **ApplyImpl** (double _in)

  Apply noise to input data value.
- virtual void **Fini** ()

  Finalize the noise model.
- **NoiseType GetNoiseType** () const

  Accessor for NoiseType.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load noise parameters from sdf.
- virtual void **SetCamera** (**rendering::CameraPtr** _camera)

  Set camera needed to create image noise.
- virtual void **SetCustomNoiseCallback** (boost::function< double(double)> _cb)

  Register a custom noise callback.

10.158.1 Detailed Description

Noise (p. 931) models for sensor output signals.

10.158.2 Member Enumeration Documentation

10.158.2.1 enum gazebo::sensors::Noise::NoiseType

Which noise types we support.

Enumerator:

NONE

CUSTOM

GAUSSIAN

10.158.3 Constructor & Destructor Documentation

10.158.3.1 gazebo::sensors::Noise::Noise (NoiseType *_type*) [explicit]

Constructor.

This should not be called directly unless creating an empty noise model. Use **NoiseFactory::NewNoiseModel** (p. 936) to instantiate a new noise model.

Parameters

in	*_type*	Type of noise model.

See also

NoiseFactory::NewNoiseModel (p. 936)

10.158.3.2 virtual gazebo::sensors::Noise::~Noise ( ) [virtual]

Destructor.

10.158.4 Member Function Documentation

10.158.4.1 double gazebo::sensors::Noise::Apply (double *_in*)

Apply noise to input data value.

Parameters

in	*_in*	Input data value.

Returns

Data with noise applied.

10.158.4.2 virtual double gazebo::sensors::Noise::ApplyImpl (double *_in*) [virtual]

Apply noise to input data value.

This gets overriden by derived classes, and called by Apply.

Parameters

in	*_in*	Input data value.

Returns

Data with noise applied.

Reimplemented in **gazebo::sensors::GaussianNoiseModel** (p. 568).

10.158.4.3 virtual void gazebo::sensors::Noise::Fini ( ) [virtual]

Finalize the noise model.

Reimplemented in **gazebo::sensors::ImageGaussianNoiseModel** (p. 645), and **gazebo::sensors::GaussianNoiseModel** (p. 569).

10.158.4.4 NoiseType gazebo::sensors::Noise::GetNoiseType ( ) const

Accessor for NoiseType.

Returns

Type of noise currently in use.

10.158.4.5 virtual void gazebo::sensors::Noise::Load (sdf::ElementPtr *_sdf*) [virtual]

Load noise parameters from sdf.

Parameters

in	*_sdf*	SDF parameters.
in	*_sensor*	Type of sensor.

Reimplemented in **gazebo::sensors::ImageGaussianNoiseModel** (p. 646), and **gazebo::sensors::GaussianNoiseModel** (p. 570).

10.158.4.6 virtual void gazebo::sensors::Noise::SetCamera (rendering::CameraPtr *_camera*) [virtual]

Set camera needed to create image noise.

This is only needed for image sensors, i.e. camera/multicamera/depth sensors, which

use shaders for more efficient noise generation.

Parameters

in	*_camera*	Camera associated to an image sensor

Reimplemented in **gazebo::sensors::ImageGaussianNoiseModel** (p. 646).

10.158.4.7 virtual void gazebo::sensors::Noise::SetCustomNoiseCallback (boost::function< double(double)> *_cb*) [virtual]

Register a custom noise callback.

Parameters

in	*_cb*	Callback function for applying a custom noise model. This is useful if users want to use their own noise model from a sensor plugin.

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

- **Noise.hh**

10.159 gazebo::sensors::NoiseFactory Class Reference

Use this noise manager for creating and loading noise models.

`#include <sensors/sensors.hh>`

Static Public Member Functions

- static **NoisePtr NewNoiseModel** (sdf::ElementPtr _sdf, const std::string &_sensorType="")

  Load a noise model based on the input sdf parameters and sensor type.

10.159.1 Detailed Description

Use this noise manager for creating and loading noise models.

10.159.2 Member Function Documentation

10.159.2.1 static NoisePtr gazebo::sensors::NoiseFactory::NewNoiseModel (sdf::ElementPtr *_sdf*, const std::string & *_sensorType* = " ") [static]

Load a noise model based on the input sdf parameters and sensor type.

Parameters

in	*_sdf*	**Noise** (p. 931) sdf parameters.
in	*_sensorType*	Type of sensor. This is currently used to distinguish between image and non image sensors in order to create the appropriate noise model.

Returns

Pointer to the noise model created.

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

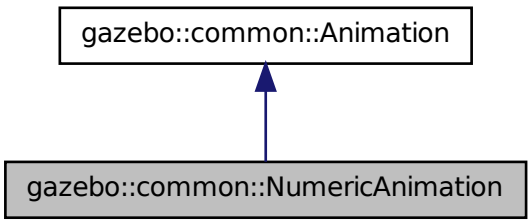
- **Noise.hh**

10.160 gazebo::common::NumericAnimation Class Reference

A numeric animation.

`#include <Animation.hh>`

Inheritance diagram for gazebo::common::NumericAnimation:



Public Member Functions

- **NumericAnimation** (const std::string &_name, double _length, bool _loop)

  Constructor.
- virtual ∼**NumericAnimation** ()

  Destructor.
- **NumericKeyFrame** ∗ **CreateKeyFrame** (double _time)

  Create a numeric keyframe at the given time.
- void **GetInterpolatedKeyFrame** (**NumericKeyFrame** &_kf) const

  Get a keyframe using the animation's current time.

10.160.1 Detailed Description

A numeric animation.

10.160.2 Constructor & Destructor Documentation

10.160.2.1 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

10.160.2.2 virtual gazebo::common::NumericAnimation::∼NumericAnimation ( ) [virtual]

Destructor.

10.160.3 Member Function Documentation

10.160.3.1 NumericKeyFrame∗ gazebo::common::NumericAnimation::CreateKeyFrame (double *_time*)

Create a numeric keyframe at the given time.

Parameters

in	*_time*	**Time** (p. 1361) at which to create the keyframe

Returns

Pointer to the new keyframe

10.160.3.2 void gazebo::common::NumericAnimation::GetInterpolatedKeyFrame (NumericKeyFrame & *_kf*) const

Get a keyframe using the animation's current time.

Parameters

out	*_kf*	**NumericKeyFrame** (p. 938) reference to hold the interpolated result

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

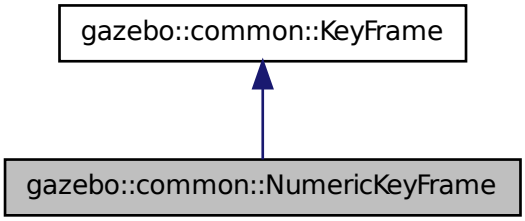
- **Animation.hh**

10.161 gazebo::common::NumericKeyFrame Class Reference

A keyframe for a **NumericAnimation** (p. 936).

`#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

10.161.1 Detailed Description

A keyframe for a **NumericAnimation** (p. 936).

10.161.2 Constructor & Destructor Documentation

10.161.2.1 gazebo::common::NumericKeyFrame::NumericKeyFrame (double *_time*)

Constructor.

Parameters

in	*_time*	**Time** (p. 1361) of the keyframe

10.161.2.2 virtual gazebo::common::NumericKeyFrame::~NumericKeyFrame ( ) [virtual]

Destructor.

10.161.3 Member Function Documentation

10.161.3.1 const double& gazebo::common::NumericKeyFrame::GetValue ( ) const

Get the value of the keyframe.

Returns

the value of the keyframe

10.161.3.2 void gazebo::common::NumericKeyFrame::SetValue (const double & _value)

Set the value of the keyframe.

Parameters

in	*_value*	The new value

10.161.4 Member Data Documentation

10.161.4.1 double gazebo::common::NumericKeyFrame::value [protected]

numeric value

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

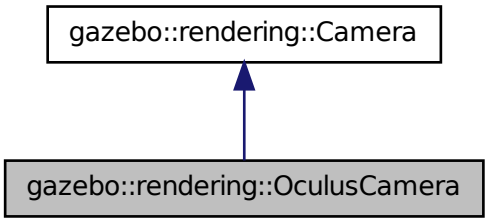
- **KeyFrame.hh**

10.162 gazebo::rendering::OculusCamera Class Reference

A camera used for user visualization of a scene.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::OculusCamera:



Public Member Functions

- **OculusCamera** (const std::string &_name, **ScenePtr** _scene)

  Constructor.
- virtual ~**OculusCamera** ()

  Destructor.
- void **AdjustAspect** (double _v)

  Change screen aspect ratio.
- void **Fini** ()

  Finialize.
- float **GetAvgFPS** () const

  Get the average frames per second.
- virtual unsigned int **GetImageHeight** () const

  Get the height of the image.
- virtual unsigned int **GetImageWidth** () const

  Get the width of the image.
- unsigned int **GetTriangleCount** () const

  Get the triangle count.
- 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.
- bool **Ready** ()

  Used to check if Oculus is plugged in and can be used.
- void **ResetSensor** ()

  Reset the Oculus Rift sensor orientation.
- void **Resize** (unsigned int _w, unsigned int _h)

  Resize the camera.
- virtual void **SetRenderTarget** (Ogre::RenderTarget *_target)

  Set to true to enable rendering.
- virtual void **Update** ()

  Render the camera.

Protected Member Functions

- virtual bool **AttachToVisualImpl** (**VisualPtr** _visual, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  Set the camera to be attached to a visual.
- virtual bool **TrackVisualImpl** (**VisualPtr** _visual)

  Set the camera to track a scene node.

Protected Attributes

- Ogre::Camera * **rightCamera**

  Ogre *(p. 163) camera for the right Oculus screen.*
- Ogre::Viewport * **rightViewport**

  View poer for the right camera.

10.162.1 Detailed Description

A camera used for user visualization of a scene.

10.162.2 Constructor & Destructor Documentation

10.162.2.1 gazebo::rendering::OculusCamera::OculusCamera (const std::string & _name, ScenePtr _scene)

Constructor.

Parameters

in	*_name*	Name of the camera.
in	*_scene*	**Scene** (p. 1097) to put the camera in.

10.162.2.2 virtual gazebo::rendering::OculusCamera::~OculusCamera ( ) [virtual]

Destructor.

10.162.3 Member Function Documentation

10.162.3.1 void gazebo::rendering::OculusCamera::AdjustAspect (double _v)

Change screen aspect ratio.

Parameters

in	*_v*	Aspect ratio.

10.162.3.2 virtual bool gazebo::rendering::OculusCamera::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	*_inheritOrientation*	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. 252).

10.162.3.3 void gazebo::rendering::OculusCamera::Fini ( ) [virtual]

Finialize.

Reimplemented from **gazebo::rendering::Camera** (p. 254).

10.162.3.4 float gazebo::rendering::OculusCamera::GetAvgFPS ( ) const [virtual]

Get the average frames per second.

Returns

The average rendering frames per second

Reimplemented from **gazebo::rendering::Camera** (p. 254).

10.162.3.5 virtual unsigned int gazebo::rendering::OculusCamera::GetImageHeight ( ) const [virtual]

Get the height of the image.

Returns

Image height

Reimplemented from **gazebo::rendering::Camera** (p. 257).

10.162.3.6 virtual unsigned int gazebo::rendering::OculusCamera::GetImageWidth ( ) const [virtual]

Get the width of the image.

Returns

Image width

Reimplemented from **gazebo::rendering::Camera** (p. 257).

10.162.3.7 unsigned int gazebo::rendering::OculusCamera::GetTriangleCount ( ) const [virtual]

Get the triangle count.

Returns

The number of triangles currently being rendered.

Reimplemented from **gazebo::rendering::Camera** (p. 260).

10.162.3.8 void gazebo::rendering::OculusCamera::Init ( ) [virtual]

Initialize.

Reimplemented from **gazebo::rendering::Camera** (p. 263).

10.162.3.9 void gazebo::rendering::OculusCamera::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the user camera.

Parameters

in	*_sdf*	Parameters for the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 264).

10.162.3.10 void gazebo::rendering::OculusCamera::Load ( ) [virtual]

Generic load function.

Reimplemented from **gazebo::rendering::Camera** (p. 264).

10.162.3.11 virtual bool gazebo::rendering::OculusCamera::MoveToPosition (const math::Pose & *_pose*, double *_time*) [virtual]

Move the camera to a position (this is an animated motion).

See also

Camera::MoveToPositions (p. 265)

Parameters

in	*_pose*	End position of the camera
in	*_time*	Duration of the camera's movement

Reimplemented from **gazebo::rendering::Camera** (p. 264).

10.162.3.12 void gazebo::rendering::OculusCamera::MoveToVisual (VisualPtr *_visual*)

Move the camera to focus on a visual.

Parameters

in	*_visual*	**Visual** (p. 1477) to move the camera to.

10.162.3.13 void gazebo::rendering::OculusCamera::MoveToVisual (const std::string & *_visualName*)

Move the camera to focus on a visual.

Parameters

in	*_visual-Name*	Name of the visual to move the camera to.

10.162.3.14 virtual void gazebo::rendering::OculusCamera::PostRender ( ) [virtual]

Post render.

Reimplemented from **gazebo::rendering::Camera** (p. 265).

10.162.3.15 bool gazebo::rendering::OculusCamera::Ready ( )

Used to check if Oculus is plugged in and can be used.

Returns

True when Oculus is ready to use.

10.162.3.16 void gazebo::rendering::OculusCamera::ResetSensor ( )

Reset the Oculus Rift sensor orientation.

10.162.3.17 void gazebo::rendering::OculusCamera::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.

10.162.3.18 virtual void gazebo::rendering::OculusCamera::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. 269).

10.162.3.19 virtual bool gazebo::rendering::OculusCamera::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. 1477) to track.

Returns

True if the camera is now tracking the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 272).

10.162.3.20 virtual void gazebo::rendering::OculusCamera::Update ( ) [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 272).

10.162.4 Member Data Documentation

10.162.4.1 Ogre::Camera* gazebo::rendering::OculusCamera::rightCamera [protected]

Ogre (p. 163) camera for the right Oculus screen.

10.162.4.2 Ogre::Viewport* gazebo::rendering::OculusCamera::rightViewport [protected]

View poer for the right camera.

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

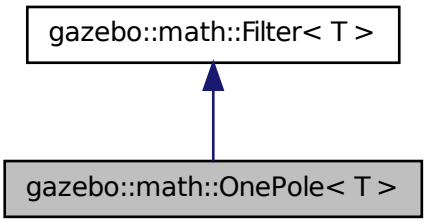
- **OculusCamera.hh**

10.163 gazebo::math::OnePole< T > Class Template Reference

A one-pole DSP filter.

`#include <math/gzmath.hh>`

Inheritance diagram for gazebo::math::OnePole< T >:



Public Member Functions

- **OnePole** ()

  Constructor.
- **OnePole** (double _fc, double _fs)

  Constructor.
- const T & **Process** (const T &_x)

  Update the filter's output.
- virtual void **SetFc** (double _fc, double _fs)

  Set the cutoff frequency and sample rate.

Protected Attributes

- double **a0**

  Input gain control.
- double **b1**

  Gain of the feedback.

10.163.1 Detailed Description

template<class T>class gazebo::math::OnePole< T >

A one-pole DSP filter.

See also

http://www.earlevel.com/main/2012/12/15/a-one-pole-filter/

10.163.2 Constructor & Destructor Documentation

10.163.2.1 template<class T> gazebo::math::OnePole< T >::OnePole ( ) [inline]

Constructor.

10.163.2.2 template<class T> gazebo::math::OnePole< T >::OnePole (double _fc, double _fs) [inline]

Constructor.

Parameters

in	*_fc*	Cutoff frequency.
in	*_fs*	Sample rate.

10.163.3 Member Function Documentation

10.163.3.1 template<class T> const T& gazebo::math::OnePole< T >::Process (const T & _x) [inline]

Update the filter's output.

[in] _x Input value.

Returns

The filter's current output.

Reimplemented in **gazebo::math::OnePoleQuaternion** (p. 953).

10.163.3.2 template<class T> virtual void gazebo::math::OnePole< T >::SetFc (double _fc, double _fs) [inline, virtual]

Set the cutoff frequency and sample rate.

Parameters

in	*_fc*	Cutoff frequency.
in	*_fs*	Sample rate.

Implements **gazebo::math::Filter< T >** (p. 555).

10.163.4 Member Data Documentation

10.163.4.1 template<class T> double gazebo::math::OnePole< T >::a0 [protected]

Input gain control.

10.163.4.2 template<class T> double gazebo::math::OnePole< T >::b1 [protected]

Gain of the feedback.

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

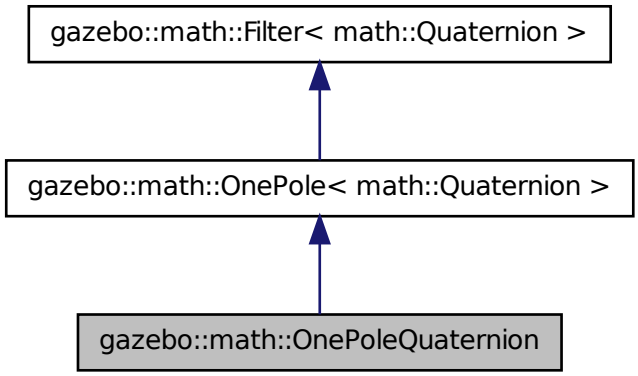
- **Filter.hh**

10.164 gazebo::math::OnePoleQuaternion Class Reference

One-pole quaternion filter.

`#include <math/gzmath.hh>`

Inheritance diagram for gazebo::math::OnePoleQuaternion:



Public Member Functions

- **OnePoleQuaternion** ()

  Constructor.
- **OnePoleQuaternion** (double _fc, double _fs)

  Constructor.
- const **math::Quaternion** & **Process** (const **math::Quaternion** &_x)

  Update the filter's output.

10.164.1 Detailed Description

One-pole quaternion filter.

10.164.2 Constructor & Destructor Documentation

10.164.2.1 gazebo::math::OnePoleQuaternion::OnePoleQuaternion ( ) [inline]

Constructor.

10.164.2.2 gazebo::math::OnePoleQuaternion::OnePoleQuaternion (double _fc, double _fs) [inline]

Constructor.

Parameters

in	*_fc*	Cutoff frequency.
in	*_fs*	Sample rate.

10.164.3 Member Function Documentation

10.164.3.1 const math::Quaternion& gazebo::math::OnePoleQuaternion::Process (const math::Quaternion & _x) [inline]

Update the filter's output.

[in] _x Input value.

Returns

The filter's current output.

Reimplemented from **gazebo::math::OnePole< math::Quaternion >** (p. 950).

References gazebo::math::Quaternion::Slerp().

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

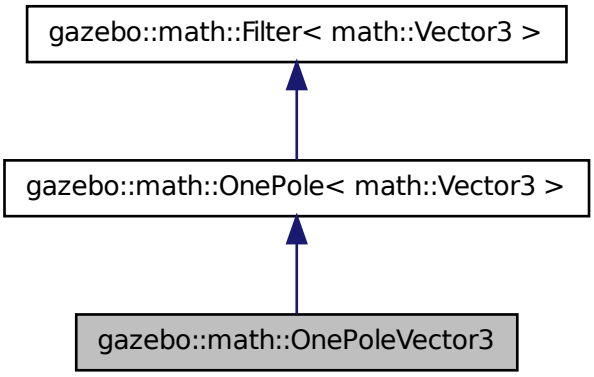
- **Filter.hh**

10.165 gazebo::math::OnePoleVector3 Class Reference

One-pole vector3 filter.

`#include <math/gzmath.hh>`

Inheritance diagram for gazebo::math::OnePoleVector3:



Public Member Functions

- **OnePoleVector3** ()

  Constructor.
- **OnePoleVector3** (double _fc, double _fs)

  Constructor.

10.165.1 Detailed Description

One-pole vector3 filter.

10.165.2 Constructor & Destructor Documentation

10.165.2.1 gazebo::math::OnePoleVector3::OnePoleVector3 ( ) [inline]

Constructor.

10.165.2.2 gazebo::math::OnePoleVector3::OnePoleVector3 (double *_fc*, double *_fs*) [inline]

Constructor.

Parameters

in	*_fc*	Cutoff frequency.
in	*_fs*	Sample rate.

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

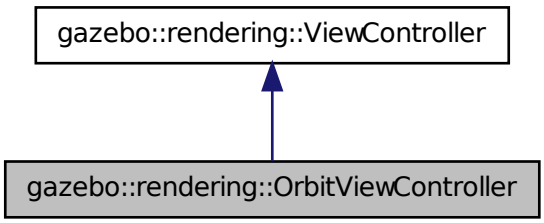
- **Filter.hh**

10.166 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.

10.166.1 Detailed Description

Orbit view controller.

10.166.2 Constructor & Destructor Documentation

10.166.2.1 gazebo::rendering::OrbitViewController::OrbitViewController (UserCameraPtr *_camera*)

Constructor.

Parameters

in	*_camera*	Pointer to the camera to control.

10.166.2.2 virtual gazebo::rendering::OrbitViewController::~OrbitViewController ( ) [virtual]

Destructor.

10.166.3 Member Function Documentation

10.166.3.1 math::Vector3 gazebo::rendering::OrbitViewController::GetFocalPoint ( ) const

Get the focal point.

Returns

The focal point

10.166.3.2 static std::string gazebo::rendering::OrbitViewController::GetTypeString ( ) [static]

Get the type name of this view controller.

Returns

The view controller name: "orbit".

10.166.3.3 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. 1475).

10.166.3.4 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. 1475).

10.166.3.5 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. 1476).

10.166.3.6 virtual void gazebo::rendering::OrbitViewController::Init ( ) [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 1476).

10.166.3.7 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. 1476).

10.166.3.8 void gazebo::rendering::OrbitViewController::SetDistance (float *_d*)

Set the distance to the focal point.

Parameters

in	*_d*	The distance from the focal point.

10.166.3.9 void gazebo::rendering::OrbitViewController::SetFocalPoint (const math::Vector3 & _fp)

Set the focal point.

Parameters

in	_fp	The focal point

10.166.3.10 virtual void gazebo::rendering::OrbitViewController::Update ( ) [virtual]

Update.

Implements **gazebo::rendering::ViewController** (p. 1476).

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

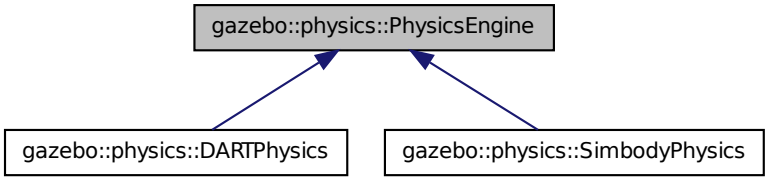
- **OrbitViewController.hh**

10.167 gazebo::physics::PhysicsEngine Class Reference

Base (p. 201) class for a physics engine.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::PhysicsEngine:



Public Member Functions

- **PhysicsEngine** (**WorldPtr** _world)

Default constructor.

- virtual ~**PhysicsEngine** ()

  Destructor.

- virtual **CollisionPtr CreateCollision** (const std::string &_shapeType, **LinkPtr** _link)=0

  Create a collision.

- **CollisionPtr CreateCollision** (const std::string &_shapeType, const std::string &_linkName)

  Create a collision.

- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent=**ModelPtr**())=0

  Create a new joint.

- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)=0

  Create a new body.

- virtual **ModelPtr CreateModel** (**BasePtr** _base)

  Create a new model.

- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)=0

  *Create a **physics::Shape** (p. 1161) 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 unsigned 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** (const std::string &_key) const

  Get an parameter of the physics engine.

- boost::recursive_mutex * **GetPhysicsUpdateMutex** () const

*returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 975).*

- double **GetRealTimeUpdateRate** () const

  Get real time update rate.
- double **GetTargetRealTimeFactor** () const

  Get target real time factor.
- virtual std::string **GetType** () const =0

  Return the physics engine type (ode|bullet|dart|simbody).
- double **GetUpdatePeriod** ()

  Get the simulation update period.
- 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** (unsigned int _maxContacts)

  : Remove this function, and replace it with a more generic property map
- void **SetMaxStepSize** (double _stepSize)

  Set max step size.
- virtual bool **SetParam** (const 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.
- void **SetTargetRealTimeFactor** (double _factor)

Set target real time factor.

- 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.

10.167.1 Detailed Description

Base (p. 201) class for a physics engine.

10.167.2 Constructor & Destructor Documentation

10.167.2.1 gazebo::physics::PhysicsEngine::PhysicsEngine (WorldPtr *_world*) [explicit]

Default constructor.

Parameters

in	*_world*	Pointer to the world.

10.167.2.2 virtual gazebo::physics::PhysicsEngine::~PhysicsEngine ( ) [virtual]

Destructor.

10.167.3 Member Function Documentation

10.167.3.1 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.

Implemented in **gazebo::physics::DARTPhysics** (p. 440), and **gazebo::physics::SimbodyPhysics** (p. 1229).

10.167.3.2 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.

10.167.3.3 virtual JointPtr gazebo::physics::PhysicsEngine::CreateJoint (const std::string & *_type*, ModelPtr *_parent* = ModelPtr()) [pure virtual]

Create a new joint.

Parameters

in	*_type*	Type of joint to create.
in	*_parent*	**Model** (p. 846) parent.

Implemented in **gazebo::physics::DARTPhysics** (p. 441), and **gazebo::physics::SimbodyPhysics** (p. 1229).

10.167.3.4 virtual LinkPtr gazebo::physics::PhysicsEngine::CreateLink (ModelPtr *_parent*) [pure virtual]

Create a new body.

Parameters

in	*_parent*	Parent model for the link.

Implemented in **gazebo::physics::DARTPhysics** (p. 441), and **gazebo::physics::SimbodyPhysics** (p. 1229).

10.167.3.5 virtual ModelPtr gazebo::physics::PhysicsEngine::CreateModel (BasePtr *_base*) [virtual]

Create a new model.

Parameters

in	*_base*	Boost shared pointer to a new model.

Reimplemented in **gazebo::physics::DARTPhysics** (p. 441), and **gazebo::physics::SimbodyPhysics** (p. 1230).

10.167.3.6 virtual ShapePtr gazebo::physics::PhysicsEngine::CreateShape (const std::string & *_shapeType*, CollisionPtr *_collision*) [pure virtual]

Create a **physics::Shape** (p. 1161) object.

Parameters

in	*_shapeType*	Type of shape to create.
in	*_collision*	**Collision** (p. 295) parent.

Implemented in **gazebo::physics::DARTPhysics** (p. 442), and **gazebo::physics::SimbodyPhysics** (p. 1230).

10.167.3.7 virtual void gazebo::physics::PhysicsEngine::DebugPrint ( ) const [pure virtual]

Debug print out of the physic engine state.

Implemented in **gazebo::physics::DARTPhysics** (p. 442), and **gazebo::physics::SimbodyPhysics** (p. 1230).

10.167.3.8 virtual void gazebo::physics::PhysicsEngine::Fini ( ) [virtual]

Finilize the physics engine.

Reimplemented in **gazebo::physics::DARTPhysics** (p. 442), and **gazebo::physics::SimbodyPhysics** (p. 1230).

10.167.3.9 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.

10.167.3.10 ContactManager* gazebo::physics::PhysicsEngine::GetContactManager ( ) const

Get a pointer to the contact manger.

Returns

Pointer to the contact manager.

10.167.3.11 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.

10.167.3.12 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. 347) suerface layer depth.

10.167.3.13 virtual math::Vector3 gazebo::physics::PhysicsEngine::GetGravity ( ) const [virtual]

Return the gavity vector.

Returns

The gavity vector.

10.167.3.14 virtual unsigned 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.

10.167.3.15 double gazebo::physics::PhysicsEngine::GetMaxStepSize ( ) const

Get max step size.

Returns

Max step size.

10.167.3.16 virtual boost::any gazebo::physics::PhysicsEngine::GetParam (const std::string & *_key*) const [virtual]

Get an parameter of the physics engine.

Parameters

in	*_attr*	String key

See also

SetParam (p. 972)

Returns

The value of the parameter

Reimplemented in **gazebo::physics::SimbodyPhysics** (p. 1231), and **gazebo::physics::DARTPhysics** (p. 442).

10.167.3.17 boost::recursive_mutex* gazebo::physics::PhysicsEngine::GetPhysicsUpdateMutex ( ) const [inline]

returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 975).

Returns

Pointer to the physics mutex.

10.167.3.18 double gazebo::physics::PhysicsEngine::GetRealTimeUpdateRate ( ) const

Get real time update rate.

Returns

Update rate

10.167.3.19 double gazebo::physics::PhysicsEngine::GetTargetRealTimeFactor ( ) const

Get target real time factor.

Returns

Target real time factor

10.167.3.20 virtual std::string gazebo::physics::PhysicsEngine::GetType ( ) const [pure virtual]

Return the physics engine type (ode|bullet|dart|simbody).

Returns

Type of the physics engine.

Implemented in **gazebo::physics::DARTPhysics** (p. 443), and **gazebo::physics::SimbodyPhysics** (p. 1231).

10.167.3.21 double gazebo::physics::PhysicsEngine::GetUpdatePeriod ( )

Get the simulation update period.

Returns

Simulation update period.

10.167.3.22 virtual double gazebo::physics::PhysicsEngine::GetWorldCFM ( ) [inline, virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 1529) CFM.

Returns

World (p. 1529) CFM.

10.167.3.23 virtual double gazebo::physics::PhysicsEngine::GetWorldERP ( ) [inline, virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 1529) ERP.

Returns

World (p. 1529) ERP.

10.167.3.24 virtual void gazebo::physics::PhysicsEngine::Init ( ) [pure virtual]

Initialize the physics engine.

Implemented in **gazebo::physics::DARTPhysics** (p. 443), and **gazebo::physics::SimbodyPhysics** (p. 1232).

10.167.3.25 virtual void gazebo::physics::PhysicsEngine::InitForThread ( ) [pure virtual]

Init the engine for threads.

Implemented in **gazebo::physics::DARTPhysics** (p. 443), and **gazebo::physics::SimbodyPhysics** (p. 1232).

10.167.3.26 virtual void gazebo::physics::PhysicsEngine::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the physics engine.

Parameters

in	*_sdf*	Pointer to the SDF parameters.

Reimplemented in **gazebo::physics::DARTPhysics** (p. 443), and **gazebo::physics::SimbodyPhysics** (p. 1233).

10.167.3.27 virtual void gazebo::physics::PhysicsEngine::OnPhysicsMsg (ConstPhysicsPtr & *_msg*) [protected, virtual]

virtual callback for gztopic "~/physics".

Parameters

in	*_msg*	Physics message.

Reimplemented in **gazebo::physics::SimbodyPhysics** (p. 1233), and **gazebo::physics::DARTPhysics** (p. 444).

10.167.3.28 virtual void gazebo::physics::PhysicsEngine::OnRequest (ConstRequestPtr & *_msg*) [protected, virtual]

virtual callback for gztopic "~/request".

Parameters

in	*_msg*	Request message.

Reimplemented in **gazebo::physics::SimbodyPhysics** (p. 1233), and **gazebo::physics::DARTPhysics** (p. 444).

10.167.3.29 virtual void gazebo::physics::PhysicsEngine::Reset ( ) [inline, virtual]

Rest the physics engine.

Reimplemented in **gazebo::physics::DARTPhysics** (p. 444), and **gazebo::physics::SimbodyPhysics** (p. 1234).

10.167.3.30 virtual void gazebo::physics::PhysicsEngine::SetAutoDisableFlag (bool *_autoDisable*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_autoDisable*	True to enable auto disabling of bodies.

10.167.3.31 virtual void gazebo::physics::PhysicsEngine::SetContactMaxCorrectingVel (double *_vel*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_vel*	Max correcting velocity.

10.167.3.32 virtual void gazebo::physics::PhysicsEngine::SetContactSurfaceLayer (double *_layerDepth*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_layerDepth*	Surface layer depth

10.167.3.33 virtual void gazebo::physics::PhysicsEngine::SetGravity (const gazebo::math::Vector3 & *_gravity*) [pure virtual]

Set the gavity vector.

Parameters

in	*_gravity*	New gravity vector.

Implemented in **gazebo::physics::DARTPhysics** (p. 444), and **gazebo::physics::SimbodyPhysics** (p. 1235).

10.167.3.34 virtual void gazebo::physics::PhysicsEngine::SetMaxContacts (unsigned int *_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.

10.167.3.35 void gazebo::physics::PhysicsEngine::SetMaxStepSize (double *_stepSize*)

Set max step size.

Parameters

in	*_stepSize*	Max step size.

10.167.3.36 virtual bool gazebo::physics::PhysicsEngine::SetParam (const std::string & *_key*, const boost::any & *_value*) [virtual]

Set a parameter of the physics engine.

See SetParam documentation for descriptions of duplicate parameters.

Parameters

in	*_key*	String key Below is a list of _key parameter definitions: 1. "solver_type" (string) - returns solver used by engine, e.g. "sequential_impulse' for Bullet, "quick" for ODE "Featherstone and Lemkes" for DART and "Spatial - Algebra and Elastic Foundation" for Simbody. -# "cfm" (double) - global CFM -# "erp" (double) - global ERP -# "precon_iters" (bool) - precondition iterations (experimental). 2. "iters" (int) - number of LCP PGS iterations. If sor_lcp_tolerance is negative, full iteration count is executed. Otherwise, PGS may stop iteration early if sor_lcp_tolerance is satisfied by the total RMS residual. 3. "sor" (double) - relaxation parameter for Projected - Gauss-Seidel (PGS) updates. 4. "contact_max_correcting_vel" (double) - truncates correction impulses from ERP by this value. 5. "contact_surface_layer" (double) - ERP is 0 for interpenetration depths below this value. 6. "max_contacts" (int) - max number of contact constraints between any pair of collision bodies. 7. "min_step_size" (double) - minimum internal step size. (defined but not used in ode). 8. "max_step_size" (double) - maximum physics step size when physics update step must return.
in	*_value*	The value to set to

Returns

true if SetParam is successful, false if operation fails.

Reimplemented in **gazebo::physics::SimbodyPhysics** (p. 1235), and **gazebo::physics::DARTPhysics** (p. 444).

10.167.3.37 void gazebo::physics::PhysicsEngine::SetRealTimeUpdateRate (double *_rate*)

Set real time update rate.

Parameters

in	*_rate*	Update rate

10.167.3.38 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.

Implemented in **gazebo::physics::DARTPhysics** (p. 445), and **gazebo::physics::SimbodyPhysics** (p. 1236).

10.167.3.39 void gazebo::physics::PhysicsEngine::SetTargetRealTimeFactor (double *_factor*)

Set target real time factor.

Parameters

in	*_factor*	Target real time factor

10.167.3.40 virtual void gazebo::physics::PhysicsEngine::SetWorldCFM (double *_cfm*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_cfm*	Constraint force mixing.

10.167.3.41 virtual void gazebo::physics::PhysicsEngine::SetWorldERP (double *_erp*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_erp*	Error reduction parameter.

10.167.3.42 virtual void gazebo::physics::PhysicsEngine::UpdateCollision ( ) [pure virtual]

Update the physics engine collision.

Implemented in **gazebo::physics::DARTPhysics** (p. 446), and **gazebo::physics::SimbodyPhysics** (p. 1237).

10.167.3.43 virtual void gazebo::physics::PhysicsEngine::UpdatePhysics ( ) [inline, virtual]

Update the physics engine.

Reimplemented in **gazebo::physics::DARTPhysics** (p. 446), and **gazebo::physics::SimbodyPhysics** (p. 1237).

10.167.4 Member Data Documentation

10.167.4.1 ContactManager* gazebo::physics::PhysicsEngine::contactManager [protected]

Class that handles all contacts generated by the physics engine.

10.167.4.2 double gazebo::physics::PhysicsEngine::maxStepSize [protected]

Real time update rate.

10.167.4.3 transport::NodePtr gazebo::physics::PhysicsEngine::node [protected]

Node for communication.

10.167.4.4 transport::SubscriberPtr gazebo::physics::PhysicsEngine::physicsSub [protected]

Subscribe to the physics topic.

10.167.4.5 boost::recursive_mutex* gazebo::physics::PhysicsEngine::physicsUpdateMutex [protected]

Mutex to protect the update cycle.

10.167.4.6 double gazebo::physics::PhysicsEngine::realTimeUpdateRate [protected]

Real time update rate.

10.167.4.7 transport::SubscriberPtr gazebo::physics::PhysicsEngine::requestSub [protected]

Subscribe to the request topic.

10.167.4.8 transport::PublisherPtr gazebo::physics::PhysicsEngine::responsePub [protected]

Response publisher.

10.167.4.9 sdf::ElementPtr gazebo::physics::PhysicsEngine::sdf [protected]

Our SDF values.

10.167.4.10 double gazebo::physics::PhysicsEngine::targetRealTimeFactor [protected]

Target real time factor.

10.167.4.11 WorldPtr gazebo::physics::PhysicsEngine::world [protected]

Pointer to the world.

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

- **PhysicsEngine.hh**

10.168 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.

10.168.1 Detailed Description

The physics factory instantiates different physics engines.

10.168.2 Member Function Documentation

10.168.2.1 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.

10.168.2.2 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. 1529) to pass to the created physics engine.

10.168.2.3 static void gazebo::physics::PhysicsFactory::RegisterAll ( ) [static]

Register everything.

10.168.2.4 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**

10.169 gazebo::common::PID Class Reference

Generic **PID** (p. 977) 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. 977) controller.*
- double **GetCmdMax** () const

  Get the maximum value for the command.
- double **GetCmdMin** () const

  Get the maximum value for the command.
- double **GetDGain** () const

  Get the derivative Gain.
- void **GetErrors** (double &_pe, double &_ie, double &_de)

  *Return **PID** (p. 977) error terms for the controller.*
- double **GetIGain** () const

  Get the integral Gain.
- double **GetIMax** () const

  Get the integral upper limit.
- double **GetIMin** () const

  Get the integral lower limit.
- double **GetPGain** () const

  Get the proportional Gain.
- 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. 977) 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.

10.169.1 Detailed Description

Generic **PID** (p. 977) 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.

10.169.2 Constructor & Destructor Documentation

10.169.2.1 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.
in	*_cmdMax*	Output max value.
in	*_cmdMin*	Output min value.

10.169.2.2 virtual gazebo::common::PID::~PID ( ) [virtual]

Destructor.

10.169.3 Member Function Documentation

10.169.3.1 double gazebo::common::PID::GetCmd ( )

Return current command for this **PID** (p. 977) controller.

Returns

the command value

10.169.3.2 double gazebo::common::PID::GetCmdMax ( ) const

Get the maximum value for the command.

Returns

The maximum value

10.169.3.3 double gazebo::common::PID::GetCmdMin ( ) const

Get the maximum value for the command.

Returns

The maximum value

10.169.3.4 double gazebo::common::PID::GetDGain ( ) const

Get the derivative Gain.

Returns

The derivative gain value

10.169.3.5 void gazebo::common::PID::GetErrors (double & *_pe*, double & *_ie*, double & *_de*)

Return **PID** (p. 977) error terms for the controller.

Parameters

in	*_pe*	The proportional error.
in	*_ie*	The integral error.
in	*_de*	The derivative error.

10.169.3.6 double gazebo::common::PID::GetIGain ( ) const

Get the integral Gain.

Returns

The integral gain value

10.169.3.7 double gazebo::common::PID::GetIMax ( ) const

Get the integral upper limit.

Returns

The integral upper limit value

10.169.3.8 double gazebo::common::PID::GetIMin ( ) const

Get the integral lower limit.

Returns

The integral lower limit value

10.169.3.9 double gazebo::common::PID::GetPGain ( ) const

Get the proportional Gain.

Returns

The proportional gain value

10.169.3.10 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.
in	*_cmdMax*	Output max value.
in	*_cmdMin*	Output min value.

10.169.3.11 PID& gazebo::common::PID::operator= (const PID & *_p*) [inline]

Assignment operator.

Parameters

in	*_p*	a reference to a **PID** (p. 977) to assign values from

Returns

reference to this instance

10.169.3.12 void gazebo::common::PID::Reset ( )

Reset the errors and command.

10.169.3.13 void gazebo::common::PID::SetCmd (double *_cmd*)

Set current target command for this **PID** (p. 977) controller.

Parameters

in	*_cmd*	New command

10.169.3.14 void gazebo::common::PID::SetCmdMax (double *_c*)

Set the maximum value for the command.

Parameters

in	*_c*	The maximum value

10.169.3.15 void gazebo::common::PID::SetCmdMin (double _c)

Set the maximum value for the command.

Parameters

in	_c	The maximum value

10.169.3.16 void gazebo::common::PID::SetDGain (double _d)

Set the derivtive Gain.

Parameters

in	_p	dertivative gain value

10.169.3.17 void gazebo::common::PID::SetIGain (double _i)

Set the integral Gain.

Parameters

in	_p	integral gain value

10.169.3.18 void gazebo::common::PID::SetIMax (double _i)

Set the integral upper limit.

Parameters

in	_p	integral upper limit value

10.169.3.19 void gazebo::common::PID::SetIMin (double _i)

Set the integral lower limit.

Parameters

in	_p	integral lower limit value

10.169.3.20 void gazebo::common::PID::SetPGain (double _p)

Set the proportional Gain.

Parameters

in	*_p*	proportional gain value

10.169.3.21 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**

10.170 gazebo::math::Plane Class Reference

A plane and related functions.

`#include <math/gzmath.hh>`

Public Member Functions

- **Plane** ()

  Constructor.
- **Plane** (const **Vector3** &_normal, double _offset=0.0)

  Constructor from a normal and a distanec.
- **Plane** (const **Vector3** &_normal, const **Vector2d** &_size, double _offset)

  Constructor.

- virtual ~**Plane** ()

  Destructor.
- double **Distance** (const **Vector3** &_origin, const **Vector3** &_dir) const

  Get distance to the plane give an origin and direction.
- **Plane** & **operator=** (const **Plane** &_p)

  Equal operator.
- void **Set** (const **Vector3** &_normal, const **Vector2d** &_size, double offset)

  Set the plane.

Public Attributes

- double **d**

  Plane *(*p. *984) offset.*
- **Vector3 normal**

  Plane *(*p. *984) normal.*
- **Vector2d size**

  Plane *(*p. *984) size.*

10.170.1 Detailed Description

A plane and related functions.

10.170.2 Constructor & Destructor Documentation

10.170.2.1 gazebo::math::Plane::Plane ( )

Constructor.

10.170.2.2 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

10.170.2.3 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

10.170.2.4 virtual gazebo::math::Plane::~Plane ( ) [virtual]

Destructor.

10.170.3 Member Function Documentation

10.170.3.1 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

10.170.3.2 Plane& gazebo::math::Plane::operator= (const Plane & _p)

Equal operator.

Parameters

_p	another plane

Returns

itself

10.170.3.3 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

10.170.4 Member Data Documentation

10.170.4.1 double gazebo::math::Plane::d

Plane (p. 984) offset.

10.170.4.2 Vector3 gazebo::math::Plane::normal

Plane (p. 984) normal.

10.170.4.3 Vector2d gazebo::math::Plane::size

Plane (p. 984) size.

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

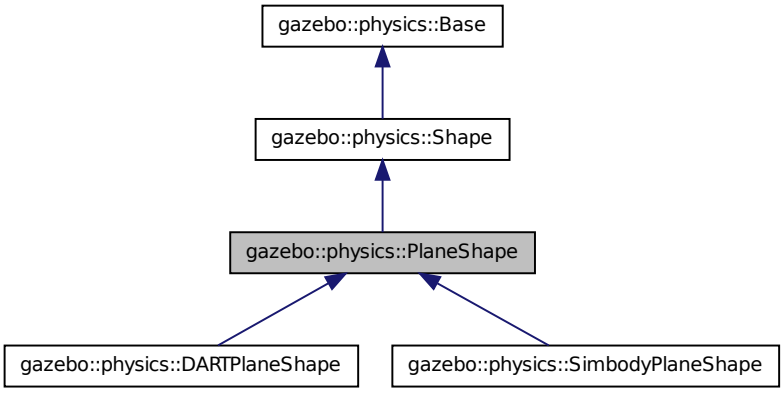
- **Plane.hh**

10.171 gazebo::physics::PlaneShape Class Reference

Collision (p. 295) 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.

- virtual void **SetScale** (const **math::Vector3** &_scale)

  Set the scale of the plane.

- void **SetSize** (const **math::Vector2d** &_size)

  Set the size.

10.171.1 Detailed Description

Collision (p. 295) 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.

10.171.2 Constructor & Destructor Documentation

10.171.2.1 gazebo::physics::PlaneShape::PlaneShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	**Link** (p. 739) to which we are attached.

10.171.2.2 virtual gazebo::physics::PlaneShape::~PlaneShape ( ) [virtual]

Destructor.

10.171.3 Member Function Documentation

10.171.3.1 virtual void gazebo::physics::PlaneShape::CreatePlane ( ) [virtual]

Create the plane.

Reimplemented in **gazebo::physics::SimbodyPlaneShape** (p. 1240), and **gazebo::physics::DARTPlaneShape** (p. 448).

Referenced by gazebo::physics::DARTPlaneShape::CreatePlane().

10.171.3.2 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. 1163).

10.171.3.3 math::Vector3 gazebo::physics::PlaneShape::GetNormal ( ) const

Get the plane normal.

Returns

The plane normal.

10.171.3.4 math::Vector2d gazebo::physics::PlaneShape::GetSize ( ) const

Get the size.

Returns

Size of the plane.

10.171.3.5 virtual void gazebo::physics::PlaneShape::Init ( ) [virtual]

Initialize the plane.

Implements **gazebo::physics::Shape** (p. 1164).

10.171.3.6 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. 1164).

10.171.3.7 virtual void gazebo::physics::PlaneShape::SetAltitude (const math::Vector3 & _pos) [virtual]

Set the altitude of the plane.

Parameters

in	*_pos*	Position of the plane.

Reimplemented in **gazebo::physics::DARTPlaneShape** (p. 448), and **gazebo::physics::SimbodyPlaneShape** (p. 1240).

Referenced by gazebo::physics::DARTPlaneShape::SetAltitude().

10.171.3.8 void gazebo::physics::PlaneShape::SetNormal (const math::Vector3 & _norm)

Set the normal.

Parameters

in	*_norm*	Plane normal.

10.171.3.9 virtual void gazebo::physics::PlaneShape::SetScale (const math::Vector3 & _scale) [virtual]

Set the scale of the plane.

Returns

_scale Scale to set the plane to.

Implements **gazebo::physics::Shape** (p. 1165).

10.171.3.10 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**

10.172 gazebo::PluginT< T > Class Template Reference

A class which all plugins must inherit from.

`#include <common/common.hh>`

Classes

- union **fptr_union_t**

  Pointer to shared library registration function definition.

Public Types

- typedef boost::shared_ptr< T > **TPtr**

  plugin pointer type definition

Public Member Functions

- **PluginT** ()

  Constructor.
- virtual ~**PluginT** ()

  Destructor.
- 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.

10.172.1 Detailed Description

template<class T>class gazebo::PluginT< T >

A class which all plugins must inherit from.

10.172.2 Member Typedef Documentation

10.172.2.1 template<class T> typedef boost::shared_ptr<T> gazebo::PluginT< T >::TPtr

plugin pointer type definition

10.172.3 Constructor & Destructor Documentation

10.172.3.1 template<class T> gazebo::PluginT< T >::PluginT ( ) [inline]

Constructor.

10.172.3.2 template<class T> virtual gazebo::PluginT< T >::~PluginT ( ) [inline, virtual]

Destructor.

10.172.4 Member Function Documentation

10.172.4.1 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

10.172.4.2 template<class T> std::string gazebo::PluginT< T >::GetFilename ( ) const [inline]

Get the name of the handler.

10.172.4.3 template<class T> std::string gazebo::PluginT< T >::GetHandle ( ) const [inline]

Get the short name of the handler.

10.172.4.4 template<class T> PluginType gazebo::PluginT< T >::GetType ( ) const [inline]

Returns the type of the plugin.

Returns

type of the plugin

10.172.5 Member Data Documentation

10.172.5.1 template<class T> std::string gazebo::PluginT< T >::filename [protected]

Path to the shared library file.

10.172.5.2 template<class T> std::string gazebo::PluginT< T >::handle [protected]

Short name.

10.172.5.3 template<class T> PluginType gazebo::PluginT< T >::type [protected]

Type of plugin.

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

- **Plugin.hh**

10.173 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 A is the transform from O to P specified in frame O B is the transform from P to Q specified in frame P then, B + A is the transform from O to Q specified in frame O.

- const **Pose** & **operator+=** (const **Pose** &_pose)

  Add-Equals operator.

- **Pose operator-** () const

  Negation operator A is the transform from O to P in frame O then -A is transform from P to O specified in frame P.

- **Pose operator-** (const **Pose** &_pose) const

  Subtraction operator A is the transform from O to P in frame O B is the transform from O to Q in frame O B - A is the transform from P to Q in frame P.

- const **Pose** & **operator-=** (const **Pose** &_pose)

  Subtraction operator.

- **Pose** & **operator=** (const **Pose** &_pose)

  Equal 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. *1440) and a **Quaternion*** (p. *1029).*

- void **Set** (const **Vector3** &_pos, const **Vector3** &_rpy)

  Set the pose from pos and rpy vectors.

- 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.

10.173.1 Detailed Description

Encapsulates a position and rotation in three space.

10.173.2 Constructor & Destructor Documentation

10.173.2.1 gazebo::math::Pose::Pose ( )

Default constructors.

10.173.2.2 gazebo::math::Pose::Pose (const Vector3 & *_pos*, const Quaternion & *_rot*)

Constructor.

Parameters

in	*_pos*	A position
in	*_rot*	A rotation

10.173.2.3 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.

10.173.2.4 gazebo::math::Pose::Pose (const Pose & _pose)

Copy constructor.

Parameters

in	*_pose*	**Pose** (p. 995) to copy

10.173.2.5 virtual gazebo::math::Pose::~Pose ( ) [virtual]

Destructor.

10.173.3 Member Function Documentation

10.173.3.1 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

10.173.3.2 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

10.173.3.3 Vector3 gazebo::math::Pose::CoordPositionAdd (const Pose & *_pose*) const

Add one point to another: result = this + pose.

Parameters

in	*_pose*	The **Pose** (p. 995) to add

Returns

The resulting position

10.173.3.4 Vector3 gazebo::math::Pose::CoordPositionSub (const Pose & *_pose*) const [inline]

Subtract one position from another: result = this - pose.

Parameters

in	*_pose*	**Pose** (p. 995) 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.

10.173.3.5 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

10.173.3.6 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(), and gazebo::math::Quaternion::-Normalize().

10.173.3.7 void gazebo::math::Pose::Correct ( ) [inline]

Fix any nan values.

10.173.3.8 Pose gazebo::math::Pose::GetInverse ( ) const

Get the inverse of this pose.

Returns

the inverse pose

10.173.3.9 bool gazebo::math::Pose::IsFinite ( ) const

See if a pose is finite (e.g., not nan)

10.173.3.10 bool gazebo::math::Pose::operator!= (const Pose & _pose) const

Inequality operator.

Parameters

in	*_pose*	**Pose** (p. 995) for comparison

Returns

True if not equal

10.173.3.11 Pose gazebo::math::Pose::operator* (const Pose & _pose)

Multiplication operator.

Parameters

in	*_pose*	the other pose

Returns

itself

10.173.3.12 Pose gazebo::math::Pose::operator+ (const Pose & _pose) const

Addition operator A is the transform from O to P specified in frame O B is the transform from P to Q specified in frame P then, B + A is the transform from O to Q specified in frame O.

Parameters

in	*_pose*	**Pose** (p. 995) to add to this pose

Returns

The resulting pose

10.173.3.13 const Pose& gazebo::math::Pose::operator+= (const Pose & _pose)

Add-Equals operator.

Parameters

in	*_pose*	**Pose** (p. 995) to add to this pose

Returns

The resulting pose

10.173.3.14 Pose gazebo::math::Pose::operator- ( ) const [inline]

Negation operator A is the transform from O to P in frame O then -A is transform from P to O specified in frame P.

Returns

The resulting pose

10.173.3.15 Pose gazebo::math::Pose::operator- (const Pose & *_pose*) const [inline]

Subtraction operator A is the transform from O to P in frame O B is the transform from O to Q in frame O B - A is the transform from P to Q in frame P.

Parameters

in	*_pose*	**Pose** (p. 995) to subtract from this one

Returns

The resulting pose

References rot.

10.173.3.16 const Pose& gazebo::math::Pose::operator-= (const Pose & *_pose*)

Subtraction operator.

Parameters

in	*_pose*	**Pose** (p. 995) to subtract from this one

Returns

The resulting pose

10.173.3.17 Pose& gazebo::math::Pose::operator= (const Pose & _pose)

Equal operator.

Parameters

in	*_pose*	**Pose** (p. 995) to copy

10.173.3.18 bool gazebo::math::Pose::operator== (const Pose & _pose) const

Equality operator.

Parameters

in	*_pose*	**Pose** (p. 995) for comparison

Returns

True if equal

10.173.3.19 void gazebo::math::Pose::Reset ( )

Reset the pose.

10.173.3.20 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

10.173.3.21 void gazebo::math::Pose::Round (int _precision)

Round all values to _precision decimal places.

Parameters

in	*_precision*	

10.173.3.22 void gazebo::math::Pose::Set (const Vector3 & _pos, const Quaternion & _rot)

Set the pose from a **Vector3** (p. 1440) and a **Quaternion** (p. 1029).

Parameters

in	*_pos*	The position.
in	*_rot*	The rotation.

10.173.3.23 void gazebo::math::Pose::Set (const Vector3 & _pos, const Vector3 & _rpy)

Set the pose from pos and rpy vectors.

Parameters

in	*_pos*	The position.
in	*_rpy*	The rotation expressed as Euler angles.

10.173.3.24 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.

10.173.4 Friends And Related Function Documentation

10.173.4.1 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

10.173.4.2 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

10.173.5 Member Data Documentation

10.173.5.1 Vector3 gazebo::math::Pose::pos

The position.

Referenced by gazebo::physics::DARTTypes::ConvPose(), and CoordPositionSub().

10.173.5.2 Quaternion gazebo::math::Pose::rot

The rotation.

Referenced by gazebo::physics::DARTTypes::ConvPose(), CoordPositionSub(), and operator-().

10.173.5.3 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:

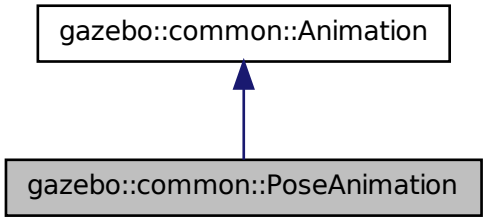
- **Pose.hh**

10.174 gazebo::common::PoseAnimation Class Reference

A pose animation.

`#include <Animation.hh>`

Inheritance diagram for gazebo::common::PoseAnimation:



Public Member Functions

- **PoseAnimation** (const std::string &_name, double _length, bool _loop)

  Constructor.
- virtual ∼**PoseAnimation** ()

  Destructor.
- **PoseKeyFrame** ∗ **CreateKeyFrame** (double _time)

  Create a pose keyframe at the given time.
- void **GetInterpolatedKeyFrame** (**PoseKeyFrame** &_kf) const

  Get a keyframe using the animation's current time.

Protected Member Functions

- void **BuildInterpolationSplines** () const

  Update the pose splines.
- void **GetInterpolatedKeyFrame** (double _time, **PoseKeyFrame** &_kf) const

  Get a keyframe using a passed in time.

10.174.1 Detailed Description

A pose animation.

10.174.2 Constructor & Destructor Documentation

10.174.2.1 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

10.174.2.2 virtual gazebo::common::PoseAnimation::~PoseAnimation ( ) [virtual]

Destructor.

10.174.3 Member Function Documentation

10.174.3.1 void gazebo::common::PoseAnimation::BuildInterpolationSplines ( ) const [protected]

Update the pose splines.

10.174.3.2 PoseKeyFrame* gazebo::common::PoseAnimation::CreateKeyFrame (double *_time*)

Create a pose keyframe at the given time.

Parameters

in	*_time*	**Time** (p. 1361) at which to create the keyframe

Returns

Pointer to the new keyframe

10.174.3.3 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (PoseKeyFrame & *_kf*) const

Get a keyframe using the animation's current time.

Parameters

out	*_kf*	**PoseKeyFrame** (p. 1008) reference to hold the interpolated result

10.174.3.4 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (double *_time,* PoseKeyFrame & *_kf*) const [protected]

Get a keyframe using a passed in time.

Parameters

in	*_time*	**Time** (p. 1361) in seconds
out	*_kf*	**PoseKeyFrame** (p. 1008) reference to hold the interpolated result

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

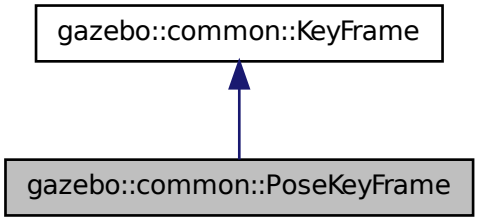
- **Animation.hh**

10.175 gazebo::common::PoseKeyFrame Class Reference

A keyframe for a **PoseAnimation** (p. 1006).

`#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

10.175.1 Detailed Description

A keyframe for a **PoseAnimation** (p. 1006).

10.175.2 Constructor & Destructor Documentation

10.175.2.1 gazebo::common::PoseKeyFrame::PoseKeyFrame (double *_time*)

Constructor.

Parameters

in	*_time*	of the keyframe

10.175.2.2 virtual gazebo::common::PoseKeyFrame::~PoseKeyFrame ( ) [virtual]

Destructor.

10.175.3 Member Function Documentation

10.175.3.1 const math::Quaternion& gazebo::common::PoseKeyFrame::GetRotation ( ) const

Get the rotation of the keyframe.

Returns

The rotation amount

10.175.3.2 const math::Vector3& gazebo::common::PoseKeyFrame::GetTranslation ( ) const

Get the translation of the keyframe.

Returns

The translation amount

10.175.3.3 void gazebo::common::PoseKeyFrame::SetRotation (const math::Quaternion & *_rot*)

Set the rotation for the keyframe.

Parameters

in	*_rot*	Rotation amount

10.175.3.4 void gazebo::common::PoseKeyFrame::SetTranslation (const math::Vector3 & *_trans*)

Set the translation for the keyframe.

Parameters

in	*_trans*	Translation amount

10.175.4 Member Data Documentation

10.175.4.1 math::Quaternion gazebo::common::PoseKeyFrame::rotate [protected]

the rotation quaternion

10.175.4.2 math::Vector3 gazebo::common::PoseKeyFrame::translate [protected]

the translation vector

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

- **KeyFrame.hh**

10.176 gazebo::rendering::Projector Class Reference

Projects a material onto surface, light a light projector.

`#include <rendering/rendering.hh>`

Classes

- class **ProjectorFrameListener**

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.

10.176.1 Detailed Description

Projects a material onto surface, light a light projector.

10.176.2 Constructor & Destructor Documentation

10.176.2.1 gazebo::rendering::Projector::Projector (VisualPtr *_parent*)

Constructor.

Parameters

in	*_parent*	Name of the parent visual.

10.176.2.2 virtual gazebo::rendering::Projector::∼Projector ( ) [virtual]

Destructor.

10.176.3 Member Function Documentation

10.176.3.1 VisualPtr gazebo::rendering::Projector::GetParent ( )

Get the parent visual.

Returns

Pointer to the parent visual.

10.176.3.2 void gazebo::rendering::Projector::Load (sdf::ElementPtr _sdf)

Load from an sdf pointer.

Parameters

in	*_sdf*	Pointer to the SDF element.

10.176.3.3 void gazebo::rendering::Projector::Load (const msgs::Projector & _msg)

Load from a message.

Parameters

in	*_msg*	Load from a message.

10.176.3.4 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	*_texture-Name*	Name of the texture to project.
in	*_nearClip*	Near clip distance.
in	*_farClip*	Far clip distance.
in	*_fov*	Field of view.

10.176.3.5 void gazebo::rendering::Projector::SetEnabled (bool *_enabled*)

Set whether the projector is enabled or disabled.

Parameters

in	*_enabled*	True to enable the projector.

10.176.3.6 void gazebo::rendering::Projector::SetTexture (const std::string & *_textureName*)

Load a texture into the projector.

Parameters

in	*_textureName*	Name of the texture to project.

10.176.3.7 void gazebo::rendering::Projector::Toggle ( )

Toggle the activation of the projector.

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

- **Projector.hh**

10.177 gazebo::transport::Publication Class Reference

A publication for a topic.

`#include <transport/transport.hh>`

Public Member Functions

- **Publication** (const std::string &_topic, const std::string &_msgType)

  Constructor.
- virtual ~**Publication** ()

  Destructor.
- void **AddPublisher** (**PublisherPtr** _pub)

  Add a publisher.
- void **AddSubscription** (const **CallbackHelperPtr** _callback)

Subscribe a callback to our topic.

- void **AddSubscription** (const **NodePtr** &_node)

  Subscribe a node to our topic.
- void **AddTransport** (const **PublicationTransportPtr** &_publink)

  Add a transport.
- unsigned int **GetCallbackCount** () const

  Get the number of callbacks.
- bool **GetLocallyAdvertised** () const

  Was the topic has been advertised from this process?
- std::string **GetMsgType** () const

  Get the type of message.
- unsigned int **GetNodeCount** () const

  Get the number of nodes.
- **MessagePtr GetPrevMsg** (uint32_t _pubId)

  Get a previous message for a publisher.
- 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)
- int **Publish** (**MessagePtr** _msg, boost::function< void(uint32_t)> _cb, uint32_t _id)

  Publish data to remote subscribers.
- void **RemovePublisher** (**PublisherPtr** _pub)

  Remove a publisher.
- 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.
- void **SetPrevMsg** (uint32_t _pubId, **MessagePtr** _msg)

  Set the previous message for a publisher.

10.177.1 Detailed Description

A publication for a topic.

This facilitates transport of messages

10.177.2 Constructor & Destructor Documentation

10.177.2.1 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

10.177.2.2 virtual gazebo::transport::Publication::~Publication ( ) [virtual]

Destructor.

10.177.3 Member Function Documentation

10.177.3.1 void gazebo::transport::Publication::AddPublisher (PublisherPtr *_pub*)

Add a publisher.

Parameters

in,out	*_pub*	Pointer to publisher object to be added

10.177.3.2 void gazebo::transport::Publication::AddSubscription (const CallbackHelperPtr *_callback*)

Subscribe a callback to our topic.

Parameters

in	*_callback*	The callback

10.177.3.3 void gazebo::transport::Publication::AddSubscription (const NodePtr & _node)

Subscribe a node to our topic.

Parameters

in	*_node*	The node

10.177.3.4 void gazebo::transport::Publication::AddTransport (const PublicationTransportPtr & _publink)

Add a transport.

Parameters

in	*_publink*	Pointer to publication transport object to be added

10.177.3.5 unsigned int gazebo::transport::Publication::GetCallbackCount ( ) const

Get the number of callbacks.

Returns

The number of callbacks

10.177.3.6 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

10.177.3.7 std::string gazebo::transport::Publication::GetMsgType ( ) const

Get the type of message.

Returns

The type of message

10.177.3.8 unsigned int gazebo::transport::Publication::GetNodeCount ( ) const

Get the number of nodes.

Returns

The number of nodes

10.177.3.9 MessagePtr gazebo::transport::Publication::GetPrevMsg (uint32_t *_publd*)

Get a previous message for a publisher.

Parameters

in	*_publd*	ID of the publisher.

Returns

Pointer to the previous message. NULL if there is no previous message.

10.177.3.10 unsigned int gazebo::transport::Publication::GetRemoteSubscriptionCount ( )

Get the number of remote subscriptions.

Returns

The number of remote subscriptions

10.177.3.11 unsigned int gazebo::transport::Publication::GetTransportCount ( ) const

Get the number of transports.

Returns

The number of transports

10.177.3.12 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

10.177.3.13 void gazebo::transport::Publication::LocalPublish (const std::string & _data)

Publish data to local subscribers (skip serialization)

Parameters

in	*_data*	The data to be published

10.177.3.14 int gazebo::transport::Publication::Publish (MessagePtr _msg, boost::function< void(uint32_t)> _cb, uint32_t _id)

Publish data to remote subscribers.

Parameters

in	*_msg*	Message to be published
in	*_cb*	Callback to be invoked after publishing is completed

Returns

Number of remote subscribers that will receive the message.

10.177.3.15 void gazebo::transport::Publication::RemovePublisher (PublisherPtr _pub)

Remove a publisher.

Parameters

in	*_pub*	Pointer to publisher object to remove.

10.177.3.16 void gazebo::transport::Publication::RemoveSubscription (const NodePtr & *_node*)

Unsubscribe a node from our topic.

Parameters

in	*_node*	The node

10.177.3.17 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

10.177.3.18 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

10.177.3.19 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

10.177.3.20 void gazebo::transport::Publication::SetPrevMsg (uint32_t *_pubId*, MessagePtr *_msg*)

Set the previous message for a publisher.

Parameters

in	*_pubId*	ID of the publisher.
in	*_msg*	The previous message.

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

- **Publication.hh**

10.178 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.

10.178.1 Detailed Description

transport/transport.hh

Reads data from a remote advertiser, and passes the data along to local subscribers

10.178.2 Constructor & Destructor Documentation

10.178.2.1 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

10.178.2.2 virtual gazebo::transport::PublicationTransport::~PublicationTransport ( ) [virtual]

Destructor.

10.178.3 Member Function Documentation

10.178.3.1 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

10.178.3.2 void gazebo::transport::PublicationTransport::Fini ( )

Finalize the transport.

10.178.3.3 const ConnectionPtr gazebo::transport::PublicationTransport::GetConnection ( ) const

Get the underlying connection.

Returns

Pointer to the underlying connection

10.178.3.4 std::string gazebo::transport::PublicationTransport::GetMsgType ( ) const

Get the topic type.

Returns

The topic type

10.178.3.5 std::string gazebo::transport::PublicationTransport::GetTopic ( ) const

Get the topic name.

Returns

The topic name

10.178.3.6 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**

10.179 gazebo::transport::Publisher Class Reference

A publisher of messages on a topic.

`#include <transport/transport.hh>`

Public Member Functions

- **Publisher** (const std::string &_topic, const std::string &_msgType, unsigned int _limit, double _hzRate)

  Constructor.
- virtual ~**Publisher** ()

  Destructor.
- void **Fini** ()

  Finalize the publisher.
- 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 **SetNode** (**NodePtr** _node)

  Set our containing node.
- 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.
- bool **WaitForConnection** (const **common::Time** &_timeout) const

  Block until a connection has been established with this publisher.

10.179.1 Detailed Description

A publisher of messages on a topic.

10.179.2 Constructor & Destructor Documentation

10.179.2.1 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.

10.179.2.2 virtual gazebo::transport::Publisher::~Publisher ( ) [virtual]

Destructor.

10.179.3 Member Function Documentation

10.179.3.1 void gazebo::transport::Publisher::Fini ( )

Finalize the publisher.

10.179.3.2 std::string gazebo::transport::Publisher::GetMsgType ( ) const

Get the message type.

Returns

The message type

10.179.3.3 unsigned int gazebo::transport::Publisher::GetOutgoingCount ( ) const

Get the number of outgoing messages.

Returns

The number of outgoing messages

10.179.3.4 std::string gazebo::transport::Publisher::GetPrevMsg ( ) const

Get the previously published message.

Returns

The previously published message, if any

10.179.3.5 MessagePtr gazebo::transport::Publisher::GetPrevMsgPtr ( ) const

Get the previously published message.

Returns

The previously published message, if any

10.179.3.6 std::string gazebo::transport::Publisher::GetTopic ( ) const

Get the topic name.

Returns

The topic name

10.179.3.7 bool gazebo::transport::Publisher::HasConnections ( ) const

Are there any connections?

Returns

true if there are any connections, false otherwise

10.179.3.8 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

10.179.3.9 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

10.179.3.10 void gazebo::transport::Publisher::SendMessage ( )

Send latest message over the wire. For internal use only.

10.179.3.11 void gazebo::transport::Publisher::SetNode (NodePtr *_node*)

Set our containing node.

Parameters

in	*_node*	Pointer to a node. Should be the node that create this publisher.

10.179.3.12 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

10.179.3.13 void gazebo::transport::Publisher::WaitForConnection ( ) const

Block until a connection has been established with this publisher.

10.179.3.14 bool gazebo::transport::Publisher::WaitForConnection (const common::Time & *_timeout*) const

Block until a connection has been established with this publisher.

Parameters

in	*_timeout*	Maxiumum time to wait. Use a negative time value to wait forever.

Returns

True if a connection was established.

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

- **Publisher.hh**

10.180 gazebo::transport::PublishTask Class Reference

`#include <Node.hh>`

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.

10.180.1 Detailed Description

Task used by **Node::Publish** (p. 916) to publish on a one-time publisher

10.180.2 Constructor & Destructor Documentation

10.180.2.1 gazebo::transport::PublishTask::PublishTask (transport::PublisherPtr *_pub*, const google::protobuf::Message & *_message*) [inline]

Constructor.

Parameters

in	*_pub*	**Publisher** (p. 1024) to publish the message on.
in	*_message*	Message to publish

10.180.3 Member Function Documentation

10.180.3.1 tbb::task* gazebo::transport::PublishTask::execute ( ) [inline]

Overridden function from tbb::task that exectues the publish task.

References NULL.

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

- **Node.hh**

10.181 gazebo::math::Quaternion Class Reference

A quaternion class.

`#include <math/gzmath.hh>`

Public Member Functions

- **Quaternion** ()

  Default Constructor.
- **Quaternion** (const double &_w, const double &_x, const double &_y, const double &_z)

  Constructor.
- **Quaternion** (const double &_roll, const double &_pitch, const double &_yaw)

  Constructor from Euler angles in radians.
- **Quaternion** (const **Vector3** &_axis, const double &_angle)

  Constructor from axis angle.
- **Quaternion** (const **Vector3** &_rpy)

  Constructor.
- **Quaternion** (const **Quaternion** &_qt)

  Copy constructor.
- ~**Quaternion** ()

  Destructor.
- void **Correct** ()

  Correct any nan.

- double **Dot** (const **Quaternion** &_q) const

  Dot product.
- void **GetAsAxis** (**Vector3** &_axis, double &_angle) const

  Return rotation as axis and angle.
- **Vector3 GetAsEuler** () const

  Return the rotation in Euler angles.
- **Matrix3 GetAsMatrix3** () const

  Get the quaternion as a 3x3 matrix.
- **Matrix4 GetAsMatrix4** () const

  Get the quaternion as a 4x4 matrix.
- **Quaternion GetExp** () const

  Return the exponent.
- **Quaternion GetInverse** () const

  Get the inverse of this quaternion.
- **Quaternion GetLog** () const

  Return the logarithm.
- double **GetPitch** ()

  Get the Euler pitch angle in radians.
- double **GetRoll** ()

  Get the Euler roll angle in radians.
- **Vector3 GetXAxis** () const

  Return the X axis.
- double **GetYaw** ()

  Get the Euler yaw angle in radians.
- **Vector3 GetYAxis** () const

  Return the Y axis.
- **Vector3 GetZAxis** () const

  Return the Z axis.
- void **Invert** ()

  Invert the quaternion.
- bool **IsFinite** () const

  See if a quatern is finite (e.g., not nan)
- void **Normalize** ()

  Normalize the quaternion.
- bool **operator!=** (const **Quaternion** &_qt) const

  Not equal to operator.
- **Quaternion operator*** (const **Quaternion** &_q) const

  Multiplication operator.
- **Quaternion operator*** (const double &_f) const

Multiplication operator.

- **Vector3 operator∗** (const **Vector3** &_v) const

  Vector3 *(*p. *1440) 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.

10.181.1 Detailed Description

A quaternion class.

10.181.2 Constructor & Destructor Documentation

10.181.2.1 gazebo::math::Quaternion::Quaternion ( )

Default Constructor.

10.181.2.2 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

10.181.2.3 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

10.181.2.4 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

10.181.2.5 gazebo::math::Quaternion::Quaternion (const Vector3 & _rpy)

Constructor.

Parameters

in	*_rpy*	euler angles

10.181.2.6 gazebo::math::Quaternion::Quaternion (const Quaternion & _qt)

Copy constructor.

Parameters

qt	**Quaternion** (p. 1029) to copy

10.181.2.7 gazebo::math::Quaternion::~Quaternion ( )

Destructor.

10.181.3 Member Function Documentation

10.181.3.1 void gazebo::math::Quaternion::Correct ( ) [inline]

Correct any nan.

References gazebo::math::equal().

10.181.3.2 double gazebo::math::Quaternion::Dot (const Quaternion & _q) const

Dot product.

Parameters

in	*_q*	the other quaternion

Returns

the product

10.181.3.3 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (const Vector3 & _vec) [static]

Convert euler angles to quatern.

Parameters

in		

10.181.3.4 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

10.181.3.5 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

10.181.3.6 Vector3 gazebo::math::Quaternion::GetAsEuler ( ) const

Return the rotation in Euler angles.

Returns

This quaternion as an Euler vector

10.181.3.7 Matrix3 gazebo::math::Quaternion::GetAsMatrix3 ( ) const

Get the quaternion as a 3x3 matrix.

10.181.3.8 Matrix4 gazebo::math::Quaternion::GetAsMatrix4 ( ) const

Get the quaternion as a 4x4 matrix.

Returns

a 4x4 matrix

10.181.3.9 Quaternion gazebo::math::Quaternion::GetExp ( ) const

Return the exponent.

Returns

the exp

10.181.3.10 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(), and gazebo::math::Pose::-CoordRotationSub().

10.181.3.11 Quaternion gazebo::math::Quaternion::GetLog ( ) const

Return the logarithm.

Returns

the log

10.181.3.12 double gazebo::math::Quaternion::GetPitch ( )

Get the Euler pitch angle in radians.

Returns

the pitch

10.181.3.13 double gazebo::math::Quaternion::GetRoll ( )

Get the Euler roll angle in radians.

Returns

the roll

10.181.3.14 Vector3 gazebo::math::Quaternion::GetXAxis ( ) const

Return the X axis.

Returns

the vector

10.181.3.15 double gazebo::math::Quaternion::GetYaw ( )

Get the Euler yaw angle in radians.

Returns

the yaw

10.181.3.16 Vector3 gazebo::math::Quaternion::GetYAxis ( ) const

Return the Y axis.

Returns

the vector

10.181.3.17 Vector3 gazebo::math::Quaternion::GetZAxis ( ) const

Return the Z axis.

Returns

the vector

10.181.3.18 void gazebo::math::Quaternion::Invert ( )

Invert the quaternion.

10.181.3.19 bool gazebo::math::Quaternion::IsFinite ( ) const

See if a quatern is finite (e.g., not nan)

Returns

True if quatern is finite

10.181.3.20 void gazebo::math::Quaternion::Normalize ( )

Normalize the quaternion.

Referenced by gazebo::math::Pose::CoordRotationSub().

10.181.3.21 bool gazebo::math::Quaternion::operator!= (const Quaternion & *_qt*) const

Not equal to operator.

Parameters

in	*_qt*	**Quaternion** (p. 1029) for comparison

Returns

True if not equal

10.181.3.22 Quaternion gazebo::math::Quaternion::operator* (const Quaternion & *_q*) const [inline]

Multiplication operator.

Parameters

in	*_qt*	**Quaternion** (p. 1029) for multiplication

Returns

This quaternion multiplied by the parameter

References w, x, y, and z.

10.181.3.23 Quaternion gazebo::math::Quaternion::operator∗ (const double & _f) const

Multiplication operator.

Parameters

in	*_f*	factor

Returns

quaternion multiplied by _f

10.181.3.24 Vector3 gazebo::math::Quaternion::operator∗ (const Vector3 & _v) const

Vector3 (p. 1440) multiplication operator.

Parameters

in	*_v*	vector to multiply

10.181.3.25 Quaternion gazebo::math::Quaternion::operator∗= (const Quaternion & *qt*)

Multiplication operator.

Parameters

in	*_qt*	**Quaternion** (p. 1029) for multiplication

Returns

This quatern multiplied by the parameter

10.181.3.26 Quaternion gazebo::math::Quaternion::operator+ (const Quaternion & _qt) const

Addition operator.

Parameters

in	*_qt*	quaternion for addition

Returns

this quaternion + _qt

10.181.3.27 Quaternion gazebo::math::Quaternion::operator+= (const Quaternion & *_qt*)

Addition operator.

Parameters

in	*_qt*	quaternion for addition

Returns

this quaternion + qt

10.181.3.28 Quaternion gazebo::math::Quaternion::operator- (const Quaternion & *_qt*) const

Substraction operator.

Parameters

in	*_qt*	quaternion to substract

Returns

this quaternion - _qt

10.181.3.29 Quaternion gazebo::math::Quaternion::operator- ( ) const

Unary minus operator.

Returns

negates each component of the quaternion

10.181.3.30 Quaternion gazebo::math::Quaternion::operator-= (const Quaternion & _qt)

Substraction operator.

Parameters

in	*_qt*	**Quaternion** (p. 1029) for substraction

Returns

This quatern - qt

10.181.3.31 Quaternion& gazebo::math::Quaternion::operator= (const Quaternion & _qt)

Equal operator.

Parameters

in	*_qt*	**Quaternion** (p. 1029) to copy

10.181.3.32 bool gazebo::math::Quaternion::operator== (const Quaternion & _qt) const

Equal to operator.

Parameters

in	*_qt*	**Quaternion** (p. 1029) for comparison

Returns

True if equal

10.181.3.33 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 gazebo::math::Vector3::x, x, gazebo::math::Vector3::y, y, gazebo::math::-Vector3::z, and z.

10.181.3.34 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

10.181.3.35 void gazebo::math::Quaternion::Round (int *_precision*)

Round all values to _precision decimal places.

Parameters

in	*_precision*	the precision

10.181.3.36 void gazebo::math::Quaternion::Scale (double *_scale*)

Scale a Quaternionion.

Parameters

in	*_scale*	Amount to scale this rotation

10.181.3.37 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

10.181.3.38 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. 173) in radians

10.181.3.39 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. 173) in radians

10.181.3.40 void gazebo::math::Quaternion::SetFromEuler (const Vector3 & _vec)

Set the quaternion from Euler angles.

The order of operations are roll, pitch, yaw.

Parameters

in	*vec*	Euler angle

10.181.3.41 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).

10.181.3.42 void gazebo::math::Quaternion::SetToIdentity ( )

Set the quatern to the identity.

10.181.3.43 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

Referenced by gazebo::math::OnePoleQuaternion::Process().

10.181.3.44 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

10.181.4 Friends And Related Function Documentation

10.181.4.1 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

10.181.4.2 std::istream& operator>> (std::istream & *_in*, gazebo::math::Quaternion & *_q*) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_q*	**Quaternion** (p. 1029) to read values into

Returns

The istream

10.181.5 Member Data Documentation

10.181.5.1 double gazebo::math::Quaternion::w

Attributes of the quaternion.

Referenced by gazebo::physics::DARTTypes::ConvQuat(), GetInverse(), and operator∗().

10.181.5.2 double gazebo::math::Quaternion::x

Attributes of the quaternion.

Referenced by gazebo::physics::DARTTypes::ConvQuat(), gazebo::math::Pose::CoordPositionSub(), GetInverse(), operator∗(), and RotateVector().

10.181.5.3 double gazebo::math::Quaternion::y

Attributes of the quaternion.

Referenced by gazebo::physics::DARTTypes::ConvQuat(), gazebo::math::Pose::CoordPositionSub(), GetInverse(), operator∗(), and RotateVector().

10.181.5.4 double gazebo::math::Quaternion::z

Attributes of the quaternion.

Referenced by gazebo::physics::DARTTypes::ConvQuat(), gazebo::math::Pose::CoordPositionSub(), GetInverse(), operator∗(), and RotateVector().

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

- **Quaternion.hh**

10.182 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.

10.182.1 Detailed Description

Random number generator class.

10.182.2 Member Function Documentation

10.182.2.1 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

10.182.2.2 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

10.182.2.3 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

10.182.2.4 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

10.182.2.5 static uint32_t gazebo::math::Rand::GetSeed ( ) [static]

Get the seed value.

Returns

The seed value used to initialize the random number generator.

10.182.2.6 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:

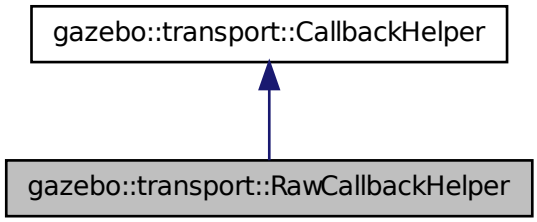
- **Rand.hh**

10.183 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, boost::function< void(uint32_t)> _cb, uint32_t _id)

  Process new incoming data.
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)

  Process new incoming message.
- virtual bool **IsLocal** () const

  Is the callback local?

10.183.1 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.

10.183.2 Constructor & Destructor Documentation

10.183.2.1 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.

10.183.3 Member Function Documentation

10.183.3.1 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. 237).

10.183.3.2 virtual bool gazebo::transport::RawCallbackHelper::HandleData (const std::string & _newdata, boost::function< void(uint32_t)> _cb, uint32_t _id) [inline, virtual]

Process new incoming data.

Parameters

in	*_newdata*	Incoming data to be processed

Returns

true if successfully processed; false otherwise

Parameters

in	*_cb*	If non-null, callback to be invoked which signals that transmission is complete.
in	*_id*	ID associated with the message data.

Implements **gazebo::transport::CallbackHelper** (p. 238).

10.183.3.3 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. 238).

10.183.3.4 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. 239).

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

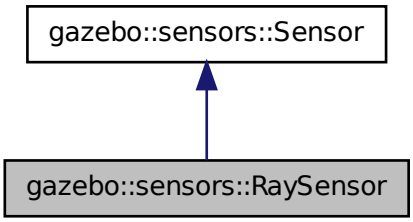
- **CallbackHelper.hh**

10.184 gazebo::sensors::RaySensor Class Reference

Sensor (p. 1130) 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** (unsigned int _index)

  Get detected fiducial value for a ray.
- **physics::MultiRayShapePtr GetLaserShape** () const

  Returns a pointer to the internal ***physics::MultiRayShape*** *(*p. 901*).*
- double **GetRange** (unsigned 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** (unsigned 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.
- double **GetVerticalAngleResolution** () const

  Get the vertical angle in radians between each range.
- 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 bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

10.184.1 Detailed Description

Sensor (p. 1130) 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).

10.184.2 Constructor & Destructor Documentation

10.184.2.1 gazebo::sensors::RaySensor::RaySensor ( )

Constructor.

10.184.2.2 virtual gazebo::sensors::RaySensor::~RaySensor ( ) [virtual]

Destructor.

10.184.3 Member Function Documentation

10.184.3.1 virtual void gazebo::sensors::RaySensor::Fini ( ) [protected, virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.184.3.2 math::Angle gazebo::sensors::RaySensor::GetAngleMax ( ) const

Get the maximum angle.

Returns

the maximum angle object

10.184.3.3 math::Angle gazebo::sensors::RaySensor::GetAngleMin ( ) const

Get the minimum angle.

Returns

The minimum angle object

10.184.3.4 double gazebo::sensors::RaySensor::GetAngleResolution ( ) const

Get the angle in radians between each range.

Returns

Resolution of the angle

10.184.3.5 int gazebo::sensors::RaySensor::GetFiducial (unsigned 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

10.184.3.6 physics::MultiRayShapePtr gazebo::sensors::RaySensor::GetLaserShape ( ) const [inline]

Returns a pointer to the internal **physics::MultiRayShape** (p. 901).

Returns

Pointer to ray shape

10.184.3.7 double gazebo::sensors::RaySensor::GetRange (unsigned 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.

10.184.3.8 int gazebo::sensors::RaySensor::GetRangeCount ( ) const

Get the range count.

Returns

The number of ranges

10.184.3.9 double gazebo::sensors::RaySensor::GetRangeMax ( ) const

Get the maximum range.

Returns

The maximum range

10.184.3.10 double gazebo::sensors::RaySensor::GetRangeMin ( ) const

Get the minimum range.

Returns

The minimum range

10.184.3.11 double gazebo::sensors::RaySensor::GetRangeResolution ( ) const

Get the range resolution.

Returns

Resolution of the range

10.184.3.12 void gazebo::sensors::RaySensor::GetRanges (std::vector< double > & *_ranges*)

Get all the ranges.

Parameters

_ranges	A vector that will contain all the range data

10.184.3.13 int gazebo::sensors::RaySensor::GetRayCount ( ) const

Get the ray count.

Returns

The number of rays

10.184.3.14 double gazebo::sensors::RaySensor::GetRetro (unsigned 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

10.184.3.15 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. 1138).

10.184.3.16 math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMax ( ) const

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.184.3.17 math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMin ( ) const

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.184.3.18 double gazebo::sensors::RaySensor::GetVerticalAngleResolution ( ) const

Get the vertical angle in radians between each range.

Returns

Resolution of the angle

10.184.3.19 int gazebo::sensors::RaySensor::GetVerticalRangeCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.184.3.20 int gazebo::sensors::RaySensor::GetVerticalRayCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.184.3.21 virtual void gazebo::sensors::RaySensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.184.3.22 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. 1139).

10.184.3.23 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. 1140).

10.184.3.24 virtual bool gazebo::sensors::RaySensor::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

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

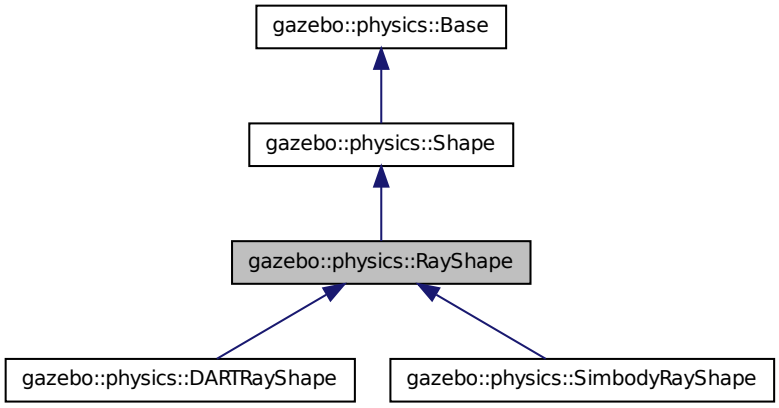
- **RaySensor.hh**

10.185 gazebo::physics::RayShape Class Reference

Base (p. 201) 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 **SetScale** (const **math::Vector3** &_scale)

  Set the scale of the 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.

10.185.1 Detailed Description

Base (p. 201) class for Ray collision geometry.

10.185.2 Constructor & Destructor Documentation

10.185.2.1 gazebo::physics::RayShape::RayShape (PhysicsEnginePtr *_physicsEngine*) [explicit]

Constructor for a global ray.

Parameters

in	*_physicsEngine*	Pointer to the physics engine.

10.185.2.2 gazebo::physics::RayShape::RayShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	**Collision** (p. 295) parent of the shape.

10.185.2.3 virtual gazebo::physics::RayShape::~RayShape ( ) [virtual]

Destructor.

10.185.3 Member Function Documentation

10.185.3.1 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. 1163).

10.185.3.2 int gazebo::physics::RayShape::GetFiducial ( ) const

Get the fiducial id detected by this ray.

Returns

Fiducial id detected.

10.185.3.3 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.

10.185.3.4 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.

Implemented in **gazebo::physics::DARTRayShape** (p. 95), and **gazebo::physics::SimbodyRayShape** (p. 1242).

10.185.3.5 double gazebo::physics::RayShape::GetLength ( ) const

Get the length of the ray.

Returns

The ray length.

10.185.3.6 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.

10.185.3.7 float gazebo::physics::RayShape::GetRetro ( ) const

Get the retro-reflectivness detected by this ray.

Returns

Retro reflectance value.

10.185.3.8 virtual void gazebo::physics::RayShape::Init ( ) [virtual]

In the ray.

Implements **gazebo::physics::Shape** (p. 1164).

10.185.3.9 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. 1164).

10.185.3.10 void gazebo::physics::RayShape::SetFiducial (int *_fid*)

Set the fiducial id detected by this ray.

Parameters

in	*_fid*	Fiducial id detected by this ray.

10.185.3.11 virtual void gazebo::physics::RayShape::SetLength (double *_len*) [virtual]

Set the length of the ray.

Parameters

in	*_len*	Length of the array.

10.185.3.12 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.

Reimplemented in **gazebo::physics::DARTRayShape** (p. 95), and **gazebo::physics::SimbodyRayShape** (p. 1243).

10.185.3.13 void gazebo::physics::RayShape::SetRetro (float *_retro*)

Set the retro-reflectivness detected by this ray.

Parameters

in	*_retro*	Retro reflectance value.

10.185.3.14 virtual void gazebo::physics::RayShape::SetScale (const math::Vector3 & *_scale*) [virtual]

Set the scale of the ray.

Implements **gazebo::physics::Shape** (p. 1165).

10.185.3.15 virtual void gazebo::physics::RayShape::Update ( ) [pure virtual]

Update the ray collision.

Reimplemented from **gazebo::physics::Base** (p. 216).

Implemented in **gazebo::physics::DARTRayShape** (p. 95), and **gazebo::physics::SimbodyRayShape** (p. 1243).

10.185.4 Member Data Documentation

10.185.4.1 int gazebo::physics::RayShape::contactFiducial [protected]

Fiducial ID value.

10.185.4.2 double gazebo::physics::RayShape::contactLen [protected]

Length of the ray.

10.185.4.3 double gazebo::physics::RayShape::contactRetro [protected]

Retro reflectance value.

10.185.4.4 math::Vector3 gazebo::physics::RayShape::globalEndPos [protected]

End position of the ray in global cs.

10.185.4.5 math::Vector3 gazebo::physics::RayShape::globalStartPos [protected]

Start position of the ray in global cs.

10.185.4.6 math::Vector3 gazebo::physics::RayShape::relativeEndPos [protected]

End position of the ray, relative to the body.

10.185.4.7 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:

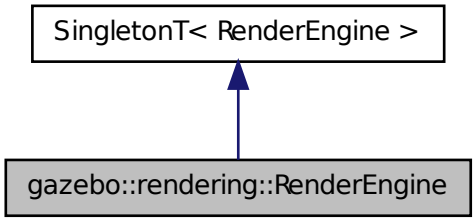
- **RayShape.hh**

10.186 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. 163) to search for resources.*
- **ScenePtr CreateScene** (const std::string &_name, bool _enableVisualizations, bool _isServer=false)

  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.

- **WindowManagerPtr GetWindowManager** () const

  Get a pointer to the window manager.

- void **Init** ()

  *Initialize **Ogre*** (p. *163). Load must happen before Init.*

- void **Load** ()

  *Load the parameters for **Ogre*** (p. *163). 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.

10.186.1 Detailed Description

Adaptor to Ogre3d.

Provides the interface to load, initialize the rendering engine.

10.186.2 Member Enumeration Documentation

10.186.2.1 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.

10.186.3 Member Function Documentation

10.186.3.1 void gazebo::rendering::RenderEngine::AddResourcePath (const std::string & *_uri*)

Add a new path for **Ogre** (p. 163) to search for resources.

Parameters

in	*_uri*	URI of the path. The uri should be of the form `file://` or model://

10.186.3.2 ScenePtr gazebo::rendering::RenderEngine::CreateScene (const std::string & *_name,* bool *_enableVisualizations,* bool *_isServer* = `false`)

Create a scene.

Parameters

in	*_name*	The name of the scene.
in	*_enable-Visualizations*	True enables visualization elements such as laser lines.

10.186.3.3 void gazebo::rendering::RenderEngine::Fini ( )

Tears down the rendering engine.

10.186.3.4 RenderPathType gazebo::rendering::RenderEngine::GetRenderPath-Type ( ) const

Get the type of rendering path to use.

This is automatically determined based on the computers capabilities

Returns

The RenderPathType

10.186.3.5 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 pointer to the **Scene** (p. 1097), or NULL if the scene doesn't exist.

10.186.3.6 ScenePtr gazebo::rendering::RenderEngine::GetScene (unsigned int _index)

Get a scene by index.

The index should be between 0 and **GetSceneCount()** (p. 1070).

Parameters

in	*_index*	The index of the scene.

Returns

A pointer to a **Scene** (p. 1097), or NULL if the index was invalid.

10.186.3.7 unsigned int gazebo::rendering::RenderEngine::GetSceneCount ( ) const

Get the number of scenes.

Returns

The number of scenes created by the **RenderEngine** (p. 1067).

10.186.3.8 WindowManagerPtr gazebo::rendering::RenderEngine::GetWindowManager ( ) const

Get a pointer to the window manager.

Returns

Pointer to the window manager.

10.186.3.9 void gazebo::rendering::RenderEngine::Init ( )

Initialize **Ogre** (p. 163). Load must happen before Init.

10.186.3.10 void gazebo::rendering::RenderEngine::Load ( )

Load the parameters for **Ogre** (p. 163). Load must happen before Init.

10.186.3.11 void gazebo::rendering::RenderEngine::RemoveScene (const std::string & *_name*)

Remove a scene.

Parameters

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

10.186.4 Member Data Documentation

10.186.4.1 void* gazebo::rendering::RenderEngine::dummyContext [protected]

GLX context used to render the scenes.Used for gui-less operation.

10.186.4.2 void* gazebo::rendering::RenderEngine::dummyDisplay [protected]

Pointer to the dummy display.Used for gui-less operation.

10.186.4.3 uint64_t gazebo::rendering::RenderEngine::dummyWindowId [protected]

ID for a dummy window. Used for gui-less operation.

10.186.4.4 Ogre::Root* gazebo::rendering::RenderEngine::root

Pointer to the root scene node.

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

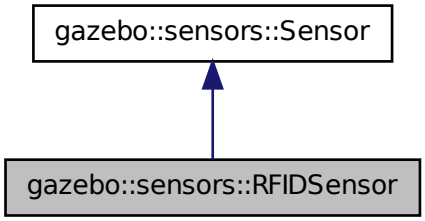
- **RenderEngine.hh**

10.187 gazebo::sensors::RFIDSensor Class Reference

Sensor (p. 1130) 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 bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

10.187.1 Detailed Description

Sensor (p. 1130) class for RFID type of sensor.

10.187.2 Constructor & Destructor Documentation

10.187.2.1 gazebo::sensors::RFIDSensor::RFIDSensor ( )

Constructor.

10.187.2.2 virtual gazebo::sensors::RFIDSensor::~RFIDSensor ( ) [virtual]

Destructor.

10.187.3 Member Function Documentation

10.187.3.1 void gazebo::sensors::RFIDSensor::AddTag (RFIDTag * *_tag*)

10.187.3.2 virtual void gazebo::sensors::RFIDSensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.187.3.3 virtual void gazebo::sensors::RFIDSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.187.3.4 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. 1130) parameters.
in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 1140).

10.187.3.5 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. 1140).

10.187.3.6 virtual bool gazebo::sensors::RFIDSensor::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

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

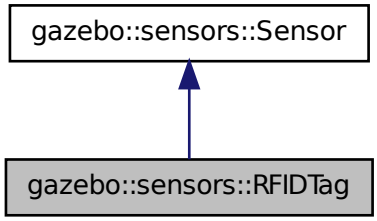
- **RFIDSensor.hh**

10.188 gazebo::sensors::RFIDTag Class Reference

RFIDTag (p. 1075) 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)

  Load the sensor with SDF parameters.

- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

Protected Member Functions

- virtual bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

10.188.1 Detailed Description

RFIDTag (p. 1075) to interact with RFIDTagSensors.

10.188.2 Constructor & Destructor Documentation

10.188.2.1 gazebo::sensors::RFIDTag::RFIDTag ( )

Constructor.

10.188.2.2 virtual gazebo::sensors::RFIDTag::~RFIDTag ( ) [virtual]

Destructor.

10.188.3 Member Function Documentation

10.188.3.1 virtual void gazebo::sensors::RFIDTag::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.188.3.2 math::Pose gazebo::sensors::RFIDTag::GetTagPose ( ) const [inline]

Returns pose of tag in world coordinate.

Returns

Pose of object.

10.188.3.3 virtual void gazebo::sensors::RFIDTag::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

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

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 1130) parameters.
in	*_worldName*	Name of world to load from.

Reimplemented from **gazebo::sensors::Sensor** (p. 1140).

10.188.3.5 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. 1140).

10.188.3.6 virtual bool gazebo::sensors::RFIDTag::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

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

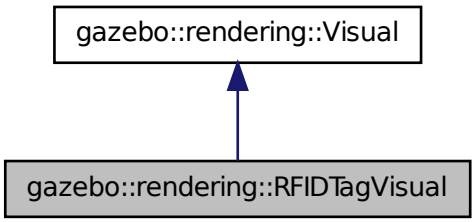
- **RFIDTag.hh**

10.189 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.

10.189.1 Detailed Description

Visualization for RFID tags sensor.

10.189.2 Constructor & Destructor Documentation

10.189.2.1 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. 1072)

10.189.2.2 virtual gazebo::rendering::RFIDTagVisual::~RFIDTagVisual ( ) [virtual]

Destructor.

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

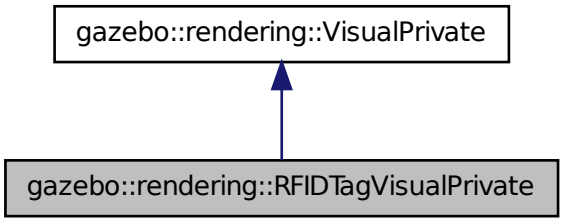
- **RFIDTagVisual.hh**

10.190 gazebo::rendering::RFIDTagVisualPrivate Class Reference

Private data for the RFID Tag **Visual** (p. 1477) class.

`#include <RFIDTagVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::RFIDTagVisualPrivate:



Public Attributes

- **transport::NodePtr node**

  Node that handles communication.
- **transport::SubscriberPtr rfidSub**

  Subscriber that receives RFID data.

10.190.1 Detailed Description

Private data for the RFID Tag **Visual** (p. 1477) class.

10.190.2 Member Data Documentation

10.190.2.1 transport::NodePtr gazebo::rendering::RFIDTagVisualPrivate::node

Node that handles communication.

10.190.2.2 transport::SubscriberPtr gazebo::rendering::RFIDTagVisualPrivate-::rfidSub

Subscriber that receives RFID data.

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

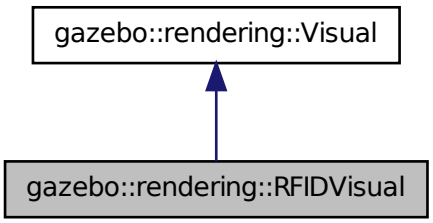
- **RFIDTagVisualPrivate.hh**

10.191 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.

10.191.1 Detailed Description

Visualization for RFID sensor.

10.191.2 Constructor & Destructor Documentation

10.191.2.1 gazebo::rendering::RFIDVisual::RFIDVisual (const std::string & *_name*, VisualPtr *_vis*, const std::string & *_topicName*)

Constructor.

Parameters

in	*_name*	Name of the **Visual** (p. 1477).
in	*_vis*	Parent **Visual** (p. 1477).
in	*_topicName*	Name of the topic which publishes RFID data.

10.191.2.2 virtual gazebo::rendering::RFIDVisual::~RFIDVisual ( ) [virtual]

Destructor.

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

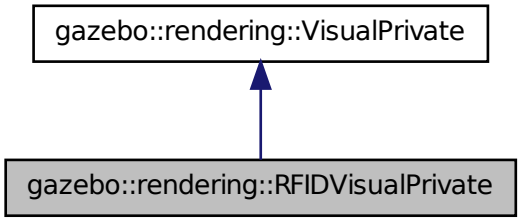
- **RFIDVisual.hh**

10.192 gazebo::rendering::RFIDVisualPrivate Class Reference

Private data for the RFID **Visual** (p. 1477) class.

`#include <RFIDVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::RFIDVisualPrivate:



Public Attributes

- **transport::NodePtr node**

  *Pointer to the **transport::Node** (p. 910) for communication.*
- **transport::SubscriberPtr rfidSub**

  *Pointer to the **transport::Subscriber** (p. 1346) for recieving data.*

10.192.1 Detailed Description

Private data for the RFID **Visual** (p. 1477) class.

10.192.2 Member Data Documentation

10.192.2.1 transport::NodePtr gazebo::rendering::RFIDVisualPrivate::node

Pointer to the **transport::Node** (p. 910) for communication.

10.192.2.2 transport::SubscriberPtr gazebo::rendering::RFIDVisualPrivate::rfidSub

Pointer to the **transport::Subscriber** (p. 1346) for recieving data.

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

- **RFIDVisualPrivate.hh**

10.193 Road Class Reference

Used to render a strip of road.

`#include <rendering/rendering.hh>`

10.193.1 Detailed Description

Used to render a strip of road.

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

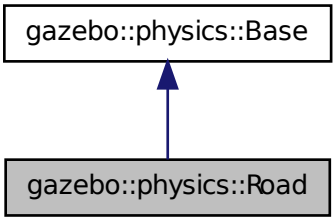
- **Road2d.hh**

10.194 gazebo::physics::Road Class Reference

for building a **Road** (p. 1083) from SDF

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Road:



Public Member Functions

- **Road** (**BasePtr** _parent)
  Constructor.
- virtual ~**Road** ()
  Destructor.
- const std::vector < **math::Vector3** > & **GetPoints** () const
  Get the point that define the road.
- double **GetWidth** () const
  Get the road width in meters.
- virtual void **Init** ()
  Initialize the road.
- void **Load** (sdf::ElementPtr _sdf)
  Load the road from SDF.

10.194.1 Detailed Description

for building a **Road** (p. 1083) from SDF

10.194.2 Constructor & Destructor Documentation

10.194.2.1 gazebo::physics::Road::Road (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of this road object.

10.194.2.2 virtual gazebo::physics::Road::~Road ( ) [virtual]

Destructor.

10.194.3 Member Function Documentation

10.194.3.1 const std::vector<math::Vector3>& gazebo::physics::Road::GetPoints ( ) const

Get the point that define the road.

Returns

The vector of points that define the road.

10.194.3.2 double gazebo::physics::Road::GetWidth ( ) const

Get the road width in meters.

Returns

Road (p. 1083) width in meters.

10.194.3.3 virtual void gazebo::physics::Road::Init ( ) [virtual]

Initialize the road.

Reimplemented from **gazebo::physics::Base** (p. 211).

10.194.3.4 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. 212).

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

- **Road.hh**

10.195 gazebo::rendering::Road2d Class Reference

`#include <Road2d.hh>`

Classes

- class **Segment**

  A road segment.

Public Member Functions

- **Road2d** ()

  Constructor.
- virtual ~**Road2d** ()

  Destructor.
- void **Load** (**VisualPtr** _parent)

  Load the visual using a parent visual.

10.195.1 Constructor & Destructor Documentation

10.195.1.1 gazebo::rendering::Road2d::Road2d ( )

Constructor.

10.195.1.2 virtual gazebo::rendering::Road2d::~Road2d ( ) [virtual]

Destructor.

10.195.2 Member Function Documentation

10.195.2.1 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**

10.196 gazebo::math::RotationSpline Class Reference

Spline (p. 1317) 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

10.196.1 Detailed Description

Spline (p. 1317) for rotations.

10.196.2 Constructor & Destructor Documentation

10.196.2.1 gazebo::math::RotationSpline::RotationSpline ( )

Constructor. Sets the autoCalc to true.

10.196.2.2 gazebo::math::RotationSpline::~RotationSpline ( )

Destructor. Nothing is done.

10.196.3 Member Function Documentation

10.196.3.1 void gazebo::math::RotationSpline::AddPoint (const Quaternion & _p)

Adds a control point to the end of the spline.

Parameters

in	*_p*	control point

10.196.3.2 void gazebo::math::RotationSpline::Clear ( )

Clears all the points in the spline.

10.196.3.3 unsigned int gazebo::math::RotationSpline::GetNumPoints ( ) const

Gets the number of control points in the spline.

Returns

the count

10.196.3.4 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)

10.196.3.5 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	*_useShortestPath*	Defines if rotation should take the shortest possible path

Returns

the rotation

10.196.3.6 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`	*_useShortestPath*	Defines if rotation should take the shortest possible path

Returns

the rotation

10.196.3.7 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.

10.196.3.8 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.

10.196.3.9 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

10.196.4 Member Data Documentation

10.196.4.1 bool gazebo::math::RotationSpline::autoCalc [protected]

Automatic recalcultation of tangeants when control points are updated.

10.196.4.2 std::vector<Quaternion> gazebo::math::RotationSpline::points [protected]

the control points

10.196.4.3 std::vector<Quaternion> gazebo::math::RotationSpline::tangents [protected]

the tangents

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

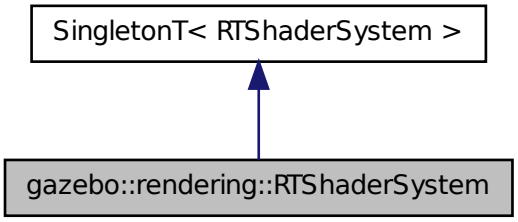
- **RotationSpline.hh**

10.197 gazebo::rendering::RTShaderSystem Class Reference

Implements **Ogre** (p. 163)'s Run-Time Shader system.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RTShaderSystem:



Public Types

- enum **LightingModel** { **SSLM_PerVertexLighting**, **SSLM_PerPixelLighting**, **SSLM_NormalMapLightingTangentSpace**, **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. 163) 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.

10.197.1 Detailed Description

Implements **Ogre** (p. 163)'s Run-Time Shader system.

This class allows Gazebo to generate per-pixel shaders for every material at run-time.

10.197.2 Member Enumeration Documentation

10.197.2.1 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.

10.197.3 Member Function Documentation

10.197.3.1 void gazebo::rendering::RTShaderSystem::AddScene (ScenePtr *_scene*)

Add a scene manager.

Parameters

in	*_scene*	The scene to process

10.197.3.2 void gazebo::rendering::RTShaderSystem::ApplyShadows (ScenePtr *_scene*)

Apply shadows to a scene.

Parameters

in	*_scene*	The scene to receive shadows.

10.197.3.3 void gazebo::rendering::RTShaderSystem::AttachEntity (Visual * *vis*)

Set an Ogre::Entity to use RT shaders.

Parameters

in	*_vis*	**Visual** (p. 1477) that will use the **RTShaderSystem** (p. 1092).

10.197.3.4 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.

10.197.3.5 void gazebo::rendering::RTShaderSystem::Clear ( )

Clear the shader system.

10.197.3.6 void gazebo::rendering::RTShaderSystem::DetachEntity (Visual * _vis)

Remove and entity.

Parameters

in	*_vis*	Remove this visual.

10.197.3.7 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.

10.197.3.8 void gazebo::rendering::RTShaderSystem::Fini ( )

Finalize the shader system.

10.197.3.9 void gazebo::rendering::RTShaderSystem::GenerateShaders (Visual * _vis)

Generate shaders for an entity.

Parameters

in	*_vis*	The visual to generate shaders for.

10.197.3.10 Ogre::PSSMShadowCameraSetup* gazebo::rendering::RTShaderSystem::GetPSSMShadowCameraSetup ( ) const

Get the **Ogre** (p. 163) PSSM Shadows camera setup.

Returns

The **Ogre** (p. 163) PSSM Shadows camera setup.

10.197.3.11 void gazebo::rendering::RTShaderSystem::Init ( )

Init the run time shader system.

10.197.3.12 void gazebo::rendering::RTShaderSystem::RemoveScene (ScenePtr *_scene*)

Remove a scene.

Parameters

in	*The*	scene to remove

10.197.3.13 void gazebo::rendering::RTShaderSystem::RemoveShadows (ScenePtr *_scene*)

Remove shadows from a scene.

Parameters

in	*_scene*	The scene to remove shadows from.

10.197.3.14 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.

10.197.3.15 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**

10.198 gazebo::rendering::Scene Class Reference

Representation of an entire scene graph.

`#include <rendering/rendering.hh>`

Public Types

- enum **SkyXMode** { **GZ_SKYX_ALL** = 0x0FFFFFFF, **GZ_SKYX_CLOUDS** = 0x0000001, **GZ_SKYX_MOON** = 0x0000002, **GZ_SKYX_NONE** = 0 }

Public Member Functions

- **Scene** (const std::string &_name, bool _enableVisualizations=false, bool _isServer=false)

  Constructor.
- virtual ~**Scene** ()

  Destructor.
- void **AddLight** (**LightPtr** _light)

  Add a light to the scene.
- void **AddVisual** (**VisualPtr** _vis)

  Add a visual to the scene.
- void **Clear** ()

  *Clear **rendering::Scene*** (p. *1097)*.
- **CameraPtr CreateCamera** (const std::string &_name, bool _autoRender=true)

  Create a camera.
- **DepthCameraPtr CreateDepthCamera** (const std::string &_name, bool _autoRender=true)

  Create depth camera.
- **GpuLaserPtr CreateGpuLaser** (const std::string &_name, bool _autoRender=true)

  Create laser that generates data from rendering.
- void **CreateGrid** (uint32_t _cellCount, float _cellLength, float _lineWidth, const **common::Color** &_color)

  Create a square grid of cells.
- **UserCameraPtr CreateUserCamera** (const std::string &_name)

  Create a user camera.
- void **DrawLine** (const **math::Vector3** &_start, const **math::Vector3** &_end, const std::string &_name)

  Draw a named line.
- **common::Color GetAmbientColor** () const

Get the ambient color.

- **common::Color GetBackgroundColor** () const

  Get the background color.

- **CameraPtr GetCamera** (uint32_t _index) const

  Get a camera based on an index.

- **CameraPtr GetCamera** (const std::string &_name) const

  Get a camera by name.

- uint32_t **GetCameraCount** () const

  Get the number of cameras in this scene.

- bool **GetFirstContact** (**CameraPtr** _camera, const **math::Vector2i** &_mouse-Pos, **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. *1097) 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.

- bool **GetShowClouds** () const

  Get whether or not clouds are displayed.

- **common::Time GetSimTime** () const

  Get the scene simulation time.

- **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 GetVisual** (uint32_t _id) const

  Get a visual by id.

- **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.

- uint32_t **GetVisualCount** () const

  Get the number of visuals.

- 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. *1097*).

- 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 **RemoveCamera** (const std::string &_name)

  Remove a camera from the scene.
- void **RemoveLight** (**LightPtr** _light)

  Remove a light to the scene.
- void **RemoveProjectors** ()

  Remove all projectors.
- 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 **SetSkyXMode** (unsigned int _mode)

  *Set **SkyX*** (p. *164)* *mode to enable/disable skyx components such as clouds and moon.*
- 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 **ShowClouds** (bool _show)

  Display clouds in the sky.
- 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.

10.198.1 Detailed Description

Representation of an entire scene graph.

Maintains all the Visuals, Lights, and Cameras for a World.

10.198.2 Member Enumeration Documentation

10.198.2.1 enum gazebo::rendering::Scene::SkyXMode

Enumerator:

GZ_SKYX_ALL

GZ_SKYX_CLOUDS

GZ_SKYX_MOON

GZ_SKYX_NONE

10.198.3 Constructor & Destructor Documentation

10.198.3.1 gazebo::rendering::Scene::Scene (const std::string & *_name*, bool *_enableVisualizations* = `false`, bool *_isServer* = `false`)

Constructor.

Parameters

`in`	*_name*	Name of the scene.
`in`	*_enable-Visualizations*	True to enable visualizations, this should be set to true for user interfaces, and false for sensor generation.

10.198.3.2 virtual gazebo::rendering::Scene::~Scene ( ) [virtual]

Destructor.

10.198.4 Member Function Documentation

10.198.4.1 void gazebo::rendering::Scene::AddLight (LightPtr _light)

Add a light to the scene.

Parameters

in	*_light*	**Light** (p. 730) to add.

10.198.4.2 void gazebo::rendering::Scene::AddVisual (VisualPtr _vis)

Add a visual to the scene.

Parameters

in	*_vis*	**Visual** (p. 1477) to add.

10.198.4.3 void gazebo::rendering::Scene::Clear ( )

Clear **rendering::Scene** (p. 1097).

10.198.4.4 CameraPtr gazebo::rendering::Scene::CreateCamera (const std::string & *_name*, bool *_autoRender* = true)

Create a camera.

Parameters

in	*_name*	Name of the new camera.
in	*_auto-Render*	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.198.4.5 DepthCameraPtr gazebo::rendering::Scene::CreateDepthCamera (const std::string & *_name,* bool *_autoRender* = `true`)

Create depth camera.

Parameters

in	*_name*	Name of the new camera.
in	*_auto-Render*	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.198.4.6 GpuLaserPtr gazebo::rendering::Scene::CreateGpuLaser (const std::string & *_name,* bool *_autoRender* = `true`)

Create laser that generates data from rendering.

Parameters

in	*_name*	Name of the new laser.
in	*_auto-Render*	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new laser.

10.198.4.7 void gazebo::rendering::Scene::CreateGrid (uint32_t *_cellCount,* float *_cellLength,* float *_lineWidth,* const common::Color & *_color*)

Create a square grid of cells.

Parameters

in	*_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.

10.198.4.8 UserCameraPtr gazebo::rendering::Scene::CreateUserCamera (const std::string & _name)

Create a user camera.

A user camera is one design for use with a GUI.

Parameters

in	*_name*	Name of the **UserCamera** (p. 1407).

Returns

A pointer to the new **UserCamera** (p. 1407).

10.198.4.9 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.

10.198.4.10 common::Color gazebo::rendering::Scene::GetAmbientColor ( ) const

Get the ambient color.

Returns

The scene's ambient color.

10.198.4.11 common::Color gazebo::rendering::Scene::GetBackgroundColor ( ) const

Get the background color.

Returns

The background color.

10.198.4.12 CameraPtr gazebo::rendering::Scene::GetCamera (uint32_t *_index*) const

Get a camera based on an index.

Index must be between 0 and **Scene::GetCameraCount** (p. 1105).

Parameters

in	*_index*	Index of the camera to get.

Returns

Pointer to the camera. Or NULL if the index is invalid.

10.198.4.13 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.

10.198.4.14 uint32_t gazebo::rendering::Scene::GetCameraCount ( ) const

Get the number of cameras in this scene.

Returns

Number of cameras.

10.198.4.15 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.

10.198.4.16 Grid* gazebo::rendering::Scene::GetGrid (uint32_t *_index*) const

Get a grid based on an index.

Index must be between 0 and **Scene::GetGridCount** (p. 1106).

Parameters

in	*_index*	Index of the grid.

10.198.4.17 uint32_t gazebo::rendering::Scene::GetGridCount ( ) const

Get the number of grids.

Returns

The number of grids.

10.198.4.18 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.

10.198.4.19 Heightmap* gazebo::rendering::Scene::GetHeightmap ( ) const

Get a pointer to the heightmap.

Returns

Pointer to the heightmap, NULL if no heightmap.

10.198.4.20 uint32_t gazebo::rendering::Scene::GetId ( ) const

Get the scene ID.

Returns

The ID of the scene.

10.198.4.21 std::string gazebo::rendering::Scene::GetIdString ( ) const

Get the scene Id as a string.

Returns

The ID as a string.

10.198.4.22 bool gazebo::rendering::Scene::GetInitialized ( ) const

Return true if the **Scene** (p. 1097) has been initialized.

10.198.4.23 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.

10.198.4.24 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. 1108).

Parameters

in	*_index*	Index of the light.

Returns

Pointer to the **Light** (p. 730) or NULL if index was invalid.

10.198.4.25 uint32_t gazebo::rendering::Scene::GetLightCount ( ) const

Get the count of the lights.

Returns

The number of lights.

10.198.4.26 Ogre::SceneManager* gazebo::rendering::Scene::GetManager ( ) const

Get the OGRE scene manager.

Returns

Pointer to the **Ogre** (p. 163) SceneManager.

10.198.4.27 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.

10.198.4.28 std::string gazebo::rendering::Scene::GetName ( ) const

Get the name of the scene.

Returns

Name of the scene.

10.198.4.29 VisualPtr gazebo::rendering::Scene::GetSelectedVisual ( ) const

Get the currently selected visual.

Returns

Pointer to the currently selected visual, or NULL if nothing is selected.

10.198.4.30 bool gazebo::rendering::Scene::GetShadowsEnabled ( ) const

Get whether shadows are on or off.

Returns

True if shadows are enabled.

10.198.4.31 bool gazebo::rendering::Scene::GetShowClouds ( ) const

Get whether or not clouds are displayed.

Returns

True if clouds are displayed.

10.198.4.32 common::Time gazebo::rendering::Scene::GetSimTime ( ) const

Get the scene simulation time.

Note this is different from World::GetSimTime() because there is a lag between the time new poses are sent out by World and when they are received and applied by the **Scene** (p. 1097).

Returns

The current simulation time in **Scene** (p. 1097)

10.198.4.33 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. 1110).

Parameters

in	*_index*	Index of the **UserCamera** (p. 1407) to get.

Returns

Pointer to the **UserCamera** (p. 1407), or NULL if the index was invalid.

10.198.4.34 uint32_t gazebo::rendering::Scene::GetUserCameraCount ( ) const

Get the number of user cameras in this scene.

Returns

The number of user cameras.

10.198.4.35 VisualPtr gazebo::rendering::Scene::GetVisual (const std::string & *_name*) const

Get a visual by name.

Parameters

in	*_name*	Name of the visual to retrieve.

Returns

Pointer to the visual, NULL if not found.

10.198.4.36 VisualPtr gazebo::rendering::Scene::GetVisual (uint32_t *_id*) const

Get a visual by id.

Parameters

in	*_id*	ID of the visual to retrieve.

Returns

Pointer to the visual, NULL if not found.

10.198.4.37 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

10.198.4.38 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.

10.198.4.39 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.

10.198.4.40 uint32_t gazebo::rendering::Scene::GetVisualCount ( ) const

Get the number of visuals.

Returns

The number of visuals in the **Scene** (p. 1097).

10.198.4.41 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.

10.198.4.42 VisualPtr gazebo::rendering::Scene::GetWorldVisual ( ) const

Get the top level world visual.

Returns

Pointer to the world visual.

10.198.4.43 void gazebo::rendering::Scene::Init ( )

Init **rendering::Scene** (p. 1097).

10.198.4.44 void gazebo::rendering::Scene::Load (sdf::ElementPtr _scene)

Load the scene from a set of parameters.

Parameters

in	*_scene*	SDF scene element to load.

10.198.4.45 void gazebo::rendering::Scene::Load ( )

Load the scene with default parameters.

10.198.4.46 void gazebo::rendering::Scene::PreRender ( )

Process all received messages.

10.198.4.47 void gazebo::rendering::Scene::PrintSceneGraph ( )

Print the scene graph to std_out.

10.198.4.48 void gazebo::rendering::Scene::RemoveCamera (const std::string & _name)

Remove a camera from the scene.

Parameters

in	*_name*	Name of the camera.

10.198.4.49 void gazebo::rendering::Scene::RemoveLight (LightPtr _light)

Remove a light to the scene.

Parameters

in	*_light*	**Light** (p. 730) to Remove.

10.198.4.50 void gazebo::rendering::Scene::RemoveProjectors ( )

Remove all projectors.

10.198.4.51 void gazebo::rendering::Scene::RemoveVisual (VisualPtr *_vis*)

Remove a visual from the scene.

Parameters

in	*_vis*	**Visual** (p. 1477) to remove.

10.198.4.52 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).

10.198.4.53 void gazebo::rendering::Scene::SetAmbientColor (const common::Color & *_color*)

Set the ambient color.

Parameters

in	*_color*	The ambient color to use.

10.198.4.54 void gazebo::rendering::Scene::SetBackgroundColor (const common::Color & *_color*)

Set the background color.

Parameters

in	*_color*	The background color.

10.198.4.55 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.

10.198.4.56 void gazebo::rendering::Scene::SetGrid (bool *_enabled*)

Set the grid on or off.

Parameters

in	*_enabled*	Set to true to turn on the grid

10.198.4.57 void gazebo::rendering::Scene::SetShadowsEnabled (bool *_value*)

Set whether shadows are on or off.

Parameters

in	*_value*	True to enable shadows, False to disable

10.198.4.58 void gazebo::rendering::Scene::SetSkyXMode (unsigned int *_mode*)

Set **SkyX** (p. 164) mode to enable/disable skyx components such as clouds and moon.

Parameters

in	*_mode*	**SkyX** (p. 164) mode bitmask.

See also

Scene::SkyXMode (p. 1101)

10.198.4.59 void gazebo::rendering::Scene::SetTransparent (bool *_show*)

Enable or disable transparency for all visuals.

Parameters

in	*_show*	True to enable transparency for all visuals.

10.198.4.60 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.

10.198.4.61 void gazebo::rendering::Scene::SetWireframe (bool *_show*)

Enable or disable wireframe for all visuals.

Parameters

in	*_show*	True to enable wireframe for all visuals.

10.198.4.62 void gazebo::rendering::Scene::ShowClouds (bool *_show*)

Display clouds in the sky.

Parameters

in	*_show*	True to display clouds.

10.198.4.63 void gazebo::rendering::Scene::ShowCollisions (bool _show)

Enable or disable collision visualization.

Parameters

in	*_show*	True to enable collision visualization.

10.198.4.64 void gazebo::rendering::Scene::ShowCOMs (bool _show)

Enable or disable center of mass visualization.

Parameters

in	*_show*	True to enable center of mass visualization.

10.198.4.65 void gazebo::rendering::Scene::ShowContacts (bool _show)

Enable or disable contact visualization.

Parameters

in	*_show*	True to enable contact visualization.

10.198.4.66 void gazebo::rendering::Scene::ShowJoints (bool _show)

Enable or disable joint visualization.

Parameters

in	*_show*	True to enable joint visualization.

10.198.4.67 void gazebo::rendering::Scene::SnapVisualToNearestBelow (const std::string & _visualName)

Move the visual to be ontop of the nearest visual below it.

Parameters

in	*_visual-Name*	Name of the visual to move.

10.198.4.68 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.

10.198.5 Member Data Documentation

10.198.5.1 SkyX::SkyX* gazebo::rendering::Scene::skyx

Pointer to the sky.

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

- **Scene.hh**

10.199 gazebo::physics::ScrewJoint< T > Class Template - Reference

A screw joint, which has both prismatic and rotational DOFs.

`#include <physics/physics.hh>`

Public Member Functions

- **ScrewJoint** (**BasePtr** _parent)
  Constructor.
- virtual ~**ScrewJoint** ()
  Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual double **GetThreadPitch** ()=0
  Get screw joint thread pitch.
- virtual void **Load** (sdf::ElementPtr _sdf)
  *Load a **ScrewJoint** (p. 1118).*
- virtual void **SetThreadPitch** (double _threadPitch)=0
  Set screw joint thread pitch.

Protected Member Functions

- virtual void **Init** ()

  Initialize joint.

Protected Attributes

- double **threadPitch**

  Pitch of the thread.

10.199.1 Detailed Description

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

A screw joint, which has both prismatic and rotational DOFs.

10.199.2 Constructor & Destructor Documentation

10.199.2.1 template<class T> gazebo::physics::ScrewJoint< T >::ScrewJoint (BasePtr *_parent*) [inline, explicit]

Constructor.

Parameters

in	*_parent*	Parent of the joint.

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

Destructor.

10.199.3 Member Function Documentation

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

10.199.3.2 template<class T> virtual double gazebo::physics::ScrewJoint< T >::GetThreadPitch ( ) [pure virtual]

Get screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class

Returns

_threadPitch Thread pitch value.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1248), and **gazebo::physics::DARTScrewJoint** (p. 456).

10.199.3.3 template<class T> virtual void gazebo::physics::ScrewJoint< T >::Init ( ) [inline, protected, virtual]

Initialize joint.

Reimplemented in **gazebo::physics::DARTScrewJoint** (p. 456).

10.199.3.4 template<class T> virtual void gazebo::physics::ScrewJoint< T >::Load (sdf::ElementPtr *_sdf*) [inline, virtual]

Load a **ScrewJoint** (p. 1118).

Parameters

in	*_sdf*	SDF value to load from

Reimplemented in **gazebo::physics::SimbodyScrewJoint** (p. 1249), and **gazebo::physics::DARTScrewJoint** (p. 457).

10.199.3.5 template<class T> virtual void gazebo::physics::ScrewJoint< T >::SetThreadPitch (double *_threadPitch*) [pure virtual]

Set screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. - This must be implemented in a child class To clarify direction, these are modeling right handed threads with positive thread_pitch, i.e. the child **Link** (p. 739) is the nut (interior threads) while the parent **Link** (p. 739) is the bolt/screw (exterior threads).

Parameters

in	*_threadPitch*	Thread pitch value.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1251), and **gazebo::physics::DARTScrewJoint** (p. 458).

10.199.4 Member Data Documentation

10.199.4.1 template<class T> double gazebo::physics::ScrewJoint< T >::threadPitch [protected]

Pitch of the thread.

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

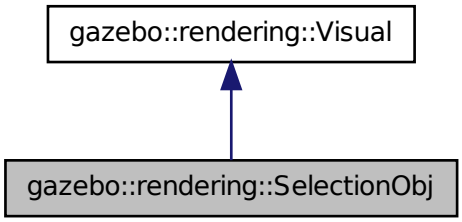
- **ScrewJoint.hh**

10.200 gazebo::rendering::SelectionObj Class Reference

Interactive selection object for models and links.

`#include <SelectionObj.hh>`

Inheritance diagram for gazebo::rendering::SelectionObj:



Public Types

- enum **SelectionMode** { **SELECTION_NONE** = 0, **TRANS**, **ROT**, **SCALE**, **TRANS_X**, **TRANS_Y**, **TRANS_Z**, **ROT_X**, **ROT_Y**, **ROT_Z**, **SCALE_X**, **SCALE_Y**, **SCALE_Z** }

Public Member Functions

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

Constructor.

- virtual ~**SelectionObj** ()

  Deconstructor.
- void **Attach** (**rendering::VisualPtr** _vis)

  Attach the selection object to the given visual.
- void **Detach** ()

  Detach the selection object from the current visual.
- **SelectionMode GetMode** ()

  Get the current selection mode.
- **SelectionMode GetState** ()

  Get the current selection state.
- void **Load** ()

  Load.
- void **SetGlobal** (bool _global)

  Set selection object to ignore local transforms.
- void **SetMode** (const std::string &_mode)

  Set the manipulation mode.
- void **SetMode** (**SelectionMode** _mode)

  Set the selection mode.
- void **SetState** (const std::string &_state)

  Set state by highlighting the corresponding selection object visual.
- void **SetState** (**SelectionMode** _state)

  Set state by highlighting the corresponding selection object visual.
- void **UpdateSize** ()

  Update selection object size to match the parent visual.

10.200.1 Detailed Description

Interactive selection object for models and links.

10.200.2 Member Enumeration Documentation

10.200.2.1 enum gazebo::rendering::SelectionObj::SelectionMode

Enumerator:

SELECTION_NONE Translation in x.

TRANS Translation mode.

ROT Rotation mode.

SCALE Scale mode.

TRANS_X Translation in x.

TRANS_Y Translation in y.

TRANS_Z Translation in z.

ROT_X Rotation in x.

ROT_Y Rotation in y.

ROT_Z Rotation in z.

SCALE_X Scale in x.

SCALE_Y Scale in y.

SCALE_Z Scale in z.

10.200.3 Constructor & Destructor Documentation

10.200.3.1 gazebo::rendering::SelectionObj::SelectionObj (const std::string & *_name*, VisualPtr *_vis*)

Constructor.

Parameters

in	*_name*	Name of selection object.
in	*_vis*	Parent visual that the selection object is attached to.

10.200.3.2 virtual gazebo::rendering::SelectionObj::~SelectionObj ( ) [virtual]

Deconstructor.

10.200.4 Member Function Documentation

10.200.4.1 void gazebo::rendering::SelectionObj::Attach (rendering::VisualPtr *_vis*)

Attach the selection object to the given visual.

Parameters

in	*_vis*	Pointer to visual to which the selection object will be attached.

10.200.4.2 void gazebo::rendering::SelectionObj::Detach ( )

Detach the selection object from the current visual.

10.200.4.3 SelectionMode gazebo::rendering::SelectionObj::GetMode ( )

Get the current selection mode.

10.200.4.4 SelectionMode gazebo::rendering::SelectionObj::GetState ( )

Get the current selection state.

10.200.4.5 void gazebo::rendering::SelectionObj::Load ( ) [virtual]

Load.

Reimplemented from **gazebo::rendering::Visual** (p. 1495).

10.200.4.6 void gazebo::rendering::SelectionObj::SetGlobal (bool *_global*)

Set selection object to ignore local transforms.

Parameters

in	*_global*	True to set the visuals to be in global frame.

10.200.4.7 void gazebo::rendering::SelectionObj::SetMode (const std::string & *_mode*)

Set the manipulation mode.

Parameters

in	*_mode*	Manipulation mode in string: translate rotate, scale.

10.200.4.8 void gazebo::rendering::SelectionObj::SetMode (SelectionMode *_mode*)

Set the selection mode.

Selection mode: TRANS, ROT, SCALE.

10.200.4.9 void gazebo::rendering::SelectionObj::SetState (const std::string & _state)

Set state by highlighting the corresponding selection object visual.

Parameters

in	*_state*	Selection state in string format.

10.200.4.10 void gazebo::rendering::SelectionObj::SetState (SelectionMode _state)

Set state by highlighting the corresponding selection object visual.

Parameters

in	*_state*	Selection state.

See also

SelectionMode (p. 1122)

10.200.4.11 void gazebo::rendering::SelectionObj::UpdateSize ( )

Update selection object size to match the parent visual.

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

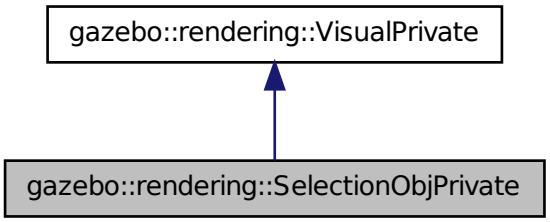
- **SelectionObj.hh**

10.201 gazebo::rendering::SelectionObjPrivate Class Reference

Private data for the Selection Obj class.

`#include <SelectionObjPrivate.hh>`

Inheritance diagram for gazebo::rendering::SelectionObjPrivate:



Public Attributes

- double **maxScale**

  Maximum scale of the selection object visual.
- double **minScale**

  Minimum scale of the selection object visual.
- **SelectionObj::SelectionMode mode**

  Current manipulation mode.
- **VisualPtr rotVisual**

  Rotation visual.
- **VisualPtr rotXVisual**

  X rotation visual.
- **VisualPtr rotYVisual**

  Y rotation visual.
- **VisualPtr rotZVisual**

  Z rotation visual.
- **VisualPtr scaleVisual**

  Scale visual.
- **VisualPtr scaleXVisual**

  X scale visual.
- **VisualPtr scaleYVisual**

  Y Scale visual.
- **VisualPtr scaleZVisual**

  Z scale visual.

- **VisualPtr selectedVis**

  Pointer to visual that is currently selected.
- **SelectionObj::SelectionMode state**

  Current selection state.
- **VisualPtr transVisual**

  Translation visual.
- **VisualPtr transXVisual**

  X translation visual.
- **VisualPtr transYVisual**

  Y translation visual.
- **VisualPtr transZVisual**

  Z translation visual.
- std::string **xAxisMat**

  Material name for the x axis.
- std::string **xAxisMatOverlay**

  Overlay material name for the x axis.
- std::string **yAxisMat**

  Material name for the y axis.
- std::string **yAxisMatOverlay**

  Overlay material name for the y axis.
- std::string **zAxisMat**

  Material name for the z axis.
- std::string **zAxisMatOverlay**

  Overlay material name for the z axis.

10.201.1 Detailed Description

Private data for the Selection Obj class.

10.201.2 Member Data Documentation

10.201.2.1 double gazebo::rendering::SelectionObjPrivate::maxScale

Maximum scale of the selection object visual.

10.201.2.2 double gazebo::rendering::SelectionObjPrivate::minScale

Minimum scale of the selection object visual.

10.201.2.3 SelectionObj::SelectionMode gazebo::rendering::SelectionObjPrivate::mode

Current manipulation mode.

10.201.2.4 VisualPtr gazebo::rendering::SelectionObjPrivate::rotVisual

Rotation visual.

10.201.2.5 VisualPtr gazebo::rendering::SelectionObjPrivate::rotXVisual

X rotation visual.

10.201.2.6 VisualPtr gazebo::rendering::SelectionObjPrivate::rotYVisual

Y rotation visual.

10.201.2.7 VisualPtr gazebo::rendering::SelectionObjPrivate::rotZVisual

Z rotation visual.

10.201.2.8 VisualPtr gazebo::rendering::SelectionObjPrivate::scaleVisual

Scale visual.

10.201.2.9 VisualPtr gazebo::rendering::SelectionObjPrivate::scaleXVisual

X scale visual.

10.201.2.10 VisualPtr gazebo::rendering::SelectionObjPrivate::scaleYVisual

Y Scale visual.

10.201.2.11 VisualPtr gazebo::rendering::SelectionObjPrivate::scaleZVisual

Z scale visual.

10.201.2.12 VisualPtr gazebo::rendering::SelectionObjPrivate::selectedVis

Pointer to visual that is currently selected.

10.201.2.13 SelectionObj::SelectionMode gazebo::rendering::SelectionObjPrivate::state

Current selection state.

10.201.2.14 VisualPtr gazebo::rendering::SelectionObjPrivate::transVisual

Translation visual.

10.201.2.15 VisualPtr gazebo::rendering::SelectionObjPrivate::transXVisual

X translation visual.

10.201.2.16 VisualPtr gazebo::rendering::SelectionObjPrivate::transYVisual

Y translation visual.

10.201.2.17 VisualPtr gazebo::rendering::SelectionObjPrivate::transZVisual

Z translation visual.

10.201.2.18 std::string gazebo::rendering::SelectionObjPrivate::xAxisMat

Material name for the x axis.

10.201.2.19 std::string gazebo::rendering::SelectionObjPrivate::xAxisMatOverlay

Overlay material name for the x axis.

10.201.2.20 std::string gazebo::rendering::SelectionObjPrivate::yAxisMat

Material name for the y axis.

10.201.2.21 std::string gazebo::rendering::SelectionObjPrivate::yAxisMatOverlay

Overlay material name for the y axis.

10.201.2.22 std::string gazebo::rendering::SelectionObjPrivate::zAxisMat

Material name for the z axis.

10.201.2.23 std::string gazebo::rendering::SelectionObjPrivate::zAxisMatOverlay

Overlay material name for the z axis.

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

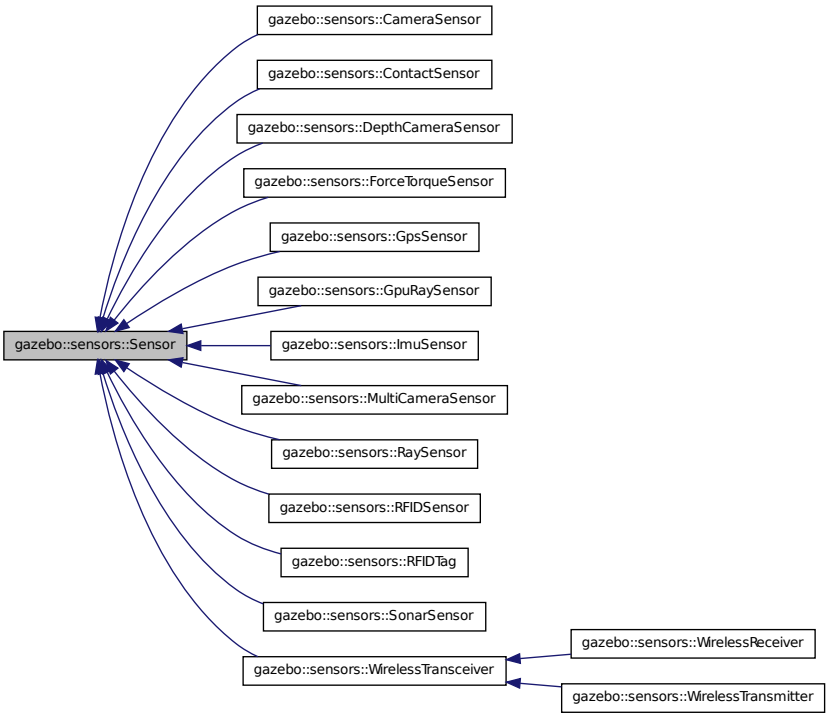
- **SelectionObjPrivate.hh**

10.202 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.
- uint32_t **GetId** () const

  Get the sensor's ID.
- **common::Time GetLastMeasurementTime** ()

  Return last measurement time.
- **common::Time GetLastUpdateTime** ()

  Return last update time.
- std::string **GetName** () const

  Get name.
- **NoisePtr GetNoise** (unsigned int _index=0) const

  Get the sensor's noise model.
- uint32_t **GetParentId** () const

  Get the sensor's parent's ID.
- 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.
- void **SetParent** (const std::string &_name, uint32_t _id)

  Set the sensor's parent.
- void **SetUpdateRate** (double _hz)

  Set the update rate of the sensor.
- void **Update** (bool _force)

  Update the sensor.

Protected Member Functions

- bool **NeedsUpdate** ()

  Return true if the sensor needs to be updated.
- virtual bool **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::vector< **NoisePtr** > **noises**

  Noise *(*p. 931*) added to sensor data.*
- uint32_t **parentId**

  The sensor's parent ID.
- 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.
- **gazebo::rendering::ScenePtr scene**

  Pointer to the Scene.
- sdf::ElementPtr **sdf**

  Pointer the the SDF element for the sensor.
- **common::Time updatePeriod**

  *Desired time between updates, set indirectly by **Sensor::SetUpdateRate** (p. 1141).*
- **gazebo::physics::WorldPtr world**

  Pointer to the world.

10.202.1 Detailed Description

Base class for sensors.

10.202.2 Constructor & Destructor Documentation

10.202.2.1 gazebo::sensors::Sensor::Sensor (SensorCategory *_cat*) [explicit]

Constructor.

Parameters

in	*_class*	

10.202.2.2 virtual gazebo::sensors::Sensor::~Sensor ( ) [virtual]

Destructor.

10.202.3 Member Function Documentation

10.202.3.1 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 pointer to the connection. This must be kept in scope.

See also

Sensor::DisconnectUpdated (p. 1135)

10.202.3.2 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. 1134)

10.202.3.3 void gazebo::sensors::Sensor::FillMsg (msgs::Sensor & *_msg*)

fills a msgs::Sensor message.

Parameters

out	*_msg*	Message to fill.

10.202.3.4 virtual void gazebo::sensors::Sensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented in **gazebo::sensors::MultiCameraSensor** (p. 897), **gazebo::sensors::ForceTorqueSensor** (p. 558), **gazebo::sensors::GpuRaySensor** (p. 595), **gazebo::sensors::RFIDSensor** (p. 1073), **gazebo::sensors::CameraSensor** (p. 281), **gazebo::sensors::ContactSensor** (p. 359), **gazebo::sensors::DepthCameraSensor** (p. 481), **gazebo::sensors::RaySensor** (p. 1054), **gazebo-**

::sensors::RFIDTag (p. 1076), **gazebo::sensors::GpsSensor** (p. 577), **gazebo::sensors::SonarSensor** (p. 1300), **gazebo::sensors::ImuSensor** (p. 652), **gazebo::sensors::WirelessTransceiver** (p. 1522), and **gazebo::sensors::WirelessReceiver** (p. 1519).

10.202.3.5 SensorCategory gazebo::sensors::Sensor::GetCategory ( ) const

Get the category of the sensor.

Returns

The category of the sensor.

See also

SensorCategory (p. 158)

10.202.3.6 uint32_t gazebo::sensors::Sensor::GetId ( ) const

Get the sensor's ID.

Returns

The sensor's ID.

10.202.3.7 common::Time gazebo::sensors::Sensor::GetLastMeasurementTime ( )

Return last measurement time.

Returns

Time of last measurement.

10.202.3.8 common::Time gazebo::sensors::Sensor::GetLastUpdateTime ( )

Return last update time.

Returns

Time of last update.

10.202.3.9 std::string gazebo::sensors::Sensor::GetName ( ) const

Get name.

Returns

Name of sensor.

10.202.3.10 NoisePtr gazebo::sensors::Sensor::GetNoise (unsigned int *_index* = 0) const

Get the sensor's noise model.

Parameters

in	*_index*	Index of the noise model. For most sensors this will be 0. For a multi camera sensor the index can be >=0.

Returns

The sensor's noise model.

10.202.3.11 uint32_t gazebo::sensors::Sensor::GetParentId ( ) const

Get the sensor's parent's ID.

Returns

The sensor's parent's ID.

10.202.3.12 std::string gazebo::sensors::Sensor::GetParentName ( ) const

Returns the name of the sensor parent.

The parent name is set by **Sensor::SetParent** (p. 1141).

Returns

Name of Parent.

10.202.3.13 virtual math::Pose gazebo::sensors::Sensor::GetPose ( ) const [virtual]

Get the current pose.

Returns

Current pose of the sensor.

10.202.3.14 std::string gazebo::sensors::Sensor::GetScopedName ( ) const

Get fully scoped name of the sensor.

Returns

world_name::model_name::link_name::sensor_name.

10.202.3.15 virtual std::string gazebo::sensors::Sensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented in **gazebo::sensors::GpuRaySensor** (p. 600), **gazebo::sensors::RaySensor** (p. 1057), **gazebo::sensors::CameraSensor** (p. 282), **gazebo::sensors::SonarSensor** (p. 1301), **gazebo::sensors::ForceTorqueSensor** (p. 559), **gazebo::sensors::MultiCameraSensor** (p. 899), and **gazebo::sensors::WirelessTransceiver** (p. 1523).

10.202.3.16 std::string gazebo::sensors::Sensor::GetType ( ) const

Get sensor type.

Returns

Type of sensor.

10.202.3.17 double gazebo::sensors::Sensor::GetUpdateRate ( )

Get the update rate of the sensor.

Returns

_hz update rate of sensor. Returns 0 if unthrottled.

10.202.3.18 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.

10.202.3.19 std::string gazebo::sensors::Sensor::GetWorldName ( ) const

Returns the name of the world the sensor is in.

Returns

Name of the world.

10.202.3.20 virtual void gazebo::sensors::Sensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented in **gazebo::sensors::GpuRaySensor** (p. 601), **gazebo::sensors::ContactSensor** (p. 362), **gazebo::sensors::RFIDSensor** (p. 1074), **gazebo::sensors::CameraSensor** (p. 283), **gazebo::sensors::DepthCameraSensor** (p. 482), **gazebo::sensors::WirelessTransmitter** (p. 1527), **gazebo::sensors::RaySensor** (p. 1058), **gazebo::sensors::RFIDTag** (p. 1077), **gazebo::sensors::GpsSensor** (p. 578), **gazebo::sensors::SonarSensor** (p. 1301), **gazebo::sensors::ForceTorqueSensor** (p. 559), **gazebo::sensors::MultiCameraSensor** (p. 899), **gazebo::sensors::ImuSensor** (p. 653), **gazebo::sensors::WirelessTransceiver** (p. 1523), and **gazebo::sensors::WirelessReceiver** (p. 1520).

10.202.3.21 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::GpuRaySensor** (p. 601), **gazebo::sensors::RaySensor** (p. 1059), **gazebo::sensors::ContactSensor** (p. 362), **gazebo::sensors::CameraSensor** (p. 283), **gazebo::sensors::MultiCameraSensor** (p. 899), **gazebo::sensors::SonarSensor** (p. 1302), **gazebo::sensors::ImuSensor** (p. 653), and **gazebo::sensors::ForceTorqueSensor** (p. 560).

10.202.3.22 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. 1130) parameters.
in	*_worldName*	Name of world to load from.

Reimplemented in **gazebo::sensors::GpuRaySensor** (p. 602), **gazebo::sensors::ContactSensor** (p. 362), **gazebo::sensors::RFIDSensor** (p. 1074), **gazebo::sensors::CameraSensor** (p. 283), **gazebo::sensors::DepthCameraSensor** (p. 482), **gazebo::sensors::RFIDTag** (p. 1077), **gazebo::sensors::GpsSensor** (p. 578), **gazebo::sensors::ForceTorqueSensor** (p. 560), and **gazebo::sensors::ImuSensor** (p. 653).

10.202.3.23 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. 602), **gazebo::sensors::ContactSensor** (p. 362), **gazebo::sensors::RFIDSensor** (p. 1074), **gazebo::sensors::CameraSensor** (p. 283), **gazebo::sensors::DepthCameraSensor** (p. 482), **gazebo::sensors::WirelessTransmitter** (p. 1527), **gazebo::sensors::RaySensor** (p. 1059), **gazebo::sensors::RFIDTag** (p. 1077), **gazebo::sensors::GpsSensor** (p. 578), **gazebo::sensors::SonarSensor** (p. 1302), **gazebo::sensors::ForceTorqueSensor** (p. 560), **gazebo::sensors::MultiCameraSensor** (p. 899), **gazebo::sensors::ImuSensor** (p. 653), **gazebo::sensors::WirelessTransceiver** (p. 1523), and **gazebo::sensors::WirelessReceiver** (p. 1520).

10.202.3.24 bool gazebo::sensors::Sensor::NeedsUpdate ( ) [protected]

Return true if the sensor needs to be updated.

Returns

True when sensor should be updated.

10.202.3.25 void gazebo::sensors::Sensor::ResetLastUpdateTime ( )

Reset the lastUpdateTime to zero.

10.202.3.26 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. 483).

10.202.3.27 void gazebo::sensors::Sensor::SetParent (const std::string & *_name*, uint32_t *_id*)

Set the sensor's parent.

Parameters

in	*_name*	The sensor's parent's name.
in	*_id*	The sensor's parent's ID.

10.202.3.28 void gazebo::sensors::Sensor::SetUpdateRate (double *_hz*)

Set the update rate of the sensor.

Parameters

in	*_hz*	update rate of sensor.

10.202.3.29 void gazebo::sensors::Sensor::Update (bool *_force*)

Update the sensor.

Parameters

in	*_force*	True to force update, false otherwise.

10.202.3.30 virtual bool gazebo::sensors::Sensor::UpdateImpl (bool) [inline, protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented in **gazebo::sensors::MultiCameraSensor** (p. 900), **gazebo::sensors::ForceTorqueSensor** (p. 560), **gazebo::sensors::GpuRaySensor** (p. 603), **gazebo::sensors::RFIDSensor** (p. 1074), **gazebo::sensors::CameraSensor** (p. 284), **gazebo::sensors::ContactSensor** (p. 363), **gazebo::sensors::DepthCameraSensor** (p. 483), **gazebo::sensors::RaySensor** (p. 1059), **gazebo::sensors::RFIDTag** (p. 1077), **gazebo::sensors::GpsSensor** (p. 578), **gazebo::sensors::SonarSensor** (p. 1302), **gazebo::sensors::WirelessTransmitter** (p. 1528), and **gazebo::sensors::ImuSensor** (p. 654).

10.202.4 Member Data Documentation

10.202.4.1 bool gazebo::sensors::Sensor::active [protected]

True if sensor generation is active.

10.202.4.2 std::vector<event::ConnectionPtr> gazebo::sensors::Sensor::connections [protected]

All event connections.

10.202.4.3 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.

10.202.4.4 common::Time gazebo::sensors::Sensor::lastUpdateTime [protected]

Time of the last update.

10.202.4.5 transport::NodePtr gazebo::sensors::Sensor::node [protected]

Node for communication.

10.202.4.6 std::vector<NoisePtr> gazebo::sensors::Sensor::noises [protected]

Noise (p. 931) added to sensor data.

10.202.4.7 uint32_t gazebo::sensors::Sensor::parentId [protected]

The sensor's parent ID.

10.202.4.8 std::string gazebo::sensors::Sensor::parentName [protected]

Name of the parent.

10.202.4.9 std::vector<SensorPluginPtr> gazebo::sensors::Sensor::plugins [protected]

All the plugins for the sensor.

10.202.4.10 math::Pose gazebo::sensors::Sensor::pose [protected]

Pose of the sensor.

10.202.4.11 transport::SubscriberPtr gazebo::sensors::Sensor::poseSub [protected]

Subscribe to pose updates.

10.202.4.12 gazebo::rendering::ScenePtr gazebo::sensors::Sensor::scene [protected]

Pointer to the Scene.

10.202.4.13 sdf::ElementPtr gazebo::sensors::Sensor::sdf [protected]

Pointer the the SDF element for the sensor.

10.202.4.14 common::Time gazebo::sensors::Sensor::updatePeriod [protected]

Desired time between updates, set indirectly by **Sensor::SetUpdateRate** (p. 1141).

10.202.4.15 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**

10.203 SensorFactor Class Reference

The sensor factory; the class is just for namespacing purposes.

`#include <sensors/sensors.hh>`

10.203.1 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**

10.204 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).

10.204.1 Member Function Documentation

10.204.1.1 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

10.204.1.2 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. 1130)

10.204.1.3 static void gazebo::sensors::SensorFactory::RegisterAll ( ) [static]

Register all known sensors.

- **sensors::CameraSensor** (p. 280)
- **sensors::DepthCameraSensor** (p. 480)
- **sensors::GpuRaySensor** (p. 591)
- **sensors::RaySensor** (p. 1051)
- **sensors::ContactSensor** (p. 358)
- **sensors::RFIDSensor** (p. 1072)
- **sensors::RFIDTag** (p. 1075)
- **sensors::WirelessTransmitter** (p. 1524)
- **sensors::WirelessReceiver** (p. 1517)

10.204.1.4 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:

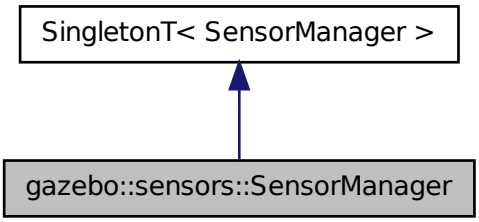
- **SensorFactory.hh**

10.205 gazebo::sensors::SensorManager Class Reference

Class to manage and update all sensors.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::SensorManager:



Classes

- class **ImageSensorContainer**
- class **SensorContainer**

Public Member Functions

- std::string **CreateSensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)

  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 **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.

10.205.1 Detailed Description

Class to manage and update all sensors.

10.205.2 Member Function Documentation

10.205.2.1 std::string gazebo::sensors::SensorManager::CreateSensor (sdf::ElementPtr *_elem*, const std::string & *_worldName*, const std::string & *_parentName*, uint32_t *_parentId*)

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

10.205.2.2 void gazebo::sensors::SensorManager::Fini ( )

Finalize all the sensors.

10.205.2.3 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 pointer to the sensor. NULL if not found.

10.205.2.4 Sensor_V gazebo::sensors::SensorManager::GetSensors ( ) const

Get all the sensors.

Returns

Vector of all the sensors.

10.205.2.5 void gazebo::sensors::SensorManager::GetSensorTypes (std::vector< std::string > & _types) const

Get all the sensor types.

Parameters

out	*All*	the sensor types.

10.205.2.6 void gazebo::sensors::SensorManager::Init ( )

Init all the sensors.

10.205.2.7 void gazebo::sensors::SensorManager::RemoveSensor (const std::string & _name)

Remove a sensor.

Parameters

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

10.205.2.8 void gazebo::sensors::SensorManager::RemoveSensors ( )

Remove all sensors.

10.205.2.9 void gazebo::sensors::SensorManager::ResetLastUpdateTimes ( )

Reset last update times in all sensors.

10.205.2.10 void gazebo::sensors::SensorManager::RunThreads ( )

Run sensor updates in separate threads.

This will only run non-image based sensor updates.

10.205.2.11 bool gazebo::sensors::SensorManager::SensorsInitialized ( )

True if SensorManager::initSensors queue is empty i.e.

all sensors managed by **SensorManager** (p. 1146) have been initialized

10.205.2.12 void gazebo::sensors::SensorManager::Stop ( )

Stop the run thread.

10.205.2.13 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:

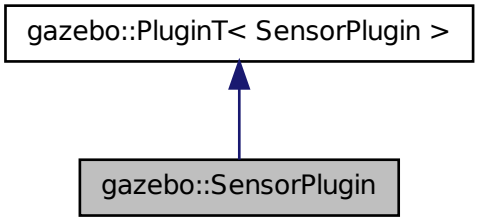
- **SensorManager.hh**

10.206 gazebo::SensorPlugin Class Reference

A plugin with access to physics::Sensor.

`#include <common/common.hh>`

Inheritance diagram for gazebo::SensorPlugin:



Public Member Functions

- **SensorPlugin** ()

  Constructor.
- virtual ~**SensorPlugin** ()

  Destructor.
- virtual void **Init** ()

  Override this method for custom plugin initialization behavior.
- virtual void **Load** (**sensors::SensorPtr** _sensor, sdf::ElementPtr _sdf)=0

  Load function.
- virtual void **Reset** ()

  Override this method for custom plugin reset behavior.

10.206.1 Detailed Description

A plugin with access to physics::Sensor.

See `reference`.

10.206.2 Constructor & Destructor Documentation

10.206.2.1 gazebo::SensorPlugin::SensorPlugin ( ) [inline]

Constructor.

References gazebo::SENSOR_PLUGIN.

10.206.2.2 virtual gazebo::SensorPlugin::~SensorPlugin ( ) [inline, virtual]

Destructor.

10.206.3 Member Function Documentation

10.206.3.1 virtual void gazebo::SensorPlugin::Init ( ) [inline, virtual]

Override this method for custom plugin initialization behavior.

10.206.3.2 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.

10.206.3.3 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:

- **Plugin.hh**

10.207 gazebo::Server Class Reference

`#include <Server.hh>`

Public Member Functions

- **Server** ()

  Constructor.
- virtual ∼**Server** ()

  Destructor.
- void **Fini** ()

  *Finalize the **Server** (*p. *1153).*
- bool **GetInitialized** () const

  *Get whether the **Server** (*p. *1153) has been initialized.*
- 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)

  *Load the **Server** (*p. *1153) from an SDF string.*
- bool **ParseArgs** (int _argc, char ∗∗_argv)

  Parse command line arguments.
- bool **PreLoad** ()

  Preload the server.
- void **PrintUsage** ()

  Output help about gzserver.
- void **Run** ()

  *Run the **Server** (*p. *1153).*
- void **SetParams** (const **common::StrStr_M** &_params)

  Set the parameters.
- void **Stop** ()

  *Stop the **Server** (*p. *1153).*

10.207.1 Constructor & Destructor Documentation

10.207.1.1 gazebo::Server::Server ( )

Constructor.

10.207.1.2 virtual gazebo::Server::~Server ( ) [virtual]

Destructor.

10.207.2 Member Function Documentation

10.207.2.1 void gazebo::Server::Fini ( )

Finalize the **Server** (p. 1153).

10.207.2.2 bool gazebo::Server::GetInitialized ( ) const

Get whether the **Server** (p. 1153) has been initialized.

Returns

True if initialized.

10.207.2.3 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*	Physics engine type (ode|bullet|dart|simbody).

Returns

True on success.

10.207.2.4 bool gazebo::Server::LoadString (const std::string & *_sdfString*)

Load the **Server** (p. 1153) from an SDF string.

Parameters

in	*_sdfString*	SDF string from which to load a World.

Returns

True on success.

10.207.2.5 bool gazebo::Server::ParseArgs (int *_argc*, char ** *_argv*)

Parse command line arguments.

Parameters

in	*_argc*	Number of arguments.
in	*_argv*	Array of argument values.

Returns

True on success.

10.207.2.6 bool gazebo::Server::PreLoad ( )

Preload the server.

Returns

True if load was successful.

10.207.2.7 void gazebo::Server::PrintUsage ( )

Output help about gzserver.

10.207.2.8 void gazebo::Server::Run ( )

Run the **Server** (p. 1153).

10.207.2.9 void gazebo::Server::SetParams (const common::StrStr_M & *_params*)

Set the parameters.

Parameters

in	*_params*	Map of string parameters

10.207.2.10 void gazebo::Server::Stop ( )

Stop the **Server** (p. 1153).

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

- **Server.hh**

10.208 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg Class Reference

Keeping the CG shader for reference.

`#include <Heightmap.hh>`

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)

10.208.1 Detailed Description

Keeping the CG shader for reference.

Utility class to help with generating shaders for Cg / HLSL.

10.208.2 Member Function Documentation

10.208.2.1 virtual void **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::defaultVpParams** (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, const Ogre::HighLevelGpuProgramPtr & *_prog*) [protected, virtual]

10.208.2.2 virtual Ogre::HighLevelGpuProgramPtr **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateFragmentProgram** (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*) [virtual]

10.208.2.3 virtual Ogre::HighLevelGpuProgramPtr **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgram** (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*) [virtual]

10.208.2.4 virtual void **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVertexProgramSource** (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*) [protected, virtual]

10.208.2.5 virtual void **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpDynamicShadows** (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*) [protected, virtual]

10.208.2.6 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]

10.208.2.7 virtual void **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperCg::generateVpFooter** (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*) [protected, virtual]

10.208.2.8 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**

10.209 gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL Class Reference

Utility class to help with generating shaders for GLSL.

`#include <Heightmap.hh>`

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)

10.209.1 Detailed Description

Utility class to help with generating shaders for GLSL.

10.209.2 Member Function Documentation

10.209.2.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::defaultVpParams (const SM2Profile ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *_tt,* const Ogre::HighLevelGpuProgramPtr & *_prog*) [protected, virtual]

10.209.2.2 void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadows (const SM2Profile ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *_tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected]

10.209.2.3 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpDynamicShadowsHelpers (const SM2Profile ∗ *_prof,* const Ogre::Terrain ∗ *_terrain,* TechniqueType *tt,* Ogre::StringUtil::StrStreamType & *_outStream*) [protected, virtual]

10.209.2.4 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]

10.209.2.5 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpFooter (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream) [protected, virtual]

10.209.2.6 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFpHeader (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType tt, Ogre::StringUtil::StrStreamType & _outStream) [protected, virtual]

10.209.2.7 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]

10.209.2.8 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgram (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt) [virtual]

10.209.2.9 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateFragmentProgramSource (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream) [protected, virtual]

10.209.2.10 virtual Ogre::HighLevelGpuProgramPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgram (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt) [virtual]

10.209.2.11 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVertexProgramSource (const SM2Profile * _prof, const Ogre::Terrain * _terrain, TechniqueType _tt, Ogre::StringUtil::StrStreamType & _outStream) [protected, virtual]

10.209.2.12 virtual void **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamicShadows** (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*) [protected, virtual]

10.209.2.13 virtual unsigned int **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpDynamicShadowsParams** (unsigned int *_texCoordStart*, const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*) [protected, virtual]

10.209.2.14 virtual void **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpFooter** (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*) [protected, virtual]

10.209.2.15 virtual void **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::generateVpHeader** (const SM2Profile * *_prof*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*, Ogre::StringUtil::StrStreamType & *_outStream*) [protected, virtual]

10.209.2.16 virtual void **gazebo::rendering::GzTerrainMatGen::SM2Profile::ShaderHelperGLSL::updateParams** (const SM2Profile * *_prof*, const Ogre::MaterialPtr & *_mat*, const Ogre::Terrain * *_terrain*, bool *_compositeMap*) [virtual]

10.209.2.17 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:

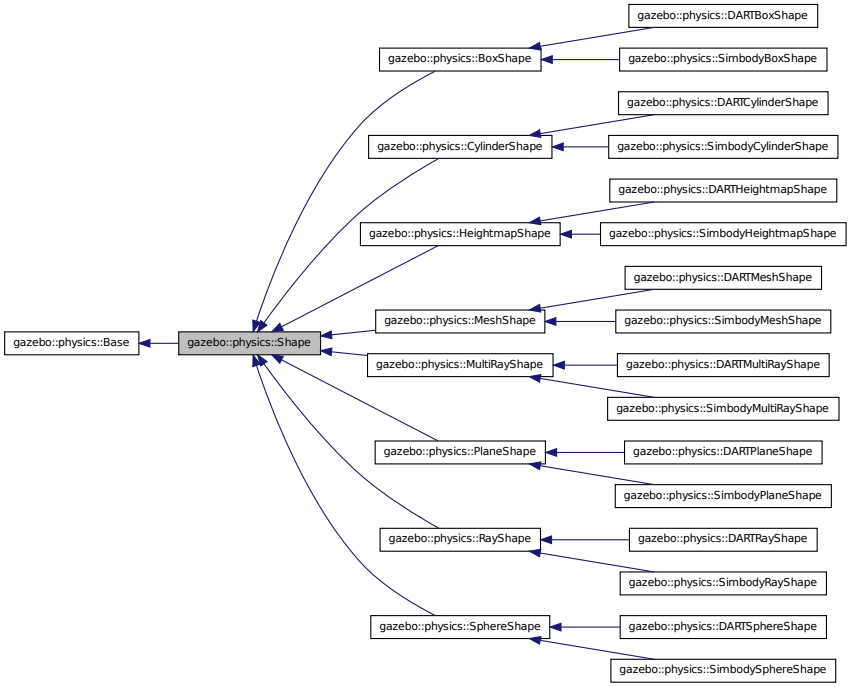
- **Heightmap.hh**

10.210 gazebo::physics::Shape Class Reference

Base (p. 201) 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 **math::Vector3 GetScale** () const

  Get the scale of the shape.
- virtual void **Init** ()=0

  Initialize the shape.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)=0

  Process a geometry message.
- virtual void **SetScale** (const **math::Vector3** &_scale)=0

Set the scale of the shape.

Protected Attributes

- **CollisionPtr collisionParent**

  This shape's collision parent.
- **math::Vector3 scale**

  This shape's scale;.

10.210.1 Detailed Description

Base (p. 201) class for all shapes.

10.210.2 Constructor & Destructor Documentation

10.210.2.1 gazebo::physics::Shape::Shape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of the shape.

10.210.2.2 virtual gazebo::physics::Shape::~Shape ( ) [virtual]

Destructor.

10.210.3 Member Function Documentation

10.210.3.1 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. 905), **gazebo::physics::RayShape** (p. 1062), **gazebo::physics::HeightmapShape** (p. 630), **gazebo::physics-**

::PlaneShape (p. 990), **gazebo::physics::MeshShape** (p. 843), **gazebo::physics::CylinderShape** (p. 372), **gazebo::physics::SphereShape** (p. 1308), and **gazebo::physics::BoxShape** (p. 229).

10.210.3.2 virtual math::Vector3 gazebo::physics::Shape::GetScale ( ) const [virtual]

Get the scale of the shape.

Returns

Scale of the shape.

10.210.3.3 virtual void gazebo::physics::Shape::Init ( ) [pure virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::Base** (p. 211).

Implemented in **gazebo::physics::RayShape** (p. 1064), **gazebo::physics::HeightmapShape** (p. 632), **gazebo::physics::MeshShape** (p. 844), **gazebo::physics::MultiRayShape** (p. 908), **gazebo::physics::PlaneShape** (p. 990), **gazebo::physics::SphereShape** (p. 1309), **gazebo::physics::BoxShape** (p. 230), **gazebo::physics::CylinderShape** (p. 372), **gazebo::physics::SimbodyHeightmapShape** (p. 1181), **gazebo::physics::SimbodyMeshShape** (p. 1221), **gazebo::physics::DARTHeightmapShape** (p. 393), and **gazebo::physics::DARTMeshShape** (p. 432).

10.210.3.4 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. 908), **gazebo::physics::RayShape** (p. 1064), **gazebo::physics::HeightmapShape** (p. 633), **gazebo::physics::PlaneShape** (p. 990), **gazebo::physics::MeshShape** (p. 844), **gazebo::physics::CylinderShape** (p. 373), **gazebo::physics::SphereShape** (p. 1309), and **gazebo::physics::BoxShape** (p. 230).

10.210.3.5 virtual void gazebo::physics::Shape::SetScale (const math::Vector3 & _scale) [pure virtual]

Set the scale of the shape.

Parameters

in	*_scale*	Scale to set the shape to.

Implemented in **gazebo::physics::RayShape** (p. 1065), **gazebo::physics::PlaneShape** (p. 991), **gazebo::physics::MeshShape** (p. 845), **gazebo::physics::CylinderShape** (p. 373), **gazebo::physics::HeightmapShape** (p. 633), **gazebo::physics::SphereShape** (p. 1309), **gazebo::physics::BoxShape** (p. 230), and **gazebo::physics::MultiRayShape** (p. 908).

10.210.4 Member Data Documentation

10.210.4.1 CollisionPtr gazebo::physics::Shape::collisionParent [protected]

This shape's collision parent.

10.210.4.2 math::Vector3 gazebo::physics::Shape::scale [protected]

This shape's scale;.

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

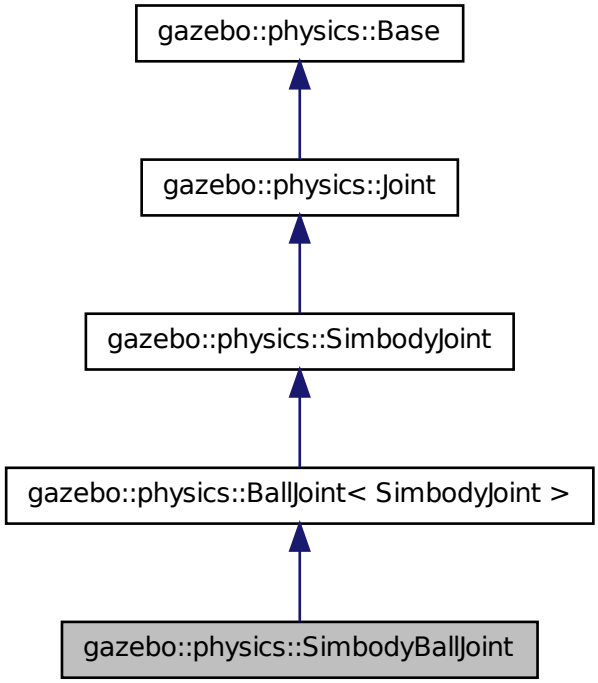
- **Shape.hh**

10.211 gazebo::physics::SimbodyBallJoint Class Reference

SimbodyBallJoint (p. 1165) class models a ball joint in Simbody.

`#include <SimbodyBallJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodyBallJoint:



Public Member Functions

- **SimbodyBallJoint** (SimTK::MultibodySystem *_world, **BasePtr** _parent)

  *Simbody Ball **Joint** (p. 669) Constructor.*
- virtual ~**SimbodyBallJoint** ()

  Destructor.
- **math::Vector3 GetAnchor** (unsigned int _index) const

  Get the anchor point.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.
- virtual **math::Vector3 GetAxis** (unsigned int) const
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

Get the axis of rotation in global cooridnate frame.

- virtual **math::Angle GetHighStop** (unsigned int _index)

  Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (unsigned int _index)

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Template to ::Load the **BallJoint** (p. 199).*
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual void **SetMaxForce** (unsigned int _index, double _t)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual void **SetVelocity** (unsigned int _index, double _angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _torque)

  *Set the force applied to this **physics::Joint** (p. 669).*

10.211.1 Detailed Description

SimbodyBallJoint (p. 1165) class models a ball joint in Simbody.

10.211.2 Constructor & Destructor Documentation

10.211.2.1 gazebo::physics::SimbodyBallJoint::SimbodyBallJoint (SimTK::MultibodySystem * *_world*, BasePtr *_parent*)

Simbody Ball **Joint** (p. 669) Constructor.

10.211.2.2 virtual gazebo::physics::SimbodyBallJoint::~SimbodyBallJoint ( ) [virtual]

Destructor.

10.211.3 Member Function Documentation

10.211.3.1 math::Vector3 gazebo::physics::SimbodyBallJoint::GetAnchor (unsigned int _index) const [virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1197).

10.211.3.2 virtual math::Angle gazebo::physics::SimbodyBallJoint::GetAngleImpl (unsigned int _index) const [virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.211.3.3 virtual math::Vector3 gazebo::physics::SimbodyBallJoint::GetAxis (unsigned *int*) const [inline, virtual]

10.211.3.4 virtual math::Vector3 gazebo::physics::SimbodyBallJoint::GetGlobalAxis (unsigned int _index) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.211.3.5 virtual math::Angle gazebo::physics::SimbodyBallJoint::GetHighStop (unsigned int _index) [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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1198).

10.211.3.6 virtual math::Angle gazebo::physics::SimbodyBallJoint::GetLowStop (unsigned int _index) [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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1200).

10.211.3.7 virtual double gazebo::physics::SimbodyBallJoint::GetMaxForce (unsigned int *_index*) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.211.3.8 virtual double gazebo::physics::SimbodyBallJoint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.211.3.9 virtual void gazebo::physics::SimbodyBallJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Template to ::Load the **BallJoint** (p. 199).

Parameters

in	*_sdf*	SDF to load the joint from.

Reimplemented from **gazebo::physics::BallJoint< SimbodyJoint >** (p. 200).

10.211.3.10 virtual void gazebo::physics::SimbodyBallJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1202).

10.211.3.11 virtual void gazebo::physics::SimbodyBallJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implements **gazebo::physics::SimbodyJoint** (p. 1203).

10.211.3.12 virtual bool gazebo::physics::SimbodyBallJoint::SetHighStop (unsigned 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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1204).

10.211.3.13 virtual bool gazebo::physics::SimbodyBallJoint::SetLowStop (unsigned 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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1204).

10.211.3.14 virtual void gazebo::physics::SimbodyBallJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.211.3.15 virtual void gazebo::physics::SimbodyBallJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

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

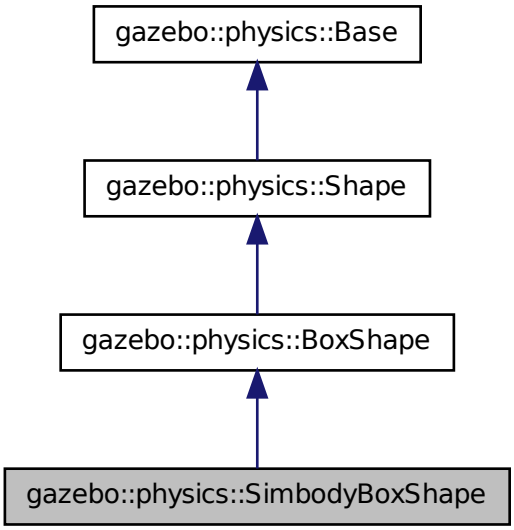
- **SimbodyBallJoint.hh**

10.212 gazebo::physics::SimbodyBoxShape Class Reference

Simbody box collision.

`#include <SimbodyBoxShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyBoxShape:



Public Member Functions

- **SimbodyBoxShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**SimbodyBoxShape** ()

  Destructor.
- void **SetSize** (const **math::Vector3** &_size)

  Set the size of the box.

10.212.1 Detailed Description

Simbody box collision.

10.212.2 Constructor & Destructor Documentation

10.212.2.1 gazebo::physics::SimbodyBoxShape::SimbodyBoxShape (CollisionPtr *_parent*) [inline]

Constructor.

10.212.2.2 virtual gazebo::physics::SimbodyBoxShape::~SimbodyBoxShape ( ) [inline, virtual]

Destructor.

10.212.3 Member Function Documentation

10.212.3.1 void gazebo::physics::SimbodyBoxShape::SetSize (const math::Vector3 & *_size*) [inline, virtual]

Set the size of the box.

Parameters

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

Reimplemented from **gazebo::physics::BoxShape** (p. 230).

References gazebo::math::equal(), gzerr, gzwarn, gazebo::physics::BoxShape::SetSize(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

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

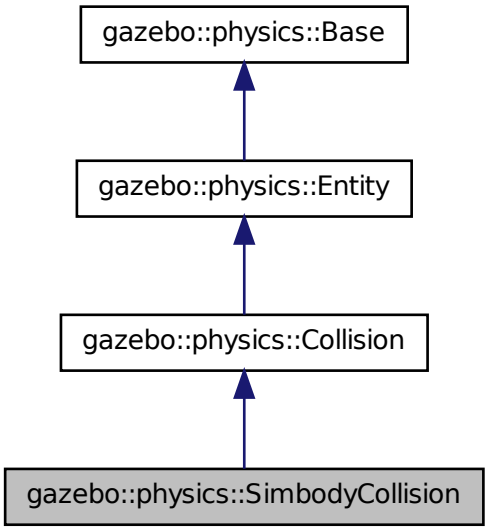
- **SimbodyBoxShape.hh**

10.213 gazebo::physics::SimbodyCollision Class Reference

Simbody collisions.

`#include <SimbodyCollision.hh>`

Inheritance diagram for gazebo::physics::SimbodyCollision:



Public Member Functions

- **SimbodyCollision** (**LinkPtr** _parent)

  Constructor.
- virtual ∼**SimbodyCollision** ()

  Destructor.
- virtual **math::Box GetBoundingBox** () const

  Get the bounding box for this collision.
- SimTK::ContactGeometry ∗ **GetCollisionShape** () const

  Get the simbody collision shape.
- virtual void **Load** (sdf::ElementPtr _ptr)

  Load the collision.
- virtual void **OnPoseChange** ()

  This function is called when the entity's (or one of its parents) pose of the parent has changed.
- virtual void **SetCategoryBits** (unsigned int _bits)

Set the category bits, used during collision detection.

- virtual void **SetCollideBits** (unsigned int _bits)

  Set the collide bits, used during collision detection.

- void **SetCollisionShape** (SimTK::ContactGeometry *_shape)

  Set the collision shape.

10.213.1 Detailed Description

Simbody collisions.

10.213.2 Constructor & Destructor Documentation

10.213.2.1 gazebo::physics::SimbodyCollision::SimbodyCollision (LinkPtr *_parent*)

Constructor.

10.213.2.2 virtual gazebo::physics::SimbodyCollision::~SimbodyCollision ( ) [virtual]

Destructor.

10.213.3 Member Function Documentation

10.213.3.1 virtual math::Box gazebo::physics::SimbodyCollision::GetBoundingBox ( ) const [virtual]

Get the bounding box for this collision.

Returns

The bounding box.

Implements **gazebo::physics::Collision** (p. 299).

10.213.3.2 SimTK::ContactGeometry* gazebo::physics::SimbodyCollision::GetCollisionShape ( ) const

Get the simbody collision shape.

Returns

SimTK (p. 164) geometry used as the collision shape.

10.213.3.3 virtual void gazebo::physics::SimbodyCollision::Load (sdf::ElementPtr _sdf) [virtual]

Load the collision.

Parameters

in	*_sdf*	SDF to load from.

Reimplemented from **gazebo::physics::Collision** (p. 304).

10.213.3.4 virtual void gazebo::physics::SimbodyCollision::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. 509).

10.213.3.5 virtual void gazebo::physics::SimbodyCollision::SetCategoryBits (unsigned int _bits) [virtual]

Set the category bits, used during collision detection.

Parameters

in	*_bits*	The bits to set.

Implements **gazebo::physics::Collision** (p. 304).

10.213.3.6 virtual void gazebo::physics::SimbodyCollision::SetCollideBits (unsigned int _bits) [virtual]

Set the collide bits, used during collision detection.

Parameters

in	*_bits*	The bits to set.

Implements **gazebo::physics::Collision** (p. 304).

10.213.3.7 void gazebo::physics::SimbodyCollision::SetCollisionShape (SimTK::ContactGeometry * *_shape*)

Set the collision shape.

Parameters

in	*_shape*	**SimTK** (p. 164) geometry to use as the collision **SimTK** (p. 164) geometry to use as the collision shape.

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

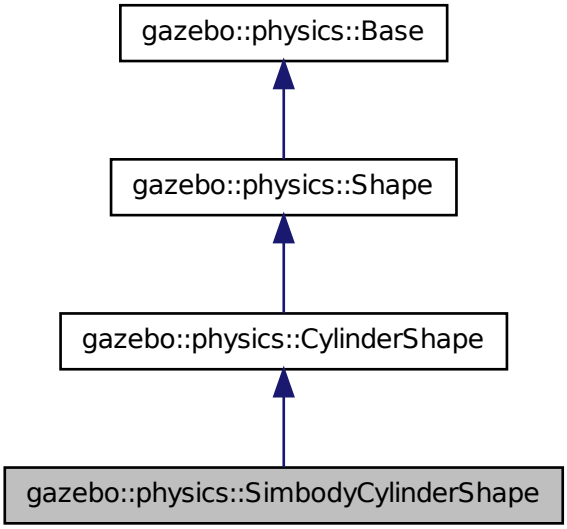
- **SimbodyCollision.hh**

10.214 gazebo::physics::SimbodyCylinderShape Class Reference

Cylinder collision.

`#include <SimbodyCylinderShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyCylinderShape:



Public Member Functions

- **SimbodyCylinderShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**SimbodyCylinderShape** ()

  Destructor.
- void **SetSize** (double _radius, double _length)

  Set the size of the cylinder.

10.214.1 Detailed Description

Cylinder collision.

10.214.2 Constructor & Destructor Documentation

10.214.2.1 gazebo::physics::SimbodyCylinderShape::SimbodyCylinderShape (CollisionPtr *_parent*) [inline]

Constructor.

10.214.2.2 virtual gazebo::physics::SimbodyCylinderShape-::~SimbodyCylinderShape ( ) [inline, virtual]

Destructor.

10.214.3 Member Function Documentation

10.214.3.1 void gazebo::physics::SimbodyCylinderShape::SetSize (double *_radius*, double *_length*) [inline, virtual]

Set the size of the cylinder.

Parameters

in	*_radius*	New radius.
in	*_lenght*	New length.

Reimplemented from **gazebo::physics::CylinderShape** (p. 373).

References gazebo::math::equal(), gzerr, gzwarn, and gazebo::physics::CylinderShape::SetSize().

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

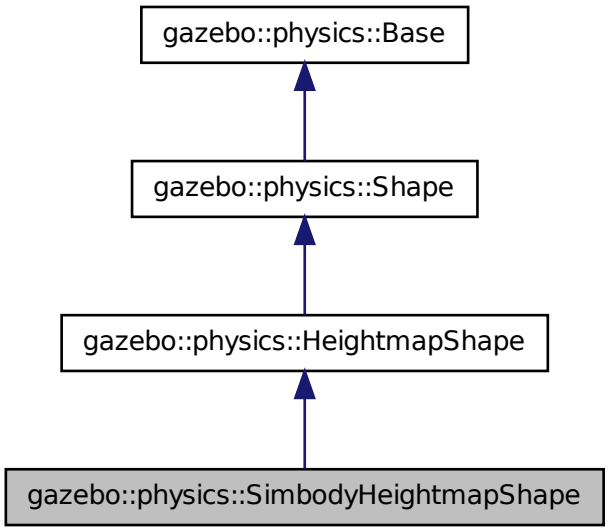
- **SimbodyCylinderShape.hh**

10.215 gazebo::physics::SimbodyHeightmapShape Class Reference

Height map collision.

`#include <SimbodyHeightmapShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyHeightmapShape:



Public Member Functions

- **SimbodyHeightmapShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**SimbodyHeightmapShape** ()

  Destructor.

- virtual void **Init** ()

  Initialize the heightmap.

10.215.1 Detailed Description

Height map collision.

10.215.2 Constructor & Destructor Documentation

10.215.2.1 gazebo::physics::SimbodyHeightmapShape::-SimbodyHeightmapShape (CollisionPtr *_parent*)

Constructor.

10.215.2.2 virtual gazebo::physics::SimbodyHeightmapShape::~SimbodyHeightmapShape ( ) [virtual]

Destructor.

10.215.3 Member Function Documentation

10.215.3.1 virtual void gazebo::physics::SimbodyHeightmapShape::Init ( ) [virtual]

Initialize the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 632).

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

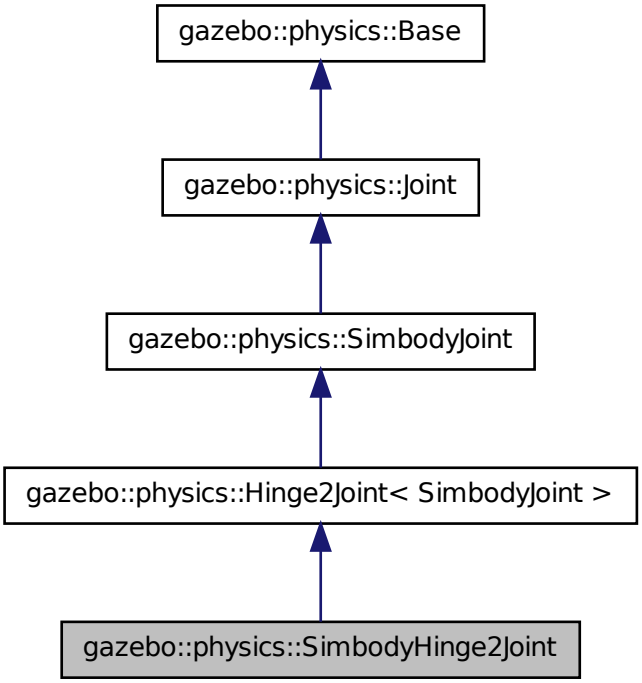
- **SimbodyHeightmapShape.hh**

10.216 gazebo::physics::SimbodyHinge2Joint Class Reference

A two axis hinge joint.

`#include <SimbodyHinge2Joint.hh>`

Inheritance diagram for gazebo::physics::SimbodyHinge2Joint:



Public Member Functions

- **SimbodyHinge2Joint** (SimTK::MultibodySystem *__world__, **BasePtr** _parent)

  Constructor.
- virtual ~**SimbodyHinge2Joint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (unsigned int _index) const

  Get the anchor point.
- virtual **math::Vector3 GetAxis** (unsigned int _index) const
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.
- virtual double **GetMaxForce** (unsigned int _index)

*Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*

- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)

- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.

- virtual void **SetMaxForce** (unsigned int _index, double _t)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*

- virtual void **SetVelocity** (unsigned int _index, double _angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the joint.

- virtual void **SetForceImpl** (unsigned int _index, double _torque)

  Set the torque.

10.216.1 Detailed Description

A two axis hinge joint.

10.216.2 Constructor & Destructor Documentation

10.216.2.1 gazebo::physics::SimbodyHinge2Joint::SimbodyHinge2Joint (SimTK::MultibodySystem * *world*, BasePtr *_parent*)

Constructor.

10.216.2.2 virtual gazebo::physics::SimbodyHinge2Joint::~SimbodyHinge2Joint ( ) [virtual]

Destructor.

10.216.3 Member Function Documentation

10.216.3.1 virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetAnchor (unsigned int _index) const [virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1197).

10.216.3.2 virtual math::Angle gazebo::physics::SimbodyHinge2Joint::GetAngleImpl (unsigned int _index) const [protected, virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.216.3.3 virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetAxis (unsigned int _index) const [virtual]

10.216.3.4 virtual math::Vector3 gazebo::physics::SimbodyHinge2Joint::GetGlobalAxis (unsigned int _index) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.216.3.5 virtual double gazebo::physics::SimbodyHinge2Joint::GetMaxForce (unsigned int *_index*) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.216.3.6 virtual double gazebo::physics::SimbodyHinge2Joint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.216.3.7 virtual void gazebo::physics::SimbodyHinge2Joint::Load (sdf::ElementPtr _sdf) [protected, virtual]

Load the joint.

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 635).

10.216.3.8 virtual void gazebo::physics::SimbodyHinge2Joint::SetAxis (unsigned int _index, const math::Vector3 & _axis) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1202).

10.216.3.9 virtual void gazebo::physics::SimbodyHinge2Joint::SetForceImpl (unsigned int _index, double _torque) [protected, virtual]

Set the torque.

Implements **gazebo::physics::SimbodyJoint** (p. 1203).

10.216.3.10 virtual void gazebo::physics::SimbodyHinge2Joint::SetMaxForce (unsigned int _index, double _force) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.216.3.11 virtual void gazebo::physics::SimbodyHinge2Joint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

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

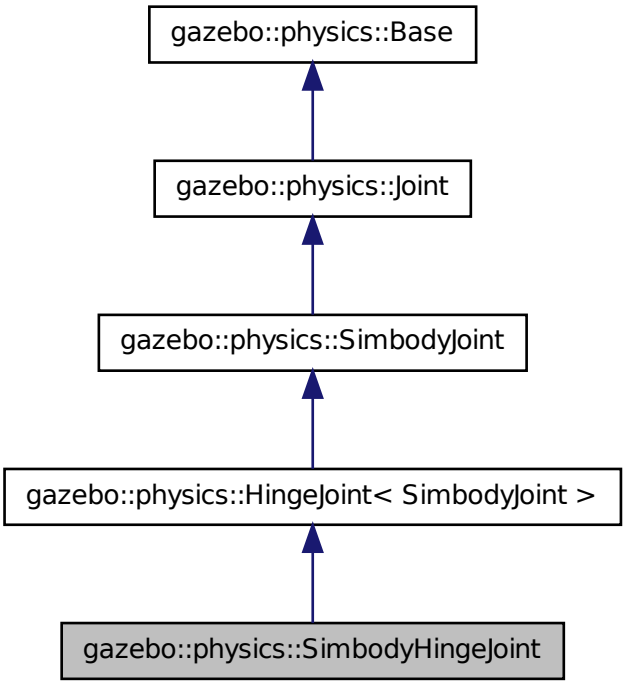
- **SimbodyHinge2Joint.hh**

10.217 gazebo::physics::SimbodyHingeJoint Class Reference

A single axis hinge joint.

`#include <SimbodyHingeJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodyHingeJoint:



Public Member Functions

- **SimbodyHingeJoint** (SimTK::MultibodySystem ∗**world**, **BasePtr** _parent)

  Constructor.
- virtual ∼**SimbodyHingeJoint** ()

  Destructor.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.
- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)

- virtual void **RestoreSimbodyState** (SimTK::State &_state)

  restore simbody state for spawning
- virtual void **SaveSimbodyState** (const SimTK::State &_state)

  save simbody state for spawning
- void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _t)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity*** (p. *701).*
- virtual void **SetVelocity** (unsigned int _index, double _rate)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load joint.
- virtual void **SetForceImpl** (unsigned int _index, double _torque)

  *Set the force applied to this **physics::Joint*** (p. *669).*

10.217.1 Detailed Description

A single axis hinge joint.

10.217.2 Constructor & Destructor Documentation

10.217.2.1 gazebo::physics::SimbodyHingeJoint::SimbodyHingeJoint (SimTK::MultibodySystem * *world,* BasePtr *_parent*)

Constructor.

10.217.2.2 virtual gazebo::physics::SimbodyHingeJoint::~SimbodyHingeJoint ( ) [virtual]

Destructor.

10.217.3 Member Function Documentation

10.217.3.1 virtual math::Angle gazebo::physics::SimbodyHingeJoint::GetAngleImpl (unsigned int _index) const [protected, virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.217.3.2 virtual math::Vector3 gazebo::physics::SimbodyHingeJoint::GetGlobalAxis (unsigned int _index) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.217.3.3 virtual double gazebo::physics::SimbodyHingeJoint::GetMaxForce (unsigned int _index) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.217.3.4 virtual double gazebo::physics::SimbodyHingeJoint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.217.3.5 virtual void gazebo::physics::SimbodyHingeJoint::Load (sdf::ElementPtr *_sdf*) [protected, virtual]

Load joint.

Parameters

in	*_sdf*	Pointer to SDF element

Reimplemented from **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 637).

10.217.3.6 virtual void gazebo::physics::SimbodyHingeJoint::RestoreSimbodyState (SimTK::State & *_state*) [virtual]

restore simbody state for spawning

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1201).

10.217.3.7 virtual void gazebo::physics::SimbodyHingeJoint::SaveSimbodyState (const SimTK::State & *_state*) [virtual]

save simbody state for spawning

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1201).

10.217.3.8 void gazebo::physics::SimbodyHingeJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1202).

10.217.3.9 virtual void gazebo::physics::SimbodyHingeJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implements **gazebo::physics::SimbodyJoint** (p. 1203).

10.217.3.10 virtual void gazebo::physics::SimbodyHingeJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.217.3.11 virtual void gazebo::physics::SimbodyHingeJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

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

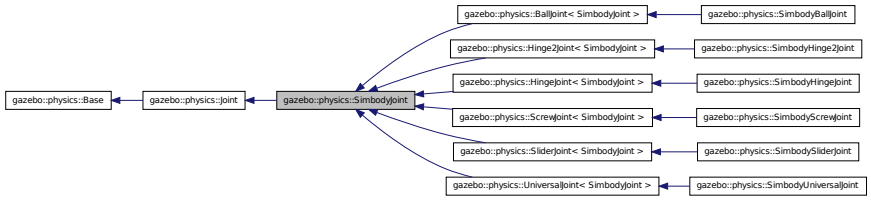
- **SimbodyHingeJoint.hh**

10.218 gazebo::physics::SimbodyJoint Class Reference

Base (p. 201) class for all joints.

`#include <SimbodyJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodyJoint:



Public Member Functions

- **SimbodyJoint** (**BasePtr** _parent)

Constructor.

- virtual ~**SimbodyJoint** ()

  Destructor.

- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const

  Determines of the two bodies are connected by a joint.

- virtual void **CacheForceTorque** ()

  *Cache **Joint** (p. 669) Force Torque Values if necessary for physics engine.*

- virtual void **Detach** ()

  Detach this joint from all links.

- virtual **math::Vector3 GetAnchor** (unsigned int _index) const

  Get the anchor point.

- virtual double **GetForce** (unsigned int _index)
- virtual **JointWrench GetForceTorque** (unsigned int _index)

  get internal force and torque values at a joint.

- virtual **math::Angle GetHighStop** (unsigned int _index)

  Get the high stop of an axis(index).

- virtual **LinkPtr GetJointLink** (unsigned int _index) const

  Get the link to which the joint is attached according the _index.

- virtual **math::Vector3 GetLinkForce** (unsigned int _index) const

  *Get the forces applied to the center of mass of a **physics::Link** (p. 739) due to the existence of this **Joint** (p. 669).*

- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const

  *Get the torque applied to the center of mass of a **physics::Link** (p. 739) due to the existence of this **Joint** (p. 669).*

- virtual **math::Angle GetLowStop** (unsigned int _index)

  Get the low stop of an axis(index).

- virtual double **GetParam** (const std::string &_key, unsigned int _index)

  Get a non-generic parameter for the joint.

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load **physics::Joint** (p. 669) from a SDF sdf::Element.*

- virtual void **Reset** ()

  Reset the joint.

- virtual void **RestoreSimbodyState** (SimTK::State &_state)
- virtual void **SaveSimbodyState** (const SimTK::State &_state)
- virtual void **SetAnchor** (unsigned int _index, const **gazebo::math::Vector3** &_anchor)

  Set the anchor point.

- virtual void **SetAttribute** (**Attribute**, unsigned int _index, double _value)

  Set a parameter for the joint.

- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

Set the axis of rotation where axis is specified in local joint frame.

- virtual void **SetDamping** (unsigned int _index, const double _damping)

  Set the joint damping.
- virtual void **SetForce** (unsigned int _index, double _force)

  *Set the force applied to this **physics::Joint*** (p. *669*).
- virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual bool **SetParam** (const std::string &_key, unsigned int _index, const boost::any &_value)

  Set a non-generic parameter for the joint.
- virtual void **SetStiffness** (unsigned int _index, const double _stiffness)

  Set the joint spring stiffness.
- virtual void **SetStiffnessDamping** (unsigned int _index, double _stiffness, double _damping, double _reference=0)

  Set the joint spring stiffness.

Public Attributes

- SimTK::Constraint **constraint**

  : isValid() if we used a constraint to model this joint.
- SimTK::Force::MobilityLinearDamper **damper** [**MAX_JOINT_AXIS**]

  : for enforcing joint damping forces.
- SimTK::Transform **defxAB**

  default mobilizer pose
- bool **isReversed**

  : if mobilizer, did it reverse parent&child? Set when we build the Simbody model.
- SimTK::Force::MobilityLinearStop **limitForce** [**MAX_JOINT_AXIS**]

  : for enforcing joint stops Set when we build the Simbody model.
- SimTK::MobilizedBody **mobod**

  Use isValid() if we used a mobilizer Set when we build the Simbody model.
- bool **mustBreakLoopHere**

  Force Simbody to break a loop by using a weld constraint.
- bool **physicsInitialized**
- SimTK::Force::MobilityLinearSpring **spring** [**MAX_JOINT_AXIS**]

  : Spring force element for enforcing joint stiffness.
- SimTK::Transform **xCB**

  child body frame to mobilizer frame
- SimTK::Transform **xPA**

  Normally A=F, B=M.

Protected Member Functions

- virtual void **SetForceImpl** (unsigned int _index, double _force)=0

  *Set the force applied to this **physics::Joint** (p. 669).*

Protected Attributes

- **SimbodyPhysicsPtr simbodyPhysics**

  keep a pointer to the simbody physics engine for convenience
- SimTK::MultibodySystem * **world**

  Simbody Multibody System.

10.218.1 Detailed Description

Base (p. 201) class for all joints.

10.218.2 Constructor & Destructor Documentation

10.218.2.1 gazebo::physics::SimbodyJoint::SimbodyJoint (BasePtr _parent)

Constructor.

10.218.2.2 virtual gazebo::physics::SimbodyJoint::~SimbodyJoint ( ) [virtual]

Destructor.

10.218.3 Member Function Documentation

10.218.3.1 virtual bool gazebo::physics::SimbodyJoint::AreConnected (LinkPtr _one, LinkPtr _two) const [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.

Implements **gazebo::physics::Joint** (p. 676).

10.218.3.2 virtual void gazebo::physics::SimbodyJoint::CacheForceTorque ( ) [virtual]

Cache **Joint** (p. 669) Force Torque Values if necessary for physics engine.

Reimplemented from **gazebo::physics::Joint** (p. 677).

10.218.3.3 virtual void gazebo::physics::SimbodyJoint::Detach ( ) [virtual]

Detach this joint from all links.

Reimplemented from **gazebo::physics::Joint** (p. 678).

10.218.3.4 virtual math::Vector3 gazebo::physics::SimbodyJoint::GetAnchor (unsigned int *_index*) const [virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Implements **gazebo::physics::Joint** (p. 679).

Reimplemented in **gazebo::physics::SimbodyUniversalJoint** (p. 1262), **gazebo::physics::SimbodyHinge2Joint** (p. 1184), and **gazebo::physics::SimbodyBallJoint** (p. 1168).

10.218.3.5 virtual double gazebo::physics::SimbodyJoint::GetForce (unsigned int *_index*) [virtual]

Todo : not yet implemented. Get external forces applied at this **Joint** (p. 669). Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The force applied to an axis.

Reimplemented from **gazebo::physics::Joint** (p. 683).

10.218.3.6 virtual JointWrench gazebo::physics::SimbodyJoint::GetForceTorque (unsigned int *_index*) [virtual]

get internal force and torque values at a joint.

The force and torque values are returned in a **JointWrench** (p. 721) data structure. - Where **JointWrench.body1Force** (p. 723) contains the force applied by the parent **Link** (p. 739) on the **Joint** (p. 669) specified in the parent **Link** (p. 739) frame, and **JointWrench.body2Force** (p. 723) contains the force applied by the child **Link** (p. 739) on the **Joint** (p. 669) specified in the child **Link** (p. 739) frame. Note that this sign convention is opposite of the reaction forces of the **Joint** (p. 669) on the Links.

FIXME TODO: change name of this function to something like: GetNegatedForceTorqueInLinkFrame and make GetForceTorque call return non-negated reaction forces in perspective **Link** (p. 739) frames.

Note that for ODE you must set <provide_feedback>true<provide_feedback> in the joint sdf to use this.

Parameters

in	*_index*	Not used right now

Returns

The force and torque at the joint, see above for details on conventions.

Implements **gazebo::physics::Joint** (p. 683).

10.218.3.7 virtual math::Angle gazebo::physics::SimbodyJoint::GetHighStop (unsigned int *_index*) [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.

Implements **gazebo::physics::Joint** (p. 684).

Reimplemented in **gazebo::physics::SimbodyBallJoint** (p. 1169), and **gazebo::physics::SimbodyScrewJoint** (p. 1247).

10.218.3.8 virtual LinkPtr gazebo::physics::SimbodyJoint::GetJointLink (unsigned int *_index*) const [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.

Implements **gazebo::physics::Joint** (p. 686).

10.218.3.9 virtual math::Vector3 gazebo::physics::SimbodyJoint::GetLinkForce (unsigned int *_index*) const [virtual]

Get the forces applied to the center of mass of a **physics::Link** (p. 739) due to the existence of this **Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*index*	The index of the link(0 or 1).

Returns

Force applied to the link.

Implements **gazebo::physics::Joint** (p. 686).

10.218.3.10 virtual math::Vector3 gazebo::physics::SimbodyJoint::GetLinkTorque (unsigned int _index) const [virtual]

Get the torque applied to the center of mass of a **physics::Link** (p. 739) due to the existence of this **Joint** (p. 669).

Note that the unit of torque should be consistent with the rest of the simulation scales.

Parameters

in	*index*	The index of the link(0 or 1)

Returns

Torque applied to the link.

Implements **gazebo::physics::Joint** (p. 687).

10.218.3.11 virtual math::Angle gazebo::physics::SimbodyJoint::GetLowStop (unsigned int _index) [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.

Implements **gazebo::physics::Joint** (p. 688).

Reimplemented in **gazebo::physics::SimbodyBallJoint** (p. 1169), and **gazebo::physics::SimbodyScrewJoint** (p. 1247).

10.218.3.12 virtual double gazebo::physics::SimbodyJoint::GetParam (const std::string & _key, unsigned int _index) [virtual]

Get a non-generic parameter for the joint.

Parameters

in	*_key*	String key.
in	*_index*	Index of the axis.

Implements **gazebo::physics::Joint** (p. 689).

Reimplemented in **gazebo::physics::SimbodyScrewJoint** (p. 1248).

10.218.3.13 virtual void gazebo::physics::SimbodyJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load **physics::Joint** (p. 669) from a SDF sdf::Element.

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::Joint** (p. 693).

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 1256), **gazebo::physics::BallJoint< SimbodyJoint >** (p. 200), **gazebo::physics::UniversalJoint< SimbodyJoint >** (p. 1406), **gazebo::physics::Hinge2Joint< SimbodyJoint >** (p. 635), **gazebo::physics::HingeJoint< SimbodyJoint >** (p. 637), **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 1120), **gazebo::physics::SliderJoint< SimbodyJoint >** (p. 1295), **gazebo::physics::SimbodyHingeJoint** (p. 1191), **gazebo::physics::SimbodyUniversalJoint** (p. 1264), **gazebo::physics::SimbodyHinge2Joint** (p. 1186), **gazebo::physics::SimbodyScrewJoint** (p. 1249), and **gazebo::physics::SimbodyBallJoint** (p. 1170).

10.218.3.14 virtual void gazebo::physics::SimbodyJoint::Reset ( ) [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Joint** (p. 694).

10.218.3.15 virtual void gazebo::physics::SimbodyJoint::RestoreSimbodyState (SimTK::State & *_state*) [virtual]

Reimplemented in **gazebo::physics::SimbodyHingeJoint** (p. 1191).

10.218.3.16 virtual void gazebo::physics::SimbodyJoint::SaveSimbodyState (const SimTK::State & *_state*) [virtual]

Reimplemented in **gazebo::physics::SimbodyHingeJoint** (p. 1191).

10.218.3.17 virtual void gazebo::physics::SimbodyJoint::SetAnchor (unsigned int *_index*, const gazebo::math::Vector3 & *_anchor*) [virtual]

Set the anchor point.

Parameters

in	*_index*	Indx of the axis.
in	*_anchor*	Anchor value.

Implements **gazebo::physics::Joint** (p. 694).

10.218.3.18 virtual void gazebo::physics::SimbodyJoint::SetAttribute (Attribute , unsigned int *_index*, double *_value*) [virtual]

Set a parameter for the joint.

10.218.3.19 virtual void gazebo::physics::SimbodyJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Implements **gazebo::physics::Joint** (p. 695).

Reimplemented in **gazebo::physics::SimbodyBallJoint** (p. 1171), **gazebo::physics::SimbodyUniversalJoint** (p. 1264), **gazebo::physics::SimbodyHinge2Joint** (p. 1186), **gazebo::physics::SimbodyHingeJoint** (p. 1192), **gazebo::physics::SimbodyScrewJoint** (p. 1249), and **gazebo::physics::SimbodySliderJoint** (p. 1256).

10.218.3.20 virtual void gazebo::physics::SimbodyJoint::SetDamping (unsigned int *_index*, const double *_damping*) [virtual]

Set the joint damping.

Parameters

in	*_index*	Index of the axis to set, currently ignored, to be implemented.

in	*_damping*	Damping value for the axis.

Implements **gazebo::physics::Joint** (p. 695).

10.218.3.21 virtual void gazebo::physics::SimbodyJoint::SetForce (unsigned int *_index*, double *_effort*) [virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForce to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)). Forces are truncated by effortLimit before applied.

Parameters

in	*_index*	Index of the axis.
in	*_effort*	Force value.

Implements **gazebo::physics::Joint** (p. 696).

10.218.3.22 virtual void gazebo::physics::SimbodyJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected, pure virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implemented in **gazebo::physics::SimbodyScrewJoint** (p. 1249), **gazebo::physics::SimbodyBallJoint** (p. 1171), **gazebo::physics::SimbodyUniversalJoint** (p. 1264), **gazebo::physics::SimbodyHinge2Joint** (p. 1186), **gazebo::physics::SimbodyHingeJoint** (p. 1192), and **gazebo::physics::SimbodySliderJoint** (p. 1257).

10.218.3.23 virtual bool gazebo::physics::SimbodyJoint::SetHighStop (unsigned 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.

Reimplemented from **gazebo::physics::Joint** (p. 696).

Reimplemented in **gazebo::physics::SimbodyBallJoint** (p. 1171), and **gazebo::physics::SimbodyScrewJoint** (p. 1250).

10.218.3.24 virtual bool gazebo::physics::SimbodyJoint::SetLowStop (unsigned 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.

Reimplemented from **gazebo::physics::Joint** (p. 697).

Reimplemented in **gazebo::physics::SimbodyBallJoint** (p. 1172), and **gazebo::physics::SimbodyScrewJoint** (p. 1250).

10.218.3.25 virtual bool gazebo::physics::SimbodyJoint::SetParam (const std::string & _key, unsigned int _index, const boost::any & _value) [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.

Implements **gazebo::physics::Joint** (p. 698).

Reimplemented in **gazebo::physics::SimbodyScrewJoint** (p. 1251).

10.218.3.26 virtual void gazebo::physics::SimbodyJoint::SetStiffness (unsigned int _index, const double _stiffness) [virtual]

Set the joint spring stiffness.

Parameters

in	*_index*	Index of the axis to set, currently ignored, to be implemented.
in	*_stiffness*	Spring stiffness value for the axis. : rename to SetSpringStiffness()

Implements **gazebo::physics::Joint** (p. 699).

10.218.3.27 virtual void gazebo::physics::SimbodyJoint::SetStiffnessDamping (unsigned int _index, double _stiffness, double _damping, double _reference = 0) [virtual]

Set the joint spring stiffness.

Parameters

in	*_index*	Index of the axis to set, currently ignored, to be implemented.
in	*_stiffness*	Stiffness value for the axis.
in	*_reference*	Spring zero load reference position. : rename to SetSpringStiffnessDamping()

Implements **gazebo::physics::Joint** (p. 700).

10.218.4 Member Data Documentation

10.218.4.1 SimTK::Constraint gazebo::physics::SimbodyJoint::constraint

: isValid() if we used a constraint to model this joint.

Set when we build the Simbody model. How this joint was modeled in the Simbody System. We used either a mobilizer or a constraint, but not both. The type of either one is the same as the joint type above.

10.218.4.2 SimTK::Force::MobilityLinearDamper gazebo::physics::SimbodyJoint::damper[MAX_JOINT_AXIS]

: for enforcing joint damping forces.

Set when we build the Simbody model. : Also, consider moving this into individual joint type subclass so we can specify custom dampers for special joints like ball joints.

10.218.4.3 SimTK::Transform gazebo::physics::SimbodyJoint::defxAB

default mobilizer pose

10.218.4.4 bool gazebo::physics::SimbodyJoint::isReversed

: if mobilizer, did it reverse parent&child? Set when we build the Simbody model.

10.218.4.5 SimTK::Force::MobilityLinearStop gazebo::physics::SimbodyJoint::limitForce[MAX_JOINT_AXIS]

: for enforcing joint stops Set when we build the Simbody model.

: Also, consider moving this into individual joint type subclass so we can specify custom dampers for special joints like ball joints. Assuming this is not used for BallJoints it's ok here for now.

10.218.4.6 SimTK::MobilizedBody gazebo::physics::SimbodyJoint::mobod

Use isValid() if we used a mobilizer Set when we build the Simbody model.

How this joint was modeled in the Simbody System. We used either a mobilizer or a constraint, but not both. The type of either one is the same as the joint type above.

10.218.4.7 bool gazebo::physics::SimbodyJoint::mustBreakLoopHere

Force Simbody to break a loop by using a weld constraint.

This flag is needed by SimbodyPhysics::MultibodyGraphMaker, so kept public.

10.218.4.8 bool gazebo::physics::SimbodyJoint::physicsInitialized

10.218.4.9 SimbodyPhysicsPtr gazebo::physics::SimbodyJoint::simbodyPhysics [protected]

keep a pointer to the simbody physics engine for convenience

10.218.4.10 SimTK::Force::MobilityLinearSpring gazebo::physics::SimbodyJoint::spring[MAX_JOINT_AXIS]

: Spring force element for enforcing joint stiffness.

The element is assigned when constructing Simbody model in SimbodyPhysics::AddDynamicModelToSimbodySystem. : Also, consider moving this into individual joint type subclass so we can specify custom springs for special joints like ball joints.

10.218.4.11 SimTK::MultibodySystem* gazebo::physics::SimbodyJoint::world [protected]

Simbody Multibody System.

Reimplemented from **gazebo::physics::Base** (p. 217).

10.218.4.12 SimTK::Transform gazebo::physics::SimbodyJoint::xCB

child body frame to mobilizer frame

10.218.4.13 SimTK::Transform gazebo::physics::SimbodyJoint::xPA

Normally A=F, B=M.

But if reversed, then B=F, A=M. parent body frame to mobilizer frame

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

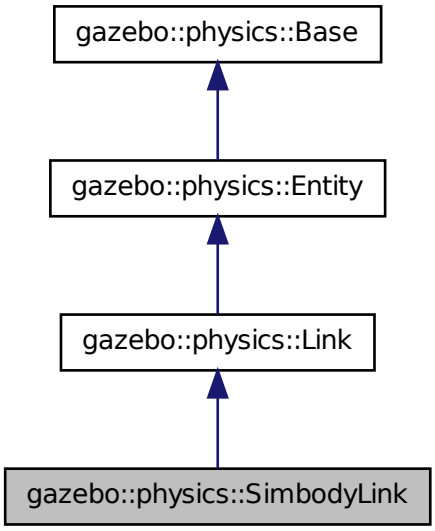
- **SimbodyJoint.hh**

10.219 gazebo::physics::SimbodyLink Class Reference

Simbody **Link** (p. 739) class.

`#include <SimbodyLink.hh>`

Inheritance diagram for gazebo::physics::SimbodyLink:



Public Member Functions

- **SimbodyLink** (**EntityPtr** _parent)

  Constructor.
- virtual ~**SimbodyLink** ()

  Destructor.
- virtual void **AddForce** (const **math::Vector3** &_force)

  Add a force to the body.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)

  Add a force to the body at position expressed to the body's own frame of reference.
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)

  Add a force to the body using a global position.
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)

  Add a force to the body, components are relative to the body's own frame of reference.

- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)

  Add a torque to the body, components are relative to the body's own frame of reference.
- virtual void **AddTorque** (const **math::Vector3** &_torque)

  Add a torque to the body.
- virtual void **Fini** ()

  Finalize the body.
- SimTK::MassProperties **GetEffectiveMassProps** (int _numFragments) const
- virtual bool **GetEnabled** () const

  Get whether this body is enabled in the physics engine.
- virtual bool **GetGravityMode** () const

  Get the gravity mode.
- SimTK::MassProperties **GetMassProperties** () const

  Convert Gazebo Inertia to Simbody MassProperties Where Simbody MassProperties contains mass, center of mass location, and unit inertia about body origin.
- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the entity in the world frame.
- virtual **math::Vector3 GetWorldCoGLinearVel** () const

  Get the linear velocity at the body's center of gravity in the world frame.
- virtual **math::Vector3 GetWorldForce** () const

  Get the force applied to the body in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** (const **math::Vector3** &_vector3) const

  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

  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

  Get the torque applied to the body in the world frame.
- virtual void **Init** ()

  Initialize the body.
- virtual void **Load** (sdf::ElementPtr _ptr)

  Load the body based on an SDF element.
- virtual void **OnPoseChange** ()

  This function is called when the entity's (or one of its parents) pose of the parent has changed.
- virtual void **RestoreSimbodyState** (SimTK::State &_state)
- virtual void **SaveSimbodyState** (const SimTK::State &_state)
- virtual void **SetAngularDamping** (double _damping)

  Set the angular damping factor.

- virtual void **SetAngularVel** (const **math::Vector3** &_vel)

  Set the angular velocity of the body.
- virtual void **SetAutoDisable** (bool _disable)

  Allow the link to auto disable.
- void **SetDirtyPose** (const **math::Pose** &_pose)
- virtual void **SetEnabled** (bool enable) const

  Set whether this body is enabled.
- virtual void **SetForce** (const **math::Vector3** &_force)

  Set the force applied to the body.
- virtual void **SetGravityMode** (bool _mode)

  Set whether gravity affects this body.
- virtual void **SetLinearDamping** (double _damping)

  Set the linear damping factor.
- virtual void **SetLinearVel** (const **math::Vector3** &_vel)

  Set the linear velocity of the body.
- virtual void **SetLinkStatic** (bool _static)

  If the inboard body of this link is ground, simply lock the inboard joint to freeze it to ground.
- virtual void **SetSelfCollide** (bool _collide)

  Set whether this body will collide with others in the model.
- virtual void **SetTorque** (const **math::Vector3** &_force)

  Set the torque applied to the body.

Public Attributes

- SimTK::MobilizedBody **masterMobod**
- bool **mustBeBaseLink**

  : Force this link to be a base body, where its inboard body is the world with 6DOF.
- bool **physicsInitialized**
- std::vector< SimTK::MobilizedBody > **slaveMobods**
- std::vector < SimTK::Constraint::Weld > **slaveWelds**

10.219.1 Detailed Description

Simbody **Link** (p. 739) class.

10.219.2 Constructor & Destructor Documentation

10.219.2.1 gazebo::physics::SimbodyLink::SimbodyLink (EntityPtr *_parent*)

Constructor.

10.219.2.2 virtual gazebo::physics::SimbodyLink::~SimbodyLink ( ) [virtual]

Destructor.

10.219.3 Member Function Documentation

10.219.3.1 virtual void gazebo::physics::SimbodyLink::AddForce (const math::Vector3 & _force) [virtual]

Add a force to the body.

Parameters

in	*_force*	Force to add.

Implements **gazebo::physics::Link** (p. 747).

10.219.3.2 virtual void gazebo::physics::SimbodyLink::AddForceAtRelativePosition (const math::Vector3 & _force, const math::Vector3 & _relPos) [virtual]

Add a force to the body at position expressed to the body's own frame of reference.

Parameters

in	*_force*	Force to add.
in	*_relPos*	Position on the link to add the force.

Implements **gazebo::physics::Link** (p. 747).

10.219.3.3 virtual void gazebo::physics::SimbodyLink::AddForceAtWorldPosition (const math::Vector3 & _force, const math::Vector3 & _pos) [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.

Implements **gazebo::physics::Link** (p. 747).

10.219.3.4 virtual void gazebo::physics::SimbodyLink::AddRelativeForce (const math::Vector3 & *_force*) [virtual]

Add a force to the body, components are relative to the body's own frame of reference.

Parameters

in	*_force*	Force to add.

Implements **gazebo::physics::Link** (p. 748).

10.219.3.5 virtual void gazebo::physics::SimbodyLink::AddRelativeTorque (const math::Vector3 & *_torque*) [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.

Implements **gazebo::physics::Link** (p. 748).

10.219.3.6 virtual void gazebo::physics::SimbodyLink::AddTorque (const math::Vector3 & *_torque*) [virtual]

Add a torque to the body.

Parameters

in	*_torque*	Torque value to add to the link.

Implements **gazebo::physics::Link** (p. 748).

10.219.3.7 virtual void gazebo::physics::SimbodyLink::Fini ( ) [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Link** (p. 750).

10.219.3.8 SimTK::MassProperties gazebo::physics::SimbodyLink::GetEffectiveMassProps (int *_numFragments*) const

10.219.3.9 virtual bool gazebo::physics::SimbodyLink::GetEnabled ( ) const [virtual]

Get whether this body is enabled in the physics engine.

Returns

True if the link is enabled.

Implements **gazebo::physics::Link** (p. 753).

10.219.3.10 virtual bool gazebo::physics::SimbodyLink::GetGravityMode ( ) const [virtual]

Get the gravity mode.

Returns

True if gravity is enabled.

Implements **gazebo::physics::Link** (p. 753).

10.219.3.11 SimTK::MassProperties gazebo::physics::SimbodyLink::GetMassProperties ( ) const

Convert Gazebo Inertia to Simbody MassProperties Where Simbody MassProperties contains mass, center of mass location, and unit inertia about body origin.

10.219.3.12 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the entity in the world frame.

Returns

A **math::Vector3** (p. 1440) for the velocity.

Reimplemented from **gazebo::physics::Entity** (p. 508).

10.219.3.13 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldCoGLinearVel ( ) const [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.

Implements **gazebo::physics::Link** (p. 757).

10.219.3.14 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldForce ( ) const [virtual]

Get the force applied to the body in the world frame.

Returns

Force applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 758).

10.219.3.15 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldLinearVel (const math::Vector3 & *_offset*) const [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. 739) frame will be returned.

Parameters

in	*_offset*	Offset of the point from the origin of the **Link** (p. 739) frame, expressed in the body-fixed frame.

Returns

Linear velocity of the point on the body

Implements **gazebo::physics::Link** (p. 759).

10.219.3.16 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldLinearVel (const math::Vector3 & *_offset,* const math::Quaternion & *_q*) const [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.

Implements **gazebo::physics::Link** (p. 760).

10.219.3.17 virtual math::Vector3 gazebo::physics::SimbodyLink::GetWorldTorque ( ) const [virtual]

Get the torque applied to the body in the world frame.

Returns

Torque applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 760).

10.219.3.18 virtual void gazebo::physics::SimbodyLink::Init ( ) [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Link** (p. 760).

10.219.3.19 virtual void gazebo::physics::SimbodyLink::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the body based on an SDF element.

Parameters

in	*_sdf*	SDF parameters.

Reimplemented from **gazebo::physics::Link** (p. 761).

10.219.3.20 virtual void gazebo::physics::SimbodyLink::OnPoseChange ( ) [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Reimplemented from **gazebo::physics::Link** (p. 761).

10.219.3.21 virtual void gazebo::physics::SimbodyLink::RestoreSimbodyState (SimTK::State & *_state*) [virtual]

10.219.3.22 virtual void gazebo::physics::SimbodyLink::SaveSimbodyState (const SimTK::State & *_state*) [virtual]

10.219.3.23 virtual void gazebo::physics::SimbodyLink::SetAngularDamping (double *_damping*) [virtual]

Set the angular damping factor.

Parameters

in	*_damping*	Angular damping factor.

Implements **gazebo::physics::Link** (p. 763).

10.219.3.24 virtual void gazebo::physics::SimbodyLink::SetAngularVel (const math::Vector3 & *_vel*) [virtual]

Set the angular velocity of the body.

Parameters

in	*_vel*	Angular velocity.

Implements **gazebo::physics::Link** (p. 763).

10.219.3.25 virtual void gazebo::physics::SimbodyLink::SetAutoDisable (bool *_disable*) [virtual]

Allow the link to auto disable.

Parameters

in	*_disable*	If true, the link is allowed to auto disable.

Implements **gazebo::physics::Link** (p. 763).

10.219.3.26 void gazebo::physics::SimbodyLink::SetDirtyPose (const math::Pose & _pose)

10.219.3.27 virtual void gazebo::physics::SimbodyLink::SetEnabled (bool _enable) const [virtual]

Set whether this body is enabled.

Parameters

in	*_enable*	True to enable the link in the physics engine.

Implements **gazebo::physics::Link** (p. 764).

10.219.3.28 virtual void gazebo::physics::SimbodyLink::SetForce (const math::Vector3 & _force) [virtual]

Set the force applied to the body.

Parameters

in	*_force*	Force value.

Implements **gazebo::physics::Link** (p. 764).

10.219.3.29 virtual void gazebo::physics::SimbodyLink::SetGravityMode (bool _mode) [virtual]

Set whether gravity affects this body.

Parameters

in	*_mode*	True to enable gravity.

Implements **gazebo::physics::Link** (p. 765).

10.219.3.30 virtual void gazebo::physics::SimbodyLink::SetLinearDamping (double _damping) [virtual]

Set the linear damping factor.

Parameters

in	*_damping*	Linear damping factor.

Implements **gazebo::physics::Link** (p. 766).

10.219.3.31 virtual void gazebo::physics::SimbodyLink::SetLinearVel (const math::Vector3 & *_vel*) [virtual]

Set the linear velocity of the body.

Parameters

in	*_vel*	Linear velocity.

Implements **gazebo::physics::Link** (p. 766).

10.219.3.32 virtual void gazebo::physics::SimbodyLink::SetLinkStatic (bool *_static*) [virtual]

If the inboard body of this link is ground, simply lock the inboard joint to freeze it to ground.

Otherwise, add a weld constraint to simulate freeze to ground effect.

Parameters

in	*_static*	if true, freeze link to ground. Otherwise unfreeze link.

Implements **gazebo::physics::Link** (p. 766).

10.219.3.33 virtual void gazebo::physics::SimbodyLink::SetSelfCollide (bool *_collide*) [virtual]

Set whether this body will collide with others in the model.

Parameters

in	*_collid*	True to enable collisions.

Implements **gazebo::physics::Link** (p. 767).

10.219.3.34 virtual void gazebo::physics::SimbodyLink::SetTorque (const math::Vector3 & *_torque*) [virtual]

Set the torque applied to the body.

Parameters

in	*_torque*	Torque value.

Implements **gazebo::physics::Link** (p. 768).

10.219.4 Member Data Documentation

10.219.4.1 SimTK::MobilizedBody gazebo::physics::SimbodyLink::masterMobod

10.219.4.2 bool gazebo::physics::SimbodyLink::mustBeBaseLink

: Force this link to be a base body, where its inboard body is the world with 6DOF.

10.219.4.3 bool gazebo::physics::SimbodyLink::physicsInitialized

10.219.4.4 std::vector<SimTK::MobilizedBody> gazebo::physics::SimbodyLink::slaveMobods

10.219.4.5 std::vector<SimTK::Constraint::Weld> gazebo::physics::SimbodyLink::slaveWelds

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

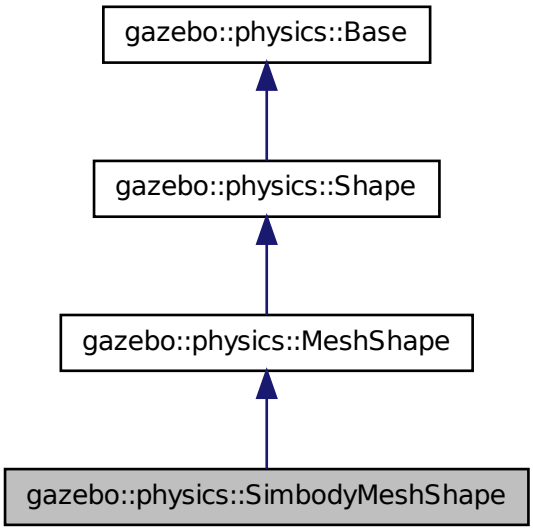
- **SimbodyLink.hh**

10.220 gazebo::physics::SimbodyMeshShape Class Reference

Triangle mesh collision.

`#include <SimbodyMeshShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyMeshShape:



Public Member Functions

- **SimbodyMeshShape** (**CollisionPtr** _parent)
  Constructor.
- virtual ~**SimbodyMeshShape** ()
  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)
  Load.

Protected Member Functions

- virtual void **Init** ()

10.220.1 Detailed Description

Triangle mesh collision.

10.220.2 Constructor & Destructor Documentation

10.220.2.1 gazebo::physics::SimbodyMeshShape::SimbodyMeshShape (CollisionPtr _parent)

Constructor.

10.220.2.2 virtual gazebo::physics::SimbodyMeshShape::~SimbodyMeshShape ( ) [virtual]

Destructor.

10.220.3 Member Function Documentation

10.220.3.1 virtual void gazebo::physics::SimbodyMeshShape::Init ( ) [protected, virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::MeshShape** (p. 844).

10.220.3.2 virtual void gazebo::physics::SimbodyMeshShape::Load (sdf::ElementPtr _sdf) [virtual]

Load.

Parameters

in	*node*	Pointer to an SDF parameters

Reimplemented from **gazebo::physics::Base** (p. 212).

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

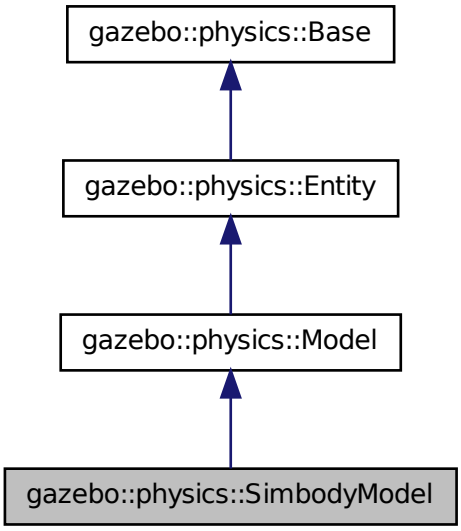
- **SimbodyMeshShape.hh**

10.221 gazebo::physics::SimbodyModel Class Reference

A model is a collection of links, joints, and plugins.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::SimbodyModel:



Public Member Functions

- **SimbodyModel** (**BasePtr** _parent)

  Constructor.
- virtual ~**SimbodyModel** ()

  Destructor.
- virtual void **Init** ()

  Initialize the model.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the model.

10.221.1 Detailed Description

A model is a collection of links, joints, and plugins.

10.221.2 Constructor & Destructor Documentation

10.221.2.1 gazebo::physics::SimbodyModel::SimbodyModel (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent object.

10.221.2.2 virtual gazebo::physics::SimbodyModel::~SimbodyModel ( ) [virtual]

Destructor.

10.221.3 Member Function Documentation

10.221.3.1 virtual void gazebo::physics::SimbodyModel::Init ( ) [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Model** (p. 857).

10.221.3.2 virtual void gazebo::physics::SimbodyModel::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the model.

Parameters

in	*_sdf*	SDF parameters to load from.

Reimplemented from **gazebo::physics::Model** (p. 857).

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

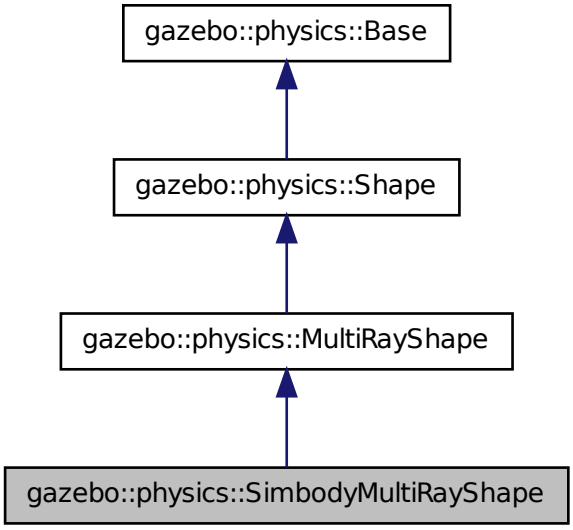
- **SimbodyModel.hh**

10.222 gazebo::physics::SimbodyMultiRayShape Class Reference

Simbody specific version of **MultiRayShape** (p. 901).

`#include <SimbodyMultiRayShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyMultiRayShape:



Public Member Functions

- **SimbodyMultiRayShape** (**CollisionPtr parent**)

  Constructor.
- virtual ~**SimbodyMultiRayShape** ()

  Destructor.
- virtual void **UpdateRays** ()

  Physics engine specific method for updating the rays.

Protected Member Functions

- virtual void **AddRay** (const **math::Vector3** &_start, const **math::Vector3** &_end)

  Add a ray to the collision.

10.222.1 Detailed Description

Simbody specific version of **MultiRayShape** (p. 901).

10.222.2 Constructor & Destructor Documentation

10.222.2.1 gazebo::physics::SimbodyMultiRayShape::SimbodyMultiRayShape (CollisionPtr *parent*)

Constructor.

10.222.2.2 virtual gazebo::physics::SimbodyMultiRayShape::~SimbodyMultiRayShape ( ) [virtual]

Destructor.

10.222.3 Member Function Documentation

10.222.3.1 virtual void gazebo::physics::SimbodyMultiRayShape::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.

Reimplemented from **gazebo::physics::MultiRayShape** (p. 904).

10.222.3.2 virtual void gazebo::physics::SimbodyMultiRayShape::UpdateRays ( ) [virtual]

Physics engine specific method for updating the rays.

Implements **gazebo::physics::MultiRayShape** (p. 909).

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

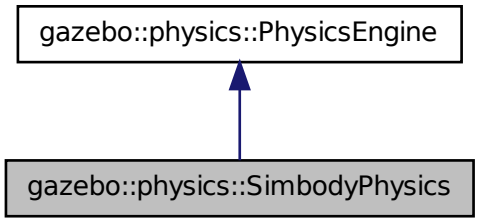
- **SimbodyMultiRayShape.hh**

10.223 gazebo::physics::SimbodyPhysics Class Reference

Simbody physics engine.

`#include <SimbodyPhysics.hh>`

Inheritance diagram for gazebo::physics::SimbodyPhysics:



Public Member Functions

- **SimbodyPhysics** (**WorldPtr** _world)

  Constructor.
- virtual ∼**SimbodyPhysics** ()

  Destructor.
- virtual **CollisionPtr CreateCollision** (const std::string &_type, **LinkPtr** _body)

  Create a collision.
- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent)

  Create a new joint.
- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)

  Create a new body.
- virtual **ModelPtr CreateModel** (**BasePtr** _parent)

  Create a new model.
- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)

  *Create a **physics::Shape** (p. 1161) object.*
- virtual void **DebugPrint** () const

  Debug print out of the physic engine state.
- virtual void **Fini** ()

Finilize the physics engine.

- SimTK::MultibodySystem * **GetDynamicsWorld** () const

  Register a joint with the dynamics world.

- virtual boost::any **GetParam** (const std::string &_key) const

  Get an parameter of the physics engine.

- virtual std::string **GetType** () const

  Return the physics engine type (ode|bullet|dart|simbody).

- virtual void **Init** ()

  Initialize the physics engine.

- virtual void **InitForThread** ()

  Init the engine for threads.

- void **InitModel** (const **physics::ModelPtr** _model)

  *Add a **Model** (p. 846) to the Simbody system.*

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the physics engine.

- virtual void **Reset** ()

  Rest the physics engine.

- virtual void **SetGravity** (const **gazebo::math::Vector3** &_gravity)

  Set the gavity vector.

- virtual bool **SetParam** (const std::string &_key, const boost::any &_value)

  Set a parameter of the physics engine.

- virtual void **SetSeed** (uint32_t _seed)

  Set the random number seed for the physics engine.

- virtual void **UpdateCollision** ()

  Update the physics engine collision.

- virtual void **UpdatePhysics** ()

  Update the physics engine.

Static Public Member Functions

- static SimTK::Transform **GetPose** (sdf::ElementPtr _element)

  If the given element contains a <pose> element, return it as a Transform.

- static std::string **GetTypeString** (unsigned int _type)

  *Convert **Base::GetType()** (p. 211) to string, this is needed by the MultibodyGraphMaker.*

- static std::string **GetTypeString** (**physics::Base::EntityType** _type)

  *Convert **Base::GetType()** (p. 211) to string, this is needed by the MultibodyGraphMaker.*

- static SimTK::Transform **Pose2Transform** (const **math::Pose** &_pose)

  Convert the given pose in x,y,z,thetax,thetay,thetaz format to a Simbody Transform.

- static SimTK::Quaternion **QuadToQuad** (const **math::Quaternion** &_q)

  *Convert **gazebo::math::Quaternion** (p. 1029) to SimTK::Quaternion.*
- static **math::Quaternion QuadToQuad** (const SimTK::Quaternion &_q)

  *Convert SimTK::Quaternion to **gazebo::math::Quaternion** (p. 1029).*
- static **math::Pose Transform2Pose** (const SimTK::Transform &_xAB)

  Convert a Simbody transform to a pose in x,y,z, thetax,thetay,thetaz format.
- static **math::Vector3 Vec3ToVector3** (const SimTK::Vec3 &_v)

  *Convert SimTK::Vec3 to **gazebo::math::Vector3** (p. 1440).*
- static SimTK::Vec3 **Vector3ToVec3** (const **math::Vector3** &_v)

  *Convert **gazebo::math::Vector3** (p. 1440) to SimTK::Vec3.*

Public Attributes

- SimTK::CompliantContactSubsystem **contact**
- SimTK::Force::DiscreteForces **discreteForces**
- SimTK::GeneralForceSubsystem **forces**
- SimTK::Force::Gravity **gravity**
- SimTK::Integrator ∗ **integ**
- SimTK::SimbodyMatterSubsystem **matter**
- bool **simbodyPhysicsInitialized**

  true if initialized
- bool **simbodyPhysicsStepped**
- SimTK::MultibodySystem **system**
- SimTK::ContactTrackerSubsystem **tracker**

Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &_msg)

  virtual callback for gztopic "∼/physics".
- virtual void **OnRequest** (ConstRequestPtr &_msg)

  virtual callback for gztopic "∼/request".

10.223.1 Detailed Description

Simbody physics engine.

10.223.2 Constructor & Destructor Documentation

10.223.2.1 gazebo::physics::SimbodyPhysics::SimbodyPhysics (WorldPtr _world)

Constructor.

10.223.2.2 virtual gazebo::physics::SimbodyPhysics::~SimbodyPhysics ( ) [virtual]

Destructor.

10.223.3 Member Function Documentation

10.223.3.1 virtual CollisionPtr gazebo::physics::SimbodyPhysics::CreateCollision (const std::string & _shapeType, LinkPtr _link) [virtual]

Create a collision.

Parameters

in	*_shapeType*	Type of collision to create.
in	*_link*	Parent link.

Implements **gazebo::physics::PhysicsEngine** (p. 963).

10.223.3.2 virtual JointPtr gazebo::physics::SimbodyPhysics::CreateJoint (const std::string & _type, ModelPtr _parent) [virtual]

Create a new joint.

Parameters

in	*_type*	Type of joint to create.
in	*_parent*	**Model** (p. 846) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 964).

10.223.3.3 virtual LinkPtr gazebo::physics::SimbodyPhysics::CreateLink (ModelPtr _parent) [virtual]

Create a new body.

Parameters

in	*_parent*	Parent model for the link.

Implements **gazebo::physics::PhysicsEngine** (p. 964).

10.223.3.4 virtual ModelPtr gazebo::physics::SimbodyPhysics::CreateModel (BasePtr *_base*) [virtual]

Create a new model.

Parameters

in	*_base*	Boost shared pointer to a new model.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 964).

10.223.3.5 virtual ShapePtr gazebo::physics::SimbodyPhysics::CreateShape (const std::string & *_shapeType,* CollisionPtr *_collision*) [virtual]

Create a **physics::Shape** (p. 1161) object.

Parameters

in	*_shapeType*	Type of shape to create.
in	*_collision*	**Collision** (p. 295) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 965).

10.223.3.6 virtual void gazebo::physics::SimbodyPhysics::DebugPrint ( ) const [virtual]

Debug print out of the physic engine state.

Implements **gazebo::physics::PhysicsEngine** (p. 965).

10.223.3.7 virtual void gazebo::physics::SimbodyPhysics::Fini ( ) [virtual]

Finilize the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 965).

10.223.3.8 SimTK::MultibodySystem* gazebo::physics::SimbodyPhysics::GetDynamicsWorld ( ) const

Register a joint with the dynamics world.

10.223.3.9 virtual boost::any gazebo::physics::SimbodyPhysics::GetParam (const std::string & *_key*) const [virtual]

Get an parameter of the physics engine.

Parameters

in	*_attr*	String key

See also

SetParam (p. 1235)

Returns

The value of the parameter

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 967).

10.223.3.10 static SimTK::Transform gazebo::physics::SimbodyPhysics::GetPose (sdf::ElementPtr *_element*) [static]

If the given element contains a <pose> element, return it as a Transform.

Otherwise return the identity Transform. If there is more than one <pose> element, only the first one is processed.

10.223.3.11 virtual std::string gazebo::physics::SimbodyPhysics::GetType ( ) const [virtual]

Return the physics engine type (ode|bullet|dart|simbody).

Returns

Type of the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 968).

10.223.3.12 static std::string gazebo::physics::SimbodyPhysics::GetTypeString (unsigned int *_type*) [static]

Convert **Base::GetType()** (p. 211) to string, this is needed by the MultibodyGraphMaker.

Parameters

in	*_type*	**Joint** (p. 669) type returned by **Joint::GetType()** (p. 211).

Returns

a hard-coded string needed by the MultibodyGraphMaker.

10.223.3.13 static std::string gazebo::physics::SimbodyPhysics::GetTypeString (physics::Base::EntityType *_type*) [static]

Convert **Base::GetType()** (p. 211) to string, this is needed by the MultibodyGraphMaker.

Parameters

in	*_type*	**Joint** (p. 669) type returned by **Joint::GetType()** (p. 211).

Returns

a hard-coded string needed by the MultibodyGraphMaker.

10.223.3.14 virtual void gazebo::physics::SimbodyPhysics::Init ( ) [virtual]

Initialize the physics engine.

Implements **gazebo::physics::PhysicsEngine** (p. 969).

10.223.3.15 virtual void gazebo::physics::SimbodyPhysics::InitForThread ( ) [virtual]

Init the engine for threads.

Implements **gazebo::physics::PhysicsEngine** (p. 969).

10.223.3.16 void gazebo::physics::SimbodyPhysics::InitModel (const physics::ModelPtr *_model*)

Add a **Model** (p. 846) to the Simbody system.

Parameters

in	*_model*	Pointer to the model to add into Simbody.

10.223.3.17 virtual void gazebo::physics::SimbodyPhysics::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the physics engine.

Parameters

in	*_sdf*	Pointer to the SDF parameters.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 969).

10.223.3.18 virtual void gazebo::physics::SimbodyPhysics::OnPhysicsMsg (ConstPhysicsPtr & *_msg*) [protected, virtual]

virtual callback for gztopic "~/physics".

Parameters

in	*_msg*	Physics message.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 969).

10.223.3.19 virtual void gazebo::physics::SimbodyPhysics::OnRequest (ConstRequestPtr & *_msg*) [protected, virtual]

virtual callback for gztopic "~/request".

Parameters

in	*_msg*	Request message.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 970).

10.223.3.20 static SimTK::Transform gazebo::physics::SimbodyPhysics::Pose2Transform (const math::Pose & *_pose*) [static]

Convert the given pose in x,y,z,thetax,thetay,thetaz format to a Simbody Transform.

The rotation angles are interpreted as a body-fixed sequence, meaning we rotation about x, then about the new y, then about the now twice-rotated z.

Parameters

in	*_pose*	Gazeb's **math::Pose** (p. 995) object

Returns

Simbody's SimTK::Transform object

10.223.3.21 static SimTK::Quaternion gazebo::physics::SimbodyPhysics::QuadToQuad (const math::Quaternion & _q) [static]

Convert **gazebo::math::Quaternion** (p. 1029) to SimTK::Quaternion.

Parameters

in	*_q*	Gazeb's **math::Quaternion** (p. 1029) object

Returns

Simbody's SimTK::Quaternion object

10.223.3.22 static math::Quaternion gazebo::physics::SimbodyPhysics::QuadToQuad (const SimTK::Quaternion & _q) [static]

Convert SimTK::Quaternion to **gazebo::math::Quaternion** (p. 1029).

Parameters

in	*_q*	Simbody's SimTK::Quaternion object

Returns

Gazeb's **math::Quaternion** (p. 1029) object

10.223.3.23 virtual void gazebo::physics::SimbodyPhysics::Reset ( ) [virtual]

Rest the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 970).

10.223.3.24 virtual void gazebo::physics::SimbodyPhysics::SetGravity (const gazebo::math::Vector3 & *_gravity*) [virtual]

Set the gavity vector.

Parameters

in	*_gravity*	New gravity vector.

Implements **gazebo::physics::PhysicsEngine** (p. 971).

10.223.3.25 virtual bool gazebo::physics::SimbodyPhysics::SetParam (const std::string & *_key*, const boost::any & *_value*) [virtual]

Set a parameter of the physics engine.

See SetParam documentation for descriptions of duplicate parameters.

Parameters

in	*_key*	String key Below is a list of _key parameter definitions: 1. "solver_type" (string) - returns solver used by engine, e.g. "sequential_impulse' for Bullet, "quick" for ODE "Featherstone and Lemkes" for DART and "Spatial - Algebra and Elastic Foundation" for Simbody. -# "cfm" (double) - global CFM -# "erp" (double) - global ERP -# "precon_iters" (bool) - precondition iterations (experimental). 2. "iters" (int) - number of LCP PGS iterations. If sor_lcp_tolerance is negative, full iteration count is executed. Otherwise, PGS may stop iteration early if sor_lcp_tolerance is satisfied by the total RMS residual. 3. "sor" (double) - relaxation parameter for Projected - Gauss-Seidel (PGS) updates. 4. "contact_max_correcting_vel" (double) - truncates correction impulses from ERP by this value. 5. "contact_surface_layer" (double) - ERP is 0 for interpenetration depths below this value. 6. "max_contacts" (int) - max number of contact constraints between any pair of collision bodies. 7. "min_step_size" (double) - minimum internal step size. (defined but not used in ode). 8. "max_step_size" (double) - maximum physics step size when physics update step must return.
in	*_value*	The value to set to

Returns

true if SetParam is successful, false if operation fails.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 972).

10.223.3.26 virtual void gazebo::physics::SimbodyPhysics::SetSeed (uint32_t *_seed*) [virtual]

Set the random number seed for the physics engine.

Parameters

in	*_seed*	The random number seed.

Implements **gazebo::physics::PhysicsEngine** (p. 973).

10.223.3.27 static math::Pose gazebo::physics::SimbodyPhysics::Transform2Pose (const SimTK::Transform & _xAB) [static]

Convert a Simbody transform to a pose in x,y,z, thetax,thetay,thetaz format.

Parameters

in	*_xAB*	Simbody's SimTK::Transform object

Returns

Gazeb's **math::Pose** (p. 995) object

10.223.3.28 virtual void gazebo::physics::SimbodyPhysics::UpdateCollision ( ) [virtual]

Update the physics engine collision.

Implements **gazebo::physics::PhysicsEngine** (p. 974).

10.223.3.29 virtual void gazebo::physics::SimbodyPhysics::UpdatePhysics ( ) [virtual]

Update the physics engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 974).

10.223.3.30 static math::Vector3 gazebo::physics::SimbodyPhysics::Vec3ToVector3 (const SimTK::Vec3 & _v) [static]

Convert SimTK::Vec3 to **gazebo::math::Vector3** (p. 1440).

Parameters

in	*_v*	Simbody's SimTK::Vec3 object

Returns

Gazeb's **math::Vector3** (p. 1440) object

10.223.3.31 static SimTK::Vec3 gazebo::physics::SimbodyPhysics::Vector3ToVec3 (const math::Vector3 & _v) [static]

Convert **gazebo::math::Vector3** (p. 1440) to SimTK::Vec3.

Parameters

in	*_v*	Gazeb's **math::Vector3** (p. 1440) object

Returns

Simbody's SimTK::Vec3 object

10.223.4 Member Data Documentation

10.223.4.1 SimTK::CompliantContactSubsystem gazebo::physics::SimbodyPhysics::contact

10.223.4.2 SimTK::Force::DiscreteForces gazebo::physics::SimbodyPhysics::discreteForces

10.223.4.3 SimTK::GeneralForceSubsystem gazebo::physics::SimbodyPhysics::forces

10.223.4.4 SimTK::Force::Gravity gazebo::physics::SimbodyPhysics::gravity

10.223.4.5 SimTK:: Integrator* gazebo::physics::SimbodyPhysics::integ

10.223.4.6 SimTK::SimbodyMatterSubsystem gazebo::physics::SimbodyPhysics::matter

10.223.4.7 bool gazebo::physics::SimbodyPhysics::simbodyPhysicsInitialized

true if initialized

10.223.4.8 bool gazebo::physics::SimbodyPhysics::simbodyPhysicsStepped

10.223.4.9 SimTK::MultibodySystem gazebo::physics::SimbodyPhysics::system

10.223.4.10 SimTK::ContactTrackerSubsystem gazebo::physics::SimbodyPhysics::tracker

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

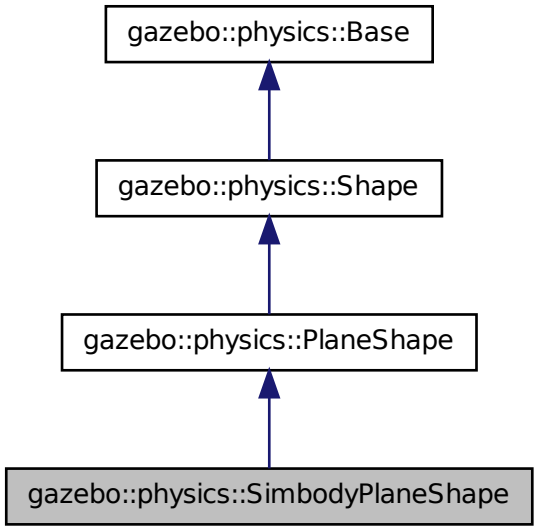
- **SimbodyPhysics.hh**

10.224 gazebo::physics::SimbodyPlaneShape Class Reference

Simbody collision for an infinite plane.

`#include <SimbodyPlaneShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyPlaneShape:



Public Member Functions

- **SimbodyPlaneShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**SimbodyPlaneShape** ()

  Destructor.
- virtual void **CreatePlane** ()

  Create the plane.
- virtual void **SetAltitude** (const **math::Vector3** &_pos)

  Set the altitude of the plane.

10.224.1 Detailed Description

Simbody collision for an infinite plane.

10.224.2 Constructor & Destructor Documentation

10.224.2.1 gazebo::physics::SimbodyPlaneShape::SimbodyPlaneShape (CollisionPtr *_parent*)

Constructor.

10.224.2.2 virtual gazebo::physics::SimbodyPlaneShape::~SimbodyPlaneShape ( ) [virtual]

Destructor.

10.224.3 Member Function Documentation

10.224.3.1 virtual void gazebo::physics::SimbodyPlaneShape::CreatePlane ( ) [virtual]

Create the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 989).

10.224.3.2 virtual void gazebo::physics::SimbodyPlaneShape::SetAltitude (const math::Vector3 & *_pos*) [virtual]

Set the altitude of the plane.

Parameters

in	*_pos*	Position of the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 991).

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

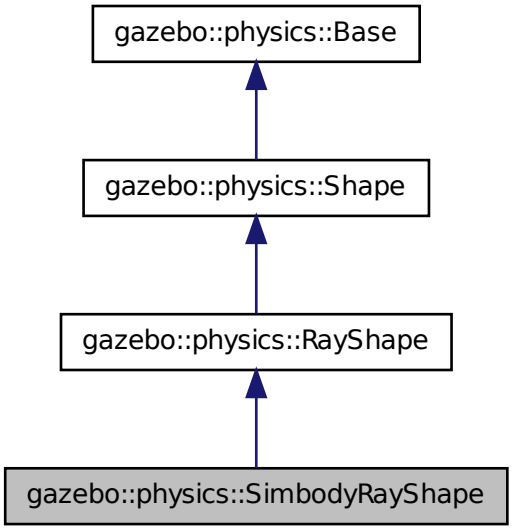
- **SimbodyPlaneShape.hh**

10.225 gazebo::physics::SimbodyRayShape Class Reference

Ray shape for simbody.

`#include <SimbodyRayShape.hh>`

Inheritance diagram for gazebo::physics::SimbodyRayShape:



Public Member Functions

- **SimbodyRayShape** (**PhysicsEnginePtr** _physicsEngine)

  Constructor.
- **SimbodyRayShape** (**CollisionPtr** _collision)

  Constructor.
- virtual ~**SimbodyRayShape** ()

  Destructor.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)

  Get the nearest intersection.
- virtual void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)

Set the ray based on starting and ending points relative to the body.

- virtual void **Update** ()

  Update the ray collision.

10.225.1 Detailed Description

Ray shape for simbody.

10.225.2 Constructor & Destructor Documentation

10.225.2.1 gazebo::physics::SimbodyRayShape::SimbodyRayShape (PhysicsEnginePtr *_physicsEngine*)

Constructor.

Parameters

in	*_physicsEngine*	Pointer to the physics engine.

10.225.2.2 gazebo::physics::SimbodyRayShape::SimbodyRayShape (CollisionPtr *_collision*)

Constructor.

Parameters

in	*_collision*	**Collision** (p. 295) the ray is attached to.

10.225.2.3 virtual gazebo::physics::SimbodyRayShape::~SimbodyRayShape ( ) [virtual]

Destructor.

10.225.3 Member Function Documentation

10.225.3.1 virtual void gazebo::physics::SimbodyRayShape::GetIntersection (double & *_dist*, std::string & *_entity*) [virtual]

Get the nearest intersection.

Parameters

out	*_dist*	Distance to the intersection.
out	*_entity*	Name of the entity the ray intersected with.

Implements **gazebo::physics::RayShape** (p. 1063).

10.225.3.2 virtual void gazebo::physics::SimbodyRayShape::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.

Reimplemented from **gazebo::physics::RayShape** (p. 1065).

10.225.3.3 virtual void gazebo::physics::SimbodyRayShape::Update ( ) [virtual]

Update the ray collision.

Implements **gazebo::physics::RayShape** (p. 1065).

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

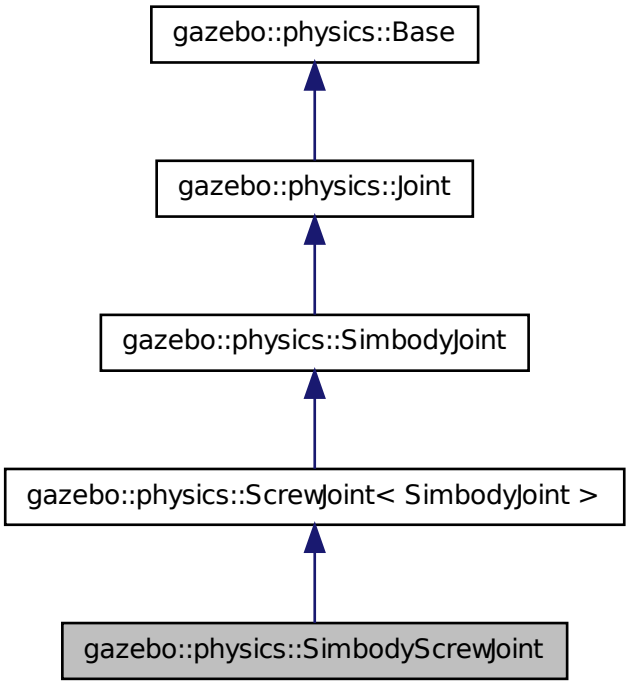
- **SimbodyRayShape.hh**

10.226 gazebo::physics::SimbodyScrewJoint Class Reference

A screw joint.

`#include <SimbodyScrewJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodyScrewJoint:



Public Member Functions

- **SimbodyScrewJoint** (SimTK::MultibodySystem *_world, **BasePtr** _parent)

  Constructor.
- virtual ~**SimbodyScrewJoint** ()

  Destructor.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.
- virtual **math::Angle GetHighStop** (unsigned int _index)

  Get the high stop of an axis(index).

- virtual **math::Angle GetLowStop** (unsigned int _index)

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual double **GetParam** (const std::string &_key, unsigned int _index)

  Get a non-generic parameter for the joint.
- virtual double **GetThreadPitch** (unsigned int)
- virtual double **GetThreadPitch** ()

  Get screw joint thread pitch.
- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual bool **SetHighStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- virtual bool **SetLowStop** (unsigned int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual void **SetMaxForce** (unsigned int _index, double _t)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual bool **SetParam** (const std::string &_key, unsigned int _index, const boost::any &_value)

  Set a non-generic parameter for the joint.
- virtual void **SetThreadPitch** (unsigned int _index, double _threadPitch)
- virtual void **SetThreadPitch** (double _threadPitch)

  Set screw joint thread pitch.
- virtual void **SetVelocity** (unsigned int _index, double _angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **ScrewJoint** (p. 1118).*
- virtual void **SetForceImpl** (unsigned int _index, double _force)

  *Set the force applied to this **physics::Joint** (p. 669).*

10.226.1 Detailed Description

A screw joint.

10.226.2 Constructor & Destructor Documentation

10.226.2.1 gazebo::physics::SimbodyScrewJoint::SimbodyScrewJoint (SimTK::MultibodySystem * _world, BasePtr _parent)

Constructor.

Parameters

in	*_world*	Pointer to the Simbody world.
in	*_parent*	Parent of the screw joint.

10.226.2.2 virtual gazebo::physics::SimbodyScrewJoint::~SimbodyScrewJoint ( ) [virtual]

Destructor.

10.226.3 Member Function Documentation

10.226.3.1 virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetAngleImpl (unsigned int _index) const [virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.226.3.2 virtual math::Vector3 gazebo::physics::SimbodyScrewJoint::GetGlobalAxis (unsigned int _index) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.226.3.3 virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetHighStop (unsigned int *_index*) [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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1198).

10.226.3.4 virtual math::Angle gazebo::physics::SimbodyScrewJoint::GetLowStop (unsigned int *_index*) [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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1200).

10.226.3.5 virtual double gazebo::physics::SimbodyScrewJoint::GetMaxForce (unsigned int *_index*) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.226.3.6 virtual double gazebo::physics::SimbodyScrewJoint::GetParam (const std::string & *_key,* unsigned int *_index*) [virtual]

Get a non-generic parameter for the joint.

Parameters

in	*_key*	String key.
in	*_index*	Index of the axis.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1200).

10.226.3.7 virtual double gazebo::physics::SimbodyScrewJoint::GetThreadPitch (unsigned *int*) [virtual]

10.226.3.8 virtual double gazebo::physics::SimbodyScrewJoint::GetThreadPitch ( ) [virtual]

Get screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. This must be implemented in a child class

Returns

_threadPitch Thread pitch value.

Implements **gazebo::physics::ScrewJoint**< **SimbodyJoint** > (p. 1119).

10.226.3.9 virtual double gazebo::physics::SimbodyScrewJoint::GetVelocity (unsigned int _index) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	_index	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.226.3.10 virtual void gazebo::physics::SimbodyScrewJoint::Load (sdf::ElementPtr _sdf) [protected, virtual]

Load a **ScrewJoint** (p. 1118).

Parameters

in	_sdf	SDF value to load from

Reimplemented from **gazebo::physics::ScrewJoint< SimbodyJoint >** (p. 1120).

10.226.3.11 virtual void gazebo::physics::SimbodyScrewJoint::SetAxis (unsigned int _index, const math::Vector3 & _axis) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	_index	Index of the axis to set.
in	_axis	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1202).

10.226.3.12 virtual void gazebo::physics::SimbodyScrewJoint::SetForceImpl (unsigned int _index, double _force) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implements **gazebo::physics::SimbodyJoint** (p. 1203).

10.226.3.13 virtual bool gazebo::physics::SimbodyScrewJoint::SetHighStop (unsigned 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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1204).

10.226.3.14 virtual bool gazebo::physics::SimbodyScrewJoint::SetLowStop (unsigned 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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1204).

10.226.3.15 virtual void gazebo::physics::SimbodyScrewJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMax-

Force. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.226.3.16 virtual bool gazebo::physics::SimbodyScrewJoint::SetParam (const std::string & *_key*, unsigned int *_index*, const boost::any & *_value*) [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.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1204).

10.226.3.17 virtual void gazebo::physics::SimbodyScrewJoint::SetThreadPitch (unsigned int *_index*, double *_threadPitch*) [virtual]

10.226.3.18 virtual void gazebo::physics::SimbodyScrewJoint::SetThreadPitch (double *_threadPitch*) [virtual]

Set screw joint thread pitch.

Thread Pitch is defined as angular motion per linear motion or rad / m in metric. - This must be implemented in a child class To clarify direction, these are modeling right handed threads with positive thread_pitch, i.e. the child **Link** (p. 739) is the nut (interior threads) while the parent **Link** (p. 739) is the bolt/screw (exterior threads).

Parameters

in	*_threadPitch*	Thread pitch value.

Implements **gazebo::physics::ScrewJoint**< **SimbodyJoint** > (p. 1120).

10.226.3.19 virtual void gazebo::physics::SimbodyScrewJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

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

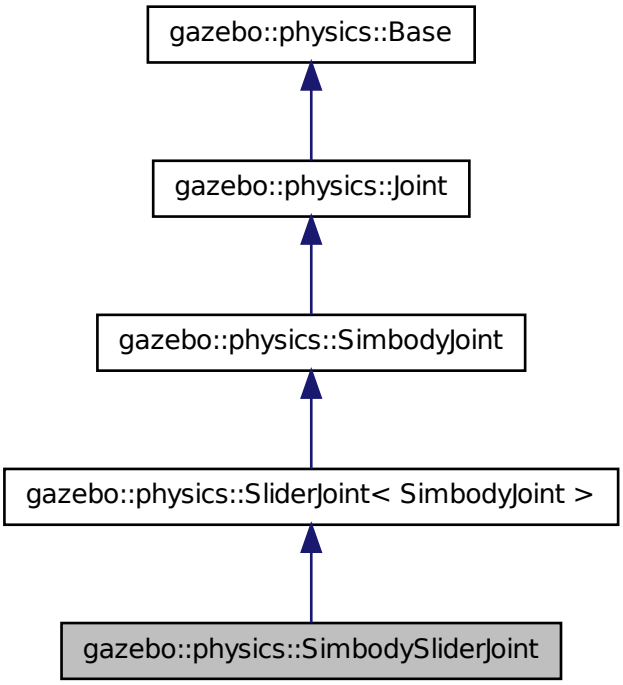
- **SimbodyScrewJoint.hh**

10.227 gazebo::physics::SimbodySliderJoint Class Reference

A slider joint.

`#include <SimbodySliderJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodySliderJoint:



Public Member Functions

- **SimbodySliderJoint** (SimTK::MultibodySystem ∗**world**, **BasePtr** _parent)

  Constructor.
- virtual ∼**SimbodySliderJoint** ()

  Destructor.
- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.
- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*

- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _t)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity*** (p. *701).*
- virtual void **SetVelocity** (unsigned int _index, double _rate)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **SliderJoint*** (p. *1294).*
- virtual void **SetForceImpl** (unsigned int _index, double _force)

  *Set the force applied to this **physics::Joint*** (p. *669).*

10.227.1 Detailed Description

A slider joint.

10.227.2 Constructor & Destructor Documentation

10.227.2.1 gazebo::physics::SimbodySliderJoint::SimbodySliderJoint (SimTK::MultibodySystem * *world*, BasePtr *_parent*)

Constructor.

Parameters

in	*_world*	Pointer to the Simbody world.
in	*_parent*	Parent of the screw joint.

10.227.2.2 virtual gazebo::physics::SimbodySliderJoint::~SimbodySliderJoint ( ) [virtual]

Destructor.

10.227.3 Member Function Documentation

10.227.3.1 virtual math::Angle gazebo::physics::SimbodySliderJoint::GetAngleImpl (unsigned int _index) const [virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.227.3.2 virtual math::Vector3 gazebo::physics::SimbodySliderJoint::GetGlobalAxis (unsigned int _index) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.227.3.3 virtual double gazebo::physics::SimbodySliderJoint::GetMaxForce (unsigned int _index) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.227.3.4 virtual double gazebo::physics::SimbodySliderJoint::GetVelocity (unsigned int *_index*) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.227.3.5 virtual void gazebo::physics::SimbodySliderJoint::Load (sdf::ElementPtr *_sdf*) [protected, virtual]

Load a **SliderJoint** (p. 1294).

Parameters

in	*_sdf*	SDF values to load from

Reimplemented from **gazebo::physics::SliderJoint**< **SimbodyJoint** > (p. 1295).

10.227.3.6 virtual void gazebo::physics::SimbodySliderJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1202).

10.227.3.7 virtual void gazebo::physics::SimbodySliderJoint::SetForceImpl (unsigned int _index, double _force) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implements **gazebo::physics::SimbodyJoint** (p. 1203).

10.227.3.8 virtual void gazebo::physics::SimbodySliderJoint::SetMaxForce (unsigned int _index, double _force) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.227.3.9 virtual void gazebo::physics::SimbodySliderJoint::SetVelocity (unsigned int _index, double _vel) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

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

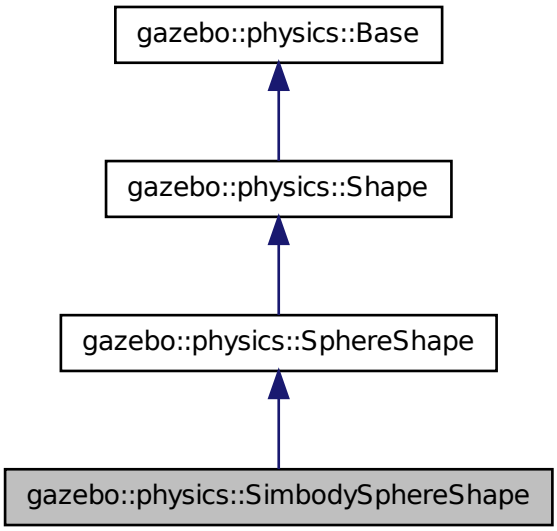
- **SimbodySliderJoint.hh**

10.228 gazebo::physics::SimbodySphereShape Class Reference

Simbody sphere collision.

`#include <SimbodySphereShape.hh>`

Inheritance diagram for gazebo::physics::SimbodySphereShape:



Public Member Functions

- **SimbodySphereShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**SimbodySphereShape** ()

  Destructor.

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

  Set the size.

10.228.1 Detailed Description

Simbody sphere collision.

10.228.2 Constructor & Destructor Documentation

10.228.2.1 gazebo::physics::SimbodySphereShape::SimbodySphereShape (CollisionPtr *_parent*) [inline]

Constructor.

Parameters

in	*_parent*	**Collision** (p. 295) parent pointer

10.228.2.2 virtual gazebo::physics::SimbodySphereShape::~SimbodySphereShape ( ) [inline, virtual]

Destructor.

10.228.3 Member Function Documentation

10.228.3.1 virtual void gazebo::physics::SimbodySphereShape::SetRadius (double *_radius*) [inline, virtual]

Set the size.

Parameters

in	*_radius*	Radius of the sphere.

Reimplemented from **gazebo::physics::SphereShape** (p. 1309).

References gazebo::math::equal(), gzerr, gzwarn, and gazebo::physics::SphereShape::SetRadius().

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

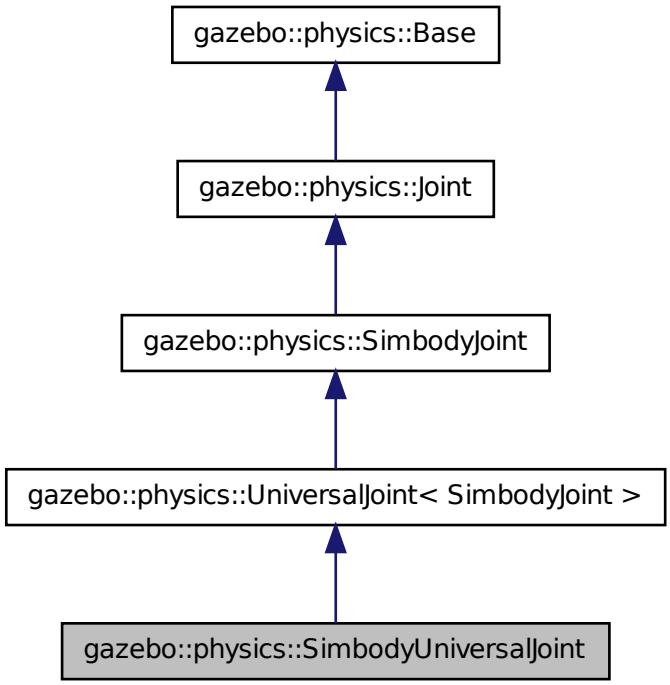
- **SimbodySphereShape.hh**

10.229 gazebo::physics::SimbodyUniversalJoint Class Reference

A simbody universal joint class.

`#include <SimbodyUniversalJoint.hh>`

Inheritance diagram for gazebo::physics::SimbodyUniversalJoint:



Public Member Functions

- **SimbodyUniversalJoint** (SimTK::MultibodySystem *_world, **BasePtr** _parent)

  Constructor.
- virtual ~**SimbodyUniversalJoint** ()

  Destuctor.

- virtual **math::Vector3 GetAnchor** (unsigned int _index) const

  Get the anchor point.
- virtual **math::Vector3 GetAxis** (unsigned int _index) const
- virtual **math::Vector3 GetGlobalAxis** (unsigned int _index) const

  Get the axis of rotation in global cooridnate frame.
- virtual double **GetMaxForce** (unsigned int _index)

  *Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual double **GetVelocity** (unsigned int _index) const

  Get the rotation rate of an axis(index)
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **UniversalJoint** (p. 1404).*
- virtual void **SetAxis** (unsigned int _index, const **math::Vector3** &_axis)

  Set the axis of rotation where axis is specified in local joint frame.
- virtual void **SetMaxForce** (unsigned int _index, double _t)

  *Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).*
- virtual void **SetVelocity** (unsigned int _index, double _rate)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (unsigned int _index) const

  Get the angle of an axis helper function.
- virtual void **SetForceImpl** (unsigned int _index, double _torque)

  *Set the force applied to this **physics::Joint** (p. 669).*

10.229.1 Detailed Description

A simbody universal joint class.

10.229.2 Constructor & Destructor Documentation

10.229.2.1 gazebo::physics::SimbodyUniversalJoint::SimbodyUniversalJoint (SimTK::MultibodySystem * *_world*, BasePtr *_parent*)

Constructor.

Parameters

in	*_world*	Pointer to the Simbody world.
in	*_parent*	Parent of the screw joint.

10.229.2.2 virtual gazebo::physics::SimbodyUniversalJoint::~SimbodyUniversalJoint ( ) [virtual]

Destuctor.

10.229.3 Member Function Documentation

10.229.3.1 virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetAnchor (unsigned int _index) const [virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1197).

10.229.3.2 virtual math::Angle gazebo::physics::SimbodyUniversalJoint::GetAngleImpl (unsigned int _index) const [protected, virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implements **gazebo::physics::Joint** (p. 681).

10.229.3.3 virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetAxis (unsigned int _index) const [virtual]

10.229.3.4 virtual math::Vector3 gazebo::physics::SimbodyUniversalJoint::GetGlobalAxis (unsigned int _index) const [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.

Implements **gazebo::physics::Joint** (p. 684).

10.229.3.5 virtual double gazebo::physics::SimbodyUniversalJoint::GetMaxForce (unsigned int _index) [virtual]

Get the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implements **gazebo::physics::Joint** (p. 688).

10.229.3.6 virtual double gazebo::physics::SimbodyUniversalJoint::GetVelocity (unsigned int _index) const [virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implements **gazebo::physics::Joint** (p. 691).

10.229.3.7 virtual void gazebo::physics::SimbodyUniversalJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load a **UniversalJoint** (p. 1404).

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::UniversalJoint**< **SimbodyJoint** > (p. 1406).

10.229.3.8 virtual void gazebo::physics::SimbodyUniversalJoint::SetAxis (unsigned int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the axis of rotation where axis is specified in local joint frame.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Vector in local joint frame of axis direction (must have length greater than zero).

Reimplemented from **gazebo::physics::SimbodyJoint** (p. 1202).

10.229.3.9 virtual void gazebo::physics::SimbodyUniversalJoint::SetForceImpl (unsigned int *_index*, double *_force*) [protected, virtual]

Set the force applied to this **physics::Joint** (p. 669).

Note that the unit of force should be consistent with the rest of the simulation scales. Force is additive (multiple calls to SetForceImpl to the same joint in the same time step will accumulate forces on that **Joint** (p. 669)).

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value. internal force, e.g. damping forces. This way, Joint::appliedForce keep track of external forces only.

Implements **gazebo::physics::SimbodyJoint** (p. 1203).

10.229.3.10 virtual void gazebo::physics::SimbodyUniversalJoint::SetMaxForce (unsigned int *_index*, double *_force*) [virtual]

Set the max allowed force of an axis(index) when using **Joint::SetVelocity** (p. 701).

Current implementation in Bullet and ODE is enforced using impulses, which enforces force/torque limits when calling **Joint::SetVelocity** (p. 701). Current implementation is engine dependent. See for example ODE implementation in ODEHingeJoint::SetMaxForce. Note this functionality is not implemented in DART and Simbody. Note that the unit of force should be consistent with the rest of the simulation scales.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implements **gazebo::physics::Joint** (p. 697).

10.229.3.11 virtual void gazebo::physics::SimbodyUniversalJoint::SetVelocity (unsigned int *_index*, double *_vel*) [virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implements **gazebo::physics::Joint** (p. 701).

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

- **SimbodyUniversalJoint.hh**

10.230 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.

10.230.1 Detailed Description

A simulation time event

10.230.2 Member Data Documentation

10.230.2.1 boost::condition_variable* gazebo::sensors::SimTimeEvent::condition

The condition to notify.

10.230.2.2 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**

10.231 gazebo::sensors::SimTimeEventHandler Class Reference

Monitors simulation time, and notifies conditions when a specified time has been reached.

`#include <SensorManager.hh>`

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.

10.231.1 Detailed Description

Monitors simulation time, and notifies conditions when a specified time has been reached.

10.231.2 Constructor & Destructor Documentation

10.231.2.1 gazebo::sensors::SimTimeEventHandler::SimTimeEventHandler ( )

Constructor.

10.231.2.2 virtual gazebo::sensors::SimTimeEventHandler::~SimTimeEventHandler ( ) [virtual]

Destructor.

10.231.3 Member Function Documentation

10.231.3.1 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:

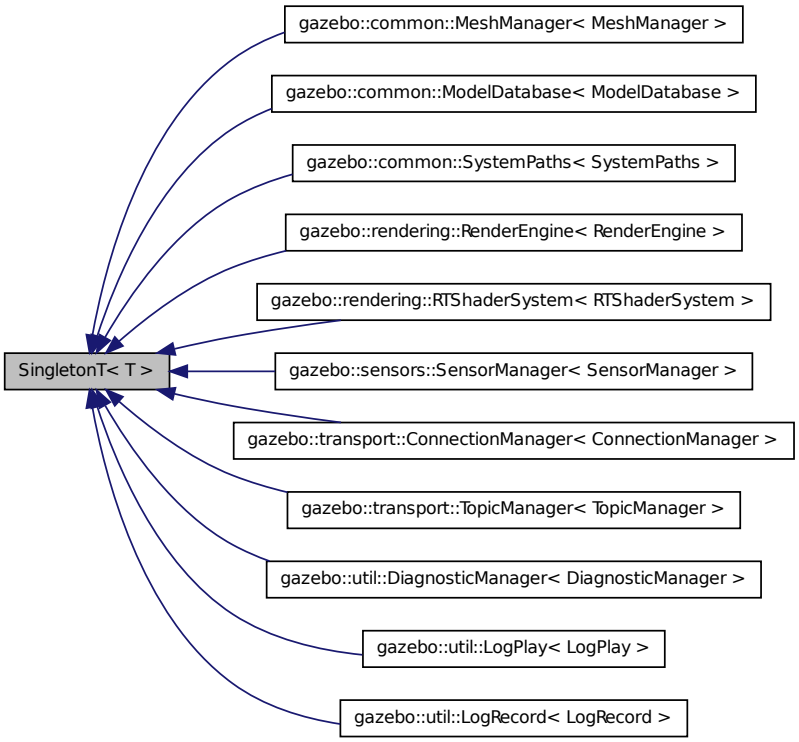
- **SensorManager.hh**

10.232 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.

10.232.1 Detailed Description

template<class T>class SingletonT< T >

Singleton template class.

10.232.2 Constructor & Destructor Documentation

10.232.2.1 template<class T> SingletonT< T >::SingletonT ( ) [inline, protected]

Constructor.

10.232.2.2 template<class T> virtual SingletonT< T >::~SingletonT ( ) [inline, protected, virtual]

Destructor.

10.232.3 Member Function Documentation

10.232.3.1 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**

10.233 gazebo::common::Skeleton Class Reference

A skeleton.

`#include <common/common.hh>`

Public Member Functions

- **Skeleton** ()

Constructor.

- **Skeleton** (**SkeletonNode** ∗_root)

  Constructor.

- virtual ∼**Skeleton** ()

  Destructor.

- void **AddAnimation** (**SkeletonAnimation** ∗_anim)

  Add an animation.

- void **AddVertNodeWeight** (unsigned int _vertex, std::string _node, double _weight)

  Add a new weight to a node (bone)

- **SkeletonAnimation** ∗ **GetAnimation** (const unsigned int _i)

  Find animation.

- **math::Matrix4 GetBindShapeTransform** ()

  Return bind pose skeletal transform.

- **SkeletonNode** ∗ **GetNodeByHandle** (unsigned int _handle)

  Find or create node with handle.

- **SkeletonNode** ∗ **GetNodeById** (std::string _id)

  Find node by index.

- **SkeletonNode** ∗ **GetNodeByName** (std::string _name)

  Find a node.

- **NodeMap GetNodes** ()

  Get a copy 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

10.233.1 Detailed Description

A skeleton.

10.233.2 Constructor & Destructor Documentation

10.233.2.1 gazebo::common::Skeleton::Skeleton ( )

Constructor.

10.233.2.2 gazebo::common::Skeleton::Skeleton (SkeletonNode * *_root*)

Constructor.

Parameters

in	*_root*	node

10.233.2.3 virtual gazebo::common::Skeleton::~Skeleton ( ) [virtual]

Destructor.

10.233.3 Member Function Documentation

10.233.3.1 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

10.233.3.2 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)

10.233.3.3 void gazebo::common::Skeleton::BuildNodeMap ( ) [protected]

Initializes the hande numbers for each node in the map using breadth first traversal.

10.233.3.4 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

10.233.3.5 math::Matrix4 gazebo::common::Skeleton::GetBindShapeTransform ( )

Return bind pose skeletal transform.

Returns

a matrix

10.233.3.6 SkeletonNode* gazebo::common::Skeleton::GetNodeByHandle (unsigned int *_handle*)

Find or create node with handle.

Parameters

in	*_handle*	

Returns

the node. A new node is created if it didn't exist

10.233.3.7 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

10.233.3.8 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

10.233.3.9 NodeMap gazebo::common::Skeleton::GetNodes ( )

Get a copy or the node dictionary.

10.233.3.10 unsigned int gazebo::common::Skeleton::GetNumAnimations ( )

Returns the number of animations.

Returns

the count

10.233.3.11 unsigned int gazebo::common::Skeleton::GetNumJoints ( )

Returns the number of joints.

Returns

the count

10.233.3.12 unsigned int gazebo::common::Skeleton::GetNumNodes ( )

Returns the node count.

Returns

the count

10.233.3.13 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

10.233.3.14 SkeletonNode* gazebo::common::Skeleton::GetRootNode ( )

Return the root.

Returns

the root

10.233.3.15 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

10.233.3.16 void gazebo::common::Skeleton::PrintTransforms ( )

Outputs the transforms to std::err stream.

10.233.3.17 void gazebo::common::Skeleton::Scale (double *_scale*)

Scale all nodes, transforms and animation data.

Parameters

in	*the*	scaling factor

10.233.3.18 void gazebo::common::Skeleton::SetBindShapeTransform (math::Matrix4 *_trans*)

Set the bind pose skeletal transform.

Parameters

in	*_trans*	the transform

10.233.3.19 void gazebo::common::Skeleton::SetNumVertAttached (unsigned int *_vertices*)

Resizes the raw node weight array.

Parameters

in	*_vertices*	the new size

10.233.3.20 void gazebo::common::Skeleton::SetRootNode (SkeletonNode * *_node*)

Change the root node.

Parameters

in	*_node*	the new node

10.233.4 Member Data Documentation

10.233.4.1 std::vector<SkeletonAnimation*> gazebo::common::Skeleton::anims [protected]

the array of animations

10.233.4.2 math::Matrix4 gazebo::common::Skeleton::bindShapeTransform [protected]

the bind pose skeletal transform

10.233.4.3 NodeMap gazebo::common::Skeleton::nodes [protected]

The dictionary of nodes, indexed by name.

10.233.4.4 RawNodeWeights gazebo::common::Skeleton::rawNW [protected]

the node weight table

10.233.4.5 SkeletonNode* gazebo::common::Skeleton::root [protected]

the root node

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

- **Skeleton.hh**

10.234 gazebo::common::SkeletonAnimation Class Reference

Skeleton (p. 1269) 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

10.234.1 Detailed Description

Skeleton (p. 1269) animation.

10.234.2 Constructor & Destructor Documentation

10.234.2.1 gazebo::common::SkeletonAnimation::SkeletonAnimation (const std::string & _name)

The Constructor.

Parameters

in	*_name*	the name of the animation

10.234.2.2 gazebo::common::SkeletonAnimation::~SkeletonAnimation ( )

The destructor.

Clears the list without destroying the animations

10.234.3 Member Function Documentation

10.234.3.1 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

10.234.3.2 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. 995)

10.234.3.3 double gazebo::common::SkeletonAnimation::GetLength ( ) const

Returns the duration of the animations.

Returns

the duration in seconds

10.234.3.4 std::string gazebo::common::SkeletonAnimation::GetName ( ) const

Returns the name.

Returns

the name

10.234.3.5 unsigned int gazebo::common::SkeletonAnimation::GetNodeCount ( ) const

Returns the number of animation nodes.

Returns

the count

10.234.3.6 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

10.234.3.7 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

10.234.3.8 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)

10.234.3.9 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

10.234.3.10 void gazebo::common::SkeletonAnimation::Scale (const double *_scale*)

Scales every animation in the animations list.

Parameters

in	*_scale*	the scaling factor

10.234.3.11 void gazebo::common::SkeletonAnimation::SetName (const std::string & *_name*)

Changes the name.

Parameters

in	*_name*	the new name

10.234.4 Member Data Documentation

10.234.4.1 std::map<std::string, NodeAnimation*> gazebo::common::SkeletonAnimation::animations [protected]

a dictionary of node animations

10.234.4.2 double gazebo::common::SkeletonAnimation::length [protected]

the duration of the longest animation

10.234.4.3 std::string gazebo::common::SkeletonAnimation::name [protected]

the node name

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

- **SkeletonAnimation.hh**

10.235 gazebo::common::SkeletonNode Class Reference

A skeleton node.

`#include <common/common.hh>`

Public Types

- enum **SkeletonNodeType** { **NODE**, **JOINT** }

  enumeration of node types

Public Member Functions

- **SkeletonNode** (**SkeletonNode** *_parent)

  Constructor.
- **SkeletonNode** (**SkeletonNode** *_parent, std::string _name, std::string _id, **SkeletonNodeType** _type=**JOINT**)

  Constructor.
- virtual ~**SkeletonNode** ()

  Destructor.
- void **AddChild** (**SkeletonNode** *_child)

  Add a new child.
- void **AddRawTransform** (**NodeTransform** _t)

  Add a raw transform.
- **SkeletonNode** * **GetChild** (unsigned int _index)

  Find a child by index.
- **SkeletonNode** * **GetChildById** (std::string _id)

  Get child by string id.
- **SkeletonNode** * **GetChildByName** (std::string _name)

  Get child by name.
- unsigned int **GetChildCount** ()

  Returns the children count.
- unsigned int **GetHandle** ()

  Get the handle index.
- std::string **GetId** ()

  Returns the index.
- **math::Matrix4 GetInverseBindTransform** ()

  Retrieve the inverse of the bind pose skeletal transform.
- **math::Matrix4 GetModelTransform** ()

  Retrieve the model transform.

- std::string **GetName** ()

  Returns the name.
- unsigned int **GetNumRawTrans** ()

  Return the raw transformations count.
- **SkeletonNode** ∗ **GetParent** ()

  Returns the parent node.
- **NodeTransform GetRawTransform** (unsigned int _i)

  Find a raw transformation.
- std::vector< **NodeTransform** > **GetRawTransforms** ()

  Retrieve the raw transformations.
- **math::Matrix4 GetTransform** ()

  Get transform relative to parent.
- std::vector< **NodeTransform** > **GetTransforms** ()

  Returns a copy of the array of transformations.
- bool **IsJoint** ()

  Is a joint query.
- bool **IsRootNode** ()

  Queries 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

10.235.1 Detailed Description

A skeleton node.

10.235.2 Member Enumeration Documentation

10.235.2.1 enum gazebo::common::SkeletonNode::SkeletonNodeType

enumeration of node types

Enumerator:

NODE

JOINT

10.235.3 Constructor & Destructor Documentation

10.235.3.1 gazebo::common::SkeletonNode::SkeletonNode (SkeletonNode * _parent)

Constructor.

Parameters

in	*_parent*	The parent node

10.235.3.2 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

10.235.3.3 virtual gazebo::common::SkeletonNode::~SkeletonNode ( ) [virtual]

Destructor.

10.235.4 Member Function Documentation

10.235.4.1 void gazebo::common::SkeletonNode::AddChild (SkeletonNode * _child)

Add a new child.

Parameters

in	*_child*	a child

10.235.4.2 void gazebo::common::SkeletonNode::AddRawTransform (NodeTransform _t)

Add a raw transform.

Parameters

in	*_t*	the transform

10.235.4.3 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

10.235.4.4 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

10.235.4.5 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

10.235.4.6 unsigned int gazebo::common::SkeletonNode::GetChildCount ( )

Returns the children count.

Returns

the count

10.235.4.7 unsigned int gazebo::common::SkeletonNode::GetHandle ( )

Get the handle index.

Returns

the handle index

10.235.4.8 std::string gazebo::common::SkeletonNode::GetId ( )

Returns the index.

Returns

the id string

10.235.4.9 math::Matrix4 gazebo::common::SkeletonNode::GetInverseBindTransform ( )

Retrieve the inverse of the bind pose skeletal transform.

Returns

the transform

10.235.4.10 math::Matrix4 gazebo::common::SkeletonNode::GetModelTransform ( )

Retrieve the model transform.

Returns

the transform

10.235.4.11 std::string gazebo::common::SkeletonNode::GetName ( )

Returns the name.

Returns

the name

10.235.4.12 unsigned int gazebo::common::SkeletonNode::GetNumRawTrans ( )

Return the raw transformations count.

Returns

the count

10.235.4.13 SkeletonNode* gazebo::common::SkeletonNode::GetParent ( )

Returns the parent node.

Returns

the parent

10.235.4.14 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

10.235.4.15 std::vector<NodeTransform> gazebo::common::SkeletonNode::GetRawTransforms ( )

Retrieve the raw transformations.

Returns

an array of transformations

10.235.4.16 math::Matrix4 gazebo::common::SkeletonNode::GetTransform ( )

Get transform relative to parent.

10.235.4.17 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)

10.235.4.18 bool gazebo::common::SkeletonNode::IsJoint ( )

Is a joint query.

Returns

true if the skeleton type is a joint, false otherwise

10.235.4.19 bool gazebo::common::SkeletonNode::IsRootNode ( )

Queries wether a node has no parent parent.

Returns

true if the node has no parent, fasle otherwise

10.235.4.20 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

10.235.4.21 void gazebo::common::SkeletonNode::SetHandle (unsigned int _h)

Assign a handle number.

Parameters

in	*_h*	the handle

10.235.4.22 void gazebo::common::SkeletonNode::SetId (std::string _id)

Change the id string.

Parameters

in	*_id*	the new id string

10.235.4.23 void gazebo::common::SkeletonNode::SetInitialTransform (math::Matrix4 _tras)

Sets the initial transformation.

Parameters

in	*_tras*	the transfromation matrix

10.235.4.24 void gazebo::common::SkeletonNode::SetInverseBindTransform (math::Matrix4 _invBM)

Assign the inverse of the bind pose skeletal transform.

Parameters

in	*_invBM*	the transform

10.235.4.25 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

10.235.4.26 void gazebo::common::SkeletonNode::SetName (std::string *_name*)

Change the name.

Parameters

in	*_name*	the new name

10.235.4.27 void gazebo::common::SkeletonNode::SetParent (SkeletonNode * *_parent*)

Set the parent node.

Parameters

in	*_parent*	the new parent

10.235.4.28 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

10.235.4.29 void gazebo::common::SkeletonNode::SetType (SkeletonNodeType _type)

Change the skeleton node type.

Parameters

in	*_type*	the new type

10.235.4.30 void gazebo::common::SkeletonNode::UpdateChildrenTransforms ( )

Apply model transformations in order for each node in the tree.

10.235.5 Member Data Documentation

10.235.5.1 std::vector<SkeletonNode*> gazebo::common::SkeletonNode::children [protected]

the children nodes

10.235.5.2 unsigned int gazebo::common::SkeletonNode::handle [protected]

handle index number

10.235.5.3 std::string gazebo::common::SkeletonNode::id [protected]

a string identifier

10.235.5.4 math::Matrix4 gazebo::common::SkeletonNode::initialTransform [protected]

the initial transformation

10.235.5.5 math::Matrix4 gazebo::common::SkeletonNode::invBindTransform [protected]

the inverse of the bind pose skeletal transform

10.235.5.6 math::Matrix4 gazebo::common::SkeletonNode::modelTransform [protected]

the model transformation

10.235.5.7 std::string gazebo::common::SkeletonNode::name [protected]

the name of the skeletal node

10.235.5.8 SkeletonNode* gazebo::common::SkeletonNode::parent [protected]

the parent node

10.235.5.9 std::vector<NodeTransform> gazebo::common::SkeletonNode::rawTransforms [protected]

the raw transformation

10.235.5.10 math::Matrix4 gazebo::common::SkeletonNode::transform [protected]

the transform

10.235.5.11 SkeletonNodeType gazebo::common::SkeletonNode::type [protected]

the type fo node

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

- **Skeleton.hh**

10.236 gazebo::physics::SliderJoint< T > Class Template - Reference

A slider joint.

`#include <physics/physics.hh>`

Public Member Functions

- **SliderJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**SliderJoint** ()

  Destructor.
- virtual unsigned int **GetAngleCount** () const
- virtual void **Load** (sdf::ElementPtr _sdf)

  *Load a **SliderJoint** (p. 1294).*

10.236.1 Detailed Description

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

A slider joint.

10.236.2 Constructor & Destructor Documentation

10.236.2.1 template<class T> gazebo::physics::SliderJoint< T >::SliderJoint (BasePtr _parent) [inline, explicit]

Constructor.

Parameters

in	*_parent*	Parent of the joint.

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

Destructor.

10.236.3 Member Function Documentation

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

10.236.3.2 template<class T> virtual void gazebo::physics::SliderJoint< T >::Load (sdf::ElementPtr _sdf) [inline, virtual]

Load a **SliderJoint** (p. 1294).

Parameters

in	*_sdf*	SDF values to load from

Reimplemented in **gazebo::physics::SimbodySliderJoint** (p. 1256), and **gazebo::physics::DARTSliderJoint** (p. 464).

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

- **SliderJoint.hh**

10.237 gazebo::rendering::GzTerrainMatGen::SM2Profile Class Reference

Shader model 2 profile target.

`#include <Heightmap.hh>`

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)

10.237.1 Detailed Description

Shader model 2 profile target.

10.237.2 Constructor & Destructor Documentation

10.237.2.1 gazebo::rendering::GzTerrainMatGen::SM2Profile::SM2Profile (Ogre::TerrainMaterialGenerator * *_parent*, const Ogre::String & *_name*, const Ogre::String & *_desc*)

Constructor.

10.237.2.2 virtual gazebo::rendering::GzTerrainMatGen::SM2Profile::~SM2Profile ( ) [virtual]

Destructor.

10.237.3 Member Function Documentation

10.237.3.1 virtual void gazebo::rendering::GzTerrainMatGen::SM2Profile::addTechnique (const Ogre::MaterialPtr & *_mat*, const Ogre::Terrain * *_terrain*, TechniqueType *_tt*) [protected, virtual]

10.237.3.2 Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generate (const Ogre::Terrain * *_terrain*)

10.237.3.3 Ogre::MaterialPtr gazebo::rendering::GzTerrainMatGen::SM2Profile::generateForCompositeMap (const Ogre::Terrain * *_terrain*)

10.237.3.4 void gazebo::rendering::GzTerrainMatGen::SM2Profile::UpdateParams (const Ogre::MaterialPtr & *_mat*, const Ogre::Terrain * *_terrain*)

10.237.3.5 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:

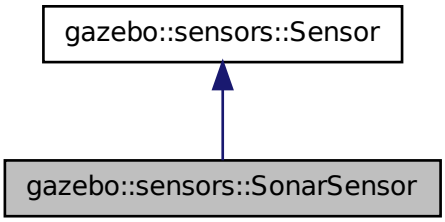
- **Heightmap.hh**

10.238 gazebo::sensors::SonarSensor Class Reference

Sensor (p. 1130) with sonar cone.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::SonarSensor:



Public Member Functions

- **SonarSensor** ()

  Constructor.
- virtual ~**SonarSensor** ()

  Destructor.
- template<typename T >
  event::ConnectionPtr ConnectUpdate (T _subscriber)

  Connect a to the new update signal.
- void **DisconnectUpdate** (**event::ConnectionPtr** &_conn)

  Disconnect from the update signal.
- double **GetRadius** () const

  Get the radius of the sonar cone at maximum range.
- double **GetRange** ()

  Get detected range for a sonar.
- double **GetRangeMax** () const

  Get the minimum range of the sonar.
- double **GetRangeMin** () const

  Get the minimum range of the sonar.
- 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.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the sensor.

- virtual bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

Protected Attributes

- **event::EventT**< void(msgs::SonarStamped)> **update**

  Update event.

10.238.1 Detailed Description

Sensor (p. 1130) with sonar cone.

This sensor uses a cone .

10.238.2 Constructor & Destructor Documentation

10.238.2.1 gazebo::sensors::SonarSensor::SonarSensor ( )

Constructor.

10.238.2.2 virtual gazebo::sensors::SonarSensor::~SonarSensor ( ) [virtual]

Destructor.

10.238.3 Member Function Documentation

10.238.3.1 template<typename T > event::ConnectionPtr gazebo::sensors::SonarSensor::ConnectUpdate (T _subscriber) [inline]

Connect a to the new update signal.

Parameters

in	*_subscriber*	Callback function.

Returns

The connection, which must be kept in scope.

10.238.3.2 void gazebo::sensors::SonarSensor::DisconnectUpdate (event::ConnectionPtr & _conn) [inline]

Disconnect from the update signal.

Parameters

in	*_conn*	Connection to remove.

10.238.3.3 virtual void gazebo::sensors::SonarSensor::Fini ( ) [protected, virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

10.238.3.4 double gazebo::sensors::SonarSensor::GetRadius ( ) const

Get the radius of the sonar cone at maximum range.

Returns

The radisu of the sonar cone at max range.

10.238.3.5 double gazebo::sensors::SonarSensor::GetRange ( )

Get detected range for a sonar.

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).

Returns

Returns DBL_MAX for no detection.

10.238.3.6 double gazebo::sensors::SonarSensor::GetRangeMax ( ) const

Get the minimum range of the sonar.

Returns

The sonar's maximum range.

10.238.3.7 double gazebo::sensors::SonarSensor::GetRangeMin ( ) const

Get the minimum range of the sonar.

Returns

The sonar's minimum range.

10.238.3.8 virtual std::string gazebo::sensors::SonarSensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 1138).

10.238.3.9 virtual void gazebo::sensors::SonarSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.238.3.10 virtual bool gazebo::sensors::SonarSensor::IsActive ( ) [virtual]

Returns true if sensor generation is active.

Returns

True if active, false if not.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

10.238.3.11 virtual void gazebo::sensors::SonarSensor::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. 1140).

10.238.3.12 virtual bool gazebo::sensors::SonarSensor::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

10.238.4 Member Data Documentation

10.238.4.1 event::EventT<void(msgs::SonarStamped)> gazebo::sensors::SonarSensor::update [protected]

Update event.

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

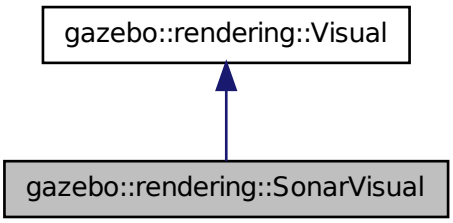
- **SonarSensor.hh**

10.239 gazebo::rendering::SonarVisual Class Reference

Visualization for sonar data.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::SonarVisual:



Public Member Functions

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

  Constructor.
- virtual ~**SonarVisual** ()

  Destructor.
- virtual void **Load** ()

  Load the visual with default parameters.

10.239.1 Detailed Description

Visualization for sonar data.

10.239.2 Constructor & Destructor Documentation

10.239.2.1 gazebo::rendering::SonarVisual::SonarVisual (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. 1477).
in	*_topicName*	Name of the topic that has sonar data.

10.239.2.2 virtual gazebo::rendering::SonarVisual::~SonarVisual ( ) [virtual]

Destructor.

10.239.3 Member Function Documentation

10.239.3.1 virtual void gazebo::rendering::SonarVisual::Load ( ) [virtual]

Load the visual with default parameters.

Reimplemented from **gazebo::rendering::Visual** (p. 1495).

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

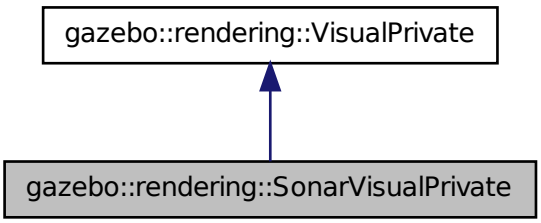
- **SonarVisual.hh**

10.240 gazebo::rendering::SonarVisualPrivate Class Reference

Private data for the Sonar **Visual** (p. 1477) class.

`#include <SonarVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::SonarVisualPrivate:



Public Attributes

- Ogre::SceneNode * **coneNode**

  Renders the sonar cone.
- std::vector< **event::ConnectionPtr** > **connections**

  All the event connections.
- boost::mutex **mutex**

  Mutex to protect the contact message.
- **transport::NodePtr node**

  Pointer to a node that handles communication.
- bool **receivedMsg**

  True if we have received a message.
- boost::shared_ptr < msgs::SonarStamped const > **sonarMsg**

  The current sonar message.
- **DynamicLines** * **sonarRay**

  Renders the sonar data reading.
- **transport::SubscriberPtr sonarSub**

  Subscription to the sonar data.

10.240.1 Detailed Description

Private data for the Sonar **Visual** (p. 1477) class.

10.240.2 Member Data Documentation

10.240.2.1 Ogre::SceneNode* gazebo::rendering::SonarVisualPrivate::coneNode

Renders the sonar cone.

10.240.2.2 std::vector<event::ConnectionPtr> gazebo::rendering::SonarVisualPrivate::connections

All the event connections.

10.240.2.3 boost::mutex gazebo::rendering::SonarVisualPrivate::mutex

Mutex to protect the contact message.

10.240.2.4 transport::NodePtr gazebo::rendering::SonarVisualPrivate::node

Pointer to a node that handles communication.

10.240.2.5 bool gazebo::rendering::SonarVisualPrivate::receivedMsg

True if we have received a message.

10.240.2.6 boost::shared_ptr<msgs::SonarStamped const> gazebo::rendering::SonarVisualPrivate::sonarMsg

The current sonar message.

10.240.2.7 DynamicLines* gazebo::rendering::SonarVisualPrivate::sonarRay

Renders the sonar data reading.

10.240.2.8 transport::SubscriberPtr gazebo::rendering::SonarVisualPrivate::sonarSub

Subscription to the sonar data.

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

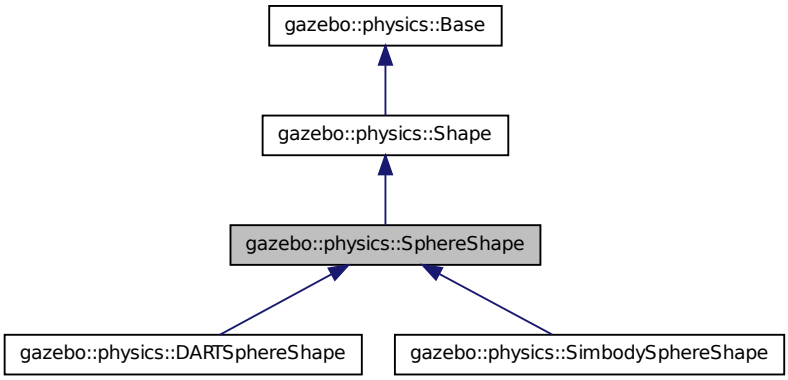
- **SonarVisualPrivate.hh**

10.241 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.
- virtual void **SetScale** (const **math::Vector3** &_scale)
  Set the scale of the sphere.

10.241.1 Detailed Description

Sphere collision shape.

10.241.2 Constructor & Destructor Documentation

10.241.2.1 gazebo::physics::SphereShape::SphereShape (CollisionPtr _parent) [explicit]

Constructor.

Parameters

in	*_parent*	Parent collision object.

10.241.2.2 virtual gazebo::physics::SphereShape::~SphereShape ( ) [virtual]

Destructor.

10.241.3 Member Function Documentation

10.241.3.1 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. 1163).

10.241.3.2 double gazebo::physics::SphereShape::GetRadius ( ) const

Get the sphere's radius.

Returns

Radius of the sphere.

10.241.3.3 virtual void gazebo::physics::SphereShape::Init ( ) [virtual]

Initialize the sphere.

Implements **gazebo::physics::Shape** (p. 1164).

10.241.3.4 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. 1164).

10.241.3.5 virtual void gazebo::physics::SphereShape::SetRadius (double *_radius*) [virtual]

Set the size.

Parameters

in	*_radius*	Radius of the sphere.

Reimplemented in **gazebo::physics::SimbodySphereShape** (p. 1259), and **gazebo::physics::DARTSphereShape** (p. 467).

Referenced by gazebo::physics::DARTSphereShape::SetRadius(), and gazebo::physics::SimbodySphereShape::SetRadius().

10.241.3.6 virtual void gazebo::physics::SphereShape::SetScale (const math::Vector3 & *_scale*) [virtual]

Set the scale of the sphere.

Parameters

in	*_scale*	Scale to set the sphere to.

Implements **gazebo::physics::Shape** (p. 1165).

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

- **SphereShape.hh**

10.242 gazebo::common::SphericalCoordinates Class Reference

Convert spherical coordinates for planetary surfaces.

`#include <commmon/common.hh>`

Public Types

- enum **SurfaceType** { **EARTH_WGS84** = 1 }

  Unique identifiers for planetary surface models.

Public Member Functions

- **SphericalCoordinates** ()

  Constructor.
- **SphericalCoordinates** (const **SurfaceType** _type)

  Constructor with surface type input.
- **SphericalCoordinates** (const **SurfaceType** _type, const **math::Angle** &_latitude, const **math::Angle** &_longitude, double _elevation, const **math::Angle** &_heading)

  Constructor with surface type, angle, and elevation inputs.
- ∼**SphericalCoordinates** ()

  Destructor.
- double **GetElevationReference** () const

  Get reference elevation in meters.
- **math::Angle GetHeadingOffset** () const

  Get heading offset for gazebo reference frame, expressed as angle from East to gazebo x-axis, or equivalently from North to gazebo y-axis.
- **math::Angle GetLatitudeReference** () const

  Get reference geodetic latitude.
- **math::Angle GetLongitudeReference** () const

  Get reference longitude.
- **SurfaceType GetSurfaceType** () const

  Get SurfaceType currently in use.
- **math::Vector3 GlobalFromLocal** (const **math::Vector3** &_xyz) const

  Convert a Cartesian velocity vector in the local gazebo frame to a global Cartesian frame with components East, North, Up.
- void **SetElevationReference** (double _elevation)

  Set reference elevation above sea level in meters.
- void **SetHeadingOffset** (const **math::Angle** &_angle)

Set heading angle offset for gazebo frame.

- void **SetLatitudeReference** (const **math::Angle** &_angle)

  Set reference geodetic latitude.
- void **SetLongitudeReference** (const **math::Angle** &_angle)

  Set reference longitude.
- void **SetSurfaceType** (const **SurfaceType** &_type)

  Set SurfaceType for planetary surface model.
- **math::Vector3 SphericalFromLocal** (const **math::Vector3** &_xyz) const

  Convert a Cartesian position vector to geodetic coordinates.

Static Public Member Functions

- static **SurfaceType Convert** (const std::string &_str)

  Convert a string to a SurfaceType.
- static double **Distance** (const **math::Angle** &_latA, const **math::Angle** &_lonA, const **math::Angle** &_latB, const **math::Angle** &_lonB)

  Get the distance between two points expressed in geographic latitude and longitude.

10.242.1 Detailed Description

Convert spherical coordinates for planetary surfaces.

10.242.2 Member Enumeration Documentation

10.242.2.1 enum gazebo::common::SphericalCoordinates::SurfaceType

Unique identifiers for planetary surface models.

Enumerator:

EARTH_WGS84 Model of reference ellipsoid for earth, based on WGS 84 standard. see wikipedia: World_Geodetic_System

10.242.3 Constructor & Destructor Documentation

10.242.3.1 gazebo::common::SphericalCoordinates::SphericalCoordinates ( )

Constructor.

10.242.3.2 gazebo::common::SphericalCoordinates::SphericalCoordinates (const SurfaceType _type)

Constructor with surface type input.

Parameters

in	*_type*	SurfaceType specification.

10.242.3.3 gazebo::common::SphericalCoordinates::SphericalCoordinates (const SurfaceType _type, const math::Angle & _latitude, const math::Angle & _longitude, double _elevation, const math::Angle & _heading)

Constructor with surface type, angle, and elevation inputs.

Parameters

in	*_type*	SurfaceType specification.
in	*_latitude*	Reference latitude.
in	*_longitude*	Reference longitude.
in	*_elevation*	Reference elevation.
in	*_heading*	Heading offset.

10.242.3.4 gazebo::common::SphericalCoordinates::~SphericalCoordinates ( )

Destructor.

10.242.4 Member Function Documentation

10.242.4.1 static SurfaceType gazebo::common::SphericalCoordinates::Convert (const std::string & _str) [static]

Convert a string to a SurfaceType.

Parameters

in	*_str*	String to convert.

Returns

Conversion to SurfaceType.

10.242.4.2 static double gazebo::common::SphericalCoordinates::Distance (const math::Angle & _latA, const math::Angle & _lonA, const math::Angle & _latB, const math::Angle & _lonB) [static]

Get the distance between two points expressed in geographic latitude and longitude.

It assumes that both points are at sea level. Example: _latA = 38.0016667 and _lonA = -123.0016667) represents the point with latitude 38d 0'6.00"N and longitude 123d 0'6.00"W.

Parameters

in	*_latA*	Latitude of point A.
in	*_longA*	Longitude of point A.
in	*_latB*	Latitude of point B.
in	*_longB*	Longitude of point B.

Returns

Distance in meters.

10.242.4.3 double gazebo::common::SphericalCoordinates::GetElevationReference ( ) const

Get reference elevation in meters.

Returns

Reference elevation.

10.242.4.4 math::Angle gazebo::common::SphericalCoordinates::GetHeadingOffset ( ) const

Get heading offset for gazebo reference frame, expressed as angle from East to gazebo x-axis, or equivalently from North to gazebo y-axis.

Returns

Heading offset of gazebo reference frame.

10.242.4.5 math::Angle gazebo::common::SphericalCoordinates::GetLatitudeReference ( ) const

Get reference geodetic latitude.

Returns

Reference geodetic latitude.

10.242.4.6 math::Angle gazebo::common::SphericalCoordinates::GetLongitudeReference ( ) const

Get reference longitude.

Returns

Reference longitude.

10.242.4.7 SurfaceType gazebo::common::SphericalCoordinates::GetSurfaceType ( ) const

Get SurfaceType currently in use.

Returns

Current SurfaceType value.

10.242.4.8 math::Vector3 gazebo::common::SphericalCoordinates::GlobalFromLocal (const math::Vector3 & *_xyz*) const

Convert a Cartesian velocity vector in the local gazebo frame to a global Cartesian frame with components East, North, Up.

Parameters

in	*_xyz*	Cartesian vector in gazebo's world frame.

Returns

Rotated vector with components (x,y,z): (East, North, Up).

10.242.4.9 void gazebo::common::SphericalCoordinates::SetElevationReference (double *_elevation*)

Set reference elevation above sea level in meters.

Parameters

in	*_elevation*	Reference elevation.

10.242.4.10 void gazebo::common::SphericalCoordinates::SetHeadingOffset (const math::Angle & *_angle*)

Set heading angle offset for gazebo frame.

Parameters

in	*_angle*	Heading offset for gazebo frame.

10.242.4.11 void gazebo::common::SphericalCoordinates::SetLatitudeReference (const math::Angle & *_angle*)

Set reference geodetic latitude.

Parameters

in	*_angle*	Reference geodetic latitude.

10.242.4.12 void gazebo::common::SphericalCoordinates::SetLongitudeReference (const math::Angle & *_angle*)

Set reference longitude.

Parameters

in	*_angle*	Reference longitude.

10.242.4.13 void gazebo::common::SphericalCoordinates::SetSurfaceType (const SurfaceType & *_type*)

Set SurfaceType for planetary surface model.

Parameters

in	*_type*	SurfaceType value.

10.242.4.14 math::Vector3 gazebo::common::SphericalCoordinates-::SphericalFromLocal (const math::Vector3 & _xyz) const

Convert a Cartesian position vector to geodetic coordinates.

Parameters

in	*_xyz*	Cartesian position vector in gazebo's world frame.

Returns

Cooordinates: geodetic latitude (deg), longitude (deg), altitude above sea level (m).

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

- **SphericalCoordinates.hh**

10.243 gazebo::common::SphericalCoordinatesPrivate Class - Reference

commmon/common.hh

`#include <SphericalCoordinatesPrivate.hh>`

Public Attributes

- double **elevationReference**

  Elevation of reference point relative to sea level in meters.

- **math::Angle headingOffset**

  Heading offset, expressed as angle from East to gazebo x-axis, or equivalently from North to gazebo y-axis.

- **math::Angle latitudeReference**

  Latitude of reference point.

- **math::Angle longitudeReference**

  Longitude of reference point.

- **SphericalCoordinates::SurfaceType surfaceType**

  Type of surface being used.

10.243.1 Detailed Description

commmon/common.hh

Private data for the **SphericalCoordinates** (p. 1310) class.

10.243.2 Member Data Documentation

10.243.2.1 double gazebo::common::SphericalCoordinatesPrivate::elevationReference

Elevation of reference point relative to sea level in meters.

10.243.2.2 math::Angle gazebo::common::SphericalCoordinatesPrivate::headingOffset

Heading offset, expressed as angle from East to gazebo x-axis, or equivalently from North to gazebo y-axis.

10.243.2.3 math::Angle gazebo::common::SphericalCoordinatesPrivate::latitudeReference

Latitude of reference point.

10.243.2.4 math::Angle gazebo::common::SphericalCoordinatesPrivate::longitudeReference

Longitude of reference point.

10.243.2.5 SphericalCoordinates::SurfaceType gazebo::common::SphericalCoordinatesPrivate::surfaceType

Type of surface being used.

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

- **SphericalCoordinatesPrivate.hh**

10.244 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.

10.244.1 Detailed Description

Splines.

10.244.2 Constructor & Destructor Documentation

10.244.2.1 gazebo::math::Spline::Spline ( )

constructor

10.244.2.2 gazebo::math::Spline::~Spline ( )

destructor

10.244.3 Member Function Documentation

10.244.3.1 void gazebo::math::Spline::AddPoint (const Vector3 & *_pt*)

Adds a control point to the end of the spline.

Parameters

in	*_pt*	point to add

10.244.3.2 void gazebo::math::Spline::Clear ( )

Clears all the points in the spline.

10.244.3.3 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

10.244.3.4 unsigned int gazebo::math::Spline::GetPointCount ( ) const

Gets the number of control points in the spline.

Returns

the count

10.244.3.5 Vector3 gazebo::math::Spline::GetTangent (unsigned int *_index*) const

Get the tangent value for a point.

Parameters

in	*_index*	the control point index

10.244.3.6 double gazebo::math::Spline::GetTension ( ) const

Get the tension value.

Returns

The value of the tension, which is between 0.0 and 1.0

10.244.3.7 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)

10.244.3.8 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

10.244.3.9 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.

10.244.3.10 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.

10.244.3.11 void gazebo::math::Spline::SetTension (double _t)

Set the tension parameter.

A value of 0 = Catmull-Rom spline.

Parameters

in	*_t*	Tension value between 0.0 and 1.0

10.244.3.12 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

10.244.4 Member Data Documentation

10.244.4.1 bool gazebo::math::Spline::autoCalc [protected]

when true, the tangents are recalculated when the control point change

10.244.4.2 Matrix4 gazebo::math::Spline::coeffs [protected]

Matrix of coefficients.

10.244.4.3 std::vector<Vector3> gazebo::math::Spline::points [protected]

control points

10.244.4.4 std::vector<Vector3> gazebo::math::Spline::tangents [protected]

tangents

10.244.4.5 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:

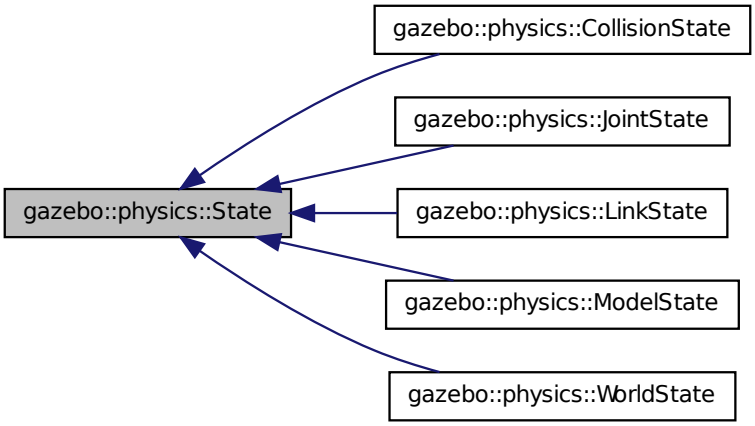
- **Spline.hh**

10.245 gazebo::physics::State Class Reference

State (p. 1323) 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. 1323).*
- **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. 1323).*
- virtual void **SetRealTime** (const **common::Time** &_time)

  Set the real time when this state was generated.
- virtual void **SetSimTime** (const **common::Time** &_time)

  Set the sim time when this state was generated.
- virtual void **SetWallTime** (const **common::Time** &_time)

  Set the wall time when this state was generated.

Protected Attributes

- std::string **name**

  *Name associated with this **State** (p. 1323).*
- **common::Time realTime**
- **common::Time simTime**
- **common::Time wallTime**

  Times for the state data.

10.245.1 Detailed Description

State (p. 1323) of an entity.

This is the base class for all **State** (p. 1323) information.

10.245.2 Constructor & Destructor Documentation

10.245.2.1 gazebo::physics::State::State ( )

Default constructor.

10.245.2.2 gazebo::physics::State::State (const std::string & *_name*, const common::Time & *_realTime*, const common::Time & *_simTime*)

Constructor.

Construct a **State** (p. 1323) object using some basic information.

Parameters

_name	Name associated with the **State** (p. 1323) information. This is typically the name of an **Entity** (p. 500). _realTime Clock time since simulation started.
_simTime	Simulation time associated with this **State** (p. 1323) info.

10.245.2.3 virtual gazebo::physics::State::~State ( ) [virtual]

Destructor.

10.245.3 Member Function Documentation

10.245.3.1 std::string gazebo::physics::State::GetName ( ) const

Get the name associated with this **State** (p. 1323).

Returns

Name associated with this state information. Typically a name of an **Entity** (p. 500).

10.245.3.2 common::Time gazebo::physics::State::GetRealTime ( ) const

Get the real time when this state was generated.

Returns

Clock time since simulation was stated.

10.245.3.3 common::Time gazebo::physics::State::GetSimTime ( ) const

Get the sim time when this state was generated.

Returns

Simulation time when the data was recorded.

10.245.3.4 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. 1323) data was recorded.

10.245.3.5 virtual void gazebo::physics::State::Load (const sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Populates the **State** (p. 1323) information from data stored in an SDF::Element

Parameters

_elem	Pointer to the SDF::Element

Reimplemented in **gazebo::physics::ModelState** (p. 877), **gazebo::physics::LinkState** (p. 776), **gazebo::physics::WorldState** (p. 1550), **gazebo::physics::JointState** (p. 717), and **gazebo::physics::CollisionState** (p. 310).

10.245.3.6 State gazebo::physics::State::operator- (const State & *_state*) const

Subtraction operator.

Parameters

in	*_pt*	A state to substract.

Returns

The resulting state.

10.245.3.7 State& gazebo::physics::State::operator= (const State & _state)

Assignment operator.

Parameters

in	*_state*	**State** (p. 1323) value

Returns

this

10.245.3.8 void gazebo::physics::State::SetName (const std::string & _name)

Set the name associated with this **State** (p. 1323).

Parameters

in	*_name*	Name associated with this state information. Typically the name of an **Entity** (p. 500).

10.245.3.9 virtual void gazebo::physics::State::SetRealTime (const common::Time & _time) [virtual]

Set the real time when this state was generated.

Parameters

in	*_time*	Clock time since simulation was stated.

Reimplemented in **gazebo::physics::ModelState** (p. 879), **gazebo::physics::LinkState** (p. 777), and **gazebo::physics::WorldState** (p. 1552).

10.245.3.10 virtual void gazebo::physics::State::SetSimTime (const common::Time & _time) [virtual]

Set the sim time when this state was generated.

Parameters

in	*_time*	Simulation time when the data was recorded.

Reimplemented in **gazebo::physics::ModelState** (p. 879), **gazebo::physics::LinkState** (p. 777), and **gazebo::physics::WorldState** (p. 1552).

10.245.3.11 virtual void gazebo::physics::State::SetWallTime (const common::Time & *_time*) [virtual]

Set the wall time when this state was generated.

Parameters

in	*_time*	The absolute clock time when the **State** (p. 1323) data was recorded.

Reimplemented in **gazebo::physics::ModelState** (p. 879), **gazebo::physics::LinkState** (p. 777), and **gazebo::physics::WorldState** (p. 1552).

10.245.4 Member Data Documentation

10.245.4.1 std::string gazebo::physics::State::name [protected]

Name associated with this **State** (p. 1323).

10.245.4.2 common::Time gazebo::physics::State::realTime [protected]

10.245.4.3 common::Time gazebo::physics::State::simTime [protected]

10.245.4.4 common::Time gazebo::physics::State::wallTime [protected]

Times for the state data.

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

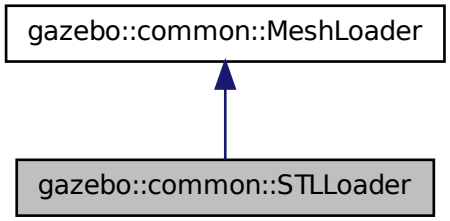
- **State.hh**

10.246 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.

10.246.1 Detailed Description

Class used to load STL mesh files.

10.246.2 Constructor & Destructor Documentation

10.246.2.1 gazebo::common::STLLoader::STLLoader ( )

Constructor.

10.246.2.2 virtual gazebo::common::STLLoader::∼STLLoader ( ) [virtual]

Destructor.

10.246.3 Member Function Documentation

10.246.3.1 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. 834).

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

- **STLLoader.hh**

10.247 gazebo::common::SubMesh Class Reference

A child mesh.

`#include <Mesh.hh>`

Public Types

- enum **PrimitiveType** { **POINTS**, **LINES**, **LINESTRIPS**, **TRIANGLES**, **TRIFANS**, **TRISTRIPS** }

  An enumeration of the geometric mesh primitives.

Public Member Functions

- **SubMesh** ()

  Constructor.
- **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.

10.247.1 Detailed Description

A child mesh.

10.247.2 Member Enumeration Documentation

10.247.2.1 enum gazebo::common::SubMesh::PrimitiveType

An enumeration of the geometric mesh primitives.

Enumerator:

POINTS

LINES

LINESTRIPS

TRIANGLES

TRIFANS

TRISTRIPS

10.247.3 Constructor & Destructor Documentation

10.247.3.1 gazebo::common::SubMesh::SubMesh ( )

Constructor.

10.247.3.2 gazebo::common::SubMesh::SubMesh (const SubMesh * *_mesh*)

Copy Constructor.

10.247.3.3 virtual gazebo::common::SubMesh::~SubMesh ( ) [virtual]

Destructor.

10.247.4 Member Function Documentation

10.247.4.1 void gazebo::common::SubMesh::AddIndex (unsigned int _i)

Add an index to the mesh.

Parameters

in	*_i*	the new vertex index

10.247.4.2 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)

10.247.4.3 void gazebo::common::SubMesh::AddNormal (const math::Vector3 & _n)

Add a normal to the mesh.

Parameters

in	*_n*	the normal

10.247.4.4 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

10.247.4.5 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

10.247.4.6 void gazebo::common::SubMesh::AddVertex (const math::Vector3 & _v)

Add a vertex to the mesh.

Parameters

in	_v	the new position

10.247.4.7 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

10.247.4.8 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.

10.247.4.9 void gazebo::common::SubMesh::CopyNormals (const std::vector< math::Vector3 > & *_norms*)

Copy normals from a vector.

Parameters

in	*_norms*	to copy from

10.247.4.10 void gazebo::common::SubMesh::CopyVertices (const std::vector< math::Vector3 > & *_verts*)

Copy vertices from a vector.

Parameters

in	*_verts*	the vertices to copy from

10.247.4.11 void gazebo::common::SubMesh::FillArrays (float ** *_vertArr,* int ** *_indArr*) const

Put all the data into flat arrays.

Parameters

in	*_verArr*	
in	*_indArr*	

10.247.4.12 void gazebo::common::SubMesh::GenSphericalTexCoord (const math::Vector3 & *_center*)

Generate texture coordinates using spherical projection from center.

Parameters

in	*_center*	

10.247.4.13 unsigned int gazebo::common::SubMesh::GetIndex (unsigned int *_i*) const

Get an index.

Parameters

in	*_i*	

10.247.4.14 unsigned int gazebo::common::SubMesh::GetIndexCount ( ) const

Return the number of indicies.

10.247.4.15 unsigned int gazebo::common::SubMesh::GetMaterialIndex ( ) const

Get the material index.

10.247.4.16 math::Vector3 gazebo::common::SubMesh::GetMax ( ) const

Get the maximun X, Y, Z values.

Returns

10.247.4.17 unsigned int gazebo::common::SubMesh::GetMaxIndex ( ) const

Get the highest index value.

10.247.4.18 math::Vector3 gazebo::common::SubMesh::GetMin ( ) const

Get the minimum X, Y, Z values.

Returns

10.247.4.19 std::string gazebo::common::SubMesh::GetName ( ) const

Get the name of this mesh.

Returns

the name

10.247.4.20 NodeAssignment gazebo::common::SubMesh::GetNodeAssignment (unsigned int _i) const

Get a vertex - skeleton node assignment.

Parameters

in	_i	the index of the assignment

10.247.4.21 unsigned int gazebo::common::SubMesh::GetNodeAssignmentsCount ( ) const

Return the number of vertex - skeleton node assignments.

10.247.4.22 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

10.247.4.23 unsigned int gazebo::common::SubMesh::GetNormalCount ( ) const

Return the number of normals.

10.247.4.24 PrimitiveType gazebo::common::SubMesh::GetPrimitiveType ( ) const

Get the primitive type.

Returns

the primitive type

10.247.4.25 math::Vector2d gazebo::common::SubMesh::GetTexCoord (unsigned int _i) const

Get a tex coord.

Parameters

in	_i	the texture index

Returns

the texture coordinates

10.247.4.26 unsigned int gazebo::common::SubMesh::GetTexCoordCount ( ) const

Return the number of texture coordinates.

10.247.4.27 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

10.247.4.28 unsigned int gazebo::common::SubMesh::GetVertexCount ( ) const

Return the number of vertices.

10.247.4.29 unsigned int gazebo::common::SubMesh::GetVertexIndex (const math::Vector3 & _v) const

Get the index of the vertex.

Parameters

in	_v	

10.247.4.30 bool gazebo::common::SubMesh::HasVertex (const math::Vector3 & _v) const

Return true if this submesh has the vertex.

Parameters

in	*_v*	

10.247.4.31 void gazebo::common::SubMesh::RecalculateNormals ( )

Recalculate all the normals.

10.247.4.32 void gazebo::common::SubMesh::Scale (double _factor)

Scale all vertices by _factor.

Parameters

in	*_factor*	Scaling factor

10.247.4.33 void gazebo::common::SubMesh::SetIndexCount (unsigned int _count)

Resize the index array.

Parameters

in	*_count*	the new size of the array

10.247.4.34 void gazebo::common::SubMesh::SetMaterialIndex (unsigned int _index)

Set the material index.

Relates to the parent mesh material list

Parameters

in	*_index*	

10.247.4.35 void gazebo::common::SubMesh::SetName (const std::string & _n)

Set the name of this mesh.

Parameters

in	*_n*	the name to set

10.247.4.36 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

10.247.4.37 void gazebo::common::SubMesh::SetNormalCount (unsigned int _count)

Resize the normal array.

Parameters

in	*_count*	the new size of the array

10.247.4.38 void gazebo::common::SubMesh::SetPrimitiveType (PrimitiveType _type)

Set the primitive type.

Parameters

in	*_type*	the type

10.247.4.39 void gazebo::common::SubMesh::SetScale (const math::Vector3 & _factor)

Scale all vertices by the _factor vector.

Parameters

in	*_factor*	Scaling vector

10.247.4.40 void gazebo::common::SubMesh::SetSubMeshCenter (math::Vector3 *_center*)

Reset mesh center to geometric center.

Parameters

in	*_center*	

10.247.4.41 void gazebo::common::SubMesh::SetTexCoord (unsigned int *_i*, const math::Vector2d & *_t*)

Set a tex coord.

Parameters

in	*_i*	
in	*_t*	

10.247.4.42 void gazebo::common::SubMesh::SetTexCoordCount (unsigned int *_count*)

Resize the texture coordinate array.

Parameters

in	*_count*	

10.247.4.43 void gazebo::common::SubMesh::SetVertex (unsigned int *_i*, const math::Vector3 & *_v*)

Set a vertex.

Parameters

in	*_i*	the index
in	*_v*	the position

10.247.4.44 void gazebo::common::SubMesh::SetVertexCount (unsigned int *_count*)

Resize the vertex array.

Parameters

in	*_count*	the new size of the array

10.247.4.45 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**

10.248 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.

10.248.1 Detailed Description

Options for a subscription.

10.248.2 Constructor & Destructor Documentation

10.248.2.1 gazebo::transport::SubscribeOptions::SubscribeOptions ( ) [inline]

Constructor.

10.248.3 Member Function Documentation

10.248.3.1 bool gazebo::transport::SubscribeOptions::GetLatching ( ) const [inline]

Are we latching?

Returns

true if we're latching the latest message, false otherwise

10.248.3.2 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

10.248.3.3 NodePtr gazebo::transport::SubscribeOptions::GetNode ( ) const [inline]

Get the node we're subscribed to.

Returns

The associated node

10.248.3.4 std::string gazebo::transport::SubscribeOptions::GetTopic ( ) const [inline]

Get the topic we're subscribed to.

Returns

The topic we're subscribed to

10.248.3.5 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().

10.248.3.6 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**

10.249 gazebo::transport::Subscriber Class Reference

A subscriber to a topic.

`#include <transport/transport.hh>`

Public Member Functions

- **Subscriber** (const std::string &_topic, **NodePtr** _node)

  Constructor.
- virtual ~**Subscriber** ()

  Destructor.
- unsigned int **GetCallbackId** () const
- std::string **GetTopic** () const

  Get the topic name.
- void **SetCallbackId** (unsigned int _id)
- void **Unsubscribe** () const

  Unsubscribe from the topic.

10.249.1 Detailed Description

A subscriber to a topic.

10.249.2 Constructor & Destructor Documentation

10.249.2.1 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

10.249.2.2 virtual gazebo::transport::Subscriber::~Subscriber ( ) [virtual]

Destructor.

10.249.3 Member Function Documentation

10.249.3.1 unsigned int gazebo::transport::Subscriber::GetCallbackId ( ) const

10.249.3.2 std::string gazebo::transport::Subscriber::GetTopic ( ) const

Get the topic name.

Returns

The topic name

10.249.3.3 void gazebo::transport::Subscriber::SetCallbackId (unsigned int *_id*)

10.249.3.4 void gazebo::transport::Subscriber::Unsubscribe ( ) const

Unsubscribe from the topic.

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

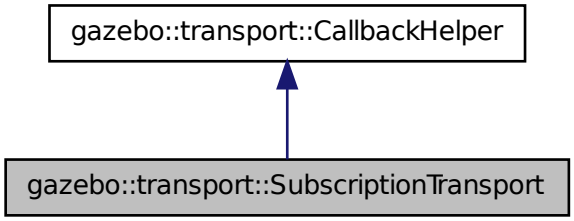
- **Subscriber.hh**

10.250 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, boost::function< void(uint32_t)> _cb, uint32_t _id)

  Output a message to a connection.
- virtual bool **HandleMessage** (**MessagePtr** _newMsg)

  Process new incoming message.
- void **Init** (**ConnectionPtr** _conn, bool _latching)

  Initialize the publication link.
- virtual bool **IsLocal** () const

  Is the callback local?

10.250.1 Detailed Description

transport/transport.hh

Handles sending data over the wire to remote subscribers

10.250.2 Constructor & Destructor Documentation

10.250.2.1 gazebo::transport::SubscriptionTransport::SubscriptionTransport ( )

Constructor.

10.250.2.2 virtual gazebo::transport::SubscriptionTransport::∼SubscriptionTransport ( ) [virtual]

Destructor.

10.250.3 Member Function Documentation

10.250.3.1 const ConnectionPtr& gazebo::transport::SubscriptionTransport::GetConnection ( ) const

Get the connection we're using.

Returns

Pointer to the connection we're using

10.250.3.2 virtual bool gazebo::transport::SubscriptionTransport::HandleData (const std::string & *_newdata,* boost::function< void(uint32_t)> *_cb,* uint32_t *_id*) [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

Parameters

in	*_cb*	If non-null, callback to be invoked after transmission is complete.
in	*_id*	ID associated with the message data.

Implements **gazebo::transport::CallbackHelper** (p. 238).

10.250.3.3 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. 238).

10.250.3.4 void gazebo::transport::SubscriptionTransport::Init (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

10.250.3.5 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. 239).

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

- **SubscriptionTransport.hh**

10.251 gazebo::physics::SurfaceParams Class Reference

SurfaceParams (p. 1350) defines various Surface contact parameters.

`#include <physics/physics.hh>`

Public Member Functions

- **SurfaceParams** ()

  Constructor.
- virtual ~**SurfaceParams** ()

  Destructor.
- virtual 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)

  Process a surface message.

Public Attributes

- bool **collideWithoutContact**

  Allow collision checking without generating a contact joint.
- unsigned int **collideWithoutContactBitmask**

  Custom collision filtering used when collideWithoutContact is true.

10.251.1 Detailed Description

SurfaceParams (p. 1350) defines various Surface contact parameters.

These parameters defines the properties of a **physics::Contact** (p. 347) constraint.

10.251.2 Constructor & Destructor Documentation

10.251.2.1 gazebo::physics::SurfaceParams::SurfaceParams ( )

Constructor.

10.251.2.2 virtual gazebo::physics::SurfaceParams::~SurfaceParams ( ) [virtual]

Destructor.

10.251.3 Member Function Documentation

10.251.3.1 virtual void gazebo::physics::SurfaceParams::FillMsg (msgs::Surface & *_msg*) [virtual]

Fill in a surface message.

Parameters

in	*_msg*	Message to fill with this object's values.

10.251.3.2 virtual void gazebo::physics::SurfaceParams::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the contact params.

Parameters

in	*_sdf*	SDF values to load from.

10.251.3.3 virtual void gazebo::physics::SurfaceParams::ProcessMsg (const msgs::Surface & *_msg*) [virtual]

Process a surface message.

Parameters

in	*_msg*	Message to read values from.

10.251.4 Member Data Documentation

10.251.4.1 bool gazebo::physics::SurfaceParams::collideWithoutContact

Allow collision checking without generating a contact joint.

10.251.4.2 unsigned int gazebo::physics::SurfaceParams::collideWithoutContact-Bitmask

Custom collision filtering used when collideWithoutContact is true.

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

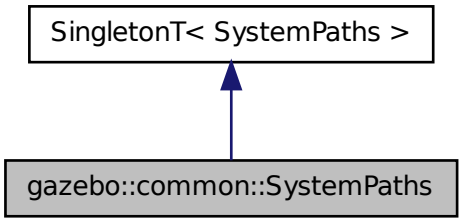
- **SurfaceParams.hh**

10.252 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.

- std::string **GetDefaultTestPath** ()

  Returns the default temporary test path.
- 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 **GetTmpInstancePath** ()

  Returns a unique temporary file for this instance of SystemPath.
- std::string **GetTmpPath** ()

  Returns the default path suitable for temporary files.
- std::string **GetWorldPathExtension** ()

  Returns the world path extension.

Public Attributes

- bool **gazeboPathsFromEnv**

  *if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1357)*
- bool **modelPathsFromEnv**

  *if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1357)*
- bool **ogrePathsFromEnv**

  *if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 1358)*
- bool **pluginPathsFromEnv**

  *if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 1358)*

10.252.1 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. 163) - RenderSystem_GL.so et. al.
- SystemPaths::pluginPaths - plugin library paths for common::WorldPlugin

10.252.2 Member Function Documentation

10.252.2.1 void gazebo::common::SystemPaths::AddGazeboPaths (const std::string & _path)

Add colon delimited paths to Gazebo install.

Parameters

in	*_path*	the directory to add

10.252.2.2 void gazebo::common::SystemPaths::AddModelPaths (const std::string & _path)

Add colon delimited paths to modelPaths.

Parameters

in	*_path*	the directory to add

10.252.2.3 void gazebo::common::SystemPaths::AddOgrePaths (const std::string & _path)

Add colon delimited paths to ogre install.

Parameters

in	*_path*	the directory to add

10.252.2.4 void gazebo::common::SystemPaths::AddPluginPaths (const std::string & _path)

Add colon delimited paths to plugins.

Parameters

in	*_path*	the directory to add

10.252.2.5 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

10.252.2.6 void gazebo::common::SystemPaths::ClearGazeboPaths ( )

clear out SystemPaths::gazeboPaths

10.252.2.7 void gazebo::common::SystemPaths::ClearModelPaths ( )

clear out SystemPaths::modelPaths

10.252.2.8 void gazebo::common::SystemPaths::ClearOgrePaths ( )

clear out SystemPaths::ogrePaths

10.252.2.9 void gazebo::common::SystemPaths::ClearPluginPaths ( )

clear out SystemPaths::pluginPaths

10.252.2.10 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	*_search-LocalPath*	True to search in the current working directory.

Returns

Returns full path name to file

10.252.2.11 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

10.252.2.12 std::string gazebo::common::SystemPaths::GetDefaultTestPath ( )

Returns the default temporary test path.

Returns

a full path name to directory. E.g.: /tmp/gazebo_test (Linux).

10.252.2.13 const std::list<std::string>& gazebo::common::SystemPaths::GetGazeboPaths ( )

Get the gazebo install paths.

Returns

a list of paths

10.252.2.14 std::string gazebo::common::SystemPaths::GetLogPath ( ) const

Get the log path.

Returns

the path

10.252.2.15 const std::list<std::string>& gazebo::common::SystemPaths::GetModelPaths ( )

Get the model paths.

Returns

a list of paths

10.252.2.16 const std::list<std::string>& gazebo::common::SystemPaths::GetOgrePaths ( )

Get the ogre install paths.

Returns

a list of paths

10.252.2.17 const std::list<std::string>& gazebo::common::SystemPaths::GetPluginPaths ( )

Get the plugin paths.

Returns

a list of paths

10.252.2.18 std::string gazebo::common::SystemPaths::GetTmpInstancePath ( )

Returns a unique temporary file for this instance of SystemPath.

Returns

a full path name to directory. E.g.: /tmp/gazebo_234123 (Linux).

10.252.2.19 std::string gazebo::common::SystemPaths::GetTmpPath ( )

Returns the default path suitable for temporary files.

Returns

a full path name to directory. E.g.: /tmp (Linux).

10.252.2.20 std::string gazebo::common::SystemPaths::GetWorldPathExtension ( )

Returns the world path extension.

Returns

Right now, it just returns "/worlds"

10.252.3 Member Data Documentation

10.252.3.1 bool gazebo::common::SystemPaths::gazeboPathsFromEnv

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1357)

10.252.3.2 bool gazebo::common::SystemPaths::modelPathsFromEnv

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 1357)

10.252.3.3 bool gazebo::common::SystemPaths::ogrePathsFromEnv

if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 1358)

10.252.3.4 bool gazebo::common::SystemPaths::pluginPathsFromEnv

if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 1358)

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

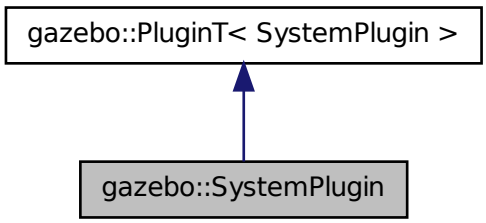
- **SystemPaths.hh**

10.253 gazebo::SystemPlugin Class Reference

A plugin loaded within the gzserver on startup.

```
#include <Plugin.hh>
```

Inheritance diagram for gazebo::SystemPlugin:



Public Member Functions

- **SystemPlugin** ()

  Constructor.
- virtual ∼**SystemPlugin** ()

  Destructor.
- virtual void **Init** ()

  Initialize the plugin.
- virtual void **Load** (int _argc=0, char ∗∗_argv=**NULL**)=0

  Load function.
- virtual void **Reset** ()

  Override this method for custom plugin reset behavior.

10.253.1 Detailed Description

A plugin loaded within the gzserver on startup.

See `reference`.

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

10.253.2 Constructor & Destructor Documentation

10.253.2.1 gazebo::SystemPlugin::SystemPlugin ( ) [inline]

Constructor.

References gazebo::SYSTEM_PLUGIN.

10.253.2.2 virtual gazebo::SystemPlugin::~SystemPlugin ( ) [inline, virtual]

Destructor.

10.253.3 Member Function Documentation

10.253.3.1 virtual void gazebo::SystemPlugin::Init ( ) [inline, virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.253.3.2 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.

10.253.3.3 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:

- **Plugin.hh**

10.254 gazebo::common::Time Class Reference

A **Time** (p. 1361) 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 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**

  Nanoseconds.

- int32_t **sec**

  Seconds.

Static Public Attributes

- static const **Time Zero**

  A 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.

10.254.1 Detailed Description

A **Time** (p. 1361) class, can be used to hold wall- or sim-time.

stored as sec and nano-sec.

10.254.2 Constructor & Destructor Documentation

10.254.2.1 gazebo::common::Time::Time ( )

Constructors.

10.254.2.2 gazebo::common::Time::Time (const Time & *_time*)

Copy constructor.

Parameters

in	*time*	**Time** (p. 1361) to copy

10.254.2.3 gazebo::common::Time::Time (const struct timeval & *_tv*)

Constructor.

Parameters

in	*_tv*	**Time** (p. 1361) to initialize to

10.254.2.4 gazebo::common::Time::Time (const struct timespec & _tv)

Constructor.

Parameters

in	*_tv*	**Time** (p. 1361) to initialize to

10.254.2.5 gazebo::common::Time::Time (int32_t _sec, int32_t _nsec)

Constructor.

Parameters

in	*_sec*	Seconds
in	*_nsec*	Nanoseconds

10.254.2.6 gazebo::common::Time::Time (double _time)

Constuctor.

Parameters

in	*_time*	**Time** (p. 1361) in double format sec.nsec

10.254.2.7 virtual gazebo::common::Time::~Time ( ) [virtual]

Destructor.

10.254.3 Member Function Documentation

10.254.3.1 double gazebo::common::Time::Double ( ) const

Get the time as a double.

Returns

Time (p. 1361) as a double in seconds

10.254.3.2 float gazebo::common::Time::Float ( ) const

Get the time as a float.

Returns

Time (p. 1361) as a float in seconds

10.254.3.3 static const Time& gazebo::common::Time::GetWallTime ( ) [static]

Get the wall time.

Returns

the current time

10.254.3.4 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.

10.254.3.5 static double gazebo::common::Time::MicToNano (double *_ms*) [inline, static]

Convert microseconds to nanoseconds.

Parameters

_ms	microseconds

Returns

nanoseconds

10.254.3.6 static double gazebo::common::Time::MilToNano (double *_ms*) [inline, static]

Convert milliseconds to nanoseconds.

Parameters

in	*_ms*	milliseconds

Returns

nanoseconds

10.254.3.7 static Time gazebo::common::Time::MSleep (unsigned int _ms) [static]

Millisecond sleep.

Parameters

in	*_ms*	milliseconds

Returns

Time (p. 1361) actually slept

10.254.3.8 static Time gazebo::common::Time::NSleep (unsigned int _ns) [static]

Nano sleep.

Parameters

in	*_ns*	nanoseconds

Returns

Time (p. 1361) actually slept

10.254.3.9 bool gazebo::common::Time::operator!= (const struct timeval & _tv) const

Equal to operator.

Parameters

in	*_tv*	the time to compare to

Returns

true if values are the same, false otherwise

10.254.3.10 bool gazebo::common::Time::operator!= (const struct timespec & *_tv*) const

Equal to operator.

Parameters

in	*_tv*	the time to compare to

Returns

true if values are the same, false otherwise

10.254.3.11 bool gazebo::common::Time::operator!= (const Time & *_time*) const

Equal to operator.

Parameters

in	*_time*	the time to compare to

Returns

true if values are the same, false otherwise

10.254.3.12 bool gazebo::common::Time::operator!= (double *_time*) const

Equal to operator.

Parameters

in	*_time*	the time to compare to

Returns

true if values are the same, false otherwise

10.254.3.13 Time gazebo::common::Time::operator∗ (const struct timeval & _tv) const

Multiplication operator.

Parameters

in	*_tv*	The scaling duration

Returns

Time (p. 1361) instance

10.254.3.14 Time gazebo::common::Time::operator∗ (const struct timespec & _tv) const

Multiplication operator.

Parameters

in	*_tv*	the scaling duration

Returns

Time (p. 1361) instance

10.254.3.15 Time gazebo::common::Time::operator∗ (const Time & _time) const

Multiplication operators.

Parameters

in	*_time*	the scaling factor

Returns

a scaled **Time** (p. 1361) instance

10.254.3.16 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

10.254.3.17 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

10.254.3.18 const Time& gazebo::common::Time::operator*= (const Time & _time)

Multiplication operators.

Parameters

in	*_time*	scale factor

Returns

a scaled **Time** (p. 1361) instance

10.254.3.19 Time gazebo::common::Time::operator+ (const struct timeval & _tv) const

Addition operators.

Parameters

in	*_tv*	the time to add

Returns

a **Time** (p. 1361) instance

10.254.3.20 Time gazebo::common::Time::operator+ (const struct timespec & _tv) const

Addition operators.

Parameters

in	*_tv*	the time to add

Returns

a **Time** (p. 1361) instance

10.254.3.21 Time gazebo::common::Time::operator+ (const Time & _time) const

Addition operators.

Parameters

in	*_time*	The time to add

Returns

a **Time** (p. 1361) instance

10.254.3.22 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

10.254.3.23 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

10.254.3.24 const Time& gazebo::common::Time::operator+= (const Time & _time)

Addition assignemtn operator.

Parameters

in	*_time*	The time to add

Returns

a **Time** (p. 1361) instance

10.254.3.25 Time gazebo::common::Time::operator- (const struct timeval & _tv) const

Subtraction operator.

Parameters

in	*_tv*	The time to subtract

Returns

a **Time** (p. 1361) instance

10.254.3.26 Time gazebo::common::Time::operator- (const struct timespec & _tv) const

Subtraction operator.

Parameters

in	*_tv*	The time to subtract

Returns

a **Time** (p. 1361) instance

10.254.3.27 Time gazebo::common::Time::operator- (const Time & *_time*) const

Subtraction operator.

Parameters

in	*_time*	The time to subtract

Returns

a **Time** (p. 1361) instance

10.254.3.28 const Time& gazebo::common::Time::operator-= (const struct timeval & *_tv*)

Subtraction assignment operator.

Parameters

in	*_tv*	The time to subtract

Returns

a **Time** (p. 1361) instance

10.254.3.29 const Time& gazebo::common::Time::operator-= (const struct timespec & *_tv*)

Subtraction assignment operator.

Parameters

in	*_tv*	The time to subtract

Returns

a **Time** (p. 1361) instance

10.254.3.30 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

10.254.3.31 Time gazebo::common::Time::operator/ (const struct timeval & _tv) const

Division operator.

Parameters

in	*_tv*	a timeval divisor

Returns

a **Time** (p. 1361) instance

10.254.3.32 Time gazebo::common::Time::operator/ (const struct timespec & _tv) const

Division operator.

Parameters

in	*_tv*	a timespec divisor

Returns

a **Time** (p. 1361) instance

10.254.3.33 Time gazebo::common::Time::operator/ (const Time & _time) const

Division operator.

Parameters

in	*_time*	the divisor

Returns

a **Time** (p. 1361) instance

10.254.3.34 const Time& gazebo::common::Time::operator/= (const struct timeval & _tv)

Division assignment operator.

Parameters

in	*_tv*	a divisor

Returns

a **Time** (p. 1361) instance

10.254.3.35 const Time& gazebo::common::Time::operator/= (const struct timespec & _tv)

Division assignment operator.

Parameters

in	*_tv*	a divisor

Returns

a **Time** (p. 1361) instance

10.254.3.36 const Time& gazebo::common::Time::operator/= (const Time & *time*)

Division assignment operator.

Parameters

in	*time*	the divisor

Returns

a **Time** (p. 1361) instance

10.254.3.37 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

10.254.3.38 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

10.254.3.39 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

10.254.3.40 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

10.254.3.41 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

10.254.3.42 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

10.254.3.43 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

10.254.3.44 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

10.254.3.45 Time& gazebo::common::Time::operator= (const struct timeval & _tv)

Assignment operator.

Parameters

in	*_tv*	the new time

Returns

a reference to this instance

10.254.3.46 Time& gazebo::common::Time::operator= (const struct timespec & _tv)

Assignment operator.

Parameters

in	*_tv*	the new time

Returns

a reference to this instance

10.254.3.47 Time& gazebo::common::Time::operator= (const Time & *_time*)

Assignment operator.

Parameters

in	*_time*	the new time

Returns

a reference to this instance

10.254.3.48 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

10.254.3.49 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

10.254.3.50 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

10.254.3.51 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

10.254.3.52 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

10.254.3.53 bool gazebo::common::Time::operator> (const struct timespec & _tv) const

Greater than operator.

Parameters

in	*_tv*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.254.3.54 bool gazebo::common::Time::operator> (const Time & _time) const

Greater than operator.

Parameters

in	*_time*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.254.3.55 bool gazebo::common::Time::operator> (double _time) const

Greater than operator.

Parameters

in	*_time*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.254.3.56 bool gazebo::common::Time::operator>= (const struct timeval & _tv) const

Greater than or equal operator.

Parameters

in	*_tv*	the time to compare with

Returns

true if tv is greater than or equal to this, false otherwise

10.254.3.57 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

10.254.3.58 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

10.254.3.59 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

10.254.3.60 static double gazebo::common::Time::SecToNano (double _sec) [inline, static]

Convert seconds to nanoseconds.

Parameters

in	*_sec*	duration in seconds

Returns

nanoseconds

10.254.3.61 void gazebo::common::Time::Set (int32_t _sec, int32_t _nsec)

Set to sec and nsec.

Parameters

in	*_sec*	Seconds
in	*_nsec*	Nanoseconds

10.254.3.62 void gazebo::common::Time::Set (double _seconds)

Set to seconds.

Parameters

in	*_seconds*	Number of seconds

10.254.3.63 void gazebo::common::Time::SetToWallTime ( )

Set the time to the wall time.

10.254.3.64 static Time gazebo::common::Time::Sleep (const common::Time & _time) [static]

Sleep for the specified time.

Parameters

in	*_time*	Sleep time

Returns

Time (p. 1361) actually slept

10.254.4 Friends And Related Function Documentation

10.254.4.1 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

10.254.4.2 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

10.254.5 Member Data Documentation

10.254.5.1 int32_t gazebo::common::Time::nsec

Nanoseconds.

10.254.5.2 int32_t gazebo::common::Time::sec

Seconds.

10.254.5.3 const Time gazebo::common::Time::Zero [static]

A static zero time variable set to common::Time(0, 0).

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

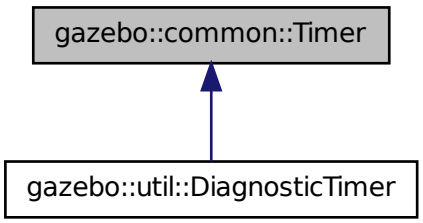
- **Time.hh**

10.255 gazebo::common::Timer Class Reference

A timer class, used to time things in real world walltime.

`#include <common/common.hh>`

Inheritance diagram for gazebo::common::Timer:



Public Member Functions

- **Timer** ()
  Constructor.
- virtual ∼**Timer** ()
  Destructor.
- **Time GetElapsed** () const
  Get the elapsed time.
- 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.

10.255.1 Detailed Description

A timer class, used to time things in real world walltime.

10.255.2 Constructor & Destructor Documentation

10.255.2.1 gazebo::common::Timer::Timer ( )

Constructor.

10.255.2.2 virtual gazebo::common::Timer::∼Timer ( ) [virtual]

Destructor.

10.255.3 Member Function Documentation

10.255.3.1 Time gazebo::common::Timer::GetElapsed ( ) const

Get the elapsed time.

Returns

The time

10.255.3.2 bool gazebo::common::Timer::GetRunning ( ) const

Returns true if the timer is running.

Returns

Tue if the timer has been started and not stopped.

10.255.3.3 virtual void gazebo::common::Timer::Start ( ) [virtual]

Start the timer.

Reimplemented in **gazebo::util::DiagnosticTimer** (p. 489).

10.255.3.4 virtual void gazebo::common::Timer::Stop ( ) [virtual]

Stop the timer.

Reimplemented in **gazebo::util::DiagnosticTimer** (p. 489).

10.255.4 Friends And Related Function Documentation

10.255.4.1 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:

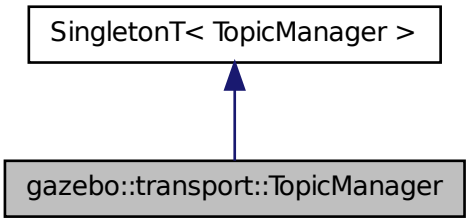
- **Timer.hh**

10.256 gazebo::transport::TopicManager Class Reference

Manages topics and their subscriptions.

`#include <transport/transport.hh>`

Inheritance diagram for gazebo::transport::TopicManager:



Public Types

- typedef std::map< std::string, std::list< **NodePtr** > > **SubNodeMap**

  A map of string->list of ***Node*** *(*p. *910) pointers.*

Public Member Functions

- void **AddNode** (**NodePtr** _node)

  Add a node to the manager.
- void **AddNodeToProcess** (**NodePtr** _ptr)

  Add a node to the list of nodes that requires processing.
- 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. *328) a local* ***Publisher*** *(*p. *1024) to a remote* ***Subscriber*** *(*p. *1346).*
- 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. *1346) to a remote* ***Publisher*** *(*p. *1024).*

- void **DisconnectPubFromSub** (const std::string &_topic, const std::string &_host, unsigned int _port)

  Disconnect a local publisher from a remote subscriber.
- void **DisconnectSubFromPub** (const std::string &_topic, const std::string &_host, unsigned int _port)

  Disconnect all local subscribers from a remote publisher.
- **PublicationPtr FindPublication** (const std::string &_topic)

  Find a publication object by topic.
- void **Fini** ()

  Finalize the manager.
- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

  Get all the topic namespaces.
- void **Init** ()

  Initialize the manager.
- bool **IsAdvertised** (const std::string &_topic)

  Has the topic been advertised?
- void **PauseIncoming** (bool _pause)

  Pause or unpause processing of incoming messages.
- void **ProcessNodes** (bool _onlyOut=false)

  Process all nodes under management.
- void **Publish** (const std::string &_topic, **MessagePtr** _message, boost::function< void(uint32_t)> _cb, uint32_t _id)

  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 **Unadvertise** (**PublisherPtr** _pub)

  Unadvertise a publisher.
- 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.

10.256.1 Detailed Description

Manages topics and their subscriptions.

10.256.2 Member Typedef Documentation

10.256.2.1 typedef std::map<std::string, std::list<NodePtr> > gazebo::transport::TopicManager::SubNodeMap

A map of string->list of **Node** (p. 910) pointers.

10.256.3 Member Function Documentation

10.256.3.1 void gazebo::transport::TopicManager::AddNode (NodePtr *_node*)

Add a node to the manager.

Parameters

in,out	*_node*	The node to be added

10.256.3.2 void gazebo::transport::TopicManager::AddNodeToProcess (NodePtr *_ptr*)

Add a node to the list of nodes that requires processing.

Parameters

in	*_ptr*	**Node** (p. 910) to process.

10.256.3.3 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. 1024)

References GZ_ASSERT, gzthrow, SingletonT< T >::Instance(), and NULL.

10.256.3.4 void gazebo::transport::TopicManager::ClearBuffers ( )

Clear all buffers.

10.256.3.5 void gazebo::transport::TopicManager::ConnectPubToSub (const std::string & _topic, const SubscriptionTransportPtr _sublink)

Connection (p. 328) a local **Publisher** (p. 1024) to a remote **Subscriber** (p. 1346).

Parameters

in	*_topic*	The topic to use
in	*_sublink*	The subscription transport object to use

10.256.3.6 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

10.256.3.7 void gazebo::transport::TopicManager::ConnectSubToPub (const msgs::Publish & _pub)

Connect a local **Subscriber** (p. 1346) to a remote **Publisher** (p. 1024).

Parameters

in	*_pub*	The publish object to use

10.256.3.8 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

10.256.3.9 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

10.256.3.10 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)

10.256.3.11 void gazebo::transport::TopicManager::Fini ( )

Finalize the manager.

10.256.3.12 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

10.256.3.13 void gazebo::transport::TopicManager::Init ( )

Initialize the manager.

10.256.3.14 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

10.256.3.15 void gazebo::transport::TopicManager::PauseIncoming (bool *_pause*)

Pause or unpause processing of incoming messages.

Parameters

in	*_pause*	If true pause processing; otherwse unpause

10.256.3.16 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

10.256.3.17 void gazebo::transport::TopicManager::Publish (const std::string & *_topic*, MessagePtr *_message*, boost::function< void(uint32_t)> *_cb*, uint32_t *_id*)

Send a message.

Use a **Publisher** (p. 1024) instead of calling this function directly.

Parameters

in	*_topic*	Name of the topic
in	*_message*	The message to send.
in	*_cb*	Callback, used when the publish is completed.
in	*_id*	ID associated with the message.

10.256.3.18 void gazebo::transport::TopicManager::RegisterTopicNamespace (const std::string & *_name*)

Register a new topic namespace.

Parameters

in	*_name*	The name of the new namespace

10.256.3.19 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

10.256.3.20 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

10.256.3.21 void gazebo::transport::TopicManager::Unadvertise (const std::string & *_topic*)

Unadvertise a topic.

Parameters

in	*_topic*	The topic to be unadvertised

10.256.3.22 void gazebo::transport::TopicManager::Unadvertise (PublisherPtr *_pub*)

Unadvertise a publisher.

Parameters

in	*_pub*	**Publisher** (p. 1024) to unadvertise.

10.256.3.23 void gazebo::transport::TopicManager::Unsubscribe (const std::string & *_topic,* const NodePtr & *_sub*)

Unsubscribe from a topic.

Use a **Subscriber** (p. 1346) rather than calling this function directly

Parameters

in	*_topic*	The topic to unsubscribe from
in	*_sub*	The node to unsubscribe

10.256.3.24 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

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

- **TopicManager.hh**

10.257 gazebo::physics::TrajectoryInfo Class Reference

Information about a trajectory for an **Actor** (p. 165).

`#include <Actor.hh>`

Public Member Functions

- **TrajectoryInfo** ()

  Constructor.

Public Attributes

- double **duration**

  Duration of the trajectory.
- double **endTime**

  End time of the trajectory.
- unsigned int **id**

  ID of the trajectory.
- double **startTime**

  Start time of the trajectory.
- bool **translated**

  True if the trajectory is tranlated.
- std::string **type**

  Type of trajectory.

10.257.1 Detailed Description

Information about a trajectory for an **Actor** (p. 165).

10.257.2 Constructor & Destructor Documentation

10.257.2.1 gazebo::physics::TrajectoryInfo::TrajectoryInfo ( )

Constructor.

10.257.3 Member Data Documentation

10.257.3.1 double gazebo::physics::TrajectoryInfo::duration

Duration of the trajectory.

10.257.3.2 double gazebo::physics::TrajectoryInfo::endTime

End time of the trajectory.

10.257.3.3 unsigned int gazebo::physics::TrajectoryInfo::id

ID of the trajectory.

10.257.3.4 double gazebo::physics::TrajectoryInfo::startTime

Start time of the trajectory.

10.257.3.5 bool gazebo::physics::TrajectoryInfo::translated

True if the trajectory is tranlated.

10.257.3.6 std::string gazebo::physics::TrajectoryInfo::type

Type of trajectory.

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

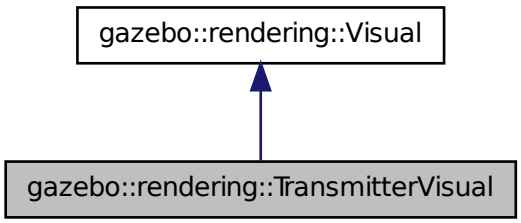
- **Actor.hh**

10.258 gazebo::rendering::TransmitterVisual Class Reference

Visualization for the wireless propagation data.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::TransmitterVisual:



Public Member Functions

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

  Constructor.
- virtual ~**TransmitterVisual** ()

  Destructor.
- virtual void **Load** ()

  Documentation inherited from parent.
- virtual void **Update** ()

  Function that runs on the OGRE thread to refresh the UI.

10.258.1 Detailed Description

Visualization for the wireless propagation data.

10.258.2 Constructor & Destructor Documentation

10.258.2.1 gazebo::rendering::TransmitterVisual::TransmitterVisual (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. 1477).
in	*_topicName*	Name of the topic that has laser data.

10.258.2.2 virtual gazebo::rendering::TransmitterVisual::~TransmitterVisual ( ) [virtual]

Destructor.

10.258.3 Member Function Documentation

10.258.3.1 virtual void gazebo::rendering::TransmitterVisual::Load ( ) [virtual]

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 1495).

10.258.3.2 virtual void gazebo::rendering::TransmitterVisual::Update ( ) [virtual]

Function that runs on the OGRE thread to refresh the UI.

Reimplemented from **gazebo::rendering::Visual** (p. 1503).

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

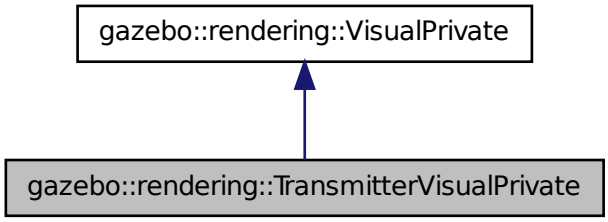
- **TransmitterVisual.hh**

10.259 gazebo::rendering::TransmitterVisualPrivate Class Reference

Private data for the Transmitter **Visual** (p. 1477) class.

`#include <TransmitterVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::TransmitterVisualPrivate:



Public Attributes

- std::vector< **event::ConnectionPtr** > **connections**

  All the event connections.
- boost::shared_ptr < msgs::PropagationGrid const > **gridMsg**

  The current contact message.
- bool **isFirst**

  Use for allocate the visuals for the grid only the first time you receive the grid.
- boost::mutex **mutex**

  Mutex to protect the contact message.
- **transport::NodePtr node**

  Pointer to a node that handles communication.
- **DynamicLines** * **points**

  Renders the points representing the signal strength.
- bool **receivedMsg**

  True if we have received a message.
- **transport::SubscriberPtr signalPropagationSub**

  Subscription to the propagation data.
- std::vector< **rendering::VisualPtr** > **vectorLink**

  Store the list of visuals.

10.259.1 Detailed Description

Private data for the Transmitter **Visual** (p. 1477) class.

10.259.2 Member Data Documentation

10.259.2.1 std::vector<event::ConnectionPtr> gazebo::rendering::TransmitterVisualPrivate::connections

All the event connections.

10.259.2.2 boost::shared_ptr<msgs::PropagationGrid const> gazebo::rendering::TransmitterVisualPrivate::gridMsg

The current contact message.

10.259.2.3 bool gazebo::rendering::TransmitterVisualPrivate::isFirst

Use for allocate the visuals for the grid only the first time you receive the grid.

The next times there are just updates.

10.259.2.4 boost::mutex gazebo::rendering::TransmitterVisualPrivate::mutex

Mutex to protect the contact message.

10.259.2.5 transport::NodePtr gazebo::rendering::TransmitterVisualPrivate::node

Pointer to a node that handles communication.

10.259.2.6 DynamicLines* gazebo::rendering::TransmitterVisualPrivate::points

Renders the points representing the signal strength.

10.259.2.7 bool gazebo::rendering::TransmitterVisualPrivate::receivedMsg

True if we have received a message.

10.259.2.8 transport::SubscriberPtr gazebo::rendering::TransmitterVisualPrivate::signalPropagationSub

Subscription to the propagation data.

10.259.2.9 std::vector<rendering::VisualPtr> gazebo::rendering::TransmitterVisualPrivate::vectorLink

Store the list of visuals.

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

- **TransmitterVisualPrivate.hh**

10.260 gazebo::physics::UniversalJoint< T > Class Template - Reference

A universal joint.

`#include <physics/physics.hh>`

Public Types

- enum **AxisIndex** { **AXIS_PARENT** = 0, **AXIS_CHILD** = 1 }

  Map joint axes to corresponding link.

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. 1404).*

Protected Member Functions

- virtual void **Init** ()

  Initialize joint.

10.260.1 Detailed Description

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

A universal joint.

Axis1 and axis2 are body-fixed, with axis1 attached to parent body and axis2 attached to child body.

10.260.2 Member Enumeration Documentation

10.260.2.1 template<class T> enum gazebo::physics::UniversalJoint::AxisIndex

Map joint axes to corresponding link.

Enumerator:

AXIS_PARENT

AXIS_CHILD

10.260.3 Constructor & Destructor Documentation

10.260.3.1 template<class T> gazebo::physics::UniversalJoint< T >::UniversalJoint (BasePtr *_parent*) [inline, explicit]

Constructor.

Parameters

in	*_parent*	Parent link of the univeral joint.

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

Destuctor.

10.260.4 Member Function Documentation

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

10.260.4.2 template<class T> virtual void gazebo::physics::UniversalJoint< T >::Init ( ) [inline, protected, virtual]

Initialize joint.

Reimplemented in **gazebo::physics::DARTUniversalJoint** (p. 472).

10.260.4.3 template<class T> virtual void gazebo::physics::UniversalJoint< T >::Load (sdf::ElementPtr *_sdf*) [inline, virtual]

Load a **UniversalJoint** (p. 1404).

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented in **gazebo::physics::SimbodyUniversalJoint** (p. 1264), and **gazebo::physics::DARTUniversalJoint** (p. 473).

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

- **UniversalJoint.hh**

10.261 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.

10.261.1 Detailed Description

Information for use in an update event.

10.261.2 Member Data Documentation

10.261.2.1 common::Time gazebo::common::UpdateInfo::realTime

Current real time.

10.261.2.2 common::Time gazebo::common::UpdateInfo::simTime

Current simulation time.

10.261.2.3 std::string gazebo::common::UpdateInfo::worldName

Name of the world.

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

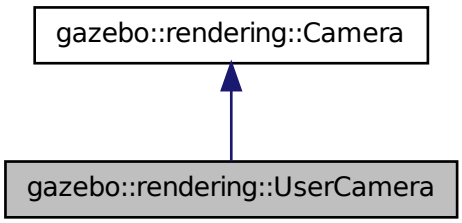
- **UpdateInfo.hh**

10.262 gazebo::rendering::UserCamera Class Reference

A camera used for user visualization of a scene.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::UserCamera:



Public Member Functions

- **UserCamera** (const std::string &_name, **ScenePtr** _scene)

  Constructor.
- virtual ~**UserCamera** ()

  Destructor.
- void **EnableViewController** (bool _value) const

  Set whether the view controller is enabled.
- void **Fini** ()

  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.
- unsigned int **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.
- bool **IsCameraSetInWorldFile** ()

  brief Show if the user camera pose has changed in the world file.
- 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 **SetUseSDFPose** (bool _value)

  brief Set if the user camera pose has changed in the world file.
- 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.

10.262.1 Detailed Description

A camera used for user visualization of a scene.

10.262.2 Constructor & Destructor Documentation

10.262.2.1 gazebo::rendering::UserCamera::UserCamera (const std::string & *_name*, ScenePtr *_scene*)

Constructor.

Parameters

in	*_name*	Name of the camera.
in	*_scene*	**Scene** (p. 1097) to put the camera in.

10.262.2.2 virtual gazebo::rendering::UserCamera::~UserCamera ( ) [virtual]

Destructor.

10.262.3 Member Function Documentation

10.262.3.1 virtual void gazebo::rendering::UserCamera::AnimationComplete ( ) [protected, virtual]

Internal function used to indicate that an animation has completed.

Reimplemented from **gazebo::rendering::Camera** (p. 251).

10.262.3.2 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. 252).

10.262.3.3 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.

10.262.3.4 void gazebo::rendering::UserCamera::Fini ( ) [virtual]

Finialize.

Reimplemented from **gazebo::rendering::Camera** (p. 254).

10.262.3.5 float gazebo::rendering::UserCamera::GetAvgFPS ( ) const [virtual]

Get the average frames per second.

Returns

The average rendering frames per second

Reimplemented from **gazebo::rendering::Camera** (p. 254).

10.262.3.6 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. 612).

10.262.3.7 virtual unsigned int gazebo::rendering::UserCamera::GetImageHeight ( ) const [virtual]

Get the height of the image.

Returns

Image height

Reimplemented from **gazebo::rendering::Camera** (p. 257).

10.262.3.8 virtual unsigned int gazebo::rendering::UserCamera::GetImageWidth ( ) const [virtual]

Get the width of the image.

Returns

Image width

Reimplemented from **gazebo::rendering::Camera** (p. 257).

10.262.3.9 unsigned int gazebo::rendering::UserCamera::GetTriangleCount ( ) const [virtual]

Get the triangle count.

Returns

The number of triangles currently being rendered.

Reimplemented from **gazebo::rendering::Camera** (p. 260).

10.262.3.10 std::string gazebo::rendering::UserCamera::GetViewControllerTypeString ( )

Get current view controller type.

Returns

Type of the current view controller: "orbit", "fps"

10.262.3.11 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

10.262.3.12 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.

10.262.3.13 void gazebo::rendering::UserCamera::HandleKeyPressEvent (const std::string & _key)

Handle a key press.

Parameters

in	*_key*	The key pressed.

10.262.3.14 void gazebo::rendering::UserCamera::HandleKeyReleaseEvent (const std::string & _key)

Handle a key release.

Parameters

in	*_key*	The key released.

10.262.3.15 void gazebo::rendering::UserCamera::HandleMouseEvent (const common::MouseEvent & *_evt*)

Handle a mouse event.

Parameters

in	*_evt*	The mouse event.

10.262.3.16 void gazebo::rendering::UserCamera::Init ( ) [virtual]

Initialize.

Reimplemented from **gazebo::rendering::Camera** (p. 263).

10.262.3.17 bool gazebo::rendering::UserCamera::IsCameraSetInWorldFile ( )

brief Show if the user camera pose has changed in the world file.

return true if the camera pose changed in the world file.

10.262.3.18 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. 264).

10.262.3.19 void gazebo::rendering::UserCamera::Load ( ) [virtual]

Generic load function.

Reimplemented from **gazebo::rendering::Camera** (p. 264).

10.262.3.20 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. 265)

Parameters

in	*_pose*	End position of the camera
in	*_time*	Duration of the camera's movement

Reimplemented from **gazebo::rendering::Camera** (p. 264).

10.262.3.21 void gazebo::rendering::UserCamera::MoveToVisual (VisualPtr *_visual*)

Move the camera to focus on a visual.

Parameters

in	*_visual*	**Visual** (p. 1477) to move the camera to.

10.262.3.22 void gazebo::rendering::UserCamera::MoveToVisual (const std::string & *_visualName*)

Move the camera to focus on a visual.

Parameters

in	*_visual-Name*	Name of the visual to move the camera to.

10.262.3.23 virtual void gazebo::rendering::UserCamera::PostRender ( ) [virtual]

Post render.

Reimplemented from **gazebo::rendering::Camera** (p. 265).

10.262.3.24 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.

10.262.3.25 void gazebo::rendering::UserCamera::SetFocalPoint (const math::Vector3 & *_pt*)

Set the point the camera should orbit around.

Parameters

in	*_pt*	The focal point

10.262.3.26 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. 269).

10.262.3.27 void gazebo::rendering::UserCamera::SetUseSDFPose (bool *_value*)

brief Set if the user camera pose has changed in the world file.

Parameters

in	*_value*	True if the camera pose changed in the world file.

10.262.3.28 void gazebo::rendering::UserCamera::SetViewController (const std::string & *_type*)

Set view controller.

Parameters

in	*_type*	The type of view controller: "orbit", "fps"

10.262.3.29 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.

10.262.3.30 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.

10.262.3.31 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. 270).

10.262.3.32 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. 1477) to track.

Returns

True if the camera is now tracking the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 272).

10.262.3.33 virtual void gazebo::rendering::UserCamera::Update ( ) [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 272).

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

- **UserCamera.hh**

10.263 gazebo::rendering::UserCameraPrivate Class Reference

Private data for the **UserCamera** (p. 1407) class.

`#include <UserCameraPrivate.hh>`

Public Attributes

- **FPSViewController** * **fpsViewController**

  A FPS view controller.
- **GUIOverlay** * **gui**

  The GUI overlay.
- bool **isCameraSetInWorldFile**

  Flag to detect if the user changed the camera pose in the world file.
- **OrbitViewController** * **orbitViewController**

  An orbit view controller.
- SelectionBuffer * **selectionBuffer**

  Draws a 3D axis in the viewport.
- **ViewController** * **viewController**

  The currently active view controller.

10.263.1 Detailed Description

Private data for the **UserCamera** (p. 1407) class.

10.263.2 Member Data Documentation

10.263.2.1 FPSViewController* gazebo::rendering::UserCameraPrivate::fpsViewController

A FPS view controller.

10.263.2.2 GUIOverlay* gazebo::rendering::UserCameraPrivate::gui

The GUI overlay.

10.263.2.3 bool gazebo::rendering::UserCameraPrivate::isCameraSetInWorldFile

Flag to detect if the user changed the camera pose in the world file.

10.263.2.4 OrbitViewController* gazebo::rendering::UserCameraPrivate::orbitViewController

An orbit view controller.

10.263.2.5 SelectionBuffer* gazebo::rendering::UserCameraPrivate::selectionBuffer

Draws a 3D axis in the viewport.

Used to select objects from mouse clicks.

10.263.2.6 ViewController* gazebo::rendering::UserCameraPrivate::viewController

The currently active view controller.

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

- **UserCameraPrivate.hh**

10.264 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.
- double **Distance** (const **Vector2d** &_pt) const

  Calc distance to the given point.
- double **Dot** (const **Vector2d** &_v) const

  Return the dot product of this vector and _v.
- 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.

10.264.1 Detailed Description

Generic double x, y vector.

10.264.2 Constructor & Destructor Documentation

10.264.2.1 gazebo::math::Vector2d::Vector2d ( )

Constructor.

10.264.2.2 gazebo::math::Vector2d::Vector2d (const double & _x, const double & _y)

Constructor.

Parameters

in	_x	value along x
in	_y	value along y

10.264.2.3 gazebo::math::Vector2d::Vector2d (const Vector2d & _v)

Copy constructor.

Parameters

in	_v	the value

10.264.2.4 virtual gazebo::math::Vector2d::~Vector2d ( ) [virtual]

Destructor.

10.264.3 Member Function Documentation

10.264.3.1 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

10.264.3.2 double gazebo::math::Vector2d::Dot (const Vector2d & _v) const

Return the dot product of this vector and _v.

Parameters

in	_v	the vector

Returns

the dot product

10.264.3.3 bool gazebo::math::Vector2d::IsFinite ( ) const

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.264.3.4 void gazebo::math::Vector2d::Normalize ( )

Normalize the vector length.

10.264.3.5 bool gazebo::math::Vector2d::operator!= (const Vector2d & _v) const

Not equal to operator.

Returns

true if elements are of diffent values (tolerence 1e-6)

10.264.3.6 const Vector2d gazebo::math::Vector2d::operator* (const Vector2d & _v) const

Multiplication operators.

Parameters

in	_v	the vector

Returns

the result

10.264.3.7 const Vector2d gazebo::math::Vector2d::operator∗ (double _v) const

Multiplication operators.

Parameters

in	_v	the scaling factor

Returns

a scaled vector

10.264.3.8 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

10.264.3.9 const Vector2d& gazebo::math::Vector2d::operator∗= (double _v)

Multiplication assignment operator.

Parameters

in	_v	the scaling factor

Returns

a scaled vector

10.264.3.10 Vector2d gazebo::math::Vector2d::operator+ (const Vector2d & _v) const

Addition operator.

Parameters

in	_v	vector to add

Returns

sum vector

10.264.3.11 const Vector2d& gazebo::math::Vector2d::operator+= (const Vector2d & _v)

Addition assignment operator.

Parameters

in	_v	the vector to add

10.264.3.12 Vector2d gazebo::math::Vector2d::operator- (const Vector2d & _v) const

Subtraction operator.

Parameters

in	_v	the vector to substract

Returns

the subtracted vector

10.264.3.13 const Vector2d& gazebo::math::Vector2d::operator-= (const Vector2d & _v)

Subtraction assignment operator.

Parameters

in	_v	the vector to substract

Returns

this

10.264.3.14 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

10.264.3.15 const Vector2d gazebo::math::Vector2d::operator/ (double _v) const

Division operator.

Parameters

in	_v	the value

Returns

a vector

10.264.3.16 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

10.264.3.17 const Vector2d& gazebo::math::Vector2d::operator/= (double _v)

Division operator.

Parameters

in	_v	the divisor

Returns

a vector

10.264.3.18 Vector2d& gazebo::math::Vector2d::operator= (const Vector2d & _v)

Assignment operator.

Parameters

in	_v	a value for x and y element

Returns

this

10.264.3.19 const Vector2d& gazebo::math::Vector2d::operator= (double _v)

Assignment operator.

Parameters

in	*_v*	the value for x and y element

Returns

this

10.264.3.20 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)

10.264.3.21 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

10.264.3.22 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

10.264.4 Friends And Related Function Documentation

10.264.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector2d & _pt) [friend]

Stream extraction operator.

Parameters

in	*_out*	output stream
in	*_pt*	**Vector2d** (p. 1420) to output

Returns

The stream

10.264.4.2 std::istream& operator>> (std::istream & _in, gazebo::math::Vector2d & _pt) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_pt*	**Vector3** (p. 1440) to read values into

Returns

The stream

10.264.5 Member Data Documentation

10.264.5.1 double gazebo::math::Vector2d::x

x data

10.264.5.2 double gazebo::math::Vector2d::y

y data

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

- **Vector2d.hh**

10.265 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.

10.265.1 Detailed Description

Generic integer x, y vector.

10.265.2 Constructor & Destructor Documentation

10.265.2.1 gazebo::math::Vector2i::Vector2i ( )

Constructor.

10.265.2.2 gazebo::math::Vector2i::Vector2i (const int & _x, const int & _y)

Constructor.

Parameters

in	_x	value along x
in	_y	value along y

10.265.2.3 gazebo::math::Vector2i::Vector2i (const Vector2i & _pt)

Copy onstructor.

Parameters

in	_pt	a point

10.265.2.4 virtual gazebo::math::Vector2i::~Vector2i ( ) [virtual]

Destructor.

10.265.3 Member Function Documentation

10.265.3.1 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

10.265.3.2 int gazebo::math::Vector2i::Distance (const Vector2i & _pt) const

Calc distance to the given point.

Parameters

in	*_pt*	a point

Returns

the distance

10.265.3.3 bool gazebo::math::Vector2i::IsFinite ( ) const

See if a point is finite (e.g., not nan)

Returns

the result

10.265.3.4 void gazebo::math::Vector2i::Normalize ( )

Normalize the vector length.

10.265.3.5 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

10.265.3.6 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

10.265.3.7 const Vector2i gazebo::math::Vector2i::operator∗ (int _v) const

Multiplication operator.

Parameters

in	_v	the scaling factor

Returns

the result

10.265.3.8 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

10.265.3.9 const Vector2i& gazebo::math::Vector2i::operator*= (int _v)

Multiplication operator.

Parameters

in	_v	scaling factor

Returns

this

10.265.3.10 Vector2i gazebo::math::Vector2i::operator+ (const Vector2i & _v) const

Addition operator.

Parameters

in	_v	the vector to add

Returns

the sum vector

10.265.3.11 const Vector2i& gazebo::math::Vector2i::operator+= (const Vector2i & _v)

Addition assignment operator.

Parameters

in	_v	the vector to add

Returns

this

10.265.3.12 Vector2i gazebo::math::Vector2i::operator- (const Vector2i & _v) const

Subtraction operator.

Parameters

in	_v	the vector to substract

Returns

the result vector

10.265.3.13 const Vector2i& gazebo::math::Vector2i::operator-= (const Vector2i & _v)

Subtraction operators.

Parameters

in	_v	the vector to substract

Returns

this

10.265.3.14 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

10.265.3.15 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

10.265.3.16 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

10.265.3.17 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

10.265.3.18 Vector2i& gazebo::math::Vector2i::operator= (const Vector2i & _v)

Assignment operator.

Parameters

in	*_v*	the value

Returns

this

10.265.3.19 const Vector2i& gazebo::math::Vector2i::operator= (int *_value*)

Assignment operator.

Parameters

in	*_value*	the value for x and y

Returns

this

10.265.3.20 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

10.265.3.21 int gazebo::math::Vector2i::operator[] (unsigned int _index) const

Array subscript operator.

Parameters

in	*_index*	the array index

10.265.3.22 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

10.265.4 Friends And Related Function Documentation

10.265.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector2i & _pt) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*pt*	**Vector2i** (p. 1430) to output

Returns

the stream

10.265.4.2 std::istream& operator>> (std::istream & _in, gazebo::math::Vector2i & _pt) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_pt*	**Vector3** (p. 1440) to read values into

Returns

The stream

10.265.5 Member Data Documentation

10.265.5.1 int gazebo::math::Vector2i::x

x data

10.265.5.2 int gazebo::math::Vector2i::y

y data

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

- **Vector2i.hh**

10.266 gazebo::math::Vector3 Class Reference

The **Vector3** (p. 1440) 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.

10.266.1 Detailed Description

The **Vector3** (p. 1440) 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.

10.266.2 Constructor & Destructor Documentation

10.266.2.1 gazebo::math::Vector3::Vector3 ( )

Constructor.

10.266.2.2 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

10.266.2.3 gazebo::math::Vector3::Vector3 (const Vector3 & _v)

Copy constructor.

Parameters

in	_v	a vector

10.266.2.4 virtual gazebo::math::Vector3::~Vector3 ( ) [virtual]

Destructor.

10.266.3 Member Function Documentation

10.266.3.1 void gazebo::math::Vector3::Correct ( ) [inline]

Corrects any nan values.

10.266.3.2 Vector3 gazebo::math::Vector3::Cross (const Vector3 & _pt) const

Return the cross product of this vector and pt.

Returns

the product

10.266.3.3 double gazebo::math::Vector3::Distance (const Vector3 & _pt) const

Calc distance to the given point.

Parameters

in	*_pt*	the point

Returns

the distance

10.266.3.4 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

10.266.3.5 double gazebo::math::Vector3::Dot (const Vector3 & _pt) const

Return the dot product of this vector and pt.

Returns

the product

10.266.3.6 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

10.266.3.7 Vector3 gazebo::math::Vector3::GetAbs ( ) const

Get the absolute value of the vector.

Returns

a vector with positive elements

10.266.3.8 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

10.266.3.9 double gazebo::math::Vector3::GetLength ( ) const

Returns the length (magnitude) of the vector \ return the length.

10.266.3.10 double gazebo::math::Vector3::GetMax ( ) const

Get the maximum value in the vector.

Returns

the maximum element

10.266.3.11 double gazebo::math::Vector3::GetMin ( ) const

Get the minimum value in the vector.

Returns

the minimum element

10.266.3.12 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

10.266.3.13 Vector3 gazebo::math::Vector3::GetPerpendicular ( ) const

Return a vector that is perpendicular to this one.

Returns

an orthogonal vector

10.266.3.14 Vector3 gazebo::math::Vector3::GetRounded ( ) const

Get a rounded version of this vector.

Returns

a rounded vector

10.266.3.15 double gazebo::math::Vector3::GetSquaredLength ( ) const

Return the square of the length (magnitude) of the vector.

Returns

the squared length

10.266.3.16 double gazebo::math::Vector3::GetSum ( ) const

Return the sum of the values.

Returns

the sum

10.266.3.17 bool gazebo::math::Vector3::IsFinite ( ) const

See if a point is finite (e.g., not nan)

10.266.3.18 Vector3 gazebo::math::Vector3::Normalize ( )

Normalize the vector length.

Returns

unit length vector

10.266.3.19 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

10.266.3.20 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	

10.266.3.21 Vector3 gazebo::math::Vector3::operator∗ (double _v) const

Multiplication operators.

Parameters

in	_v	the scaling factor

Returns

a scaled vector

10.266.3.22 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

10.266.3.23 const Vector3& gazebo::math::Vector3::operator*= (double _v)

Multiplication operator.

Parameters

in	_v	scaling factor

Returns

this

10.266.3.24 Vector3 gazebo::math::Vector3::operator+ (const Vector3 & _v) const

Addition operator.

Parameters

in	_v	vector to add

Returns

the sum vector

10.266.3.25 const Vector3& gazebo::math::Vector3::operator+= (const Vector3 & _v)

Addition assignment operator.

Parameters

in	_v	vector to add

10.266.3.26 Vector3 gazebo::math::Vector3::operator- ( ) const [inline]

Negation operator.

Returns

negative of this vector

10.266.3.27 Vector3 gazebo::math::Vector3::operator- (const Vector3 & _pt) const [inline]

Subtraction operators.

Parameters

in	*_pt*	a vector to substract

Returns

a vector

References x, y, and z.

10.266.3.28 const Vector3& gazebo::math::Vector3::operator-= (const Vector3 & _pt)

Subtraction operators.

Parameters

in	*_pt*	subtrahend

10.266.3.29 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

10.266.3.30 const Vector3 gazebo::math::Vector3::operator/ (double _v) const

Division operator.

Remarks

this is an element wise division

Returns

a vector

10.266.3.31 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

10.266.3.32 const Vector3& gazebo::math::Vector3::operator/= (double _v)

Division operator.

Remarks

this is an element wise division

Returns

this

10.266.3.33 Vector3& gazebo::math::Vector3::operator= (const Vector3 & _v)

Assignment operator.

Parameters

in	_v	a new value

Returns

this

10.266.3.34 Vector3& gazebo::math::Vector3::operator= (double *_value*)

Assignment operator.

Parameters

in	*_value*	assigned to all elements

Returns

this

10.266.3.35 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

10.266.3.36 double gazebo::math::Vector3::operator[] (unsigned int *index*) const

[] operator

10.266.3.37 Vector3 gazebo::math::Vector3::Round ( )

Round to near whole number, return the result.

Returns

the result

10.266.3.38 void gazebo::math::Vector3::Round (int _precision)

Round all values to _precision decimal places.

Parameters

in	*_precision*	the decimal places

10.266.3.39 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

10.266.3.40 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

10.266.3.41 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

10.266.4 Friends And Related Function Documentation

10.266.4.1 Vector3 operator* (double _s, const Vector3 & _v) [friend]

Multiplication operators.

Parameters

in	*_s*	the scaling factor
in	*_v*	input vector

Returns

a scaled vector

10.266.4.2 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Vector3 & *_pt*) [friend]

Stream insertion operator.

Parameters

_out	output stream
_pt	**Vector3** (p. 1440) to output

Returns

the stream

10.266.4.3 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

10.266.5 Member Data Documentation

10.266.5.1 const Vector3 gazebo::math::Vector3::One [static]

math::Vector3(1, 1, 1)

10.266.5.2 const Vector3 gazebo::math::Vector3::UnitX [static]

math::Vector3(1, 0, 0)

10.266.5.3 const Vector3 gazebo::math::Vector3::UnitY [static]

math::Vector3(0, 1, 0)

10.266.5.4 const Vector3 gazebo::math::Vector3::UnitZ [static]

math::Vector3(0, 0, 1)

10.266.5.5 double gazebo::math::Vector3::x

X location.

Referenced by gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::Pose::-CoordPositionSub(), gazebo::math::Matrix3::operator∗(), operator-(), gazebo::math-::Quaternion::RotateVector(), gazebo::physics::SimbodyBoxShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

10.266.5.6 double gazebo::math::Vector3::y

Y location.

Referenced by gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::Pose::-CoordPositionSub(), gazebo::math::Matrix3::operator∗(), operator-(), gazebo::math-::Quaternion::RotateVector(), gazebo::physics::SimbodyBoxShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

10.266.5.7 double gazebo::math::Vector3::z

Z location.

Referenced by gazebo::physics::DARTTypes::ConvVec3(), gazebo::math::Pose::-CoordPositionSub(), gazebo::math::Matrix3::operator∗(), operator-(), gazebo::math-::Quaternion::RotateVector(), gazebo::physics::SimbodyBoxShape::SetSize(), and gazebo::physics::DARTBoxShape::SetSize().

10.266.5.8 const Vector3 gazebo::math::Vector3::Zero [static]

math::Vector3(0, 0, 0)

Referenced by gazebo::physics::Link::GetWorldLinearVel().

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

- **Vector3.hh**

10.267 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.

10.267.1 Detailed Description

double Generic x, y, z, w vector

10.267.2 Constructor & Destructor Documentation

10.267.2.1 gazebo::math::Vector4::Vector4 ( )

Constructor.

10.267.2.2 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

10.267.2.3 gazebo::math::Vector4::Vector4 (const Vector4 & _v)

Copy constructor.

Parameters

in	_v	vector

10.267.2.4 virtual gazebo::math::Vector4::~Vector4 ( ) [virtual]

Destructor.

10.267.3 Member Function Documentation

10.267.3.1 double gazebo::math::Vector4::Distance (const Vector4 & _pt) const

Calc distance to the given point.

Parameters

in	*_pt*	the point

Returns

the distance

10.267.3.2 double gazebo::math::Vector4::GetLength ( ) const

Returns the length (magnitude) of the vector.

10.267.3.3 double gazebo::math::Vector4::GetSquaredLength ( ) const

Return the square of the length (magnitude) of the vector.

Returns

the length

10.267.3.4 bool gazebo::math::Vector4::IsFinite ( ) const

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.267.3.5 void gazebo::math::Vector4::Normalize ( )

Normalize the vector length.

10.267.3.6 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

10.267.3.7 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

10.267.3.8 const Vector4 gazebo::math::Vector4::operator* (const Matrix4 & _m) const

Matrix multiplication operator.

Parameters

in	*_m*	matrix

Returns

the vector multiplied by _m

10.267.3.9 const Vector4 gazebo::math::Vector4::operator∗ (double _v) const

Multiplication operators.

Parameters

in	_v	scaling factor

Returns

a scaled vector

10.267.3.10 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

10.267.3.11 const Vector4& gazebo::math::Vector4::operator∗= (double _v)

Multiplication assignment operator.

Parameters

in	_v	scaling factor

Returns

this

10.267.3.12 Vector4 gazebo::math::Vector4::operator+ (const Vector4 & _v) const

Addition operator.

Parameters

in	_v	the vector to add

Returns

a sum vector

10.267.3.13 const Vector4& gazebo::math::Vector4::operator+= (const Vector4 & _v)

Addition operator.

Parameters

in	_v	the vector to add

Returns

this vector

10.267.3.14 Vector4 gazebo::math::Vector4::operator- (const Vector4 & _v) const

Subtraction operator.

Parameters

in	_v	the vector to substract

Returns

a vector

10.267.3.15 const Vector4& gazebo::math::Vector4::operator-= (const Vector4 & _v)

Subtraction assigment operators.

Parameters

in	*_v*	the vector to substract

Returns

this vector

10.267.3.16 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

10.267.3.17 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

10.267.3.18 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

10.267.3.19 const Vector4& gazebo::math::Vector4::operator/= (double _v)

Division operator.

Parameters

in	_v	scaling factor

Returns

a vector

10.267.3.20 Vector4& gazebo::math::Vector4::operator= (const Vector4 & _v)

Assignment operator.

Parameters

in	_v	the vector

Returns

a reference to this vector

10.267.3.21 Vector4& gazebo::math::Vector4::operator= (double _value)

Assignment operator.

Parameters

in	_value	

10.267.3.22 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

10.267.3.23 double gazebo::math::Vector4::operator[] (unsigned int _index) const

Array subscript operator.

Parameters

in	_index	

10.267.3.24 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

10.267.4 Friends And Related Function Documentation

10.267.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector4 & _pt) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_pt*	**Vector4** (p. 1457) to output

Returns

The stream

10.267.4.2 std::istream& operator>> (std::istream & _in, gazebo::math::Vector4 & _pt) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_pt*	**Vector4** (p. 1457) to read values into

Returns

the stream

10.267.5 Member Data Documentation

10.267.5.1 double gazebo::math::Vector4::w

W value.

10.267.5.2 double gazebo::math::Vector4::x

X value.

10.267.5.3 double gazebo::math::Vector4::y

Y value.

10.267.5.4 double gazebo::math::Vector4::z

Z value.

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

- **Vector4.hh**

10.268 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.

10.268.1 Detailed Description

Handle video encoding and decoding using libavcodec.

10.268.2 Constructor & Destructor Documentation

10.268.2.1 gazebo::common::Video::Video ( )

Constructor.

10.268.2.2 virtual gazebo::common::Video::~Video ( ) [virtual]

Destructor.

10.268.3 Member Function Documentation

10.268.3.1 int gazebo::common::Video::GetHeight ( ) const

Get the height of the video in pixels.

Returns

the height

10.268.3.2 bool gazebo::common::Video::GetNextFrame (unsigned char ** *_buffer*)

Get the next frame of the video.

Parameters

out	*_img*	**Image** (p. 637) in which the frame is stored

Returns

false if HAVE_FFMPEG is not defined, true otherwise

10.268.3.3 int gazebo::common::Video::GetWidth ( ) const

Get the width of the video in pixels.

Returns

the width

10.268.3.4 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:

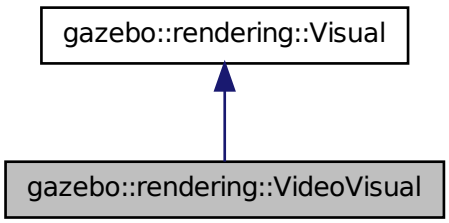
- **Video.hh**

10.269 gazebo::rendering::VideoVisual Class Reference

A visual element that displays a video as a texture.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::VideoVisual:



Public Member Functions

- **VideoVisual** (const std::string &_name, **VisualPtr** _parent)

  Constructor.
- virtual ~**VideoVisual** ()

  Destructor.

10.269.1 Detailed Description

A visual element that displays a video as a texture.

10.269.2 Constructor & Destructor Documentation

10.269.2.1 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.

10.269.2.2 virtual gazebo::rendering::VideoVisual::~VideoVisual ( ) [virtual]

Destructor.

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

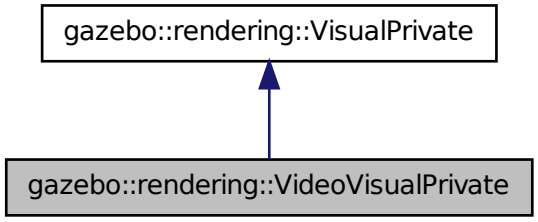
- **VideoVisual.hh**

10.270 gazebo::rendering::VideoVisualPrivate Class Reference

Private data for the Video **Visual** (p. 1477) class.

`#include <VideoVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::VideoVisualPrivate:



Public Attributes

- std::vector< **event::ConnectionPtr** > **connections**

  All the event connections.
- int **height**

  Height of the video.
- unsigned char * **imageBuffer**

  One frame of the viedeo.
- Ogre::TexturePtr **texture**

  Texture to draw the video onto.
- **common::Video** * **video**

  Load a video.
- int **width**

  Width of the video.

10.270.1 Detailed Description

Private data for the Video **Visual** (p. 1477) class.

10.270.2 Member Data Documentation

10.270.2.1 std::vector<event::ConnectionPtr> gazebo::rendering::VideoVisualPrivate::connections

All the event connections.

10.270.2.2 int gazebo::rendering::VideoVisualPrivate::height

Height of the video.

10.270.2.3 unsigned char* gazebo::rendering::VideoVisualPrivate::imageBuffer

One frame of the viedeo.

10.270.2.4 Ogre::TexturePtr gazebo::rendering::VideoVisualPrivate::texture

Texture to draw the video onto.

10.270.2.5 common::Video* gazebo::rendering::VideoVisualPrivate::video

Load a video.

10.270.2.6 int gazebo::rendering::VideoVisualPrivate::width

Width of the video.

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

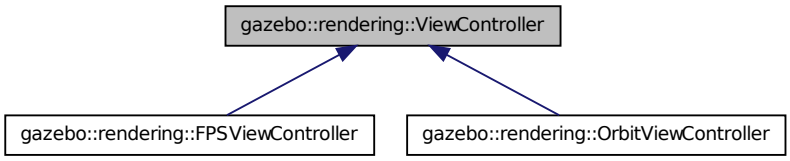
- **VideoVisualPrivate.hh**

10.271 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. 242).

Protected Attributes

- **UserCameraPtr camera**

  Pointer to the camera to control.

- bool **enabled**

  True if enabled.

- std::string **typeString**

  Type of view controller.

10.271.1 Detailed Description

Base class for view controllers.

10.271.2 Constructor & Destructor Documentation

10.271.2.1 gazebo::rendering::ViewController::ViewController (UserCameraPtr *_camera*)

Constructor.

Parameters

in	*_camera*	The user camera to controll.

10.271.2.2 virtual gazebo::rendering::ViewController::~ViewController ( ) [virtual]

Destructor.

10.271.3 Member Function Documentation

10.271.3.1 std::string gazebo::rendering::ViewController::GetTypeString ( ) const

Get the type of view controller.

Returns

The view controller type string.

10.271.3.2 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. 957), and **gazebo::rendering::FPSViewController** (p. 563).

10.271.3.3 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. 957), and **gazebo::rendering::FPSViewController** (p. 563).

10.271.3.4 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. 958), and **gazebo::rendering::FPSViewController** (p. 563).

10.271.3.5 virtual void gazebo::rendering::ViewController::Init ( ) [pure virtual]

Initialize the view controller.

Implemented in **gazebo::rendering::OrbitViewController** (p. 958), and **gazebo::rendering::FPSViewController** (p. 564).

10.271.3.6 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. 958).

10.271.3.7 void gazebo::rendering::ViewController::SetEnabled (bool _value)

Set whether the controller is enabled.

Parameters

in	*_value*	True if the controller is enabled.

10.271.3.8 virtual void gazebo::rendering::ViewController::Update ( ) [pure virtual]

Update the controller, which should update the position of the **Camera** (p. 242).

Implemented in **gazebo::rendering::OrbitViewController** (p. 959), and **gazebo::rendering::FPSViewController** (p. 564).

10.271.4 Member Data Documentation

10.271.4.1 UserCameraPtr gazebo::rendering::ViewController::camera [protected]

Pointer to the camera to control.

10.271.4.2 bool gazebo::rendering::ViewController::enabled [protected]

True if enabled.

10.271.4.3 std::string gazebo::rendering::ViewController::typeString [protected]

Type of view controller.

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

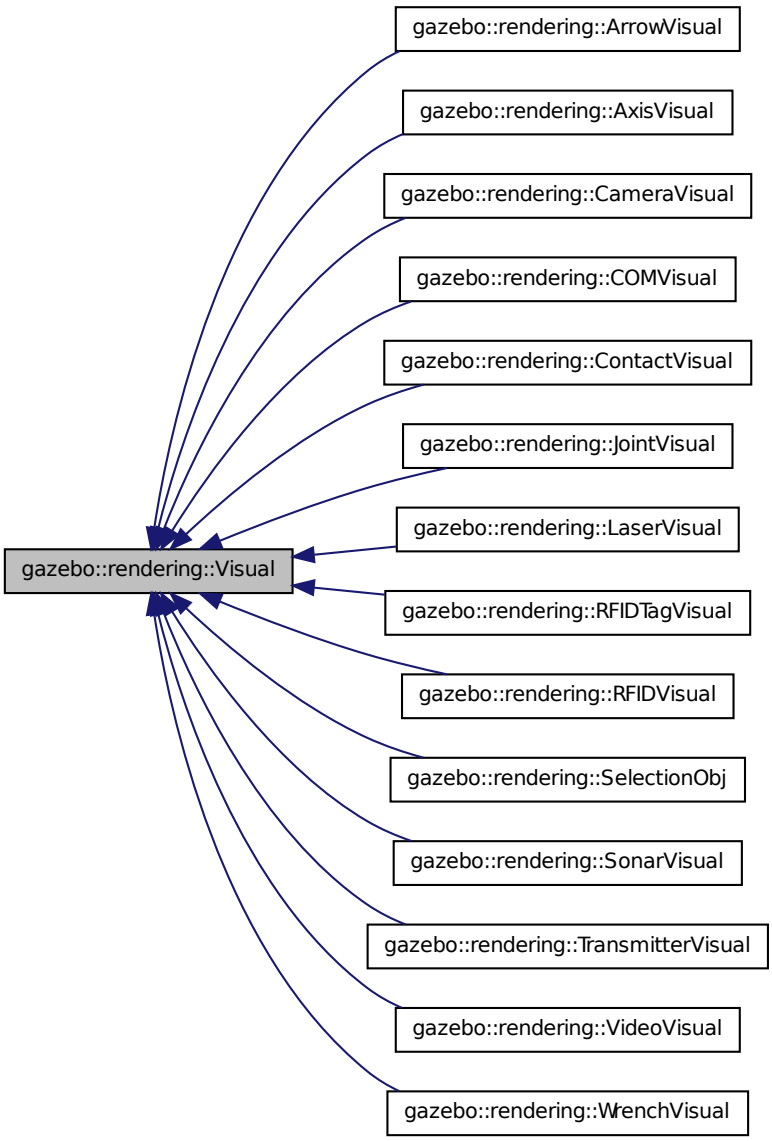
- **ViewController.hh**

10.272 gazebo::rendering::Visual Class Reference

A renderable object.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::Visual:



Public Member Functions

- **Visual** (const std::string &_name, **VisualPtr** _parent, bool _useRTShader=true)

  Constructor.
- **Visual** (const std::string &_name, **ScenePtr** _scene, bool _useRTShader=true)

  Constructor.
- virtual ∼**Visual** ()

  Destructor.
- void **AttachAxes** ()

  Attach visualization axes.
- void **AttachLineVertex** (**DynamicLines** ∗_line, unsigned int _index)

  Attach a vertex of a line to the position of the visual.
- Ogre::MovableObject ∗ **AttachMesh** (const std::string &_meshName, const std::string &_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.
- bool **GetHighlighted** () const

  Get whether or not the visual is visually highlighted.
- uint32_t **GetId** () const

  Get the id associated with this visual.
- 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. 163).*

- 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 **SetId** (uint32_t _id)

  Set the id associated with this visual.
- void **SetLighting** (bool _lighting)

  Set whether or not to enable or disable lighting.
- 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. 163).*

Protected Member Functions

- **Visual** (**VisualPrivate** &_dataPtr, const std::string &_name, **VisualPtr** _parent, bool _useRTShader=true)
- **Visual** (**VisualPrivate** &_dataPtr, const std::string &_name, **ScenePtr** _scene, bool _useRTShader=true)

Protected Attributes

- **VisualPrivate** ∗ **dataPtr**

10.272.1 Detailed Description

A renderable object.

10.272.2 Constructor & Destructor Documentation

10.272.2.1 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	*_useRT-Shader*	True if the visual should use the real-time shader system (RTShader).

10.272.2.2 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. 1097) containing the visual.
in	*_useRT-Shader*	True if the visual should use the real-time shader system (RTShader).

10.272.2.3 virtual gazebo::rendering::Visual::~Visual ( ) [virtual]

Destructor.

10.272.2.4 gazebo::rendering::Visual::Visual (VisualPrivate & *_dataPtr,* const std::string & *_name,* VisualPtr *_parent,* bool *_useRTShader* = true) [protected]

10.272.2.5 gazebo::rendering::Visual::Visual (VisualPrivate & *_dataPtr,* const std::string & *_name,* ScenePtr *_scene,* bool *_useRTShader* = true) [protected]

10.272.3 Member Function Documentation

10.272.3.1 void gazebo::rendering::Visual::AttachAxes ( )

Attach visualization axes.

10.272.3.2 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.

10.272.3.3 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.

10.272.3.4 void gazebo::rendering::Visual::AttachObject (Ogre::MovableObject * *_obj*)

Attach a renerable object to the visual.

Parameters

in	*_obj*	A movable object to attach to the visual.

10.272.3.5 void gazebo::rendering::Visual::AttachVisual (VisualPtr *_vis*)

Attach a visual to this visual.

Parameters

in	*_vis*	**Visual** (p. 1477) to attach.

10.272.3.6 void gazebo::rendering::Visual::ClearParent ( )

Clear parents.

10.272.3.7 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. 1477).
in	*_newParent*	Parent of the cloned **Visual** (p. 1477).

Returns

The visual.

10.272.3.8 DynamicLines* gazebo::rendering::Visual::CreateDynamicLine (RenderOpType *_type* = RENDERING_LINE_STRIP)

Add a line to the visual.

Parameters

in	*_type*	The type of line to make.

Returns

A pointer to the new dynamic line.

10.272.3.9 void gazebo::rendering::Visual::DeleteDynamicLine (DynamicLines * _line)

Delete a dynamic line.

Parameters

in	*_line*	Pointer to the line to delete.

10.272.3.10 void gazebo::rendering::Visual::DetachObjects ( )

Detach all objects.

10.272.3.11 void gazebo::rendering::Visual::DetachVisual (VisualPtr _vis)

Detach a visual.

Parameters

in	*_vis*	**Visual** (p. 1477) to detach.

10.272.3.12 void gazebo::rendering::Visual::DetachVisual (const std::string & _name)

Detach a visual.

Parameters

in	*_name*	Name of the visual to detach.

10.272.3.13 void gazebo::rendering::Visual::DisableTrackVisual ( )

Disable tracking of a visual.

10.272.3.14 void gazebo::rendering::Visual::EnableTrackVisual (VisualPtr _vis)

Set one visual to track/follow another.

Parameters

in	*_vis*	**Visual** (p. 1477) to track.

10.272.3.15 void gazebo::rendering::Visual::Fini ( )

Helper for the destructor.

10.272.3.16 unsigned int gazebo::rendering::Visual::GetAttachedObjectCount ( ) const

Return the number of attached movable objects.

Returns

The number of attached movable objects.

10.272.3.17 math::Box gazebo::rendering::Visual::GetBoundingBox ( ) const

Get the bounding box for the visual.

Returns

The bounding box in world coordinates.

10.272.3.18 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. 1489).

Parameters

in	*_index*	Index of the child to retreive.

Returns

Pointer to the child visual, NULL if index is invalid.

10.272.3.19 unsigned int gazebo::rendering::Visual::GetChildCount ( )

Get the number of attached visuals.

Returns

The number of children.

10.272.3.20 bool gazebo::rendering::Visual::GetHighlighted ( ) const

Get whether or not the visual is visually highlighted.

This is most often means that an object is selected by a user via the GUI.

Returns

True if the visual is highlighted.

10.272.3.21 uint32_t gazebo::rendering::Visual::GetId ( ) const

Get the id associated with this visual.

10.272.3.22 std::string gazebo::rendering::Visual::GetMaterialName ( ) const

Get the name of the material.

Returns

The name of the visual applied to this visual.

10.272.3.23 std::string gazebo::rendering::Visual::GetMeshName ( ) const

The name of the mesh set in the visual's SDF.

Returns

Name of the mesh.

10.272.3.24 std::string gazebo::rendering::Visual::GetName ( ) const

Get the name of the visual.

Returns

The name of the visual.

10.272.3.25 std::string gazebo::rendering::Visual::GetNormalMap ( ) const

Get the normal map.

Returns

The name of the normal map material.

10.272.3.26 VisualPtr gazebo::rendering::Visual::GetParent ( ) const

Get the parent visual, if one exists.

Returns

Pointer to the parent visual, NULL if no parent.

10.272.3.27 math::Pose gazebo::rendering::Visual::GetPose ( ) const

Get the pose of the visual.

Returns

The **Visual** (p. 1477)'s pose.

10.272.3.28 math::Vector3 gazebo::rendering::Visual::GetPosition ( ) const

Get the position of the visual.

Returns

The visual's position.

10.272.3.29 VisualPtr gazebo::rendering::Visual::GetRootVisual ( )

Get the root visual.

Returns

The root visual, which is one level below the world visual.

10.272.3.30 math::Quaternion gazebo::rendering::Visual::GetRotation ( ) const

Get the rotation of the visual.

Returns

The visual's rotation.

10.272.3.31 math::Vector3 gazebo::rendering::Visual::GetScale ( )

Get the scale.

Returns

The scaling factor.

10.272.3.32 ScenePtr gazebo::rendering::Visual::GetScene ( ) const

Get current.

Returns

Pointer to the scene.

10.272.3.33 Ogre::SceneNode* gazebo::rendering::Visual::GetSceneNode ( ) const

Return the scene Node of this visual entity.

Returns

The **Ogre** (p. 163) scene node.

10.272.3.34 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".

10.272.3.35 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.

10.272.3.36 float gazebo::rendering::Visual::GetTransparency ( )

Get the transparency.

Returns

The transparency.

10.272.3.37 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. 1748)
GZ_VISIBILITY_GUI (p. 1748)
GZ_VISIBILITY_SELECTABLE (p. 1748)

10.272.3.38 bool gazebo::rendering::Visual::GetVisible ( ) const

Get whether the visual is visible.

Returns

True if the visual is visible.

10.272.3.39 math::Pose gazebo::rendering::Visual::GetWorldPose ( ) const

Get the global pose of the node.

Returns

The pose in the world coordinate frame.

10.272.3.40 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.

10.272.3.41 void gazebo::rendering::Visual::Init ( )

Helper for the contructor.

10.272.3.42 void gazebo::rendering::Visual::InsertMesh (const std::string & *_meshName*, const std::string & *_subMesh* = " ", bool *_centerSubmesh* = false)

Insert a mesh into **Ogre** (p. 163).

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.

10.272.3.43 static void gazebo::rendering::Visual::InsertMesh (const common::Mesh * _mesh, const std::string & _subMesh = "", bool _centerSubmesh = false) [static]

Insert a mesh into **Ogre** (p. 163).

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.

10.272.3.44 bool gazebo::rendering::Visual::IsPlane ( ) const

Return true if the visual is a plane.

Returns

True if a plane.

10.272.3.45 bool gazebo::rendering::Visual::IsStatic ( ) const

Return true if the visual is a static geometry.

Returns

True if the visual is static.

10.272.3.46 void gazebo::rendering::Visual::Load (sdf::ElementPtr _sdf)

Load the visual with a set of parameters.

Parameters

in	*_sdf*	Load from an SDF element.

Reimplemented in **gazebo::rendering::COMVisual** (p. 326).

10.272.3.47 virtual void gazebo::rendering::Visual::Load ( ) [virtual]

Load the visual with default parameters.

Reimplemented in **gazebo::rendering::SelectionObj** (p. 1124), **gazebo::rendering::SonarVisual** (p. 1304), **gazebo::rendering::TransmitterVisual** (p. 1401), **gazebo::rendering::AxisVisual** (p. 196), and **gazebo::rendering::ArrowVisual** (p. 189).

10.272.3.48 void gazebo::rendering::Visual::LoadFromMsg (ConstVisualPtr & *_msg*)

Load from a message.

Parameters

in	*_msg*	A visual message.

10.272.3.49 void gazebo::rendering::Visual::LoadPlugin (const std::string & *_filename*, const std::string & *_name*, sdf::ElementPtr *_sdf*)

Load a plugin.

Parameters

_filename	The filename of the plugin
_name	A unique name for the plugin
_sdf	The SDF to pass into the plugin.

10.272.3.50 void gazebo::rendering::Visual::MakeStatic ( )

Make the visual objects static renderables.

10.272.3.51 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.

10.272.3.52 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.

10.272.3.53 void gazebo::rendering::Visual::RemovePlugin (const std::string & *_name*)

Remove a running plugin.

Parameters

_name	The unique name of the plugin to remove

10.272.3.54 void gazebo::rendering::Visual::SetAmbient (const common::Color & *_color*)

Set the ambient color of the visual.

Parameters

in	*_color*	The ambient color.

10.272.3.55 void gazebo::rendering::Visual::SetCastShadows (bool *_shadows*)

Set whether the visual should cast shadows.

Parameters

in	*_shadows*	True to enable shadows.

10.272.3.56 void gazebo::rendering::Visual::SetDiffuse (const common::Color & _color)

Set the diffuse color of the visual.

Parameters

in	*_color*	Set the diffuse color.

10.272.3.57 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. 728).

10.272.3.58 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.

10.272.3.59 void gazebo::rendering::Visual::SetId (uint32_t _id)

Set the id associated with this visual.

10.272.3.60 void gazebo::rendering::Visual::SetLighting (bool _lighting)

Set whether or not to enable or disable lighting.

Parameters

in	*_lighting*	True to enable lighting.

10.272.3.61 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.

10.272.3.62 void gazebo::rendering::Visual::SetName (const std::string & *_name*)

Set the name of the visual.

Parameters

in	*_name*	Name of the visual

10.272.3.63 void gazebo::rendering::Visual::SetNormalMap (const std::string & *_nmap*)

Set the normal map.

Parameters

in	*_nmap*	Name of the normal map material.

10.272.3.64 void gazebo::rendering::Visual::SetPose (const math::Pose & *_pose*)

Set the pose of the visual.

Parameters

in	*_pose*	The new pose of the visual.

10.272.3.65 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.

10.272.3.66 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.

10.272.3.67 void gazebo::rendering::Visual::SetRotation (const math::Quaternion & _rot)

Set the rotation of the visual.

Parameters

in	*_rot*	The rotation of the visual.

10.272.3.68 void gazebo::rendering::Visual::SetScale (const math::Vector3 & _scale)

Set the scale.

Parameters

in	*_scale*	The scaling factor for the visual.

10.272.3.69 void gazebo::rendering::Visual::SetScene (ScenePtr _scene)

Set current scene.

Parameters

in	*_scene*	Pointer to the scene.

10.272.3.70 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".

10.272.3.71 void gazebo::rendering::Visual::SetSkeletonPose (const msgs::PoseAnimation & _pose)

Set animation skeleton pose.

Parameters

in	*_pose*	Skelton message

10.272.3.72 void gazebo::rendering::Visual::SetSpecular (const common::Color & _color)

Set the specular color of the visual.

Parameters

in	*_color*	Specular color.

10.272.3.73 void gazebo::rendering::Visual::SetTransparency (float _trans)

Set the transparency.

Parameters

in	*_trans*	The transparency, between 0 and 1 where 0 is no transparency.

10.272.3.74 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. 1748)
GZ_VISIBILITY_GUI (p. 1748)
GZ_VISIBILITY_SELECTABLE (p. 1748)

10.272.3.75 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.

10.272.3.76 void gazebo::rendering::Visual::SetWireframe (bool *_show*)

Enable or disable wireframe for this visual.

Parameters

in	*_show*	True to enable wireframe for this visual.

10.272.3.77 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.

10.272.3.78 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.

10.272.3.79 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.

10.272.3.80 void gazebo::rendering::Visual::ShowBoundingBox ( )

Display the bounding box visual.

10.272.3.81 void gazebo::rendering::Visual::ShowCollision (bool *_show*)

Display the collision visuals.

Parameters

in	*_show*	True to show visuals labeled as collision objects.

10.272.3.82 void gazebo::rendering::Visual::ShowCOM (bool *_show*)

Display Center of Mass visuals.

Parameters

in	*_show*	True to show center of mass visualizations.

10.272.3.83 void gazebo::rendering::Visual::ShowJoints (bool *_show*)

Display joint visuals.

Parameters

in	*_show*	True to show joint visualizations.

10.272.3.84 void gazebo::rendering::Visual::ShowSkeleton (bool *_show*)

Display the skeleton visuals.

Parameters

in	*_show*	True to show skeleton visuals.

10.272.3.85 void gazebo::rendering::Visual::ToggleVisible ( )

Toggle whether this visual is visible.

10.272.3.86 void gazebo::rendering::Visual::Update ( )

Update the visual.

Reimplemented in **gazebo::rendering::TransmitterVisual** (p. 1401).

10.272.3.87 void gazebo::rendering::Visual::UpdateFromMsg (ConstVisualPtr & *_msg*)

Update a visual based on a message.

Parameters

in	*_msg*	The visual message.

10.272.4 Member Data Documentation

10.272.4.1 VisualPrivate* gazebo::rendering::Visual::dataPtr [protected]

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

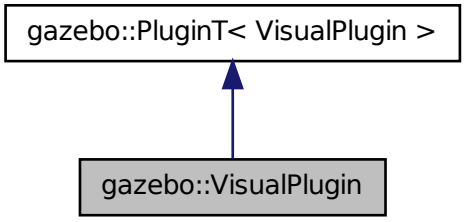
- **Visual.hh**

10.273 gazebo::VisualPlugin Class Reference

A plugin loaded within the gzserver on startup.

`#include <Plugin.hh>`

Inheritance diagram for gazebo::VisualPlugin:



Public Member Functions

- **VisualPlugin** ()
- virtual void **Init** ()

  Initialize the plugin.
- virtual void **Load** (**rendering::VisualPtr** _visual, sdf::ElementPtr _sdf)=0

  Load function.
- virtual void **Reset** ()

  Override this method for custom plugin reset behavior.

10.273.1 Detailed Description

A plugin loaded within the gzserver on startup.

See reference.

10.273.2 Constructor & Destructor Documentation

10.273.2.1 gazebo::VisualPlugin::VisualPlugin ( ) [inline]

References gazebo::VISUAL_PLUGIN.

10.273.3 Member Function Documentation

10.273.3.1 virtual void gazebo::VisualPlugin::Init ( ) [inline, virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.273.3.2 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.

10.273.3.3 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:

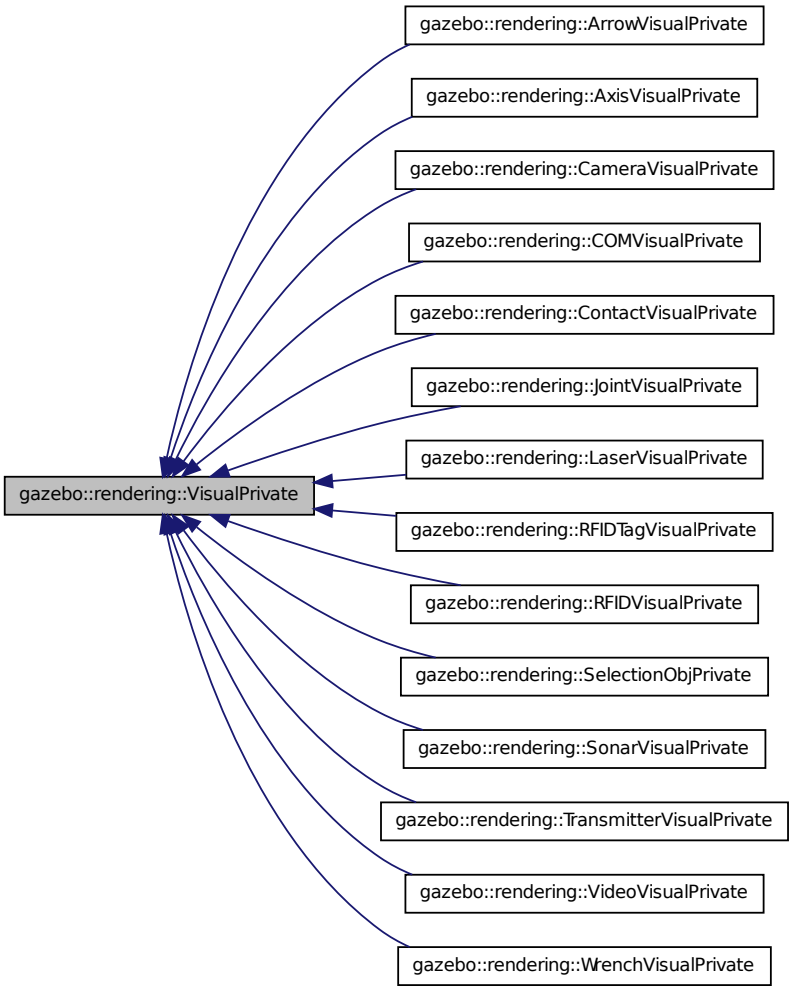
- **Plugin.hh**

10.274 gazebo::rendering::VisualPrivate Class Reference

Private data for the **Visual** (p. 1477) class.

#include <VisualPrivate.hh>

Inheritance diagram for gazebo::rendering::VisualPrivate:



Public Attributes

- Ogre::AnimationState * **animState**

  Used to animate the visual.
- **WireBox** * **boundingBox**

A wire frame bounding box.

- std::vector< **VisualPtr** > **children**

  Children visuals.

- uint32_t **id**

  Unique id of this visual.

- bool **initialized**

  True if initialized.

- bool **isStatic**

  *True if the visual is static, which allows **Ogre** (p. 163) to improve performance.*

- bool **lighting**

  True if lighting will be applied to this visual.

- std::list< **DynamicLines** * > **lines**

  List of all the lines created.

- std::list< std::pair < **DynamicLines** *, unsigned int > > **lineVertices**

  Lines and their vertices connected to this visual.

- std::string **myMaterialName**

  The unique name for the visual's material.

- std::string **name**

  Name of the visual.

- boost::function< void()> **onAnimationComplete**

  Callback for the animation complete event.

- std::string **origMaterialName**

  The original name for the visual's material.

- **VisualPtr parent**

  Parent visual.

- std::vector< **VisualPluginPtr** > **plugins**

  A list of visual plugins.

- **event::ConnectionPtr preRenderConnection**

  Connection for the pre render event.

- **common::Time prevAnimTime**

  Time of the previous animation step.

- Ogre::RibbonTrail * **ribbonTrail**

  The ribbon train created by the visual.

- **math::Vector3 scale**

  Scale of visual.

- **ScenePtr scene**

  Pointer to the visual's scene.

- Ogre::SceneNode * **sceneNode**

  *Pointer to the visual's scene node in **Ogre** (p. 163).*

- sdf::ElementPtr **sdf**

The SDF element for the visual.

- Ogre::SkeletonInstance * **skeleton**

  The visual's skeleton, used only for person simulation.

- Ogre::StaticGeometry * **staticGeom**

  Pointer to the static geometry.

- float **transparency**

  Transparency value.

- bool **useRTShader**

  True to use RT shader system.

- bool **visible**

  True if rendered.

Static Public Attributes

- static uint32_t **visualIdCount**

  Counter used to create unique ids.

10.274.1 Detailed Description

Private data for the **Visual** (p. 1477) class.

10.274.2 Member Data Documentation

10.274.2.1 Ogre::AnimationState* gazebo::rendering::VisualPrivate::animState

Used to animate the visual.

10.274.2.2 WireBox* gazebo::rendering::VisualPrivate::boundingBox

A wire frame bounding box.

10.274.2.3 std::vector<VisualPtr> gazebo::rendering::VisualPrivate::children

Children visuals.

10.274.2.4 uint32_t gazebo::rendering::VisualPrivate::id

Unique id of this visual.

10.274.2.5 bool gazebo::rendering::VisualPrivate::initialized

True if initialized.

10.274.2.6 bool gazebo::rendering::VisualPrivate::isStatic

True if the visual is static, which allows **Ogre** (p. 163) to improve performance.

10.274.2.7 bool gazebo::rendering::VisualPrivate::lighting

True if lighting will be applied to this visual.

10.274.2.8 std::list<DynamicLines*> gazebo::rendering::VisualPrivate::lines

List of all the lines created.

10.274.2.9 std::list< std::pair<DynamicLines*, unsigned int> > gazebo::rendering::VisualPrivate::lineVertices

Lines and their vertices connected to this visual.

10.274.2.10 std::string gazebo::rendering::VisualPrivate::myMaterialName

The unique name for the visual's material.

10.274.2.11 std::string gazebo::rendering::VisualPrivate::name

Name of the visual.

10.274.2.12 boost::function<void()> gazebo::rendering::VisualPrivate::onAnimationComplete

Callback for the animation complete event.

10.274.2.13 std::string gazebo::rendering::VisualPrivate::origMaterialName

The original name for the visual's material.

10.274.2.14 VisualPtr gazebo::rendering::VisualPrivate::parent

Parent visual.

10.274.2.15 std::vector<VisualPluginPtr> gazebo::rendering::VisualPrivate::plugins

A list of visual plugins.

10.274.2.16 event::ConnectionPtr gazebo::rendering::VisualPrivate::preRenderConnection

Connection for the pre render event.

10.274.2.17 common::Time gazebo::rendering::VisualPrivate::prevAnimTime

Time of the previous animation step.

10.274.2.18 Ogre::RibbonTrail* gazebo::rendering::VisualPrivate::ribbonTrail

The ribbon train created by the visual.

10.274.2.19 math::Vector3 gazebo::rendering::VisualPrivate::scale

Scale of visual.

10.274.2.20 ScenePtr gazebo::rendering::VisualPrivate::scene

Pointer to the visual's scene.

10.274.2.21 Ogre::SceneNode* gazebo::rendering::VisualPrivate::sceneNode

Pointer to the visual's scene node in **Ogre** (p. 163).

10.274.2.22 sdf::ElementPtr gazebo::rendering::VisualPrivate::sdf

The SDF element for the visual.

10.274.2.23 Ogre::SkeletonInstance* gazebo::rendering::VisualPrivate::skeleton

The visual's skeleton, used only for person simulation.

10.274.2.24 Ogre::StaticGeometry* gazebo::rendering::VisualPrivate::staticGeom

Pointer to the static geometry.

10.274.2.25 float gazebo::rendering::VisualPrivate::transparency

Transparency value.

10.274.2.26 bool gazebo::rendering::VisualPrivate::useRTShader

True to use RT shader system.

10.274.2.27 bool gazebo::rendering::VisualPrivate::visible

True if rendered.

10.274.2.28 uint32_t gazebo::rendering::VisualPrivate::visualIdCount [static]

Counter used to create unique ids.

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

- **VisualPrivate.hh**

10.275 gazebo::rendering::WindowManager Class Reference

Class to mangage render windows.

`#include <rendering/rendering.hh>`

Public Member Functions

- **WindowManager** ()

  Constructor.
- virtual ~**WindowManager** ()

Destructor.

- 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. 163) 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.

10.275.1 Detailed Description

Class to mangage render windows.

10.275.2 Constructor & Destructor Documentation

10.275.2.1 gazebo::rendering::WindowManager::WindowManager ( )

Constructor.

10.275.2.2 virtual gazebo::rendering::WindowManager::~WindowManager ( ) [virtual]

Destructor.

10.275.3 Member Function Documentation

10.275.3.1 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.

10.275.3.2 void gazebo::rendering::WindowManager::Fini ( )

Shutdown all the windows.

10.275.3.3 float gazebo::rendering::WindowManager::GetAvgFPS (uint32_t *_id*)

Get the average FPS.

Parameters

in	*_id*	ID of the window.

Returns

The frames per second.

10.275.3.4 uint32_t gazebo::rendering::WindowManager::GetTriangleCount (uint32_t *_id*)

Get the triangle count.

Parameters

in	*_id*	ID of the window.

Returns

The triangle count.

10.275.3.5 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.

10.275.3.6 void gazebo::rendering::WindowManager::Moved (uint32_t *_id*)

Tells **Ogre** (p. 163) the window has moved, and needs updating.

Parameters

in	*_id*	ID of the window.

10.275.3.7 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.

10.275.3.8 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**

10.276 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.
- **math::Box GetBox** () const

  Get the wireframe box.
- bool **GetVisible** () const

  Get the visibility of the box.
- void **Init** (const **math::Box** &_box)

  Builds the wireframe line list.
- void **SetVisible** (bool _visible)

  Set the visibility of the box.

10.276.1 Detailed Description

Draws a wireframe box.

10.276.2 Constructor & Destructor Documentation

10.276.2.1 gazebo::rendering::WireBox::WireBox (VisualPtr *_parent,* const math::Box & *_box*) [explicit]

Constructor.

Parameters

in	*_box*	Dimension of the box to draw.
in	*_parent*	Parent visual of the box.

10.276.2.2 gazebo::rendering::WireBox::~WireBox ( )

Destructor.

10.276.3 Member Function Documentation

10.276.3.1 math::Box gazebo::rendering::WireBox::GetBox ( ) const

Get the wireframe box.

Returns

The wireframe box.

10.276.3.2 bool gazebo::rendering::WireBox::GetVisible ( ) const

Get the visibility of the box.

Returns

True if the box is visual.

10.276.3.3 void gazebo::rendering::WireBox::Init (const math::Box & *_box*)

Builds the wireframe line list.

Parameters

in	*_box*	Box to build a wireframe from.

10.276.3.4 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**

10.277 gazebo::rendering::WireBoxPrivate Class Reference

Private data for the **WireBox** (p. 1515) class.

```
#include <WireBoxPrivate.hh>
```

Public Attributes

- **DynamicLines ∗ lines**

  The lines which outline the box.

- **VisualPtr parent**

  The visual which this box is attached to.

10.277.1 Detailed Description

Private data for the **WireBox** (p. 1515) class.

10.277.2 Member Data Documentation

10.277.2.1 DynamicLines∗ gazebo::rendering::WireBoxPrivate::lines

The lines which outline the box.

10.277.2.2 VisualPtr gazebo::rendering::WireBoxPrivate::parent

The visual which this box is attached to.

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

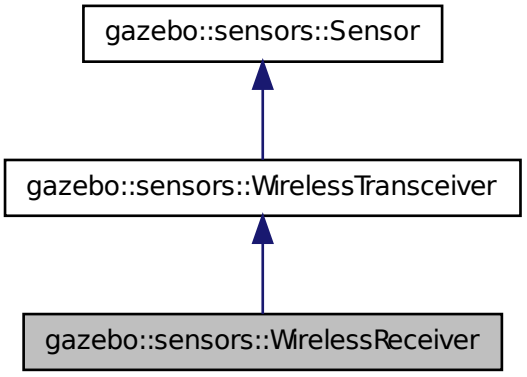
- **WireBoxPrivate.hh**

10.278 gazebo::sensors::WirelessReceiver Class Reference

Sensor (p. 1130) class for receiving wireless signals.

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

Inheritance diagram for gazebo::sensors::WirelessReceiver:



Public Member Functions

- **WirelessReceiver** ()

  Constructor.
- virtual ~**WirelessReceiver** ()

  Constructor.
- virtual void **Fini** ()

  Finalize the sensor.
- double **GetMaxFreqFiltered** () const

  Returns the maximum frequency filtered (MHz).
- double **GetMinFreqFiltered** () const

  Returns the minimum frequency filtered (MHz).
- double **GetSensitivity** () const

  Returns the receiver sensitivity (dBm).
- virtual void **Init** ()

  Initialize the sensor.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

10.278.1 Detailed Description

Sensor (p. 1130) class for receiving wireless signals.

10.278.2 Constructor & Destructor Documentation

10.278.2.1 gazebo::sensors::WirelessReceiver::WirelessReceiver ( )

Constructor.

10.278.2.2 virtual gazebo::sensors::WirelessReceiver::~WirelessReceiver ( ) [virtual]

Constructor.

10.278.3 Member Function Documentation

10.278.3.1 virtual void gazebo::sensors::WirelessReceiver::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1522).

10.278.3.2 double gazebo::sensors::WirelessReceiver::GetMaxFreqFiltered ( ) const

Returns the maximum frequency filtered (MHz).

Returns

Reception frequency (MHz).

10.278.3.3 double gazebo::sensors::WirelessReceiver::GetMinFreqFiltered ( ) const

Returns the minimum frequency filtered (MHz).

Returns

Reception frequency (MHz).

10.278.3.4 double gazebo::sensors::WirelessReceiver::GetSensitivity ( ) const

Returns the receiver sensitivity (dBm).

Returns

Receiver sensitivity (dBm).

10.278.3.5 virtual void gazebo::sensors::WirelessReceiver::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1523).

10.278.3.6 virtual void gazebo::sensors::WirelessReceiver::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::WirelessTransceiver** (p. 1523).

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

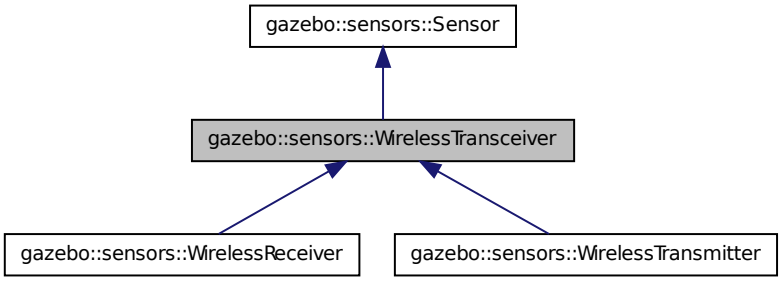
- **WirelessReceiver.hh**

10.279 gazebo::sensors::WirelessTransceiver Class Reference

Sensor (p. 1130) class for receiving wireless signals.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::WirelessTransceiver:



Public Member Functions

- **WirelessTransceiver** ()

  Constructor.
- ∼**WirelessTransceiver** ()

  Constructor.
- virtual void **Fini** ()

  Finalize the sensor.
- double **GetGain** () const

  Returns the antenna's gain of the receiver (dBi).
- double **GetPower** () const

  Returns the receiver power (dBm).
- virtual std::string **GetTopic** () const

  Returns the topic name as set in SDF.
- virtual void **Init** ()

  Initialize the sensor.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

Protected Attributes

- double **gain**

  Antenna's gain of the receiver (dBi).
- boost::weak_ptr< **physics::Link** > **parentEntity**

Parent entity which the sensor is attached to.

- double **power**

  Receiver's power (dBm).

- **transport::PublisherPtr pub**

  Publisher to publish propagation model data.

- **math::Pose referencePose**

  Sensor *(p. 1130) reference pose.*

10.279.1 Detailed Description

Sensor (p. 1130) class for receiving wireless signals.

10.279.2 Constructor & Destructor Documentation

10.279.2.1 gazebo::sensors::WirelessTransceiver::WirelessTransceiver ( )

Constructor.

10.279.2.2 gazebo::sensors::WirelessTransceiver::~WirelessTransceiver ( )

Constructor.

10.279.3 Member Function Documentation

10.279.3.1 virtual void gazebo::sensors::WirelessTransceiver::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1135).

Reimplemented in **gazebo::sensors::WirelessReceiver** (p. 1519).

10.279.3.2 double gazebo::sensors::WirelessTransceiver::GetGain ( ) const

Returns the antenna's gain of the receiver (dBi).

Returns

Antenna's gain of the receiver (dBi).

10.279.3.3 double gazebo::sensors::WirelessTransceiver::GetPower ( ) const

Returns the receiver power (dBm).

Returns

Receiver power (dBm).

10.279.3.4 virtual std::string gazebo::sensors::WirelessTransceiver::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

Returns

Topic name.

Reimplemented from **gazebo::sensors::Sensor** (p. 1138).

10.279.3.5 virtual void gazebo::sensors::WirelessTransceiver::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 1139).

Reimplemented in **gazebo::sensors::WirelessTransmitter** (p. 1527), and **gazebo::sensors::WirelessReceiver** (p. 1520).

10.279.3.6 virtual void gazebo::sensors::WirelessTransceiver::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. 1140).

Reimplemented in **gazebo::sensors::WirelessTransmitter** (p. 1527), and **gazebo::sensors::WirelessReceiver** (p. 1520).

10.279.4 Member Data Documentation

10.279.4.1 double gazebo::sensors::WirelessTransceiver::gain [protected]

Antenna's gain of the receiver (dBi).

10.279.4.2 boost::weak_ptr<physics::Link> gazebo::sensors::WirelessTransceiver::parentEntity [protected]

Parent entity which the sensor is attached to.

10.279.4.3 double gazebo::sensors::WirelessTransceiver::power [protected]

Receiver's power (dBm).

10.279.4.4 transport::PublisherPtr gazebo::sensors::WirelessTransceiver::pub [protected]

Publisher to publish propagation model data.

10.279.4.5 math::Pose gazebo::sensors::WirelessTransceiver::referencePose [protected]

Sensor (p. 1130) reference pose.

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

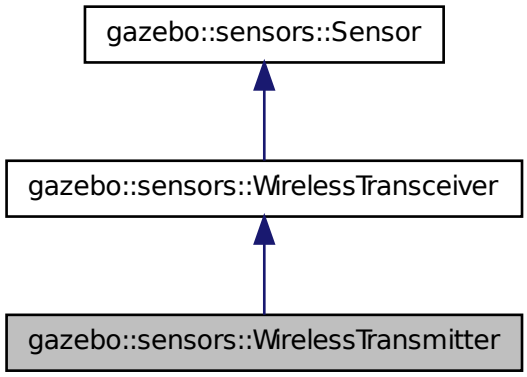
- **WirelessTransceiver.hh**

10.280 gazebo::sensors::WirelessTransmitter Class Reference

Transmitter to send wireless signals.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::WirelessTransmitter:



Public Member Functions

- **WirelessTransmitter** ()

  Constructor.
- virtual ~**WirelessTransmitter** ()

  Destructor.
- std::string **GetESSID** () const

  Returns the Service Set Identifier (network name).
- double **GetFreq** () const

  Returns reception frequency (MHz).
- double **GetSignalStrength** (const **math::Pose** &_receiver, const double rx-Gain)

  Returns the signal strength in a given world's point (dBm).
- virtual void **Init** ()

  Initialize the sensor.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

Static Public Attributes

- static const double **ModelStdDesv**

  Std desv of the Gaussian random variable used in the propagation model.

- static const double **NEmpty**

  Constant used in the propagation model when there are no obstacles between transmitter and receiver.

- static const double **NObstacle**

  Constant used in the propagation model when there are obstacles between transmitter and receiver.

Protected Member Functions

- virtual bool **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

Protected Attributes

- double **freq**

  Reception frequency (MHz).

10.280.1 Detailed Description

Transmitter to send wireless signals.

10.280.2 Constructor & Destructor Documentation

10.280.2.1 gazebo::sensors::WirelessTransmitter::WirelessTransmitter ( )

Constructor.

10.280.2.2 virtual gazebo::sensors::WirelessTransmitter::~WirelessTransmitter ( ) [virtual]

Destructor.

10.280.3 Member Function Documentation

10.280.3.1 std::string gazebo::sensors::WirelessTransmitter::GetESSID ( ) const

Returns the Service Set Identifier (network name).

Returns

Service Set Identifier (network name).

10.280.3.2 double gazebo::sensors::WirelessTransmitter::GetFreq ( ) const

Returns reception frequency (MHz).

Returns

Reception frequency (MHz).

10.280.3.3 double gazebo::sensors::WirelessTransmitter::GetSignalStrength (const math::Pose & *_receiver,* const double *rxGain*)

Returns the signal strength in a given world's point (dBm).

Returns

Signal strength in a world's point (dBm).

10.280.3.4 virtual void gazebo::sensors::WirelessTransmitter::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::WirelessTransceiver** (p. 1523).

10.280.3.5 virtual void gazebo::sensors::WirelessTransmitter::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::WirelessTransceiver** (p. 1523).

10.280.3.6 virtual bool gazebo::sensors::WirelessTransmitter::UpdateImpl (bool) [protected, virtual]

This gets overwritten by derived sensor types.

This function is called during **Sensor::Update** (p. 1142). And in turn, **Sensor::Update** (p. 1142) is called by **SensorManager::Update** (p. 1150)

Parameters

in	*_force*	True if update is forced, false if not

Returns

True if the sensor was updated.

Reimplemented from **gazebo::sensors::Sensor** (p. 1142).

10.280.4 Member Data Documentation

10.280.4.1 double gazebo::sensors::WirelessTransmitter::freq [protected]

Reception frequency (MHz).

10.280.4.2 const double gazebo::sensors::WirelessTransmitter::ModelStdDesv [static]

Std desv of the Gaussian random variable used in the propagation model.

10.280.4.3 const double gazebo::sensors::WirelessTransmitter::NEmpty [static]

Constant used in the propagation model when there are no obstacles between transmitter and receiver.

10.280.4.4 const double gazebo::sensors::WirelessTransmitter::NObstacle [static]

Constant used in the propagation model when there are obstacles between transmitter and receiver.

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

- **WirelessTransmitter.hh**

10.281 gazebo::physics::World Class Reference

The world provides access to all other object within a simulated environment.

`#include <physics/physics.hh>`

Public Member Functions

- **World** (const std::string &_name="")

  Constructor.
- ~**World** ()

  Destructor.
- void **Clear** ()

  Remove all entities from the world.
- void **ClearModels** ()

  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. 500) based on a name.*
- **EntityPtr GetEntityBelowPoint** (const **math::Vector3** &_pt)

  Get the nearest entity below a point.
- uint32_t **GetIterations** () const

  Get the total number of iterations.
- **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 and not encapsulating 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).
- bool **GetRunning** () const

  Return the running state of the world.
- msgs::Scene **GetSceneMsg** () const

  Get the current scene in message form.
- **EntityPtr GetSelectedEntity** () const

  *Get the selected **Entity** (p. 500).*
- 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. 1368).*
- **common::SphericalCoordinatesPtr GetSphericalCoordinates** () const

  Return the spherical coordinates converter.
- **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. 500) 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** (unsigned int _iterations=0)

  Run the world in a thread.
- void **RunBlocking** (unsigned int _iterations=0)

  Run the world. This call blocks. Run the update loop.
- 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 **Step** (unsigned int _steps)

  Step the world forward in time.
- 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. 512) on the **physics::Link** (p. 739) in World::Update.*

10.281.1 Detailed Description

The world provides access to all other object within a simulated environment.

The **World** (p. 1529) is the container for all models and their components (links, joints, sensors, plugins, etc), and **WorldPlugin** (p. 1544) instances. Many core function are also handled in the **World** (p. 1529), including physics update, model updates, and message processing.

10.281.2 Constructor & Destructor Documentation

10.281.2.1 gazebo::physics::World::World (const std::string & *_name* = " ") [explicit]

Constructor.

Constructor for the **World** (p. 1529). Must specify a unique name.

Parameters

in	*_name*	Name of the world.

10.281.2.2 gazebo::physics::World::~World ( )

Destructor.

10.281.3 Member Function Documentation

10.281.3.1 void gazebo::physics::World::Clear ( )

Remove all entities from the world.

This function has delayed effect. Models are cleared at the end of the current update iteration.

10.281.3.2 void gazebo::physics::World::ClearModels ( )

Remove all entities from the world.

Implementation of **World::Clear** (p. 1532)

10.281.3.3 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.

10.281.3.4 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.

10.281.3.5 void gazebo::physics::World::EnablePhysicsEngine (bool *_enable*) [inline]

enable/disable physics engine during World::Update.

Parameters

in	*_enable*	True to enable the physics engine.

10.281.3.6 void gazebo::physics::World::Fini ( )

Finalize the world.

Call this function to tear-down the world.

10.281.3.7 BasePtr gazebo::physics::World::GetByName (const std::string & *_name*)

Get an element by name.

Searches the list of entities, and return a pointer to the model with a matching _name.

Parameters

in	*_name*	The name of the **Model** (p. 846) to find.

Returns

A pointer to the entity, or NULL if no entity was found.

10.281.3.8 bool gazebo::physics::World::GetEnablePhysicsEngine ( ) [inline]

check if physics engine is enabled/disabled.

Parameters

True	if the physics engine is enabled.

10.281.3.9 EntityPtr gazebo::physics::World::GetEntity (const std::string & *_name*)

Get a pointer to an **Entity** (p. 500) 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. 500) to find.

Returns

A pointer to the **Entity** (p. 500), or NULL if no **Entity** (p. 500) was found.

10.281.3.10 EntityPtr gazebo::physics::World::GetEntityBelowPoint (const math::Vector3 & *_pt*)

Get the nearest entity below a point.

Projects a Ray down (-Z axis) starting at the given point. The first entity hit by the Ray is returned.

Parameters

in	*_pt*	The 3D point to search below

Returns

A pointer to nearest **Entity** (p. 500), NULL if none is found.

10.281.3.11 uint32_t gazebo::physics::World::GetIterations ( ) const

Get the total number of iterations.

Returns

Number of iterations that simulation has taken.

10.281.3.12 ModelPtr gazebo::physics::World::GetModel (unsigned int *_index*) const

Get a model based on an index.

Get a **Model** (p. 846) using an index, where index must be greater than zero and less than **World::GetModelCount()** (p. 1536)

Parameters

in	*_index*	The index of the model [0..GetModelCount)

Returns

A pointer to the **Model** (p. 846). NULL if _index is invalid.

10.281.3.13 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. 846) to find.

Returns

A pointer to the **Model** (p. 846), or NULL if no model was found.

10.281.3.14 ModelPtr gazebo::physics::World::GetModelBelowPoint (const math::Vector3 & *_pt*)

Get the nearest model below and not encapsulating a point.

Only objects below the start point can be returned. Any object that encapsulates the

start point can not be returned from this function. This function makes use of **World::GetEntityBelowPoint** (p. 1534).

Parameters

in	*_pt*	The 3D point to search below.

Returns

A pointer to nearest **Model** (p. 846), NULL if none is found.

10.281.3.15 unsigned int gazebo::physics::World::GetModelCount ( ) const

Get the number of models.

Returns

The number of models in the **World** (p. 1529).

10.281.3.16 Model_V gazebo::physics::World::GetModels ( ) const

Get a list of all the models.

Returns

A list of all the Models in the world.

10.281.3.17 std::string gazebo::physics::World::GetName ( ) const

Get the name of the world.

Returns

The name of the world.

10.281.3.18 common::Time gazebo::physics::World::GetPauseTime ( ) const

Get the amount of time simulation has been paused.

Returns

The pause time.

10.281.3.19 PhysicsEnginePtr gazebo::physics::World::GetPhysicsEngine ( ) const

Return the physics engine.

Get a pointer to the physics engine used by the world.

Returns

Pointer to the physics engine.

10.281.3.20 common::Time gazebo::physics::World::GetRealTime ( ) const

Get the real time (elapsed time).

Returns

The real time.

10.281.3.21 bool gazebo::physics::World::GetRunning ( ) const

Return the running state of the world.

Returns

True if the world is running.

10.281.3.22 msgs::Scene gazebo::physics::World::GetSceneMsg ( ) const

Get the current scene in message form.

Returns

The scene state as a protobuf message.

10.281.3.23 EntityPtr gazebo::physics::World::GetSelectedEntity ( ) const

Get the selected **Entity** (p. 500).

The selected entity is set via the GUI.

Returns

A point to the **Entity** (p. 500), NULL if nothing is selected.

10.281.3.24 boost::mutex* gazebo::physics::World::GetSetWorldPoseMutex ( ) const [inline]

Get the set world pose mutex.

Returns

Pointer to the mutex.

10.281.3.25 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. 1368).

Returns

The current simulation time

10.281.3.26 common::SphericalCoordinatesPtr gazebo::physics::World::GetSphericalCoordinates ( ) const

Return the spherical coordinates converter.

Returns

Pointer to the spherical coordinates converter.

10.281.3.27 common::Time gazebo::physics::World::GetStartTime ( ) const

Get the wall time simulation was started.

Returns

The start time.

10.281.3.28 void gazebo::physics::World::Init ( )

Initialize the world.

This is called after Load.

10.281.3.29 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).

10.281.3.30 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 reference to an SDF object.

10.281.3.31 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 string containing valid SDF markup.

10.281.3.32 bool gazebo::physics::World::IsLoaded ( ) const

Return true if the world has been loaded.

Returns

True if **World::Load** (p. 1540) has completed.

10.281.3.33 bool gazebo::physics::World::IsPaused ( ) const

Returns the state of the simulation true if paused.

Returns

True if paused.

10.281.3.34 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.

10.281.3.35 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 unique name for the plugin.
in	*_sdf*	The SDF to pass into the plugin.

10.281.3.36 void gazebo::physics::World::PrintEntityTree ( )

Print **Entity** (p. 500) tree.

Prints alls the entities to stdout.

10.281.3.37 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.

10.281.3.38 void gazebo::physics::World::RemovePlugin (const std::string & _name)

Remove a running plugin.

Parameters

in	*_name*	The unique name of the plugin to remove.

10.281.3.39 void gazebo::physics::World::Reset ( )

Reset time and model poses, configurations in simulation.

10.281.3.40 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. 205).

Parameters

in	*_type*	The type of reset.

10.281.3.41 void gazebo::physics::World::ResetTime ( )

Reset simulation time back to zero.

10.281.3.42 void gazebo::physics::World::Run (unsigned int _iterations = 0)

Run the world in a thread.

Run the update loop.

Parameters

in	*_iterations*	Run for this many iterations, then stop. A value of zero disables run stop.

10.281.3.43 void gazebo::physics::World::RunBlocking (unsigned int _iterations = 0)

Run the world. This call blocks. Run the update loop.

Todo In gazebo 3.0 this should be move to the proper section.

Parameters

in	*_iterations*	Run for this many iterations, then stop. A value of zero disables run stop.

10.281.3.44 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.

10.281.3.45 void gazebo::physics::World::SetPaused (bool *_p*)

Set whether the simulation is paused.

Parameters

in	*_p*	True pauses the simulation. False runs the simulation.

10.281.3.46 void gazebo::physics::World::SetSimTime (const common::Time & *_t*)

Set the sim time.

Parameters

in	*_t*	The new simulation time

10.281.3.47 void gazebo::physics::World::SetState (const WorldState & *_state*)

Set the current world state.

Parameters

_state	The state to set the **World** (p. 1529) to.

10.281.3.48 void gazebo::physics::World::Step (unsigned int _steps)

Step the world forward in time.

Parameters

in	*_steps*	The number of steps the **World** (p. 1529) should take.

10.281.3.49 void gazebo::physics::World::Stop ( )

Stop the world.

Stop the update loop.

10.281.3.50 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.

10.281.3.51 void gazebo::physics::World::UpdateStateSDF ( )

Update the state SDF value from the current state.

10.281.4 Member Data Documentation

10.281.4.1 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. 512) on the **physics::Link** (p. 739) in World::Update.

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

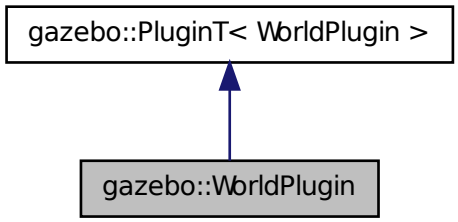
- **World.hh**

10.282 gazebo::WorldPlugin Class Reference

A plugin with access to **physics::World** (p. 1529).

`#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** ()

10.282.1 Detailed Description

A plugin with access to **physics::World** (p. 1529).

See `reference`.

10.282.2 Constructor & Destructor Documentation

10.282.2.1 gazebo::WorldPlugin::WorldPlugin ( ) [inline]

Constructor.

References gazebo::WORLD_PLUGIN.

10.282.2.2 virtual gazebo::WorldPlugin::~WorldPlugin ( ) [inline, virtual]

Destructor.

10.282.3 Member Function Documentation

10.282.3.1 virtual void gazebo::WorldPlugin::Init ( ) [inline, virtual]

10.282.3.2 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.

10.282.3.3 virtual void gazebo::WorldPlugin::Reset ( ) [inline, virtual]

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

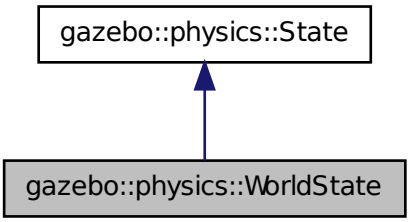
- **Plugin.hh**

10.283 gazebo::physics::WorldState Class Reference

Store state information of a **physics::World** (p. 1529) 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** (const std::string &_modelName) const

  Get a model state by model name.
- unsigned int **GetModelStateCount** () const

  Get the number of model states.
- **ModelState_M GetModelStates** (const boost::regex &_regex) const

  Get model states based on a regular expression.
- const **ModelState_M** & **GetModelStates** () const

  Get the model states.
- bool **HasModelState** (const std::string &_modelName) const

  *Return true if **WorldState** (p. 1545) has a **ModelState** (p. 869) with the given name.*
- bool **IsZero** () const

  Return true if the values in the state are zero.

- void **Load** (const **WorldPtr** _world)

  *Load from a **World** (p. 1529) pointer.*
- 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.
- virtual void **SetRealTime** (const **common::Time** &_time)

  Set the real time when this state was generated.
- virtual void **SetSimTime** (const **common::Time** &_time)

  Set the sim time when this state was generated.
- virtual void **SetWallTime** (const **common::Time** &_time)

  Set the wall time when this state was generated.
- void **SetWorld** (const **WorldPtr** _world)

  Set the world.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::-WorldState** &_state)

  Stream insertion operator.

10.283.1 Detailed Description

Store state information of a **physics::World** (p. 1529) object.

Instances of this class contain the state of a **World** (p. 1529) at a specific time. **World** (p. 1529) state includes the state of all models, and their children.

10.283.2 Constructor & Destructor Documentation

10.283.2.1 gazebo::physics::WorldState::WorldState ( )

Default constructor.

10.283.2.2 gazebo::physics::WorldState::WorldState (const WorldPtr _world) [explicit]

Constructor.

Generate a **WorldState** (p. 1545) from an instance of a **World** (p. 1529).

Parameters

in	*_world*	Pointer to a world

10.283.2.3 gazebo::physics::WorldState::WorldState (const sdf::ElementPtr _sdf) [explicit]

Constructor.

Build a **WorldState** (p. 1545) from SDF data

Parameters

in	*_sdf*	SDF data to load a world state from.

10.283.2.4 virtual gazebo::physics::WorldState::~WorldState ( ) [virtual]

Destructor.

10.283.3 Member Function Documentation

10.283.3.1 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.

10.283.3.2 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. 548)*	When the _modelName doesn't exist.

10.283.3.3 unsigned int gazebo::physics::WorldState::GetModelStateCount ( ) const

Get the number of model states.

Returns the number of models in this instance.

Returns

Number of models.

10.283.3.4 ModelState_M gazebo::physics::WorldState::GetModelStates (const boost::regex & *_regex*) const

Get model states based on a regular expression.

Parameters

in	*_regex*	The regular expression.

Returns

List of model states whose names match the regular expression.

10.283.3.5 const ModelState_M& gazebo::physics::WorldState::GetModelStates ( ) const

Get the model states.

Returns

A vector of model states.

10.283.3.6 bool gazebo::physics::WorldState::HasModelState (const std::string & _modelName) const

Return true if **WorldState** (p. 1545) has a **ModelState** (p. 869) with the given name.

Parameters

in	*_modelName*	Name of the model to search for.

Returns

True if the **ModelState** (p. 869) exists.

10.283.3.7 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.

10.283.3.8 void gazebo::physics::WorldState::Load (const WorldPtr _world)

Load from a **World** (p. 1529) pointer.

Generate a **WorldState** (p. 1545) from an instance of a **World** (p. 1529).

Parameters

in	*_world*	Pointer to a world

10.283.3.9 virtual void gazebo::physics::WorldState::Load (const sdf::ElementPtr _elem) [virtual]

Load state from SDF element.

Set a **WorldState** (p. 1545) from an SDF element containing **WorldState** (p. 1545) info.

Parameters

in	*_elem*	Pointer to the **WorldState** (p. 1545) SDF element.

Reimplemented from **gazebo::physics::State** (p. 1326).

10.283.3.10 WorldState gazebo::physics::WorldState::operator+ (const WorldState & _state) const

Addition operator.

Parameters

in	*_pt*	A state to add.

Returns

The resulting state.

10.283.3.11 WorldState gazebo::physics::WorldState::operator- (const WorldState & _state) const

Subtraction operator.

Parameters

in	*_pt*	A state to substract.

Returns

The resulting state.

10.283.3.12 WorldState& gazebo::physics::WorldState::operator= (const WorldState & _state)

Assignment operator.

Parameters

in	*_state*	**State** (p. 1323) value

Returns

Reference to this

10.283.3.13 virtual void gazebo::physics::WorldState::SetRealTime (const common::Time & _time) [virtual]

Set the real time when this state was generated.

Parameters

in	*_time*	Clock time since simulation was stated.

Reimplemented from **gazebo::physics::State** (p. 1327).

10.283.3.14 virtual void gazebo::physics::WorldState::SetSimTime (const common::Time & _time) [virtual]

Set the sim time when this state was generated.

Parameters

in	*_time*	Simulation time when the data was recorded.

Reimplemented from **gazebo::physics::State** (p. 1327).

10.283.3.15 virtual void gazebo::physics::WorldState::SetWallTime (const common::Time & _time) [virtual]

Set the wall time when this state was generated.

Parameters

in	*_time*	The absolute clock time when the **State** (p. 1323) data was recorded.

Reimplemented from **gazebo::physics::State** (p. 1328).

10.283.3.16 void gazebo::physics::WorldState::SetWorld (const WorldPtr _world)

Set the world.

Parameters

in	*_world*	Pointer to the world.

10.283.4 Friends And Related Function Documentation

10.283.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::physics::WorldState & *_state*) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_state*	**World** (p. 1529) state to output

Returns

the stream

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

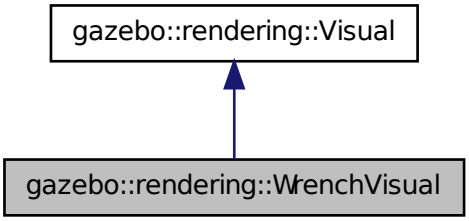
- **WorldState.hh**

10.284 gazebo::rendering::WrenchVisual Class Reference

Visualization for sonar data.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::WrenchVisual:



Public Member Functions

- **WrenchVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_-topicName)

  Constructor.
- virtual ~**WrenchVisual** ()

  Destructor.
- void **Load** (ConstJointPtr &_msg)

  Load the visual based on a message.
- void **SetEnabled** (bool _enabled)

  Set to true to enable wrench visualization.

10.284.1 Detailed Description

Visualization for sonar data.

10.284.2 Constructor & Destructor Documentation

10.284.2.1 gazebo::rendering::WrenchVisual::WrenchVisual (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. 1477).
in	*_topicName*	Name of the topic that has sonar data.

10.284.2.2 virtual gazebo::rendering::WrenchVisual::~WrenchVisual ( ) [virtual]

Destructor.

10.284.3 Member Function Documentation

10.284.3.1 void gazebo::rendering::WrenchVisual::Load (ConstJointPtr & *_msg*)

Load the visual based on a message.

Parameters

in	*_msg*	Joint message

10.284.3.2 void gazebo::rendering::WrenchVisual::SetEnabled (bool *_enabled*)

Set to true to enable wrench visualization.

Parameters

in	*_enabled*	True to show wrenches, false to hide.

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

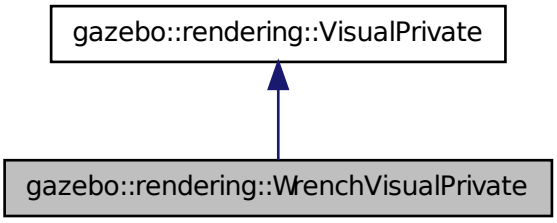
- **WrenchVisual.hh**

10.285 gazebo::rendering::WrenchVisualPrivate Class Reference

Private data for the Wrench **Visual** (p. 1477) class.

`#include <WrenchVisualPrivate.hh>`

Inheritance diagram for gazebo::rendering::WrenchVisualPrivate:



Public Attributes

- Ogre::SceneNode * **coneXNode**

  Scene *(p. 1097) node for X torque visualization.*

- Ogre::SceneNode ∗ **coneYNode**

  Scene *(p. 1097) node for Y torque visualization.*
- Ogre::SceneNode ∗ **coneZNode**

  Scene *(p. 1097) node for Z torque visualization.*
- std::vector< **event::ConnectionPtr** > **connections**

  All the event connections.
- bool **enabled**

  True if this visualization is enabled.
- **DynamicLines** ∗ **forceLine**

  Line to visualize force.
- Ogre::SceneNode ∗ **forceNode**

  Scene *(p. 1097) node for force visualization.*
- boost::mutex **mutex**

  Mutex to protect the contact message.
- **transport::NodePtr node**

  Pointer to a node that handles communication.
- bool **receivedMsg**

  True if we have received a message.
- boost::shared_ptr < msgs::WrenchStamped const > **wrenchMsg**

  The current wrench message.
- **transport::SubscriberPtr wrenchSub**

  Subscription to the sonar data.

10.285.1 Detailed Description

Private data for the Wrench **Visual** (p. 1477) class.

10.285.2 Member Data Documentation

10.285.2.1 Ogre::SceneNode∗ gazebo::rendering::WrenchVisualPrivate::coneXNode

Scene (p. 1097) node for X torque visualization.

10.285.2.2 Ogre::SceneNode∗ gazebo::rendering::WrenchVisualPrivate::coneYNode

Scene (p. 1097) node for Y torque visualization.

10.285.2.3 Ogre::SceneNode∗ gazebo::rendering::WrenchVisualPrivate::coneZNode

Scene (p. 1097) node for Z torque visualization.

10.285.2.4 std::vector<event::ConnectionPtr> gazebo::rendering::WrenchVisualPrivate::connections

All the event connections.

10.285.2.5 bool gazebo::rendering::WrenchVisualPrivate::enabled

True if this visualization is enabled.

10.285.2.6 DynamicLines* gazebo::rendering::WrenchVisualPrivate::forceLine

Line to visualize force.

10.285.2.7 Ogre::SceneNode* gazebo::rendering::WrenchVisualPrivate::forceNode

Scene (p. 1097) node for force visualization.

10.285.2.8 boost::mutex gazebo::rendering::WrenchVisualPrivate::mutex

Mutex to protect the contact message.

10.285.2.9 transport::NodePtr gazebo::rendering::WrenchVisualPrivate::node

Pointer to a node that handles communication.

10.285.2.10 bool gazebo::rendering::WrenchVisualPrivate::receivedMsg

True if we have received a message.

10.285.2.11 boost::shared_ptr<msgs::WrenchStamped const> gazebo::rendering::WrenchVisualPrivate::wrenchMsg

The current wrench message.

10.285.2.12 transport::SubscriberPtr gazebo::rendering::WrenchVisualPrivate::wrenchSub

Subscription to the sonar data.

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

- **WrenchVisualPrivate.hh**

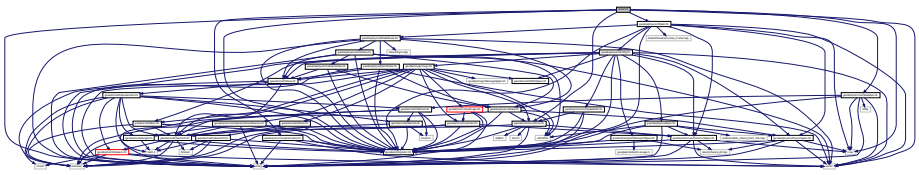
Chapter 11

File Documentation

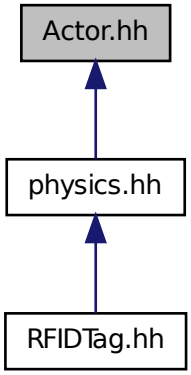
11.1 Actor.hh File Reference

```
#include <string>   #include <map>   #include <vector> ×
#include "gazebo/physics/Model.hh" #include "gazebo/common/-
Time.hh"   #include "gazebo/common/Skeleton.hh"   #include
"gazebo/common/Animation.hh" #include "gazebo/util/system.-
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. *165) class enables GPU based mesh model / skeleton scriptable animation.*

- class **gazebo::physics::TrajectoryInfo**

  Information about a trajectory for an ***Actor*** *(*p. *165).*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::common**

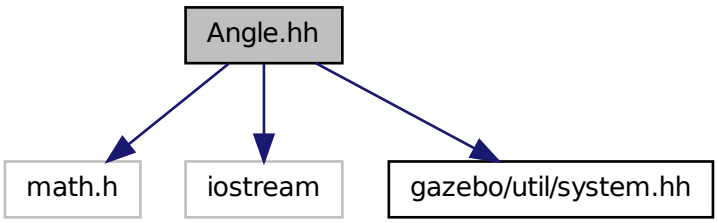
  Common namespace.

- namespace **gazebo::physics**

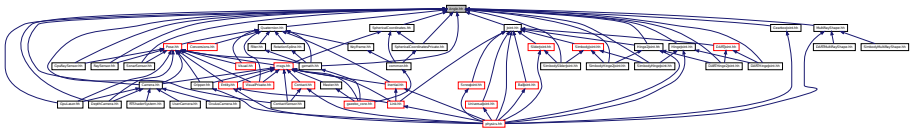
  namespace for physics

11.2 Angle.hh File Reference

`#include <math.h>` `#include <iostream>` `#include "gazebo/util/system.hh"` 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.

Defines

- #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 Define 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

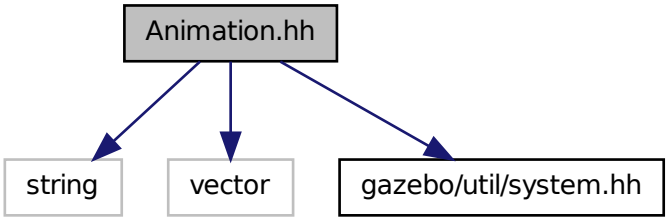
in	*radians*	

Returns

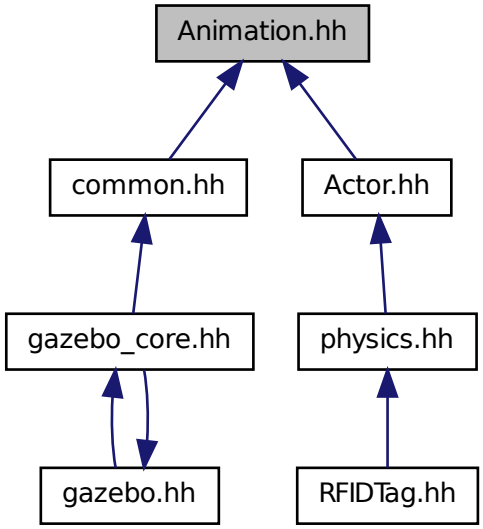
degrees

11.3 Animation.hh File Reference

`#include <string>` `#include <vector>` `#include "gazebo/util/system.-hh"` Include dependency graph for Animation.hh:



This graph shows which files directly or indirectly include this file:



Classes

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

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

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

  A pose animation.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

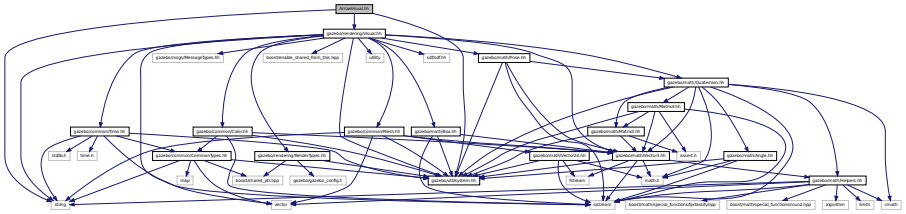
  Common namespace.

- namespace **gazebo::math**

  Math namespace.

11.4 ArrowVisual.hh File Reference

`#include <string>  #include "gazebo/rendering/Visual.hh"  #include "gazebo/util/system.hh"` Include dependency graph for ArrowVisual.hh:



Classes

- class **gazebo::rendering::ArrowVisual**

  Basic arrow visualization.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

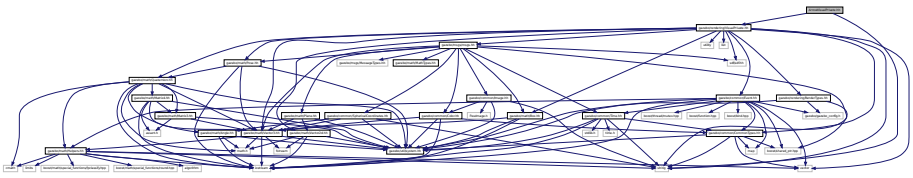
- namespace **gazebo::rendering**

  Rendering namespace.

11.5 ArrowVisualPrivate.hh File Reference

`#include <string> #include "gazebo/rendering/VisualPrivate.-`

hh" Include dependency graph for ArrowVisualPrivate.hh:



Classes

- class **gazebo::rendering::ArrowVisualPrivate**

  *Private data for the Arrow **Visual*** (p. *1477*) *class.*

Namespaces

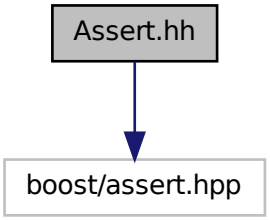
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

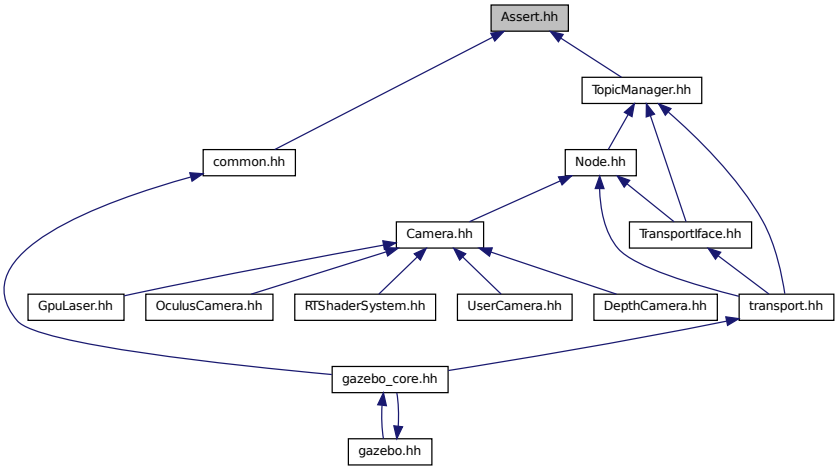
  Rendering namespace.
- namespace **ogre**

11.6 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:



Defines

- #define **GZ_ASSERT**(_expr, _msg) BOOST_ASSERT_MSG(_expr, _msg)

  This macro define the standard way of launching an exception inside gazebo.

11.6.1 Define Documentation

11.6.1.1 #define GZ_ASSERT(*_expr, _msg*) BOOST_ASSERT_MSG(_expr, _msg)

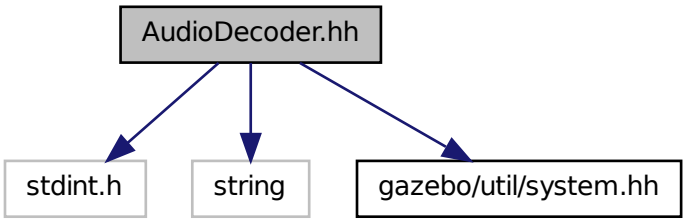
This macro define the standard way of launching an exception inside gazebo.

Referenced by gazebo::transport::TopicManager::Advertise().

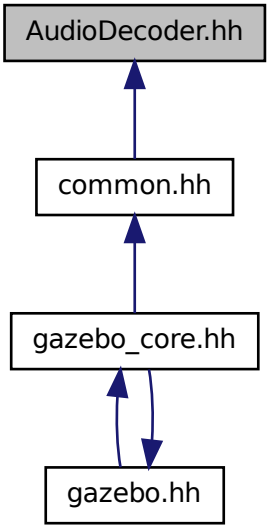
11.7 AudioDecoder.hh File Reference

```
#include <stdint.h> #include <string> #include "gazebo/util/system.-
```

hh" Include dependency graph for AudioDecoder.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::AudioDecoder**

  An audio decoder based on FFMPEG.

Namespaces

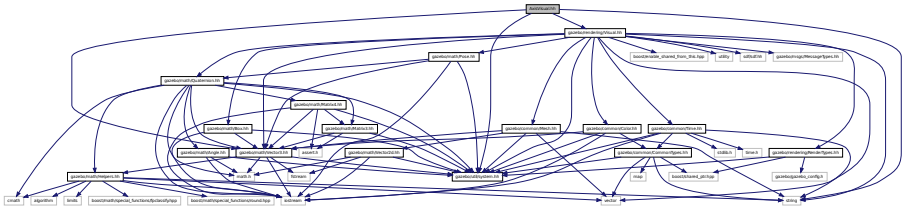
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.8 AxisVisual.hh File Reference

`#include <string>` `#include "gazebo/math/Vector3.hh"` × `#include "gazebo/rendering/Visual.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for AxisVisual.hh:



Classes

- class **gazebo::rendering::AxisVisual**

  Basic axis visualization.

Namespaces

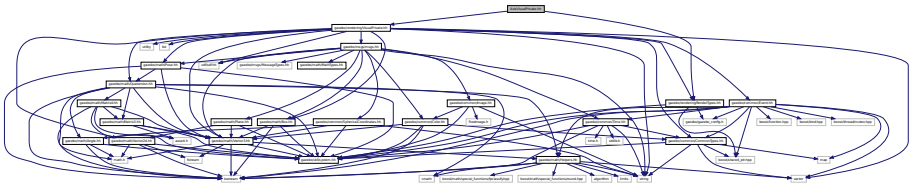
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.9 AxisVisualPrivate.hh File Reference

`#include "gazebo/rendering/RenderTypes.hh"` `#include "gazebo/rendering/VisualPrivate.hh"` Include dependency graph for AxisVisualPrivate.hh:



Classes

- class **gazebo::rendering::AxisVisualPrivate**

  *Private data for the Axis **Visual*** (p. *1477*) *class.*

Namespaces

- namespace **gazebo**

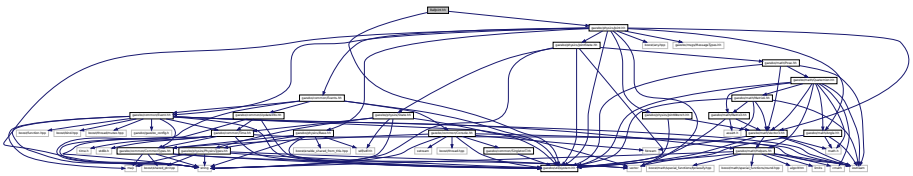
  Forward declarations for the common classes.

- namespace **gazebo::rendering**

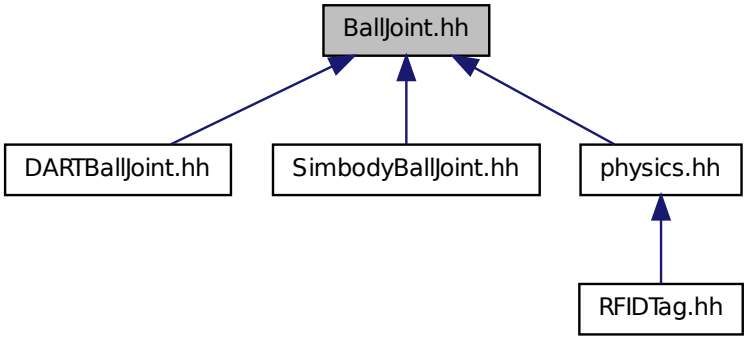
  Rendering namespace.

11.10 BallJoint.hh File Reference

`#include "gazebo/physics/Joint.hh"` `#include "gazebo/util/system.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. *201*) *class for a ball joint.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

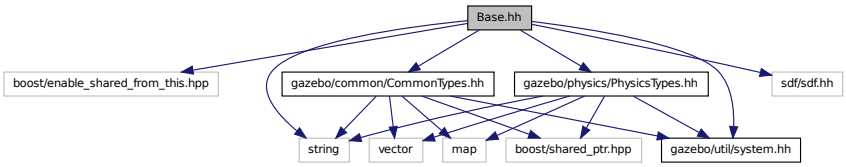
- namespace **gazebo::physics**

  namespace for physics

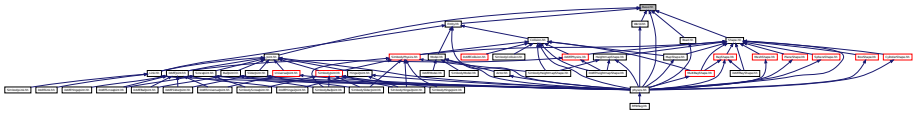
11.11 Base.hh File Reference

`#include <boost/enable_shared_from_this.hpp>` `#include <string>` `#include <sdf/sdf.hh>` `#include "gazebo/common/CommonTypes.hh"` `#include "gazebo/physics/PhysicsTypes.hh"` `#include "gazebo/util/system.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. *201) class for most physics classes.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

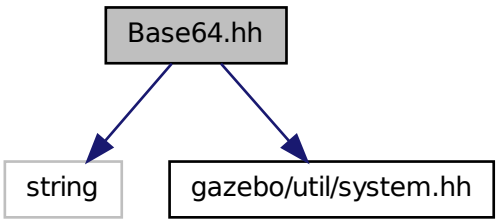
Variables

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

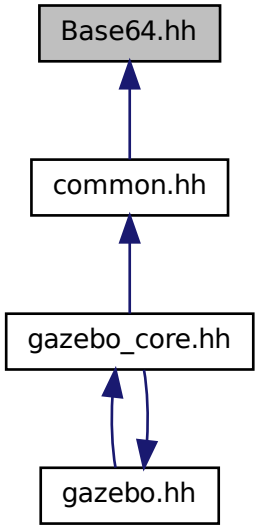
  String names for the different entity types.

11.12 Base64.hh File Reference

`#include <string>` `#include "gazebo/util/system.hh"` Include dependency graph for Base64.hh:



This graph shows which files directly or indirectly include this file:



Functions

- **GAZEBO_VISIBLE** std::string **Base64Decode** (const std::string &_encoded-String)

  Decode a base64 string.
- **GAZEBO_VISIBLE** void **Base64Encode** (const char *_bytesToEncode, unsigned int _len, std::string &_result)

  Encode a binary string into base 64.

11.12.1 Function Documentation

11.12.1.1 GAZEBO_VISIBLE std::string Base64Decode (const std::string & *_encodedString*)

Decode a base64 string.

Parameters

in	*_encoded-String*	A base 64 encoded string.

Returns

The decoded string.

11.12.1.2 GAZEBO_VISIBLE void Base64Encode (const char * *_bytesToEncode*, unsigned int *_len*, std::string & *_result*)

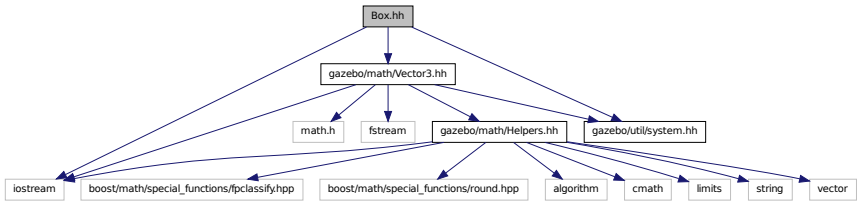
Encode a binary string into base 64.

Parameters

in	*_bytesTo-Encode*	String of bytes to encode.
in	*_len*	Length of _bytesToEncode.
out	*_result*	Based64 string is appended to this string.

11.13 Box.hh File Reference

`#include <iostream>` `#include "gazebo/math/Vector3.hh"` × `#include "gazebo/util/system.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**

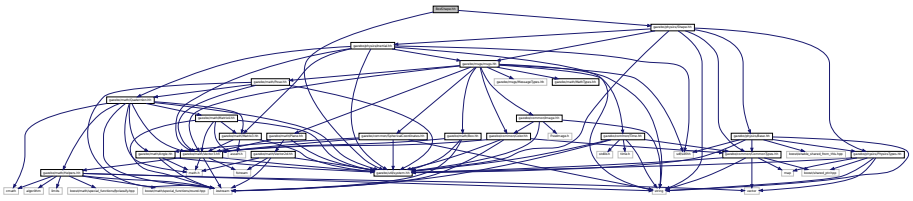
  Forward declarations for the common classes.

- namespace **gazebo::math**

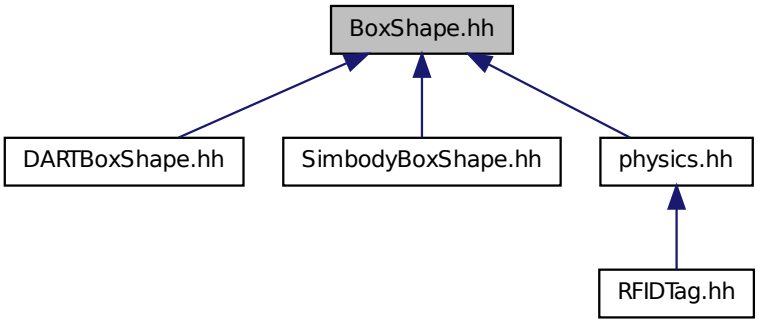
  Math namespace.

11.14 BoxShape.hh File Reference

`#include "gazebo/physics/Shape.hh"` `#include "gazebo/util/system.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.

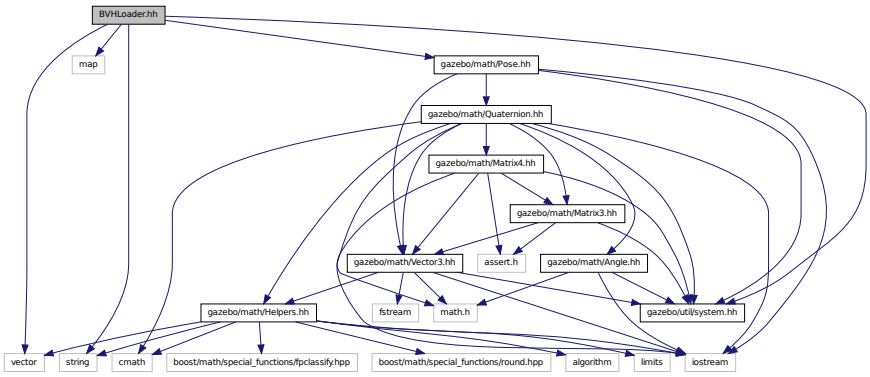
- namespace **gazebo::physics**

  namespace for physics

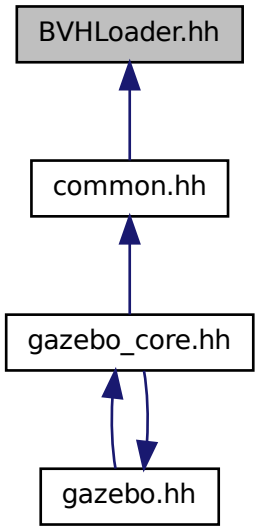
11.15 BVHLoader.hh File Reference

```
#include <vector>   #include <map>   #include <string>×
#include "gazebo/math/Pose.hh"#include "gazebo/util/system.-
```

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.

Defines

- #define **X_POSITION** 0
- #define **X_ROTATION** 3
- #define **Y_POSITION** 1
- #define **Y_ROTATION** 4
- #define **Z_POSITION** 2
- #define **Z_ROTATION** 5

11.15.1 Define Documentation

11.15.1.1 #define X_POSITION 0

11.15.1.2 #define X_ROTATION 3

11.15.1.3 #define Y_POSITION 1

11.15.1.4 #define Y_ROTATION 4

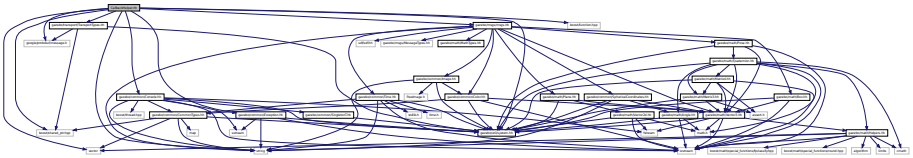
11.15.1.5 #define Z_POSITION 2

11.15.1.6 #define Z_ROTATION 5

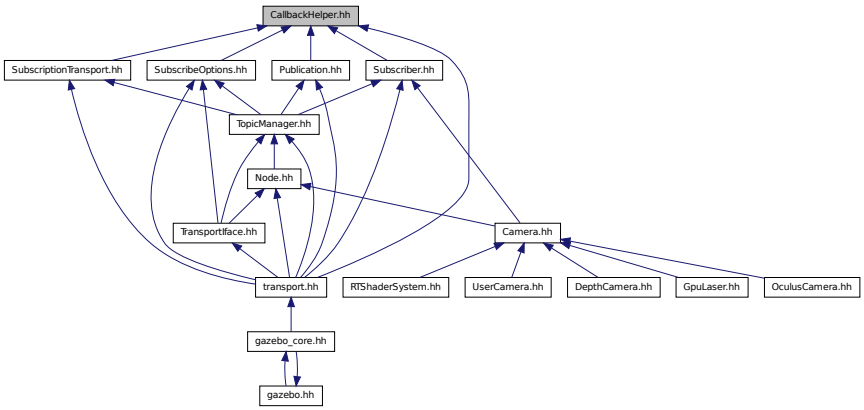
11.16 CallbackHelper.hh File Reference

```
#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 "gazebo/util/system.hh"` Include dependency graph for CallbackHelper.hh:



This graph shows which files directly or indirectly include this file:



Classes

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

  A helper class to handle callbacks when messages arrive.

- class **gazebo::transport::CallbackHelperT< M >**

  Callback helper Template.

- class **gazebo::transport::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

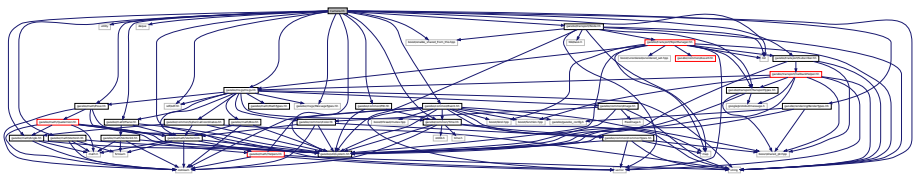
- typedef boost::shared_ptr < CallbackHelper > **gazebo::transport::CallbackHelperPtr**

  *boost shared pointer to **transport::CallbackHelper** (p. 235)*

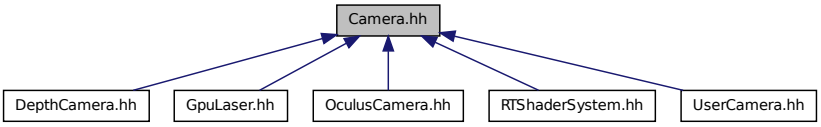
11.17 Camera.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>     #include
<string>  #include <utility>  #include <list>  #include
<vector> #include <deque> #include <sdf/sdf.hh> #include
"gazebo/msgs/msgs.hh"   #include "gazebo/transport/Node.-
hh"   #include "gazebo/transport/Subscriber.hh"   #include
"gazebo/common/Event.hh"   #include "gazebo/common/PID.hh"
#include "gazebo/common/Time.hh"   #include "gazebo/math/-
Angle.hh" #include "gazebo/math/Pose.hh" #include "gazebo/math/-
Plane.hh"   #include "gazebo/math/Vector2i.hh"   #include
"gazebo/msgs/MessageTypes.hh" #include "gazebo/rendering/-
RenderTypes.hh" #include "gazebo/util/system.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

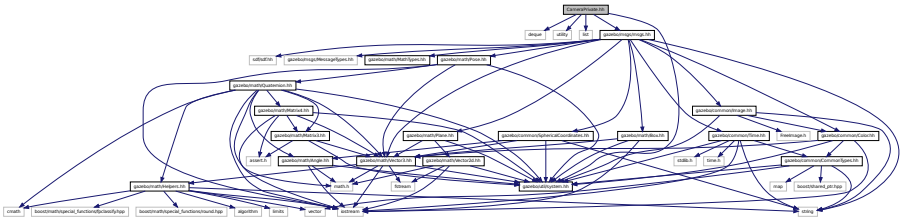
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.18 CameraPrivate.hh File Reference

`#include <deque>  #include <utility>  #include <list>` × `#include "gazebo/msgs/msgs.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for CameraPrivate.hh:



Classes

- class **gazebo::rendering::CameraPrivate**

  *Private data for the **Camera** (p. 242) class.*

Namespaces

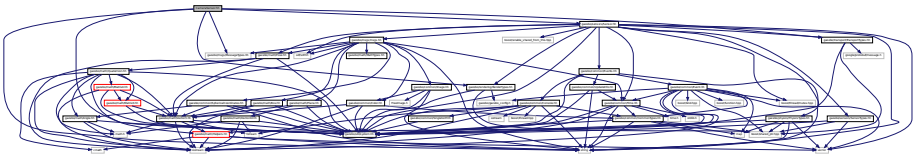
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.19 CameraSensor.hh File Reference

`#include <string>` `#include "gazebo/sensors/Sensor.hh"` `#include "gazebo/msgs/MessageTypes.hh"` `#include "gazebo/transport/TransportTypes.hh"` `#include "gazebo/rendering/RenderTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for CameraSensor.hh:



Classes

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

  Basic camera sensor.

Namespaces

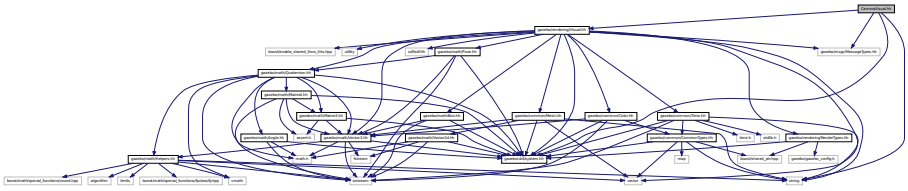
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.20 CameraVisual.hh File Reference

`#include <string> #include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh" #include "gazebo/util/system.-
hh"` Include dependency graph for CameraVisual.hh:



Classes

- class **gazebo::rendering::CameraVisual**

  Basic camera visualization.

Namespaces

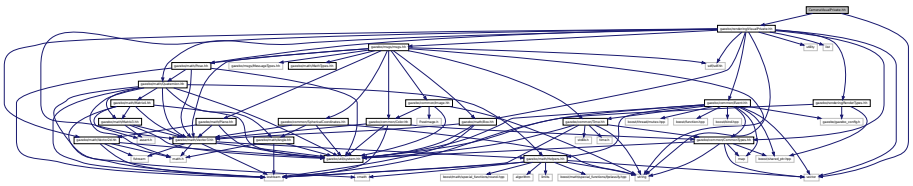
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.21 CameraVisualPrivate.hh File Reference

`#include <vector> #include "gazebo/rendering/VisualPrivate.-
hh"` Include dependency graph for CameraVisualPrivate.hh:



Classes

- class **gazebo::rendering::CameraVisualPrivate**

Namespaces

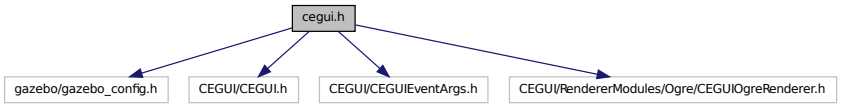
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

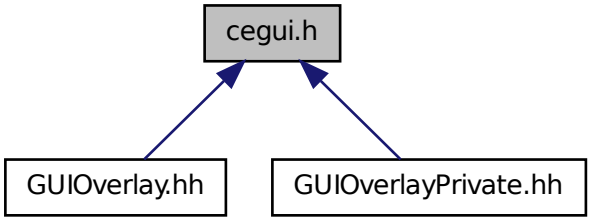
  Rendering namespace.

11.22 cegui.h File Reference

`#include "gazebo/gazebo_config.h"  #include <CEGUI/CEGUI.h> #include <CEGUI/CEGUIEventArgs.h> #include <CEGUI/RendererModules/Ogre/CEGUIOgreRenderer.h>` Include dependency graph for cegui.h:

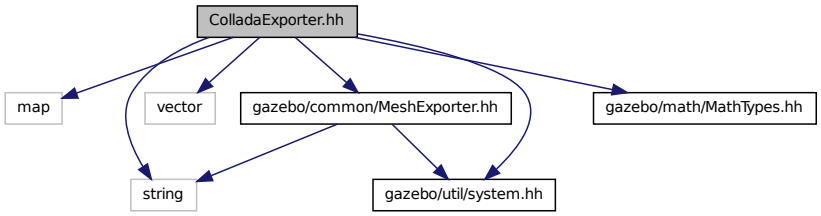


This graph shows which files directly or indirectly include this file:



11.23 ColladaExporter.hh File Reference

`#include <map>  #include <string>  #include <vector>` × `#include "gazebo/common/MeshExporter.hh"` `#include "gazebo/math/-MathTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for ColladaExporter.hh:



Classes

- class **gazebo::common::ColladaExporter**

  Class used to export Collada mesh files.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

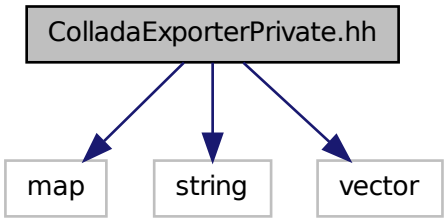
- namespace **gazebo::common**

  Common namespace.

11.24 ColladaExporterPrivate.hh File Reference

`#include <map>` `#include <string>` `#include <vector>` Include

dependency graph for ColladaExporterPrivate.hh:



Classes

- class **gazebo::common::ColladaExporterPrivate**

  *Private data for the **ColladaExporter*** (p. 287) *class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

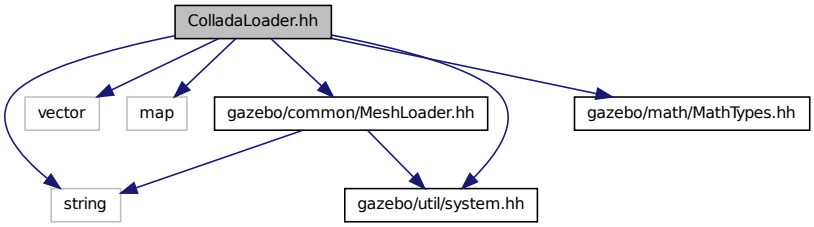
- namespace **gazebo::common**

  Common namespace.

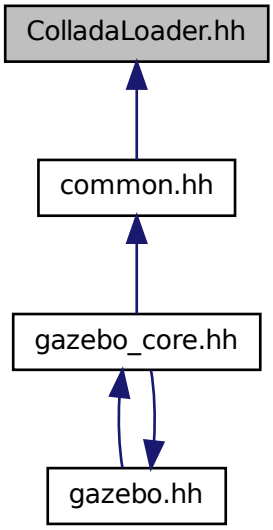
11.25 ColladaLoader.hh File Reference

`#include <string>  #include <vector>  #include <map>` ×
`#include "gazebo/common/MeshLoader.hh"` `#include "gazebo/math/-MathTypes.hh"` `#include "gazebo/util/system.hh"` Include depen-

dency 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

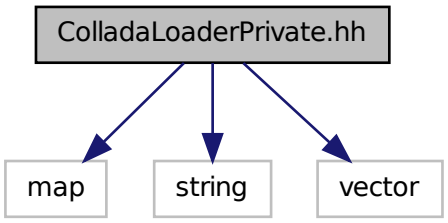
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.26 ColladaLoaderPrivate.hh File Reference

`#include <map>` `#include <string>` `#include <vector>` Include dependency graph for ColladaLoaderPrivate.hh:



Classes

- class **gazebo::common::ColladaLoaderPrivate**

  *Private data for the **ColladaLoader** (p. 291) class.*
- class **gazebo::common::GeometryIndices**

  Helper data structure for loading collada geometries.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

11.27 Collision.hh File Reference

`#include <string> #include <vector> #include "gazebo/common/-Event.hh" #include "gazebo/common/CommonTypes.hh" #include "gazebo/physics/PhysicsTypes.hh" #include "gazebo/physics/-CollisionState.hh" #include "gazebo/physics/Entity.hh" × #include "gazebo/util/system.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. *201) class for all collision entities.*

Namespaces

- namespace **gazebo**

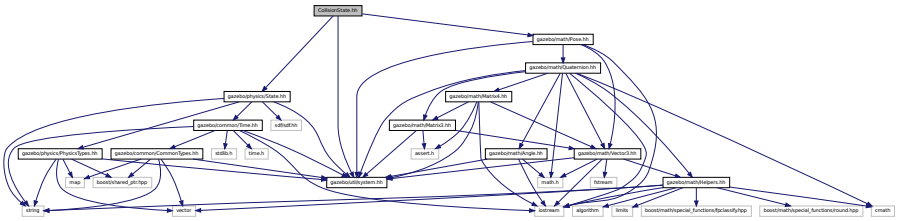
  Forward declarations for the common classes.

- namespace **gazebo::physics**

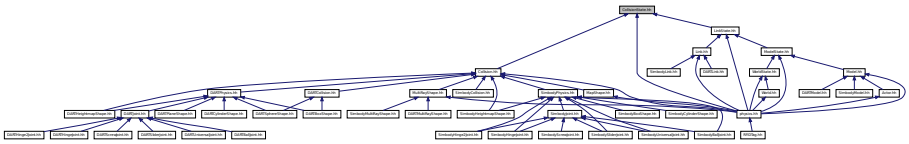
  namespace for physics

11.28 CollisionState.hh File Reference

`#include "gazebo/physics/State.hh" #include "gazebo/math/-Pose.hh" #include "gazebo/util/system.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. 295) object.*

Namespaces

- namespace **gazebo**

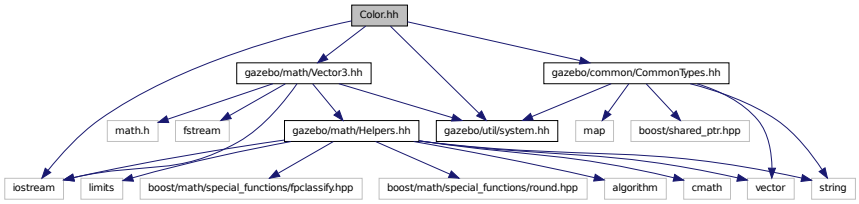
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.29 Color.hh File Reference

`#include <iostream> #include "gazebo/common/CommonTypes.-`

hh" #include "gazebo/math/Vector3.hh" #include "gazebo/util/system.-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.

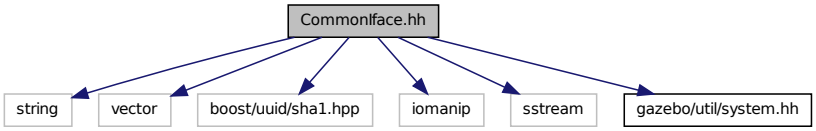
- namespace **gazebo::common**

  Common namespace.

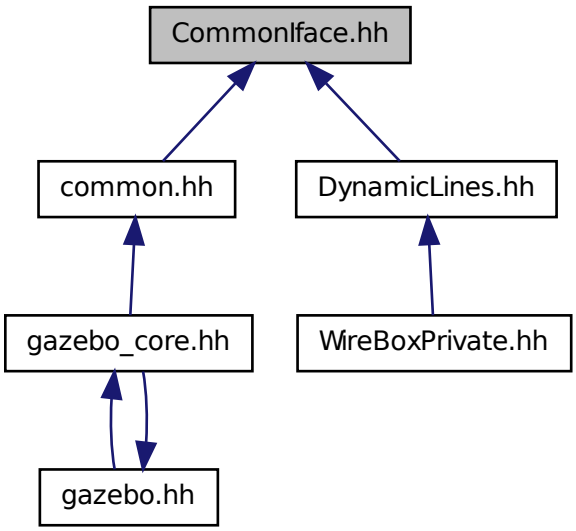
11.30 CommonIface.hh File Reference

#include <string> #include <vector> #include <boost/uuid/sha1.-hpp> #include <iomanip> #include <sstream> #include "gazebo/util/system.-

hh" Include dependency graph for CommonIface.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

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

  *add path sufix to **common::SystemPaths** (p. 1352)*
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file** (const std::string &_file)

  *search for file in **common::SystemPaths** (p. 1352)*
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath)

  *search for file in **common::SystemPaths** (p. 1352)*
- **GAZEBO_VISIBLE** std::string **gazebo::common::find_file_path** (const std::string &_file)

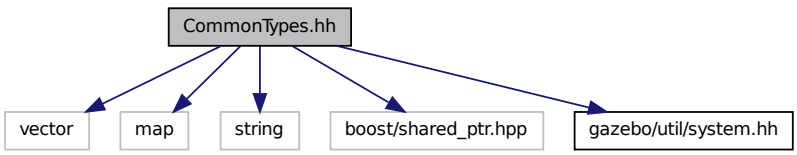
  *search for a file in **common::SystemPaths** (p. 1352)*
- template<typename T >
  GAZEBO_VISIBLE std::string **gazebo::common::get_sha1** (const T &_buffer)

  Compute the SHA1 hash of an array of bytes.
- **GAZEBO_VISIBLE** void **gazebo::common::load** ()

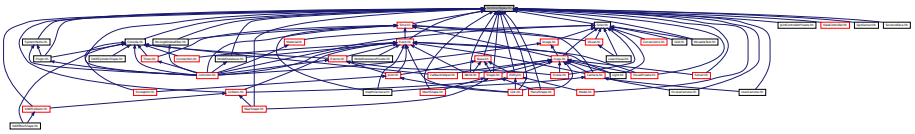
  Load the common library.

11.31 CommonTypes.hh File Reference

`#include <vector>  #include <map>  #include <string>` × `#include <boost/shared_ptr.hpp>` `#include "gazebo/util/system.hh"` Include dependency graph for CommonTypes.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.
- namespace **gazebo::event**

  Event *(p. 514) namespace.*

Defines

- #define **GAZEBO_DEPRECATED**(version) ()
- #define **GAZEBO_FORCEINLINE**
- #define **NULL** 0

Typedefs

- typedef boost::shared_ptr < Animation > **gazebo::common::AnimationPtr**
- typedef std::vector < ConnectionPtr > **gazebo::event::Connection_V**
- typedef boost::shared_ptr < Connection > **gazebo::event::ConnectionPtr**
- typedef boost::shared_ptr < DiagnosticTimer > **gazebo::common::DiagnosticTimerPtr**
- typedef boost::shared_ptr < GUIPlugin > **gazebo::GUIPluginPtr**
- typedef boost::shared_ptr < ModelPlugin > **gazebo::ModelPluginPtr**
- typedef boost::shared_ptr < NumericAnimation > **gazebo::common::NumericAnimationPtr**
- typedef std::vector < common::Param * > **gazebo::common::Param_V**
- typedef boost::shared_ptr < PoseAnimation > **gazebo::common::PoseAnimationPtr**
- typedef boost::shared_ptr < SensorPlugin > **gazebo::SensorPluginPtr**
- typedef boost::shared_ptr < SphericalCoordinates > **gazebo::common::SphericalCoordinatesPtr**
- typedef std::map< std::string, std::string > **gazebo::common::StrStr_M**

- typedef boost::shared_ptr < SystemPlugin > **gazebo::SystemPluginPtr**
- typedef boost::shared_ptr < VisualPlugin > **gazebo::VisualPluginPtr**
- typedef boost::shared_ptr < WorldPlugin > **gazebo::WorldPluginPtr**

Variables

- static const double **gazebo::common::SpeedOfLight** = 299792458

  Speed of light.

11.31.1 Detailed Description

11.31.2 Define Documentation

11.31.2.1 #define GAZEBO_DEPRECATED(*version*) ()

11.31.2.2 #define GAZEBO_FORCEINLINE

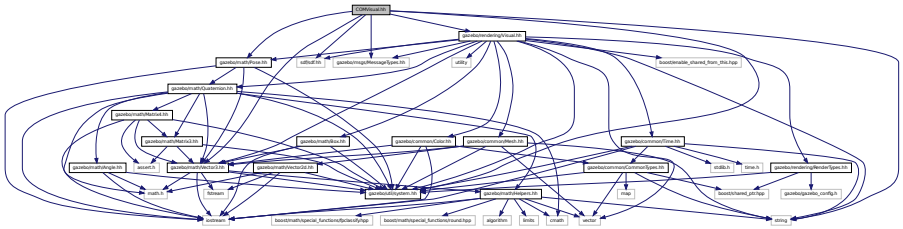
11.31.2.3 #define NULL 0

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::event::EventT< T >::Disconnect(), gazebo::transport::PublishTask::execute(), gazebo::transport::ConnectionReadTask::execute(), gazebo::common::get_sha1(), gazebo::transport::CallbackHelperT< M >::GetMsgType(), gazebo::transport::SubscribeOptions::Init(), gazebo::PluginT< ModelPlugin >::PluginT(), gazebo::physics::DARTSphereShape::SetRadius(), gazebo::physics::DARTCylinderShape::SetSize(), gazebo::physics::DARTBoxShape::SetSize(), and gazebo::common::MovingWindowFilter< T >::~MovingWindowFilter().

11.32 COMVisual.hh File Reference

`#include <string>` `#include <sdf/sdf.hh>` `#include "gazebo/math/Pose.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/msgs/MessageTypes.hh"` `#include "gazebo/rendering/Visual.hh"` × `#include "gazebo/util/system.hh"` Include dependency graph for COM-

Visual.hh:



Classes

- class **gazebo::rendering::COMVisual**

  Basic Center of Mass visualization.

Namespaces

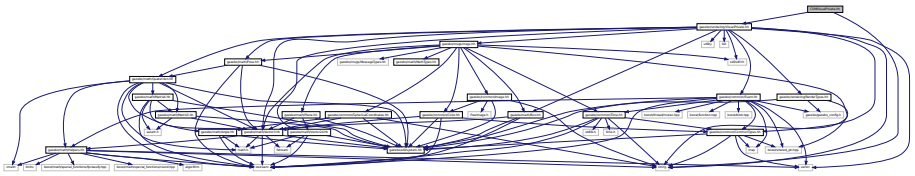
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.33 COMVisualPrivate.hh File Reference

`#include <string>` `#include "gazebo/rendering/VisualPrivate.-hh"` Include dependency graph for COMVisualPrivate.hh:



Classes

- class **gazebo::rendering::COMVisualPrivate**

  *Private data for the COM **Visual** (p. 1477) class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

  Rendering namespace.

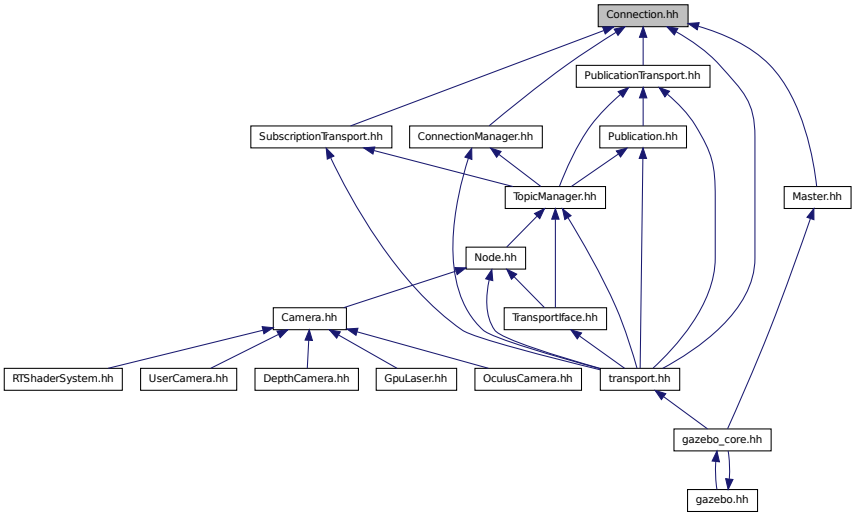
- namespace **ogre**

11.34 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 <utility> #include "gazebo/common/-
Event.hh"   #include "gazebo/common/Console.hh"   #include
"gazebo/common/Exception.hh" #include "gazebo/util/system.-
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**

Defines

- #define **HEADER_LENGTH** 8

Typedefs

- typedef boost::shared_ptr < Connection > **gazebo::transport::ConnectionPtr**

Functions

- bool **gazebo::transport::is_stopped** ()

  Is the transport system stopped?

11.34.1 Define Documentation

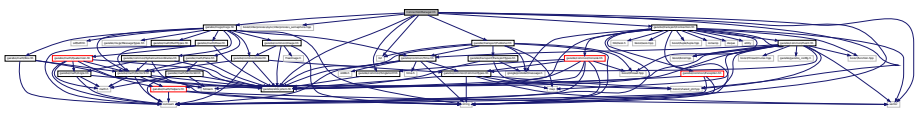
11.34.1.1 #define HEADER_LENGTH 8

Referenced by gazebo::transport::Connection::AsyncRead().

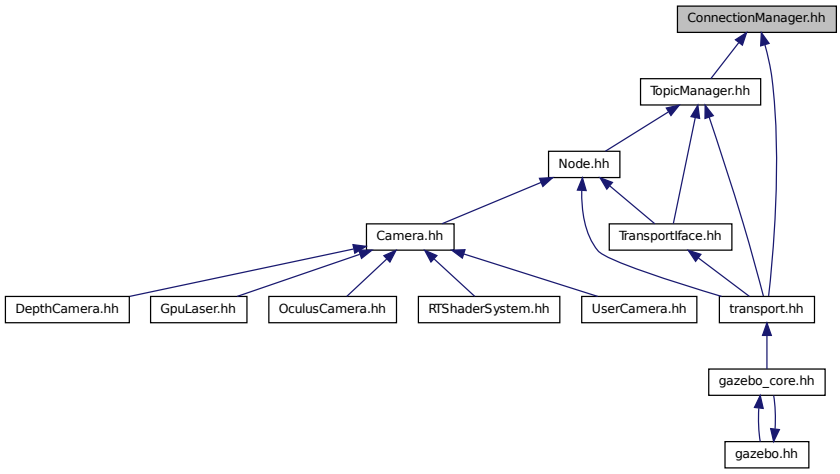
11.35 ConnectionManager.hh File Reference

```
#include <boost/shared_ptr.hpp> #include <boost/interprocess/sync/interproce
_semaphore.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
"gazebo/util/system.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**

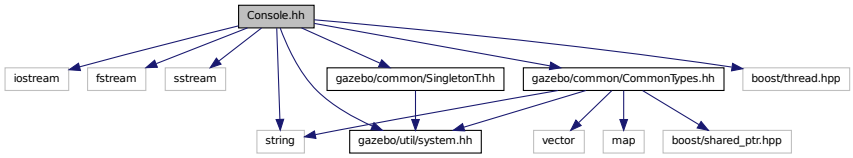
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.36 Console.hh File Reference

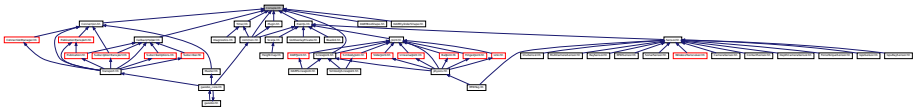
```
#include <iostream> #include <fstream> #include <sstream> ×
#include <string>  #include <boost/thread.hpp>  #include
"gazebo/common/SingletonT.hh"   #include "gazebo/common/-
CommonTypes.hh" #include "gazebo/util/system.hh"
```

Include de-

pendency graph for Console.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Logger::Buffer**

  String buffer for the base logger.
- class **gazebo::common::FileLogger::Buffer**

  String buffer for the file logger.
- class **gazebo::common::Console**

  Container for loggers, and global logging options (such as verbose vs.
- class **gazebo::common::FileLogger**

  A logger that outputs messages to a file.
- class **gazebo::common::Logger**

  Terminal logger.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Defines

- #define **gzdbg** (**gazebo::common::Console::dbg**(__FILE__, __LINE__))

  Output a debug message.

- #define **gzerr** (**gazebo::common::Console::err**(__FILE__, __LINE__))

  Output an error message.

- #define **gzlog** (**gazebo::common::Console::log**())

  Output a message to a log file.

- #define **gzLogInit**(_str) (gazebo::common::Console::log.Init(_str))

  Initialize log file with filename given by _str.

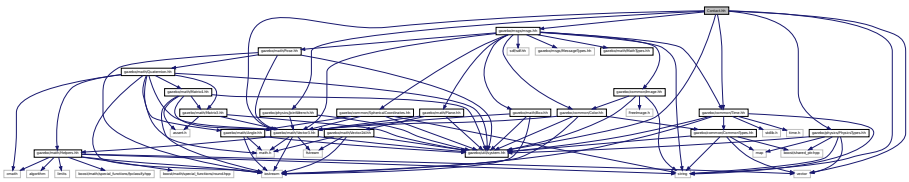
- #define **gzmsg** (**gazebo::common::Console::msg**())

- #define **gzwarn** (**gazebo::common::Console::warn**(__FILE__, __LINE__))

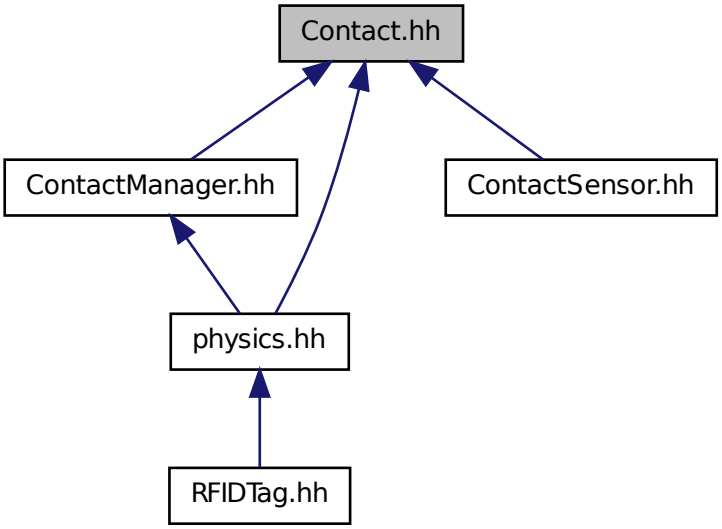
  Output a warning message.

11.37 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 "gazebo/util/system.hh"` Include dependency graph for Contact.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Contact**

  A contact between two collisions.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Defines

- #define **MAX_COLLIDE_RETURNS** 250
- #define **MAX_CONTACT_JOINTS** 32

11.37.1 Define Documentation

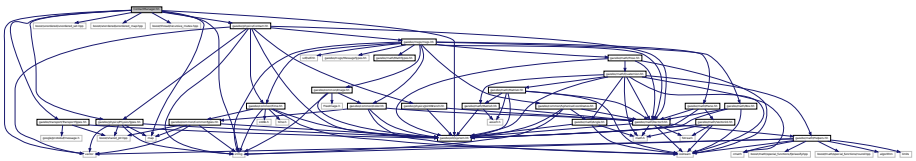
11.37.1.1 #define MAX_COLLIDE_RETURNS 250

11.37.1.2 #define MAX_CONTACT_JOINTS 32

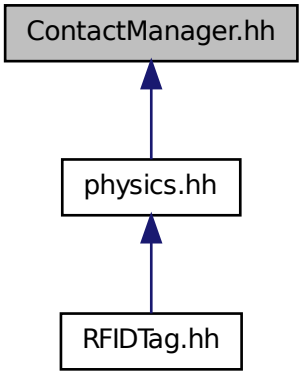
11.38 ContactManager.hh File Reference

```
#include <vector>   #include <string>   #include <map> ×
#include <boost/unordered/unordered_set.hpp>     #include
<boost/unordered/unordered_map.hpp> #include <boost/thread/recursive-
_mutex.hpp>   #include "gazebo/transport/TransportTypes.-
hh"   #include "gazebo/physics/PhysicsTypes.hh"   #include
"gazebo/physics/Contact.hh" #include "gazebo/util/system.-
hh"
```

Include dependency graph for ContactManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ContactManager**

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

  A custom contact publisher created for each contact filter in the ***Contact*** (p. 347) *- Manager.*

Namespaces

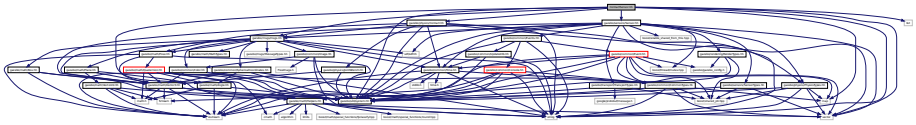
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.39 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 "gazebo/util/system.-hh"` Include dependency graph for ContactSensor.hh:



Classes

- class **gazebo::sensors::ContactSensor**

  Contact sensor.

Namespaces

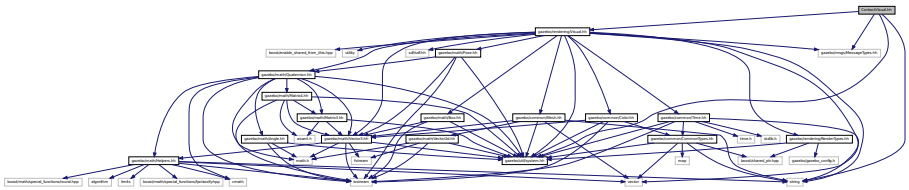
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.40 ContactVisual.hh File Reference

`#include <string> #include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh" #include "gazebo/util/system.-
hh"` Include dependency graph for ContactVisual.hh:



Classes

- class **gazebo::rendering::ContactVisual**

  Contact visualization.

Namespaces

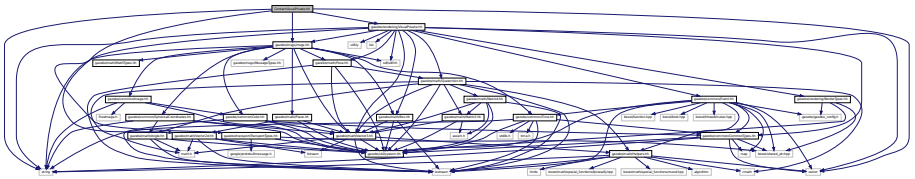
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.41 ContactVisualPrivate.hh File Reference

`#include <string> #include <vector> #include "gazebo/msgs/msgs.-
hh" #include "gazebo/transport/TransportTypes.hh" #include
"gazebo/rendering/VisualPrivate.hh"` Include dependency graph for ContactVisualPrivate.hh:



Classes

- class **gazebo::rendering::ContactVisualPrivate::ContactPoint**

  A contact point visualization.

- class **gazebo::rendering::ContactVisualPrivate**

  *Private data for the Arrow **Visual*** (p. *1477) class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

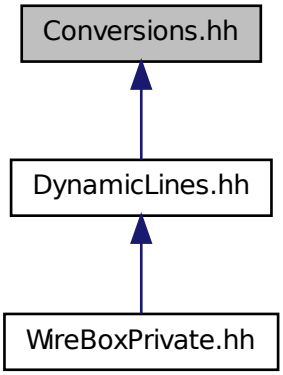
  Rendering namespace.

- namespace **Ogre**

11.42 Conversions.hh File Reference

`#include "gazebo/rendering/ogre_gazebo.h"` `#include "gazebo/common/-Color.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/math/-Quaternion.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for Conversions.hh:

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Conversions**

  Conversions *(p. 367)* ***Conversions.hh*** *(p. 1608)* ***rendering/Conversions.hh*** *(p. 1608).*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

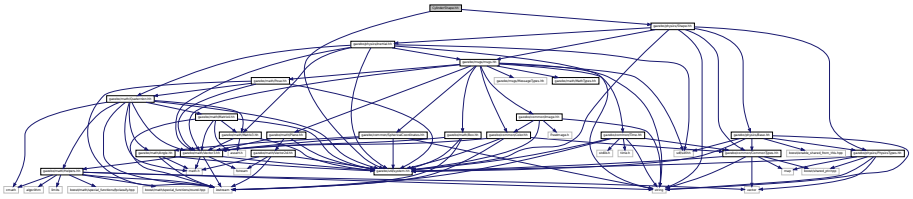
- namespace **gazebo::rendering**

  Rendering namespace.

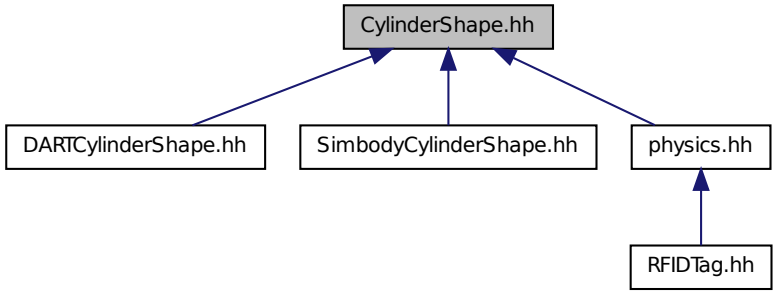
11.43 CylinderShape.hh File Reference

`#include "gazebo/physics/Shape.hh" #include "gazebo/util/system.-`

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**

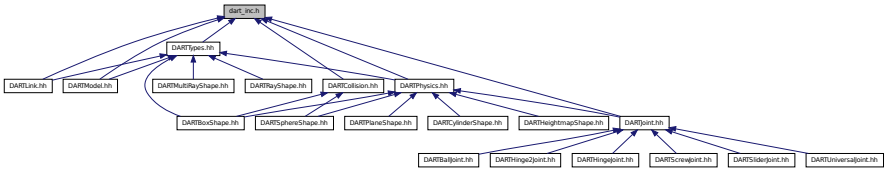
  namespace for physics

11.44 dart_inc.h File Reference

```
#include <dart/math/Helpers.h>      #include <dart/math/-
Geometry.h>  #include <dart/collision/CollisionDetector.-
h>   #include <dart/collision/CollisionNode.h>   #include
<dart/collision/dart/DARTCollisionDetector.h>    #include
<dart/collision/fcl_mesh/FCLMeshCollisionDetector.h> ×
#include <dart/integration/Integrator.h> #include <dart/integration/-
EulerIntegrator.h> #include <dart/integration/RK4Integrator.-
h> #include <dart/dynamics/BallJoint.h> #include <dart/dynamics/-
BodyNode.h>  #include <dart/dynamics/BoxShape.h>  #include
<dart/dynamics/CylinderShape.h> #include <dart/dynamics/-
EllipsoidShape.h>    #include <dart/dynamics/FreeJoint.h>
#include <dart/dynamics/Joint.h> #include <dart/dynamics/-
MeshShape.h> #include <dart/dynamics/PointMass.h> #include
<dart/dynamics/PrismaticJoint.h> #include <dart/dynamics/-
RevoluteJoint.h> #include <dart/dynamics/Shape.h> #include
<dart/dynamics/Skeleton.h>      #include <dart/dynamics/-
ScrewJoint.h>   #include <dart/dynamics/UniversalJoint.h>
#include <dart/dynamics/WeldJoint.h> #include <dart/dynamics/-
SoftBodyNode.h>   #include <dart/dynamics/SoftMeshShape.-
h> #include <dart/constraint/Constraint.h> #include <dart/constraint/-
ConstraintSolver.h>    #include <dart/constraint/Contact-
Constraint.h> #include <dart/constraint/JointLimitConstraint.-
h>     #include <dart/constraint/WeldJointConstraint.h> ×
#include <dart/simulation/World.h>
```

Include dependency graph for dart_inc.h:

This graph shows which files directly or indirectly include this file:



11.45 DARTBallJoint.hh File Reference

`#include "gazebo/physics/BallJoint.hh"` `#include "gazebo/physics/dart/DARTJoint.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for DARTBallJoint.hh:

Classes

- class **gazebo::physics::DARTBallJoint**

  *An **DARTBallJoint** (p. 374).*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.46 DARTBoxShape.hh File Reference

`#include "gazebo/common/Console.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/physics/dart/DARTPhysics.hh"` `#include "gazebo/physics/dart/DARTTypes.hh"` `#include "gazebo/physics/dart/DARTCollision.hh"` `#include "gazebo/physics/PhysicsTypes.hh"` `#include "gazebo/physics/BoxShape.hh"` × `#include "gazebo/util/system.hh"` Include dependency graph for DARTBoxShape.hh:

Classes

- class **gazebo::physics::DARTBoxShape**

DART Box shape.

Namespaces

- namespace **gazebo**

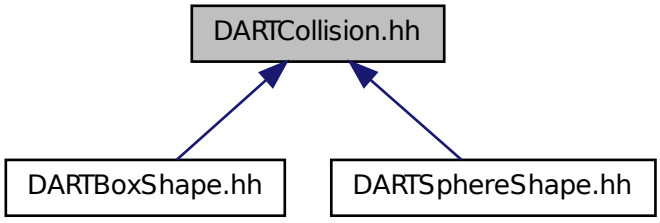
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.47 DARTCollision.hh File Reference

`#include "gazebo/common/CommonTypes.hh"` `#include "gazebo/physics/PhysicsTypes.hh"` `#include "gazebo/physics/Collision.hh"` `#include "gazebo/physics/dart/dart_inc.h"` `#include "gazebo/util/system.hh"` Include dependency graph for DARTCollision.hh:

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::DARTCollision**

  Base *(p. 201) class for all DART collisions.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::physics**

  namespace for physics

11.48 DARTCylinderShape.hh File Reference

`#include "gazebo/common/Console.hh" #include "gazebo/physics/-CylinderShape.hh" #include "gazebo/physics/dart/DART-Physics.hh" #include "gazebo/util/system.hh"` Include dependency graph for DARTCylinderShape.hh:

Classes

- class **gazebo::physics::DARTCylinderShape**

  DART cylinder shape.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

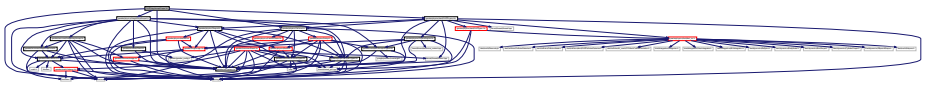
- namespace **gazebo::physics**

  namespace for physics

11.49 DARTHeightmapShape.hh File Reference

`#include <vector> #include "gazebo/physics/Heightmap-Shape.hh" #include "gazebo/physics/dart/DARTPhysics.hh"`

`#include "gazebo/physics/Collision.hh"` `#include "gazebo/util/system.-hh"` Include dependency graph for DARTHeightmapShape.hh:



Classes

- class **gazebo::physics::DARTHeightmapShape**

  DART Height map collision.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.50 DARTHinge2Joint.hh File Reference

`#include "gazebo/math/Angle.hh"` `#include "gazebo/math/-Vector3.hh"` `#include "gazebo/physics/Hinge2Joint.hh"` × `#include "gazebo/physics/dart/DARTJoint.hh"` `#include "gazebo/util/system.-hh"` Include dependency graph for DARTHinge2Joint.hh:

Classes

- class **gazebo::physics::DARTHinge2Joint**

  A two axis hinge joint.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

  namespace for physics

11.51 DARTHingeJoint.hh File Reference

`#include "gazebo/math/Angle.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/physics/HingeJoint.hh"` `#include "gazebo/physics/dart/DARTJoint.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for DARTHingeJoint.hh:

Classes

- class **gazebo::physics::DARTHingeJoint**

  A single axis hinge joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

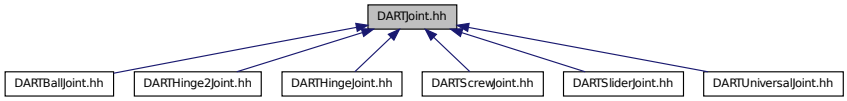
- namespace **gazebo::physics**

  namespace for physics

11.52 DARTJoint.hh File Reference

`#include <boost/any.hpp>` `#include <string>` `#include "gazebo/common/Exception.hh"` `#include "gazebo/physics/Joint.hh"` `#include "gazebo/physics/dart/dart_inc.h"` `#include "gazebo/physics/dart/DARTPhysics.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for DARTJoint.hh:

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::DARTJoint**

  DART joint interface.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.53 DARTLink.hh File Reference

`#include <vector>` `#include "gazebo/physics/Link.hh"` × `#include "gazebo/physics/dart/dart_inc.h"` `#include "gazebo/physics/dart/DARTTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for DARTLink.hh:

Classes

- class **gazebo::physics::DARTLink**

  *DART **Link** (p. 739) class.*

Namespaces

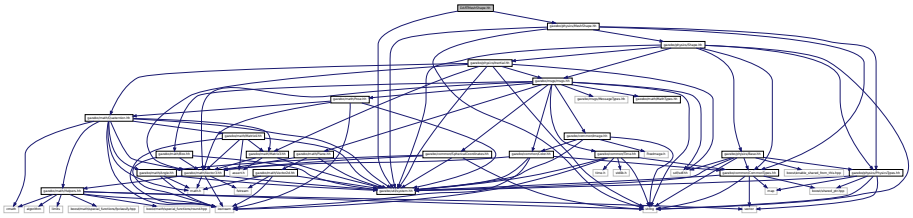
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.54 DARTMeshShape.hh File Reference

`#include "gazebo/physics/MeshShape.hh"` `#include "gazebo/util/system.-hh"` Include dependency graph for DARTMeshShape.hh:



Classes

- class **gazebo::physics::DARTMeshShape**

  Triangle mesh collision.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.55 DARTModel.hh File Reference

`#include "gazebo/physics/dart/dart_inc.h"` `#include "gazebo/physics/dart/-DARTTypes.hh"` `#include "gazebo/physics/Model.hh"` `#include`

"gazebo/util/system.hh" Include dependency graph for DARTModel.hh:

Classes

- class **gazebo::physics::DARTModel**
  DART model class.

Namespaces

- namespace **gazebo**
  Forward declarations for the common classes.
- namespace **gazebo::physics**
  namespace for physics

11.56 DARTMultiRayShape.hh File Reference

#include "gazebo/physics/MultiRayShape.hh" #include "gazebo/physics/dart/-DARTTypes.hh" #include "gazebo/util/system.hh" Include dependency graph for DARTMultiRayShape.hh:

Classes

- class **gazebo::physics::DARTMultiRayShape**
  DART specific version of ***MultiRayShape*** (p. *901*).

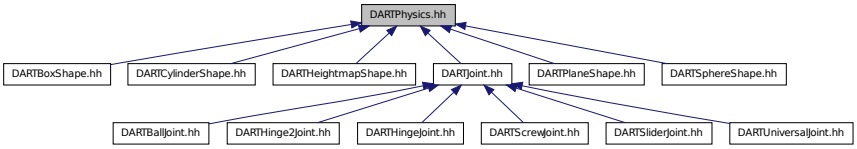
Namespaces

- namespace **gazebo**
  Forward declarations for the common classes.
- namespace **gazebo::physics**
  namespace for physics

11.57 DARTPhysics.hh File Reference

`#include <string>` `#include <boost/thread/thread.hpp>` `#include <boost/thread/mutex.hpp>` `#include "gazebo/physics/PhysicsEngine.hh"` `#include "gazebo/physics/Collision.hh"` `#include "gazebo/physics/Shape.hh"` `#include "gazebo/physics/dart/dart_inc.h"` `#include "gazebo/physics/dart/DARTTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for DARTPhysics.hh:

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::DARTPhysics**

  DART physics engine.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.58 DARTPlaneShape.hh File Reference

`#include "gazebo/physics/PlaneShape.hh"` `#include "gazebo/physics/dart/DARTPhysics.hh"` `#include "gazebo/util/system.hh"` Include depen-

dency graph for DARTPlaneShape.hh:

Classes

- class **gazebo::physics::DARTPlaneShape**

  An DART Plane shape.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.59 DARTRayShape.hh File Reference

`#include <string>  #include "gazebo/physics/RayShape.hh" #include "gazebo/physics/Shape.hh" #include "gazebo/physics/dart/DARTTypes.hh" #include "gazebo/util/system.hh"` Include dependency graph for DARTRayShape.hh:

Classes

- class **gazebo::physics::DARTRayShape**

  Ray collision.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::physics**

  namespace for physics

11.60 DARTScrewJoint.hh File Reference

`#include "gazebo/physics/ScrewJoint.hh"` `#include "gazebo/physics/dart/-DARTJoint.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for DARTScrewJoint.hh:

Classes

- class **gazebo::physics::DARTScrewJoint**

  A screw joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.61 DARTSliderJoint.hh File Reference

`#include "gazebo/physics/SliderJoint.hh"` `#include "gazebo/physics/dart/-DARTJoint.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for DARTSliderJoint.hh:

Classes

- class **gazebo::physics::DARTSliderJoint**

  A slider joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.62 DARTSphereShape.hh File Reference

`#include "gazebo/physics/dart/DARTPhysics.hh"  #include "gazebo/physics/dart/DARTCollision.hh" #include "gazebo/physics/PhysicsTypes.hh"  #include "gazebo/physics/SphereShape.hh" #include "gazebo/util/system.hh"` Include dependency graph for DARTSphereShape.hh:

Classes

- class **gazebo::physics::DARTSphereShape**

  A DART sphere shape.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

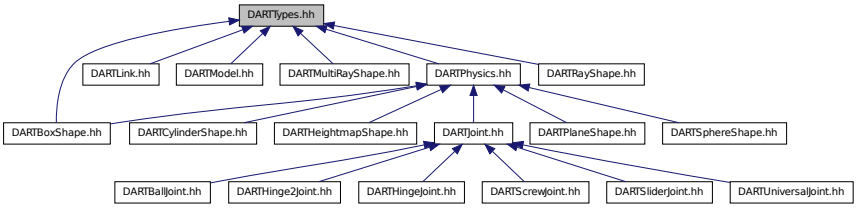
11.63 DARTTypes.hh File Reference

DART wrapper forward declarations and typedefs.

`#include <boost/shared_ptr.hpp>  #include "gazebo/math/Pose.hh" #include "gazebo/physics/dart/dart_inc.h" #include`

"gazebo/util/system.hh" Include dependency graph for DARTTypes.hh:

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::DARTTypes**

  A set of functions for converting between the math types used by gazebo and dart.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Typedefs

- typedef boost::shared_ptr < DARTCollision > **gazebo::physics::DARTCollisionPtr**
- typedef boost::shared_ptr < DARTJoint > **gazebo::physics::DARTJointPtr**
- typedef boost::shared_ptr < DARTLink > **gazebo::physics::DARTLinkPtr**
- typedef boost::shared_ptr < DARTModel > **gazebo::physics::DARTModelPtr**
- typedef boost::shared_ptr < DARTPhysics > **gazebo::physics::DARTPhysicsPtr**
- typedef boost::shared_ptr < DARTRayShape > **gazebo::physics::DARTRayShapePtr**

11.63.1 Detailed Description

DART wrapper forward declarations and typedefs.

11.64 DARTUniversalJoint.hh File Reference

`#include "gazebo/physics/UniversalJoint.hh"` `#include "gazebo/physics/dart/-DARTJoint.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for DARTUniversalJoint.hh:

Classes

- class **gazebo::physics::DARTUniversalJoint**

  A universal joint.

Namespaces

- namespace **gazebo**

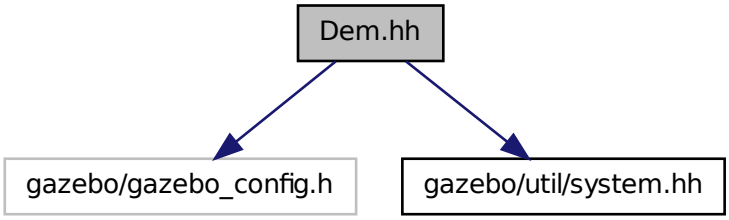
  Forward declarations for the common classes.

- namespace **gazebo::physics**

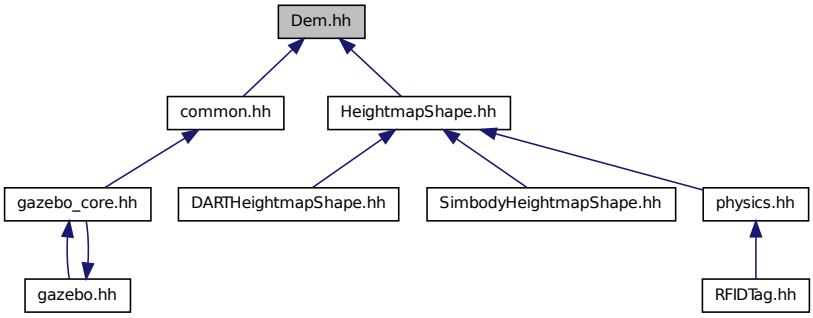
  namespace for physics

11.65 Dem.hh File Reference

`#include <gazebo/gazebo_config.h>` `#include <gazebo/util/system.-hh>` Include dependency graph for Dem.hh:



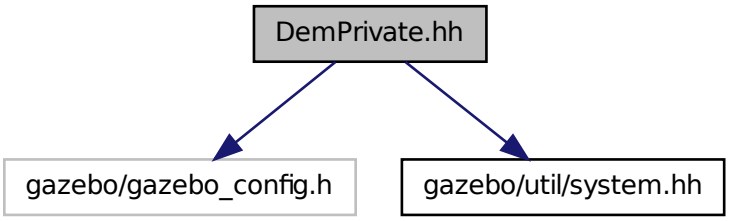
This graph shows which files directly or indirectly include this file:



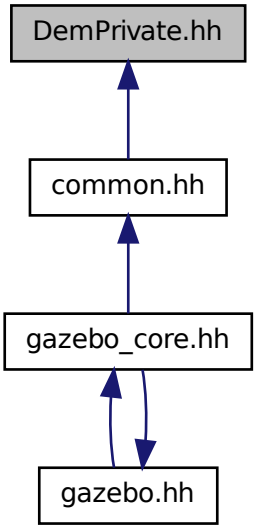
11.66 DemPrivate.hh File Reference

`#include <gazebo/gazebo_config.h>` `#include <gazebo/util/system.-`

hh> Include dependency graph for DemPrivate.hh:

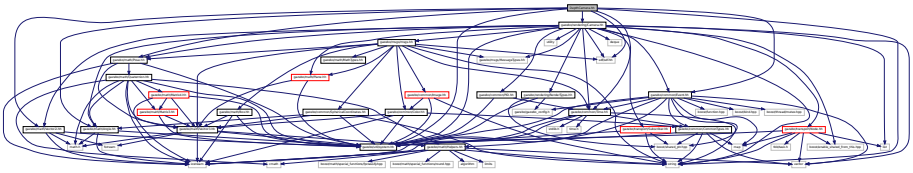


This graph shows which files directly or indirectly include this file:



11.67 DepthCamera.hh File Reference

`#include <string>` `#include <sdf/sdf.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/rendering/Camera.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for DepthCamera.hh:



Classes

- class **gazebo::rendering::DepthCamera**

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

Namespaces

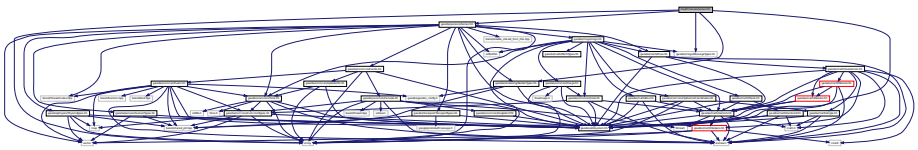
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.68 DepthCameraSensor.hh File Reference

`#include <string>` `#include "gazebo/sensors/Sensor.hh"` `#include "gazebo/msgs/MessageTypes.hh"` `#include "gazebo/rendering/-RenderTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for DepthCameraSensor.hh:



Classes

- class **gazebo::sensors::DepthCameraSensor**

Namespaces

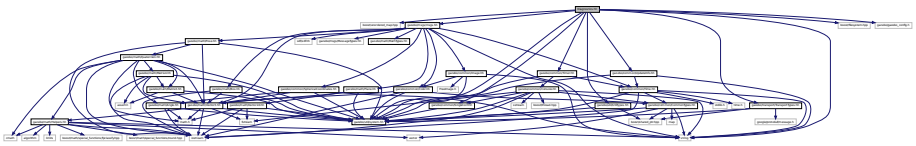
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.69 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 "gazebo/util/system.hh"` Include dependency graph for Diagnostics.hh:



Classes

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

  A diagnostic manager class.
- class **gazebo::util::DiagnosticTimer**

  A timer designed for diagnostics.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::util**

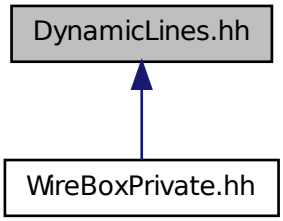
Defines

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

11.70 DynamicLines.hh File Reference

`#include <vector> #include <string> #include "gazebo/common/-CommonIface.hh" #include "gazebo/math/Vector3.hh" #include "gazebo/rendering/Conversions.hh" #include "gazebo/rendering/-DynamicRenderable.hh" #include "gazebo/util/system.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**

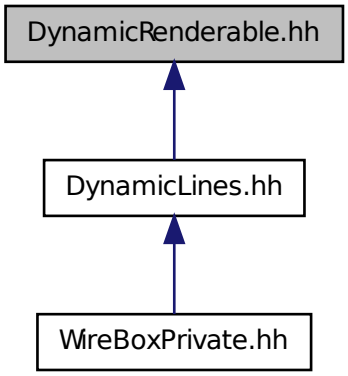
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.71 DynamicRenderable.hh File Reference

`#include <string>` `#include "gazebo/rendering/ogre_gazebo.h"` `#include "gazebo/rendering/RenderTypes.hh"` `#include "gazebo/util/system.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

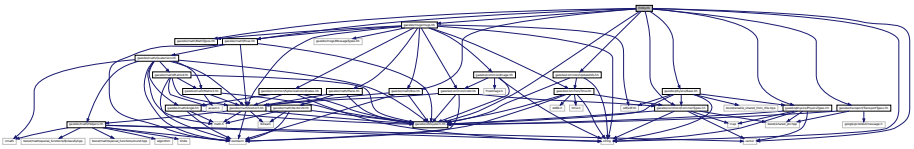
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.72 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 "gazebo/util/system.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. *201) class for all physics objects in Gazebo.*

Namespaces

- namespace **boost**

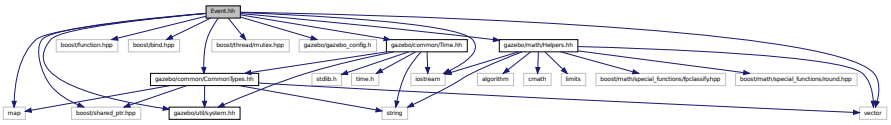
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.73 Event.hh File Reference

`#include <iostream>  #include <vector>  #include <map>` ×
`#include <boost/function.hpp>  #include <boost/bind.hpp>`
`#include <boost/shared_ptr.hpp>` `#include <boost/thread/mutex.-hpp>` `#include <gazebo/gazebo_config.h>` `#include <gazebo/common/-Time.hh>` `#include <gazebo/common/CommonTypes.hh>` `#include <gazebo/math/Helpers.hh>  #include "gazebo/util/system.-hh"` Include dependency graph for Event.hh:



This graph shows which files directly or indirectly include this file:

Classes

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

  A class that encapsulates a connection.
- class **gazebo::event::ConnectionPrivate**
- class **gazebo::event::Event**

  Base class for all events.
- class **gazebo::event::EventPrivate**
- class **gazebo::event::EventT< T >**

  A class for event processing.
- class **gazebo::event::EventTPrivate< T >**

Namespaces

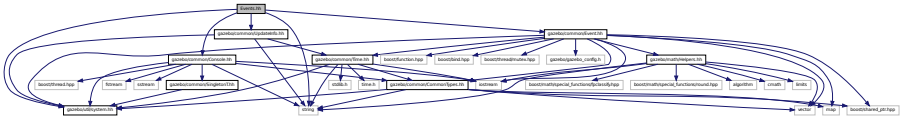
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::event**

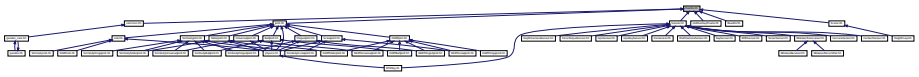
  Event *(p. 514) namespace.*

11.74 Events.hh File Reference

`#include <string>` `#include "gazebo/common/Console.hh"` `#include "gazebo/common/UpdateInfo.hh"` `#include "gazebo/common/Event.hh"` `#include "gazebo/util/system.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. 514) class to get notifications for simulator events.*

Namespaces

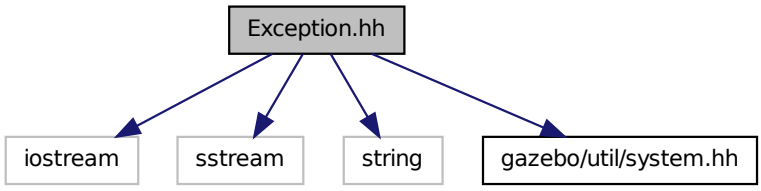
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::event**

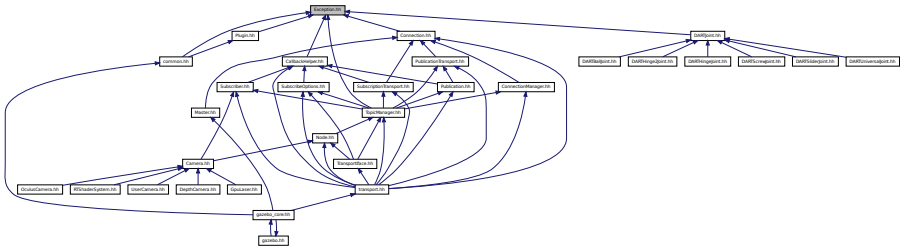
  Event *(p. 514) namespace.*

11.75 Exception.hh File Reference

`#include <iostream>` `#include <sstream>` `#include <string>` × `#include "gazebo/util/system.hh"` 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.

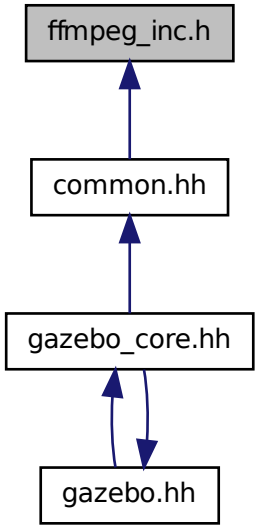
Defines

- #define **gzthrow**(msg)

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

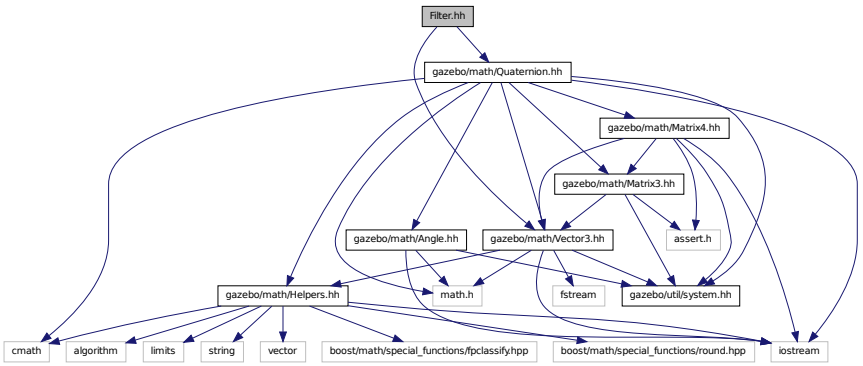
11.76 ffmpeg_inc.h File Reference

This graph shows which files directly or indirectly include this file:

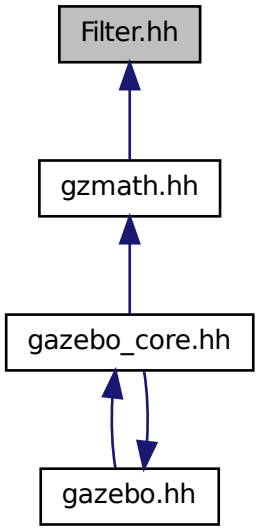


11.77 Filter.hh File Reference

`#include <gazebo/math/Vector3.hh> #include <gazebo/math/-Quaternion.hh>` Include dependency graph for Filter.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::BiQuad< T >**

  Bi-quad filter base class.
- class **gazebo::math::BiQuadVector3**

  BiQuad (p. *217) vector3 filter.*
- class **gazebo::math::Filter< T >**

  Filter (p. *553) base class.*
- class **gazebo::math::OnePole< T >**

  A one-pole DSP filter.
- class **gazebo::math::OnePoleQuaternion**

  One-pole quaternion filter.
- class **gazebo::math::OnePoleVector3**

  One-pole vector3 filter.

Namespaces

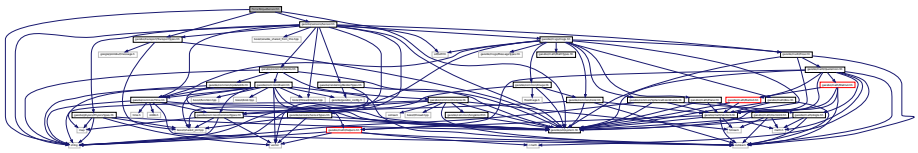
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

11.78 ForceTorqueSensor.hh File Reference

`#include <string>  #include "gazebo/transport/Transport-Types.hh"  #include "gazebo/sensors/Sensor.hh"  #include "gazebo/util/system.hh"` Include dependency graph for ForceTorqueSensor.-hh:



Classes

- class **gazebo::sensors::ForceTorqueSensor**

  Sensor *(*p. *1130) for measure force and torque on a joint.*

Namespaces

- namespace **gazebo**

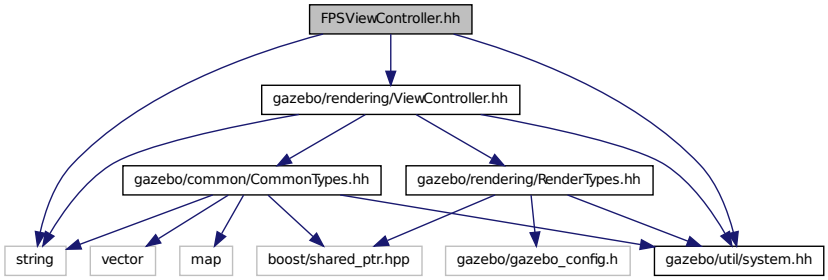
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.79 FPSViewController.hh File Reference

`#include <string> #include "gazebo/rendering/ViewController.-hh" #include "gazebo/util/system.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.

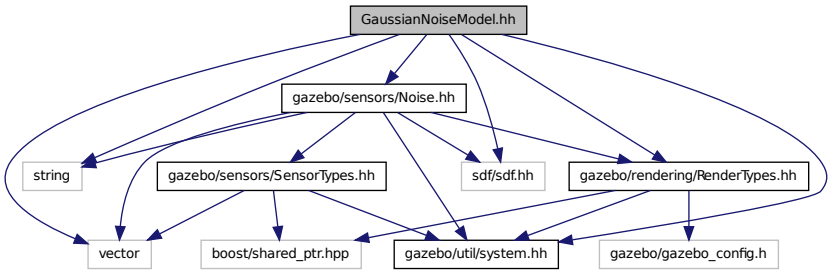
- namespace **gazebo::rendering**

  Rendering namespace.

11.80 GaussianNoiseModel.hh File Reference

```
#include <vector> #include <string> #include <sdf/sdf.-
hh> #include "gazebo/rendering/RenderTypes.hh" #include
"gazebo/sensors/Noise.hh" #include "gazebo/util/system.-
```

hh" Include dependency graph for GaussianNoiseModel.hh:



Classes

- class **gazebo::sensors::GaussianNoiseModel**

  Gaussian noise class.

- class **gazebo::sensors::ImageGaussianNoiseModel**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::sensors**

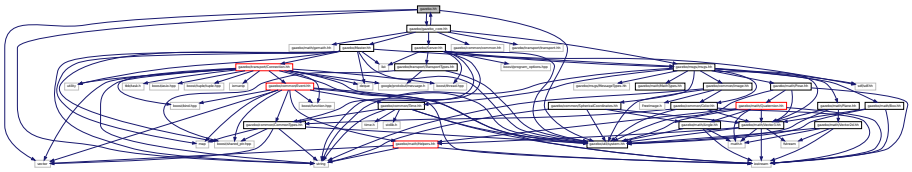
  Sensors namespace.

- namespace **Ogre**

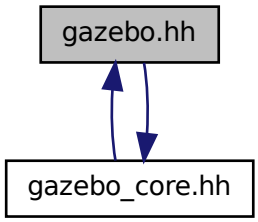
11.81 gazebo.hh File Reference

```
#include <gazebo/gazebo_core.hh>      #include <string>×
#include <vector>  #include "gazebo/util/system.hh"  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

- **GAZEBO_VISIBLE** void **gazebo::addPlugin** (const std::string &_filename)

  Add a system plugin.
- **GAZEBO_VISIBLE** **gazebo::physics::WorldPtr** **gazebo::loadWorld** (const std::string &_worldFile)

  Create and load a new world from an SDF world file.
- **GAZEBO_VISIBLE** void **gazebo::printVersion** ()

  Output version information to the terminal.

- **GAZEBO_VISIBLE** void **gazebo::runWorld** (**gazebo::physics::WorldPtr** _world, unsigned int _iterations)

  Run a world for a specific number of iterations.

- **GAZEBO_VISIBLE** bool **gazebo::setupClient** (int _argc=0, char ∗∗_argv=0)

  Start a gazebo client.

- **GAZEBO_VISIBLE** bool **gazebo::setupClient** (const std::vector< std::string > &_args)

  Start a gazebo client.

- **GAZEBO_VISIBLE** bool **gazebo::setupServer** (int _argc=0, char ∗∗_argv=0)

  Start a gazebo server.

- **GAZEBO_VISIBLE** bool **gazebo::setupServer** (const std::vector< std::string > &_args)

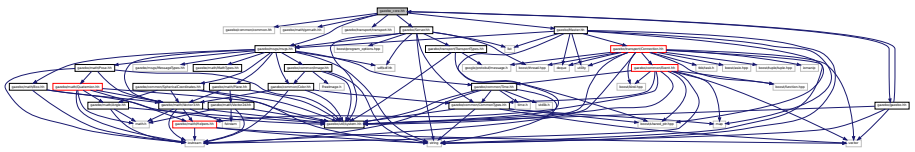
  Start a gazebo server.

- **GAZEBO_VISIBLE** bool **gazebo::shutdown** ()

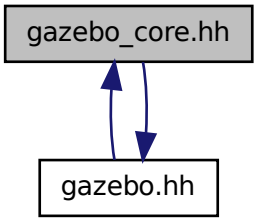
  Stop and cleanup simulation.

11.82 gazebo_core.hh File Reference

`#include <gazebo/common/common.hh>` `#include <gazebo/math/gzmath.hh>` `#include <gazebo/msgs/msgs.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:

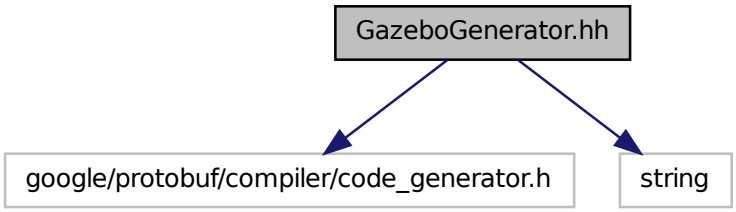


This graph shows which files directly or indirectly include this file:



11.83 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**

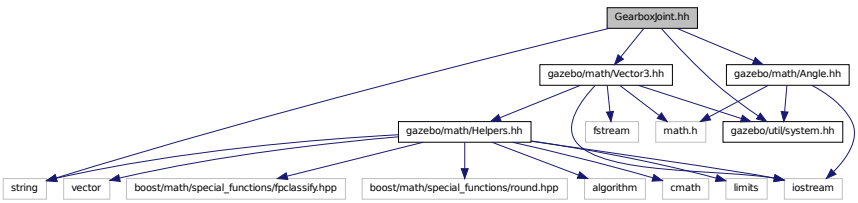
  *Google protobuf message generator for **gazebo::msgs** (p. 135).*

Namespaces

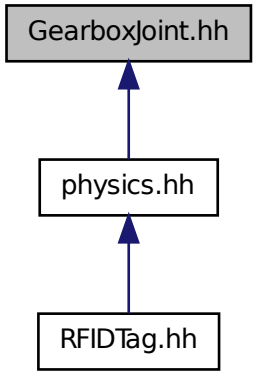
- namespace **google**
- namespace **google::protobuf**
- namespace **google::protobuf::compiler**
- namespace **google::protobuf::compiler::cpp**

11.84 GearboxJoint.hh File Reference

`#include <string>` `#include "gazebo/math/Angle.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for GearboxJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

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

  A double axis gearbox joint.

Namespaces

- namespace **gazebo**

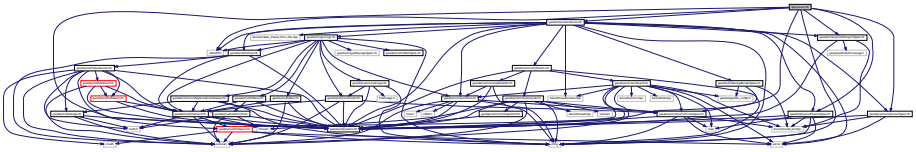
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.85 GpsSensor.hh File Reference

```
#include <string> #include <sdf/sdf.hh> #include "gazebo/sensors/-
Sensor.hh" #include "gazebo/common/CommonTypes.hh" #include
"gazebo/physics/PhysicsTypes.hh" #include "gazebo/sensors/-
SensorTypes.hh"      #include "gazebo/transport/Transport-
```

`Types.hh" #include "gazebo/util/system.hh"` Include dependency graph for GpsSensor.hh:



Classes

- class **gazebo::sensors::GpsSensor**

  GpsSensor *(p. 575) to provide position measurement.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.86 GpuLaser.hh File Reference

`#include <string> #include <vector> #include <sdf/sdf.hh> #include "gazebo/rendering/ogre_gazebo.h" #include "gazebo/rendering/Camera.hh" #include "gazebo/rendering/RenderTypes.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/util/system.hh"` Include dependency graph for GpuLaser.hh:

Classes

- class **gazebo::rendering::GpuLaser**

  GPU based laser distance sensor.

Namespaces

- namespace **gazebo**

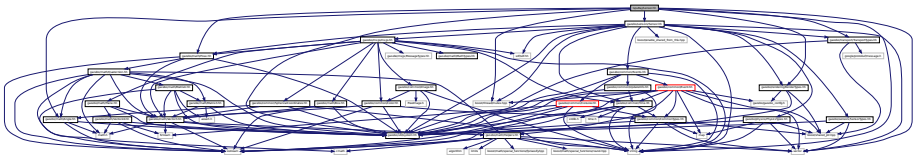
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.87 GpuRaySensor.hh File Reference

`#include <vector>` `#include <string>` `#include <boost/thread/mutex.hpp>` `#include "gazebo/math/Angle.hh"` `#include "gazebo/math/Pose.hh"` `#include "gazebo/transport/TransportTypes.hh"` `#include "gazebo/sensors/Sensor.hh"` `#include "gazebo/rendering/RenderTypes.hh"` `#include "gazebo/util/system.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.

11.88 Grid.hh File Reference

`#include <stdint.h> #include <vector> #include <string> #include "gazebo/rendering/ogre_gazebo.h" #include "gazebo/common/-Color.hh" #include "gazebo/util/system.hh"` Include dependency graph for Grid.hh:

Classes

- class **gazebo::rendering::Grid**

  Displays a grid of cells, drawn with lines.

Namespaces

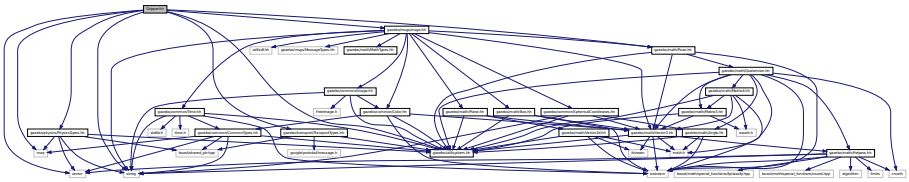
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

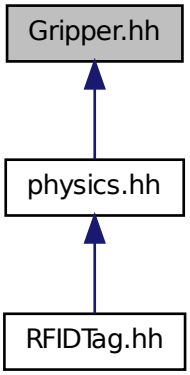
  Rendering namespace.
- namespace **Ogre**

11.89 Gripper.hh File Reference

`#include <map> #include <vector> #include <string> × #include "gazebo/msgs/msgs.hh" #include "gazebo/transport/-TransportTypes.hh" #include "gazebo/math/Pose.hh" #include "gazebo/physics/PhysicsTypes.hh" #include "gazebo/util/system.-hh"` Include dependency graph for Gripper.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Gripper**

  A gripper abstraction.

Namespaces

- namespace **gazebo**

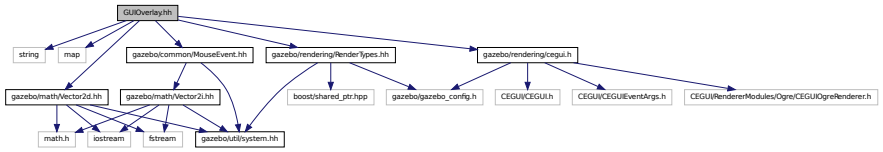
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.90 GUIOverlay.hh File Reference

```
#include <string> #include <map> #include "gazebo/common/-
MouseEvent.hh" #include "gazebo/math/Vector2d.hh" #include
```

"gazebo/rendering/cegui.h" #include "gazebo/rendering/-RenderTypes.hh" Include dependency graph for GUIOverlay.hh:



Classes

- class **gazebo::rendering::GUIOverlay**

  A class that creates a CEGUI overlay on a render window.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.91 GUIOverlayPrivate.hh File Reference

#include <string> #include <vector> #include "gazebo/common/-Events.hh" #include "gazebo/rendering/cegui.h" Include dependency graph for GUIOverlayPrivate.hh:

Classes

- class **gazebo::rendering::GUIOverlayPrivate**

  *Private data for the **GUIOverlay** (p. 612) class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.92 Heightmap.hh File Reference

`#include <string> #include <vector> #include <boost/filesystem.-hpp> #include "gazebo/rendering/ogre_gazebo.h" #include "gazebo/common/Image.hh" #include "gazebo/math/Vector3.-hh" #include "gazebo/math/Vector2d.hh" #include "gazebo/rendering/-Scene.hh" #include "gazebo/util/system.hh"` Include dependency graph for Heightmap.hh:

Classes

- class **gazebo::rendering::DummyPageProvider**

  Pretends to provide procedural page content to avoid page loading.
- 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::ShaderHelperGLS-L**

  Utility class to help with generating shaders for GLSL.
- class **gazebo::rendering::GzTerrainMatGen::SM2Profile**

  Shader model 2 profile target.

Namespaces

- namespace **gazebo**

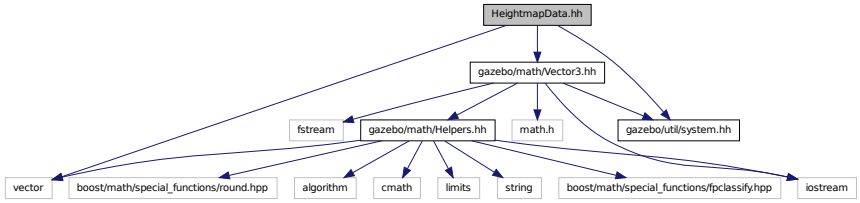
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

Rendering namespace.

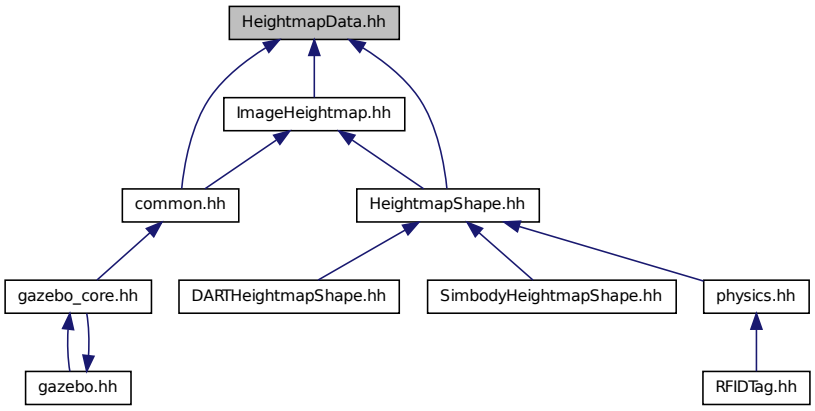
- namespace **Ogre**

11.93 HeightmapData.hh File Reference

`#include <vector>` `#include "gazebo/math/Vector3.hh"` × `#include "gazebo/util/system.hh"` Include dependency graph for - HeightmapData.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::HeightmapData**

Encapsulates a generic heightmap data file.

Namespaces

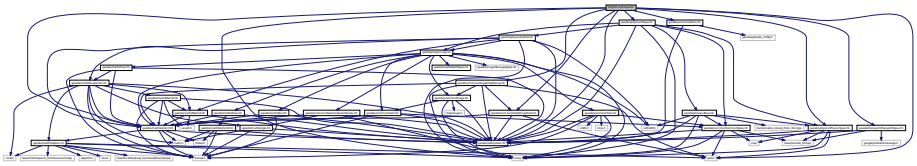
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

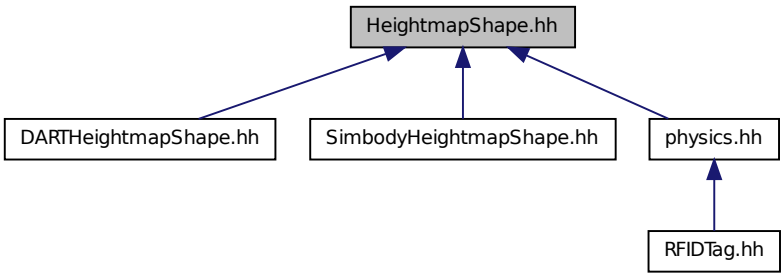
  Common namespace.

11.94 HeightmapShape.hh File Reference

`#include <string> #include <vector> #include "gazebo/common/-ImageHeightmap.hh" #include "gazebo/common/HeightmapData.-hh" #include "gazebo/common/Dem.hh" #include "gazebo/math/-Vector3.hh" #include "gazebo/transport/TransportTypes.-hh" #include "gazebo/physics/PhysicsTypes.hh" #include "gazebo/physics/Shape.hh" #include "gazebo/util/system.-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. *628*) *collision shape builds a heightmap from an image.*

Namespaces

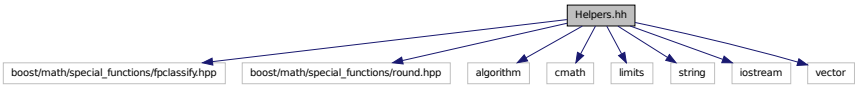
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

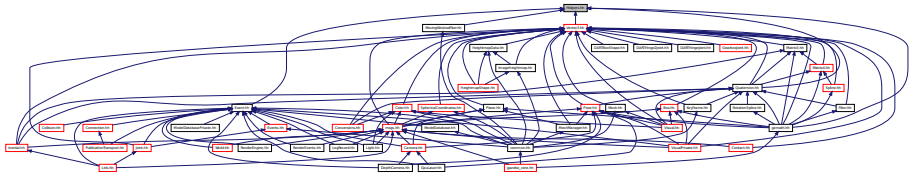
  namespace for physics

11.95 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.

Defines

- #define **GZ_DBL_MAX** std::numeric_limits<double>::max()

  Double maximum value.
- #define **GZ_DBL_MIN** std::numeric_limits<double>::min()

  Double min value.
- #define **GZ_FLT_MAX** std::numeric_limits<float>::max()

  Float maximum value.
- #define **GZ_FLT_MIN** std::numeric_limits<float>::min()

  Float minimum value.
- #define **GZ_INT32_MAX** std::numeric_limits<int32_t>::max()

  32bit integer maximum value
- #define **GZ_INT32_MIN** std::numeric_limits<int32_t>::min()

  32bit integer minimum value
- #define **GZ_UINT32_MAX** std::numeric_limits<uint32_t>::max()

  32bit unsigned integer maximum value
- #define **GZ_UINT32_MIN** std::numeric_limits<uint32_t>::min()

  32bit unsigned integer minimum value

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
- float **gazebo::math::fixnan** (float _v)

  Fix a nan value.
- double **gazebo::math::fixnan** (double _v)

  Fix a nan value.
- 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
- unsigned int **gazebo::math::roundUpPowerOfTwo** (unsigned int _x)

  Get the smallest power of two that is greater or equal to a given value.
- 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)

11.95.1 Define Documentation

11.95.1.1 #define GZ_DBL_MAX std::numeric_limits<double>::max()

Double maximum value.

11.95.1.2 #define GZ_DBL_MIN std::numeric_limits<double>::min()

Double min value.

11.95.1.3 #define GZ_FLT_MAX std::numeric_limits<float>::max()

Float maximum value.

11.95.1.4 #define GZ_FLT_MIN std::numeric_limits<float>::min()

Float minimum value.

11.95.1.5 #define GZ_INT32_MAX std::numeric_limits<int32_t>::max()

32bit integer maximum value

11.95.1.6 #define GZ_INT32_MIN std::numeric_limits<int32_t>::min()

32bit integer minimum value

11.95.1.7 #define GZ_UINT32_MAX std::numeric_limits<uint32_t>::max()

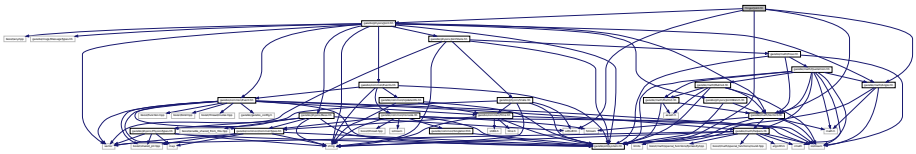
32bit unsigned integer maximum value

11.95.1.8 #define GZ_UINT32_MIN std::numeric_limits<uint32_t>::min()

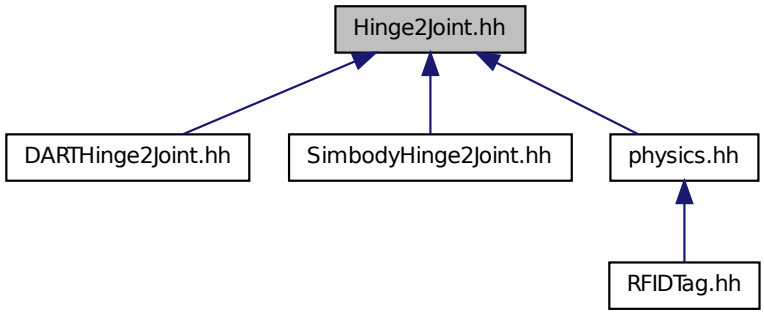
32bit unsigned integer minimum value

11.96 Hinge2Joint.hh File Reference

`#include <sdf/sdf.hh>` `#include "gazebo/math/Angle.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/physics/Joint.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for Hinge2Joint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Hinge2Joint**< **T** >

  A two axis hinge joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

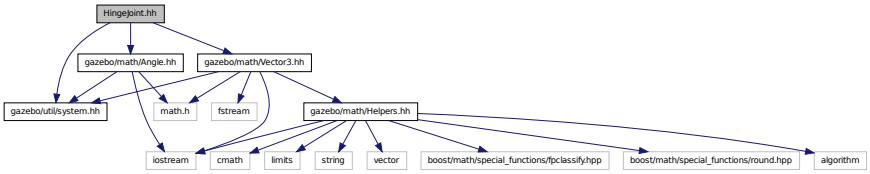
- namespace **gazebo::physics**

  namespace for physics

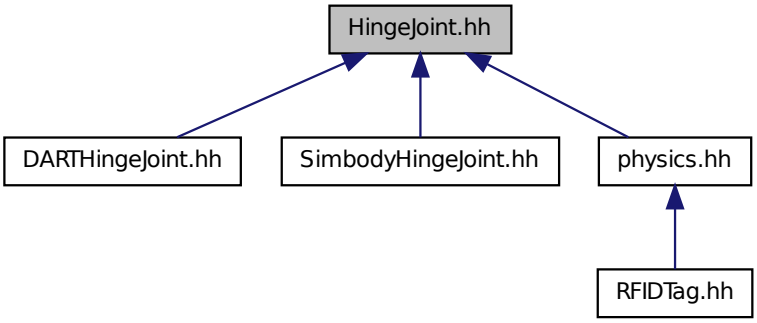
11.97 HingeJoint.hh File Reference

`#include "gazebo/math/Angle.hh"` `#include "gazebo/math/-Vector3.hh"` `#include "gazebo/util/system.hh"` Include dependency

graph for HingeJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

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

  A single axis hinge joint.

Namespaces

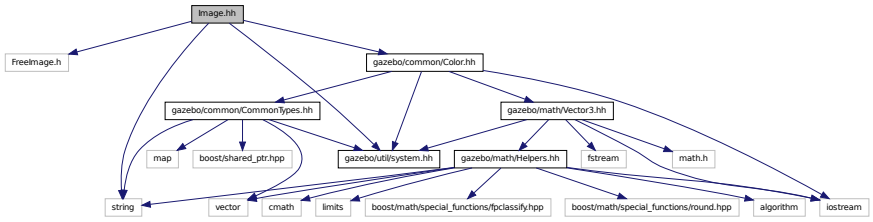
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

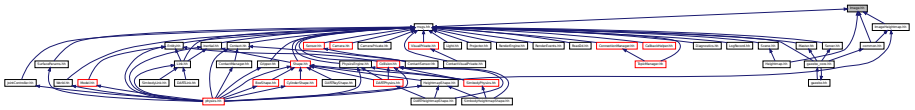
  namespace for physics

11.98 Image.hh File Reference

`#include <FreeImage.h>` `#include <string>` `#include "gazebo/common/-Color.hh"` `#include "gazebo/util/system.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.

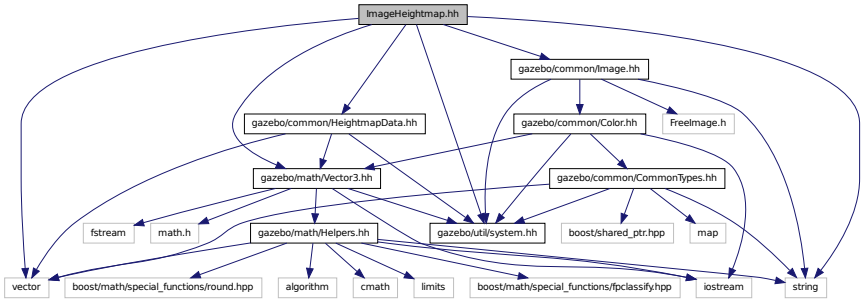
Variables

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

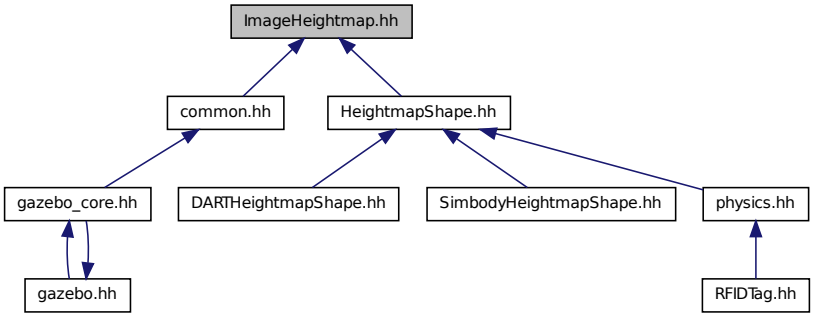
  String names for the pixel formats.

11.99 ImageHeightmap.hh File Reference

`#include <string> #include <vector> #include "gazebo/common/-HeightmapData.hh" #include "gazebo/common/Image.hh" #include "gazebo/math/Vector3.hh" #include "gazebo/util/system.-hh"` Include dependency graph for ImageHeightmap.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::ImageHeightmap**

  Encapsulates an image that will be interpreted as a heightmap.

Namespaces

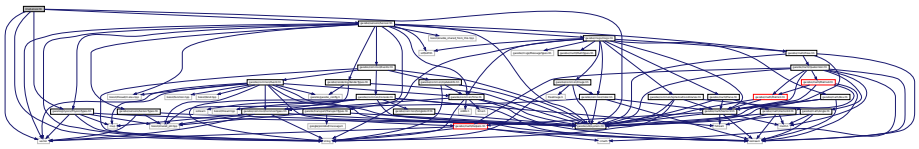
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.100 ImuSensor.hh File Reference

`#include <vector>` `#include <string>` `#include "gazebo/physics/PhysicsTypes.hh"` `#include "gazebo/sensors/Sensor.hh"` × `#include "gazebo/util/system.hh"` Include dependency graph for ImuSensor.hh:



Classes

- class **gazebo::sensors::ImuSensor**

  An IMU sensor.

Namespaces

- namespace **gazebo**

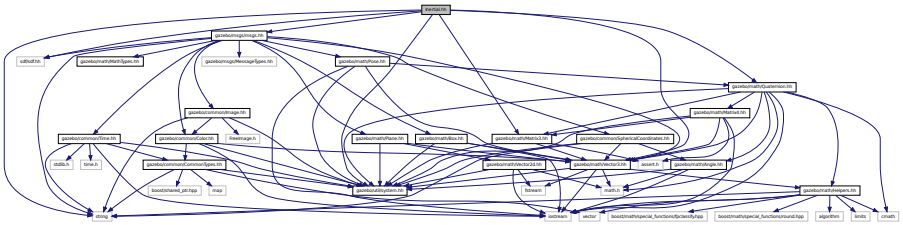
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

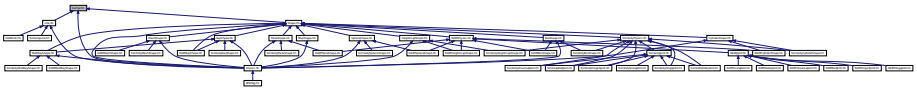
11.101 Inertial.hh File Reference

`#include <string>` `#include <sdf/sdf.hh>` `#include "gazebo/msgs/msgs.hh"` `#include "gazebo/math/Quaternion.hh"` `#include "gazebo/math/`

`Vector3.hh"  #include "gazebo/math/Matrix3.hh"  #include "gazebo/util/system.hh"` Include dependency graph for Inertial.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Inertial**

  A class for inertial information about a link.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

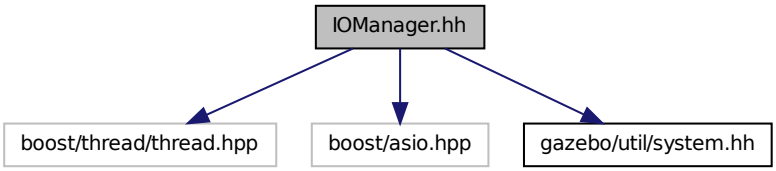
- namespace **gazebo::physics**

  namespace for physics

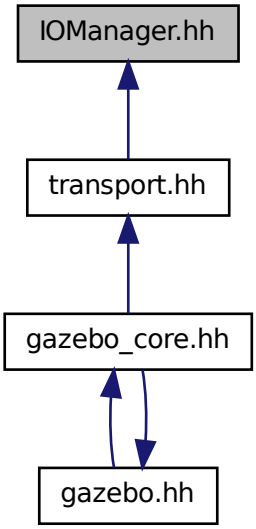
11.102 IOManager.hh File Reference

`#include <boost/thread/thread.hpp> #include <boost/asio.-hpp> #include "gazebo/util/system.hh"` 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.

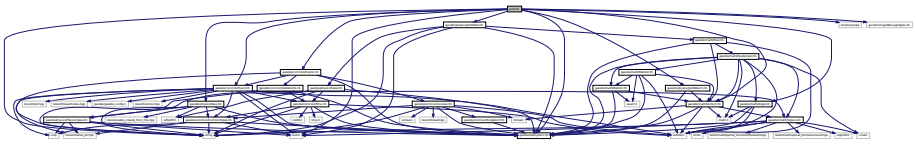
Namespaces

- namespace **gazebo**

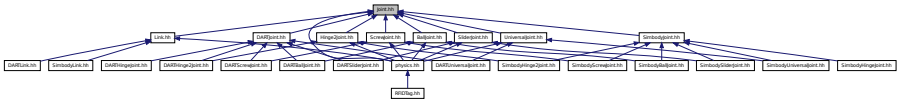
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.103 Joint.hh File Reference

`#include <string> #include <vector> #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 "gazebo/util/system.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. 201) class for all joints.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::physics**

  namespace for physics

Defines

- #define **MAX_JOINT_AXIS** 2

  maximum number of axis per joint anticipated.

11.103.1 Define Documentation

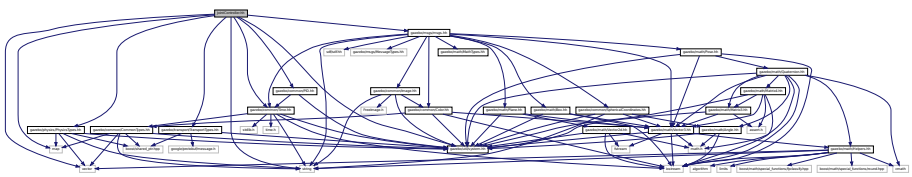
11.103.1.1 #define MAX_JOINT_AXIS 2

maximum number of axis per joint anticipated.

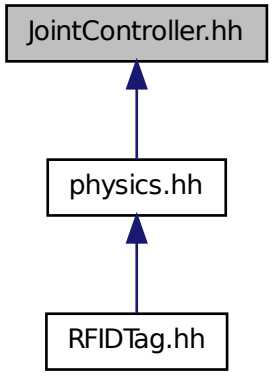
Currently, this is 2 as 3-axis joints (e.g. ball) actuation, control is not there yet.

11.104 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 "gazebo/util/system.hh"` Include dependency graph for JointController.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointController**

  *A class for manipulating **physics::Joint*** (p. *669*).

Namespaces

- namespace **gazebo**

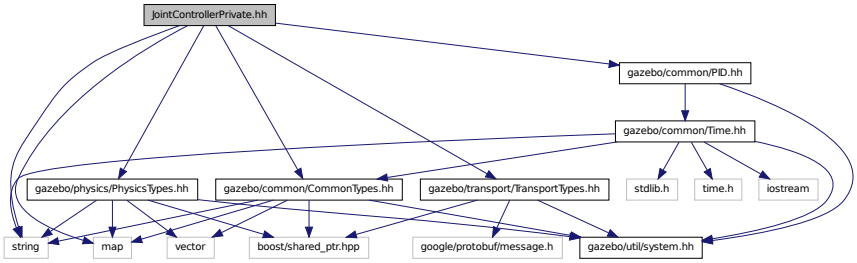
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.105 JointControllerPrivate.hh File Reference

```
#include <string> #include <map> #include "gazebo/transport/-
TransportTypes.hh"  #include "gazebo/common/CommonTypes.-
hh" #include "gazebo/common/PID.hh" #include "gazebo/physics/-
```

PhysicsTypes.hh" Include dependency graph for JointControllerPrivate.hh:



Classes

- class **gazebo::physics::JointControllerPrivate**

Namespaces

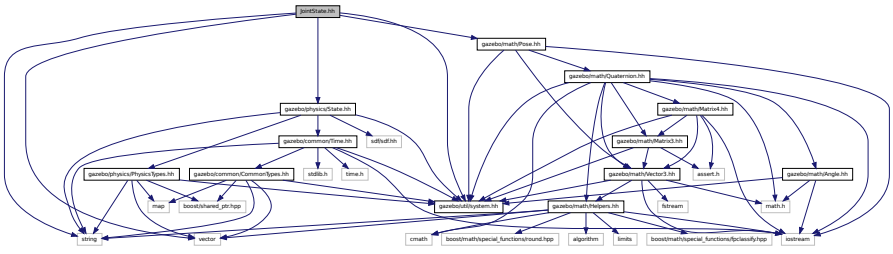
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

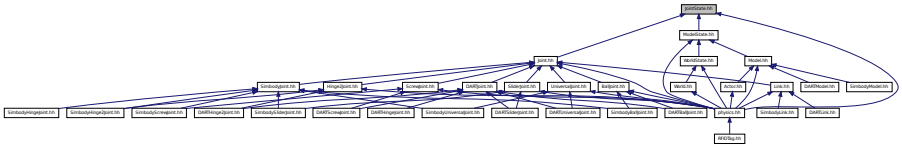
  namespace for physics

11.106 JointState.hh File Reference

#include <vector> #include <string> #include "gazebo/physics/-State.hh" #include "gazebo/math/Pose.hh" #include "gazebo/util/system.-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. 669)

Namespaces

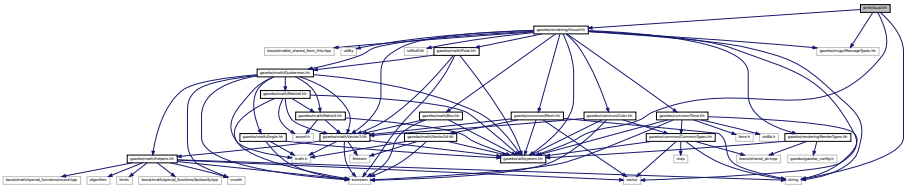
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.107 JointVisual.hh File Reference

`#include <string>  #include "gazebo/msgs/MessageTypes.hh" #include "gazebo/rendering/Visual.hh"#include "gazebo/util/system.hh"` Include dependency graph for JointVisual.hh:



Classes

- class **gazebo::rendering::JointVisual**

  Visualization for joints.

Namespaces

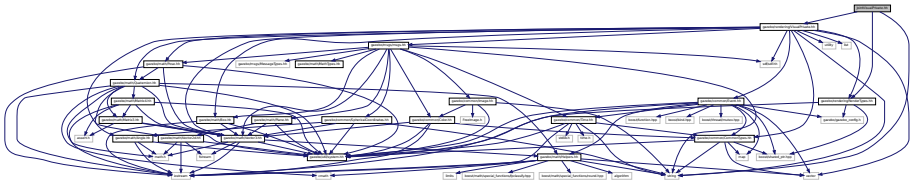
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.108 JointVisualPrivate.hh File Reference

`#include <string>` `#include "gazebo/rendering/RenderTypes.-hh"` `#include "gazebo/rendering/VisualPrivate.hh"` Include dependency graph for JointVisualPrivate.hh:



Classes

- class **gazebo::rendering::JointVisualPrivate**

  *Private data for the Joint **Visual*** (p. *1477*) *class.*

Namespaces

- namespace **gazebo**

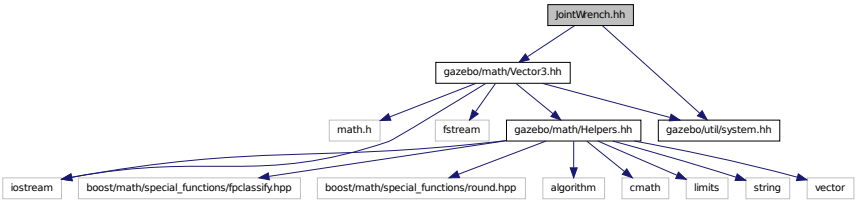
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

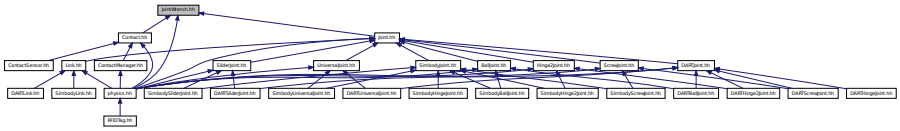
11.109 JointWrench.hh File Reference

`#include "gazebo/math/Vector3.hh"` `#include "gazebo/util/system.-`

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.

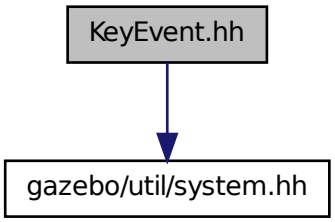
- namespace **gazebo::physics**

  namespace for physics

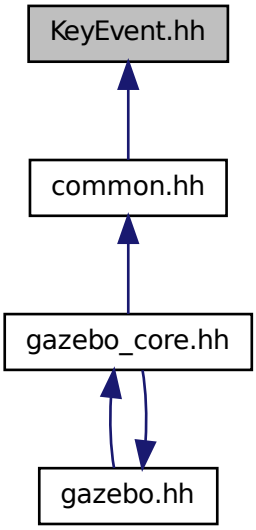
11.110 KeyEvent.hh File Reference

#include "gazebo/util/system.hh" Include dependency graph for Key-

Event.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::KeyEvent**

  Generic description of a keyboard event.

Namespaces

- namespace **gazebo**

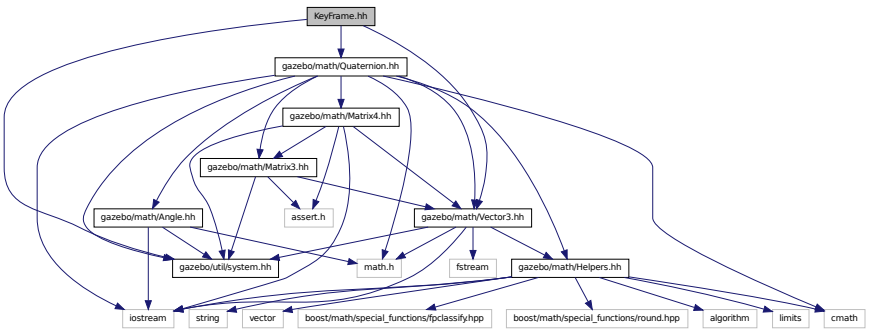
  Forward declarations for the common classes.

- namespace **gazebo::common**

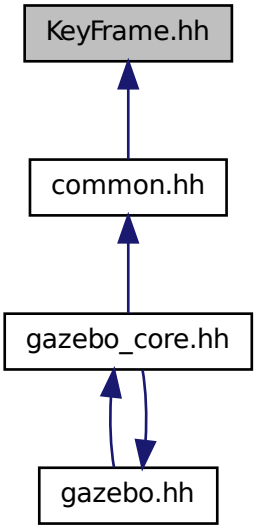
  Common namespace.

11.111 KeyFrame.hh File Reference

`#include "gazebo/math/Vector3.hh" #include "gazebo/math/-Quaternion.hh" #include "gazebo/util/system.hh"` Include dependency graph for KeyFrame.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::KeyFrame**

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

  *A keyframe for a **NumericAnimation*** (p. *936*).
- class **gazebo::common::PoseKeyFrame**

  *A keyframe for a **PoseAnimation*** (p. *1006*).

Namespaces

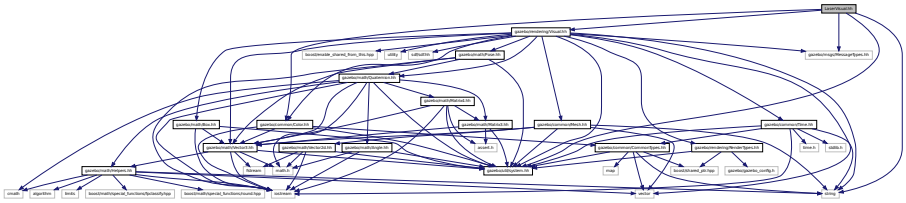
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.112 LaserVisual.hh File Reference

`#include <string>` `#include "gazebo/common/Color.hh"` × `#include "gazebo/msgs/MessageTypes.hh"` `#include "gazebo/rendering/Visual.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for LaserVisual.hh:



Classes

- class **gazebo::rendering::LaserVisual**

  Visualization for laser data.

Namespaces

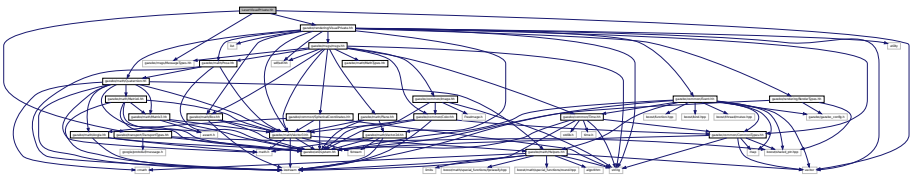
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.113 LaserVisualPrivate.hh File Reference

`#include <vector>` `#include "gazebo/msgs/MessageTypes.hh"` `#include "gazebo/transport/TransportTypes.hh"` `#include "gazebo/rendering/VisualPrivate.hh"` Include dependency graph for LaserVisualPrivate.hh:



Classes

- class **gazebo::rendering::LaserVisualPrivate**

  *Private data for the Laser **Visual*** (p. *1477*) *class.*

Namespaces

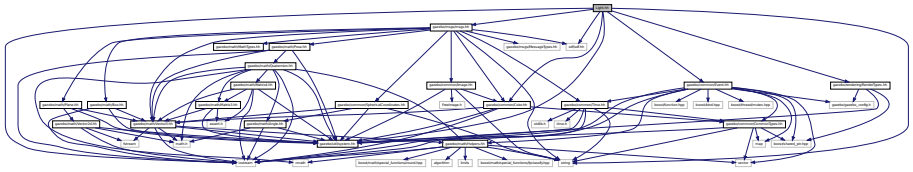
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.114 Light.hh File Reference

`#include <string> #include <iostream> #include <sdf/sdf.-hh> #include "gazebo/msgs/msgs.hh" #include "gazebo/rendering/-RenderTypes.hh" #include "gazebo/common/Event.hh" #include "gazebo/common/Color.hh" #include "gazebo/util/system.-hh"` Include dependency graph for Light.hh:



Classes

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

  A light source.

Namespaces

- namespace **gazebo**

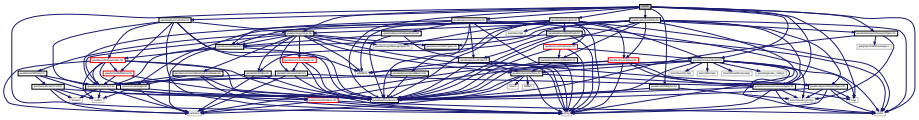
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

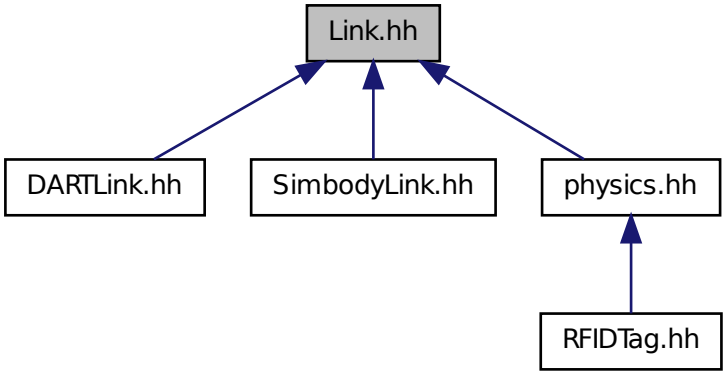
11.115 Link.hh File Reference

```
#include <map>   #include <vector>   #include <string> ×
#include "gazebo/msgs/msgs.hh" #include "gazebo/transport/-
TransportTypes.hh"    #include "gazebo/util/UtilTypes.hh"
#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
"gazebo/util/system.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. *739*) *class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.*

Namespaces

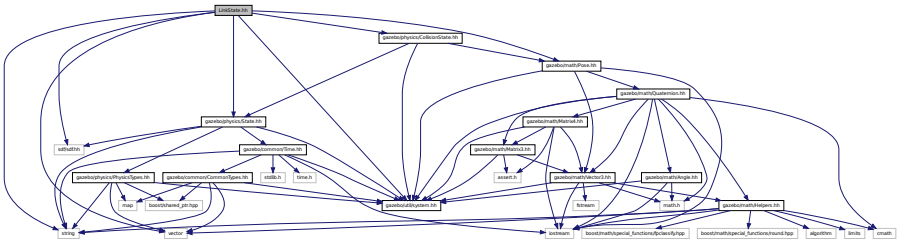
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

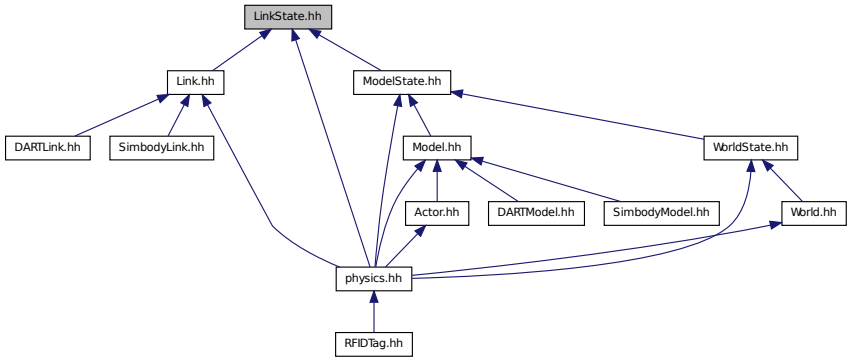
  namespace for physics
- namespace **gazebo::util**

11.116 LinkState.hh File Reference

`#include <vector> #include <string> #include <sdf/sdf.-hh> #include "gazebo/physics/State.hh" #include "gazebo/physics/-CollisionState.hh" #include "gazebo/math/Pose.hh" #include "gazebo/util/system.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. 739) object.*

Namespaces

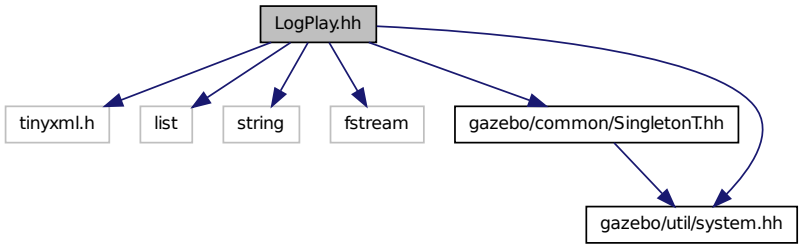
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.117 LogPlay.hh File Reference

`#include <tinyxml.h> #include <list> #include <string> #include <fstream> #include "gazebo/common/SingletonT.hh" #include "gazebo/util/system.hh"` Include dependency graph for LogPlay.hh:



Classes

- class **gazebo::util::LogPlay**

Namespaces

- namespace **gazebo**

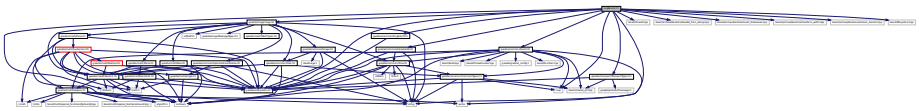
  Forward declarations for the common classes.

- namespace **gazebo::util**

11.118 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/msgs/msgs.hh" #include "gazebo/transport/Transport-
Types.hh"  #include "gazebo/common/UpdateInfo.hh"  #include
"gazebo/common/Event.hh" #include "gazebo/common/Singleton-
T.hh" #include "gazebo/util/system.hh"
```

Include dependency graph for LogRecord.hh:



Classes

- class **gazebo::util::LogRecord**

  addtogroup gazebo_util

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::util**

Defines

- #define **GZ_LOG_VERSION** "1.0"

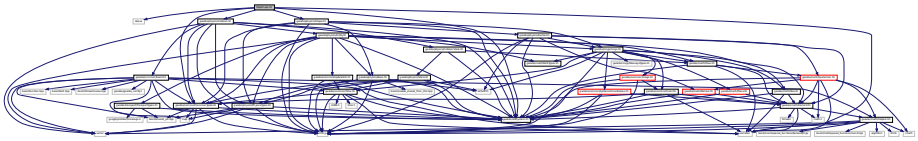
11.118.1 Define Documentation

11.118.1.1 #define GZ_LOG_VERSION "1.0"

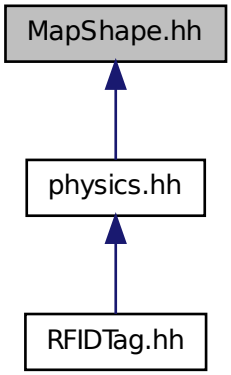
11.119 mainpage.html File Reference

11.120 MapShape.hh File Reference

`#include <deque> #include <string> #include "gazebo/common/-CommonTypes.hh" #include "gazebo/physics/Collision.hh" #include "gazebo/physics/Shape.hh" #include "gazebo/util/system.-hh"` Include dependency graph for MapShape.hh:



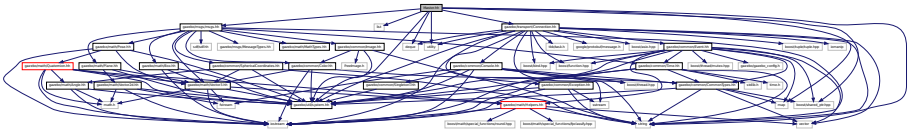
This graph shows which files directly or indirectly include this file:



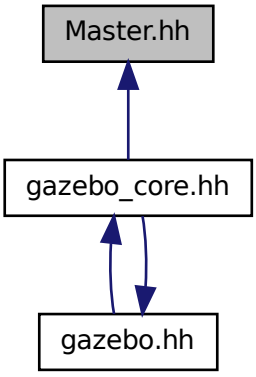
11.121 Master.hh File Reference

`#include <string> #include <list> #include <deque> #include <utility> #include <map> #include <boost/shared-`

_ptr.hpp> #include "gazebo/msgs/msgs.hh" #include "gazebo/transport/-Connection.hh" #include "gazebo/util/system.hh" Include dependency graph for Master.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Master**

  A manager that directs topic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

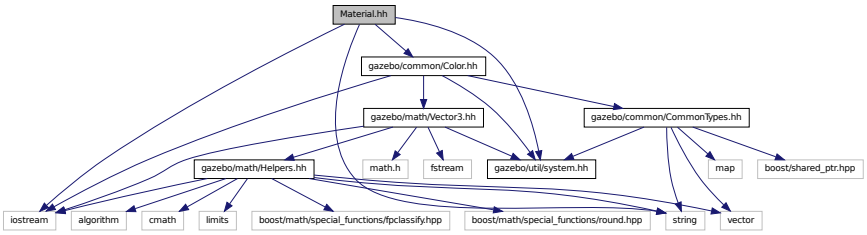
Namespaces

- namespace **gazebo**

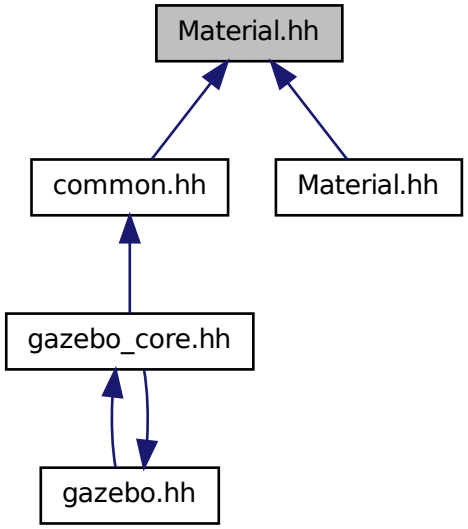
  Forward declarations for the common classes.

11.122 Material.hh File Reference

`#include <string>` `#include <iostream>` `#include "gazebo/common/-Color.hh"` `#include "gazebo/util/system.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

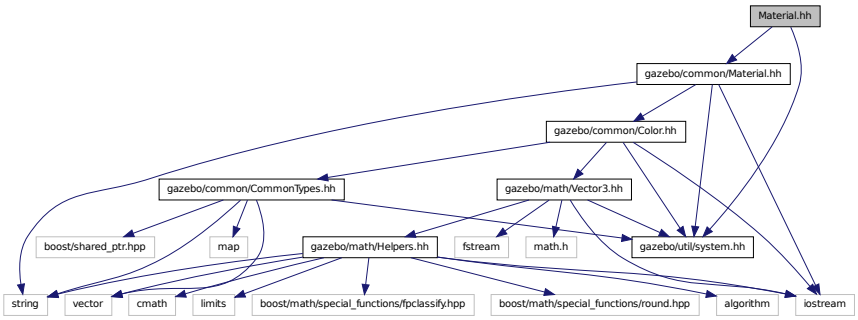
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.123 Material.hh File Reference

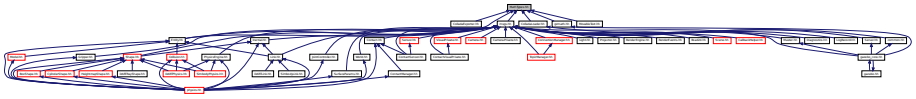
`#include "gazebo/common/Material.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for rendering/Material.hh:



11.124 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**

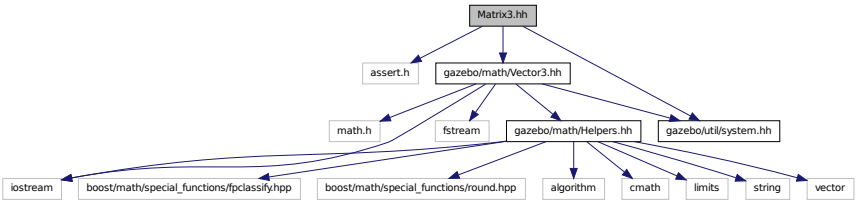
  Math namespace.

11.124.1 Detailed Description

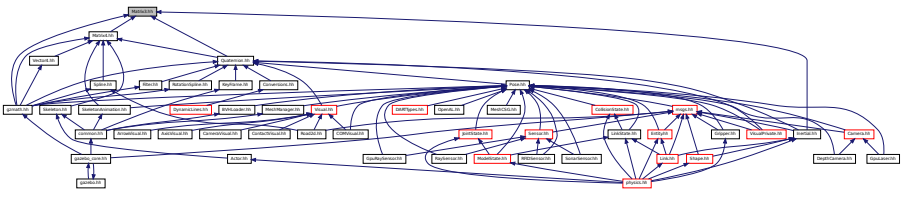
Forward declarations for the math classes.

11.125 Matrix3.hh File Reference

`#include <assert.h>` `#include "gazebo/math/Vector3.hh"` × `#include "gazebo/util/system.hh"` Include dependency graph for - Matrix3.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Matrix3**

A 3x3 matrix class.

Namespaces

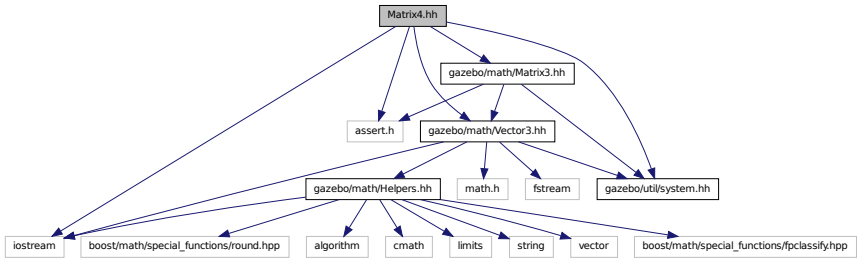
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

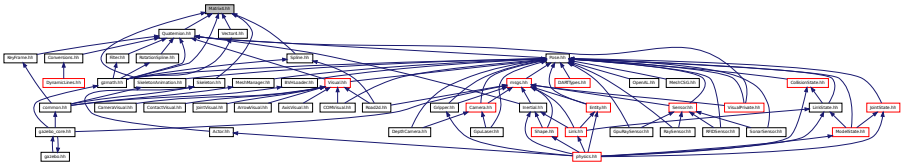
  Math namespace.

11.126 Matrix4.hh File Reference

`#include <assert.h> #include <iostream> #include "gazebo/math/-Vector3.hh" #include "gazebo/math/Matrix3.hh" #include "gazebo/util/system.hh"` Include dependency graph for Matrix4.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Matrix4**

  A 3x3 matrix class.

Namespaces

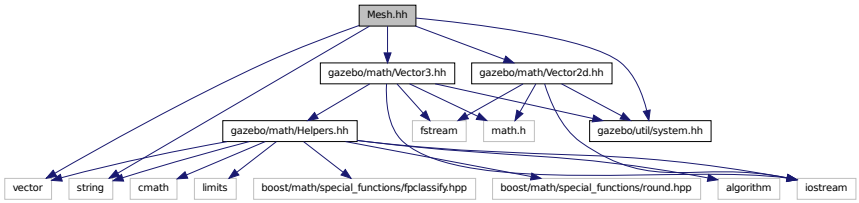
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

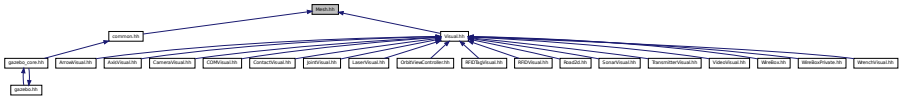
  Math namespace.

11.127 Mesh.hh File Reference

`#include <vector>` `#include <string>` `#include "gazebo/math/-Vector3.hh"` `#include "gazebo/math/Vector2d.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for Mesh.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Mesh**

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

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

  A child mesh.

Namespaces

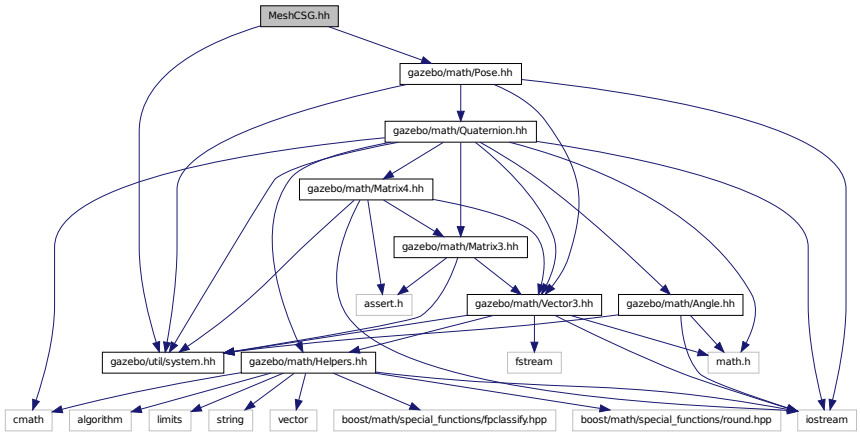
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.128 MeshCSG.hh File Reference

`#include "gazebo/math/Pose.hh"` `#include "gazebo/util/system.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**

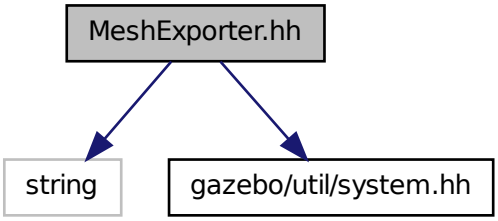
11.128.1 Typedef Documentation

11.128.1.1 typedef _GPtrArray GPtrArray

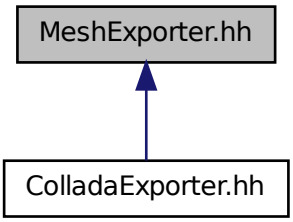
11.128.1.2 typedef _GtsSurface GtsSurface

11.129 MeshExporter.hh File Reference

`#include <string>` `#include "gazebo/util/system.hh"` Include dependency graph for MeshExporter.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MeshExporter**

  Base class for exporting meshes.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

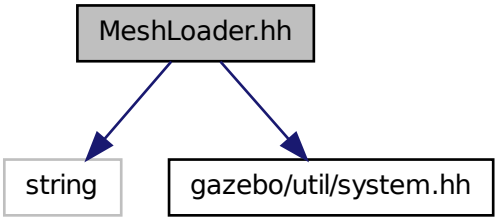
- namespace **gazebo::common**

  Common namespace.

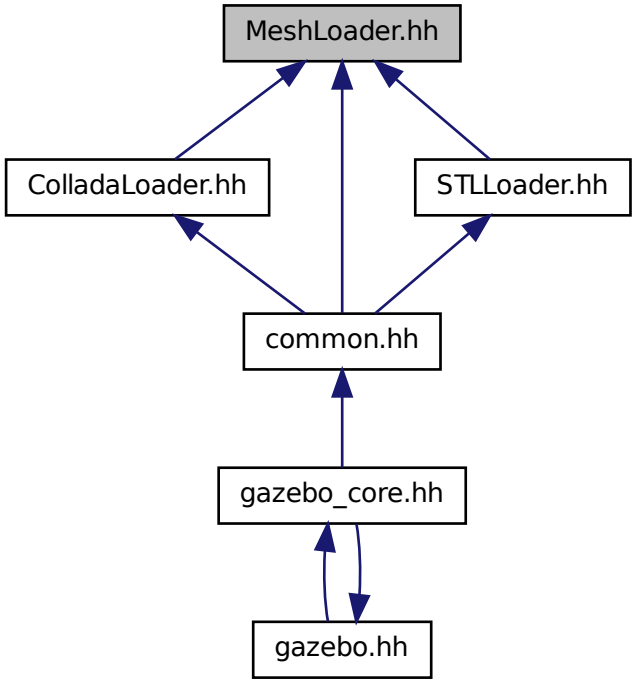
11.130 MeshLoader.hh File Reference

`#include <string>` `#include "gazebo/util/system.hh"` 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**

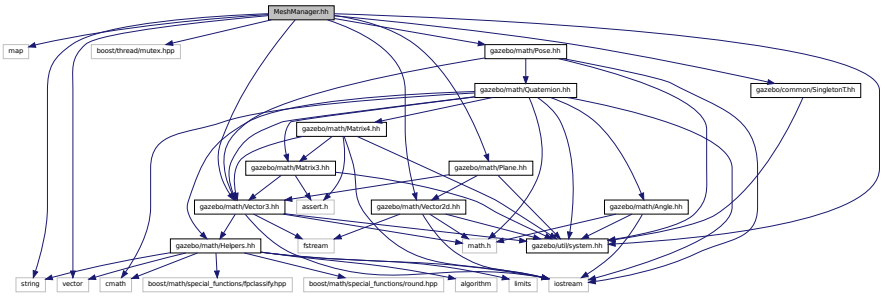
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

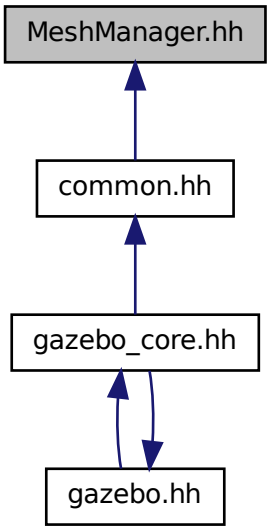
11.131 MeshManager.hh File Reference

```
#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 "gazebo/util/system.-
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**

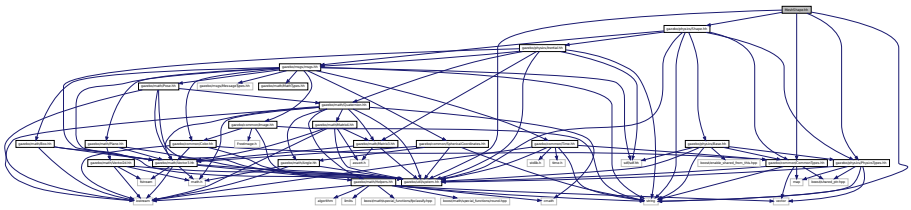
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

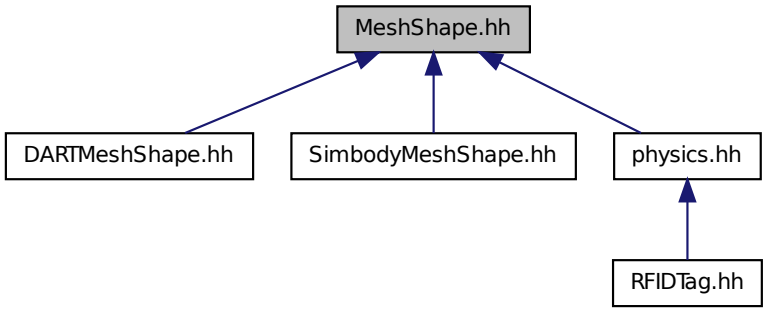
11.132 MeshShape.hh File Reference

```
#include <string>   #include "gazebo/common/CommonTypes.-
```

hh" #include "gazebo/physics/PhysicsTypes.hh" #include "gazebo/physics/Shape.hh" #include "gazebo/util/system.-hh" Include dependency graph for MeshShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::MeshShape**

  Triangle mesh collision shape.

Namespaces

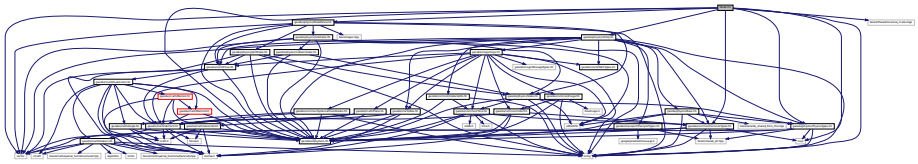
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

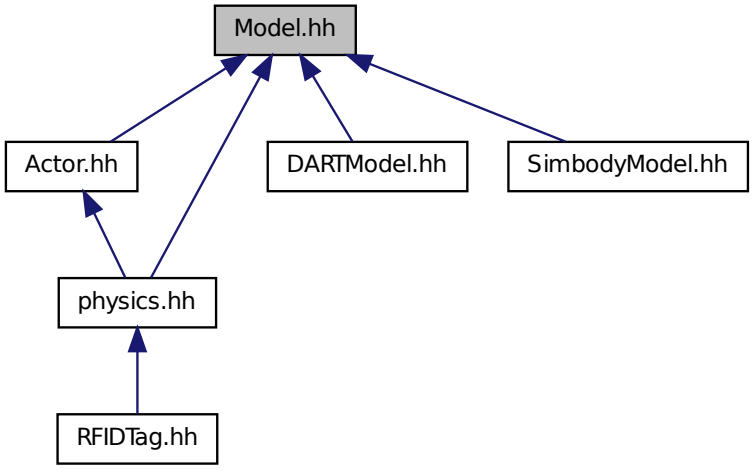
  namespace for physics

11.133 Model.hh File Reference

`#include <string>  #include <map>  #include <vector>` × `#include <boost/thread/recursive_mutex.hpp>` `#include "gazebo/common/-CommonTypes.hh"  #include "gazebo/physics/PhysicsTypes.-hh"` `#include "gazebo/physics/ModelState.hh"` `#include "gazebo/physics/-Entity.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for Model.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Model**

A model is a collection of links, joints, and plugins.

Namespaces

- namespace **boost**
- namespace **gazebo**

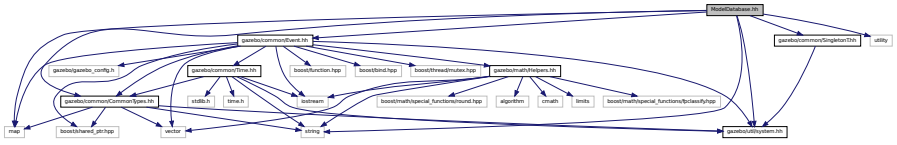
  Forward declarations for the common classes.

- namespace **gazebo::physics**

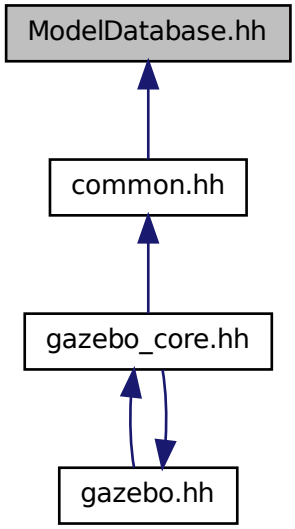
  namespace for physics

11.134 ModelDatabase.hh File Reference

`#include <string>  #include <map>  #include <utility>` × `#include "gazebo/common/Event.hh"` `#include "gazebo/common/SingletonT.hh"  #include "gazebo/common/CommonTypes.hh"` × `#include "gazebo/util/system.hh"` Include dependency graph for ModelDatabase.hh:



This graph shows which files directly or indirectly include this file:



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.

Defines

- #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.

11.134.1 Define Documentation

11.134.1.1 #define GZ_MODEL_DB_MANIFEST_FILENAME "database.config"

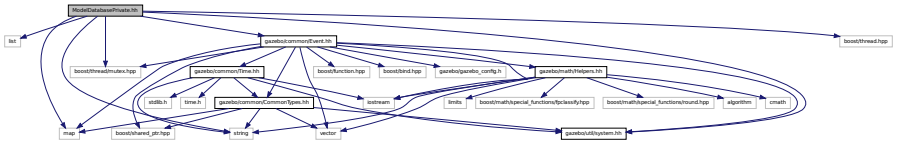
The file name of model database XML configuration.

11.134.1.2 #define GZ_MODEL_MANIFEST_FILENAME "model.config"

The file name of model XML configuration.

11.135 ModelDatabasePrivate.hh File Reference

`#include <list> #include <map> #include <string> #include <boost/thread.hpp> #include <boost/thread/mutex.hpp> #include "gazebo/common/Event.hh" #include "gazebo/util/system.hh"` Include dependency graph for ModelDatabasePrivate.hh:



Classes

- class **gazebo::common::ModelDatabasePrivate**

  *Private class attributes for **ModelDatabase*** (p. *864*).

Namespaces

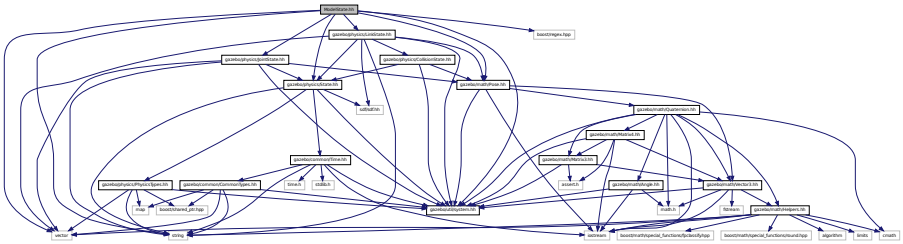
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

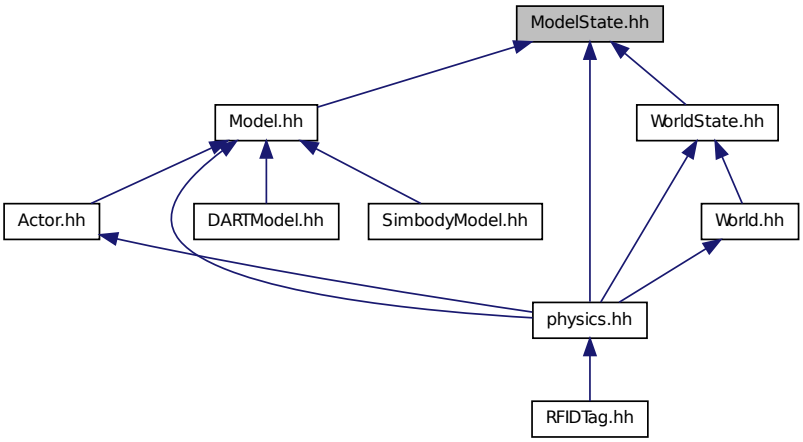
  Common namespace.

11.136 ModelState.hh File Reference

`#include <vector>` `#include <string>` `#include <boost/regex.hpp>` `#include "gazebo/math/Pose.hh"` `#include "gazebo/physics/State.hh"` `#include "gazebo/physics/LinkState.hh"` `#include "gazebo/physics/JointState.hh"` `#include "gazebo/util/system.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. 846) *object.*

Namespaces

- namespace **gazebo**

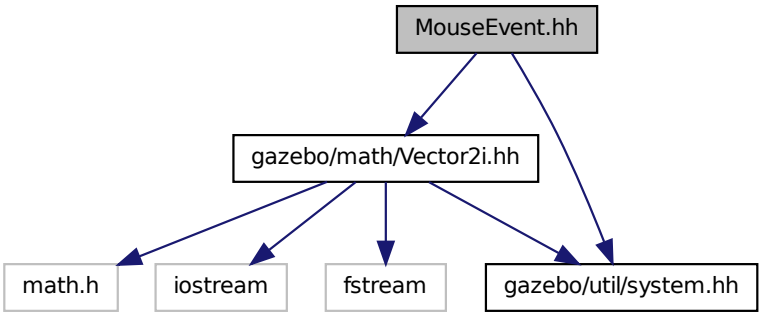
  Forward declarations for the common classes.

- namespace **gazebo::physics**

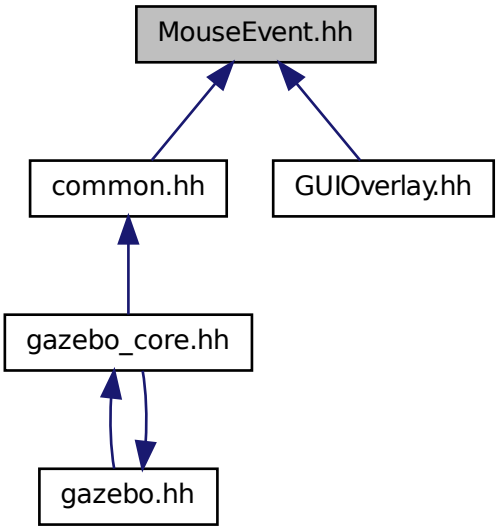
  namespace for physics

11.137 MouseEvent.hh File Reference

`#include "gazebo/math/Vector2i.hh"` `#include "gazebo/util/system.-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.

11.138 MovableText.hh File Reference

```
#include <string> #include "gazebo/rendering/ogre_gazebo.-
```

h" #include "gazebo/common/CommonTypes.hh" #include "gazebo/common/-Color.hh" #include "gazebo/math/MathTypes.hh" #include "gazebo/util/system.hh" Include dependency graph for MovableText.hh:

Classes

- class **gazebo::rendering::MovableText**

  Movable text.

Namespaces

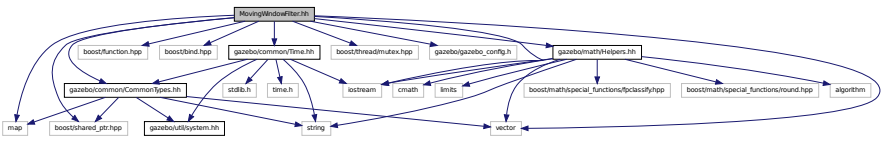
- namespace **boost**
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

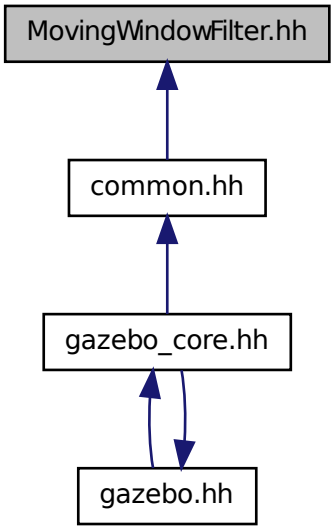
  Rendering namespace.

11.139 MovingWindowFilter.hh File Reference

#include <iostream> #include <vector> #include <map> × #include <boost/function.hpp> #include <boost/bind.hpp> #include <boost/shared_ptr.hpp> #include <boost/thread/mutex.-hpp> #include <gazebo/gazebo_config.h> #include <gazebo/common/-Time.hh> #include <gazebo/common/CommonTypes.hh> #include <gazebo/math/Helpers.hh> Include dependency graph for MovingWindow-Filter.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MovingWindowFilter**< **T** >

  *Base class for **MovingWindowFilter*** (p. *890*).
- class **gazebo::common::MovingWindowFilterPrivate**< **T** >

Namespaces

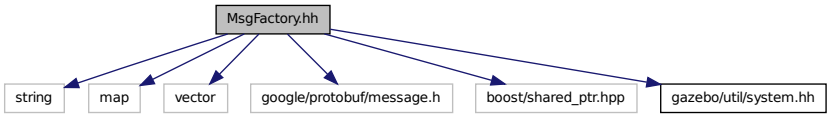
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.140 MsgFactory.hh File Reference

`#include <string>  #include <map>  #include <vector>` × `#include <google/protobuf/message.h>` `#include <boost/shared_ptr.hpp>` `#include "gazebo/util/system.hh"` Include dependency graph for MsgFactory.hh:



Classes

- class **gazebo::msgs::MsgFactory**

  A factory that generates protobuf message based on a string type.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::msgs**

  Messages namespace.

Defines

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

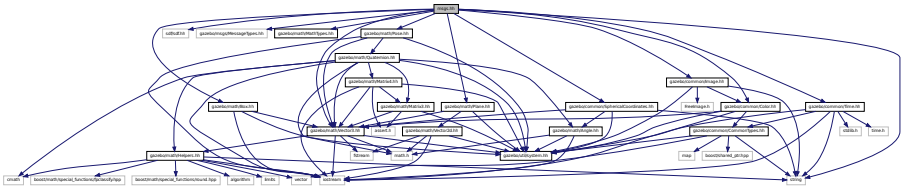
  Static message registration macro.

Typedefs

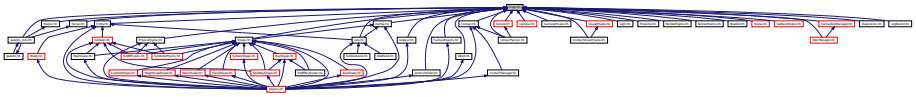
- typedef boost::shared_ptr < google::protobuf::Message >(∗ **gazebo::msgs::-MsgFactoryFn**)()

11.141 msgs.hh File Reference

`#include <string>` `#include <sdf/sdf.hh>` `#include "gazebo/msgs/-MessageTypes.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/SphericalCoordinates.hh"` `#include "gazebo/common/Color.hh"` `#include "gazebo/common/-Time.hh"` `#include "gazebo/common/Image.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

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

  *Convert a **math::Vector3** (p. 1440) to a msgs::Vector3d.*
- **GAZEBO_VISIBLE** msgs::Quaternion **gazebo::msgs::Convert** (const math::-Quaternion &_q)

  *Convert a **math::Quaternion** (p. 1029) to a msgs::Quaternion.*

- **GAZEBO_VISIBLE** msgs::Pose **gazebo::msgs::Convert** (const math::Pose &-_p)

  *Convert a **math::Pose** (p. 995) to a msgs::Pose.*
- **GAZEBO_VISIBLE** msgs::Color **gazebo::msgs::Convert** (const common::-Color &_c)

  *Convert a **common::Color** (p. 312) to a msgs::Color.*
- **GAZEBO_VISIBLE** msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)

  *Convert a **common::Time** (p. 1361) to a msgs::Time.*
- **GAZEBO_VISIBLE** msgs::PlaneGeom **gazebo::msgs::Convert** (const math::-Plane &_p)

  *Convert a **math::Plane** (p. 984) to a msgs::PlaneGeom.*
- **GAZEBO_VISIBLE** math::Vector3 **gazebo::msgs::Convert** (const msgs::-Vector3d &_v)

  Convert a msgs::Vector3d to a math::Vector.
- **GAZEBO_VISIBLE** math::Quaternion **gazebo::msgs::Convert** (const msgs::-Quaternion &_q)

  *Convert a msgs::Quaternion to a **math::Quaternion** (p. 1029).*
- **GAZEBO_VISIBLE** math::Pose **gazebo::msgs::Convert** (const msgs::Pose &-_p)

  *Convert a msgs::Pose to a **math::Pose** (p. 995).*
- **GAZEBO_VISIBLE** common::Color **gazebo::msgs::Convert** (const msgs::-Color &_c)

  *Convert a msgs::Color to a **common::Color** (p. 312).*
- **GAZEBO_VISIBLE** common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time** (p. 1361).*
- **GAZEBO_VISIBLE** math::Plane **gazebo::msgs::Convert** (const msgs::Plane-Geom &_p)

  Convert a msgs::PlaneGeom to a common::Plane.
- **GAZEBO_VISIBLE** msgs::Request * **gazebo::msgs::CreateRequest** (const std::string &_request, const std::string &_data="")

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

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

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

Get the header from a protobuf message.

- **GAZEBO_VISIBLE** msgs::GUI **gazebo::msgs::GUIFromSDF** (sdf::ElementPtr _sdf)

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

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

  Create a msgs::Light from a light SDF element.
- **GAZEBO_VISIBLE** sdf::ElementPtr **gazebo::msgs::LightToSDF** (const msgs::Light &_msg, sdf::ElementPtr _sdf=sdf::ElementPtr())

  Create an SDF element from a msgs::Scene.
- **GAZEBO_VISIBLE** msgs::MeshGeom **gazebo::msgs::MeshFromSDF** (sdf::ElementPtr _sdf)

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

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

  *Convert a msgs::Image to a **common::Image** (p. 637).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)

  *Set a msgs::Image from a **common::Image** (p. 637).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 1440).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 1440).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 1029).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 995).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)

  *Set a msgs::Color from a **common::Color** (p. 312).*
- **GAZEBO_VISIBLE** void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)

*Set a msgs::Time from a **common::Time** (p. 1361).*

- void **gazebo::msgs::Set** (msgs::SphericalCoordinates *_s, const common::SphericalCoordinates &_v)

  *Set a msgs::SphericalCoordinates from a **common::SphericalCoordinates** (p. 1310) object.*

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

  *Set a msgs::Plane from a **math::Plane** (p. 984).*

- **GAZEBO_VISIBLE** void **gazebo::msgs::Stamp** (msgs::Header *_header)

  Time stamp a header.

- **GAZEBO_VISIBLE** void **gazebo::msgs::Stamp** (msgs::Time *_time)

  Set the time in a time message.

- **GAZEBO_VISIBLE** msgs::TrackVisual **gazebo::msgs::TrackVisualFromSDF** (sdf::ElementPtr _sdf)

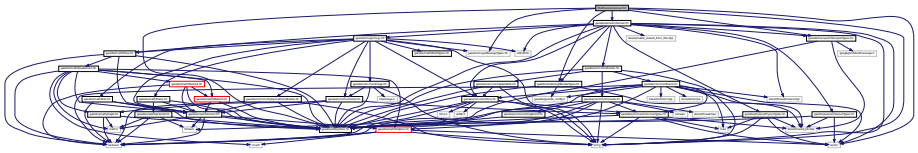
  Create a msgs::TrackVisual from a track visual SDF element.

- **GAZEBO_VISIBLE** msgs::Visual **gazebo::msgs::VisualFromSDF** (sdf::ElementPtr _sdf)

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

11.142 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 "gazebo/util/system.hh"` Include dependency graph for MultiCameraSensor.hh:



Classes

- class **gazebo::sensors::MultiCameraSensor**

  Multiple camera sensor.

Namespaces

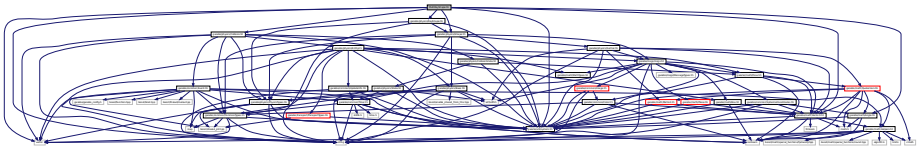
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

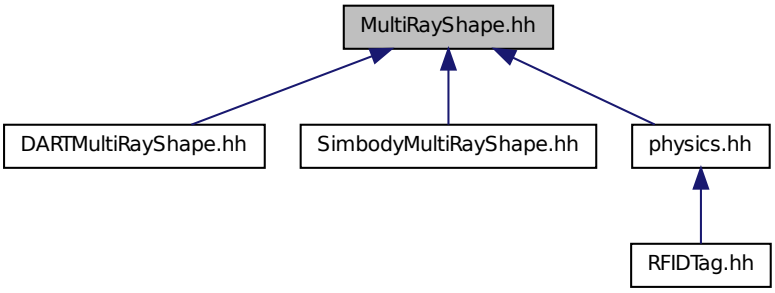
  Sensors namespace.

11.143 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 "gazebo/util/system.-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

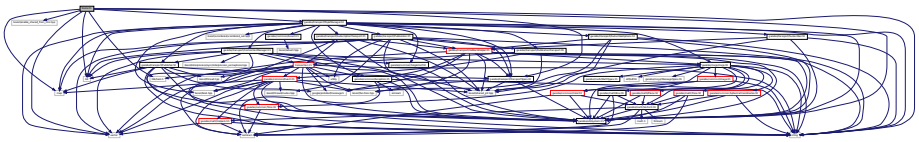
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

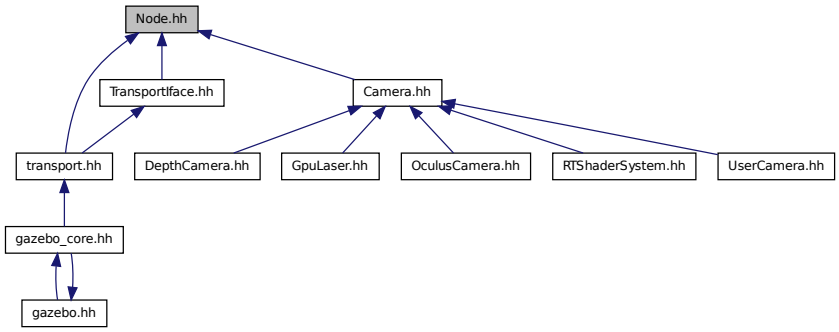
  namespace for physics

11.144 Node.hh File Reference

`#include <tbb/task.h>  #include <boost/enable_shared_from_this.hpp>  #include <map>  #include <list>  #include <string>  #include <vector>  #include "gazebo/transport/TransportTypes.hh" #include "gazebo/transport/TopicManager.hh" #include "gazebo/util/system.hh"` Include dependency graph for Node.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Node**

  A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

- class **gazebo::transport::PublishTask**

Namespaces

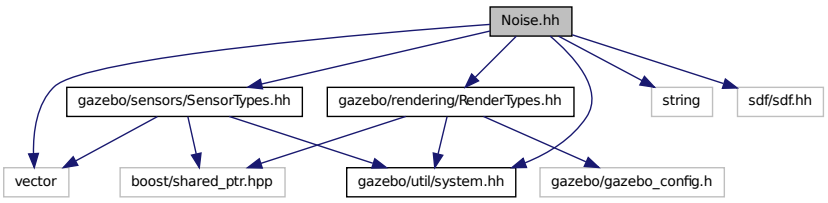
- namespace **gazebo**

  Forward declarations for the common classes.

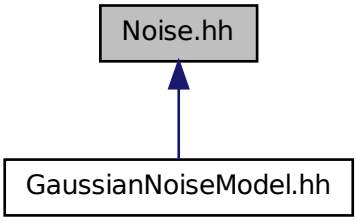
- namespace **gazebo::transport**

11.145 Noise.hh File Reference

`#include <vector> #include <string> #include <sdf/sdf.-hh> #include "gazebo/rendering/RenderTypes.hh" #include "gazebo/sensors/SensorTypes.hh" #include "gazebo/util/system.-hh"` Include dependency graph for Noise.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::Noise**

  Noise *(p. 931) models for sensor output signals.*

- class **gazebo::sensors::NoiseFactory**

  Use this noise manager for creating and loading noise models.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

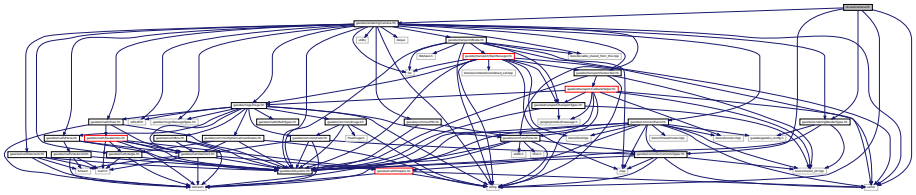
- namespace **gazebo::sensors**

  Sensors namespace.

11.146 OculusCamera.hh File Reference

`#include <string>` `#include <vector>` `#include "gazebo/rendering/-Camera.hh"` `#include "gazebo/rendering/RenderTypes.hh"` × `#include "gazebo/common/CommonTypes.hh"` Include dependency graph

for OculusCamera.hh:



Classes

- class **gazebo::rendering::OculusCamera**

  A camera used for user visualization of a scene.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

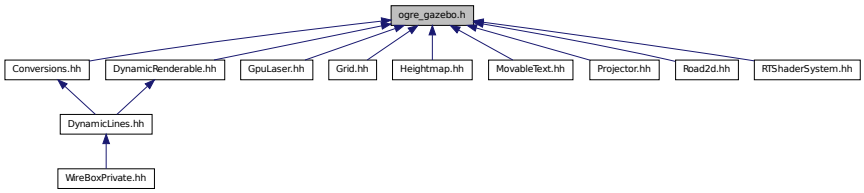
  Rendering namespace.
- namespace **Ogre**
- namespace **OVR**
- namespace **OVR::Util**
- namespace **OVR::Util::Render**

11.147 ogre_gazebo.h File Reference

```
#include <OGRE/Ogre.h>  #include <OGRE/OgreImageCodec.h>
#include <OGRE/OgreMovableObject.h>  #include <OGRE/Ogre-
Renderable.h>  #include <OGRE/OgrePlugin.h>  #include <O-
GRE/OgreDataStream.h>  #include <OGRE/OgreLogManager.h> ×
#include <OGRE/OgreWindowEventUtilities.h>  #include <OG-
RE/OgreSceneQuery.h> #include <OGRE/OgreRoot.h> #include
<OGRE/OgreSceneManager.h>  #include <OGRE/OgreSceneNode.-
h>   #include <OGRE/OgreVector3.h>   #include <OGRE/Ogre-
ManualObject.h>   #include <OGRE/OgreMaterialManager.h> ×
#include <OGRE/OgreColourValue.h>    #include <OGRE/Ogre-
Quaternion.h>   #include <OGRE/OgreMesh.h>   #include <O-
GRE/OgreHardwareBufferManager.h>     #include <OGRE/Ogre-
Camera.h> #include <OGRE/OgreNode.h> #include <OGRE/Ogre-
```

SimpleRenderable.h> #include <OGRE/OgreFrameListener.-
h> #include <OGRE/OgreTexture.h> #include <OGRE/Ogre-
RenderObjectListener.h> #include <OGRE/OgreTechnique.h>
#include <OGRE/OgrePass.h> #include <OGRE/OgreTexture-
UnitState.h> #include <OGRE/OgreGpuProgramManager.h> ×
#include <OGRE/OgreHighLevelGpuProgramManager.h> #include
<OGRE/OgreHardwarePixelBuffer.h> #include <OGRE/Ogre-
ShadowCameraSetupPSSM.h> #include <OGRE/Paging/OgrePage-
Manager.h> #include <OGRE/Paging/OgrePagedWorld.h> #include
<OGRE/Terrain/OgreTerrainPaging.h> #include <OGRE/Terrain/-
OgreTerrainMaterialGeneratorA.h> #include <OGRE/Terrain/-
OgreTerrain.h> #include <OGRE/Terrain/OgreTerrainGroup.-
h> #include <OGRE/OgreFontManager.h> Include dependency graph for ogre_gazebo.h:

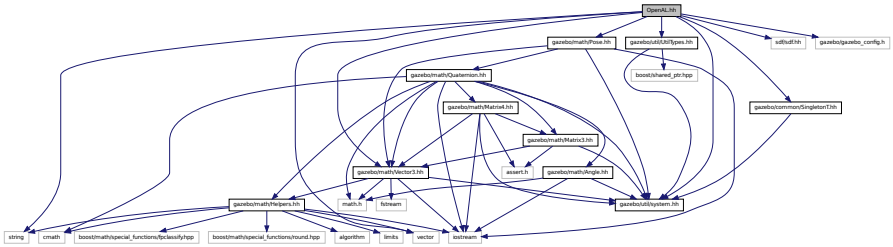
This graph shows which files directly or indirectly include this file:



11.148 OpenAL.hh File Reference

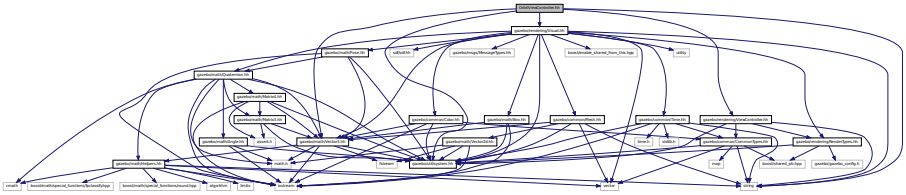
#include <string> #include <vector> #include <sdf/sdf.-
hh> #include "gazebo/math/Vector3.hh" #include "gazebo/math/-
Pose.hh" #include "gazebo/common/SingletonT.hh" #include
"gazebo/util/UtilTypes.hh" #include "gazebo/gazebo_config.-
h" #include "gazebo/util/system.hh" Include dependency graph for

OpenAL.hh:



11.149 OrbitViewController.hh File Reference

`#include <string>` `#include "gazebo/rendering/Visual.hh"` `#include "gazebo/rendering/ViewController.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for OrbitViewController.hh:



Classes

- class **gazebo::rendering::OrbitViewController**

  Orbit view controller.

Namespaces

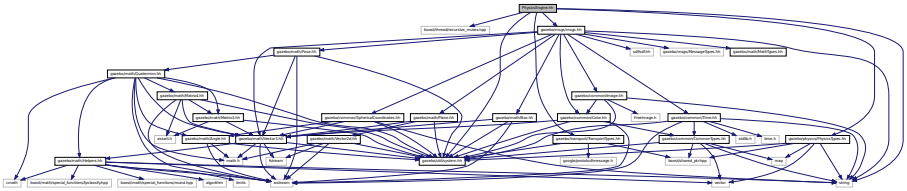
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

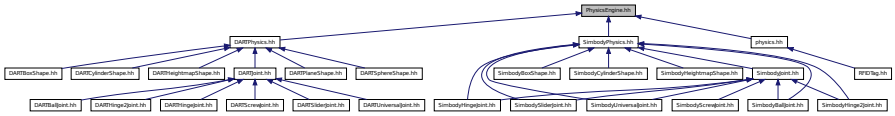
  Rendering namespace.

11.150 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 "gazebo/util/system.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. *201) class for a physics engine.*

Namespaces

- namespace **gazebo**

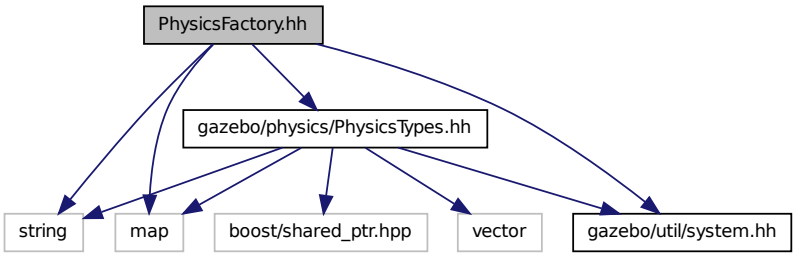
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

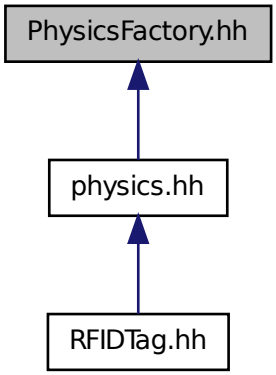
11.151 PhysicsFactory.hh File Reference

`#include <string>` `#include <map>` `#include "gazebo/physics/-`

`PhysicsTypes.hh" #include "gazebo/util/system.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

Defines

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

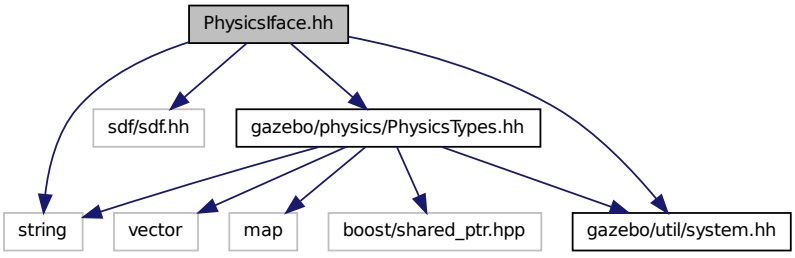
  Static physics registration macro.

Typedefs

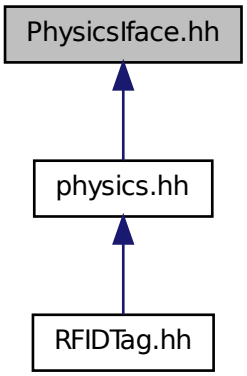
- typedef PhysicsEnginePtr(∗ **gazebo::physics::PhysicsFactoryFn**)(WorldPtr world)

11.152 PhysicsIface.hh File Reference

`#include <string>` `#include <sdf/sdf.hh>` `#include "gazebo/physics/-PhysicsTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for PhysicsIface.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

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

  Create a world given a name.
- **GAZEBO_VISIBLE** bool **gazebo::physics::fini** ()

  *Finalize transport by calling **gazebo::transport::fini*** (p. *113*).
- **GAZEBO_VISIBLE** WorldPtr **gazebo::physics::get_world** (const std::string &_name="")

  Returns a pointer to a world by name.
- **GAZEBO_VISIBLE** uint32_t **gazebo::physics::getUniqueId** ()

  Get a unique ID.
- **GAZEBO_VISIBLE** void **gazebo::physics::init_world** (WorldPtr _world)

Init world given a pointer to it.

- **GAZEBO_VISIBLE** void **gazebo::physics::init_worlds** ()

  initialize multiple worlds stored in static variable gazebo::g_worlds

- **GAZEBO_VISIBLE** bool **gazebo::physics::load** ()

  *Setup **gazebo::SystemPlugin** (p. 1359)'s and call **gazebo::transport::init** (p. 115).*

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

  Load world from sdf::Element pointer.

- **GAZEBO_VISIBLE** void **gazebo::physics::load_worlds** (sdf::ElementPtr _sdf)

  load multiple worlds from single sdf::Element pointer

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

  *Pause world by calling **World::SetPaused** (p. 1542).*

- **GAZEBO_VISIBLE** void **gazebo::physics::pause_worlds** (bool pause)

  pause multiple worlds stored in static variable gazebo::g_worlds

- **GAZEBO_VISIBLE** void **gazebo::physics::remove_worlds** ()

  remove multiple worlds stored in static variable gazebo::g_worlds

- **GAZEBO_VISIBLE** void **gazebo::physics::run_world** (WorldPtr _world, unsigned int _iterations=0)

  *Run world by calling **World::Run()** (p. 1541) given a pointer to it.*

- **GAZEBO_VISIBLE** void **gazebo::physics::run_worlds** (unsigned int _iterations=0)

  Run multiple worlds stored in static variable gazebo::g_worlds.

- **GAZEBO_VISIBLE** void **gazebo::physics::stop_world** (WorldPtr _world)

  *Stop world by calling **World::Stop()** (p. 1543) given a pointer to it.*

- **GAZEBO_VISIBLE** void **gazebo::physics::stop_worlds** ()

  stop multiple worlds stored in static variable gazebo::g_worlds

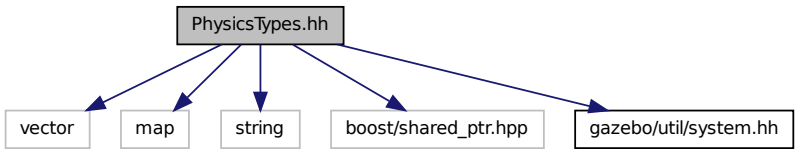
- **GAZEBO_VISIBLE** bool **gazebo::physics::worlds_running** ()

  Return true if any world is running.

11.153 PhysicsTypes.hh File Reference

default namespace for gazebo

`#include <vector>  #include <map>  #include <string>` × `#include <boost/shared_ptr.hpp>` `#include "gazebo/util/system.-hh"` 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

Defines

- #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 boost::shared_ptr< Actor > **gazebo::physics::ActorPtr**
- typedef std::vector< BasePtr > **gazebo::physics::Base_V**
- typedef boost::shared_ptr< Base > **gazebo::physics::BasePtr**
- typedef boost::shared_ptr < BoxShape > **gazebo::physics::BoxShapePtr**
- typedef std::vector< CollisionPtr > **gazebo::physics::Collision_V**
- typedef boost::shared_ptr < Collision > **gazebo::physics::CollisionPtr**
- typedef boost::shared_ptr < Contact > **gazebo::physics::ContactPtr**
- typedef boost::shared_ptr < CylinderShape > **gazebo::physics::CylinderShapePtr**
- typedef boost::shared_ptr< Entity > **gazebo::physics::EntityPtr**
- typedef boost::shared_ptr < Gripper > **gazebo::physics::GripperPtr**
- typedef boost::shared_ptr < HeightmapShape > **gazebo::physics::HeightmapShapePtr**
- typedef boost::shared_ptr < Inertial > **gazebo::physics::InertialPtr**
- typedef std::vector< JointPtr > **gazebo::physics::Joint_V**
- typedef std::vector < JointControllerPtr > **gazebo::physics::JointController_V**
- typedef boost::shared_ptr < JointController > **gazebo::physics::JointControllerPtr**
- typedef boost::shared_ptr< Joint > **gazebo::physics::JointPtr**
- typedef std::map< std::string, JointState > **gazebo::physics::JointState_M**
- typedef std::vector< LinkPtr > **gazebo::physics::Link_V**
- typedef boost::shared_ptr< Link > **gazebo::physics::LinkPtr**
- typedef std::map< std::string, LinkState > **gazebo::physics::LinkState_M**
- typedef boost::shared_ptr < MeshShape > **gazebo::physics::MeshShapePtr**
- typedef std::vector< ModelPtr > **gazebo::physics::Model_V**
- typedef boost::shared_ptr< Model > **gazebo::physics::ModelPtr**
- typedef std::map< std::string, ModelState > **gazebo::physics::ModelState_M**
- typedef boost::shared_ptr < MultiRayShape > **gazebo::physics::MultiRayShapePtr**
- typedef boost::shared_ptr < PhysicsEngine > **gazebo::physics::PhysicsEnginePtr**
- typedef boost::shared_ptr < RayShape > **gazebo::physics::RayShapePtr**
- typedef boost::shared_ptr< **Road** > **gazebo::physics::RoadPtr**
- typedef boost::shared_ptr< Shape > **gazebo::physics::ShapePtr**
- typedef boost::shared_ptr < SphereShape > **gazebo::physics::SphereShapePtr**
- typedef boost::shared_ptr < SurfaceParams > **gazebo::physics::SurfaceParamsPtr**
- typedef boost::shared_ptr< World > **gazebo::physics::WorldPtr**

11.153.1 Detailed Description

default namespace for gazebo

11.153.2 Define Documentation

11.153.2.1 #define GZ_ALL_COLLIDE 0x0FFFFFFF

Default collision bitmask.

Collision objects will collide with everything.

11.153.2.2 #define GZ_FIXED_COLLIDE 0x00000001

Collision object will collide only with fixed objects.

11.153.2.3 #define GZ_GHOST_COLLIDE 0x10000000

Collides with everything else but other ghost.

11.153.2.4 #define GZ_NONE_COLLIDE 0x00000000

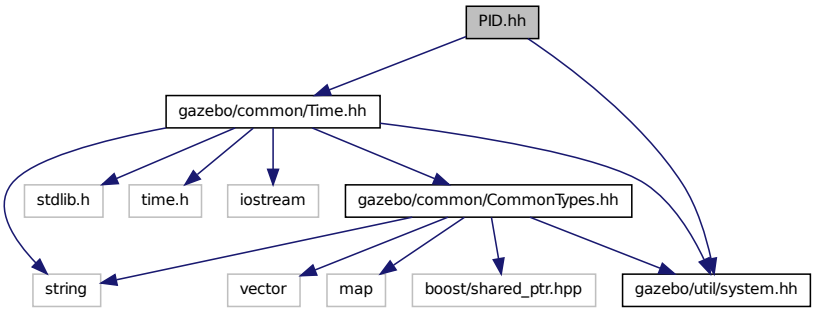
Collision object will collide with nothing.

11.153.2.5 #define GZ_SENSOR_COLLIDE 0x00000002

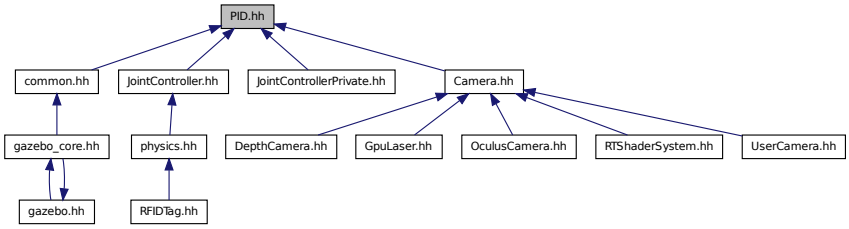
Collision object will collide only with sensors.

11.154 PID.hh File Reference

`#include "gazebo/common/Time.hh"` `#include "gazebo/util/system.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. 977) controller class.*

Namespaces

- namespace **gazebo**

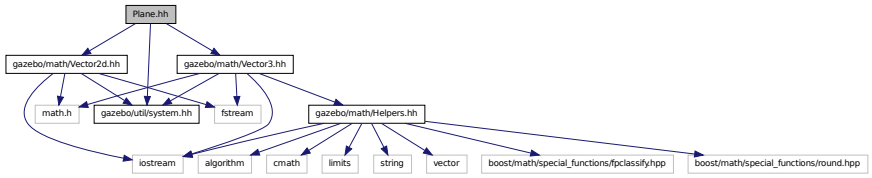
  Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

11.155 Plane.hh File Reference

`#include "gazebo/math/Vector3.hh"` `#include "gazebo/math/-Vector2d.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for Plane.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::math::Plane**

  A plane and related functions.

Namespaces

- namespace **gazebo**

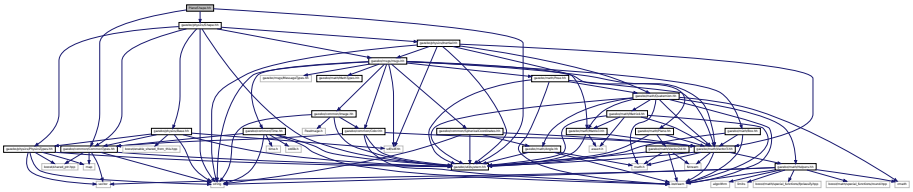
  Forward declarations for the common classes.

- namespace **gazebo::math**

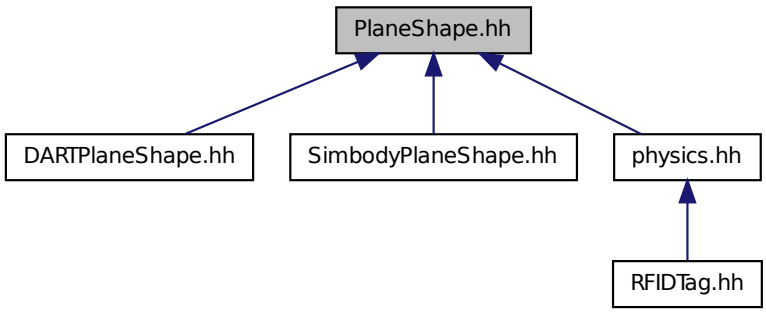
  Math namespace.

11.156 PlaneShape.hh File Reference

`#include "gazebo/common/CommonTypes.hh"` `#include "gazebo/physics/-Shape.hh"` `#include "gazebo/util/system.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. 295) for an infinite plane.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

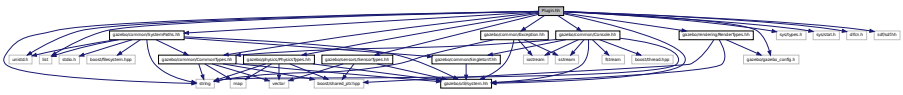
- namespace **gazebo::physics**

namespace for physics

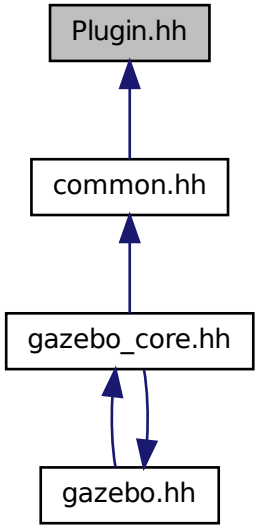
11.157 Plugin.hh File Reference

```
#include <unistd.h> #include <sys/types.h> #include <sys/stat.-
h> #include <gazebo/gazebo_config.h> #include <dlfcn.h>
#include <list> #include <string> #include <sdf/sdf.hh>
#include "gazebo/common/CommonTypes.hh" #include "gazebo/common/-
SystemPaths.hh" #include "gazebo/common/Console.hh" #include
"gazebo/common/Exception.hh" #include "gazebo/physics/-
PhysicsTypes.hh" #include "gazebo/sensors/SensorTypes.-
hh" #include "gazebo/rendering/RenderTypes.hh" #include
"gazebo/util/system.hh"
```

Include dependency graph for Plugin.hh:



This graph shows which files directly or indirectly include this file:



Classes

- union **gazebo::PluginT< T >::fptr_union_t**

  Pointer to shared library registration function definition.
- class **gazebo::ModelPlugin**

  A plugin with access to ***physics::Model*** *(p. 846).*
- class **gazebo::PluginT< T >**

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

  A plugin with access to physics::Sensor.
- class **gazebo::SystemPlugin**

  A plugin loaded within the gzserver on startup.
- class **gazebo::VisualPlugin**

  A plugin loaded within the gzserver on startup.
- class **gazebo::WorldPlugin**

  A plugin with access to ***physics::World*** *(p. 1529).*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

Defines

- #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.

11.157.1 Define Documentation

11.157.1.1 #define GZ_REGISTER_MODEL_PLUGIN(*classname*)

Value:

```
extern "C" GAZEBO_VISIBLE gazebo::ModelPlugin *RegisterPlugin(); \
  GAZEBO_VISIBLE \
  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

11.157.1.2 #define GZ_REGISTER_SENSOR_PLUGIN(*classname*)

Value:

```
extern "C" GAZEBO_VISIBLE gazebo::SensorPlugin *RegisterPlugin(); \
  GAZEBO_VISIBLE \
  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

11.157.1.3 #define GZ_REGISTER_SYSTEM_PLUGIN(*classname*)

Value:

```
extern "C" GAZEBO_VISIBLE gazebo::SystemPlugin *RegisterPlugin(); \
  GAZEBO_VISIBLE \
  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

11.157.1.4 #define GZ_REGISTER_VISUAL_PLUGIN(*classname*)

Value:

```
extern "C" GAZEBO_VISIBLE gazebo::VisualPlugin *RegisterPlugin(); \
  GAZEBO_VISIBLE \
  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

11.157.1.5 #define GZ_REGISTER_WORLD_PLUGIN(*classname*)

Value:

```
extern "C" GAZEBO_VISIBLE gazebo::WorldPlugin *RegisterPlugin(); \
  GAZEBO_VISIBLE \
  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.

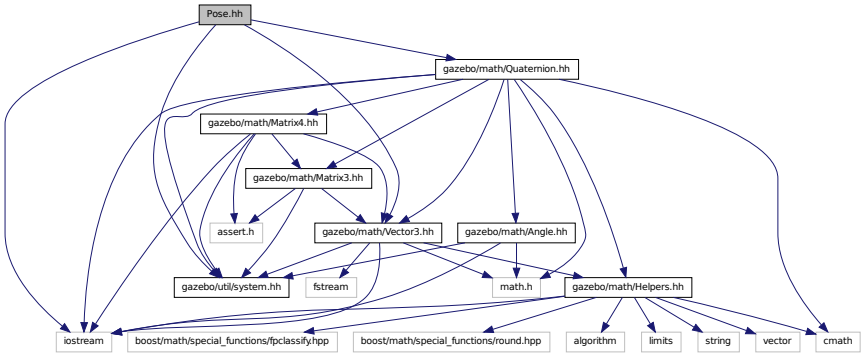
Returns

the name of the registered plugin

11.158 Pose.hh File Reference

```
#include <iostream>     #include "gazebo/math/Vector3.hh"
#include "gazebo/math/Quaternion.hh" #include "gazebo/util/system.-
```

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.

11.159 Projector.hh File Reference

```
#include <string> #include <map> #include <list> #include
<sdf/sdf.hh>    #include "gazebo/rendering/ogre_gazebo.h"
```

#include "gazebo/msgs/msgs.hh" #include "gazebo/transport/transport.-hh" #include "gazebo/rendering/RenderTypes.hh" #include "gazebo/util/system.hh" Include dependency graph for Projector.hh:

Classes

- class **gazebo::rendering::Projector**

  Projects a material onto surface, light a light projector.

- class **gazebo::rendering::Projector::ProjectorFrameListener**

Namespaces

- namespace **gazebo**

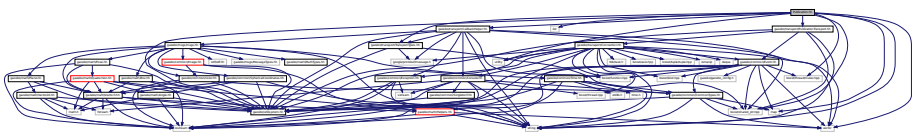
  Forward declarations for the common classes.

- namespace **gazebo::rendering**

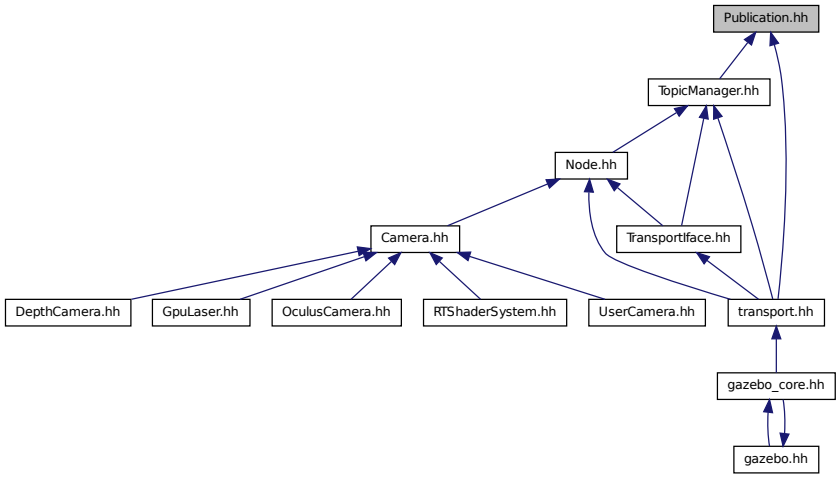
  Rendering namespace.

11.160 Publication.hh File Reference

#include <utility> #include <boost/shared_ptr.hpp> × #include <boost/thread/mutex.hpp> #include <list> #include <string> #include <vector> #include <map> #include "gazebo/transport/-CallbackHelper.hh" #include "gazebo/transport/Transport-Types.hh" #include "gazebo/transport/PublicationTransport.-hh" #include "gazebo/util/system.hh" Include dependency graph for Publication.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Publication**

  A publication for a topic.

Namespaces

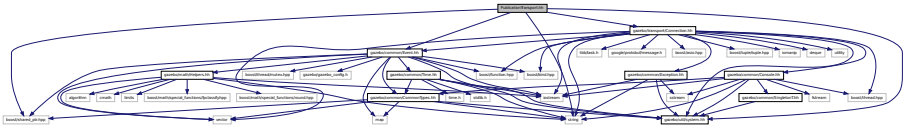
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::transport**

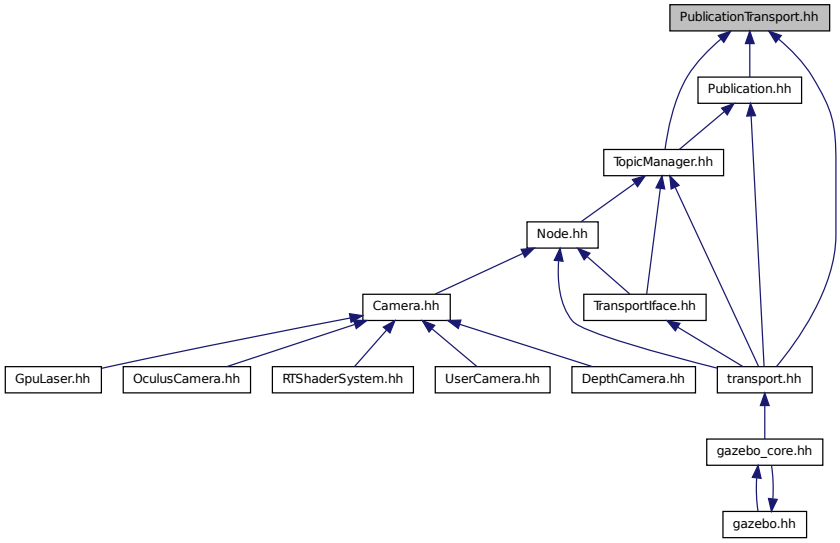
11.161 PublicationTransport.hh File Reference

`#include <boost/shared_ptr.hpp>` `#include <string>` `#include "gazebo/transport/Connection.hh"` `#include "gazebo/common/-Event.hh"` `#include "gazebo/util/system.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

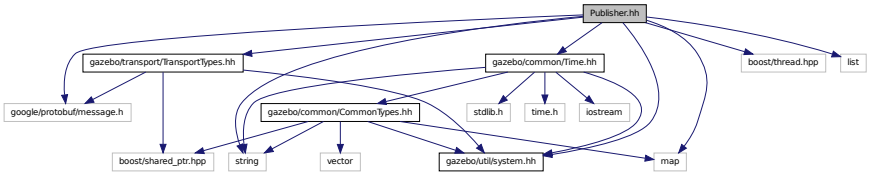
Namespaces

- namespace **gazebo**

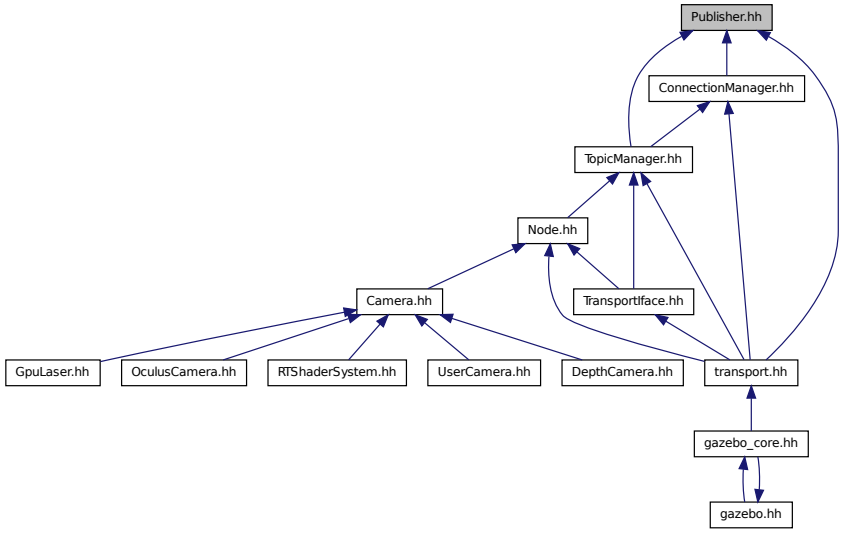
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.162 Publisher.hh File Reference

`#include <google/protobuf/message.h>` `#include <boost/thread.hpp>` `#include <string>` `#include <list>` `#include <map>` `#include "gazebo/common/Time.hh"` `#include "gazebo/transport/TransportTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for Publisher.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Publisher**

A publisher of messages on a topic.

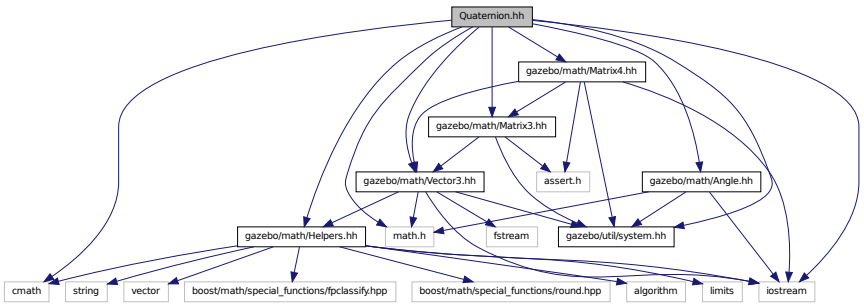
Namespaces

- namespace **gazebo**

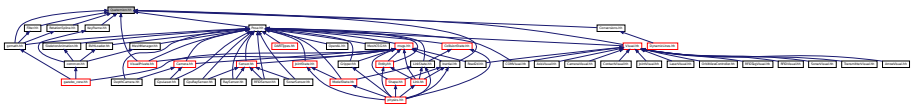
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.163 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 "gazebo/util/system.hh"` Include dependency graph for Quaternion.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Quaternion**

  A quaternion class.

Namespaces

- namespace **gazebo**

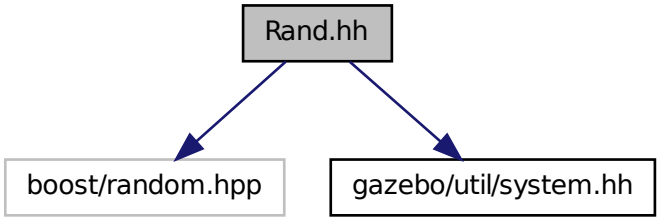
  Forward declarations for the common classes.

- namespace **gazebo::math**

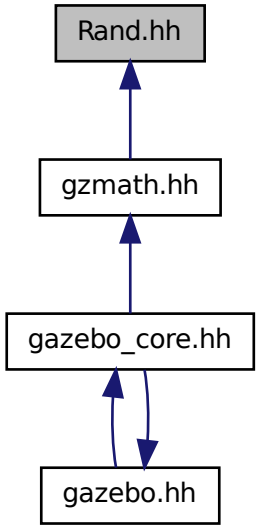
  Math namespace.

11.164 Rand.hh File Reference

`#include <boost/random.hpp>` `#include "gazebo/util/system.-hh"` 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.

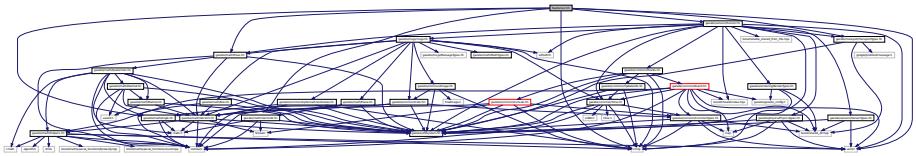
Typedefs

- 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**

11.165 RaySensor.hh File Reference

`#include <vector>` `#include <string>` `#include "gazebo/math/-Angle.hh"` `#include "gazebo/math/Pose.hh"` `#include "gazebo/transport/-TransportTypes.hh"` `#include "gazebo/sensors/Sensor.hh"` × `#include "gazebo/util/system.hh"` Include dependency graph for RaySensor.hh:



Classes

- class **gazebo::sensors::RaySensor**

  Sensor (p. *1130*) *with one or more rays.*

Namespaces

- namespace **gazebo**

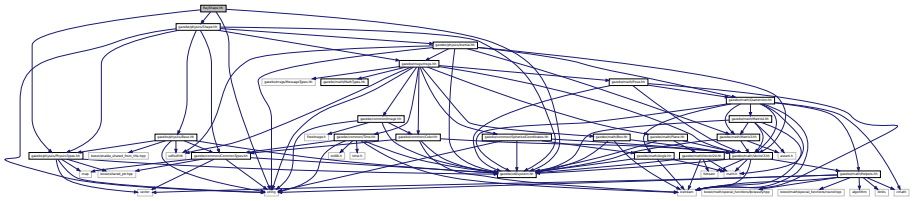
  Forward declarations for the common classes.

- namespace **gazebo::sensors**

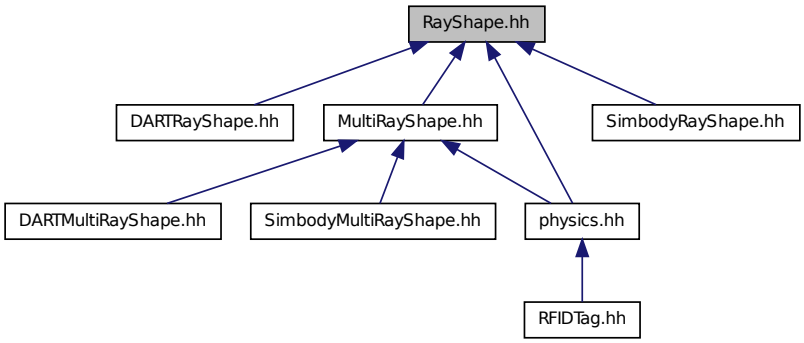
  Sensors namespace.

11.166 RayShape.hh File Reference

`#include <string> #include "gazebo/physics/PhysicsTypes.-hh" #include "gazebo/physics/Shape.hh" #include "gazebo/util/system.-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. 201) class for Ray collision geometry.*

Namespaces

- namespace **gazebo**

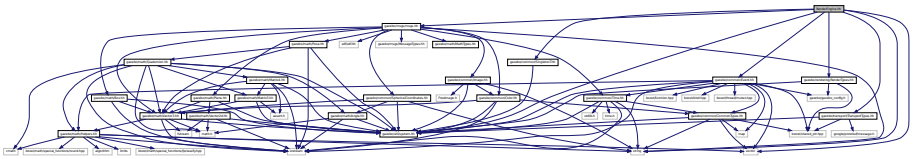
  Forward declarations for the common classes.

- namespace **gazebo::physics**

  namespace for physics

11.167 RenderEngine.hh File Reference

`#include <vector> #include <string> #include "gazebo/msgs/msgs.hh" #include "gazebo/common/SingletonT.hh" #include "gazebo/common/Event.hh" #include "gazebo/transport/TransportTypes.hh" #include "gazebo/rendering/RenderTypes.hh" #include "gazebo/util/system.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**

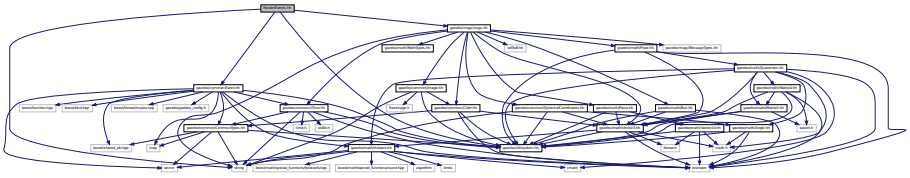
  Rendering namespace.

- namespace **Ogre**

11.168 RenderEvents.hh File Reference

`#include <string> #include "gazebo/common/Event.hh" #include "gazebo/msgs/msgs.hh" #include "gazebo/util/system.`

hh" Include dependency graph for RenderEvents.hh:



Classes

- class **gazebo::rendering::Events**

  Base class for rendering events.

Namespaces

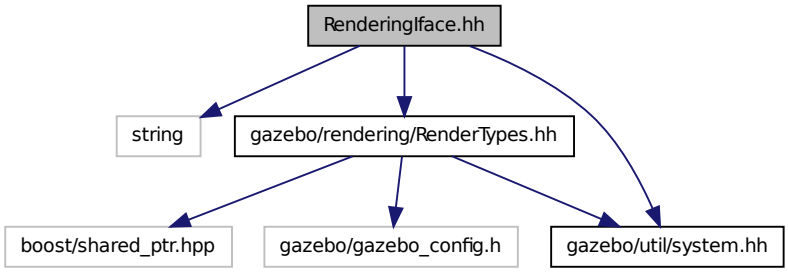
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.169 RenderingIface.hh File Reference

#include <string> #include "gazebo/rendering/RenderTypes.-hh" #include "gazebo/util/system.hh" Include dependency graph for RenderingIface.hh:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

Functions

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

  *create **rendering::Scene*** (p. *1097*) *by name.*
- **GAZEBO_VISIBLE** bool **gazebo::rendering::fini** ()

  teardown rendering engine.
- **GAZEBO_VISIBLE** rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name="")

  *get pointer to **rendering::Scene*** (p. *1097*) *by name.*
- **GAZEBO_VISIBLE** bool **gazebo::rendering::init** ()

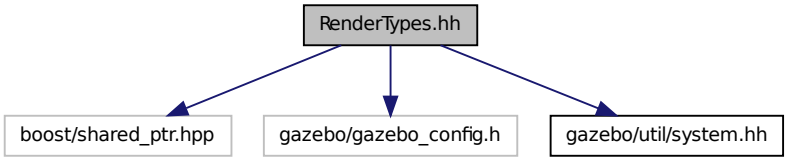
  init rendering engine.
- **GAZEBO_VISIBLE** bool **gazebo::rendering::load** ()

  load rendering engine.
- **GAZEBO_VISIBLE** void **gazebo::rendering::remove_scene** (const std::string &_name)

  *remove a **rendering::Scene*** (p. *1097*) *by name*

11.170 RenderTypes.hh File Reference

`#include <boost/shared_ptr.hpp>  #include "gazebo/gazebo_config.h" #include "gazebo/util/system.hh"` 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.

Defines

- #define **GZ_VISIBILITY_ALL** 0x0FFFFFFF

  Render everything visibility mask.
- #define **GZ_VISIBILITY_GUI** 0x00000001

  Render GUI visuals mask.
- #define **GZ_VISIBILITY_SELECTABLE** 0x00000002

  Render visuals that are selectable mask.
- #define **GZ_VISIBILITY_SELECTION** 0x10000000

  Renders only objects that can be selected.

Typedefs

- typedef boost::shared_ptr < ArrowVisual > **gazebo::rendering::ArrowVisualPtr**
- typedef boost::shared_ptr < AxisVisual > **gazebo::rendering::AxisVisualPtr**
- typedef boost::shared_ptr< Camera > **gazebo::rendering::CameraPtr**
- typedef boost::shared_ptr < CameraVisual > **gazebo::rendering::CameraVisualPtr**
- typedef boost::shared_ptr < COMVisual > **gazebo::rendering::COMVisualPtr**
- typedef boost::shared_ptr < ContactVisual > **gazebo::rendering::ContactVisualPtr**
- typedef boost::shared_ptr < DepthCamera > **gazebo::rendering::DepthCameraPtr**
- typedef boost::shared_ptr < DynamicLines > **gazebo::rendering::DynamicLinesPtr**
- typedef boost::shared_ptr < GpuLaser > **gazebo::rendering::GpuLaserPtr**
- typedef boost::shared_ptr < JointVisual > **gazebo::rendering::JointVisualPtr**

- typedef boost::shared_ptr < LaserVisual > **gazebo::rendering::LaserVisualPtr**
- typedef boost::shared_ptr< Light > **gazebo::rendering::LightPtr**
- typedef boost::shared_ptr < RFIDTagVisual > **gazebo::rendering::RFIDTagVisualPtr**
- typedef boost::shared_ptr < RFIDVisual > **gazebo::rendering::RFIDVisualPtr**
- typedef boost::shared_ptr< Scene > **gazebo::rendering::ScenePtr**
- typedef boost::shared_ptr < SelectionObj > **gazebo::rendering::SelectionObjPtr**
- typedef boost::shared_ptr < SonarVisual > **gazebo::rendering::SonarVisualPtr**
- typedef boost::shared_ptr < UserCamera > **gazebo::rendering::UserCameraPtr**
- typedef boost::shared_ptr< Visual > **gazebo::rendering::VisualPtr**
- typedef boost::shared_ptr < WindowManager > **gazebo::rendering::WindowManagerPtr**
- typedef boost::shared_ptr < WrenchVisual > **gazebo::rendering::WrenchVisualPtr**

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.

11.170.1 Define Documentation

11.170.1.1 #define GZ_VISIBILITY_ALL 0x0FFFFFFF

Render everything visibility mask.

11.170.1.2 #define GZ_VISIBILITY_GUI 0x00000001

Render GUI visuals mask.

11.170.1.3 #define GZ_VISIBILITY_SELECTABLE 0x00000002

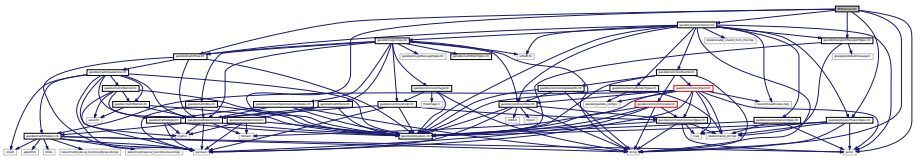
Render visuals that are selectable mask.

11.170.1.4 #define GZ_VISIBILITY_SELECTION 0x10000000

Renders only objects that can be selected.

11.171 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 "gazebo/util/system.hh"` Include dependency graph for RFIDSensor.hh:



Classes

- class **gazebo::sensors::RFIDSensor**

  Sensor (p. *1130*) *class for RFID type of sensor.*

Namespaces

- namespace **gazebo**

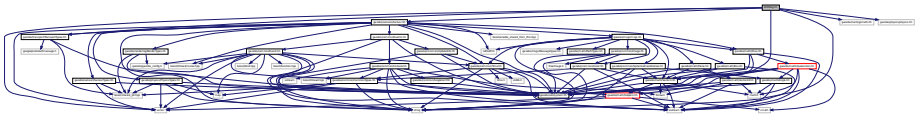
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.172 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 "gazebo/util/system.hh"` Include dependency graph for

RFIDTag.hh:



Classes

- class **gazebo::sensors::RFIDTag**

  RFIDTag *(p. 1075) to interact with RFIDTagSensors.*

Namespaces

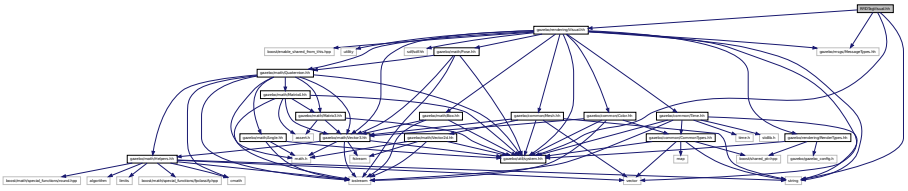
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.173 RFIDTagVisual.hh File Reference

`#include <string>  #include "gazebo/msgs/MessageTypes.hh" #include "gazebo/rendering/Visual.hh" #include "gazebo/util/system.hh"` Include dependency graph for RFIDTagVisual.hh:



Classes

- class **gazebo::rendering::RFIDTagVisual**

  Visualization for RFID tags sensor.

Namespaces

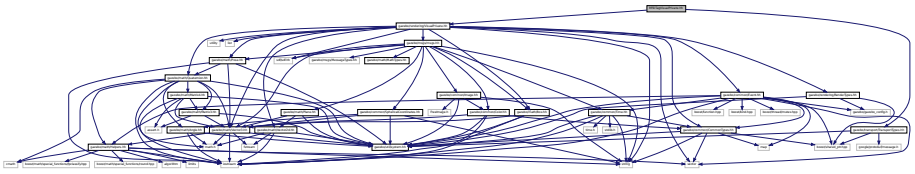
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.174 RFIDTagVisualPrivate.hh File Reference

`#include "gazebo/transport/TransportTypes.hh"` `#include "gazebo/rendering/VisualPrivate.hh"` Include dependency graph for RFIDTagVisualPrivate.hh:



Classes

- class **gazebo::rendering::RFIDTagVisualPrivate**

  *Private data for the RFID Tag **Visual*** (p. *1477*) *class.*

Namespaces

- namespace **gazebo**

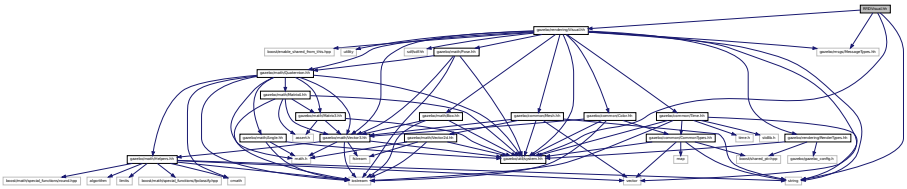
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.175 RFIDVisual.hh File Reference

`#include <string>` `#include "gazebo/msgs/MessageTypes.hh"` `#include "gazebo/rendering/Visual.hh"` `#include "gazebo/util/system.-`

hh" Include dependency graph for RFIDVisual.hh:



Classes

- class **gazebo::rendering::RFIDVisual**

  Visualization for RFID sensor.

Namespaces

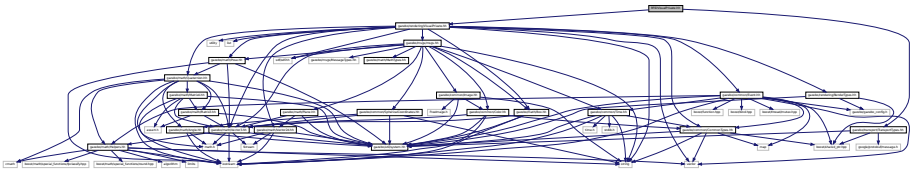
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.176 RFIDVisualPrivate.hh File Reference

#include "gazebo/transport/TransportTypes.hh" #include "gazebo/rendering/VisualPrivate.hh" Include dependency graph for RFIDVisualPrivate.hh:



Classes

- class **gazebo::rendering::RFIDVisualPrivate**

  *Private data for the RFID **Visual** (p. 1477) class.*

Namespaces

- namespace **gazebo**

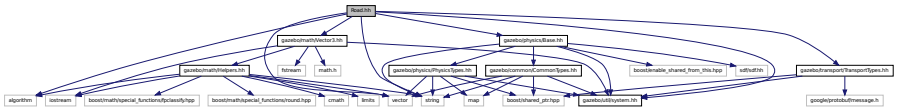
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

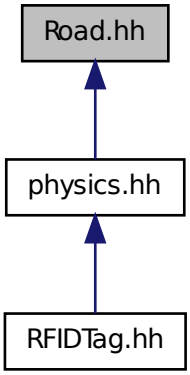
11.177 Road.hh File Reference

```
#include <string> #include <vector> #include <algorithm> ×
#include <gazebo/math/Vector3.hh> #include "gazebo/transport/-
TransportTypes.hh"     #include "gazebo/physics/Base.hh" ×
#include "gazebo/util/system.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. 1083) from SDF*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.178 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 "gazebo/util/system.-
hh"
```

Include dependency graph for Road2d.hh:

Classes

- class **gazebo::rendering::Road2d**
- class **gazebo::rendering::Road2d::Segment**

  A road segment.

Namespaces

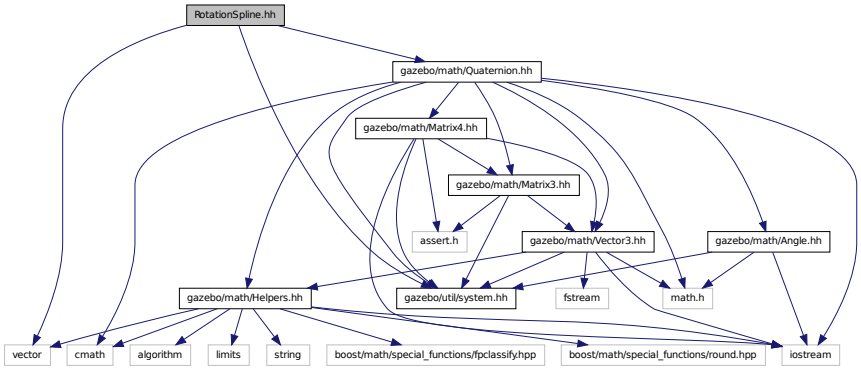
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

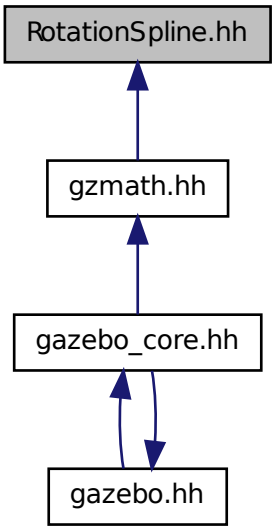
  Rendering namespace.

11.179 RotationSpline.hh File Reference

`#include <vector>  #include "gazebo/math/Quaternion.hh"` × `#include "gazebo/util/system.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. 1317) for rotations.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

11.180 RTShaderSystem.hh File Reference

```
#include <list>  #include <string>  #include <vector> ×
```

#include "gazebo/rendering/ogre_gazebo.h" #include "gazebo/gazebo-_config.h" #include "gazebo/rendering/Camera.hh" #include "gazebo/common/SingletonT.hh" #include "gazebo/util/system.-hh" Include dependency graph for RTShaderSystem.hh:

Classes

- class **gazebo::rendering::RTShaderSystem**

  *Implements **Ogre*** (p. *163*)*'s Run-Time Shader system.*

Namespaces

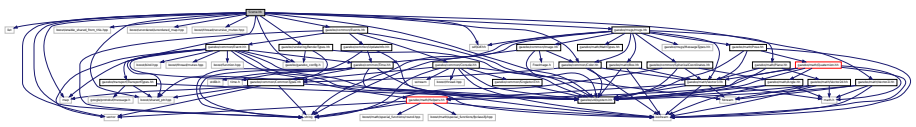
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

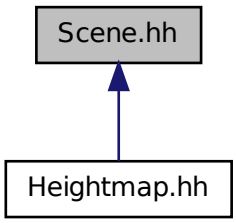
  Rendering namespace.

11.181 Scene.hh File Reference

#include <list> #include <map> #include <string> #include <vector> #include <boost/enable_shared_from_this.hpp> #include <boost/shared_ptr.hpp> #include <boost/unordered/unordered-_map.hpp> #include <boost/thread/recursive_mutex.hpp> × #include <sdf/sdf.hh> #include "gazebo/common/Events.hh" #include "gazebo/common/Color.hh" #include "gazebo/gazebo-_config.h" #include "gazebo/math/Vector2i.hh" #include "gazebo/msgs/msgs.hh" #include "gazebo/rendering/Render-Types.hh" #include "gazebo/transport/TransportTypes.hh" #include "gazebo/util/system.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**

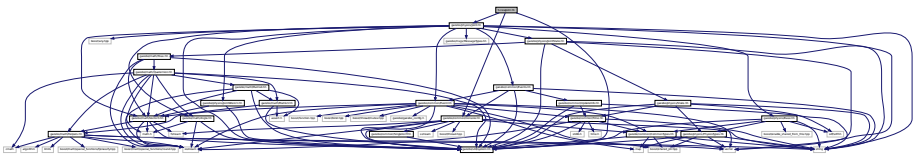
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**
- namespace **SkyX**

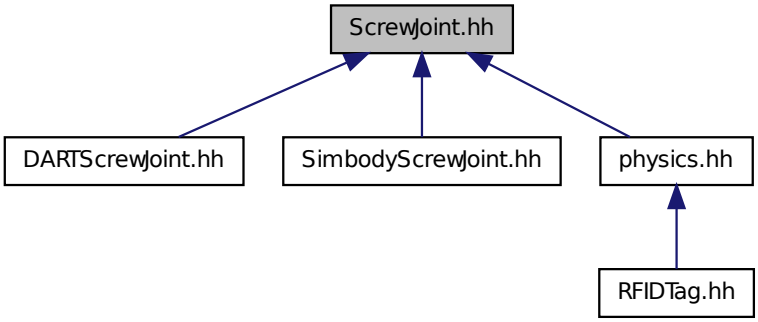
11.182 ScrewJoint.hh File Reference

`#include "gazebo/physics/Joint.hh"` `#include "gazebo/common/-Console.hh"` `#include "gazebo/util/system.hh"` Include dependency

graph for ScrewJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

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

  A screw joint, which has both prismatic and rotational DOFs.

Namespaces

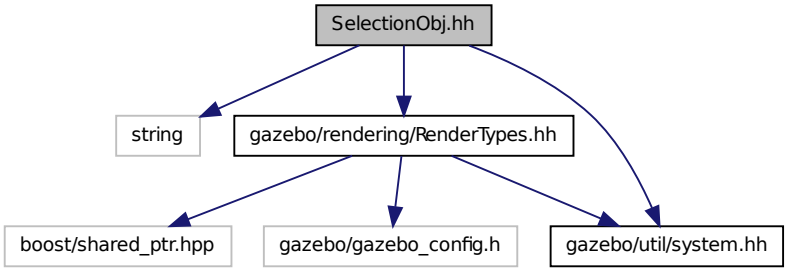
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

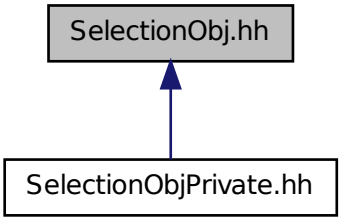
  namespace for physics

11.183 SelectionObj.hh File Reference

`#include <string>` `#include "gazebo/rendering/RenderTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SelectionObj.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::SelectionObj**

  Interactive selection object for models and links.

Namespaces

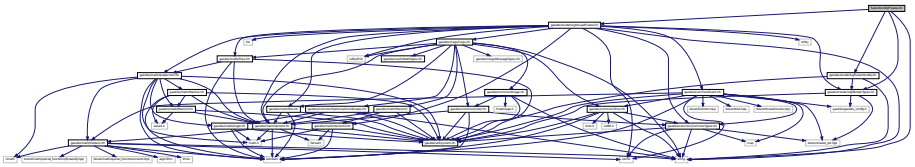
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.184 SelectionObjPrivate.hh File Reference

`#include <string>` `#include "gazebo/rendering/RenderTypes.hh"` `#include "gazebo/rendering/VisualPrivate.hh"` `#include "gazebo/rendering/SelectionObj.hh"` Include dependency graph for SelectionObjPrivate.hh:



Classes

- class **gazebo::rendering::SelectionObjPrivate**

  Private data for the Selection Obj class.

Namespaces

- namespace **gazebo**

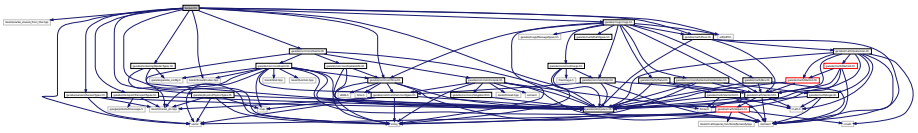
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.185 Sensor.hh File Reference

`#include <boost/enable_shared_from_this.hpp>` `#include <boost/thread/mutex.hpp>` `#include <vector>` `#include <string>` `#include <sdf/sdf.hh>` `#include "gazebo/physics/PhysicsTypes.hh"` `#include "gazebo/rendering/RenderTypes.hh"`

`#include "gazebo/sensors/SensorTypes.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 "gazebo/util/system.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.

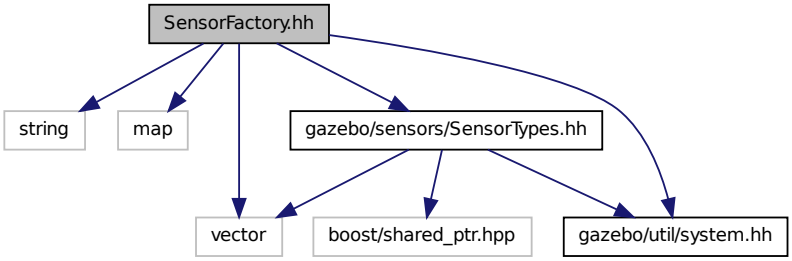
Enumerations

- 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.

11.186 SensorFactory.hh File Reference

`#include <string>  #include <map>  #include <vector>` × `#include "gazebo/sensors/SensorTypes.hh"` `#include "gazebo/util/system.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.

Defines

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

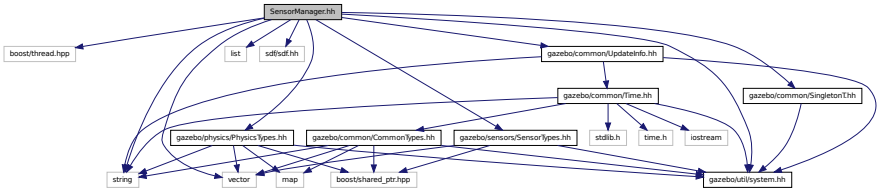
  Static sensor registration macro.

Typedefs

- typedef Sensor ∗(∗ **gazebo::sensors::SensorFactoryFn**)()

11.187 SensorManager.hh File Reference

`#include <boost/thread.hpp> #include <string> #include <vector> #include <list> #include <sdf/sdf.hh> #include "gazebo/physics/PhysicsTypes.hh" #include "gazebo/common/SingletonT.hh" #include "gazebo/common/UpdateInfo.hh" #include "gazebo/sensors/SensorTypes.hh" #include "gazebo/util/system.hh"` Include dependency graph for SensorManager.hh:



Classes

- class **gazebo::sensors::SensorManager::ImageSensorContainer**
- class **gazebo::sensors::SensorManager::SensorContainer**
- 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

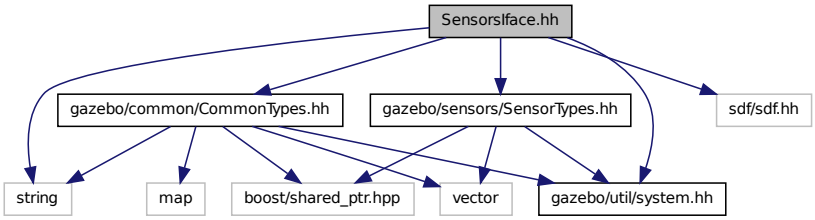
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.188 SensorsIface.hh File Reference

`#include <string>` `#include <sdf/sdf.hh>` `#include "gazebo/common/-CommonTypes.hh"` `#include "gazebo/sensors/SensorTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for -SensorsIface.hh:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

Functions

- **GAZEBO_VISIBLE** std::string **gazebo::sensors::create_sensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName, uint32_t _parentId)

  Create a sensor using SDF.
- **GAZEBO_VISIBLE** void **gazebo::sensors::disable** ()

  Disable sensors.
- **GAZEBO_VISIBLE** void **gazebo::sensors::enable** ()

  Enable sensors.
- **GAZEBO_VISIBLE** bool **gazebo::sensors::fini** ()

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

  Get a sensor using by name.

- **GAZEBO_VISIBLE** bool **gazebo::sensors::init** ()

  initialize the sensor generation loop.

- **GAZEBO_VISIBLE** bool **gazebo::sensors::load** ()

  Load the sensor library.

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

  Remove a sensor by name.

- **GAZEBO_VISIBLE** bool **gazebo::sensors::remove_sensors** ()

  Remove all sensors.

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

  Run the sensor generation one step.

- **GAZEBO_VISIBLE** void **gazebo::sensors::run_threads** ()

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

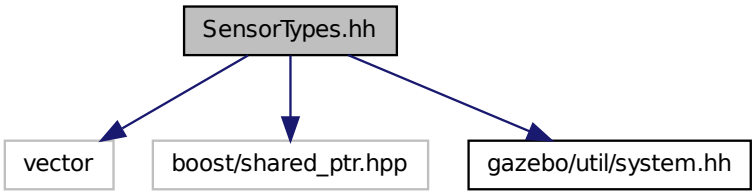
- **GAZEBO_VISIBLE** void **gazebo::sensors::stop** ()

  Stop the sensor generation loop.

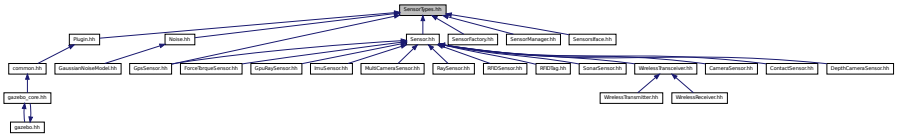
11.189 SensorTypes.hh File Reference

Forward declarations and typedefs for sensors.

`#include <vector>` `#include <boost/shared_ptr.hpp>` `#include "gazebo/util/system.hh"` 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 boost::shared_ptr < CameraSensor > **gazebo::sensors::CameraSensorPtr**
- typedef std::vector < ContactSensorPtr > **gazebo::sensors::ContactSensor_V**
- typedef boost::shared_ptr < ContactSensor > **gazebo::sensors::ContactSensorPtr**
- typedef std::vector < DepthCameraSensorPtr > **gazebo::sensors::DepthCameraSensor_V**
- typedef boost::shared_ptr < DepthCameraSensor > **gazebo::sensors::DepthCameraSensorPtr**
- typedef boost::shared_ptr < ForceTorqueSensor > **gazebo::sensors::ForceTorqueSensorPtr**
- typedef boost::shared_ptr < GaussianNoiseModel > **gazebo::sensors::GaussianNoiseModelPtr**
- typedef boost::shared_ptr < GpsSensor > **gazebo::sensors::GpsSensorPtr**
- typedef std::vector < GpuRaySensorPtr > **gazebo::sensors::GpuRaySensor_V**
- typedef boost::shared_ptr < GpuRaySensor > **gazebo::sensors::GpuRaySensorPtr**
- typedef boost::shared_ptr < ImageGaussianNoiseModel > **gazebo::sensors::ImageGaussianNoiseModelPtr**

  *Shared pointer to **Noise** (p. 931).*
- typedef std::vector< ImuSensorPtr > **gazebo::sensors::ImuSensor_V**

- typedef boost::shared_ptr < ImuSensor > **gazebo::sensors::ImuSensorPtr**
- typedef std::vector < MultiCameraSensorPtr > **gazebo::sensors::MultiCameraSensor_V**
- typedef boost::shared_ptr < MultiCameraSensor > **gazebo::sensors::MultiCameraSensorPtr**
- typedef boost::shared_ptr< Noise > **gazebo::sensors::NoisePtr**
- typedef std::vector< RaySensorPtr > **gazebo::sensors::RaySensor_V**
- typedef boost::shared_ptr < RaySensor > **gazebo::sensors::RaySensorPtr**
- typedef std::vector< RFIDSensor > **gazebo::sensors::RFIDSensor_V**
- typedef boost::shared_ptr < RFIDSensor > **gazebo::sensors::RFIDSensorPtr**
- typedef std::vector< RFIDTag > **gazebo::sensors::RFIDTag_V**
- typedef boost::shared_ptr < RFIDTag > **gazebo::sensors::RFIDTagPtr**
- typedef std::vector< SensorPtr > **gazebo::sensors::Sensor_V**
- typedef boost::shared_ptr< Sensor > **gazebo::sensors::SensorPtr**
- typedef boost::shared_ptr < SonarSensor > **gazebo::sensors::SonarSensorPtr**
- typedef std::vector < WirelessReceiver > **gazebo::sensors::WirelessReceiver_V**
- typedef boost::shared_ptr < WirelessReceiver > **gazebo::sensors::WirelessReceiverPtr**
- typedef std::vector < WirelessTransceiver > **gazebo::sensors::WirelessTransceiver_V**
- typedef boost::shared_ptr < WirelessTransceiver > **gazebo::sensors::WirelessTransceiverPtr**
- typedef std::vector < WirelessTransmitter > **gazebo::sensors::WirelessTransmitter_V**
- typedef boost::shared_ptr < WirelessTransmitter > **gazebo::sensors::WirelessTransmitterPtr**

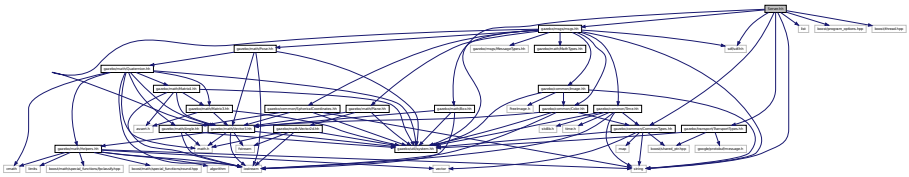
11.189.1 Detailed Description

Forward declarations and typedefs for sensors.

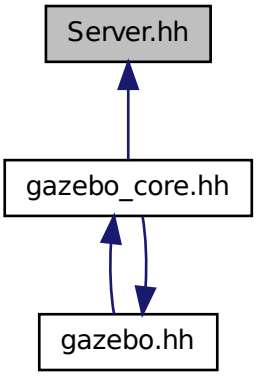
11.190 Server.hh File Reference

```
#include <string> #include <list> #include <boost/program-
_options.hpp> #include <boost/thread.hpp> #include <sdf/sdf.-
hh> #include "gazebo/msgs/msgs.hh" #include "gazebo/transport/-
TransportTypes.hh"  #include "gazebo/common/CommonTypes.-
```

hh" #include "gazebo/util/system.hh" Include dependency graph for Server.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Server**

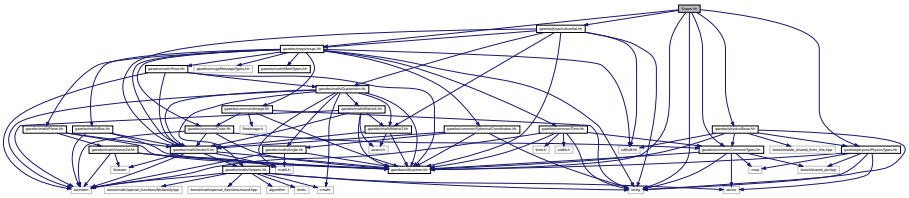
Namespaces

- namespace **boost**
- namespace **gazebo**

  Forward declarations for the common classes.

11.191 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 "gazebo/util/system.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. *201) class for all shapes.*

Namespaces

- namespace **gazebo**

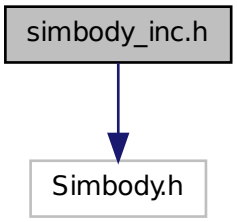
  Forward declarations for the common classes.

- namespace **gazebo::physics**

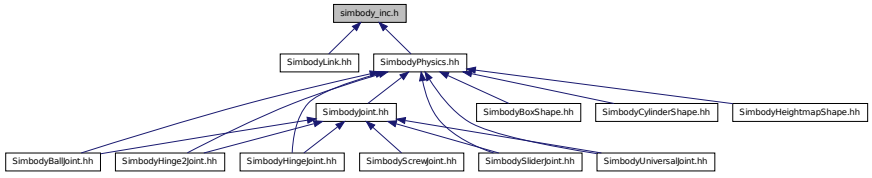
  namespace for physics

11.192 simbody_inc.h File Reference

`#include <Simbody.h>` Include dependency graph for simbody_inc.h:

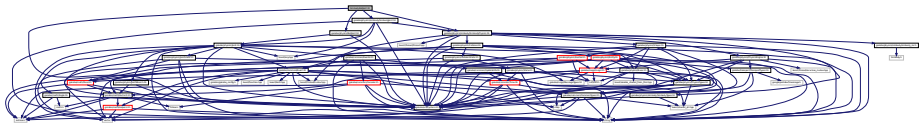


This graph shows which files directly or indirectly include this file:



11.193 SimbodyBallJoint.hh File Reference

`#include "gazebo/physics/BallJoint.hh"` `#include "gazebo/physics/simbody/SimbodyJoint.hh"` `#include "gazebo/physics/simbody/SimbodyPhysics.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodyBallJoint.hh:



Classes

- class **gazebo::physics::SimbodyBallJoint**

  SimbodyBallJoint *(p. 1165) class models a ball joint in Simbody.*

Namespaces

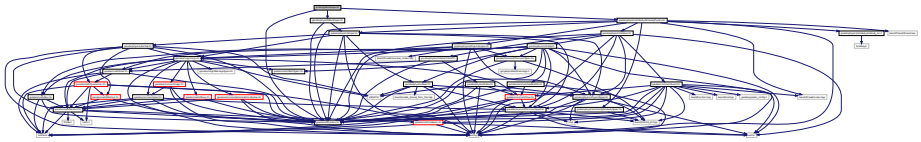
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.194 SimbodyBoxShape.hh File Reference

`#include "gazebo/physics/simbody/SimbodyPhysics.hh"` `#include "gazebo/physics/BoxShape.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodyBoxShape.hh:



Classes

- class **gazebo::physics::SimbodyBoxShape**

  Simbody box collision.

Namespaces

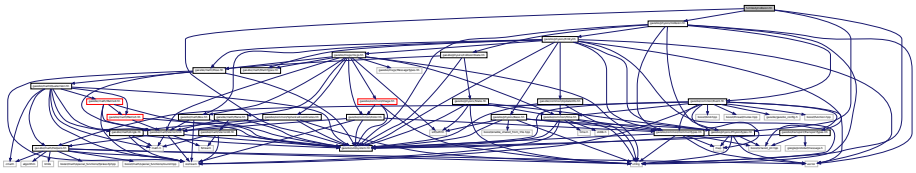
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.195 SimbodyCollision.hh File Reference

`#include <string> #include "gazebo/physics/PhysicsTypes.-hh" #include "gazebo/physics/Collision.hh" #include "gazebo/util/system.-hh"` Include dependency graph for SimbodyCollision.hh:



Classes

- class **gazebo::physics::SimbodyCollision**

  Simbody collisions.

Namespaces

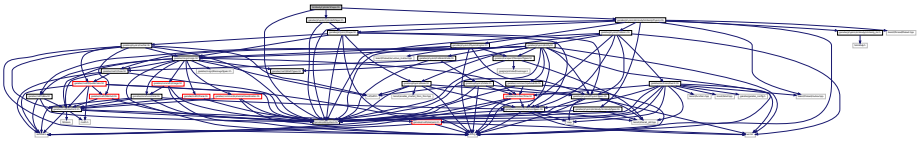
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics
- namespace **SimTK**

11.196 SimbodyCylinderShape.hh File Reference

`#include "gazebo/physics/simbody/SimbodyPhysics.hh" #include "gazebo/physics/CylinderShape.hh" #include "gazebo/util/system.-hh"` Include dependency graph for SimbodyCylinderShape.hh:



Classes

- class **gazebo::physics::SimbodyCylinderShape**

  Cylinder collision.

Namespaces

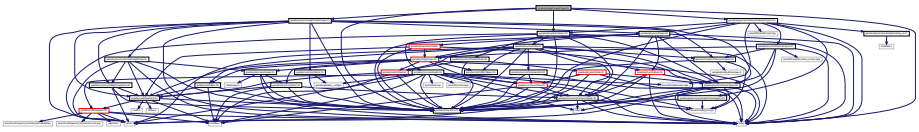
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.197 SimbodyHeightmapShape.hh File Reference

`#include <string>` `#include "gazebo/physics/HeightmapShape.hh"` `#include "gazebo/physics/simbody/SimbodyPhysics.hh"` `#include "gazebo/physics/Collision.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodyHeightmapShape.hh:



Classes

- class **gazebo::physics::SimbodyHeightmapShape**

  Height map collision.

Namespaces

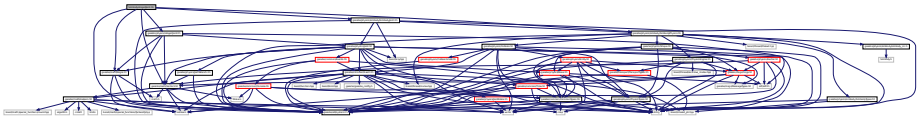
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.198 SimbodyHinge2Joint.hh File Reference

`#include "gazebo/math/Angle.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/physics/Hinge2Joint.hh"` `#include "gazebo/physics/simbody/SimbodyJoint.hh"` `#include "gazebo/physics/simbody/SimbodyPhysics.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodyHinge2Joint.hh:



Classes

- class **gazebo::physics::SimbodyHinge2Joint**

  A two axis hinge joint.

Namespaces

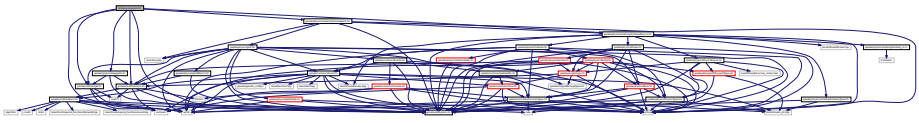
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.199 SimbodyHingeJoint.hh File Reference

`#include <vector>` `#include "gazebo/math/Angle.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/physics/HingeJoint.hh"` `#include "gazebo/physics/simbody/SimbodyJoint.hh"` `#include "gazebo/physics/simbody/SimbodyPhysics.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodyHingeJoint.hh:



Classes

- class **gazebo::physics::SimbodyHingeJoint**

  A single axis hinge joint.

Namespaces

- namespace **gazebo**

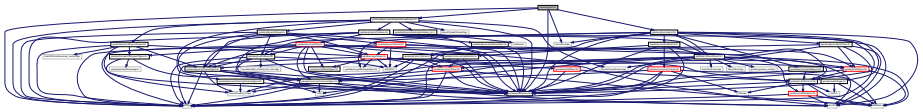
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

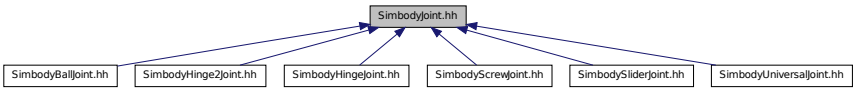
11.200 SimbodyJoint.hh File Reference

```
#include <boost/any.hpp> #include <string> #include "gazebo/physics/simbody/
SimbodyPhysics.hh"    #include "gazebo/physics/Joint.hh" ×
#include "gazebo/util/system.hh"
```

Include dependency graph for - SimbodyJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SimbodyJoint**

  Base *(*p. *201) class for all joints.*

Namespaces

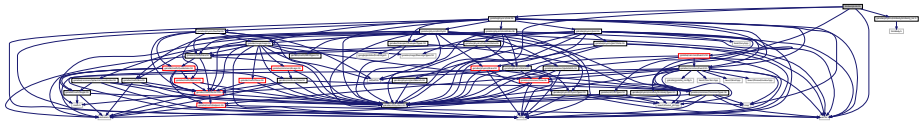
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.201 SimbodyLink.hh File Reference

`#include <vector>` `#include "gazebo/physics/simbody/SimbodyTypes.hh"` `#include "gazebo/physics/Link.hh"` `#include "gazebo/physics/simbody/simbody_inc.h"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodyLink.hh:



Classes

- class **gazebo::physics::SimbodyLink**

  *Simbody **Link*** (p. *739*) *class.*

Namespaces

- namespace **gazebo**

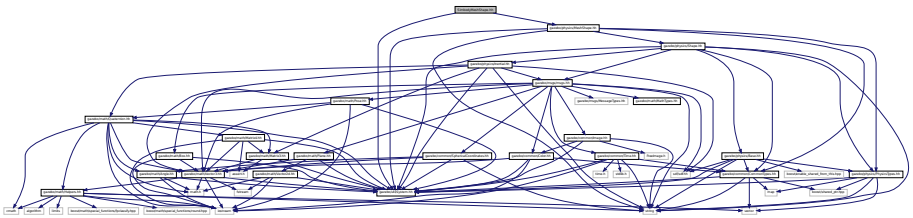
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.202 SimbodyMeshShape.hh File Reference

`#include "gazebo/physics/MeshShape.hh"` `#include "gazebo/util/system.-`

hh" Include dependency graph for SimbodyMeshShape.hh:



Classes

- class **gazebo::physics::SimbodyMeshShape**

  Triangle mesh collision.

Namespaces

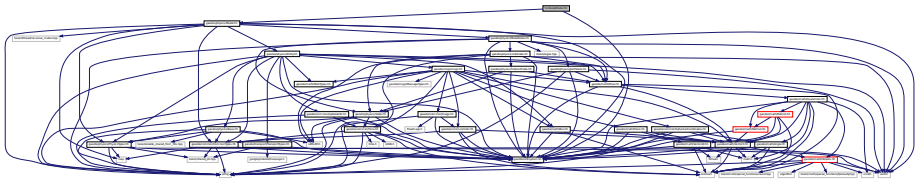
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.203 SimbodyModel.hh File Reference

#include "gazebo/physics/Model.hh" #include "gazebo/util/system.-hh" Include dependency graph for SimbodyModel.hh:



Classes

- class **gazebo::physics::SimbodyModel**

  A model is a collection of links, joints, and plugins.

Namespaces

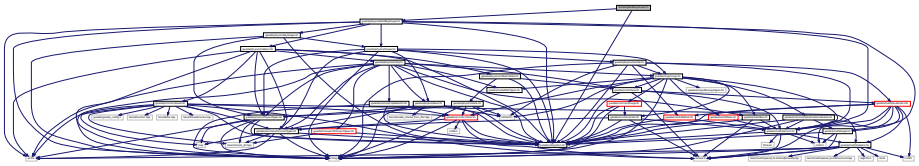
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.204 SimbodyMultiRayShape.hh File Reference

`#include "gazebo/physics/MultiRayShape.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodyMultiRayShape.hh:



Classes

- class **gazebo::physics::SimbodyMultiRayShape**

  Simbody specific version of ***MultiRayShape*** *(*p. *901).*

Namespaces

- namespace **gazebo**

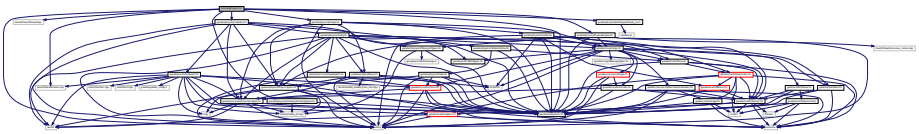
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

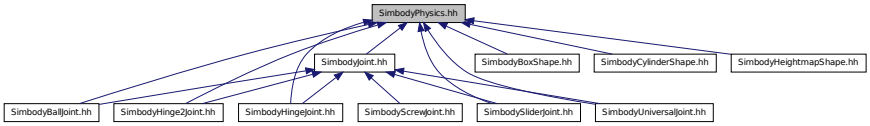
11.205 SimbodyPhysics.hh File Reference

`#include <string>` `#include <boost/thread/thread.hpp>` `#include <boost/thread/mutex.hpp>` `#include "gazebo/physics/PhysicsEngine.hh"` `#include "gazebo/physics/Collision.hh"`

`#include "gazebo/physics/Shape.hh"` `#include "gazebo/physics/simbody/-SimbodyTypes.hh"` `#include "gazebo/physics/simbody/simbody-_inc.h"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodyPhysics.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SimbodyPhysics**

  Simbody physics engine.

Namespaces

- namespace **gazebo**

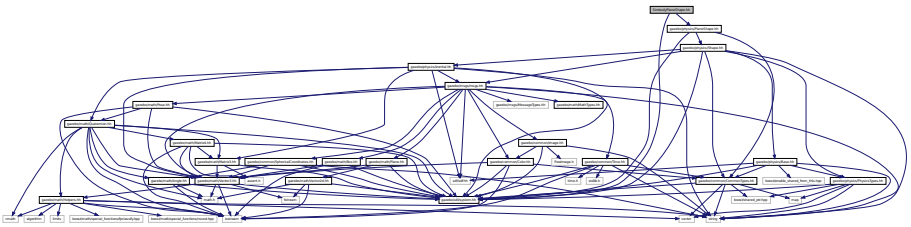
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.206 SimbodyPlaneShape.hh File Reference

`#include "gazebo/physics/PlaneShape.hh"` `#include "gazebo/util/system.-`

hh" Include dependency graph for SimbodyPlaneShape.hh:



Classes

- class **gazebo::physics::SimbodyPlaneShape**

  Simbody collision for an infinite plane.

Namespaces

- namespace **gazebo**

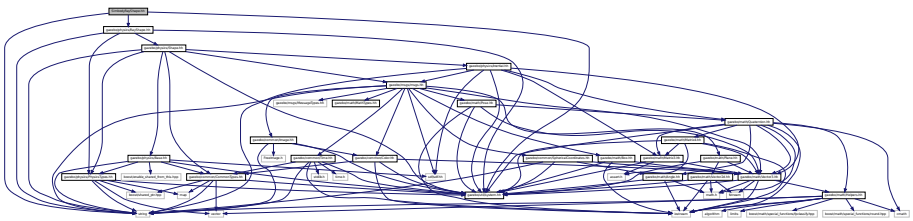
  Forward declarations for the common classes.

- namespace **gazebo::physics**

  namespace for physics

11.207 SimbodyRayShape.hh File Reference

#include <string> #include "gazebo/physics/RayShape.hh" #include "gazebo/util/system.hh" Include dependency graph for - SimbodyRayShape.hh:



Classes

- class **gazebo::physics::SimbodyRayShape**

  Ray shape for simbody.

Namespaces

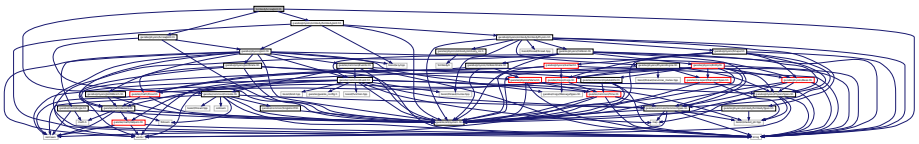
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.208 SimbodyScrewJoint.hh File Reference

`#include <string>` `#include "gazebo/physics/simbody/SimbodyJoint.hh"` `#include "gazebo/physics/ScrewJoint.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodyScrewJoint.hh:



Classes

- class **gazebo::physics::SimbodyScrewJoint**

  A screw joint.

Namespaces

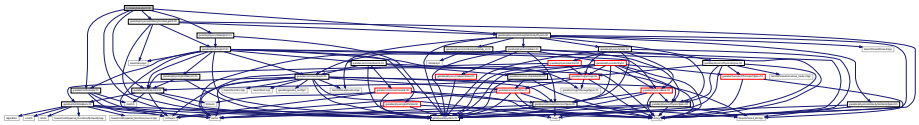
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.209 SimbodySliderJoint.hh File Reference

`#include "gazebo/math/Angle.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/physics/simbody/SimbodyJoint.hh"` `#include "gazebo/physics/SliderJoint.hh"` `#include "gazebo/physics/simbody/SimbodyPhysics.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodySliderJoint.hh:



Classes

- class **gazebo::physics::SimbodySliderJoint**

  A slider joint.

Namespaces

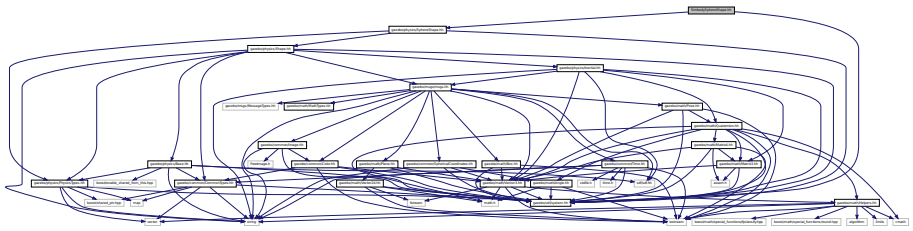
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.210 SimbodySphereShape.hh File Reference

`#include "gazebo/physics/SphereShape.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodySphereShape.hh:



Classes

- class **gazebo::physics::SimbodySphereShape**

  Simbody sphere collision.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

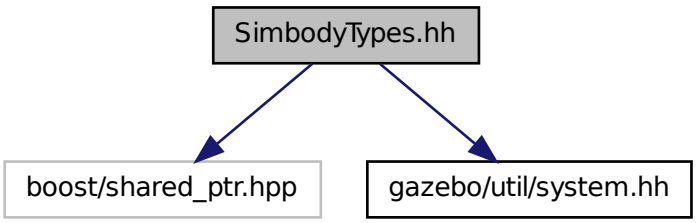
- namespace **gazebo::physics**

  namespace for physics

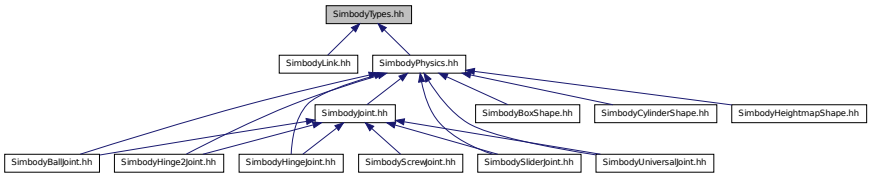
11.211 SimbodyTypes.hh File Reference

Simbody wrapper forward declarations and typedefs.

`#include <boost/shared_ptr.hpp>` `#include "gazebo/util/system.hh"` Include dependency graph for SimbodyTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Typedefs

- typedef boost::shared_ptr < SimbodyCollision > **gazebo::physics::SimbodyCollisionPtr**
- typedef boost::shared_ptr < SimbodyLink > **gazebo::physics::SimbodyLinkPtr**
- typedef boost::shared_ptr < SimbodyModel > **gazebo::physics::SimbodyModelPtr**
- typedef boost::shared_ptr < SimbodyPhysics > **gazebo::physics::SimbodyPhysicsPtr**
- typedef boost::shared_ptr < SimbodyRayShape > **gazebo::physics::SimbodyRayShapePtr**

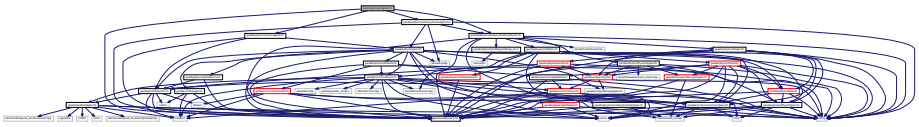
11.211.1 Detailed Description

Simbody wrapper forward declarations and typedefs.

11.212 SimbodyUniversalJoint.hh File Reference

```
#include "gazebo/physics/UniversalJoint.hh" #include "gazebo/physics/simbody/-
SimbodyJoint.hh" #include "gazebo/physics/simbody/Simbody-
```

`Physics.hh" #include "gazebo/util/system.hh"` Include dependency graph for SimbodyUniversalJoint.hh:



Classes

- class **gazebo::physics::SimbodyUniversalJoint**

  A simbody universal joint class.

Namespaces

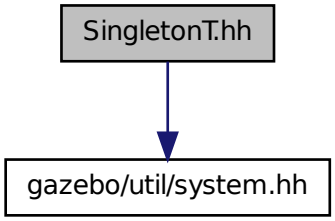
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

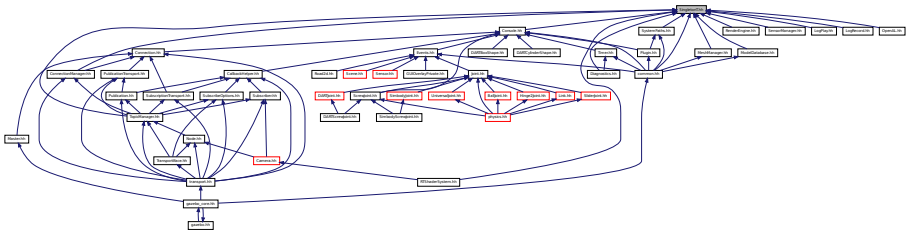
  namespace for physics

11.213 SingletonT.hh File Reference

`#include "gazebo/util/system.hh"` Include dependency graph for - SingletonT.hh:



This graph shows which files directly or indirectly include this file:



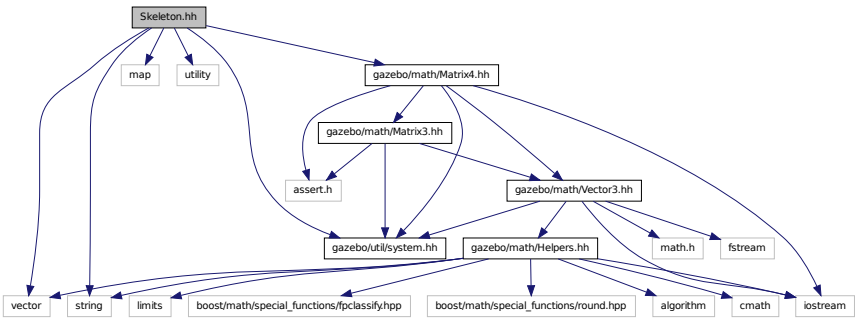
Classes

- class **SingletonT**< **T** >

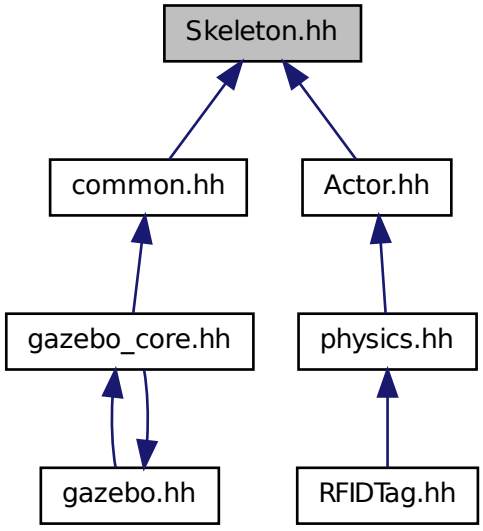
  Singleton template class.

11.214 Skeleton.hh File Reference

`#include <vector>` `#include <string>` `#include <map>` × `#include <utility>` `#include "gazebo/math/Matrix4.hh"` × `#include "gazebo/util/system.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. *925)* ***Skeleton.hh*** *(*p. *1787) common/common.hh*
- class **gazebo::common::Skeleton**

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

  A skeleton node.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

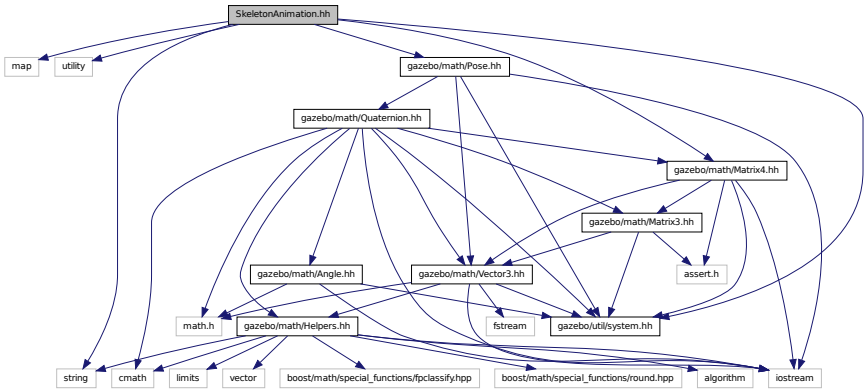
  Common namespace.

Typedefs

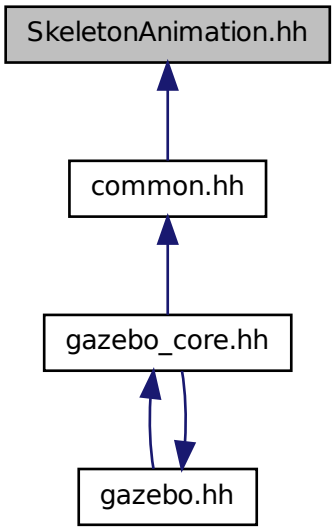
- typedef std::map< unsigned int, SkeletonNode * > **gazebo::common::NodeMap**
- typedef std::map< unsigned int, SkeletonNode * >::iterator **gazebo::common::NodeMapIter**
- typedef std::map< double, std::vector< NodeTransform > > **gazebo::common::RawNodeAnim**
- typedef std::vector < std::vector< std::pair < std::string, double > > > **gazebo::common::RawNodeWeights**
- typedef std::map< std::string, RawNodeAnim > **gazebo::common::RawSkeletonAnim**

11.215 SkeletonAnimation.hh File Reference

`#include <map>` `#include <utility>` `#include <string>` × `#include "gazebo/math/Matrix4.hh"` `#include "gazebo/math/Pose.hh"` `#include "gazebo/util/system.hh"` 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. 1269) animation.*

Namespaces

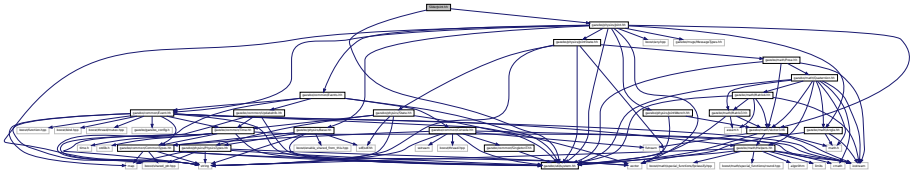
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

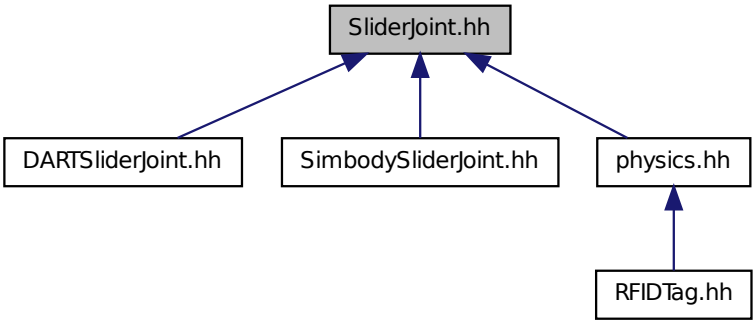
  Common namespace.

11.216 SliderJoint.hh File Reference

`#include "gazebo/physics/Joint.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SliderJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

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

  A slider joint.

Namespaces

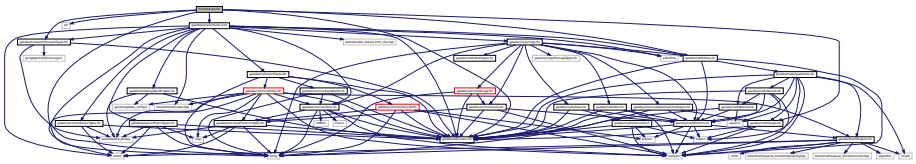
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

namespace for physics

11.217 SonarSensor.hh File Reference

`#include <string> #include <list> #include "gazebo/math/-Angle.hh" #include "gazebo/math/Pose.hh" #include "gazebo/transport/-TransportTypes.hh" #include "gazebo/sensors/Sensor.hh" × #include "gazebo/util/system.hh"` Include dependency graph for SonarSensor.hh:



Classes

- class **gazebo::sensors::SonarSensor**

  Sensor (p. *1130) with sonar cone.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

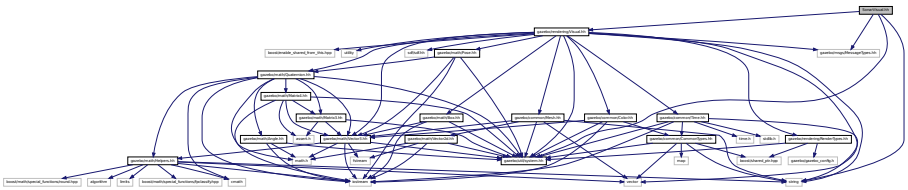
- namespace **gazebo::sensors**

  Sensors namespace.

11.218 SonarVisual.hh File Reference

`#include <string> #include "gazebo/msgs/MessageTypes.hh" #include "gazebo/rendering/Visual.hh" #include "gazebo/util/system.-`

hh" Include dependency graph for SonarVisual.hh:



Classes

- class **gazebo::rendering::SonarVisual**

  Visualization for sonar data.

Namespaces

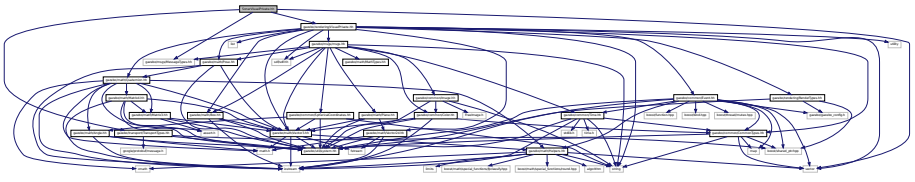
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.219 SonarVisualPrivate.hh File Reference

#include <vector> #include "gazebo/msgs/MessageTypes.hh" #include "gazebo/transport/TransportTypes.hh" #include "gazebo/rendering/VisualPrivate.hh" Include dependency graph for SonarVisualPrivate.hh:



Classes

- class **gazebo::rendering::SonarVisualPrivate**

  *Private data for the Sonar **Visual** (p. 1477) class.*

Namespaces

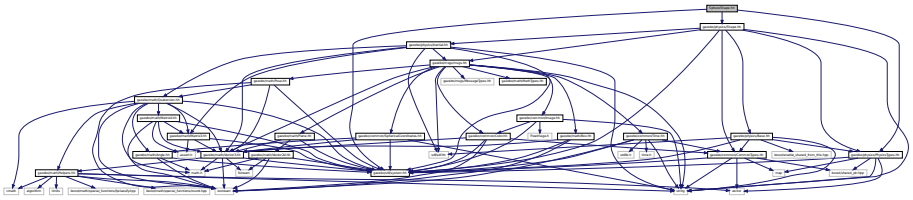
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

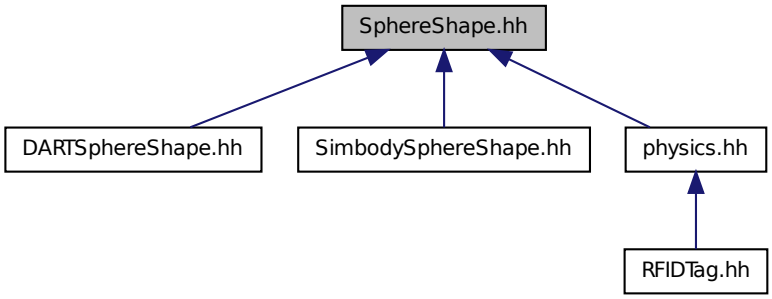
  Rendering namespace.

11.220 SphereShape.hh File Reference

`#include "gazebo/physics/Shape.hh"` `#include "gazebo/physics/PhysicsTypes.hh"` `#include "gazebo/util/system.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**

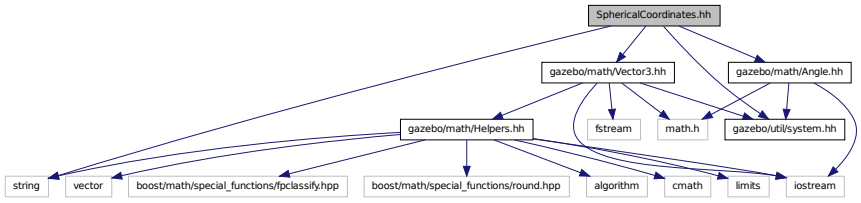
  Forward declarations for the common classes.

- namespace **gazebo::physics**

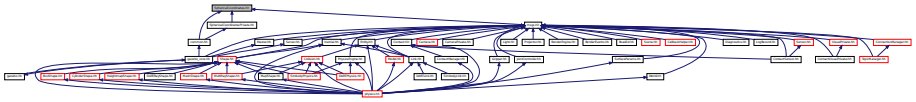
  namespace for physics

11.221 SphericalCoordinates.hh File Reference

`#include <string>` `#include "gazebo/math/Angle.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for SphericalCoordinates.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::SphericalCoordinates**

  Convert spherical coordinates for planetary surfaces.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

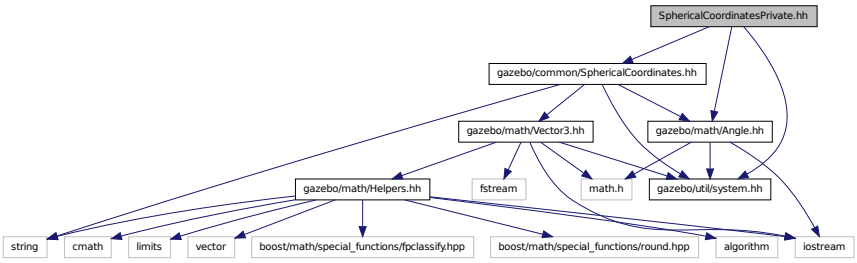
- namespace **gazebo::common**

  Common namespace.

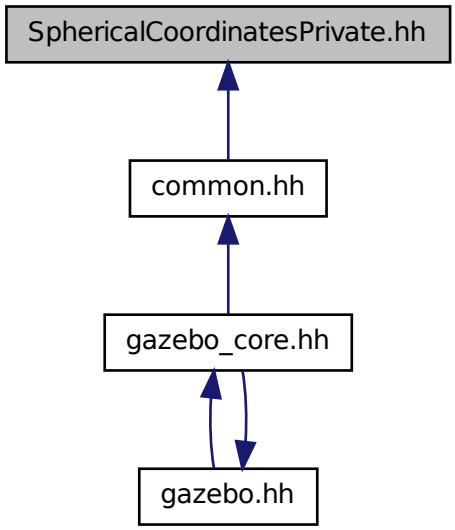
11.222 SphericalCoordinatesPrivate.hh File Reference

```
#include "gazebo/common/SphericalCoordinates.hh" #include
"gazebo/math/Angle.hh" #include "gazebo/util/system.hh"
```

Include dependency graph for SphericalCoordinatesPrivate.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::SphericalCoordinatesPrivate**

  commmon/common.hh

Namespaces

- namespace **gazebo**

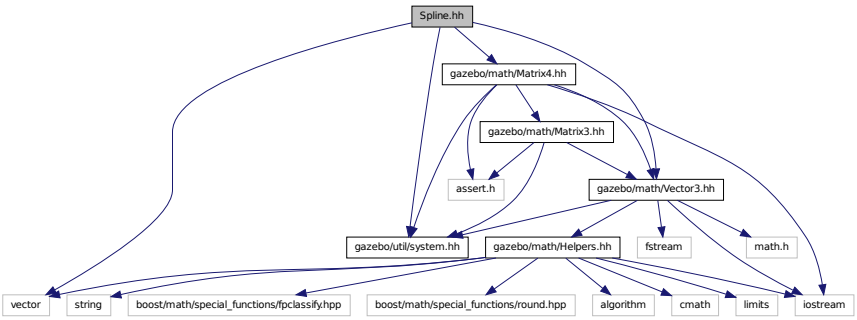
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

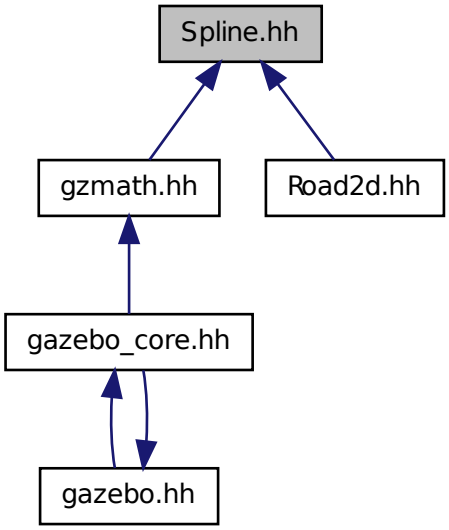
11.223 Spline.hh File Reference

```
#include <vector>      #include "gazebo/math/Vector3.hh" ×
```

`#include "gazebo/math/Matrix4.hh"` `#include "gazebo/util/system.-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

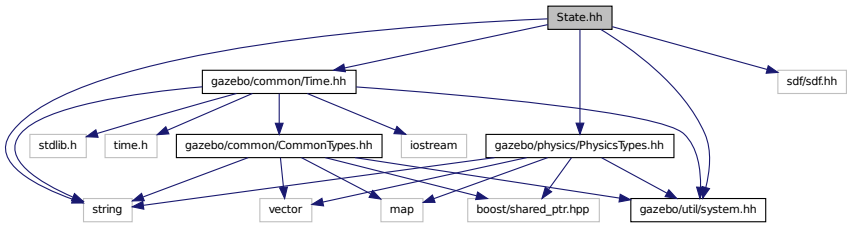
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

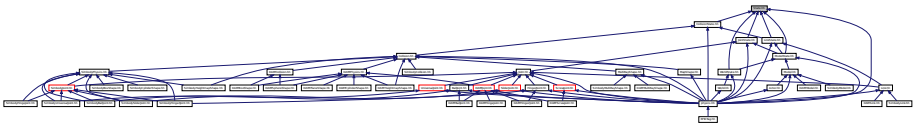
  Math namespace.

11.224 State.hh File Reference

`#include <string>` `#include <sdf/sdf.hh>` `#include "gazebo/physics/-PhysicsTypes.hh"` `#include "gazebo/common/Time.hh"` `#include "gazebo/util/system.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. *1323*) *of an entity.*

Namespaces

- namespace **gazebo**

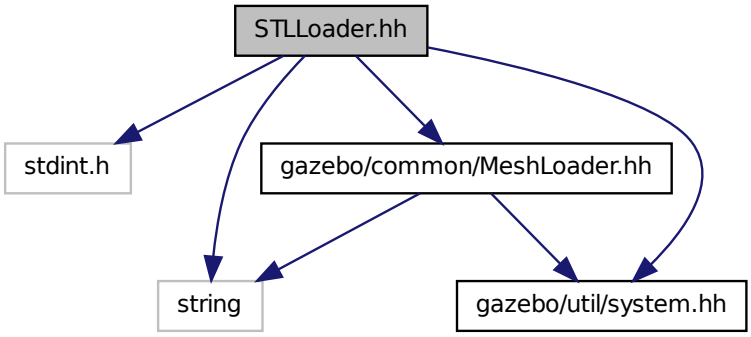
  Forward declarations for the common classes.

- namespace **gazebo::physics**

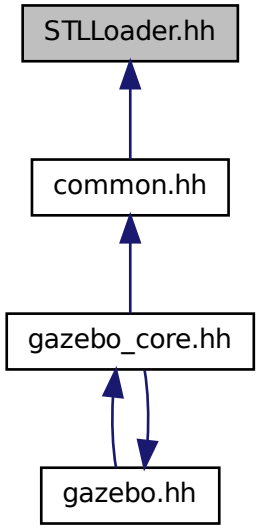
  namespace for physics

11.225 STLLoader.hh File Reference

`#include <stdint.h>` `#include <string>` `#include "gazebo/common/-MeshLoader.hh"` `#include "gazebo/util/system.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.

Defines

- #define **COR3_MAX** 200000

- #define **FACE_MAX** 200000
- #define **LINE_MAX_LEN** 256
- #define **ORDER_MAX** 10

11.225.1 Define Documentation

11.225.1.1 #define COR3_MAX 200000

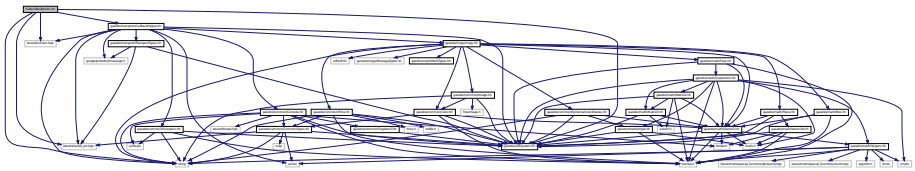
11.225.1.2 #define FACE_MAX 200000

11.225.1.3 #define LINE_MAX_LEN 256

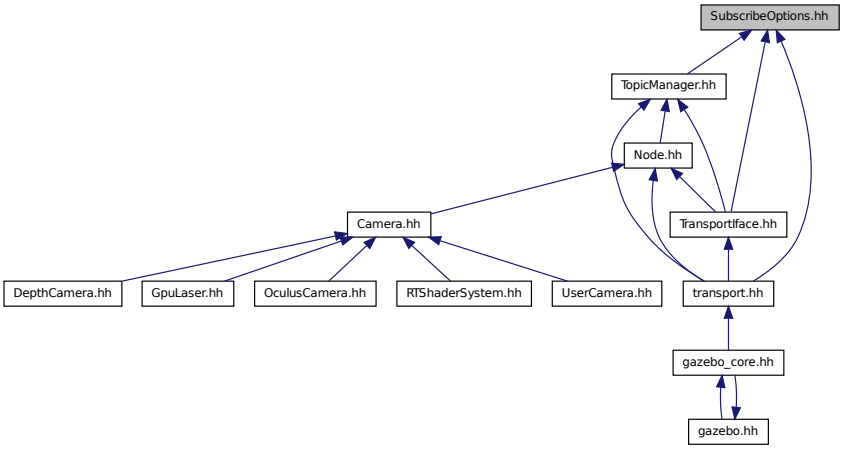
11.225.1.4 #define ORDER_MAX 10

11.226 SubscribeOptions.hh File Reference

`#include <boost/function.hpp> #include <boost/shared_ptr.hpp> #include <string> #include "gazebo/transport/CallbackHelper.hh" #include "gazebo/util/system.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**

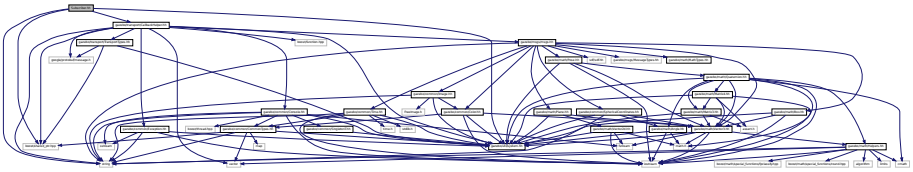
  Forward declarations for the common classes.

- namespace **gazebo::transport**

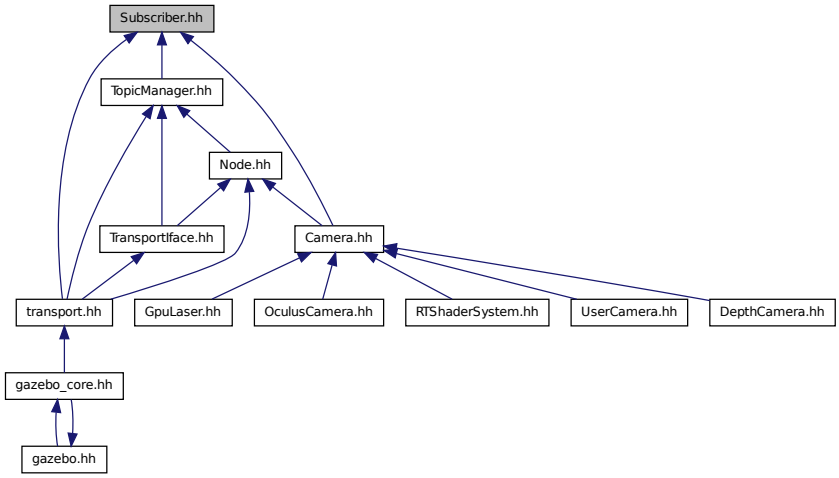
11.227 Subscriber.hh File Reference

```
#include <string> #include <boost/shared_ptr.hpp> #include
"gazebo/transport/CallbackHelper.hh" #include "gazebo/util/system.-
```

hh" Include dependency graph for Subscriber.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Subscriber**

  A subscriber to a topic.

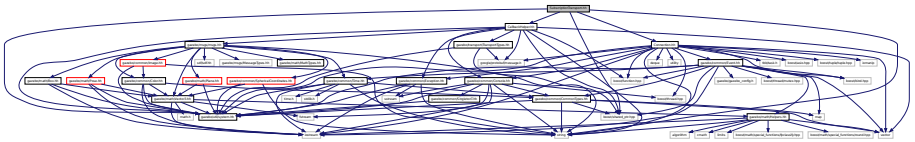
Namespaces

- namespace **gazebo**

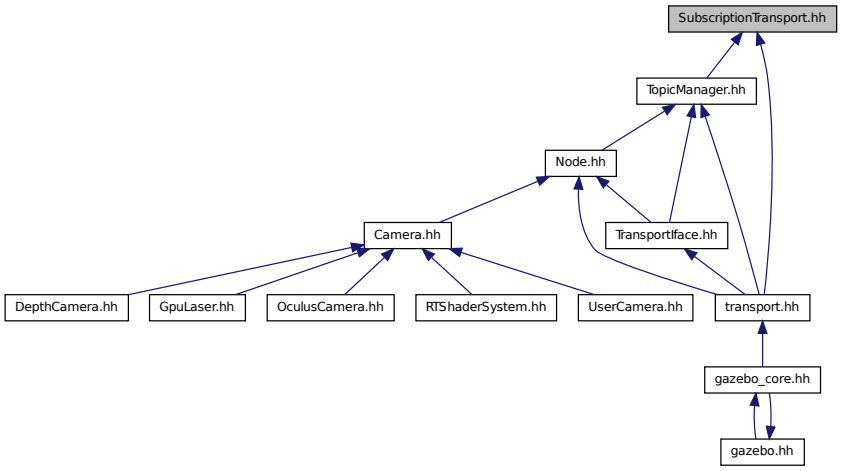
  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.228 SubscriptionTransport.hh File Reference

`#include <boost/shared_ptr.hpp>` `#include <string>` `#include "Connection.hh"` `#include "CallbackHelper.hh"` `#include "gazebo/util/system.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

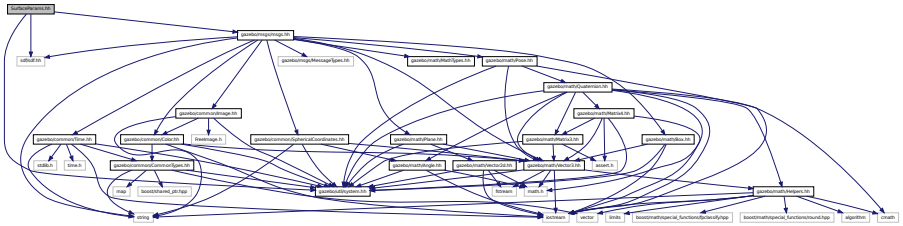
- namespace **gazebo**

Forward declarations for the common classes.

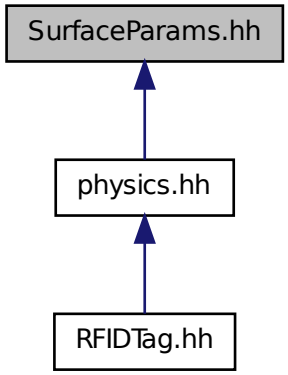
- namespace **gazebo::transport**

11.229 SurfaceParams.hh File Reference

`#include <sdf/sdf.hh>` `#include "gazebo/msgs/msgs.hh"` × `#include "gazebo/util/system.hh"` Include dependency graph for - SurfaceParams.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::FrictionPyramid**

  Parameters used for friction pyramid model.
- class **gazebo::physics::SurfaceParams**

  SurfaceParams *(*p. *1350) defines various Surface contact parameters.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.230 system.hh File Reference

Defines

- #define **GAZEBO_HIDDEN**

  Use to represent "symbol hidden" if supported.
- #define **GAZEBO_VISIBLE**

  Use to represent "symbol visible" if supported.

11.230.1 Define Documentation

11.230.1.1 #define GAZEBO_HIDDEN

Use to represent "symbol hidden" if supported.

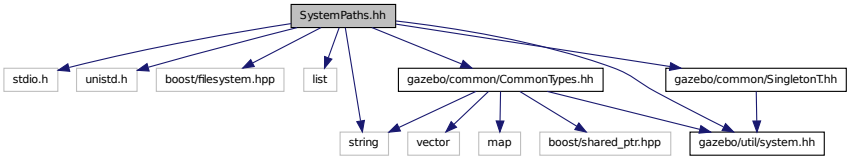
11.230.1.2 #define GAZEBO_VISIBLE

Use to represent "symbol visible" if supported.

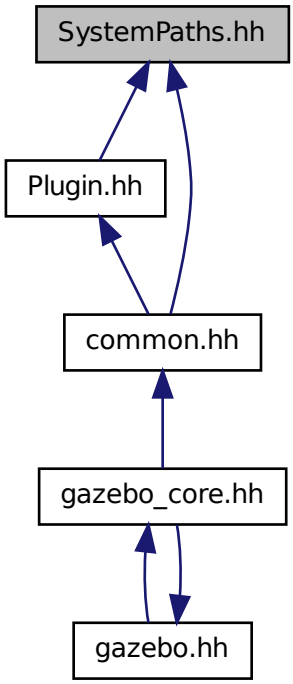
11.231 SystemPaths.hh File Reference

```
#include <stdio.h> #include <unistd.h> #include <boost/filesystem.-
hpp> #include <list> #include <string> #include "gazebo/common/-
CommonTypes.hh"   #include "gazebo/common/SingletonT.hh" ×
```

`#include "gazebo/util/system.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.

Defines

- #define **GetCurrentDir** getcwd
- #define **LINUX**

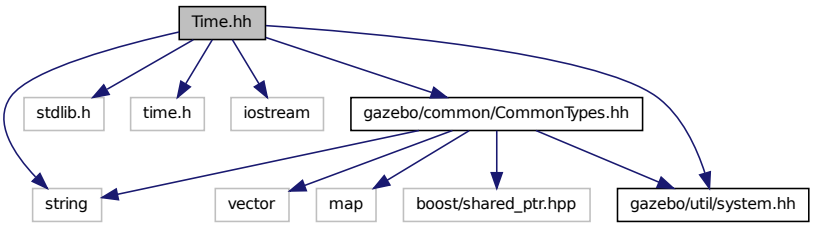
11.231.1 Define Documentation

11.231.1.1 #define GetCurrentDir getcwd

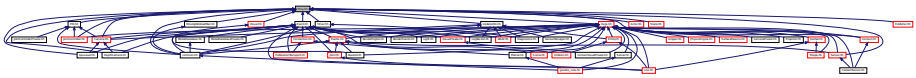
11.231.1.2 #define LINUX

11.232 Time.hh File Reference

`#include <string> #include <stdlib.h> #include <time.h> #include <iostream> #include "gazebo/common/CommonTypes.hh" #include "gazebo/util/system.hh"` Include dependency graph for Time.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Time**

  *A **Time** (p. 1361) class, can be used to hold wall- or sim-time.*

Namespaces

- namespace **gazebo**

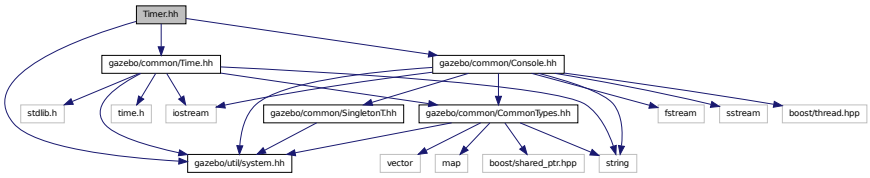
  Forward declarations for the common classes.

- namespace **gazebo::common**

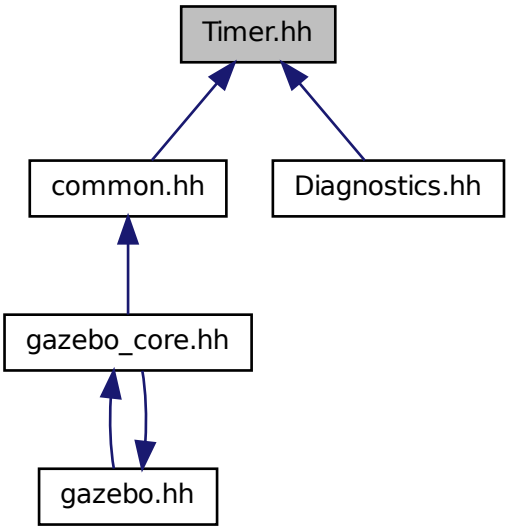
  Common namespace.

11.233 Timer.hh File Reference

`#include "gazebo/common/Console.hh"` `#include "gazebo/common/-Time.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for Timer.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Timer**

  A timer class, used to time things in real world walltime.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

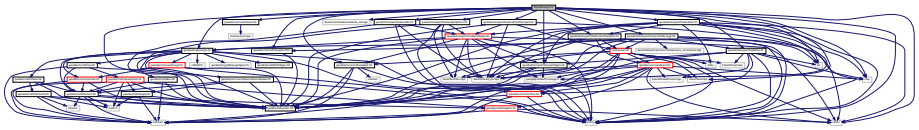
  Common namespace.

11.234 TopicManager.hh File Reference

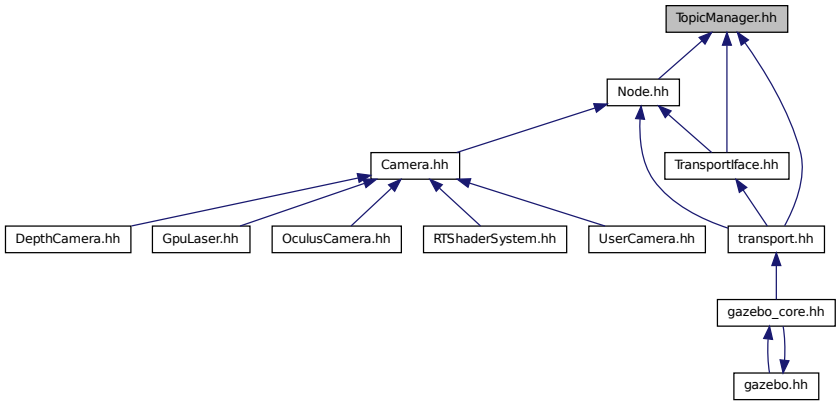
```
#include <boost/bind.hpp> #include <map> #include <list> ×
```

```
#include <string> #include <vector> #include <boost/unordered/unordered-
_set.hpp>   #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/Subscribe-
Options.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 "gazebo/util/system.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.

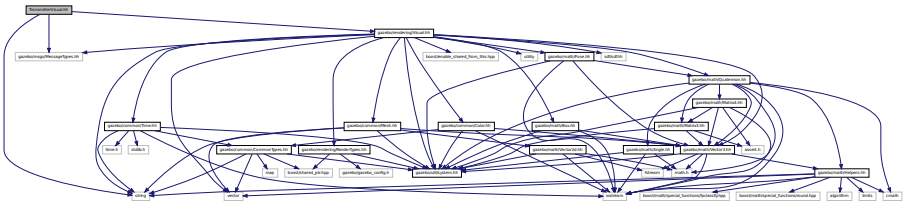
Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::transport**

11.235 TransmitterVisual.hh File Reference

`#include <string>` `#include "gazebo/msgs/MessageTypes.hh"` × `#include "gazebo/rendering/Visual.hh"` Include dependency graph for TransmitterVisual.hh:



Classes

- class **gazebo::rendering::TransmitterVisual**

  Visualization for the wireless propagation data.

Namespaces

- namespace **gazebo**

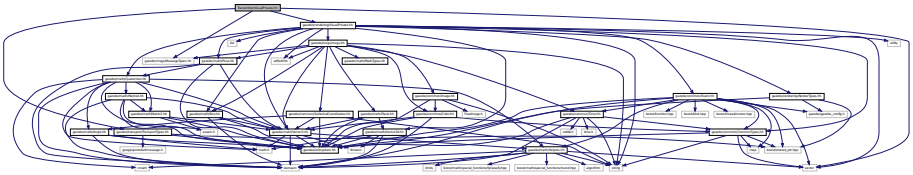
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.236 TransmitterVisualPrivate.hh File Reference

`#include <vector>` `#include "gazebo/msgs/MessageTypes.hh"` `#include "gazebo/transport/TransportTypes.hh"` `#include`

"gazebo/rendering/VisualPrivate.hh" Include dependency graph for TransmitterVisualPrivate.hh:



Classes

- class **gazebo::rendering::TransmitterVisualPrivate**

  *Private data for the Transmitter **Visual*** (p. *1477*) *class.*

Namespaces

- namespace **gazebo**

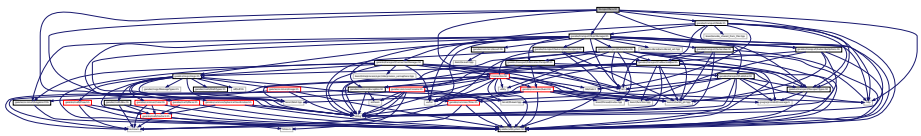
  Forward declarations for the common classes.

- namespace **gazebo::rendering**

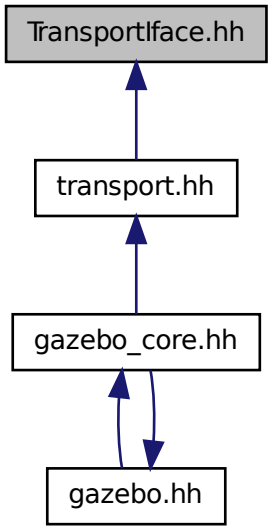
  Rendering namespace.

11.237 TransportIface.hh File Reference

#include <boost/bind.hpp> #include <string> #include <list> #include <map> #include "gazebo/transport/TransportTypes.hh" #include "gazebo/transport/SubscribeOptions.hh" #include "gazebo/transport/Node.hh" #include "gazebo/transport/TopicManager.hh" Include dependency graph for TransportIface.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

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

  Clear any remaining communication buffers.
- **GAZEBO_VISIBLE** transport::ConnectionPtr **gazebo::transport::connectToMaster** ()

  Create a connection to master.
- **GAZEBO_VISIBLE** void **gazebo::transport::fini** ()

  Cleanup the transport component.

- **GAZEBO_VISIBLE** 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.
- **GAZEBO_VISIBLE** void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)

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

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

  Get a list of all the unique advertised topic names.
- **GAZEBO_VISIBLE** bool **gazebo::transport::getMinimalComms** ()

  Get whether minimal comms has been enabled.
- **GAZEBO_VISIBLE** std::string **gazebo::transport::getTopicMsgType** (const std::string &_topicName)

  Get the message typename that is published on the given topic.
- **GAZEBO_VISIBLE** bool **gazebo::transport::init** (const std::string &_masterHost="", unsigned int _masterPort=0, uint32_t _timeoutIterations=30)

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

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

  Pause or unpause incoming messages.
- template<typename M >
  GAZEBO_VISIBLE void **gazebo::transport::publish** (const std::string &_topic, const google::protobuf::Message &_message)

  A convenience function for a one-time publication of a message.
- **GAZEBO_VISIBLE** boost::shared_ptr < msgs::Response > **gazebo::transport::request** (const std::string &_worldName, const std::string &_request, const std::string &_data="")

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

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

  Run the transport component.

- **GAZEBO_VISIBLE** void **gazebo::transport::setMinimalComms** (bool _enabled)

  Set whether minimal comms should be used.
- **GAZEBO_VISIBLE** void **gazebo::transport::stop** ()

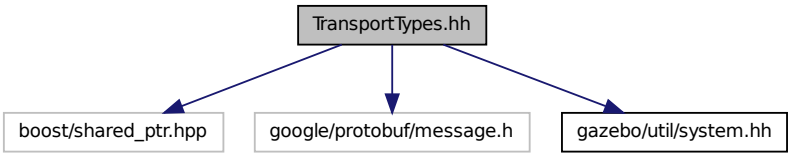
  Stop the transport component from running.
- **GAZEBO_VISIBLE** bool **gazebo::transport::waitForNamespaces** (const **gazebo::common::Time** &_maxWait)

  *Blocks while waiting for topic namespaces from the **Master** (p. 792).*

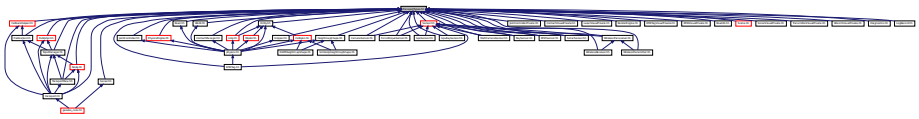
11.238 TransportTypes.hh File Reference

Forward declarations for transport.

`#include <boost/shared_ptr.hpp>` `#include <google/protobuf/message.h>` `#include "gazebo/util/system.hh"` 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

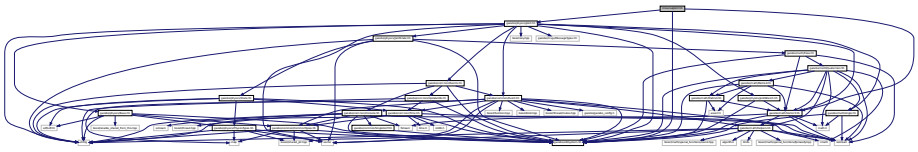
- typedef boost::shared_ptr < google::protobuf::Message > **gazebo::transport::-MessagePtr**
- typedef boost::shared_ptr< Node > **gazebo::transport::NodePtr**
- typedef boost::shared_ptr < Publication > **gazebo::transport::PublicationPtr**
- typedef boost::shared_ptr < PublicationTransport > **gazebo::transport::-PublicationTransportPtr**
- typedef boost::shared_ptr < Publisher > **gazebo::transport::PublisherPtr**
- typedef boost::shared_ptr < Subscriber > **gazebo::transport::SubscriberPtr**
- typedef boost::shared_ptr < SubscriptionTransport > **gazebo::transport::-SubscriptionTransportPtr**

11.238.1 Detailed Description

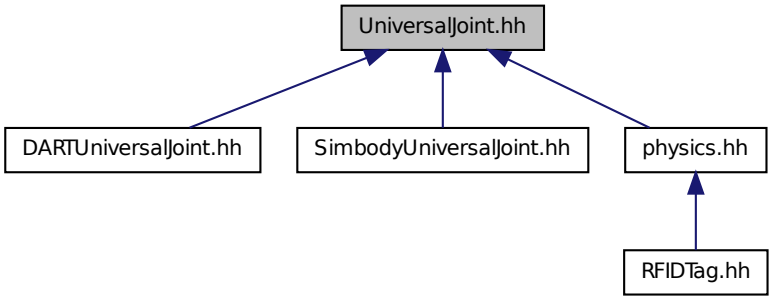
Forward declarations for transport.

11.239 UniversalJoint.hh File Reference

`#include "gazebo/math/Vector3.hh"` `#include "gazebo/physics/-Joint.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for UniversalJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

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

  A universal joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

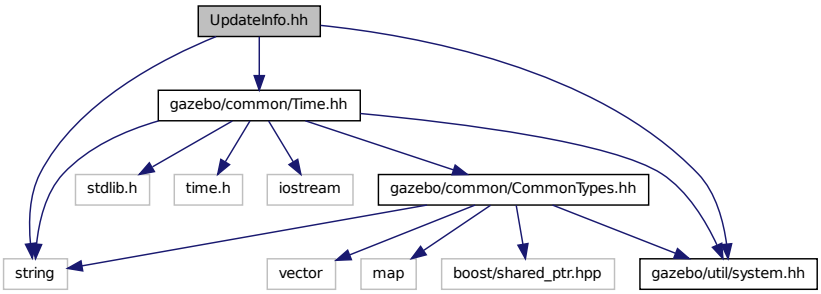
- namespace **gazebo::physics**

  namespace for physics

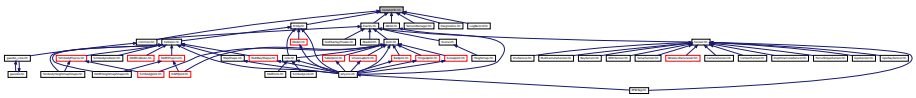
11.240 UpdateInfo.hh File Reference

`#include <string> #include "gazebo/common/Time.hh" #include`

"gazebo/util/system.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**

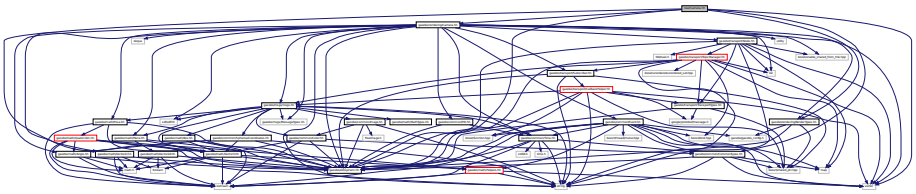
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.241 UserCamera.hh File Reference

```
#include <string> #include <vector> #include "gazebo/rendering/-
Camera.hh"      #include "gazebo/rendering/RenderTypes.hh"
```

`#include "gazebo/common/CommonTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for UserCamera.hh:



Classes

- class **gazebo::rendering::UserCamera**

  A camera used for user visualization of a scene.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.242 UserCameraPrivate.hh File Reference

Classes

- class **gazebo::rendering::UserCameraPrivate**

  *Private data for the **UserCamera** (p. 1407) class.*

Namespaces

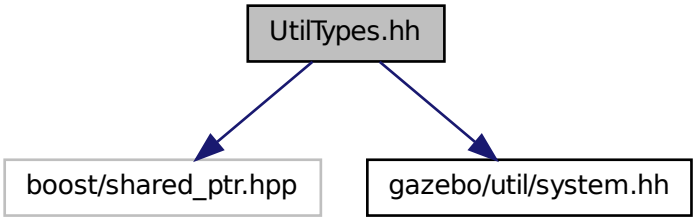
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

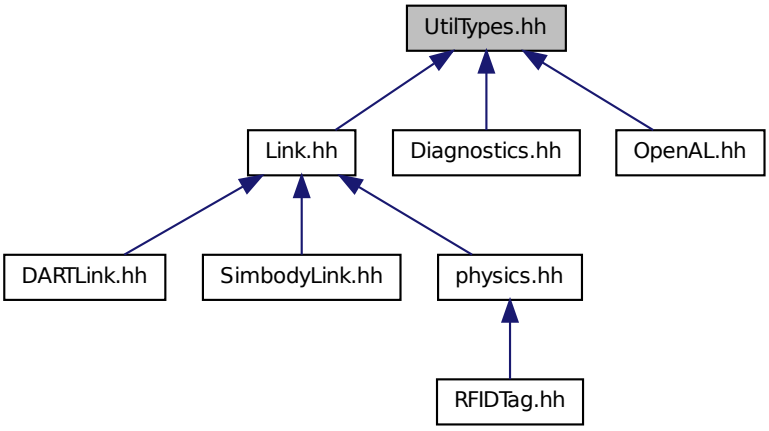
  Rendering namespace.

11.243 UtilTypes.hh File Reference

`#include <boost/shared_ptr.hpp>` `#include "gazebo/util/system.hh"` 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**

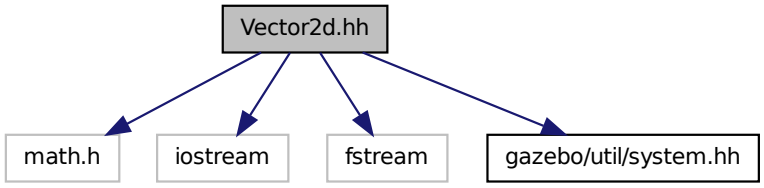
Typedefs

- typedef boost::shared_ptr < DiagnosticTimer > **gazebo::util::DiagnosticTimerPtr**
- typedef boost::shared_ptr < OpenALSink > **gazebo::util::OpenALSinkPtr**
- typedef boost::shared_ptr < OpenALSource > **gazebo::util::OpenALSourcePtr**

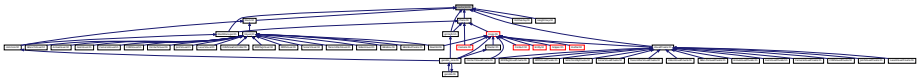
11.243.1 Detailed Description

11.244 Vector2d.hh File Reference

`#include <math.h>` `#include <iostream>` `#include <fstream>` × `#include "gazebo/util/system.hh"` 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**

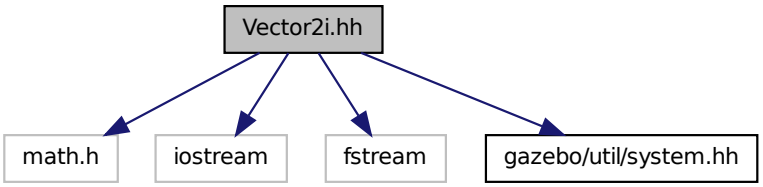
  Forward declarations for the common classes.

- namespace **gazebo::math**

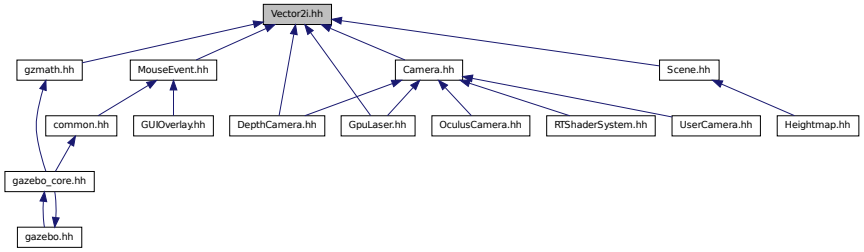
  Math namespace.

11.245 Vector2i.hh File Reference

`#include <math.h>` `#include <iostream>` `#include <fstream>` × `#include "gazebo/util/system.hh"` 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

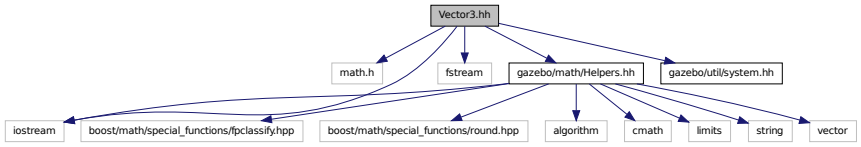
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

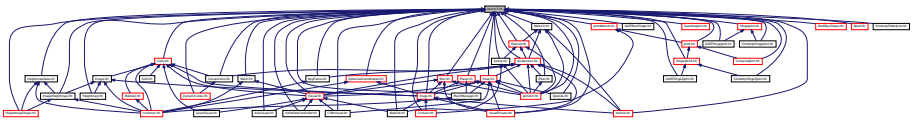
  Math namespace.

11.246 Vector3.hh File Reference

`#include <math.h>` `#include <iostream>` `#include <fstream>` × `#include "gazebo/math/Helpers.hh"` `#include "gazebo/util/system.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. 1440) class represents the generic vector containing 3 elements.*

Namespaces

- namespace **gazebo**

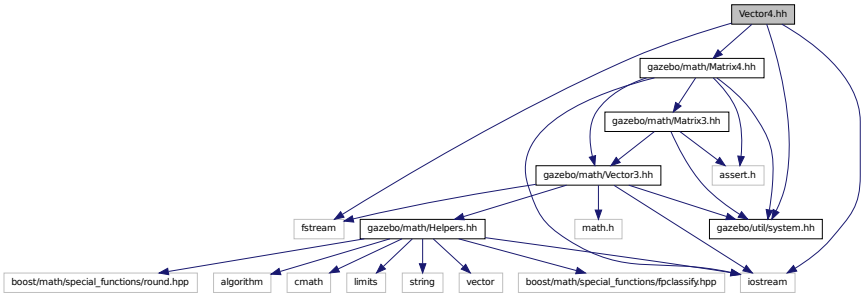
  Forward declarations for the common classes.

- namespace **gazebo::math**

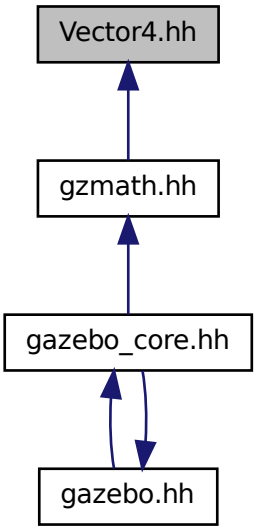
  Math namespace.

11.247 Vector4.hh File Reference

`#include <iostream>` `#include <fstream>` `#include "gazebo/math/-Matrix4.hh"` `#include "gazebo/util/system.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**

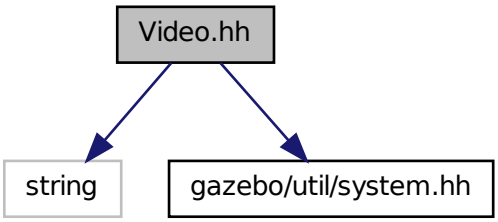
  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

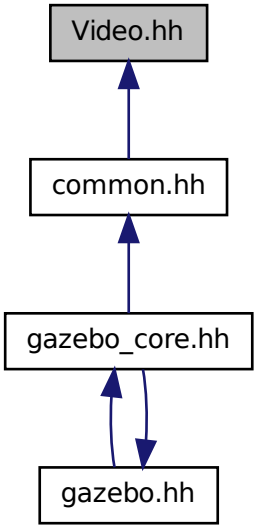
11.248 Video.hh File Reference

`#include <string>` `#include "gazebo/util/system.hh"` 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

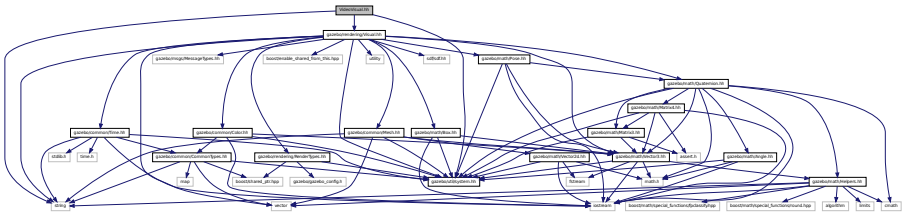
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.249 VideoVisual.hh File Reference

`#include <string>` `#include "gazebo/rendering/Visual.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for VideoVisual.hh:



Classes

- class **gazebo::rendering::VideoVisual**

  A visual element that displays a video as a texture.

Namespaces

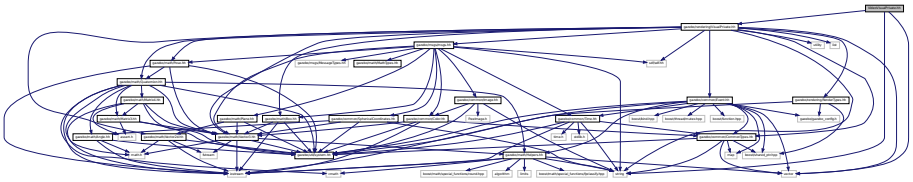
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.250 VideoVisualPrivate.hh File Reference

`#include <string> #include <vector> #include "gazebo/rendering/-VisualPrivate.hh"` Include dependency graph for VideoVisualPrivate.hh:



Classes

- class **gazebo::rendering::VideoVisualPrivate**

  *Private data for the Video **Visual*** (p. *1477*) *class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

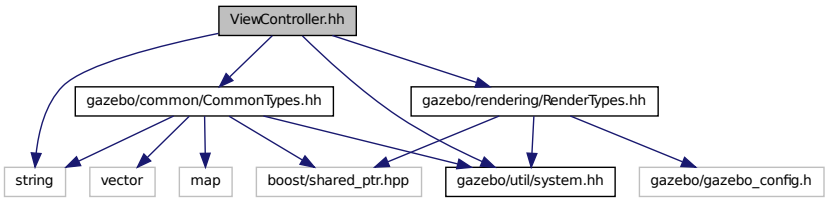
- namespace **gazebo::rendering**

  Rendering namespace.

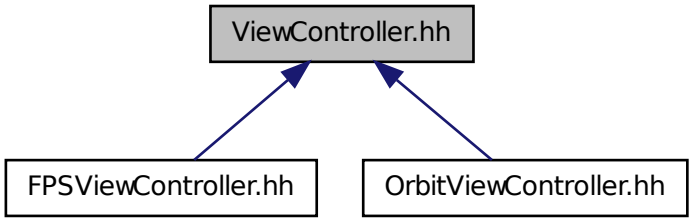
11.251 ViewController.hh File Reference

`#include <string> #include "gazebo/common/CommonTypes.-hh" #include "gazebo/rendering/RenderTypes.hh" #include`

"gazebo/util/system.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

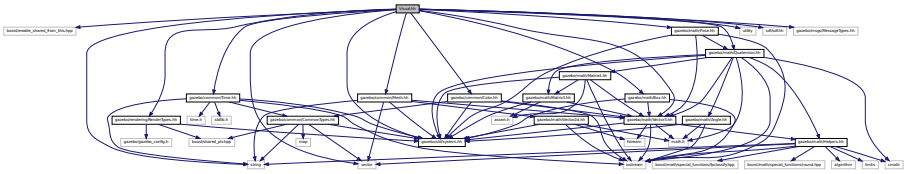
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.252 Visual.hh File Reference

`#include <boost/enable_shared_from_this.hpp>` `#include <string>` `#include <utility>` `#include <vector>` `#include <sdf/sdf.hh>` `#include "gazebo/common/Color.hh"` `#include "gazebo/common/Mesh.hh"` `#include "gazebo/common/Time.hh"` `#include "gazebo/msgs/MessageTypes.hh"` `#include "gazebo/math/Box.hh"` `#include "gazebo/math/Pose.hh"` `#include "gazebo/math/Quaternion.hh"` `#include "gazebo/math/Vector3.hh"` `#include "gazebo/rendering/RenderTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for Visual.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::rendering::Visual**

  A renderable object.

Namespaces

- namespace **gazebo**

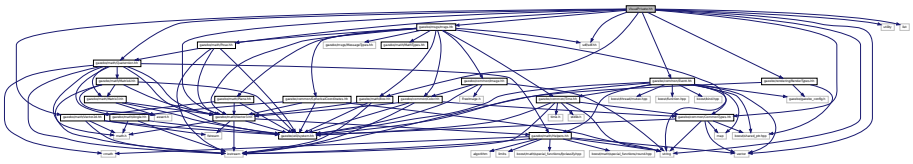
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.253 VisualPrivate.hh File Reference

```
#include <string>  #include <utility>  #include <list>
#include <vector> #include <sdf/sdf.hh> #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/rendering/-
RenderTypes.hh"  #include "gazebo/common/CommonTypes.hh" ×
```

Include dependency graph for VisualPrivate.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::rendering::VisualPrivate**

  *Private data for the **Visual*** (p. *1477*) *class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

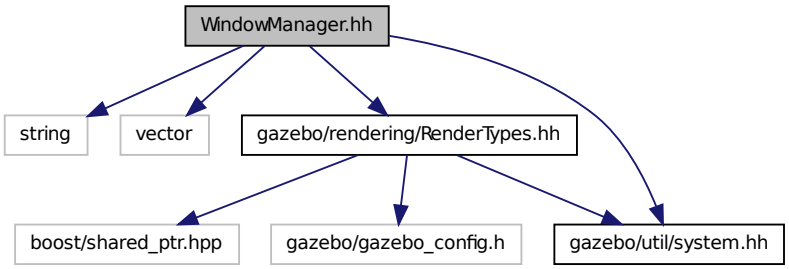
- namespace **gazebo::rendering**

  Rendering namespace.

- namespace **Ogre**

11.254 WindowManager.hh File Reference

`#include <string>` `#include <vector>` `#include "gazebo/rendering/-RenderTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for WindowManager.hh:



Classes

- class **gazebo::rendering::WindowManager**

  Class to mangage render windows.

Namespaces

- namespace **gazebo**

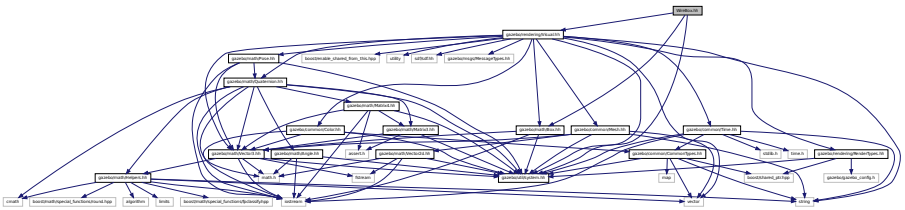
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.255 WireBox.hh File Reference

`#include "gazebo/math/Box.hh"` `#include "gazebo/rendering/-Visual.hh"` `#include "gazebo/util/system.hh"` Include dependency

graph for WireBox.hh:



Classes

- class **gazebo::rendering::WireBox**

  Draws a wireframe box.

Namespaces

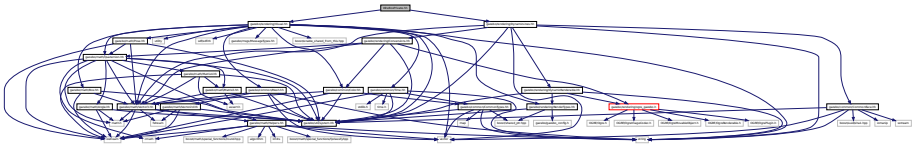
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.256 WireBoxPrivate.hh File Reference

`#include "gazebo/rendering/Visual.hh"` `#include "gazebo/rendering/-DynamicLines.hh"` Include dependency graph for WireBoxPrivate.hh:



Classes

- class **gazebo::rendering::WireBoxPrivate**

  *Private data for the **WireBox** (p. 1515) class.*

Namespaces

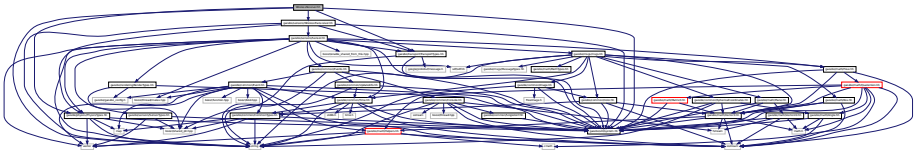
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.257 WirelessReceiver.hh File Reference

`#include <string> #include "gazebo/physics/PhysicsTypes.-hh" #include "gazebo/sensors/WirelessTransceiver.hh" × #include "gazebo/transport/TransportTypes.hh" #include "gazebo/util/system.hh"` Include dependency graph for WirelessReceiver.hh:



Classes

- class **gazebo::sensors::WirelessReceiver**

  Sensor (p. *1130*) *class for receiving wireless signals.*

Namespaces

- namespace **gazebo**

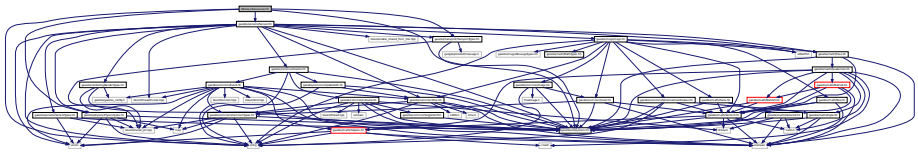
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

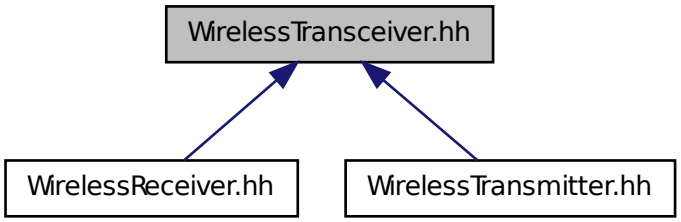
11.258 WirelessTransceiver.hh File Reference

`#include <string> #include "gazebo/physics/PhysicsTypes.-hh" #include "gazebo/sensors/Sensor.hh" #include "gazebo/transport/-`

`TransportTypes.hh" #include "gazebo/util/system.hh"` Include dependency graph for WirelessTransceiver.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::WirelessTransceiver**

  Sensor *(*p. *1130) class for receiving wireless signals.*

Namespaces

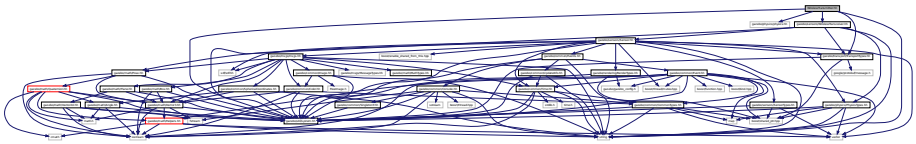
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.259 WirelessTransmitter.hh File Reference

`#include <string>  #include "gazebo/physics/physics.hh"` ×
`#include "gazebo/sensors/WirelessTransceiver.hh"  #include "gazebo/transport/TransportTypes.hh"` `#include "gazebo/util/system.hh"` Include dependency graph for WirelessTransmitter.hh:



Classes

- class **gazebo::sensors::WirelessTransmitter**

  Transmitter to send wireless signals.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

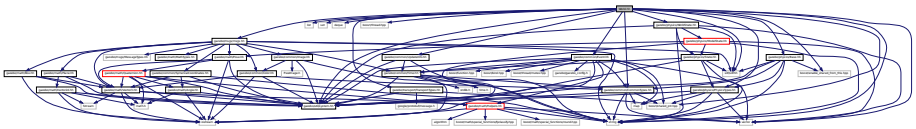
- namespace **gazebo::sensors**

  Sensors namespace.

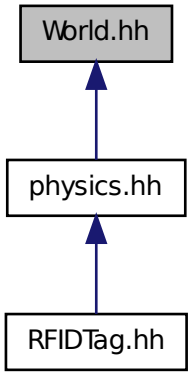
11.260 World.hh File Reference

`#include <vector>` `#include <list>` `#include <set>` `#include <deque>  #include <string>  #include <boost/thread.hpp>` ×
`#include <boost/enable_shared_from_this.hpp>  #include <boost/shared_ptr.hpp>  #include <sdf/sdf.hh>  #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/util/system.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

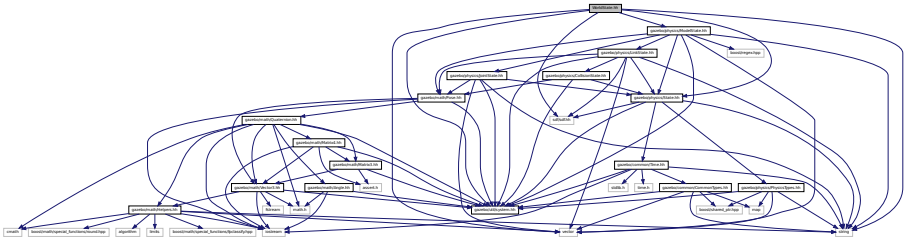
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

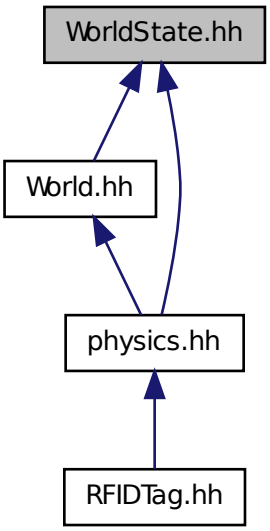
  namespace for physics

11.261 WorldState.hh File Reference

`#include <string> #include <vector> #include <sdf/sdf.hh> #include "gazebo/physics/State.hh" #include "gazebo/physics/ModelState.hh" #include "gazebo/util/system.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. 1529) object.*

Namespaces

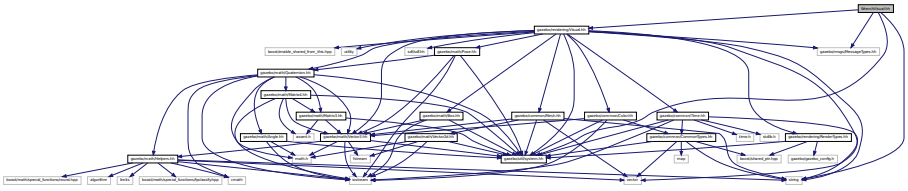
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.262 WrenchVisual.hh File Reference

`#include <string> #include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/rendering/Visual.hh"#include "gazebo/util/system.-
hh"` Include dependency graph for WrenchVisual.hh:



Classes

- class **gazebo::rendering::WrenchVisual**

  Visualization for sonar data.

Namespaces

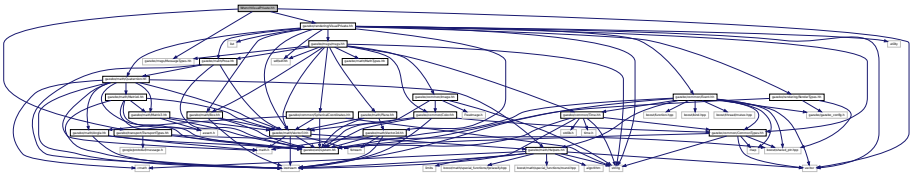
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.263 WrenchVisualPrivate.hh File Reference

`#include <vector>` `#include "gazebo/msgs/MessageTypes.hh"` `#include "gazebo/transport/TransportTypes.hh"` `#include "gazebo/rendering/VisualPrivate.hh"` Include dependency graph for WrenchVisualPrivate.hh:



Classes

- class **gazebo::rendering::WrenchVisualPrivate**

  *Private data for the Wrench **Visual*** (p. *1477*) *class.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.