Gazebo

1.2.5

Generated by Doxygen 1.8.2

Thu Nov 8 2012 10:16:26

Contents

Chapter 1

Gazebo API Reference

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

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

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

`Wiki`: A collection of user supported documentation.

Chapter 2

Todo List

Member gazebo::physics::Joint::GetForce (p. **430) (int _index)**

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

Member gazebo::sensors::CameraSensor::GetTopic (p. **258) () const**

to be implemented

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

Document me

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

Document me

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

Document me

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

Document me

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

Document me

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

Document me

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

Document me

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

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

Class gazebo::SystemPlugin (p. **838)**

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

Member urdf2gazebo::URDF2Gazebo::parseGazeboExtension (p. **875) (TiXmlDocument &urdf_xml)**

: do this using sdf definitions, not hard coded stuff

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

T

Chapter 6

Class Index

6.1 Class List

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

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Module Documentation

8.1 Common

Files

- file **CommonTypes.hh**

Namespaces

- namespace **gazebo::common**

  Common namespace.

Classes

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

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

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

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

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

  Message, error, warning, and logging functionality.
- class **gazebo::common::DiagnosticManager**

  A diagnostic manager class.
- class **gazebo::common::DiagnosticTimer**

  A timer designed for diagnostics.
- class **gazebo::common::Exception**

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

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

A key frame in an animation.

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

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

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

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

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

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

  *A plugin with access to **physics::Model** (p. 521).*
- class **gazebo::common::MouseEvent**

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

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

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

  A transformation node.
- class **gazebo::common::NumericAnimation**

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

  *A keyframe for a **NumericAnimation** (p. 569).*
- class **gazebo::common::PID**

  *Generic **PID** (p. 664) controller class.*
- class **gazebo::PluginT< T >**

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

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

  *A keyframe for a **PoseAnimation** (p. 685).*
- class **gazebo::SensorPlugin**

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

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

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

  ***Skeleton** (p. 784) animation.*
- class **gazebo::common::SkeletonNode**

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

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

  A child mesh.

- class **gazebo::common::SystemPaths**

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

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

  *A **Time** (p. 840) class, can be used to hold wall- or sim-time.*
- class **gazebo::common::Timer**

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

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

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

  *A plugin with access to **physics::World** (p. 954).*

Macros

- #define **DIAG_TIMER**(name) DiagnosticManager::Instance()->CreateTimer(name);

  Create an instance of common::DiagnosticManager.
- #define **gzclr_end** "\033[0m"

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

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

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

  Output an error message.
- #define **gzlog**

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

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

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

  Output a warning message.

Typedefs

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

Enumerations

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

  Used to specify the type of plugin.

Functions

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

  *add path prefix to **common::SystemPaths** (p. 834)*
- static void **gazebo::common::ModelDatabase::DownloadDependencies** (const std::string &_path)

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

  *search for file in **common::SystemPaths** (p. 834)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)

  *search for a file in **common::SystemPaths** (p. 834)*
- static std::string **gazebo::common::ModelDatabase::GetManifest** (const std::string &_uri)

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

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

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

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

  Returns the dictionary of all the model names.
- static std::string **gazebo::common::ModelDatabase::GetURI** ()

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

  Returns true if the model exists on the database.

8.1.1 Detailed Description

8.1.2 Macro Definition Documentation

8.1.2.1 #define DIAG_TIMER(*name*) DiagnosticManager::Instance()->CreateTimer(name);

Create an instance of common::DiagnosticManager.

8.1.2.2 #define gzclr_end "\033[0m"

end marker

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

start marker

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

Output a debug message.

8.1.2.5 #define gzerr

Value:

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

Output an error message.

Referenced by gazebo::transport::Connection::AsyncRead(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::physics::BulletJoint::GetAnchor(), gazebo::physics::BulletJoint::GetLinkForce(), gazebo::physics::BulletJoint::GetLinkTorque(), gazebo::physics::ScrewJoint< BulletJoint >::Load(), sdf::ParamT< std::string >::Set(), gazebo::physics::BulletJoint::SetAnchor(), gazebo::physics::BulletJoint::SetAttribute(), and gazebo::physics::BulletJoint::SetDamping().

8.1.2.6 #define gzlog

Value:

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

Log a message.

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

Output a message.

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

8.1.2.8 #define gzthrow(*msg*)

Value:

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

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

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

8.1.2.9 #define gzwarn

Value:

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

Output a warning message.

8.1.3 Typedef Documentation

8.1.3.1 typedef DiagnosticTimer* gazebo::common::DiagnosticTimerPtr

8.1.4 Enumeration Type Documentation

8.1.4.1 enum gazebo::PluginType

Used to specify the type of plugin.

Enumerator:

WORLD_PLUGIN A World plugin.

MODEL_PLUGIN A Model plugin.

SENSOR_PLUGIN A Sensor plugin.

SYSTEM_PLUGIN A System plugin.

VISUAL_PLUGIN A Visual plugin.

8.1.5 Function Documentation

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

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

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

Download all dependencies for a give model path.

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

Parameters

in	*_path*	Path to a model.

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

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

Parameters

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

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

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

Parameters

in	*_file*	the file name to look for

Returns

The path containing the file

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

Return the manifest.xml file as a string.

Returns

the manifest file from the model database.

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

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

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

Parameters

in	*_uri*	The URI of the model

Returns

The full path and filename to the SDF file

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

Get the name of a model based on a URI.

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

Parameters

in	*_uri*	the model uri

Returns

the model's name.

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

Get the local path to a model.

Get the path to a model based on a URI. If the model is on a remote server, then the model fetched and installed locally. param[in] _uri the model uri

Returns

path to a model directory

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

Returns the dictionary of all the model names.

Returns

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

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

Returns the the global model database URI.

Returns

the URI.

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

Returns true if the model exists on the database.

Parameters

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

Returns

True if the model was found.

8.2 Events

Namespaces

- namespace **gazebo::event**

  Event *(*p. *350) namespace.*

Classes

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

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

  Base class for all events.
- class **gazebo::event::EventT< T >**

  A class for event processing.

Functions

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

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

  Connect a callback to this event.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectAddEntity** (T _subscriber)

  Connect a boost::slot the the add entity signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectCreateEntity** (T _subscriber)

  Connect a boost::slot the the add entity signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectDeleteEntity** (T _subscriber)

  Connect a boost::slot the delete entity.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectDiagTimerStart** (T _subscriber)

  Connect a boost::slot the diagnostic timer start signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectDiagTimerStop** (T _subscriber)

  Connect a boost::slot the diagnostic timer stop signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectPause** (T _subscriber)

  Connect a boost::slot the the pause signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectPostRender** (T _subscriber)

  Connect a boost::slot the post render update signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectPreRender** (T _subscriber)

  Render start signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectRender** (T _subscriber)

Connect a boost::slot the render update signal.

- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectSetSelectedEntity** (T _subscriber)

  Connect a boost::slot the set selected entity.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectStep** (T _subscriber)

  Connect a boost::slot the the step signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectStop** (T _subscriber)

  Connect a boost::slot the the stop signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectWorldCreated** (T _subscriber)

  Connect a boost::slot the the world created signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectWorldUpdateEnd** (T _subscriber)

  Connect a boost::slot the the world update end signal.
- template<typename T >
  static ConnectionPtr **gazebo::event::Events::ConnectWorldUpdateStart** (T _subscriber)

  Connect a boost::slot the the world update start signal.
- virtual void **gazebo::event::EventT< T >::Disconnect** (ConnectionPtr _c)

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

  Disconnect a callback to this event.
- static void **gazebo::event::Events::DisconnectAddEntity** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the the add entity signal.
- static void **gazebo::event::Events::DisconnectCreateEntity** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the the add entity signal.
- static void **gazebo::event::Events::DisconnectDeleteEntity** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the delete entity.
- static void **gazebo::event::Events::DisconnectDiagTimerStart** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the diagnostic timer start signal.
- static void **gazebo::event::Events::DisconnectDiagTimerStop** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the diagnostic timer stop signal.
- static void **gazebo::event::Events::DisconnectPause** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the the pause signal.
- static void **gazebo::event::Events::DisconnectPostRender** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the post render update signal.
- static void **gazebo::event::Events::DisconnectPreRender** (ConnectionPtr _subscriber)

  Disconnect a render start signal.
- static void **gazebo::event::Events::DisconnectRender** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the render update signal.
- static void **gazebo::event::Events::DisconnectSetSelectedEntity** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the set selected entity.
- static void **gazebo::event::Events::DisconnectStep** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the the step signal.
- static void **gazebo::event::Events::DisconnectStop** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the the stop signal.
- static void **gazebo::event::Events::DisconnectWorldCreated** (ConnectionPtr _subscriber)

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

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

  Disconnect a boost::slot the the world update end signal.
- static void **gazebo::event::Events::DisconnectWorldUpdateStart** (ConnectionPtr _subscriber)

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

Variables

- static EventT< void(std::string)> **gazebo::event::Events::addEntity**

  An entity has been added.
- static EventT< void(std::string)> **gazebo::event::Events::deleteEntity**

  An entity has been deleted.
- static EventT< void(std::string)> **gazebo::event::Events::diagTimerStart**

  Diagnostic timer start.
- static EventT< void(std::string)> **gazebo::event::Events::diagTimerStop**

  Diagnostic timer stop.
- static EventT< void(std::string)> **gazebo::event::Events::entityCreated**

  An entity has been created.
- static EventT< void(bool)> **gazebo::event::Events::pause**

  Pause signal.
- static EventT< void()> **gazebo::event::Events::postRender**

  Post-Render.
- static EventT< void()> **gazebo::event::Events::preRender**

  Pre-render.
- static EventT< void()> **gazebo::event::Events::render**

  Render.
- static EventT< void(std::string, std::string)> **gazebo::event::Events::setSelectedEntity**

  An entity has been selected.
- static EventT< void()> **gazebo::event::Events::step**

  Step the simulation once signal.
- static EventT< void()> **gazebo::event::Events::stop**

  Simulation stop signal.
- static EventT< void(std::string)> **gazebo::event::Events::worldCreated**

  A world has been created.
- static EventT< void()> **gazebo::event::Events::worldUpdateEnd**

  World update has ended.
- static EventT< void()> **gazebo::event::Events::worldUpdateStart**

  World update has started.

8.2.1 Detailed Description

8.2.2 Function Documentation

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

Destructor.

Destructor. Deletes all the associated connections.

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

Connect a callback to this event.

Adds a connection.

Parameters

in	*_subscriber*	Pointer to a callback function

Returns

A **Connection** (p. 287) object, which will automatically call Disconnect when it goes out of scope

Parameters

in	*_subscriber*	the subscriber to connect

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

Connect a boost::slot the delete entity.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

Connect a boost::slot the the pause signal.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

Render start signal.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

Connect a boost::slot the render update signal.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

8.2.2.12 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectSetSelectedEntity (T *_subscriber*) [inline], [static]

Connect a boost::slot the set selected entity.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

8.2.2.13 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStep (T *_subscriber*) [inline], [static]

Connect a boost::slot the the step signal.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

8.2.2.14 template<typename T > static ConnectionPtr gazebo::event::Events::ConnectStop (T *_subscriber*) [inline], [static]

Connect a boost::slot the the stop signal.

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

Returns

a connection

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

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

Disconnect a callback to this event.

Removes a connection.

Parameters

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

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

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

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

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

Disconnect a callback to this event.

Removes a connection.

Parameters

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

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

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

8.2.2.21 static void gazebo::event::Events::DisconnectCreateEntity (ConnectionPtr *_subscriber*) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

8.2.2.22 static void gazebo::event::Events::DisconnectDeleteEntity (ConnectionPtr _subscriber) [inline], [static]

Disconnect a boost::slot the delete entity.

Parameters

in	*_subscriber*	the subscriber to this event

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

8.2.2.23 static void gazebo::event::Events::DisconnectDiagTimerStart (ConnectionPtr _subscriber) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

8.2.2.24 static void gazebo::event::Events::DisconnectDiagTimerStop (ConnectionPtr _subscriber) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

8.2.2.25 static void gazebo::event::Events::DisconnectPause (ConnectionPtr _subscriber) [inline], [static]

Disconnect a boost::slot the the pause signal.

Parameters

in	*_subscriber*	the subscriber to this event

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

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

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

Parameters

in	*_subscriber*	the subscriber to this event

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

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

Disconnect a render start signal.

Parameters

in	*_subscriber*	the subscriber to this event

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

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

Disconnect a boost::slot the render update signal.

Parameters

in	*_subscriber*	the subscriber to this event

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

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

Disconnect a boost::slot the set selected entity.

Parameters

in	*_subscriber*	the subscriber to this event

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

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

Disconnect a boost::slot the the step signal.

Parameters

in	*_subscriber*	the subscriber to this event

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

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

Disconnect a boost::slot the the stop signal.

Parameters

in	*_subscriber*	the subscriber to this event

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

8.2.2.32 static void gazebo::event::Events::DisconnectWorldCreated (ConnectionPtr *_subscriber*) [inline], [static]

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

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

8.2.2.33 static void gazebo::event::Events::DisconnectWorldUpdateEnd (ConnectionPtr *_subscriber*) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

8.2.2.34 static void gazebo::event::Events::DisconnectWorldUpdateStart (ConnectionPtr *_subscriber*) [inline], [static]

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

Parameters

in	*_subscriber*	the subscriber to this event

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

8.2.3 Variable Documentation

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

An entity has been added.

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

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

An entity has been deleted.

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

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

Diagnostic timer start.

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

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

Diagnostic timer stop.

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

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

An entity has been created.

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

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

Pause signal.

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

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

Post-Render.

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

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

Pre-render.

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

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

Render.

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

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

An entity has been selected.

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

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

Step the simulation once signal.

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

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

Simulation stop signal.

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

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

A world has been created.

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

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

World update has ended.

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

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

World update has started.

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

8.3 Math

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

Files

- file **MathTypes.hh**

  Forward declarations for the math classes.

Namespaces

- namespace **gazebo::math**

  Math namespace.

Classes

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

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

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

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

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

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

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

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

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

  Spline *(p. 808) for rotations.*
- class **gazebo::math::Spline**

  Splines.
- class **gazebo::math::Vector2d**

  Generic double x, y vector.
- class **gazebo::math::Vector2i**

  Generic integer x, y vector.
- class **gazebo::math::Vector3**

  The ***Vector3*** *(p. 902) class represents the generic vector containing 3 elements.*
- class **gazebo::math::Vector4**

  double Generic x, y, z, w vector

Functions

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

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

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

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

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

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

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

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

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

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

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

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

  get variance of vector of values

Variables

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

  Not a number.
- static const double **gazebo::math::NAN_I** = std::numeric_limits<int>::quiet_NaN()

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

8.3.1 Detailed Description

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

8.3.2 Function Documentation

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

simple clamping function

Parameters

in	*_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(), and gazebo::math::Quaternion::GetInverse().

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

check if a float is NaN

Parameters

in	*_v*	the value

Returns

true if _v is not a number, false otherwise

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

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

check if a double is NaN

Parameters

in	*_v*	the value

Returns

true if _v is not a number, false otherwise

References gazebo::math::isnan().

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

is this a power of 2?

Parameters

in	*_x*	the number

Returns

true if _x is a power of 2, false otherwise

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

get the maximum value of vector of values

Parameters

in	*_values*	the vector of values

Returns

maximum

References gazebo::math::min().

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

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

get mean of vector of values

Parameters

in	*_values*	the vector of values

Returns

the mean

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

get the minimum value of vector of values

Parameters

in	*_values*	the vector of values

Returns

minimum

References gazebo::math::max().

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

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

parse string into float

Parameters

_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.

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

parse string into an integer

Parameters

in	*_input*	the string

Returns

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

References gazebo::math::NAN_I.

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

get value at a specified precision

Parameters

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

Returns

the value for the specified precision

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

get variance of vector of values

Parameters

_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]

Not a number.

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

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

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

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

8.4 Messages

All messages and helper functions.

Namespaces

- namespace **gazebo::msgs**

  Messages namespace.

Classes

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

  *Google protobuf message generator for **gazebo::msgs*** (p. *101*).

Functions

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

  *Convert a **math::Vector3*** (p. *902*) *to a msgs::Vector3d.*
- msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)

  *Convert a **math::Quaternion*** (p. *697*) *to a msgs::Quaternion.*
- msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)

  *Convert a **math::Pose*** (p. *677*) *to a msgs::Pose.*
- msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)

  *Convert a **common::Color*** (p. *274*) *to a msgs::Color.*
- msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)

  *Convert a **common::Time*** (p. *840*) *to a msgs::Time.*
- msgs::PlaneGeom **gazebo::msgs::Convert** (const math::Plane &_p)

  *Convert a **math::Plane*** (p. *668*) *to a msgs::PlaneGeom.*
- math::Vector3 **gazebo::msgs::Convert** (const msgs::Vector3d &_v)

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

  *Convert a msgs::Quaternion to a **math::Quaternion*** (p. *697*).
- math::Pose **gazebo::msgs::Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose*** (p. *677*).
- common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color*** (p. *274*).
- common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time*** (p. *840*).
- math::Plane **gazebo::msgs::Convert** (const msgs::PlaneGeom &_p)

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

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

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

  Get the header from a protobuf message.

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

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

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

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

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

  *Convert a msgs::Image to a **common::Image** (p. 407).*
- void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)

  *Set a msgs::Image from a **common::Image** (p. 407).*
- void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 902).*
- void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 902).*
- void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 697).*
- void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 677).*
- void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)

  *Set a msgs::Color from a **common::Color** (p. 274).*
- void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)

  *Set a msgs::Time from a **common::Time** (p. 840).*
- void **gazebo::msgs::Set** (msgs::PlaneGeom *_p, const math::Plane &_v)

  *Set a msgs::Plane from a **math::Plane** (p. 668).*
- void **gazebo::msgs::Stamp** (msgs::Header *_header)

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

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

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

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

8.4.1 Detailed Description

All messages and helper functions.

8.4.2 Function Documentation

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

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

Parameters

_v	The vector to convert

Returns

A msgs::Vector3d object

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

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

Parameters

_q	The quaternion to convert

Returns

A msgs::Quaternion object

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

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

Parameters

_p	The pose to convert

Returns

A msgs::Pose object

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

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

Parameters

_c	The color to convert

Returns

A msgs::Color object

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

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

Parameters

_t	The time to convert

Returns

A msgs::Time object

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

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

Parameters

_p	The plane to convert

Returns

A msgs::PlaneGeom object

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

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

Parameters

_v	The plane to convert

Returns

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

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

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

Parameters

_q	The quaternion to convert

Returns

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

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

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

Parameters

_q	The pose to convert

Returns

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

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

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

Parameters

_c	The color to convert

Returns

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

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

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

Parameters

_t	The time to convert

Returns

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

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

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

Parameters

_p	The plane to convert

Returns

A common::Plane object

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

Create a request message.

Parameters

_request	Request string
_data	Optional data string

Returns

A Request message

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

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

Parameters

_sdf	The sdf element

Returns

The new msgs::Fog object

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

Get the header from a protobuf message.

Parameters

_message	A google protobuf message

Returns

A pointer to the message's header

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

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

Parameters

_sdf	The sdf element

Returns

The new msgs::GUI object

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

Initialize a message.

Parameters

_message	Message to initialize
_id	Optional string id

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

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

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

Parameters

_sdf	The sdf element

Returns

The new msgs::Light object

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

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

Parameters

_sdf	The sdf element

Returns

The new msgs::Scene object

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

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

Parameters

_img	The **common::Image** (p. 407) container
_msg	The Image message to convert

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

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

Parameters

_msg	A msgs::Image pointer
_i	A **common::Image** (p. 407) reference

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

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

Parameters

_pt	A msgs::Vector3d pointer
_v	A **math::Vector3** (p. 902) reference

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

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

Parameters

_pt	A msgs::Vector2d pointer
_v	A **math::Vector2d** (p. 885) reference

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

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

Parameters

_q	A msgs::Quaternion pointer
_v	A **math::Quaternion** (p. 697) reference

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

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

Parameters

_p	A msgs::Pose pointer
_v	A **math::Pose** (p. 677) reference

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

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

Parameters

_p	A msgs::Color pointer
_v	A **common::Color** (p. 274) reference

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

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

Parameters

_p	A msgs::Time pointer
_v	A **common::Time** (p. 840) reference

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

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

Parameters

_p	A msgs::Plane pointer
_v	A **math::Plane** (p. 668) reference

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

Time stamp a header.

Parameters

_header	Header to stamp

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

Set the time in a time message.

Parameters

_time	A Time message

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

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

Parameters

_sdf	The sdf element

Returns

The new msgs::TrackVisual object

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

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

Parameters

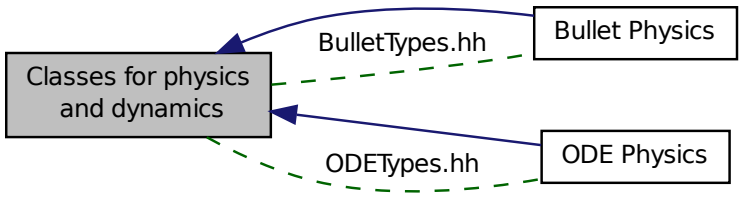
_sdf	The sdf element

Returns

The new msgs::Visual object

8.5 Classes for physics and dynamics

Collaboration diagram for Classes for physics and dynamics:



Modules

- **Bullet Physics**

  bullet physics engine wrapper
- **ODE Physics**

  ODE physics engine wrapper.

Files

- file **BulletTypes.hh**

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

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

  default namespace for gazebo

Namespaces

- namespace **gazebo::physics**

  namespace for physics

Classes

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

  Actor *(*p. *121) class enables GPU based mesh model / skeleton scriptable animation.*
- class **gazebo::physics::BallJoint**< **T** >

  Base *(*p. *145) class for a ball joint.*
- class **gazebo::physics::Base**

  Base *(*p. *145) class for most physics classes.*
- class **gazebo::physics::BoxShape**

Box geometry primitive.

- class **gazebo::physics::Collision**

  Base *(*p. *145) class for all collision entities.*
- class **gazebo::physics::CollisionState**

  Store state information of a ***physics::Collision*** *(*p. *262) object.*
- class **gazebo::physics::Contact**

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

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

  Base *(*p. *145) class for all physics objects in Gazebo.*
- class **gazebo::physics::Gripper**

  A gripper abstraction.
- class **gazebo::physics::HeightmapShape**

  HeightmapShape *(*p. *399) collision shape builds a heightmap from an image.*
- class **gazebo::physics::Hinge2Joint**< **T** >

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

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

  A class for inertial information about a link.
- class **gazebo::physics::Joint**

  Base *(*p. *145) class for all joints.*
- class **gazebo::physics::JointController**

  A class for manipulating ***physics::Joint*** *(*p. *423).*
- class **gazebo::physics::JointFeedback**

  Feedback information from a joint.
- class **gazebo::physics::JointState**

  keeps track of state of a ***physics::Joint*** *(*p. *423)*
- class **gazebo::physics::Link**

  Link *(*p. *454) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.*
- class **gazebo::physics::LinkState**

  Store state information of a ***physics::Link*** *(*p. *454) object.*
- class **gazebo::physics::Logger**

  Handles logging of data to disk.
- class **gazebo::physics::Logplay**

  Open and playback log files that were recorded using ***Logger*** *(*p. *477).*
- class **gazebo::physics::MapShape**

  Creates box extrusions based on an image.
- class **gazebo::physics::Model**

  A model is a collection of links, joints, and plugins.
- class **gazebo::physics::ModelState**

  Store state information of a ***physics::Model*** *(*p. *521) object.*
- class **gazebo::physics::MultiRayShape**

  Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.
- class **gazebo::physics::PhysicsEngine**

Base *(p. 145) class for a physics engine.*

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

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

  Collision *(p. 262) for an infinite plane.*
- class **gazebo::physics::RayShape**

  Base *(p. 145) class for Ray collision geometry.*
- class **gazebo::physics::Road**

  for building a ***Road*** *(p. 736) from SDF*
- class **gazebo::physics::ScrewJoint< T >**

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

  Base *(p. 145) class for all shapes.*
- class **gazebo::physics::SliderJoint< T >**

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

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

  State *(p. 813) of an entity.*
- class **gazebo::physics::SurfaceParams**

  SurfaceParams *(p. 830) defines various Surface contact parameters.*
- class **gazebo::physics::TrimeshShape**

  Triangle mesh collision shape.
- class **gazebo::physics::UniversalJoint< T >**

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

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

  Store state information of a ***physics::World*** *(p. 954) object.*

Macros

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

  Static physics registration macro.

Typedefs

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

Functions

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

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

  Finalize transport by calling ***gazebo::transport::fini*** *(p. 90).*
- WorldPtr **gazebo::physics::get_world** (const std::string &_name="")

Returns a pointer to a world by name.

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

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

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

  *Setup **gazebo::SystemPlugin** (p. 838)'s and call **gazebo::transport::init** (p. 90).*
- void **gazebo::physics::load_world** (WorldPtr _world, **sdf::ElementPtr** _sdf)

  *Load world from **sdf::Element** (p. 332) pointer.*
- void **gazebo::physics::load_worlds** (**sdf::ElementPtr** _sdf)

  *load multiple worlds from single **sdf::Element** (p. 332) pointer*
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)

  *Pause world by calling **World::SetPaused** (p. 964).*
- void **gazebo::physics::pause_worlds** (bool pause)

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

  remove multiple worlds stored in static variable gazebo::g_worlds
- void **gazebo::physics::run_world** (WorldPtr _world)

  *Run world by calling **World::Run()** (p. 964) given a pointer to it.*
- void **gazebo::physics::run_worlds** ()

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

  *Stop world by calling **World::Stop()** (p. 965) given a pointer to it.*
- void **gazebo::physics::stop_worlds** ()

  stop multiple worlds stored in static variable gazebo::g_worlds

Variables

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

  String names for the different entity types.

8.5.1 Detailed Description

8.5.2 Macro Definition Documentation

8.5.2.1 #define GZ_REGISTER_PHYSICS_ENGINE(*name, classname*)

Value:

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

Static physics registration macro.

Use this macro to register physics engine with the server.

Parameters

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

8.5.3 Typedef Documentation

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

8.5.4 Function Documentation

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

Create a world given a name.

Parameters

in	*_name*	Name of the world to create.

Returns

Pointer to the new world.

8.5.4.2 bool gazebo::physics::fini ( )

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

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

Returns a pointer to a world by name.

Parameters

in	*_name*	Name of the world to get.

Returns

Pointer to the world.

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

Init world given a pointer to it.

Parameters

in	*_world*	**World** (p. 954) to initialize.

8.5.4.5 void gazebo::physics::init_worlds ( )

initialize multiple worlds stored in static variable gazebo::g_worlds

8.5.4.6 bool gazebo::physics::load ( )

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

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

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

Parameters

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

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

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

Parameters

in	*_sdf*	SDF values used to create worlds.

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

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

Parameters

in	*_world*	**World** (p. 954) to pause or unpause.
in	*_pause*	True to pause, False to unpause.

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

pause multiple worlds stored in static variable gazebo::g_worlds

Parameters

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

8.5.4.11 void gazebo::physics::remove_worlds ( )

remove multiple worlds stored in static variable gazebo::g_worlds

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

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

Parameters

in	*_world*	**World** (p. 954) to run.

8.5.4.13 void gazebo::physics::run_worlds ( )

run multiple worlds stored in static variable gazebo::g_worlds

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

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

Parameters

in	*_world*	**World** (p. 954) to stop.

8.5.4.15 void gazebo::physics::stop_worlds ( )

stop multiple worlds stored in static variable gazebo::g_worlds

8.5.5 Variable Documentation

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

Initial value:

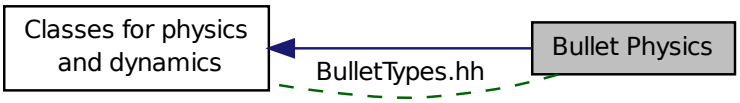
```
= {
      "common",
      "entity",
      "model",
      "actor",
      "link",
      "collision",
      "light",
      "visual",
      "joint",
      "ball",
      "hinge2",
      "hinge",
      "slider",
      "universal",
      "shape",
      "box",
      "cylinder",
      "heightmap",
      "map",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

String names for the different entity types.

8.6 Bullet Physics

bullet physics engine wrapper

Collaboration diagram for Bullet Physics:



Files

- file **BulletTypes.hh**

  Bullet wrapper forward declarations and typedefs.

Classes

- class **gazebo::physics::BulletBallJoint**

  BulletBallJoint *(p. 166) class models a ball joint in Bullet.*
- class **gazebo::physics::BulletBoxShape**

  Bullet box collision.
- class **gazebo::physics::BulletCollision**

  Bullet collisions.
- class **gazebo::physics::BulletCylinderShape**

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

  Height map collision.
- class **gazebo::physics::BulletHinge2Joint**

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

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

  Base *(p. 145) class for all joints.*
- class **gazebo::physics::BulletLink**

  Bullet ***Link*** *(p. 454) class.*
- class **gazebo::physics::BulletMotionState**

  Bullet btMotionState encapsulation.
- class **gazebo::physics::BulletMultiRayShape**

  Bullet specific version of ***MultiRayShape*** *(p. 549).*
- class **gazebo::physics::BulletPhysics**

  Bullet physics engine.
- class **gazebo::physics::BulletPlaneShape**

Bullet collision for an infinite plane.

- class **gazebo::physics::BulletRaySensor**

  An Bullet Ray sensor.
- class **gazebo::physics::BulletRayShape**

  Ray shape for bullet.
- class **gazebo::physics::BulletScrewJoint**

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

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

  Bullet sphere collision.
- class **gazebo::physics::BulletTrimeshShape**

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

  A bullet universal joint class.

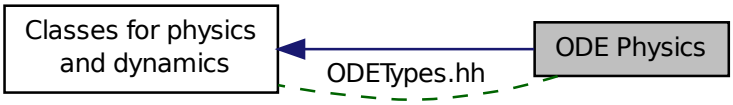
8.6.1 Detailed Description

bullet physics engine wrapper

8.7 ODE Physics

ODE physics engine wrapper.

Collaboration diagram for ODE Physics:



Files

- file **ODETypes.hh**

  ODE wrapper forward declarations and typedefs.

Classes

- class **gazebo::physics::ContactFeedback**

  data structure for contact feedbacks
- class **gazebo::physics::ODEBallJoint**

  *An **ODEBallJoint** (p. 573).*
- class **gazebo::physics::ODEBoxShape**

  ODE Box shape.
- class **gazebo::physics::ODECollision**

  ***Base** (p. 145) class for all ODE collisions.*
- class **gazebo::physics::ODECylinderShape**

  ODE cylinder shape.
- class **gazebo::physics::ODEHeightmapShape**

  ODE Height map collision.
- class **gazebo::physics::ODEHinge2Joint**

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

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

  ODE joint interface.
- class **gazebo::physics::ODELink**

  *ODE **Link** (p. 454) class.*
- class **gazebo::physics::ODEMultiRayShape**

  *ODE specific version of **MultiRayShape** (p. 549).*
- class **gazebo::physics::ODEPhysics**

  ODE physics engine.

- class **gazebo::physics::ODEPlaneShape**

  An ODE Plane shape.
- class **gazebo::physics::ODERayShape**

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

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

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

  A ODE sphere shape.
- class **gazebo::physics::ODESurfaceParams**

  Surface params.
- class **gazebo::physics::ODETrimeshShape**

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

  A universal joint.

Typedefs

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

8.7.1 Detailed Description

ODE physics engine wrapper.

8.7.2 Typedef Documentation

8.7.2.1 typedef ODEPhysics ∗ gazebo::physics::ODEPhysicsPtr

8.7.2.2 typedef ODESurfaceParams∗ gazebo::physics::ODESurfaceParamsPtr

8.8 Rendering

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

Namespaces

- namespace **gazebo::rendering**

  Rendering namespace.

Classes

- class **gazebo::rendering::ArrowVisual**

  Basic arrow visualization.
- class **gazebo::rendering::AxisVisual**

  Basic axis visualization.
- class **gazebo::rendering::Camera**

  Basic camera sensor.
- class **gazebo::rendering::CameraVisual**

  Basic camera visualization.
- class **gazebo::rendering::COMVisual**

  Basic Center of Mass visualization.
- class **gazebo::rendering::ContactVisual**

  Contact visualization.
- class **gazebo::rendering::Conversions**

  Conversions *(*p. *305)* ***Conversions.hh*** *(*p. *1018)* ***rendering/Conversions.hh*** *(*p. *1018).*
- class **gazebo::rendering::DepthCamera**

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

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

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

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

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

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

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

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

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

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

Visualization for laser data.

- class **gazebo::rendering::Light**
  A light source.
- class **gazebo::rendering::MovableText**
  Movable text.
- class **gazebo::rendering::OrbitViewController**
  Orbit view controller.
- class **gazebo::rendering::Projector**
  Projects a material onto surface, light a light projector.
- class **gazebo::rendering::RenderEngine**
  Adaptor to Ogre3d.
- class **gazebo::rendering::RFIDTagVisual**
  Visualization for RFID tags sensor.
- class **gazebo::rendering::RFIDVisual**
  Visualization for RFID sensor.
- class **Road**
  Used to render a strip of road.
- class **gazebo::rendering::Road2d**
- class **gazebo::rendering::RTShaderSystem**
  *Implements **Ogre** (p. 118)'s Run-Time Shader system.*
- class **gazebo::rendering::Scene**
  Representation of an entire scene graph.
- class **gazebo::rendering::SelectionObj**
  A graphical selection object.
- class **gazebo::rendering::UserCamera**
  A camera used for user visualization of a scene.
- class **gazebo::rendering::VideoVisual**
  A visual element that displays a video as a texture.
- class **gazebo::rendering::ViewController**
  Base class for view controllers.
- class **gazebo::rendering::Visual**
  A renderable object.
- class **gazebo::rendering::WindowManager**
  Class to mangage render windows.

Functions

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

  *create **rendering::Scene** (p. 746) by name.*
- bool **gazebo::rendering::fini** ()
  teardown rendering engine.
- rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name)
  *get pointer to **rendering::Scene** (p. 746) by name.*
- bool **gazebo::rendering::init** ()
  init rendering engine.
- bool **gazebo::rendering::load** ()
  load rendering engine.
- void **gazebo::rendering::remove_scene** (const std::string &_name)
  *remove a **rendering::Scene** (p. 746) by name*

8.8.1 Detailed Description

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

8.8.2 Function Documentation

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

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

Parameters

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

8.8.2.2 bool gazebo::rendering::fini ( )

teardown rendering engine.

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

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

Parameters

in	*_name*	Name of the scene to retreive.

8.8.2.4 bool gazebo::rendering::init ( )

init rendering engine.

8.8.2.5 bool gazebo::rendering::load ( )

load rendering engine.

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

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

Parameters

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

8.9 Gazebo_parser

Namespaces

- namespace **sdf**

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

  namespace for URDF to SDF parser

Classes

- class **sdf::Element**

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

  Base ***SDF*** *(*p. *762) class.*
- class **urdf2gazebo::URDF2Gazebo**

Typedefs

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

8.9.1 Detailed Description

8.9.2 Typedef Documentation

8.9.2.1 typedef urdf::Collision∗ urdf2gazebo::CollisionPtr

8.9.2.2 typedef urdf::Visual∗ urdf2gazebo::VisualPtr

8.10 Sensors

A set of sensor classes, functions, and definitions.

Files

- file **SensorTypes.hh**

  Forward declarations and typedefs for sensors.

Namespaces

- namespace **gazebo::sensors**

  Sensors namespace.

Classes

- class **gazebo::sensors::CameraSensor**
- class **gazebo::sensors::ContactSensor**
- class **gazebo::sensors::DepthCameraSensor**
- class **gazebo::sensors::GpuRaySensor**
- class **gazebo::sensors::ImuSensor**

  An IMU sensor.
- class **gazebo::sensors::RaySensor**
- class **gazebo::sensors::RFIDSensor**

  Sensor *(*p. *765) class for RFID type of sensor.*
- class **gazebo::sensors::RFIDTag**

  RFIDTag *(*p. *729) to interact with RFIDTagSensors Nate check.*
- class **gazebo::sensors::RFIDTagManager**

  Nate fill in
- class **gazebo::sensors::Sensor**
- class **SensorFactor**

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

  Class to manage and update all sensors.

Macros

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

  Static sensor registration macro.

Functions

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

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

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

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

  initialize the sensor generation loop.
- bool **gazebo::sensors::load** ()
- void **gazebo::sensors::remove_sensor** (const std::string &_sensorName)

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

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

  Run sensor generation continuously. This is a blocking call.
- void **gazebo::sensors::run_once** (bool _force=true)

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

  Stop the sensor generation loop.

8.10.1 Detailed Description

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

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

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

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

GPU based laser sensor.

Contact sensor.

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

This sensor detects and reports contacts between objects

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

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

LoadChild instead of Load, InitChild instead of Init

Base class for sensors

Returns

True if successfully loaded, false if not Nate check

8.10.2 Macro Definition Documentation

8.10.2.1 #define GZ_REGISTER_STATIC_SENSOR(*name, classname*)

Value:

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

Static sensor registration macro.

Use this macro to register sensors with the server.

Parameters

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

Create a sensor using SDF.

Parameters

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

Returns

The name of the new sensor

8.10.3.2 bool gazebo::sensors::fini ( )

shutdown the sensor generation loop.

Returns

True if successfully finalized, false if not

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

Get a sensor using by name.

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

Parameters

in	*_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.

8.10.3.4 bool gazebo::sensors::init ( )

initialize the sensor generation loop.

Returns

True if successfully initialized, false if not

8.10.3.5 bool gazebo::sensors::load ( )

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

Remove a sensor by name.

Parameters

in	*_sensorName*	Name of sensor to remove

8.10.3.7 bool gazebo::sensors::remove_sensors ( )

Remove all sensors.

Returns

True if all successfully removed, false if not

8.10.3.8 void gazebo::sensors::run ( )

Run sensor generation continuously. This is a blocking call.

8.10.3.9 void gazebo::sensors::run_once (bool *_force* = `true`)

Run the sensor generation one step.

Parameters

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

8.10.3.10 void gazebo::sensors::stop ( )

Stop the sensor generation loop.

8.11 Transport

Handles transportation of messages.

Files

- file **TransportTypes.hh**

  Forward declarations for transport.

Namespaces

- namespace **gazebo::transport**

  Transport namespace.

Classes

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

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

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

  *TCP/IP **Connection*** *(p. 288).*
- class **gazebo::transport::ConnectionManager**

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

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

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

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

  Reads data from a remote advertiser, and passes the data along to local subscribers.
- class **gazebo::transport::Publisher**

  A publisher of messages on a topic.
- class **gazebo::transport::SubscribeOptions**

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

  A subscriber to a topic.
- class **gazebo::transport::SubscriptionTransport**

  Handles sending data over the wire to remote subscribers.
- class **gazebo::transport::TopicManager**

  Manages topics and their subscriptions.

Typedefs

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

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

Functions

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

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

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

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

  Return all the namespace (world names) on the master.
- bool **gazebo::transport::init** (const std::string &master_host="", unsigned int master_port=0)

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

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

  Set to true to pause incoming messages.
- msgs::Response **gazebo::transport::request** (const std::string &_worldName, const msgs::Request &_request)

  Send a request, and receive a response.
- void **gazebo::transport::run** ()

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

  Stop the transport component from running.

8.11.1 Detailed Description

Handles transportation of messages.

8.11.2 Typedef Documentation

8.11.2.1 typedef CallbackHelper∗ gazebo::transport::CallbackHelperPtr

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

8.11.3 Function Documentation

8.11.3.1 void gazebo::transport::clear_buffers ( )

clear any remaining communication buffers

8.11.3.2 void gazebo::transport::fini ( )

Cleanup the transport component.

8.11.3.3 bool gazebo::transport::get_master_uri (std::string & *master_host*, unsigned int & *master_port*)

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

Parameters

master_host	The hostname of the master is set to this param
master_port	The port of the master is set to this param

Returns

False if the GAZEBO_MASTER_URI was not found

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

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

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

Initialize the transport system.

Parameters

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

8.11.3.6 bool gazebo::transport::is_stopped ( )

Return true if the transport system is stopped.

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

Set to true to pause incoming messages.

They are still queued for later delivery

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

Send a request, and receive a response.

8.11.3.9 void gazebo::transport::run ( )

Run the transport component.

This starts message passing. This is a blocking call

8.11.3.10 void gazebo::transport::stop ( )

Stop the transport component from running.

Chapter 9

Namespace Documentation

9.1 boost Namespace Reference

9.2 gazebo Namespace Reference

Forward declarations for the common classes.

Namespaces

- namespace **common**

  Common namespace.
- namespace **event**

  Event *(p. 350) namespace.*
- namespace **math**

  Math namespace.
- namespace **msgs**

  Messages namespace.
- namespace **physics**

  namespace for physics
- namespace **rendering**

  Rendering namespace.
- namespace **sensors**

  Sensors namespace.
- namespace **transport**

  Transport namespace.

Classes

- class **Master**

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

  A plugin with access to ***physics::Model*** *(p. 521).*

- class **PluginT**

  A class which all plugins must inherit from.
- class **SensorPlugin**

  A plugin with access to physics::Sensor.
- class **Server**
- class **SystemPlugin**

  A plugin loaded within the gzserver on startup.
- class **VisualPlugin**

  A plugin loaded within the gzserver on startup.
- class **WorldPlugin**

  *A plugin with access to **physics::World** (p. 954).*

Typedefs

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

Enumerations

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

  Used to specify the type of plugin.

Functions

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

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

9.2.1 Detailed Description

Forward declarations for the common classes. Nate check this base class and I can propogate changes to subclasses

9.2.2 Typedef Documentation

9.2.2.1 typedef GUIPlugin* gazebo::GUIPluginPtr

9.2.2.2 typedef ModelPlugin* gazebo::ModelPluginPtr

9.2.2.3 typedef SensorPlugin* gazebo::SensorPluginPtr

9.2.2.4 typedef SystemPlugin* gazebo::SystemPluginPtr

9.2.2.5 typedef VisualPlugin* gazebo::VisualPluginPtr

9.2.2.6 typedef WorldPlugin* gazebo::WorldPluginPtr

9.2.3 Function Documentation

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

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

Find a file in the gazebo search paths.

9.2.3.3 void gazebo::fini ( )

9.2.3.4 bool gazebo::init ( )

9.2.3.5 bool gazebo::load (int *argc* = 0, char ** *argv* = 0)

9.2.3.6 void gazebo::print_version ( )

9.2.3.7 void gazebo::run ( )

9.2.3.8 void gazebo::stop ( )

9.3 gazebo::common Namespace Reference

Common namespace.

Classes

- class **Animation**

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

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

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

  Defines a color.
- class **Console**

Message, error, warning, and logging functionality.

- class **DiagnosticManager**

  A diagnostic manager class.
- class **DiagnosticTimer**

  A timer designed for diagnostics.
- class **Exception**

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

  Encapsulates an image.
- class **KeyFrame**

  A key frame in an animation.
- class **Material**

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

  A 3D mesh.
- class **MeshLoader**

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

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

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

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

  Node animation.
- struct **NodeAssignment**

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

  A transformation node.
- class **NumericAnimation**

  A numeric animation.
- class **NumericKeyFrame**

  *A keyframe for a **NumericAnimation** (p. 569).*
- class **PID**

  *Generic **PID** (p. 664) controller class.*
- class **PoseAnimation**

  A pose animation.
- class **PoseKeyFrame**

  *A keyframe for a **PoseAnimation** (p. 685).*
- class **Skeleton**

  A skeleton.
- class **SkeletonAnimation**

  ***Skeleton** (p. 784) animation.*
- class **SkeletonNode**

  A skeleton node.
- class **STLLoader**

  Class used to load STL mesh files.

- class **SubMesh**

  A child mesh.
- class **SystemPaths**

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

  *A **Time** (p. 840) class, can be used to hold wall- or sim-time.*
- class **Timer**

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

  Handle video encoding and decoding using libavcodec.

Typedefs

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

Functions

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

  *add path prefix to **common::SystemPaths** (p. 834)*
- std::string **find_file** (const std::string &_file, bool _searchLocalPath=true)

  *search for file in **common::SystemPaths** (p. 834)*
- std::string **find_file_path** (const std::string &_file)

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

9.3.1 Detailed Description

Common namespace.

9.3.2 Typedef Documentation

9.3.2.1 typedef Animation* gazebo::common::AnimationPtr

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

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

9.3.2.4 typedef NumericAnimation* gazebo::common::NumericAnimationPtr

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

9.3.2.6 typedef PoseAnimation* gazebo::common::PoseAnimationPtr

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

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

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

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

9.4 gazebo::event Namespace Reference

Event (p. 350) namespace.

Classes

- class **Connection**

  A class that encapsulates a connection.
- class **Event**

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

  *An **Event** (p. 350) class to get notifications for simulator events.*
- class **EventT**

  A class for event processing.

Typedefs

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

9.4.1 Detailed Description

Event (p. 350) namespace.

9.4.2 Typedef Documentation

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

9.4.2.2 typedef Connection* gazebo::event::ConnectionPtr

9.5 gazebo::math Namespace Reference

Math namespace.

Classes

- class **Angle**

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

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

  A 3x3 matrix class.
- class **Matrix4**

  A 3x3 matrix class.
- class **Plane**

  A plane and related functions.
- class **Pose**

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

  A quaternion class.
- class **Rand**

  Random number generator class.
- class **RotationSpline**

  Spline *(p. 808) for rotations.*
- class **Spline**

  Splines.
- class **Vector2d**

  Generic double x, y vector.
- class **Vector2i**

  Generic integer x, y vector.
- class **Vector3**

  The ***Vector3*** *(p. 902) class represents the generic vector containing 3 elements.*
- class **Vector4**

  double Generic x, y, z, w vector

Typedefs

- typedef boost::mt19937 **GeneratorType**
- typedef boost::normal_distribution < double > **NormalRealDist**

- typedef boost::variate_generator < **GeneratorType** &, **NormalRealDist** > **NRealGen**
- typedef boost::variate_generator < **GeneratorType** &, **UniformIntDist** > **UIntGen**
- typedef boost::uniform_int< int > **UniformIntDist**
- typedef boost::uniform_real < double > **UniformRealDist**
- typedef boost::variate_generator < **GeneratorType** &, **UniformRealDist** > **URealGen**

Functions

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

  simple clamping function
- template<typename T >
  bool **equal** (const T &_a, const T &_b, const T &_epsilon=1e-6)

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

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

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

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

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

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

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

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

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

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

  get variance of vector of values

Variables

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

  Not a number.
- static const double **NAN_I** = std::numeric_limits<int>::quiet_NaN()

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

9.5.1 Detailed Description

Math namespace.

9.5.2 Typedef Documentation

9.5.2.1 typedef boost::mt19937 gazebo::math::GeneratorType

9.5.2.2 typedef boost::normal_distribution<double> gazebo::math::NormalRealDist

9.5.2.3 typedef boost::variate_generator<GeneratorType&, NormalRealDist > gazebo::math::NRealGen

9.5.2.4 typedef boost::variate_generator<GeneratorType&, UniformIntDist > gazebo::math::UIntGen

9.5.2.5 typedef boost::uniform_int<int> gazebo::math::UniformIntDist

9.5.2.6 typedef boost::uniform_real<double> gazebo::math::UniformRealDist

9.5.2.7 typedef boost::variate_generator<GeneratorType&, UniformRealDist > gazebo::math::URealGen

9.6 gazebo::msgs Namespace Reference

Messages namespace.

Functions

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

  *Convert a **math::Vector3** (p. 902) to a msgs::Vector3d.*
- msgs::Quaternion **Convert** (const **math::Quaternion** &_q)

  *Convert a **math::Quaternion** (p. 697) to a msgs::Quaternion.*
- msgs::Pose **Convert** (const **math::Pose** &_p)

  *Convert a **math::Pose** (p. 677) to a msgs::Pose.*
- msgs::Color **Convert** (const **common::Color** &_c)

  *Convert a **common::Color** (p. 274) to a msgs::Color.*
- msgs::Time **Convert** (const **common::Time** &_t)

  *Convert a **common::Time** (p. 840) to a msgs::Time.*
- msgs::PlaneGeom **Convert** (const **math::Plane** &_p)

  *Convert a **math::Plane** (p. 668) to a msgs::PlaneGeom.*
- **math::Vector3 Convert** (const msgs::Vector3d &_v)

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

- **math::Quaternion Convert** (const msgs::Quaternion &_q)

  *Convert a msgs::Quaternion to a **math::Quaternion*** (p. *697*).
- **math::Pose Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose*** (p. *677*).
- **common::Color Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color*** (p. *274*).
- **common::Time Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time*** (p. *840*).
- **math::Plane Convert** (const msgs::PlaneGeom &_p)

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

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

  Create a msgs::Fog from a fog SDF element.
- msgs::Header * **GetHeader** (google::protobuf::Message &_message)

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

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

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

  Create a msgs::Light from a light SDF element.
- msgs::Scene **SceneFromSDF** (**sdf::ElementPtr** _sdf)

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

  *Convert a msgs::Image to a **common::Image*** (p. *407*).
- void **Set** (msgs::Image *_msg, const **common::Image** &_i)

  *Set a msgs::Image from a **common::Image*** (p. *407*).
- void **Set** (msgs::Vector3d *_pt, const **math::Vector3** &_v)

  *Set a msgs::Vector3d from a **math::Vector3*** (p. *902*).
- void **Set** (msgs::Vector2d *_pt, const **math::Vector2d** &_v)

  *Set a msgs::Vector2d from a **math::Vector3*** (p. *902*).
- void **Set** (msgs::Quaternion *_q, const **math::Quaternion** &_v)

  *Set a msgs::Quaternion from a **math::Quaternion*** (p. *697*).
- void **Set** (msgs::Pose *_p, const **math::Pose** &_v)

  *Set a msgs::Pose from a **math::Pose*** (p. *677*).
- void **Set** (msgs::Color *_c, const **common::Color** &_v)

  *Set a msgs::Color from a **common::Color*** (p. *274*).
- void **Set** (msgs::Time *_t, const **common::Time** &_v)

  *Set a msgs::Time from a **common::Time*** (p. *840*).
- void **Set** (msgs::PlaneGeom *_p, const **math::Plane** &_v)

  *Set a msgs::Plane from a **math::Plane*** (p. *668*).
- void **Stamp** (msgs::Header *_header)

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

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

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

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

9.6.1 Detailed Description

Messages namespace.

9.7 gazebo::physics Namespace Reference

namespace for physics

Classes

- class **Actor**

  Actor *(p. 121) class enables GPU based mesh model / skeleton scriptable animation.*
- class **BallJoint**

  Base *(p. 145) class for a ball joint.*
- class **Base**

  Base *(p. 145) class for most physics classes.*
- class **BoxShape**

  Box geometry primitive.
- class **BulletBallJoint**

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

  Bullet box collision.
- class **BulletCollision**

  Bullet collisions.
- class **BulletCylinderShape**

  Cylinder collision.
- class **BulletHeightmapShape**

  Height map collision.
- class **BulletHinge2Joint**

  A two axis hinge joint.
- class **BulletHingeJoint**

  A single axis hinge joint.
- class **BulletJoint**

  Base *(p. 145) class for all joints.*
- class **BulletLink**

  *Bullet **Link** (p. 454) class.*
- class **BulletMotionState**

  Bullet btMotionState encapsulation.
- class **BulletMultiRayShape**

  *Bullet specific version of **MultiRayShape** (p. 549).*
- class **BulletPhysics**

  Bullet physics engine.
- class **BulletPlaneShape**

  Bullet collision for an infinite plane.
- class **BulletRaySensor**

  An Bullet Ray sensor.

- class **BulletRayShape**

  Ray shape for bullet.
- class **BulletScrewJoint**

  A screw joint.
- class **BulletSliderJoint**

  A slider joint.
- class **BulletSphereShape**

  Bullet sphere collision.
- class **BulletTrimeshShape**

  Triangle mesh collision.
- class **BulletUniversalJoint**

  A bullet universal joint class.
- class **Collision**

  Base *(*p. *145) class for all collision entities.*
- class **CollisionState**

  Store state information of a ***physics::Collision*** *(*p. *262) object.*
- class **Contact**

  A contact between two collisions.
- class **ContactFeedback**

  data structure for contact feedbacks
- class **CylinderShape**

  Cylinder collision.
- class **Entity**

  Base *(*p. *145) class for all physics objects in Gazebo.*
- class **Gripper**

  A gripper abstraction.
- class **HeightmapShape**

  HeightmapShape *(*p. *399) collision shape builds a heightmap from an image.*
- class **Hinge2Joint**

  A two axis hinge joint.
- class **HingeJoint**

  A single axis hinge joint.
- class **Inertial**

  A class for inertial information about a link.
- class **Joint**

  Base *(*p. *145) class for all joints.*
- class **JointController**

  A class for manipulating ***physics::Joint*** *(*p. *423).*
- class **JointFeedback**

  Feedback information from a joint.
- class **JointState**

  keeps track of state of a ***physics::Joint*** *(*p. *423)*
- class **Link**

  Link *(*p. *454) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.*
- class **LinkState**

  Store state information of a ***physics::Link*** *(*p. *454) object.*

- class **Logger**

  Handles logging of data to disk.
- class **Logplay**

  Open and playback log files that were recorded using ***Logger*** *(*p. *477).*
- class **MapShape**

  Creates box extrusions based on an image.
- class **Model**

  A model is a collection of links, joints, and plugins.
- class **ModelState**

  Store state information of a ***physics::Model*** *(*p. *521) object.*
- class **MultiRayShape**

  Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.
- class **ODEBallJoint**

  An ***ODEBallJoint*** *(*p. *573).*
- class **ODEBoxShape**

  ODE Box shape.
- class **ODECollision**

  Base *(*p. *145) class for all ODE collisions.*
- class **ODECylinderShape**

  ODE cylinder shape.
- class **ODEHeightmapShape**

  ODE Height map collision.
- class **ODEHinge2Joint**

  A two axis hinge joint.
- class **ODEHingeJoint**

  A single axis hinge joint.
- class **ODEJoint**

  ODE joint interface.
- class **ODELink**

  ODE ***Link*** *(*p. *454) class.*
- class **ODEMultiRayShape**

  ODE specific version of ***MultiRayShape*** *(*p. *549).*
- class **ODEPhysics**

  ODE physics engine.
- class **ODEPlaneShape**

  An ODE Plane shape.
- class **ODERayShape**

  Ray collision.
- class **ODEScrewJoint**

  A screw joint.
- class **ODESliderJoint**

  A slider joint.
- class **ODESphereShape**

  A ODE sphere shape.
- class **ODESurfaceParams**

  Surface params.
- class **ODETrimeshShape**

Triangle mesh collision.

- class **ODEUniversalJoint**

  A universal joint.
- class **PhysicsEngine**

  Base *(p. 145) class for a physics engine.*
- class **PhysicsFactory**

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

  Collision *(p. 262) for an infinite plane.*
- class **RayShape**

  Base *(p. 145) class for Ray collision geometry.*
- class **Road**

  for building a ***Road*** *(p. 736) from SDF*
- class **ScrewJoint**

  A screw joint.
- class **Shape**

  Base *(p. 145) class for all shapes.*
- class **SliderJoint**

  A slider joint.
- class **SphereShape**

  Sphere collision.
- class **State**

  State *(p. 813) of an entity.*
- class **SurfaceParams**

  SurfaceParams *(p. 830) defines various Surface contact parameters.*
- struct **TrajectoryInfo**
- class **TrimeshShape**

  Triangle mesh collision shape.
- class **UniversalJoint**

  A universal joint.
- class **World**

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

  Store state information of a ***physics::World*** *(p. 954) object.*

Typedefs

- typedef std::vector< **ActorPtr** > **Actor_V**
- typedef **Actor** * **ActorPtr**
- typedef std::vector< **BasePtr** > **Base_V**
- typedef **Base** * **BasePtr**
- typedef **BoxShape** * **BoxShapePtr**
- typedef **BulletCollision** * **BulletCollisionPtr**
- typedef **BulletLink** * **BulletLinkPtr**
- typedef **BulletPhysics** * **BulletPhysicsPtr**
- typedef **BulletRayShape** * **BulletRayShapePtr**
- typedef std::vector< **CollisionPtr** > **Collision_V**
- typedef **Collision** * **CollisionPtr**

- typedef **Contact** ∗ **ContactPtr**
- typedef **CylinderShape** ∗ **CylinderShapePtr**
- typedef **Entity** ∗ **EntityPtr**
- typedef **HeightmapShape** ∗ **HeightmapShapePtr**
- typedef **Inertial** ∗ **InertialPtr**
- typedef std::vector< **JointPtr** > **Joint_V**
- typedef **Joint** ∗ **JointPtr**
- typedef std::vector< **LinkPtr** > **Link_V**
- typedef **Link** ∗ **LinkPtr**
- typedef MeshShape ∗ **MeshShapePtr**
- typedef std::vector< **ModelPtr** > **Model_V**
- typedef **Model** ∗ **ModelPtr**
- typedef **MultiRayShape** ∗ **MultiRayShapePtr**
- typedef **ODECollision** ∗ **ODECollisionPtr**
- typedef **ODELink** ∗ **ODELinkPtr**
- typedef **ODEPhysics** ∗ **ODEPhysicsPtr**
- typedef **ODERayShape** ∗ **ODERayShapePtr**
- typedef **ODESurfaceParams** ∗ **ODESurfaceParamsPtr**
- typedef **PhysicsEngine** ∗ **PhysicsEnginePtr**
- typedef **PhysicsEnginePtr**(∗ **PhysicsFactoryFn**)(**WorldPtr** world)
- typedef **RayShape** ∗ **RayShapePtr**
- typedef **Road** ∗ **RoadPtr**
- typedef **Shape** ∗ **ShapePtr**
- typedef **SphereShape** ∗ **SphereShapePtr**
- typedef **SurfaceParams** ∗ **SurfaceParamsPtr**
- typedef **World** ∗ **WorldPtr**

Functions

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

  Create a world given a name.
- bool **fini** ()

  *Finalize transport by calling **gazebo::transport::fini** (p. 90).*
- **WorldPtr get_world** (const std::string &_name="")

  Returns a pointer to a world by name.
- void **init_world** (**WorldPtr** _world)

  Init world given a pointer to it.
- void **init_worlds** ()

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

  *Setup **gazebo::SystemPlugin** (p. 838)'s and call **gazebo::transport::init** (p. 90).*
- void **load_world** (**WorldPtr** _world, **sdf::ElementPtr** _sdf)

  *Load world from **sdf::Element** (p. 332) pointer.*
- void **load_worlds** (**sdf::ElementPtr** _sdf)

  *load multiple worlds from single **sdf::Element** (p. 332) pointer*
- void **pause_world** (**WorldPtr** _world, bool _pause)

  *Pause world by calling **World::SetPaused** (p. 964).*
- void **pause_worlds** (bool pause)

  pause multiple worlds stored in static variable gazebo::g_worlds

- void **remove_worlds** ()

  remove multiple worlds stored in static variable gazebo::g_worlds
- void **run_world** (**WorldPtr** _world)

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

  run multiple worlds stored in static variable gazebo::g_worlds
- void **stop_world** (**WorldPtr** _world)

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

  stop multiple worlds stored in static variable gazebo::g_worlds

Variables

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

  String names for the different entity types.

9.7.1 Detailed Description

namespace for physics Physics forward declarations and type defines.

physics namespace

9.7.2 Typedef Documentation

9.7.2.1 typedef std::vector<ActorPtr> gazebo::physics::Actor_V

9.7.2.2 typedef Actor* gazebo::physics::ActorPtr

9.7.2.3 typedef std::vector<BasePtr> gazebo::physics::Base_V

9.7.2.4 typedef Base* gazebo::physics::BasePtr

9.7.2.5 typedef BoxShape* gazebo::physics::BoxShapePtr

9.7.2.6 typedef BulletCollision* gazebo::physics::BulletCollisionPtr

9.7.2.7 typedef BulletLink* gazebo::physics::BulletLinkPtr

9.7.2.8 typedef BulletPhysics* gazebo::physics::BulletPhysicsPtr

9.7.2.9 typedef BulletRayShape* gazebo::physics::BulletRayShapePtr

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

9.7.2.11 typedef Collision* gazebo::physics::CollisionPtr

9.7.2.12 typedef Contact* gazebo::physics::ContactPtr

9.7.2.13 typedef CylinderShape* gazebo::physics::CylinderShapePtr

9.7.2.14 typedef Entity* gazebo::physics::EntityPtr

9.7.2.15 typedef HeightmapShape* gazebo::physics::HeightmapShapePtr

9.7.2.16 typedef Inertial* gazebo::physics::InertialPtr

9.7.2.17 typedef std::vector<JointPtr> gazebo::physics::Joint_V

9.7.2.18 typedef Joint* gazebo::physics::JointPtr

9.7.2.19 typedef std::vector<LinkPtr> gazebo::physics::Link_V

9.7.2.20 typedef Link* gazebo::physics::LinkPtr

9.7.2.21 typedef MeshShape* gazebo::physics::MeshShapePtr

9.7.2.22 typedef std::vector<ModelPtr> gazebo::physics::Model_V

9.7.2.23 typedef Model* gazebo::physics::ModelPtr

9.7.2.24 typedef MultiRayShape* gazebo::physics::MultiRayShapePtr

9.7.2.25 typedef ODECollision* gazebo::physics::ODECollisionPtr

9.7.2.26 typedef ODELink* gazebo::physics::ODELinkPtr

9.7.2.27 typedef ODERayShape* gazebo::physics::ODERayShapePtr

9.7.2.28 typedef PhysicsEngine* gazebo::physics::PhysicsEnginePtr

9.7.2.29 typedef RayShape* gazebo::physics::RayShapePtr

9.7.2.30 typedef Road* gazebo::physics::RoadPtr

9.7.2.31 typedef Shape* gazebo::physics::ShapePtr

9.7.2.32 typedef SphereShape* gazebo::physics::SphereShapePtr

9.7.2.33 typedef SurfaceParams* gazebo::physics::SurfaceParamsPtr

9.7.2.34 typedef World* gazebo::physics::WorldPtr

9.8 gazebo::rendering Namespace Reference

Rendering namespace.

Classes

- class **ArrowVisual**

  Basic arrow visualization.
- class **AxisVisual**

Basic axis visualization.

- class **Camera**

  Basic camera sensor.
- class **CameraVisual**

  Basic camera visualization.
- class **COMVisual**

  Basic Center of Mass visualization.
- class **ContactVisual**

  Contact visualization.
- class **Conversions**

  Conversions *(*p. *305)* ***Conversions.hh*** *(*p. *1018)* ***rendering/Conversions.hh*** *(*p. *1018).*
- class **DepthCamera**

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

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

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

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

  First Person Shooter style view controller.
- class **GpuLaser**

  GPU based laser distance sensor.
- class **Grid**

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

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

  Rendering a terrain using heightmap information.
- class **JointVisual**

  Visualization for joints.
- class **LaserVisual**

  Visualization for laser data.
- class **Light**

  A light source.
- class **MovableText**

  Movable text.
- class **OrbitViewController**

  Orbit view controller.
- class **Projector**

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

  Adaptor to Ogre3d.
- class **RFIDTagVisual**

  Visualization for RFID tags sensor.
- class **RFIDVisual**

  Visualization for RFID sensor.

- class **Road2d**
- class **RTShaderSystem**

  *Implements **Ogre** (p. 118)'s Run-Time Shader system.*
- class **Scene**

  Representation of an entire scene graph.
- class **SelectionObj**

  A graphical selection object.
- class **UserCamera**

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

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

  Base class for view controllers.
- class **Visual**

  A renderable object.
- class **WindowManager**

  Class to mangage render windows.

Typedefs

- typedef **ArrowVisual** ∗ **ArrowVisualPtr**
- typedef **AxisVisual** ∗ **AxisVisualPtr**
- typedef **Camera** ∗ **CameraPtr**
- typedef **CameraVisual** ∗ **CameraVisualPtr**
- typedef **COMVisual** ∗ **COMVisualPtr**
- typedef **ContactVisual** ∗ **ContactVisualPtr**
- typedef **DepthCamera** ∗ **DepthCameraPtr**
- typedef **DynamicLines** ∗ **DynamicLinesPtr**
- typedef **GpuLaser** ∗ **GpuLaserPtr**
- typedef **JointVisual** ∗ **JointVisualPtr**
- typedef **LaserVisual** ∗ **LaserVisualPtr**
- typedef **Light** ∗ **LightPtr**
- typedef **RFIDTagVisual** ∗ **RFIDTagVisualPtr**
- typedef **RFIDVisual** ∗ **RFIDVisualPtr**
- typedef **Scene** ∗ **ScenePtr**
- typedef **UserCamera** ∗ **UserCameraPtr**
- typedef **Visual** ∗ **VisualPtr**

Enumerations

- enum **RenderOpType** {
  RENDERING_POINT_LIST = 0, **RENDERING_LINE_LIST** = 1, **RENDERING_LINE_STRIP** = 2, **RENDERING-_TRIANGLE_LIST** = 3,
  RENDERING_TRIANGLE_STRIP = 4, **RENDERING_TRIANGLE_FAN** = 5, **RENDERING_MESH_RESOURCE** = 6 }

  Type of render operation for a drawable.

Functions

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

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

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

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

  init rendering engine.
- bool **load** ()

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

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

9.8.1 Detailed Description

Rendering namespace.

9.8.2 Typedef Documentation

9.8.2.1 typedef ArrowVisual∗ gazebo::rendering::ArrowVisualPtr

9.8.2.2 typedef AxisVisual∗ gazebo::rendering::AxisVisualPtr

9.8.2.3 typedef Camera∗ gazebo::rendering::CameraPtr

9.8.2.4 typedef CameraVisual∗ gazebo::rendering::CameraVisualPtr

9.8.2.5 typedef COMVisual∗ gazebo::rendering::COMVisualPtr

9.8.2.6 typedef ContactVisual∗ gazebo::rendering::ContactVisualPtr

9.8.2.7 typedef DepthCamera∗ gazebo::rendering::DepthCameraPtr

9.8.2.8 typedef DynamicLines∗ gazebo::rendering::DynamicLinesPtr

9.8.2.9 typedef GpuLaser∗ gazebo::rendering::GpuLaserPtr

9.8.2.10 typedef JointVisual∗ gazebo::rendering::JointVisualPtr

9.8.2.11 typedef LaserVisual∗ gazebo::rendering::LaserVisualPtr

9.8.2.12 typedef Light∗ gazebo::rendering::LightPtr

9.8.2.13 typedef RFIDTagVisual∗ gazebo::rendering::RFIDTagVisualPtr

9.8.2.14 typedef RFIDVisual∗ gazebo::rendering::RFIDVisualPtr

9.8.2.15 typedef Scene* gazebo::rendering::ScenePtr

9.8.2.16 typedef UserCamera* gazebo::rendering::UserCameraPtr

9.8.2.17 typedef Visual* gazebo::rendering::VisualPtr

9.8.3 Enumeration Type Documentation

9.8.3.1 enum gazebo::rendering::RenderOpType

Type of render operation for a drawable.

Enumerator:

RENDERING_POINT_LIST A list of points, 1 vertex per point.

RENDERING_LINE_LIST A list of lines, 2 vertices per line.

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

RENDERING_TRIANGLE_LIST A list of triangles, 3 vertices per triangle.

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

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

RENDERING_MESH_RESOURCE N/A.

9.9 gazebo::sensors Namespace Reference

Sensors namespace.

Classes

- class **CameraSensor**
- class **ContactSensor**
- class **DepthCameraSensor**
- class **GpuRaySensor**
- class **ImuSensor**

  An IMU sensor.
- class **RaySensor**
- class **RFIDSensor**

  Sensor (p. *765) class for RFID type of sensor.*
- class **RFIDTag**

  RFIDTag (p. *729) to interact with RFIDTagSensors Nate check.*
- class **RFIDTagManager**

  Nate fill in
- class **Sensor**
- class **SensorFactory**
- class **SensorManager**

  Class to manage and update all sensors.

Typedefs

- typedef std::vector < **CameraSensorPtr** > **CameraSensor_V**
- typedef **CameraSensor** * **CameraSensorPtr**
- typedef std::vector < **ContactSensorPtr** > **ContactSensor_V**
- typedef **ContactSensor** * **ContactSensorPtr**
- typedef std::vector < **DepthCameraSensorPtr** > **DepthCameraSensor_V**
- typedef **DepthCameraSensor** * **DepthCameraSensorPtr**
- typedef std::vector < **GpuRaySensorPtr** > **GpuRaySensor_V**
- typedef **GpuRaySensor** * **GpuRaySensorPtr**
- typedef std::vector< **RaySensorPtr** > **RaySensor_V**
- typedef **RaySensor** * **RaySensorPtr**
- typedef std::vector< **RFIDSensor** > **RFIDSensor_V**
- typedef **RFIDSensor** * **RFIDSensorPtr**
- typedef std::vector< **RFIDTag** > **RFIDTag_V**
- typedef **RFIDTag** * **RFIDTagPtr**
- typedef std::vector< **SensorPtr** > **Sensor_V**
- typedef **Sensor** *(* **SensorFactoryFn**)()
- typedef **Sensor** * **SensorPtr**

Functions

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

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

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

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

  initialize the sensor generation loop.
- bool **load** ()
- void **remove_sensor** (const std::string &_sensorName)

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

  Remove all sensors.
- void **run** ()

  Run sensor generation continuously. This is a blocking call.
- void **run_once** (bool _force=true)

  Run the sensor generation one step.
- void **stop** ()

  Stop the sensor generation loop.

9.9.1 Detailed Description

Sensors namespace.

9.9.2 Typedef Documentation

9.9.2.1 typedef std::vector<CameraSensorPtr> gazebo::sensors::CameraSensor_V

9.9.2.2 typedef CameraSensor* gazebo::sensors::CameraSensorPtr

9.9.2.3 typedef std::vector<ContactSensorPtr> gazebo::sensors::ContactSensor_V

9.9.2.4 typedef ContactSensor* gazebo::sensors::ContactSensorPtr

9.9.2.5 typedef std::vector<DepthCameraSensorPtr> gazebo::sensors::DepthCameraSensor_V

9.9.2.6 typedef DepthCameraSensor* gazebo::sensors::DepthCameraSensorPtr

9.9.2.7 typedef std::vector<GpuRaySensorPtr> gazebo::sensors::GpuRaySensor_V

9.9.2.8 typedef GpuRaySensor* gazebo::sensors::GpuRaySensorPtr

9.9.2.9 typedef std::vector<RaySensorPtr> gazebo::sensors::RaySensor_V

9.9.2.10 typedef RaySensor* gazebo::sensors::RaySensorPtr

9.9.2.11 typedef std::vector<RFIDSensor> gazebo::sensors::RFIDSensor_V

9.9.2.12 typedef RFIDSensor* gazebo::sensors::RFIDSensorPtr

9.9.2.13 typedef std::vector<RFIDTag> gazebo::sensors::RFIDTag_V

9.9.2.14 typedef RFIDTag* gazebo::sensors::RFIDTagPtr

9.9.2.15 typedef std::vector<SensorPtr> gazebo::sensors::Sensor_V

9.9.2.16 typedef Sensor*(* gazebo::sensors::SensorFactoryFn)()

9.9.2.17 typedef Sensor* gazebo::sensors::SensorPtr

9.10 gazebo::transport Namespace Reference

Transport namespace.

Classes

- class **CallbackHelper**

  A helper class to handle callbacks when messages arrive.
- class **CallbackHelperT**

  Callback helper Template.
- class **Connection**

  *TCP/IP **Connection** (p. 288).*
- class **ConnectionManager**

  Manager of connections.

- class **DebugCallbackHelper**
- class **IOManager**

  Managers boost::asio IO.
- class **Node**

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

  A publication for a topic.
- class **PublicationTransport**

  Reads data from a remote advertiser, and passes the data along to local subscribers.
- class **Publisher**

  A publisher of messages on a topic.
- class **SubscribeOptions**

  Options for a subscription.
- class **Subscriber**

  A subscriber to a topic.
- class **SubscriptionTransport**

  Handles sending data over the wire to remote subscribers.
- class **TopicManager**

  Manages topics and their subscriptions.

Typedefs

- typedef **CallbackHelper** ∗ **CallbackHelperPtr**

  boost shared pointer to ***transport::CallbackHelper*** *(*p. *230)*
- typedef **Connection** ∗ **ConnectionPtr**
- typedef **Node** ∗ **NodePtr**
- typedef **Publication** ∗ **PublicationPtr**
- typedef **PublicationTransport** ∗ **PublicationTransportPtr**
- typedef **Publisher** ∗ **PublisherPtr**
- typedef **Subscriber** ∗ **SubscriberPtr**
- typedef **SubscriptionTransport** ∗ **SubscriptionTransportPtr**

Functions

- void **clear_buffers** ()

  clear any remaining communication buffers
- void **fini** ()

  Cleanup the transport component.
- bool **get_master_uri** (std::string &master_host, unsigned int &master_port)

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

  Return all the namespace (world names) on the master.
- bool **init** (const std::string &master_host="", unsigned int master_port=0)

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

  Return true if the transport system is stopped.
- void **pause_incoming** (bool _pause)

Set to true to pause incoming messages.

- msgs::Response **request** (const std::string &_worldName, const msgs::Request &_request)

  Send a request, and receive a response.

- void **run** ()

  Run the transport component.

- void **stop** ()

  Stop the transport component from running.

9.10.1 Detailed Description

Transport namespace.

9.10.2 Typedef Documentation

9.10.2.1 typedef Connection* gazebo::transport::ConnectionPtr

9.10.2.2 typedef Node* gazebo::transport::NodePtr

9.10.2.3 typedef Publication* gazebo::transport::PublicationPtr

9.10.2.4 typedef PublicationTransport* gazebo::transport::PublicationTransportPtr

9.10.2.5 typedef Publisher* gazebo::transport::PublisherPtr

9.10.2.6 typedef Subscriber* gazebo::transport::SubscriberPtr

9.10.2.7 typedef SubscriptionTransport* gazebo::transport::SubscriptionTransportPtr

9.11 google Namespace Reference

Namespaces

- namespace **protobuf**

9.12 google::protobuf Namespace Reference

Namespaces

- namespace **compiler**

9.13 google::protobuf::compiler Namespace Reference

Namespaces

- namespace **cpp**

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

Classes

- class **GazeboGenerator**

  *Google protobuf message generator for **gazebo::msgs*** (p. *101*).

9.15 Ogre Namespace Reference

9.16 ogre Namespace Reference

9.17 sdf Namespace Reference

namespace for Simulation Description Format parser

Classes

- class **Converter**

  *Convert from one version of **SDF*** (p. *762*) *to another.*
- class **Element**

  SDF (p. *762*) ***Element*** (p. *332*) *class.*
- class **Param**

  A parameter class.
- class **ParamT**

  Templatized parameter class.
- class **Plugin**
- class **SDF**

  *Base **SDF*** (p. *762*) *class.*

Typedefs

- typedef **Element** ∗ **ElementPtr**
- typedef std::vector< **ElementPtr** > **ElementPtr_V**
- typedef std::vector< **ParamPtr** > **Param_V**
- typedef **Param** ∗ **ParamPtr**
- typedef **SDF** ∗ **SDFPtr**

Functions

- void **addNestedModel** (**ElementPtr** _sdf, **ElementPtr** _includeSDF)
- void **copyChildren** (**ElementPtr** _sdf, TiXmlElement ∗_xml)
- bool **init** (**SDFPtr** _sdf)

  Init based on the installed sdf_format.xml file.
- bool **initDoc** (TiXmlDocument ∗_xmlDoc, **SDFPtr** _sdf)
- bool **initDoc** (TiXmlDocument ∗_xmlDoc, **ElementPtr** _sdf)

- bool **initFile** (const std::string &_filename, **SDFPtr** _sdf)
- bool **initFile** (const std::string &_filename, **ElementPtr** _sdf)
- bool **initString** (const std::string &_xmlString, **SDFPtr** _sdf)
- bool **initXml** (TiXmlElement *_xml, **ElementPtr** _sdf)
- bool **readDoc** (TiXmlDocument *_xmlDoc, **SDFPtr** _sdf, const std::string &_source)

  *Populate the **SDF** (p. 762) values from a TinyXML document.*
- bool **readDoc** (TiXmlDocument *_xmlDoc, **ElementPtr** _sdf, const std::string &_source)
- bool **readFile** (const std::string &_filename, **SDFPtr** _sdf)

  *Populate the **SDF** (p. 762) values from a file.*
- bool **readString** (const std::string &_xmlString, **SDFPtr** _sdf)

  *Populate the **SDF** (p. 762) values from a string.*
- bool **readString** (const std::string &_xmlString, **ElementPtr** _sdf)
- bool **readXml** (TiXmlElement *_xml, **ElementPtr** _sdf)

9.17.1 Detailed Description

namespace for Simulation Description Format parser

9.17.2 Typedef Documentation

9.17.2.1 typedef Element* sdf::ElementPtr

9.17.2.2 typedef std::vector< ElementPtr > sdf::ElementPtr_V

9.17.2.3 typedef std::vector< ParamPtr > sdf::Param_V

9.17.2.4 typedef Param* sdf::ParamPtr

9.17.2.5 typedef SDF* sdf::SDFPtr

9.17.3 Function Documentation

9.17.3.1 void sdf::addNestedModel (ElementPtr *_sdf,* ElementPtr *_includeSDF*)

9.17.3.2 void sdf::copyChildren (ElementPtr *_sdf,* TiXmlElement * *_xml*)

9.17.3.3 bool sdf::init (SDFPtr *_sdf*)

Init based on the installed sdf_format.xml file.

9.17.3.4 bool sdf::initDoc (TiXmlDocument * *_xmlDoc,* SDFPtr *_sdf*)

9.17.3.5 bool sdf::initDoc (TiXmlDocument * *_xmlDoc,* ElementPtr *_sdf*)

9.17.3.6 bool sdf::initFile (const std::string & *_filename,* SDFPtr *_sdf*)

9.17.3.7 bool sdf::initFile (const std::string & *_filename,* ElementPtr *_sdf*)

9.17.3.8 bool sdf::initString (const std::string & *_xmlString,* SDFPtr *_sdf*)

9.17.3.9 bool sdf::initXml (TiXmlElement * *_xml,* ElementPtr *_sdf*)

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

Populate the **SDF** (p. 762) values from a TinyXML document.

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

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

Populate the **SDF** (p. 762) values from a file.

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

Populate the **SDF** (p. 762) values from a string.

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

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

9.18 SkyX Namespace Reference

9.19 urdf2gazebo Namespace Reference

namespace for URDF to SDF parser

Classes

- class **GazeboExtension**
- class **URDF2Gazebo**

Typedefs

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

9.19.1 Detailed Description

namespace for URDF to SDF parser

Chapter 10

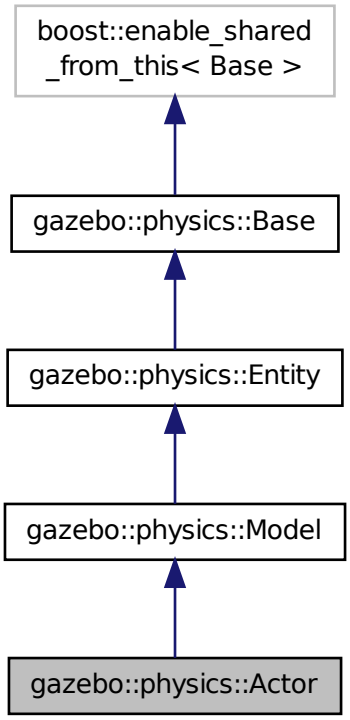
Class Documentation

10.1 gazebo::physics::Actor Class Reference

Actor (p. 121) class enables GPU based mesh model / skeleton scriptable animation.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Actor:



Public Member Functions

- **Actor** (**BasePtr parent**)

  Constructor.
- virtual ∼**Actor** ()

  Destructor.
- virtual void **Fini** ()

  Finalize the actor.
- virtual const **sdf::ElementPtr GetSDF** ()

  Get the SDF values for the actor.
- virtual void **Init** ()

  Initialize the actor.
- virtual bool **IsActive** ()

  Returns true when actor is playing animation.
- void **Load** (**sdf::ElementPtr** _sdf)

  Load the actor.
- virtual void **Play** ()

  Start playing the script.
- virtual void **Stop** ()

  Stop playing the script.
- void **Update** ()

  Update the actor.
- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  update the parameters using new sdf values.

Protected Attributes

- bool **active**

  True if the actor is being updated.
- bool **autoStart**

  True if the actor should start running automatically.
- **transport::PublisherPtr bonePosePub**

  Where to send bone info.
- std::map< std::string, bool > **interpolateX**

  True to interpolate along x direction.
- **math::Vector3 lastPos**

  Last position of the actor.
- double **lastScriptTime**

  Time the scipt was last updated.
- unsigned int **lastTraj**

  THe last trajectory.
- bool **loop**

  True if the animation should loop.
- **LinkPtr mainLink**

  Base *(*p. *145) link.*
- const **common::Mesh** ∗ **mesh**

  Pointer to the actor's mesh.

- std::string **oldAction**

  THe old action.
- double **pathLength**

  Length of the actor's path.
- **common::Time playStartTime**

  Time when the animation was started.
- **common::Time prevFrameTime**

  Time of the previous frame.
- double **scriptLength**

  Time length of a scipt.
- std::map< std::string, **common::SkeletonAnimation** * > **skelAnimation**

  Skeleton animations.
- **common::Skeleton** * **skeleton**

  The actor's skeleton.
- std::map< std::string, std::map< std::string, std::string > > **skelNodesMap**

  Skeleton to naode map.
- std::string **skinFile**

  Filename for the skin.
- double **skinScale**

  Scaling factor to apply to the skin.
- double **startDelay**

  Amount of time to delay start by.
- std::map< unsigned int, **common::PoseAnimation** * > **trajectories**

  All the trajectories.
- std::vector< **TrajectoryInfo** > **trajInfo**

  Trajectory information.
- std::string **visualName**

  Name of the visual.

Additional Inherited Members

10.1.1 Detailed Description

Actor (p. 121) class enables GPU based mesh model / skeleton scriptable animation.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 gazebo::physics::Actor::Actor (**BasePtr** *parent*) [explicit]

Constructor.

Parameters

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. 525).

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

Get the SDF values for the actor.

Returns

Pointer to the SDF values.

Reimplemented from **gazebo::physics::Model** (p. 528).

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

Initialize the actor.

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

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

Returns true when actor is playing animation.

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

Load the actor.

Parameters

in	*_sdf*	SDF parameters

Reimplemented from **gazebo::physics::Entity** (p. 346).

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

Start playing the script.

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

Stop playing the script.

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

Update the actor.

Reimplemented from **gazebo::physics::Base** (p. 157).

10.1.3.9 virtual void gazebo::physics::Actor::UpdateParameters (sdf::ElementPtr _sdf) [virtual]

update the parameters using new sdf values.

Parameters

in	*_sdf*	SDF values to update from.

Reimplemented from **gazebo::physics::Model** (p. 533).

10.1.4 Member Data Documentation

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

True if the actor is being updated.

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

True if the actor should start running automatically.

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

Where to send bone info.

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

True to interpolate along x direction.

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

Last position of the actor.

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

Time the 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. 145) link.

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

Pointer to the actor's mesh.

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

THe old action.

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

Length of the actor's path.

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

Time when the animation was started.

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

Time of the previous frame.

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

Time length of a 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 std::string gazebo::physics::Actor::visualName [protected]

Name of the visual.

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

- **Actor.hh**

10.2 gazebo::math::Angle Class Reference

An angle and related functions.

`#include <Angle.hh>`

Public Member Functions

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

Get the angle in degrees.

- double **GetAsRadian** () const __attribute__((deprecated))

  Get the angle in radians.
- void **Normalize** ()

  Normalize the angle in the range -Pi to Pi.
- bool **operator!=** (const **Angle** &_angle) const

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

  Dereference operator.
- **Angle operator*** (const **Angle** &_angle) const

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

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

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

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

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

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

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

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

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

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

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

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

  Greater or equal operator.
- double **Radian** () const

  Get the angle in radians.
- void **SetFromDegree** (double _degree)

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

  Set the value from an angle in radians.

Friends

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

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

  Stream extraction operator.

10.2.1 Detailed Description

An angle and related functions.

10.2.2 Constructor & Destructor Documentation

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

Constructor.

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

Copy Constructor.

Parameters

in	*_radian*	Radians

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

Copy constructor.

Parameters

in	*_angle*	**Angle** (p. 127) to copy

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

Destructor.

10.2.3 Member Function Documentation

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

Get the angle in degrees.

Returns

double containing the angle's degree value

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

Get the angle in degrees.

Returns

double containing the angle's degree value

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

Get the angle in radians.

Returns

double containing the angle's radian value

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

Normalize the angle in the range -Pi to Pi.

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

Inequality.

Parameters

in	*_angle*	**Angle** (p. 127) to check for inequality

Returns

true if this != _angle

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

Dereference operator.

Returns

Double containing the angle's radian value

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

Multiplication operator, result = this * _angle.

Parameters

in	*_angle*	**Angle** (p. 127) for multiplication

Returns

the new angle

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

Multiplication set, this = this * _angle.

Parameters

in	*_angle*	**Angle** (p. 127) for multiplication

Returns

angle

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

Addition operator, result = this + _angle.

Parameters

in	*_angle*	**Angle** (p. 127) for addition

Returns

the new angle

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

Addition set, this = this + _angle.

Parameters

in	*_angle*	**Angle** (p. 127) for addition

Returns

angle

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

Substraction, result = this - _angle.

Parameters

in	*_angle*	**Angle** (p. 127) for substraction

Returns

the new angle

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

Subtraction set, this = this - _angle.

Parameters

in	*_angle*	**Angle** (p. 127) for subtraction

Returns

angle

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

Division, result = this / _angle.

Parameters

in	*_angle*	**Angle** (p. 127) for division

Returns

the new angle

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

Division set, this = this / _angle.

Parameters

in	*_angle*	**Angle** (p. 127) for division

Returns

angle

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

Less than operator.

Parameters

in	*_angle*	**Angle** (p. 127) to check

Returns

true if this < _angle

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

Less or equal operator.

Parameters

in	*_angle*	**Angle** (p. 127) to check

Returns

true if this <= _angle

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

Equality operator, result = this == _angle.

Parameters

in	*_angle*	**Angle** (p. 127) to check for equality

Returns

true if this == _angle

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

Greater than operator.

Parameters

in	*_angle*	**Angle** (p. 127) to check

Returns

true if this > _angle

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

Greater or equal operator.

Parameters

in	*_angle*	**Angle** (p. 127) to check

Returns

true if this >= _angle

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

Get the angle in radians.

Returns

double containing the angle's radian value

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

Set the value from an angle in degrees.

Parameters

in	*_degree*	Degree value

10.2.3.22 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

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

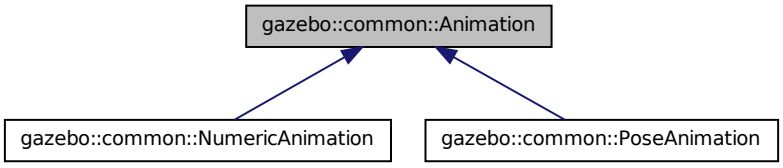
- **Angle.hh**

10.3 gazebo::common::Animation Class Reference

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

`#include <Animation.hh>`

Inheritance diagram for gazebo::common::Animation:



Public Member Functions

- **Animation** (const std::string &_name, double _length, bool _loop)

  Constructor.
- virtual ∼**Animation** ()

  Destructor.
- void **AddTime** (double _time)

  Add time to the animation.
- **KeyFrame** ∗ **GetKeyFrame** (unsigned int _index) const

  Get a key frame using an index value.
- unsigned int **GetKeyFrameCount** () const

  Return the number of key frames in the animation.
- double **GetLength** () const

  Return the duration of the animation.
- double **GetTime** () const

  Return the current time position.
- void **SetLength** (double _len)

  Set the duration of the animation.
- void **SetTime** (double _time)

  Set the current time position of the animation.

Protected Types

- typedef std::vector< **KeyFrame** * > **KeyFrame_V**

  array of keyframe type alias

Protected Member Functions

- double **GetKeyFramesAtTime** (double _time, **KeyFrame** **_kf1, **KeyFrame** **_kf2, unsigned int &_firstKeyIndex) const

  Get the two key frames that bound a time value.

Protected Attributes

- bool **build**

  determines if the interpolation splines need building
- **KeyFrame_V keyFrames**

  array of key frames
- double **length**

  animation duration
- bool **loop**

  true if animation repeats
- std::string **name**

  animation name
- double **timePos**

  current time position

10.3.1 Detailed Description

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

10.3.2 Member Typedef Documentation

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

array of keyframe type alias

10.3.3 Constructor & Destructor Documentation

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

Constructor.

Parameters

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:

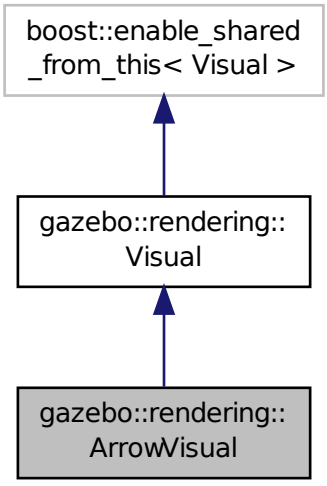
- **Animation.hh**

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.

Additional Inherited Members

10.4.1 Detailed Description

Basic arrow visualization.

10.4.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_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. 943).

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

Show the rotation of the visual.

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

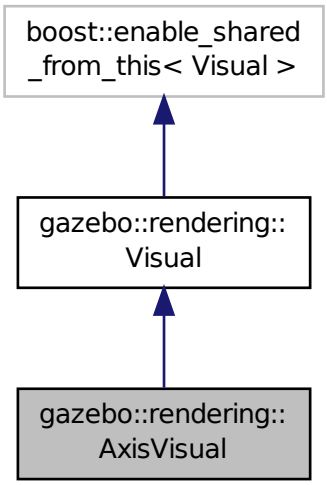
- **ArrowVisual.hh**

10.5 gazebo::rendering::AxisVisual Class Reference

Basic axis visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::AxisVisual:



Public Member Functions

- **AxisVisual** (const std::string &_name, **VisualPtr** _vis)
  Constructor.
- virtual ~**AxisVisual** ()
  Destructor.
- virtual void **Load** ()
  Load the axis visual.
- void **ScaleXAxis** (const **math::Vector3** &_scale)
  Scale the X axis.
- void **ScaleYAxis** (const **math::Vector3** &_scale)
  Scale the Y axis.
- void **ScaleZAxis** (const **math::Vector3** &_scale)
  Scale the Z axis.
- void **SetAxisMaterial** (unsigned int _axis, const std::string &_material)
  Set the material used to render and axis.
- void **ShowRotation** (unsigned int _axis)
  Load the rotation tube.

Additional Inherited Members

10.5.1 Detailed Description

Basic axis visualization.

10.5.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_name*	Name of the **AxisVisual** (p. 140)
in	*_vis*	Parent visual

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

Destructor.

10.5.3 Member Function Documentation

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

Load the axis visual.

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

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

Scale the X axis.

Parameters

in	*_scale*	Scaling factor

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

Scale the Y axis.

Parameters

in	*_scale*	Scaling factor

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

Scale the Z axis.

Parameters

in	*_scale*	Scaling factor

10.5.3.5 void gazebo::rendering::AxisVisual::SetAxisMaterial (unsigned int *_axis*, const std::string & *_material*)

Set the material used to render and axis.

Parameters

in	*_axis*	The number of the axis (0, 1, 2 = x,y,z)
in	*_material*	The name of the material to apply to the axis

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

Load the rotation tube.

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

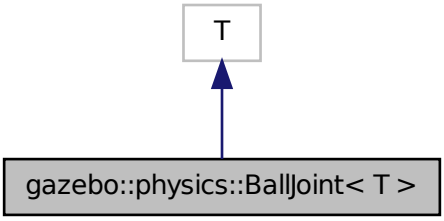
- **AxisVisual.hh**

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

Base (p. 145) class for a ball joint.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::BallJoint< T >:



Public Member Functions

- **BallJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**BallJoint** ()

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

Protected Member Functions

- void **Load** (**sdf::ElementPtr** _sdf)

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

10.6.1 Detailed Description

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

Base (p. 145) class for a ball joint.

Each physics engine should implement this class.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 template<class T> gazebo::physics::BallJoint< T >::BallJoint (BasePtr *_parent*) [inline]

Constructor.

Parameters

in	*_parent*	Pointer to the parent link.

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

Destructor.

10.6.3 Member Function Documentation

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

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

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

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

Parameters

in	*_sdf*	SDF to load the joint from.

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

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

10.6.3.6 template<class T> virtual void gazebo::physics::BallJoint< T >::SetLowStop (int , math::Angle)

[inline], [virtual]

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

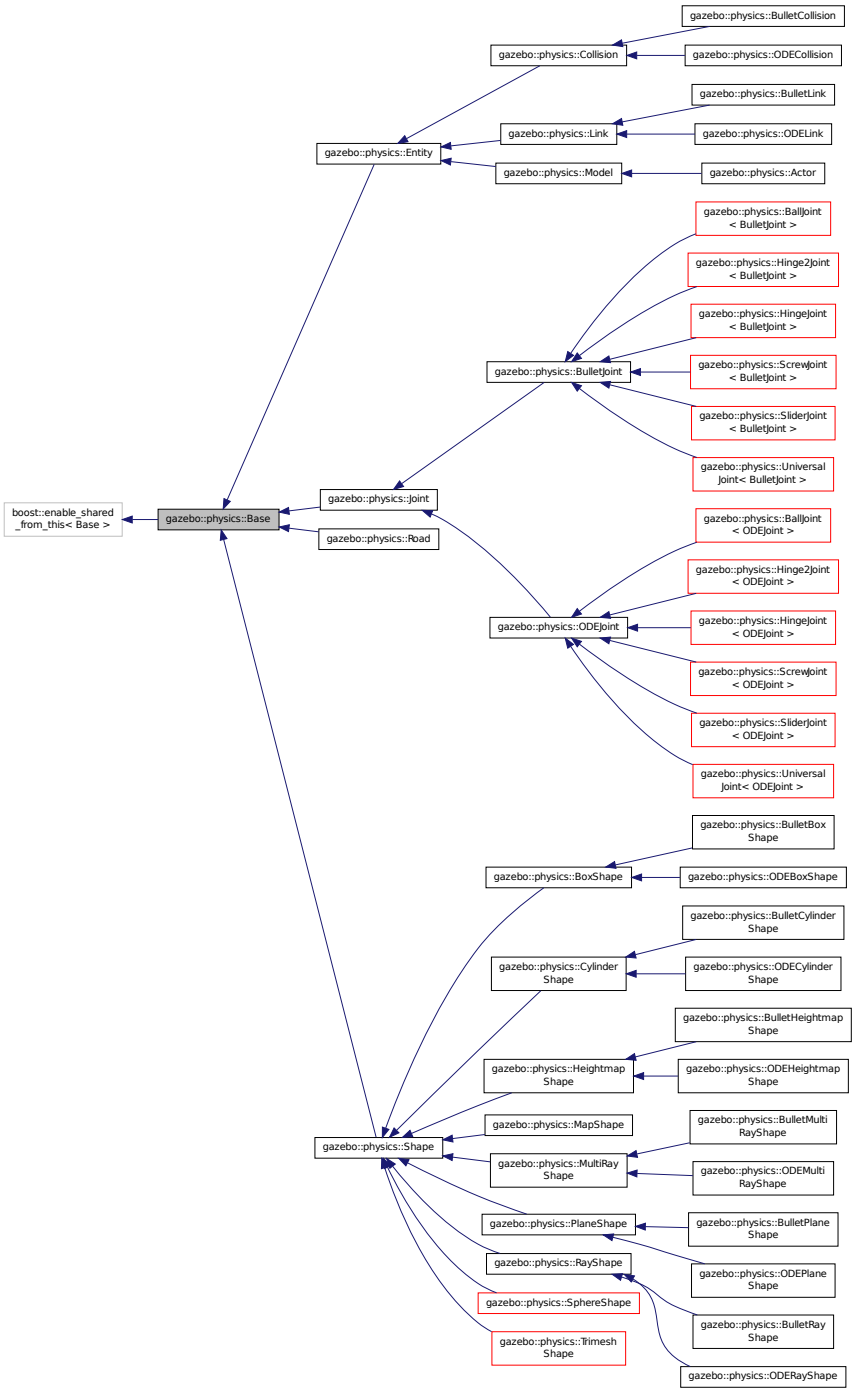
- **BallJoint.hh**

10.7 gazebo::physics::Base Class Reference

Base (p. 145) class for most physics classes.

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

Inheritance diagram for gazebo::physics::Base:



Public Types

- enum **EntityType** {
  BASE = 0x00000000, **ENTITY** = 0x00000001, **MODEL** = 0x00000002, **LINK** = 0x00000004,
  COLLISION = 0x00000008, **ACTOR** = 0x00000016, **LIGHT** = 0x00000010, **VISUAL** = 0x00000020,
  JOINT = 0x00000040, **BALL_JOINT** = 0x00000080, **HINGE2_JOINT** = 0x00000100, **HINGE_JOINT** = 0x00000200,
  SLIDER_JOINT = 0x00000400, **SCREW_JOINT** = 0x00000800, **UNIVERSAL_JOINT** = 0x00001000, **SHAPE** = 0x00002000,
  BOX_SHAPE = 0x00004000, **CYLINDER_SHAPE** = 0x00008000, **HEIGHTMAP_SHAPE** = 0x00010000, **MAP-_SHAPE** = 0x00020000,
  MULTIRAY_SHAPE = 0x00040000, **RAY_SHAPE** = 0x00080000, **PLANE_SHAPE** = 0x00100000, **SPHERE_-SHAPE** = 0x00200000,
  TRIMESH_SHAPE = 0x00400000 }

  Unique identifiers for all entity types.

Public Member Functions

- **Base** (**BasePtr parent**)

  Constructor.
- virtual ∼**Base** ()

  Destructor.
- void **AddChild** (**BasePtr** _child)

  Add a child to this entity.
- void **AddType** (**EntityType** _type)

  Add a type specifier.
- virtual void **Fini** ()

  Finialize the object.
- **BasePtr GetById** (unsigned int _id) const

  Get a child or self by id.
- **BasePtr GetByName** (const std::string &_name)

  Get by name.
- **BasePtr GetChild** (unsigned int _i) const

  Get a child by index.
- **BasePtr GetChild** (const std::string &_name)

  Get a child by name.
- unsigned int **GetChildCount** () const

  Get the number of children.
- unsigned int **GetId** () const

  Return the ID of this entity.
- std::string **GetName** () const

  Return the name of the entity.
- **BasePtr GetParent** () const

  Get the parent.
- int **GetParentId** () const

  Return the ID of the parent.
- bool **GetSaveable** () const

  Get whether the object should be "saved", when the user selects to save the world to xml.
- std::string **GetScopedName** () const

Return the name of this entity with the model scope world::model1::...::modelN::entityName.

- virtual const **sdf::ElementPtr GetSDF** ()

  Get the SDF values for the object.

- unsigned int **GetType** () const

  Get the full type definition.

- const **WorldPtr** & **GetWorld** () const

  *Get the **World** (p. 954) this object is in.*

- bool **HasType** (const **EntityType** &_t) const

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

- virtual void **Init** ()

  Initialize the object.

- bool **IsSelected** () const

  True if the entity is selected by the user.

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load.

- bool **operator==** (const **Base** &_ent) const

  Returns true if the entities are the same.

- void **Print** (const std::string &_prefix)

  Print this object to screen via gzmsg.

- virtual void **RemoveChild** (unsigned int _id)

  Remove a child from this entity.

- void **RemoveChild** (const std::string &_name)

  Remove a child by name.

- void **RemoveChildren** ()

  Remove all children.

- virtual void **Reset** ()

  Reset the object.

- virtual void **Reset** (**Base::EntityType** _resetType)

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

- virtual void **SetName** (const std::string &_name)

  Set the name of the entity.

- void **SetParent** (**BasePtr** _parent)

  Set the parent.

- void **SetSaveable** (bool _v)

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

- virtual bool **SetSelected** (bool _show)

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

- void **SetWorld** (const **WorldPtr** &_newWorld)

  Set the world this object belongs to.

- virtual void **Update** ()

  Update the object.

- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  Update the parameters using new sdf values.

Protected Attributes

- **Base_V children**

  Children of this entity.
- Base_V::iterator **childrenEnd**

  End of the children vector.
- **BasePtr parent**

  Parent of this entity.
- **sdf::ElementPtr sdf**

  The SDF values for this object.
- **WorldPtr world**

  Pointer to the world.

10.7.1 Detailed Description

Base (p. 145) class for most physics classes.

10.7.2 Member Enumeration Documentation

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

Unique identifiers for all entity types.

Enumerator:

BASE **Base** (p. 145) type.

ENTITY **Entity** (p. 338) type.

MODEL **Model** (p. 521) type.

LINK **Link** (p. 454) type.

COLLISION **Collision** (p. 262) type.

ACTOR **Actor** (p. 121) type.

LIGHT Light type.

VISUAL Visual type.

JOINT **Joint** (p. 423) type.

BALL_JOINT **BallJoint** (p. 143) type.

HINGE2_JOINT Hing2Joint type.

HINGE_JOINT **HingeJoint** (p. 405) type.

SLIDER_JOINT **SliderJoint** (p. 804) type.

SCREW_JOINT **ScrewJoint** (p. 760) type.

UNIVERSAL_JOINT **UniversalJoint** (p. 868) type.

SHAPE **Shape** (p. 779) type.

BOX_SHAPE **BoxShape** (p. 162) type.

CYLINDER_SHAPE **CylinderShape** (p. 307) type.

HEIGHTMAP_SHAPE **HeightmapShape** (p. 399) type.

MAP_SHAPE **MapShape** (p. 483) type.

MULTIRAY_SHAPE **MultiRayShape** (p. 549) type.

RAY_SHAPE **RayShape** (p. 718) type.

PLANE_SHAPE **PlaneShape** (p. 671) type.

SPHERE_SHAPE **SphereShape** (p. 806) type.

TRIMESH_SHAPE **TrimeshShape** (p. 866) type.

10.7.3 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*parent*	Parent of this object

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

Destructor.

10.7.4 Member Function Documentation

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

Add a child to this entity.

Parameters

in	*_child*	Child entity.

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

Add a type specifier.

Parameters

in	*_type*	New type to append to this objects type definition.

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

Finialize the object.

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

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

Get a child or self by id.

Parameters

in	*_id*	ID of the object to retreive.

Returns

A pointer to the object, NULL if not found

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

Get by name.

Parameters

in	*_name*	Get a child (or self) object by name

Returns

A pointer to the object, NULL if not found

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

Get a child by index.

Parameters

in	*_i*	Index of the child to retreive.

Returns

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

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

Get a child by name.

Parameters

in	*_name*	Name of the child.

Returns

A pointer to the object, NULL if not found

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

Get the number of children.

Returns

The number of children.

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

Return the ID of this entity.

This id is unique.

Returns

Integer ID.

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

Return the name of the entity.

Returns

Name of the entity.

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

Get the parent.

Returns

Pointer to the parent entity.

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

Return the ID of the parent.

Returns

Integer ID.

10.7.4.13 bool gazebo::physics::Base::GetSaveable ( ) const

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

Returns

True if the object is saveable.

10.7.4.14 std::string gazebo::physics::Base::GetScopedName ( ) const

Return the name of this entity with the model scope world::model1::...::modelN::entityName.

Returns

The scoped name.

10.7.4.15 virtual const sdf::ElementPtr gazebo::physics::Base::GetSDF ( ) [virtual]

Get the SDF values for the object.

Returns

The SDF values for the object.

Reimplemented in **gazebo::physics::Actor** (p. 124), and **gazebo::physics::Model** (p. 528).

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

Get the full type definition.

Returns

The full type definition.

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

Get the **World** (p. 954) this object is in.

Returns

The **World** (p. 954) this object is part of.

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

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

Parameters

in	_t	Type to check.

Returns

True if this object's type definition has the.

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

Initialize the object.

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

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

True if the entity is selected by the user.

Returns

True if the entity is selected.

10.7.4.21 virtual void gazebo::physics::Base::Load (sdf::ElementPtr *_sdf*) [virtual]

Load.

Parameters

in	*node*	Pointer to an SDF parameters

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

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

Returns true if the entities are the same.

Checks only the name.

Parameters

in	*_ent*	**Base** (p. 145) object to compare with.

Returns

True if the entities are the same.

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

Print this object to screen via gzmsg.

Parameters

in	*_prefix*	Usually a set of spaces.

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

Remove a child from this entity.

Parameters

in	*_id*	ID of the child to remove.

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

Remove a child by name.

Parameters

in	*_name*	Name of the child.

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

Remove all children.

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

Reset the object.

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

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

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

Parameters

in	*_resetType*	The type of reset operation

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

Set the name of the entity.

Parameters

in	*_name*	New name.

Reimplemented in **gazebo::physics::Entity** (p. 347).

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

Set the parent.

Parameters

in	*_parent*	Parent object.

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

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

Parameters

in	*_v*	Set to True if the object should be saved.

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

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

Parameters

in	*_show*	True to set this entity as selected.

Reimplemented in **gazebo::physics::Link** (p. 471).

10.7.4.33 void gazebo::physics::Base::SetWorld (const WorldPtr & *_newWorld*)

Set the world this object belongs to.

This will also set the world for all children.

Parameters

in	*_newWorld*	The new **World** (p. 954) this object is part of.

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

Update the object.

Reimplemented in **gazebo::physics::MultiRayShape** (p. 555), **gazebo::physics::Joint** (p. 437), **gazebo::physics::Actor** (p. 125), **gazebo::physics::RayShape** (p. 722), **gazebo::physics::Link** (p. 472), **gazebo::physics::Model** (p. 533), **gazebo::physics::BulletLink** (p. 197), **gazebo::physics::ODELink** (p. 606), **gazebo::physics::MapShape** (p. 487), **gazebo::physics::ODERayShape** (p. 621), **gazebo::physics::BulletRayShape** (p. 212), **gazebo::physics::TrimeshShape** (p. 868), and **gazebo::physics::ODETrimeshShape** (p. 634).

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

Update the parameters using new sdf values.

Parameters

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

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

10.7.5 Member Data Documentation

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

Children of this entity.

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

End of the children vector.

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

Parent of this entity.

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

The SDF values for this object.

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

Pointer to the world.

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

- **Base.hh**

10.8 gazebo::math::Box Class Reference

Mathematical representation of a box and related functions.

`#include <Box.hh>`

Public Member Functions

- **Box** ()

  Default constructor.
- **Box** (const **Vector3** &_min, const **Vector3** &_max)

  Constructor.
- **Box** (const **Box** &_b)

  Copy Constructor.
- virtual ~**Box** ()

  Destructor.
- **math::Vector3 GetCenter** () const

  Get the box center.
- **math::Vector3 GetSize** () const

  Get the size of the box.
- double **GetXLength** () const

  Get the length along the x dimension.
- double **GetYLength** () const

  Get the length along the y dimension.
- double **GetZLength** () const

  Get the length along the z dimension.
- void **Merge** (const **Box** &_box)

  Merge a box with this box.
- **Box operator+** (const **Box** &_b) const

  Addition operator.
- const **Box** & **operator+=** (const **Box** &_b)

  Addition set operator.
- **Box operator-** (const **Vector3** &_v)

  Subtract a vector from the min and max values.
- **Box** & **operator=** (const **Box** &_b)

  Assignment operator.
- bool **operator==** (const **Box** &_b)

  Equality test 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.

10.8.1 Detailed Description

Mathematical representation of a box and related functions.

10.8.2 Constructor & Destructor Documentation

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

Default constructor.

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

Constructor.

Parameters

in	*_min*	Minimum corner of the box
in	*_max*	Maximum corner of the box

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

Copy Constructor.

Parameters

in	*_b*	**Box** (p. 158) to copy

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

Destructor.

10.8.3 Member Function Documentation

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

Get the box center.

Returns

The center position of the box

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

Get the size of the box.

Returns

Size of the box

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

Get the length along the x dimension.

Returns

Double value of the length in the x dimension

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

Get the length along the y dimension.

Returns

Double value of the length in the y dimension

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

Get the length along the z dimension.

Returns

Double value of the length in the z dimension

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

Merge a box with this box.

Parameters

in	*_box*	**Box** (p. 158) to add to this box

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

Addition operator.

result = this + _b

Parameters

in	*_b*	**Box** (p. 158) to add

Returns

The new box

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

Addition set operator.

this = this + _b

Parameters

in	*_b*	**Box** (p. 158) to add

Returns

This new box

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

Subtract a vector from the min and max values.

Parameters

_v	The vector to use during subtraction

Returns

The new box

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

Assignment operator.

Set this box to the parameter

Parameters

in	*_b*	**Box** (p. 158) to copy

Returns

The new box.

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

Equality test operatoer.

Parameters

in	*_b*	**Box** (p. 158) to test

Returns

True if equal

10.8.4 Friends And Related Function Documentation

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

Output operator.

Parameters

in	*_out*	Output stream
in	*_b*	**Box** (p. 158) to output to the stream

Returns

The stream

10.8.5 Member Data Documentation

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

Maximum corner of the box.

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

Minimum corner of the box.

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

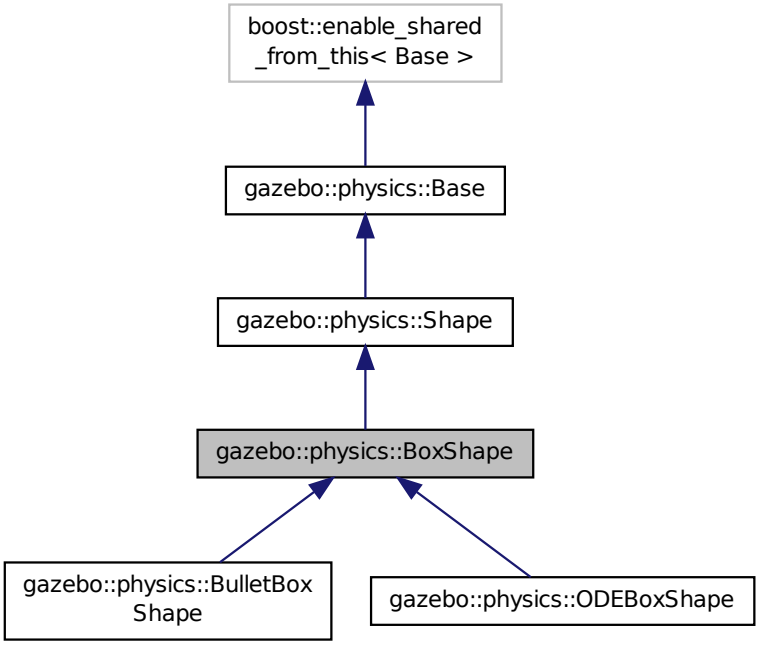
- **Box.hh**

10.9 gazebo::physics::BoxShape Class Reference

Box geometry primitive.

`#include <physics/physcs.hh>`

Inheritance diagram for gazebo::physics::BoxShape:



Public Member Functions

- **BoxShape** (**CollisionPtr** _parent)
  Constructor.
- virtual ~**BoxShape** ()
  Destructor.
- void **FillMsg** (msgs::Geometry &_msg)
  Fill in the values for a geomertry message.
- void **FillShapeMsg** (msgs::Geometry &_msg) **GAZEBO_DEPRECATED**
  Deprecated.
- virtual void **GetInertial** (double _mass, **InertialPtr** _inertial) const
  Get inertial for a shape.
- virtual double **GetMass** (double _density) const
  Get the mass of the box give a density value.
- **math::Vector3 GetSize** () const
  Get the size of the box.
- virtual void **Init** ()
  Initialize the box.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
  Process a geometry message.
- virtual void **SetSize** (const **math::Vector3** &_size)
  Set the size of the box.

Additional Inherited Members

10.9.1 Detailed Description

Box geometry primitive.

10.9.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

in	*_parent*	Parent **Collision** (p. 262).

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

Destructor.

10.9.3 Member Function Documentation

10.9.3.1 void gazebo::physics::BoxShape::FillMsg (msgs::Geometry & _msg) [virtual]

Fill in the values for a geomertry message.

Parameters

out	*_msg*	The geometry message to fill.

Implements **gazebo::physics::Shape** (p. 781).

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

Deprecated.

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

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

Get inertial for a shape.

Parameters

in	*_mass*	Mass of the box.
out	*_inertial*	**Inertial** (p. 414) element to populate with the result.

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

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

Get the mass of the box give a density value.

Parameters

in	*_density*	Density to compute a mass from.

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

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

Get the size of the box.

Returns

The size of each side of the box.

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

Initialize the box.

Implements **gazebo::physics::Shape** (p. 781).

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

Process a geometry message.

Parameters

in	*_msg*	The message to set values from.

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

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

Set the size of the box.

Parameters

in	*_size*	Size of each side of the box.

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

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

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

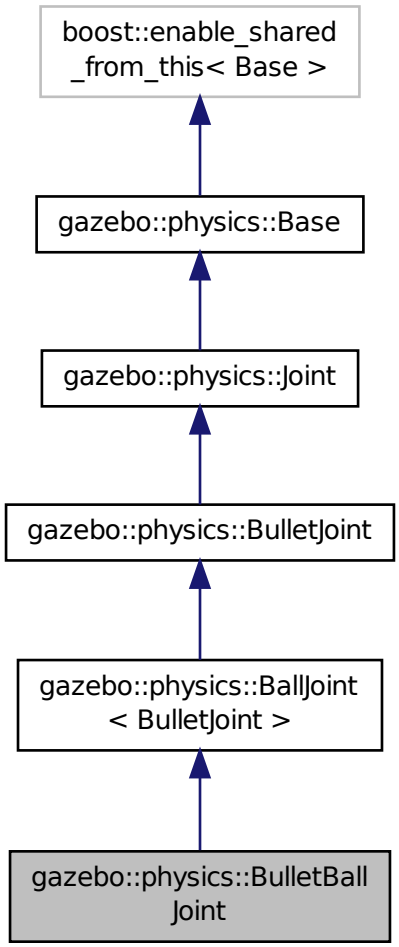
- **BoxShape.hh**

10.10 gazebo::physics::BulletBallJoint Class Reference

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

`#include <BulletBallJoint.hh>`

Inheritance diagram for gazebo::physics::BulletBallJoint:



Public Member Functions

- **BulletBallJoint** (btDynamicsWorld ∗_world, **BasePtr** _parent)

  *Bullet Ball **Joint** (p. 423) Constructor.*
- virtual ∼**BulletBallJoint** ()

  Destructor.
- void **Attach** (**LinkPtr** _one, **LinkPtr** _two)

Attach the two bodies with this joint.

- **math::Vector3 GetAnchor** (int _index) const

  Get joint's anchor point.

- virtual **math::Angle GetAngle** (int _index) const

  Get the angle of rotation of an axis(index)

- virtual **math::Angle GetAngleImpl** (int _index) const

  Get the angle of rotation.

- virtual **math::Vector3 GetAxis** (int) const

  Get the axis of rotation.

- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  Get the axis of rotation.

- virtual double **GetMaxForce** (int _index)

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

- virtual double **GetVelocity** (int _index) const

  Get the rotation rate of an axis(index)

- void **SetAnchor** (int _index, const **math::Vector3** &_anchor)

  Set joint's anchor point.

- virtual void **SetDamping** (int _index, double _damping)

  Set joint damping, not yet implemented.

- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).

- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).

- virtual void **SetMaxForce** (int _index, double _t)

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

- virtual void **SetVelocity** (int _index, double _angle)

  Set the velocity of an axis(index).

Additional Inherited Members

10.10.1 Detailed Description

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

10.10.2 Constructor & Destructor Documentation

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

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

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

Destructor.

10.10.3 Member Function Documentation

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

Attach the two bodies with this joint.

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

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

Get joint's anchor point.

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

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

Get the angle of rotation of an axis(index)

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

Get the angle of rotation.

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

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

Get the axis of rotation.

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

Get the axis of rotation.

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

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

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

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

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

Get the rotation rate of an axis(index)

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

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

Set joint's anchor point.

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

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

Set joint damping, not yet implemented.

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

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

Set the high stop of an axis(index).

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

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

Set the low stop of an axis(index).

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

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

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

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

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

Set the velocity of an axis(index).

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

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

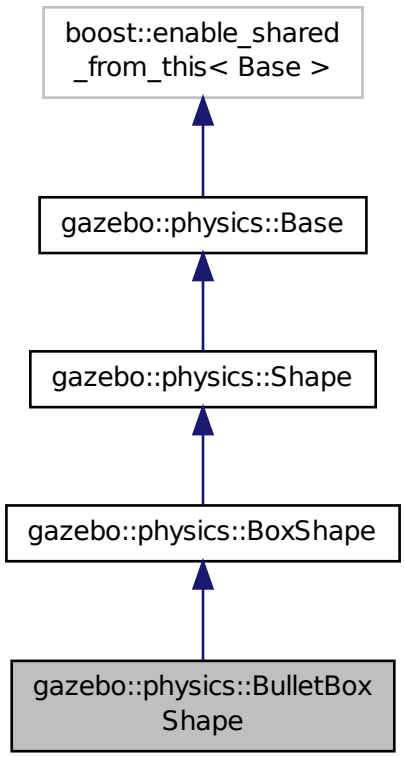
- **BulletBallJoint.hh**

10.11 gazebo::physics::BulletBoxShape Class Reference

Bullet box collision.

`#include <BulletBoxShape.hh>`

Inheritance diagram for gazebo::physics::BulletBoxShape:



Public Member Functions

- **BulletBoxShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**BulletBoxShape** ()

  Destructor.
- void **SetSize** (const **math::Vector3** &_size)

  Set the size of the box.

Additional Inherited Members

10.11.1 Detailed Description

Bullet box collision.

10.11.2 Constructor & Destructor Documentation

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

Constructor.

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

Destructor.

10.11.3 Member Function Documentation

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

Set the size of the box.

Bullet requires the half-extents of the box

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

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

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

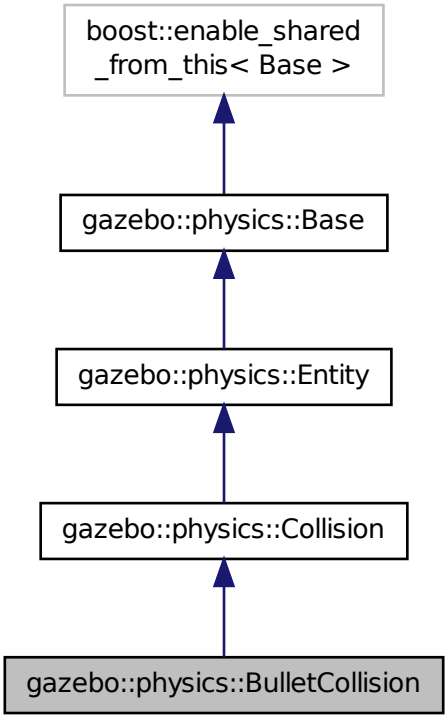
- **BulletBoxShape.hh**

10.12 gazebo::physics::BulletCollision Class Reference

Bullet collisions.

`#include <BulletCollision.hh>`

Inheritance diagram for gazebo::physics::BulletCollision:



Public Member Functions

- **BulletCollision** (**LinkPtr** _parent)

  Constructor.
- virtual ~**BulletCollision** ()

  Destructor.
- virtual **math::Box GetBoundingBox** () const

  Get the bounding box, defined by the physics engine.
- btCollisionShape * **GetCollisionShape** () const

  Get the bullet collision shape.
- virtual void **Load** (**sdf::ElementPtr** _ptr)

  Load the collision.
- virtual void **OnPoseChange** ()

  On pose change.
- virtual void **SetCategoryBits** (unsigned int bits)

  Set the category bits, used during collision detection.
- virtual void **SetCollideBits** (unsigned int bits)

  Set the collide bits, used during collision detection.

- void **SetCollisionShape** (btCollisionShape ∗**shape**)

  Set the collision shape.
- void **SetCompoundShapeIndex** (int index)

  Set the index of the compound shape.

Protected Attributes

- btCollisionShape ∗ **collisionShape**

Additional Inherited Members

10.12.1 Detailed Description

Bullet collisions.

10.12.2 Constructor & Destructor Documentation

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

Constructor.

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

Destructor.

10.12.3 Member Function Documentation

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

Get the bounding box, defined by the physics engine.

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

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

Get the bullet collision shape.

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

Load the collision.

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

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

On pose change.

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

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

Set the category bits, used during collision detection.

Parameters

bits	The bits

Implements **gazebo::physics::Collision** (p. 270).

10.12.3.6 virtual void gazebo::physics::BulletCollision::SetCollideBits (unsigned int *bits*) [virtual]

Set the collide bits, used during collision detection.

Parameters

bits	The bits

Implements **gazebo::physics::Collision** (p. 270).

10.12.3.7 void gazebo::physics::BulletCollision::SetCollisionShape (btCollisionShape * *shape*)

Set the collision shape.

Referenced by gazebo::physics::BulletPlaneShape::CreatePlane(), gazebo::physics::BulletSphereShape::SetRadius(), gazebo::physics::BulletBoxShape::SetSize(), and gazebo::physics::BulletCylinderShape::SetSize().

10.12.3.8 void gazebo::physics::BulletCollision::SetCompoundShapeIndex (int *index*)

Set the index of the compound shape.

10.12.4 Member Data Documentation

10.12.4.1 btCollisionShape* gazebo::physics::BulletCollision::collisionShape [protected]

The documentation for this class was generated from the following file:

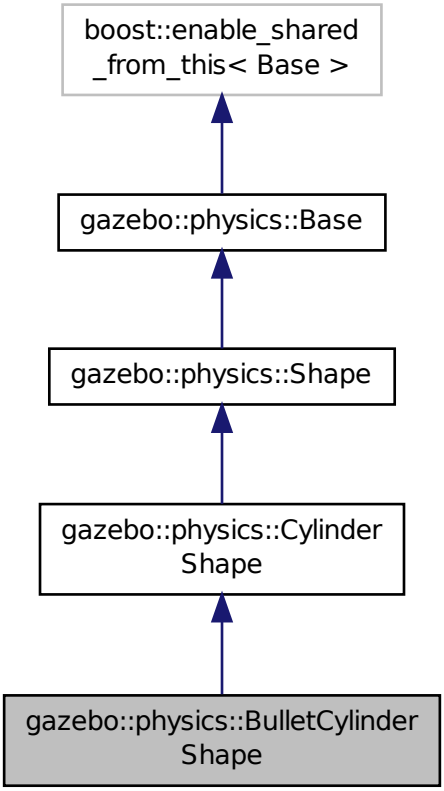
- **BulletCollision.hh**

10.13 gazebo::physics::BulletCylinderShape Class Reference

Cylinder collision.

`#include <BulletCylinderShape.hh>`

Inheritance diagram for gazebo::physics::BulletCylinderShape:



Public Member Functions

- **BulletCylinderShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**BulletCylinderShape** ()

  Destructor.
- void **SetSize** (const double &_radius, const double &_length)

  Set the size of the cylinder.

Additional Inherited Members

10.13.1 Detailed Description

Cylinder collision.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 gazebo::physics::BulletCylinderShape::BulletCylinderShape (CollisionPtr *_parent*) [inline]

Constructor.

10.13.2.2 virtual gazebo::physics::BulletCylinderShape::~BulletCylinderShape ( ) [inline], [virtual]

Destructor.

10.13.3 Member Function Documentation

10.13.3.1 void gazebo::physics::BulletCylinderShape::SetSize (const double & *_radius,* const double & *_length*) [inline]

Set the size of the cylinder.

References gazebo::physics::Shape::collisionParent, gazebo::physics::BulletCollision::SetCollisionShape(), and gazebo::physics::CylinderShape::SetSize().

The documentation for this class was generated from the following file:

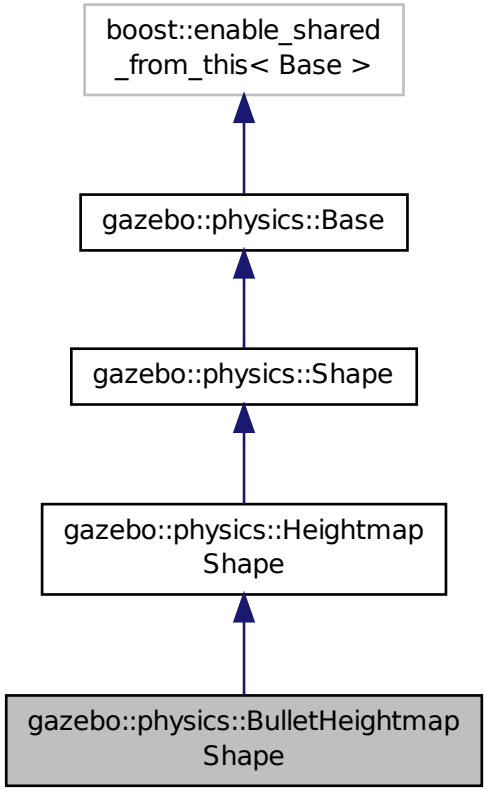
- **BulletCylinderShape.hh**

10.14 gazebo::physics::BulletHeightmapShape Class Reference

Height map collision.

`#include <BulletHeightmapShape.hh>`

Inheritance diagram for gazebo::physics::BulletHeightmapShape:



Public Member Functions

- **BulletHeightmapShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**BulletHeightmapShape** ()

  Destructor.
- virtual void **Init** ()

  Load the heightmap.

Additional Inherited Members

10.14.1 Detailed Description

Height map collision.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 gazebo::physics::BulletHeightmapShape::BulletHeightmapShape (CollisionPtr *_parent*)

Constructor.

10.14.2.2 virtual gazebo::physics::BulletHeightmapShape::~BulletHeightmapShape ( ) [virtual]

Destructor.

10.14.3 Member Function Documentation

10.14.3.1 virtual void gazebo::physics::BulletHeightmapShape::Init ( ) [virtual]

Load the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 403).

The documentation for this class was generated from the following file:

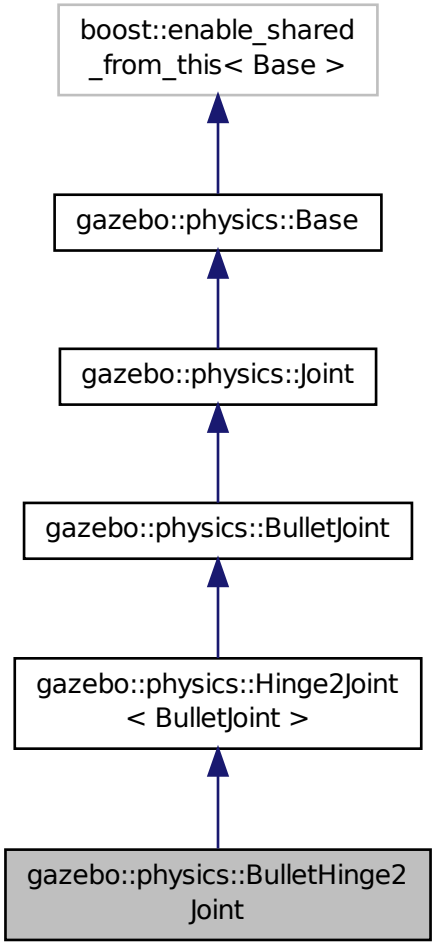
- **BulletHeightmapShape.hh**

10.15 gazebo::physics::BulletHinge2Joint Class Reference

A two axis hinge joint.

`#include <BulletHinge2Joint.hh>`

Inheritance diagram for gazebo::physics::BulletHinge2Joint:



Public Member Functions

- **BulletHinge2Joint** (btDynamicsWorld ∗**world**, **BasePtr** _parent)

  Constructor.
- virtual ∼**BulletHinge2Joint** ()

  Destructor.
- virtual void **Attach** (**LinkPtr** _one, **LinkPtr** _two)

  Attach the two bodies with this joint.
- virtual **math::Vector3 GetAnchor** (int _index) const

  Get anchor point.
- **math::Angle GetAngle** (int _index) const

  Get angle of rotation about first axis.

- virtual **math::Angle GetAngleImpl** (int _index) const

  Get the angle of rotation.
- virtual **math::Vector3 GetAxis** (int _index) const

  Get first axis of rotation.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  Get the axis of rotation.
- virtual **math::Angle GetHighStop** (int _index)

  Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)

  Get the max allowed force of an axis(index).
- double **GetVelocity** (int _index) const

  Get rate of rotation about first axis.
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)

  Set the anchor point.
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)

  Set the first axis of rotation.
- virtual void **SetDamping** (int _index, double _damping)

  Set joint damping, not yet implemented.
- void **SetForce** (int _index, double _torque)

  Set the torque.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)

  Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load the **BulletHinge2Joint*** (p. *178*).

Additional Inherited Members

10.15.1 Detailed Description

A two axis hinge joint.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 gazebo::physics::BulletHinge2Joint::BulletHinge2Joint (btDynamicsWorld * *world,* BasePtr *_parent*)

Constructor.

10.15.2.2 virtual gazebo::physics::BulletHinge2Joint::~BulletHinge2Joint ( ) [virtual]

Destructor.

10.15.3 Member Function Documentation

10.15.3.1 virtual void gazebo::physics::BulletHinge2Joint::Attach (LinkPtr *_one*, LinkPtr *_two*) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.15.3.2 virtual math::Vector3 gazebo::physics::BulletHinge2Joint::GetAnchor (int *_index*) const [virtual]

Get anchor point.

Reimplemented from **gazebo::physics::BulletJoint** (p. 190).

10.15.3.3 math::Angle gazebo::physics::BulletHinge2Joint::GetAngle (int *_index*) const

Get angle of rotation about first axis.

10.15.3.4 virtual math::Angle gazebo::physics::BulletHinge2Joint::GetAngleImpl (int *_index*) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.15.3.5 virtual math::Vector3 gazebo::physics::BulletHinge2Joint::GetAxis (int *_index*) const [virtual]

Get first axis of rotation.

10.15.3.6 virtual math::Vector3 gazebo::physics::BulletHinge2Joint::GetGlobalAxis (int *_index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.15.3.7 virtual math::Angle gazebo::physics::BulletHinge2Joint::GetHighStop (int *_index*) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

10.15.3.8 virtual math::Angle gazebo::physics::BulletHinge2Joint::GetLowStop (int *_index*) [virtual]

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.15.3.9 virtual double gazebo::physics::BulletHinge2Joint::GetMaxForce (int *_index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.15.3.10 double gazebo::physics::BulletHinge2Joint::GetVelocity (int *_index*) const [virtual]

Get rate of rotation about first axis.

Implements **gazebo::physics::Joint** (p. 433).

10.15.3.11 virtual void gazebo::physics::BulletHinge2Joint::Load (sdf::ElementPtr *_sdf*) [protected], [virtual]

Load the **BulletHinge2Joint** (p. 178).

Reimplemented from **gazebo::physics::Hinge2Joint**< **BulletJoint** > (p. 405).

10.15.3.12 virtual void gazebo::physics::BulletHinge2Joint::SetAnchor (int *_index*, const math::Vector3 & *_anchor*) [virtual]

Set the anchor point.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.15.3.13 virtual void gazebo::physics::BulletHinge2Joint::SetAxis (int *_index*, const math::Vector3 & *_axis*) [virtual]

Set the first axis of rotation.

Implements **gazebo::physics::Joint** (p. 435).

10.15.3.14 virtual void gazebo::physics::BulletHinge2Joint::SetDamping (int *_index*, double *_damping*) [virtual]

Set joint damping, not yet implemented.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.15.3.15 void gazebo::physics::BulletHinge2Joint::SetForce (int *_index*, double *_torque*) [virtual]

Set the torque.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.15.3.16 virtual void gazebo::physics::BulletHinge2Joint::SetHighStop (int *_index*, const math::Angle & *_angle*) [virtual]

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.15.3.17 virtual void gazebo::physics::BulletHinge2Joint::SetLowStop (int *_index*, const math::Angle & *_angle*) [virtual]

Set the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.15.3.18 virtual void gazebo::physics::BulletHinge2Joint::SetMaxForce (int *_index*, double *_t*) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.15.3.19 virtual void gazebo::physics::BulletHinge2Joint::SetVelocity (int *_index*, double *_angle*) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

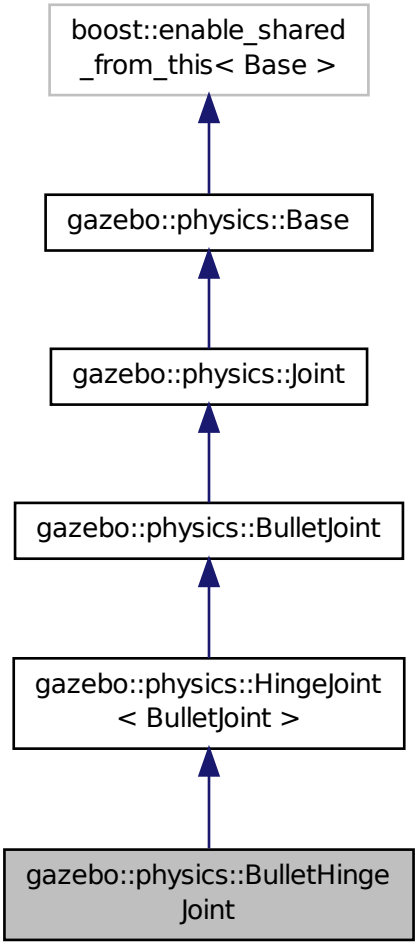
- **BulletHinge2Joint.hh**

10.16 gazebo::physics::BulletHingeJoint Class Reference

A single axis hinge joint.

`#include <BulletHingeJoint.hh>`

Inheritance diagram for gazebo::physics::BulletHingeJoint:



Public Member Functions

- **BulletHingeJoint** (btDynamicsWorld ∗**world**, **BasePtr** _parent)

  Constructor.
- virtual ∼**BulletHingeJoint** ()

  Destructor.
- virtual void **Attach** (**LinkPtr** _one, **LinkPtr** _two)

  Attach the two bodies with this joint.
- virtual **math::Vector3 GetAnchor** (int _index) const

  Get the anchor point.
- virtual **math::Angle GetAngle** (int _index) const

  Get the angle of rotation.

- virtual **math::Angle GetAngleImpl** (int _index) const

  Get the angle of rotation.
- virtual double **GetForce** (int _index)

  Get the torque of a joint.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  Get the axis of rotation.
- virtual **math::Angle GetHighStop** (int _index)

  Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)

  Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const

  Get the rotation rate.
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)

  Set the anchor point.
- void **SetAxis** (int _index, const **math::Vector3** &_axis)

  Set the axis of rotation.
- virtual void **SetDamping** (int _index, double _damping)

  Set joint damping, not yet implemented.
- void **SetForce** (int _index, double _torque)

  Set the torque of a joint.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)

  Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load the **BulletHingeJoint** (*p. *183)**.*

Additional Inherited Members

10.16.1 Detailed Description

A single axis hinge joint.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 gazebo::physics::BulletHingeJoint::BulletHingeJoint (btDynamicsWorld * *world,* BasePtr *_parent*)

Constructor.

10.16.2.2 virtual gazebo::physics::BulletHingeJoint::~BulletHingeJoint ( ) [virtual]

Destructor.

10.16.3 Member Function Documentation

10.16.3.1 virtual void gazebo::physics::BulletHingeJoint::Attach (LinkPtr *_one,* LinkPtr *_two*) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.16.3.2 virtual math::Vector3 gazebo::physics::BulletHingeJoint::GetAnchor (int *_index*) const [virtual]

Get the anchor point.

Reimplemented from **gazebo::physics::BulletJoint** (p. 190).

10.16.3.3 virtual math::Angle gazebo::physics::BulletHingeJoint::GetAngle (int *_index*) const [virtual]

Get the angle of rotation.

10.16.3.4 virtual math::Angle gazebo::physics::BulletHingeJoint::GetAngleImpl (int *_index*) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.16.3.5 virtual double gazebo::physics::BulletHingeJoint::GetForce (int *_index*) [virtual]

Get the torque of a joint.

Reimplemented from **gazebo::physics::Joint** (p. 430).

10.16.3.6 virtual math::Vector3 gazebo::physics::BulletHingeJoint::GetGlobalAxis (int *_index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.16.3.7 virtual math::Angle gazebo::physics::BulletHingeJoint::GetHighStop (int *_index*) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

10.16.3.8 virtual math::Angle gazebo::physics::BulletHingeJoint::GetLowStop (int *_index*) [virtual]

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.16.3.9 virtual double gazebo::physics::BulletHingeJoint::GetMaxForce (int _index) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.16.3.10 virtual double gazebo::physics::BulletHingeJoint::GetVelocity (int _index) const [virtual]

Get the rotation rate.

Implements **gazebo::physics::Joint** (p. 433).

10.16.3.11 virtual void gazebo::physics::BulletHingeJoint::Load (sdf::ElementPtr _sdf) [protected], [virtual]

Load the **BulletHingeJoint** (p. 183).

Reimplemented from **gazebo::physics::HingeJoint< BulletJoint >** (p. 407).

10.16.3.12 virtual void gazebo::physics::BulletHingeJoint::SetAnchor (int _index, const math::Vector3 & _anchor) [virtual]

Set the anchor point.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.16.3.13 void gazebo::physics::BulletHingeJoint::SetAxis (int _index, const math::Vector3 & _axis) [virtual]

Set the axis of rotation.

Implements **gazebo::physics::Joint** (p. 435).

10.16.3.14 virtual void gazebo::physics::BulletHingeJoint::SetDamping (int _index, double _damping) [virtual]

Set joint damping, not yet implemented.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.16.3.15 void gazebo::physics::BulletHingeJoint::SetForce (int _index, double _torque) [virtual]

Set the torque of a joint.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.16.3.16 virtual void gazebo::physics::BulletHingeJoint::SetHighStop (int _index, const math::Angle & _angle) [virtual]

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.16.3.17 virtual void gazebo::physics::BulletHingeJoint::SetLowStop (int *_index*, const math::Angle & *_angle*) [virtual]

Set the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.16.3.18 virtual void gazebo::physics::BulletHingeJoint::SetMaxForce (int *_index*, double *_t*) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.16.3.19 virtual void gazebo::physics::BulletHingeJoint::SetVelocity (int *_index*, double *_angle*) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

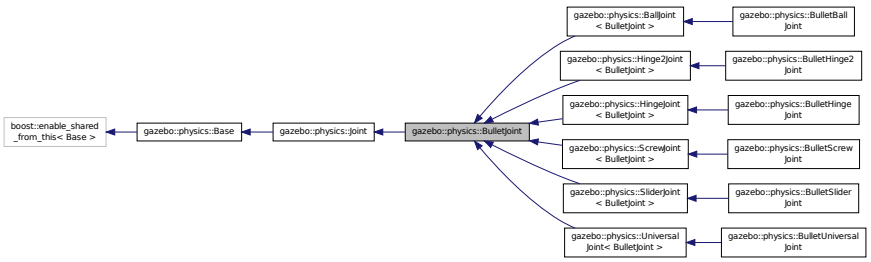
- **BulletHingeJoint.hh**

10.17 gazebo::physics::BulletJoint Class Reference

Base (p. 145) class for all joints.

`#include <BulletJoint.hh>`

Inheritance diagram for gazebo::physics::BulletJoint:



Public Member Functions

- **BulletJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**BulletJoint** ()

  Destructor.
- bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const

  Determines of the two bodies are connected by a joint.
- virtual void **Detach** ()

Detach this joint from all bodies.

- virtual **math::Vector3 GetAnchor** (int) const

  Get the anchor point.
- **LinkPtr GetJointLink** (int _index) const

  Get the body to which the joint is attached according the _index.
- virtual **math::Vector3 GetLinkForce** (unsigned int) const

  Get the force the joint applies to the first body.
- virtual **math::Vector3 GetLinkTorque** (unsigned int) const

  Get the torque the joint applies to the first body.
- void **Load** (**sdf::ElementPtr** _sdf)

  *Load a **BulletJoint** (p. 188).*
- virtual void **Reset** ()

  Reset the joint.
- virtual void **SetAnchor** (int, const **gazebo::math::Vector3** &)

  Set the anchor point.
- virtual void **SetAttribute** (**Attribute**, int, double)

  Set a parameter for the joint.
- virtual void **SetDamping** (int, const double)

  Set the joint damping.

Protected Attributes

- btTypedConstraint ∗ **constraint**
- btDynamicsWorld ∗ **world**

Additional Inherited Members

10.17.1 Detailed Description

Base (p. 145) class for all joints.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 gazebo::physics::BulletJoint::BulletJoint (BasePtr *_parent*)

Constructor.

10.17.2.2 virtual gazebo::physics::BulletJoint::∼BulletJoint ( ) [virtual]

Destructor.

10.17.3 Member Function Documentation

10.17.3.1 bool gazebo::physics::BulletJoint::AreConnected (LinkPtr *_one,* LinkPtr *_two*) const [virtual]

Determines of the two bodies are connected by a joint.

Implements **gazebo::physics::Joint** (p. 427).

10.17.3.2 virtual void gazebo::physics::BulletJoint::Detach ( ) [virtual]

Detach this joint from all bodies.

Reimplemented from **gazebo::physics::Joint** (p. 428).

10.17.3.3 virtual math::Vector3 gazebo::physics::BulletJoint::GetAnchor (int) const [inline], [virtual]

Get the anchor point.

Implements **gazebo::physics::Joint** (p. 428).

Reimplemented in **gazebo::physics::ScrewJoint< BulletJoint >** (p. 761), **gazebo::physics::BulletHinge2Joint** (p. 181), **gazebo::physics::SliderJoint< BulletJoint >** (p. 805), **gazebo::physics::BulletHingeJoint** (p. 186), **gazebo::physics::BulletUniversalJoint** (p. 227), and **gazebo::physics::BulletBallJoint** (p. 168).

References gzerr.

10.17.3.4 LinkPtr gazebo::physics::BulletJoint::GetJointLink (int *_index*) const [virtual]

Get the body to which the joint is attached according the _index.

Implements **gazebo::physics::Joint** (p. 431).

10.17.3.5 virtual math::Vector3 gazebo::physics::BulletJoint::GetLinkForce (unsigned int) const [inline], [virtual]

Get the force the joint applies to the first body.

Parameters

index	The index of the body(0 or 1)

Implements **gazebo::physics::Joint** (p. 431).

References gzerr.

10.17.3.6 virtual math::Vector3 gazebo::physics::BulletJoint::GetLinkTorque (unsigned int) const [inline], [virtual]

Get the torque the joint applies to the first body.

Parameters

index	The index of the body(0 or 1)

Implements **gazebo::physics::Joint** (p. 431).

References gzerr.

10.17.3.7 void gazebo::physics::BulletJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load a **BulletJoint** (p. 188).

Reimplemented from **gazebo::physics::Joint** (p. 433).

Reimplemented in **gazebo::physics::BallJoint< BulletJoint >** (p. 144), **gazebo::physics::BulletScrewJoint** (p. 216), **gazebo::physics::ScrewJoint< BulletJoint >** (p. 761), **gazebo::physics::BulletSliderJoint** (p. 220), **gazebo::physics::SliderJoint< BulletJoint >** (p. 805), **gazebo::physics::Hinge2Joint< BulletJoint >** (p. 405), **gazebo::physics::HingeJoint< BulletJoint >** (p. 407), and **gazebo::physics::UniversalJoint< BulletJoint >** (p. 870).

10.17.3.8 virtual void gazebo::physics::BulletJoint::Reset ( ) [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Joint** (p. 434).

10.17.3.9 virtual void gazebo::physics::BulletJoint::SetAnchor (int , const gazebo::math::Vector3 &) [inline], [virtual]

Set the anchor point.

Implements **gazebo::physics::Joint** (p. 434).

Reimplemented in **gazebo::physics::ScrewJoint< BulletJoint >** (p. 761), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::SliderJoint< BulletJoint >** (p. 806), and **gazebo::physics::BulletBallJoint** (p. 168).

References gzerr.

10.17.3.10 virtual void gazebo::physics::BulletJoint::SetAttribute (Attribute , int , double) [inline], [virtual]

Set a parameter for the joint.

Implements **gazebo::physics::Joint** (p. 435).

References gzerr.

10.17.3.11 virtual void gazebo::physics::BulletJoint::SetDamping (int , const double) [inline], [virtual]

Set the joint damping.

Implements **gazebo::physics::Joint** (p. 435).

Reimplemented in **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::BulletScrewJoint** (p. 216), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::BulletSliderJoint** (p. 220), and **gazebo::physics::BulletBallJoint** (p. 169).

References gzerr.

10.17.4 Member Data Documentation

10.17.4.1 btTypedConstraint* gazebo::physics::BulletJoint::constraint [protected]

10.17.4.2 btDynamicsWorld* gazebo::physics::BulletJoint::world [protected]

The documentation for this class was generated from the following file:

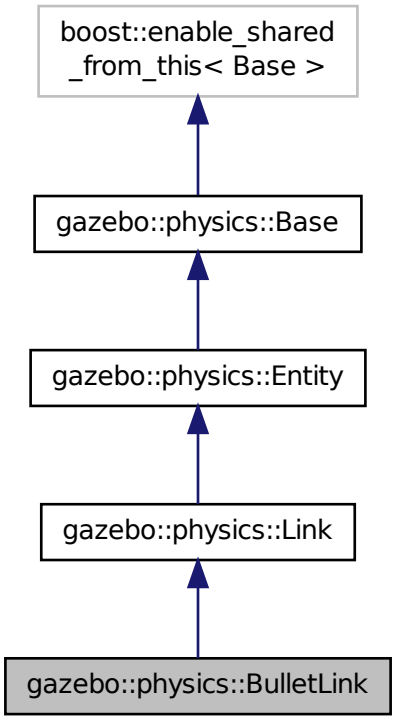
- **BulletJoint.hh**

10.18 gazebo::physics::BulletLink Class Reference

Bullet **Link** (p. 454) class.

`#include <BulletLink.hh>`

Inheritance diagram for gazebo::physics::BulletLink:



Public Member Functions

- **BulletLink** (**EntityPtr** _parent)

  Constructor.
- virtual ~**BulletLink** ()

  Destructor.
- virtual void **AddForce** (const **math::Vector3** &_force)

  Set the relative pose of a child collision.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)

  Add a force to the body at position expressed to the body's own frame of reference.
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)

  Add a force to the body using a global position.
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)

Add a force to the body, components are relative to the body's own frame of reference.

- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)

  Add a torque to the body, components are relative to the body's own frame of reference.
- virtual void **AddTorque** (const **math::Vector3** &_torque)

  Add a torque to the body.
- virtual void **Fini** ()

  Finalize the body.
- btRigidBody * **GetBulletLink** () const

  Get the bullet rigid body.
- virtual bool **GetEnabled** () const

  Get whether this body is enabled in the physics engine.
- virtual bool **GetGravityMode** ()

  Get the gravity mode.
- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the body in the world frame.
- virtual **math::Vector3 GetWorldForce** () const

  Get the force applied to the body in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the body in the world frame.
- virtual **math::Vector3 GetWorldTorque** () const

  Get the torque applied to the body in the world frame.
- virtual void **Init** ()

  Initialize the body.
- virtual void **Load** (**sdf::ElementPtr** _ptr)

  Load the body based on an common::XMLConfig node.
- virtual void **OnPoseChange** ()

  Called when the pose of the entity (or one of its parents) has changed.
- virtual void **SetAngularDamping** (double damping)

  Set the angular damping factor.
- virtual void **SetAngularVel** (const **math::Vector3** &vel)

  Set the angular velocity of the body.
- virtual void **SetAutoDisable** (bool _disable)

  Allow the link to auto disable.
- virtual void **SetEnabled** (bool enable) const

  Set whether this body is enabled.
- virtual void **SetForce** (const **math::Vector3** &force)

  Set the force applied to the body.
- virtual void **SetGravityMode** (bool mode)

  Set whether gravity affects this body.
- virtual void **SetLinearDamping** (double damping)

  Set the linear damping factor.
- virtual void **SetLinearVel** (const **math::Vector3** &vel)

  Set the linear velocity of the body.
- void **SetSelfCollide** (bool collide)

  Set whether this body will collide with others in the model.
- virtual void **SetTorque** (const **math::Vector3** &force)

  Set the torque applied to the body.

- virtual void **Update** ()

  Update the body.
- virtual void **UpdateCoM** ()

  Update the center of mass.

Protected Attributes

- **math::Pose pose**

Additional Inherited Members

10.18.1 Detailed Description

Bullet **Link** (p. 454) class.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 gazebo::physics::BulletLink::BulletLink (EntityPtr *_parent*)

Constructor.

10.18.2.2 virtual gazebo::physics::BulletLink::~BulletLink ( ) [virtual]

Destructor.

10.18.3 Member Function Documentation

10.18.3.1 virtual void gazebo::physics::BulletLink::AddForce (const math::Vector3 & *_force*) [virtual]

Set the relative pose of a child collision.

Add a force to the body

Implements **gazebo::physics::Link** (p. 459).

10.18.3.2 virtual void gazebo::physics::BulletLink::AddForceAtRelativePosition (const math::Vector3 & *_force,* const math::Vector3 & *_relpos*) [virtual]

Add a force to the body at position expressed to the body's own frame of reference.

Implements **gazebo::physics::Link** (p. 460).

10.18.3.3 virtual void gazebo::physics::BulletLink::AddForceAtWorldPosition (const math::Vector3 & *_force,* const math::Vector3 & *_pos*) [virtual]

Add a force to the body using a global position.

Implements **gazebo::physics::Link** (p. 460).

10.18.3.4 virtual void gazebo::physics::BulletLink::AddRelativeForce (const math::Vector3 & _force) [virtual]

Add a force to the body, components are relative to the body's own frame of reference.

Implements **gazebo::physics::Link** (p. 460).

10.18.3.5 virtual void gazebo::physics::BulletLink::AddRelativeTorque (const math::Vector3 & _torque) [virtual]

Add a torque to the body, components are relative to the body's own frame of reference.

Implements **gazebo::physics::Link** (p. 460).

10.18.3.6 virtual void gazebo::physics::BulletLink::AddTorque (const math::Vector3 & _torque) [virtual]

Add a torque to the body.

Implements **gazebo::physics::Link** (p. 461).

10.18.3.7 virtual void gazebo::physics::BulletLink::Fini ( ) [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Entity** (p. 342).

10.18.3.8 btRigidBody* gazebo::physics::BulletLink::GetBulletLink ( ) const

Get the bullet rigid body.

10.18.3.9 virtual bool gazebo::physics::BulletLink::GetEnabled ( ) const [inline], [virtual]

Get whether this body is enabled in the physics engine.

Implements **gazebo::physics::Link** (p. 463).

10.18.3.10 virtual bool gazebo::physics::BulletLink::GetGravityMode ( ) [virtual]

Get the gravity mode.

Implements **gazebo::physics::Link** (p. 463).

10.18.3.11 virtual math::Vector3 gazebo::physics::BulletLink::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.18.3.12 virtual math::Vector3 gazebo::physics::BulletLink::GetWorldForce ( ) const [virtual]

Get the force applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 467).

10.18.3.13 virtual math::Vector3 gazebo::physics::BulletLink::GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 345).

10.18.3.14 virtual math::Vector3 gazebo::physics::BulletLink::GetWorldTorque ( ) const [virtual]

Get the torque applied to the body in the world frame.

Implements **gazebo::physics::Link** (p. 467).

10.18.3.15 virtual void gazebo::physics::BulletLink::Init ( ) [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Link** (p. 467).

10.18.3.16 virtual void gazebo::physics::BulletLink::Load (sdf::ElementPtr *_ptr*) [virtual]

Load the body based on an common::XMLConfig node.

Reimplemented from **gazebo::physics::Link** (p. 468).

10.18.3.17 virtual void gazebo::physics::BulletLink::OnPoseChange ( ) [virtual]

Called when the pose of the entity (or one of its parents) has changed.

Reimplemented from **gazebo::physics::Link** (p. 468).

10.18.3.18 virtual void gazebo::physics::BulletLink::SetAngularDamping (double *damping*) [virtual]

Set the angular damping factor.

Implements **gazebo::physics::Link** (p. 469).

10.18.3.19 virtual void gazebo::physics::BulletLink::SetAngularVel (const math::Vector3 & *vel*) [virtual]

Set the angular velocity of the body.

Implements **gazebo::physics::Link** (p. 469).

10.18.3.20 virtual void gazebo::physics::BulletLink::SetAutoDisable (bool *_disable*) [virtual]

Allow the link to auto disable.

Parameters

in	*_disable*	If true, the link is allowed to auto disable.

Implements **gazebo::physics::Link** (p. 469).

10.18.3.21 virtual void gazebo::physics::BulletLink::SetEnabled (bool *enable*) const [virtual]

Set whether this body is enabled.

Implements **gazebo::physics::Link** (p. 470).

10.18.3.22 virtual void gazebo::physics::BulletLink::SetForce (const math::Vector3 & *force*) [virtual]

Set the force applied to the body.

Implements **gazebo::physics::Link** (p. 470).

10.18.3.23 virtual void gazebo::physics::BulletLink::SetGravityMode (bool *mode*) [virtual]

Set whether gravity affects this body.

Implements **gazebo::physics::Link** (p. 470).

10.18.3.24 virtual void gazebo::physics::BulletLink::SetLinearDamping (double *damping*) [virtual]

Set the linear damping factor.

Implements **gazebo::physics::Link** (p. 471).

10.18.3.25 virtual void gazebo::physics::BulletLink::SetLinearVel (const math::Vector3 & *vel*) [virtual]

Set the linear velocity of the body.

Implements **gazebo::physics::Link** (p. 471).

10.18.3.26 void gazebo::physics::BulletLink::SetSelfCollide (bool *collide*) [virtual]

Set whether this body will collide with others in the model.

Implements **gazebo::physics::Link** (p. 471).

10.18.3.27 virtual void gazebo::physics::BulletLink::SetTorque (const math::Vector3 & *force*) [virtual]

Set the torque applied to the body.

Implements **gazebo::physics::Link** (p. 472).

10.18.3.28 virtual void gazebo::physics::BulletLink::Update ( ) [virtual]

Update the body.

Reimplemented from **gazebo::physics::Link** (p. 472).

10.18.3.29 virtual void gazebo::physics::BulletLink::UpdateCoM ( ) [virtual]

Update the center of mass.

10.18.4 Member Data Documentation

10.18.4.1 math::Pose gazebo::physics::BulletLink::pose [protected]

The documentation for this class was generated from the following file:

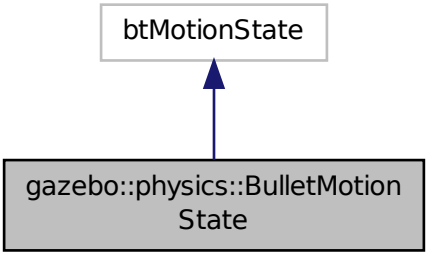
- **BulletLink.hh**

10.19 gazebo::physics::BulletMotionState Class Reference

Bullet btMotionState encapsulation.

`#include <BulletMotionState.hh>`

Inheritance diagram for gazebo::physics::BulletMotionState:



Public Member Functions

- **BulletMotionState** (**Link** *_link)

  Constructor.
- virtual ~**BulletMotionState** ()

  Constructor.
- **math::Pose GetWorldPose** () const

  Get the pose.
- virtual void **getWorldTransform** (btTransform &_worldTrans) const

  Get the world transform.
- void **SetCoG** (const **math::Vector3** &_cog)

  Set the center of mass offset.
- void **SetWorldPose** (const **math::Pose** &_pose)

  Set the pose.
- virtual void **SetWorldPosition** (const **math::Vector3** &_pos)

  Set the position of the body.
- virtual void **SetWorldRotation** (const **math::Quaternion** &_rot)

  Set the rotation of the body.

- virtual void **setWorldTransform** (const btTransform &_worldTrans)

  Set the world transform.

10.19.1 Detailed Description

Bullet btMotionState encapsulation.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 gazebo::physics::BulletMotionState::BulletMotionState (Link * _link)

Constructor.

10.19.2.2 virtual gazebo::physics::BulletMotionState::~BulletMotionState ( ) [virtual]

Constructor.

Destructor

10.19.3 Member Function Documentation

10.19.3.1 math::Pose gazebo::physics::BulletMotionState::GetWorldPose ( ) const

Get the pose.

10.19.3.2 virtual void gazebo::physics::BulletMotionState::getWorldTransform (btTransform & _worldTrans) const [virtual]

Get the world transform.

10.19.3.3 void gazebo::physics::BulletMotionState::SetCoG (const math::Vector3 & _cog)

Set the center of mass offset.

10.19.3.4 void gazebo::physics::BulletMotionState::SetWorldPose (const math::Pose & _pose)

Set the pose.

10.19.3.5 virtual void gazebo::physics::BulletMotionState::SetWorldPosition (const math::Vector3 & _pos) [virtual]

Set the position of the body.

Parameters

pos	math::Vector position

10.19.3.6 virtual void gazebo::physics::BulletMotionState::SetWorldRotation (const math::Quaternion & *_rot*) [virtual]

Set the rotation of the body.

Parameters

rot	Quaternion rotation

10.19.3.7 virtual void gazebo::physics::BulletMotionState::setWorldTransform (const btTransform & *_worldTrans*) [virtual]

Set the world transform.

The documentation for this class was generated from the following file:

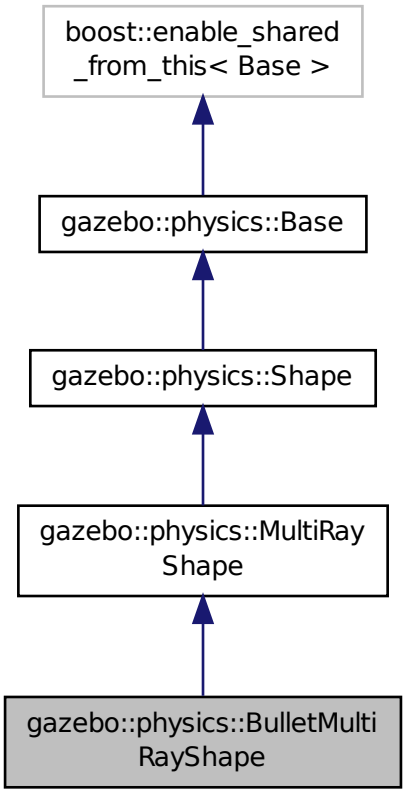
- **BulletMotionState.hh**

10.20 gazebo::physics::BulletMultiRayShape Class Reference

Bullet specific version of **MultiRayShape** (p. 549).

`#include <BulletMultiRayShape.hh>`

Inheritance diagram for gazebo::physics::BulletMultiRayShape:



Public Member Functions

- **BulletMultiRayShape** (**CollisionPtr parent**)

  Constructor.
- virtual ~**BulletMultiRayShape** ()

  Destructor.
- virtual void **UpdateRays** ()

  Update the rays.

Protected Member Functions

- void **AddRay** (const **math::Vector3** &start, const **math::Vector3** &end)

  Add a ray to the collision.

Additional Inherited Members

10.20.1 Detailed Description

Bullet specific version of **MultiRayShape** (p. 549).

10.20.2 Constructor & Destructor Documentation

10.20.2.1 gazebo::physics::BulletMultiRayShape::BulletMultiRayShape (CollisionPtr *parent*)

Constructor.

10.20.2.2 virtual gazebo::physics::BulletMultiRayShape::~BulletMultiRayShape ( ) [virtual]

Destructor.

10.20.3 Member Function Documentation

10.20.3.1 void gazebo::physics::BulletMultiRayShape::AddRay (const math::Vector3 & *start,* const math::Vector3 & *end*) [protected], [virtual]

Add a ray to the collision.

Reimplemented from **gazebo::physics::MultiRayShape** (p. 551).

10.20.3.2 virtual void gazebo::physics::BulletMultiRayShape::UpdateRays ( ) [virtual]

Update the rays.

Implements **gazebo::physics::MultiRayShape** (p. 555).

The documentation for this class was generated from the following file:

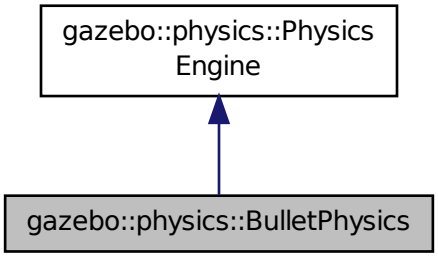
- **BulletMultiRayShape.hh**

10.21 gazebo::physics::BulletPhysics Class Reference

Bullet physics engine.

`#include <BulletPhysics.hh>`

Inheritance diagram for gazebo::physics::BulletPhysics:



Public Member Functions

- **BulletPhysics** (**WorldPtr** _world)

  Constructor.
- virtual ∼**BulletPhysics** ()

  Destructor.
- virtual void **ConvertMass** (**InertialPtr** _inertial, void ∗_engineMass)

  Create a physics based ray sensor.
- virtual void **ConvertMass** (void ∗_engineMass, **InertialPtr** _inertial)

  Convert an gazebo Mass to a bullet Mass.
- virtual **CollisionPtr CreateCollision** (const std::string &_type, **LinkPtr** _body)

  Create a new collision.
- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent)

  Create a new joint.
- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)

  Create a new body.
- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)

  *Create a **physics::Shape** (p. 779) object.*
- virtual void **DebugPrint** () const

  Debug print out of the physic engine state.
- virtual void **Fini** ()

  Finilize the Bullet engine.
- btDynamicsWorld ∗ **GetDynamicsWorld** () const

  Register a joint with the dynamics world.
- virtual double **GetStepTime** ()

  Get the simulation step time.
- virtual void **Init** ()

  Initialize the Bullet engine.
- virtual void **InitForThread** ()

  Init the engine for threads.

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the Bullet engine.
- virtual void **SetGravity** (const **gazebo::math::Vector3** &gravity)

  Set the gavity vector.
- virtual void **SetStepTime** (double _value)

  Set the simulation step time.
- virtual void **UpdateCollision** ()

  Update the Bullet collision.
- virtual void **UpdatePhysics** ()

  Update the Bullet engine.

Static Public Member Functions

- static **math::Pose ConvertPose** (const btTransform &_bt)

  Convert a bullet transform to a gazebo pose.
- static btTransform **ConvertPose** (const **math::Pose** &_pose)

  Convert a gazebo pose to a bullet transform.

Additional Inherited Members

10.21.1 Detailed Description

Bullet physics engine.

10.21.2 Constructor & Destructor Documentation

10.21.2.1 gazebo::physics::BulletPhysics::BulletPhysics (WorldPtr *_world*)

Constructor.

10.21.2.2 virtual gazebo::physics::BulletPhysics::~BulletPhysics ( ) [virtual]

Destructor.

10.21.3 Member Function Documentation

10.21.3.1 virtual void gazebo::physics::BulletPhysics::ConvertMass (InertialPtr *_inertial*, void * *_engineMass*) [virtual]

Create a physics based ray sensor.

Convert an bullet mass to a gazebo Mass

10.21.3.2 virtual void gazebo::physics::BulletPhysics::ConvertMass (void * *_engineMass*, InertialPtr *_inertial*) [virtual]

Convert an gazebo Mass to a bullet Mass.

10.21.3.3 static math::Pose gazebo::physics::BulletPhysics::ConvertPose (const btTransform & *_bt*) [static]

Convert a bullet transform to a gazebo pose.

10.21.3.4 static btTransform gazebo::physics::BulletPhysics::ConvertPose (const math::Pose & *_pose*) [static]

Convert a gazebo pose to a bullet transform.

10.21.3.5 virtual CollisionPtr gazebo::physics::BulletPhysics::CreateCollision (const std::string & *_type,* LinkPtr *_body*) [virtual]

Create a new collision.

Implements **gazebo::physics::PhysicsEngine** (p. 654).

10.21.3.6 virtual JointPtr gazebo::physics::BulletPhysics::CreateJoint (const std::string & *_type,* ModelPtr *_parent*) [virtual]

Create a new joint.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.21.3.7 virtual LinkPtr gazebo::physics::BulletPhysics::CreateLink (ModelPtr *_parent*) [virtual]

Create a new body.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.21.3.8 virtual ShapePtr gazebo::physics::BulletPhysics::CreateShape (const std::string & *_shapeType,* CollisionPtr *_collision*) [virtual]

Create a **physics::Shape** (p. 779) object.

Parameters

in	*_shapeType*	Type of shape to create.
in	*_collision*	**Collision** (p. 262) parent.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.21.3.9 virtual void gazebo::physics::BulletPhysics::DebugPrint ( ) const [virtual]

Debug print out of the physic engine state.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.21.3.10 virtual void gazebo::physics::BulletPhysics::Fini ( ) [virtual]

Finilize the Bullet engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 656).

10.21.3.11 btDynamicsWorld* gazebo::physics::BulletPhysics::GetDynamicsWorld ( ) const [inline]

Register a joint with the dynamics world.

10.21.3.12 virtual double gazebo::physics::BulletPhysics::GetStepTime ( ) [virtual]

Get the simulation step time.

Implements **gazebo::physics::PhysicsEngine** (p. 658).

10.21.3.13 virtual void gazebo::physics::BulletPhysics::Init ( ) [virtual]

Initialize the Bullet engine.

Implements **gazebo::physics::PhysicsEngine** (p. 659).

10.21.3.14 virtual void gazebo::physics::BulletPhysics::InitForThread ( ) [virtual]

Init the engine for threads.

Implements **gazebo::physics::PhysicsEngine** (p. 659).

10.21.3.15 virtual void gazebo::physics::BulletPhysics::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the Bullet engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 659).

10.21.3.16 virtual void gazebo::physics::BulletPhysics::SetGravity (const gazebo::math::Vector3 & *gravity*) [virtual]

Set the gavity vector.

Implements **gazebo::physics::PhysicsEngine** (p. 660).

10.21.3.17 virtual void gazebo::physics::BulletPhysics::SetStepTime (double *_value*) [virtual]

Set the simulation step time.

Implements **gazebo::physics::PhysicsEngine** (p. 661).

10.21.3.18 virtual void gazebo::physics::BulletPhysics::UpdateCollision ( ) [virtual]

Update the Bullet collision.

Implements **gazebo::physics::PhysicsEngine** (p. 662).

10.21.3.19 virtual void gazebo::physics::BulletPhysics::UpdatePhysics ( ) [virtual]

Update the Bullet engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 662).

The documentation for this class was generated from the following file:

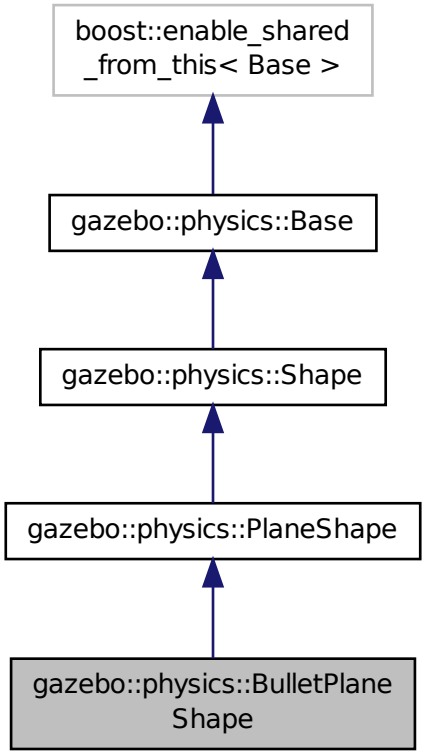
- **BulletPhysics.hh**

10.22 gazebo::physics::BulletPlaneShape Class Reference

Bullet collision for an infinite plane.

`#include <BulletPlaneShape.hh>`

Inheritance diagram for gazebo::physics::BulletPlaneShape:



Public Member Functions

- **BulletPlaneShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**BulletPlaneShape** ()

  Destructor.
- void **CreatePlane** ()

  Create the plane.
- void **SetAltitude** (const **math::Vector3** &pos)

  Set the altitude of the plane.

Additional Inherited Members

10.22.1 Detailed Description

Bullet collision for an infinite plane.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 gazebo::physics::BulletPlaneShape::BulletPlaneShape (CollisionPtr *_parent*) [inline]

Constructor.

10.22.2.2 virtual gazebo::physics::BulletPlaneShape::~BulletPlaneShape ( ) [inline], [virtual]

Destructor.

10.22.3 Member Function Documentation

10.22.3.1 void gazebo::physics::BulletPlaneShape::CreatePlane ( ) [inline], [virtual]

Create the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 673).

References gazebo::physics::Shape::collisionParent, gazebo::physics::PlaneShape::CreatePlane(), gazebo::physics::PlaneShape::GetNormal(), and gazebo::physics::BulletCollision::SetCollisionShape().

10.22.3.2 void gazebo::physics::BulletPlaneShape::SetAltitude (const math::Vector3 & *pos*) [inline], [virtual]

Set the altitude of the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 673).

References gazebo::physics::PlaneShape::SetAltitude().

The documentation for this class was generated from the following file:

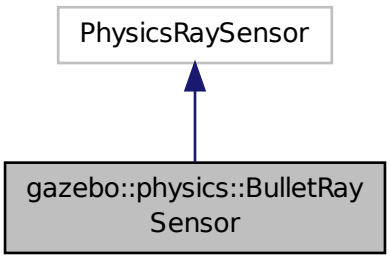
- **BulletPlaneShape.hh**

10.23 gazebo::physics::BulletRaySensor Class Reference

An Bullet Ray sensor.

`#include <BulletRaySensor.hh>`

Inheritance diagram for gazebo::physics::BulletRaySensor:



Public Member Functions

- **BulletRaySensor** (**Link** ∗body)

  Constructor.
- virtual ∼**BulletRaySensor** ()

  Destructor.
- void **AddRay** (**math::Vector3** start, **math::Vector3** end, double minRange, double maxRange, bool display)

  Add a ray to the sensor.
- int **GetCount** () const

  Get the number of rays.
- double **GetFiducial** (int index) const

  Get the fiducial value of a ray.
- double **GetRange** (int index) const

  Get the range of a ray.
- void **GetRelativePoints** (int index, **math::Vector3** &a, **math::Vector3** &b)

  Get the relative starting and ending points of a ray.
- double **GetRetro** (int index) const

  Get the retro reflectance value of a ray.
- virtual void **Update** ()

  Update the ray sensor.

10.23.1 Detailed Description

An Bullet Ray sensor.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 gazebo::physics::BulletRaySensor::BulletRaySensor (Link ∗ *body*)

Constructor.

10.23.2.2 virtual gazebo::physics::BulletRaySensor::~BulletRaySensor ( ) [virtual]

Destructor.

10.23.3 Member Function Documentation

10.23.3.1 void gazebo::physics::BulletRaySensor::AddRay (math::Vector3 *start*, math::Vector3 *end*, double *minRange*, double *maxRange*, bool *display*)

Add a ray to the sensor.

10.23.3.2 int gazebo::physics::BulletRaySensor::GetCount ( ) const

Get the number of rays.

10.23.3.3 double gazebo::physics::BulletRaySensor::GetFiducial (int *index*) const

Get the fiducial value of a ray.

10.23.3.4 double gazebo::physics::BulletRaySensor::GetRange (int *index*) const

Get the range of a ray.

10.23.3.5 void gazebo::physics::BulletRaySensor::GetRelativePoints (int *index*, math::Vector3 & *a*, math::Vector3 & *b*)

Get the relative starting and ending points of a ray.

10.23.3.6 double gazebo::physics::BulletRaySensor::GetRetro (int *index*) const

Get the retro reflectance value of a ray.

10.23.3.7 virtual void gazebo::physics::BulletRaySensor::Update ( ) [virtual]

Update the ray sensor.

The documentation for this class was generated from the following file:

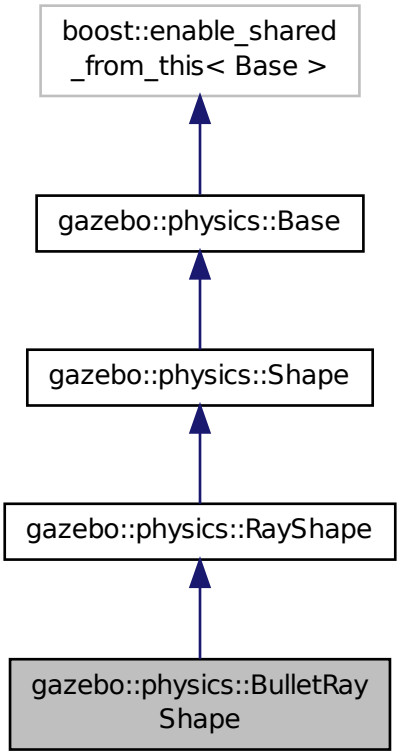
- **BulletRaySensor.hh**

10.24 gazebo::physics::BulletRayShape Class Reference

Ray shape for bullet.

`#include <BulletRayShape.hh>`

Inheritance diagram for gazebo::physics::BulletRayShape:



Public Member Functions

- **BulletRayShape** (**PhysicsEnginePtr** _physicsEngine)
- **BulletRayShape** (**CollisionPtr** _collision)

  Constructor.
- virtual ~**BulletRayShape** ()

  Destructor.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)

  Get the nearest intersection.
- void **SetPoints** (const **math::Vector3** &_posStart, const **math::Vector3** &_posEnd)

  Set the ray based on starting and ending points relative to the body.
- virtual void **Update** ()

  Update the ray collision.

Additional Inherited Members

10.24.1 Detailed Description

Ray shape for bullet.

10.24.2 Constructor & Destructor Documentation

10.24.2.1 gazebo::physics::BulletRayShape::BulletRayShape (PhysicsEnginePtr *_physicsEngine*)

10.24.2.2 gazebo::physics::BulletRayShape::BulletRayShape (CollisionPtr *_collision*)

Constructor.

Parameters

body	**Link** (p. 454) the ray is attached to

10.24.2.3 virtual gazebo::physics::BulletRayShape::~BulletRayShape ( ) [virtual]

Destructor.

10.24.3 Member Function Documentation

10.24.3.1 virtual void gazebo::physics::BulletRayShape::GetIntersection (double & *_dist,* std::string & *_entity*) [virtual]

Get the nearest intersection.

Implements **gazebo::physics::RayShape** (p. 720).

10.24.3.2 void gazebo::physics::BulletRayShape::SetPoints (const math::Vector3 & *_posStart,* const math::Vector3 & *_posEnd*) [virtual]

Set the ray based on starting and ending points relative to the body.

Parameters

posStart	Start position, relative the body
posEnd	End position, relative to the body

Reimplemented from **gazebo::physics::RayShape** (p. 721).

10.24.3.3 virtual void gazebo::physics::BulletRayShape::Update ( ) [virtual]

Update the ray collision.

Implements **gazebo::physics::RayShape** (p. 722).

The documentation for this class was generated from the following file:

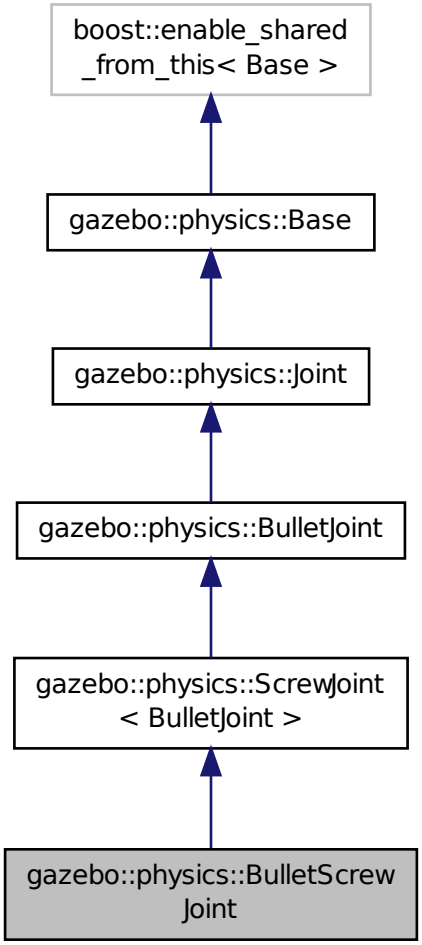
- **BulletRayShape.hh**

10.25 gazebo::physics::BulletScrewJoint Class Reference

A screw joint.

`#include <BulletScrewJoint.hh>`

Inheritance diagram for gazebo::physics::BulletScrewJoint:



Public Member Functions

- **BulletScrewJoint** (btDynamicsWorld ∗**world**, **BasePtr** _parent)

  Constructor.
- virtual ∼**BulletScrewJoint** ()

  Destructor.
- void **Attach** (**LinkPtr** _one, **LinkPtr** _two)

Attach the two bodies with this joint.

- virtual **math::Angle GetAngle** (int _index) const

  Get the position of the joint.
- virtual **math::Angle GetAngleImpl** (int _index) const

  Get the angle of rotation.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  Get the axis of rotation.
- virtual **math::Angle GetHighStop** (int _index)

  Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)

  Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const

  Get the rate of change.
- void **SetAxis** (int _index, const **math::Vector3** &_axis)

  Set the axis of motion.
- virtual void **SetDamping** (int _index, double _damping)

  Set joint damping, not yet implemented.
- virtual void **SetForce** (int _index, double _force)

  Set the screw force.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)

  Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load the **BulletScrewJoint** (*p. *213**).*

Additional Inherited Members

10.25.1 Detailed Description

A screw joint.

10.25.2 Constructor & Destructor Documentation

10.25.2.1 gazebo::physics::BulletScrewJoint::BulletScrewJoint (btDynamicsWorld * *world,* BasePtr *_parent*)

Constructor.

10.25.2.2 virtual gazebo::physics::BulletScrewJoint::~BulletScrewJoint ( ) [virtual]

Destructor.

10.25.3 Member Function Documentation

10.25.3.1 void gazebo::physics::BulletScrewJoint::Attach (LinkPtr *_one*, LinkPtr *_two*) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.25.3.2 virtual math::Angle gazebo::physics::BulletScrewJoint::GetAngle (int *_index*) const [virtual]

Get the position of the joint.

10.25.3.3 virtual math::Angle gazebo::physics::BulletScrewJoint::GetAngleImpl (int *_index*) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.25.3.4 virtual math::Vector3 gazebo::physics::BulletScrewJoint::GetGlobalAxis (int *_index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.25.3.5 virtual math::Angle gazebo::physics::BulletScrewJoint::GetHighStop (int *_index*) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

10.25.3.6 virtual math::Angle gazebo::physics::BulletScrewJoint::GetLowStop (int *_index*) [virtual]

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.25.3.7 virtual double gazebo::physics::BulletScrewJoint::GetMaxForce (int *_index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.25.3.8 virtual double gazebo::physics::BulletScrewJoint::GetVelocity (int *_index*) const [virtual]

Get the rate of change.

Implements **gazebo::physics::Joint** (p. 433).

10.25.3.9 virtual void gazebo::physics::BulletScrewJoint::Load (sdf::ElementPtr _sdf) [protected], [virtual]

Load the **BulletScrewJoint** (p. 213).

Reimplemented from **gazebo::physics::ScrewJoint< BulletJoint >** (p. 761).

10.25.3.10 void gazebo::physics::BulletScrewJoint::SetAxis (int _index, const math::Vector3 & _axis) [virtual]

Set the axis of motion.

Implements **gazebo::physics::Joint** (p. 435).

10.25.3.11 virtual void gazebo::physics::BulletScrewJoint::SetDamping (int _index, double _damping) [virtual]

Set joint damping, not yet implemented.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.25.3.12 virtual void gazebo::physics::BulletScrewJoint::SetForce (int _index, double _force) [virtual]

Set the screw force.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.25.3.13 virtual void gazebo::physics::BulletScrewJoint::SetHighStop (int _index, const math::Angle & _angle) [virtual]

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.25.3.14 virtual void gazebo::physics::BulletScrewJoint::SetLowStop (int _index, const math::Angle & _angle) [virtual]

Set the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.25.3.15 virtual void gazebo::physics::BulletScrewJoint::SetMaxForce (int _index, double _t) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.25.3.16 virtual void gazebo::physics::BulletScrewJoint::SetVelocity (int _index, double _angle) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

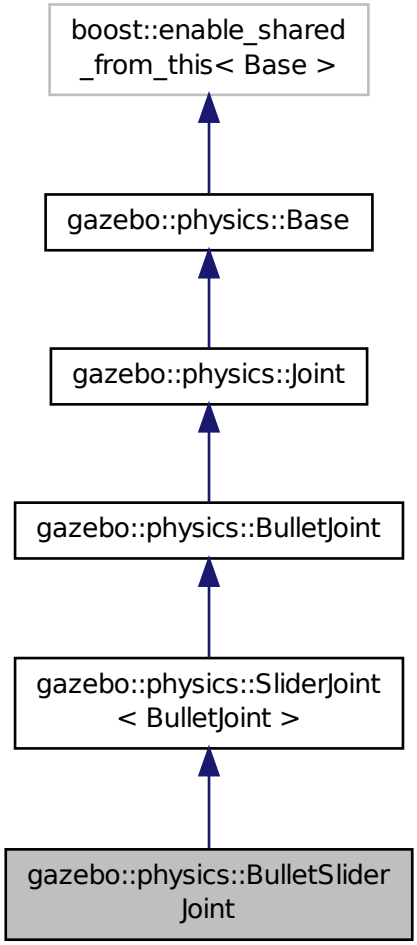
- **BulletScrewJoint.hh**

10.26 gazebo::physics::BulletSliderJoint Class Reference

A slider joint.

`#include <BulletSliderJoint.hh>`

Inheritance diagram for gazebo::physics::BulletSliderJoint:



Public Member Functions

- **BulletSliderJoint** (btDynamicsWorld ∗**world**, **BasePtr** _parent)

  Constructor.
- virtual ∼**BulletSliderJoint** ()

  Destructor.
- void **Attach** (**LinkPtr** _one, **LinkPtr** _two)

Attach the two bodies with this joint.

- virtual **math::Angle GetAngle** (int _index) const

  Get the position of the joint.
- virtual **math::Angle GetAngleImpl** (int _index) const

  Get the angle of rotation.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  Get the axis of rotation.
- virtual **math::Angle GetHighStop** (int _index)

  Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)

  Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const

  Get the rate of change.
- void **SetAxis** (int _index, const **math::Vector3** &_axis)

  Set the axis of motion.
- virtual void **SetDamping** (int _index, const double _damping)

  Set joint damping, not yet implemented.
- virtual void **SetForce** (int _index, double _force)

  Set the slider force.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)

  Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load the **BulletSliderJoint** (p. 217).*

Additional Inherited Members

10.26.1 Detailed Description

A slider joint.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 gazebo::physics::BulletSliderJoint::BulletSliderJoint (btDynamicsWorld * *world,* BasePtr *_parent*)

Constructor.

10.26.2.2 virtual gazebo::physics::BulletSliderJoint::~BulletSliderJoint ( ) [virtual]

Destructor.

10.26.3 Member Function Documentation

10.26.3.1 void gazebo::physics::BulletSliderJoint::Attach (LinkPtr *_one,* LinkPtr *_two*) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.26.3.2 virtual math::Angle gazebo::physics::BulletSliderJoint::GetAngle (int *_index*) const [virtual]

Get the position of the joint.

10.26.3.3 virtual math::Angle gazebo::physics::BulletSliderJoint::GetAngleImpl (int *_index*) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.26.3.4 virtual math::Vector3 gazebo::physics::BulletSliderJoint::GetGlobalAxis (int *_index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.26.3.5 virtual math::Angle gazebo::physics::BulletSliderJoint::GetHighStop (int *_index*) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

10.26.3.6 virtual math::Angle gazebo::physics::BulletSliderJoint::GetLowStop (int *_index*) [virtual]

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.26.3.7 virtual double gazebo::physics::BulletSliderJoint::GetMaxForce (int *_index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.26.3.8 virtual double gazebo::physics::BulletSliderJoint::GetVelocity (int *_index*) const [virtual]

Get the rate of change.

Implements **gazebo::physics::Joint** (p. 433).

10.26.3.9 virtual void gazebo::physics::BulletSliderJoint::Load (sdf::ElementPtr _sdf) [protected], [virtual]

Load the **BulletSliderJoint** (p. 217).

Reimplemented from **gazebo::physics::SliderJoint**< **BulletJoint** > (p. 805).

10.26.3.10 void gazebo::physics::BulletSliderJoint::SetAxis (int _index, const math::Vector3 & _axis) [virtual]

Set the axis of motion.

Implements **gazebo::physics::Joint** (p. 435).

10.26.3.11 virtual void gazebo::physics::BulletSliderJoint::SetDamping (int _index, const double _damping) [virtual]

Set joint damping, not yet implemented.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.26.3.12 virtual void gazebo::physics::BulletSliderJoint::SetForce (int _index, double _force) [virtual]

Set the slider force.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.26.3.13 virtual void gazebo::physics::BulletSliderJoint::SetHighStop (int _index, const math::Angle & _angle) [virtual]

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.26.3.14 virtual void gazebo::physics::BulletSliderJoint::SetLowStop (int _index, const math::Angle & _angle) [virtual]

Set the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.26.3.15 virtual void gazebo::physics::BulletSliderJoint::SetMaxForce (int _index, double _t) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.26.3.16 virtual void gazebo::physics::BulletSliderJoint::SetVelocity (int _index, double _angle) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

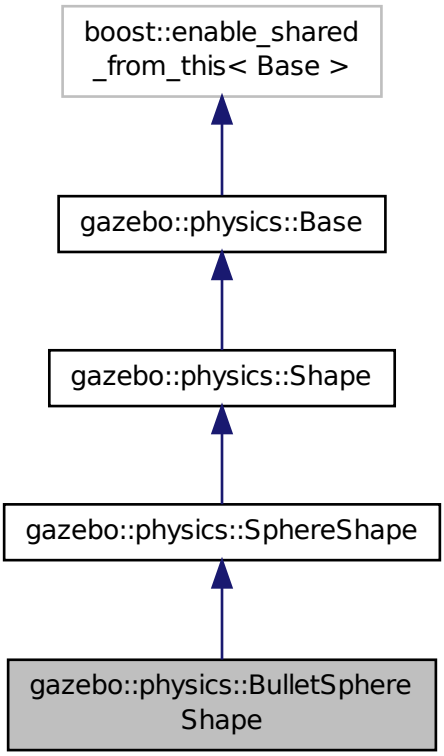
- **BulletSliderJoint.hh**

10.27 gazebo::physics::BulletSphereShape Class Reference

Bullet sphere collision.

`#include <BulletSphereShape.hh>`

Inheritance diagram for gazebo::physics::BulletSphereShape:



Public Member Functions

- **BulletSphereShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**BulletSphereShape** ()

  Destructor.
- void **SetRadius** (const double &_radius)

  Set the radius.

Additional Inherited Members

10.27.1 Detailed Description

Bullet sphere collision.

10.27.2 Constructor & Destructor Documentation

10.27.2.1 gazebo::physics::BulletSphereShape::BulletSphereShape (CollisionPtr *_parent*) [inline]

Constructor.

10.27.2.2 virtual gazebo::physics::BulletSphereShape::~BulletSphereShape ( ) [inline], [virtual]

Destructor.

10.27.3 Member Function Documentation

10.27.3.1 void gazebo::physics::BulletSphereShape::SetRadius (const double & *_radius*) [inline]

Set the radius.

References gazebo::physics::Shape::collisionParent, gazebo::physics::BulletCollision::SetCollisionShape(), and gazebo::physics::SphereShape::SetRadius().

The documentation for this class was generated from the following file:

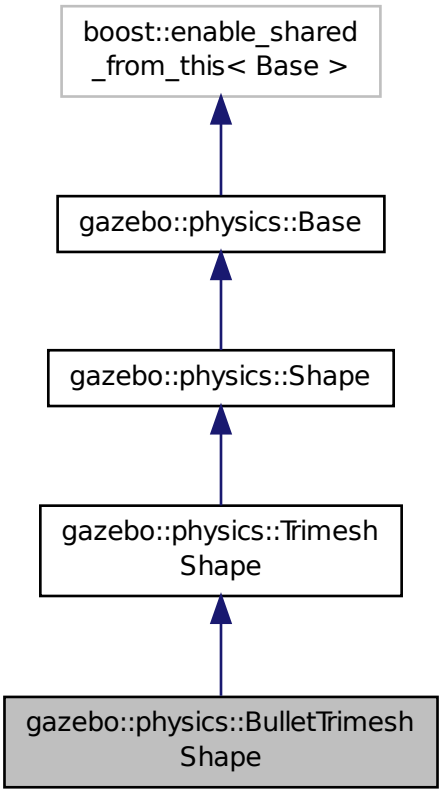
- **BulletSphereShape.hh**

10.28 gazebo::physics::BulletTrimeshShape Class Reference

Triangle mesh collision.

`#include <BulletTrimeshShape.hh>`

Inheritance diagram for gazebo::physics::BulletTrimeshShape:



Public Member Functions

- **BulletTrimeshShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**BulletTrimeshShape** ()

  Destructor.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the trimesh.

Protected Member Functions

- virtual void **Init** ()

  Init the trimesh shape.

Additional Inherited Members

10.28.1 Detailed Description

Triangle mesh collision.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 gazebo::physics::BulletTrimeshShape::BulletTrimeshShape (CollisionPtr *_parent*)

Constructor.

10.28.2.2 virtual gazebo::physics::BulletTrimeshShape::~BulletTrimeshShape ( ) [virtual]

Destructor.

10.28.3 Member Function Documentation

10.28.3.1 virtual void gazebo::physics::BulletTrimeshShape::Init ( ) [protected], [virtual]

Init the trimesh shape.

Reimplemented from **gazebo::physics::TrimeshShape** (p. 868).

10.28.3.2 virtual void gazebo::physics::BulletTrimeshShape::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the trimesh.

Reimplemented from **gazebo::physics::Base** (p. 154).

The documentation for this class was generated from the following file:

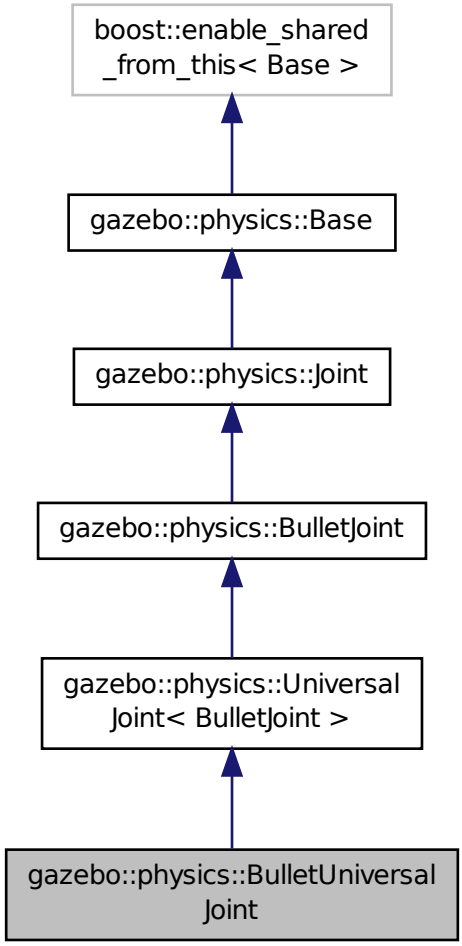
- **BulletTrimeshShape.hh**

10.29 gazebo::physics::BulletUniversalJoint Class Reference

A bullet universal joint class.

`#include <BulletUniversalJoint.hh>`

Inheritance diagram for gazebo::physics::BulletUniversalJoint:



Public Member Functions

- **BulletUniversalJoint** (btDynamicsWorld ∗**world**, **BasePtr** _parent)

  Constructor.
- virtual ∼**BulletUniversalJoint** ()

  Destuctor.
- void **Attach** (**LinkPtr** _one, **LinkPtr** _two)

  Attach the two bodies with this joint.
- virtual **math::Vector3 GetAnchor** (int _index) const

  Get the anchor point.
- virtual **math::Angle GetAngle** (int _index) const

  Get the angle of axis 1.

- virtual **math::Angle GetAngleImpl** (int _index) const

  Get the angle of rotation.
- virtual **math::Vector3 GetAxis** (int _index) const

  Get the first axis of rotation.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const

  Get the axis of rotation.
- virtual **math::Angle GetHighStop** (int _index)

  Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int _index)

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)

  Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int _index) const

  Get the angular rate of axis 1.
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)

  Set the anchor point.
- void **SetAxis** (int _index, const **math::Vector3** &_axis)

  Set the first axis of rotation.
- virtual void **SetDamping** (int _index, const double _damping)

  Set joint damping, not yet implemented.
- virtual void **SetForce** (int _index, double _torque)

  Set the torque of a joint.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _t)

  Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int _index, double _angle)

  Set the velocity of an axis(index).

Additional Inherited Members

10.29.1 Detailed Description

A bullet universal joint class.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 gazebo::physics::BulletUniversalJoint::BulletUniversalJoint (btDynamicsWorld * *world,* **BasePtr** *_parent*)

Constructor.

10.29.2.2 virtual gazebo::physics::BulletUniversalJoint::~BulletUniversalJoint ( ) [virtual]

Destuctor.

10.29.3 Member Function Documentation

10.29.3.1 void gazebo::physics::BulletUniversalJoint::Attach (LinkPtr _one, LinkPtr _two) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.29.3.2 virtual math::Vector3 gazebo::physics::BulletUniversalJoint::GetAnchor (int _index) const [virtual]

Get the anchor point.

Reimplemented from **gazebo::physics::BulletJoint** (p. 190).

10.29.3.3 virtual math::Angle gazebo::physics::BulletUniversalJoint::GetAngle (int _index) const [virtual]

Get the angle of axis 1.

10.29.3.4 virtual math::Angle gazebo::physics::BulletUniversalJoint::GetAngleImpl (int _index) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.29.3.5 virtual math::Vector3 gazebo::physics::BulletUniversalJoint::GetAxis (int _index) const [virtual]

Get the first axis of rotation.

10.29.3.6 virtual math::Vector3 gazebo::physics::BulletUniversalJoint::GetGlobalAxis (int _index) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.29.3.7 virtual math::Angle gazebo::physics::BulletUniversalJoint::GetHighStop (int _index) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

10.29.3.8 virtual math::Angle gazebo::physics::BulletUniversalJoint::GetLowStop (int _index) [virtual]

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.29.3.9 virtual double gazebo::physics::BulletUniversalJoint::GetMaxForce (int _index) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.29.3.10 virtual double gazebo::physics::BulletUniversalJoint::GetVelocity (int _index) const [virtual]

Get the angular rate of axis 1.

Implements **gazebo::physics::Joint** (p. 433).

10.29.3.11 virtual void gazebo::physics::BulletUniversalJoint::SetAnchor (int _index, const math::Vector3 & _anchor) [virtual]

Set the anchor point.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.29.3.12 void gazebo::physics::BulletUniversalJoint::SetAxis (int _index, const math::Vector3 & _axis) [virtual]

Set the first axis of rotation.

Implements **gazebo::physics::Joint** (p. 435).

10.29.3.13 virtual void gazebo::physics::BulletUniversalJoint::SetDamping (int _index, const double _damping) [virtual]

Set joint damping, not yet implemented.

Reimplemented from **gazebo::physics::BulletJoint** (p. 191).

10.29.3.14 virtual void gazebo::physics::BulletUniversalJoint::SetForce (int _index, double _torque) [virtual]

Set the torque of a joint.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.29.3.15 virtual void gazebo::physics::BulletUniversalJoint::SetHighStop (int _index, const math::Angle & _angle) [virtual]

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.29.3.16 virtual void gazebo::physics::BulletUniversalJoint::SetLowStop (int _index, const math::Angle & _angle) [virtual]

Set the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.29.3.17 virtual void gazebo::physics::BulletUniversalJoint::SetMaxForce (int _index, double _t) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.29.3.18 virtual void gazebo::physics::BulletUniversalJoint::SetVelocity (int _index, double _angle) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

- **BulletUniversalJoint.hh**

10.30 gazebo::common::BVHLoader Class Reference

Handles loading BVH animation files.

`#include <BVHLoader.hh>`

Public Member Functions

- **BVHLoader** ()
  Constructor.
- ~**BVHLoader** ()
  Desutrctor.
- **Skeleton** * **Load** (const std::string &_filename, double _scale)
  Load a BVH file.

10.30.1 Detailed Description

Handles loading BVH animation files.

10.30.2 Constructor & Destructor Documentation

10.30.2.1 gazebo::common::BVHLoader::BVHLoader ( )

Constructor.

10.30.2.2 gazebo::common::BVHLoader::~BVHLoader ( )

Desutrctor.

10.30.3 Member Function Documentation

10.30.3.1 Skeleton* gazebo::common::BVHLoader::Load (const std::string & _filename, double _scale)

Load a BVH file.

Parameters

in	*_filename*	BVH file to load
in	*_scale*	Scaling factor to apply to the skeleton

Returns

A pointer to a new **Skeleton** (p. 784)

The documentation for this class was generated from the following file:

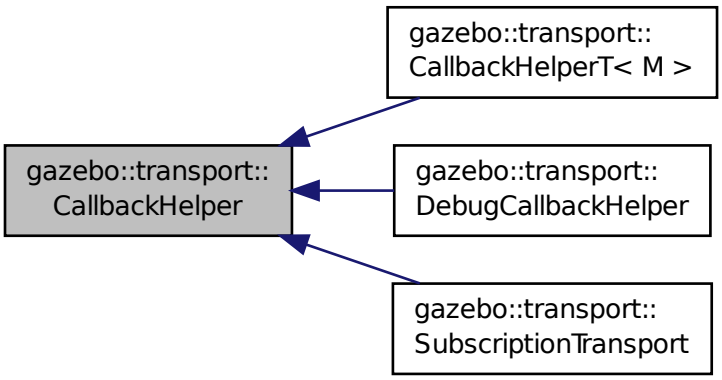
- **BVHLoader.hh**

10.31 gazebo::transport::CallbackHelper Class Reference

A helper class to handle callbacks when messages arrive.

`#include <CallbackHelper.hh>`

Inheritance diagram for gazebo::transport::CallbackHelper:



Public Member Functions

- **CallbackHelper** ()
- virtual ~**CallbackHelper** ()
- bool **GetLatching** () const
- virtual std::string **GetMsgType** () const

  Get the typename of the message that is handled.
- virtual bool **HandleData** (const std::string &newdata)=0
- virtual bool **IsLocal** () const =0

  Return true if the callback is local, false if the callback is tied to a remote connection.

Protected Attributes

- bool **latching**

10.31.1 Detailed Description

A helper class to handle callbacks when messages arrive.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 gazebo::transport::CallbackHelper::CallbackHelper ( ) [inline]

10.31.2.2 virtual gazebo::transport::CallbackHelper::~CallbackHelper ( ) [inline], [virtual]

10.31.3 Member Function Documentation

10.31.3.1 bool gazebo::transport::CallbackHelper::GetLatching ( ) const [inline]

References latching.

10.31.3.2 virtual std::string gazebo::transport::CallbackHelper::GetMsgType ( ) const [inline], [virtual]

Get the typename of the message that is handled.

Reimplemented in **gazebo::transport::DebugCallbackHelper** (p. 312), and **gazebo::transport::CallbackHelperT< M >** (p. 232).

10.31.3.3 virtual bool gazebo::transport::CallbackHelper::HandleData (const std::string & *newdata*) [pure virtual]

Implemented in **gazebo::transport::DebugCallbackHelper** (p. 312), **gazebo::transport::CallbackHelperT< M >** (p. 233), and **gazebo::transport::SubscriptionTransport** (p. 829).

10.31.3.4 virtual bool gazebo::transport::CallbackHelper::IsLocal ( ) const [pure virtual]

Return true if the callback is local, false if the callback is tied to a remote connection.

Implemented in **gazebo::transport::DebugCallbackHelper** (p. 312), **gazebo::transport::CallbackHelperT< M >** (p. 233), and **gazebo::transport::SubscriptionTransport** (p. 830).

10.31.4 Member Data Documentation

10.31.4.1 bool gazebo::transport::CallbackHelper::latching [protected]

Referenced by GetLatching().

The documentation for this class was generated from the following file:

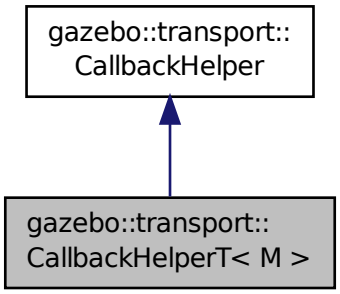
- **CallbackHelper.hh**

10.32 gazebo::transport::CallbackHelperT< M > Class Template Reference

Callback helper Template.

`#include <CallbackHelper.hh>`

Inheritance diagram for gazebo::transport::CallbackHelperT< M >:



Public Member Functions

- **CallbackHelperT** (const boost::function< void(const M const > &)*&cb)
- std::string **GetMsgType** () const

  Get the typename of the message that is handled.
- virtual bool **HandleData** (const std::string &newdata)
- virtual bool **IsLocal** () const

  Return true if the callback is local, false if the callback is tied to a remote connection.

Additional Inherited Members

10.32.1 Detailed Description

template<class M>class gazebo::transport::CallbackHelperT< M >

Callback helper Template.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 template<class M > gazebo::transport::CallbackHelperT< M >::CallbackHelperT (const boost::function< void(const M const > &) [inline]

10.32.3 Member Function Documentation

10.32.3.1 template<class M > std::string gazebo::transport::CallbackHelperT< M >::GetMsgType ( ) const [inline], [virtual]

Get the typename of the message that is handled.

Reimplemented from **gazebo::transport::CallbackHelper** (p. 231).

References gzthrow, and NULL.

10.32.3.2 template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::HandleData (const std::string & *newdata*) [inline], [virtual]

Implements **gazebo::transport::CallbackHelper** (p. 231).

10.32.3.3 template<class M > virtual bool gazebo::transport::CallbackHelperT< M >::IsLocal ( ) const [inline], [virtual]

Return true if the callback is local, false if the callback is tied to a remote connection.

Implements **gazebo::transport::CallbackHelper** (p. 231).

The documentation for this class was generated from the following file:

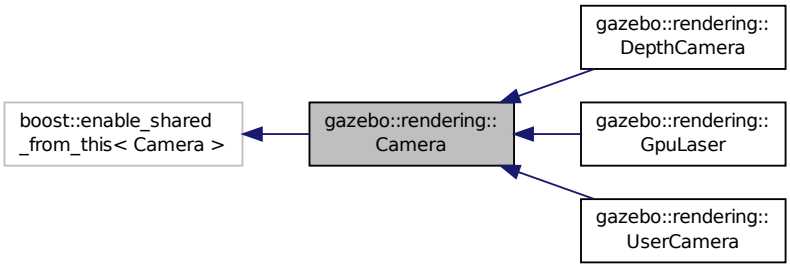
- **CallbackHelper.hh**

10.33 gazebo::rendering::Camera Class Reference

Basic camera sensor.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::Camera:



Public Member Functions

- **Camera** (const std::string &_namePrefix, **Scene** *_scene, bool _autoRender=true)

  Constructor.
- virtual ~**Camera** ()

  Destructor.
- void **AttachToVisual** (const std::string &_visualName, bool _inheritOrientation, double _minDist=0.0, double _maxDist=0.0)

  Attach the camera to a scene node.
- template<typename T >
  event::ConnectionPtr ConnectNewImageFrame (T _subscriber)

  Connect a to the new image signal.
- void **CreateRenderTexture** (const std::string &_textureName)

Set the render target.

- void **DisconnectNewImageFrame** (**event::ConnectionPtr** &_c)

  Disconnect from an image frame.
- void **EnableSaveFrame** (bool _enable)

  Enable or disable saving.
- void **Fini** ()

  Finalize the camera.
- float **GetAspectRatio** () const

  Get the apect ratio.
- virtual float **GetAvgFPS** ()

  Get the average FPS.
- void **GetCameraToViewportRay** (int _screenx, int _screeny, **math::Vector3** &_origin, **math::Vector3** &_dir)

  Get a world space ray as cast from the camera through the viewport.
- **math::Vector3 GetDirection** () const

  Get the camera's direction vector.
- double **GetFarClip** ()

  Get the far clip distance.
- **math::Angle GetHFOV** () const

  Get the camera FOV (horizontal)
- size_t **GetImageByteSize** () const

  Get the image size in bytes.
- virtual const unsigned char * **GetImageData** (unsigned int i=0)

  Get a pointer to the image data.
- unsigned int **GetImageDepth** () const

  Get the depth of the image.
- std::string **GetImageFormat** () const

  Get the height of the image.
- unsigned int **GetImageHeight** () const

  Get the height of the image.
- unsigned int **GetImageWidth** () const

  Get the width of the image.
- bool **GetInitialized** () const

  Returns true if initialized.
- **common::Time GetLastRenderWallTime** ()

  Get the last time the camera was rendered.
- std::string **GetName** () const

  Get the camera's name.
- double **GetNearClip** ()

  Get the near clip distance.
- Ogre::Camera * **GetOgreCamera** () const

  Get a pointer to the ogre camera.
- Ogre::Texture * **GetRenderTexture** () const

  Get the render texture.
- **math::Vector3 GetRight** ()

  Get the viewport right vector.
- **Scene** * **GetScene** () const

  Get the scene this camera is in.

- Ogre::SceneNode * **GetSceneNode** () const

  Get the camera's scene node.
- unsigned int **GetTextureHeight** () const

  Get the height of the off-screen render texture.
- unsigned int **GetTextureWidth** () const

  Get the width of the off-screen render texture.
- virtual unsigned int **GetTriangleCount** ()

  Get the triangle count.
- **math::Vector3 GetUp** ()

  Get the viewport up vector.
- **math::Angle GetVFOV** () const

  Get the camera FOV (vertical)
- Ogre::Viewport * **GetViewport** () const

  Get a pointer to the Ogre::Viewport.
- unsigned int **GetViewportHeight** () const

  Get the viewport height in pixels.
- unsigned int **GetViewportWidth** () const

  Get the viewport width in pixels.
- unsigned int **GetWindowId** () const

  Get the ID of the window this camera is rendering into.
- bool **GetWorldPointOnPlane** (int _x, int _y, const **math::Plane** &_plane, **math::Vector3** &_result)

  Get point on a plane.
- **math::Pose GetWorldPose** ()

  Get the global pose of the camera.
- **math::Vector3 GetWorldPosition** () const

  Get the camera position in the world.
- **math::Quaternion GetWorldRotation** () const

  Get the camera's orientation in the world.
- double **GetZValue** (int _x, int _y)

  Get the Z-buffer value at the given image coordinate.
- void **Init** ()

  Initialize the camera.
- bool **IsInitialized** () const

  Return true if the camera has been initialized.
- bool **IsVisible** (**VisualPtr** _visual)

  Return true if the visual is within the camera's view frustum.
- bool **IsVisible** (const std::string &_visualName)

  Return true if the visual is within the camera's view frustum.
- void **Load** (**sdf::ElementPtr** _sdf)

  Load the camera with a set of parmeters.
- void **Load** ()

  Load the camera with default parmeters.
- virtual bool **MoveToPosition** (const **math::Pose** &_pose, double _time)

  Move the camera to a position.
- bool **MoveToPositions** (const std::vector< **math::Pose** > &_pts, double _time, boost::function< void()> _onComplete=**NULL**)

  Move the camera to a series of positions.

- virtual void **PostRender** ()

  Post render.
- void **Render** ()

  Render the camera.
- void **RotatePitch** (**math::Angle** _angle)

  Rotate the camera around the pitch axis.
- void **RotateYaw** (**math::Angle** _angle)

  Rotate the camera around the yaw axis.
- bool **SaveFrame** (const std::string &_filename)

  Save the last frame to disk.
- void **SetAspectRatio** (float _ratio)

  Set the aspect ratio.
- void **SetCaptureData** (bool _value)

  Set whether to capture data.
- void **SetClipDist** (float _near, float _far)

  Set the clip distances.
- void **SetHFOV** (**math::Angle** _angle)

  Set the camera FOV (horizontal)
- void **SetImageHeight** (unsigned int _h)

  Set the image height.
- void **SetImageSize** (unsigned int _w, unsigned int _h)

  Set the image size.
- void **SetImageWidth** (unsigned int _w)

  Set the image height.
- void **SetName** (const std::string &_name)

  Set the camera's name.
- void **SetRenderRate** (double _hz)

  Set the render Hz rate.
- virtual void **SetRenderTarget** (Ogre::RenderTarget *_target)

  Set the camera's render target.
- void **SetSaveFramePathname** (const std::string &_pathname)

  Set the save frame pathname.
- void **SetScene** (**Scene** *_scene)

  Set the scene this camera is viewing.
- void **SetSceneNode** (Ogre::SceneNode *_node)

  Set the camera's scene node.
- void **SetWindowId** (unsigned int _windowId)
- virtual void **SetWorldPose** (const **math::Pose** &_pose)

  Set the global pose of the camera.
- void **SetWorldPosition** (const **math::Vector3** &_pos)

  Set the world position.
- void **SetWorldRotation** (const **math::Quaternion** &_quat)

  Set the world orientation.
- void **ShowWireframe** (bool _s)

  Set whether to view the world in wireframe.
- void **ToggleShowWireframe** ()

  Toggle whether to view the world in wireframe.

- void **TrackVisual** (const std::string &_visualName)

  Set the camera to track a scene node.
- void **Translate** (const **math::Vector3** &_direction)

  Translate the camera.
- virtual void **Update** ()

Static Public Member Functions

- static size_t **GetImageByteSize** (unsigned int _width, unsigned int _height, const std::string &_format)

  Calculate image byte size base on a few parameters.
- static bool **SaveFrame** (const unsigned char *_image, unsigned int _width, unsigned int _height, int _depth, const std::string &_format, const std::string &_filename)

  Save a frame using an image buffer.

Protected Member Functions

- virtual bool **AttachToVisualImpl** (const std::string &_name, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  Attach the camera to a scene node.
- virtual bool **AttachToVisualImpl** (**VisualPtr** _visual, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  Attach the camera to a visual.
- std::string **GetFrameFilename** ()

  Get the next frame filename based on SDF parameters.
- virtual void **RenderImpl** ()

  Implementation of the render call.
- bool **TrackVisualImpl** (const std::string &_visualName)

  *Implementation of the **Camera::TrackVisual*** (p. 253) *call.*
- virtual bool **TrackVisualImpl** (**VisualPtr** _visual)

  Set the camera to track a scene node.

Protected Attributes

- Ogre::AnimationState * **animState**
- unsigned char * **bayerFrameBuffer**
- Ogre::Camera * **camera**
- bool **captureData**
- std::vector< **event::ConnectionPtr** > **connections**
- int **imageFormat**
- int **imageHeight**
- int **imageWidth**
- bool **initialized**
- **common::Time lastRenderWallTime**
- std::string **name**
- bool **newData**
- **event::EventT**< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)> **newImageFrame**

- boost::function< void() > **onAnimationComplete**
- Ogre::SceneNode * **pitchNode**
- **common::Time prevAnimTime**
- Ogre::RenderTarget * **renderTarget**
- Ogre::Texture * **renderTexture**
- std::list< msgs::Request > **requests**
- unsigned int **saveCount**
- unsigned char * **saveFrameBuffer**
- **Scene** * **scene**
- Ogre::SceneNode * **sceneNode**
- **sdf::ElementPtr sdf**
- unsigned int **textureHeight**
- unsigned int **textureWidth**
- Ogre::Viewport * **viewport**
- unsigned int **windowId**

10.33.1 Detailed Description

Basic camera sensor.

This is the base class for all cameras.

10.33.2 Constructor & Destructor Documentation

10.33.2.1 gazebo::rendering::Camera::Camera (const std::string & *_namePrefix,* Scene * *_scene,* bool *_autoRender* = `true`)

Constructor.

Parameters

in	*_namePrefix*	Unique prefix name for the camera.
in	*_scene*	**Scene** (p. 746) that will contain the camera
in	*_autoRender*	Almost everyone should leave this as true.

10.33.2.2 virtual gazebo::rendering::Camera::~Camera ( ) [virtual]

Destructor.

10.33.3 Member Function Documentation

10.33.3.1 void gazebo::rendering::Camera::AttachToVisual (const std::string & *_visualName,* bool *_inheritOrientation,* double *_minDist* = `0.0`, double *_maxDist* = `0.0`)

Attach the camera to a scene node.

Parameters

in	*_visualName*	Name of the visual to attach the camera to
in	*_inherit-Orientation*	True means camera acquires the visual's orientation

in	*_minDist*	Minimum distance the camera is allowed to get to the visual
in	*_maxDist*	Maximum distance the camera is allowd to get from the visual

10.33.3.2 virtual bool gazebo::rendering::Camera::AttachToVisualImpl (const std::string & *_name,* bool *_inheritOrientation,* double *_minDist* = 0, double *_maxDist* = 0) [protected], [virtual]

Attach the camera to a scene node.

Parameters

in	*_visualName*	Name of the visual to attach the camera to
in	*_inherit-Orientation*	True means camera acquires the visual's orientation
in	*_minDist*	Minimum distance the camera is allowed to get to the visual
in	*_maxDist*	Maximum distance the camera is allowd to get from the visual

Returns

True on success

10.33.3.3 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. 880).

10.33.3.4 template<typename T > event::ConnectionPtr gazebo::rendering::Camera::ConnectNewImageFrame (T *_subscriber*) [inline]

Connect a to the new image signal.

Parameters

in	*_subscriber*	Callback that is called when a new image is generated

Returns

A pointer to the connection. This must be kept in scope.

References gazebo::event::EventT< T >::Connect(), and newImageFrame.

10.33.3.5 void gazebo::rendering::Camera::CreateRenderTexture (const std::string & *_textureName*)

Set the render target.

Parameters

in	*_textureName*	Name of the new render texture

10.33.3.6 void gazebo::rendering::Camera::DisconnectNewImageFrame (event::ConnectionPtr & *_c*) [inline]

Disconnect from an image frame.

Parameters

in	*_c*	The connection to disconnect

References gazebo::event::EventT< T >::Disconnect(), and newImageFrame.

10.33.3.7 void gazebo::rendering::Camera::EnableSaveFrame (bool *_enable*)

Enable or disable saving.

Parameters

in	*_enable*	Set to True to enable saving of frames

10.33.3.8 void gazebo::rendering::Camera::Fini ( )

Finalize the camera.

This function is called before the camera is destructed

10.33.3.9 float gazebo::rendering::Camera::GetAspectRatio ( ) const

Get the apect ratio.

Returns

The aspect ratio (width / height) in pixels

10.33.3.10 virtual float gazebo::rendering::Camera::GetAvgFPS ( ) [inline], [virtual]

Get the average FPS.

Returns

The average frames per second

10.33.3.11 void gazebo::rendering::Camera::GetCameraToViewportRay (int *_screenx,* int *_screeny,* math::Vector3 & *_origin,* math::Vector3 & *_dir*)

Get a world space ray as cast from the camera through the viewport.

Parameters

in	*_screenx*	X coordinate in the camera's viewport, in pixels.
in	*_screeny*	Y coordinate in the camera's viewport, in pixels.
out	*_origin*	Origing in the world coordinate frame of the resulting ray
out	*_dir*	Direction of the resulting ray

10.33.3.12 math::Vector3 gazebo::rendering::Camera::GetDirection ( ) const

Get the camera's direction vector.

Returns

Direction the camera is facing

10.33.3.13 double gazebo::rendering::Camera::GetFarClip ( )

Get the far clip distance.

Returns

Far clip distance

10.33.3.14 std::string gazebo::rendering::Camera::GetFrameFilename ( ) [protected]

Get the next frame filename based on SDF parameters.

Returns

The frame's filename

10.33.3.15 math::Angle gazebo::rendering::Camera::GetHFOV ( ) const

Get the camera FOV (horizontal)

Returns

The horizontal field of view

10.33.3.16 size_t gazebo::rendering::Camera::GetImageByteSize ( ) const

Get the image size in bytes.

Returns

Size in bytes

10.33.3.17 static size_t gazebo::rendering::Camera::GetImageByteSize (unsigned int *_width*, unsigned int *_height*, const std::string & *_format*) [static]

Calculate image byte size base on a few parameters.

Parameters

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

10.33.3.18 virtual const unsigned char* gazebo::rendering::Camera::GetImageData (unsigned int *i* = 0) [virtual]

Get a pointer to the image data.

Get the raw image data from a camera's buffer.

Parameters

in	*_i*	Index of the camera's texture (0 = RGB, 1 = depth).

Returns

Pointer to the raw data, null if data is not available.

10.33.3.19 unsigned int gazebo::rendering::Camera::GetImageDepth ( ) const

Get the depth of the image.

Returns

Depth of the image

10.33.3.20 std::string gazebo::rendering::Camera::GetImageFormat ( ) const

Get the height of the image.

Returns

String representation of the image format.

10.33.3.21 unsigned int gazebo::rendering::Camera::GetImageHeight ( ) const

Get the height of the image.

Returns

Image height

10.33.3.22 unsigned int gazebo::rendering::Camera::GetImageWidth ( ) const

Get the width of the image.

Returns

Image width

10.33.3.23 bool gazebo::rendering::Camera::GetInitialized ( ) const

Returns true if initialized.

Returns

Ture if the camera is initialized

10.33.3.24 common::Time gazebo::rendering::Camera::GetLastRenderWallTime ( )

Get the last time the camera was rendered.

Returns

Time the camera was last rendered

10.33.3.25 std::string gazebo::rendering::Camera::GetName ( ) const

Get the camera's name.

Returns

The name of the camera

10.33.3.26 double gazebo::rendering::Camera::GetNearClip ( )

Get the near clip distance.

Returns

Near clip distance

10.33.3.27 Ogre::Camera* gazebo::rendering::Camera::GetOgreCamera ( ) const

Get a pointer to the ogre camera.

Returns

Pointer to the OGRE camera

10.33.3.28 Ogre::Texture* gazebo::rendering::Camera::GetRenderTexture ( ) const

Get the render texture.

Returns

Pointer to the render texture

10.33.3.29 math::Vector3 gazebo::rendering::Camera::GetRight ( )

Get the viewport right vector.

Returns

The viewport right vector

10.33.3.30 Scene* gazebo::rendering::Camera::GetScene ( ) const

Get the scene this camera is in.

Returns

Pointer to scene containing this camera

10.33.3.31 Ogre::SceneNode* gazebo::rendering::Camera::GetSceneNode ( ) const

Get the camera's scene node.

Returns

The scene node the camera is attached to

10.33.3.32 unsigned int gazebo::rendering::Camera::GetTextureHeight ( ) const

Get the height of the off-screen render texture.

Returns

Render texture height

10.33.3.33 unsigned int gazebo::rendering::Camera::GetTextureWidth ( ) const

Get the width of the off-screen render texture.

Returns

Render texture width

10.33.3.34 virtual unsigned int gazebo::rendering::Camera::GetTriangleCount ( ) [inline], [virtual]

Get the triangle count.

Returns

The current triangle count

10.33.3.35 math::Vector3 gazebo::rendering::Camera::GetUp ( )

Get the viewport up vector.

Returns

The viewport up vector

10.33.3.36 math::Angle gazebo::rendering::Camera::GetVFOV ( ) const

Get the camera FOV (vertical)

Returns

The vertical field of view

10.33.3.37 Ogre::Viewport* gazebo::rendering::Camera::GetViewport ( ) const

Get a pointer to the Ogre::Viewport.

Returns

Pointer to the Ogre::Viewport

10.33.3.38 unsigned int gazebo::rendering::Camera::GetViewportHeight ( ) const

Get the viewport height in pixels.

Returns

The viewport height

10.33.3.39 unsigned int gazebo::rendering::Camera::GetViewportWidth ( ) const

Get the viewport width in pixels.

Returns

The viewport width

10.33.3.40 unsigned int gazebo::rendering::Camera::GetWindowId ( ) const

Get the ID of the window this camera is rendering into.

Returns

The ID of the window.

10.33.3.41 bool gazebo::rendering::Camera::GetWorldPointOnPlane (int *_x*, int *_y*, const math::Plane & *_plane*, math::Vector3 & *_result*)

Get point on a plane.

Parameters

in	*_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

10.33.3.42 math::Pose gazebo::rendering::Camera::GetWorldPose ( )

Get the global pose of the camera.

Returns

Pose of the camera in the world coordinate frame

10.33.3.43 math::Vector3 gazebo::rendering::Camera::GetWorldPosition ( ) const

Get the camera position in the world.

Returns

The world position of the camera

10.33.3.44 math::Quaternion gazebo::rendering::Camera::GetWorldRotation ( ) const

Get the camera's orientation in the world.

Returns

The camera's orientation as a **math::Quaternion** (p. 697)

10.33.3.45 double gazebo::rendering::Camera::GetZValue (int _x, int _y)

Get the Z-buffer value at the given image coordinate.

Parameters

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.

10.33.3.46 void gazebo::rendering::Camera::Init ( )

Initialize the camera.

10.33.3.47 bool gazebo::rendering::Camera::IsInitialized ( ) const [inline]

Return true if the camera has been initialized.

Returns

True if initialized was successful

References initialized.

10.33.3.48 bool gazebo::rendering::Camera::IsVisible (VisualPtr _visual)

Return true if the visual is within the camera's view frustum.

Parameters

in	*_visual*	The visual to check for visibility

Returns

True if the _visual is in the camera's frustum

10.33.3.49 bool gazebo::rendering::Camera::IsVisible (const std::string & _visualName)

Return true if the visual is within the camera's view frustum.

Parameters

in	*_visualName*	Name of the visual to check for visibility

Returns

True if the _visual is in the camera's frustum

10.33.3.50 void gazebo::rendering::Camera::Load (sdf::ElementPtr *_sdf*)

Load the camera with a set of parmeters.

Parameters

in	*_sdf*	The SDF camera info

10.33.3.51 void gazebo::rendering::Camera::Load ( )

Load the camera with default parmeters.

10.33.3.52 virtual bool gazebo::rendering::Camera::MoveToPosition (const math::Pose & *_pose,* double *_time*) [virtual]

Move the camera to a position.

This is an animated motion

Parameters

in	*_pose*	End position of the camera
in	*_time*	Duration of the camera's movement

Reimplemented in **gazebo::rendering::UserCamera** (p. 883).

10.33.3.53 bool gazebo::rendering::Camera::MoveToPositions (const std::vector< math::Pose > & *_pts,* double *_time,* boost::function< void() > *_onComplete* = NULL)

Move the camera to a series of positions.

Parameters

in	*_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

10.33.3.54 virtual void gazebo::rendering::Camera::PostRender ( ) [virtual]

Post render.

Called afer the render signal.

Reimplemented in **gazebo::rendering::GpuLaser** (p. 378), **gazebo::rendering::DepthCamera** (p. 316), and **gazebo-**

::rendering::UserCamera (p. 884).

10.33.3.55 void gazebo::rendering::Camera::Render ( )

Render the camera.

Called after the pre-render signal. This function will generate camera images

10.33.3.56 virtual void gazebo::rendering::Camera::RenderImpl ( ) [protected], [virtual]

Implementation of the render call.

10.33.3.57 void gazebo::rendering::Camera::RotatePitch (math::Angle *_angle*)

Rotate the camera around the pitch axis.

Parameters

in	*_angle*	Pitch amount

10.33.3.58 void gazebo::rendering::Camera::RotateYaw (math::Angle *_angle*)

Rotate the camera around the yaw axis.

Parameters

in	*_angle*	Rotation amount

10.33.3.59 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

10.33.3.60 static bool gazebo::rendering::Camera::SaveFrame (const unsigned char * *_image*, unsigned int *_width*, unsigned int *_height*, int *_depth*, const std::string & *_format*, const std::string & *_filename*) [static]

Save a frame using an image buffer.

Parameters

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

10.33.3.61 void gazebo::rendering::Camera::SetAspectRatio (float *_ratio*)

Set the aspect ratio.

Parameters

in	*_ratio*	The aspect ratio (width / height) in pixels

10.33.3.62 void gazebo::rendering::Camera::SetCaptureData (bool *_value*)

Set whether to capture data.

Parameters

in	*_value*	Set to true to capture data into a memory buffer.

10.33.3.63 void gazebo::rendering::Camera::SetClipDist (float *_near,* float *_far*)

Set the clip distances.

Parameters

in	*_near*	Near clip distance in meters
in	*_far*	Far clip distance in meters

10.33.3.64 void gazebo::rendering::Camera::SetHFOV (math::Angle *_angle*)

Set the camera FOV (horizontal)

Parameters

in	*_radians*	Horizontal field of view

10.33.3.65 void gazebo::rendering::Camera::SetImageHeight (unsigned int *_h*)

Set the image height.

Parameters

in	*_h*	Image height

10.33.3.66 void gazebo::rendering::Camera::SetImageSize (unsigned int *_w*, unsigned int *_h*)

Set the image size.

Parameters

in	*_w*	Image width
in	*_h*	Image height

10.33.3.67 void gazebo::rendering::Camera::SetImageWidth (unsigned int *_w*)

Set the image height.

Parameters

in	*_w*	Image width

10.33.3.68 void gazebo::rendering::Camera::SetName (const std::string & *_name*)

Set the camera's name.

Parameters

in	*_name*	New name for the camera

10.33.3.69 void gazebo::rendering::Camera::SetRenderRate (double *_hz*)

Set the render Hz rate.

Parameters

in	*_hz*	The Hz rate

10.33.3.70 virtual void gazebo::rendering::Camera::SetRenderTarget (Ogre::RenderTarget * *_target*) [virtual]

Set the camera's render target.

Parameters

in	*_target*	Pointer to the render target

Reimplemented in **gazebo::rendering::UserCamera** (p. 884).

10.33.3.71 void gazebo::rendering::Camera::SetSaveFramePathname (const std::string & *_pathname*)

Set the save frame pathname.

Parameters

in	*_pathname*	Directory in which to store saved image frames

10.33.3.72 void gazebo::rendering::Camera::SetScene (Scene * *_scene*)

Set the scene this camera is viewing.

Parameters

in	*_scene*	Pointer to the scene

10.33.3.73 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

10.33.3.74 void gazebo::rendering::Camera::SetWindowId (unsigned int *_windowId*)

10.33.3.75 virtual void gazebo::rendering::Camera::SetWorldPose (const math::Pose & *_pose*) [virtual]

Set the global pose of the camera.

Parameters

in	*_pose*	The new **math::Pose** (p. 677) of the camera

Reimplemented in **gazebo::rendering::UserCamera** (p. 885).

10.33.3.76 void gazebo::rendering::Camera::SetWorldPosition (const math::Vector3 & *_pos*)

Set the world position.

Parameters

in	*_pos*	The new position of the camera

10.33.3.77 void gazebo::rendering::Camera::SetWorldRotation (const math::Quaternion & *_quat*)

Set the world orientation.

Parameters

in	*_quat*	The new orientation of the camera

10.33.3.78 void gazebo::rendering::Camera::ShowWireframe (bool *_s*)

Set whether to view the world in wireframe.

Parameters

in	*_s*	Set to True to render objects as wireframe

10.33.3.79 void gazebo::rendering::Camera::ToggleShowWireframe ( )

Toggle whether to view the world in wireframe.

10.33.3.80 void gazebo::rendering::Camera::TrackVisual (const std::string & *_visualName*)

Set the camera to track a scene node.

Parameters

in	*_visualName*	Name of the visual to track

10.33.3.81 bool gazebo::rendering::Camera::TrackVisualImpl (const std::string & *_visualName*) [protected]

Implementation of the **Camera::TrackVisual** (p. 253) call.

Parameters

in	*_visualName*	Name of the visual to track

Returns

True if able to track the visual

10.33.3.82 virtual bool gazebo::rendering::Camera::TrackVisualImpl (VisualPtr *_visual*) [protected], [virtual]

Set the camera to track a scene node.

Parameters

in	*_visual*	The visual to track

Returns

True if able to track the visual

Reimplemented in **gazebo::rendering::UserCamera** (p. 885).

10.33.3.83 void gazebo::rendering::Camera::Translate (const math::Vector3 & _direction)

Translate the camera.

Parameters

in	*_direction*	The translation vector

10.33.3.84 virtual void gazebo::rendering::Camera::Update ( ) [virtual]

Reimplemented in **gazebo::rendering::UserCamera** (p. 885).

10.33.4 Member Data Documentation

10.33.4.1 Ogre::AnimationState* gazebo::rendering::Camera::animState [protected]

10.33.4.2 unsigned char* gazebo::rendering::Camera::bayerFrameBuffer [protected]

10.33.4.3 Ogre::Camera* gazebo::rendering::Camera::camera [protected]

10.33.4.4 bool gazebo::rendering::Camera::captureData [protected]

10.33.4.5 std::vector<event::ConnectionPtr> gazebo::rendering::Camera::connections [protected]

10.33.4.6 int gazebo::rendering::Camera::imageFormat [protected]

10.33.4.7 int gazebo::rendering::Camera::imageHeight [protected]

10.33.4.8 int gazebo::rendering::Camera::imageWidth [protected]

10.33.4.9 bool gazebo::rendering::Camera::initialized [protected]

Referenced by IsInitialized().

10.33.4.10 common::Time gazebo::rendering::Camera::lastRenderWallTime [protected]

10.33.4.11 std::string gazebo::rendering::Camera::name [protected]

10.33.4.12 bool gazebo::rendering::Camera::newData [protected]

10.33.4.13 event::EventT<void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)> gazebo::rendering::Camera::newImageFrame [protected]

Referenced by ConnectNewImageFrame(), and DisconnectNewImageFrame().

10.33.4.14 boost::function<void()> gazebo::rendering::Camera::onAnimationComplete [protected]

10.33.4.15 Ogre::SceneNode* gazebo::rendering::Camera::pitchNode [protected]

10.33.4.16 common::Time gazebo::rendering::Camera::prevAnimTime [protected]

10.33.4.17 Ogre::RenderTarget* gazebo::rendering::Camera::renderTarget [protected]

10.33.4.18 Ogre::Texture* gazebo::rendering::Camera::renderTexture [protected]

10.33.4.19 std::list<msgs::Request> gazebo::rendering::Camera::requests [protected]

10.33.4.20 unsigned int gazebo::rendering::Camera::saveCount [protected]

10.33.4.21 unsigned char* gazebo::rendering::Camera::saveFrameBuffer [protected]

10.33.4.22 Scene* gazebo::rendering::Camera::scene [protected]

10.33.4.23 Ogre::SceneNode* gazebo::rendering::Camera::sceneNode [protected]

10.33.4.24 sdf::ElementPtr gazebo::rendering::Camera::sdf [protected]

10.33.4.25 unsigned int gazebo::rendering::Camera::textureHeight [protected]

10.33.4.26 unsigned int gazebo::rendering::Camera::textureWidth [protected]

10.33.4.27 Ogre::Viewport* gazebo::rendering::Camera::viewport [protected]

10.33.4.28 unsigned int gazebo::rendering::Camera::windowId [protected]

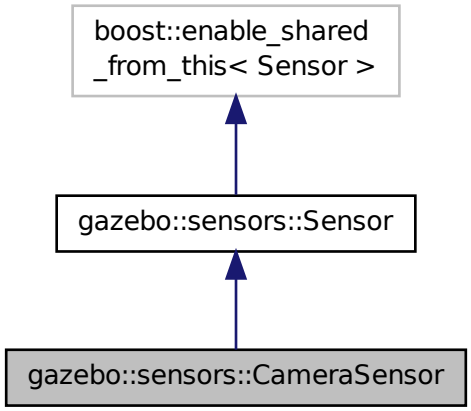
The documentation for this class was generated from the following file:

- **Camera.hh**

10.34 gazebo::sensors::CameraSensor Class Reference

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::CameraSensor:



Public Member Functions

- **CameraSensor** ()
  Constructor.
- virtual ~**CameraSensor** ()
  Destructor.
- **rendering::CameraPtr GetCamera** () const
  *Returns a pointer to the **rendering::Camera** (p. 233).*
- const unsigned char * **GetImageData** ()
  Gets the raw image data from the sensor.
- unsigned int **GetImageHeight** () const
  Gets the height of the image in pixels.
- unsigned int **GetImageWidth** () const
  Gets the width of the image in pixels.
- virtual std::string **GetTopic** () const
  Gets the topic name of the sensor.
- virtual void **Init** ()
  Initialize the camera.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** _sdf)
  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
  Load the sensor with default parameters.
- bool **SaveFrame** (const std::string &_filename)
  Saves the image to the disk.
- virtual void **SetActive** (bool _value)
  Set whether the sensor is active or not.
- virtual void **SetParent** (const std::string &_name)
  Set the parent of the sensor.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the camera.
- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.

Additional Inherited Members

10.34.1 Constructor & Destructor Documentation

10.34.1.1 gazebo::sensors::CameraSensor::CameraSensor ( )

Constructor.

10.34.1.2 virtual gazebo::sensors::CameraSensor::~CameraSensor ( ) [virtual]

Destructor.

10.34.2 Member Function Documentation

10.34.2.1 virtual void gazebo::sensors::CameraSensor::Fini ( ) [protected], [virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

10.34.2.2 rendering::CameraPtr gazebo::sensors::CameraSensor::GetCamera ( ) const [inline]

Returns a pointer to the **rendering::Camera** (p. 233).

Returns

The Pointer to the camera sensor : nate check

10.34.2.3 const unsigned char* gazebo::sensors::CameraSensor::GetImageData ( )

Gets the raw image data from the sensor.

Returns

The pointer to the data array : nate check

10.34.2.4 unsigned int gazebo::sensors::CameraSensor::GetImageHeight ( ) const

Gets the height of the image in pixels.

Returns

The height in pixels of the image : nate check

10.34.2.5 unsigned int gazebo::sensors::CameraSensor::GetImageWidth ( ) const

Gets the width of the image in pixels.

Returns

The width in pixels of the image : nate check

10.34.2.6 virtual std::string gazebo::sensors::CameraSensor::GetTopic ( ) const [virtual]

Gets the topic name of the sensor.

Returns

Topic name

Todo to be implemented

Reimplemented from **gazebo::sensors::Sensor** (p. 769).

10.34.2.7 virtual void gazebo::sensors::CameraSensor::Init ( ) [virtual]

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.34.2.8 virtual void gazebo::sensors::CameraSensor::Load (const std::string & *_worldName,* **sdf::ElementPtr** *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 765) parameters
in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.34.2.9 virtual void gazebo::sensors::CameraSensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.34.2.10 bool gazebo::sensors::CameraSensor::SaveFrame (const std::string & *_filename*)

Saves the image to the disk.

Parameters

in	*&_filename*	The name of the file to be saved

Returns

True if successful, false if unsuccessful : nate check

10.34.2.11 virtual void gazebo::sensors::CameraSensor::SetActive (bool *_value*) [virtual]

Set whether the sensor is active or not.

Parameters

in	*_value*	True if active, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.34.2.12 virtual void gazebo::sensors::CameraSensor::SetParent (const std::string & *_name*) [virtual]

Set the parent of the sensor.

Parameters

_name	The name of the parent

Reimplemented from **gazebo::sensors::Sensor** (p. 771).

10.34.2.13 virtual void gazebo::sensors::CameraSensor::UpdateImpl (bool *_force*) [protected], [virtual]

Update the sensor information.

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 771).

The documentation for this class was generated from the following file:

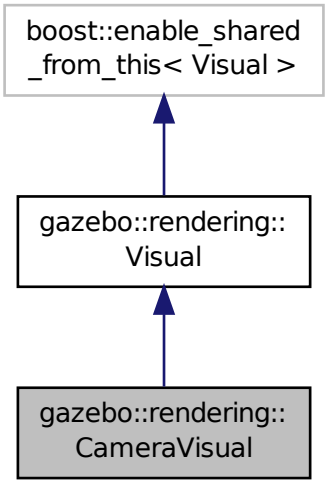
- **CameraSensor.hh**

10.35 gazebo::rendering::CameraVisual Class Reference

Basic camera visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::CameraVisual:



Public Member Functions

- **CameraVisual** (const std::string &_name, **VisualPtr** _vis)

  Constructor.
- virtual ~**CameraVisual** ()

  Destructor.
- void **Load** (unsigned int _width, unsigned int _height)

  *Load the **Visual** (p. 931).*

Additional Inherited Members

10.35.1 Detailed Description

Basic camera visualization.

This class is used to visualize a camera image generated from a CameraSensor. The sensor's image is drawn on a billboard in the 3D environment.

10.35.2 Constructor & Destructor Documentation

10.35.2.1 gazebo::rendering::CameraVisual::CameraVisual (const std::string & *_name,* VisualPtr *_vis*)

Constructor.

Parameters

in	*_name*	Name of the **Visual** (p. 931)
in	*_vis*	Pointer to the parent **Visual** (p. 931)

10.35.2.2 virtual gazebo::rendering::CameraVisual::~CameraVisual ( ) [virtual]

Destructor.

10.35.3 Member Function Documentation

10.35.3.1 void gazebo::rendering::CameraVisual::Load (unsigned int *_width,* unsigned int *_height*)

Load the **Visual** (p. 931).

Parameters

in	*_width*	Width of the **Camera** (p. 233) image
in	*_height*	Height of the **Camera** (p. 233) image

The documentation for this class was generated from the following file:

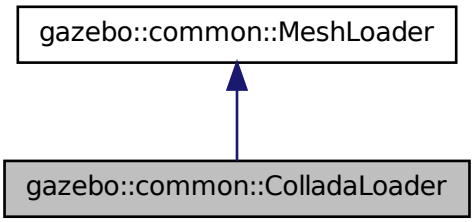
- **CameraVisual.hh**

10.36 gazebo::common::ColladaLoader Class Reference

Class used to load Collada mesh files.

`#include <ColladaLoader.hh>`

Inheritance diagram for gazebo::common::ColladaLoader:



Public Member Functions

- **ColladaLoader** ()

Constructor.

- virtual ∼**ColladaLoader** ()

  Destructor.

- virtual **Mesh** ∗ **Load** (const std::string &_filename)

  Load a mesh.

10.36.1 Detailed Description

Class used to load Collada mesh files.

10.36.2 Constructor & Destructor Documentation

10.36.2.1 gazebo::common::ColladaLoader::ColladaLoader ( )

Constructor.

10.36.2.2 virtual gazebo::common::ColladaLoader::∼ColladaLoader ( ) [virtual]

Destructor.

10.36.3 Member Function Documentation

10.36.3.1 virtual Mesh∗ gazebo::common::ColladaLoader::Load (const std::string & *_filename*) [virtual]

Load a mesh.

Parameters

in	*_filename*	Collada file to load

Returns

Pointer to a new **Mesh** (p. 509)

Implements **gazebo::common::MeshLoader** (p. 515).

The documentation for this class was generated from the following file:

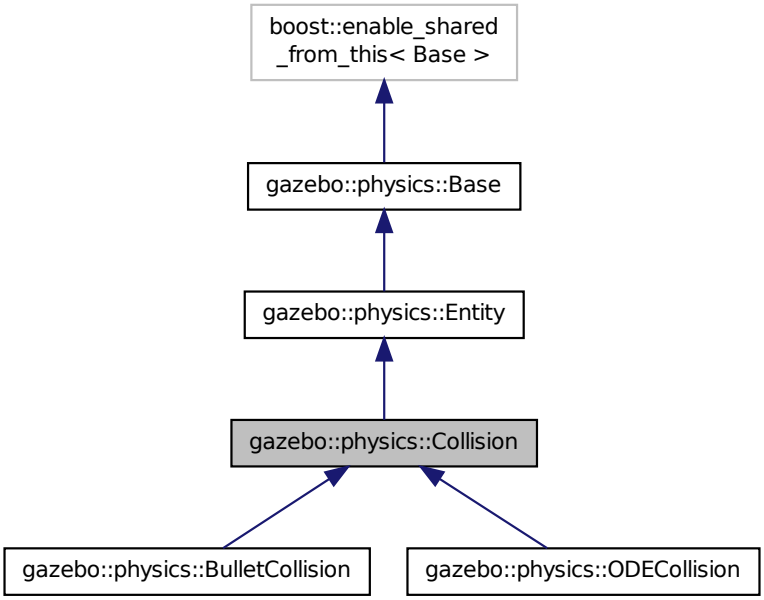
- **ColladaLoader.hh**

10.37 gazebo::physics::Collision Class Reference

Base (p. 145) class for all collision entities.

`#include <Collision.hh>`

Inheritance diagram for gazebo::physics::Collision:



Public Member Functions

- **Collision** (**LinkPtr** _link)

  Constructor.
- virtual ~**Collision** ()

  Destructor.
- void **AddContact** (const **Contact** &_contact)

  Add an occurance of a contact to this collision.
- template<typename T >
  event::ConnectionPtr ConnectContact (T _subscriber)

  Setup callback for contact event.
- void **DisconnectContact** (**event::ConnectionPtr** &_conn)

  Disconnect callback for contact event.
- void **FillCollisionMsg** (msgs::Collision &_msg) **GAZEBO_DEPRECATED**

  DEPRECATED.
- void **FillMsg** (msgs::Collision &_msg)

  Fill a collision message.
- void **Fini** ()

  Finalize the collision.
- virtual **math::Box GetBoundingBox** () const =0

  Get the bounding box for this collision.
- bool **GetContactsEnabled** () const

Return true of contacts are on.

- float **GetLaserRetro** () const

  Get the laser retro reflectiveness.

- **LinkPtr GetLink** () const

  Get the link this collision belongs to.

- **ModelPtr GetModel** () const

  Get the model this collision belongs to.

- virtual **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the collision.

- virtual **math::Vector3 GetRelativeAngularVel** () const

  Get the angular velocity of the collision.

- virtual **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the collision.

- virtual **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the collision.

- **ShapePtr GetShape** () const

  Get the collision shape.

- unsigned int **GetShapeType** ()

  Get the shape type.

- **CollisionState GetState** ()

  Get the collision state.

- **SurfaceParamsPtr GetSurface** () const

  Get the surface parameters.

- virtual **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the collision in the world frame.

- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the collision in the world frame.

- virtual **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the collision in the world frame.

- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the collision in the world frame.

- virtual void **Init** ()

  Initialize the collision.

- bool **IsPlaceable** () const

  Return whether this collision is movable.

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the collision.

- void **ProcessMsg** (const msgs::Collision &_msg)

  Update parameters from a message.

- virtual void **SetCategoryBits** (unsigned int _bits)=0

  Set the category bits, used during collision detection.

- virtual void **SetCollideBits** (unsigned int _bits)=0

  Set the collide bits, used during collision detection.

- void **SetCollision** (bool _placeable)

  Set the encapsulated collsion object.

- void **SetContactsEnabled** (bool _enable)

  Turn contact recording on or off.

- void **SetLaserRetro** (float _retro)

  Set the laser retro reflectiveness.
- void **SetShape** (**ShapePtr** _shape)

  Set the shape for this collision.
- void **SetState** (const **CollisionState** &_state)

  Set the current collision state.
- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  Update the parameters using new sdf values.

Protected Attributes

- **LinkPtr link**

  The link this collision belongs to.
- bool **placeable**

  Flag for placeable.
- **ShapePtr shape**

  *Pointer to **physics::Shape** (p. 779).*

Additional Inherited Members

10.37.1 Detailed Description

Base (p. 145) class for all collision entities.

10.37.2 Constructor & Destructor Documentation

10.37.2.1 gazebo::physics::Collision::Collision (LinkPtr *_link*) [explicit]

Constructor.

Parameters

in	*_link*	**Link** (p. 454) that contains this collision object.

10.37.2.2 virtual gazebo::physics::Collision::~Collision ( ) [virtual]

Destructor.

10.37.3 Member Function Documentation

10.37.3.1 void gazebo::physics::Collision::AddContact (const Contact & *_contact*)

Add an occurance of a contact to this collision.

Parameters

in	*_contact*	The contact which was detected by a collision engine.

10.37.3.2 template<typename T > event::ConnectionPtr gazebo::physics::Collision::ConnectContact (T _subscriber) [inline]

Setup callback for contact event.

Parameters

in	*_subscriber*	The callback.

Returns

Pointer to the new connection, which must be kept in scope.

References gazebo::event::EventT< T >::Connect().

10.37.3.3 void gazebo::physics::Collision::DisconnectContact (event::ConnectionPtr & _conn) [inline]

Disconnect callback for contact event.

Parameters

in	*_conn*	The connection to disconnect.

References gazebo::event::EventT< T >::Disconnect().

10.37.3.4 void gazebo::physics::Collision::FillCollisionMsg (msgs::Collision & _msg)

DEPRECATED.

10.37.3.5 void gazebo::physics::Collision::FillMsg (msgs::Collision & _msg)

Fill a collision message.

Parameters

out	*_msg*	The message to fill with this collision's data.

10.37.3.6 void gazebo::physics::Collision::Fini ( ) [virtual]

Finalize the collision.

Reimplemented from **gazebo::physics::Entity** (p. 342).

Reimplemented in **gazebo::physics::ODECollision** (p. 580).

10.37.3.7 virtual math::Box gazebo::physics::Collision::GetBoundingBox ( ) const [pure virtual]

Get the bounding box for this collision.

Returns

The bounding box.

Reimplemented from **gazebo::physics::Entity** (p. 342).

Implemented in **gazebo::physics::ODECollision** (p. 580), and **gazebo::physics::BulletCollision** (p. 173).

10.37.3.8 bool gazebo::physics::Collision::GetContactsEnabled ( ) const

Return true of contacts are on.

Returns

True of contact are on.

10.37.3.9 float gazebo::physics::Collision::GetLaserRetro ( ) const

Get the laser retro reflectiveness.

Returns

The laser retro value.

10.37.3.10 LinkPtr gazebo::physics::Collision::GetLink ( ) const

Get the link this collision belongs to.

Returns

The parent **Link** (p. 454).

10.37.3.11 ModelPtr gazebo::physics::Collision::GetModel ( ) const

Get the model this collision belongs to.

Returns

The parent model.

10.37.3.12 virtual math::Vector3 gazebo::physics::Collision::GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the collision.

Returns

The angular acceleration of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 343).

10.37.3.13 virtual math::Vector3 gazebo::physics::Collision::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the collision.

Returns

The angular velocity of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 343).

10.37.3.14 virtual math::Vector3 gazebo::physics::Collision::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the collision.

Returns

The linear acceleration of the collision.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.37.3.15 virtual math::Vector3 gazebo::physics::Collision::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the collision.

Returns

The linear velocity relative to the parent model.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.37.3.16 ShapePtr gazebo::physics::Collision::GetShape ( ) const

Get the collision shape.

Returns

The collision shape.

10.37.3.17 unsigned int gazebo::physics::Collision::GetShapeType ( )

Get the shape type.

Returns

The shape type.

See Also

EntityType (p. 149)

10.37.3.18 CollisionState gazebo::physics::Collision::GetState ( )

Get the collision state.

Returns

The collision state.

10.37.3.19 SurfaceParamsPtr gazebo::physics::Collision::GetSurface ( ) const [inline]

Get the surface parameters.

Returns

The surface parameters.

10.37.3.20 virtual math::Vector3 gazebo::physics::Collision::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the collision in the world frame.

Returns

The angular acceleration of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.37.3.21 virtual math::Vector3 gazebo::physics::Collision::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the collision in the world frame.

Returns

The angular velocity of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.37.3.22 virtual math::Vector3 gazebo::physics::Collision::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the collision in the world frame.

Returns

The linear acceleration of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 345).

10.37.3.23 virtual math::Vector3 gazebo::physics::Collision::GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the collision in the world frame.

Returns

The linear velocity of the collision in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 345).

10.37.3.24 virtual void gazebo::physics::Collision::Init ( ) [virtual]

Initialize the collision.

Reimplemented from **gazebo::physics::Base** (p. 153).

10.37.3.25 bool gazebo::physics::Collision::IsPlaceable ( ) const

Return whether this collision is movable.

Example on an immovable object is a ray.

Returns

True if the object is immovable.

10.37.3.26 virtual void gazebo::physics::Collision::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the collision.

Parameters

in	*_sdf*	SDF to load from.

Reimplemented from **gazebo::physics::Entity** (p. 346).

Reimplemented in **gazebo::physics::BulletCollision** (p. 173), and **gazebo::physics::ODECollision** (p. 581).

10.37.3.27 void gazebo::physics::Collision::ProcessMsg (const msgs::Collision & *_msg*)

Update parameters from a message.

Parameters

in	*_msg*	Message to update from.

10.37.3.28 virtual void gazebo::physics::Collision::SetCategoryBits (unsigned int *_bits*) [pure virtual]

Set the category bits, used during collision detection.

Parameters

in	*_bits*	The bits to set.

Implemented in **gazebo::physics::ODECollision** (p. 581), and **gazebo::physics::BulletCollision** (p. 174).

10.37.3.29 virtual void gazebo::physics::Collision::SetCollideBits (unsigned int *_bits*) [pure virtual]

Set the collide bits, used during collision detection.

Parameters

in	*_bits*	The bits to set.

Implemented in **gazebo::physics::ODECollision** (p. 581), and **gazebo::physics::BulletCollision** (p. 174).

10.37.3.30 void gazebo::physics::Collision::SetCollision (bool *_placeable*)

Set the encapsulated collsion object.

Parameters

in	*_placeable*	True to make the object m.

10.37.3.31 void gazebo::physics::Collision::SetContactsEnabled (bool *_enable*)

Turn contact recording on or off.

Parameters

in	*_enable*	True to enable collision contacts.

10.37.3.32 void gazebo::physics::Collision::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	*_retro*	The laser retro value.

10.37.3.33 void gazebo::physics::Collision::SetShape (ShapePtr *_shape*)

Set the shape for this collision.

Parameters

in	*_shape*	The shape for this collision object.

10.37.3.34 void gazebo::physics::Collision::SetState (const CollisionState & *_state*)

Set the current collision state.

Parameters

in	*The*	collision state.

10.37.3.35 virtual void gazebo::physics::Collision::UpdateParameters (sdf::ElementPtr _sdf) [virtual]

Update the parameters using new sdf values.

Parameters

in	*_sdf*	SDF values to update from.

Reimplemented from **gazebo::physics::Entity** (p. 348).

10.37.4 Member Data Documentation

10.37.4.1 LinkPtr gazebo::physics::Collision::link [protected]

The link this collision belongs to.

10.37.4.2 bool gazebo::physics::Collision::placeable [protected]

Flag for placeable.

10.37.4.3 ShapePtr gazebo::physics::Collision::shape [protected]

Pointer to **physics::Shape** (p. 779).

The documentation for this class was generated from the following file:

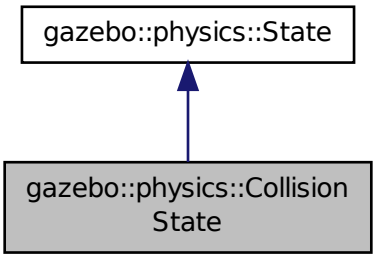
- **Collision.hh**

10.38 gazebo::physics::CollisionState Class Reference

Store state information of a **physics::Collision** (p. 262) object.

`#include <physics/phyiscs.hh>`

Inheritance diagram for gazebo::physics::CollisionState:



Public Member Functions

- **CollisionState** ()

  Default constructor.
- **CollisionState** (const **CollisionPtr** _collision)

  Constructor.
- virtual ~**CollisionState** ()

  Destructor.
- **math::Pose GetPose** () const

  *Get the **Collision** (p. 262) pose.*
- virtual void **Load** (**sdf::ElementPtr** _elem)

  Load state from SDF element.

Additional Inherited Members

10.38.1 Detailed Description

Store state information of a **physics::Collision** (p. 262) object.

This class captures the entire state of a **Collision** (p. 262) at one specific time during a simulation run.

State (p. 813) of a **Collision** (p. 262) is its Pose.

10.38.2 Constructor & Destructor Documentation

10.38.2.1 gazebo::physics::CollisionState::CollisionState ( )

Default constructor.

10.38.2.2 gazebo::physics::CollisionState::CollisionState (const CollisionPtr *_collision*)

Constructor.

Build a **CollisionState** (p. 272) from an existing **Collision** (p. 262).

Parameters

in	*_model*	Pointer to the **Link** (p. 454) from which to gather state info.

10.38.2.3 virtual gazebo::physics::CollisionState::~CollisionState ( ) [virtual]

Destructor.

10.38.3 Member Function Documentation

10.38.3.1 math::Pose gazebo::physics::CollisionState::GetPose ( ) const

Get the **Collision** (p. 262) pose.

10.38.3.2 virtual void gazebo::physics::CollisionState::Load (sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Load **CollisionState** (p. 272) information from stored data in and SDF::Element

Parameters

in	*_elem*	Pointer to the SDF::Element containing state info.

Implements **gazebo::physics::State** (p. 815).

The documentation for this class was generated from the following file:

- **CollisionState.hh**

10.39 gazebo::common::Color Class Reference

Defines a color.

`#include <Color.hh>`

Public Types

- typedef unsigned int **ABGR**
- typedef unsigned int **ARGB**
- typedef unsigned int **BGRA**
- typedef unsigned int **RGBA**

Public Member Functions

- **Color** ()

  Constructor.
- **Color** (float _r, float _g, float _b, float _a=1.0)

  Constructor.
- **Color** (const **Color** &_clr)

  Copy Constructor.
- virtual ~**Color** ()

  Destructor.
- **ABGR GetAsABGR** () const

  Get as uint32 ABGR packed value.
- **ARGB GetAsARGB** () const

  Get as uint32 ARGB packed value.
- **BGRA GetAsBGRA** () const

  Get as uint32 BGRA packed value.
- **math::Vector3 GetAsHSV** () const

  Get the color in HSV colorspace.
- **RGBA GetAsRGBA** () const

  Get as uint32 RGBA packed value.

- **math::Vector3 GetAsYUV** () const
  Get the color in YUV colorspace.
- bool **operator!=** (const **Color** &_pt) const
  Inequality operator.
- const **Color operator*** (const **Color** &_pt) const
  Multiplication operator.
- const **Color operator*** (const float &_v) const
  Multiply all color components by _v.
- const **Color** & **operator*=** (const **Color** &_pt)
  Multiplication equal operator.
- **Color operator+** (const **Color** &_pt) const
  Addition operator (this + _pt)
- **Color operator+** (const float &_v) const
  Add _v to all color components.
- const **Color** & **operator+=** (const **Color** &_pt)
  Addition equal operator.
- **Color operator-** (const **Color** &_pt) const
  Subtraction operator.
- **Color operator-** (const float &_v) const
  Subtract _v from all color components.
- const **Color** & **operator-=** (const **Color** &_pt)
  Subtraction equal operator.
- const **Color operator**/ (const **Color** &_pt) const
  Division operator.
- const **Color operator**/ (const float &_v) const
  Divide all color component by _v.
- const **Color** & **operator/=** (const **Color** &_pt)
  Division equal operator.
- **Color** & **operator=** (const **Color** &_pt)
  Equal operator.
- bool **operator==** (const **Color** &_pt) const
  Equality operator.
- float **operator[]** (unsigned int _index)
  Array index operator.
- void **Reset** ()
  Reset the color to default values.
- void **Set** (float _r=1, float _g=1, float _b=1, float _a=1)
  Set the contents of the vector.
- void **SetFromABGR** (const **ABGR** _v)
  Set from uint32 ABGR packed value.
- void **SetFromARGB** (const **ARGB** _v)
  Set from uint32 ARGB packed value.
- void **SetFromBGRA** (const **BGRA** _v)
  Set from uint32 BGRA packed value.
- void **SetFromHSV** (float _h, float _s, float _v)
  Set a color based on HSV values.
- void **SetFromRGBA** (const **RGBA** _v)
  Set from uint32 RGBA packed value.
- void **SetFromYUV** (float _y, float _u, float _v)
  Set from yuv.

Public Attributes

- float **a**
- float **b**
- float **g**
- float **r**

Static Public Attributes

- static const **Color Black**

  (0, 0, 0)
- static const **Color Blue**

  (0, 0, 1)
- static const **Color Green**

  (0, 1, 0)
- static const **Color Purple**

  (1, 0, 1)
- static const **Color Red**

  (1, 0, 0)
- static const **Color White**

  (1, 1, 1)
- static const **Color Yellow**

  (1, 1, 0)

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **Color** &_pt)

  Stream insertion operator.
- std::istream & **operator**>> (std::istream &_in, **Color** &_pt)

  Stream insertion operator.

10.39.1 Detailed Description

Defines a color.

10.39.2 Member Typedef Documentation

10.39.2.1 typedef unsigned int gazebo::common::Color::ABGR

10.39.2.2 typedef unsigned int gazebo::common::Color::ARGB

10.39.2.3 typedef unsigned int gazebo::common::Color::BGRA

10.39.2.4 typedef unsigned int gazebo::common::Color::RGBA

10.39.3 Constructor & Destructor Documentation

10.39.3.1 gazebo::common::Color::Color ( )

Constructor.

10.39.3.2 gazebo::common::Color::Color (float _r, float _g, float _b, float _a = 1.0)

Constructor.

Parameters

in	*_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)

10.39.3.3 gazebo::common::Color::Color (const Color & _clr)

Copy Constructor.

Parameters

in	*_clr*	**Color** (p. 274) to copy

10.39.3.4 virtual gazebo::common::Color::~Color ( ) [virtual]

Destructor.

10.39.4 Member Function Documentation

10.39.4.1 ABGR gazebo::common::Color::GetAsABGR ( ) const

Get as uint32 ABGR packed value.

Returns

the color

10.39.4.2 ARGB gazebo::common::Color::GetAsARGB ( ) const

Get as uint32 ARGB packed value.

Returns

the color

10.39.4.3 BGRA gazebo::common::Color::GetAsBGRA ( ) const

Get as uint32 BGRA packed value.

Returns

the color

10.39.4.4 math::Vector3 gazebo::common::Color::GetAsHSV ( ) const

Get the color in HSV colorspace.

Returns

HSV values in a **math::Vector3** (p. 902) format

10.39.4.5 RGBA gazebo::common::Color::GetAsRGBA ( ) const

Get as uint32 RGBA packed value.

Returns

the color

10.39.4.6 math::Vector3 gazebo::common::Color::GetAsYUV ( ) const

Get the color in YUV colorspace.

Returns

the YUV color

10.39.4.7 bool gazebo::common::Color::operator!= (const Color & *_pt*) const

Inequality operator.

Parameters

in	*_pt*	The color to check for inequality

Returns

True if the this color does not equal _pt

10.39.4.8 const Color gazebo::common::Color::operator* (const Color & *_pt*) const

Multiplication operator.

Parameters

in	*_pt*	The color to muliply by

Returns

The resulting color

10.39.4.9 const Color gazebo::common::Color::operator∗ (const float & _v) const

Multiply all color components by _v.

Parameters

in	*_v*	The value to multiply by

Returns

The resulting color

10.39.4.10 const Color& gazebo::common::Color::operator∗= (const Color & _pt)

Multiplication equal operator.

Parameters

in	*_pt*	The color to muliply by

Returns

The resulting color

10.39.4.11 Color gazebo::common::Color::operator+ (const Color & _pt) const

Addition operator (this + _pt)

Parameters

in	*_pt*	**Color** (p. 274) to add

Returns

The resulting color

10.39.4.12 Color gazebo::common::Color::operator+ (const float & _v) const

Add _v to all color components.

Parameters

in	*_v*	Value to add to each color component

Returns

The resulting color

10.39.4.13 const Color& gazebo::common::Color::operator+= (const Color & _pt)

Addition equal operator.

Parameters

in	*_pt*	**Color** (p. 274) to add

Returns

The resulting color

10.39.4.14 Color gazebo::common::Color::operator- (const Color & _pt) const

Subtraction operator.

Parameters

in	*_pt*	The color to substract

Returns

The resulting color

10.39.4.15 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

10.39.4.16 const Color& gazebo::common::Color::operator-= (const Color & _pt)

Subtraction equal operator.

Parameters

in	*_pt*	**Color** (p. 274) to subtract

Returns

The resulting color

10.39.4.17 const Color gazebo::common::Color::operator/ (const Color & _pt) const

Division operator.

Parameters

in	*_pt*	**Color** (p. 274) to divide by

Returns

The resulting color

10.39.4.18 const Color gazebo::common::Color::operator/ (const float & _v) const

Divide all color component by _v.

Parameters

in	*_v*	The value to divide by

Returns

The resulting color

10.39.4.19 const Color& gazebo::common::Color::operator/= (const Color & _pt)

Division equal operator.

Parameters

in	*_pt*	**Color** (p. 274) to divide by

Returns

The resulting color

10.39.4.20 Color& gazebo::common::Color::operator= (const Color & _pt)

Equal operator.

Parameters

in	*_pt*	**Color** (p. 274) to copy

Returns

Reference to this color

10.39.4.21 bool gazebo::common::Color::operator== (const Color & _pt) const

Equality operator.

Parameters

in	*_pt*	The color to check for equality

Returns

True if the this color equals _pt

10.39.4.22 float gazebo::common::Color::operator[] (unsigned int _index)

Array index operator.

Parameters

in	*_index*	**Color** (p. 274) component index(0=red, 1=green, 2=blue)

Returns

r, g, b, or a when _index is 0, 1, 2 or 3

10.39.4.23 void gazebo::common::Color::Reset ( )

Reset the color to default values.

10.39.4.24 void gazebo::common::Color::Set (float _r = 1, float _g = 1, float _b = 1, float _a = 1)

Set the contents of the vector.

Parameters

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)

10.39.4.25 void gazebo::common::Color::SetFromABGR (const ABGR _v)

Set from uint32 ABGR packed value.

Parameters

in	_v	the new color

10.39.4.26 void gazebo::common::Color::SetFromARGB (const ARGB _v)

Set from uint32 ARGB packed value.

Parameters

in	_v	the new color

10.39.4.27 void gazebo::common::Color::SetFromBGRA (const BGRA _v)

Set from uint32 BGRA packed value.

Parameters

in	_v	the new color

10.39.4.28 void gazebo::common::Color::SetFromHSV (float _h, float _s, float _v)

Set a color based on HSV values.

Parameters

in	_h	Hue(0..360)
in	_s	Saturation(0..1)
in	_v	Value(0..1)

10.39.4.29 void gazebo::common::Color::SetFromRGBA (const RGBA _v)

Set from uint32 RGBA packed value.

Parameters

in	_v	the new color

10.39.4.30 void gazebo::common::Color::SetFromYUV (float _y, float _u, float _v)

Set from yuv.

Parameters

in	_y	value
in	_u	value
in	_v	value

10.39.5 Friends And Related Function Documentation

10.39.5.1 std::ostream& operator<< (std::ostream & _out, const Color & _pt) [friend]

Stream insertion operator.

Parameters

in	_out	the output stream
in	_pt	the color

Returns

the output stream

10.39.5.2 std::istream& operator>> (std::istream & _in, Color & _pt) [friend]

Stream insertion operator.

Parameters

in	_in	the input stream
in	_pt	

10.39.6 Member Data Documentation

10.39.6.1 float gazebo::common::Color::a

10.39.6.2 float gazebo::common::Color::b

10.39.6.3 const Color gazebo::common::Color::Black [static]

(0, 0, 0)

10.39.6.4 const Color gazebo::common::Color::Blue [static]

(0, 0, 1)

10.39.6.5 float gazebo::common::Color::g

10.39.6.6 const Color gazebo::common::Color::Green [static]

(0, 1, 0)

10.39.6.7 const Color gazebo::common::Color::Purple [static]

(1, 0, 1)

10.39.6.8 float gazebo::common::Color::r

10.39.6.9 const Color gazebo::common::Color::Red [static]

(1, 0, 0)

10.39.6.10 const Color gazebo::common::Color::White [static]

(1, 1, 1)

10.39.6.11 const Color gazebo::common::Color::Yellow [static]

(1, 1, 0)

The documentation for this class was generated from the following file:

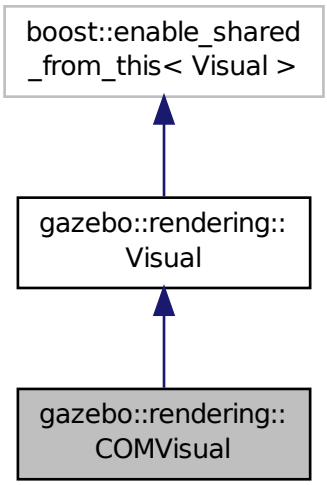
- **Color.hh**

10.40 gazebo::rendering::COMVisual Class Reference

Basic Center of Mass visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::COMVisual:



Public Member Functions

- **COMVisual** (const std::string &_name, **VisualPtr** _vis)

  Constructor.
- virtual ~**COMVisual** ()

  Destructor.
- virtual void **Load** (**sdf::ElementPtr** _elem)

  *Load the **Visual** (p. 931) from an SDF pointer.*
- virtual void **Load** (ConstLinkPtr &_msg)

  Load from a message.

Additional Inherited Members

10.40.1 Detailed Description

Basic Center of Mass visualization.

10.40.2 Constructor & Destructor Documentation

10.40.2.1 gazebo::rendering::COMVisual::COMVisual (const std::string & *_name,* VisualPtr *_vis*)

Constructor.

Parameters

in	*_name*	Name of the **Visual** (p. 931)
in	*_vis*	Parent **Visual** (p. 931)

10.40.2.2 virtual gazebo::rendering::COMVisual::~COMVisual ( ) [virtual]

Destructor.

10.40.3 Member Function Documentation

10.40.3.1 virtual void gazebo::rendering::COMVisual::Load (sdf::ElementPtr *_elem*) [virtual]

Load the **Visual** (p. 931) from an SDF pointer.

Parameters

in	*_elem*	SDF Element pointer

10.40.3.2 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**

10.41 gazebo::event::Connection Class Reference

A class that encapsulates a connection.

`#include <Event.hh>`

Public Member Functions

- **Connection** ()
  Constructor.
- **Connection** (**Event** ∗_e, int _i)
  Constructor.
- ∼**Connection** ()
  Destructor.
- int **GetId** () const
  Get the id of this connection.

10.41.1 Detailed Description

A class that encapsulates a connection.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 gazebo::event::Connection::Connection ( ) [inline]

Constructor.

10.41.2.2 gazebo::event::Connection::Connection (**Event** ∗ *_e,* int *_i*)

Constructor.

Parameters

in	*_e*	**Event** (p. 350) pointer to connect with
in	*_i*	Unique id

10.41.2.3 gazebo::event::Connection::∼Connection ( )

Destructor.

10.41.3 Member Function Documentation

10.41.3.1 int gazebo::event::Connection::GetId ( ) const

Get the id of this connection.

Returns

The id of this connection

Referenced by gazebo::event::EventT< T >::Disconnect().

The documentation for this class was generated from the following file:

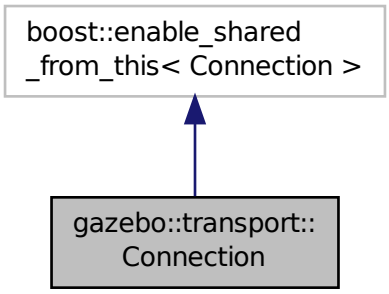
- **Event.hh**

10.42 gazebo::transport::Connection Class Reference

TCP/IP **Connection** (p. 288).

`#include <Connection.hh>`

Inheritance diagram for gazebo::transport::Connection:



Public Types

- typedef boost::function< void(const **ConnectionPtr** &)> **AcceptCallback**
- typedef boost::function< void(const std::string &data)> **ReadCallback**

Public Member Functions

- **Connection** ()

  Constructor.

- virtual ~**Connection** ()

  Destructor.
- template<typename Handler >
  void **AsyncRead** (Handler handler)

  Peform and asyncronous read.
- void **Cancel** ()

  Cancel all async operations on an open socket.
- bool **Connect** (const std::string &host, unsigned int port)

  Connect to a remote host.
- **event::ConnectionPtr ConnectToShutdown** (boost::function< void()> subscriber_)
- void **DisconnectShutdown** (**event::ConnectionPtr** subscriber_)
- void **EnqueueMsg** (const std::string &_buffer, bool _force=false)

  Write data to the socket.
- std::string **GetLocalAddress** () const

  Get the address of this connection.
- std::string **GetLocalHostname** () const

  Get the local hostname.
- unsigned int **GetLocalPort** () const

  Get the port of this connection.
- std::string **GetLocalURI** () const

  Get the local URI.
- std::string **GetRemoteAddress** () const

  Get the remote address.
- std::string **GetRemoteHostname** () const

  Get the remote hostname.
- unsigned int **GetRemotePort** () const

  Get the remote port number.
- std::string **GetRemoteURI** () const

  Get the remote URI.
- bool **IsOpen** () const

  Return true if the connection is open.
- void **Listen** (unsigned int port, const **AcceptCallback** &accept_cb)

  Start a server that listens on a port.
- void **ProcessWriteQueue** ()

  Handle on write callbacks.
- bool **Read** (std::string &data)

  Read data from the socket.
- void **Shutdown** ()

  Shutdown the socket.
- void **StartRead** (const **ReadCallback** &cb)

  Start a thread that reads from the connection, and passes new message to the ReadCallback.
- void **StopRead** ()

  Stop the read loop.

Public Attributes

- unsigned int **id**
- unsigned int **writeCount**

10.42.1 Detailed Description

TCP/IP **Connection** (p. 288).

10.42.2 Member Typedef Documentation

10.42.2.1 typedef boost::function<void(const ConnectionPtr&)> gazebo::transport::Connection::AcceptCallback

10.42.2.2 typedef boost::function<void(const std::string &data)> gazebo::transport::Connection::ReadCallback

10.42.3 Constructor & Destructor Documentation

10.42.3.1 gazebo::transport::Connection::Connection ( )

Constructor.

10.42.3.2 virtual gazebo::transport::Connection::~Connection ( ) [virtual]

Destructor.

10.42.4 Member Function Documentation

10.42.4.1 template<typename Handler > void gazebo::transport::Connection::AsyncRead (Handler *handler*) [inline]

Peform and asyncronous read.

References gzerr, HEADER_LENGTH, and IsOpen().

10.42.4.2 void gazebo::transport::Connection::Cancel ( )

Cancel all async operations on an open socket.

10.42.4.3 bool gazebo::transport::Connection::Connect (const std::string & *host*, unsigned int *port*)

Connect to a remote host.

10.42.4.4 event::ConnectionPtr gazebo::transport::Connection::ConnectToShutdown (boost::function< void()> *subscriber_*) [inline]

References gazebo::event::EventT< T >::Connect().

10.42.4.5 void gazebo::transport::Connection::DisconnectShutdown (event::ConnectionPtr *subscriber_*) [inline]

References gazebo::event::EventT< T >::Disconnect().

10.42.4.6 void gazebo::transport::Connection::EnqueueMsg (const std::string & *_buffer*, bool *_force* = false)

Write data to the socket.

10.42.4.7 std::string gazebo::transport::Connection::GetLocalAddress ( ) const

Get the address of this connection.

10.42.4.8 std::string gazebo::transport::Connection::GetLocalHostname ( ) const

Get the local hostname.

10.42.4.9 unsigned int gazebo::transport::Connection::GetLocalPort ( ) const

Get the port of this connection.

10.42.4.10 std::string gazebo::transport::Connection::GetLocalURI ( ) const

Get the local URI.

10.42.4.11 std::string gazebo::transport::Connection::GetRemoteAddress ( ) const

Get the remote address.

10.42.4.12 std::string gazebo::transport::Connection::GetRemoteHostname ( ) const

Get the remote hostname.

10.42.4.13 unsigned int gazebo::transport::Connection::GetRemotePort ( ) const

Get the remote port number.

10.42.4.14 std::string gazebo::transport::Connection::GetRemoteURI ( ) const

Get the remote URI.

10.42.4.15 bool gazebo::transport::Connection::IsOpen ( ) const

Return true if the connection is open.

Referenced by AsyncRead().

10.42.4.16 void gazebo::transport::Connection::Listen (unsigned int *port,* const **AcceptCallback** & *accept_cb*)

Start a server that listens on a port.

10.42.4.17 void gazebo::transport::Connection::ProcessWriteQueue ( )

Handle on write callbacks.

10.42.4.18 bool gazebo::transport::Connection::Read (std::string & *data*)

Read data from the socket.

10.42.4.19 void gazebo::transport::Connection::Shutdown ( )

Shutdown the socket.

10.42.4.20 void gazebo::transport::Connection::StartRead (const ReadCallback & *cb*)

Start a thread that reads from the connection, and passes new message to the ReadCallback.

10.42.4.21 void gazebo::transport::Connection::StopRead ( )

Stop the read loop.

10.42.5 Member Data Documentation

10.42.5.1 unsigned int gazebo::transport::Connection::id

10.42.5.2 unsigned int gazebo::transport::Connection::writeCount

The documentation for this class was generated from the following file:

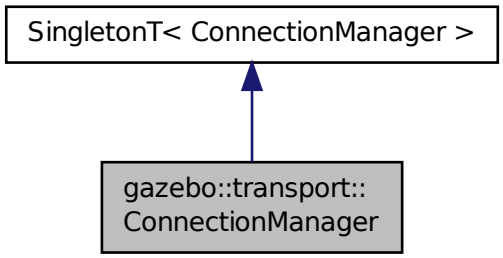
- **Connection.hh**

10.43 gazebo::transport::ConnectionManager Class Reference

Manager of connections.

`#include <ConnectionManager.hh>`

Inheritance diagram for gazebo::transport::ConnectionManager:



Public Member Functions

- void **Advertise** (const std::string &topic, const std::string &msgType)
- **ConnectionPtr ConnectToRemoteHost** (const std::string &host, unsigned int port)

  Connect to a remote server.
- void **Fini** ()

  Finalize the conneciton manager.
- void **GetAllPublishers** (std::list< msgs::Publish > &publishers)

  Explicitly update the publisher list.
- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

  Get all the topic namespaces.
- bool **Init** (const std::string &master_host, unsigned int master_port)
- bool **IsRunning** () const

  Return true if running (not stopped)
- void **RegisterTopicNamespace** (const std::string &_name)

  Register a new topic namespace.
- void **RemoveConnection** (**ConnectionPtr** &conn)

  Remove a connection.
- void **Run** ()

  Run the connection manager loop.
- void **RunUpdate** ()
- void **Stop** ()

  Stop the conneciton manager.
- void **Subscribe** (const std::string &_topic, const std::string &_msgType, bool _latching)
- void **Unadvertise** (const std::string &topic)
- void **Unsubscribe** (const msgs::Subscribe &_sub)
- void **Unsubscribe** (const std::string &_topic, const std::string &_msgType)

Protected Attributes

- std::vector< **event::ConnectionPtr** > **eventConnections**

Additional Inherited Members

10.43.1 Detailed Description

Manager of connections.

10.43.2 Member Function Documentation

10.43.2.1 void gazebo::transport::ConnectionManager::Advertise (const std::string & *topic,* const std::string & *msgType*)

10.43.2.2 ConnectionPtr gazebo::transport::ConnectionManager::ConnectToRemoteHost (const std::string & *host,* unsigned int *port*)

Connect to a remote server.

10.43.2.3 void gazebo::transport::ConnectionManager::Fini ( )

Finalize the conneciton manager.

10.43.2.4 void gazebo::transport::ConnectionManager::GetAllPublishers (std::list< msgs::Publish > & *publishers*)

Explicitly update the publisher list.

10.43.2.5 void gazebo::transport::ConnectionManager::GetTopicNamespaces (std::list< std::string > & *_namespaces*)

Get all the topic namespaces.

10.43.2.6 bool gazebo::transport::ConnectionManager::Init (const std::string & *master_host,* unsigned int *master_port*)

10.43.2.7 bool gazebo::transport::ConnectionManager::IsRunning ( ) const

Return true if running (not stopped)

10.43.2.8 void gazebo::transport::ConnectionManager::RegisterTopicNamespace (const std::string & *_name*)

Register a new topic namespace.

10.43.2.9 void gazebo::transport::ConnectionManager::RemoveConnection (ConnectionPtr & *conn*)

Remove a connection.

10.43.2.10 void gazebo::transport::ConnectionManager::Run ( )

Run the connection manager loop.

10.43.2.11 void gazebo::transport::ConnectionManager::RunUpdate ( )

10.43.2.12 void gazebo::transport::ConnectionManager::Stop ( )

Stop the conneciton manager.

10.43.2.13 void gazebo::transport::ConnectionManager::Subscribe (const std::string & *_topic,* const std::string & *_msgType,* bool *_latching*)

10.43.2.14 void gazebo::transport::ConnectionManager::Unadvertise (const std::string & *topic*)

10.43.2.15 void gazebo::transport::ConnectionManager::Unsubscribe (const msgs::Subscribe & *_sub*)

10.43.2.16 void gazebo::transport::ConnectionManager::Unsubscribe (const std::string & *_topic,* const std::string & *_msgType*)

10.43.3 Member Data Documentation

10.43.3.1 std::vector<event::ConnectionPtr> gazebo::transport::ConnectionManager::eventConnections [protected]

The documentation for this class was generated from the following file:

- **ConnectionManager.hh**

10.44 gazebo::common::Console Class Reference

Message, error, warning, and logging functionality.

`#include <Console.hh>`

Public Member Functions

- std::ostream & **ColorErr** (const std::string &_lbl, const std::string &_file, unsigned int _line, int _color)

  Use this to output an error to the terminal.
- std::ostream & **ColorMsg** (const std::string &_lbl, int _color)

  Use this to output a colored message to the terminal.
- void **Load** ()

  Load the message parameters.
- std::ofstream & **Log** ()

  Use this to output a message to a log file.
- void **SetQuiet** (bool _q)

  Set quiet output.

Static Public Member Functions

- static **Console** * **Instance** ()

  Return an instance to this class.

10.44.1 Detailed Description

Message, error, warning, and logging functionality.

10.44.2 Member Function Documentation

10.44.2.1 std::ostream& gazebo::common::Console::ColorErr (const std::string & *_lbl*, const std::string & *_file*, unsigned int *_line*, int *_color*)

Use this to output an error to the terminal.

Parameters

in	*_lbl*	Text label
in	*_file*	File containing the error
in	*_line*	Line containing the error
in	*_color*	**Color** (p. 274) to make the label

Returns

Reference to an output stream

10.44.2.2 std::ostream& gazebo::common::Console::ColorMsg (const std::string & *_lbl*, int *_color*)

Use this to output a colored message to the terminal.

Parameters

in	*_lbl*	Text label
in	*_color*	**Color** (p. 274) to make the label

Returns

Reference to an output stream

10.44.2.3 static Console* gazebo::common::Console::Instance ( ) [static]

Return an instance to this class.

10.44.2.4 void gazebo::common::Console::Load ( )

Load the message parameters.

10.44.2.5 std::ofstream& gazebo::common::Console::Log ( )

Use this to output a message to a log file.

Returns

Reference to output stream

10.44.2.6 void gazebo::common::Console::SetQuiet (bool *_q*)

Set quiet output.

Parameters

in	*q*	True to prevent warning

The documentation for this class was generated from the following file:

- **Console.hh**

10.45 gazebo::physics::Contact Class Reference

A contact between two collisions.

```
#include <physics/physics.hh>
```

Public Member Functions

- **Contact** ()

  Constructor.
- **Contact** (const **Contact** &_contact)

  Copy constructor.
- virtual ~**Contact** ()

  Destructor.
- **Contact Clone** () const

  Clone the contact.
- **Contact** & **operator=** (const **Contact** &_contact)

  Operator =.
- void **Reset** ()

  Reset to default values.

Public Attributes

- **Collision** * **collision1**

  Pointer to the first collision in the contact.
- **Collision** * **collision2**

  Pointer to the second collision in the contact.
- int **count**

  Length of all the arrays.
- double **depths** [32]

  Array of contact depths.
- **JointFeedback forces** [32]

  Array of forces for the contact.
- **math::Vector3 normals** [32]

  Array of force normals.
- **math::Vector3 positions** [32]

  Array of force positions.
- **common::Time time**

  Time at which the contact occured.

10.45.1 Detailed Description

A contact between two collisions.

Each contact can consist of a number of contact points

10.45.2 Constructor & Destructor Documentation

10.45.2.1 gazebo::physics::Contact::Contact ( )

Constructor.

10.45.2.2 gazebo::physics::Contact::Contact (const Contact & _contact)

Copy constructor.

Parameters

in	*_contact*	**Contact** (p. 296) to copy.

10.45.2.3 virtual gazebo::physics::Contact::~Contact ( ) [virtual]

Destructor.

10.45.3 Member Function Documentation

10.45.3.1 Contact gazebo::physics::Contact::Clone ( ) const

Clone the contact.

Returns

A cope of this contact.

10.45.3.2 Contact& gazebo::physics::Contact::operator= (const Contact & _contact)

Operator =.

Parameters

in	*_contact*	**Contact** (p. 296) to copy.

Returns

Reference to this contact

10.45.3.3 void gazebo::physics::Contact::Reset ( )

Reset to default values.

10.45.4 Member Data Documentation

10.45.4.1 Collision* gazebo::physics::Contact::collision1

Pointer to the first collision in the contact.

10.45.4.2 Collision* gazebo::physics::Contact::collision2

Pointer to the second collision in the contact.

10.45.4.3 int gazebo::physics::Contact::count

Length of all the arrays.

10.45.4.4 double gazebo::physics::Contact::depths[32]

Array of contact depths.

10.45.4.5 JointFeedback gazebo::physics::Contact::forces[32]

Array of forces for the contact.

10.45.4.6 math::Vector3 gazebo::physics::Contact::normals[32]

Array of force normals.

10.45.4.7 math::Vector3 gazebo::physics::Contact::positions[32]

Array of force positions.

10.45.4.8 common::Time gazebo::physics::Contact::time

Time at which the contact occured.

The documentation for this class was generated from the following file:

- **Contact.hh**

10.46 gazebo::physics::ContactFeedback Class Reference

data structure for contact feedbacks

`#include <ODEPhysics.hh>`

Public Attributes

- **Contact contact**
- int **feedbackCount**
- dJointFeedback **feedbacks** [**MAX_CONTACT_JOINTS**]

10.46.1 Detailed Description

data structure for contact feedbacks

10.46.2 Member Data Documentation

10.46.2.1 Contact gazebo::physics::ContactFeedback::contact

10.46.2.2 int gazebo::physics::ContactFeedback::feedbackCount

10.46.2.3 dJointFeedback gazebo::physics::ContactFeedback::feedbacks[MAX_CONTACT_JOINTS]

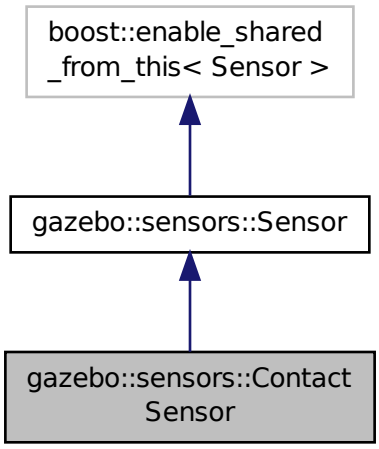
The documentation for this class was generated from the following file:

- **ODEPhysics.hh**

10.47 gazebo::sensors::ContactSensor Class Reference

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::ContactSensor:



Public Member Functions

- **ContactSensor** ()

  Constructor.
- virtual ~**ContactSensor** ()

  Destructor.
- **physics::Contact GetCollisionContact** (const std::string &_collisionName, unsigned int _index) const

  Get a contact for a collision by index.
- unsigned int **GetCollisionContactCount** (const std::string &_collisionName) const

  Return the number of contacts for an observed collision.

- unsigned int **GetCollisionCount** () const

  Get the number of collisions that the sensor is observing.
- std::string **GetCollisionName** (unsigned int _index) const

  Get a collision name at index _index.
- std::map< std::string, **physics::Contact** > **GetContacts** (const std::string &_collisionName)

  Gets contacts of a collision.
- boost::mutex * **GetUpdateMutex** () const

  returns a pointer to the mutex for locking while reading internally kept map of map of collision names and contacts
- virtual void **Init** ()

  Initialize the sensor.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** _sdf)

  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the sensor.
- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.

Additional Inherited Members

10.47.1 Constructor & Destructor Documentation

10.47.1.1 gazebo::sensors::ContactSensor::ContactSensor ( )

Constructor.

Parameters

body	The underlying collision test uses an ODE collision, so ray sensors must be attached to a body.

10.47.1.2 virtual gazebo::sensors::ContactSensor::~ContactSensor ( ) [virtual]

Destructor.

10.47.2 Member Function Documentation

10.47.2.1 virtual void gazebo::sensors::ContactSensor::Fini ( ) [protected], [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

10.47.2.2 physics::Contact gazebo::sensors::ContactSensor::GetCollisionContact (const std::string & *_collisionName*, unsigned int *_index*) const

Get a contact for a collision by index.

Parameters

in	*_collisionName*	Name of collision contact
in	*_index*	Index of collision to get

Returns

Contact object : nate check

10.47.2.3 unsigned int gazebo::sensors::ContactSensor::GetCollisionContactCount (const std::string & *_collisionName*) const

Return the number of contacts for an observed collision.

Parameters

in	*_collisionName*	The name of the observed collision

Returns

The collision contact count

10.47.2.4 unsigned int gazebo::sensors::ContactSensor::GetCollisionCount ( ) const

Get the number of collisions that the sensor is observing.

Returns

Number of collisions

10.47.2.5 std::string gazebo::sensors::ContactSensor::GetCollisionName (unsigned int *_index*) const

Get a collision name at index _index.

Parameters

in	*_index*	Index of collision in collection of collisions

Returns

name of collision

10.47.2.6 std::map<std::string, physics::Contact> gazebo::sensors::ContactSensor::GetContacts (const std::string & *_collisionName*)

Gets contacts of a collision.

Parameters

in	*_collisionName*	Name of collision

Returns

Container of contacts : nate check

10.47.2.7 boost::mutex* gazebo::sensors::ContactSensor::GetUpdateMutex ( ) const [inline]

returns a pointer to the mutex for locking while reading internally kept map of map of collision names and contacts

Returns

The mutex for the sensor

10.47.2.8 virtual void gazebo::sensors::ContactSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.47.2.9 virtual void gazebo::sensors::ContactSensor::Load (const std::string & *_worldName,* sdf::ElementPtr *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 765) parameters
in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.47.2.10 virtual void gazebo::sensors::ContactSensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.47.2.11 virtual void gazebo::sensors::ContactSensor::UpdateImpl (bool *_force*) [protected], [virtual]

Update the sensor information.

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 771).

The documentation for this class was generated from the following file:

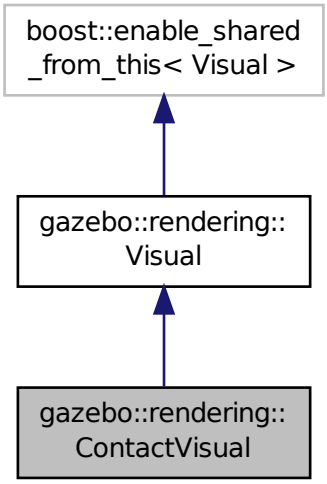
- **ContactSensor.hh**

10.48 gazebo::rendering::ContactVisual Class Reference

Contact visualization.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::ContactVisual:



Public Member Functions

- **ContactVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

  Constructor.
- virtual ~**ContactVisual** ()

  Destructor.

Additional Inherited Members

10.48.1 Detailed Description

Contact visualization.

This class visualizes contact points by drawing arrows in the 3D environment.

10.48.2 Constructor & Destructor Documentation

10.48.2.1 gazebo::rendering::ContactVisual::ContactVisual (const std::string & *_name,* VisualPtr *_vis,* const std::string & *_topicName*)

Constructor.

Parameters

in	*_name*	Name of the **ContactVisual** (p. 304)
in	*_vis*	Pointer the parent **Visual** (p. 931)
in	*_topicName*	Name of the topic which publishes the contact information.

10.48.2.2 virtual gazebo::rendering::ContactVisual::~ContactVisual ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **ContactVisual.hh**

10.49 gazebo::rendering::Conversions Class Reference

Conversions (p. 305) **Conversions.hh** (p. 1018) **rendering/Conversions.hh** (p. 1018).

```
#include <Conversions.hh>
```

Static Public Member Functions

- static Ogre::ColourValue **Convert** (const **common::Color** &_clr)
  Return the equivalent ogre color.
- static Ogre::Vector3 **Convert** (const **math::Vector3** &_v)
  *return **Ogre** (p. 118) Vector from Gazebo Vector3*
- static **math::Vector3 Convert** (const Ogre::Vector3 &_v)
  return gazebo Vector from ogre Vector3
- static Ogre::Quaternion **Convert** (const **math::Quaternion** &_v)
  *Gazebo quaternion to **Ogre** (p. 118) quaternion.*
- static **math::Quaternion Convert** (const Ogre::Quaternion &_v)
  ***Ogre** (p. 118) quaternion to Gazebo quaternion.*

10.49.1 Detailed Description

Conversions (p. 305) **Conversions.hh** (p. 1018) **rendering/Conversions.hh** (p. 1018).

A set of utility function to convert between Gazebo and **Ogre** (p. 118) data types

10.49.2 Member Function Documentation

10.49.2.1 static Ogre::ColourValue gazebo::rendering::Conversions::Convert (const common::Color & _clr) [static]

Return the equivalent ogre color.

Parameters

in	*_clr*	Gazebo color to convert

Returns

Ogre (p. 118) color value

10.49.2.2 static Ogre::Vector3 gazebo::rendering::Conversions::Convert (const math::Vector3 & _v) [static]

return **Ogre** (p. 118) Vector from Gazebo Vector3

Parameters

in	*_v*	Gazebo vector

Returns

Ogre (p. 118) vector

10.49.2.3 static math::Vector3 gazebo::rendering::Conversions::Convert (const Ogre::Vector3 & _v) [static]

return gazebo Vector from ogre Vector3

Parameters

in	*_v*	**Ogre** (p. 118) vector

Returns

Gazebo vector

10.49.2.4 static Ogre::Quaternion gazebo::rendering::Conversions::Convert (const math::Quaternion & _v) [static]

Gazebo quaternion to **Ogre** (p. 118) quaternion.

Parameters

in	*_v*	Gazebo quaternion

Returns

Ogre (p. 118) quaternion

10.49.2.5 static math::Quaternion gazebo::rendering::Conversions::Convert (const Ogre::Quaternion & _v) [static]

Ogre (p. 118) quaternion to Gazebo quaternion.

Parameters

in	*_v*	**Ogre** (p. 118) quaternion return Gazebo quaternion

The documentation for this class was generated from the following file:

- **Conversions.hh**

10.50 sdf::Converter Class Reference

Convert from one version of **SDF** (p. 762) to another.

`#include <Converter.hh>`

Static Public Member Functions

- static bool **Convert** (TiXmlElement *_elem, const std::string &_toVersion, bool _quiet=false)

10.50.1 Detailed Description

Convert from one version of **SDF** (p. 762) to another.

10.50.2 Member Function Documentation

10.50.2.1 static bool sdf::Converter::Convert (TiXmlElement * *_elem*, const std::string & *_toVersion*, bool *_quiet* = false) [static]

The documentation for this class was generated from the following file:

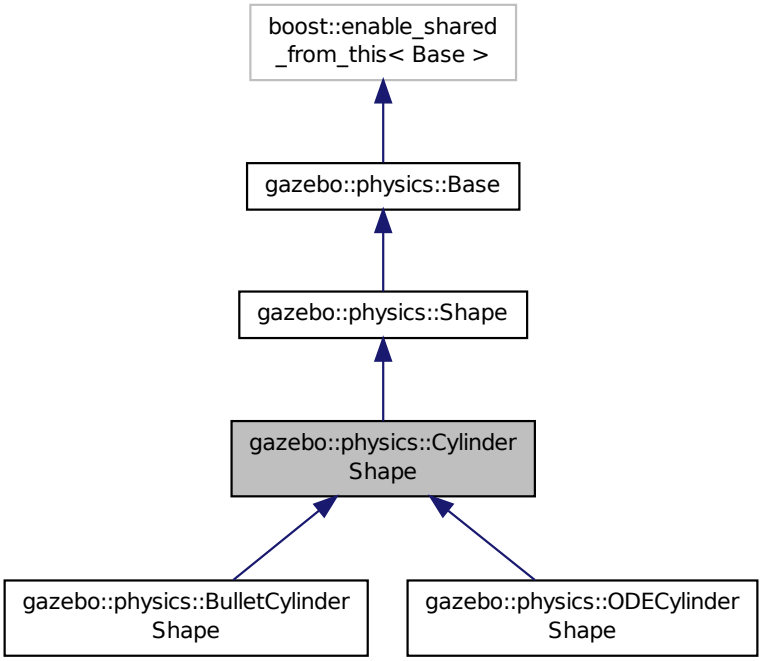
- **Converter.hh**

10.51 gazebo::physics::CylinderShape Class Reference

Cylinder collision.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::CylinderShape:



Public Member Functions

- **CylinderShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**CylinderShape** ()

  Destructor.
- void **FillMsg** (msgs::Geometry &_msg)

  Fill in the values for a geomertry message.
- virtual void **GetInertial** (double _mass, **InertialPtr** _inertial) const

  Get inertial for a shape.
- double **GetLength** () const

  Get length.
- virtual double **GetMass** (double _density) const

  Get the mass of the shape based on a density.
- double **GetRadius** () const

  Get radius.
- void **Init** ()

  Initialize the cylinder.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  Update values based on a message.

- void **SetLength** (double _length)

  Set length.
- void **SetRadius** (double _radius)

  Set radius.
- virtual void **SetSize** (double _radius, double _length)

  Set the size of the cylinder.

Additional Inherited Members

10.51.1 Detailed Description

Cylinder collision.

10.51.2 Constructor & Destructor Documentation

10.51.2.1 gazebo::physics::CylinderShape::CylinderShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of the shape.

10.51.2.2 virtual gazebo::physics::CylinderShape::~CylinderShape ( ) [virtual]

Destructor.

10.51.3 Member Function Documentation

10.51.3.1 void gazebo::physics::CylinderShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Fill in the values for a geomertry message.

Parameters

out	*_msg*	The geometry message to fill.

Implements **gazebo::physics::Shape** (p. 781).

10.51.3.2 virtual void gazebo::physics::CylinderShape::GetInertial (double *_mass,* InertialPtr *_inertial*) const [virtual]

Get inertial for a shape.

Parameters

in	*_mass*	Mass of the cylinder.
out	*_inertial*	**Inertial** (p. 414) element to populate with the result.

Reimplemented from **gazebo::physics::Shape** (p. 781).

10.51.3.3 double gazebo::physics::CylinderShape::GetLength ( ) const

Get length.

Returns

The cylinder length.

10.51.3.4 virtual double gazebo::physics::CylinderShape::GetMass (double *_density*) const [virtual]

Get the mass of the shape based on a density.

Parameters

in	*_density*	Density of the cylinder.

Reimplemented from **gazebo::physics::Shape** (p. 781).

10.51.3.5 double gazebo::physics::CylinderShape::GetRadius ( ) const

Get radius.

Returns

The cylinder radius.

10.51.3.6 void gazebo::physics::CylinderShape::Init ( ) [virtual]

Initialize the cylinder.

Implements **gazebo::physics::Shape** (p. 781).

10.51.3.7 virtual void gazebo::physics::CylinderShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Update values based on a message.

Parameters

in	*_msg*	Message to update from.

Implements **gazebo::physics::Shape** (p. 782).

10.51.3.8 void gazebo::physics::CylinderShape::SetLength (double *_length*)

Set length.

Parameters

in	*_length*	New length of the cylinder.

10.51.3.9 void gazebo::physics::CylinderShape::SetRadius (double *_radius*)

Set radius.

Parameters

in}	_radius New radius of the cylinder.

10.51.3.10 virtual void gazebo::physics::CylinderShape::SetSize (double *_radius,* double *_length*) [virtual]

Set the size of the cylinder.

Parameters

in	*_radius*	New radius.
in	*_lenght*	New length.

Reimplemented in **gazebo::physics::ODECylinderShape** (p. 584).

Referenced by gazebo::physics::ODECylinderShape::SetSize(), and gazebo::physics::BulletCylinderShape::SetSize().

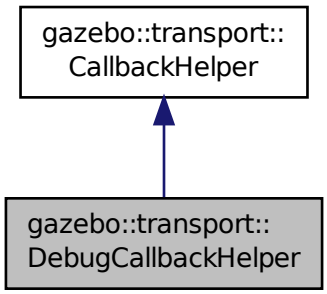
The documentation for this class was generated from the following file:

- **CylinderShape.hh**

10.52 gazebo::transport::DebugCallbackHelper Class Reference

`#include <CallbackHelper.hh>`

Inheritance diagram for gazebo::transport::DebugCallbackHelper:



Public Member Functions

- **DebugCallbackHelper** (const boost::function< void(ConstGzStringPtr &)> &cb)
- std::string **GetMsgType** () const

Get the typename of the message that is handled.

- virtual bool **HandleData** (const std::string &newdata)
- virtual bool **IsLocal** () const

  Return true if the callback is local, false if the callback is tied to a remote connection.

Additional Inherited Members

10.52.1 Constructor & Destructor Documentation

10.52.1.1 gazebo::transport::DebugCallbackHelper::DebugCallbackHelper (const boost::function< void(ConstGzStringPtr &)> & *cb*) [inline]

10.52.2 Member Function Documentation

10.52.2.1 std::string gazebo::transport::DebugCallbackHelper::GetMsgType ( ) const [inline], [virtual]

Get the typename of the message that is handled.

Reimplemented from **gazebo::transport::CallbackHelper** (p. 231).

10.52.2.2 virtual bool gazebo::transport::DebugCallbackHelper::HandleData (const std::string & *newdata*) [inline], [virtual]

Implements **gazebo::transport::CallbackHelper** (p. 231).

10.52.2.3 virtual bool gazebo::transport::DebugCallbackHelper::IsLocal ( ) const [inline], [virtual]

Return true if the callback is local, false if the callback is tied to a remote connection.

Implements **gazebo::transport::CallbackHelper** (p. 231).

The documentation for this class was generated from the following file:

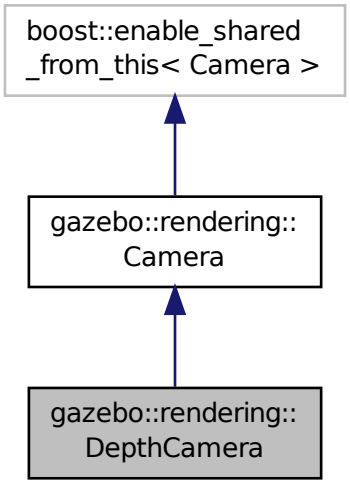
- **CallbackHelper.hh**

10.53 gazebo::rendering::DepthCamera Class Reference

Depth camera used to render depth data into an image buffer.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::DepthCamera:



Public Member Functions

- **DepthCamera** (const std::string &_namePrefix, **Scene** *_scene, bool _autoRender=true)

  Constructor.
- virtual ~**DepthCamera** ()

  Destructor.
- template<typename T >
  event::ConnectionPtr ConnectNewDepthFrame (T _subscriber)

  Connect a to the new depth image signal.
- template<typename T >
  event::ConnectionPtr ConnectNewRGBPointCloud (T subscriber)

  Connect a to the new rgb point cloud signal.
- void **CreateDepthTexture** (const std::string &_textureName)

  Create a texture which will hold the depth data.
- void **DisconnectNewDepthFrame** (**event::ConnectionPtr** &_c)

  Disconnect from an depth image singal.
- void **DisconnectNewRGBPointCloud** (**event::ConnectionPtr** &c)

  Disconnect from an rgb point cloud singal.
- void **Fini** ()

  Finalize the camera.
- virtual const float * **GetDepthData** ()

  All things needed to get back z buffer for depth data.
- void **Init** ()

  Initialize the camera.
- void **Load** (**sdf::ElementPtr** &_sdf)

Load the camera with a set of parmeters.

- void **Load** ()

  Load the camera with default parmeters.

- virtual void **PostRender** ()

  Render the camera.

- virtual void **SetDepthTarget** (Ogre::RenderTarget ∗_target)

  Set the render target, which renders the depth data.

Protected Attributes

- Ogre::RenderTarget ∗ **depthTarget**

  Pointer to the depth target.

- Ogre::Texture ∗ **depthTexture**

  Pointer to the depth texture.

- Ogre::Viewport ∗ **depthViewport**

  Pointer to the depth viewport.

Additional Inherited Members

10.53.1 Detailed Description

Depth camera used to render depth data into an image buffer.

10.53.2 Constructor & Destructor Documentation

10.53.2.1 gazebo::rendering::DepthCamera::DepthCamera (const std::string & *_namePrefix,* Scene ∗ *_scene,* bool *_autoRender* = `true`)

Constructor.

Parameters

in	*_namePrefix*	Unique prefix name for the camera.
in	*_scene*	**Scene** (p. 746) that will contain the camera
in	*_autoRender*	Almost everyone should leave this as true.

10.53.2.2 virtual gazebo::rendering::DepthCamera::∼DepthCamera ( ) `[virtual]`

Destructor.

10.53.3 Member Function Documentation

10.53.3.1 template<typename T > event::ConnectionPtr gazebo::rendering::DepthCamera::ConnectNewDepthFrame (T *_subscriber*) `[inline]`

Connect a to the new depth image signal.

Parameters

in	*_subscriber*	Subscriber callback function

Returns

Pointer to the new Connection. This must be kept in scope

References gazebo::event::EventT< T >::Connect().

10.53.3.2 template<typename T > event::ConnectionPtr gazebo::rendering::DepthCamera::ConnectNewRGBPointCloud (T *subscriber*) [inline]

Connect a to the new rgb point cloud signal.

Parameters

in	*_subscriber*	Subscriber callback function

Returns

Pointer to the new Connection. This must be kept in scope

References gazebo::event::EventT< T >::Connect().

10.53.3.3 void gazebo::rendering::DepthCamera::CreateDepthTexture (const std::string & *_textureName*)

Create a texture which will hold the depth data.

Parameters

in	*_textureName*	Name of the texture to create

10.53.3.4 void gazebo::rendering::DepthCamera::DisconnectNewDepthFrame (event::ConnectionPtr & *_c*) [inline]

Disconnect from an depth image singal.

Parameters

in	*_c*	The connection to disconnect

References gazebo::event::EventT< T >::Disconnect().

10.53.3.5 void gazebo::rendering::DepthCamera::DisconnectNewRGBPointCloud (event::ConnectionPtr & *c*) [inline]

Disconnect from an rgb point cloud singal.

Parameters

in	*_c*	The connection to disconnect

References gazebo::event::EventT< T >::Disconnect().

10.53.3.6 void gazebo::rendering::DepthCamera::Fini ( )

Finalize the camera.

10.53.3.7 virtual const float* gazebo::rendering::DepthCamera::GetDepthData ( ) [virtual]

All things needed to get back z buffer for depth data.

Returns

The z-buffer as a float array

10.53.3.8 void gazebo::rendering::DepthCamera::Init ( )

Initialize the camera.

10.53.3.9 void gazebo::rendering::DepthCamera::Load (sdf::ElementPtr & *_sdf*)

Load the camera with a set of parmeters.

Parameters

in	*_sdf*	The SDF camera info

10.53.3.10 void gazebo::rendering::DepthCamera::Load ( )

Load the camera with default parmeters.

10.53.3.11 virtual void gazebo::rendering::DepthCamera::PostRender ( ) [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 248).

10.53.3.12 virtual void gazebo::rendering::DepthCamera::SetDepthTarget (Ogre::RenderTarget * *_target*) [virtual]

Set the render target, which renders the depth data.

Parameters

in	*_target*	Pointer to the render target

10.53.4 Member Data Documentation

10.53.4.1 Ogre::RenderTarget* gazebo::rendering::DepthCamera::depthTarget [protected]

Pointer to the depth target.

10.53.4.2 Ogre::Texture* gazebo::rendering::DepthCamera::depthTexture [protected]

Pointer to the depth texture.

10.53.4.3 Ogre::Viewport* gazebo::rendering::DepthCamera::depthViewport [protected]

Pointer to the depth viewport.

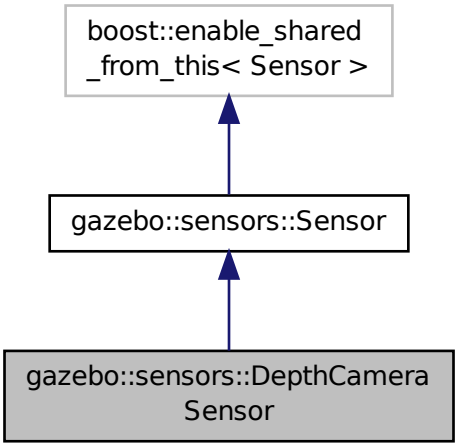
The documentation for this class was generated from the following file:

- **DepthCamera.hh**

10.54 gazebo::sensors::DepthCameraSensor Class Reference

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::DepthCameraSensor:



Public Member Functions

- **DepthCameraSensor** ()

  Constructor.
- virtual ~**DepthCameraSensor** ()

  Destructor.

- **rendering::DepthCameraPtr GetDepthCamera** () const
  *Returns a pointer to the **rendering::DepthCamera** (p. 312).*
- bool **SaveFrame** (const std::string &_filename)
  Saves an image frame of depth camera sensor to file.
- virtual void **SetActive** (bool _value)
  Set whether the sensor is active or not.
- virtual void **SetParent** (const std::string &_name)
  Set the parent of the sensor.

Protected Member Functions

- virtual void **Fini** ()
  Finalize the camera.
- virtual void **Init** ()
  Initialize the camera.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** &_sdf)
  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
  Load the sensor with default parameters.
- virtual void **UpdateImpl** (bool _force)
  Update the sensor information.

Additional Inherited Members

10.54.1 Constructor & Destructor Documentation

10.54.1.1 gazebo::sensors::DepthCameraSensor::DepthCameraSensor ( )

Constructor.

10.54.1.2 virtual gazebo::sensors::DepthCameraSensor::~DepthCameraSensor ( ) [virtual]

Destructor.

10.54.2 Member Function Documentation

10.54.2.1 virtual void gazebo::sensors::DepthCameraSensor::Fini ( ) [protected], [virtual]

Finalize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

10.54.2.2 rendering::DepthCameraPtr gazebo::sensors::DepthCameraSensor::GetDepthCamera ( ) const [inline]

Returns a pointer to the **rendering::DepthCamera** (p. 312).

Returns

Depth Camera pointer

10.54.2.3 virtual void gazebo::sensors::DepthCameraSensor::Init ( ) [protected], [virtual]

Initialize the camera.

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.54.2.4 virtual void gazebo::sensors::DepthCameraSensor::Load (const std::string & *_worldName,* sdf::ElementPtr & *_sdf*) [protected], [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 765) parameters
in	*_worldName*	Name of world to load from

10.54.2.5 virtual void gazebo::sensors::DepthCameraSensor::Load (const std::string & *_worldName*) [protected], [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.54.2.6 bool gazebo::sensors::DepthCameraSensor::SaveFrame (const std::string & *_filename*)

Saves an image frame of depth camera sensor to file.

Parameters

in	*Name*	of file to save as

Returns

True if saved, false if not

10.54.2.7 virtual void gazebo::sensors::DepthCameraSensor::SetActive (bool *_value*) [virtual]

Set whether the sensor is active or not.

Parameters

in	*_value*	True if active, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.54.2.8 virtual void gazebo::sensors::DepthCameraSensor::SetParent (const std::string & *_name*) [virtual]

Set the parent of the sensor.

Parameters

in	*_name*	Name of parent

Reimplemented from **gazebo::sensors::Sensor** (p. 771).

10.54.2.9 virtual void gazebo::sensors::DepthCameraSensor::UpdateImpl (bool *_force*) [protected], [virtual]

Update the sensor information.

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 771).

The documentation for this class was generated from the following file:

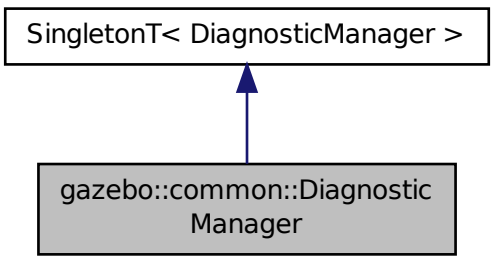
- **DepthCameraSensor.hh**

10.55 gazebo::common::DiagnosticManager Class Reference

A diagnostic manager class.

`#include <Diagnostics.hh>`

Inheritance diagram for gazebo::common::DiagnosticManager:



Public Member Functions

- **DiagnosticTimerPtr CreateTimer** (const std::string &_name)

  Create a new timer instance.

- bool **GetEnabled** () const

  Get whether the timers are enabled.
- std::string **GetLabel** (int _index) const

  Get a label for a timer.
- **Time GetTime** (int _index) const

  Get the time of a timer instance.
- **Time GetTime** (const std::string &_label) const

  Get a time based on a label.
- int **GetTimerCount** () const

  Get the number of timers.
- void **SetEnabled** (bool _e)

  Set whether timers are enabled.
- void **TimerStart** (**DiagnosticTimer** *_timer)

  A diagnostic timer has started.
- void **TimerStop** (**DiagnosticTimer** *_timer)

  A diagnostic timer has stopped.

Additional Inherited Members

10.55.1 Detailed Description

A diagnostic manager class.

10.55.2 Member Function Documentation

10.55.2.1 DiagnosticTimerPtr gazebo::common::DiagnosticManager::CreateTimer (const std::string & *_name*)

Create a new timer instance.

Parameters

in	*_name*	Name of the timer.

Returns

A pointer to the new diagnostic timer

10.55.2.2 bool gazebo::common::DiagnosticManager::GetEnabled ( ) const [inline]

Get whether the timers are enabled.

Returns

TRue if the timers are enabled

10.55.2.3 std::string gazebo::common::DiagnosticManager::GetLabel (int *_index*) const

Get a label for a timer.

Parameters

in	*_index*	Index of a timer instance

Returns

Label of the specified timer

10.55.2.4 Time gazebo::common::DiagnosticManager::GetTime (int *_index*) const

Get the time of a timer instance.

Parameters

in	*_index*	The index of a timer instance

Returns

Time (p. 840) of the specified timer

10.55.2.5 Time gazebo::common::DiagnosticManager::GetTime (const std::string & *_label*) const

Get a time based on a label.

Parameters

in	*_label*	Name of the timer instance

Returns

Time (p. 840) of the specified timer

10.55.2.6 int gazebo::common::DiagnosticManager::GetTimerCount ( ) const

Get the number of timers.

Returns

The number of timers

10.55.2.7 void gazebo::common::DiagnosticManager::SetEnabled (bool *_e*) [inline]

Set whether timers are enabled.

Parameters

in	*_e*	True = timers are enabled

10.55.2.8 void gazebo::common::DiagnosticManager::TimerStart (DiagnosticTimer * _timer)

A diagnostic timer has started.

Parameters

in	*_timer*	The timer to start

Referenced by gazebo::common::DiagnosticTimer::DiagnosticTimer().

10.55.2.9 void gazebo::common::DiagnosticManager::TimerStop (DiagnosticTimer * _timer)

A diagnostic timer has stopped.

Parameters

in	*_time*	The timer to stop

Referenced by gazebo::common::DiagnosticTimer::~DiagnosticTimer().

The documentation for this class was generated from the following file:

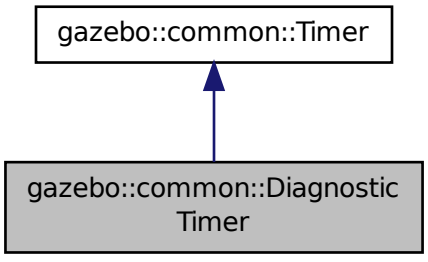
- **Diagnostics.hh**

10.56 gazebo::common::DiagnosticTimer Class Reference

A timer designed for diagnostics.

`#include <Diagnostics.hh>`

Inheritance diagram for gazebo::common::DiagnosticTimer:



Public Member Functions

- **DiagnosticTimer** (const std::string &_name)

  Constructor.

- virtual ~**DiagnosticTimer** ()

  Destructor.
- const std::string **GetName** () const

  Get the name of the timer.

10.56.1 Detailed Description

A timer designed for diagnostics.

10.56.2 Constructor & Destructor Documentation

10.56.2.1 gazebo::common::DiagnosticTimer::DiagnosticTimer (const std::string & *_name*) [inline]

Constructor.

Parameters

in	*_name*	Name of the timer

References gazebo::common::Timer::Start(), and gazebo::common::DiagnosticManager::TimerStart().

10.56.2.2 virtual gazebo::common::DiagnosticTimer::~DiagnosticTimer ( ) [inline], [virtual]

Destructor.

References gazebo::common::DiagnosticManager::TimerStop().

10.56.3 Member Function Documentation

10.56.3.1 const std::string gazebo::common::DiagnosticTimer::GetName ( ) const [inline]

Get the name of the timer.

Returns

The name of timer

The documentation for this class was generated from the following file:

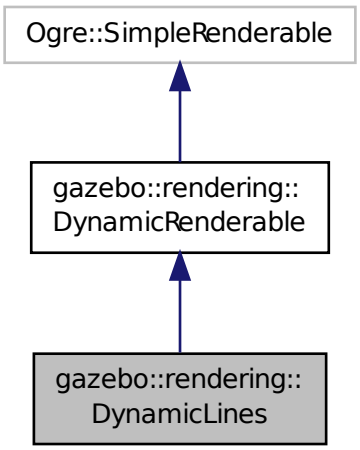
- **Diagnostics.hh**

10.57 gazebo::rendering::DynamicLines Class Reference

Class for drawing lines that can change.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::DynamicLines:



Public Member Functions

- **DynamicLines** (**RenderOpType** _opType=**RENDERING_LINE_STRIP**)

  Constructor.
- virtual ~**DynamicLines** ()

  Destructor.
- void **AddPoint** (const **math::Vector3** &_pt)

  Add a point to the point list.
- void **Clear** ()

  Remove all points from the point list.
- virtual const Ogre::String & **getMovableType** () const

  *Overridden function from **Ogre** (p. 118)'s base class.*
- const **math::Vector3** & **GetPoint** (unsigned int _index) const

  Return the location of an existing point in the point list.
- unsigned int **GetPointCount** () const

  Return the total number of points in the point list.
- void **SetPoint** (unsigned int _index, const **math::Vector3** &_value)

  Change the location of an existing point in the point list.
- void **Update** ()

  Call this to update the hardware buffer after making changes.

Static Public Member Functions

- static std::string **GetMovableType** ()

  Get type of movable.

Protected Member Functions

- virtual void **CreateVertexDeclaration** ()

  Implementation ***DynamicRenderable*** *(*p. *328), creates a simple vertex-only decl.*
- virtual void **FillHardwareBuffers** ()

  Implementation ***DynamicRenderable*** *(*p. *328), pushes point list out to hardware memory.*

Additional Inherited Members

10.57.1 Detailed Description

Class for drawing lines that can change.

10.57.2 Constructor & Destructor Documentation

10.57.2.1 gazebo::rendering::DynamicLines::DynamicLines (RenderOpType *_opType* = RENDERING_LINE_STRIP)

Constructor.

Parameters

in	*_opType*	The type of Line

10.57.2.2 virtual gazebo::rendering::DynamicLines::~DynamicLines ( ) [virtual]

Destructor.

10.57.3 Member Function Documentation

10.57.3.1 void gazebo::rendering::DynamicLines::AddPoint (const math::Vector3 & *_pt*)

Add a point to the point list.

Parameters

in	*pt*	**math::Vector3** (p. 902) point

10.57.3.2 void gazebo::rendering::DynamicLines::Clear ( )

Remove all points from the point list.

10.57.3.3 virtual void gazebo::rendering::DynamicLines::CreateVertexDeclaration ( ) [protected], [virtual]

Implementation **DynamicRenderable** (p. 328), creates a simple vertex-only decl.

Implements **gazebo::rendering::DynamicRenderable** (p. 329).

10.57.3.4 virtual void gazebo::rendering::DynamicLines::FillHardwareBuffers ( ) [protected], [virtual]

Implementation **DynamicRenderable** (p. 328), pushes point list out to hardware memory.

Implements **gazebo::rendering::DynamicRenderable** (p. 330).

10.57.3.5 static std::string gazebo::rendering::DynamicLines::GetMovableType ( ) [static]

Get type of movable.

Returns

This returns "gazebo::dynamiclines"

10.57.3.6 virtual const Ogre::String& gazebo::rendering::DynamicLines::getMovableType ( ) const [virtual]

Overridden function from **Ogre** (p. 118)'s base class.

Returns

Returns "gazebo::ogredynamicslines"

10.57.3.7 const math::Vector3& gazebo::rendering::DynamicLines::GetPoint (unsigned int _index) const

Return the location of an existing point in the point list.

Parameters

in	*_index*	Number of the point to return

Returns

math::Vector3 (p. 902) value of the point

10.57.3.8 unsigned int gazebo::rendering::DynamicLines::GetPointCount ( ) const

Return the total number of points in the point list.

Returns

Number of points

10.57.3.9 void gazebo::rendering::DynamicLines::SetPoint (unsigned int _index, const math::Vector3 & _value)

Change the location of an existing point in the point list.

Parameters

in	*_index*	Index of the point to set
in	*_value*	**math::Vector3** (p. 902) value to set the point to

10.57.3.10 void gazebo::rendering::DynamicLines::Update ( )

Call this to update the hardware buffer after making changes.

The documentation for this class was generated from the following file:

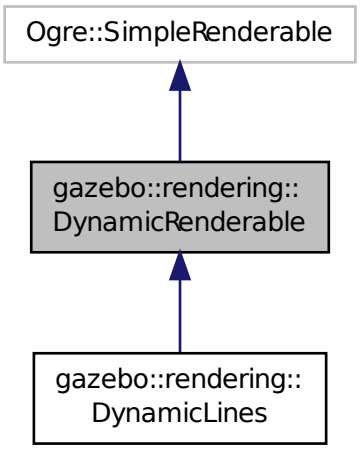
- **DynamicLines.hh**

10.58 gazebo::rendering::DynamicRenderable Class Reference

Abstract base class providing mechanisms for dynamically growing hardware buffers.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::DynamicRenderable:



Public Member Functions

- **DynamicRenderable** ()

  Constructor.
- virtual ~**DynamicRenderable** ()

  Virtual destructor.
- virtual Ogre::Real **getBoundingRadius** () const

  Implementation of Ogre::SimpleRenderable.
- **RenderOpType GetOperationType** () const

  Get the render operation type.
- virtual Ogre::Real **getSquaredViewDepth** (const Ogre::Camera *_cam) const

  Implementation of Ogre::SimpleRenderable.
- void **Init** (**RenderOpType** _opType, bool _useIndices=false)

Initializes the dynamic renderable.

- void **SetOperationType** (**RenderOpType** _opType)

  Set the render operation type.

Protected Member Functions

- virtual void **CreateVertexDeclaration** ()=0

  Creates the vertex declaration.

- virtual void **FillHardwareBuffers** ()=0

  Fills the hardware vertex and index buffers with data.

- void **PrepareHardwareBuffers** (size_t _vertexCount, size_t _indexCount)

  Prepares the hardware buffers for the requested vertex and index counts.

Protected Attributes

- size_t **indexBufferCapacity**

  Maximum capacity of the currently allocated index buffer.

- size_t **vertexBufferCapacity**

  Maximum capacity of the currently allocated vertex buffer.

10.58.1 Detailed Description

Abstract base class providing mechanisms for dynamically growing hardware buffers.

10.58.2 Constructor & Destructor Documentation

10.58.2.1 gazebo::rendering::DynamicRenderable::DynamicRenderable ( )

Constructor.

10.58.2.2 virtual gazebo::rendering::DynamicRenderable::~DynamicRenderable ( ) [virtual]

Virtual destructor.

10.58.3 Member Function Documentation

10.58.3.1 virtual void gazebo::rendering::DynamicRenderable::CreateVertexDeclaration ( ) [protected], [pure virtual]

Creates the vertex declaration.

Remarks

Override and set mRenderOp.vertexData->vertexDeclaration here. mRenderOp.vertexData will be created for you before this method is called.

Implemented in **gazebo::rendering::DynamicLines** (p. 326).

10.58.3.2 virtual void gazebo::rendering::DynamicRenderable::FillHardwareBuffers ( ) [protected], [pure virtual]

Fills the hardware vertex and index buffers with data.

Remarks

This function must call prepareHardwareBuffers() before locking the buffers to ensure the they are large enough for the data to be written. Afterwards the vertex and index buffers (if using indices) can be locked, and data can be written to them.

Implemented in **gazebo::rendering::DynamicLines** (p. 327).

10.58.3.3 virtual Ogre::Real gazebo::rendering::DynamicRenderable::getBoundingRadius ( ) const [virtual]

Implementation of Ogre::SimpleRenderable.

Returns

The bounding radius

10.58.3.4 RenderOpType gazebo::rendering::DynamicRenderable::GetOperationType ( ) const

Get the render operation type.

Returns

The render operation type.

10.58.3.5 virtual Ogre::Real gazebo::rendering::DynamicRenderable::getSquaredViewDepth (const Ogre::Camera * *_cam*) const [virtual]

Implementation of Ogre::SimpleRenderable.

Parameters

in	*_cam*	Pointer to the **Ogre** (p. 118) camera that views the renderable.

Returns

The squared depth in the **Camera** (p. 233)'s view

10.58.3.6 void gazebo::rendering::DynamicRenderable::Init (RenderOpType *_opType*, bool *_useIndices* = false)

Initializes the dynamic renderable.

Remarks

This function should only be called once. It initializes the render operation, and calls the abstract function **CreateVertexDeclaration()** (p. 329).

Parameters

in	*_opType*	The type of render operation to perform.
in	*_useIndices*	Specifies whether to use indices to determine the vertices to use as input.

10.58.3.7 void gazebo::rendering::DynamicRenderable::PrepareHardwareBuffers (size_t *_vertexCount*, size_t *_indexCount*) [protected]

Prepares the hardware buffers for the requested vertex and index counts.

Remarks

This function must be called before locking the buffers in fillHardwareBuffers(). It guarantees that the hardware buffers are large enough to hold at least the requested number of vertices and indices (if using indices). The buffers are possibly reallocated to achieve this.

The vertex and index count in the render operation are set to

the values of vertexCount and indexCount respectively.

Parameters

in	*_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.

10.58.3.8 void gazebo::rendering::DynamicRenderable::SetOperationType (RenderOpType *_opType*)

Set the render operation type.

Parameters

in	*_opType*	The type of render operation to perform.

10.58.4 Member Data Documentation

10.58.4.1 size_t gazebo::rendering::DynamicRenderable::indexBufferCapacity [protected]

Maximum capacity of the currently allocated index buffer.

10.58.4.2 size_t gazebo::rendering::DynamicRenderable::vertexBufferCapacity [protected]

Maximum capacity of the currently allocated vertex buffer.

The documentation for this class was generated from the following file:

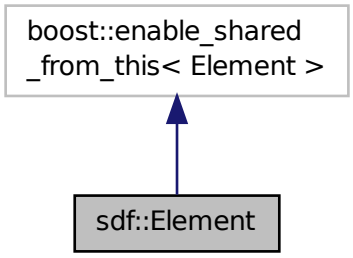
- **DynamicRenderable.hh**

10.59 sdf::Element Class Reference

SDF (p. 762) **Element** (p. 332) class.

`#include <SDF.hh>`

Inheritance diagram for sdf::Element:



Public Member Functions

- **Element** ()
- virtual ~**Element** ()
- void **AddAttribute** (const std::string &_key, const std::string &_type, const std::string &_defaultvalue, bool _required, const std::string &_description="")
- **ElementPtr AddElement** (const std::string &_name)
- void **AddElementDescription** (**ElementPtr** _elem)

  Add a new element description.
- void **AddValue** (const std::string &_type, const std::string &_defaultValue, bool _required, const std::string &_description="")
- void **ClearElements** ()
- **Element** * **Clone** () const
- void **Copy** (const **ElementPtr** _elem)

  *Copy values from an **Element** (p. 332).*
- **ParamPtr GetAttribute** (const std::string &_key)

  Get the param of an attribute.
- **ParamPtr GetAttribute** (unsigned int _index) const

  Get an attribute using an index.
- unsigned int **GetAttributeCount** () const

  Get the number of attributes.
- bool **GetAttributeSet** (const std::string &_key)

  Return true if the attribute was set (i.e. not default value)
- bool **GetCopyChildren** () const
- std::string **GetDescription** () const

  Get a text description of the element.
- **ElementPtr GetElement** (const std::string &_name) const

- **ElementPtr GetElement** (const std::string &_name)
- **ElementPtr GetElementDescription** (unsigned int _index) const

  Get an element description using an index.
- **ElementPtr GetElementDescription** (const std::string &_key) const

  Get an element descriptio using a key.
- unsigned int **GetElementDescriptionCount** () const

  Get the number of element descriptions.
- **ElementPtr GetElementImpl** (const std::string &_name) const
- **ElementPtr GetFirstElement** () const
- std::string **GetInclude** () const
- const std::string & **GetName** () const
- **ElementPtr GetNextElement** (const std::string &_name="") const
- **ElementPtr GetParent** () const
- const std::string & **GetRequired** () const
- **ParamPtr GetValue** ()

  Get the param of the elements value.
- bool **GetValueBool** (const std::string &_key="")
- char **GetValueChar** (const std::string &_key="")
- **gazebo::common::Color GetValueColor** (const std::string &_key="")
- double **GetValueDouble** (const std::string &_key="")
- float **GetValueFloat** (const std::string &_key="")
- int **GetValueInt** (const std::string &_key="")
- **gazebo::math::Pose GetValuePose** (const std::string &_key="")
- **gazebo::math::Quaternion GetValueQuaternion** (const std::string &_key="")
- std::string **GetValueString** (const std::string &_key="")
- **gazebo::common::Time GetValueTime** (const std::string &_key="")
- unsigned int **GetValueUInt** (const std::string &_key="")
- **gazebo::math::Vector2d GetValueVector2d** (const std::string &_key="")
- **gazebo::math::Vector3 GetValueVector3** (const std::string &_key="")
- bool **HasAttribute** (const std::string &_key)
- bool **HasElement** (const std::string &_name) const
- bool **HasElementDescription** (const std::string &_name)

  Return true if an element description exists.
- void **InsertElement** (**ElementPtr** _elem)
- void **PrintDescription** (std::string _prefix)
- void **PrintDocLeftPane** (std::string &_html, int _spacing, int &_index)

  *Helper function for **SDF::PrintDoc*** (p. *763*).
- void **PrintDocRightPane** (std::string &_html, int _spacing)

  *Helper function for **SDF::PrintDoc*** (p. *763*).
- void **PrintValues** (std::string _prefix)
- void **PrintWiki** (std::string _prefix)
- void **Reset** ()
- bool **Set** (const bool &_value)
- bool **Set** (const int &_value)
- bool **Set** (const unsigned int &_value)
- bool **Set** (const float &_value)
- bool **Set** (const double &_value)
- bool **Set** (const char &_value)
- bool **Set** (const std::string &_value)
- bool **Set** (const char *_value)

- bool **Set** (const **gazebo::math::Vector3** &_value)
- bool **Set** (const **gazebo::math::Vector2i** &_value)
- bool **Set** (const **gazebo::math::Vector2d** &_value)
- bool **Set** (const **gazebo::math::Quaternion** &_value)
- bool **Set** (const **gazebo::math::Pose** &_value)
- bool **Set** (const **gazebo::common::Color** &_value)
- bool **Set** (const **gazebo::common::Time** &_value)
- void **SetCopyChildren** (bool _value)
- void **SetDescription** (const std::string &_desc)

  Set a text description for the element.
- void **SetInclude** (const std::string &_filename)
- void **SetName** (const std::string &_name)
- void **SetParent** (const **ElementPtr** _parent)
- void **SetRequired** (const std::string &_req)
- std::string **ToString** (const std::string &_prefix) const
- void **Update** ()

10.59.1 Detailed Description

SDF (p. 762) **Element** (p. 332) class.

10.59.2 Constructor & Destructor Documentation

10.59.2.1 sdf::Element::Element ( )

10.59.2.2 virtual sdf::Element::~Element ( ) [virtual]

10.59.3 Member Function Documentation

10.59.3.1 void sdf::Element::AddAttribute (const std::string & *_key,* const std::string & *_type,* const std::string & *_defaultvalue,* bool *_required,* const std::string & *_description* = " ")

10.59.3.2 ElementPtr sdf::Element::AddElement (const std::string & *_name*)

10.59.3.3 void sdf::Element::AddElementDescription (ElementPtr *_elem*)

Add a new element description.

10.59.3.4 void sdf::Element::AddValue (const std::string & *_type,* const std::string & *_defaultValue,* bool *_required,* const std::string & *_description* = " ")

10.59.3.5 void sdf::Element::ClearElements ( )

10.59.3.6 Element* sdf::Element::Clone ( ) const

10.59.3.7 void sdf::Element::Copy (const ElementPtr *_elem*)

Copy values from an **Element** (p. 332).

10.59.3.8 ParamPtr sdf::Element::GetAttribute (const std::string & _key)

Get the param of an attribute.

Parameters

_key	the name of the attribute

10.59.3.9 ParamPtr sdf::Element::GetAttribute (unsigned int _index) const

Get an attribute using an index.

10.59.3.10 unsigned int sdf::Element::GetAttributeCount ( ) const

Get the number of attributes.

10.59.3.11 bool sdf::Element::GetAttributeSet (const std::string & _key)

Return true if the attribute was set (i.e. not default value)

10.59.3.12 bool sdf::Element::GetCopyChildren ( ) const

10.59.3.13 std::string sdf::Element::GetDescription ( ) const

Get a text description of the element.

10.59.3.14 ElementPtr sdf::Element::GetElement (const std::string & _name) const

Referenced by gazebo::physics::Hinge2Joint< BulletJoint >::Load(), and gazebo::physics::ScrewJoint< BulletJoint >-::Load().

10.59.3.15 ElementPtr sdf::Element::GetElement (const std::string & _name)

10.59.3.16 ElementPtr sdf::Element::GetElementDescription (unsigned int _index) const

Get an element description using an index.

10.59.3.17 ElementPtr sdf::Element::GetElementDescription (const std::string & _key) const

Get an element descriptio using a key.

10.59.3.18 unsigned int sdf::Element::GetElementDescriptionCount ( ) const

Get the number of element descriptions.

10.59.3.19 **ElementPtr** sdf::Element::GetElementImpl (const std::string & *_name*) const

10.59.3.20 **ElementPtr** sdf::Element::GetFirstElement ( ) const

10.59.3.21 std::string sdf::Element::GetInclude ( ) const

10.59.3.22 const std::string& sdf::Element::GetName ( ) const

10.59.3.23 **ElementPtr** sdf::Element::GetNextElement (const std::string & *_name* = " ") const

10.59.3.24 **ElementPtr** sdf::Element::GetParent ( ) const

10.59.3.25 const std::string& sdf::Element::GetRequired ( ) const

10.59.3.26 **ParamPtr** sdf::Element::GetValue ( )

Get the param of the elements value.

10.59.3.27 bool sdf::Element::GetValueBool (const std::string & *_key* = " ")

10.59.3.28 char sdf::Element::GetValueChar (const std::string & *_key* = " ")

10.59.3.29 **gazebo::common::Color** sdf::Element::GetValueColor (const std::string & *_key* = " ")

10.59.3.30 double sdf::Element::GetValueDouble (const std::string & *_key* = " ")

Referenced by gazebo::physics::ScrewJoint< BulletJoint >::Load().

10.59.3.31 float sdf::Element::GetValueFloat (const std::string & *_key* = " ")

10.59.3.32 int sdf::Element::GetValueInt (const std::string & *_key* = " ")

10.59.3.33 **gazebo::math::Pose** sdf::Element::GetValuePose (const std::string & *_key* = " ")

10.59.3.34 **gazebo::math::Quaternion** sdf::Element::GetValueQuaternion (const std::string & *_key* = " ")

10.59.3.35 std::string sdf::Element::GetValueString (const std::string & *_key* = " ")

10.59.3.36 **gazebo::common::Time** sdf::Element::GetValueTime (const std::string & *_key* = " ")

10.59.3.37 unsigned int sdf::Element::GetValueUInt (const std::string & *_key* = " ")

10.59.3.38 **gazebo::math::Vector2d** sdf::Element::GetValueVector2d (const std::string & *_key* = " ")

10.59.3.39 **gazebo::math::Vector3** sdf::Element::GetValueVector3 (const std::string & *_key* = " ")

Referenced by gazebo::physics::Hinge2Joint< BulletJoint >::Load(), and gazebo::physics::ScrewJoint< BulletJoint >-::Load().

10.59.3.40 bool sdf::Element::HasAttribute (const std::string & *_key*)

10.59.3.41 bool sdf::Element::HasElement (const std::string & _name) const

Referenced by gazebo::physics::ScrewJoint< BulletJoint >::Load().

10.59.3.42 bool sdf::Element::HasElementDescription (const std::string & _name)

Return true if an element description exists.

10.59.3.43 void sdf::Element::InsertElement (ElementPtr _elem)

10.59.3.44 void sdf::Element::PrintDescription (std::string _prefix)

10.59.3.45 void sdf::Element::PrintDocLeftPane (std::string & _html, int _spacing, int & _index)

Helper function for **SDF::PrintDoc** (p. 763).

This generates the **SDF** (p. 762) html documentation.

Parameters

out	_html	Accumulated HTML for output.
in	_spacing	Amount of spacing for this element.
in	_index	Unique index for this element.

10.59.3.46 void sdf::Element::PrintDocRightPane (std::string & _html, int _spacing)

Helper function for **SDF::PrintDoc** (p. 763).

This generates the **SDF** (p. 762) html documentation.

Parameters

out	_html	Accumulated HTML for output
in	_spacing	Amount of spacing for this element.

10.59.3.47 void sdf::Element::PrintValues (std::string _prefix)

10.59.3.48 void sdf::Element::PrintWiki (std::string _prefix)

10.59.3.49 void sdf::Element::Reset ( )

10.59.3.50 bool sdf::Element::Set (const bool & _value)

10.59.3.51 bool sdf::Element::Set (const int & _value)

10.59.3.52 bool sdf::Element::Set (const unsigned int & _value)

10.59.3.53 bool sdf::Element::Set (const float & _value)

10.59.3.54 bool sdf::Element::Set (const double & _value)

10.59.3.55 bool sdf::Element::Set (const char & *_value*)

10.59.3.56 bool sdf::Element::Set (const std::string & *_value*)

10.59.3.57 bool sdf::Element::Set (const char * *_value*)

10.59.3.58 bool sdf::Element::Set (const **gazebo::math::Vector3** & *_value*)

10.59.3.59 bool sdf::Element::Set (const **gazebo::math::Vector2i** & *_value*)

10.59.3.60 bool sdf::Element::Set (const **gazebo::math::Vector2d** & *_value*)

10.59.3.61 bool sdf::Element::Set (const **gazebo::math::Quaternion** & *_value*)

10.59.3.62 bool sdf::Element::Set (const **gazebo::math::Pose** & *_value*)

10.59.3.63 bool sdf::Element::Set (const **gazebo::common::Color** & *_value*)

10.59.3.64 bool sdf::Element::Set (const **gazebo::common::Time** & *_value*)

10.59.3.65 void sdf::Element::SetCopyChildren (bool *_value*)

10.59.3.66 void sdf::Element::SetDescription (const std::string & *_desc*)

Set a text description for the element.

10.59.3.67 void sdf::Element::SetInclude (const std::string & *_filename*)

10.59.3.68 void sdf::Element::SetName (const std::string & *_name*)

10.59.3.69 void sdf::Element::SetParent (const **ElementPtr** *_parent*)

10.59.3.70 void sdf::Element::SetRequired (const std::string & *_req*)

10.59.3.71 std::string sdf::Element::ToString (const std::string & *_prefix*) const

10.59.3.72 void sdf::Element::Update ( )

The documentation for this class was generated from the following file:

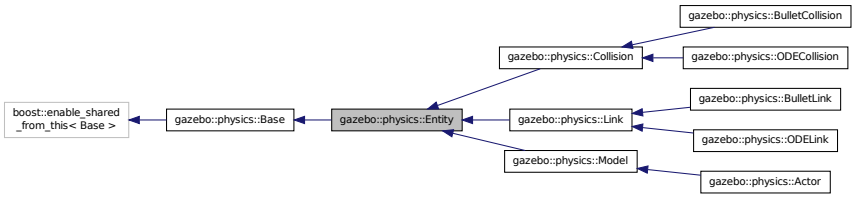
- **SDF.hh**

10.60 gazebo::physics::Entity Class Reference

Base (p. 145) class for all physics objects in Gazebo.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Entity:



Public Member Functions

- **Entity** (**BasePtr** _parent)

  Constructor.
- virtual ∼**Entity** ()

  Destructor.
- virtual void **Fini** ()

  Finalize the entity.
- virtual **math::Box GetBoundingBox** () const

  Return the bounding box for the entity.
- **CollisionPtr GetChildCollision** (const std::string &_name)

  Get a child collision entity, if one exists.
- **LinkPtr GetChildLink** (const std::string &_name)

  Get a child linke entity, if one exists.
- **math::Box GetCollisionBoundingBox** () const

  Returns collision bounding box.
- const **math::Pose** & **GetDirtyPose** () const

  *Returns **Entity::dirtyPose** (p. 349).*
- void **GetNearestEntityBelow** (double &_distBelow, std::string &_entityName)

  Get the distance to the nearest entity below (along the Z-axis) this entity.
- **ModelPtr GetParentModel** ()

  Get the parent model, if one exists.
- virtual **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the entity.
- virtual **math::Vector3 GetRelativeAngularVel** () const

  Get the angular velocity of the entity.
- virtual **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the entity.
- virtual **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the entity.
- **math::Pose GetRelativePose** () const

  Get the pose of the entity relative to its parent.
- virtual **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the entity in the world frame.
- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the entity in the world frame.

- virtual **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the entity in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the entity in the world frame.
- const **math::Pose** & **GetWorldPose** () const

  Get the absolute pose of the entity.
- bool **IsCanonicalLink** () const

  A helper function that checks if this is a canonical body.
- bool **IsStatic** () const

  Return whether this entity is static.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the entity.
- void **PlaceOnEntity** (const std::string &_entityName)

  Move this entity to be ontop of another entity by name.
- void **PlaceOnNearestEntityBelow** ()

  Move this entity to be ontop of the nearest entity below.
- virtual void **Reset** ()

  Reset the entity.
- void **SetAnimation** (const **common::PoseAnimationPtr** &_anim, boost::function< void()> _onComplete)

  Set an animation for this entity.
- void **SetAnimation** (**common::PoseAnimationPtr** _anim)

  Set an animation for this entity.
- void **SetCanonicalLink** (bool _value)

  Set to true if this entity is a canonical link for a model.
- void **SetInitialRelativePose** (const **math::Pose** &_pose)

  Set the initial pose.
- virtual void **SetName** (const std::string &_name)

  Set the name of the entity.
- void **SetRelativePose** (const **math::Pose** &_pose, bool _notify=true, bool _publish=true)

  Set the pose of the entity relative to its parent.
- void **SetStatic** (const bool &_static)

  Set whether this entity is static: immovable.
- void **SetWorldPose** (const **math::Pose** &_pose, bool _notify=true, bool _publish=true)

  Set the world pose of the entity.
- void **SetWorldTwist** (const **math::Vector3** &_linear, const **math::Vector3** &_angular, bool _updateChildren=true)

  *Set angular and linear rates of an **physics::Entity*** (p. 338).
- virtual void **StopAnimation** ()

  Stop the current animation, if any.
- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  Update the parameters using new sdf values.

Protected Member Functions

- virtual void **OnPoseChange** ()=0

  This function is called when the entity's (or one of its parents) pose of the parent has changed.

Protected Attributes

- **common::PoseAnimationPtr animation**

  Current pose animation.
- **event::ConnectionPtr animationConnection**

  Connection used to update an animation.
- **math::Pose animationStartPose**

  Start pose of an animation.
- std::vector< **event::ConnectionPtr** > **connections**

  All our event connections.
- **math::Pose dirtyPose**

  The pose set by a physics engine.
- **transport::NodePtr node**

  Communication node.
- **EntityPtr parentEntity**

  A helper that prevents numerous dynamic_casts.
- msgs::Pose * **poseMsg**

  Pose message containr.
- **common::Time prevAnimationTime**

  Previous time an animation was updated.
- **transport::PublisherPtr requestPub**

  Request publisher.
- **transport::PublisherPtr visPub**

  Visual publisher.
- msgs::Visual * **visualMsg**

  Visual message container.

Additional Inherited Members

10.60.1 Detailed Description

Base (p. 145) class for all physics objects in Gazebo.

10.60.2 Constructor & Destructor Documentation

10.60.2.1 gazebo::physics::Entity::Entity (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of the entity.

10.60.2.2 virtual gazebo::physics::Entity::~Entity ( ) [virtual]

Destructor.

10.60.3 Member Function Documentation

10.60.3.1 virtual void gazebo::physics::Entity::Fini ( ) [virtual]

Finalize the entity.

Reimplemented from **gazebo::physics::Base** (p. 150).

Reimplemented in **gazebo::physics::Actor** (p. 124), **gazebo::physics::Model** (p. 525), **gazebo::physics::Link** (p. 462), **gazebo::physics::ODECollision** (p. 580), **gazebo::physics::BulletLink** (p. 195), **gazebo::physics::ODELink** (p. 603), and **gazebo::physics::Collision** (p. 266).

10.60.3.2 virtual math::Box gazebo::physics::Entity::GetBoundingBox ( ) const [virtual]

Return the bounding box for the entity.

Returns

The bounding box.

Reimplemented in **gazebo::physics::Link** (p. 462), **gazebo::physics::Model** (p. 525), **gazebo::physics::Collision** (p. 266), **gazebo::physics::ODECollision** (p. 580), and **gazebo::physics::BulletCollision** (p. 173).

10.60.3.3 CollisionPtr gazebo::physics::Entity::GetChildCollision (const std::string & *_name*)

Get a child collision entity, if one exists.

Parameters

in	*_name*	Name of the child collision object.

Returns

Pointer to the **Collision** (p. 262) object, or NULL if not found.

10.60.3.4 LinkPtr gazebo::physics::Entity::GetChildLink (const std::string & *_name*)

Get a child linke entity, if one exists.

Parameters

in	*_name*	Name of the child **Link** (p. 454) object.

Returns

Pointer to the **Link** (p. 454) object, or NULL if not found.

10.60.3.5 math::Box gazebo::physics::Entity::GetCollisionBoundingBox ( ) const

Returns collision bounding box.

Returns

Collsiion boundin box.

10.60.3.6 const math::Pose& gazebo::physics::Entity::GetDirtyPose ( ) const

Returns **Entity::dirtyPose** (p. 349).

The dirty pose is the pose set by the physics engine before it's value is propagated to the rest of the simulator.

Returns

The dirty pose of the entity.

10.60.3.7 void gazebo::physics::Entity::GetNearestEntityBelow (double & *_distBelow,* std::string & *_entityName*)

Get the distance to the nearest entity below (along the Z-axis) this entity.

Parameters

out	*_distBelow*	The distance to the nearest entity below.
out	*_entityName*	The name of the nearest entity below.

10.60.3.8 ModelPtr gazebo::physics::Entity::GetParentModel ( )

Get the parent model, if one exists.

Returns

Pointer to a model, or NULL if no parent model exists.

10.60.3.9 virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularAccel ( ) const [inline], [virtual]

Get the angular acceleration of the entity.

Returns

A **math::Vector3** (p. 902) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 465), **gazebo::physics::Collision** (p. 267), and **gazebo::physics::Model** (p. 527).

10.60.3.10 virtual math::Vector3 gazebo::physics::Entity::GetRelativeAngularVel ( ) const [inline], [virtual]

Get the angular velocity of the entity.

Returns

A **math::Vector3** (p. 902) for the velocity.

Reimplemented in **gazebo::physics::Link** (p. 465), **gazebo::physics::Collision** (p. 268), and **gazebo::physics::Model** (p. 527).

10.60.3.11 virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearAccel ( ) const [inline], [virtual]

Get the linear acceleration of the entity.

Returns

A **math::Vector3** (p. 902) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 465), **gazebo::physics::Collision** (p. 268), and **gazebo::physics::Model** (p. 527).

10.60.3.12 virtual math::Vector3 gazebo::physics::Entity::GetRelativeLinearVel ( ) const [inline], [virtual]

Get the linear velocity of the entity.

Returns

A **math::Vector3** (p. 902) for the linear velocity.

Reimplemented in **gazebo::physics::Link** (p. 465), **gazebo::physics::Collision** (p. 268), and **gazebo::physics::Model** (p. 527).

10.60.3.13 math::Pose gazebo::physics::Entity::GetRelativePose ( ) const

Get the pose of the entity relative to its parent.

Returns

The pose of the entity relative to its parent.

10.60.3.14 virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularAccel ( ) const [inline], [virtual]

Get the angular acceleration of the entity in the world frame.

Returns

A **math::Vector3** (p. 902) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 467), **gazebo::physics::Collision** (p. 269), and **gazebo::physics::Model** (p. 528).

10.60.3.15 virtual math::Vector3 gazebo::physics::Entity::GetWorldAngularVel ( ) const [inline], [virtual]

Get the angular velocity of the entity in the world frame.

Returns

A **math::Vector3** (p. 902) for the velocity.

Reimplemented in **gazebo::physics::Collision** (p. 269), **gazebo::physics::Model** (p. 528), **gazebo::physics::ODELink** (p. 604), and **gazebo::physics::BulletLink** (p. 195).

10.60.3.16 virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearAccel ( ) const [inline], [virtual]

Get the linear acceleration of the entity in the world frame.

Returns

A **math::Vector3** (p. 902) for the acceleration.

Reimplemented in **gazebo::physics::Link** (p. 467), **gazebo::physics::Collision** (p. 269), and **gazebo::physics::Model** (p. 528).

10.60.3.17 virtual math::Vector3 gazebo::physics::Entity::GetWorldLinearVel ( ) const [inline], [virtual]

Get the linear velocity of the entity in the world frame.

Returns

A **math::Vector3** (p. 902) for the linear velocity.

Reimplemented in **gazebo::physics::Collision** (p. 269), **gazebo::physics::Model** (p. 529), **gazebo::physics::ODELink** (p. 604), and **gazebo::physics::BulletLink** (p. 196).

10.60.3.18 const math::Pose& gazebo::physics::Entity::GetWorldPose ( ) const [inline]

Get the absolute pose of the entity.

Returns

The absolute pose of the entity.

Referenced by gazebo::sensors::RFIDTag::GetTagPose().

10.60.3.19 bool gazebo::physics::Entity::IsCanonicalLink ( ) const [inline]

A helper function that checks if this is a canonical body.

Returns

True if the link is canonical.

10.60.3.20 bool gazebo::physics::Entity::IsStatic ( ) const

Return whether this entity is static.

Returns

True if static.

10.60.3.21 virtual void gazebo::physics::Entity::Load (sdf::ElementPtr _sdf) [virtual]

Load the entity.

Parameters

in	*_sdf*	Pointer to an SDF element.

Reimplemented from **gazebo::physics::Base** (p. 154).

Reimplemented in **gazebo::physics::Actor** (p. 124), **gazebo::physics::Model** (p. 529), **gazebo::physics::Link** (p. 468), **gazebo::physics::BulletCollision** (p. 173), **gazebo::physics::Collision** (p. 270), **gazebo::physics::BulletLink** (p. 196), **gazebo::physics::ODECollision** (p. 581), and **gazebo::physics::ODELink** (p. 605).

10.60.3.22 virtual void gazebo::physics::Entity::OnPoseChange ( ) [protected], [pure virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implemented in **gazebo::physics::Link** (p. 468), **gazebo::physics::Model** (p. 529), **gazebo::physics::ODECollision** (p. 581), **gazebo::physics::BulletLink** (p. 196), **gazebo::physics::BulletCollision** (p. 173), and **gazebo::physics::ODELink** (p. 605).

10.60.3.23 void gazebo::physics::Entity::PlaceOnEntity (const std::string & _entityName)

Move this entity to be ontop of another entity by name.

Parameters

in	*_entityName*	Name of the **Entity** (p. 338) this **Entity** (p. 338) should be ontop of.

10.60.3.24 void gazebo::physics::Entity::PlaceOnNearestEntityBelow ( )

Move this entity to be ontop of the nearest entity below.

10.60.3.25 virtual void gazebo::physics::Entity::Reset ( ) [virtual]

Reset the entity.

Reimplemented from **gazebo::physics::Base** (p. 155).

Reimplemented in **gazebo::physics::Model** (p. 530), and **gazebo::physics::Link** (p. 468).

10.60.3.26 void gazebo::physics::Entity::SetAnimation (const common::PoseAnimationPtr & _anim, boost::function< void() > _onComplete)

Set an animation for this entity.

Parameters

in	*_anim*	Pose animation.
in	*_onComplete*	Callback for when the animation completes.

10.60.3.27 void gazebo::physics::Entity::SetAnimation (common::PoseAnimationPtr _anim)

Set an animation for this entity.

Parameters

in	*_anim*	Pose animation.

10.60.3.28 void gazebo::physics::Entity::SetCanonicalLink (bool _value)

Set to true if this entity is a canonical link for a model.

Parameters

in	*_value*	True if the link is canonical.

10.60.3.29 void gazebo::physics::Entity::SetInitialRelativePose (const math::Pose & _pose)

Set the initial pose.

Parameters

in	*_pose*	The initial pose.

10.60.3.30 virtual void gazebo::physics::Entity::SetName (const std::string & _name) [virtual]

Set the name of the entity.

Parameters

in	*_name*	The new name.

Reimplemented from **gazebo::physics::Base** (p. 156).

10.60.3.31 void gazebo::physics::Entity::SetRelativePose (const math::Pose & _pose, bool _notify = true, bool _publish = true)

Set the pose of the entity relative to its parent.

Parameters

in	*_pose*	The new pose.
in	*_notify*	True = tell children of the pose change.
in	*_publish*	True to publish the pose.

10.60.3.32 void gazebo::physics::Entity::SetStatic (const bool & _static)

Set whether this entity is static: immovable.

Parameters

in	*_static*	True = static.

10.60.3.33 void gazebo::physics::Entity::SetWorldPose (const math::Pose & *_pose*, bool *_notify* = true, bool *_publish* = true)

Set the world pose of the entity.

Parameters

in	*_pose*	The new world pose.
in	*_notify*	True = tell children of the pose change.
in	*_publish*	True to publish the pose.

10.60.3.34 void gazebo::physics::Entity::SetWorldTwist (const math::Vector3 & *_linear*, const math::Vector3 & *_angular*, bool *_updateChildren* = true)

Set angular and linear rates of an **physics::Entity** (p. 338).

Parameters

in	*_linear*	Linear twist.
in	*_angular*	Angular twist.
in	*_updateChildren*	True to pass this update to child entities.

10.60.3.35 virtual void gazebo::physics::Entity::StopAnimation ( ) [virtual]

Stop the current animation, if any.

Reimplemented in **gazebo::physics::Model** (p. 533).

10.60.3.36 virtual void gazebo::physics::Entity::UpdateParameters (sdf::ElementPtr *_sdf*) [virtual]

Update the parameters using new sdf values.

Parameters

in	*_sdf*	SDF to update from.

Reimplemented from **gazebo::physics::Base** (p. 157).

Reimplemented in **gazebo::physics::Actor** (p. 125), **gazebo::physics::Model** (p. 533), **gazebo::physics::Link** (p. 472), and **gazebo::physics::Collision** (p. 272).

10.60.4 Member Data Documentation

10.60.4.1 common::PoseAnimationPtr gazebo::physics::Entity::animation [protected]

Current pose animation.

10.60.4.2 event::ConnectionPtr gazebo::physics::Entity::animationConnection [protected]

Connection used to update an animation.

10.60.4.3 math::Pose gazebo::physics::Entity::animationStartPose [protected]

Start pose of an animation.

10.60.4.4 std::vector<event::ConnectionPtr> gazebo::physics::Entity::connections [protected]

All our event connections.

10.60.4.5 math::Pose gazebo::physics::Entity::dirtyPose [protected]

The pose set by a physics engine.

10.60.4.6 transport::NodePtr gazebo::physics::Entity::node [protected]

Communication node.

10.60.4.7 EntityPtr gazebo::physics::Entity::parentEntity [protected]

A helper that prevents numerous dynamic_casts.

10.60.4.8 msgs::Pose* gazebo::physics::Entity::poseMsg [protected]

Pose message containr.

10.60.4.9 common::Time gazebo::physics::Entity::prevAnimationTime [protected]

Previous time an animation was updated.

10.60.4.10 transport::PublisherPtr gazebo::physics::Entity::requestPub [protected]

Request publisher.

10.60.4.11 transport::PublisherPtr gazebo::physics::Entity::visPub [protected]

Visual publisher.

10.60.4.12 msgs::Visual* gazebo::physics::Entity::visualMsg [protected]

Visual message container.

The documentation for this class was generated from the following file:

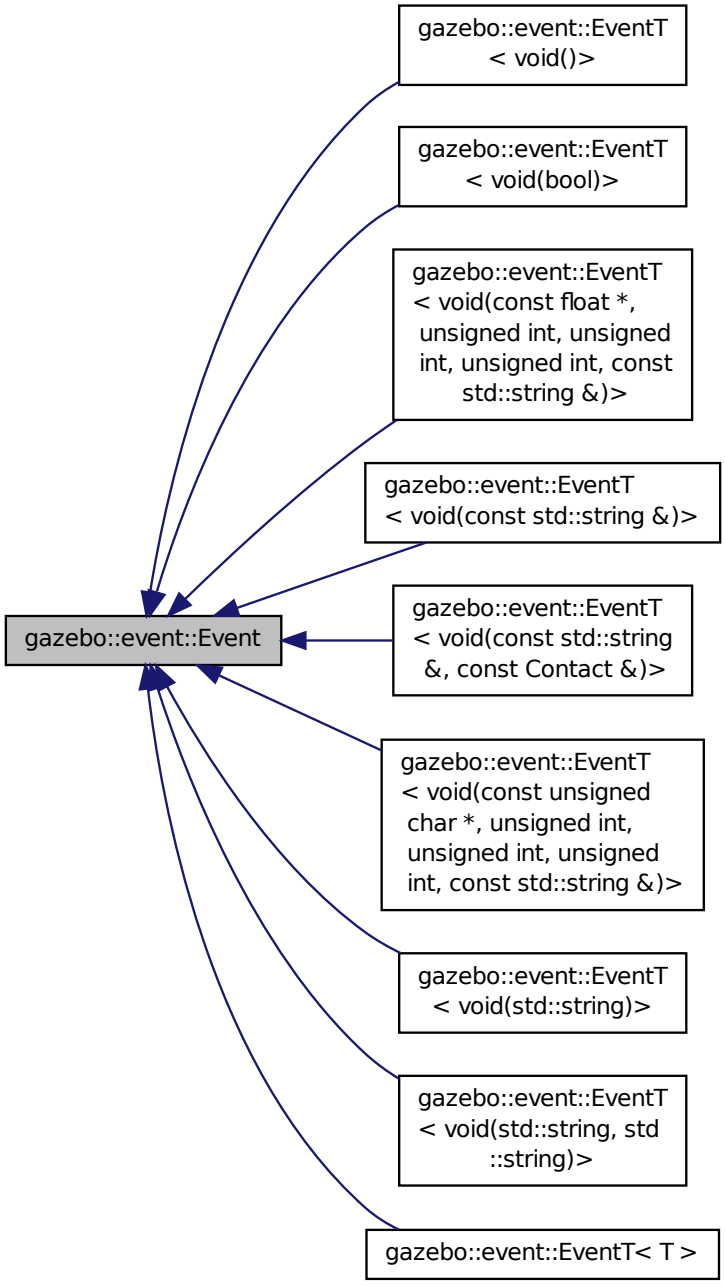
- **Entity.hh**

10.61 gazebo::event::Event Class Reference

Base class for all events.

`#include <Event.hh>`

Inheritance diagram for gazebo::event::Event:



Public Member Functions

- virtual ~**Event** ()

  Constructor.
- virtual void **Disconnect** (**ConnectionPtr** _c)=0

  Disconnect.
- virtual void **Disconnect** (int _id)=0

  Disconnect.

10.61.1 Detailed Description

Base class for all events.

10.61.2 Constructor & Destructor Documentation

10.61.2.1 virtual gazebo::event::Event::~Event ( ) [inline], [virtual]

Constructor.

10.61.3 Member Function Documentation

10.61.3.1 virtual void gazebo::event::Event::Disconnect (ConnectionPtr _c) [pure virtual]

Disconnect.

Parameters

in	*_c*	A pointer to a connection

Implemented in **gazebo::event::EventT< T >** (p. 46), **gazebo::event::EventT< void(std::string)>** (p. 46), **gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 46), **gazebo::event::EventT< void(const std::string &)>** (p. 46), **gazebo::event::EventT< void()>** (p. 46), **gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 46), **gazebo::event::EventT< void(const std::string &, const Contact &)>** (p. 46), **gazebo::event::EventT< void(std::string, std::string)>** (p. 46), and **gazebo::event::EventT< void(bool)>** (p. 46).

10.61.3.2 virtual void gazebo::event::Event::Disconnect (int _id) [pure virtual]

Disconnect.

Parameters

in	*_id*	Integer ID of a connection

Implemented in **gazebo::event::EventT< T >** (p. 47), **gazebo::event::EventT< void(std::string)>** (p. 47), **gazebo::event::EventT< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 47), **gazebo::event::EventT< void(const std::string &)>** (p. 47), **gazebo::event::EventT< void()>** (p. 47), **gazebo::event::EventT< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)>** (p. 47), **gazebo::event::EventT< void(const std::string &, const Contact &)>** (p. 47), **gazebo::event::EventT< void(std::string, std::string)>** (p. 47), and **gazebo::event::EventT< void(bool)>** (p. 47).

The documentation for this class was generated from the following file:

- **Event.hh**

10.62 gazebo::event::Events Class Reference

An **Event** (p. 350) class to get notifications for simulator events.

`#include <Events.hh>`

Static Public Member Functions

- template<typename T >
  static **ConnectionPtr ConnectAddEntity** (T _subscriber)

  Connect a boost::slot the the add entity signal.
- template<typename T >
  static **ConnectionPtr ConnectCreateEntity** (T _subscriber)

  Connect a boost::slot the the add entity signal.
- template<typename T >
  static **ConnectionPtr ConnectDeleteEntity** (T _subscriber)

  Connect a boost::slot the delete entity.
- template<typename T >
  static **ConnectionPtr ConnectDiagTimerStart** (T _subscriber)

  Connect a boost::slot the diagnostic timer start signal.
- template<typename T >
  static **ConnectionPtr ConnectDiagTimerStop** (T _subscriber)

  Connect a boost::slot the diagnostic timer stop signal.
- template<typename T >
  static **ConnectionPtr ConnectPause** (T _subscriber)

  Connect a boost::slot the the pause signal.
- template<typename T >
  static **ConnectionPtr ConnectPostRender** (T _subscriber)

  Connect a boost::slot the post render update signal.
- template<typename T >
  static **ConnectionPtr ConnectPreRender** (T _subscriber)

  Render start signal.
- template<typename T >
  static **ConnectionPtr ConnectRender** (T _subscriber)

  Connect a boost::slot the render update signal.
- template<typename T >
  static **ConnectionPtr ConnectSetSelectedEntity** (T _subscriber)

  Connect a boost::slot the set selected entity.
- template<typename T >
  static **ConnectionPtr ConnectStep** (T _subscriber)

  Connect a boost::slot the the step signal.
- template<typename T >
  static **ConnectionPtr ConnectStop** (T _subscriber)

  Connect a boost::slot the the stop signal.

- template<typename T >
  static **ConnectionPtr ConnectWorldCreated** (T _subscriber)

  Connect a boost::slot the the world created signal.
- template<typename T >
  static **ConnectionPtr ConnectWorldUpdateEnd** (T _subscriber)

  Connect a boost::slot the the world update end signal.
- template<typename T >
  static **ConnectionPtr ConnectWorldUpdateStart** (T _subscriber)

  Connect a boost::slot the the world update start signal.
- static void **DisconnectAddEntity** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the add entity signal.
- static void **DisconnectCreateEntity** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the add entity signal.
- static void **DisconnectDeleteEntity** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the delete entity.
- static void **DisconnectDiagTimerStart** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the diagnostic timer start signal.
- static void **DisconnectDiagTimerStop** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the diagnostic timer stop signal.
- static void **DisconnectPause** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the pause signal.
- static void **DisconnectPostRender** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the post render update signal.
- static void **DisconnectPreRender** (**ConnectionPtr** _subscriber)

  Disconnect a render start signal.
- static void **DisconnectRender** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the render update signal.
- static void **DisconnectSetSelectedEntity** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the set selected entity.
- static void **DisconnectStep** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the step signal.
- static void **DisconnectStop** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the stop signal.
- static void **DisconnectWorldCreated** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the world created signal.
- static void **DisconnectWorldUpdateEnd** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the world update end signal.
- static void **DisconnectWorldUpdateStart** (**ConnectionPtr** _subscriber)

  Disconnect a boost::slot the the world update start signal.

Static Public Attributes

- static **EventT**< void(std::string)> **addEntity**

  An entity has been added.
- static **EventT**< void(std::string)> **deleteEntity**

  An entity has been deleted.
- static **EventT**< void(std::string)> **diagTimerStart**

Diagnostic timer start.

- static **EventT**< void(std::string)> **diagTimerStop**

  Diagnostic timer stop.
- static **EventT**< void(std::string)> **entityCreated**

  An entity has been created.
- static **EventT**< void(bool)> **pause**

  Pause signal.
- static **EventT**< void()> **postRender**

  Post-Render.
- static **EventT**< void()> **preRender**

  Pre-render.
- static **EventT**< void()> **render**

  Render.
- static **EventT**< void(std::string, std::string)> **setSelectedEntity**

  An entity has been selected.
- static **EventT**< void()> **step**

  Step the simulation once signal.
- static **EventT**< void()> **stop**

  Simulation stop signal.
- static **EventT**< void(std::string)> **worldCreated**

  A world has been created.
- static **EventT**< void()> **worldUpdateEnd**

  World update has ended.
- static **EventT**< void()> **worldUpdateStart**

  World update has started.

10.62.1 Detailed Description

An **Event** (p. 350) class to get notifications for simulator events.

The documentation for this class was generated from the following file:

- **Events.hh**

10.63 gazebo::rendering::Events Class Reference

Base class for rendering events.

`#include <rendering/rendering.hh>`

Static Public Member Functions

- template<typename T >
  static **event::ConnectionPtr ConnectCreateScene** (T _subscriber)

  Connect to a scene created event.
- template<typename T >
  static **event::ConnectionPtr ConnectRemoveScene** (T _subscriber)

Connect to a scene removed event.

- static void **DisconnectCreateScene** (**event::ConnectionPtr** _connection)

  Disconnect from a scene created event.

- static void **DisconnectRemoveScene** (**event::ConnectionPtr** _connection)

  Disconnect from a scene removed event.

Static Public Attributes

- static **event::EventT**< void(const std::string &)> **createScene**

  The event used to trigger a create scene event.

- static **event::EventT**< void(const std::string &)> **removeScene**

  The event used to trigger a remve scene event.

10.63.1 Detailed Description

Base class for rendering events.

10.63.2 Member Function Documentation

10.63.2.1 template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectCreateScene (T *_subscriber*) [inline], [static]

Connect to a scene created event.

Parameters

in	*_subscriber*	Callback to trigger when event occurs.

Returns

Pointer the connection. This must stay in scope.

References gazebo::event::EventT< T >::Connect(), and createScene.

10.63.2.2 template<typename T > static event::ConnectionPtr gazebo::rendering::Events::ConnectRemoveScene (T *_subscriber*) [inline], [static]

Connect to a scene removed event.

Parameters

in	*_subscriber*	Callback to trigger when event occurs.

Returns

Pointer the connection. This must stay in scope.

References gazebo::event::EventT< T >::Connect(), and removeScene.

10.63.2.3 static void gazebo::rendering::Events::DisconnectCreateScene (event::ConnectionPtr _connection) [inline], [static]

Disconnect from a scene created event.

Parameters

in	*_connection*	The connection to disconnect.

References createScene, and gazebo::event::EventT< T >::Disconnect().

10.63.2.4 static void gazebo::rendering::Events::DisconnectRemoveScene (event::ConnectionPtr _connection) [inline], [static]

Disconnect from a scene removed event.

Parameters

in	*_connection*	The connection to disconnect.

References gazebo::event::EventT< T >::Disconnect(), and removeScene.

10.63.3 Member Data Documentation

10.63.3.1 event::EventT<void (const std::string &)> gazebo::rendering::Events::createScene [static]

The event used to trigger a create scene event.

Referenced by ConnectCreateScene(), and DisconnectCreateScene().

10.63.3.2 event::EventT<void (const std::string &)> gazebo::rendering::Events::removeScene [static]

The event used to trigger a remve scene event.

Referenced by ConnectRemoveScene(), and DisconnectRemoveScene().

The documentation for this class was generated from the following file:

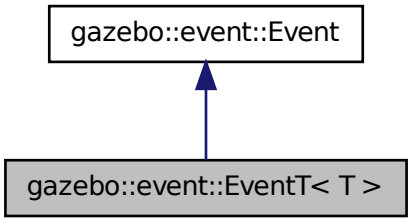
- **RenderEvents.hh**

10.64 gazebo::event::EventT< T > Class Template Reference

A class for event processing.

`#include <Event.hh>`

Inheritance diagram for gazebo::event::EventT< T >:



Public Member Functions

- virtual ~**EventT** ()

  Destructor.
- **ConnectionPtr Connect** (const boost::function< T > &_subscriber)

  Connect a callback to this event.
- virtual void **Disconnect** (**ConnectionPtr** _c)

  Disconnect a callback to this event.
- virtual void **Disconnect** (int _id)

  Disconnect a callback to this event.
- void **operator()** ()

  Access the signal.
- template<typename P >
  void **operator()** (const P &_p)

  Signal the event with one parameter.
- template<typename P1 , typename P2 >
  void **operator()** (const P1 &_p1, const P2 &_p2)

  Signal the event with two parameters.
- template<typename P1 , typename P2 , typename P3 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3)

  Signal the event with three parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4)

  Signal the event with four parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5)

  Signal the event with five parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6)

  Signal the event with six parameters.
- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7)

Signal the event with seven parameters.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8)

  Signal the event with eight parameters.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)

  Signal the event with nine parameters.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >
  void **operator()** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)

  Signal the event with ten parameters.

- void **Signal** ()

  Signal the event for all subscribers.

- template<typename P >
  void **Signal** (const P &_p)

  Signal the event with one parameter.

- template<typename P1 , typename P2 >
  void **Signal** (const P1 &_p1, const P2 &_p2)

  Signal the event with two parameter.

- template<typename P1 , typename P2 , typename P3 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3)

  Signal the event with three parameter.

- template<typename P1 , typename P2 , typename P3 , typename P4 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4)

  Signal the event with four parameter.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5)

  Signal the event with five parameter.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6)

  Signal the event with six parameter.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7)

  Signal the event with seven parameter.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8)

  Signal the event with eight parameter.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9)

  Signal the event with nine parameter.

- template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 >
  void **Signal** (const P1 &_p1, const P2 &_p2, const P3 &_p3, const P4 &_p4, const P5 &_p5, const P6 &_p6, const P7 &_p7, const P8 &_p8, const P9 &_p9, const P10 &_p10)

  Signal the event with ten parameter.

10.64.1 Detailed Description

template<typename T>class gazebo::event::EventT< T >

A class for event processing.

10.64.2 Member Function Documentation

10.64.2.1 template<typename T> void gazebo::event::EventT< T >::operator() ( ) [inline]

Access the signal.

10.64.2.2 template<typename T> template<typename P > void gazebo::event::EventT< T >::operator() (const P & *_p*) [inline]

Signal the event with one parameter.

Parameters

in	*_p*	the parameter

10.64.2.3 template<typename T> template<typename P1 , typename P2 > void gazebo::event::EventT< T >::operator() (const P1 & *_p1,* const P2 & *_p2*) [inline]

Signal the event with two parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter

10.64.2.4 template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::operator() (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3*) [inline]

Signal the event with three parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter

10.64.2.5 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4) [inline]

Signal the event with four parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter

10.64.2.6 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5) [inline]

Signal the event with five parameters.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fift parameter

10.64.2.7 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6) [inline]

Signal the event with six parameters.

Parameters

in	*_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

10.64.2.8 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7) [inline]

Signal the event with seven parameters.

Parameters

in	*_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

10.64.2.9 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 > void gazebo::event::EventT< T >::operator() (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6,* const P7 & *_p7,* const P8 & *_p8*) [inline]

Signal the event with eight parameters.

Parameters

in	*_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

10.64.2.10 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 > void gazebo::event::EventT< T >::operator() (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6,* const P7 & *_p7,* const P8 & *_p8,* const P9 & *_p9*) [inline]

Signal the event with nine parameters.

Parameters

in	*_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

10.64.2.11 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 > void gazebo::event::EventT< T >::operator() (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9, const P10 & _p10) [inline]

Signal the event with ten parameters.

Parameters

in	_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

10.64.2.12 template<typename T> void gazebo::event::EventT< T >::Signal ( ) [inline]

Signal the event for all subscribers.

Referenced by gazebo::event::EventT< void(bool)>::operator()().

10.64.2.13 template<typename T> template<typename P > void gazebo::event::EventT< T >::Signal (const P & _p) [inline]

Signal the event with one parameter.

Parameters

in	_p	parameter

10.64.2.14 template<typename T> template<typename P1 , typename P2 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2) [inline]

Signal the event with two parameter.

Parameters

in	_p1	the first parameter
in	_p2	the second parameter

10.64.2.15 template<typename T> template<typename P1 , typename P2 , typename P3 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3) [inline]

Signal the event with three parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter

10.64.2.16 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 > void gazebo::event::EventT< T >::Signal (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4*) [inline]

Signal the event with four parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter

10.64.2.17 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 > void gazebo::event::EventT< T >::Signal (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5*) [inline]

Signal the event with five parameter.

Parameters

in	*_p1*	the first parameter
in	*_p2*	the second parameter
in	*_p3*	the second parameter
in	*_p4*	the first parameter
in	*_p5*	the fifth parameter

10.64.2.18 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 > void gazebo::event::EventT< T >::Signal (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6*) [inline]

Signal the event with six parameter.

Parameters

in	*_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

10.64.2.19 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7) [inline]

Signal the event with seven parameter.

Parameters

in	_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

10.64.2.20 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8) [inline]

Signal the event with eight parameter.

Parameters

in	_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

10.64.2.21 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 > void gazebo::event::EventT< T >::Signal (const P1 & _p1, const P2 & _p2, const P3 & _p3, const P4 & _p4, const P5 & _p5, const P6 & _p6, const P7 & _p7, const P8 & _p8, const P9 & _p9) [inline]

Signal the event with nine parameter.

Parameters

in	_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

10.64.2.22 template<typename T> template<typename P1 , typename P2 , typename P3 , typename P4 , typename P5 , typename P6 , typename P7 , typename P8 , typename P9 , typename P10 > void gazebo::event::EventT< T >::Signal (const P1 & *_p1,* const P2 & *_p2,* const P3 & *_p3,* const P4 & *_p4,* const P5 & *_p5,* const P6 & *_p6,* const P7 & *_p7,* const P8 & *_p8,* const P9 & *_p9,* const P10 & *_p10*) [inline]

Signal the event with ten parameter.

Parameters

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**

10.65 gazebo::common::Exception Class Reference

Class for generating exceptions.

`#include <Exception.hh>`

Public Member Functions

- **Exception** ()

  Constructor.
- **Exception** (const char *file, int line, std::string msg)

  Default constructor.
- virtual ~**Exception** ()

  Destructor.
- std::string **GetErrorFile** () const

  Return the error function.
- std::string **GetErrorStr** () const

  Return the error string.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Exception** &_err)

  stream insertion operator for Gazebo Error

10.65.1 Detailed Description

Class for generating exceptions.

10.65.2 Constructor & Destructor Documentation

10.65.2.1 gazebo::common::Exception::Exception ( )

Constructor.

10.65.2.2 gazebo::common::Exception::Exception (const char * *file*, int *line*, std::string *msg*)

Default constructor.

Parameters

in	*file*	File name
in	*line*	Line number where the error occurred
in	*msg*	Error message

10.65.2.3 virtual gazebo::common::Exception::~Exception ( ) [virtual]

Destructor.

10.65.3 Member Function Documentation

10.65.3.1 std::string gazebo::common::Exception::GetErrorFile ( ) const

Return the error function.

Returns

The error function name

10.65.3.2 std::string gazebo::common::Exception::GetErrorStr ( ) const

Return the error string.

Returns

The error string

10.65.4 Friends And Related Function Documentation

10.65.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::common::Exception & *_err*) [friend]

stream insertion operator for Gazebo Error

Parameters

in	*_out*	the output stream
in	*_err*	the exception

The documentation for this class was generated from the following file:

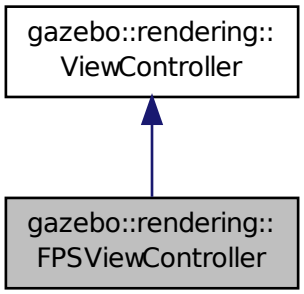
- **Exception.hh**

10.66 gazebo::rendering::FPSViewController Class Reference

First Person Shooter style view controller.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::FPSViewController:



Public Member Functions

- **FPSViewController** (**UserCameraPtr** _camera)

  Constructor.
- virtual ~**FPSViewController** ()

  Destructor.
- void **HandleKeyPressEvent** (const std::string &_key)

  Handle a key press event.
- void **HandleKeyReleaseEvent** (const std::string &_key)

  Handle a key release event.
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)

  Handle a mouse event.
- virtual void **Init** ()

  Initialize the controller.
- virtual void **Update** ()

  Update the camera position.

Static Public Member Functions

- static std::string **GetTypeString** ()

  Get the type name of this view controller.

Additional Inherited Members

10.66.1 Detailed Description

First Person Shooter style view controller.

10.66.2 Constructor & Destructor Documentation

10.66.2.1 gazebo::rendering::FPSViewController::FPSViewController (UserCameraPtr *_camera*)

Constructor.

Parameters

in	***Camera*** *(p. 233)*	to controll

10.66.2.2 virtual gazebo::rendering::FPSViewController::~FPSViewController ( ) [virtual]

Destructor.

10.66.3 Member Function Documentation

10.66.3.1 static std::string gazebo::rendering::FPSViewController::GetTypeString ( ) [static]

Get the type name of this view controller.

Returns

The name of the controller type: "fps"

10.66.3.2 void gazebo::rendering::FPSViewController::HandleKeyPressEvent (const std::string & *_key*) [virtual]

Handle a key press event.

Parameters

in	*_key*	The key that was pressed.

Implements **gazebo::rendering::ViewController** (p. 929).

10.66.3.3 void gazebo::rendering::FPSViewController::HandleKeyReleaseEvent (const std::string & *_key*) [virtual]

Handle a key release event.

Parameters

in	*_key*	The key that was released.

Implements **gazebo::rendering::ViewController** (p. 929).

10.66.3.4 virtual void gazebo::rendering::FPSViewController::HandleMouseEvent (const common::MouseEvent & *_event*) [virtual]

Handle a mouse event.

Parameters

in	*_event*	The mouse position.

Implements **gazebo::rendering::ViewController** (p. 929).

10.66.3.5 virtual void gazebo::rendering::FPSViewController::Init ( ) [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 930).

10.66.3.6 virtual void gazebo::rendering::FPSViewController::Update ( ) [virtual]

Update the camera position.

Implements **gazebo::rendering::ViewController** (p. 930).

The documentation for this class was generated from the following file:

- **FPSViewController.hh**

10.67 urdf2gazebo::GazeboExtension Class Reference

`#include <parser_urdf.hh>`

Public Member Functions

- **GazeboExtension** ()
- **GazeboExtension** (const **GazeboExtension** &ge)

Public Attributes

- std::vector< TiXmlElement * > **blobs**
- double **damping_factor**
- std::string **fdir1**
- double **fudge_factor**
- bool **gravity**
- double **initial_joint_position**

- bool **is_damping_factor**
- bool **is_fudge_factor**
- bool **is_initial_joint_position**
- bool **is_kd**
- bool **is_kp**
- bool **is_laser_retro**
- bool **is_maxVel**
- bool **is_minDepth**
- bool **is_mu1**
- bool **is_mu2**
- bool **is_stop_cfm**
- bool **is_stop_erp**
- double **kd**
- double **kp**
- double **laser_retro**
- std::string **material**
- double **maxVel**
- double **minDepth**
- double **mu1**
- double **mu2**
- std::string **old_link_name**
- bool **provideFeedback**
- **gazebo::math::Pose reduction_transform**
- bool **self_collide**
- bool **setStaticFlag**
- double **stop_cfm**
- double **stop_erp**

10.67.1 Constructor & Destructor Documentation

10.67.1.1 urdf2gazebo::GazeboExtension::GazeboExtension ( ) [inline]

References blobs, damping_factor, fdir1, fudge_factor, gravity, initial_joint_position, is_damping_factor, is_fudge_factor, is_initial_joint_position, is_kd, is_kp, is_laser_retro, is_maxVel, is_minDepth, is_mu1, is_mu2, is_stop_cfm, is_stop_erp, kd, kp, laser_retro, material, maxVel, minDepth, mu1, mu2, provideFeedback, self_collide, setStaticFlag, stop_cfm, and stop_erp.

10.67.1.2 urdf2gazebo::GazeboExtension::GazeboExtension (const GazeboExtension & *ge*) [inline]

References blobs, damping_factor, fdir1, fudge_factor, gravity, initial_joint_position, is_damping_factor, is_fudge_factor, is_initial_joint_position, is_kd, is_kp, is_laser_retro, is_maxVel, is_minDepth, is_mu1, is_mu2, is_stop_cfm, is_stop_erp, kd, kp, laser_retro, material, maxVel, minDepth, mu1, mu2, old_link_name, provideFeedback, reduction_transform, self_collide, setStaticFlag, stop_cfm, and stop_erp.

10.67.2 Member Data Documentation

10.67.2.1 std::vector<TiXmlElement*> urdf2gazebo::GazeboExtension::blobs

Referenced by GazeboExtension().

10.67.2.2 double urdf2gazebo::GazeboExtension::damping_factor

Referenced by GazeboExtension().

10.67.2.3 std::string urdf2gazebo::GazeboExtension::fdir1

Referenced by GazeboExtension().

10.67.2.4 double urdf2gazebo::GazeboExtension::fudge_factor

Referenced by GazeboExtension().

10.67.2.5 bool urdf2gazebo::GazeboExtension::gravity

Referenced by GazeboExtension().

10.67.2.6 double urdf2gazebo::GazeboExtension::initial_joint_position

Referenced by GazeboExtension().

10.67.2.7 bool urdf2gazebo::GazeboExtension::is_damping_factor

Referenced by GazeboExtension().

10.67.2.8 bool urdf2gazebo::GazeboExtension::is_fudge_factor

Referenced by GazeboExtension().

10.67.2.9 bool urdf2gazebo::GazeboExtension::is_initial_joint_position

Referenced by GazeboExtension().

10.67.2.10 bool urdf2gazebo::GazeboExtension::is_kd

Referenced by GazeboExtension().

10.67.2.11 bool urdf2gazebo::GazeboExtension::is_kp

Referenced by GazeboExtension().

10.67.2.12 bool urdf2gazebo::GazeboExtension::is_laser_retro

Referenced by GazeboExtension().

10.67.2.13 bool urdf2gazebo::GazeboExtension::is_maxVel

Referenced by GazeboExtension().

10.67.2.14 bool urdf2gazebo::GazeboExtension::is_minDepth

Referenced by GazeboExtension().

10.67.2.15 bool urdf2gazebo::GazeboExtension::is_mu1

Referenced by GazeboExtension().

10.67.2.16 bool urdf2gazebo::GazeboExtension::is_mu2

Referenced by GazeboExtension().

10.67.2.17 bool urdf2gazebo::GazeboExtension::is_stop_cfm

Referenced by GazeboExtension().

10.67.2.18 bool urdf2gazebo::GazeboExtension::is_stop_erp

Referenced by GazeboExtension().

10.67.2.19 double urdf2gazebo::GazeboExtension::kd

Referenced by GazeboExtension().

10.67.2.20 double urdf2gazebo::GazeboExtension::kp

Referenced by GazeboExtension().

10.67.2.21 double urdf2gazebo::GazeboExtension::laser_retro

Referenced by GazeboExtension().

10.67.2.22 std::string urdf2gazebo::GazeboExtension::material

Referenced by GazeboExtension().

10.67.2.23 double urdf2gazebo::GazeboExtension::maxVel

Referenced by GazeboExtension().

10.67.2.24 double urdf2gazebo::GazeboExtension::minDepth

Referenced by GazeboExtension().

10.67.2.25 double urdf2gazebo::GazeboExtension::mu1

Referenced by GazeboExtension().

10.67.2.26 double urdf2gazebo::GazeboExtension::mu2

Referenced by GazeboExtension().

10.67.2.27 std::string urdf2gazebo::GazeboExtension::old_link_name

Referenced by GazeboExtension().

10.67.2.28 bool urdf2gazebo::GazeboExtension::provideFeedback

Referenced by GazeboExtension().

10.67.2.29 gazebo::math::Pose urdf2gazebo::GazeboExtension::reduction_transform

Referenced by GazeboExtension().

10.67.2.30 bool urdf2gazebo::GazeboExtension::self_collide

Referenced by GazeboExtension().

10.67.2.31 bool urdf2gazebo::GazeboExtension::setStaticFlag

Referenced by GazeboExtension().

10.67.2.32 double urdf2gazebo::GazeboExtension::stop_cfm

Referenced by GazeboExtension().

10.67.2.33 double urdf2gazebo::GazeboExtension::stop_erp

Referenced by GazeboExtension().

The documentation for this class was generated from the following file:

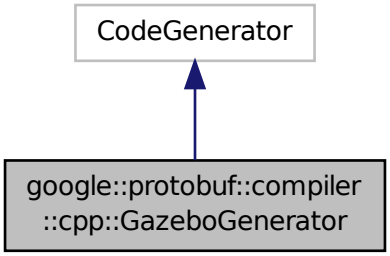
- **parser_urdf.hh**

10.68 google::protobuf::compiler::cpp::GazeboGenerator Class Reference

Google protobuf message generator for **gazebo::msgs** (p. 101).

`#include <GazeboGenerator.hh>`

Inheritance diagram for google::protobuf::compiler::cpp::GazeboGenerator:



Public Member Functions

- **GazeboGenerator** (const std::string &_name)
- virtual ~**GazeboGenerator** ()
- virtual bool **Generate** (const FileDescriptor *file, const string &parameter, OutputDirectory *directory, string *error) const

10.68.1 Detailed Description

Google protobuf message generator for **gazebo::msgs** (p. 101).

10.68.2 Constructor & Destructor Documentation

10.68.2.1 google::protobuf::compiler::cpp::GazeboGenerator::GazeboGenerator (const std::string & *_name*)

10.68.2.2 virtual google::protobuf::compiler::cpp::GazeboGenerator::~GazeboGenerator ( ) [virtual]

10.68.3 Member Function Documentation

10.68.3.1 virtual bool google::protobuf::compiler::cpp::GazeboGenerator::Generate (const FileDescriptor * *file,* const string & *parameter,* OutputDirectory * *directory,* string * *error*) const [virtual]

The documentation for this class was generated from the following file:

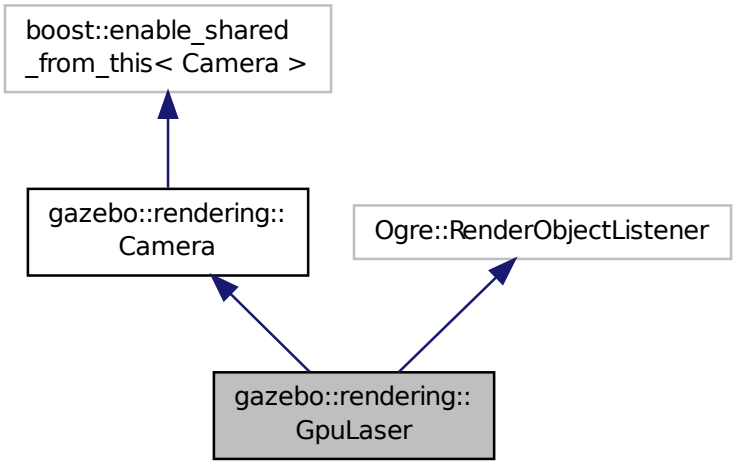
- **GazeboGenerator.hh**

10.69 gazebo::rendering::GpuLaser Class Reference

GPU based laser distance sensor.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::GpuLaser:



Public Member Functions

- **GpuLaser** (const std::string &_namePrefix, **Scene** *_scene, bool _autoRender=true)

  Constructor.
- virtual ~**GpuLaser** ()

  Destructor.
- template<typename T >
  event::ConnectionPtr ConnectNewLaserFrame (T _subscriber)

  Connect to a laser frame signal.
- void **CreateLaserTexture** (const std::string &_textureName)

  Create the texture which is used to render laser data.
- void **DisconnectNewLaserFrame** (**event::ConnectionPtr** &_c)

  Disconnect from a laser frame signal.
- void **Fini** ()

  Finalize the camera.
- virtual const float * **GetLaserData** ()

  All things needed to get back z buffer for laser data.
- void **Init** ()

  Initialize the camera.
- void **Load** (**sdf::ElementPtr** &_sdf)

  Load the camera with a set of parmeters.

- 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** ()

  Render the camera.
- void **SetParentSensor** (**sensors::GpuRaySensor** ∗_parent)

  Set the parent sensor.
- void **SetRangeCount** (unsigned int _w, unsigned int _h=1)

  Set the number of laser samples in the width and height.

Additional Inherited Members

10.69.1 Detailed Description

GPU based laser distance sensor.

This is the base class for all cameras.

10.69.2 Constructor & Destructor Documentation

10.69.2.1 gazebo::rendering::GpuLaser::GpuLaser (const std::string & *_namePrefix,* **Scene** ∗ *_scene,* bool *_autoRender* = `true`)

Constructor.

Parameters

in	*_namePrefix*	Unique prefix name for the camera.
in	*_scene*	**Scene** (p. 746) that will contain the camera
in	*_autoRender*	Almost everyone should leave this as true.

10.69.2.2 virtual gazebo::rendering::GpuLaser::∼GpuLaser ( ) `[virtual]`

Destructor.

10.69.3 Member Function Documentation

10.69.3.1 template<typename T > event::ConnectionPtr gazebo::rendering::GpuLaser::ConnectNewLaserFrame (T *_subscriber*) `[inline]`

Connect to a laser frame signal.

Parameters

in	*_subscriber*	Callback that is called when a new image is generated

Returns

A pointer to the connection. This must be kept in scope.

References gazebo::event::EventT< T >::Connect().

10.69.3.2 void gazebo::rendering::GpuLaser::CreateLaserTexture (const std::string & _textureName)

Create the texture which is used to render laser data.

Parameters

in	*_textureName*	Name of the new texture

10.69.3.3 void gazebo::rendering::GpuLaser::DisconnectNewLaserFrame (event::ConnectionPtr & _c) [inline]

Disconnect from a laser frame signal.

Parameters

in	*_c*	The connection to disconnect

References gazebo::event::EventT< T >::Disconnect().

10.69.3.4 void gazebo::rendering::GpuLaser::Fini ( )

Finalize the camera.

10.69.3.5 virtual const float* gazebo::rendering::GpuLaser::GetLaserData ( ) [virtual]

All things needed to get back z buffer for laser data.

Returns

Array of laser data

10.69.3.6 void gazebo::rendering::GpuLaser::Init ( )

Initialize the camera.

10.69.3.7 void gazebo::rendering::GpuLaser::Load (sdf::ElementPtr & _sdf)

Load the camera with a set of parmeters.

Parameters

in	*_sdf*	The SDF camera info

10.69.3.8 void gazebo::rendering::GpuLaser::Load ( )

Load the camera with default parmeters.

10.69.3.9 virtual void gazebo::rendering::GpuLaser::notifyRenderSingleObject (Ogre::Renderable * *_rend*, const Ogre::Pass * *_p*, const Ogre::AutoParamDataSource * *_s*, const Ogre::LightList * *_ll*, bool *_supp*) [virtual]

10.69.3.10 virtual void gazebo::rendering::GpuLaser::PostRender ( ) [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 248).

10.69.3.11 void gazebo::rendering::GpuLaser::SetParentSensor (sensors::GpuRaySensor * *_parent*)

Set the parent sensor.

Parameters

in	*_parent*	Pointer to a **sensors::GpuRaySensor** (p. 378)

10.69.3.12 void gazebo::rendering::GpuLaser::SetRangeCount (unsigned int *_w*, unsigned int *_h* = 1)

Set the number of laser samples in the width and height.

Parameters

in	*_w*	Number of samples in the horizontal sweep
in	*_h*	Number of samples in the vertical sweep

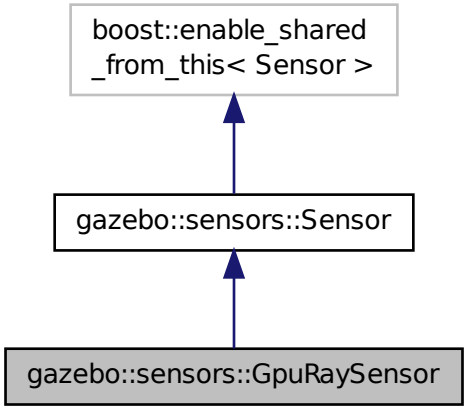
The documentation for this class was generated from the following file:

- **GpuLaser.hh**

10.70 gazebo::sensors::GpuRaySensor Class Reference

```
#include <sensors/sensors.hh>
```

Inheritance diagram for gazebo::sensors::GpuRaySensor:



Public Member Functions

- **GpuRaySensor** ()

  Constructor.
- virtual ~**GpuRaySensor** ()

  Destructor.
- **event::ConnectionPtr ConnectNewLaserFrame** (boost::function< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)> subscriber)

  Connect a to the add entity signal Nate do these parameters need to be specified here?
- void **DisconnectNewLaserFrame** (**event::ConnectionPtr** &c)

  Disconnect Laser Frame.
- double **Get1stRatio** ()

  Nate fill in.
- double **Get2ndRatio** ()
- **math::Angle GetAngleMax** () const

  Get the maximum angle.
- **math::Angle GetAngleMin** () const

  Get the minimum angle.
- double **GetAngleResolution** () const

  Get radians between each range.
- unsigned int **GetCameraCount** ()

  Gets the camera count.
- double **GetCHFOV** ()
- double **GetCVFOV** ()
- int **GetFiducial** (int _index)

  Get detected fiducial value for a ray.
- double **GetHAngle** ()

- double **GetHFOV** ()
- **rendering::GpuLaserPtr GetLaserCamera** () const

  *Returns a pointer to the internally kept **rendering::GpuLaser** (p. 375).*
- double **GetRange** (int _index)

  Get detected range for a ray.
- int **GetRangeCount** () const

  Get the range count.
- double **GetRangeMax** () const

  Get the maximum range.
- double **GetRangeMin** () const

  Get the minimum range.
- double **GetRangeResolution** () const

  Get the range resolution If RangeResolution is 1, the number of simulated rays is equal to the number of returned range readings.
- void **GetRanges** (std::vector< double > &_ranges)

  Get all the ranges.
- int **GetRayCount** () const

  Get the ray count.
- double **GetRetro** (int _index)

  Get detected retro (intensity) value for a ray.
- double **GetVAngle** ()
- **math::Angle GetVerticalAngleMax** () const

  Get the vertical scan line top angle.
- **math::Angle GetVerticalAngleMin** () const

  Get the vertical scan bottom angle.
- int **GetVerticalRangeCount** () const

  Get the vertical scan line count.
- int **GetVerticalRayCount** () const

  Get the vertical scan line count.
- double **GetVFOV** ()
- virtual void **Init** ()

  Initialize the ray.
- bool **IsHorizontal** ()

  Gets if sensor is horizontal.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** &_sdf)

  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.
- void **SetAngleMax** (double _angle)

  Set the scan maximum angle.
- void **SetAngleMin** (double _angle)

  Set the scan minimum angle.
- void **SetVerticalAngleMax** (double _angle)

  Set the vertical scan line top angle.
- void **SetVerticalAngleMin** (double _angle)

  Set the vertical scan bottom angle.

Protected Member Functions

- virtual void **Fini** ()

  Finalize the ray.
- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.

Protected Attributes

- unsigned int **cameraCount**
- **sdf::ElementPtr cameraElem**
- double **chfov**
- double **cvfov**
- double **far**
- double **hang**
- unsigned int **height_1st**
- unsigned int **height_2nd**
- double **hfov**
- **sdf::ElementPtr horzElem**
- bool **isHorizontal**
- double **near**
- **math::Vector3 offset**
- **sdf::ElementPtr rangeElem**
- double **ratio_1st**
- double **ratio_2nd**
- **sdf::ElementPtr rayElem**
- **sdf::ElementPtr scanElem**
- double **vang**
- **sdf::ElementPtr vertElem**
- double **vfov**
- unsigned int **width_1st**
- unsigned int **width_2nd**

10.70.1 Constructor & Destructor Documentation

10.70.1.1 gazebo::sensors::GpuRaySensor::GpuRaySensor ( )

Constructor.

10.70.1.2 virtual gazebo::sensors::GpuRaySensor::~GpuRaySensor ( ) [virtual]

Destructor.

10.70.2 Member Function Documentation

10.70.2.1 event::ConnectionPtr gazebo::sensors::GpuRaySensor::ConnectNewLaserFrame (boost::function< void(const float *, unsigned int, unsigned int, unsigned int, const std::string &)> *subscriber*)

Connect a to the add entity signal Nate do these parameters need to be specified here?

10.70.2.2 void gazebo::sensors::GpuRaySensor::DisconnectNewLaserFrame (event::ConnectionPtr & *c*)

Disconnect Laser Frame.

Parameters

Connection	pointer to disconnect

10.70.2.3 virtual void gazebo::sensors::GpuRaySensor::Fini ( ) [protected], [virtual]

Finalize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

10.70.2.4 double gazebo::sensors::GpuRaySensor::Get1stRatio ( )

Nate fill in.

I'm not sure what these what these sensor parameters refer to

10.70.2.5 double gazebo::sensors::GpuRaySensor::Get2ndRatio ( )

Todo Document me

10.70.2.6 math::Angle gazebo::sensors::GpuRaySensor::GetAngleMax ( ) const

Get the maximum angle.

Returns

the maximum angle

10.70.2.7 math::Angle gazebo::sensors::GpuRaySensor::GetAngleMin ( ) const

Get the minimum angle.

Returns

The minimum angle

10.70.2.8 double gazebo::sensors::GpuRaySensor::GetAngleResolution ( ) const

Get radians between each range.

10.70.2.9 unsigned int gazebo::sensors::GpuRaySensor::GetCameraCount ( )

Gets the camera count.

Returns

Number of cameras

10.70.2.10 double gazebo::sensors::GpuRaySensor::GetCHFOV ( )

Todo Document me

10.70.2.11 double gazebo::sensors::GpuRaySensor::GetCVFOV ( )

Todo Document me

10.70.2.12 int gazebo::sensors::GpuRaySensor::GetFiducial (int *_index*)

Get detected fiducial value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index of specific ray

Returns

Fiducial value of ray

10.70.2.13 double gazebo::sensors::GpuRaySensor::GetHAngle ( )

Todo Document me

10.70.2.14 double gazebo::sensors::GpuRaySensor::GetHFOV ( )

Todo Document me

10.70.2.15 rendering::GpuLaserPtr gazebo::sensors::GpuRaySensor::GetLaserCamera ( ) const [inline]

Returns a pointer to the internally kept **rendering::GpuLaser** (p. 375).

Returns

Pointer to GpuLaser

10.70.2.16 double gazebo::sensors::GpuRaySensor::GetRange (int *_index*)

Get detected range for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index of specific ray

Returns

Returns DBL_MAX for no detection.

10.70.2.17 int gazebo::sensors::GpuRaySensor::GetRangeCount ( ) const

Get the range count.

Returns

The number of ranges

10.70.2.18 double gazebo::sensors::GpuRaySensor::GetRangeMax ( ) const

Get the maximum range.

Returns

The maximum range

10.70.2.19 double gazebo::sensors::GpuRaySensor::GetRangeMin ( ) const

Get the minimum range.

Returns

The minimum range

10.70.2.20 double gazebo::sensors::GpuRaySensor::GetRangeResolution ( ) const

Get the range resolution If RangeResolution is 1, the number of simulated rays is equal to the number of returned range readings.

If it's less than 1, fewer simulated rays than actual returned range readings are used, the results are interpolated from two nearest neighbors, and vice versa.

Returns

The Range Resolution

10.70.2.21 void gazebo::sensors::GpuRaySensor::GetRanges (std::vector< double > & *_ranges*)

Get all the ranges.

Parameters

_range	A vector that will contain all the range data

10.70.2.22 int gazebo::sensors::GpuRaySensor::GetRayCount ( ) const

Get the ray count.

Returns

The number of rays

10.70.2.23 double gazebo::sensors::GpuRaySensor::GetRetro (int *_index*)

Get detected retro (intensity) value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index of specific ray

Returns

Intensity value of ray

10.70.2.24 double gazebo::sensors::GpuRaySensor::GetVAngle ( )

Todo Document me

10.70.2.25 math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMax ( ) const

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.70.2.26 math::Angle gazebo::sensors::GpuRaySensor::GetVerticalAngleMin ( ) const

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.70.2.27 int gazebo::sensors::GpuRaySensor::GetVerticalRangeCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.70.2.28 int gazebo::sensors::GpuRaySensor::GetVerticalRayCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.70.2.29 double gazebo::sensors::GpuRaySensor::GetVFOV ( )

Todo Document me

10.70.2.30 virtual void gazebo::sensors::GpuRaySensor::Init ( ) [virtual]

Initialize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.70.2.31 bool gazebo::sensors::GpuRaySensor::IsHorizontal ( )

Gets if sensor is horizontal.

Returns

True if horizontal, false if not

10.70.2.32 virtual void gazebo::sensors::GpuRaySensor::Load (const std::string & *_worldName,* sdf::ElementPtr & *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 765) parameters
in	*_worldName*	Name of world to load from

10.70.2.33 virtual void gazebo::sensors::GpuRaySensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.70.2.34 void gazebo::sensors::GpuRaySensor::SetAngleMax (double *_angle*)

Set the scan maximum angle.

Parameters

in	*_angle*	The maximum angle

10.70.2.35 void gazebo::sensors::GpuRaySensor::SetAngleMin (double *_angle*)

Set the scan minimum angle.

Parameters

in	*_angle*	The minimum angle

10.70.2.36 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMax (double *_angle*)

Set the vertical scan line top angle.

Parameters

in	*_angle*	The Maximum angle of the scan block

10.70.2.37 void gazebo::sensors::GpuRaySensor::SetVerticalAngleMin (double *_angle*)

Set the vertical scan bottom angle.

Parameters

in	*_angle*	The minimum angle of the scan block

10.70.2.38 virtual void gazebo::sensors::GpuRaySensor::UpdateImpl (bool *_force*) [protected], [virtual]

Update the sensor information.

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 771).

10.70.3 Member Data Documentation

10.70.3.1 unsigned int gazebo::sensors::GpuRaySensor::cameraCount [protected]

10.70.3.2 sdf::ElementPtr gazebo::sensors::GpuRaySensor::cameraElem [protected]

10.70.3.3 double gazebo::sensors::GpuRaySensor::chfov [protected]

10.70.3.4 double gazebo::sensors::GpuRaySensor::cvfov [protected]

10.70.3.5 double gazebo::sensors::GpuRaySensor::far [protected]

10.70.3.6 double gazebo::sensors::GpuRaySensor::hang [protected]

10.70.3.7 unsigned int gazebo::sensors::GpuRaySensor::height_1st [protected]

10.70.3.8 unsigned int gazebo::sensors::GpuRaySensor::height_2nd [protected]

10.70.3.9 double gazebo::sensors::GpuRaySensor::hfov [protected]

10.70.3.10 sdf::ElementPtr gazebo::sensors::GpuRaySensor::horzElem [protected]

10.70.3.11 bool gazebo::sensors::GpuRaySensor::isHorizontal [protected]

10.70.3.12 double gazebo::sensors::GpuRaySensor::near [protected]

10.70.3.13 math::Vector3 gazebo::sensors::GpuRaySensor::offset [protected]

10.70.3.14 sdf::ElementPtr gazebo::sensors::GpuRaySensor::rangeElem [protected]

10.70.3.15 double gazebo::sensors::GpuRaySensor::ratio_1st [protected]

10.70.3.16 double gazebo::sensors::GpuRaySensor::ratio_2nd [protected]

10.70.3.17 sdf::ElementPtr gazebo::sensors::GpuRaySensor::rayElem [protected]

10.70.3.18 sdf::ElementPtr gazebo::sensors::GpuRaySensor::scanElem [protected]

10.70.3.19 double gazebo::sensors::GpuRaySensor::vang [protected]

10.70.3.20 sdf::ElementPtr gazebo::sensors::GpuRaySensor::vertElem [protected]

10.70.3.21 double gazebo::sensors::GpuRaySensor::vfov [protected]

10.70.3.22 unsigned int gazebo::sensors::GpuRaySensor::width_1st [protected]

10.70.3.23 unsigned int gazebo::sensors::GpuRaySensor::width_2nd [protected]

The documentation for this class was generated from the following file:

- **GpuRaySensor.hh**

10.71 gazebo::rendering::Grid Class Reference

Displays a grid of cells, drawn with lines.

`#include <rendering/rendering.hh>`

Public Member Functions

- **Grid** (**Scene** *_scene, uint32_t _cellCount, float _cellLength, float _lineWidth, const **common::Color** &_color)
  Constructor.
- ~**Grid** ()
  Destructor.
- void **Enable** (bool _enable)
  Enable or disable the grid.
- uint32_t **GetCellCount** () const
  Get the numb.
- float **GetCellLength** () const
  Get the cell length.
- **common::Color GetColor** () const
  Return the grid color.
- uint32_t **GetHeight** () const
  Get the height of the grid.
- float **GetLineWidth** () const
  Get the width of the grid line.
- Ogre::SceneNode * **GetSceneNode** ()
  *Get the **Ogre** (p. 118) scene node associated with this grid.*
- void **Init** ()
  Initialize the grid.
- void **SetCellCount** (uint32_t _count)
  Set the number of cells.
- void **SetCellLength** (float _len)
  Set the cell length.
- void **SetColor** (const **common::Color** &_color)
  Sets the color of the grid.

- void **SetHeight** (uint32_t _count)

  Set the height of the grid.
- void **SetLineWidth** (float _width)

  Set the line width.
- void **SetUserData** (const Ogre::Any &_data)

  Sets user data on all ogre objects we own.

10.71.1 Detailed Description

Displays a grid of cells, drawn with lines.

Displays a grid of cells, drawn with lines. A grid with an identity orientation is drawn along the XY plane.

10.71.2 Constructor & Destructor Documentation

10.71.2.1 gazebo::rendering::Grid::Grid (Scene * *_scene*, uint32_t *_cellCount*, float *_cellLength*, float *_lineWidth*, const common::Color & *_color*)

Constructor.

Parameters

in	***Scene*** *(p. 746)*	The scene this object is part of
in	*cell_count*	The number of cells to draw
in	*cell_length*	The size of each cell
in	*r*	Red color component, in the range [0, 1]
in	*g*	Green color component, in the range [0, 1]
in	*b*	Blue color component, in the range [0, 1]

10.71.2.2 gazebo::rendering::Grid::~Grid ( )

Destructor.

10.71.3 Member Function Documentation

10.71.3.1 void gazebo::rendering::Grid::Enable (bool *_enable*)

Enable or disable the grid.

Parameters

in	*_enable*	Set to true to view the grid, false to make invisible.

10.71.3.2 uint32_t gazebo::rendering::Grid::GetCellCount ( ) const [inline]

Get the numb.

10.71.3.3 float gazebo::rendering::Grid::GetCellLength ( ) const [inline]

Get the cell length.

Returns

The cell length

10.71.3.4 common::Color gazebo::rendering::Grid::GetColor ( ) const [inline]

Return the grid color.

Returns

The grid color

10.71.3.5 uint32_t gazebo::rendering::Grid::GetHeight ( ) const [inline]

Get the height of the grid.

Returns

The height

10.71.3.6 float gazebo::rendering::Grid::GetLineWidth ( ) const [inline]

Get the width of the grid line.

Returns

The line width

10.71.3.7 Ogre::SceneNode* gazebo::rendering::Grid::GetSceneNode ( ) [inline]

Get the **Ogre** (p. 118) scene node associated with this grid.

Returns

The **Ogre** (p. 118) scene node associated with this grid

10.71.3.8 void gazebo::rendering::Grid::Init ( )

Initialize the grid.

10.71.3.9 void gazebo::rendering::Grid::SetCellCount (uint32_t *_count*)

Set the number of cells.

Parameters

in	*The*	number of cells

10.71.3.10 void gazebo::rendering::Grid::SetCellLength (float *_len*)

Set the cell length.

Parameters

in	*The*	cell length

10.71.3.11 void gazebo::rendering::Grid::SetColor (const common::Color & *_color*)

Sets the color of the grid.

Parameters

in	*_color*	The grid color

10.71.3.12 void gazebo::rendering::Grid::SetHeight (uint32_t *_count*)

Set the height of the grid.

Parameters

in	*_count*	**Grid** (p. 389) height

10.71.3.13 void gazebo::rendering::Grid::SetLineWidth (float *_width*)

Set the line width.

Parameters

in	*_width*	The width of the grid

10.71.3.14 void gazebo::rendering::Grid::SetUserData (const Ogre::Any & *_data*)

Sets user data on all ogre objects we own.

Parameters

in	*_data*	The user data

The documentation for this class was generated from the following file:

- **Grid.hh**

10.72 gazebo::physics::Gripper Class Reference

A gripper abstraction.

`#include <physics/physics.hh>`

Public Member Functions

- **Gripper** (**ModelPtr** _model)
  Constructor.
- virtual ~**Gripper** ()
  Destructor.
- virtual void **Init** ()
  Initialize.
- virtual void **Load** (**sdf::ElementPtr** _sdf)
  Load the gripper.

10.72.1 Detailed Description

A gripper abstraction.

A gripper is a collection of links that act as a gripper. This class will intelligently generate fixed joints between the gripper and an object within the gripper. This allows the object to be manipulated without falling or behaving poorly.

10.72.2 Constructor & Destructor Documentation

10.72.2.1 gazebo::physics::Gripper::Gripper (**ModelPtr** *_model*) [explicit]

Constructor.

Parameters

in	*_model*	The model which contains the **Gripper** (p. 393).

10.72.2.2 virtual gazebo::physics::Gripper::~Gripper ( ) [virtual]

Destructor.

10.72.3 Member Function Documentation

10.72.3.1 virtual void gazebo::physics::Gripper::Init ( ) [virtual]

Initialize.

10.72.3.2 virtual void gazebo::physics::Gripper::Load (**sdf::ElementPtr** *_sdf*) [virtual]

Load the gripper.

Parameters

in	*_sdf*	Shared point to an sdf element that contains the list of links in the gripper.

The documentation for this class was generated from the following file:

- **Gripper.hh**

10.73 gazebo::rendering::GUIOverlay Class Reference

A class that creates a CEGUI overlay on a render window.

`#include <rendering/rendering.hh>`

Public Member Functions

- **GUIOverlay** ()
  Constructor.
- virtual ~**GUIOverlay** ()
  Destructor.
- bool **AttachCameraToImage** (**CameraPtr** &_camera, const std::string &_windowName)
  *Use this function to draw the output from a **rendering::Camera** (p. 233) to and overlay window.*
- bool **AttachCameraToImage** (**DepthCameraPtr** &_camera, const std::string &_windowName)
  *Use this function to draw the output from a **rendering::DepthCamera** (p. 312) to and overlay window.*
- template<typename T >
  void **ButtonCallback** (const std::string &_buttonName, void(T::*_fp)(), T *_obj)
  Register a CEGUI button callback.
- void **CreateWindow** (const std::string &_type, const std::string &_name, const std::string &_parent, const **math::Vector2d** &_position, const **math::Vector2d** &_size, const std::string &_text)
  Create a new window on the overlay.
- bool **HandleKeyPressEvent** (const std::string &_key)
  Handle a key press event.
- bool **HandleKeyReleaseEvent** (const std::string &_key)
  Handle a key release event.
- bool **HandleMouseEvent** (const **common::MouseEvent** &_evt)
  Handle a mouse event.
- void **Hide** ()
  Make the overlay invisible.
- void **Init** (Ogre::RenderTarget *_renderTarget)
  Initialize the overlay.
- bool **IsInitialized** ()
  Return true if the overlay has been initialized.
- void **LoadLayout** (const std::string &_filename)
  Load a CEGUI layout file.
- void **Resize** (unsigned int _width, unsigned int _height)
  Resize the window.
- void **Show** ()
  Make the overlay visible.
- void **Update** ()
  Update the overlay's objects.

10.73.1 Detailed Description

A class that creates a CEGUI overlay on a render window.

10.73.2 Constructor & Destructor Documentation

10.73.2.1 gazebo::rendering::GUIOverlay::GUIOverlay ( )

Constructor.

10.73.2.2 virtual gazebo::rendering::GUIOverlay::~GUIOverlay ( ) [virtual]

Destructor.

10.73.3 Member Function Documentation

10.73.3.1 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (CameraPtr & *_camera*, const std::string & *_windowName*)

Use this function to draw the output from a **rendering::Camera** (p. 233) to and overlay window.

Parameters

in	*_camera*	Pointer to the camera.
in	*_windowName*	Name of the window to receive the camera image

Returns

True if successful

10.73.3.2 bool gazebo::rendering::GUIOverlay::AttachCameraToImage (DepthCameraPtr & *_camera*, const std::string & *_windowName*)

Use this function to draw the output from a **rendering::DepthCamera** (p. 312) to and overlay window.

Parameters

in	*_camera*	Pointer to the camera.
in	*_windowName*	Name of the window to receive the camera image

Returns

True if successful

10.73.3.3 template<typename T > void gazebo::rendering::GUIOverlay::ButtonCallback (const std::string & *_buttonName*, void(T::*)() *_fp*, T * *_obj*) [inline]

Register a CEGUI button callback.

Assign a callback to a name button.

Parameters

in	*_buttonName*	Name of the button.
in	*_fp*	Function pointer to the callback.
in	*_obj*	Class pointer that contains _fp.

10.73.3.4 void gazebo::rendering::GUIOverlay::CreateWindow (const std::string & *_type*, const std::string & *_name*, const std::string & *_parent*, const **math::Vector2d** & *_position*, const **math::Vector2d** & *_size*, const std::string & *_text*)

Create a new window on the overlay.

Parameters

in	*_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.

10.73.3.5 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.

10.73.3.6 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.

10.73.3.7 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.

10.73.3.8 void gazebo::rendering::GUIOverlay::Hide ( )

Make the overlay invisible.

10.73.3.9 void gazebo::rendering::GUIOverlay::Init (Ogre::RenderTarget * *_renderTarget*)

Initialize the overlay.

Parameters

in	*_renderTarget*	The render target which will have the overlay.

10.73.3.10 bool gazebo::rendering::GUIOverlay::IsInitialized ( )

Return true if the overlay has been initialized.

Returns

True if initialized

10.73.3.11 void gazebo::rendering::GUIOverlay::LoadLayout (const std::string & *_filename*)

Load a CEGUI layout file.

Parameters

in	*_filename*	Name of the layout file.

10.73.3.12 void gazebo::rendering::GUIOverlay::Resize (unsigned int *_width*, unsigned int *_height*)

Resize the window.

10.73.3.13 void gazebo::rendering::GUIOverlay::Show ( )

Make the overlay visible.

10.73.3.14 void gazebo::rendering::GUIOverlay::Update ( )

Update the overlay's objects.

The documentation for this class was generated from the following file:

- **GUIOverlay.hh**

10.74 gazebo::rendering::Heightmap Class Reference

Rendering a terrain using heightmap information.

`#include <rendering/rendering.hh>`

Public Member Functions

- **Heightmap** (**ScenePtr** _scene)
  Constructor.
- virtual ~**Heightmap** ()
  Destructor.
- double **GetHeight** (double _x, double _y)
  Get the height at a location.
- void **Load** ()
  Load the heightmap.
- void **LoadFromMsg** (ConstVisualPtr &_msg)
  Load the heightmap from a visual message.

10.74.1 Detailed Description

Rendering a terrain using heightmap information.

10.74.2 Constructor & Destructor Documentation

10.74.2.1 gazebo::rendering::Heightmap::Heightmap (ScenePtr *_scene*)

Constructor.

Parameters

in	*_scene*	Pointer to the scene that will contain the heightmap

10.74.2.2 virtual gazebo::rendering::Heightmap::~Heightmap ( ) [virtual]

Destructor.

10.74.3 Member Function Documentation

10.74.3.1 double gazebo::rendering::Heightmap::GetHeight (double *_x*, double *_y*)

Get the height at a location.

Parameters

in	*_x*	X location
in	*_y*	Y location

Returns

The height at the specified location

10.74.3.2 void gazebo::rendering::Heightmap::Load ( )

Load the heightmap.

10.74.3.3 void gazebo::rendering::Heightmap::LoadFromMsg (ConstVisualPtr & *_msg*)

Load the heightmap from a visual message.

Parameters

in	*_msg*	The visual message containing heightmap info

The documentation for this class was generated from the following file:

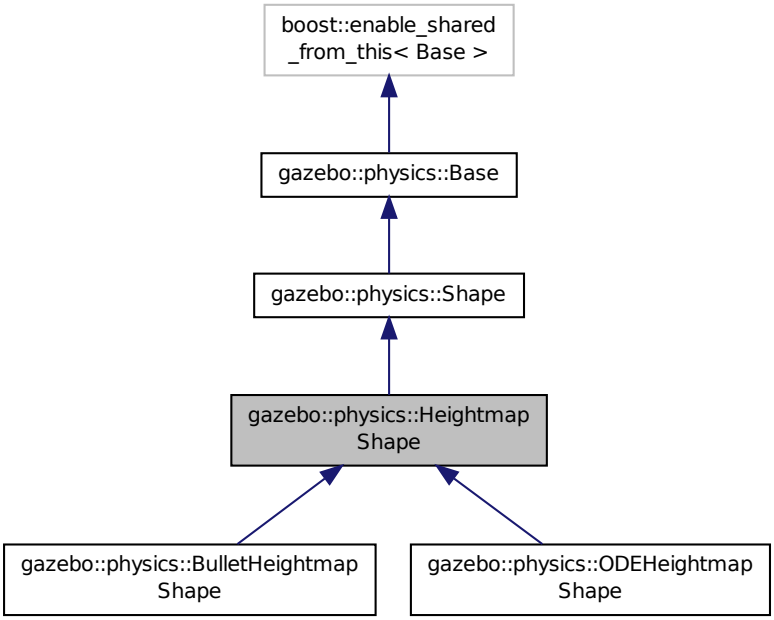
- **Heightmap.hh**

10.75 gazebo::physics::HeightmapShape Class Reference

HeightmapShape (p. 399) collision shape builds a heightmap from an image.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::HeightmapShape:



Public Member Functions

- **HeightmapShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**HeightmapShape** ()

  Destructor.
- void **FillMsg** (msgs::Geometry &_msg)

  Fill a geometry message with this shape's data.
- float **GetHeight** (int _x, int _y)

  Get a height at a position.
- float **GetMaxHeight** () const

  Get the maximum height.
- float **GetMinHeight** () const

  Get the minimum height.
- **math::Vector3 GetPos** () const

  Get the origin in world coordinate frame.
- **math::Vector3 GetSize** () const

  Get the size in meters.
- std::string **GetURI** () const

  Get the URI of the heightmap image.
- **math::Vector2i GetVertexCount** () const

  Return the number of vertices, which equals the size of the image used to load the heightmap.

- virtual void **Init** ()

  Initialize the heightmap.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the heightmap.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  Update the heightmap from a message.

Protected Attributes

- std::vector< float > **heights**

  Lookup table of heights.
- **common::Image img**

  Image used to generate the heights.
- **math::Vector3 scale**

  Scaling factor.
- int **subSampling**

  Level of subsampling.
- unsigned int **vertSize**

  Size of the height lookup table.

Additional Inherited Members

10.75.1 Detailed Description

HeightmapShape (p. 399) collision shape builds a heightmap from an image.

The supplied image must be square with N∗N+1 pixels per side, where N is an integer.

10.75.2 Constructor & Destructor Documentation

10.75.2.1 gazebo::physics::HeightmapShape::HeightmapShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent **Collision** (p. 262) object.

10.75.2.2 virtual gazebo::physics::HeightmapShape::~HeightmapShape ( ) [virtual]

Destructor.

10.75.3 Member Function Documentation

10.75.3.1 void gazebo::physics::HeightmapShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Fill a geometry message with this shape's data.

Parameters

in	*_msg*	Message to fill.

Implements **gazebo::physics::Shape** (p. 781).

10.75.3.2 float gazebo::physics::HeightmapShape::GetHeight (int _x, int _y)

Get a height at a position.

Parameters

in	*_x*	X position.
in	*_y*	Y position.

Returns

The height at a the specified location.

10.75.3.3 float gazebo::physics::HeightmapShape::GetMaxHeight ( ) const

Get the maximum height.

Returns

The maximum height.

10.75.3.4 float gazebo::physics::HeightmapShape::GetMinHeight ( ) const

Get the minimum height.

Returns

The minimum height.

10.75.3.5 math::Vector3 gazebo::physics::HeightmapShape::GetPos ( ) const

Get the origin in world coordinate frame.

Returns

The origin in world coordinate frame.

10.75.3.6 math::Vector3 gazebo::physics::HeightmapShape::GetSize ( ) const

Get the size in meters.

Returns

The size in meters.

10.75.3.7 std::string gazebo::physics::HeightmapShape::GetURI ( ) const

Get the URI of the heightmap image.

Returns

The heightmap image URI.

10.75.3.8 math::Vector2i gazebo::physics::HeightmapShape::GetVertexCount ( ) const

Return the number of vertices, which equals the size of the image used to load the heightmap.

Returns

math::Vector2i (p. 894), result.x = width, result.y = length/height.

10.75.3.9 virtual void gazebo::physics::HeightmapShape::Init ( ) [virtual]

Initialize the heightmap.

Implements **gazebo::physics::Shape** (p. 781).

Reimplemented in **gazebo::physics::BulletHeightmapShape** (p. 178), and **gazebo::physics::ODEHeightmapShape** (p. 586).

10.75.3.10 virtual void gazebo::physics::HeightmapShape::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the heightmap.

Parameters

in	*_sdf*	SDF value to load from.

Reimplemented from **gazebo::physics::Base** (p. 154).

10.75.3.11 virtual void gazebo::physics::HeightmapShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Update the heightmap from a message.

Parameters

in	*_msg*	Message to update from.

Implements **gazebo::physics::Shape** (p. 782).

10.75.4 Member Data Documentation

10.75.4.1 std::vector<float> gazebo::physics::HeightmapShape::heights [protected]

Lookup table of heights.

10.75.4.2 common::Image gazebo::physics::HeightmapShape::img [protected]

Image used to generate the heights.

10.75.4.3 math::Vector3 gazebo::physics::HeightmapShape::scale [protected]

Scaling factor.

10.75.4.4 int gazebo::physics::HeightmapShape::subSampling [protected]

Level of subsampling.

10.75.4.5 unsigned int gazebo::physics::HeightmapShape::vertSize [protected]

Size of the height lookup table.

The documentation for this class was generated from the following file:

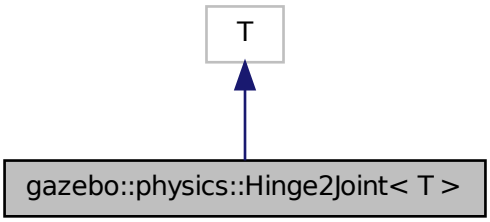
- **HeightmapShape.hh**

10.76 gazebo::physics::Hinge2Joint< T > Class Template Reference

A two axis hinge joint.

`#include <Hinge2Joint.hh>`

Inheritance diagram for gazebo::physics::Hinge2Joint< T >:



Public Member Functions

- **Hinge2Joint** (**BasePtr** _parent)

  Constructor.
- virtual ~**Hinge2Joint** ()

  Destructor.

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the joint.

10.76.1 Detailed Description

template<class T>class gazebo::physics::Hinge2Joint< T >

A two axis hinge joint.

10.76.2 Constructor & Destructor Documentation

10.76.2.1 template<class T> gazebo::physics::Hinge2Joint< T >::Hinge2Joint (BasePtr *_parent*) [inline]

Constructor.

10.76.2.2 template<class T> virtual gazebo::physics::Hinge2Joint< T >::~Hinge2Joint ( ) [inline], [virtual]

Destructor.

10.76.3 Member Function Documentation

10.76.3.1 template<class T> virtual void gazebo::physics::Hinge2Joint< T >::Load (sdf::ElementPtr *_sdf*) [inline], [protected], [virtual]

Load the joint.

Reimplemented in **gazebo::physics::BulletHinge2Joint** (p. 182), and **gazebo::physics::ODEHinge2Joint** (p. 589).

The documentation for this class was generated from the following file:

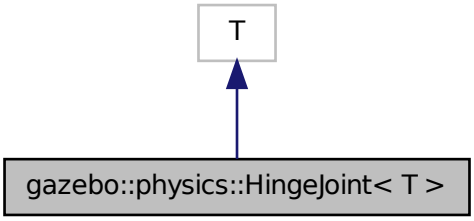
- **Hinge2Joint.hh**

10.77 gazebo::physics::HingeJoint< T > Class Template Reference

A single axis hinge joint.

`#include <HingeJoint.hh>`

Inheritance diagram for gazebo::physics::HingeJoint< T >:



Public Member Functions

- **HingeJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**HingeJoint** ()

  Destructor.

Protected Member Functions

- virtual void **Init** ()
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load joint.

10.77.1 Detailed Description

template<class T>class gazebo::physics::HingeJoint< T >

A single axis hinge joint.

10.77.2 Constructor & Destructor Documentation

10.77.2.1 template<class T> gazebo::physics::HingeJoint< T >::HingeJoint (BasePtr *_parent*) [inline]

Constructor.

10.77.2.2 template<class T> virtual gazebo::physics::HingeJoint< T >::~HingeJoint ( ) [inline], [virtual]

Destructor.

10.77.3 Member Function Documentation

10.77.3.1 template<class T> virtual void gazebo::physics::HingeJoint< T >::Init ( ) [inline], [protected], [virtual]

10.77.3.2 template<class T> virtual void gazebo::physics::HingeJoint< T >::Load (sdf::ElementPtr *_sdf*) [inline], [protected], [virtual]

Load joint.

Reimplemented in **gazebo::physics::BulletHingeJoint** (p. 187), and **gazebo::physics::ODEHingeJoint** (p. 593).

The documentation for this class was generated from the following file:

- **HingeJoint.hh**

10.78 gazebo::common::Image Class Reference

Encapsulates an image.

`#include <Image.hh>`

Public Types

- enum **PixelFormat** {
  UNKNOWN, **L_INT8**, **L_INT16**, **RGB_INT8**,
  RGBA_INT8, **BGRA_INT8**, **RGB_INT16**, **RGB_INT32**,
  BGR_INT8, **BGR_INT16**, **BGR_INT32**, **R_FLOAT16**,
  RGB_FLOAT16, **R_FLOAT32**, **RGB_FLOAT32**, **BAYER_RGGB8**,
  BAYER_RGGR8, **BAYER_GBRG8**, **BAYER_GRBG8** }

  Pixel formats enumeration.

Public Member Functions

- **Image** (const std::string &_filename="")

  Constructor.
- virtual ~**Image** ()

  Destructor.
- **Color GetAvgColor** ()

  Get the average color.
- unsigned int **GetBPP** () const

  Get the size of one pixel in bits.
- void **GetData** (unsigned char **_data, unsigned int &_count) const

  Get the image as a data array.
- std::string **GetFilename** () const

  Get the full filename of the image.
- unsigned int **GetHeight** () const

  Get the height.
- **Color GetMaxColor** ()

  Get the max color.

- int **GetPitch** () const
- **Color GetPixel** (unsigned int _x, unsigned int _y)

  Get a pixel color value.
- **PixelFormat GetPixelFormat** () const

  Get the pixel format.
- void **GetRGBData** (unsigned char **_data, unsigned int &_count) const

  Get only the RGB data from the image.
- unsigned int **GetWidth** () const

  Get the width.
- int **Load** (const std::string &_filename)

  Load an image.
- void **Rescale** (int _width, int _height)

  Rescale the image.
- void **SavePNG** (const std::string &_filename)

  Save the image in PNG format.
- void **SetFromData** (const unsigned char *_data, unsigned int _width, unsigned int _height, **Image::PixelFormat** _format)

  Set the image from raw data.
- bool **Valid** () const

  Returns whether this is a valid image.

10.78.1 Detailed Description

Encapsulates an image.

10.78.2 Member Enumeration Documentation

10.78.2.1 enum gazebo::common::Image::PixelFormat

Pixel formats enumeration.

Enumerator:

UNKNOWN

L_INT8

L_INT16

RGB_INT8

RGBA_INT8

BGRA_INT8

RGB_INT16

RGB_INT32

BGR_INT8

BGR_INT16

BGR_INT32

R_FLOAT16

RGB_FLOAT16

R_FLOAT32

RGB_FLOAT32

BAYER_RGGB8

BAYER_RGGR8

BAYER_GBRG8

BAYER_GRBG8

10.78.3 Constructor & Destructor Documentation

10.78.3.1 gazebo::common::Image::Image (const std::string & *_filename* = " ")

Constructor.

Parameters

in	*_filename*	the path to the image

10.78.3.2 virtual gazebo::common::Image::~Image ( ) [virtual]

Destructor.

10.78.4 Member Function Documentation

10.78.4.1 Color gazebo::common::Image::GetAvgColor ( )

Get the average color.

Returns

The average color

10.78.4.2 unsigned int gazebo::common::Image::GetBPP ( ) const

Get the size of one pixel in bits.

Returns

The BPP of the image

10.78.4.3 void gazebo::common::Image::GetData (unsigned char ** *_data*, unsigned int & *_count*) const

Get the image as a data array.

Parameters

out	*_data*	Pointer to a NULL array of char.
out	*_count*	The resulting data array size

10.78.4.4 std::string gazebo::common::Image::GetFilename ( ) const

Get the full filename of the image.

Returns

The filename used to load the image

10.78.4.5 unsigned int gazebo::common::Image::GetHeight ( ) const

Get the height.

Returns

The image height

10.78.4.6 Color gazebo::common::Image::GetMaxColor ( )

Get the max color.

Returns

The max color

10.78.4.7 int gazebo::common::Image::GetPitch ( ) const

Returns

The pitch of the image

10.78.4.8 Color gazebo::common::Image::GetPixel (unsigned int _x, unsigned int _y)

Get a pixel color value.

Parameters

in	_x	Column location in the image
in	_y	Row location in the image

10.78.4.9 PixelFormat gazebo::common::Image::GetPixelFormat ( ) const

Get the pixel format.

Returns

PixelFormat

10.78.4.10 void gazebo::common::Image::GetRGBData (unsigned char ** _data, unsigned int & _count) const

Get only the RGB data from the image.

This will drop the alpha channel if one is present.

Parameters

out	*_data*	Pointer to a NULL array of char.
out	*_count*	The resulting data array size

10.78.4.11 unsigned int gazebo::common::Image::GetWidth ( ) const

Get the width.

Returns

The image width

10.78.4.12 int gazebo::common::Image::Load (const std::string & _filename)

Load an image.

Return 0 on success

Parameters

in	*_filename*	the path to the image file

10.78.4.13 void gazebo::common::Image::Rescale (int _width, int _height)

Rescale the image.

Parameters

in	*_width*	New image width
in	*_height*	New image height

10.78.4.14 void gazebo::common::Image::SavePNG (const std::string & _filename)

Save the image in PNG format.

Parameters

in	*_filename*	The name of the saved image

10.78.4.15 void gazebo::common::Image::SetFromData (const unsigned char * _data, unsigned int _width, unsigned int _height, Image::PixelFormat _format)

Set the image from raw data.

Parameters

in	*_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

10.78.4.16 bool gazebo::common::Image::Valid ( ) const

Returns whether this is a valid image.

Returns

true if image has a bitmap

The documentation for this class was generated from the following file:

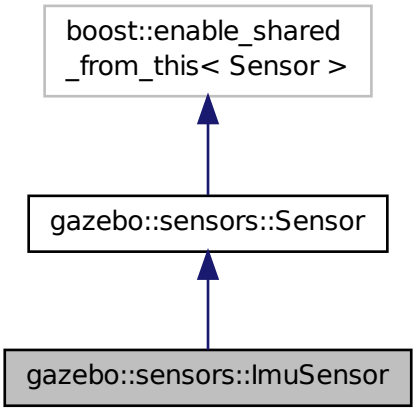
- **Image.hh**

10.79 gazebo::sensors::ImuSensor Class Reference

An IMU sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::ImuSensor:



Public Member Functions

- **ImuSensor** (Body *_body)

Constructor.

- virtual ~**ImuSensor** ()

  Destructor.

- Pose **GetVelocity** ()

  *Returns velocity as a **math::Pose** (p. 677) FIXME storing x,y,z components in a quaternion seems like a bad idea.*

Protected Member Functions

- virtual void **FiniChild** ()

  Finalize the ray.

- virtual void **InitChild** ()

  Initialize the ray.

- virtual void **LoadChild** (XMLConfigNode *_node)

  *Load the **ImuSensor** (p. 412) from XMLConfigNode.*

- virtual void **SaveChild** (std::string &_prefix, std::ostream &_stream)

  Save the sensor info in XML format.

- virtual void **UpdateChild** ()

  Update sensed values.

Additional Inherited Members

10.79.1 Detailed Description

An IMU sensor.

10.79.2 Constructor & Destructor Documentation

10.79.2.1 gazebo::sensors::ImuSensor::ImuSensor (Body * *_body*)

Constructor.

Parameters

_body	The IMU sensor must be attached to a body.

10.79.2.2 virtual gazebo::sensors::ImuSensor::~ImuSensor ( ) [virtual]

Destructor.

10.79.3 Member Function Documentation

10.79.3.1 virtual void gazebo::sensors::ImuSensor::FiniChild ( ) [protected], [virtual]

Finalize the ray.

10.79.3.2 Pose gazebo::sensors::ImuSensor::GetVelocity ( )

Returns velocity as a **math::Pose** (p. 677) FIXME storing x,y,z components in a quaternion seems like a bad idea.

Todo storing x,y,z components in a quaternion seems like a bad idea

Returns

velocity data stored in Pose Nate check

10.79.3.3 virtual void gazebo::sensors::ImuSensor::InitChild ( ) [protected], [virtual]

Initialize the ray.

10.79.3.4 virtual void gazebo::sensors::ImuSensor::LoadChild (XMLConfigNode * *_node*) [protected], [virtual]

Load the **ImuSensor** (p. 412) from XMLConfigNode.

Parameters

_node	The XMLConfig node

10.79.3.5 virtual void gazebo::sensors::ImuSensor::SaveChild (std::string & *_prefix*, std::ostream & *_stream*) [protected], [virtual]

Save the sensor info in XML format.

Parameters

_prefix	
_stream	Nate fill in

10.79.3.6 virtual void gazebo::sensors::ImuSensor::UpdateChild ( ) [protected], [virtual]

Update sensed values.

The documentation for this class was generated from the following file:

- **ImuSensor.hh**

10.80 gazebo::physics::Inertial Class Reference

A class for inertial information about a link.

`#include <physics/physics.hh>`

Public Member Functions

- **Inertial** ()

  Default Constructor.
- **Inertial** (double _mass)

  Constructor.
- **Inertial** (const **Inertial** &_inertial)

  Copy constructor.
- virtual ~**Inertial** ()

  Destructor.
- const **math::Vector3** & **GetCoG** () const

  Get the center of gravity.
- double **GetIXX** () const

  Get IXX.
- double **GetIXY** () const

  Get IXY.
- double **GetIXZ** () const

  Get IXZ.
- double **GetIYY** () const

  Get IYY.
- double **GetIYZ** () const

  Get IXZ.
- double **GetIZZ** () const

  Get IZZ.
- double **GetMass** () const

  Get the mass.
- const **math::Pose GetPose** () const

  *Get the pose about which the mass and inertia matrix is specified in the **Link** (p. 454) frame.*
- **math::Vector3 GetPrincipalMoments** () const

  Get the principal moments of inertia (Ixx, Iyy, Izz).
- **math::Vector3 GetProductsofInertia** () const

  Get the products of inertia (Ixy, Ixy, Iyz).
- void **Load** (**sdf::ElementPtr** _sdf)

  Load from SDF values.
- **Inertial operator+** (const **Inertial** &_inertial) const

  Addition operator.
- const **Inertial** & **operator+=** (const **Inertial** &_inertial)

  Addition equal operator.
- **Inertial** & **operator=** (const **Inertial** &_inertial)

  Equal operator.
- void **ProcessMsg** (const msgs::Inertial &_msg)

  Update parameters from a message.
- void **Reset** ()

  Reset all the mass properties.
- void **Rotate** (const **math::Quaternion** &rot)

  Rotate this mass.
- void **SetCoG** (double _cx, double _cy, double _cz)

Set the center of gravity.

- void **SetCoG** (const **math::Vector3** &_center)

  Set the center of gravity.
- void **SetInertiaMatrix** (double _ixx, double _iyy, double _izz, double _ixy, double _ixz, double iyz)

  Set the mass matrix.
- void **SetIXX** (double _v)

  Set IXX.
- void **SetIXY** (double _v)

  Set IXY.
- void **SetIXZ** (double _v)

  Set IXZ.
- void **SetIYY** (double _v)

  Set IYY.
- void **SetIYZ** (double _v)

  Set IYZ.
- void **SetIZZ** (double _v)

  Set IZZ.
- void **SetMass** (double m)

  Set the mass.
- void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  update the parameters using new sdf values.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::Inertial** &_inertial)

  Output operator.

10.80.1 Detailed Description

A class for inertial information about a link.

10.80.2 Constructor & Destructor Documentation

10.80.2.1 gazebo::physics::Inertial::Inertial ( )

Default Constructor.

10.80.2.2 gazebo::physics::Inertial::Inertial (double *_mass*)

Constructor.

Parameters

in	*_mass*	Mass value in kg if using metric.

10.80.2.3 gazebo::physics::Inertial::Inertial (const Inertial & *_inertial*)

Copy constructor.

Parameters

in	*_inertial*	**Inertial** (p. 414) element to copy

10.80.2.4 virtual gazebo::physics::Inertial::~Inertial ( ) [virtual]

Destructor.

10.80.3 Member Function Documentation

10.80.3.1 const math::Vector3& gazebo::physics::Inertial::GetCoG ( ) const [inline]

Get the center of gravity.

Returns

The cneter of gravity.

10.80.3.2 double gazebo::physics::Inertial::GetIXX ( ) const

Get IXX.

Returns

IXX value

10.80.3.3 double gazebo::physics::Inertial::GetIXY ( ) const

Get IXY.

Returns

IXY value

10.80.3.4 double gazebo::physics::Inertial::GetIXZ ( ) const

Get IXZ.

Returns

IXZ value

10.80.3.5 double gazebo::physics::Inertial::GetIYY ( ) const

Get IYY.

Returns

IYY value

10.80.3.6 double gazebo::physics::Inertial::GetIYZ ( ) const

Get IXZ.

Returns

IYZ value

10.80.3.7 double gazebo::physics::Inertial::GetIZZ ( ) const

Get IZZ.

Returns

IZZ value

10.80.3.8 double gazebo::physics::Inertial::GetMass ( ) const

Get the mass.

10.80.3.9 const math::Pose gazebo::physics::Inertial::GetPose ( ) const [inline]

Get the pose about which the mass and inertia matrix is specified in the **Link** (p. 454) frame.

Returns

The inertial pose.

10.80.3.10 math::Vector3 gazebo::physics::Inertial::GetPrincipalMoments ( ) const

Get the principal moments of inertia (Ixx, Iyy, Izz).

Returns

The principal moments.

10.80.3.11 math::Vector3 gazebo::physics::Inertial::GetProductsofInertia ( ) const

Get the products of inertia (Ixy, Ixy, Iyz).

Returns

The products of inertia.

10.80.3.12 void gazebo::physics::Inertial::Load (sdf::ElementPtr _sdf)

Load from SDF values.

Parameters

in	*_sdf*	SDF value to load from.

10.80.3.13 Inertial gazebo::physics::Inertial::operator+ (const Inertial & _inertial) const

Addition operator.

Parameters

in	*_inertial*	**Inertial** (p. 414) to add.

Returns

The result of the addition.

10.80.3.14 const Inertial& gazebo::physics::Inertial::operator+= (const Inertial & _inertial)

Addition equal operator.

Parameters

in	*_inertial*	**Inertial** (p. 414) to add.

Returns

Reference to this object.

10.80.3.15 Inertial& gazebo::physics::Inertial::operator= (const Inertial & _inertial)

Equal operator.

Parameters

in	*_inertial*	**Inertial** (p. 414) to copy.

Returns

Reference to this object.

10.80.3.16 void gazebo::physics::Inertial::ProcessMsg (const msgs::Inertial & _msg)

Update parameters from a message.

Parameters

in	*_msg*	Message to read

10.80.3.17 void gazebo::physics::Inertial::Reset ( )

Reset all the mass properties.

10.80.3.18 void gazebo::physics::Inertial::Rotate (const math::Quaternion & *rot*)

Rotate this mass.

Parameters

in	*_rot*	Rotation amount.

10.80.3.19 void gazebo::physics::Inertial::SetCoG (double *_cx*, double *_cy*, double *_cz*)

Set the center of gravity.

Parameters

in	*_cx*	X position.
in	*_cy*	Y position.
in	*_cz*	Z position.

10.80.3.20 void gazebo::physics::Inertial::SetCoG (const math::Vector3 & *_center*)

Set the center of gravity.

Parameters

in	*_center*	Center of the gravity.

10.80.3.21 void gazebo::physics::Inertial::SetInertiaMatrix (double *_ixx*, double *_iyy*, double *_izz*, double *_ixy*, double *_ixz*, double *iyz*)

Set the mass matrix.

Parameters

in	*_ixx*	X second moment of inertia about x axis.
in	*_iyy*	Y second moment of inertia about y axis.
in	*_izz*	Z second moment of inertia about z axis.
in	*_ixy*	XY inertia.
in	*_ixz*	XZ inertia.
in	*_iyz*	YZ inertia.

10.80.3.22 void gazebo::physics::Inertial::SetIXX (double _v)

Set IXX.

Parameters

in	_v	IXX value

10.80.3.23 void gazebo::physics::Inertial::SetIXY (double _v)

Set IXY.

Parameters

in	_v	IXY value

10.80.3.24 void gazebo::physics::Inertial::SetIXZ (double _v)

Set IXZ.

Parameters

in	_v	IXZ value

10.80.3.25 void gazebo::physics::Inertial::SetIYY (double _v)

Set IYY.

Parameters

in	_v	IYY value

10.80.3.26 void gazebo::physics::Inertial::SetIYZ (double _v)

Set IYZ.

Parameters

in	_v	IXX value

10.80.3.27 void gazebo::physics::Inertial::SetIZZ (double _v)

Set IZZ.

Parameters

in	_v	IZZ value

10.80.3.28 void gazebo::physics::Inertial::SetMass (double *m*)

Set the mass.

10.80.3.29 void gazebo::physics::Inertial::UpdateParameters (sdf::ElementPtr *_sdf*)

update the parameters using new sdf values.

Parameters

in	*_sdf*	Update values from.

10.80.4 Friends And Related Function Documentation

10.80.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::physics::Inertial & *_inertial*) [friend]

Output operator.

Parameters

in	*_out*	Output stream.
in	*_inertial*	**Inertial** (p. 414) object to output.

The documentation for this class was generated from the following file:

- **Inertial.hh**

10.81 gazebo::transport::IOManager Class Reference

Managers boost::asio IO.

`#include <IOManager.hh>`

Public Member Functions

- **IOManager** ()
- ~**IOManager** ()
- void **DecCount** ()
- unsigned int **GetCount** () const
- boost::asio::io_service & **GetIO** ()
- void **IncCount** ()
- void **Stop** ()

10.81.1 Detailed Description

Managers boost::asio IO.

10.81.2 Constructor & Destructor Documentation

10.81.2.1 gazebo::transport::IOManager::IOManager ( )

10.81.2.2 gazebo::transport::IOManager::~IOManager ( )

10.81.3 Member Function Documentation

10.81.3.1 void gazebo::transport::IOManager::DecCount ( )

10.81.3.2 unsigned int gazebo::transport::IOManager::GetCount ( ) const

10.81.3.3 boost::asio::io_service& gazebo::transport::IOManager::GetIO ( )

10.81.3.4 void gazebo::transport::IOManager::IncCount ( )

10.81.3.5 void gazebo::transport::IOManager::Stop ( )

The documentation for this class was generated from the following file:

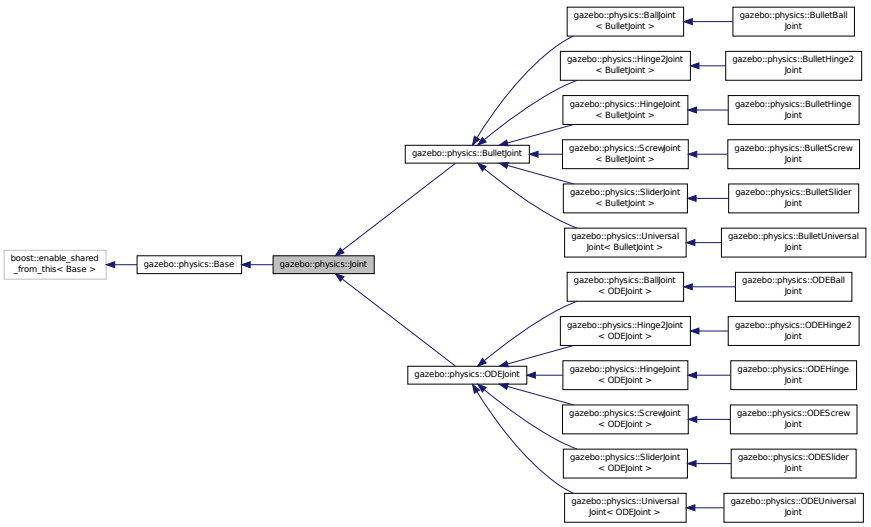
- **IOManager.hh**

10.82 gazebo::physics::Joint Class Reference

Base (p. 145) class for all joints.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Joint:



Public Types

- enum **Attribute** {
  FUDGE_FACTOR, **SUSPENSION_ERP**, **SUSPENSION_CFM**, **STOP_ERP**,
  STOP_CFM, **ERP**, **CFM**, **FMAX**,
  VEL, **HI_STOP**, **LO_STOP** }

  Joint *(p. 423) attribute types.*

Public Member Functions

- **Joint** (**BasePtr** _parent)

  Constructor.
- virtual ~**Joint** ()

  Destructor.
- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const =0

  Determines of the two bodies are connected by a joint.
- virtual void **Attach** (**LinkPtr** _parent, **LinkPtr** _child)

  Attach the two bodies with this joint.
- template<typename T >
  event::ConnectionPtr ConnectJointUpdate (T _subscriber)

  Connect a boost::slot the the joint update signal.
- virtual void **Detach** ()

  Detach this joint from all links.
- void **DisconnectJointUpdate** (**event::ConnectionPtr** &_conn)

  Disconnect a boost::slot the the joint update signal.
- void **FillJointMsg** (msgs::Joint &_msg) **GAZEBO_DEPRECATED**

  DEPRECATED.
- void **FillMsg** (msgs::Joint &_msg)

  Fill a joint message.
- virtual **math::Vector3 GetAnchor** (int _index) const =0

  Get the anchor point.
- **math::Angle GetAngle** (int _index) const

  Get the angle of rotation of an axis(index)
- **LinkPtr GetChild** () const

  Get the child link.
- virtual double **GetForce** (int _index)
- virtual **math::Vector3 GetGlobalAxis** (int _index) const =0

  Get the axis of rotation in global cooridnate frame.
- virtual **math::Angle GetHighStop** (int _index)=0

  Get the high stop of an axis(index).
- virtual **LinkPtr GetJointLink** (int _index) const =0

  Get the link to which the joint is attached according the _index.
- virtual **math::Vector3 GetLinkForce** (unsigned int _index) const =0

  Get the forces applied to the center of mass of a ***physics::Link*** *(p. 454) due to the existence of this* ***Joint*** *(p. 423).*
- virtual **math::Vector3 GetLinkTorque** (unsigned int _index) const =0

  Get the torque applied to the center of mass of a ***physics::Link*** *(p. 454) due to the existence of this* ***Joint*** *(p. 423).*
- **math::Vector3 GetLocalAxis** (int _index) const

  Get the axis of rotation.

- virtual **math::Angle GetLowStop** (int _index)=0

  Get the low stop of an axis(index).
- virtual double **GetMaxForce** (int _index)=0

  Get the max allowed force of an axis(index).
- **LinkPtr GetParent** () const

  Get the parent link.
- **JointState GetState** ()

  Get the joint state.
- virtual double **GetVelocity** (int _index) const =0

  Get the rotation rate of an axis(index)
- virtual void **Init** ()

  Initialize a joint.
- void **Load** (**LinkPtr** _parent, **LinkPtr** _child, const **math::Pose** &_pose)

  *Set pose, parent and child links of a **physics::Joint** (p. 423).*
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load **physics::Joint** (p. 423) from a SDF **sdf::Element** (p. 332).*
- virtual void **Reset** ()

  Reset the joint.
- virtual void **SetAnchor** (int _index, const **math::Vector3** &_anchor)=0

  Set the anchor point.
- void **SetAngle** (int _index, **math::Angle** _angle)

  *If the **Joint** (p. 423) is static, Gazebo stores the state of this **Joint** (p. 423) as a scalar inside the **Joint** (p. 423) class, so this call will NOT move the joint dynamically for a static **Model** (p. 521).*
- virtual void **SetAttribute** (**Attribute** _attr, int _index, double _value)=0

  Set a parameter for the joint.
- virtual void **SetAxis** (int _index, const **math::Vector3** &_axis)=0

  Set the axis of rotation.
- virtual void **SetDamping** (int _index, double _damping)=0

  Set the joint damping.
- virtual void **SetForce** (int _index, double _force)

  *Set the force applied to this **physics::Joint** (p. 423).*
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)=0

  Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)=0

  Set the low stop of an axis(index).
- virtual void **SetMaxForce** (int _index, double _force)=0

  Set the max allowed force of an axis(index).
- void **SetModel** (**ModelPtr** _model)

  Set the model this joint belongs too.
- void **SetState** (const **JointState** &_state)

  Set the joint state.
- virtual void **SetVelocity** (int _index, double _vel)=0

  Set the velocity of an axis(index).
- void **Update** ()

  Update the joint.
- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  Update the parameters using new sdf values.

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (int _index) const =0

  Get the angle of an axis helper function.

Protected Attributes

- **LinkPtr anchorLink**

  Anchor link.
- **math::Vector3 anchorPos**

  Anchor pose.
- **LinkPtr childLink**

  The first link this joint connects to.
- double **damping_coefficient**

  joint damping_coefficient
- **ModelPtr model**

  Pointer to the parent model.
- **LinkPtr parentLink**

  The second link this joint connects to.

10.82.1 Detailed Description

Base (p. 145) class for all joints.

10.82.2 Member Enumeration Documentation

10.82.2.1 enum gazebo::physics::Joint::Attribute

Joint (p. 423) attribute types.

Enumerator:

FUDGE_FACTOR Fudge factor.

SUSPENSION_ERP Suspension error reduction parameter.

SUSPENSION_CFM Suspension constraint force mixing.

STOP_ERP Stop limit error reduction parameter.

STOP_CFM Stop limit constraint force mixing.

ERP Error reduction parameter.

CFM Constraint force mixing.

FMAX Maximum force.

VEL Velocity.

HI_STOP High stop angle.

LO_STOP Low stop angle.

10.82.3 Constructor & Destructor Documentation

10.82.3.1 gazebo::physics::Joint::Joint (BasePtr *_parent*) [explicit]

Constructor.

Parameters

in	***Joint*** *(p. 423)*	parent

10.82.3.2 virtual gazebo::physics::Joint::~Joint ( ) [virtual]

Destructor.

10.82.4 Member Function Documentation

10.82.4.1 virtual bool gazebo::physics::Joint::AreConnected (LinkPtr *_one,* LinkPtr *_two*) const [pure virtual]

Determines 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::BulletJoint** (p. 189), and **gazebo::physics::ODEJoint** (p. 596).

10.82.4.2 virtual void gazebo::physics::Joint::Attach (LinkPtr *_parent,* LinkPtr *_child*) [virtual]

Attach the two bodies with this joint.

Parameters

in	*_parent*	Parent link.
in	*_child*	Child link.

Reimplemented in **gazebo::physics::ODEJoint** (p. 596), **gazebo::physics::BulletBallJoint** (p. 168), **gazebo::physics::BulletScrewJoint** (p. 215), **gazebo::physics::BulletHinge2Joint** (p. 181), **gazebo::physics::BulletHingeJoint** (p. 186), **gazebo::physics::BulletSliderJoint** (p. 219), and **gazebo::physics::BulletUniversalJoint** (p. 227).

10.82.4.3 template<typename T > event::ConnectionPtr gazebo::physics::Joint::ConnectJointUpdate (T *_subscriber*) [inline]

Connect a boost::slot the the joint update signal.

Parameters

in	*_subscriber*	Callback for the connection.

Returns

Connection pointer, which must be kept in scope.

References gazebo::event::EventT< T >::Connect().

10.82.4.4 virtual void gazebo::physics::Joint::Detach ( ) [virtual]

Detach this joint from all links.

Reimplemented in **gazebo::physics::ODEJoint** (p. 597), and **gazebo::physics::BulletJoint** (p. 190).

10.82.4.5 void gazebo::physics::Joint::DisconnectJointUpdate (event::ConnectionPtr & *_conn*) [inline]

Disconnect a boost::slot the the joint update signal.

Parameters

in	*_conn*	Connection to disconnect.

References gazebo::event::EventT< T >::Disconnect().

10.82.4.6 void gazebo::physics::Joint::FillJointMsg (msgs::Joint & *_msg*)

DEPRECATED.

See Also

Joint::FillMsg (p. 428)

10.82.4.7 void gazebo::physics::Joint::FillMsg (msgs::Joint & *_msg*)

Fill a joint message.

Parameters

out	*_msg*	Message to fill with this joint's properties.

10.82.4.8 virtual math::Vector3 gazebo::physics::Joint::GetAnchor (int *_index*) const [pure virtual]

Get the anchor point.

Parameters

in	*_index*	Index of the axis.

Returns

Anchor value for the axis.

Implemented in **gazebo::physics::ScrewJoint< ODEJoint >** (p. 761), **gazebo::physics::ScrewJoint< BulletJoint >** (p. 761), **gazebo::physics::BulletJoint** (p. 190), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::BulletHinge2Joint** (p. 181), **gazebo::physics::SliderJoint< ODEJoint >** (p. 805), **gazebo::physics::SliderJoint< BulletJoint >** (p. 805), **gazebo::physics::BulletHingeJoint** (p. 186), **gazebo::physics::BulletUniversalJoint** (p. 227), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::BulletBallJoint** (p. 168), **gazebo::physics::ODEBallJoint** (p. 575), and **gazebo::physics::ODEUniversalJoint** (p. 636).

10.82.4.9 math::Angle gazebo::physics::Joint::GetAngle (int *_index*) const

Get the angle of rotation of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

10.82.4.10 virtual math::Angle gazebo::physics::Joint::GetAngleImpl (int *_index*) const [protected], [pure virtual]

Get the angle of an axis helper function.

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the axis.

Implemented in **gazebo::physics::BulletHinge2Joint** (p. 181), **gazebo::physics::BulletHingeJoint** (p. 186), **gazebo::physics::BulletUniversalJoint** (p. 227), **gazebo::physics::BulletScrewJoint** (p. 215), **gazebo::physics::BulletSliderJoint** (p. 219), **gazebo::physics::BulletBallJoint** (p. 168), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEBallJoint** (p. 575), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::ODESliderJoint** (p. 628), and **gazebo::physics::ODEUniversalJoint** (p. 636).

10.82.4.11 LinkPtr gazebo::physics::Joint::GetChild ( ) const

Get the child link.

Returns

Pointer to the child link.

10.82.4.12 virtual double gazebo::physics::Joint::GetForce (int _index) [virtual]

Todo : not yet implemented. Get the internal forces at a this **Joint** (p. 423). Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

in	*_index*	Index of the axis.

Returns

The force applied to an axis.

Reimplemented in **gazebo::physics::BulletHingeJoint** (p. 186).

10.82.4.13 virtual math::Vector3 gazebo::physics::Joint::GetGlobalAxis (int _index) const [pure virtual]

Get the axis of rotation in global cooridnate frame.

Parameters

in	*_index*	Index of the axis to get.

Returns

Axis value for the provided index.

Implemented in **gazebo::physics::BulletHinge2Joint** (p. 181), **gazebo::physics::BulletHingeJoint** (p. 186), **gazebo::physics::BulletUniversalJoint** (p. 227), **gazebo::physics::BulletScrewJoint** (p. 215), **gazebo::physics::BulletSliderJoint** (p. 219), **gazebo::physics::BulletBallJoint** (p. 168), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::ODEUniversalJoint** (p. 637), **gazebo::physics::ODEBallJoint** (p. 575), **gazebo::physics::ODEScrewJoint** (p. 624), and **gazebo::physics::ODESliderJoint** (p. 628).

10.82.4.14 virtual math::Angle gazebo::physics::Joint::GetHighStop (int _index) [pure virtual]

Get the high stop of an axis(index).

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the high stop value.

Implemented in **gazebo::physics::BulletHinge2Joint** (p. 181), **gazebo::physics::BulletHingeJoint** (p. 186), **gazebo::physics::ODEJoint** (p. 597), **gazebo::physics::BulletUniversalJoint** (p. 227), **gazebo::physics::BallJoint< ODEJoint >** (p. 144), **gazebo::physics::BallJoint< BulletJoint >** (p. 144), **gazebo::physics::BulletScrewJoint** (p. 215), and **gazebo::physics::BulletSliderJoint** (p. 219).

10.82.4.15 virtual LinkPtr gazebo::physics::Joint::GetJointLink (int _index) const [pure virtual]

Get the link to which the joint is attached according the _index.

Parameters

in	*_index*	Index of the link to retreive.

Returns

Pointer to the request link. NULL if the index was invalid.

Implemented in **gazebo::physics::BulletJoint** (p. 190), and **gazebo::physics::ODEJoint** (p. 597).

10.82.4.16 virtual math::Vector3 gazebo::physics::Joint::GetLinkForce (unsigned int _index) const [pure virtual]

Get the forces applied to the center of mass of a **physics::Link** (p. 454) due to the existence of this **Joint** (p. 423).

Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

in	*index*	The index of the link(0 or 1).

Returns

Force applied to the link.

Implemented in **gazebo::physics::ODEJoint** (p. 597), and **gazebo::physics::BulletJoint** (p. 190).

10.82.4.17 virtual math::Vector3 gazebo::physics::Joint::GetLinkTorque (unsigned int _index) const [pure virtual]

Get the torque applied to the center of mass of a **physics::Link** (p. 454) due to the existence of this **Joint** (p. 423).

Note that the unit of torque should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons-Meters. If using imperial units (sorry), then unit of force is lb-force-inches not (lb-mass-inches), etc.

Parameters

in	*index*	The index of the link(0 or 1)

Returns

Torque applied to the link.

Implemented in **gazebo::physics::ODEJoint** (p. 597), and **gazebo::physics::BulletJoint** (p. 190).

10.82.4.18 math::Vector3 gazebo::physics::Joint::GetLocalAxis (int _index) const

Get the axis of rotation.

Parameters

in	*_index*	Index of the axis to get.

Returns

Axis value for the provided index.

10.82.4.19 virtual math::Angle gazebo::physics::Joint::GetLowStop (int *_index*) [pure virtual]

Get the low stop of an axis(index).

Parameters

in	*_index*	Index of the axis.

Returns

Angle of the low stop value.

Implemented in **gazebo::physics::BulletHinge2Joint** (p. 181), **gazebo::physics::BulletHingeJoint** (p. 186), **gazebo::physics::ODEJoint** (p. 598), **gazebo::physics::BulletUniversalJoint** (p. 227), **gazebo::physics::BallJoint< ODEJoint >** (p. 144), **gazebo::physics::BallJoint< BulletJoint >** (p. 144), **gazebo::physics::BulletScrewJoint** (p. 215), and **gazebo::physics::BulletSliderJoint** (p. 219).

10.82.4.20 virtual double gazebo::physics::Joint::GetMaxForce (int *_index*) [pure virtual]

Get the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

in	*_index*	Index of the axis.

Returns

The maximum force.

Implemented in **gazebo::physics::BulletScrewJoint** (p. 215), **gazebo::physics::BulletSliderJoint** (p. 219), **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::BulletUniversalJoint** (p. 227), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::ODESliderJoint** (p. 628), **gazebo::physics::ODEUniversalJoint** (p. 637), **gazebo::physics::BulletBallJoint** (p. 168), and **gazebo::physics::ODEBallJoint** (p. 576).

10.82.4.21 LinkPtr gazebo::physics::Joint::GetParent ( ) const

Get the parent link.

Returns

Pointer to the parent link.

10.82.4.22 JointState gazebo::physics::Joint::GetState ( )

Get the joint state.

Returns

The current joint state.

10.82.4.23 virtual double gazebo::physics::Joint::GetVelocity (int *_index*) const [pure virtual]

Get the rotation rate of an axis(index)

Parameters

in	*_index*	Index of the axis.

Returns

The rotaional velocity of the joint axis.

Implemented in **gazebo::physics::BulletScrewJoint** (p. 215), **gazebo::physics::BulletSliderJoint** (p. 219), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::BulletBallJoint** (p. 168), **gazebo::physics::ODESliderJoint** (p. 628), **gazebo::physics::ODEUniversalJoint** (p. 637), and **gazebo::physics::ODEBallJoint** (p. 576).

10.82.4.24 virtual void gazebo::physics::Joint::Init ( ) [virtual]

Initialize a joint.

Reimplemented from **gazebo::physics::Base** (p. 153).

Reimplemented in **gazebo::physics::HingeJoint< ODEJoint >** (p. 407), and **gazebo::physics::HingeJoint< BulletJoint >** (p. 407).

10.82.4.25 void gazebo::physics::Joint::Load (LinkPtr *_parent*, LinkPtr *_child*, const math::Pose & *_pose*)

Set pose, parent and child links of a **physics::Joint** (p. 423).

Parameters

in	*_parent*	Parent link.
in	*_child*	Child link.
in	*_pose*	Pose of the link.

10.82.4.26 virtual void gazebo::physics::Joint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load **physics::Joint** (p. 423) from a SDF **sdf::Element** (p. 332).

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::Base** (p. 154).

Reimplemented in **gazebo::physics::BallJoint< ODEJoint >** (p. 144), **gazebo::physics::BallJoint< BulletJoint >** (p. 144), **gazebo::physics::BulletScrewJoint** (p. 216), **gazebo::physics::ScrewJoint< ODEJoint >** (p. 761), **gazebo::physics::ScrewJoint< BulletJoint >** (p. 761), **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::BulletSliderJoint** (p. 220), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::SliderJoint< ODEJoint >** (p. 805), **gazebo::physics::SliderJoint< BulletJoint >** (p. 805), **gazebo::physics::Hinge2Joint< ODEJoint >** (p. 405), **gazebo::physics::Hinge2Joint< BulletJoint >** (p. 405), **gazebo::physics::HingeJoint< ODEJoint >** (p. 407), **gazebo::physics::HingeJoint< BulletJoint >** (p. 407), **gazebo::physics::UniversalJoint< ODEJoint >** (p. 870), **gazebo::physics::UniversalJoint< BulletJoint >** (p. 870), **gazebo::physics::BulletJoint** (p. 190), **gazebo::physics::ODEJoint** (p. 598), and **gazebo::physics::ODESliderJoint** (p. 628).

10.82.4.27 virtual void gazebo::physics::Joint::Reset ( ) [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Base** (p. 155).

Reimplemented in **gazebo::physics::BulletJoint** (p. 191), and **gazebo::physics::ODEJoint** (p. 598).

10.82.4.28 virtual void gazebo::physics::Joint::SetAnchor (int *_index*, const math::Vector3 & *_anchor*) [pure virtual]

Set the anchor point.

Parameters

in	*_index*	Indx of the axis.
in	*_anchor*	Anchor value.

Implemented in **gazebo::physics::ScrewJoint< ODEJoint >** (p. 761), **gazebo::physics::ScrewJoint< BulletJoint >** (p. 761), **gazebo::physics::BulletJoint** (p. 191), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::SliderJoint< ODEJoint >** (p. 806), **gazebo::physics::SliderJoint< BulletJoint >** (p. 806), **gazebo::physics::BulletBallJoint** (p. 168), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::ODEBallJoint** (p. 576), and **gazebo::physics::ODEUniversalJoint** (p. 637).

10.82.4.29 void gazebo::physics::Joint::SetAngle (int *_index*, math::Angle *_angle*)

If the **Joint** (p. 423) is static, Gazebo stores the state of this **Joint** (p. 423) as a scalar inside the **Joint** (p. 423) class, so this call will NOT move the joint dynamically for a static **Model** (p. 521).

But if this **Model** (p. 521) is not static, then it is updated dynamically, all the conencted children **Link** (p. 454)'s are moved as a result of the **Joint** (p. 423) angle setting. Dynamic **Joint** (p. 423) angle update is accomplished by calling **JointController::SetJointPosition** (p. 439).

Parameters

in	*_index*	Index of the axis.
in	*_angle*	Angle to set the joint to.

10.82.4.30 virtual void gazebo::physics::Joint::SetAttribute (Attribute _attr, int _index, double _value) [pure virtual]

Set a parameter for the joint.

Parameters

in	*_attr*	Attribute to set.
in	*_index*	Index of the axis.
in	*_value*	Value of the attribute.

Implemented in **gazebo::physics::ODEJoint** (p. 598), and **gazebo::physics::BulletJoint** (p. 191).

10.82.4.31 virtual void gazebo::physics::Joint::SetAxis (int _index, const math::Vector3 & _axis) [pure virtual]

Set the axis of rotation.

Parameters

in	*_index*	Index of the axis to set.
in	*_axis*	Axis value.

Implemented in **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::ODEHingeJoint** (p. 594), **gazebo::physics::BallJoint< ODEJoint >** (p. 144), **gazebo::physics::BallJoint< BulletJoint >** (p. 144), **gazebo::physics::BulletScrewJoint** (p. 216), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::ODEUniversalJoint** (p. 637), **gazebo::physics::BulletSliderJoint** (p. 220), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEHinge2Joint** (p. 590), and **gazebo::physics::ODESliderJoint** (p. 628).

10.82.4.32 virtual void gazebo::physics::Joint::SetDamping (int _index, double _damping) [pure virtual]

Set the joint damping.

Parameters

in	*_index*	Index of the axis to set.
in	*_damping*	Damping value for the axis.

Implemented in **gazebo::physics::BulletJoint** (p. 191), **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::ODEHingeJoint** (p. 594), **gazebo::physics::BulletScrewJoint** (p. 216), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::BulletSliderJoint** (p. 220), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEHinge2Joint** (p. 590), **gazebo::physics::BulletBallJoint** (p. 169), **gazebo::physics::ODEBallJoint** (p. 576), **gazebo::physics::ODESliderJoint** (p. 628), and **gazebo::physics::ODEUniversalJoint** (p. 637).

10.82.4.33 virtual void gazebo::physics::Joint::SetForce (int _index, double _force) [virtual]

Set the force applied to this **physics::Joint** (p. 423).

Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Force value.

Reimplemented in **gazebo::physics::BulletScrewJoint** (p. 216), **gazebo::physics::ODEHingeJoint** (p. 594), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::BulletSliderJoint** (p. 220), **gazebo::physics::ODEHinge2Joint** (p. 590), **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::ODEScrewJoint** (p. 625), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::ODESliderJoint** (p. 628), and **gazebo::physics::ODEUniversalJoint** (p. 637).

10.82.4.34 virtual void gazebo::physics::Joint::SetHighStop (int *_index*, const math::Angle & *_angle*) [pure virtual]

Set the high stop of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_angle*	High stop angle.

Implemented in **gazebo::physics::BulletHinge2Joint** (p. 182), **gazebo::physics::BulletHingeJoint** (p. 187), **gazebo::physics::ODEJoint** (p. 598), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::BulletBallJoint** (p. 169), **gazebo::physics::BulletScrewJoint** (p. 216), and **gazebo::physics::BulletSliderJoint** (p. 220).

10.82.4.35 virtual void gazebo::physics::Joint::SetLowStop (int *_index*, const math::Angle & *_angle*) [pure virtual]

Set the low stop of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_angle*	Low stop angle.

Implemented in **gazebo::physics::BulletHinge2Joint** (p. 183), **gazebo::physics::BulletHingeJoint** (p. 188), **gazebo::physics::ODEJoint** (p. 599), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::BulletBallJoint** (p. 169), **gazebo::physics::BulletScrewJoint** (p. 216), and **gazebo::physics::BulletSliderJoint** (p. 220).

10.82.4.36 virtual void gazebo::physics::Joint::SetMaxForce (int *_index*, double *_force*) [pure virtual]

Set the max allowed force of an axis(index).

Note that the unit of force should be consistent with the rest of the simulation scales. E.g. if you are using metric units, the unit for force is Newtons. If using imperial units (sorry), then unit of force is lb-force not (lb-mass), etc.

Parameters

in	*_index*	Index of the axis.
in	*_force*	Maximum force that can be applied to the axis.

Implemented in **gazebo::physics::BulletScrewJoint** (p. 216), **gazebo::physics::BulletSliderJoint** (p. 220), **gazebo::physics::BulletHinge2Joint** (p. 183), **gazebo::physics::ODEScrewJoint** (p. 625), **gazebo::physics::BulletUniversalJoint** (p. 228), **gazebo::physics::ODEHingeJoint** (p. 594), **gazebo::physics::BulletHingeJoint** (p. 188), **gazebo::physics::ODEHinge2Joint** (p. 590), **gazebo::physics::BulletBallJoint** (p. 169), **gazebo::physics::ODE-**

SliderJoint (p. 629), **gazebo::physics::ODEUniversalJoint** (p. 637), and **gazebo::physics::ODEBallJoint** (p. 576).

10.82.4.37 void gazebo::physics::Joint::SetModel (ModelPtr _model)

Set the model this joint belongs too.

Parameters

in	*_model*	Pointer to a model.

10.82.4.38 void gazebo::physics::Joint::SetState (const JointState & _state)

Set the joint state.

Parameters

in	*_state*	**Joint** (p. 423) state

10.82.4.39 virtual void gazebo::physics::Joint::SetVelocity (int _index, double _vel) [pure virtual]

Set the velocity of an axis(index).

Parameters

in	*_index*	Index of the axis.
in	*_vel*	Velocity.

Implemented in **gazebo::physics::BulletScrewJoint** (p. 216), **gazebo::physics::BulletSliderJoint** (p. 220), **gazebo::physics::BulletHinge2Joint** (p. 183), **gazebo::physics::ODEScrewJoint** (p. 625), **gazebo::physics::ODEHinge2Joint** (p. 590), **gazebo::physics::ODEHingeJoint** (p. 594), **gazebo::physics::BulletHingeJoint** (p. 188), **gazebo::physics::BulletUniversalJoint** (p. 229), **gazebo::physics::ODESliderJoint** (p. 629), **gazebo::physics::ODEUniversalJoint** (p. 638), **gazebo::physics::BulletBallJoint** (p. 169), and **gazebo::physics::ODEBallJoint** (p. 576).

10.82.4.40 void gazebo::physics::Joint::Update ( ) [virtual]

Update the joint.

Reimplemented from **gazebo::physics::Base** (p. 157).

10.82.4.41 virtual void gazebo::physics::Joint::UpdateParameters (sdf::ElementPtr _sdf) [virtual]

Update the parameters using new sdf values.

Parameters

in	*_sdf*	SDF values to update from.

Reimplemented from **gazebo::physics::Base** (p. 157).

10.82.5 Member Data Documentation

10.82.5.1 LinkPtr gazebo::physics::Joint::anchorLink [protected]

Anchor link.

10.82.5.2 math::Vector3 gazebo::physics::Joint::anchorPos [protected]

Anchor pose.

10.82.5.3 LinkPtr gazebo::physics::Joint::childLink [protected]

The first link this joint connects to.

10.82.5.4 double gazebo::physics::Joint::damping_coefficient [protected]

joint damping_coefficient

10.82.5.5 ModelPtr gazebo::physics::Joint::model [protected]

Pointer to the parent model.

10.82.5.6 LinkPtr gazebo::physics::Joint::parentLink [protected]

The second link this joint connects to.

The documentation for this class was generated from the following file:

- **Joint.hh**

10.83 gazebo::physics::JointController Class Reference

A class for manipulating **physics::Joint** (p. 423).

`#include <physics/physics.hh>`

Public Member Functions

- **JointController** (**ModelPtr** _model)
  Constructor.
- void **AddJoint** (**JointPtr** _joint)
  Add a joint to control.
- void **Reset** ()
  Reset all commands.
- void **SetJointPosition** (const std::string &_name, double _position)
  *Set the positions of a **Joint** (p. 423) by name.*
- void **SetJointPosition** (**JointPtr** _joint, double _position)

*Set the positions of a **Joint** (p. 423) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 804) and radians for **HingeJoint** (p. 405), etc.*

- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)

  *Set the positions of a set of **Joint** (p. 423)'s.*

- void **Update** ()

  Update the joint control.

10.83.1 Detailed Description

A class for manipulating **physics::Joint** (p. 423).

10.83.2 Constructor & Destructor Documentation

10.83.2.1 gazebo::physics::JointController::JointController (ModelPtr *_model*) [explicit]

Constructor.

Parameters

in	*_model*	**Model** (p. 521) that uses this joint controller.

10.83.3 Member Function Documentation

10.83.3.1 void gazebo::physics::JointController::AddJoint (JointPtr *_joint*)

Add a joint to control.

Parameters

in	*_joint*	**Joint** (p. 423) to control.

10.83.3.2 void gazebo::physics::JointController::Reset ( )

Reset all commands.

10.83.3.3 void gazebo::physics::JointController::SetJointPosition (const std::string & *_name,* double *_position*)

Set the positions of a **Joint** (p. 423) by name.

See Also

JointController::SetJointPosition(JointPtr, double) (p. 439)

10.83.3.4 void gazebo::physics::JointController::SetJointPosition (JointPtr *_joint,* double *_position*)

Set the positions of a **Joint** (p. 423) by name The position is specified in native units, which means, if you are using metric system, it's meters for **SliderJoint** (p. 804) and radians for **HingeJoint** (p. 405), etc.

Implementation: In order to change the position of a **Joint** (p. 423) inside a **Model** (p. 521), this call must recursively crawl through all the connected children **Link** (p. 454)'s in this **Model** (p. 521), and update each **Link** (p. 454) Pose affected by this **Joint** (p. 423) angle update. Warning: There is no constraint satisfaction being done here, traversal through the kinematic graph has unexpected behavior if you try to set the joint position of a link inside a loop structure.

Parameters

in	*_joint*	**Joint** (p. 423) to set.
in	*_position*	Position of the joint.

10.83.3.5 void gazebo::physics::JointController::SetJointPositions (const std::map< std::string, double > & *_jointPositions*)

Set the positions of a set of **Joint** (p. 423)'s.

See Also

JointController::SetJointPosition(JointPtr, double) (p. 439)

10.83.3.6 void gazebo::physics::JointController::Update ( )

Update the joint control.

The documentation for this class was generated from the following file:

- **JointController.hh**

10.84 gazebo::physics::JointFeedback Class Reference

Feedback information from a joint.

`#include <physics/physics.hh>`

Public Member Functions

- **JointFeedback** & **operator=** (const **JointFeedback** &_feedback)

  Operator =.

Public Attributes

- **math::Vector3 body1Force**

  Force on the first link.
- **math::Vector3 body1Torque**

  Torque on the first link.
- **math::Vector3 body2Force**

  Force on the second link.
- **math::Vector3 body2Torque**

  Torque on the second link.

10.84.1 Detailed Description

Feedback information from a joint.

These are forces and torques on parent and child **Link** (p. 454)'s

10.84.2 Member Function Documentation

10.84.2.1 JointFeedback& gazebo::physics::JointFeedback::operator= (const JointFeedback & *_feedback*) [inline]

Operator =.

Parameters

in	*_feedback*	**Joint** (p. 423) feedback to set from.

Returns

∗this

References body1Force, body1Torque, body2Force, and body2Torque.

10.84.3 Member Data Documentation

10.84.3.1 math::Vector3 gazebo::physics::JointFeedback::body1Force

Force on the first link.

Referenced by operator=().

10.84.3.2 math::Vector3 gazebo::physics::JointFeedback::body1Torque

Torque on the first link.

Referenced by operator=().

10.84.3.3 math::Vector3 gazebo::physics::JointFeedback::body2Force

Force on the second link.

Referenced by operator=().

10.84.3.4 math::Vector3 gazebo::physics::JointFeedback::body2Torque

Torque on the second link.

Referenced by operator=().

The documentation for this class was generated from the following file:

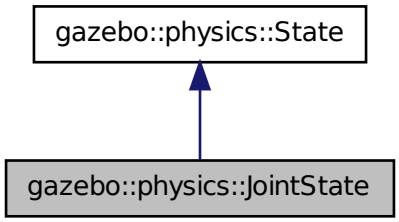
- **JointFeedback.hh**

10.85 gazebo::physics::JointState Class Reference

keeps track of state of a **physics::Joint** (p. 423)

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::JointState:



Public Member Functions

- **JointState** ()

  Default constructor.
- **JointState** (**JointPtr** _joint)

  Constructor.
- virtual ~**JointState** ()

  Destructor.
- **math::Angle GetAngle** (unsigned int _axis) const

  Get the joint angle.
- unsigned int **GetAngleCount** () const

  Get the number of angles.
- virtual void **Load** (**sdf::ElementPtr** _elem)

  Load state from SDF element.

Additional Inherited Members

10.85.1 Detailed Description

keeps track of state of a **physics::Joint** (p. 423)

10.85.2 Constructor & Destructor Documentation

10.85.2.1 gazebo::physics::JointState::JointState ( )

Default constructor.

10.85.2.2 gazebo::physics::JointState::JointState (JointPtr _joint) [explicit]

Constructor.

Parameters

in	*_joint*	**Joint** (p. 423) to get the state of.

10.85.2.3 virtual gazebo::physics::JointState::~JointState ( ) [virtual]

Destructor.

10.85.3 Member Function Documentation

10.85.3.1 math::Angle gazebo::physics::JointState::GetAngle (unsigned int _axis) const

Get the joint angle.

Parameters

in	*_axis*	The axis index.

Returns

Angle of the axis.

10.85.3.2 unsigned int gazebo::physics::JointState::GetAngleCount ( ) const

Get the number of angles.

Returns

The number of angles.

10.85.3.3 virtual void gazebo::physics::JointState::Load (sdf::ElementPtr _elem) [virtual]

Load state from SDF element.

Parameters

in	*_elem*	SDf values to load from.

Implements **gazebo::physics::State** (p. 815).

The documentation for this class was generated from the following file:

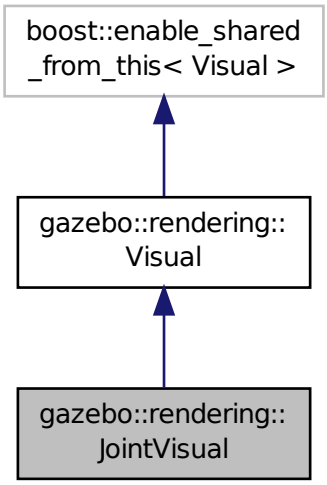
- **JointState.hh**

10.86 gazebo::rendering::JointVisual Class Reference

Visualization for joints.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::JointVisual:



Public Member Functions

- **JointVisual** (const std::string &_name, **VisualPtr** _vis)
  Constructor.
- virtual ~**JointVisual** ()
  Destructor.
- void **Load** (ConstJointPtr &_msg)
  Load the visual based on a message.

Additional Inherited Members

10.86.1 Detailed Description

Visualization for joints.

10.86.2 Constructor & Destructor Documentation

10.86.2.1 gazebo::rendering::JointVisual::JointVisual (const std::string & *_name*, VisualPtr *_vis*)

Constructor.

Parameters

in	*_name*	Name of the visual
in	*_vis*	Pointer to the parent visual

10.86.2.2 virtual gazebo::rendering::JointVisual::~JointVisual ( ) [virtual]

Destructor.

10.86.3 Member Function Documentation

10.86.3.1 void gazebo::rendering::JointVisual::Load (ConstJointPtr & *_msg*)

Load the visual based on a message.

Parameters

in	*_msg*	Joint message

The documentation for this class was generated from the following file:

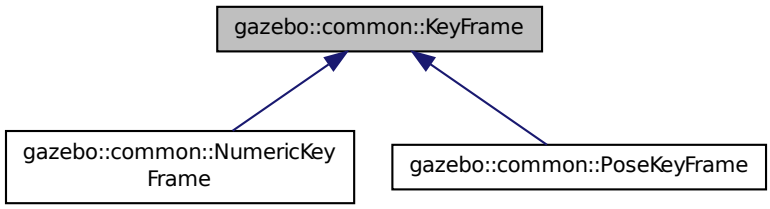
- **JointVisual.hh**

10.87 gazebo::common::KeyFrame Class Reference

A key frame in an animation.

`#include <KeyFrame.hh>`

Inheritance diagram for gazebo::common::KeyFrame:



Public Member Functions

- **KeyFrame** (double _time)

  Constructor.
- virtual ~**KeyFrame** ()

  Destructor.

- double **GetTime** () const

  Get the time of the keyframe.

Protected Attributes

- double **time**

  time of key frame

10.87.1 Detailed Description

A key frame in an animation.

10.87.2 Constructor & Destructor Documentation

10.87.2.1 gazebo::common::KeyFrame::KeyFrame (double *_time*)

Constructor.

Parameters

in	*_time*	**Time** (p. 840) of the keyframe in seconds

10.87.2.2 virtual gazebo::common::KeyFrame::~KeyFrame ( ) [virtual]

Destructor.

10.87.3 Member Function Documentation

10.87.3.1 double gazebo::common::KeyFrame::GetTime ( ) const

Get the time of the keyframe.

Returns

the time

10.87.4 Member Data Documentation

10.87.4.1 double gazebo::common::KeyFrame::time [protected]

time of key frame

The documentation for this class was generated from the following file:

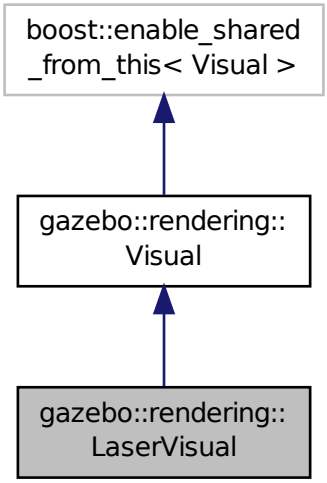
- **KeyFrame.hh**

10.88 gazebo::rendering::LaserVisual Class Reference

Visualization for laser data.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::LaserVisual:



Public Member Functions

- **LaserVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

  Constructor.
- virtual ~**LaserVisual** ()

  Destructor.
- virtual void **SetEmissive** (const **common::Color** &_color)

  Documentation inherited from parent.

Additional Inherited Members

10.88.1 Detailed Description

Visualization for laser data.

10.88.2 Constructor & Destructor Documentation

10.88.2.1 gazebo::rendering::LaserVisual::LaserVisual (const std::string & _name, VisualPtr _vis, const std::string & _topicName)

Constructor.

Parameters

in	*_name*	Name of the visual.
in	*_vis*	Pointer to the parent **Visual** (p. 931).
in	*_topicName*	Name of the topic that has laser data.

10.88.2.2 virtual gazebo::rendering::LaserVisual::~LaserVisual ( ) [virtual]

Destructor.

10.88.3 Member Function Documentation

10.88.3.1 virtual void gazebo::rendering::LaserVisual::SetEmissive (const common::Color & _color) [virtual]

Documentation inherited from parent.

Reimplemented from **gazebo::rendering::Visual** (p. 944).

The documentation for this class was generated from the following file:

- **LaserVisual.hh**

10.89 gazebo::rendering::Light Class Reference

A light source.

`#include <rendering/rendering.hh>`

Public Member Functions

- **Light** (**Scene** *_scene)
  Constructor.
- virtual ~**Light** ()
  Destructor.
- void **FillMsg** (msgs::Light &_msg) const
  Fill the contents of a light message.
- **common::Color GetDiffuseColor** () const
  Get the diffuse color.
- **math::Vector3 GetDirection** () const
  Get the direction.
- std::string **GetName** () const
  Get the name of the visual.
- **math::Vector3 GetPosition** () const
  Get the position of the light.

- **common::Color GetSpecularColor** () const

  Get the specular color.
- std::string **GetType** () const

  Get the type of the light.
- void **Load** (**sdf::ElementPtr** _sdf)

  Load the light using a set of SDF parameters.
- void **Load** ()

  Load the light using default parameters.
- void **LoadFromMsg** (ConstLightPtr &_msg)

  Load from a light message.
- void **SetAttenuation** (double _constant, double _linear, double _quadratic)

  Set the attenuation.
- void **SetCastShadows** (const bool &_cast)

  Set cast shadows.
- void **SetDiffuseColor** (const **common::Color** &_color)

  Set the diffuse color.
- void **SetDirection** (const **math::Vector3** &_dir)

  Set the direction.
- void **SetLightType** (const std::string &_type)

  Set the light type.
- void **SetName** (const std::string &_name)

  Set the name of the visual.
- void **SetPosition** (const **math::Vector3** &_p)

  Set the position of the light.
- void **SetRange** (const double &_range)

  Set the range.
- virtual bool **SetSelected** (bool _s)

  Set whether this entity has been selected by the user through the gui.
- void **SetSpecularColor** (const **common::Color** &_color)

  Set the specular color.
- void **SetSpotFalloff** (const double &_value)

  Set the spot light falloff.
- void **SetSpotInnerAngle** (const double &_angle)

  Set the spot light inner angle.
- void **SetSpotOuterAngle** (const double &_angle)

  Set the spot light outer angle.
- void **ShowVisual** (bool _s)

  Set whether to show the visual.
- void **ToggleShowVisual** ()
- void **UpdateFromMsg** (ConstLightPtr &_msg)

  Update a light source from a message.

Protected Member Functions

- virtual void **OnPoseChange** ()

  On pose change callback.

10.89.1 Detailed Description

A light source.

There are three types of lights: Point, Spot, and Directional. This class encapsulates all three. Point lights are light light bulbs, spot lights project a cone of light, and directional lights are light sun light.

10.89.2 Constructor & Destructor Documentation

10.89.2.1 gazebo::rendering::Light::Light (Scene * _scene)

Constructor.

Parameters

in	*_scene*	Pointer to the scene that contains the **Light** (p. 448).

10.89.2.2 virtual gazebo::rendering::Light::~Light ( ) [virtual]

Destructor.

10.89.3 Member Function Documentation

10.89.3.1 void gazebo::rendering::Light::FillMsg (msgs::Light & _msg) const

Fill the contents of a light message.

Parameters

out	*_msg*	Message to fill.

10.89.3.2 common::Color gazebo::rendering::Light::GetDiffuseColor ( ) const

Get the diffuse color.

Returns

The light's diffuse color.

10.89.3.3 math::Vector3 gazebo::rendering::Light::GetDirection ( ) const

Get the direction.

Returns

The light's direction.

10.89.3.4 std::string gazebo::rendering::Light::GetName ( ) const

Get the name of the visual.

Returns

The light's name.

10.89.3.5 math::Vector3 gazebo::rendering::Light::GetPosition ( ) const

Get the position of the light.

Returns

The position of the light

10.89.3.6 common::Color gazebo::rendering::Light::GetSpecularColor ( ) const

Get the specular color.

Returns

The specular color

10.89.3.7 std::string gazebo::rendering::Light::GetType ( ) const

Get the type of the light.

Returns

The light type: "point", "spot", "directional".

10.89.3.8 void gazebo::rendering::Light::Load (sdf::ElementPtr *_sdf*)

Load the light using a set of SDF parameters.

Parameters

in	*_sdf*	Pointer to the SDF containing the **Light** (p. 448) description.

10.89.3.9 void gazebo::rendering::Light::Load ( )

Load the light using default parameters.

10.89.3.10 void gazebo::rendering::Light::LoadFromMsg (ConstLightPtr & *_msg*)

Load from a light message.

Parameters

in	*_msg*	Containing the light information.

10.89.3.11 virtual void gazebo::rendering::Light::OnPoseChange ( ) [inline], [protected], [virtual]

On pose change callback.

10.89.3.12 void gazebo::rendering::Light::SetAttenuation (double *_constant,* double *_linear,* double *_quadratic*)

Set the attenuation.

Parameters

in	*_contant*	Constant attenuation
in	*_linear*	Linear attenuation
in	*_quadratic*	Quadratic attenuation

10.89.3.13 void gazebo::rendering::Light::SetCastShadows (const bool & *_cast*)

Set cast shadows.

Parameters

in	*_cast*	Set to true to cast shadows.

10.89.3.14 void gazebo::rendering::Light::SetDiffuseColor (const **common::Color** & *_color*)

Set the diffuse color.

Parameters

in	*_color*	**Light** (p. 448) diffuse color.

10.89.3.15 void gazebo::rendering::Light::SetDirection (const **math::Vector3** & *_dir*)

Set the direction.

Parameters

in	*_dir*	Set the light's direction. Only applicable to spot and directional lights.

10.89.3.16 void gazebo::rendering::Light::SetLightType (const std::string & *_type*)

Set the light type.

Parameters

in	*_type*	The light type: "point", "spot", "directional"

10.89.3.17 void gazebo::rendering::Light::SetName (const std::string & _name)

Set the name of the visual.

Parameters

in	*_name*	Name of the light source.

10.89.3.18 void gazebo::rendering::Light::SetPosition (const math::Vector3 & _p)

Set the position of the light.

Parameters

in	*_p*	New position for the light

10.89.3.19 void gazebo::rendering::Light::SetRange (const double & _range)

Set the range.

Parameters

in	*_range*	Rage of the light in meters.

10.89.3.20 virtual bool gazebo::rendering::Light::SetSelected (bool _s) [virtual]

Set whether this entity has been selected by the user through the gui.

Parameters

in	*_s*	Set to True when the light is selected by the user.

10.89.3.21 void gazebo::rendering::Light::SetSpecularColor (const common::Color & _color)

Set the specular color.

Parameters

in	*_color*	The specular color

10.89.3.22 void gazebo::rendering::Light::SetSpotFalloff (const double & _value)

Set the spot light falloff.

Parameters

in	*_value*	Falloff value

10.89.3.23 void gazebo::rendering::Light::SetSpotInnerAngle (const double & _angle)

Set the spot light inner angle.

Parameters

in	*_angle*	Inner angle in radians

10.89.3.24 void gazebo::rendering::Light::SetSpotOuterAngle (const double & _angle)

Set the spot light outer angle.

Parameters

in	*_angle*	Outer angle in radians

10.89.3.25 void gazebo::rendering::Light::ShowVisual (bool _s)

Set whether to show the visual.

Parameters

in	*_s*	Set to true to draw a representation of the light.

10.89.3.26 void gazebo::rendering::Light::ToggleShowVisual ( )

10.89.3.27 void gazebo::rendering::Light::UpdateFromMsg (ConstLightPtr & _msg)

Update a light source from a message.

Parameters

in	*_msg*	**Light** (p. 448) message to update from

The documentation for this class was generated from the following file:

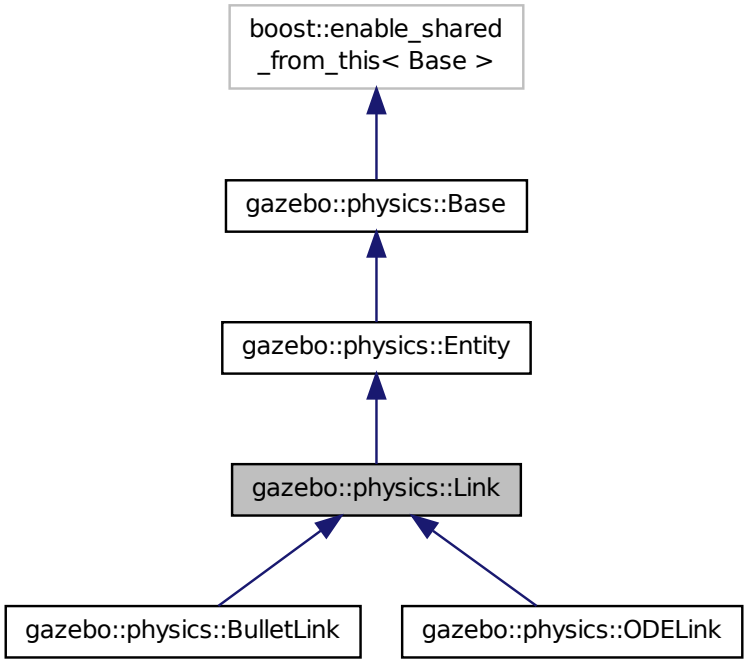
- **Light.hh**

10.90 gazebo::physics::Link Class Reference

Link (p. 454) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Link:



Public Member Functions

- **Link** (**EntityPtr** _parent)

  Constructor.
- virtual ~**Link** ()

  Destructor.
- void **AddChildJoint** (**JointPtr** _joint)

  *Joints that have this **Link** (p. 454) as a parent **Link** (p. 454).*
- virtual void **AddForce** (const **math::Vector3** &_force)=0

  Add a force to the body.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relPos)=0

  Add a force to the body at position expressed to the body's own frame of reference.
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)=0

  Add a force to the body using a global position.
- void **AddParentJoint** (**JointPtr** _joint)

  *Joints that have this **Link** (p. 454) as a child **Link** (p. 454).*
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)=0

  Add a force to the body, components are relative to the body's own frame of reference.
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)=0

  Add a torque to the body, components are relative to the body's own frame of reference.

- virtual void **AddTorque** (const **math::Vector3** &_torque)=0

  Add a torque to the body.
- void **AttachStaticModel** (**ModelPtr** &_model, const **math::Pose** &_offset)

  Attach a static model to this link.
- template<typename T >
  event::ConnectionPtr ConnectEnabled (T _subscriber)

  Connect to the add entity signal.
- void **DetachAllStaticModels** ()

  Detach all static models from this link.
- void **DetachStaticModel** (const std::string &_modelName)

  Detach a static model from this link.
- void **DisconnectEnabled** (**event::ConnectionPtr** &_conn)

  Disconnect to the add entity signal.
- void **FillLinkMsg** (msgs::Link &_msg) **GAZEBO_DEPRECATED**

  DEPRECATED.
- void **FillMsg** (msgs::Link &_msg)

  Fill a link message.
- void **Fini** ()

  Finalize the body.
- double **GetAngularDamping** () const

  Get the angular damping factor.
- virtual **math::Box GetBoundingBox** () const

  Get the bounding box for the link and all the child elements.
- **Link_V GetChildJointsLinks** () const

  Returns a vector of children Links connected by joints.
- **CollisionPtr GetCollision** (const std::string &_name)

  Get a child collision by name.
- **CollisionPtr GetCollision** (unsigned int _index) const

  Get a child collision by index.
- **CollisionPtr GetCollisionById** (unsigned int _id) const

  Get a collision by id.
- virtual bool **GetEnabled** () const =0

  Get whether this body is enabled in the physics engine.
- virtual bool **GetGravityMode** ()=0

  Get the gravity mode.
- **InertialPtr GetInertial** () const

  Get the inertia of the link.
- virtual bool **GetKinematic** () const

  Implement this function.
- double **GetLinearDamping** () const

  Get the linear damping factor.
- **ModelPtr GetModel** () const

  Get the model that this body belongs to.
- **Link_V GetParentJointsLinks** () const

  Returns a vector of parent Links connected by joints.
- **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the body.

- **math::Vector3 GetRelativeAngularVel** () const

  Get the angular velocity of the body.
- **math::Vector3 GetRelativeForce** () const

  Get the force applied to the body.
- **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the body.
- **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the body.
- **math::Vector3 GetRelativeTorque** () const

  Get the torque applied to the body.
- bool **GetSelfCollide** ()

  Get Self-Collision Flag, if this is true, this body will collide with other bodies even if they share the same parent.
- unsigned int **GetSensorCount** () const

  Get sensor count.
- std::string **GetSensorName** (unsigned int _index) const

  Get sensor name.
- **LinkState GetState** ()

  Get the link state.
- **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the body in the world frame.
- virtual **math::Vector3 GetWorldForce** () const =0

  Get the force applied to the body in the world frame.
- **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the body in the world frame.
- virtual **math::Vector3 GetWorldTorque** () const =0

  Get the torque applied to the body in the world frame.
- virtual void **Init** ()

  Initialize the body.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the body based on an SDF element.
- virtual void **OnPoseChange** ()

  This function is called when the entity's (or one of its parents) pose of the parent has changed.
- void **ProcessMsg** (const msgs::Link &_msg)

  Update parameters from a message.
- void **RemoveChildJoint** (**JointPtr** _joint)

  *Remove Joints that have this **Link** (p. 454) as a parent **Link** (p. 454).*
- void **RemoveParentJoint** (**JointPtr** _joint)

  *Remove Joints that have this **Link** (p. 454) as a child **Link** (p. 454).*
- void **Reset** ()

  Reset the link.
- void **SetAngularAccel** (const **math::Vector3** &_accel)

  Set the angular acceleration of the body.
- virtual void **SetAngularDamping** (double _damping)=0

  Set the angular damping factor.
- virtual void **SetAngularVel** (const **math::Vector3** &_vel)=0

  Set the angular velocity of the body.
- virtual void **SetAutoDisable** (bool _disable)=0

Allow the link to auto disable.

- void **SetCollideMode** (const std::string &_mode)

  Set the collide mode of the body.
- virtual void **SetEnabled** (bool _enable) const =0

  Set whether this body is enabled.
- virtual void **SetForce** (const **math::Vector3** &_force)=0

  Set the force applied to the body.
- virtual void **SetGravityMode** (bool _mode)=0

  Set whether gravity affects this body.
- void **SetInertial** (const **InertialPtr** &_inertial)

  Set the mass of the link.
- virtual void **SetKinematic** (const bool &_kinematic)

  Implement this function.
- void **SetLaserRetro** (float _retro)

  Set the laser retro reflectiveness.
- void **SetLinearAccel** (const **math::Vector3** &_accel)

  Set the linear acceleration of the body.
- virtual void **SetLinearDamping** (double _damping)=0

  Set the linear damping factor.
- virtual void **SetLinearVel** (const **math::Vector3** &_vel)=0

  Set the linear velocity of the body.
- virtual bool **SetSelected** (bool _set)

  Set whether this entity has been selected by the user through the gui.
- virtual void **SetSelfCollide** (bool _collide)=0

  Set whether this body will collide with others in the model.
- void **SetState** (const **LinkState** &_state)

  Set the current link state.
- virtual void **SetTorque** (const **math::Vector3** &_torque)=0

  Set the torque applied to the body.
- virtual void **Update** ()

  Update the body.
- virtual void **UpdateMass** ()

  Update the mass matrix.
- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  Update the parameters using new sdf values.
- virtual void **UpdateSurface** ()

  Update surface parameters.

Protected Attributes

- **math::Vector3 angularAccel**

  Angular acceleration.
- std::vector< **math::Pose** > **attachedModelsOffset**

  Offsets for the attached models.
- std::vector< std::string > **cgVisuals**

  Center of gravity visual elements.
- **InertialPtr inertial**

Inertial *(p. 414) properties.*

- **math::Vector3 linearAccel**

  Linear acceleration.

- std::vector< std::string > **visuals**

  Link *(p. 454) visual elements.*

Additional Inherited Members

10.90.1 Detailed Description

Link (p. 454) class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.

10.90.2 Constructor & Destructor Documentation

10.90.2.1 gazebo::physics::Link::Link (EntityPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent of this link.

10.90.2.2 virtual gazebo::physics::Link::~Link ( ) [virtual]

Destructor.

10.90.3 Member Function Documentation

10.90.3.1 void gazebo::physics::Link::AddChildJoint (JointPtr *_joint*)

Joints that have this **Link** (p. 454) as a parent **Link** (p. 454).

Parameters

in	*_joint*	**Joint** (p. 423) that is a child of this link.

10.90.3.2 virtual void gazebo::physics::Link::AddForce (const math::Vector3 & *_force*) [pure virtual]

Add a force to the body.

Parameters

in	*_force*	Force to add.

Implemented in **gazebo::physics::BulletLink** (p. 194), and **gazebo::physics::ODELink** (p. 602).

10.90.3.3 virtual void gazebo::physics::Link::AddForceAtRelativePosition (const math::Vector3 & _force, const math::Vector3 & _relPos) [pure virtual]

Add a force to the body at position expressed to the body's own frame of reference.

Parameters

in	*_force*	Force to add.
in	*_relPos*	Position on the link to add the force.

Implemented in **gazebo::physics::BulletLink** (p. 194), and **gazebo::physics::ODELink** (p. 603).

10.90.3.4 virtual void gazebo::physics::Link::AddForceAtWorldPosition (const math::Vector3 & _force, const math::Vector3 & _pos) [pure virtual]

Add a force to the body using a global position.

Parameters

in	*_force*	Force to add.
in	*_pos*	Position in global coord frame to add the force.

Implemented in **gazebo::physics::BulletLink** (p. 194), and **gazebo::physics::ODELink** (p. 603).

10.90.3.5 void gazebo::physics::Link::AddParentJoint (JointPtr _joint)

Joints that have this **Link** (p. 454) as a child **Link** (p. 454).

Parameters

in	*_joint*	**Joint** (p. 423) that is a parent of this link.

10.90.3.6 virtual void gazebo::physics::Link::AddRelativeForce (const math::Vector3 & _force) [pure virtual]

Add a force to the body, components are relative to the body's own frame of reference.

Parameters

in	*_force*	Force to add.

Implemented in **gazebo::physics::BulletLink** (p. 195), and **gazebo::physics::ODELink** (p. 603).

10.90.3.7 virtual void gazebo::physics::Link::AddRelativeTorque (const math::Vector3 & _torque) [pure virtual]

Add a torque to the body, components are relative to the body's own frame of reference.

Parameters

in	*_torque*	Torque value to add.

Implemented in **gazebo::physics::BulletLink** (p. 195), and **gazebo::physics::ODELink** (p. 603).

10.90.3.8 virtual void gazebo::physics::Link::AddTorque (const math::Vector3 & _torque) [pure virtual]

Add a torque to the body.

Parameters

in	*_torque*	Torque value to add to the link.

Implemented in **gazebo::physics::BulletLink** (p. 195), and **gazebo::physics::ODELink** (p. 603).

10.90.3.9 void gazebo::physics::Link::AttachStaticModel (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.

10.90.3.10 template<typename T > event::ConnectionPtr gazebo::physics::Link::ConnectEnabled (T _subscriber) [inline]

Connect to the add entity signal.

Parameters

in	*_subscriber*	Subsciber callback function.

Returns

Pointer to the connection, which must be kept in scope.

References gazebo::event::EventT< T >::Connect().

10.90.3.11 void gazebo::physics::Link::DetachAllStaticModels ( )

Detach all static models from this link.

10.90.3.12 void gazebo::physics::Link::DetachStaticModel (const std::string & _modelName)

Detach a static model from this link.

Parameters

in	*_modelName*	Name of an attached model to detach.

10.90.3.13 void gazebo::physics::Link::DisconnectEnabled (event::ConnectionPtr & _conn) [inline]

Disconnect to the add entity signal.

Parameters

in	*_conn*	Connection pointer to disconnect.

References gazebo::event::EventT< T >::Disconnect().

10.90.3.14 void gazebo::physics::Link::FillLinkMsg (msgs::Link & *_msg*)

DEPRECATED.

10.90.3.15 void gazebo::physics::Link::FillMsg (msgs::Link & *_msg*)

Fill a link message.

Parameters

out	*_msg*	Message to fill

10.90.3.16 void gazebo::physics::Link::Fini ( ) [virtual]

Finalize the body.

Reimplemented from **gazebo::physics::Entity** (p. 342).

Reimplemented in **gazebo::physics::ODELink** (p. 603).

10.90.3.17 double gazebo::physics::Link::GetAngularDamping ( ) const

Get the angular damping factor.

Returns

Angular damping.

10.90.3.18 virtual math::Box gazebo::physics::Link::GetBoundingBox ( ) const [virtual]

Get the bounding box for the link and all the child elements.

Returns

The link's bounding box.

Reimplemented from **gazebo::physics::Entity** (p. 342).

10.90.3.19 Link_V gazebo::physics::Link::GetChildJointsLinks ( ) const

Returns a vector of children Links connected by joints.

Returns

A vector of children Links connected by joints.

10.90.3.20 CollisionPtr gazebo::physics::Link::GetCollision (const std::string & _name)

Get a child collision by name.

Parameters

in	*_name*	Name of the collision object.

Returns

Pointer to the collision, NULL if the name was not found.

10.90.3.21 CollisionPtr gazebo::physics::Link::GetCollision (unsigned int _index) const

Get a child collision by index.

Parameters

in	*_index*	Index of the collision object.

Returns

Pointer to the collision, NULL if the name was not found.

10.90.3.22 CollisionPtr gazebo::physics::Link::GetCollisionById (unsigned int _id) const

Get a collision by id.

Parameters

in	*_id*	Id of the collision object to find.

Returns

Pointer to the collision, NULL if the id is invalid.

10.90.3.23 virtual bool gazebo::physics::Link::GetEnabled ( ) const [pure virtual]

Get whether this body is enabled in the physics engine.

Returns

True if the link is enabled.

Implemented in **gazebo::physics::BulletLink** (p. 195), and **gazebo::physics::ODELink** (p. 603).

10.90.3.24 virtual bool gazebo::physics::Link::GetGravityMode ( ) [pure virtual]

Get the gravity mode.

Returns

True if gravity is enabled.

Implemented in **gazebo::physics::ODELink** (p. 603), and **gazebo::physics::BulletLink** (p. 195).

10.90.3.25 InertialPtr gazebo::physics::Link::GetInertial ( ) const [inline]

Get the inertia of the link.

Returns

Inertia of the link.

References inertial.

10.90.3.26 virtual bool gazebo::physics::Link::GetKinematic ( ) const [inline], [virtual]

Implement this function.

Get whether this body is in the kinematic state.

Returns

True if the link is kinematic only.

Reimplemented in **gazebo::physics::ODELink** (p. 604).

10.90.3.27 double gazebo::physics::Link::GetLinearDamping ( ) const

Get the linear damping factor.

Returns

Linear damping.

10.90.3.28 ModelPtr gazebo::physics::Link::GetModel ( ) const

Get the model that this body belongs to.

Returns

Model (p. 521) that this body belongs to.

10.90.3.29 Link_V gazebo::physics::Link::GetParentJointsLinks ( ) const

Returns a vector of parent Links connected by joints.

Returns

Vector of parent Links connected by joints.

10.90.3.30 math::Vector3 gazebo::physics::Link::GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the body.

Returns

Angular acceleration of the body.

Reimplemented from **gazebo::physics::Entity** (p. 343).

10.90.3.31 math::Vector3 gazebo::physics::Link::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the body.

Returns

Angular velocity of the body.

Reimplemented from **gazebo::physics::Entity** (p. 343).

10.90.3.32 math::Vector3 gazebo::physics::Link::GetRelativeForce ( ) const

Get the force applied to the body.

Returns

Force applied to the body.

10.90.3.33 math::Vector3 gazebo::physics::Link::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the body.

Returns

Linear acceleration of the body.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.90.3.34 math::Vector3 gazebo::physics::Link::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the body.

Returns

Linear velocity of the body.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.90.3.35 math::Vector3 gazebo::physics::Link::GetRelativeTorque ( ) const

Get the torque applied to the body.

Returns

Torque applied to the body.

10.90.3.36 bool gazebo::physics::Link::GetSelfCollide ( )

Get Self-Collision Flag, if this is true, this body will collide with other bodies even if they share the same parent.

Returns

True if self collision is enabled.

10.90.3.37 unsigned int gazebo::physics::Link::GetSensorCount ( ) const

Get sensor count.

This will return the number of sensors created by the link when it was loaded. This function is commonly used with **Link::GetSensorName** (p. 466).

Returns

The number of sensors created by the link.

10.90.3.38 std::string gazebo::physics::Link::GetSensorName (unsigned int *_index*) const

Get sensor name.

Get the name of a sensor based on an index. The index should be in the range of 0...**Link::GetSensorCount()** (p. 466).

Note

A **Link** (p. 454) does not manage or maintain a pointer to a **sensors::Sensor** (p. 765). Access to a Sensor object is accomplished through the **sensors::SensorManager** (p. 774). This was done to separate the physics engine from the sensor engine.

Parameters

in	*_index*	Index of the sensor name.

Returns

The name of the sensor, or empty string if the index is out of bounds.

10.90.3.39 LinkState gazebo::physics::Link::GetState ( )

Get the link state.

Returns

The state of this link.

10.90.3.40 math::Vector3 gazebo::physics::Link::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the body in the world frame.

Returns

Angular acceleration of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.90.3.41 virtual math::Vector3 gazebo::physics::Link::GetWorldForce ( ) const [pure virtual]

Get the force applied to the body in the world frame.

Returns

Force applied to the body in the world frame.

Implemented in **gazebo::physics::ODELink** (p. 604), and **gazebo::physics::BulletLink** (p. 195).

10.90.3.42 math::Vector3 gazebo::physics::Link::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the body in the world frame.

Returns

Linear acceleration of the body in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 345).

10.90.3.43 virtual math::Vector3 gazebo::physics::Link::GetWorldTorque ( ) const [pure virtual]

Get the torque applied to the body in the world frame.

Returns

Torque applied to the body in the world frame.

Implemented in **gazebo::physics::ODELink** (p. 604), and **gazebo::physics::BulletLink** (p. 196).

10.90.3.44 virtual void gazebo::physics::Link::Init ( ) [virtual]

Initialize the body.

Reimplemented from **gazebo::physics::Base** (p. 153).

Reimplemented in **gazebo::physics::BulletLink** (p. 196), and **gazebo::physics::ODELink** (p. 604).

10.90.3.45 virtual void gazebo::physics::Link::Load (sdf::ElementPtr _sdf) [virtual]

Load the body based on an SDF element.

Parameters

in	*_sdf*	SDF parameters.

Reimplemented from **gazebo::physics::Entity** (p. 346).

Reimplemented in **gazebo::physics::BulletLink** (p. 196), and **gazebo::physics::ODELink** (p. 605).

10.90.3.46 virtual void gazebo::physics::Link::OnPoseChange ( ) [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implements **gazebo::physics::Entity** (p. 346).

Reimplemented in **gazebo::physics::BulletLink** (p. 196), and **gazebo::physics::ODELink** (p. 605).

10.90.3.47 void gazebo::physics::Link::ProcessMsg (const msgs::Link & _msg)

Update parameters from a message.

Parameters

in	*_msg*	Message to read.

10.90.3.48 void gazebo::physics::Link::RemoveChildJoint (JointPtr _joint)

Remove Joints that have this **Link** (p. 454) as a parent **Link** (p. 454).

Parameters

in	*_joint*	**Joint** (p. 423) that is a child of this link.

10.90.3.49 void gazebo::physics::Link::RemoveParentJoint (JointPtr _joint)

Remove Joints that have this **Link** (p. 454) as a child **Link** (p. 454).

Parameters

in	*_joint*	**Joint** (p. 423) that is a parent of this link.

10.90.3.50 void gazebo::physics::Link::Reset ( ) [virtual]

Reset the link.

Reimplemented from **gazebo::physics::Entity** (p. 346).

10.90.3.51 void gazebo::physics::Link::SetAngularAccel (const math::Vector3 & _accel)

Set the angular acceleration of the body.

Parameters

in	*_accel*	Angular acceleration.

10.90.3.52 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::ODELink** (p. 605), and **gazebo::physics::BulletLink** (p. 196).

10.90.3.53 virtual void gazebo::physics::Link::SetAngularVel (const math::Vector3 & _vel) [pure virtual]

Set the angular velocity of the body.

Parameters

in	*_vel*	Angular velocity.

Implemented in **gazebo::physics::ODELink** (p. 605), and **gazebo::physics::BulletLink** (p. 196).

10.90.3.54 virtual void gazebo::physics::Link::SetAutoDisable (bool _disable) [pure virtual]

Allow the link to auto disable.

Parameters

in	*_disable*	If true, the link is allowed to auto disable.

Implemented in **gazebo::physics::ODELink** (p. 605), and **gazebo::physics::BulletLink** (p. 196).

10.90.3.55 void gazebo::physics::Link::SetCollideMode (const std::string & _mode)

Set the collide mode of the body.

Parameters

in	*_mode*	**Collision** (p. 262) Mode, this can be: [all|none|sensors|fixed|ghost] all: collides with everything none: collides with nothing sensors: collides with everything else but other sensors fixed: collides with everything else but other fixed ghost: collides with everything else but other ghost

10.90.3.56 virtual void gazebo::physics::Link::SetEnabled (bool *_enable*) const [pure virtual]

Set whether this body is enabled.

Parameters

in	*_enable*	True to enable the link in the physics engine.

Implemented in **gazebo::physics::BulletLink** (p. 197), and **gazebo::physics::ODELink** (p. 605).

10.90.3.57 virtual void gazebo::physics::Link::SetForce (const math::Vector3 & *_force*) [pure virtual]

Set the force applied to the body.

Parameters

in	*_force*	Force value.

Implemented in **gazebo::physics::ODELink** (p. 605), and **gazebo::physics::BulletLink** (p. 197).

10.90.3.58 virtual void gazebo::physics::Link::SetGravityMode (bool *_mode*) [pure virtual]

Set whether gravity affects this body.

Parameters

in	*_mode*	True to enable gravity.

Implemented in **gazebo::physics::ODELink** (p. 606), and **gazebo::physics::BulletLink** (p. 197).

10.90.3.59 void gazebo::physics::Link::SetInertial (const InertialPtr & *_inertial*)

Set the mass of the link.

[in] _inertial **Inertial** (p. 414) value for the link.

10.90.3.60 virtual void gazebo::physics::Link::SetKinematic (const bool & *_kinematic*) [virtual]

Implement this function.

Set whether this body is in the kinematic state.

Parameters

in	*_kinematic*	True to make the link kinematic only.

Reimplemented in **gazebo::physics::ODELink** (p. 606).

10.90.3.61 void gazebo::physics::Link::SetLaserRetro (float *_retro*)

Set the laser retro reflectiveness.

Parameters

in	*_retro*	Retro value for all child collisions.

10.90.3.62 void gazebo::physics::Link::SetLinearAccel (const math::Vector3 & *_accel*)

Set the linear acceleration of the body.

Parameters

in	*_accel*	Linear acceleration.

10.90.3.63 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::ODELink** (p. 606), and **gazebo::physics::BulletLink** (p. 197).

10.90.3.64 virtual void gazebo::physics::Link::SetLinearVel (const math::Vector3 & *_vel*) [pure virtual]

Set the linear velocity of the body.

Parameters

in	*_vel*	Linear velocity.

Implemented in **gazebo::physics::ODELink** (p. 606), and **gazebo::physics::BulletLink** (p. 197).

10.90.3.65 virtual bool gazebo::physics::Link::SetSelected (bool *_set*) [virtual]

Set whether this entity has been selected by the user through the gui.

Parameters

in	*_set*	True to set the link as selected.

Reimplemented from **gazebo::physics::Base** (p. 156).

10.90.3.66 virtual void gazebo::physics::Link::SetSelfCollide (bool *_collide*) [pure virtual]

Set whether this body will collide with others in the model.

Parameters

in	*_collid*	True to enable collisions.

Implemented in **gazebo::physics::ODELink** (p. 606), and **gazebo::physics::BulletLink** (p. 197).

10.90.3.67 void gazebo::physics::Link::SetState (const LinkState & *_state*)

Set the current link state.

Parameters

in	*_state*	The state to set the link to.

10.90.3.68 virtual void gazebo::physics::Link::SetTorque (const math::Vector3 & *_torque*) [pure virtual]

Set the torque applied to the body.

Parameters

in	*_torque*	Torque value.

Implemented in **gazebo::physics::ODELink** (p. 606), and **gazebo::physics::BulletLink** (p. 197).

10.90.3.69 virtual void gazebo::physics::Link::Update ( ) [virtual]

Update the body.

Reimplemented from **gazebo::physics::Base** (p. 157).

Reimplemented in **gazebo::physics::BulletLink** (p. 197), and **gazebo::physics::ODELink** (p. 606).

10.90.3.70 virtual void gazebo::physics::Link::UpdateMass ( ) [inline], [virtual]

Update the mass matrix.

Reimplemented in **gazebo::physics::ODELink** (p. 607).

10.90.3.71 virtual void gazebo::physics::Link::UpdateParameters (sdf::ElementPtr *_sdf*) [virtual]

Update the parameters using new sdf values.

Parameters

in	*_sdf*	SDF values to load from.

Reimplemented from **gazebo::physics::Entity** (p. 348).

10.90.3.72 virtual void gazebo::physics::Link::UpdateSurface ( ) [inline], [virtual]

Update surface parameters.

Reimplemented in **gazebo::physics::ODELink** (p. 607).

10.90.4 Member Data Documentation

10.90.4.1 math::Vector3 gazebo::physics::Link::angularAccel [protected]

Angular acceleration.

10.90.4.2 std::vector<math::Pose> gazebo::physics::Link::attachedModelsOffset [protected]

Offsets for the attached models.

10.90.4.3 std::vector<std::string> gazebo::physics::Link::cgVisuals [protected]

Center of gravity visual elements.

10.90.4.4 InertialPtr gazebo::physics::Link::inertial [protected]

Inertial (p. 414) properties.

Referenced by GetInertial().

10.90.4.5 math::Vector3 gazebo::physics::Link::linearAccel [protected]

Linear acceleration.

10.90.4.6 std::vector<std::string> gazebo::physics::Link::visuals [protected]

Link (p. 454) visual elements.

The documentation for this class was generated from the following file:

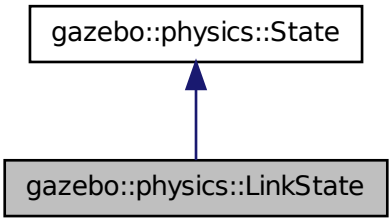
- **Link.hh**

10.91 gazebo::physics::LinkState Class Reference

Store state information of a **physics::Link** (p. 454) object.

`#include <physics/LinkState.hh>`

Inheritance diagram for gazebo::physics::LinkState:



Public Member Functions

- **LinkState** ()

  Default constructor.
- **LinkState** (const **LinkPtr** _link)

  Constructor.
- virtual ~**LinkState** ()

  Destructor.
- void **FillStateSDF** (**sdf::ElementPtr** _elem)

  *Fill a **State** (p. 813) SDF element with state info.*
- **math::Pose GetAcceleration** () const

  Get the link acceleration.
- **CollisionState GetCollisionState** (unsigned int _index) const

  Get a collision state.
- **CollisionState GetCollisionState** (const std::string &_collisionName) const

  Get a link state by link name.
- unsigned int **GetCollisionStateCount** () const

  Get the number of link states.
- std::list< **math::Pose** > **GetForces** () const

  *Get the forces applied to the **Link** (p. 454).*
- **math::Pose GetPose** () const

  Get the link pose.
- **math::Pose GetVelocity** () const

  Get the link velocity.
- virtual void **Load** (**sdf::ElementPtr** _elem)

  Load state from SDF element.
- void **UpdateLinkSDF** (**sdf::ElementPtr** _elem)

  *Update a **Link** (p. 454) SDF element with this state info.*

Additional Inherited Members

10.91.1 Detailed Description

Store state information of a **physics::Link** (p. 454) object.

This class captures the entire state of a **Link** (p. 454) at one specific time during a simulation run.

State (p. 813) of a **Link** (p. 454) includes the state of itself all its child **Collision** (p. 262) entities.

10.91.2 Constructor & Destructor Documentation

10.91.2.1 gazebo::physics::LinkState::LinkState ( )

Default constructor.

10.91.2.2 gazebo::physics::LinkState::LinkState (const LinkPtr *_link*) [explicit]

Constructor.

Build a **LinkState** (p. 473) from an existing **Link** (p. 454).

Parameters

in	*_model*	Pointer to the **Link** (p. 454) from which to gather state info.

10.91.2.3 virtual gazebo::physics::LinkState::~LinkState ( ) [virtual]

Destructor.

10.91.3 Member Function Documentation

10.91.3.1 void gazebo::physics::LinkState::FillStateSDF (sdf::ElementPtr *_elem*)

Fill a **State** (p. 813) SDF element with state info.

Stored state information into an SDF::Element pointer.

Parameters

in	*_elem*	Pointer to the SDF::Element which recieves the data.

10.91.3.2 math::Pose gazebo::physics::LinkState::GetAcceleration ( ) const

Get the link acceleration.

Returns

The acceleration represented as a **math::Pose** (p. 677)

10.91.3.3 CollisionState gazebo::physics::LinkState::GetCollisionState (unsigned int _index) const

Get a collision state.

Get a **Collision** (p. 262) **State** (p. 813) based on an index, where index is in the range of 0...**LinkState::GetCollisionStateCount** (p. 476)

Parameters

in	*_index*	Index of the **CollisionState** (p. 272)

Returns

State (p. 813) of the **Collision** (p. 262)

10.91.3.4 CollisionState gazebo::physics::LinkState::GetCollisionState (const std::string & _collisionName) const

Get a link state by link name.

Searches through all CollisionStates. Returns the **CollisionState** (p. 272) with the matching name, if any.

Parameters

in	*_collisionName*	Name of the **CollisionState** (p. 272)

Returns

State (p. 813) of the **Collision** (p. 262).

10.91.3.5 unsigned int gazebo::physics::LinkState::GetCollisionStateCount ( ) const

Get the number of link states.

This returns the number of Collisions recorded.

Returns

Number of **CollisionState** (p. 272) recorded

10.91.3.6 std::list<math::Pose> gazebo::physics::LinkState::GetForces ( ) const

Get the forces applied to the **Link** (p. 454).

Returns

The list of forces represented as a **math::Pose** (p. 677)

10.91.3.7 math::Pose gazebo::physics::LinkState::GetPose ( ) const

Get the link pose.

Returns

The **math::Pose** (p. 677) of the **Link** (p. 454)

10.91.3.8 math::Pose gazebo::physics::LinkState::GetVelocity ( ) const

Get the link velocity.

Returns

The velocity represented as a **math::Pose** (p. 677)

10.91.3.9 virtual void gazebo::physics::LinkState::Load (sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Load **LinkState** (p. 473) information from stored data in and SDF::Element

Parameters

in	*_elem*	Pointer to the SDF::Element containing state info.

Implements **gazebo::physics::State** (p. 815).

10.91.3.10 void gazebo::physics::LinkState::UpdateLinkSDF (sdf::ElementPtr *_elem*)

Update a **Link** (p. 454) SDF element with this state info.

Set the values in a Links's SDF::Element with the information stored in this instance.

Parameters

in	*_elem*	Pointer to a Links's SDF::Element

The documentation for this class was generated from the following file:

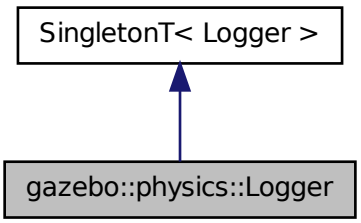
- **LinkState.hh**

10.92 gazebo::physics::Logger Class Reference

Handles logging of data to disk.

`#include <physics/Logger.hh>`

Inheritance diagram for gazebo::physics::Logger:



Public Member Functions

- bool **Add** (const std::string &_entityName, const std::string &_filename)

  Add an entity to a log file.
- bool **Remove** (const std::string &_entity)

  Remove an entity from a log.

Additional Inherited Members

10.92.1 Detailed Description

Handles logging of data to disk.

The **Logger** (p. 477) class is a Singleton that manages data logging of any entity within a running simulation. An entity may be a **World** (p. 954), **Model** (p. 521), or any of their child entities. This class only writes log files, see LogPlay for playback functionality.

State (p. 813) information for an entity may be logged through the **Logger::Add** (p. 478) function, and stopped through the **Logger::Remove** (p. 479) function. Data may be logged into a single file, or split into many separate files by specifying different filenames for the **Logger::Add** (p. 478) function.

The **Logger** (p. 477) is updated at the start of each simulation step. This guarantees that all data is stored.

See Also

Logplay (p. 479), **State** (p. 813)

10.92.2 Member Function Documentation

10.92.2.1 bool gazebo::physics::Logger::Add (const std::string & *_entityName,* const std::string & *_filename*)

Add an entity to a log file.

Add a new entity to a log. An entity can be any valid named object in the world, including the world itself. Duplicate additions are ignored. Entites can be added to the same file by passing specifying the same _filename.

Parameters

_entityName	Name of the entity to log.
_filename	Filename of the log file.

Returns

True if the **Entity** (p. 338) was added to a log. False if the entity is already being logged.

10.92.2.2 bool gazebo::physics::Logger::Remove (const std::string & *_entity*)

Remove an entity from a log.

Removes an entity from the logger. The stops data recording for the entity and all its children. For example, specifying a world will stop all data logging.

Parameters

_entity	Name of the entity to stop logging

Returns

True if the entity existed and was removed. False if the entity was not registered with the logger.

The documentation for this class was generated from the following file:

- **Logger.hh**

10.93 gazebo::physics::Logplay Class Reference

Open and playback log files that were recorded using **Logger** (p. 477).

`#include <LogPlay.hh>`

Public Member Functions

- Iterator **Begin** ()

  Get an iterator to the beginning of the log file.
- Iterator **End** ()

  Get an iterator to the end of the log file.
- bool **Open** (const std::string &_logFile)

  Open a log file for reading.
- bool **Play** (**WorldPtr** _world)

  *Play a log file in a given **World** (p. 954).*
- bool **Play** (**WorldPtr** _world, const std::string &_entityName)

  *Play a log file in a given **World** (p. 954) using only data for a given entity.*
- bool **Play** (**WorldPtr** _world, const std::list< std::string > &_entityNames)

  *Play a log file in a given **World** (p. 954) using only data for a list of entities.*
- bool **Play** (**WorldPtr** _world, **common::Time** _start, **common::Time** _stop)

*Play a segment of a log file in a given **World** (p. 954).*

- bool **Play** (**WorldPtr** _world, const std::string &_entityName, **common::Time** _start, **common::Time** _stop)

  *Play a segment of a log file in a given **World** (p. 954) using only data for a given entity.*

- bool **Play** (**WorldPtr** _world, const std::list< std::string > &_entityNames, **common::Time** _start, **common::Time** _stop)

  *Play a segment of a log file in a given **World** (p. 954) using only data for a list of entities.*

10.93.1 Detailed Description

Open and playback log files that were recorded using **Logger** (p. 477).

Use **Logplay** (p. 479) to open a log file (**Logplay::Open** (p. 481)), and access the recorded state information. Iterators are available to step through the state information. It is also possible to replay the data in a **World** (p. 954) using the Play functions. Replay involves reading and applying state information to a **World** (p. 954).

See Also

Logger (p. 477), **State** (p. 813)

10.93.2 Member Function Documentation

10.93.2.1 Iterator gazebo::physics::Logplay::Begin ( )

Get an iterator to the beginning of the log file.

Requires a log file to be opened first. Use the iterator to step through a log file.

See Also

Logplay::Open (p. 481).

Returns

Iterator the beginning of the opened log file

10.93.2.2 Iterator gazebo::physics::Logplay::End ( )

Get an iterator to the end of the log file.

Requires a log file to be opened first Use the iterator to step through a log file.

See Also

Logplay::Open (p. 481).

Returns

Iterator the end of the opened log file

10.93.2.3 bool gazebo::physics::Logplay::Open (const std::string & _logFile)

Open a log file for reading.

Open a log file that was previously recorded.

Parameters

_logFile	The file to load

Returns

True if the log file was successfully loaded

10.93.2.4 bool gazebo::physics::Logplay::Play (WorldPtr _world)

Play a log file in a given **World** (p. 954).

Replay a complete log file in a **World** (p. 954). Requires a log file to be opened first,

See Also

Logplay::Open (p. 481).

Parameters

_world	Pointer to the **World** (p. 954)

Returns

True if replay was successful

10.93.2.5 bool gazebo::physics::Logplay::Play (WorldPtr _world, const std::string & _entityName)

Play a log file in a given **World** (p. 954) using only data for a given entity.

Replay a log file in a **World** (p. 954), but only apply data for the given entity. Requires a log file to be opened first

See Also

Logplay::Open (p. 481).

Parameters

_world	Pointer to the **World** (p. 954)
_entityName	Name of the entity to search the log for

Returns

True if replay was successful

10.93.2.6 bool gazebo::physics::Logplay::Play (WorldPtr *_world,* const std::list< std::string > & *_entityNames*)

Play a log file in a given **World** (p. 954) using only data for a list of entities.

Replay a log file in a **World** (p. 954), but only apply data for the given list of entities. Requires a log file to be opened first,

See Also

Logplay::Open (p. 481).

Parameters

_world	Pointer to the **World** (p. 954)
_entityNames	Names of the entities to search the log for

Returns

True if replay was successful

10.93.2.7 bool gazebo::physics::Logplay::Play (WorldPtr *_world,* common::Time *_start,* common::Time *_stop*)

Play a segment of a log file in a given **World** (p. 954).

Replay a segment of log file in a **World** (p. 954). The segment is defined by a start and end time, where time is simulation time. Requires a log file to be opened first,

See Also

Logplay::Open (p. 481).

Parameters

_world	Pointer to the **World** (p. 954).
_start	Start time, in simulation time.
_stop	Stop time, in simulation time.

Returns

True if replay was successful.

10.93.2.8 bool gazebo::physics::Logplay::Play (WorldPtr *_world,* const std::string & *_entityName,* common::Time *_start,* common::Time *_stop*)

Play a segment of a log file in a given **World** (p. 954) using only data for a given entity.

Replay a segment of log file in a **World** (p. 954), but only apply data for the given entity. The segment is defined by a start and end time, where time is simulation time. Requires a log file to be opened first,

See Also

Logplay::Open (p. 481).

Parameters

_world	Pointer to the **World** (p. 954).
_entityName	Name of the entity to search the log for
_start	Start time, in simulation time.
_stop	Stop time, in simulation time.

Returns

True if replay was successful.

10.93.2.9 bool gazebo::physics::Logplay::Play (WorldPtr *_world,* const std::list< std::string > & *_entityNames,* common::Time *_start,* common::Time *_stop*)

Play a segment of a log file in a given **World** (p. 954) using only data for a list of entities.

Replay a segment of a log file in a **World** (p. 954), but only apply data for the given list of entities. Requires a log file to be opened first,

See Also

Logplay::Open (p. 481).

Parameters

_world	Pointer to the **World** (p. 954)
_entityNames	Names of the entities to search the log for
_start	Start time, in simulation time.
_stop	Stop time, in simulation time.

Returns

True if replay was successful

The documentation for this class was generated from the following file:

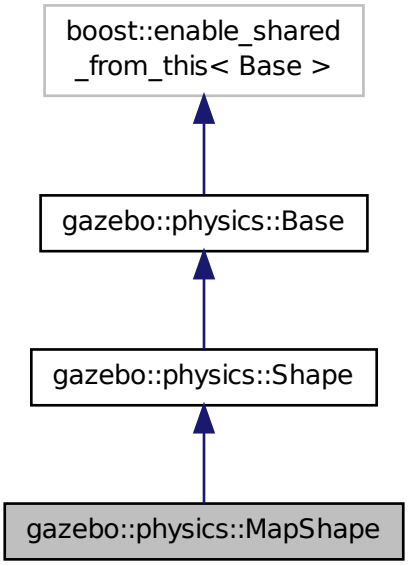
- **LogPlay.hh**

10.94 gazebo::physics::MapShape Class Reference

Creates box extrusions based on an image.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::MapShape:



Public Member Functions

- **MapShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**MapShape** ()

  Destructor.
- void **FillMsg** (msgs::Geometry &_msg)

  Fills out a msgs::Geometry message containing information about this map geometry object.
- int **GetGranularity** () const

  Returns granularity of this geometry.
- double **GetHeight** () const

  Returns height of this geometry.
- double **GetScale** () const

  Returns scaling factor for this geometry.
- int **GetThreshold** () const

  Returns image threshold for this geometry.
- std::string **GetURI** () const

  Returns the image URI for this geometry.
- virtual void **Init** ()

  Init the map.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the map.

- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  : Implement this function.
- void **Update** ()

  Update function.

Additional Inherited Members

10.94.1 Detailed Description

Creates box extrusions based on an image.

This function is not yet complete, to be implemented.

10.94.2 Constructor & Destructor Documentation

10.94.2.1 gazebo::physics::MapShape::MapShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent collision object.

10.94.2.2 virtual gazebo::physics::MapShape::~MapShape ( ) [virtual]

Destructor.

10.94.3 Member Function Documentation

10.94.3.1 void gazebo::physics::MapShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Fills out a msgs::Geometry message containing information about this map geometry object.

Parameters

in	*_msg*	Message to fill with this object's data.

Implements **gazebo::physics::Shape** (p. 781).

10.94.3.2 int gazebo::physics::MapShape::GetGranularity ( ) const

Returns granularity of this geometry.

Returns

Granularity (amount of error betweent the image pixels and the 3D shapes created).

10.94.3.3 double gazebo::physics::MapShape::GetHeight ( ) const

Returns height of this geometry.

All regions in image with value larger than MapShape::scale will be replaced by boxes with MapShape::height.

Returns

Height of the map shapes.

10.94.3.4 double gazebo::physics::MapShape::GetScale ( ) const

Returns scaling factor for this geometry.

Returns

Scaling factor.

10.94.3.5 int gazebo::physics::MapShape::GetThreshold ( ) const

Returns image threshold for this geometry.

All regions in image with value larger than MapShape::scale will be replaced by boxes with MapShape::height.

Returns

Image threshold value.

10.94.3.6 std::string gazebo::physics::MapShape::GetURI ( ) const

Returns the image URI for this geometry.

Returns

The image URI that was used to load the map.

10.94.3.7 virtual void gazebo::physics::MapShape::Init ( ) [virtual]

Init the map.

Implements **gazebo::physics::Shape** (p. 781).

10.94.3.8 virtual void gazebo::physics::MapShape::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the map.

Parameters

in	*_sdf*	Load the map from SDF values.

Reimplemented from **gazebo::physics::Base** (p. 154).

10.94.3.9 virtual void gazebo::physics::MapShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

: Implement this function.

Parameters

in	*_msg*	Message to process, which will alter the map.

Implements **gazebo::physics::Shape** (p. 782).

10.94.3.10 void gazebo::physics::MapShape::Update ( ) [virtual]

Update function.

Reimplemented from **gazebo::physics::Base** (p. 157).

The documentation for this class was generated from the following file:

- **MapShape.hh**

10.95 gazebo::Master Class Reference

A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

`#include <gazebo_core.hh>`

Public Member Functions

- **Master** ()

  Constructor.
- virtual ~**Master** ()

  Destructor.
- void **Fini** ()

  Finalize the master.
- void **Init** (uint16_t _port)

  Initialize.
- void **Run** ()

  Run the master.
- void **RunOnce** ()

  Run the master one iteration.
- void **RunThread** ()

  Run the master in a new thread.
- void **Stop** ()

  Stop the master.

10.95.1 Detailed Description

A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

Base class for simulation server that handles commandline options, starts a **Master** (p. 487), runs World update and sensor generation loops.

10.95.2 Constructor & Destructor Documentation

10.95.2.1 gazebo::Master::Master ( )

Constructor.

10.95.2.2 virtual gazebo::Master::~Master ( ) [virtual]

Destructor.

10.95.3 Member Function Documentation

10.95.3.1 void gazebo::Master::Fini ( )

Finalize the master.

10.95.3.2 void gazebo::Master::Init (uint16_t *_port*)

Initialize.

Parameters

in	*_port*	The master's port

10.95.3.3 void gazebo::Master::Run ( )

Run the master.

10.95.3.4 void gazebo::Master::RunOnce ( )

Run the master one iteration.

10.95.3.5 void gazebo::Master::RunThread ( )

Run the master in a new thread.

10.95.3.6 void gazebo::Master::Stop ( )

Stop the master.

The documentation for this class was generated from the following file:

- **Master.hh**

10.96 gazebo::common::Material Class Reference

Encapsulates description of a material.

`#include <Material.hh>`

Public Types

- enum **BlendMode** { **ADD**, **MODULATE**, **REPLACE**, **BLEND_COUNT** }
- enum **ShadeMode** {
  FLAT, **GOURAUD**, **PHONG**, **BLINN**,
  SHADE_COUNT }

Public Member Functions

- **Material** ()

  Constructor.
- **Material** (const **Color** &_clr)

  Create a material with a default color.
- virtual ~**Material** ()

  Destructor.
- **Color GetAmbient** () const

  Get the ambient color.
- void **GetBlendFactors** (double &_srcFactor, double &_dstFactor)

  Get the blend factors.
- **BlendMode GetBlendMode** () const

  Get the blending mode.
- bool **GetDepthWrite** () const

  Get depth write.
- **Color GetDiffuse** () const

  Get the diffuse color.
- **Color GetEmissive** () const

  Get the emissive color.
- bool **GetLighting** () const

  Get lighting enabled.
- std::string **GetName** () const

  Get the name of the material.
- double **GetPointSize** () const

  Get the point size.
- **ShadeMode GetShadeMode** () const

  Get the shading mode.
- double **GetShininess** () const

  Get the shininess.

- **Color GetSpecular** () const

  Get the specular color.
- std::string **GetTextureImage** () const

  Get a texture image.
- double **GetTransparency** () const

  Get the transparency percentage (0..1)
- void **SetAmbient** (const **Color** &_clr)

  Set the ambient color.
- void **SetBlendFactors** (double _srcFactor, double _dstFactor)

  Set the blende factors.
- void **SetBlendMode** (**BlendMode** _b)

  Set the blending mode.
- void **SetDepthWrite** (bool _value)

  Set depth write.
- void **SetDiffuse** (const **Color** &_clr)

  Set the diffuse color.
- void **SetEmissive** (const **Color** &_clr)

  Set the emissive color.
- void **SetLighting** (bool _value)

  Set lighting enabled.
- void **SetPointSize** (double _size)

  Set the point size.
- void **SetShadeMode** (**ShadeMode** _b)

  Set the shading mode param[in] the shading mode.
- void **SetShininess** (double _t)

  Set the shininess.
- void **SetSpecular** (const **Color** &_clr)

  Set the specular color.
- void **SetTextureImage** (const std::string &_tex)

  Set a texture image.
- void **SetTextureImage** (const std::string &_tex, const std::string &_resourcePath)

  Set a texture image.
- void **SetTransparency** (double _t)

  Set the transparency percentage (0..1)

Static Public Attributes

- static std::string **BlendModeStr** [**BLEND_COUNT**]
- static std::string **ShadeModeStr** [**SHADE_COUNT**]

Protected Attributes

- **Color ambient**

  the ambient light color
- **BlendMode blendMode**

  blend mode
- **Color diffuse**

the diffuse 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.

10.96.1 Detailed Description

Encapsulates description of a material.

10.96.2 Member Enumeration Documentation

10.96.2.1 enum gazebo::common::Material::BlendMode

Enumerator:

ADD

MODULATE

REPLACE

BLEND_COUNT

10.96.2.2 enum gazebo::common::Material::ShadeMode

Enumerator:

FLAT

GOURAUD

PHONG

BLINN

SHADE_COUNT

10.96.3 Constructor & Destructor Documentation

10.96.3.1 gazebo::common::Material::Material ( )

Constructor.

10.96.3.2 virtual gazebo::common::Material::~Material ( ) [virtual]

Destructor.

10.96.3.3 gazebo::common::Material::Material (const Color & _clr)

Create a material with a default color.

Parameters

in	*_clr*	**Color** (p. 274) of the material

10.96.4 Member Function Documentation

10.96.4.1 Color gazebo::common::Material::GetAmbient ( ) const

Get the ambient color.

Returns

The ambient color

10.96.4.2 void gazebo::common::Material::GetBlendFactors (double & _srcFactor, double & _dstFactor)

Get the blend factors.

Parameters

in	*_srcFactor*	Source factor is returned in this variable
in	*_dstFactor*	Destination factor is returned in this variable

10.96.4.3 BlendMode gazebo::common::Material::GetBlendMode ( ) const

Get the blending mode.

Returns

the blend mode

10.96.4.4 bool gazebo::common::Material::GetDepthWrite ( ) const

Get depth write.

Returns

the depth write enabled state

10.96.4.5 Color gazebo::common::Material::GetDiffuse ( ) const

Get the diffuse color.

Returns

The diffuse color

10.96.4.6 Color gazebo::common::Material::GetEmissive ( ) const

Get the emissive color.

Returns

The emissive color

10.96.4.7 bool gazebo::common::Material::GetLighting ( ) const

Get lighting enabled.

Returns

the lighting enabled state

10.96.4.8 std::string gazebo::common::Material::GetName ( ) const

Get the name of the material.

Returns

The name of the material

10.96.4.9 double gazebo::common::Material::GetPointSize ( ) const

Get the point size.

Returns

the point size

10.96.4.10 ShadeMode gazebo::common::Material::GetShadeMode ( ) const

Get the shading mode.

Returns

the shading mode

10.96.4.11 double gazebo::common::Material::GetShininess ( ) const

Get the shininess.

Returns

The shininess value

10.96.4.12 Color gazebo::common::Material::GetSpecular ( ) const

Get the specular color.

Returns

The specular color

10.96.4.13 std::string gazebo::common::Material::GetTextureImage ( ) const

Get a texture image.

Returns

The name of the texture image (if one exists) or an empty string

10.96.4.14 double gazebo::common::Material::GetTransparency ( ) const

Get the transparency percentage (0..1)

Returns

The transparency percentage

10.96.4.15 void gazebo::common::Material::SetAmbient (const Color & *_clr*)

Set the ambient color.

Parameters

in	*_clr*	The ambient color

10.96.4.16 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

10.96.4.17 void gazebo::common::Material::SetBlendMode (BlendMode _b)

Set the blending mode.

Parameters

in	*_b*	the blend mode

10.96.4.18 void gazebo::common::Material::SetDepthWrite (bool _value)

Set depth write.

Parameters

in	*_value*	the depth write enabled state

10.96.4.19 void gazebo::common::Material::SetDiffuse (const Color & _clr)

Set the diffuse color.

Parameters

in	*_clr*	The diffuse color

10.96.4.20 void gazebo::common::Material::SetEmissive (const Color & _clr)

Set the emissive color.

Parameters

in	*_clr*	The emissive color

10.96.4.21 void gazebo::common::Material::SetLighting (bool _value)

Set lighting enabled.

Parameters

in	*_value*	the lighting enabled state

10.96.4.22 void gazebo::common::Material::SetPointSize (double _size)

Set the point size.

Parameters

in	*_size*	the size

10.96.4.23 void gazebo::common::Material::SetShadeMode (ShadeMode _b)

Set the shading mode param[in] the shading mode.

10.96.4.24 void gazebo::common::Material::SetShininess (double _t)

Set the shininess.

Parameters

in	*_t*	The shininess value

10.96.4.25 void gazebo::common::Material::SetSpecular (const Color & _clr)

Set the specular color.

Parameters

in	*_clr*	The specular color

10.96.4.26 void gazebo::common::Material::SetTextureImage (const std::string & _tex)

Set a texture image.

Parameters

i]	_tex The name of the texture, which must be in Gazebo's resource path

10.96.4.27 void gazebo::common::Material::SetTextureImage (const std::string & _tex, const std::string & _resourcePath)

Set a texture image.

Parameters

i]	_tex The name of the texture
_resourcePath]	_resourcePath Path which contains _tex

10.96.4.28 void gazebo::common::Material::SetTransparency (double _t)

Set the transparency percentage (0..1)

Parameters

in	*_t*	The amount of transparency (0..1)

10.96.5 Friends And Related Function Documentation

10.96.5.1 std::ostream& operator<< (std::ostream & _out, const gazebo::common::Material & _m) [friend]

Stream insertion operator param[in] _out the output stream to extract from param[out] _m the material information.

10.96.6 Member Data Documentation

10.96.6.1 Color gazebo::common::Material::ambient [protected]

the ambient light color

10.96.6.2 BlendMode gazebo::common::Material::blendMode [protected]

blend mode

10.96.6.3 std::string gazebo::common::Material::BlendModeStr[BLEND_COUNT] [static]

10.96.6.4 Color gazebo::common::Material::diffuse [protected]

the diffuse ligth color

10.96.6.5 Color gazebo::common::Material::emissive [protected]

the emissive light color

10.96.6.6 std::string gazebo::common::Material::name [protected]

the name of the material

10.96.6.7 double gazebo::common::Material::pointSize [protected]

point size

10.96.6.8 ShadeMode gazebo::common::Material::shadeMode [protected]

the shade mode

10.96.6.9 std::string gazebo::common::Material::ShadeModeStr[SHADE_COUNT] [static]

10.96.6.10 double gazebo::common::Material::shininess [protected]

shininess value (0 to 1)

10.96.6.11 Color gazebo::common::Material::specular [protected]

the specular light color

10.96.6.12 std::string gazebo::common::Material::texImage [protected]

the texture image file name

10.96.6.13 double gazebo::common::Material::transparency [protected]

transparency value in the range 0 to 1

The documentation for this class was generated from the following file:

- **common/Material.hh**

10.97 gazebo::math::Matrix3 Class Reference

A 3x3 matrix class.

`#include <Matrix3.hh>`

Public Member Functions

- **Matrix3** ()

  Constructor.
- **Matrix3** (const **Matrix3** &_m)

  Copy constructor.
- **Matrix3** (double _v00, double _v01, double _v02, double _v10, double _v11, double _v12, double _v20, double _v21, double _v22)

  Constructor.
- virtual ~**Matrix3** ()

  Desctructor.
- bool **operator==** (const **Matrix3** &_m) const

  Equality test operatorr.
- const double * **operator[]** (size_t _row) const

  Array subscript operator.
- double * **operator[]** (size_t _row)

  Array subscript operator.
- void **SetCol** (unsigned int _c, const **Vector3** &_v)

  Set a column.
- void **SetFromAxes** (const **Vector3** &_xAxis, const **Vector3** &_yAxis, const **Vector3** &_zAxis)

  Set the matrix from three axis (1 per column)
- void **SetFromAxis** (const **Vector3** &_axis, double _angle)

  Set the matrix from an axis and angle.

Protected Attributes

- double **m** [3][3]

  the 3x3 matrix

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Matrix3** &_m)
  Stream insertion operator.

10.97.1 Detailed Description

A 3x3 matrix class.

10.97.2 Constructor & Destructor Documentation

10.97.2.1 gazebo::math::Matrix3::Matrix3 ( )

Constructor.

10.97.2.2 gazebo::math::Matrix3::Matrix3 (const Matrix3 & *_m*)

Copy constructor.

Parameters

_m	Matrix to copy

10.97.2.3 gazebo::math::Matrix3::Matrix3 (double *_v00,* double *_v01,* double *_v02,* double *_v10,* double *_v11,* double *_v12,* double *_v20,* double *_v21,* double *_v22*)

Constructor.

Parameters

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

10.97.2.4 virtual gazebo::math::Matrix3::~Matrix3 ( ) [virtual]

Desctructor.

10.97.3 Member Function Documentation

10.97.3.1 bool gazebo::math::Matrix3::operator== (const Matrix3 & _m) const

Equality test operatorr.

Parameters

in	*_m*	**Matrix3** (p. 498) to test

Returns

True if equal (using the default tolerance of 1e-6)

10.97.3.2 const double* gazebo::math::Matrix3::operator[] (size_t _row) const [inline]

Array subscript operator.

Parameters

in	*_row*	row index

Returns

a pointer to the row

References m.

10.97.3.3 double* gazebo::math::Matrix3::operator[] (size_t _row) [inline]

Array subscript operator.

Parameters

in	*_row*	row index

Returns

a pointer to the row

References m.

10.97.3.4 void gazebo::math::Matrix3::SetCol (unsigned int _c, const Vector3 & _v)

Set a column.

Parameters

in	*_c*	The colum index (0, 1, 2)
in	*_v*	The value to set in each row of the column

10.97.3.5 void gazebo::math::Matrix3::SetFromAxes (const Vector3 & _xAxis, const Vector3 & _yAxis, const Vector3 & _zAxis)

Set the matrix from three axis (1 per column)

Parameters

in	*_xAxis*	The x axis
in	*_yAxis*	The y axis
in	*_zAxis*	The z axis

10.97.3.6 void gazebo::math::Matrix3::SetFromAxis (const Vector3 & _axis, double _angle)

Set the matrix from an axis and angle.

Parameters

in	*_axis*	the axis
in	*_angle*	ccw rotation around the axis in radians

10.97.4 Friends And Related Function Documentation

10.97.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Matrix3 & _m) [friend]

Stream insertion operator.

Parameters

in	*_out*	Output stream
in	*_m*	Matrix to output

Returns

the stream

10.97.5 Member Data Documentation

10.97.5.1 double gazebo::math::Matrix3::m[3][3] [protected]

the 3x3 matrix

Referenced by operator[]().

The documentation for this class was generated from the following file:

- **Matrix3.hh**

10.98 gazebo::math::Matrix4 Class Reference

A 3x3 matrix class.

`#include <Matrix4.hh>`

Public Member Functions

- **Matrix4** ()

  Constructor.
- **Matrix4** (const **Matrix4** &_m)

  Copy constructor.
- **Matrix4** (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

  Constructor.
- virtual ∼**Matrix4** ()

  Destructor.
- **math::Pose GetAsPose** () const

  *Get the transformation as **math::Pose** (*p. *677)*.
- **Vector3 GetEulerRotation** (unsigned int solution_number=1) const

  Get the rotation as a Euler angles.
- **Quaternion GetRotation** () const

  Get the rotation as a quaternion.
- **Vector3 GetTranslation** () const

  *Get the translational values as a **Vector3** (*p. *902)*.
- **Matrix4 Inverse** () const

  Return the inverse matrix.
- bool **IsAffine** () const

  Return true if the matrix is affine.
- **Matrix4 operator**∗ (const **Matrix4** &_mat) const

  Multiplication operator.
- **Matrix4 operator**∗ (const **Matrix3** &_mat) const

  Multiplication operator.
- **Vector3 operator**∗ (const **Vector3** &_vec) const

  Multiplication operator.
- **Matrix4** & **operator=** (const **Matrix4** &_mat)

  Equal operator.
- const **Matrix4** & **operator=** (const **Matrix3** &_mat)

  Equal operator for 3x3 matrix.
- bool **operator==** (const **Matrix4** &_m) const

  Equality operator.
- double ∗ **operator[]** (size_t _row)

  Array subscript operator.
- const double ∗ **operator[]** (size_t _row) const
- void **Set** (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

  Change the values.
- void **SetScale** (const **Vector3** &_s)

  Set the scale.
- void **SetTranslate** (const **Vector3** &_t)

  Set the translational values [(0, 3) (1, 3) (2, 3)].
- **Vector3 TransformAffine** (const **Vector3** &_v) const

  Perform an affine transformation.

Static Public Attributes

- static const **Matrix4 IDENTITY**

  Identity matrix.
- static const **Matrix4 ZERO**

  Zero matrix.

Protected Attributes

- double **m** [4][4]

  The 4x4 matrix.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Matrix4** &_m)

  Stream insertion operator.

10.98.1 Detailed Description

A 3x3 matrix class.

10.98.2 Constructor & Destructor Documentation

10.98.2.1 gazebo::math::Matrix4::Matrix4 ( )

Constructor.

10.98.2.2 gazebo::math::Matrix4::Matrix4 (const Matrix4 & *_m*)

Copy constructor.

Parameters

_m	Matrix to copy

10.98.2.3 gazebo::math::Matrix4::Matrix4 (double *_v00*, double *_v01*, double *_v02*, double *_v03*, double *_v10*, double *_v11*, double *_v12*, double *_v13*, double *_v20*, double *_v21*, double *_v22*, double *_v23*, double *_v30*, double *_v31*, double *_v32*, double *_v33*)

Constructor.

Parameters

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

10.98.2.4 virtual gazebo::math::Matrix4::~Matrix4 ( ) [virtual]

Destructor.

10.98.3 Member Function Documentation

10.98.3.1 math::Pose gazebo::math::Matrix4::GetAsPose ( ) const

Get the transformation as **math::Pose** (p. 677).

Returns

the pose

10.98.3.2 Vector3 gazebo::math::Matrix4::GetEulerRotation (unsigned int *solution_number* = 1) const

Get the rotation as a Euler angles.

Returns

the rotation

10.98.3.3 Quaternion gazebo::math::Matrix4::GetRotation ( ) const

Get the rotation as a quaternion.

Returns

the rotation

10.98.3.4 Vector3 gazebo::math::Matrix4::GetTranslation ( ) const

Get the translational values as a **Vector3** (p. 902).

Returns

x,y,z

10.98.3.5 Matrix4 gazebo::math::Matrix4::Inverse ( ) const

Return the inverse matrix.

10.98.3.6 bool gazebo::math::Matrix4::IsAffine ( ) const

Return true if the matrix is affine.

Returns

true if the matrix is affine, false otherwise

10.98.3.7 Matrix4 gazebo::math::Matrix4::operator* (const Matrix4 & *_mat*) const

Multiplication operator.

Parameters

_mat	Incoming matrix

Returns

This matrix * _mat

10.98.3.8 Matrix4 gazebo::math::Matrix4::operator* (const Matrix3 & *_mat*) const

Multiplication operator.

Parameters

_mat	Incoming matrix

Returns

This matrix * _mat

10.98.3.9 Vector3 gazebo::math::Matrix4::operator* (const Vector3 & *_vec*) const

Multiplication operator.

Parameters

_vec	**Vector3** (p. 902)

Returns

Resulting vector from multiplication

10.98.3.10 Matrix4& gazebo::math::Matrix4::operator= (const Matrix4 & _mat)

Equal operator.

this = _mat

Parameters

_mat	Incoming matrix

Returns

itself

10.98.3.11 const Matrix4& gazebo::math::Matrix4::operator= (const Matrix3 & _mat)

Equal operator for 3x3 matrix.

Parameters

_mat	Incoming matrix

Returns

itself

10.98.3.12 bool gazebo::math::Matrix4::operator== (const Matrix4 & _m) const

Equality operator.

Parameters

in	*_m*	**Matrix3** (p. 498) to test

Returns

true if the 2 matrices are equal (using the tolerance 1e-6), false otherwise

10.98.3.13 double* gazebo::math::Matrix4::operator[] (size_t _row) [inline]

Array subscript operator.

Parameters

in	*_row*	the row index

Returns

the row

References m.

10.98.3.14 const double* gazebo::math::Matrix4::operator[] (size_t _row) const [inline]

Parameters

in	*_row*	the row index

Returns

the row

References m.

10.98.3.15 void gazebo::math::Matrix4::Set (double _v00, double _v01, double _v02, double _v03, double _v10, double _v11, double _v12, double _v13, double _v20, double _v21, double _v22, double _v23, double _v30, double _v31, double _v32, double _v33)

Change the values.

Parameters

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

10.98.3.16 void gazebo::math::Matrix4::SetScale (const Vector3 & _s)

Set the scale.

Parameters

in	*_s*	scale

10.98.3.17 void gazebo::math::Matrix4::SetTranslate (const Vector3 & _t)

Set the translational values [(0, 3) (1, 3) (2, 3)].

Parameters

in	*_t*	Values to set

10.98.3.18 Vector3 gazebo::math::Matrix4::TransformAffine (const Vector3 & _v) const

Perform an affine transformation.

Parameters

_v	**Vector3** (p. 902) value for the transformation

Returns

The result of the transformation

10.98.4 Friends And Related Function Documentation

10.98.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Matrix4 & _m) [friend]

Stream insertion operator.

Parameters

_out	output stream
_m	Matrix to output

Returns

the stream

10.98.5 Member Data Documentation

10.98.5.1 const Matrix4 gazebo::math::Matrix4::IDENTITY [static]

Identity matrix.

10.98.5.2 double gazebo::math::Matrix4::m[4][4] [protected]

The 4x4 matrix.

Referenced by operator[]().

10.98.5.3 const Matrix4 gazebo::math::Matrix4::ZERO [static]

Zero matrix.

The documentation for this class was generated from the following file:

- **Matrix4.hh**

10.99 gazebo::common::Mesh Class Reference

A 3D mesh.

`#include <Mesh.hh>`

Public Member Functions

- **Mesh** ()

  Constructor.
- virtual ~**Mesh** ()

  Destructor.
- int **AddMaterial** (**Material** *_mat)

  Add a material to the mesh.
- void **AddSubMesh** (**SubMesh** *_child)

  Add a submesh mesh.
- void **FillArrays** (float **_vertArr, int **_indArr) const

  Put all the data into flat arrays.
- void **GenSphericalTexCoord** (const **math::Vector3** &_center)

  Generate texture coordinates using spherical projection from center.
- void **GetAABB** (**math::Vector3** &_center, **math::Vector3** &_min_xyz, **math::Vector3** &_max_xyz) const

  Get AABB coordinate.
- unsigned int **GetIndexCount** () const

  Return the number of indices.
- const **Material** * **GetMaterial** (int _index) const

  Get a material.
- unsigned int **GetMaterialCount** () const

  Get the number of materials.
- **math::Vector3 GetMax** () const

  Get the 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.
- unsigned int **GetSubMeshCount** () const

  Get the number of children.
- unsigned int **GetTexCoordCount** () const

  Return the number of texture coordinates.
- unsigned int **GetVertexCount** () const

Return the number of vertices.

- bool **HasSkeleton** () const

  Return true if mesh is attached to a skeleton.

- void **RecalculateNormals** ()

  Recalculate all the normals of each face defined by three indices.

- void **Scale** (double _factor)

  Scale all vertices by _factor.

- void **SetName** (const std::string &_n)

  Set the name of this mesh.

- void **SetPath** (const std::string &_path)

  Set the path which contains the mesh resource.

- void **SetSkeleton** (**Skeleton** ∗_skel)

  Set the mesh skeleton.

10.99.1 Detailed Description

A 3D mesh.

10.99.2 Constructor & Destructor Documentation

10.99.2.1 gazebo::common::Mesh::Mesh ( )

Constructor.

10.99.2.2 virtual gazebo::common::Mesh::∼Mesh ( ) [virtual]

Destructor.

10.99.3 Member Function Documentation

10.99.3.1 int gazebo::common::Mesh::AddMaterial (**Material** ∗ *_mat*)

Add a material to the mesh.

Parameters

in	*_mat*	the material

Returns

Index of this material

10.99.3.2 void gazebo::common::Mesh::AddSubMesh (**SubMesh** ∗ *_child*)

Add a submesh mesh.

The **Mesh** (p. 509) object takes ownership of the submesh.

Parameters

in	*_child*	the submesh

10.99.3.3 void gazebo::common::Mesh::FillArrays (float ** *_vertArr*, int ** *_indArr*) const

Put all the data into flat arrays.

Parameters

out	*_vertArr*	the vertex array
out	*_indArr*	the index array

10.99.3.4 void gazebo::common::Mesh::GenSphericalTexCoord (const math::Vector3 & *_center*)

Generate texture coordinates using spherical projection from center.

Parameters

in	*_center*	the center of the projection

10.99.3.5 void gazebo::common::Mesh::GetAABB (math::Vector3 & *_center*, math::Vector3 & *_min_xyz*, math::Vector3 & *_max_xyz*) const

Get AABB coordinate.

Parameters

out	*_center*	of the bounding box
out	*_min_xyz*	bounding box minimum values
out	*_max_xyz*	bounding box maximum values

10.99.3.6 unsigned int gazebo::common::Mesh::GetIndexCount ( ) const

Return the number of indices.

Returns

the count

10.99.3.7 const Material* gazebo::common::Mesh::GetMaterial (int *_index*) const

Get a material.

Parameters

in	*_index*	the index

Returns

the material or NULL if the index is out of bounds

10.99.3.8 unsigned int gazebo::common::Mesh::GetMaterialCount ( ) const

Get the number of materials.

Returns

the count

10.99.3.9 math::Vector3 gazebo::common::Mesh::GetMax ( ) const

Get the maximun X, Y, Z values.

Returns

the upper bounds of the bounding box

10.99.3.10 math::Vector3 gazebo::common::Mesh::GetMin ( ) const

Get the minimum X, Y, Z values.

Returns

the lower bounds of the bounding box

10.99.3.11 std::string gazebo::common::Mesh::GetName ( ) const

Get the name of this mesh.

Returns

the name

10.99.3.12 unsigned int gazebo::common::Mesh::GetNormalCount ( ) const

Return the number of normals.

Returns

the count

10.99.3.13 std::string gazebo::common::Mesh::GetPath ( ) const

Get the path which contains the mesh resource.

Returns

the path to the mesh resource

10.99.3.14 Skeleton* gazebo::common::Mesh::GetSkeleton ( ) const

Get the skeleton to which this mesh is attached.

Returns

pointer to skeleton, or NULL if none is present.

10.99.3.15 const SubMesh* gazebo::common::Mesh::GetSubMesh (unsigned int _i) const

Get a child mesh.

Parameters

in	_i	the index

Returns

the submesh. An exception is thrown if the index is out of bounds

10.99.3.16 unsigned int gazebo::common::Mesh::GetSubMeshCount ( ) const

Get the number of children.

Returns

the count

10.99.3.17 unsigned int gazebo::common::Mesh::GetTexCoordCount ( ) const

Return the number of texture coordinates.

Returns

the count

10.99.3.18 unsigned int gazebo::common::Mesh::GetVertexCount ( ) const

Return the number of vertices.

Returns

the count

10.99.3.19 bool gazebo::common::Mesh::HasSkeleton ( ) const

Return true if mesh is attached to a skeleton.

10.99.3.20 void gazebo::common::Mesh::RecalculateNormals ( )

Recalculate all the normals of each face defined by three indices.

10.99.3.21 void gazebo::common::Mesh::Scale (double *_factor*)

Scale all vertices by _factor.

Parameters

_factor	Scaling factor

10.99.3.22 void gazebo::common::Mesh::SetName (const std::string & *_n*)

Set the name of this mesh.

Parameters

in	*_n*	the name to set

10.99.3.23 void gazebo::common::Mesh::SetPath (const std::string & *_path*)

Set the path which contains the mesh resource.

Parameters

in	*_path*	the file path

10.99.3.24 void gazebo::common::Mesh::SetSkeleton (Skeleton * *_skel*)

Set the mesh skeleton.

The documentation for this class was generated from the following file:

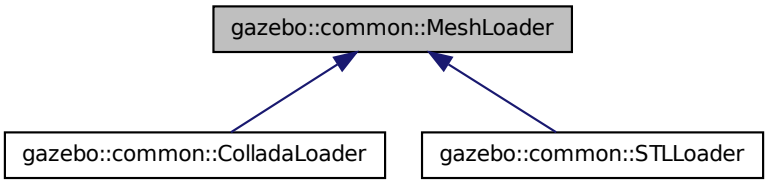
- **Mesh.hh**

10.100 gazebo::common::MeshLoader Class Reference

Base class for loading meshes.

`#include <MeshLoader.hh>`

Inheritance diagram for gazebo::common::MeshLoader:



Public Member Functions

- **MeshLoader** ()

  Constructor.
- virtual ∼**MeshLoader** ()

  Destructor.
- virtual **Mesh** ∗ **Load** (const std::string &filename)=0

  Load a 3D mesh.

10.100.1 Detailed Description

Base class for loading meshes.

10.100.2 Constructor & Destructor Documentation

10.100.2.1 gazebo::common::MeshLoader::MeshLoader ( )

Constructor.

10.100.2.2 virtual gazebo::common::MeshLoader::∼MeshLoader ( ) [virtual]

Destructor.

10.100.3 Member Function Documentation

10.100.3.1 virtual Mesh∗ gazebo::common::MeshLoader::Load (const std::string & *filename*) [pure virtual]

Load a 3D mesh.

Parameters

in	*the*	path to the mesh

Returns

a pointer to the created mesh

Implemented in **gazebo::common::ColladaLoader** (p. 262), and **gazebo::common::STLLoader** (p. 817).

The documentation for this class was generated from the following file:

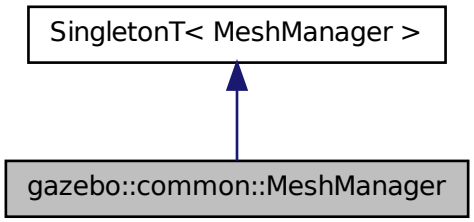
- **MeshLoader.hh**

10.101 gazebo::common::MeshManager Class Reference

Maintains and manages all meshes.

`#include <MeshManager.hh>`

Inheritance diagram for gazebo::common::MeshManager:



Public Member Functions

- void **AddMesh** (**Mesh** *_mesh)

  Add a mesh to the manager.
- void **CreateBox** (const std::string &_name, const **math::Vector3** &_sides, const **math::Vector2d** &_uvCoords)

  Create a Box mesh.
- void **CreateCamera** (const std::string &_name, float _scale)

  Create a Camera mesh.
- void **CreateCone** (const std::string &_name, float _radius, float _height, int _rings, int _segments)

  Create a cone mesh.
- void **CreateCylinder** (const std::string &_name, float _radius, float _height, int _rings, int _segments)

  Create a cylinder mesh.
- void **CreatePlane** (const std::string &_name, const **math::Plane** &_plane, const **math::Vector2d** &_segments, const **math::Vector2d** &_uvTile)

  Create mesh for a plane.
- void **CreatePlane** (const std::string &_name, const **math::Vector3** &_normal, double _d, const **math::Vector2d** &_size, const **math::Vector2d** &_segments, const **math::Vector2d** &_uvTile)

  Create mesh for a plane.

- void **CreateSphere** (const std::string &_name, float _radius, int _rings, int _segments)

  Create a sphere mesh.
- void **CreateTube** (const std::string &_name, float _innerRadius, float _outterRadius, float _height, int _rings, int _segments)

  Create a tube mesh.
- void **GenSphericalTexCoord** (const **Mesh** ∗_mesh, **math::Vector3** _center)

  generate spherical texture coordinates
- const **Mesh** ∗ **GetMesh** (const std::string &_name) const

  Get a mesh by name.
- void **GetMeshAABB** (const **Mesh** ∗_mesh, **math::Vector3** &_center, **math::Vector3** &_min_xyz, **math::Vector3** &_max_xyz)

  Get mesh aabb and center.
- bool **HasMesh** (const std::string &_name) const

  Return true if the mesh exists.
- bool **IsValidFilename** (const std::string &_filename)

  Checks a path extension against the list of valid extensions.
- const **Mesh** ∗ **Load** (const std::string &_filename)

  Load a mesh from a file.

Additional Inherited Members

10.101.1 Detailed Description

Maintains and manages all meshes.

10.101.2 Member Function Documentation

10.101.2.1 void gazebo::common::MeshManager::AddMesh (Mesh ∗ *_mesh*)

Add a mesh to the manager.

This **MeshManager** (p. 516) takes ownership of the mesh and will destroy it. See ∼MeshManager.

Parameters

in	*the*	mesh to add.

10.101.2.2 void gazebo::common::MeshManager::CreateBox (const std::string & *_name,* const math::Vector3 & *_sides,* const math::Vector2d & *_uvCoords*)

Create a Box mesh.

Parameters

in	*_name*	the name of the new mesh
in	*_sides*	the x y x dimentions of eah side in meter
in	*_uvCoords*	the texture coordinates

10.101.2.3 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

10.101.2.4 void gazebo::common::MeshManager::CreateCone (const std::string & *_name,* float *_radius,* float *_height,* int *_rings,* int *_segments*)

Create a cone mesh.

Parameters

in	*_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

10.101.2.5 void gazebo::common::MeshManager::CreateCylinder (const std::string & *_name,* float *_radius,* float *_height,* int *_rings,* int *_segments*)

Create a cylinder mesh.

Parameters

in	*_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

10.101.2.6 void gazebo::common::MeshManager::CreatePlane (const std::string & *_name,* const **math::Plane** & *_plane,* const **math::Vector2d** & *_segments,* const **math::Vector2d** & *_uvTile*)

Create mesh for a plane.

Parameters

in	*_name*	
in	*_plane*	plane parameters
in	*_segments*	number of segments in x and y
in	*_uvTile*	the texture tile size in x and y

10.101.2.7 void gazebo::common::MeshManager::CreatePlane (const std::string & *_name*, const math::Vector3 & *_normal*, double *_d*, const math::Vector2d & *_size*, const math::Vector2d & *_segments*, const math::Vector2d & *_uvTile*)

Create mesh for a plane.

Parameters

in	*_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

10.101.2.8 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

10.101.2.9 void gazebo::common::MeshManager::CreateTube (const std::string & *_name*, float *_innerRadius*, float *_outterRadius*, float *_height*, int *_rings*, int *_segments*)

Create a tube mesh.

Generates rings inside and outside the cylinder Needs at least two rings and 3 segments

Parameters

in	*_name*	the name of the new mesh
in	*_innerRadius*	the inner radius of the tube in the x y plane
in	*_outterRadius*	the outer radius of the tube in the x y plane
in	*_height*	the height along z
in	*_rings*	the number of circles along the height
in	*_segments*	the number of segment per circle

10.101.2.10 void gazebo::common::MeshManager::GenSphericalTexCoord (const Mesh * *_mesh*, math::Vector3 *_center*)

generate spherical texture coordinates

10.101.2.11 const Mesh* gazebo::common::MeshManager::GetMesh (const std::string & *_name*) const

Get a mesh by name.

Parameters

in	*_name*	the name of the mesh to look for

Returns

the mesh or NULL if not found

10.101.2.12 void gazebo::common::MeshManager::GetMeshAABB (const Mesh * *_mesh*, math::Vector3 & *_center*, math::Vector3 & *_min_xyz*, math::Vector3 & *_max_xyz*)

Get mesh aabb and center.

Parameters

in	*_mesh*	the mesh
out	*_center*	the AAB center position
out	*_min_xyz*	the bounding box minimum
out	*_max_xyz*	the bounding box maximum

10.101.2.13 bool gazebo::common::MeshManager::HasMesh (const std::string & *_name*) const

Return true if the mesh exists.

Parameters

in	*_name*	the name of the mesh

10.101.2.14 bool gazebo::common::MeshManager::IsValidFilename (const std::string & *_filename*)

Checks a path extension against the list of valid extensions.

Returns

true if the file extension is loadable

10.101.2.15 const Mesh* gazebo::common::MeshManager::Load (const std::string & *_filename*)

Load a mesh from a file.

Parameters

in	*_filename*	the path to the mesh

Returns

a pointer to the created mesh

The documentation for this class was generated from the following file:

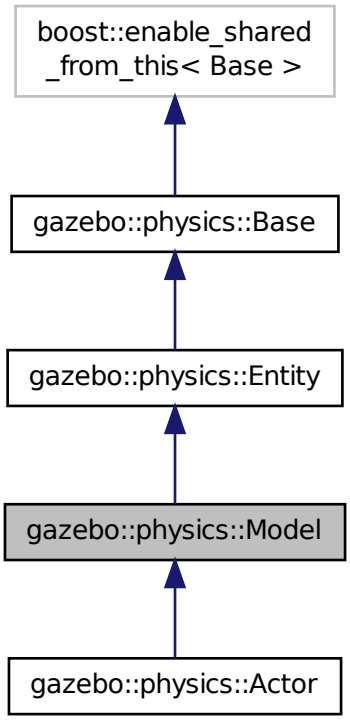
- **MeshManager.hh**

10.102 gazebo::physics::Model Class Reference

A model is a collection of links, joints, and plugins.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::Model:



Public Member Functions

- **Model** (**BasePtr** _parent)

  Constructor.
- virtual ~**Model** ()

  Destructor.
- void **AttachStaticModel** (**ModelPtr** &_model, **math::Pose** _offset)

  Attach a static model to this model.
- void **DetachStaticModel** (const std::string &_model)

  Detach a static model from this model.
- void **FillModelMsg** (msgs::Model &_msg) **GAZEBO_DEPRECATED**

  DEPRECATED.
- void **FillMsg** (msgs::Model &_msg)

Fill a model message.

- virtual void **Fini** ()

  Finalize the model.

- **Link_V GetAllLinks** () const

  *Construct and return a vector of **Link** (p. 454)'s in this model.*

- virtual **math::Box GetBoundingBox** () const

  Get the size of the bounding box.

- **JointPtr GetJoint** (unsigned int _index) const

  Get a joint by index.

- **JointPtr GetJoint** (const std::string &_name)

  Get a joint.

- unsigned int **GetJointCount** () const

  Get the number of joints.

- **LinkPtr GetLink** (const std::string &_name="canonical") const

  Get a link by name.

- **LinkPtr GetLink** (unsigned int _index) const

  Get a child link by index.

- **LinkPtr GetLinkById** (unsigned int _id) const

  Get a link by id.

- virtual **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the entity.

- virtual **math::Vector3 GetRelativeAngularVel** () const

  Get the angular velocity of the entity.

- virtual **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the entity.

- virtual **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the entity.

- virtual const **sdf::ElementPtr GetSDF** ()

  Get the SDF values for the model.

- **ModelState GetState** ()

  Get the current model state.

- virtual **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the entity in the world frame.

- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the entity in the world frame.

- virtual **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the entity in the world frame.

- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the entity in the world frame.

- virtual void **Init** ()

  Initialize the model.

- void **Load** (**sdf::ElementPtr** _sdf)

  Load the model.

- void **LoadPlugins** ()

  Load all plugins.

- void **ProcessMsg** (const msgs::Model &_msg)

  Update parameters from a model message.

- virtual void **RemoveChild** (**EntityPtr** _child)

  Remove a child.
- void **Reset** ()

  Reset the model.
- void **SetAngularAccel** (const **math::Vector3** &_vel)

  Set the angular acceleration of the model, and all its links.
- void **SetAngularVel** (const **math::Vector3** &_vel)

  Set the angular velocity of the model, and all its links.
- void **SetAutoDisable** (bool _disable)

  Allow the model the auto disable.
- void **SetCollideMode** (const std::string &_mode)

  *This is not implemented in **Link** (p. 454), which means this function doesn't do anything.*
- void **SetEnabled** (bool _enabled)

  Enable all the links in all the models.
- void **SetGravityMode** (const bool &_value)

  Set the gravity mode of the model.
- void **SetJointAnimation** (const std::map< std::string, **common::NumericAnimationPtr** > _anim, boost::function< void()> _onComplete=**NULL**)

  ***Joint** (p. 423) Animation.*
- void **SetJointPosition** (const std::string &_jointName, double _position)

  *Set the positions of a **Joint** (p. 423) by name.*
- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)

  Set the positions of a set of joints.
- void **SetLaserRetro** (const float _retro)

  Set the laser retro reflectiveness of the model.
- void **SetLinearAccel** (const **math::Vector3** &_vel)

  Set the linear acceleration of the model, and all its links.
- void **SetLinearVel** (const **math::Vector3** &_vel)

  Set the linear velocity of the model, and all its links.
- void **SetLinkWorldPose** (const **math::Pose** &_pose, std::string _linkName)

  *Set the Pose of the entire **Model** (p. 521) by specifying desired Pose of a **Link** (p. 454) within the **Model** (p. 521).*
- void **SetLinkWorldPose** (const **math::Pose** &_pose, const **LinkPtr** &_link)

  *Set the Pose of the entire **Model** (p. 521) by specifying desired Pose of a **Link** (p. 454) within the **Model** (p. 521).*
- void **SetState** (const **ModelState** &_state)

  Set the current model state.
- virtual void **StopAnimation** ()

  Stop the current animations.
- void **Update** ()

  Update the model.
- virtual void **UpdateParameters** (**sdf::ElementPtr** _sdf)

  Update the parameters using new sdf values.

Protected Member Functions

- virtual void **OnPoseChange** ()

  Callback when the pose of the model has been changed.

Protected Attributes

- std::vector< **ModelPtr** > **attachedModels**

  Used by ***Model::AttachStaticModel*** *(*p. *524).*
- std::vector< **math::Pose** > **attachedModelsOffset**

  used by ***Model::AttachStaticModel*** *(*p. *524).*

Additional Inherited Members

10.102.1 Detailed Description

A model is a collection of links, joints, and plugins.

10.102.2 Constructor & Destructor Documentation

10.102.2.1 gazebo::physics::Model::Model (**BasePtr** *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent object.

10.102.2.2 virtual gazebo::physics::Model::~Model ( ) [virtual]

Destructor.

10.102.3 Member Function Documentation

10.102.3.1 void gazebo::physics::Model::AttachStaticModel (**ModelPtr** & *_model,* **math::Pose** *_offset*)

Attach a static model to this model.

This function takes as input a static **Model** (p. 521), which is a **Model** (p. 521) that has been marked as static (no physics simulation), and attaches it to this **Model** (p. 521) with a given offset.

This function is useful when you want to simulate a grasp of a static object, or move a static object around using a dynamic model.

If you are in doubt, do not use this function.

Parameters

in	*_model*	Pointer to the static model.
in	*_offset*	Offset, relative to this **Model** (p. 521), to place _model.

10.102.3.2 void gazebo::physics::Model::DetachStaticModel (const std::string & *_model*)

Detach a static model from this model.

Parameters

in	*_model*	Name of an attached static model to remove.

See Also

Model::AttachStaticModel (p. 524).

10.102.3.3 void gazebo::physics::Model::FillModelMsg (msgs::Model & *_msg*)

DEPRECATED.

10.102.3.4 void gazebo::physics::Model::FillMsg (msgs::Model & *_msg*)

Fill a model message.

Parameters

in	*_msg*	Message to fill using this model's data.

10.102.3.5 virtual void gazebo::physics::Model::Fini ( ) [virtual]

Finalize the model.

Reimplemented from **gazebo::physics::Entity** (p. 342).

Reimplemented in **gazebo::physics::Actor** (p. 124).

10.102.3.6 Link_V gazebo::physics::Model::GetAllLinks ( ) const

Construct and return a vector of **Link** (p. 454)'s in this model.

Note this constructs the vector of **Link** (p. 454)'s on the fly, could be costly.

Returns

A vector of **Link** (p. 454)'s in this model.

10.102.3.7 virtual math::Box gazebo::physics::Model::GetBoundingBox ( ) const [virtual]

Get the size of the bounding box.

Returns

The bounding box.

Reimplemented from **gazebo::physics::Entity** (p. 342).

10.102.3.8 JointPtr gazebo::physics::Model::GetJoint (unsigned int *_index*) const

Get a joint by index.

Parameters

in	*_index*	Index of the joint.

Returns

A pointer to the joint, NULL if the index is invalid.

10.102.3.9 JointPtr gazebo::physics::Model::GetJoint (const std::string & *_name*)

Get a joint.

Parameters

in	*_name*	The name of the joint, specified in the world file.

Returns

Pointer to the joint, NULL if the name is invalid.

10.102.3.10 unsigned int gazebo::physics::Model::GetJointCount ( ) const

Get the number of joints.

Returns

Get the number of joints.

10.102.3.11 LinkPtr gazebo::physics::Model::GetLink (const std::string & *_name* = "canonical") const

Get a link by name.

Parameters

in	*_name*	Name of the link to get.

Returns

Pointer to the link, NULL if the name is invalid.

10.102.3.12 LinkPtr gazebo::physics::Model::GetLink (unsigned int *_index*) const

Get a child link by index.

Parameters

in	*_index*	Index of the link.

Returns

Point to the link, NULL if the index is invalid.

10.102.3.13 LinkPtr gazebo::physics::Model::GetLinkById (unsigned int *_id*) const

Get a link by id.

Returns

Pointer to the link, NULL if the id is invalid.

10.102.3.14 virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 343).

10.102.3.15 virtual math::Vector3 gazebo::physics::Model::GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the entity.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 343).

10.102.3.16 virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.102.3.17 virtual math::Vector3 gazebo::physics::Model::GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the entity.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.102.3.18 virtual const sdf::ElementPtr gazebo::physics::Model::GetSDF ( ) [virtual]

Get the SDF values for the model.

Returns

The SDF value for this model.

Reimplemented from **gazebo::physics::Base** (p. 153).

Reimplemented in **gazebo::physics::Actor** (p. 124).

10.102.3.19 ModelState gazebo::physics::Model::GetState ( )

Get the current model state.

Returns

The current model state.

10.102.3.20 virtual math::Vector3 gazebo::physics::Model::GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.102.3.21 virtual math::Vector3 gazebo::physics::Model::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the entity in the world frame.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.102.3.22 virtual math::Vector3 gazebo::physics::Model::GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 345).

10.102.3.23 virtual math::Vector3 gazebo::physics::Model::GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the entity in the world frame.

Returns

math::Vector3 (p. 902), set to 0, 0, 0 if the model has no body.

Reimplemented from **gazebo::physics::Entity** (p. 345).

10.102.3.24 virtual void gazebo::physics::Model::Init ( ) [virtual]

Initialize the model.

Reimplemented from **gazebo::physics::Base** (p. 153).

Reimplemented in **gazebo::physics::Actor** (p. 124).

10.102.3.25 void gazebo::physics::Model::Load (sdf::ElementPtr _sdf) [virtual]

Load the model.

Parameters

in	*_sdf*	SDF parameters to load from.

Reimplemented from **gazebo::physics::Entity** (p. 346).

10.102.3.26 void gazebo::physics::Model::LoadPlugins ( )

Load all plugins.

Load all plugins specified in the SDF for the model.

10.102.3.27 virtual void gazebo::physics::Model::OnPoseChange ( ) [protected], [virtual]

Callback when the pose of the model has been changed.

Implements **gazebo::physics::Entity** (p. 346).

10.102.3.28 void gazebo::physics::Model::ProcessMsg (const msgs::Model & _msg)

Update parameters from a model message.

Parameters

in	*_msg*	Message to process.

10.102.3.29 virtual void gazebo::physics::Model::RemoveChild (EntityPtr _child) [virtual]

Remove a child.

Parameters

in	*_child*	Remove a child entity.

10.102.3.30 void gazebo::physics::Model::Reset ( ) [virtual]

Reset the model.

Reimplemented from **gazebo::physics::Entity** (p. 346).

10.102.3.31 void gazebo::physics::Model::SetAngularAccel (const math::Vector3 & *_vel*)

Set the angular acceleration of the model, and all its links.

Parameters

in	*_vel*	The new angular acceleration

10.102.3.32 void gazebo::physics::Model::SetAngularVel (const math::Vector3 & *_vel*)

Set the angular velocity of the model, and all its links.

Parameters

in	*_vel*	The new angular velocity.

10.102.3.33 void gazebo::physics::Model::SetAutoDisable (bool *_disable*)

Allow the model the auto disable.

This is ignored if the model has joints.

Parameters

in	*_disable*	If true, the model is allowed to auto disable.

10.102.3.34 void gazebo::physics::Model::SetCollideMode (const std::string & *_mode*)

This is not implemented in **Link** (p. 454), which means this function doesn't do anything.

Set the collide mode of the model.

Parameters

in	*_mode*	The collision mode

10.102.3.35 void gazebo::physics::Model::SetEnabled (bool *_enabled*)

Enable all the links in all the models.

Parameters

in	*_enabled*	True to enable all the links.

10.102.3.36 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.

10.102.3.37 void gazebo::physics::Model::SetJointAnimation (const std::map< std::string, common::NumericAnimationPtr > *_anim,* boost::function< void() > *_onComplete* = NULL)

Joint (p. 423) Animation.

Parameters

in	*_anim*	Map of joint names to their position animation.
in	*_onComplete*	Callback function for when the animation completes.

10.102.3.38 void gazebo::physics::Model::SetJointPosition (const std::string & *_jointName,* double *_position*)

Set the positions of a **Joint** (p. 423) by name.

See Also

JointController::SetJointPosition (p. 439)

Parameters

in	*_jointName*	Name of the joint to set.
in	*_position*	Position to set the joint to.

10.102.3.39 void gazebo::physics::Model::SetJointPositions (const std::map< std::string, double > & *_jointPositions*)

Set the positions of a set of joints.

See Also

JointController::SetJointPositions (p. 440).

Parameters

in	*_jointPositions*	Map of joint names to their positions.

10.102.3.40 void gazebo::physics::Model::SetLaserRetro (const float _retro)

Set the laser retro reflectiveness of the model.

Parameters

in	*_retro*	Retro reflectance value.

10.102.3.41 void gazebo::physics::Model::SetLinearAccel (const math::Vector3 & _vel)

Set the linear acceleration of the model, and all its links.

Parameters

in	*_vel*	The new linear acceleration.

10.102.3.42 void gazebo::physics::Model::SetLinearVel (const math::Vector3 & _vel)

Set the linear velocity of the model, and all its links.

Parameters

in	*_vel*	The new linear velocity.

10.102.3.43 void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & _pose, std::string _linkName)

Set the Pose of the entire **Model** (p. 521) by specifying desired Pose of a **Link** (p. 454) within the **Model** (p. 521).

Doing so, keeps the configuration of the **Model** (p. 521) unchanged, i.e. all **Joint** (p. 423) angles are unchanged.

Parameters

in	*_pose*	Pose to set the link to.
in	*_linkName*	Name of the link to set.

10.102.3.44 void gazebo::physics::Model::SetLinkWorldPose (const math::Pose & _pose, const LinkPtr & _link)

Set the Pose of the entire **Model** (p. 521) by specifying desired Pose of a **Link** (p. 454) within the **Model** (p. 521).

Doing so, keeps the configuration of the **Model** (p. 521) unchanged, i.e. all **Joint** (p. 423) angles are unchanged.

Parameters

in	*_pose*	Pose to set the link to.
in	*_link*	Pointer to the link to set.

10.102.3.45 void gazebo::physics::Model::SetState (const ModelState & _state)

Set the current model state.

Parameters

in	*_state*	**State** (p. 813) to set the model to.

10.102.3.46 virtual void gazebo::physics::Model::StopAnimation ( ) [virtual]

Stop the current animations.

Reimplemented from **gazebo::physics::Entity** (p. 348).

10.102.3.47 void gazebo::physics::Model::Update ( ) [virtual]

Update the model.

Reimplemented from **gazebo::physics::Base** (p. 157).

10.102.3.48 virtual void gazebo::physics::Model::UpdateParameters (sdf::ElementPtr *_sdf*) [virtual]

Update the parameters using new sdf values.

Parameters

in	*_sdf*	SDF values to update from.

Reimplemented from **gazebo::physics::Entity** (p. 348).

Reimplemented in **gazebo::physics::Actor** (p. 125).

10.102.4 Member Data Documentation

10.102.4.1 std::vector<ModelPtr> gazebo::physics::Model::attachedModels [protected]

Used by **Model::AttachStaticModel** (p. 524).

10.102.4.2 std::vector<math::Pose> gazebo::physics::Model::attachedModelsOffset [protected]

used by **Model::AttachStaticModel** (p. 524).

The documentation for this class was generated from the following file:

- **Model.hh**

10.103 gazebo::common::ModelDatabase Class Reference

Connects to model database, and has utility functions to find models.

`#include <common/common.hh>`

Static Public Member Functions

- static void **DownloadDependencies** (const std::string &_path)

Download all dependencies for a give model path.

- static std::string **GetManifest** (const std::string &_uri)

  Return the manifest.xml file as a string.

- static std::string **GetModelFile** (const std::string &_uri)

  Get a model's SDF file based on a URI.

- static std::string **GetModelName** (const std::string &_uri)

  Get the name of a model based on a URI.

- static std::string **GetModelPath** (const std::string &_uri)

  Get the local path to a model.

- static std::map< std::string, std::string > **GetModels** ()

  Returns the dictionary of all the model names.

- static std::string **GetURI** ()

  Returns the the global model database URI.

- static bool **HasModel** (const std::string &_modelName)

  Returns true if the model exists on the database.

10.103.1 Detailed Description

Connects to model database, and has utility functions to find models.

The documentation for this class was generated from the following file:

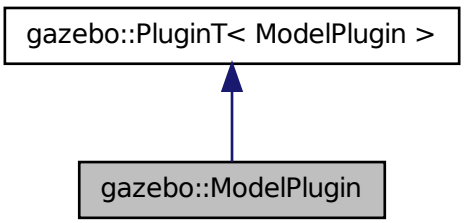
- **ModelDatabase.hh**

10.104 gazebo::ModelPlugin Class Reference

A plugin with access to **physics::Model** (p. 521).

`#include <Plugin.hh>`

Inheritance diagram for gazebo::ModelPlugin:



Public Member Functions

- **ModelPlugin** ()

  Constructor.
- virtual ∼**ModelPlugin** ()

  Destructor.
- virtual void **Init** ()

  Override this method for custom plugin initialization behavior.
- virtual void **Load** (**physics::ModelPtr** _model, **sdf::ElementPtr** _sdf)=0

  Load function.
- virtual void **Reset** ()

  Override this method for custom plugin reset behavior.

Additional Inherited Members

10.104.1 Detailed Description

A plugin with access to **physics::Model** (p. 521).

See `reference`.

10.104.2 Constructor & Destructor Documentation

10.104.2.1 gazebo::ModelPlugin::ModelPlugin ( ) [inline]

Constructor.

References gazebo::MODEL_PLUGIN, and gazebo::PluginT< ModelPlugin >::type.

10.104.2.2 virtual gazebo::ModelPlugin::∼ModelPlugin ( ) [inline], [virtual]

Destructor.

10.104.3 Member Function Documentation

10.104.3.1 virtual void gazebo::ModelPlugin::Init ( ) [inline], [virtual]

Override this method for custom plugin initialization behavior.

10.104.3.2 virtual void gazebo::ModelPlugin::Load (physics::ModelPtr *_model*, sdf::ElementPtr *_sdf*) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

_model	Pointer the Model
_sdf	Pointer the the SDF element of the plugin.

10.104.3.3 virtual void gazebo::ModelPlugin::Reset ( ) [inline], [virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

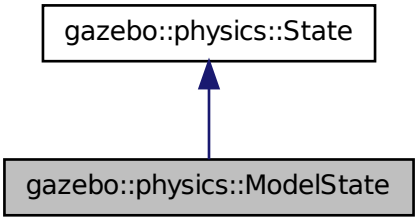
- **common/Plugin.hh**

10.105 gazebo::physics::ModelState Class Reference

Store state information of a **physics::Model** (p. 521) object.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::ModelState:



Public Member Functions

- **ModelState** ()
  Default constructor.
- **ModelState** (**ModelPtr** _model)
  Constructor.
- virtual ~**ModelState** ()
  Destructor.
- void **FillStateSDF** (**sdf::ElementPtr** _elem)
  *Fill a **State** (p. 813) SDF element with state info.*
- **JointState GetJointState** (unsigned int _index) const
  *Get a **Joint** (p. 423) state.*
- **JointState GetJointState** (const std::string &_jointName) const
  *Get a **Joint** (p. 423) state by **Joint** (p. 423) name.*
- unsigned int **GetJointStateCount** () const
  Get the number of joint states.
- **LinkState GetLinkState** (unsigned int _index) const
  Get a link state.
- **LinkState GetLinkState** (const std::string &_linkName) const

*Get a link state by **Link** (p. 454) name.*

- unsigned int **GetLinkStateCount** () const

  Get the number of link states.

- **math::Pose GetPose** () const

  Get the stored model pose.

- virtual void **Load** (**sdf::ElementPtr** _elem)

  Load state from SDF element.

- void **UpdateModelSDF** (**sdf::ElementPtr** _elem)

  *Update a **Model** (p. 521) SDF element with this state info.*

Additional Inherited Members

10.105.1 Detailed Description

Store state information of a **physics::Model** (p. 521) object.

This class captures the entire state of a **Model** (p. 521) at one specific time during a simulation run.

State (p. 813) of a **Model** (p. 521) includes the state of all its child Links and Joints.

10.105.2 Constructor & Destructor Documentation

10.105.2.1 gazebo::physics::ModelState::ModelState ( )

Default constructor.

10.105.2.2 gazebo::physics::ModelState::ModelState (**ModelPtr** *_model*) [explicit]

Constructor.

Build a **ModelState** (p. 536) from an existing **Model** (p. 521).

Parameters

in	*_model*	Pointer to the model from which to gather state info.

10.105.2.3 virtual gazebo::physics::ModelState::~ModelState ( ) [virtual]

Destructor.

10.105.3 Member Function Documentation

10.105.3.1 void gazebo::physics::ModelState::FillStateSDF (**sdf::ElementPtr** *_elem*)

Fill a **State** (p. 813) SDF element with state info.

Stored state information into an SDF::Element pointer.

Parameters

in	*_elem*	Pointer to the SDF::Element which recieves the data.

10.105.3.2 JointState gazebo::physics::ModelState::GetJointState (unsigned int *_index*) const

Get a **Joint** (p. 423) state.

Return a **JointState** (p. 442) based on a index, where index is between 0...**ModelState::GetJointStateCount()** (p. 538).

Parameters

in	*_index*	Index of a **JointState** (p. 442).

Returns

State (p. 813) of a **Joint** (p. 423).

10.105.3.3 JointState gazebo::physics::ModelState::GetJointState (const std::string & *_jointName*) const

Get a **Joint** (p. 423) state by **Joint** (p. 423) name.

Searches through all JointStates. Returns the **JointState** (p. 442) with the matching name, if any.

Parameters

in	*_jointName*	Name of the **JointState** (p. 442).

Returns

State (p. 813) of the **Joint** (p. 423).

10.105.3.4 unsigned int gazebo::physics::ModelState::GetJointStateCount ( ) const

Get the number of joint states.

Returns the number of JointStates recorded.

Returns

Number of JointStates.

10.105.3.5 LinkState gazebo::physics::ModelState::GetLinkState (unsigned int *_index*) const

Get a link state.

Get a **Link** (p. 454) **State** (p. 813) based on an index, where index is in the range of 0...**ModelState::GetLinkStateCount** (p. 539)

Parameters

in	*_index*	Index of the **LinkState** (p. 473)

Returns

State (p. 813) of the **Link** (p. 454)

10.105.3.6 LinkState gazebo::physics::ModelState::GetLinkState (const std::string & *_linkName*) const

Get a link state by **Link** (p. 454) name.

Searches through all LinkStates. Returns the **LinkState** (p. 473) with the matching name, if any.

Parameters

in	*_linkName*	Name of the **LinkState** (p. 473)

Returns

State (p. 813) of the **Link** (p. 454).

10.105.3.7 unsigned int gazebo::physics::ModelState::GetLinkStateCount ( ) const

Get the number of link states.

This returns the number of Links recorded.

Returns

Number of **LinkState** (p. 473) recorded

10.105.3.8 math::Pose gazebo::physics::ModelState::GetPose ( ) const

Get the stored model pose.

Returns

The **math::Pose** (p. 677) of the **Model** (p. 521)

10.105.3.9 virtual void gazebo::physics::ModelState::Load (sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Load **ModelState** (p. 536) information from stored data in and SDF::Element

Parameters

in	*_elem*	Pointer to the SDF::Element containing state info.

Implements **gazebo::physics::State** (p. 815).

10.105.3.10 void gazebo::physics::ModelState::UpdateModelSDF (sdf::ElementPtr *_elem*)

Update a **Model** (p. 521) SDF element with this state info.

Set the values in a **Model** (p. 521)'s SDF::Element with the information. stored in this instance.

Parameters

in	*_elem*	Pointer to a Models's SDF::Element.

The documentation for this class was generated from the following file:

- **ModelState.hh**

10.106 gazebo::common::MouseEvent Class Reference

Generic description of a mouse event.

`#include <MouseEvent.hh>`

Public Types

- enum **Buttons** { **NO_BUTTON** = 0x0, **LEFT** = 0x1, **MIDDLE** = 0x2, **RIGHT** = 0x4 }

  Standard mouse buttons enumeration.
- enum **EventType** {
  NO_EVENT, **MOVE**, **PRESS**, **RELEASE**,
  SCROLL }

  Mouse event types enumeration.

Public Member Functions

- **MouseEvent** ()

  Constructor.

Public Attributes

- bool **alt**

  Alt key press flag.
- unsigned int **button**

  The button which caused the event.
- unsigned int **buttons**

  State of the buttons when the event was generated.
- bool **control**

  Control key press flag.
- bool **dragging**

  Flag for mouse drag motion.
- float **moveScale**

  Scaling factor.
- **math::Vector2i pos**

  Mouse pointer position on the screen.
- **math::Vector2i pressPos**

Position of button press.

- **math::Vector2i prevPos**

  Previous position.

- **math::Vector2i scroll**

  Scroll position.

- bool **shift**

  Shift key press flag.

- **EventType type**

  Event type.

10.106.1 Detailed Description

Generic description of a mouse event.

10.106.2 Member Enumeration Documentation

10.106.2.1 enum gazebo::common::MouseEvent::Buttons

Standard mouse buttons enumeration.

Enumerator:

NO_BUTTON

LEFT

MIDDLE

RIGHT

10.106.2.2 enum gazebo::common::MouseEvent::EventType

Mouse event types enumeration.

Enumerator:

NO_EVENT

MOVE

PRESS

RELEASE

SCROLL

10.106.3 Constructor & Destructor Documentation

10.106.3.1 gazebo::common::MouseEvent::MouseEvent ( ) [inline]

Constructor.

10.106.4 Member Data Documentation

10.106.4.1 bool gazebo::common::MouseEvent::alt

Alt key press flag.

10.106.4.2 unsigned int gazebo::common::MouseEvent::button

The button which caused the event.

10.106.4.3 unsigned int gazebo::common::MouseEvent::buttons

State of the buttons when the event was generated.

10.106.4.4 bool gazebo::common::MouseEvent::control

Control key press flag.

10.106.4.5 bool gazebo::common::MouseEvent::dragging

Flag for mouse drag motion.

10.106.4.6 float gazebo::common::MouseEvent::moveScale

Scaling factor.

10.106.4.7 math::Vector2i gazebo::common::MouseEvent::pos

Mouse pointer position on the screen.

10.106.4.8 math::Vector2i gazebo::common::MouseEvent::pressPos

Position of button press.

10.106.4.9 math::Vector2i gazebo::common::MouseEvent::prevPos

Previous position.

10.106.4.10 math::Vector2i gazebo::common::MouseEvent::scroll

Scroll position.

10.106.4.11 bool gazebo::common::MouseEvent::shift

Shift key press flag.

10.106.4.12 EventType gazebo::common::MouseEvent::type

Event type.

The documentation for this class was generated from the following file:

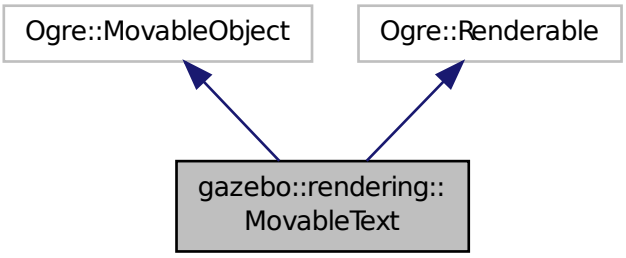
- **MouseEvent.hh**

10.107 gazebo::rendering::MovableText Class Reference

Movable text.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::MovableText:



Public Types

- enum **HorizAlign** { **H_LEFT**, **H_CENTER** }

  Horizontal alignment.
- enum **VertAlign** { **V_BELOW**, **V_ABOVE** }

  vertical alignment

Public Member Functions

- **MovableText** ()

  Constructor.
- virtual ~**MovableText** ()

  Destructor.
- **math::Box GetAABB** ()

  Get the axis aligned bounding box of the text.
- float **GetBaseline** () const

  Get the baseline height.
- float **GetCharHeight** () const

Set the height of a characters return Height of the characters.

- const **common::Color** & **GetColor** () const

  Get the text color.
- const std::string & **GetFont** () const

  Get the font.
- bool **GetShowOnTop** () const

  True = text is displayed on top.
- float **GetSpaceWidth** () const

  Get the width of a space.
- const std::string & **GetText** () const

  Get the displayed text.
- void **Load** (const std::string &_name, const std::string &_text, const std::string &_fontName="Arial", float _charHeight=1.0, const **common::Color** &_color=**common::Color::White**)

  Loads text and font info.
- void **SetBaseline** (float _height)

  Set the baseline height of the text.
- void **SetCharHeight** (float _height)

  Set the height of a character.
- void **SetColor** (const **common::Color** &_color)

  Set the text color.
- void **SetFontName** (const std::string &_font)

  Set the font.
- void **SetShowOnTop** (bool _show)

  True = text always is displayed 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

10.107.1 Detailed Description

Movable text.

10.107.2 Member Enumeration Documentation

10.107.2.1 enum gazebo::rendering::MovableText::HorizAlign

Horizontal alignment.

Enumerator:

H_LEFT Left alignment.

H_CENTER Center alignment.

10.107.2.2 enum gazebo::rendering::MovableText::VertAlign

vertical alignment

Enumerator:

V_BELOW Align below.

V_ABOVE Align above.

10.107.3 Constructor & Destructor Documentation

10.107.3.1 gazebo::rendering::MovableText::MovableText ( )

Constructor.

10.107.3.2 virtual gazebo::rendering::MovableText::~MovableText ( ) [virtual]

Destructor.

10.107.4 Member Function Documentation

10.107.4.1 void gazebo::rendering::MovableText::_setupGeometry ( ) [protected]

10.107.4.2 void gazebo::rendering::MovableText::_updateColors ( ) [protected]

10.107.4.3 math::Box gazebo::rendering::MovableText::GetAABB ( )

Get the axis aligned bounding box of the text.

Returns

The axis aligned bounding box.

10.107.4.4 float gazebo::rendering::MovableText::GetBaseline ( ) const

Get the baseline height.

Returns

Baseline height

10.107.4.5 float gazebo::rendering::MovableText::getBoundingRadius ( ) const [protected]

10.107.4.6 float gazebo::rendering::MovableText::GetCharHeight ( ) const

Set the height of a characters return Height of the characters.

10.107.4.7 const common::Color& gazebo::rendering::MovableText::GetColor ( ) const

Get the text color.

Returns

Textuer color.

10.107.4.8 const std::string& gazebo::rendering::MovableText::GetFont ( ) const

Get the font.

Returns

The font name

10.107.4.9 const Ogre::LightList& gazebo::rendering::MovableText::getLights (void) const [protected]

10.107.4.10 const Ogre::MaterialPtr& gazebo::rendering::MovableText::getMaterial (void) const [protected]

10.107.4.11 void gazebo::rendering::MovableText::getRenderOperation (Ogre::RenderOperation & *op*) [protected]

10.107.4.12 bool gazebo::rendering::MovableText::GetShowOnTop ( ) const

True = text is displayed on top.

Returns

True if MovableText::SetShownOnTop(true) was called.

10.107.4.13 float gazebo::rendering::MovableText::GetSpaceWidth ( ) const

Get the width of a space.

Returns

Space width

10.107.4.14 float gazebo::rendering::MovableText::getSquaredViewDepth (const Ogre::Camera * *cam*) const [protected]

10.107.4.15 const std::string& gazebo::rendering::MovableText::GetText ( ) const

Get the displayed text.

Returns

The displayed text.

10.107.4.16 void gazebo::rendering::MovableText::getWorldTransforms (Ogre::Matrix4 * *xform*) const [protected]

10.107.4.17 void gazebo::rendering::MovableText::Load (const std::string & *_name*, const std::string & *_text*, const std::string & *_fontName* = "Arial", float *_charHeight* = 1.0, const common::Color & *_color* = common::Color::White)

Loads text and font info.

Parameters

in	*_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

10.107.4.18 void gazebo::rendering::MovableText::SetBaseline (float *_height*)

Set the baseline height of the text.

Parameters

in	*_height*	Baseline height

10.107.4.19 void gazebo::rendering::MovableText::SetCharHeight (float *_height*)

Set the height of a character.

Parameters

in	*_height*	Height of the characters.

10.107.4.20 void gazebo::rendering::MovableText::SetColor (const common::Color & *_color*)

Set the text color.

Parameters

in	*_color*	Text color.

10.107.4.21 void gazebo::rendering::MovableText::SetFontName (const std::string & *_font*)

Set the font.

Parameters

in	*_font*	Name of the font

10.107.4.22 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.

10.107.4.23 void gazebo::rendering::MovableText::SetSpaceWidth (float *_width*)

Set the width of a space.

Parameters

in	*_width*	space width

10.107.4.24 void gazebo::rendering::MovableText::SetText (const std::string & *_text*)

Set the text to display.

Parameters

in	*The*	text to display.

10.107.4.25 void gazebo::rendering::MovableText::SetTextAlignment (const HorizAlign & *_hAlign,* const VertAlign & *_vAlign*)

Set the alignment of the text.

Parameters

in	*_hAlign*	Horizontal alignment
in	*_vAlign*	Vertical alignment

10.107.4.26 void gazebo::rendering::MovableText::Update ( )

Update the text.

10.107.4.27 virtual void gazebo::rendering::MovableText::visitRenderables (Ogre::Renderable::Visitor * *visitor,* bool *debug =* `false`) [virtual]

The documentation for this class was generated from the following file:

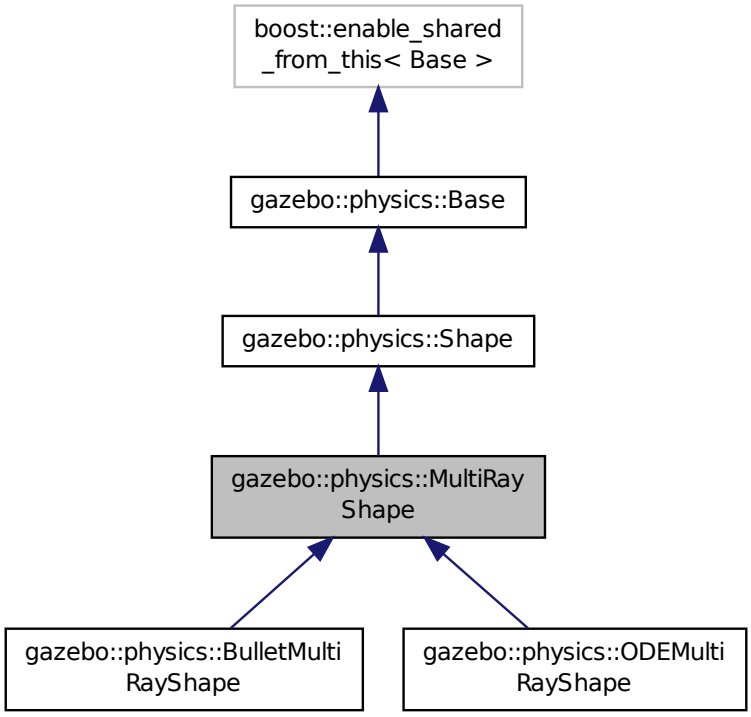
- **MovableText.hh**

10.108 gazebo::physics::MultiRayShape Class Reference

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::MultiRayShape:



Public Member Functions

- **MultiRayShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**MultiRayShape** ()

  Destructor.
- template<typename T >
  event::ConnectionPtr ConnectNewLaserScans (T _subscriber)

  Connect a to the new laser scan signal.
- void **DisconnectNewLaserScans** (**event::ConnectionPtr** &_conn)

  Disconnect from the new laser scans signal.
- void **FillMsg** (msgs::Geometry &_msg)

  This function is not implemented.
- int **GetFiducial** (int _index)

Get detected fiducial value for a ray.

- **math::Angle GetMaxAngle** () const

  Get the maximum angle.

- double **GetMaxRange** () const

  Get the maximum range.

- **math::Angle GetMinAngle** () const

  Get the minimum angle.

- double **GetMinRange** () const

  Get the minimum range.

- double **GetRange** (int _index)

  Get detected range for a ray.

- double **GetResRange** () const

  Get the range resolution.

- double **GetRetro** (int _index)

  Get detected retro (intensity) value for a ray.

- int **GetSampleCount** () const

  Get the horizontal sample count.

- double **GetScanResolution** () const

  Get the horizontal resolution.

- **math::Angle GetVerticalMaxAngle** () const

  Get the vertical max angle.

- **math::Angle GetVerticalMinAngle** () const

  Get the vertical min angle.

- int **GetVerticalSampleCount** () const

  Get the vertical sample count.

- double **GetVerticalScanResolution** () const

  Get the vertical range resolution.

- virtual void **Init** ()

  Init the shape.

- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

  This function is not implemented.

- void **Update** ()

  Update the ray collisions.

Protected Member Functions

- virtual void **AddRay** (const **math::Vector3** &_start, const **math::Vector3** &_end)

  Add a ray to the collision.

- virtual void **UpdateRays** ()=0

  Physics engine specific method for updating the rays.

Protected Attributes

- **sdf::ElementPtr horzElem**

  Horizontal SDF element pointer.
- **event::EventT**< void()> **newLaserScans**

  New laser scans event.
- **math::Pose offset**

  Pose offset of all the rays.
- **sdf::ElementPtr rangeElem**

  Range SDF element pointer.
- **sdf::ElementPtr rayElem**

  Ray SDF element pointer.
- std::vector< **RayShapePtr** > **rays**

  Ray data.
- **sdf::ElementPtr scanElem**

  Scan SDF element pointer.
- **sdf::ElementPtr vertElem**

  Vertical SDF element pointer.

Additional Inherited Members

10.108.1 Detailed Description

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

10.108.2 Constructor & Destructor Documentation

10.108.2.1 gazebo::physics::MultiRayShape::MultiRayShape (CollisionPtr *_parent*) [explicit]

Constructor.

Parameters

in	*_parent*	Parent collision shape.

10.108.2.2 virtual gazebo::physics::MultiRayShape::~MultiRayShape ( ) [virtual]

Destructor.

10.108.3 Member Function Documentation

10.108.3.1 virtual void gazebo::physics::MultiRayShape::AddRay (const math::Vector3 & *_start,* const math::Vector3 & *_end*) [protected], [virtual]

Add a ray to the collision.

Parameters

in	*_start*	Start of the ray.
in	*_end*	End of the ray.

Reimplemented in **gazebo::physics::ODEMultiRayShape** (p. 609), and **gazebo::physics::BulletMultiRayShape** (p. 202).

10.108.3.2 template<typename T > event::ConnectionPtr gazebo::physics::MultiRayShape::ConnectNewLaserScans (T *_subscriber*) [inline]

Connect a to the new laser scan signal.

Parameters

in	*_subscriber*	Callback function.

Returns

The connection, which must be kept in scope.

References gazebo::event::EventT< T >::Connect(), and newLaserScans.

10.108.3.3 void gazebo::physics::MultiRayShape::DisconnectNewLaserScans (event::ConnectionPtr & *_conn*) [inline]

Disconnect from the new laser scans signal.

Parameters

in	*_conn*	Connection to remove.

References gazebo::event::EventT< T >::Disconnect(), and newLaserScans.

10.108.3.4 void gazebo::physics::MultiRayShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

This function is not implemented.

Fill a message with this shape's values.

Parameters

out	*_msg*	Message that contains the shape's values.

Implements **gazebo::physics::Shape** (p. 781).

10.108.3.5 int gazebo::physics::MultiRayShape::GetFiducial (int *_index*)

Get detected fiducial value for a ray.

Parameters

in	*_index*	Index of the ray.

Returns

Fiducial value for the ray.

10.108.3.6 math::Angle gazebo::physics::MultiRayShape::GetMaxAngle ( ) const

Get the maximum angle.

Returns

Maximum angle of ray scan.

10.108.3.7 double gazebo::physics::MultiRayShape::GetMaxRange ( ) const

Get the maximum range.

Returns

Maximum range of all the rays.

10.108.3.8 math::Angle gazebo::physics::MultiRayShape::GetMinAngle ( ) const

Get the minimum angle.

Returns

Minimum angle of ray scan.

10.108.3.9 double gazebo::physics::MultiRayShape::GetMinRange ( ) const

Get the minimum range.

Returns

Minimum range of all the rays.

10.108.3.10 double gazebo::physics::MultiRayShape::GetRange (int *_index*)

Get detected range for a ray.

Parameters

in	*_index*	Index of the ray.

Returns

Returns DBL_MAX for no detection.

10.108.3.11 double gazebo::physics::MultiRayShape::GetResRange ( ) const

Get the range resolution.

Returns

Range resolution of all the rays.

10.108.3.12 double gazebo::physics::MultiRayShape::GetRetro (int *_index*)

Get detected retro (intensity) value for a ray.

Parameters

in	*_index*	Index of the ray.

Returns

Retro value for the ray.

10.108.3.13 int gazebo::physics::MultiRayShape::GetSampleCount ( ) const

Get the horizontal sample count.

Returns

Horizontal sample count.

10.108.3.14 double gazebo::physics::MultiRayShape::GetScanResolution ( ) const

Get the horizontal resolution.

Returns

Horizontal resolution.

10.108.3.15 math::Angle gazebo::physics::MultiRayShape::GetVerticalMaxAngle ( ) const

Get the vertical max angle.

Returns

Verticam max angle.

10.108.3.16 math::Angle gazebo::physics::MultiRayShape::GetVerticalMinAngle ( ) const

Get the vertical min angle.

Returns

Vertical min angle.

10.108.3.17 int gazebo::physics::MultiRayShape::GetVerticalSampleCount ( ) const

Get the vertical sample count.

Returns

Verical sample count.

10.108.3.18 double gazebo::physics::MultiRayShape::GetVerticalScanResolution ( ) const

Get the vertical range resolution.

Returns

Vertical range resolution.

10.108.3.19 virtual void gazebo::physics::MultiRayShape::Init ( ) [virtual]

Init the shape.

Implements **gazebo::physics::Shape** (p. 781).

10.108.3.20 virtual void gazebo::physics::MultiRayShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

This function is not implemented.

Update the ray based on a message.

Parameters

in	*_msg*	Message to update from.

Implements **gazebo::physics::Shape** (p. 782).

10.108.3.21 void gazebo::physics::MultiRayShape::Update ( ) [virtual]

Update the ray collisions.

Reimplemented from **gazebo::physics::Base** (p. 157).

10.108.3.22 virtual void gazebo::physics::MultiRayShape::UpdateRays ( ) [protected], [pure virtual]

Physics engine specific method for updating the rays.

Implemented in **gazebo::physics::BulletMultiRayShape** (p. 202), and **gazebo::physics::ODEMultiRayShape** (p. 609).

10.108.4 Member Data Documentation

10.108.4.1 sdf::ElementPtr gazebo::physics::MultiRayShape::horzElem [protected]

Horizontal SDF element pointer.

10.108.4.2 event::EventT<void()> gazebo::physics::MultiRayShape::newLaserScans [protected]

New laser scans event.

Referenced by ConnectNewLaserScans(), and DisconnectNewLaserScans().

10.108.4.3 math::Pose gazebo::physics::MultiRayShape::offset [protected]

Pose offset of all the rays.

10.108.4.4 sdf::ElementPtr gazebo::physics::MultiRayShape::rangeElem [protected]

Range SDF element pointer.

10.108.4.5 sdf::ElementPtr gazebo::physics::MultiRayShape::rayElem [protected]

Ray SDF element pointer.

10.108.4.6 std::vector<RayShapePtr> gazebo::physics::MultiRayShape::rays [protected]

Ray data.

10.108.4.7 sdf::ElementPtr gazebo::physics::MultiRayShape::scanElem [protected]

Scan SDF element pointer.

10.108.4.8 sdf::ElementPtr gazebo::physics::MultiRayShape::vertElem [protected]

Vertical SDF element pointer.

The documentation for this class was generated from the following file:

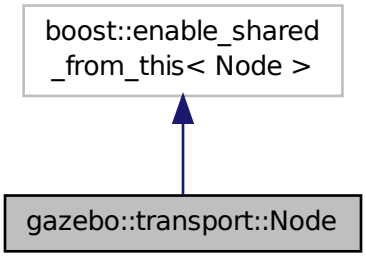
- **MultiRayShape.hh**

10.109 gazebo::transport::Node Class Reference

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

`#include <Node.hh>`

Inheritance diagram for gazebo::transport::Node:



Public Member Functions

- **Node** ()

  Constructor.
- virtual ~**Node** ()

  Destructor.
- template<typename M >
  transport::PublisherPtr Advertise (const std::string &topic, unsigned int _queueLimit=1000, bool _latch=false)

  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** ()
- unsigned int **GetId** () const

  Get the unique ID of the node.
- std::string **GetMsgType** (const std::string &_topic) const

  Get the message type for a topic.
- std::string **GetTopicNamespace** () const

  Get the topic namespace for this node.
- bool **HandleData** (const std::string &_topic, const std::string &_msg)
- void **Init** (const std::string &_space="")

  Init the node.
- void **InsertLatchedMsg** (const std::string &_topic, const std::string &_msg)

  Add a latched message to the node for publication.
- void **ProcessIncoming** ()
- void **ProcessPublishers** ()

  Process all publishers, which has each publisher send it's most recent message over the wire.
- template<typename M , typename T >
  SubscriberPtr Subscribe (const std::string &topic, void(T::*fp)(const M const *&), T *obj, bool _latching=false)

Subscribe to a topic, and return data on the callback.

- template<typename M >
  SubscriberPtr Subscribe (const std::string &topic, void(*fp)(const M const *&), bool _latching=false)

  Subscribe to a topic, and return data on the callback.

10.109.1 Detailed Description

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

10.109.2 Constructor & Destructor Documentation

10.109.2.1 gazebo::transport::Node::Node ( )

Constructor.

10.109.2.2 virtual gazebo::transport::Node::~Node ( ) [virtual]

Destructor.

10.109.3 Member Function Documentation

10.109.3.1 template<typename M > transport::PublisherPtr gazebo::transport::Node::Advertise (const std::string & *topic*, unsigned int *_queueLimit* = 1000, bool *_latch* = false) [inline]

Adverise a topic.

References DecodeTopicName(), and SingletonT< T >::Instance().

10.109.3.2 std::string gazebo::transport::Node::DecodeTopicName (const std::string & *topic*)

Decode a topic name.

Referenced by Advertise(), and Subscribe().

10.109.3.3 std::string gazebo::transport::Node::EncodeTopicName (const std::string & *topic*)

Encode a topic name.

10.109.3.4 void gazebo::transport::Node::Fini ( )

10.109.3.5 unsigned int gazebo::transport::Node::GetId ( ) const

Get the unique ID of the node.

10.109.3.6 std::string gazebo::transport::Node::GetMsgType (const std::string & *_topic*) const

Get the message type for a topic.

10.109.3.7 std::string gazebo::transport::Node::GetTopicNamespace ( ) const

Get the topic namespace for this node.

Returns

The namespace

10.109.3.8 bool gazebo::transport::Node::HandleData (const std::string & *_topic,* const std::string & *_msg*)

10.109.3.9 void gazebo::transport::Node::Init (const std::string & *_space* = " ")

Init the node.

Parameters

space	Set the global namespace of all topics. If left blank, the topic will initialize to the first namespace on the **Master** (p. 487)

10.109.3.10 void gazebo::transport::Node::InsertLatchedMsg (const std::string & *_topic,* const std::string & *_msg*)

Add a latched message to the node for publication.

This is called when a subscription is connected to a publication.

Parameters

in	*_topic*	Name of the topic to publish data on.
in	*_msg*	The message to publish.

10.109.3.11 void gazebo::transport::Node::ProcessIncoming ( )

10.109.3.12 void gazebo::transport::Node::ProcessPublishers ( )

Process all publishers, which has each publisher send it's most recent message over the wire.

This is for internal use only

10.109.3.13 template<typename M , typename T > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & *topic,* void(T::*)(const M const *&) *fp,* T * *obj,* bool *_latching* = `false`) [inline]

Subscribe to a topic, and return data on the callback.

References DecodeTopicName(), and SingletonT< T >::Instance().

10.109.3.14 template<typename M > SubscriberPtr gazebo::transport::Node::Subscribe (const std::string & *topic,* void(*)(const M const *&) *fp,* bool *_latching* = `false`) [inline]

Subscribe to a topic, and return data on the callback.

References DecodeTopicName(), and SingletonT< T >::Instance().

The documentation for this class was generated from the following file:

- **Node.hh**

10.110 gazebo::common::NodeAnimation Class Reference

Node animation.

`#include <SkeletonAnimation.hh>`

Public Member Functions

- **NodeAnimation** (const std::string &_name)

  constructor
- ~**NodeAnimation** ()

  Destructor. It empties the key frames list.
- void **AddKeyFrame** (const double _time, const **math::Matrix4** _trans)

  Adds a key frame at a specific time.
- void **AddKeyFrame** (const double _time, const **math::Pose** _pose)

  Adds a key fram at a specific time.
- **math::Matrix4 GetFrameAt** (double _time, bool _loop=true) const

  Returns a frame transformation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.
- unsigned int **GetFrameCount** () const

  Returns the number of key frames.
- void **GetKeyFrame** (const unsigned int _i, double &_time, **math::Matrix4** &_trans) const

  Finds a key frame using the index.
- std::pair< double, **math::Matrix4** > **GetKeyFrame** (const unsigned int _i) const

  Returns a key frame using the index.
- double **GetLength** () const

  Returns the duration of the animations.
- std::string **GetName** () const

  Returns the name.
- double **GetTimeAtX** (const double _x) const

  Returns the time where a transformation's translational value along the X axis is equal to _x.
- void **Scale** (const double _scale)

  Scales each transformation in the key frames.
- void **SetName** (const std::string &_name)

  Changes the name of the animation.

Protected Attributes

- std::map< double, **math::Matrix4** > **keyFrames**

  the dictionary of key frames, indexed by time
- double **length**

  the duration of the animations (time of last key frame)
- std::string **name**

  the name of the animation

10.110.1 Detailed Description

Node animation.

10.110.2 Constructor & Destructor Documentation

10.110.2.1 gazebo::common::NodeAnimation::NodeAnimation (const std::string & *_name*)

constructor

Parameters

in	*_name*	the name of the node

10.110.2.2 gazebo::common::NodeAnimation::~NodeAnimation ( )

Destructor. It empties the key frames list.

10.110.3 Member Function Documentation

10.110.3.1 void gazebo::common::NodeAnimation::AddKeyFrame (const double *_time,* const math::Matrix4 *_trans*)

Adds a key frame at a specific time.

Parameters

in	*_time*	the time of the key frame
in	*_trans*	the transformation

10.110.3.2 void gazebo::common::NodeAnimation::AddKeyFrame (const double *_time,* const math::Pose *_pose*)

Adds a key fram at a specific time.

Parameters

in	*_time*	the tiem of the key frame
in	*_pose*	the pose

10.110.3.3 math::Matrix4 gazebo::common::NodeAnimation::GetFrameAt (double *_time,* bool *_loop* = `true`) const

Returns a frame transformation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.

Parameters

in	*_time*	the time
in	*_loop*	when true, the time is divided by the duration (see GetLength)

10.110.3.4 unsigned int gazebo::common::NodeAnimation::GetFrameCount ( ) const

Returns the number of key frames.

Returns

the count

10.110.3.5 void gazebo::common::NodeAnimation::GetKeyFrame (const unsigned int *_i*, double & *_time*, **math::Matrix4** & *_trans*) const

Finds a key frame using the index.

Note the index of a key frame can change as frames are added.

Parameters

in	*_i*	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

10.110.3.6 std::pair<double, **math::Matrix4**> gazebo::common::NodeAnimation::GetKeyFrame (const unsigned int *_i*) const

Returns a key frame using the index.

Note the index of a key frame can change as frames are added.

Parameters

in	*_i*	the index

Returns

a pair that contains the time and transformation. **Time** (p. 840) is -1 if the index is out of bounds

10.110.3.7 double gazebo::common::NodeAnimation::GetLength ( ) const

Returns the duration of the animations.

Returns

the time of the last animation

10.110.3.8 std::string gazebo::common::NodeAnimation::GetName ( ) const

Returns the name.

Returns

the name

10.110.3.9 double gazebo::common::NodeAnimation::GetTimeAtX (const double _x) const

Returns the time where a transformation's translational value along the X axis is equal to _x.

When no transformation is found (within a tolerance of 1e-6), the time is interpolated.

Parameters

in	*_x*	the value along x. You must ensure that _x is within a valid range.

10.110.3.10 void gazebo::common::NodeAnimation::Scale (const double _scale)

Scales each transformation in the key frames.

This only affects the translational values.

Parameters

in	*_scale*	the scaling factor

10.110.3.11 void gazebo::common::NodeAnimation::SetName (const std::string & _name)

Changes the name of the animation.

Parameters

in	*the*	new name

10.110.4 Member Data Documentation

10.110.4.1 std::map<double, math::Matrix4> gazebo::common::NodeAnimation::keyFrames [protected]

the dictionary of key frames, indexed by time

10.110.4.2 double gazebo::common::NodeAnimation::length [protected]

the duration of the animations (time of last key frame)

10.110.4.3 std::string gazebo::common::NodeAnimation::name [protected]

the name of the animation

The documentation for this class was generated from the following file:

- **SkeletonAnimation.hh**

10.111 gazebo::common::NodeAssignment Struct Reference

Vertex to node weighted assignement for skeleton animation visualization.

`#include <Mesh.hh>`

Public Attributes

- unsigned int **nodeIndex**

  node (or bone) index
- unsigned int **vertexIndex**

  index of the vertex
- float **weight**

  the weight (between 0 and 1)

10.111.1 Detailed Description

Vertex to node weighted assignement for skeleton animation visualization.

10.111.2 Member Data Documentation

10.111.2.1 unsigned int gazebo::common::NodeAssignment::nodeIndex

node (or bone) index

10.111.2.2 unsigned int gazebo::common::NodeAssignment::vertexIndex

index of the vertex

10.111.2.3 float gazebo::common::NodeAssignment::weight

the weight (between 0 and 1)

The documentation for this struct was generated from the following file:

- **Mesh.hh**

10.112 gazebo::common::NodeTransform Class Reference

A transformation node.

`#include <Skeleton.hh>`

Public Types

- enum **TransformType** { **TRANSLATE**, **ROTATE**, **SCALE**, **MATRIX** }

  Enumeration of the transform types.

Public Member Functions

- **NodeTransform** (**TransformType** _type=**MATRIX**)
  Constructor.
- **NodeTransform** (**math::Matrix4** _mat, std::string _sid="_default_", TransformType _type=**MATRIX**)
  Constructor.
- ∼**NodeTransform** ()
  Destructor. It does nothing.
- **math::Matrix4 Get** ()
  Returns the transformation matrix.
- std::string **GetSID** ()
  Returns thr SID.
- **TransformType GetType** ()
  Returns the transformation type.
- **math::Matrix4 operator()** ()
  Matrix cast operator.
- **math::Matrix4 operator**∗ (**NodeTransform** _t)
  Node transform multiplication operator.
- **math::Matrix4 operator**∗ (**math::Matrix4** _m)
  Matrix multiplication operator.
- void **PrintSource** ()
  Prints the transform matrix to std::err stream.
- void **RecalculateMatrix** ()
  Sets the transform matrix from the source according to the type.
- void **Set** (**math::Matrix4** _mat)
  Assign a transformation.
- void **SetComponent** (unsigned int _idx, double _value)
  Set a transformation matrix component value.
- void **SetSID** (std::string _sid)
  Set the SID.
- void **SetSourceValues** (**math::Matrix4** _mat)
  Set source data values _ param[in] _mat the values.
- void **SetSourceValues** (**math::Vector3** _vec)
  Set source data values.
- void **SetSourceValues** (**math::Vector3** _axis, double _angle)
  Sets source matrix values from roation.
- void **SetType** (**TransformType** _type)
  Set transform type.

Protected Attributes

- std::string **sid**
  the sid
- std::vector< double > **source**
  source data values (can be a matrix, a position or rotation)
- **math::Matrix4 transform**
  transform
- **TransformType type**
  transform type

10.112.1 Detailed Description

A transformation node.

10.112.2 Member Enumeration Documentation

10.112.2.1 enum gazebo::common::NodeTransform::TransformType

Enumeration of the transform types.

Enumerator:

TRANSLATE

ROTATE

SCALE

MATRIX

10.112.3 Constructor & Destructor Documentation

10.112.3.1 gazebo::common::NodeTransform::NodeTransform (TransformType *_type* = MATRIX)

Constructor.

Parameters

in	*_type*	the type of transform

10.112.3.2 gazebo::common::NodeTransform::NodeTransform (math::Matrix4 *_mat*, std::string *_sid* = "_default_", TransformType *_type* = MATRIX)

Constructor.

Parameters

in	*_mat*	the matrix
in	*_sid*	identifier
in	*_type*	the type of transform

10.112.3.3 gazebo::common::NodeTransform::~NodeTransform ( )

Destructor. It does nothing.

10.112.4 Member Function Documentation

10.112.4.1 math::Matrix4 gazebo::common::NodeTransform::Get ( )

Returns the transformation matrix.

Returns

the matrix

10.112.4.2 std::string gazebo::common::NodeTransform::GetSID ( )

Returns thr SID.

Returns

the SID

10.112.4.3 TransformType gazebo::common::NodeTransform::GetType ( )

Returns the transformation type.

Returns

the type

10.112.4.4 math::Matrix4 gazebo::common::NodeTransform::operator() ( )

Matrix cast operator.

Returns

the transform

10.112.4.5 math::Matrix4 gazebo::common::NodeTransform::operator* (NodeTransform _t)

Node transform multiplication operator.

Parameters

in	_t	a transform

Returns

transform matrix multiplied by _t's transform

10.112.4.6 math::Matrix4 gazebo::common::NodeTransform::operator* (math::Matrix4 _m)

Matrix multiplication operator.

Parameters

in	_m	a matrix

Returns

transform matrix multiplied by _m

10.112.4.7 void gazebo::common::NodeTransform::PrintSource ( )

Prints the transform matrix to std::err stream.

10.112.4.8 void gazebo::common::NodeTransform::RecalculateMatrix ( )

Sets the transform matrix from the source according to the type.

10.112.4.9 void gazebo::common::NodeTransform::Set (math::Matrix4 *_mat*)

Assign a transformation.

Parameters

in	*_mat*	the transform

10.112.4.10 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

10.112.4.11 void gazebo::common::NodeTransform::SetSID (std::string *_sid*)

Set the SID.

Parameters

in	*_sid*	the sid

10.112.4.12 void gazebo::common::NodeTransform::SetSourceValues (math::Matrix4 *_mat*)

Set source data values _ param[in] _mat the values.

10.112.4.13 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 *_vec*)

Set source data values.

10.112.4.14 void gazebo::common::NodeTransform::SetSourceValues (math::Vector3 _axis, double _angle)

Sets source matrix values from roation.

Parameters

in	*_axis*	of rotation
in	*_angle*	of rotation

10.112.4.15 void gazebo::common::NodeTransform::SetType (TransformType _type)

Set transform type.

Parameters

in	*_type*	the type

10.112.5 Member Data Documentation

10.112.5.1 std::string gazebo::common::NodeTransform::sid [protected]

the sid

10.112.5.2 std::vector<double> gazebo::common::NodeTransform::source [protected]

source data values (can be a matrix, a position or rotation)

10.112.5.3 math::Matrix4 gazebo::common::NodeTransform::transform [protected]

transform

10.112.5.4 TransformType gazebo::common::NodeTransform::type [protected]

transform type

The documentation for this class was generated from the following file:

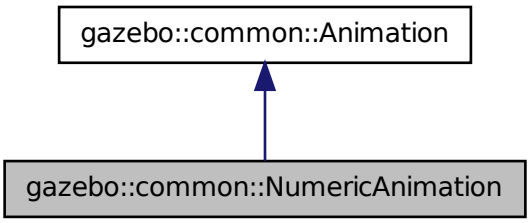
- **Skeleton.hh**

10.113 gazebo::common::NumericAnimation Class Reference

A numeric animation.

`#include <Animation.hh>`

Inheritance diagram for gazebo::common::NumericAnimation:



Public Member Functions

- **NumericAnimation** (const std::string &_name, double _length, bool _loop)

  Constructor.
- virtual ∼**NumericAnimation** ()

  Destructor.
- **NumericKeyFrame** ∗ **CreateKeyFrame** (double _time)

  Create a numeric keyframe at the given time.
- void **GetInterpolatedKeyFrame** (**NumericKeyFrame** &_kf) const

  Get a keyframe using the animation's current time.

Additional Inherited Members

10.113.1 Detailed Description

A numeric animation.

10.113.2 Constructor & Destructor Documentation

10.113.2.1 gazebo::common::NumericAnimation::NumericAnimation (const std::string & *_name,* double *_length,* bool *_loop*)

Constructor.

Parameters

in	*_name*	String name of the animation. This should be unique.
in	*_length*	Length of the animation in seconds
in	*_loop*	True == loop the animation

10.113.2.2 virtual gazebo::common::NumericAnimation::∼NumericAnimation ( ) [virtual]

Destructor.

10.113.3 Member Function Documentation

10.113.3.1 NumericKeyFrame* gazebo::common::NumericAnimation::CreateKeyFrame (double *_time*)

Create a numeric keyframe at the given time.

Parameters

in	*_time*	**Time** (p. 840) at which to create the keyframe

Returns

Pointer to the new keyframe

10.113.3.2 void gazebo::common::NumericAnimation::GetInterpolatedKeyFrame (NumericKeyFrame & *_kf*) const

Get a keyframe using the animation's current time.

Parameters

out	*_kf*	**NumericKeyFrame** (p. 571) reference to hold the interpolated result

The documentation for this class was generated from the following file:

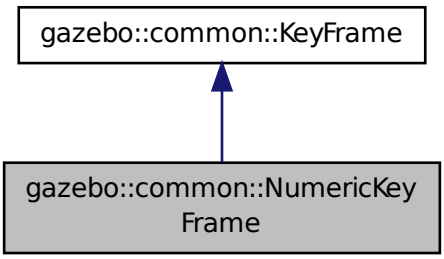
- **Animation.hh**

10.114 gazebo::common::NumericKeyFrame Class Reference

A keyframe for a **NumericAnimation** (p. 569).

`#include <KeyFrame.hh>`

Inheritance diagram for gazebo::common::NumericKeyFrame:



Public Member Functions

- **NumericKeyFrame** (double _time)

  Constructor.
- virtual ~**NumericKeyFrame** ()

  Destructor.
- const double & **GetValue** () const

  Get the value of the keyframe.
- void **SetValue** (const double &_value)

  Set the value of the keyframe.

Protected Attributes

- double **value**

  numeric value

10.114.1 Detailed Description

A keyframe for a **NumericAnimation** (p. 569).

10.114.2 Constructor & Destructor Documentation

10.114.2.1 gazebo::common::NumericKeyFrame::NumericKeyFrame (double *_time*)

Constructor.

Parameters

in	***Time*** *(p. 840)*	of the keyframe

10.114.2.2 virtual gazebo::common::NumericKeyFrame::~NumericKeyFrame ( ) [virtual]

Destructor.

10.114.3 Member Function Documentation

10.114.3.1 const double& gazebo::common::NumericKeyFrame::GetValue ( ) const

Get the value of the keyframe.

Returns

the value of the keyframe

10.114.3.2 void gazebo::common::NumericKeyFrame::SetValue (const double & *_value*)

Set the value of the keyframe.

Parameters

in	*_value*	The new value

10.114.4 Member Data Documentation

10.114.4.1 double gazebo::common::NumericKeyFrame::value [protected]

numeric value

The documentation for this class was generated from the following file:

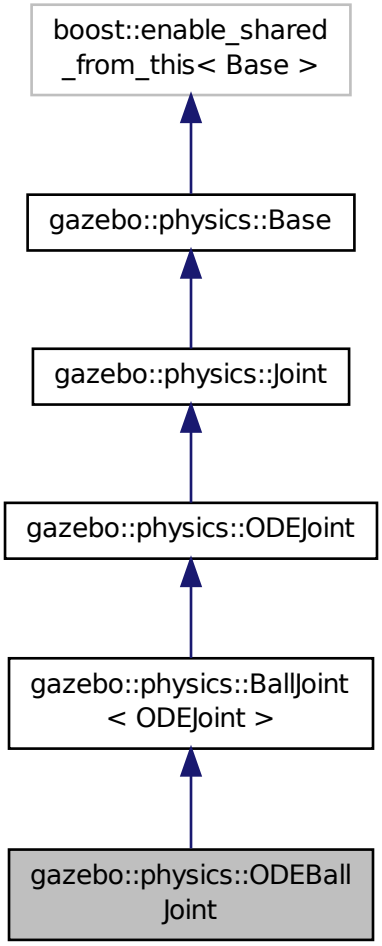
- **KeyFrame.hh**

10.115 gazebo::physics::ODEBallJoint Class Reference

An **ODEBallJoint** (p. 573).

`#include <ODEBallJoint.hh>`

Inheritance diagram for gazebo::physics::ODEBallJoint:



Public Member Functions

- **ODEBallJoint** (dWorldID worldId, **BasePtr** _parent)

  Constructor.
- virtual ~**ODEBallJoint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (int index) const

  Get joint's anchor point.
- virtual **math::Angle GetAngleImpl** (int) const

  Get the angle of rotation of an axis(index)
- virtual **math::Vector3 GetGlobalAxis** (int) const

  Get the axis of rotation.

- virtual double **GetMaxForce** (int)

  Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int) const

  Get the rotation rate of an axis(index)
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)

  Set joint's anchor point.
- virtual void **SetDamping** (int _index, double _damping)

  Set joint damping, not yet implemented.
- virtual void **SetMaxForce** (int, double)

  Set the max allowed force of an axis(index).
- virtual void **SetVelocity** (int, double)

  Set the velocity of an axis(index).

Additional Inherited Members

10.115.1 Detailed Description

An **ODEBallJoint** (p. 573).

10.115.2 Constructor & Destructor Documentation

10.115.2.1 gazebo::physics::ODEBallJoint::ODEBallJoint (dWorldID *worldId,* BasePtr *_parent*)

Constructor.

10.115.2.2 virtual gazebo::physics::ODEBallJoint::~ODEBallJoint ( ) [virtual]

Destructor.

10.115.3 Member Function Documentation

10.115.3.1 virtual math::Vector3 gazebo::physics::ODEBallJoint::GetAnchor (int *index*) const [virtual]

Get joint's anchor point.

Implements **gazebo::physics::Joint** (p. 428).

10.115.3.2 virtual math::Angle gazebo::physics::ODEBallJoint::GetAngleImpl (int) const [inline], [virtual]

Get the angle of rotation of an axis(index)

Implements **gazebo::physics::Joint** (p. 429).

10.115.3.3 virtual math::Vector3 gazebo::physics::ODEBallJoint::GetGlobalAxis (int) const [inline], [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.115.3.4 virtual double gazebo::physics::ODEBallJoint::GetMaxForce (int) [inline], [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.115.3.5 virtual double gazebo::physics::ODEBallJoint::GetVelocity (int) const [inline], [virtual]

Get the rotation rate of an axis(index)

Implements **gazebo::physics::Joint** (p. 433).

10.115.3.6 virtual void gazebo::physics::ODEBallJoint::SetAnchor (int *index*, const **math::Vector3** & *anchor*) [virtual]

Set joint's anchor point.

Implements **gazebo::physics::Joint** (p. 434).

10.115.3.7 virtual void gazebo::physics::ODEBallJoint::SetDamping (int *_index*, double *_damping*) [virtual]

Set joint damping, not yet implemented.

Implements **gazebo::physics::Joint** (p. 435).

10.115.3.8 virtual void gazebo::physics::ODEBallJoint::SetMaxForce (int , double) [inline], [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.115.3.9 virtual void gazebo::physics::ODEBallJoint::SetVelocity (int , double) [inline], [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

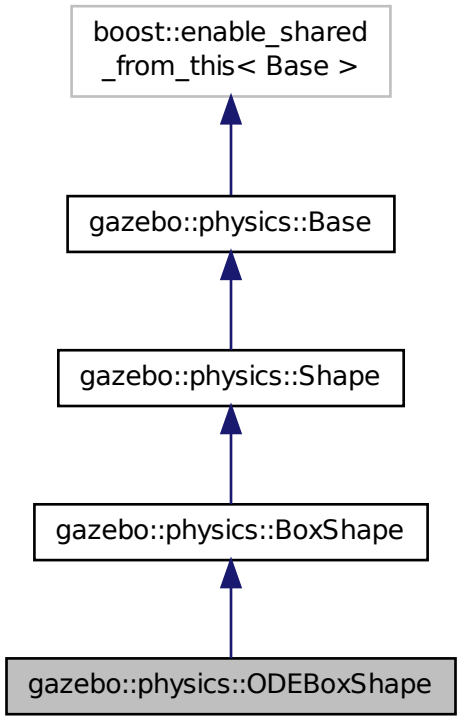
- **ODEBallJoint.hh**

10.116 gazebo::physics::ODEBoxShape Class Reference

ODE Box shape.

`#include <ODEBoxShape.hh>`

Inheritance diagram for gazebo::physics::ODEBoxShape:



Public Member Functions

- **ODEBoxShape** (**ODECollisionPtr** _parent)
- virtual ~**ODEBoxShape** ()
- virtual void **SetSize** (const **math::Vector3** &size)

  Set the size of the box.

Additional Inherited Members

10.116.1 Detailed Description

ODE Box shape.

10.116.2 Constructor & Destructor Documentation

10.116.2.1 gazebo::physics::ODEBoxShape::ODEBoxShape (**ODECollisionPtr** *_parent*) [inline]

10.116.2.2 virtual gazebo::physics::ODEBoxShape::~ODEBoxShape ( ) [inline], [virtual]

10.116.3 Member Function Documentation

10.116.3.1 virtual void gazebo::physics::ODEBoxShape::SetSize (const math::Vector3 & _size) [inline], [virtual]

Set the size of the box.

Parameters

in	*_size*	Size of each side of the box.

Reimplemented from **gazebo::physics::BoxShape** (p. 165).

References gazebo::physics::Shape::collisionParent, gazebo::physics::ODECollision::GetCollisionId(), NULL, gazebo::physics::ODECollision::SetCollision(), gazebo::physics::BoxShape::SetSize(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

The documentation for this class was generated from the following file:

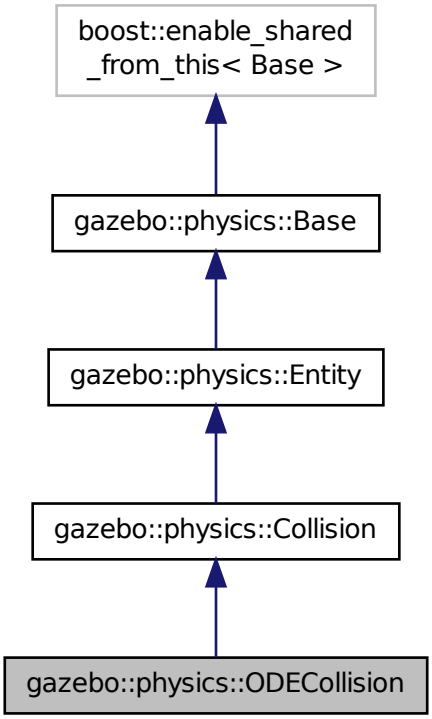
- **ODEBoxShape.hh**

10.117 gazebo::physics::ODECollision Class Reference

Base (p. 145) class for all ODE collisions.

`#include <ODECollision.hh>`

Inheritance diagram for gazebo::physics::ODECollision:



Public Member Functions

- **ODECollision** (**LinkPtr link**)

  Constructor.
- virtual ~**ODECollision** ()

  Destructor.
- void **Fini** ()

  Finalize the collision.
- virtual **math::Box GetBoundingBox** () const

  Get the bounding box, defined by the physics engine.
- int **GetCollisionClass** () const

  Get the ODE collision class.
- dGeomID **GetCollisionId** () const

  Return the collision id.
- dSpaceID **GetSpaceId** () const

  Get the collision's space ID.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the collision.

- virtual void **OnPoseChange** ()

  This function is called when the entity's (or one of its parents) pose of the parent has changed.
- virtual void **SetCategoryBits** (unsigned int bits)

  Set the category bits, used during collision detection.
- virtual void **SetCollideBits** (unsigned int bits)

  Set the collide bits, used during collision detection.
- void **SetCollision** (dGeomID **collisionId**, bool **placeable**)

  Set the encapsulated geometry object.
- void **SetSpaceId** (dSpaceID spaceid)

  Set the collision's space ID.

Protected Attributes

- dGeomID **collisionId**

  ID for the sub-collision.
- dSpaceID **spaceId**

Additional Inherited Members

10.117.1 Detailed Description

Base (p. 145) class for all ODE collisions.

10.117.2 Constructor & Destructor Documentation

10.117.2.1 gazebo::physics::ODECollision::ODECollision (LinkPtr *link*)

Constructor.

10.117.2.2 virtual gazebo::physics::ODECollision::~ODECollision ( ) [virtual]

Destructor.

10.117.3 Member Function Documentation

10.117.3.1 void gazebo::physics::ODECollision::Fini ( ) [virtual]

Finalize the collision.

Reimplemented from **gazebo::physics::Collision** (p. 266).

10.117.3.2 virtual math::Box gazebo::physics::ODECollision::GetBoundingBox ( ) const [virtual]

Get the bounding box, defined by the physics engine.

Implements **gazebo::physics::Collision** (p. 266).

10.117.3.3 int gazebo::physics::ODECollision::GetCollisionClass ( ) const

Get the ODE collision class.

10.117.3.4 dGeomID gazebo::physics::ODECollision::GetCollisionId ( ) const

Return the collision id.

Returns

The collision id

Referenced by gazebo::physics::ODEPlaneShape::CreatePlane(), gazebo::physics::ODEPlaneShape::SetAltitude(), gazebo::physics::ODESphereShape::SetRadius(), gazebo::physics::ODECylinderShape::SetSize(), and gazebo::physics::ODEBoxShape::SetSize().

10.117.3.5 dSpaceID gazebo::physics::ODECollision::GetSpaceId ( ) const

Get the collision's space ID.

Referenced by gazebo::physics::ODEPlaneShape::CreatePlane().

10.117.3.6 virtual void gazebo::physics::ODECollision::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the collision.

Reimplemented from **gazebo::physics::Collision** (p. 270).

10.117.3.7 virtual void gazebo::physics::ODECollision::OnPoseChange ( ) [virtual]

This function is called when the entity's (or one of its parents) pose of the parent has changed.

Implements **gazebo::physics::Entity** (p. 346).

10.117.3.8 virtual void gazebo::physics::ODECollision::SetCategoryBits (unsigned int *bits*) [virtual]

Set the category bits, used during collision detection.

Parameters

bits	The bits

Implements **gazebo::physics::Collision** (p. 270).

10.117.3.9 virtual void gazebo::physics::ODECollision::SetCollideBits (unsigned int *bits*) [virtual]

Set the collide bits, used during collision detection.

Parameters

bits	The bits

Implements **gazebo::physics::Collision** (p. 270).

10.117.3.10 void gazebo::physics::ODECollision::SetCollision (dGeomID *collisionId,* bool *placeable*)

Set the encapsulated geometry object.

Referenced by gazebo::physics::ODEPlaneShape::CreatePlane(), gazebo::physics::ODESphereShape::SetRadius(), gazebo::physics::ODECylinderShape::SetSize(), and gazebo::physics::ODEBoxShape::SetSize().

10.117.3.11 void gazebo::physics::ODECollision::SetSpaceId (dSpaceID *spaceid*)

Set the collision's space ID.

10.117.4 Member Data Documentation

10.117.4.1 dGeomID gazebo::physics::ODECollision::collisionId [protected]

ID for the sub-collision.

10.117.4.2 dSpaceID gazebo::physics::ODECollision::spaceId [protected]

The documentation for this class was generated from the following file:

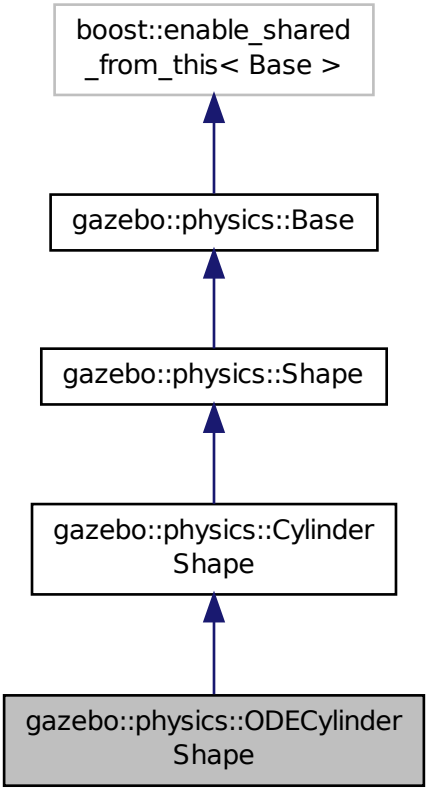
- **ODECollision.hh**

10.118 gazebo::physics::ODECylinderShape Class Reference

ODE cylinder shape.

`#include <ODECylinderShape.hh>`

Inheritance diagram for gazebo::physics::ODECylinderShape:



Public Member Functions

- **ODECylinderShape** (**CollisionPtr** _parent)

  constructor
- virtual ~**ODECylinderShape** ()

  destructor
- void **SetSize** (double _radius, double _length)

  Set radius and length of a cylinder.

Additional Inherited Members

10.118.1 Detailed Description

ODE cylinder shape.

10.118.2 Constructor & Destructor Documentation

10.118.2.1 gazebo::physics::ODECylinderShape::ODECylinderShape (CollisionPtr *_parent*) [inline]

constructor

10.118.2.2 virtual gazebo::physics::ODECylinderShape::~ODECylinderShape ( ) [inline], [virtual]

destructor

10.118.3 Member Function Documentation

10.118.3.1 void gazebo::physics::ODECylinderShape::SetSize (double *_radius,* double *_length*) [inline], [virtual]

Set radius and length of a cylinder.

Reimplemented from **gazebo::physics::CylinderShape** (p. 311).

References gazebo::physics::Shape::collisionParent, gazebo::physics::ODECollision::GetCollisionId(), NULL, gazebo::physics::ODECollision::SetCollision(), and gazebo::physics::CylinderShape::SetSize().

The documentation for this class was generated from the following file:

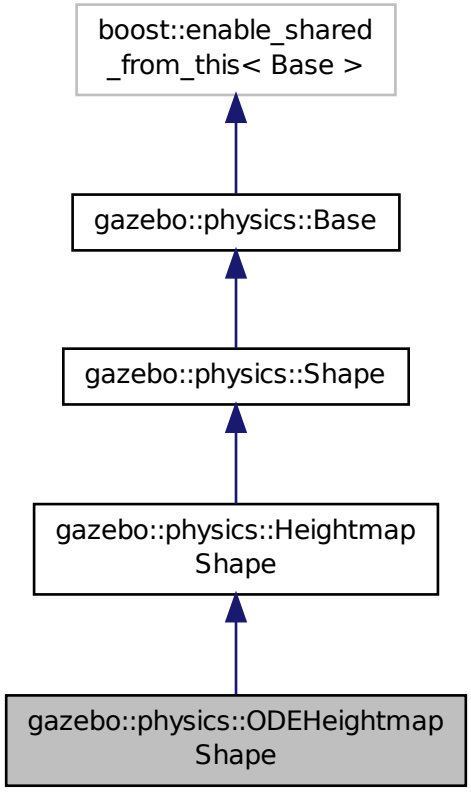
- **ODECylinderShape.hh**

10.119 gazebo::physics::ODEHeightmapShape Class Reference

ODE Height map collision.

`#include <ODEHeightmapShape.hh>`

Inheritance diagram for gazebo::physics::ODEHeightmapShape:



Public Member Functions

- **ODEHeightmapShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**ODEHeightmapShape** ()

  Destructor.
- virtual void **Init** ()

  Load the heightmap.

Additional Inherited Members

10.119.1 Detailed Description

ODE Height map collision.

10.119.2 Constructor & Destructor Documentation

10.119.2.1 gazebo::physics::ODEHeightmapShape::ODEHeightmapShape (CollisionPtr *_parent*)

Constructor.

10.119.2.2 virtual gazebo::physics::ODEHeightmapShape::~ODEHeightmapShape ( ) [virtual]

Destructor.

10.119.3 Member Function Documentation

10.119.3.1 virtual void gazebo::physics::ODEHeightmapShape::Init ( ) [virtual]

Load the heightmap.

Reimplemented from **gazebo::physics::HeightmapShape** (p. 403).

The documentation for this class was generated from the following file:

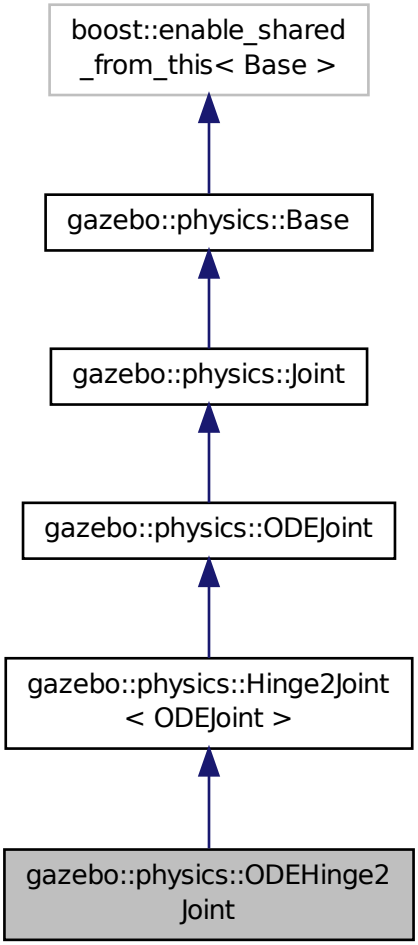
- **ODEHeightmapShape.hh**

10.120 gazebo::physics::ODEHinge2Joint Class Reference

A two axis hinge joint.

`#include <ODEHinge2Joint.hh>`

Inheritance diagram for gazebo::physics::ODEHinge2Joint:



Public Member Functions

- **ODEHinge2Joint** (dWorldID worldId, **BasePtr** _parent)

  Constructor.
- virtual ~**ODEHinge2Joint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (int index) const

  Get anchor point.
- virtual **math::Angle GetAngleImpl** (int index) const

  Get angle of rotation about first axis.
- virtual **math::Vector3 GetGlobalAxis** (int index) const

  Get first axis of rotation.

- virtual double **GetMaxForce** (int index)

  Get the max allowed force of an axis(index).
- virtual double **GetParam** (int parameter) const

  Get the specified parameter.
- virtual double **GetVelocity** (int index) const

  Get rate of rotation about first axis.
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)

  Set the anchor point.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)

  Set the first axis of rotation.
- virtual void **SetDamping** (int _index, double _damping)

  Set joint damping, not yet implemented.
- virtual void **SetForce** (int index, double torque)

  Set the torque.
- virtual void **SetMaxForce** (int index, double t)

  Set the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)

  Set _parameter with _value.
- virtual void **SetVelocity** (int index, double angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load the **ODEHinge2Joint** (p. 586).*

Additional Inherited Members

10.120.1 Detailed Description

A two axis hinge joint.

10.120.2 Constructor & Destructor Documentation

10.120.2.1 gazebo::physics::ODEHinge2Joint::ODEHinge2Joint (dWorldID *worldId,* BasePtr *_parent*)

Constructor.

10.120.2.2 virtual gazebo::physics::ODEHinge2Joint::~ODEHinge2Joint ( ) [virtual]

Destructor.

10.120.3 Member Function Documentation

10.120.3.1 virtual math::Vector3 gazebo::physics::ODEHinge2Joint::GetAnchor (int *index*) const [virtual]

Get anchor point.

Implements **gazebo::physics::Joint** (p. 428).

10.120.3.2 virtual math::Angle gazebo::physics::ODEHinge2Joint::GetAngleImpl (int *index*) const [virtual]

Get angle of rotation about first axis.

Implements **gazebo::physics::Joint** (p. 429).

10.120.3.3 virtual math::Vector3 gazebo::physics::ODEHinge2Joint::GetGlobalAxis (int *index*) const [virtual]

Get first axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.120.3.4 virtual double gazebo::physics::ODEHinge2Joint::GetMaxForce (int *index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.120.3.5 virtual double gazebo::physics::ODEHinge2Joint::GetParam (int *parameter*) const [virtual]

Get the specified parameter.

Reimplemented from **gazebo::physics::ODEJoint** (p. 598).

10.120.3.6 virtual double gazebo::physics::ODEHinge2Joint::GetVelocity (int *index*) const [virtual]

Get rate of rotation about first axis.

Implements **gazebo::physics::Joint** (p. 433).

10.120.3.7 virtual void gazebo::physics::ODEHinge2Joint::Load (sdf::ElementPtr *_sdf*) [protected], [virtual]

Load the **ODEHinge2Joint** (p. 586).

Reimplemented from **gazebo::physics::Hinge2Joint< ODEJoint >** (p. 405).

10.120.3.8 virtual void gazebo::physics::ODEHinge2Joint::SetAnchor (int *index,* const math::Vector3 & *anchor*) [virtual]

Set the anchor point.

Implements **gazebo::physics::Joint** (p. 434).

10.120.3.9 virtual void gazebo::physics::ODEHinge2Joint::SetAxis (int *index,* const math::Vector3 & *axis*) [virtual]

Set the first axis of rotation.

Implements **gazebo::physics::Joint** (p. 435).

10.120.3.10 virtual void gazebo::physics::ODEHinge2Joint::SetDamping (int *_index,* double *_damping*) [virtual]

Set joint damping, not yet implemented.

Implements **gazebo::physics::Joint** (p. 435).

10.120.3.11 virtual void gazebo::physics::ODEHinge2Joint::SetForce (int *index,* double *torque*) [virtual]

Set the torque.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.120.3.12 virtual void gazebo::physics::ODEHinge2Joint::SetMaxForce (int *index,* double *t*) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.120.3.13 virtual void gazebo::physics::ODEHinge2Joint::SetParam (int *parameter,* double *value*) [virtual]

Set _parameter with _value.

Reimplemented from **gazebo::physics::ODEJoint** (p. 599).

10.120.3.14 virtual void gazebo::physics::ODEHinge2Joint::SetVelocity (int *index,* double *angle*) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

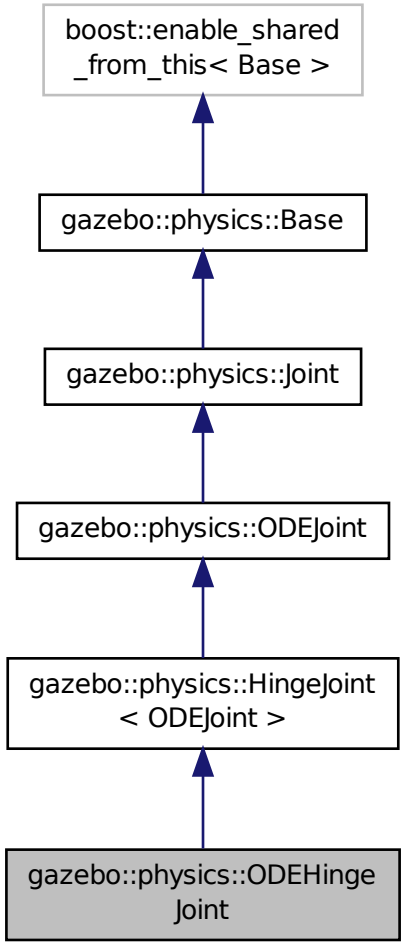
- **ODEHinge2Joint.hh**

10.121 gazebo::physics::ODEHingeJoint Class Reference

A single axis hinge joint.

`#include <ODEHingeJoint.hh>`

Inheritance diagram for gazebo::physics::ODEHingeJoint:



Public Member Functions

- **ODEHingeJoint** (dWorldID worldId, **BasePtr** _parent)

  Constructor.
- virtual ~**ODEHingeJoint** ()

  Destructor.
- void **ApplyDamping** ()

  callback to apply damping force to joint
- virtual **math::Vector3 GetAnchor** (int index) const

  Get the anchor point.
- virtual **math::Angle GetAngleImpl** (int index) const

  Get the angle of rotation.

- virtual **math::Vector3 GetGlobalAxis** (int index) const

  Get the axis of rotation.
- virtual double **GetMaxForce** (int index)

  Get the max allowed force of an axis(index).
- virtual double **GetParam** (int parameter) const

  Get the specified parameter.
- virtual double **GetVelocity** (int index) const

  Get the rotation rate of an axis(index)
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)

  Set the anchor point.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)

  Set the axis of rotation.
- virtual void **SetDamping** (int _index, double _damping)

  Set the joint damping.
- virtual void **SetForce** (int index, double torque)

  Set the torque of a joint.
- virtual void **SetMaxForce** (int index, double t)

  Set the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)

  Set the parameter to value.
- virtual void **SetVelocity** (int index, double angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load joint.

Additional Inherited Members

10.121.1 Detailed Description

A single axis hinge joint.

10.121.2 Constructor & Destructor Documentation

10.121.2.1 gazebo::physics::ODEHingeJoint::ODEHingeJoint (dWorldID *worldId,* BasePtr *_parent*)

Constructor.

10.121.2.2 virtual gazebo::physics::ODEHingeJoint::~ODEHingeJoint ( ) [virtual]

Destructor.

10.121.3 Member Function Documentation

10.121.3.1 void gazebo::physics::ODEHingeJoint::ApplyDamping ( )

callback to apply damping force to joint

10.121.3.2 virtual math::Vector3 gazebo::physics::ODEHingeJoint::GetAnchor (int *index*) const [virtual]

Get the anchor point.

Implements **gazebo::physics::Joint** (p. 428).

10.121.3.3 virtual math::Angle gazebo::physics::ODEHingeJoint::GetAngleImpl (int *index*) const [virtual]

Get the angle of rotation.

Implements **gazebo::physics::Joint** (p. 429).

10.121.3.4 virtual math::Vector3 gazebo::physics::ODEHingeJoint::GetGlobalAxis (int *index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.121.3.5 virtual double gazebo::physics::ODEHingeJoint::GetMaxForce (int *index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.121.3.6 virtual double gazebo::physics::ODEHingeJoint::GetParam (int *parameter*) const [virtual]

Get the specified parameter.

Reimplemented from **gazebo::physics::ODEJoint** (p. 598).

10.121.3.7 virtual double gazebo::physics::ODEHingeJoint::GetVelocity (int *index*) const [virtual]

Get the rotation rate of an axis(index)

Implements **gazebo::physics::Joint** (p. 433).

10.121.3.8 virtual void gazebo::physics::ODEHingeJoint::Load (sdf::ElementPtr *_sdf*) [protected], [virtual]

Load joint.

Reimplemented from **gazebo::physics::HingeJoint**< **ODEJoint** > (p. 407).

10.121.3.9 virtual void gazebo::physics::ODEHingeJoint::SetAnchor (int *index*, const math::Vector3 & *anchor*) [virtual]

Set the anchor point.

Implements **gazebo::physics::Joint** (p. 434).

10.121.3.10 virtual void gazebo::physics::ODEHingeJoint::SetAxis (int *index,* const math::Vector3 & *axis*) [virtual]

Set the axis of rotation.

Implements **gazebo::physics::Joint** (p. 435).

10.121.3.11 virtual void gazebo::physics::ODEHingeJoint::SetDamping (int *_index,* double *_damping*) [virtual]

Set the joint damping.

Implements **gazebo::physics::Joint** (p. 435).

10.121.3.12 virtual void gazebo::physics::ODEHingeJoint::SetForce (int *index,* double *torque*) [virtual]

Set the torque of a joint.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.121.3.13 virtual void gazebo::physics::ODEHingeJoint::SetMaxForce (int *index,* double *t*) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.121.3.14 virtual void gazebo::physics::ODEHingeJoint::SetParam (int *parameter,* double *value*) [virtual]

Set the parameter to value.

Reimplemented from **gazebo::physics::ODEJoint** (p. 599).

10.121.3.15 virtual void gazebo::physics::ODEHingeJoint::SetVelocity (int *index,* double *angle*) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

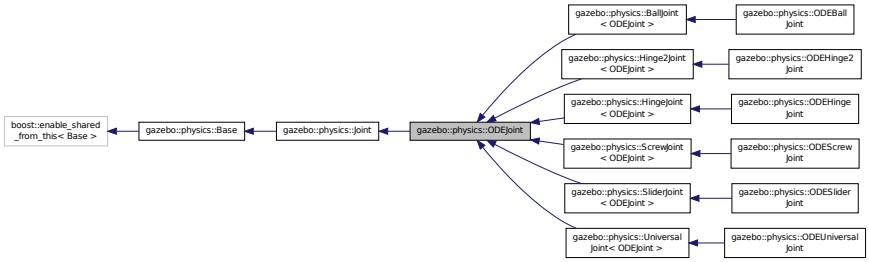
- **ODEHingeJoint.hh**

10.122 gazebo::physics::ODEJoint Class Reference

ODE joint interface.

`#include <ODEJoint.hh>`

Inheritance diagram for gazebo::physics::ODEJoint:



Public Member Functions

- **ODEJoint** (**BasePtr** _parent)

  Constructor.
- virtual ∼**ODEJoint** ()

  Destructor.
- virtual bool **AreConnected** (**LinkPtr** _one, **LinkPtr** _two) const

  Determines 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 bodies.
- double **GetCFM** ()

  Get the ERP of this joint.
- double **GetERP** ()

  Get the ERP of this joint.
- dJointFeedback ∗ **GetFeedback** ()

  Get the feedback data structure for this joint, if set.
- virtual **math::Angle GetHighStop** (int index)

  Get the high stop of an axis(index).
- virtual **LinkPtr GetJointLink** (int _index) const

  Get the link to which the joint is attached according the _index.
- virtual **math::Vector3 GetLinkForce** (unsigned int index) const

  Get the force the joint applies to the first link.
- virtual **math::Vector3 GetLinkTorque** (unsigned int index) const

  Get the torque the joint applies to the first link.
- virtual **math::Angle GetLowStop** (int index)

  Get the low stop of an axis(index).
- virtual double **GetParam** (int _parameter) const

  The default function does nothing.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load **ODEJoint** (p. 594).*
- virtual void **Reset** ()

  Reset the joint.

- virtual void **SetAttribute** (**Attribute**, int index, double value)

  Set a parameter for the joint.
- void **SetCFM** (double newCFM)

  Set the CFM of this joint.
- void **SetERP** (double newERP)

  Set the ERP of this joint.
- virtual void **SetHighStop** (int _index, const **math::Angle** &_angle)

  Set the high stop of an axis(index).
- virtual void **SetLowStop** (int _index, const **math::Angle** &_angle)

  Set the low stop of an axis(index).
- virtual void **SetParam** (int _parameter, double _value)

  By default this does nothing.

Protected Attributes

- dJointID **jointId**

  This is our id.

Additional Inherited Members

10.122.1 Detailed Description

ODE joint interface.

10.122.2 Constructor & Destructor Documentation

10.122.2.1 gazebo::physics::ODEJoint::ODEJoint (BasePtr *_parent*)

Constructor.

10.122.2.2 virtual gazebo::physics::ODEJoint::~ODEJoint ( ) [virtual]

Destructor.

10.122.3 Member Function Documentation

10.122.3.1 virtual bool gazebo::physics::ODEJoint::AreConnected (LinkPtr *_one,* LinkPtr *_two*) const [virtual]

Determines of the two bodies are connected by a joint.

Implements **gazebo::physics::Joint** (p. 427).

10.122.3.2 virtual void gazebo::physics::ODEJoint::Attach (LinkPtr *parent,* LinkPtr *child*) [virtual]

Attach the two bodies with this joint.

Reimplemented from **gazebo::physics::Joint** (p. 427).

10.122.3.3 virtual void gazebo::physics::ODEJoint::Detach ( ) [virtual]

Detach this joint from all bodies.

Reimplemented from **gazebo::physics::Joint** (p. 428).

10.122.3.4 double gazebo::physics::ODEJoint::GetCFM ( )

Get the ERP of this joint.

10.122.3.5 double gazebo::physics::ODEJoint::GetERP ( )

Get the ERP of this joint.

10.122.3.6 dJointFeedback* gazebo::physics::ODEJoint::GetFeedback ( )

Get the feedback data structure for this joint, if set.

10.122.3.7 virtual math::Angle gazebo::physics::ODEJoint::GetHighStop (int *index*) [virtual]

Get the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 430).

Reimplemented in **gazebo::physics::BallJoint< ODEJoint >** (p. 144).

10.122.3.8 virtual LinkPtr gazebo::physics::ODEJoint::GetJointLink (int *_index*) const [virtual]

Get the link to which the joint is attached according the _index.

Implements **gazebo::physics::Joint** (p. 431).

10.122.3.9 virtual math::Vector3 gazebo::physics::ODEJoint::GetLinkForce (unsigned int *index*) const [virtual]

Get the force the joint applies to the first link.

Parameters

index	The index of the link(0 or 1)

Implements **gazebo::physics::Joint** (p. 431).

10.122.3.10 virtual math::Vector3 gazebo::physics::ODEJoint::GetLinkTorque (unsigned int *index*) const [virtual]

Get the torque the joint applies to the first link.

Parameters

index	The index of the link(0 or 1)

Implements **gazebo::physics::Joint** (p. 431).

10.122.3.11 virtual math::Angle gazebo::physics::ODEJoint::GetLowStop (int *index*) [virtual]

Get the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

Reimplemented in **gazebo::physics::BallJoint< ODEJoint >** (p. 144).

10.122.3.12 virtual double gazebo::physics::ODEJoint::GetParam (int *_parameter*) const [virtual]

The default function does nothing.

This should be overriden in the child classes where appropriate

Reimplemented in **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEHinge2Joint** (p. 589), and **gazebo::physics::ODESliderJoint** (p. 628).

10.122.3.13 virtual void gazebo::physics::ODEJoint::Load (sdf::ElementPtr *_sdf*) [virtual]

Load **ODEJoint** (p. 594).

Reimplemented from **gazebo::physics::Joint** (p. 433).

Reimplemented in **gazebo::physics::BallJoint< ODEJoint >** (p. 144), **gazebo::physics::ScrewJoint< ODEJoint >** (p. 761), **gazebo::physics::ODEScrewJoint** (p. 624), **gazebo::physics::ODEHinge2Joint** (p. 589), **gazebo::physics::ODEHingeJoint** (p. 593), **gazebo::physics::SliderJoint< ODEJoint >** (p. 805), **gazebo::physics::Hinge2Joint< ODEJoint >** (p. 405), **gazebo::physics::HingeJoint< ODEJoint >** (p. 407), **gazebo::physics::UniversalJoint< ODEJoint >** (p. 870), and **gazebo::physics::ODESliderJoint** (p. 628).

10.122.3.14 virtual void gazebo::physics::ODEJoint::Reset ( ) [virtual]

Reset the joint.

Reimplemented from **gazebo::physics::Joint** (p. 434).

10.122.3.15 virtual void gazebo::physics::ODEJoint::SetAttribute (Attribute , int *index,* double *value*) [virtual]

Set a parameter for the joint.

Implements **gazebo::physics::Joint** (p. 435).

10.122.3.16 void gazebo::physics::ODEJoint::SetCFM (double *newCFM*)

Set the CFM of this joint.

10.122.3.17 void gazebo::physics::ODEJoint::SetERP (double *newERP*)

Set the ERP of this joint.

10.122.3.18 virtual void gazebo::physics::ODEJoint::SetHighStop (int *_index,* const math::Angle & *_angle*) [virtual]

Set the high stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.122.3.19 virtual void gazebo::physics::ODEJoint::SetLowStop (int _index, const math::Angle & _angle) [virtual]

Set the low stop of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.122.3.20 virtual void gazebo::physics::ODEJoint::SetParam (int _parameter, double _value) [virtual]

By default this does nothing.

It should be overridden in child classes where appropriate

Reimplemented in **gazebo::physics::ODEHingeJoint** (p. 594), **gazebo::physics::ODEScrewJoint** (p. 625), **gazebo::physics::ODEHinge2Joint** (p. 590), **gazebo::physics::ODESliderJoint** (p. 629), and **gazebo::physics::ODEUniversalJoint** (p. 637).

10.122.4 Member Data Documentation

10.122.4.1 dJointID gazebo::physics::ODEJoint::jointId [protected]

This is our id.

The documentation for this class was generated from the following file:

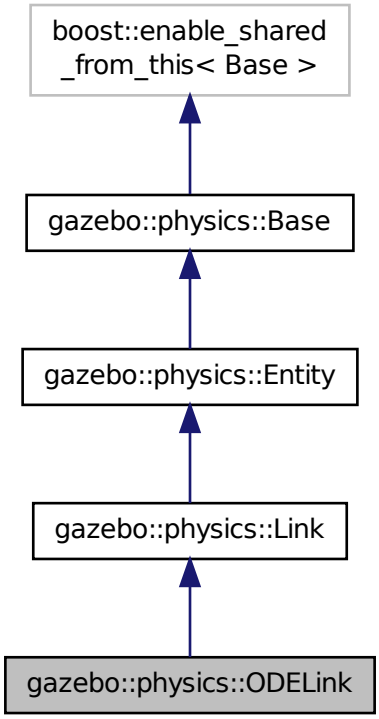
- **ODEJoint.hh**

10.123 gazebo::physics::ODELink Class Reference

ODE **Link** (p. 454) class.

`#include <ODELink.hh>`

Inheritance diagram for gazebo::physics::ODELink:



Public Member Functions

- **ODELink** (**EntityPtr** _parent)

  Constructor.
- virtual ~**ODELink** ()

  Destructor.
- virtual void **AddForce** (const **math::Vector3** &_force)

  Add a force to the body.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)

  Set the force applied to the body (add by Stefano)
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)

  Add a force to the body using a global position.
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)

  Add a force to the body.
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)

  Add a torque to the body relative frame.
- virtual void **AddTorque** (const **math::Vector3** &_torque)

  Add a torque to the body.

- virtual void **Fini** ()

  Finalize the link.
- virtual bool **GetEnabled** () const

  Get whether this link is enabled in the physics engine.
- virtual bool **GetGravityMode** ()

  Get the gravity mode.
- virtual bool **GetKinematic** () const

  Get whether this link is in the kinematic state.
- dBodyID **GetODEId** () const

  Return the ID of this link.
- dSpaceID **GetSpaceId** () const

  Get the link's space ID.
- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the link in the world frame.
- virtual **math::Vector3 GetWorldForce** () const

  Get the force applied to the link in the world frame.
- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the link in the world frame.
- virtual **math::Vector3 GetWorldTorque** () const

  Get the torque applied to the link in the world frame.
- virtual void **Init** ()

  Initialize the link.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the link based on SDF parameters.
- virtual void **OnPoseChange** ()

  Called when the pose of the entity (or one of its parents) has changed.
- virtual void **SetAngularDamping** (double damping)

  Set the angular damping factor.
- virtual void **SetAngularVel** (const **math::Vector3** &vel)

  Set the angular velocity of the link.
- virtual void **SetAutoDisable** (bool _disable)

  Allow the link to auto disable.
- virtual void **SetEnabled** (bool enable) const

  Set whether this link is enabled.
- virtual void **SetForce** (const **math::Vector3** &_force)

  Set the force applied to the link.
- virtual void **SetGravityMode** (bool mode)

  Set whether gravity affects this link.
- virtual void **SetKinematic** (const bool &state)

  Set whether this link is in the kinematic state.
- virtual void **SetLinearDamping** (double damping)

  Set the linear damping factor.
- virtual void **SetLinearVel** (const **math::Vector3** &vel)

  Set the linear velocity of the link.
- void **SetSelfCollide** (bool collide)

  Set whether this link will collide with others in the model.
- void **SetSpaceId** (dSpaceID spaceid)

Set the link's space ID.

- virtual void **SetTorque** (const **math::Vector3** &_torque)

  Set the torque applied to the link.

- virtual void **Update** ()

  Update the link.

- virtual void **UpdateMass** ()

  Update the mass matrix.

- virtual void **UpdateSurface** ()

  Update other parameters for ODE.

Static Public Member Functions

- static void **DisabledCallback** (dBodyID _id)

  callback when ODE determines a body is disabled

- static void **MoveCallback** (dBodyID id)

  when ODE updates dynamics bodies, this callback propagates the chagnes in pose back to Gazebo

Protected Attributes

- **math::Pose pose**

Additional Inherited Members

10.123.1 Detailed Description

ODE **Link** (p. 454) class.

10.123.2 Constructor & Destructor Documentation

10.123.2.1 gazebo::physics::ODELink::ODELink (EntityPtr *_parent*)

Constructor.

10.123.2.2 virtual gazebo::physics::ODELink::~ODELink ( ) [virtual]

Destructor.

10.123.3 Member Function Documentation

10.123.3.1 virtual void gazebo::physics::ODELink::AddForce (const math::Vector3 & *_force*) [virtual]

Add a force to the body.

Implements **gazebo::physics::Link** (p. 459).

10.123.3.2 virtual void gazebo::physics::ODELink::AddForceAtRelativePosition (const math::Vector3 & _force, const math::Vector3 & _relpos) [virtual]

Set the force applied to the body (add by Stefano)

Implements **gazebo::physics::Link** (p. 460).

10.123.3.3 virtual void gazebo::physics::ODELink::AddForceAtWorldPosition (const math::Vector3 & _force, const math::Vector3 & _pos) [virtual]

Add a force to the body using a global position.

Implements **gazebo::physics::Link** (p. 460).

10.123.3.4 virtual void gazebo::physics::ODELink::AddRelativeForce (const math::Vector3 & _force) [virtual]

Add a force to the body.

Implements **gazebo::physics::Link** (p. 460).

10.123.3.5 virtual void gazebo::physics::ODELink::AddRelativeTorque (const math::Vector3 & _torque) [virtual]

Add a torque to the body relative frame.

Implements **gazebo::physics::Link** (p. 460).

10.123.3.6 virtual void gazebo::physics::ODELink::AddTorque (const math::Vector3 & _torque) [virtual]

Add a torque to the body.

Implements **gazebo::physics::Link** (p. 461).

10.123.3.7 static void gazebo::physics::ODELink::DisabledCallback (dBodyID _id) [static]

callback when ODE determines a body is disabled

10.123.3.8 virtual void gazebo::physics::ODELink::Fini ( ) [virtual]

Finalize the link.

Reimplemented from **gazebo::physics::Link** (p. 462).

10.123.3.9 virtual bool gazebo::physics::ODELink::GetEnabled ( ) const [virtual]

Get whether this link is enabled in the physics engine.

Implements **gazebo::physics::Link** (p. 463).

10.123.3.10 virtual bool gazebo::physics::ODELink::GetGravityMode ( ) [virtual]

Get the gravity mode.

Implements **gazebo::physics::Link** (p. 463).

10.123.3.11 virtual bool gazebo::physics::ODELink::GetKinematic ( ) const [virtual]

Get whether this link is in the kinematic state.

Reimplemented from **gazebo::physics::Link** (p. 464).

10.123.3.12 dBodyID gazebo::physics::ODELink::GetODEId ( ) const

Return the ID of this link.

Returns

ODE link id

10.123.3.13 dSpaceID gazebo::physics::ODELink::GetSpaceId ( ) const

Get the link's space ID.

10.123.3.14 virtual math::Vector3 gazebo::physics::ODELink::GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the link in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 344).

10.123.3.15 virtual math::Vector3 gazebo::physics::ODELink::GetWorldForce ( ) const [virtual]

Get the force applied to the link in the world frame.

Implements **gazebo::physics::Link** (p. 467).

10.123.3.16 virtual math::Vector3 gazebo::physics::ODELink::GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the link in the world frame.

Reimplemented from **gazebo::physics::Entity** (p. 345).

10.123.3.17 virtual math::Vector3 gazebo::physics::ODELink::GetWorldTorque ( ) const [virtual]

Get the torque applied to the link in the world frame.

Implements **gazebo::physics::Link** (p. 467).

10.123.3.18 virtual void gazebo::physics::ODELink::Init ( ) [virtual]

Initialize the link.

Reimplemented from **gazebo::physics::Link** (p. 467).

10.123.3.19 virtual void gazebo::physics::ODELink::Load (sdf::ElementPtr _sdf) [virtual]

Load the link based on SDF parameters.

Parameters

_sdf	the sdf parameters

Reimplemented from **gazebo::physics::Link** (p. 468).

10.123.3.20 static void gazebo::physics::ODELink::MoveCallback (dBodyID *id*) [static]

when ODE updates dynamics bodies, this callback propagates the chagnes in pose back to Gazebo

10.123.3.21 virtual void gazebo::physics::ODELink::OnPoseChange ( ) [virtual]

Called when the pose of the entity (or one of its parents) has changed.

Reimplemented from **gazebo::physics::Link** (p. 468).

10.123.3.22 virtual void gazebo::physics::ODELink::SetAngularDamping (double *damping*) [virtual]

Set the angular damping factor.

Implements **gazebo::physics::Link** (p. 469).

10.123.3.23 virtual void gazebo::physics::ODELink::SetAngularVel (const math::Vector3 & *vel*) [virtual]

Set the angular velocity of the link.

Implements **gazebo::physics::Link** (p. 469).

10.123.3.24 virtual void gazebo::physics::ODELink::SetAutoDisable (bool _disable) [virtual]

Allow the link to auto disable.

Parameters

_disable	If true, the link is allowed to auto disable.

Implements **gazebo::physics::Link** (p. 469).

10.123.3.25 virtual void gazebo::physics::ODELink::SetEnabled (bool *enable*) const [virtual]

Set whether this link is enabled.

Implements **gazebo::physics::Link** (p. 470).

10.123.3.26 virtual void gazebo::physics::ODELink::SetForce (const math::Vector3 & _force) [virtual]

Set the force applied to the link.

Implements **gazebo::physics::Link** (p. 470).

10.123.3.27 virtual void gazebo::physics::ODELink::SetGravityMode (bool *mode*) [virtual]

Set whether gravity affects this link.

Implements **gazebo::physics::Link** (p. 470).

10.123.3.28 virtual void gazebo::physics::ODELink::SetKinematic (const bool & *state*) [virtual]

Set whether this link is in the kinematic state.

Reimplemented from **gazebo::physics::Link** (p. 470).

10.123.3.29 virtual void gazebo::physics::ODELink::SetLinearDamping (double *damping*) [virtual]

Set the linear damping factor.

Implements **gazebo::physics::Link** (p. 471).

10.123.3.30 virtual void gazebo::physics::ODELink::SetLinearVel (const **math::Vector3** & *vel*) [virtual]

Set the linear velocity of the link.

Implements **gazebo::physics::Link** (p. 471).

10.123.3.31 void gazebo::physics::ODELink::SetSelfCollide (bool *collide*) [virtual]

Set whether this link will collide with others in the model.

Implements **gazebo::physics::Link** (p. 471).

10.123.3.32 void gazebo::physics::ODELink::SetSpaceId (dSpaceID *spaceid*)

Set the link's space ID.

10.123.3.33 virtual void gazebo::physics::ODELink::SetTorque (const **math::Vector3** & *_torque*) [virtual]

Set the torque applied to the link.

Implements **gazebo::physics::Link** (p. 472).

10.123.3.34 virtual void gazebo::physics::ODELink::Update ( ) [virtual]

Update the link.

Reimplemented from **gazebo::physics::Link** (p. 472).

10.123.3.35 virtual void gazebo::physics::ODELink::UpdateMass ( ) [virtual]

Update the mass matrix.

Reimplemented from **gazebo::physics::Link** (p. 472).

10.123.3.36 virtual void gazebo::physics::ODELink::UpdateSurface ( ) [virtual]

Update other parameters for ODE.

Reimplemented from **gazebo::physics::Link** (p. 472).

10.123.4 Member Data Documentation

10.123.4.1 math::Pose gazebo::physics::ODELink::pose [protected]

The documentation for this class was generated from the following file:

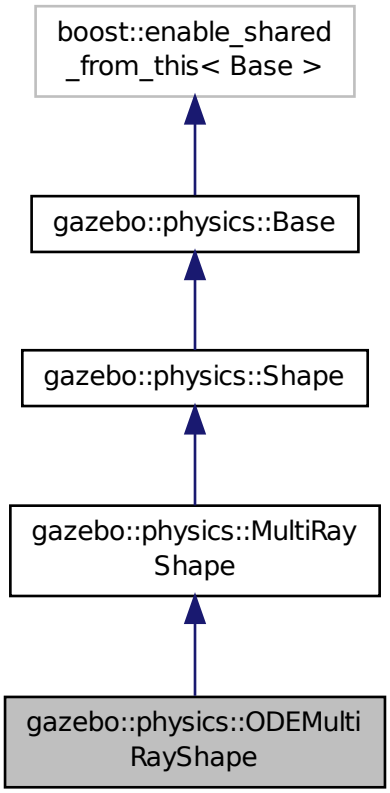
- **ODELink.hh**

10.124 gazebo::physics::ODEMultiRayShape Class Reference

ODE specific version of **MultiRayShape** (p. 549).

`#include <ODEMultiRayShape.hh>`

Inheritance diagram for gazebo::physics::ODEMultiRayShape:



Public Member Functions

- **ODEMultiRayShape** (**CollisionPtr parent**)

  Constructor.
- virtual ~**ODEMultiRayShape** ()

  Destructor.
- virtual void **UpdateRays** ()

  Update the rays.

Protected Member Functions

- void **AddRay** (const **math::Vector3** &start, const **math::Vector3** &end)

  Add a ray to the collision.

Additional Inherited Members

10.124.1 Detailed Description

ODE specific version of **MultiRayShape** (p. 549).

10.124.2 Constructor & Destructor Documentation

10.124.2.1 gazebo::physics::ODEMultiRayShape::ODEMultiRayShape (CollisionPtr *parent*)

Constructor.

10.124.2.2 virtual gazebo::physics::ODEMultiRayShape::~ODEMultiRayShape ( ) [virtual]

Destructor.

10.124.3 Member Function Documentation

10.124.3.1 void gazebo::physics::ODEMultiRayShape::AddRay (const math::Vector3 & *start,* const math::Vector3 & *end*) [protected], [virtual]

Add a ray to the collision.

Reimplemented from **gazebo::physics::MultiRayShape** (p. 551).

10.124.3.2 virtual void gazebo::physics::ODEMultiRayShape::UpdateRays ( ) [virtual]

Update the rays.

Implements **gazebo::physics::MultiRayShape** (p. 555).

The documentation for this class was generated from the following file:

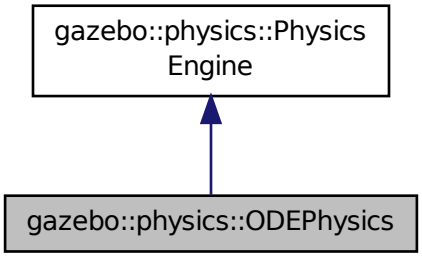
- **ODEMultiRayShape.hh**

10.125 gazebo::physics::ODEPhysics Class Reference

ODE physics engine.

`#include <ODEPhysics.hh>`

Inheritance diagram for gazebo::physics::ODEPhysics:



Public Member Functions

- **ODEPhysics** (**WorldPtr world**)

  Constructor.
- virtual ∼**ODEPhysics** ()

  Destructor.
- void **Collide** (**ODECollision** ∗collision1, **ODECollision** ∗collision2, dContactGeom ∗contactCollisions)

  Collide two collisions.
- virtual **CollisionPtr CreateCollision** (const std::string &_shapeType, **LinkPtr** _parent)

  Create a collision.
- void **CreateContact** (**ODECollision** ∗collision1, **ODECollision** ∗collision2)

  Not yet implemented.
- virtual **JointPtr CreateJoint** (const std::string &type, **ModelPtr** _parent)

  Create a new joint.
- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)

  Create a new body.
- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)

  *Create a collision **physics::Shape** (p. 779) object given its type.*
- virtual void **DebugPrint** () const

  Debug print out of the physic engine state.
- virtual void **Fini** ()

  Finilize the ODE engine.
- double **GetContactMaxCorrectingVel** ()

  access functions to set ODE parameters
- double **GetContactSurfaceLayer** ()

  access functions to set ODE parameters
- **math::Vector3 GetGravity** () const

  Get gravity vector.
- int **GetMaxContacts** ()

  access functions to set ODE parameters

- int **GetSORPGSIters** ()

  access functions to set ODE parameters
- int **GetSORPGSPreconIters** ()

  access functions to set ODE parameters
- double **GetSORPGSW** ()

  access functions to set ODE parameters
- dSpaceID **GetSpaceId** () const

  Return the space id.
- virtual double **GetStepTime** ()

  Get the simulation step time.
- virtual std::string **GetStepType** () const

  Get the step type.
- double **GetWorldCFM** ()

  access functions to set ODE parameters
- double **GetWorldERP** ()

  access functions to set ODE parameters
- dWorldID **GetWorldId** ()

  Get the world id.
- virtual void **Init** ()

  Initialize the ODE engine.
- virtual void **InitForThread** ()

  Init the engine for threads.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the ODE engine.
- void **ProcessContactFeedback** (**ContactFeedback** *feedback, msgs::Contact *_msg)

  *process contact feedbacks into **physics::ContactFeedback** (p. 299)*
- void **Reset** ()

  Empties dynamically created contact joints.
- void **SetContactMaxCorrectingVel** (double vel)

  access functions to set ODE parameters
- void **SetContactSurfaceLayer** (double layer_depth)

  access functions to set ODE parameters
- virtual void **SetGravity** (const **gazebo::math::Vector3** &gravity)

  Set the gavity vector.
- void **SetMaxContacts** (unsigned int max_contacts)

  access functions to set ODE parameters
- void **SetSORPGSIters** (unsigned int iters)

  access functions to set ODE parameters
- void **SetSORPGSPreconIters** (unsigned int iters)

  access functions to set ODE parameters
- void **SetSORPGSW** (double w)

  access functions to set ODE parameters
- void **SetStepTime** (double _value)

  Set the step time.
- virtual void **SetStepType** (const std::string &_type)

  Set the step type.
- void **SetWorldCFM** (double cfm)

access functions to set ODE parameters

- void **SetWorldERP** (double erp)

  access functions to set ODE parameters
- virtual void **UpdateCollision** ()

  Update the ODE collision.
- virtual void **UpdatePhysics** ()

  Update the ODE engine.

Static Public Member Functions

- static void **ConvertMass** (**InertialPtr** _interial, void ∗odeMass)

  Convert an odeMass to Mass.
- static void **ConvertMass** (void ∗odeMass, **InertialPtr** _inertial)

  Convert an odeMass to Mass.

Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &)

  act on a msgs::Physics message request
- virtual void **OnRequest** (ConstRequestPtr &)

  act on a msgs::Request for physics

Additional Inherited Members

10.125.1 Detailed Description

ODE physics engine.

10.125.2 Constructor & Destructor Documentation

10.125.2.1 gazebo::physics::ODEPhysics::ODEPhysics (WorldPtr *world*)

Constructor.

10.125.2.2 virtual gazebo::physics::ODEPhysics::∼ODEPhysics ( ) [virtual]

Destructor.

10.125.3 Member Function Documentation

10.125.3.1 void gazebo::physics::ODEPhysics::Collide (ODECollision ∗ *collision1,* ODECollision ∗ *collision2,* dContactGeom ∗ *contactCollisions*)

Collide two collisions.

10.125.3.2 static void gazebo::physics::ODEPhysics::ConvertMass (InertialPtr *_interial,* void * *odeMass*) [static]

Convert an odeMass to Mass.

10.125.3.3 static void gazebo::physics::ODEPhysics::ConvertMass (void * *odeMass,* InertialPtr *_inertial*) [static]

Convert an odeMass to Mass.

10.125.3.4 virtual CollisionPtr gazebo::physics::ODEPhysics::CreateCollision (const std::string & *_shapeType,* LinkPtr *_parent*) [virtual]

Create a collision.

Implements **gazebo::physics::PhysicsEngine** (p. 654).

10.125.3.5 void gazebo::physics::ODEPhysics::CreateContact (ODECollision * *collision1,* ODECollision * *collision2*)

Not yet implemented.

10.125.3.6 virtual JointPtr gazebo::physics::ODEPhysics::CreateJoint (const std::string & *type,* ModelPtr *_parent*) [virtual]

Create a new joint.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.125.3.7 virtual LinkPtr gazebo::physics::ODEPhysics::CreateLink (ModelPtr *_parent*) [virtual]

Create a new body.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.125.3.8 virtual ShapePtr gazebo::physics::ODEPhysics::CreateShape (const std::string & *_shapeType,* CollisionPtr *_collision*) [virtual]

Create a collision **physics::Shape** (p. 779) object given its type.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.125.3.9 virtual void gazebo::physics::ODEPhysics::DebugPrint ( ) const [virtual]

Debug print out of the physic engine state.

Implements **gazebo::physics::PhysicsEngine** (p. 655).

10.125.3.10 virtual void gazebo::physics::ODEPhysics::Fini ( ) [virtual]

Finilize the ODE engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 656).

10.125.3.11 double gazebo::physics::ODEPhysics::GetContactMaxCorrectingVel ( ) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 656).

10.125.3.12 double gazebo::physics::ODEPhysics::GetContactSurfaceLayer ( ) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 656).

10.125.3.13 math::Vector3 gazebo::physics::ODEPhysics::GetGravity ( ) const

Get gravity vector.

10.125.3.14 int gazebo::physics::ODEPhysics::GetMaxContacts ( ) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 656).

10.125.3.15 int gazebo::physics::ODEPhysics::GetSORPGSIters ( ) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 657).

10.125.3.16 int gazebo::physics::ODEPhysics::GetSORPGSPreconIters ( ) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 657).

10.125.3.17 double gazebo::physics::ODEPhysics::GetSORPGSW ( ) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 657).

10.125.3.18 dSpaceID gazebo::physics::ODEPhysics::GetSpaceId ( ) const

Return the space id.

10.125.3.19 virtual double gazebo::physics::ODEPhysics::GetStepTime ( ) [virtual]

Get the simulation step time.

Implements **gazebo::physics::PhysicsEngine** (p. 658).

10.125.3.20 virtual std::string gazebo::physics::ODEPhysics::GetStepType ( ) const [virtual]

Get the step type.

10.125.3.21 double gazebo::physics::ODEPhysics::GetWorldCFM ( ) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 658).

10.125.3.22 double gazebo::physics::ODEPhysics::GetWorldERP ( ) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 658).

10.125.3.23 dWorldID gazebo::physics::ODEPhysics::GetWorldId ( )

Get the world id.

10.125.3.24 virtual void gazebo::physics::ODEPhysics::Init ( ) [virtual]

Initialize the ODE engine.

Implements **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.25 virtual void gazebo::physics::ODEPhysics::InitForThread ( ) [virtual]

Init the engine for threads.

Implements **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.26 virtual void gazebo::physics::ODEPhysics::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the ODE engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.27 virtual void gazebo::physics::ODEPhysics::OnPhysicsMsg (ConstPhysicsPtr &) [protected], [virtual]

act on a msgs::Physics message request

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.28 virtual void gazebo::physics::ODEPhysics::OnRequest (ConstRequestPtr &) [protected], [virtual]

act on a msgs::Request for physics

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.29 void gazebo::physics::ODEPhysics::ProcessContactFeedback (ContactFeedback * *feedback*, msgs::Contact * *_msg*)

process contact feedbacks into **physics::ContactFeedback** (p. 299)

10.125.3.30 void gazebo::physics::ODEPhysics::Reset ( ) [virtual]

Empties dynamically created contact joints.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 659).

10.125.3.31 void gazebo::physics::ODEPhysics::SetContactMaxCorrectingVel (double *vel*) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 660).

10.125.3.32 void gazebo::physics::ODEPhysics::SetContactSurfaceLayer (double *layer_depth*) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 660).

10.125.3.33 virtual void gazebo::physics::ODEPhysics::SetGravity (const gazebo::math::Vector3 & *gravity*) [virtual]

Set the gavity vector.

Implements **gazebo::physics::PhysicsEngine** (p. 660).

10.125.3.34 void gazebo::physics::ODEPhysics::SetMaxContacts (unsigned int *max_contacts*)

access functions to set ODE parameters

10.125.3.35 void gazebo::physics::ODEPhysics::SetSORPGSIters (unsigned int *iters*) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 661).

10.125.3.36 void gazebo::physics::ODEPhysics::SetSORPGSPreconIters (unsigned int *iters*) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 661).

10.125.3.37 void gazebo::physics::ODEPhysics::SetSORPGSW (double *w*) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 661).

10.125.3.38 void gazebo::physics::ODEPhysics::SetStepTime (double *_value*) [virtual]

Set the step time.

Implements **gazebo::physics::PhysicsEngine** (p. 661).

10.125.3.39 virtual void gazebo::physics::ODEPhysics::SetStepType (const std::string & *_type*) [virtual]

Set the step type.

10.125.3.40 void gazebo::physics::ODEPhysics::SetWorldCFM (double *cfm*) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 662).

10.125.3.41 void gazebo::physics::ODEPhysics::SetWorldERP (double *erp*) [virtual]

access functions to set ODE parameters

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 662).

10.125.3.42 virtual void gazebo::physics::ODEPhysics::UpdateCollision ( ) [virtual]

Update the ODE collision.

Implements **gazebo::physics::PhysicsEngine** (p. 662).

10.125.3.43 virtual void gazebo::physics::ODEPhysics::UpdatePhysics ( ) [virtual]

Update the ODE engine.

Reimplemented from **gazebo::physics::PhysicsEngine** (p. 662).

The documentation for this class was generated from the following file:

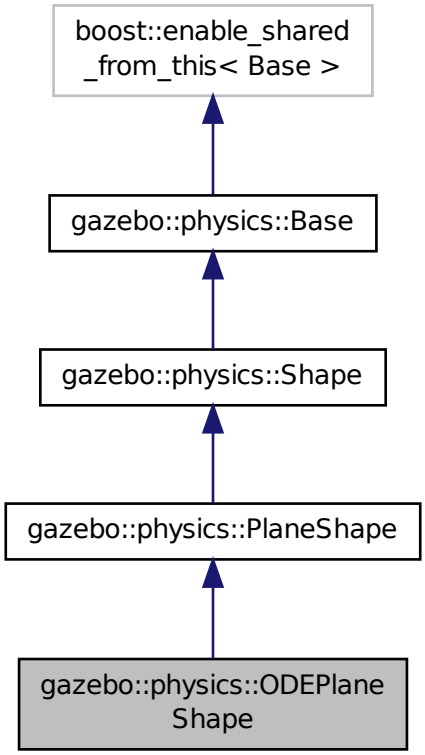
- **ODEPhysics.hh**

10.126 gazebo::physics::ODEPlaneShape Class Reference

An ODE Plane shape.

`#include <ODEPlaneShape.hh>`

Inheritance diagram for gazebo::physics::ODEPlaneShape:



Public Member Functions

- **ODEPlaneShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**ODEPlaneShape** ()

  Destructor.
- void **CreatePlane** ()

  Create the plane.
- void **SetAltitude** (const **math::Vector3** &pos)

  Set the altitude of the plane.

Additional Inherited Members

10.126.1 Detailed Description

An ODE Plane shape.

10.126.2 Constructor & Destructor Documentation

10.126.2.1 gazebo::physics::ODEPlaneShape::ODEPlaneShape (CollisionPtr *_parent*) [inline]

Constructor.

10.126.2.2 virtual gazebo::physics::ODEPlaneShape::~ODEPlaneShape ( ) [inline], [virtual]

Destructor.

10.126.3 Member Function Documentation

10.126.3.1 void gazebo::physics::ODEPlaneShape::CreatePlane ( ) [inline], [virtual]

Create the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 673).

References gazebo::physics::Shape::collisionParent, gazebo::physics::PlaneShape::CreatePlane(), gazebo::physics::ODECollision::GetCollisionId(), gazebo::physics::PlaneShape::GetNormal(), gazebo::physics::ODECollision::GetSpaceId(), NULL, gazebo::physics::ODECollision::SetCollision(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

10.126.3.2 void gazebo::physics::ODEPlaneShape::SetAltitude (const math::Vector3 & *pos*) [inline], [virtual]

Set the altitude of the plane.

Reimplemented from **gazebo::physics::PlaneShape** (p. 673).

References gazebo::physics::Shape::collisionParent, gazebo::physics::ODECollision::GetCollisionId(), gazebo::physics::PlaneShape::SetAltitude(), gazebo::math::Vector3::x, gazebo::math::Vector3::y, and gazebo::math::Vector3::z.

The documentation for this class was generated from the following file:

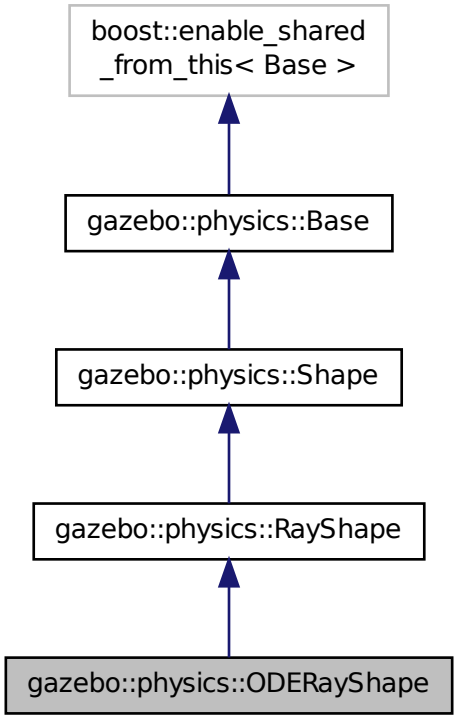
- **ODEPlaneShape.hh**

10.127 gazebo::physics::ODERayShape Class Reference

Ray collision.

`#include <ODERayShape.hh>`

Inheritance diagram for gazebo::physics::ODERayShape:



Public Member Functions

- **ODERayShape** (**PhysicsEnginePtr** _physicsEngine)

  Constructor for a global ray.
- **ODERayShape** (**CollisionPtr** collision)

  Constructor.
- virtual ~**ODERayShape** ()

  Destructor.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)

  Get the nearest intersection.
- virtual void **SetPoints** (const **math::Vector3** &posStart, const **math::Vector3** &posEnd)

  Set the ray based on starting and ending points relative to the body.
- virtual void **Update** ()

  Update the tay collision.

Additional Inherited Members

10.127.1 Detailed Description

Ray collision.

10.127.2 Constructor & Destructor Documentation

10.127.2.1 gazebo::physics::ODERayShape::ODERayShape (PhysicsEnginePtr *_physicsEngine*)

Constructor for a global ray.

10.127.2.2 gazebo::physics::ODERayShape::ODERayShape (CollisionPtr *collision*)

Constructor.

Parameters

body	**Link** (p. 454) the ray is attached to

10.127.2.3 virtual gazebo::physics::ODERayShape::~ODERayShape ( ) [virtual]

Destructor.

10.127.3 Member Function Documentation

10.127.3.1 virtual void gazebo::physics::ODERayShape::GetIntersection (double & *_dist,* std::string & *_entity*) [virtual]

Get the nearest intersection.

Implements **gazebo::physics::RayShape** (p. 720).

10.127.3.2 virtual void gazebo::physics::ODERayShape::SetPoints (const math::Vector3 & *posStart,* const math::Vector3 & *posEnd*) [virtual]

Set the ray based on starting and ending points relative to the body.

Parameters

posStart	Start position, relative the body
posEnd	End position, relative to the body

Reimplemented from **gazebo::physics::RayShape** (p. 721).

10.127.3.3 virtual void gazebo::physics::ODERayShape::Update ( ) [virtual]

Update the tay collision.

Implements **gazebo::physics::RayShape** (p. 722).

The documentation for this class was generated from the following file:

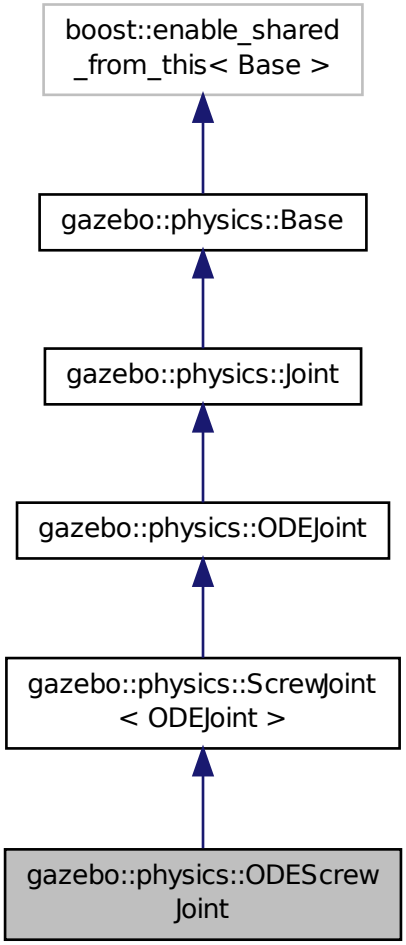
- **ODERayShape.hh**

10.128 gazebo::physics::ODEScrewJoint Class Reference

A screw joint.

`#include <ODEScrewJoint.hh>`

Inheritance diagram for gazebo::physics::ODEScrewJoint:



Public Member Functions

- **ODEScrewJoint** (dWorldID worldId, **BasePtr** _parent)

  Constructor.
- virtual ~**ODEScrewJoint** ()

  Destructor.
- void **ApplyDamping** ()

callback to apply damping force to joint

- virtual **math::Angle GetAngleImpl** (int index) const

  Get the position of the joint.
- virtual **math::Vector3 GetGlobalAxis** (int index) const

  *Get the axis of rotation of an **ODEScrewJoint** (p. 622).*
- virtual double **GetMaxForce** (int index)

  Get the max allowed force of an axis(index).
- virtual double **GetParam** (int parameter) const

  Get the _parameter.
- virtual double **GetVelocity** (int index) const

  Get the rate of change.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)

  Set the axis of motion.
- virtual void **SetDamping** (int _index, double _damping)
- virtual void **SetForce** (int index, double force)

  Set the screw force.
- virtual void **SetMaxForce** (int index, double t)

  Set the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)

  Set the _parameter.
- virtual void **SetThreadPitch** (int _index, double _threadPitch)

  Set screw joint thread pitch.
- virtual void **SetVelocity** (int index, double angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load the **ODEScrewJoint** (p. 622) from ::Element.*

Additional Inherited Members

10.128.1 Detailed Description

A screw joint.

10.128.2 Constructor & Destructor Documentation

10.128.2.1 gazebo::physics::ODEScrewJoint::ODEScrewJoint (dWorldID *worldId,* BasePtr *_parent*)

Constructor.

10.128.2.2 virtual gazebo::physics::ODEScrewJoint::~ODEScrewJoint ( ) [virtual]

Destructor.

10.128.3 Member Function Documentation

10.128.3.1 void gazebo::physics::ODEScrewJoint::ApplyDamping ( )

callback to apply damping force to joint

10.128.3.2 virtual math::Angle gazebo::physics::ODEScrewJoint::GetAngleImpl (int *index*) const [virtual]

Get the position of the joint.

Implements **gazebo::physics::Joint** (p. 429).

10.128.3.3 virtual math::Vector3 gazebo::physics::ODEScrewJoint::GetGlobalAxis (int *index*) const [virtual]

Get the axis of rotation of an **ODEScrewJoint** (p. 622).

Implements **gazebo::physics::Joint** (p. 430).

10.128.3.4 virtual double gazebo::physics::ODEScrewJoint::GetMaxForce (int *index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.128.3.5 virtual double gazebo::physics::ODEScrewJoint::GetParam (int *parameter*) const [virtual]

Get the _parameter.

Reimplemented from **gazebo::physics::ODEJoint** (p. 598).

10.128.3.6 virtual double gazebo::physics::ODEScrewJoint::GetVelocity (int *index*) const [virtual]

Get the rate of change.

Implements **gazebo::physics::Joint** (p. 433).

10.128.3.7 virtual void gazebo::physics::ODEScrewJoint::Load (sdf::ElementPtr *_sdf*) [protected], [virtual]

Load the **ODEScrewJoint** (p. 622) from ::Element.

Reimplemented from **gazebo::physics::ScrewJoint**< **ODEJoint** > (p. 761).

10.128.3.8 virtual void gazebo::physics::ODEScrewJoint::SetAxis (int *index,* const math::Vector3 & *axis*) [virtual]

Set the axis of motion.

Implements **gazebo::physics::Joint** (p. 435).

10.128.3.9 virtual void gazebo::physics::ODEScrewJoint::SetDamping (int *_index,* double *_damping*) [virtual]

Implements **gazebo::physics::Joint** (p. 435).

10.128.3.10 virtual void gazebo::physics::ODEScrewJoint::SetForce (int *index,* double *force*) [virtual]

Set the screw force.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.128.3.11 virtual void gazebo::physics::ODEScrewJoint::SetMaxForce (int *index,* double *t*) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.128.3.12 virtual void gazebo::physics::ODEScrewJoint::SetParam (int *parameter,* double *value*) [virtual]

Set the _parameter.

Reimplemented from **gazebo::physics::ODEJoint** (p. 599).

10.128.3.13 virtual void gazebo::physics::ODEScrewJoint::SetThreadPitch (int *_index,* double *_threadPitch*) [virtual]

Set screw joint thread pitch.

This must be implemented in a child class

Parameters

in	*_index*	Index of the axis.
in	*_threadPitch*	Thread pitch value.

Implements **gazebo::physics::ScrewJoint**< **ODEJoint** > (p. 762).

10.128.3.14 virtual void gazebo::physics::ODEScrewJoint::SetVelocity (int *index,* double *angle*) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

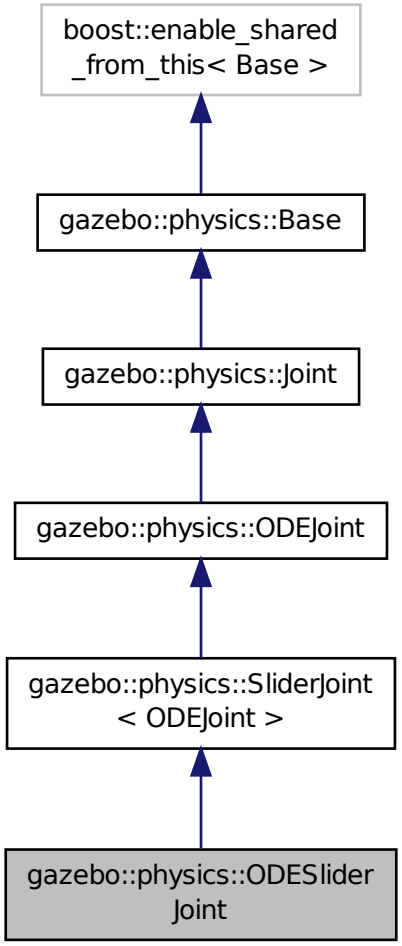
- **ODEScrewJoint.hh**

10.129 gazebo::physics::ODESliderJoint Class Reference

A slider joint.

`#include <ODESliderJoint.hh>`

Inheritance diagram for gazebo::physics::ODESliderJoint:



Public Member Functions

- **ODESliderJoint** (dWorldID worldId, **BasePtr** _parent)

  Constructor.
- virtual ~**ODESliderJoint** ()

  Destructor.
- void **ApplyDamping** ()

  callback to apply damping force to joint
- virtual **math::Angle GetAngleImpl** (int index) const

  Get the position of the joint.
- virtual **math::Vector3 GetGlobalAxis** (int index) const

  Get the axis of rotation.

- virtual double **GetMaxForce** (int index)

  Get the max allowed force of an axis(index).
- virtual double **GetParam** (int parameter) const

  Get the _parameter.
- virtual double **GetVelocity** (int index) const

  Get the rate of change.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)

  Set the axis of motion.
- virtual void **SetDamping** (int _index, double _damping)

  Set joint damping, not yet implemented.
- virtual void **SetForce** (int index, double force)

  Set the slider force.
- virtual void **SetMaxForce** (int index, double t)

  Set the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)

  Set the _parameter.
- virtual void **SetVelocity** (int index, double angle)

  Set the velocity of an axis(index).

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load the **ODESliderJoint** (p. 625).*

Additional Inherited Members

10.129.1 Detailed Description

A slider joint.

10.129.2 Constructor & Destructor Documentation

10.129.2.1 gazebo::physics::ODESliderJoint::ODESliderJoint (dWorldID *worldId,* BasePtr *_parent*)

Constructor.

10.129.2.2 virtual gazebo::physics::ODESliderJoint::~ODESliderJoint ( ) [virtual]

Destructor.

10.129.3 Member Function Documentation

10.129.3.1 void gazebo::physics::ODESliderJoint::ApplyDamping ( )

callback to apply damping force to joint

10.129.3.2 virtual math::Angle gazebo::physics::ODESliderJoint::GetAngleImpl (int *index*) const [virtual]

Get the position of the joint.

Implements **gazebo::physics::Joint** (p. 429).

10.129.3.3 virtual math::Vector3 gazebo::physics::ODESliderJoint::GetGlobalAxis (int *index*) const [virtual]

Get the axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.129.3.4 virtual double gazebo::physics::ODESliderJoint::GetMaxForce (int *index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.129.3.5 virtual double gazebo::physics::ODESliderJoint::GetParam (int *parameter*) const [virtual]

Get the _parameter.

Reimplemented from **gazebo::physics::ODEJoint** (p. 598).

10.129.3.6 virtual double gazebo::physics::ODESliderJoint::GetVelocity (int *index*) const [virtual]

Get the rate of change.

Implements **gazebo::physics::Joint** (p. 433).

10.129.3.7 virtual void gazebo::physics::ODESliderJoint::Load (sdf::ElementPtr *_sdf*) [protected], [virtual]

Load the **ODESliderJoint** (p. 625).

Reimplemented from **gazebo::physics::SliderJoint**< **ODEJoint** > (p. 805).

10.129.3.8 virtual void gazebo::physics::ODESliderJoint::SetAxis (int *index*, const math::Vector3 & *axis*) [virtual]

Set the axis of motion.

Implements **gazebo::physics::Joint** (p. 435).

10.129.3.9 virtual void gazebo::physics::ODESliderJoint::SetDamping (int *_index*, double *_damping*) [virtual]

Set joint damping, not yet implemented.

Implements **gazebo::physics::Joint** (p. 435).

10.129.3.10 virtual void gazebo::physics::ODESliderJoint::SetForce (int *index*, double *force*) [virtual]

Set the slider force.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.129.3.11 virtual void gazebo::physics::ODESliderJoint::SetMaxForce (int *index,* double *t*) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.129.3.12 virtual void gazebo::physics::ODESliderJoint::SetParam (int *parameter,* double *value*) [virtual]

Set the _parameter.

Reimplemented from **gazebo::physics::ODEJoint** (p. 599).

10.129.3.13 virtual void gazebo::physics::ODESliderJoint::SetVelocity (int *index,* double *angle*) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

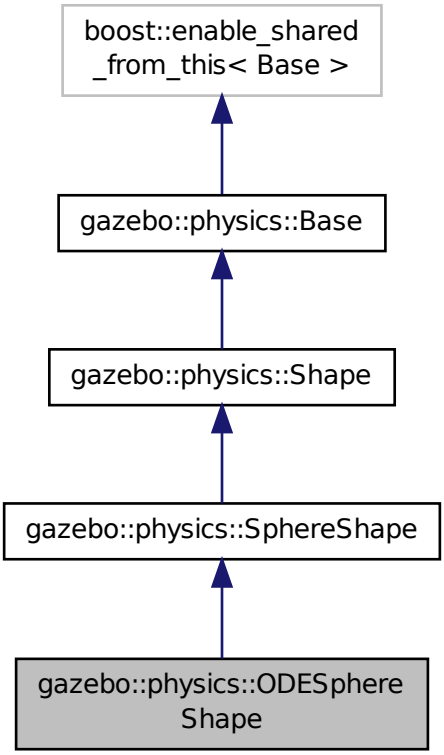
- **ODESliderJoint.hh**

10.130 gazebo::physics::ODESphereShape Class Reference

A ODE sphere shape.

`#include <ODESphereShape.hh>`

Inheritance diagram for gazebo::physics::ODESphereShape:



Public Member Functions

- **ODESphereShape** (**ODECollisionPtr** _parent)

  Constructor.
- virtual ~**ODESphereShape** ()

  Destructor.
- void **SetRadius** (double _radius)

  Set the radius.

Additional Inherited Members

10.130.1 Detailed Description

A ODE sphere shape.

10.130.2 Constructor & Destructor Documentation

10.130.2.1 gazebo::physics::ODESphereShape::ODESphereShape (ODECollisionPtr *_parent*) [inline]

Constructor.

10.130.2.2 virtual gazebo::physics::ODESphereShape::~ODESphereShape ( ) [inline], [virtual]

Destructor.

10.130.3 Member Function Documentation

10.130.3.1 void gazebo::physics::ODESphereShape::SetRadius (double *_radius*) [inline], [virtual]

Set the radius.

Reimplemented from **gazebo::physics::SphereShape** (p. 808).

References gazebo::physics::Shape::collisionParent, gazebo::physics::ODECollision::GetCollisionId(), NULL, gazebo::physics::ODECollision::SetCollision(), and gazebo::physics::SphereShape::SetRadius().

The documentation for this class was generated from the following file:

- **ODESphereShape.hh**

10.131 gazebo::physics::ODESurfaceParams Class Reference

Surface params.

`#include <ODESurfaceParams.hh>`

Public Member Functions

- **ODESurfaceParams** ()

  Constructor.
- virtual ~**ODESurfaceParams** ()

  Destructor.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the contact params.

10.131.1 Detailed Description

Surface params.

10.131.2 Constructor & Destructor Documentation

10.131.2.1 gazebo::physics::ODESurfaceParams::ODESurfaceParams ( )

Constructor.

10.131.2.2 virtual gazebo::physics::ODESurfaceParams::~ODESurfaceParams ( ) [virtual]

Destructor.

10.131.3 Member Function Documentation

10.131.3.1 virtual void gazebo::physics::ODESurfaceParams::Load (sdf::ElementPtr _sdf) [virtual]

Load the contact params.

The documentation for this class was generated from the following file:

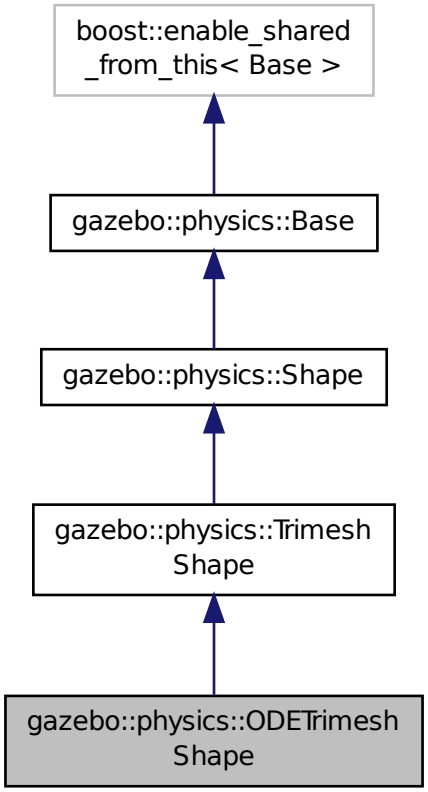
- **ODESurfaceParams.hh**

10.132 gazebo::physics::ODETrimeshShape Class Reference

Triangle mesh collision.

`#include <ODETrimeshShape.hh>`

Inheritance diagram for gazebo::physics::ODETrimeshShape:



Public Member Functions

- **ODETrimeshShape** (**CollisionPtr** _parent)

  Constructor.
- virtual ~**ODETrimeshShape** ()

  Destructor.
- void **Update** ()

  Update function.

Protected Member Functions

- virtual void **Init** ()

  Init the trimesh shape.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the trimesh.

Additional Inherited Members

10.132.1 Detailed Description

Triangle mesh collision.

10.132.2 Constructor & Destructor Documentation

10.132.2.1 gazebo::physics::ODETrimeshShape::ODETrimeshShape (CollisionPtr *_parent*)

Constructor.

10.132.2.2 virtual gazebo::physics::ODETrimeshShape::~ODETrimeshShape ( ) [virtual]

Destructor.

10.132.3 Member Function Documentation

10.132.3.1 virtual void gazebo::physics::ODETrimeshShape::Init ( ) [protected], [virtual]

Init the trimesh shape.

Reimplemented from **gazebo::physics::TrimeshShape** (p. 868).

10.132.3.2 virtual void gazebo::physics::ODETrimeshShape::Load (sdf::ElementPtr *_sdf*) [protected], [virtual]

Load the trimesh.

Reimplemented from **gazebo::physics::Base** (p. 154).

10.132.3.3 void gazebo::physics::ODETrimeshShape::Update ( ) [virtual]

Update function.

Reimplemented from **gazebo::physics::TrimeshShape** (p. 868).

The documentation for this class was generated from the following file:

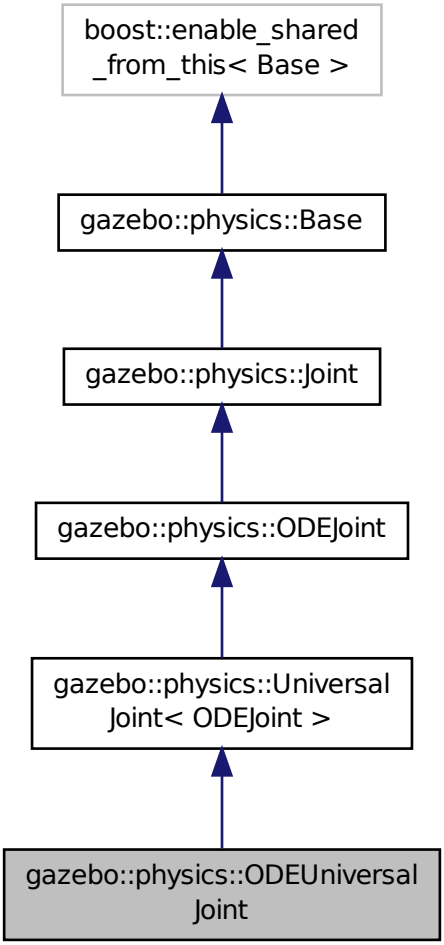
- **ODETrimeshShape.hh**

10.133 gazebo::physics::ODEUniversalJoint Class Reference

A universal joint.

`#include <ODEUniversalJoint.hh>`

Inheritance diagram for gazebo::physics::ODEUniversalJoint:



Public Member Functions

- **ODEUniversalJoint** (dWorldID worldId, **BasePtr** _parent)

  Constructor.
- virtual ~**ODEUniversalJoint** ()

  Destuctor.
- virtual **math::Vector3 GetAnchor** (int index) const

  Get the anchor point.
- virtual **math::Angle GetAngleImpl** (int index) const

  Get the angle of axis.
- virtual **math::Vector3 GetGlobalAxis** (int index) const

  Get the first axis of rotation.

- virtual double **GetMaxForce** (int index)

  Get the max allowed force of an axis(index).
- virtual double **GetVelocity** (int index) const

  Get the angular rate of an axis.
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)

  Set the anchor point.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)

  Set the first axis of rotation.
- virtual void **SetDamping** (int index, double damping)

  Set joint damping, not yet implemented.
- virtual void **SetForce** (int index, double torque)

  Set the torque of a joint.
- virtual void **SetMaxForce** (int index, double t)

  Set the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)

  Set the parameter to value.
- virtual void **SetVelocity** (int index, double angle)

  Set the velocity of an axis(index).

Additional Inherited Members

10.133.1 Detailed Description

A universal joint.

10.133.2 Constructor & Destructor Documentation

10.133.2.1 gazebo::physics::ODEUniversalJoint::ODEUniversalJoint (dWorldID *worldId,* BasePtr *_parent*)

Constructor.

10.133.2.2 virtual gazebo::physics::ODEUniversalJoint::~ODEUniversalJoint ( ) [virtual]

Destuctor.

10.133.3 Member Function Documentation

10.133.3.1 virtual math::Vector3 gazebo::physics::ODEUniversalJoint::GetAnchor (int *index*) const [virtual]

Get the anchor point.

Implements **gazebo::physics::Joint** (p. 428).

10.133.3.2 virtual math::Angle gazebo::physics::ODEUniversalJoint::GetAngleImpl (int *index*) const [virtual]

Get the angle of axis.

Implements **gazebo::physics::Joint** (p. 429).

10.133.3.3 virtual math::Vector3 gazebo::physics::ODEUniversalJoint::GetGlobalAxis (int *index*) const [virtual]

Get the first axis of rotation.

Implements **gazebo::physics::Joint** (p. 430).

10.133.3.4 virtual double gazebo::physics::ODEUniversalJoint::GetMaxForce (int *index*) [virtual]

Get the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 432).

10.133.3.5 virtual double gazebo::physics::ODEUniversalJoint::GetVelocity (int *index*) const [virtual]

Get the angular rate of an axis.

Implements **gazebo::physics::Joint** (p. 433).

10.133.3.6 virtual void gazebo::physics::ODEUniversalJoint::SetAnchor (int *index,* const math::Vector3 & *anchor*) [virtual]

Set the anchor point.

Implements **gazebo::physics::Joint** (p. 434).

10.133.3.7 virtual void gazebo::physics::ODEUniversalJoint::SetAxis (int *index,* const math::Vector3 & *axis*) [virtual]

Set the first axis of rotation.

Implements **gazebo::physics::Joint** (p. 435).

10.133.3.8 virtual void gazebo::physics::ODEUniversalJoint::SetDamping (int *index,* double *damping*) [virtual]

Set joint damping, not yet implemented.

Implements **gazebo::physics::Joint** (p. 435).

10.133.3.9 virtual void gazebo::physics::ODEUniversalJoint::SetForce (int *index,* double *torque*) [virtual]

Set the torque of a joint.

Reimplemented from **gazebo::physics::Joint** (p. 435).

10.133.3.10 virtual void gazebo::physics::ODEUniversalJoint::SetMaxForce (int *index,* double *t*) [virtual]

Set the max allowed force of an axis(index).

Implements **gazebo::physics::Joint** (p. 436).

10.133.3.11 virtual void gazebo::physics::ODEUniversalJoint::SetParam (int *parameter,* double *value*) [virtual]

Set the parameter to value.

Reimplemented from **gazebo::physics::ODEJoint** (p. 599).

10.133.3.12 virtual void gazebo::physics::ODEUniversalJoint::SetVelocity (int *index*, double *angle*) [virtual]

Set the velocity of an axis(index).

Implements **gazebo::physics::Joint** (p. 437).

The documentation for this class was generated from the following file:

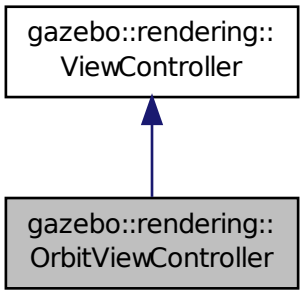
- **ODEUniversalJoint.hh**

10.134 gazebo::rendering::OrbitViewController Class Reference

Orbit view controller.

`#include <OrbitViewController.hh>`

Inheritance diagram for gazebo::rendering::OrbitViewController:



Public Member Functions

- **OrbitViewController** (**UserCameraPtr** _camera)

  Constructor.
- virtual ~**OrbitViewController** ()

  Destructor.
- **math::Vector3 GetFocalPoint** () const

  Get the focal point.
- virtual void **HandleKeyPressEvent** (const std::string &_key)

  Handle a key press event.
- void **HandleKeyReleaseEvent** (const std::string &_key)

  Handle a key release event.
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)

Handle a mouse event.

- virtual void **Init** ()

  Initialize the controller.
- virtual void **Init** (const **math::Vector3** &_focalPoint)

  Initialze the controller with a focal point.
- void **SetDistance** (float _d)

  Set the distance to the focal point.
- void **SetDistanceRange** (double _minDist, double _maxDist)

  Set the min and max distance from the focal point.
- void **SetFocalPoint** (const **math::Vector3** &_fp)

  Set the focal point.
- void **SetPitch** (double _pitch)

  Set the pitch angle of the camera.
- void **SetYaw** (double _yaw)

  Set the yaw angle of the camera.
- virtual void **Update** ()

  Update.

Static Public Member Functions

- static std::string **GetTypeString** ()

  Get the type name of this view controller.

Additional Inherited Members

10.134.1 Detailed Description

Orbit view controller.

10.134.2 Constructor & Destructor Documentation

10.134.2.1 gazebo::rendering::OrbitViewController::OrbitViewController (UserCameraPtr *_camera*)

Constructor.

Parameters

in	*_camera*	Pointer to the camera to control.

10.134.2.2 virtual gazebo::rendering::OrbitViewController::~OrbitViewController ( ) [virtual]

Destructor.

10.134.3 Member Function Documentation

10.134.3.1 math::Vector3 gazebo::rendering::OrbitViewController::GetFocalPoint ( ) const

Get the focal point.

Returns

The focal point

10.134.3.2 static std::string gazebo::rendering::OrbitViewController::GetTypeString ( ) [static]

Get the type name of this view controller.

Returns

The view controller name: "orbit".

10.134.3.3 virtual void gazebo::rendering::OrbitViewController::HandleKeyPressEvent (const std::string & *_key*) [virtual]

Handle a key press event.

Parameters

in	*_key*	The key that was pressed.

Implements **gazebo::rendering::ViewController** (p. 929).

10.134.3.4 void gazebo::rendering::OrbitViewController::HandleKeyReleaseEvent (const std::string & *_key*) [virtual]

Handle a key release event.

Parameters

in	*_key*	The key that was released.

Implements **gazebo::rendering::ViewController** (p. 929).

10.134.3.5 virtual void gazebo::rendering::OrbitViewController::HandleMouseEvent (const common::MouseEvent & *_event*) [virtual]

Handle a mouse event.

Parameters

in	*_event*	The mouse event.

Implements **gazebo::rendering::ViewController** (p. 929).

10.134.3.6 virtual void gazebo::rendering::OrbitViewController::Init ( ) [virtual]

Initialize the controller.

Implements **gazebo::rendering::ViewController** (p. 930).

10.134.3.7 virtual void gazebo::rendering::OrbitViewController::Init (const math::Vector3 & *_focalPoint*) [virtual]

Initialze the controller with a focal point.

Parameters

in	*_focalPoint*	Point to look at.

Reimplemented from **gazebo::rendering::ViewController** (p. 930).

10.134.3.8 void gazebo::rendering::OrbitViewController::SetDistance (float *_d*)

Set the distance to the focal point.

Parameters

in	*_d*	The distance from the focal point.

10.134.3.9 void gazebo::rendering::OrbitViewController::SetDistanceRange (double *_minDist*, double *_maxDist*)

Set the min and max distance from the focal point.

Parameters

in	*_minDist*	Min distance to the focal point.
in	*_maxDist*	Max distance from the focal point.

10.134.3.10 void gazebo::rendering::OrbitViewController::SetFocalPoint (const math::Vector3 & *_fp*)

Set the focal point.

Parameters

in	*_fp*	The focal point

10.134.3.11 void gazebo::rendering::OrbitViewController::SetPitch (double *_pitch*)

Set the pitch angle of the camera.

Parameters

in	*Pitch*	angle in radians.

10.134.3.12 void gazebo::rendering::OrbitViewController::SetYaw (double *_yaw*)

Set the yaw angle of the camera.

Parameters

in	*Yaw*	angle in radians

10.134.3.13 virtual void gazebo::rendering::OrbitViewController::Update ( ) [virtual]

Update.

Implements **gazebo::rendering::ViewController** (p. 930).

The documentation for this class was generated from the following file:

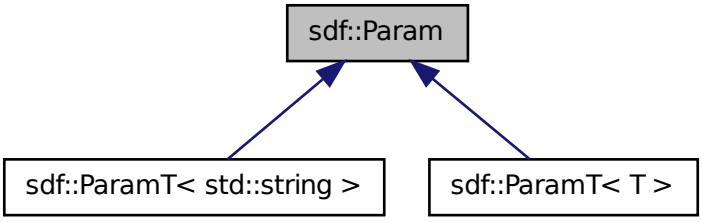
- **OrbitViewController.hh**

10.135 sdf::Param Class Reference

A parameter class.

`#include <Param.hh>`

Inheritance diagram for sdf::Param:



Public Member Functions

- **Param** (**Param** *_newParam)

  Constructor.
- virtual ~**Param** ()

  Destructor.
- virtual **Param** * **Clone** () const =0
- bool **Get** (bool &_value)
- bool **Get** (int &_value)
- bool **Get** (unsigned int &_value)
- bool **Get** (float &_value)
- bool **Get** (double &_value)
- bool **Get** (char &_value)
- bool **Get** (std::string &_value)

- bool **Get** (**gazebo::math::Vector3** &_value)
- bool **Get** (**gazebo::math::Vector2i** &_value)
- bool **Get** (**gazebo::math::Vector2d** &_value)
- bool **Get** (**gazebo::math::Quaternion** &_value)
- bool **Get** (**gazebo::math::Pose** &_value)
- bool **Get** (**gazebo::common::Color** &_value)
- bool **Get** (**gazebo::common::Time** &_value)
- virtual std::string **GetAsString** () const

  Get the type.
- virtual std::string **GetDefaultAsString** () const
- std::string **GetDescription** () const

  Get the description of the parameter.
- const std::string & **GetKey** () const
- bool **GetRequired** () const
- bool **GetSet** () const

  Return true if the parameter has been set.
- std::string **GetTypeName** () const
- bool **IsBool** () const
- bool **IsChar** () const
- bool **IsColor** () const
- bool **IsDouble** () const
- bool **IsFloat** () const
- bool **IsInt** () const
- bool **IsPose** () const
- bool **IsQuaternion** () const
- bool **IsStr** () const
- bool **IsTime** () const
- bool **IsUInt** () const
- bool **IsVector2d** () const
- bool **IsVector2i** () const
- bool **IsVector3** () const
- virtual void **Reset** ()=0

  Reset the parameter.
- bool **Set** (const bool &_value)
- bool **Set** (const int &_value)
- bool **Set** (const unsigned int &_value)
- bool **Set** (const float &_value)
- bool **Set** (const double &_value)
- bool **Set** (const char &_value)
- bool **Set** (const std::string &_value)
- bool **Set** (const char *_value)
- bool **Set** (const **gazebo::math::Vector3** &_value)
- bool **Set** (const **gazebo::math::Vector2i** &_value)
- bool **Set** (const **gazebo::math::Vector2d** &_value)
- bool **Set** (const **gazebo::math::Quaternion** &_value)
- bool **Set** (const **gazebo::math::Pose** &_value)
- bool **Set** (const **gazebo::common::Color** &_value)
- bool **Set** (const **gazebo::common::Time** &_value)
- void **SetDescription** (const std::string &_desc)

  Set the description of the parameter.

- virtual bool **SetFromString** (const std::string &)

  Set the parameter value from a string.
- template<typename T >
  void **SetUpdateFunc** (T _updateFunc)

  Update function.
- virtual void **Update** ()=0

Protected Attributes

- std::string **description**
- std::string **key**
- bool **required**
- bool **set**
- std::string **typeName**
- boost::function< boost::any()> **updateFunc**

10.135.1 Detailed Description

A parameter class.

10.135.2 Constructor & Destructor Documentation

10.135.2.1 sdf::Param::Param (Param * *_newParam*)

Constructor.

10.135.2.2 virtual sdf::Param::~Param ( ) [virtual]

Destructor.

10.135.3 Member Function Documentation

10.135.3.1 virtual Param* sdf::Param::Clone ( ) const [pure virtual]

Implemented in **sdf::ParamT< T >** (p. 650), and **sdf::ParamT< std::string >** (p. 650).

10.135.3.2 bool sdf::Param::Get (bool & *_value*)

10.135.3.3 bool sdf::Param::Get (int & *_value*)

10.135.3.4 bool sdf::Param::Get (unsigned int & *_value*)

10.135.3.5 bool sdf::Param::Get (float & *_value*)

10.135.3.6 bool sdf::Param::Get (double & *_value*)

10.135.3.7 bool sdf::Param::Get (char & *_value*)

10.135.3.8 bool sdf::Param::Get (std::string & *_value*)

10.135.3.9 bool sdf::Param::Get (gazebo::math::Vector3 & *_value*)

10.135.3.10 bool sdf::Param::Get (gazebo::math::Vector2i & *_value*)

10.135.3.11 bool sdf::Param::Get (gazebo::math::Vector2d & *_value*)

10.135.3.12 bool sdf::Param::Get (gazebo::math::Quaternion & *_value*)

10.135.3.13 bool sdf::Param::Get (gazebo::math::Pose & *_value*)

10.135.3.14 bool sdf::Param::Get (gazebo::common::Color & *_value*)

10.135.3.15 bool sdf::Param::Get (gazebo::common::Time & *_value*)

10.135.3.16 virtual std::string sdf::Param::GetAsString ( ) const [inline], [virtual]

Get the type.

Reimplemented in **sdf::ParamT< T >** (p. 650), and **sdf::ParamT< std::string >** (p. 650).

10.135.3.17 virtual std::string sdf::Param::GetDefaultAsString ( ) const [inline], [virtual]

Reimplemented in **sdf::ParamT< T >** (p. 650), and **sdf::ParamT< std::string >** (p. 650).

10.135.3.18 std::string sdf::Param::GetDescription ( ) const

Get the description of the parameter.

10.135.3.19 const std::string& sdf::Param::GetKey ( ) const [inline]

References key.

Referenced by sdf::ParamT< std::string >::Clone(), and sdf::ParamT< std::string >::Set().

10.135.3.20 bool sdf::Param::GetRequired ( ) const [inline]

References required.

10.135.3.21 bool sdf::Param::GetSet ( ) const [inline]

Return true if the parameter has been set.

10.135.3.22 std::string sdf::Param::GetTypeName ( ) const

10.135.3.23 bool sdf::Param::IsBool ( ) const

10.135.3.24 bool sdf::Param::IsChar ( ) const

10.135.3.25 bool sdf::Param::IsColor ( ) const

10.135.3.26 bool sdf::Param::IsDouble ( ) const

10.135.3.27 bool sdf::Param::IsFloat ( ) const

10.135.3.28 bool sdf::Param::IsInt ( ) const

10.135.3.29 bool sdf::Param::IsPose ( ) const

10.135.3.30 bool sdf::Param::IsQuaternion ( ) const

10.135.3.31 bool sdf::Param::IsStr ( ) const

10.135.3.32 bool sdf::Param::IsTime ( ) const

10.135.3.33 bool sdf::Param::IsUInt ( ) const

10.135.3.34 bool sdf::Param::IsVector2d ( ) const

10.135.3.35 bool sdf::Param::IsVector2i ( ) const

10.135.3.36 bool sdf::Param::IsVector3 ( ) const

10.135.3.37 virtual void sdf::Param::Reset ( ) [pure virtual]

Reset the parameter.

Implemented in **sdf::ParamT< T >** (p. 650), and **sdf::ParamT< std::string >** (p. 650).

10.135.3.38 bool sdf::Param::Set (const bool & *_value*)

Referenced by sdf::ParamT< std::string >::Update().

10.135.3.39 bool sdf::Param::Set (const int & *_value*)

10.135.3.40 bool sdf::Param::Set (const unsigned int & *_value*)

10.135.3.41 bool sdf::Param::Set (const float & *_value*)

10.135.3.42 bool sdf::Param::Set (const double & *_value*)

10.135.3.43 bool sdf::Param::Set (const char & *_value*)

10.135.3.44 bool sdf::Param::Set (const std::string & *_value*)

10.135.3.45 bool sdf::Param::Set (const char * *_value*)

10.135.3.46 bool sdf::Param::Set (const gazebo::math::Vector3 & *_value*)

10.135.3.47 bool sdf::Param::Set (const gazebo::math::Vector2i & *_value*)

10.135.3.48 bool sdf::Param::Set (const gazebo::math::Vector2d & *_value*)

10.135.3.49 bool sdf::Param::Set (const gazebo::math::Quaternion & *_value*)

10.135.3.50 bool sdf::Param::Set (const gazebo::math::Pose & *_value*)

10.135.3.51 bool sdf::Param::Set (const gazebo::common::Color & *_value*)

10.135.3.52 bool sdf::Param::Set (const gazebo::common::Time & *_value*)

10.135.3.53 void sdf::Param::SetDescription (const std::string & *_desc*)

Set the description of the parameter.

10.135.3.54 virtual bool sdf::Param::SetFromString (const std::string &) [inline], [virtual]

Set the parameter value from a string.

Reimplemented in **sdf::ParamT< T >** (p. 650), and **sdf::ParamT< std::string >** (p. 650).

10.135.3.55 template<typename T > void sdf::Param::SetUpdateFunc (T *_updateFunc*) [inline]

Update function.

References updateFunc.

10.135.3.56 virtual void sdf::Param::Update ( ) [pure virtual]

Implemented in **sdf::ParamT< T >** (p. 651), and **sdf::ParamT< std::string >** (p. 651).

10.135.4 Member Data Documentation

10.135.4.1 std::string sdf::Param::description [protected]

Referenced by sdf::ParamT< std::string >::Clone(), and sdf::ParamT< std::string >::ParamT().

10.135.4.2 std::string sdf::Param::key [protected]

Referenced by GetKey(), sdf::ParamT< std::string >::ParamT(), and sdf::ParamT< std::string >::Set().

10.135.4.3 bool sdf::Param::required [protected]

Referenced by sdf::ParamT< std::string >::Clone(), GetRequired(), sdf::ParamT< std::string >::ParamT(), and sdf::ParamT< std::string >::Set().

10.135.4.4 bool sdf::Param::set [protected]

10.135.4.5 std::string sdf::Param::typeName [protected]

Referenced by sdf::ParamT< std::string >::Clone(), and sdf::ParamT< std::string >::ParamT().

10.135.4.6 boost::function<boost::any ()> sdf::Param::updateFunc [protected]

Referenced by SetUpdateFunc(), and sdf::ParamT< std::string >::Update().

The documentation for this class was generated from the following file:

- **Param.hh**

10.136 ParamT< T > Class Template Reference

`#include <CommonTypes.hh>`

The documentation for this class was generated from the following file:

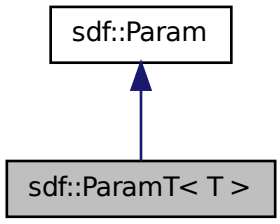
- **CommonTypes.hh**

10.137 sdf::ParamT< T > Class Template Reference

Templatized parameter class.

`#include <Param.hh>`

Inheritance diagram for sdf::ParamT< T >:



Public Member Functions

- **ParamT** (const std::string &_key, const std::string &_default, bool _required, const std::string &_typeName="", const std::string &_description="")

  Constructor.
- virtual ~**ParamT** ()

  Destructor.

- virtual **Param** * **Clone** () const
- virtual std::string **GetAsString** () const

  Get the parameter value as a string.
- virtual std::string **GetDefaultAsString** () const
- T **GetDefaultValue** () const

  Get the value.
- T **GetValue** () const

  Get the value.
- T **operator*** () const
- virtual void **Reset** ()

  Reset to default value.
- virtual bool **Set** (const std::string &_str)

  Set the parameter value from a string.
- virtual bool **SetFromString** (const std::string &_value)

  Set the parameter value from a string.
- void **SetValue** (const T &_value)

  Set the value of the parameter.
- virtual void **Update** ()

Protected Attributes

- T **defaultValue**
- T **value**

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **ParamT**< T > &_p)

10.137.1 Detailed Description

template<typename T>class sdf::ParamT< T >

Templatized parameter class.

10.137.2 Constructor & Destructor Documentation

10.137.2.1 template<typename T> sdf::ParamT< T >::ParamT (const std::string & *_key,* const std::string & *_default,* bool *_required,* const std::string & *_typeName = " ",* const std::string & *_description = " ")* [inline]

Constructor.

10.137.2.2 template<typename T> virtual sdf::ParamT< T >::~ParamT ( ) [inline], [virtual]

Destructor.

10.137.3 Member Function Documentation

10.137.3.1 template<typename T> virtual Param* sdf::ParamT< T >::Clone ( ) const [inline], [virtual]

Implements **sdf::Param** (p. 644).

10.137.3.2 template<typename T> virtual std::string sdf::ParamT< T >::GetAsString ( ) const [inline], [virtual]

Get the parameter value as a string.

Reimplemented from **sdf::Param** (p. 645).

Referenced by sdf::ParamT< std::string >::Clone().

10.137.3.3 template<typename T> virtual std::string sdf::ParamT< T >::GetDefaultAsString ( ) const [inline], [virtual]

Reimplemented from **sdf::Param** (p. 645).

10.137.3.4 template<typename T> T sdf::ParamT< T >::GetDefaultValue ( ) const [inline]

Get the value.

10.137.3.5 template<typename T> T sdf::ParamT< T >::GetValue ( ) const [inline]

Get the value.

10.137.3.6 template<typename T> T sdf::ParamT< T >::operator* ( ) const [inline]

10.137.3.7 template<typename T> virtual void sdf::ParamT< T >::Reset ( ) [inline], [virtual]

Reset to default value.

Implements **sdf::Param** (p. 646).

10.137.3.8 template<typename T> virtual bool sdf::ParamT< T >::Set (const std::string & *_str*) [inline], [virtual]

Set the parameter value from a string.

Referenced by sdf::ParamT< std::string >::ParamT(), and sdf::ParamT< std::string >::SetFromString().

10.137.3.9 template<typename T> virtual bool sdf::ParamT< T >::SetFromString (const std::string & *_value*) [inline], [virtual]

Set the parameter value from a string.

Reimplemented from **sdf::Param** (p. 647).

10.137.3.10 template<typename T> void sdf::ParamT< T >::SetValue (const T & _value) [inline]

Set the value of the parameter.

10.137.3.11 template<typename T> virtual void sdf::ParamT< T >::Update ( ) [inline], [virtual]

Implements **sdf::Param** (p. 647).

10.137.4 Friends And Related Function Documentation

10.137.4.1 template<typename T> std::ostream& operator<< (std::ostream & _out, const ParamT< T > & _p) [friend]

10.137.5 Member Data Documentation

10.137.5.1 template<typename T> T sdf::ParamT< T >::defaultValue [protected]

Referenced by sdf::ParamT< std::string >::GetDefaultAsString(), sdf::ParamT< std::string >::GetDefaultValue(), sdf::ParamT< std::string >::ParamT(), sdf::ParamT< std::string >::Reset(), and sdf::ParamT< std::string >::Set().

10.137.5.2 template<typename T> T sdf::ParamT< T >::value [protected]

Referenced by sdf::ParamT< std::string >::GetAsString(), sdf::ParamT< std::string >::GetValue(), sdf::ParamT< std::string >::operator*(), sdf::ParamT< std::string >::ParamT(), sdf::ParamT< std::string >::Reset(), sdf::ParamT< std::string >::Set(), and sdf::ParamT< std::string >::SetValue().

The documentation for this class was generated from the following file:

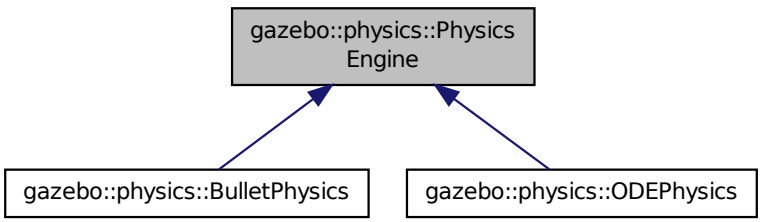
- **Param.hh**

10.138 gazebo::physics::PhysicsEngine Class Reference

Base (p. 145) class for a physics engine.

`#include <physics/physics.hh>`

Inheritance diagram for gazebo::physics::PhysicsEngine:



Public Member Functions

- **PhysicsEngine** (**WorldPtr** _world)

  Default constructor.
- virtual ~**PhysicsEngine** ()

  Destructor.
- virtual **CollisionPtr CreateCollision** (const std::string &_shapeType, **LinkPtr** _link)=0

  Create a collision.
- **CollisionPtr CreateCollision** (const std::string &_shapeType, const std::string &_linkName)

  Create a collision.
- virtual **JointPtr CreateJoint** (const std::string &_type, **ModelPtr** _parent)=0

  Create a new joint.
- virtual **LinkPtr CreateLink** (**ModelPtr** _parent)=0

  Create a new body.
- virtual **ShapePtr CreateShape** (const std::string &_shapeType, **CollisionPtr** _collision)=0

  *Create a **physics::Shape** (p. 779) object.*
- virtual void **DebugPrint** () const =0

  Debug print out of the physic engine state.
- virtual void **Fini** ()

  Finilize the physics engine.
- virtual bool **GetAutoDisableFlag** ()

  : Remove this function, and replace it with a more generic property map
- virtual double **GetContactMaxCorrectingVel** ()

  : Remove this function, and replace it with a more generic property map.
- virtual double **GetContactSurfaceLayer** ()

  : Remove this function, and replace it with a more generic property map.
- **math::Vector3 GetGravity** () const

  Return the gavity vector.
- virtual int **GetMaxContacts** ()

  : Remove this function, and replace it with a more generic property map.
- boost::recursive_mutex * **GetPhysicsUpdateMutex** () const

  *returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 663).*
- virtual int **GetSORPGSIters** ()

  : Remove this function, and replace it with a more generic property map
- virtual int **GetSORPGSPreconIters** ()

  : Remove this function, and replace it with a more generic property map
- virtual double **GetSORPGSW** ()

  : Remove this function, and replace it with a more generic property map.
- virtual double **GetStepTime** ()=0

  Get the simulation step time.
- double **GetUpdatePeriod** ()

  Get the simulation update period.
- double **GetUpdateRate** ()

  Get the simulation update rate.
- virtual double **GetWorldCFM** ()

  : Remove this function, and replace it with a more generic property map
- virtual double **GetWorldERP** ()

: Remove this function, and replace it with a more generic property map

- virtual void **Init** ()=0

  Initialize the physics engine.

- virtual void **InitForThread** ()=0

  Init the engine for threads.

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the physics engine.

- virtual void **Reset** ()

  Rest the physics engine.

- virtual void **SetAutoDisableFlag** (bool _autoDisable)

  : Remove this function, and replace it with a more generic property map

- virtual void **SetContactMaxCorrectingVel** (double _vel)

  : Remove this function, and replace it with a more generic property map

- virtual void **SetContactSurfaceLayer** (double _layerDepth)

  : Remove this function, and replace it with a more generic property map

- virtual void **SetGravity** (const **gazebo::math::Vector3** &_gravity)=0

  Set the gavity vector.

- virtual void **SetMaxContacts** (double _maxContacts)

  : Remove this function, and replace it with a more generic property map

- virtual void **SetSORPGSIters** (unsigned int _iters)

  : Remove this function, and replace it with a more generic property map

- virtual void **SetSORPGSPreconIters** (unsigned int _iters)

  : Remove this function, and replace it with a more generic property map

- virtual void **SetSORPGSW** (double _w)

  : Remove this function, and replace it with a more generic property map

- virtual void **SetStepTime** (double _value)=0

  Set the simulation step time.

- void **SetUpdateRate** (double _value)

  Set the simulation update rate.

- virtual void **SetWorldCFM** (double _cfm)

  : Remove this function, and replace it with a more generic property map

- virtual void **SetWorldERP** (double _erp)

  : Remove this function, and replace it with a more generic property map

- virtual void **UpdateCollision** ()=0

  Update the physics engine collision.

- virtual void **UpdatePhysics** ()

  Update the physics engine.

Protected Member Functions

- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &_msg)

  virtual callback for gztopic "~/physics".

- virtual void **OnRequest** (ConstRequestPtr &_msg)

  virtual callback for gztopic "~/request".

Protected Attributes

- **transport::PublisherPtr contactPub**

  Contact *(p. 296) publisher.*
- **transport::NodePtr node**

  Node for communication.
- **transport::SubscriberPtr physicsSub**

  Subscribe to the physics topic.
- boost::recursive_mutex * **physicsUpdateMutex**

  Mutex to protect the update cycle.
- **transport::SubscriberPtr requestSub**

  Subscribe to the request topic.
- **transport::PublisherPtr responsePub**

  Response publisher.
- **sdf::ElementPtr sdf**

  Our SDF values.
- **WorldPtr world**

  Pointer to the world.

10.138.1 Detailed Description

Base (p. 145) class for a physics engine.

10.138.2 Constructor & Destructor Documentation

10.138.2.1 gazebo::physics::PhysicsEngine::PhysicsEngine (WorldPtr _world) [explicit]

Default constructor.

Parameters

in	*_world*	Pointer to the world.

10.138.2.2 virtual gazebo::physics::PhysicsEngine::~PhysicsEngine ( ) [virtual]

Destructor.

10.138.3 Member Function Documentation

10.138.3.1 virtual CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & _shapeType, LinkPtr _link) [pure virtual]

Create a collision.

Parameters

in	*_shapeType*	Type of collision to create.
in	*_link*	Parent link.

Implemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.2 CollisionPtr gazebo::physics::PhysicsEngine::CreateCollision (const std::string & *_shapeType,* const std::string & *_linkName*)

Create a collision.

Parameters

in	*_shapeType*	Type of collision to create.
in	*_linkName*	Name of the parent link.

10.138.3.3 virtual JointPtr gazebo::physics::PhysicsEngine::CreateJoint (const std::string & *_type,* ModelPtr *_parent*) [pure virtual]

Create a new joint.

Parameters

in	*_type*	Type of joint to create.
in	*_parent*	**Model** (p. 521) parent.

Implemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.4 virtual LinkPtr gazebo::physics::PhysicsEngine::CreateLink (ModelPtr *_parent*) [pure virtual]

Create a new body.

Parameters

in	*_parent*	Parent model for the link.

Implemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.5 virtual ShapePtr gazebo::physics::PhysicsEngine::CreateShape (const std::string & *_shapeType,* CollisionPtr *_collision*) [pure virtual]

Create a **physics::Shape** (p. 779) object.

Parameters

in	*_shapeType*	Type of shape to create.
in	*_collision*	**Collision** (p. 262) parent.

Implemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.6 virtual void gazebo::physics::PhysicsEngine::DebugPrint ( ) const [pure virtual]

Debug print out of the physic engine state.

Implemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.7 virtual void gazebo::physics::PhysicsEngine::Fini ( ) [virtual]

Finilize the physics engine.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 613), and **gazebo::physics::BulletPhysics** (p. 205).

10.138.3.8 virtual bool gazebo::physics::PhysicsEngine::GetAutoDisableFlag ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

access functions to set ODE parameters..

Returns

Auto disable flag.

10.138.3.9 virtual double gazebo::physics::PhysicsEngine::GetContactMaxCorrectingVel ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters.

Returns

Max correcting velocity.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.10 virtual double gazebo::physics::PhysicsEngine::GetContactSurfaceLayer ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters.

Returns

Contact (p. 296) suerface layer depth.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.11 math::Vector3 gazebo::physics::PhysicsEngine::GetGravity ( ) const

Return the gavity vector.

Returns

The gavity vector.

10.138.3.12 virtual int gazebo::physics::PhysicsEngine::GetMaxContacts ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.

access functions to set ODE parameters.

Returns

Maximum number of allows contacts.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.13 boost::recursive_mutex* gazebo::physics::PhysicsEngine::GetPhysicsUpdateMutex ( ) const [inline]

returns a pointer to the **PhysicsEngine::physicsUpdateMutex** (p. 663).

Returns

Pointer to the physics mutex.

References physicsUpdateMutex.

10.138.3.14 virtual int gazebo::physics::PhysicsEngine::GetSORPGSIters ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map
access functions to set ODE parameters.

Returns

SORPGS iterations.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.15 virtual int gazebo::physics::PhysicsEngine::GetSORPGSPreconIters ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map
access functions to set ODE parameters.

Returns

SORPGS precondition iterations.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.16 virtual double gazebo::physics::PhysicsEngine::GetSORPGSW ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map.
access functions to set ODE parameters

Returns

SORPGSW value.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 614).

10.138.3.17 virtual double gazebo::physics::PhysicsEngine::GetStepTime ( ) [pure virtual]

Get the simulation step time.

Returns

Simulation step time.

Implemented in **gazebo::physics::ODEPhysics** (p. 614), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.18 double gazebo::physics::PhysicsEngine::GetUpdatePeriod ( )

Get the simulation update period.

Returns

Simulation update period.

10.138.3.19 double gazebo::physics::PhysicsEngine::GetUpdateRate ( )

Get the simulation update rate.

Returns

Update rate.

10.138.3.20 virtual double gazebo::physics::PhysicsEngine::GetWorldCFM ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 954) CFM.

Returns

World (p. 954) CFM.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 615).

10.138.3.21 virtual double gazebo::physics::PhysicsEngine::GetWorldERP ( ) [inline], [virtual]

: Remove this function, and replace it with a more generic property map

Get **World** (p. 954) ERP.

Returns

World (p. 954) ERP.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 615).

10.138.3.22 virtual void gazebo::physics::PhysicsEngine::Init ( ) [pure virtual]

Initialize the physics engine.

Implemented in **gazebo::physics::ODEPhysics** (p. 615), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.23 virtual void gazebo::physics::PhysicsEngine::InitForThread ( ) [pure virtual]

Init the engine for threads.

Implemented in **gazebo::physics::ODEPhysics** (p. 615), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.24 virtual void gazebo::physics::PhysicsEngine::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the physics engine.

Parameters

in	*_sdf*	Pointer to the SDF parameters.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 615), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.25 virtual void gazebo::physics::PhysicsEngine::OnPhysicsMsg (ConstPhysicsPtr & *_msg*) [protected], [virtual]

virtual callback for gztopic "~/physics".

Parameters

in	*_msg*	Physics message.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 615).

10.138.3.26 virtual void gazebo::physics::PhysicsEngine::OnRequest (ConstRequestPtr & *_msg*) [protected], [virtual]

virtual callback for gztopic "~/request".

Parameters

in	*_msg*	Request message.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 615).

10.138.3.27 virtual void gazebo::physics::PhysicsEngine::Reset ( ) [inline], [virtual]

Rest the physics engine.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 616).

10.138.3.28 virtual void gazebo::physics::PhysicsEngine::SetAutoDisableFlag (bool *_autoDisable*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_autoDisable*	True to enable auto disabling of bodies.

10.138.3.29 virtual void gazebo::physics::PhysicsEngine::SetContactMaxCorrectingVel (double *_vel*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_vel*	Max correcting velocity.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 616).

10.138.3.30 virtual void gazebo::physics::PhysicsEngine::SetContactSurfaceLayer (double *_layerDepth*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_layerDepth*	Surface layer depth

Reimplemented in **gazebo::physics::ODEPhysics** (p. 616).

10.138.3.31 virtual void gazebo::physics::PhysicsEngine::SetGravity (const gazebo::math::Vector3 & *_gravity*) [pure virtual]

Set the gavity vector.

Parameters

in	*_gravity*	New gravity vector.

Implemented in **gazebo::physics::ODEPhysics** (p. 616), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.32 virtual void gazebo::physics::PhysicsEngine::SetMaxContacts (double *_maxContacts*) [virtual]

: Remove this function, and replace it with a more generic property map

access functions to set ODE parameters

Parameters

in	*_maxContacts*	Maximum number of contacts.

10.138.3.33 virtual void gazebo::physics::PhysicsEngine::SetSORPGSIters (unsigned int *_iters*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_iter*	Number of iterations.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 616).

10.138.3.34 virtual void gazebo::physics::PhysicsEngine::SetSORPGSPreconIters (unsigned int *_iters*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_iter*	Number of iterations.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 616).

10.138.3.35 virtual void gazebo::physics::PhysicsEngine::SetSORPGSW (double *_w*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_w*	SORPGSW value.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 616).

10.138.3.36 virtual void gazebo::physics::PhysicsEngine::SetStepTime (double *_value*) [pure virtual]

Set the simulation step time.

Parameters

in	*_value*	Value of the step time.

Implemented in **gazebo::physics::ODEPhysics** (p. 617), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.37 void gazebo::physics::PhysicsEngine::SetUpdateRate (double *_value*)

Set the simulation update rate.

Parameters

in	*_value*	Value of the update rate.

10.138.3.38 virtual void gazebo::physics::PhysicsEngine::SetWorldCFM (double *_cfm*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_cfm*	Constraint force mixing.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 617).

10.138.3.39 virtual void gazebo::physics::PhysicsEngine::SetWorldERP (double *_erp*) [virtual]

: Remove this function, and replace it with a more generic property map

Access functions to set ODE parameters.

Parameters

in	*_erp*	Error reduction parameter.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 617).

10.138.3.40 virtual void gazebo::physics::PhysicsEngine::UpdateCollision ( ) [pure virtual]

Update the physics engine collision.

Implemented in **gazebo::physics::ODEPhysics** (p. 617), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.3.41 virtual void gazebo::physics::PhysicsEngine::UpdatePhysics ( ) [inline], [virtual]

Update the physics engine.

Reimplemented in **gazebo::physics::ODEPhysics** (p. 617), and **gazebo::physics::BulletPhysics** (p. 206).

10.138.4 Member Data Documentation

10.138.4.1 transport::PublisherPtr gazebo::physics::PhysicsEngine::contactPub [protected]

Contact (p. 296) publisher.

10.138.4.2 transport::NodePtr gazebo::physics::PhysicsEngine::node [protected]

Node for communication.

10.138.4.3 transport::SubscriberPtr gazebo::physics::PhysicsEngine::physicsSub [protected]

Subscribe to the physics topic.

10.138.4.4 boost::recursive_mutex* gazebo::physics::PhysicsEngine::physicsUpdateMutex [protected]

Mutex to protect the update cycle.

Referenced by GetPhysicsUpdateMutex().

10.138.4.5 transport::SubscriberPtr gazebo::physics::PhysicsEngine::requestSub [protected]

Subscribe to the request topic.

10.138.4.6 transport::PublisherPtr gazebo::physics::PhysicsEngine::responsePub [protected]

Response publisher.

10.138.4.7 sdf::ElementPtr gazebo::physics::PhysicsEngine::sdf [protected]

Our SDF values.

10.138.4.8 WorldPtr gazebo::physics::PhysicsEngine::world [protected]

Pointer to the world.

The documentation for this class was generated from the following file:

- **PhysicsEngine.hh**

10.139 gazebo::physics::PhysicsFactory Class Reference

The physics factory instantiates different physics engines.

`#include <physics/physics.hh>`

Static Public Member Functions

- static **PhysicsEnginePtr NewPhysicsEngine** (const std::string &_className, **WorldPtr** _world)
  Create a new instance of a physics engine.
- static void **RegisterAll** ()
  Register everything.
- static void **RegisterPhysicsEngine** (std::string _className, **PhysicsFactoryFn** _factoryfn)
  Register a physics class.

10.139.1 Detailed Description

The physics factory instantiates different physics engines.

10.139.2 Member Function Documentation

10.139.2.1 static PhysicsEnginePtr gazebo::physics::PhysicsFactory::NewPhysicsEngine (const std::string & *_className,* WorldPtr *_world*) [static]

Create a new instance of a physics engine.

Parameters

in	*_className*	Name of the physics class.
in	*_world*	**World** (p. 954) to pass to the created physics engine.

10.139.2.2 static void gazebo::physics::PhysicsFactory::RegisterAll ( ) [static]

Register everything.

10.139.2.3 static void gazebo::physics::PhysicsFactory::RegisterPhysicsEngine (std::string *_className,* PhysicsFactoryFn *_factoryfn*) [static]

Register a physics class.

Parameters

in	*_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**

10.140 gazebo::common::PID Class Reference

Generic **PID** (p. 664) controller class.

`#include <PID.hh>`

Public Member Functions

- **PID** (double _p=0.0, double _i=0.0, double _d=0.0, double _imax=0.0, double _imin=0.0, double _cmdMax=0.0, double _cmdMin=0.0)

  Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[I1:I2].
- virtual ~**PID** ()

  Destructor.
- double **GetCmd** ()

  *Return current command for this **PID** (p. 664) controller.*
- void **GetErrors** (double &_pe, double &_ie, double &_de)

  *Return **PID** (p. 664) error terms for the controller.*
- void **Init** (double _p=0.0, double _i=0.0, double _d=0.0, double _imax=0.0, double _imin=0.0, double _cmdMax=0.0, double _cmdMin=0.0)

Initialize PID-gains and integral term limits:[iMax:iMin]-[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. 664) controller.*
- void **SetCmdMax** (double _c)

  Set the maximum value for the command.
- void **SetCmdMin** (double _c)

  Set the maximum value for the command.
- void **SetDGain** (double _d)

  Set the derivtive Gain.
- void **SetIGain** (double _i)

  Set the integral Gain.
- void **SetIMax** (double _i)

  Set the integral upper limit.
- void **SetIMin** (double _i)

  Set the integral lower limit.
- void **SetPGain** (double _p)

  Set the proportional Gain.
- double **Update** (double _error, **common::Time** _dt)

  Update the Pid loop with nonuniform time step size.

10.140.1 Detailed Description

Generic **PID** (p. 664) controller class.

Generic 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.

10.140.2 Constructor & Destructor Documentation

10.140.2.1 gazebo::common::PID::PID (double _p = 0.0, double _i = 0.0, double _d = 0.0, double _imax = 0.0, double _imin = 0.0, double _cmdMax = 0.0, double _cmdMin = 0.0)

Constructor, zeros out Pid values when created and initialize Pid-gains and integral term limits:[iMax:iMin]-[I1:I2].

Parameters

in	*_p*	The proportional gain.
in	*_i*	The integral gain.
in	*_d*	The derivative gain.
in	*_imax*	The integral upper limit.
in	*_imin*	The integral lower limit.

10.140.2.2 virtual gazebo::common::PID::~PID ( ) [virtual]

Destructor.

10.140.3 Member Function Documentation

10.140.3.1 double gazebo::common::PID::GetCmd ( )

Return current command for this **PID** (p. 664) controller.

Returns

the command value

10.140.3.2 void gazebo::common::PID::GetErrors (double & *_pe*, double & *_ie*, double & *_de*)

Return **PID** (p. 664) error terms for the controller.

Parameters

in	*_pe*	The proportional error.
in	*_ie*	The integral error.
in	*_de*	The derivative error.

10.140.3.3 void gazebo::common::PID::Init (double *_p* = 0.0, double *_i* = 0.0, double *_d* = 0.0, double *_imax* = 0.0, double *_imin* = 0.0, double *_cmdMax* = 0.0, double *_cmdMin* = 0.0)

Initialize PID-gains and integral term limits:[iMax:iMin]-[I1:I2].

Parameters

in	*_p*	The proportional gain.
in	*_i*	The integral gain.
in	*_d*	The derivative gain.
in	*_imax*	The integral upper limit.
in	*_imin*	The integral lower limit.

10.140.3.4 PID& gazebo::common::PID::operator= (const PID & *_p*) [inline]

Assignment operator.

Parameters

in	*_p*	a reference to a **PID** (p. 664) to assign values from

Returns

reference to this instance

References Reset().

10.140.3.5 void gazebo::common::PID::Reset ( )

Reset the errors and command.

Referenced by operator=().

10.140.3.6 void gazebo::common::PID::SetCmd (double _cmd)

Set current target command for this **PID** (p. 664) controller.

Parameters

in	*_cmd*	New command

10.140.3.7 void gazebo::common::PID::SetCmdMax (double _c)

Set the maximum value for the command.

Parameters

in	*_c*	The maximum value

10.140.3.8 void gazebo::common::PID::SetCmdMin (double _c)

Set the maximum value for the command.

Parameters

in	*_c*	The maximum value

10.140.3.9 void gazebo::common::PID::SetDGain (double _d)

Set the derivtive Gain.

Parameters

in	*_p*	dertivative gain value

10.140.3.10 void gazebo::common::PID::SetIGain (double _i)

Set the integral Gain.

Parameters

in	*_p*	integral gain value

10.140.3.11 void gazebo::common::PID::SetIMax (double _i)

Set the integral upper limit.

Parameters

in	*_p*	integral upper limit value

10.140.3.12 void gazebo::common::PID::SetIMin (double *_i*)

Set the integral lower limit.

Parameters

in	*_p*	integral lower limit value

10.140.3.13 void gazebo::common::PID::SetPGain (double *_p*)

Set the proportional Gain.

Parameters

in	*_p*	proportional gain value

10.140.3.14 double gazebo::common::PID::Update (double *_error,* common::Time *_dt*)

Update the Pid loop with nonuniform time step size.

Parameters

_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**

10.141 gazebo::math::Plane Class Reference

A plane and related functions.

`#include <Plane.hh>`

Public Member Functions

- **Plane** ()

  Constructor.
- **Plane** (const **Vector3** &_normal, double _offset=0.0)

Constructor from a normal and a 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. *668) offset.*
- **Vector3 normal**

  Plane *(*p. *668) normal.*
- **Vector2d size**

  Plane *(*p. *668) size.*

10.141.1 Detailed Description

A plane and related functions.

10.141.2 Constructor & Destructor Documentation

10.141.2.1 gazebo::math::Plane::Plane ( )

Constructor.

10.141.2.2 gazebo::math::Plane::Plane (const Vector3 & *_normal,* double *_offset* = 0.0)

Constructor from a normal and a distanec.

Parameters

in	*_normal*	The plane normal
in	*_offset*	Offset along the normal

10.141.2.3 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

10.141.2.4 virtual gazebo::math::Plane::~Plane ( ) [virtual]

Destructor.

10.141.3 Member Function Documentation

10.141.3.1 double gazebo::math::Plane::Distance (const Vector3 & *_origin,* const Vector3 & *_dir*) const

Get distance to the plane give an origin and direction.

Parameters

in	*_origin*	the origin
in	*_dir*	a direction

Returns

the shortest distance

10.141.3.2 Plane& gazebo::math::Plane::operator= (const Plane & *_p*)

Equal operator.

Parameters

_p	another plane

Returns

itself

10.141.3.3 void gazebo::math::Plane::Set (const Vector3 & *_normal,* const Vector2d & *_size,* double *offset*)

Set the plane.

Parameters

in	*_normal*	The plane normal
in	*_size*	Size of the plane
in	*_offset*	Offset along the normal

10.141.4 Member Data Documentation

10.141.4.1 double gazebo::math::Plane::d

Plane (p. 668) offset.

10.141.4.2 Vector3 gazebo::math::Plane::normal

Plane (p. 668) normal.

10.141.4.3 Vector2d gazebo::math::Plane::size

Plane (p. 668) size.

The documentation for this class was generated from the following file:

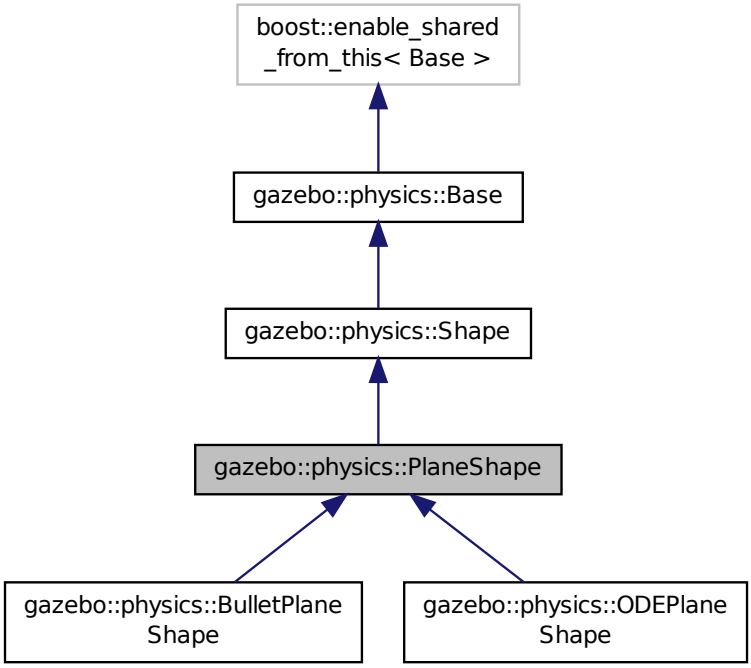
- **Plane.hh**

10.142 gazebo::physics::PlaneShape Class Reference

Collision (p. 262) for an infinite plane.

`#include <PlaneShape.hh>`

Inheritance diagram for gazebo::physics::PlaneShape:



Public Member Functions

- **PlaneShape** (**CollisionPtr parent**)

  Constructor.
- virtual ∼**PlaneShape** ()

  Destructor.
- virtual void **CreatePlane** ()

  Create the plane.
- void **FillMsg** (msgs::Geometry &_msg)
- **math::Vector3 GetNormal** () const
- **math::Vector2d GetSize** () const

  Get the size.
- virtual void **Init** ()

  Initialize the plane.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
- virtual void **SetAltitude** (const **math::Vector3** &_pos)

  Set the altitude of the plane.
- void **SetNormal** (const **math::Vector3** &_norm)

  Set the normal.
- void **SetSize** (const **math::Vector2d** &_size)

  Set the size.

Additional Inherited Members

10.142.1 Detailed Description

Collision (p. 262) for an infinite plane.

This collision is used primarily for ground planes. Note that while the plane in infinite, only the part near the camera is drawn.

10.142.2 Constructor & Destructor Documentation

10.142.2.1 gazebo::physics::PlaneShape::PlaneShape (CollisionPtr *parent*)

Constructor.

Parameters

body	**Link** (p. 454) to which we are attached.

10.142.2.2 virtual gazebo::physics::PlaneShape::∼PlaneShape ( ) [virtual]

Destructor.

10.142.3 Member Function Documentation

10.142.3.1 virtual void gazebo::physics::PlaneShape::CreatePlane ( ) [virtual]

Create the plane.

Reimplemented in **gazebo::physics::BulletPlaneShape** (p. 208), and **gazebo::physics::ODEPlaneShape** (p. 619).

Referenced by gazebo::physics::ODEPlaneShape::CreatePlane(), and gazebo::physics::BulletPlaneShape::CreatePlane().

10.142.3.2 void gazebo::physics::PlaneShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Implements **gazebo::physics::Shape** (p. 781).

10.142.3.3 math::Vector3 gazebo::physics::PlaneShape::GetNormal ( ) const

Referenced by gazebo::physics::ODEPlaneShape::CreatePlane(), and gazebo::physics::BulletPlaneShape::CreatePlane().

10.142.3.4 math::Vector2d gazebo::physics::PlaneShape::GetSize ( ) const

Get the size.

10.142.3.5 virtual void gazebo::physics::PlaneShape::Init ( ) [virtual]

Initialize the plane.

Implements **gazebo::physics::Shape** (p. 781).

10.142.3.6 virtual void gazebo::physics::PlaneShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Implements **gazebo::physics::Shape** (p. 782).

10.142.3.7 virtual void gazebo::physics::PlaneShape::SetAltitude (const math::Vector3 & *_pos*) [virtual]

Set the altitude of the plane.

Reimplemented in **gazebo::physics::ODEPlaneShape** (p. 619), and **gazebo::physics::BulletPlaneShape** (p. 208).

Referenced by gazebo::physics::BulletPlaneShape::SetAltitude(), and gazebo::physics::ODEPlaneShape::SetAltitude().

10.142.3.8 void gazebo::physics::PlaneShape::SetNormal (const math::Vector3 & *_norm*)

Set the normal.

10.142.3.9 void gazebo::physics::PlaneShape::SetSize (const math::Vector2d & *_size*)

Set the size.

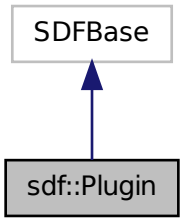
The documentation for this class was generated from the following file:

- **PlaneShape.hh**

10.143 sdf::Plugin Class Reference

`#include <Plugin.hh>`

Inheritance diagram for sdf::Plugin:



Public Member Functions

- **Plugin** ()
- void **Clear** ()
- void **Print** (const std::string &prefix)

Public Attributes

- std::vector< **ParamT** < std::string > > **data**
- **ParamT**< std::string > **filename**
- **ParamT**< std::string > **name**

10.143.1 Constructor & Destructor Documentation

10.143.1.1 sdf::Plugin::Plugin ( ) [inline]

10.143.2 Member Function Documentation

10.143.2.1 void sdf::Plugin::Clear ( ) [inline]

References data.

10.143.2.2 void sdf::Plugin::Print (const std::string & *prefix*) [inline]

References filename, and name.

10.143.3 Member Data Documentation

10.143.3.1 std::vector<ParamT<std::string> > sdf::Plugin::data

Referenced by Clear().

10.143.3.2 ParamT<std::string> sdf::Plugin::filename

Referenced by Print().

10.143.3.3 ParamT<std::string> sdf::Plugin::name

Referenced by Print().

The documentation for this class was generated from the following file:

- **sdf/interface/Plugin.hh**

10.144 gazebo::PluginT< T > Class Template Reference

A class which all plugins must inherit from.

`#include <Plugin.hh>`

Public Types

- typedef T * **TPtr**

  plugin pointer type definition

Public Member Functions

- std::string **GetFilename** () const

  Get the name of the handler.
- std::string **GetHandle** () const

  Get the short name of the handler.
- **PluginType GetType** () const

  Returns the type of the plugin.

Static Public Member Functions

- static **TPtr Create** (const std::string &_filename, const std::string &_handle)

  a class method that creates a plugin from a file name.

Protected Attributes

- std::string **filename**

  Path to the shared library file.
- std::string **handle**

  Short name.
- **PluginType type**

  Type of plugin.

10.144.1 Detailed Description

template<class T>class gazebo::PluginT< T >

A class which all plugins must inherit from.

10.144.2 Member Typedef Documentation

10.144.2.1 template<class T> typedef T* gazebo::PluginT< T >::TPtr

plugin pointer type definition

10.144.3 Member Function Documentation

10.144.3.1 template<class T> static TPtr gazebo::PluginT< T >::Create (const std::string & *_filename,* const std::string & *_handle*) [inline], [static]

a class method that creates a plugin from a file name.

It locates the shared library and loads it dynamically.

Parameters

in	*_filename*	the path to the shared library.
in	*_handle*	short name of the handler

10.144.3.2 template<class T> std::string gazebo::PluginT< T >::GetFilename ( ) const [inline]

Get the name of the handler.

10.144.3.3 template<class T> std::string gazebo::PluginT< T >::GetHandle ( ) const [inline]

Get the short name of the handler.

10.144.3.4 template<class T> PluginType gazebo::PluginT< T >::GetType ( ) const [inline]

Returns the type of the plugin.

10.144.4 Member Data Documentation

10.144.4.1 template<class T> std::string gazebo::PluginT< T >::filename [protected]

Path to the shared library file.

Referenced by gazebo::PluginT< ModelPlugin >::Create(), and gazebo::PluginT< ModelPlugin >::GetFilename().

10.144.4.2 template<class T> std::string gazebo::PluginT< T >::handle [protected]

Short name.

Referenced by gazebo::PluginT< ModelPlugin >::Create(), and gazebo::PluginT< ModelPlugin >::GetHandle().

10.144.4.3 template<class T> PluginType gazebo::PluginT< T >::type [protected]

Type of plugin.

Referenced by gazebo::PluginT< ModelPlugin >::GetType().

The documentation for this class was generated from the following file:

- **common/Plugin.hh**

10.145 gazebo::math::Pose Class Reference

Encapsulates a position and rotation in three space.

`#include <Pose.hh>`

Public Member Functions

- **Pose** ()
  Default constructors.
- **Pose** (const **Vector3** &_pos, const **Quaternion** &_rot)
  Constructor.
- **Pose** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)
  Constructor.
- **Pose** (const **Pose** &_pose)
  Copy constructor.
- virtual ~**Pose** ()
  Destructor.
- **Pose CoordPoseSolve** (const **Pose** &_b) const
  Find the inverse of a pose; i.e., if b = this + a, given b and this, find a.
- **Vector3 CoordPositionAdd** (const **Vector3** &_pos) const
  Add one point to a vector: result = this + pos.
- **Vector3 CoordPositionAdd** (const **Pose** &_pose) const
  Add one point to another: result = this + pose.
- **Vector3 CoordPositionSub** (const **Pose** &_pose) const
  Subtract one position from another: result = this - pose.

- **Quaternion CoordRotationAdd** (const **Quaternion** &_rot) const

  Add one rotation to another: result = this->rot + rot.
- **Quaternion CoordRotationSub** (const **Quaternion** &_rot) const

  Subtract one rotation from another: result = this->rot - rot.
- void **Correct** ()

  Fix any nan values.
- **Pose GetInverse** () const

  Get the inverse of this pose.
- bool **IsFinite** () const

  See if a pose is finite (e.g., not nan)
- bool **operator!=** (const **Pose** &_pose) const

  Inequality operator.
- **Pose operator*** (const **Pose** &_pose)

  Multiplication operator.
- **Pose operator+** (const **Pose** &_pose) const

  Addition operator.
- const **Pose** & **operator+=** (const **Pose** &_pose)

  Add-Equals operator.
- **Pose operator-** (const **Pose** &_pose) const

  Subtraction operator.
- const **Pose** & **operator-=** (const **Pose** &_pose)

  Subtraction operator.
- bool **operator==** (const **Pose** &_pose) const

  Equality operator.
- void **Reset** ()

  Reset the pose.
- **Pose RotatePositionAboutOrigin** (const **Quaternion** &_rot) const

  Rotate vector part of a pose about the origin.
- void **Round** (int _precision)

  Round all values to _precision decimal places.
- void **Set** (const **Vector3** &_pos, const **Quaternion** &_rot)

  *Set the pose from a **Vector3** (p. 902) and a **Quaternion** (p. 697).*
- void **Set** (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

  Set the pose from a six tuple.

Public Attributes

- **Vector3 pos**

  The position.
- **Quaternion rot**

  The rotation.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Pose** &_pose)

  Stream insertion operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Pose** &_pose)

  Stream extraction operator.

10.145.1 Detailed Description

Encapsulates a position and rotation in three space.

10.145.2 Constructor & Destructor Documentation

10.145.2.1 gazebo::math::Pose::Pose ( )

Default constructors.

Referenced by operator-().

10.145.2.2 gazebo::math::Pose::Pose (const Vector3 & _pos, const Quaternion & _rot)

Constructor.

Parameters

pos	A position
rot	A rotation

10.145.2.3 gazebo::math::Pose::Pose (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

Constructor.

10.145.2.4 gazebo::math::Pose::Pose (const Pose & _pose)

Copy constructor.

Parameters

pose	**Pose** (p. 677) to copy

10.145.2.5 virtual gazebo::math::Pose::~Pose ( ) [virtual]

Destructor.

10.145.3 Member Function Documentation

10.145.3.1 Pose gazebo::math::Pose::CoordPoseSolve (const Pose & _b) const

Find the inverse of a pose; i.e., if b = this + a, given b and this, find a.

Parameters

in	*_b*	the other pose

10.145.3.2 Vector3 gazebo::math::Pose::CoordPositionAdd (const Vector3 & _pos) const

Add one point to a vector: result = this + pos.

Parameters

in	*pos*	Position to add to this pose

Returns

the resulting position

10.145.3.3 Vector3 gazebo::math::Pose::CoordPositionAdd (const Pose & _pose) const

Add one point to another: result = this + pose.

Parameters

in	*_pose*	The **Pose** (p. 677) to add

Returns

The resulting position

10.145.3.4 Vector3 gazebo::math::Pose::CoordPositionSub (const Pose & _pose) const [inline]

Subtract one position from another: result = this - pose.

Parameters

in	*_pose*	**Pose** (p. 677) to subtract

Returns

The resulting position

References gazebo::math::Quaternion::GetInverse(), pos, rot, gazebo::math::Vector3::x, gazebo::math::Quaternion::x, gazebo::math::Vector3::y, gazebo::math::Quaternion::y, gazebo::math::Vector3::z, and gazebo::math::Quaternion::z.

Referenced by operator-().

10.145.3.5 Quaternion gazebo::math::Pose::CoordRotationAdd (const Quaternion & _rot) const

Add one rotation to another: result = this->rot + rot.

Parameters

in	*_rot*	Rotation to add

Returns

The resulting rotation

10.145.3.6 Quaternion gazebo::math::Pose::CoordRotationSub (const Quaternion & *_rot*) const [inline]

Subtract one rotation from another: result = this->rot - rot.

Parameters

in	*_rot*	The rotation to subtract

Returns

The resulting rotation

References gazebo::math::Quaternion::GetInverse(), gazebo::math::Quaternion::Normalize(), and rot.

Referenced by operator-().

10.145.3.7 void gazebo::math::Pose::Correct ( ) [inline]

Fix any nan values.

References gazebo::math::Vector3::Correct(), gazebo::math::Quaternion::Correct(), pos, and rot.

10.145.3.8 Pose gazebo::math::Pose::GetInverse ( ) const

Get the inverse of this pose.

Returns

the inverse pose

10.145.3.9 bool gazebo::math::Pose::IsFinite ( ) const

See if a pose is finite (e.g., not nan)

10.145.3.10 bool gazebo::math::Pose::operator!= (const Pose & *_pose*) const

Inequality operator.

Parameters

in	*_pose*	**Pose** (p. 677) for comparison

Returns

True if not equal

10.145.3.11 Pose gazebo::math::Pose::operator∗ (const Pose & *_pose*)

Multiplication operator.

Parameters

in	*_pose*	the other pose

Returns

itself

10.145.3.12 Pose gazebo::math::Pose::operator+ (const Pose & *_pose*) const

Addition operator.

Parameters

in	*pose*	**Pose** (p. 677) to add to this pose

Returns

The resulting pose

10.145.3.13 const Pose& gazebo::math::Pose::operator+= (const Pose & *_pose*)

Add-Equals operator.

Parameters

in	*pose*	**Pose** (p. 677) to add to this pose

Returns

The resulting pose

10.145.3.14 Pose gazebo::math::Pose::operator- (const Pose & *_pose*) const [inline]

Subtraction operator.

Parameters

in	*pose*	**Pose** (p. 677) to subtract from this one

Returns

The resulting pose

References CoordPositionSub(), CoordRotationSub(), Pose(), and rot.

10.145.3.15 const Pose& gazebo::math::Pose::operator-= (const Pose & _pose)

Subtraction operator.

Parameters

in	*pose*	**Pose** (p. 677) to subtract from this one

Returns

The resulting pose

10.145.3.16 bool gazebo::math::Pose::operator== (const Pose & _pose) const

Equality operator.

Parameters

in	*_pose*	**Pose** (p. 677) for comparison

Returns

True if equal

10.145.3.17 void gazebo::math::Pose::Reset ( )

Reset the pose.

10.145.3.18 Pose gazebo::math::Pose::RotatePositionAboutOrigin (const Quaternion & _rot) const

Rotate vector part of a pose about the origin.

Parameters

in	*_rot*	rotation

Returns

the rotated pose

10.145.3.19 void gazebo::math::Pose::Round (int _precision)

Round all values to _precision decimal places.

Parameters

in	*_precision*	

10.145.3.20 void gazebo::math::Pose::Set (const Vector3 & _pos, const Quaternion & _rot)

Set the pose from a **Vector3** (p. 902) and a **Quaternion** (p. 697).

Parameters

in	*_pos*	The position.
in	*_rot*	The rotation.

10.145.3.21 void gazebo::math::Pose::Set (double _x, double _y, double _z, double _roll, double _pitch, double _yaw)

Set the pose from a six tuple.

Parameters

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.

10.145.4 Friends And Related Function Documentation

10.145.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Pose & _pose) [friend]

Stream insertion operator.

Parameters

out	output stream
pose	pose to output

Returns

the stream

10.145.4.2 std::istream& operator>> (std::istream & _in, gazebo::math::Pose & _pose) [friend]

Stream extraction operator.

Parameters

in	*_in*	the input stream
in	*_pose*	the pose

Returns

the stream

10.145.5 Member Data Documentation

10.145.5.1 Vector3 gazebo::math::Pose::pos

The position.

Referenced by CoordPositionSub(), and Correct().

10.145.5.2 Quaternion gazebo::math::Pose::rot

The rotation.

Referenced by CoordPositionSub(), CoordRotationSub(), Correct(), and operator-().

The documentation for this class was generated from the following file:

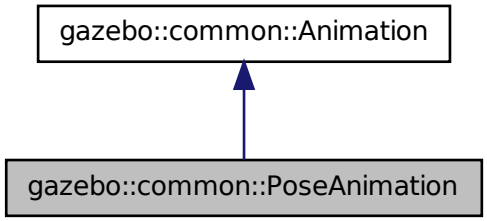
- **Pose.hh**

10.146 gazebo::common::PoseAnimation Class Reference

A pose animation.

`#include <Animation.hh>`

Inheritance diagram for gazebo::common::PoseAnimation:



Public Member Functions

- **PoseAnimation** (const std::string &_name, double _length, bool _loop)

  Constructor.
- virtual ∼**PoseAnimation** ()

  Destructor.
- **PoseKeyFrame** ∗ **CreateKeyFrame** (double _time)

  Create a pose keyframe at the given time.
- void **GetInterpolatedKeyFrame** (**PoseKeyFrame** &_kf) const

  Get a keyframe using the animation's current time.

Protected Member Functions

- void **BuildInterpolationSplines** () const

  Update the pose splines.
- void **GetInterpolatedKeyFrame** (double _time, **PoseKeyFrame** &_kf) const

  Get a keyframe using a passed in time.

Additional Inherited Members

10.146.1 Detailed Description

A pose animation.

10.146.2 Constructor & Destructor Documentation

10.146.2.1 gazebo::common::PoseAnimation::PoseAnimation (const std::string & *_name,* double *_length,* bool *_loop*)

Constructor.

Parameters

in	*_name*	String name of the animation. This should be unique.
in	*_length*	Length of the animation in seconds
in	*_loop*	True == loop the animation

10.146.2.2 virtual gazebo::common::PoseAnimation::~PoseAnimation ( ) [virtual]

Destructor.

10.146.3 Member Function Documentation

10.146.3.1 void gazebo::common::PoseAnimation::BuildInterpolationSplines ( ) const [protected]

Update the pose splines.

10.146.3.2 PoseKeyFrame* gazebo::common::PoseAnimation::CreateKeyFrame (double *_time*)

Create a pose keyframe at the given time.

Parameters

in	*_time*	**Time** (p. 840) at which to create the keyframe

Returns

Pointer to the new keyframe

10.146.3.3 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (PoseKeyFrame & _kf) const

Get a keyframe using the animation's current time.

Parameters

out	*_kf*	**PoseKeyFrame** (p. 687) reference to hold the interpolated result

10.146.3.4 void gazebo::common::PoseAnimation::GetInterpolatedKeyFrame (double _time, PoseKeyFrame & _kf) const [protected]

Get a keyframe using a passed in time.

Parameters

in	*_time*	**Time** (p. 840) in seconds
out	*_kf*	**PoseKeyFrame** (p. 687) reference to hold the interpolated result

The documentation for this class was generated from the following file:

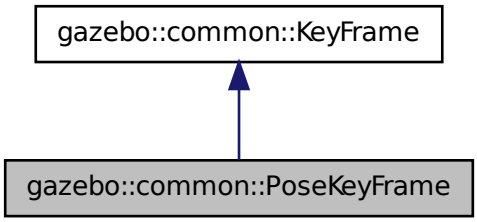
- **Animation.hh**

10.147 gazebo::common::PoseKeyFrame Class Reference

A keyframe for a **PoseAnimation** (p. 685).

`#include <KeyFrame.hh>`

Inheritance diagram for gazebo::common::PoseKeyFrame:



Public Member Functions

- **PoseKeyFrame** (double _time)

  Constructor.
- virtual ~**PoseKeyFrame** ()

  Destructor.

- const **math::Quaternion** & **GetRotation** () const

  Get the rotation of the keyframe.
- const **math::Vector3** & **GetTranslation** () const

  Get the translation of the keyframe.
- void **SetRotation** (const **math::Quaternion** &_rot)

  Set the rotation for the keyframe.
- void **SetTranslation** (const **math::Vector3** &_trans)

  Set the translation for the keyframe.

Protected Attributes

- **math::Quaternion rotate**

  the rotation quaternion
- **math::Vector3 translate**

  the translation vector

10.147.1 Detailed Description

A keyframe for a **PoseAnimation** (p. 685).

10.147.2 Constructor & Destructor Documentation

10.147.2.1 gazebo::common::PoseKeyFrame::PoseKeyFrame (double *_time*)

Constructor.

Parameters

in	*_time*	of the keyframe

10.147.2.2 virtual gazebo::common::PoseKeyFrame::~PoseKeyFrame ( ) [virtual]

Destructor.

10.147.3 Member Function Documentation

10.147.3.1 const math::Quaternion& gazebo::common::PoseKeyFrame::GetRotation ( ) const

Get the rotation of the keyframe.

Returns

The rotation amount

10.147.3.2 const math::Vector3& gazebo::common::PoseKeyFrame::GetTranslation ( ) const

Get the translation of the keyframe.

Returns

The translation amount

10.147.3.3 void gazebo::common::PoseKeyFrame::SetRotation (const math::Quaternion & *_rot*)

Set the rotation for the keyframe.

Parameters

in	*_trans*	Rotation amount

10.147.3.4 void gazebo::common::PoseKeyFrame::SetTranslation (const math::Vector3 & *_trans*)

Set the translation for the keyframe.

Parameters

in	*_trans*	Translation amount

10.147.4 Member Data Documentation

10.147.4.1 math::Quaternion gazebo::common::PoseKeyFrame::rotate [protected]

the rotation quaternion

10.147.4.2 math::Vector3 gazebo::common::PoseKeyFrame::translate [protected]

the translation vector

The documentation for this class was generated from the following file:

- **KeyFrame.hh**

10.148 gazebo::rendering::Projector Class Reference

Projects a material onto surface, light a light projector.

`#include <rendering/rendering.hh>`

Public Member Functions

- **Projector** (**VisualPtr** _parent)
  Constructor.
- virtual ~**Projector** ()
  Destructor.
- **VisualPtr GetParent** ()
  Get the parent visual.

- void **Load** (**sdf::ElementPtr** _sdf)

  Load from an sdf pointer.
- void **Load** (const msgs::Projector &_msg)

  Load from a message.
- void **Load** (const std::string &_name, const **math::Pose** &_pose=**math::Pose**(0, 0, 0, 0, 0, 0), const std::string &_textureName="", double _nearClip=0.25, double _farClip=15.0, double _fov=M_PI ∗0.25)

  Load the projector.
- void **SetEnabled** (bool _enabled)

  Set whether the projector is enabled or disabled.
- void **SetTexture** (const std::string &_textureName)

  Load a texture into the projector.
- void **Toggle** ()

  Toggle the activation of the projector.

10.148.1 Detailed Description

Projects a material onto surface, light a light projector.

10.148.2 Constructor & Destructor Documentation

10.148.2.1 gazebo::rendering::Projector::Projector (VisualPtr *_parent*)

Constructor.

Parameters

in	*_parent*	Name of the parent visual.

10.148.2.2 virtual gazebo::rendering::Projector::∼Projector ( ) [virtual]

Destructor.

10.148.3 Member Function Documentation

10.148.3.1 VisualPtr gazebo::rendering::Projector::GetParent ( )

Get the parent visual.

Returns

Pointer ot the parent visual.

10.148.3.2 void gazebo::rendering::Projector::Load (sdf::ElementPtr *_sdf*)

Load from an sdf pointer.

Parameters

in	*_sdf*	Pointer to the SDF element.

10.148.3.3 void gazebo::rendering::Projector::Load (const msgs::Projector & *_msg*)

Load from a message.

Parameters

in	*_msg*	Load from a message.

10.148.3.4 void gazebo::rendering::Projector::Load (const std::string & *_name*, const math::Pose & *_pose* = math::Pose(0, 0, 0, 0, 0, 0), const std::string & *_textureName* = "", double *_nearClip* = 0.25, double *_farClip* = 15.0, double *_fov* = M_PI *0.25)

Load the projector.

Parameters

in	*_name*	Name of the projector.
in	*_pos*	Pose of the projector.
in	*_textureName*	Name of the texture to project.
in	*_nearClip*	Near clip distance.
in	*_farClip*	Far clip distance.
in	*_fov*	Field of view.

10.148.3.5 void gazebo::rendering::Projector::SetEnabled (bool *_enabled*)

Set whether the projector is enabled or disabled.

Parameters

in	*_enabled*	True to enable the projector.

10.148.3.6 void gazebo::rendering::Projector::SetTexture (const std::string & *_textureName*)

Load a texture into the projector.

Parameters

in	*_textureName*	Name of the texture to project.

10.148.3.7 void gazebo::rendering::Projector::Toggle ( )

Toggle the activation of the projector.

The documentation for this class was generated from the following file:

- **Projector.hh**

10.149 gazebo::transport::Publication Class Reference

A publication for a topic.

`#include <Publication.hh>`

Public Member Functions

- **Publication** (const std::string &topic, const std::string &msgType)
  Constructor.
- virtual ~**Publication** ()
  Destructor.
- void **AddPublisher** (**PublisherPtr** _pub)
- void **AddSubscription** (const **CallbackHelperPtr** &callback)
- void **AddSubscription** (const **NodePtr** &_node)
- void **AddTransport** (const **PublicationTransportPtr** &publink)
- unsigned int **GetCallbackCount** () const
- bool **GetLocallyAdvertised** () const
  Return true if the topic has been advertised from this process.
- std::string **GetMsgType** () const
  Get the type of message.
- unsigned int **GetNodeCount** () const
- unsigned int **GetRemoteSubscriptionCount** ()
- unsigned int **GetTransportCount** () const
- bool **HasTransport** (const std::string &_host, unsigned int _port)
- void **LocalPublish** (const std::string &data)
  Publish data.
- void **Publish** (const google::protobuf::Message &msg, const boost::function< void()> &cb=**NULL**)
- void **RemoveSubscription** (const **NodePtr** &_node)
  Remove a subscription.
- void **RemoveSubscription** (const std::string &host, unsigned int port)
  Remove a subscription.
- void **RemoveTransport** (const std::string &host, unsigned int port)
- void **SetLocallyAdvertised** (bool _value)
  Set whether this topic has been advertised from this process.

10.149.1 Detailed Description

A publication for a topic.

This facilitates transport of messages

10.149.2 Constructor & Destructor Documentation

10.149.2.1 gazebo::transport::Publication::Publication (const std::string & *topic,* const std::string & *msgType*)

Constructor.

10.149.2.2 virtual gazebo::transport::Publication::~Publication ( ) [virtual]

Destructor.

10.149.3 Member Function Documentation

10.149.3.1 void gazebo::transport::Publication::AddPublisher (PublisherPtr _pub)

Referenced by gazebo::transport::TopicManager::Advertise().

10.149.3.2 void gazebo::transport::Publication::AddSubscription (const CallbackHelperPtr & callback)

Referenced by gazebo::transport::TopicManager::Advertise().

10.149.3.3 void gazebo::transport::Publication::AddSubscription (const NodePtr & _node)

10.149.3.4 void gazebo::transport::Publication::AddTransport (const PublicationTransportPtr & publink)

10.149.3.5 unsigned int gazebo::transport::Publication::GetCallbackCount ( ) const

10.149.3.6 bool gazebo::transport::Publication::GetLocallyAdvertised ( ) const

Return true if the topic has been advertised from this process.

Referenced by gazebo::transport::TopicManager::Advertise().

10.149.3.7 std::string gazebo::transport::Publication::GetMsgType ( ) const

Get the type of message.

10.149.3.8 unsigned int gazebo::transport::Publication::GetNodeCount ( ) const

10.149.3.9 unsigned int gazebo::transport::Publication::GetRemoteSubscriptionCount ( )

10.149.3.10 unsigned int gazebo::transport::Publication::GetTransportCount ( ) const

10.149.3.11 bool gazebo::transport::Publication::HasTransport (const std::string & _host, unsigned int _port)

10.149.3.12 void gazebo::transport::Publication::LocalPublish (const std::string & data)

Publish data.

10.149.3.13 void gazebo::transport::Publication::Publish (const google::protobuf::Message & msg, const boost::function< void() > & cb = NULL)

10.149.3.14 void gazebo::transport::Publication::RemoveSubscription (const NodePtr & _node)

Remove a subscription.

10.149.3.15 void gazebo::transport::Publication::RemoveSubscription (const std::string & *host,* unsigned int *port*)

Remove a subscription.

10.149.3.16 void gazebo::transport::Publication::RemoveTransport (const std::string & *host,* unsigned int *port*)

10.149.3.17 void gazebo::transport::Publication::SetLocallyAdvertised (bool *_value*)

Set whether this topic has been advertised from this process.

Referenced by gazebo::transport::TopicManager::Advertise().

The documentation for this class was generated from the following file:

- **Publication.hh**

10.150 gazebo::transport::PublicationTransport Class Reference

Reads data from a remote advertiser, and passes the data along to local subscribers.

`#include <PublicationTransport.hh>`

Public Member Functions

- **PublicationTransport** (const std::string &topic, const std::string &msgType)
- virtual ~**PublicationTransport** ()
- void **AddCallback** (const boost::function< void(const std::string &)> &cb)
- void **Fini** ()
- const **ConnectionPtr GetConnection** () const
- std::string **GetMsgType** () const
- std::string **GetTopic** () const
- void **Init** (const **ConnectionPtr** &conn)

10.150.1 Detailed Description

Reads data from a remote advertiser, and passes the data along to local subscribers.

10.150.2 Constructor & Destructor Documentation

10.150.2.1 gazebo::transport::PublicationTransport::PublicationTransport (const std::string & *topic,* const std::string & *msgType*)

10.150.2.2 virtual gazebo::transport::PublicationTransport::~PublicationTransport ( ) [virtual]

10.150.3 Member Function Documentation

10.150.3.1 void gazebo::transport::PublicationTransport::AddCallback (const boost::function< void(const std::string &)> & *cb*)

10.150.3.2 void gazebo::transport::PublicationTransport::Fini ( )

10.150.3.3 const ConnectionPtr gazebo::transport::PublicationTransport::GetConnection ( ) const

10.150.3.4 std::string gazebo::transport::PublicationTransport::GetMsgType ( ) const

10.150.3.5 std::string gazebo::transport::PublicationTransport::GetTopic ( ) const

10.150.3.6 void gazebo::transport::PublicationTransport::Init (const ConnectionPtr & *conn*)

The documentation for this class was generated from the following file:

- **PublicationTransport.hh**

10.151 gazebo::transport::Publisher Class Reference

A publisher of messages on a topic.

`#include <Publisher.hh>`

Public Member Functions

- **Publisher** (const std::string &topic, const std::string &msg_type, unsigned int _limit, bool _latch)

  Use this constructor.
- virtual ~**Publisher** ()

  Destructor.
- bool **GetLatching** () const
- std::string **GetMsgType** () const

  Get the message type.
- unsigned int **GetOutgoingCount** () const
- std::string **GetPrevMsg** () const
- std::string **GetTopic** () const

  Get the topic name.
- bool **HasConnections** () const
- void **Publish** (const google::protobuf::Message &_message, bool _block=false)

  Publish a message on the topic.
- template<typename M >
  void **Publish** (M _message, bool _block=false)
- void **SendMessage** ()

  Send latest message over the wire. For internal use only.
- void **SetPublication** (**PublicationPtr** &_publication, int _i)
- void **WaitForConnection** () const

  Block until a connection has been established with this publisher.

10.151.1 Detailed Description

A publisher of messages on a topic.

10.151.2 Constructor & Destructor Documentation

10.151.2.1 gazebo::transport::Publisher::Publisher (const std::string & *topic,* const std::string & *msg_type,* unsigned int *_limit,* bool *_latch*)

Use this constructor.

Parameters

topic	Name of topic
msg_type	Type of the message which is to be published

10.151.2.2 virtual gazebo::transport::Publisher::~Publisher ( ) [virtual]

Destructor.

10.151.3 Member Function Documentation

10.151.3.1 bool gazebo::transport::Publisher::GetLatching ( ) const

10.151.3.2 std::string gazebo::transport::Publisher::GetMsgType ( ) const

Get the message type.

10.151.3.3 unsigned int gazebo::transport::Publisher::GetOutgoingCount ( ) const

10.151.3.4 std::string gazebo::transport::Publisher::GetPrevMsg ( ) const

10.151.3.5 std::string gazebo::transport::Publisher::GetTopic ( ) const

Get the topic name.

10.151.3.6 bool gazebo::transport::Publisher::HasConnections ( ) const

10.151.3.7 void gazebo::transport::Publisher::Publish (const google::protobuf::Message & *_message,* bool *_block* = false) [inline]

Publish a message on the topic.

10.151.3.8 template<typename M > void gazebo::transport::Publisher::Publish (M *_message,* bool *_block* = false) [inline]

10.151.3.9 void gazebo::transport::Publisher::SendMessage ( )

Send latest message over the wire. For internal use only.

10.151.3.10 void gazebo::transport::Publisher::SetPublication (PublicationPtr & *_publication,* int *_i*)

10.151.3.11 void gazebo::transport::Publisher::WaitForConnection ( ) const

Block until a connection has been established with this publisher.

The documentation for this class was generated from the following file:

- **Publisher.hh**

10.152 gazebo::math::Quaternion Class Reference

A quaternion class.

`#include <Quaternion.hh>`

Public Member Functions

- **Quaternion** ()
  Default Constructor.
- **Quaternion** (const double &_w, const double &_x, const double &_y, const double &_z)
  Constructor.
- **Quaternion** (const double &_roll, const double &_pitch, const double &_yaw)
  Constructor from Euler angles.
- **Quaternion** (const **Vector3** &_axis, const double &_angle)
  Constructor from axis angle.
- **Quaternion** (const **Vector3** &_rpy)
  Constructor.
- **Quaternion** (const **Quaternion** &_qt)
  Copy constructor.
- ∼**Quaternion** ()
  Destructor.
- void **Correct** ()
  Correct any nan.
- double **Dot** (const **Quaternion** &_q) const
  Dot product.
- void **GetAsAxis** (**Vector3** &_axis, double &_angle) const
  Return rotation as axis and angle.
- **Vector3 GetAsEuler** () const
  Return the rotation in Euler angles.
- **Matrix3 GetAsMatrix3** () const
  Get the quaternion as a 3x3 matrix.
- **Matrix4 GetAsMatrix4** () const
  Get the quaternion as a 4x4 matrix.
- **Quaternion GetExp** () const
  Return the exponent.
- **Quaternion GetInverse** () const
  Get the inverse of this quaternion.
- **Quaternion GetLog** () const
  Return the logarithm.

- double **GetPitch** ()

  Get the Euler pitch angle in radians.
- double **GetRoll** ()

  Get the Euler roll angle in radians.
- **Vector3 GetXAxis** () const

  Return the X axis.
- double **GetYaw** ()

  Get the Euler yaw angle in radians.
- **Vector3 GetYAxis** () const

  Return the Y axis.
- **Vector3 GetZAxis** () const

  Return the Z axis.
- void **Invert** ()

  Invert the quaternion.
- bool **IsFinite** () const

  See if a quatern is finite (e.g., not nan)
- void **Normalize** ()

  Normalize the quaternion.
- bool **operator!=** (const **Quaternion** &_qt) const

  Not equal to operator.
- **Quaternion operator**∗ (const **Quaternion** &_q) const

  Multiplication operator.
- **Quaternion operator**∗ (const double &_f) const

  Multiplication operator.
- **Vector3 operator**∗ (const **Vector3** &v) const

  Vector3 *(*p. *902) multiplication operator.*
- **Quaternion operator∗=** (const **Quaternion** &qt)

  Multiplication operator.
- **Quaternion operator+** (const **Quaternion** &_qt) const

  Addition operator.
- **Quaternion operator+=** (const **Quaternion** &_qt)

  Addition operator.
- **Quaternion operator-** (const **Quaternion** &_qt) const

  Substraction operator.
- **Quaternion operator-** () const

  Unary minus operator.
- **Quaternion operator-=** (const **Quaternion** &_qt)

  Substraction operator.
- **Quaternion** & **operator=** (const **Quaternion** &qt)

  Equal operator.
- bool **operator==** (const **Quaternion** &_qt) const

  Equal to operator.
- **Vector3 RotateVector** (const **Vector3** &_vec) const

  Rotate a vector using the quaternion.
- **Vector3 RotateVectorReverse** (**Vector3** _vec) const

  Do the reverse rotation of a vector by this quaternion.
- void **Round** (int _precision)

Round all values to _precision decimal places.

- void **Scale** (double _scale)

  Scale a Quaternionion.

- void **Set** (double _u, double _x, double _y, double _z)

  Set this quaternion from 4 floating numbers.

- void **SetFromAxis** (double _x, double _y, double _z, double _a)

  Set the quaternion from an axis and angle.

- void **SetFromAxis** (const **Vector3** &_axis, double _a)

  Set the quaternion from an axis and angle.

- void **SetFromEuler** (const **Vector3** &_vec)

  Set the quaternion from Euler angles.

- void **SetToIdentity** ()

  Set the quatern to the identity.

Static Public Member Functions

- static **Quaternion EulerToQuaternion** (const **Vector3** &_vec)

  Convert euler angles to quatern.

- static **Quaternion EulerToQuaternion** (double _x, double _y, double _z)

  Convert euler angles to quatern.

- static **Quaternion Slerp** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkQ, bool _shortestPath=false)

  Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.

- static **Quaternion Squad** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkA, const **Quaternion** &_rkB, const **Quaternion** &_rkQ, bool _shortestPath=false)

  Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.

Public Attributes

- double **w**

  Attributes of the quaternion.

- double **x**

  Attributes of the quaternion.

- double **y**

  Attributes of the quaternion.

- double **z**

  Attributes of the quaternion.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Quaternion** &_q)

  Stream insertion operator.

- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Quaternion** &_q)

  Stream extraction operator.

10.152.1 Detailed Description

A quaternion class.

10.152.2 Constructor & Destructor Documentation

10.152.2.1 gazebo::math::Quaternion::Quaternion ( )

Default Constructor.

Referenced by operator∗().

10.152.2.2 gazebo::math::Quaternion::Quaternion (const double & *_w*, const double & *_x*, const double & *_y*, const double & *_z*)

Constructor.

Parameters

in	*_w*	W param
in	*_x*	X param
in	*_y*	Y param
in	*_z*	Z param

10.152.2.3 gazebo::math::Quaternion::Quaternion (const double & *_roll*, const double & *_pitch*, const double & *_yaw*)

Constructor from Euler angles.

Parameters

in	*_roll*	roll
in	*_pitch*	pitch
in	*_yaw*	yaw

10.152.2.4 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

10.152.2.5 gazebo::math::Quaternion::Quaternion (const Vector3 & *_rpy*)

Constructor.

Parameters

in	*_rpy*	euler angles

10.152.2.6 gazebo::math::Quaternion::Quaternion (const Quaternion & _qt)

Copy constructor.

Parameters

qt	**Quaternion** (p. 697) to copy

10.152.2.7 gazebo::math::Quaternion::~Quaternion ( )

Destructor.

10.152.3 Member Function Documentation

10.152.3.1 void gazebo::math::Quaternion::Correct ( ) [inline]

Correct any nan.

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.152.3.2 double gazebo::math::Quaternion::Dot (const Quaternion & _q) const

Dot product.

Parameters

in	*the*	other quaternion

Returns

the product

10.152.3.3 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (const Vector3 & _vec) [static]

Convert euler angles to quatern.

Parameters

in		

10.152.3.4 static Quaternion gazebo::math::Quaternion::EulerToQuaternion (double _x, double _y, double _z) [static]

Convert euler angles to quatern.

Parameters

_x	rotation along x
_y	rotation along y
_z	rotation along z

10.152.3.5 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

10.152.3.6 Vector3 gazebo::math::Quaternion::GetAsEuler ( ) const

Return the rotation in Euler angles.

Returns

This quaternion as an Euler vector

10.152.3.7 Matrix3 gazebo::math::Quaternion::GetAsMatrix3 ( ) const

Get the quaternion as a 3x3 matrix.

10.152.3.8 Matrix4 gazebo::math::Quaternion::GetAsMatrix4 ( ) const

Get the quaternion as a 4x4 matrix.

Returns

a 4x4 matrix

10.152.3.9 Quaternion gazebo::math::Quaternion::GetExp ( ) const

Return the exponent.

Returns

the exp

10.152.3.10 Quaternion gazebo::math::Quaternion::GetInverse ( ) const [inline]

Get the inverse of this quaternion.

Returns

Inverse quarenion

References gazebo::math::equal(), w, x, y, and z.

Referenced by gazebo::math::Pose::CoordPositionSub(), gazebo::math::Pose::CoordRotationSub(), and RotateVector().

10.152.3.11 Quaternion gazebo::math::Quaternion::GetLog ( ) const

Return the logarithm.

Returns

the log

10.152.3.12 double gazebo::math::Quaternion::GetPitch ( )

Get the Euler pitch angle in radians.

Returns

the pitch

10.152.3.13 double gazebo::math::Quaternion::GetRoll ( )

Get the Euler roll angle in radians.

Returns

the roll

10.152.3.14 Vector3 gazebo::math::Quaternion::GetXAxis ( ) const

Return the X axis.

Returns

the vector

10.152.3.15 double gazebo::math::Quaternion::GetYaw ( )

Get the Euler yaw angle in radians.

Returns

the yaw

10.152.3.16 Vector3 gazebo::math::Quaternion::GetYAxis ( ) const

Return the Y axis.

Returns

the vector

10.152.3.17 Vector3 gazebo::math::Quaternion::GetZAxis ( ) const

Return the Z axis.

Returns

the vector

10.152.3.18 void gazebo::math::Quaternion::Invert ( )

Invert the quaternion.

10.152.3.19 bool gazebo::math::Quaternion::IsFinite ( ) const

See if a quatern is finite (e.g., not nan)

Returns

True if quatern is finite

10.152.3.20 void gazebo::math::Quaternion::Normalize ( )

Normalize the quaternion.

Referenced by gazebo::math::Pose::CoordRotationSub().

10.152.3.21 bool gazebo::math::Quaternion::operator!= (const Quaternion & _qt) const

Not equal to operator.

Parameters

in	*_qt*	**Quaternion** (p. 697) for comparison

Returns

True if not equal

10.152.3.22 Quaternion gazebo::math::Quaternion::operator* (const Quaternion & _q) const [inline]

Multiplication operator.

Parameters

in	*_qt*	**Quaternion** (p. 697) for multiplication

Returns

This quaternion multiplied by the parameter

References Quaternion(), w, x, y, and z.

10.152.3.23 Quaternion gazebo::math::Quaternion::operator∗ (const double & _f) const

Multiplication operator.

Parameters

in	*_f*	factor

Returns

quaternion multiplied by _f

10.152.3.24 Vector3 gazebo::math::Quaternion::operator∗ (const Vector3 & *v*) const

Vector3 (p. 902) multiplication operator.

10.152.3.25 Quaternion gazebo::math::Quaternion::operator∗= (const Quaternion & *qt*)

Multiplication operator.

Parameters

in	*_qt*	**Quaternion** (p. 697) for multiplication

Returns

This quatern multiplied by the parameter

10.152.3.26 Quaternion gazebo::math::Quaternion::operator+ (const Quaternion & _qt) const

Addition operator.

Parameters

_qt[in]	quaternion for addition

Returns

this quaternion + _qt

10.152.3.27 Quaternion gazebo::math::Quaternion::operator+= (const Quaternion & _qt)

Addition operator.

Parameters

in	*_qt*	quaternion for addition

Returns

this quaternion + qt

10.152.3.28 Quaternion gazebo::math::Quaternion::operator- (const Quaternion & *_qt*) const

Substraction operator.

Parameters

in	*_qt*	quaternion to substract

Returns

this quaternion - _qt

10.152.3.29 Quaternion gazebo::math::Quaternion::operator- ( ) const

Unary minus operator.

Returns

negates each component of the quaternion

10.152.3.30 Quaternion gazebo::math::Quaternion::operator-= (const Quaternion & *_qt*)

Substraction operator.

Parameters

in	*_qt*	**Quaternion** (p. 697) for substraction

Returns

This quatern - qt

10.152.3.31 Quaternion& gazebo::math::Quaternion::operator= (const Quaternion & *qt*)

Equal operator.

Parameters

in	*qt*	**Quaternion** (p. 697) to copy

10.152.3.32 bool gazebo::math::Quaternion::operator== (const Quaternion & _qt) const

Equal to operator.

Parameters

in	*_qt*	**Quaternion** (p. 697) for comparison

Returns

True if equal

10.152.3.33 Vector3 gazebo::math::Quaternion::RotateVector (const Vector3 & _vec) const [inline]

Rotate a vector using the quaternion.

Parameters

in	*_vec*	vector to rotate

Returns

the rotated vector

References GetInverse(), gazebo::math::Vector3::x, x, gazebo::math::Vector3::y, y, gazebo::math::Vector3::z, and z.

10.152.3.34 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

10.152.3.35 void gazebo::math::Quaternion::Round (int _precision)

Round all values to _precision decimal places.

Parameters

in	*_precision*	the precision

10.152.3.36 void gazebo::math::Quaternion::Scale (double _scale)

Scale a Quaternionion.

Parameters

in	*_scale*	Amount to scale this rotation

10.152.3.37 void gazebo::math::Quaternion::Set (double _u, double _x, double _y, double _z)

Set this quaternion from 4 floating numbers.

Parameters

in	*_u*	u
in	*_x*	x
in	*_y*	y
in	*_z*	z

10.152.3.38 void gazebo::math::Quaternion::SetFromAxis (double _x, double _y, double _z, double _a)

Set the quaternion from an axis and angle.

Parameters

in	*_x*	X axis
in	*_y*	Y axis
in	*_z*	Z axis
in	*_a*	**Angle** (p. 127) in radians

10.152.3.39 void gazebo::math::Quaternion::SetFromAxis (const Vector3 & _axis, double _a)

Set the quaternion from an axis and angle.

Parameters

in	*_axis*	Axis
in	*_a*	**Angle** (p. 127) in radians

10.152.3.40 void gazebo::math::Quaternion::SetFromEuler (const Vector3 & _vec)

Set the quaternion from Euler angles.

Parameters

in	*vec*	Euler angle

10.152.3.41 void gazebo::math::Quaternion::SetToIdentity ( )

Set the quatern to the identity.

10.152.3.42 static Quaternion gazebo::math::Quaternion::Slerp (double *_fT,* const Quaternion & *_rkP,* const Quaternion & *_rkQ,* bool *_shortestPath* = false) [static]

Spherical linear interpolation between 2 quaternions, given the ends and an interpolation parameter between 0 and 1.

Parameters

in	*_ft*	the interpolation parameter
in	*_rkP*	the beginning quaternion
in	*_rkQ*	the end quaternion
in	*_shortestPath*	when true, the rotation may be inverted to get to minimize rotation

10.152.3.43 static Quaternion gazebo::math::Quaternion::Squad (double *_fT,* const Quaternion & *_rkP,* const Quaternion & *_rkA,* const Quaternion & *_rkB,* const Quaternion & *_rkQ,* bool *_shortestPath* = false) [static]

Spherical quadratic interpolation given the ends and an interpolation parameter between 0 and 1.

Parameters

in	*_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

10.152.4 Friends And Related Function Documentation

10.152.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::math::Quaternion & *_q*) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_q*	quaternion to output

Returns

the stream

10.152.4.2 std::istream& operator>> (std::istream & *_in,* gazebo::math::Quaternion & *_q*) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_q*	**Quaternion** (p. 697) to read values into

Returns

The istream

10.152.5 Member Data Documentation

10.152.5.1 double gazebo::math::Quaternion::w

Attributes of the quaternion.

Referenced by Correct(), GetInverse(), and operator∗().

10.152.5.2 double gazebo::math::Quaternion::x

Attributes of the quaternion.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), GetInverse(), operator∗(), and RotateVector().

10.152.5.3 double gazebo::math::Quaternion::y

Attributes of the quaternion.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), GetInverse(), operator∗(), and RotateVector().

10.152.5.4 double gazebo::math::Quaternion::z

Attributes of the quaternion.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), GetInverse(), operator∗(), and RotateVector().

The documentation for this class was generated from the following file:

- **Quaternion.hh**

10.153 gazebo::math::Rand Class Reference

Random number generator class.

`#include <Rand.hh>`

Static Public Member Functions

- static double **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.

10.153.1 Detailed Description

Random number generator class.

10.153.2 Member Function Documentation

10.153.2.1 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

10.153.2.2 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

10.153.2.3 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

10.153.2.4 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

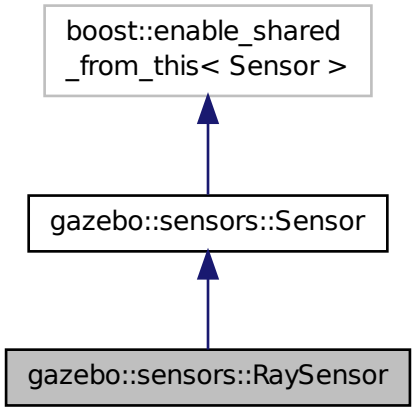
The documentation for this class was generated from the following file:

- **Rand.hh**

10.154 gazebo::sensors::RaySensor Class Reference

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::RaySensor:



Public Member Functions

- **RaySensor** ()

  Constructor.
- virtual ~**RaySensor** ()

  Destructor.
- **math::Angle GetAngleMax** () const

  Get the maximum angle.
- **math::Angle GetAngleMin** () const

  Get the minimum angle.
- double **GetAngleResolution** () const

  Get radians between each range.
- int **GetFiducial** (int _index)

  Get detected fiducial value for a ray.
- **physics::MultiRayShapePtr GetLaserShape** () const

  *Returns a pointer to the internal **physics::MultiRayShape** (p. 549).*
- double **GetRange** (int _index)

  Get detected range for a ray.
- int **GetRangeCount** () const

  Get the range count.
- double **GetRangeMax** () const

  Get the maximum range.
- double **GetRangeMin** () const

  Get the minimum range.
- double **GetRangeResolution** () const

  Get the range resolution.

- void **GetRanges** (std::vector< double > &_ranges)
  Get all the ranges.
- int **GetRayCount** () const
  Get the ray count.
- double **GetRetro** (int _index)
  Get detected retro (intensity) value for a ray.
- virtual std::string **GetTopic** () const
  Gets the topic name of the sensor.
- **math::Angle GetVerticalAngleMax** () const
  Get the vertical scan line top angle.
- **math::Angle GetVerticalAngleMin** () const
  Get the vertical scan bottom angle.
- int **GetVerticalRangeCount** () const
  Get the vertical scan line count.
- int **GetVerticalRayCount** () const
  Get the vertical scan line count.
- virtual void **Init** ()
  Initialize the ray.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** _sdf)
  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)
  Load the sensor with default parameters.

Protected Member Functions

- virtual void **Fini** ()
  Finalize the ray.
- virtual void **UpdateImpl** (bool _force)
  Update the sensor information.

Additional Inherited Members

10.154.1 Constructor & Destructor Documentation

10.154.1.1 gazebo::sensors::RaySensor::RaySensor ( )

Constructor.

10.154.1.2 virtual gazebo::sensors::RaySensor::~RaySensor ( ) [virtual]

Destructor.

10.154.2 Member Function Documentation

10.154.2.1 virtual void gazebo::sensors::RaySensor::Fini ( ) [protected], [virtual]

Finalize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

10.154.2.2 math::Angle gazebo::sensors::RaySensor::GetAngleMax ( ) const

Get the maximum angle.

Returns

the maximum angle

10.154.2.3 math::Angle gazebo::sensors::RaySensor::GetAngleMin ( ) const

Get the minimum angle.

Returns

The minimum angle

10.154.2.4 double gazebo::sensors::RaySensor::GetAngleResolution ( ) const

Get radians between each range.

Returns

Resolution of the angle

10.154.2.5 int gazebo::sensors::RaySensor::GetFiducial (int *_index*)

Get detected fiducial value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index value of specific ray

Returns

Fiducial value

10.154.2.6 physics::MultiRayShapePtr gazebo::sensors::RaySensor::GetLaserShape ( ) const [inline]

Returns a pointer to the internal **physics::MultiRayShape** (p. 549).

Returns

Pointer to ray shape

10.154.2.7 double gazebo::sensors::RaySensor::GetRange (int _index)

Get detected range for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index of specific ray

Returns

Returns DBL_MAX for no detection.

10.154.2.8 int gazebo::sensors::RaySensor::GetRangeCount ( ) const

Get the range count.

Returns

The number of ranges

10.154.2.9 double gazebo::sensors::RaySensor::GetRangeMax ( ) const

Get the maximum range.

Returns

The maximum range

10.154.2.10 double gazebo::sensors::RaySensor::GetRangeMin ( ) const

Get the minimum range.

Returns

The minimum range

10.154.2.11 double gazebo::sensors::RaySensor::GetRangeResolution ( ) const

Get the range resolution.

Returns

Resolution of the range

10.154.2.12 void gazebo::sensors::RaySensor::GetRanges (std::vector< double > & *_ranges*)

Get all the ranges.

Parameters

_ranges	A vector that will contain all the range data

10.154.2.13 int gazebo::sensors::RaySensor::GetRayCount ( ) const

Get the ray count.

Returns

The number of rays

10.154.2.14 double gazebo::sensors::RaySensor::GetRetro (int *_index*)

Get detected retro (intensity) value for a ray.

```
Warning: If you are accessing all the ray data in a loop
it's possible that the Ray will update in the middle of
your access loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Parameters

in	*_index*	Index of specific ray

Returns

Retro (intensity) value for ray

10.154.2.15 virtual std::string gazebo::sensors::RaySensor::GetTopic ( ) const [virtual]

Gets the topic name of the sensor.

Returns

Topic name

Reimplemented from **gazebo::sensors::Sensor** (p. 769).

10.154.2.16 math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMax ( ) const

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

10.154.2.17 math::Angle gazebo::sensors::RaySensor::GetVerticalAngleMin ( ) const

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

10.154.2.18 int gazebo::sensors::RaySensor::GetVerticalRangeCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.154.2.19 int gazebo::sensors::RaySensor::GetVerticalRayCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

10.154.2.20 virtual void gazebo::sensors::RaySensor::Init ( ) [virtual]

Initialize the ray.

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.154.2.21 virtual void gazebo::sensors::RaySensor::Load (const std::string & *_worldName*, sdf::ElementPtr *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 765) parameters
in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.154.2.22 virtual void gazebo::sensors::RaySensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.154.2.23 virtual void gazebo::sensors::RaySensor::UpdateImpl (bool *_force*) [protected], [virtual]

Update the sensor information.

Reimplemented from **gazebo::sensors::Sensor** (p. 771).

The documentation for this class was generated from the following file:

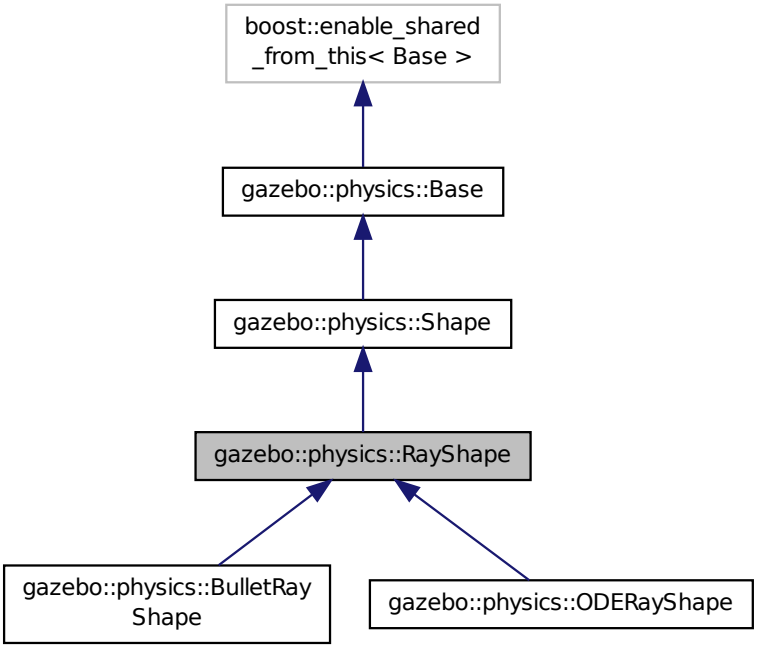
- **RaySensor.hh**

10.155 gazebo::physics::RayShape Class Reference

Base (p. 145) class for Ray collision geometry.

`#include <RayShape.hh>`

Inheritance diagram for gazebo::physics::RayShape:



Public Member Functions

- **RayShape** (**PhysicsEnginePtr** _physicsEngine)

  Constructor for a global ray.
- **RayShape** (**CollisionPtr parent**)

Constructor.

- virtual ~**RayShape** ()

  Destructor.
- void **FillMsg** (msgs::Geometry &)
- int **GetFiducial** () const

  Get the fiducial id detected by this ray.
- virtual void **GetGlobalPoints** (**math::Vector3** &posA, **math::Vector3** &posB)

  Get the global starting and ending points.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)=0

  Get the nearest intersection.
- double **GetLength** () const

  Get the length of the ray.
- virtual void **GetRelativePoints** (**math::Vector3** &posA, **math::Vector3** &posB)

  Get the relative starting and ending points.
- float **GetRetro** () const

  Get the retro-reflectivness detected by this ray.
- virtual void **Init** ()

  In the ray.
- virtual void **ProcessMsg** (const msgs::Geometry &)
- void **SetFiducial** (int fid)

  Set the fiducial id detected by this ray.
- virtual void **SetLength** (double len)

  Set the length of the ray.
- virtual void **SetPoints** (const **math::Vector3** &posStart, const **math::Vector3** &posEnd)

  Set the ray based on starting and ending points relative to the body.
- void **SetRetro** (float retro)

  Set the retro-reflectivness detected by this ray.
- virtual void **Update** ()=0

  Update the ray collision.

Protected Attributes

- int **contactFiducial**
- double **contactLen**

  Contact *(*p. *296) information; this is filled out during collision detection.*
- double **contactRetro**
- **math::Vector3 globalEndPos**
- **math::Vector3 globalStartPos**

  Start and end positions of the ray in global cs.
- **math::Vector3 relativeEndPos**
- **math::Vector3 relativeStartPos**

  Start and end positions of the ray, relative to the body.

Additional Inherited Members

10.155.1 Detailed Description

Base (p. 145) class for Ray collision geometry.

10.155.2 Constructor & Destructor Documentation

10.155.2.1 gazebo::physics::RayShape::RayShape (PhysicsEnginePtr *_physicsEngine*)

Constructor for a global ray.

10.155.2.2 gazebo::physics::RayShape::RayShape (CollisionPtr *parent*)

Constructor.

Parameters

body	**Link** (p. 454) the ray is attached to
displayRays	Indicates if the rays should be displayed when rendering images

10.155.2.3 virtual gazebo::physics::RayShape::~RayShape ( ) [virtual]

Destructor.

10.155.3 Member Function Documentation

10.155.3.1 void gazebo::physics::RayShape::FillMsg (msgs::Geometry &) [inline], [virtual]

Implements **gazebo::physics::Shape** (p. 781).

10.155.3.2 int gazebo::physics::RayShape::GetFiducial ( ) const

Get the fiducial id detected by this ray.

10.155.3.3 virtual void gazebo::physics::RayShape::GetGlobalPoints (math::Vector3 & *posA,* math::Vector3 & *posB*) [virtual]

Get the global starting and ending points.

Parameters

posA	Returns the starting point
posB	Returns the ending point

10.155.3.4 virtual void gazebo::physics::RayShape::GetIntersection (double & *_dist,* std::string & *_entity*) [pure virtual]

Get the nearest intersection.

Implemented in **gazebo::physics::ODERayShape** (p. 621), and **gazebo::physics::BulletRayShape** (p. 212).

10.155.3.5 double gazebo::physics::RayShape::GetLength ( ) const

Get the length of the ray.

10.155.3.6 virtual void gazebo::physics::RayShape::GetRelativePoints (math::Vector3 & *posA*, math::Vector3 & *posB*) [virtual]

Get the relative starting and ending points.

Parameters

posA	Returns the starting point
posB	Returns the ending point

10.155.3.7 float gazebo::physics::RayShape::GetRetro ( ) const

Get the retro-reflectivness detected by this ray.

10.155.3.8 virtual void gazebo::physics::RayShape::Init ( ) [virtual]

In the ray.

Implements **gazebo::physics::Shape** (p. 781).

10.155.3.9 virtual void gazebo::physics::RayShape::ProcessMsg (const msgs::Geometry &) [inline], [virtual]

Implements **gazebo::physics::Shape** (p. 782).

10.155.3.10 void gazebo::physics::RayShape::SetFiducial (int *fid*)

Set the fiducial id detected by this ray.

10.155.3.11 virtual void gazebo::physics::RayShape::SetLength (double *len*) [virtual]

Set the length of the ray.

Parameters

len	Length of the array

10.155.3.12 virtual void gazebo::physics::RayShape::SetPoints (const math::Vector3 & *posStart*, const math::Vector3 & *posEnd*) [virtual]

Set the ray based on starting and ending points relative to the body.

Parameters

posStart	Start position, relative the body
posEnd	End position, relative to the body

Reimplemented in **gazebo::physics::ODERayShape** (p. 621), and **gazebo::physics::BulletRayShape** (p. 212).

10.155.3.13 void gazebo::physics::RayShape::SetRetro (float *retro*)

Set the retro-reflectivness detected by this ray.

10.155.3.14 virtual void gazebo::physics::RayShape::Update ( ) [pure virtual]

Update the ray collision.

Reimplemented from **gazebo::physics::Base** (p. 157).

Implemented in **gazebo::physics::ODERayShape** (p. 621), and **gazebo::physics::BulletRayShape** (p. 212).

10.155.4 Member Data Documentation

10.155.4.1 int gazebo::physics::RayShape::contactFiducial [protected]

10.155.4.2 double gazebo::physics::RayShape::contactLen [protected]

Contact (p. 296) information; this is filled out during collision detection.

10.155.4.3 double gazebo::physics::RayShape::contactRetro [protected]

10.155.4.4 math::Vector3 gazebo::physics::RayShape::globalEndPos [protected]

10.155.4.5 math::Vector3 gazebo::physics::RayShape::globalStartPos [protected]

Start and end positions of the ray in global cs.

10.155.4.6 math::Vector3 gazebo::physics::RayShape::relativeEndPos [protected]

10.155.4.7 math::Vector3 gazebo::physics::RayShape::relativeStartPos [protected]

Start and end positions of the ray, relative to the body.

The documentation for this class was generated from the following file:

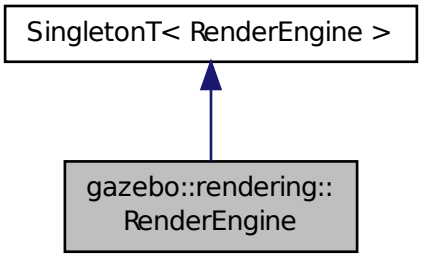
- **RayShape.hh**

10.156 gazebo::rendering::RenderEngine Class Reference

Adaptor to Ogre3d.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RenderEngine:



Public Types

- enum **RenderPathType** {
  NONE, **VERTEX**, **FORWARD**, **DEFERRED**,
  RENDER_PATH_COUNT }

  The type of rendering path used by the rendering engine.

Public Member Functions

- void **AddResourcePath** (const std::string &_uri)

  *Add a new path for **Ogre** (p. 118) to search for resources.*
- **ScenePtr CreateScene** (const std::string &_name, bool _enableVisualizations)

  Create a scene.
- void **Fini** ()

  Tears down the rendering engine.
- **RenderPathType GetRenderPathType** () const

  Get the type of rendering path to use.
- **ScenePtr GetScene** (const std::string &_name)

  Get a scene by name.
- **ScenePtr GetScene** (unsigned int _index)

  Get a scene by index.
- unsigned int **GetSceneCount** () const

  Get the number of scenes.
- void **Init** ()

  *Initialize **Ogre** (p. 118). Load must happen before Init.*
- void **Load** ()

  *Load the parameters for **Ogre** (p. 118). Load must happen before Init.*
- void **RemoveScene** (const std::string &_name)

  Remove a scene.

Public Attributes

- Ogre::Root * **root**

  Pointer to the root scene node.

Protected Attributes

- void * **dummyContext**

  GLX context used to render the scenes.Used for gui-less operation.
- void * **dummyDisplay**

  Pointer to the dummy display.Used for gui-less operation.
- uint64_t **dummyWindowId**

  ID for a dummy window. Used for gui-less operation.

Additional Inherited Members

10.156.1 Detailed Description

Adaptor to Ogre3d.

Provides the interface to load, initialize the rendering engine.

10.156.2 Member Enumeration Documentation

10.156.2.1 enum gazebo::rendering::RenderEngine::RenderPathType

The type of rendering path used by the rendering engine.

Enumerator:

NONE No rendering is done.

VERTEX Most basic rendering, with least fidelity.

FORWARD Utilizes the RTT shader system.

DEFERRED Utilizes deferred rendering. Best fidelity.

RENDER_PATH_COUNT Count of the rendering path enums.

10.156.3 Member Function Documentation

10.156.3.1 void gazebo::rendering::RenderEngine::AddResourcePath (const std::string & *_uri*)

Add a new path for **Ogre** (p. 118) to search for resources.

Parameters

in	*_uri*	URI of the path. The uri should be of the form `file://` or model://

10.156.3.2 ScenePtr gazebo::rendering::RenderEngine::CreateScene (const std::string & _name, bool _enableVisualizations)

Create a scene.

Parameters

in	*_name*	The name of the scene.
in	*_enable-Visualizations*	True enables visualization elements such as laser lines.

10.156.3.3 void gazebo::rendering::RenderEngine::Fini ( )

Tears down the rendering engine.

10.156.3.4 RenderPathType gazebo::rendering::RenderEngine::GetRenderPathType ( ) const

Get the type of rendering path to use.

This is automatically determined based on the computers capabilities

Returns

The RenderPathType

10.156.3.5 ScenePtr gazebo::rendering::RenderEngine::GetScene (const std::string & _name)

Get a scene by name.

Parameters

in	*_name*	Name of the scene to retreive.

Returns

A pointer to the **Scene** (p. 746), or NULL if the scene doesn't exist.

10.156.3.6 ScenePtr gazebo::rendering::RenderEngine::GetScene (unsigned int _index)

Get a scene by index.

The index should be between 0 and **GetSceneCount()** (p. 726).

Parameters

in	*_index*	The index of the scene.

Returns

A pointer to a **Scene** (p. 746), or NULL if the index was invalid.

10.156.3.7 unsigned int gazebo::rendering::RenderEngine::GetSceneCount ( ) const

Get the number of scenes.

Returns

The number of scenes created by the **RenderEngine** (p. 722).

10.156.3.8 void gazebo::rendering::RenderEngine::Init ( )

Initialize **Ogre** (p. 118). Load must happen before Init.

10.156.3.9 void gazebo::rendering::RenderEngine::Load ( )

Load the parameters for **Ogre** (p. 118). Load must happen before Init.

10.156.3.10 void gazebo::rendering::RenderEngine::RemoveScene (const std::string & *_name*)

Remove a scene.

Parameters

in	*_name*	The name of the scene to remove.

10.156.4 Member Data Documentation

10.156.4.1 void* gazebo::rendering::RenderEngine::dummyContext [protected]

GLX context used to render the scenes.Used for gui-less operation.

10.156.4.2 void* gazebo::rendering::RenderEngine::dummyDisplay [protected]

Pointer to the dummy display.Used for gui-less operation.

10.156.4.3 uint64_t gazebo::rendering::RenderEngine::dummyWindowId [protected]

ID for a dummy window. Used for gui-less operation.

10.156.4.4 Ogre::Root* gazebo::rendering::RenderEngine::root

Pointer to the root scene node.

The documentation for this class was generated from the following file:

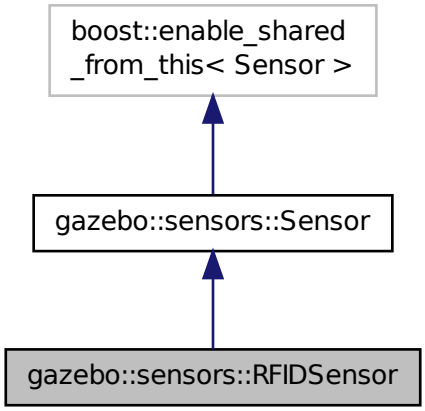
- **RenderEngine.hh**

10.157 gazebo::sensors::RFIDSensor Class Reference

Sensor (p. 765) class for RFID type of sensor.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::RFIDSensor:



Public Member Functions

- **RFIDSensor** ()

  Constructor.
- virtual ~**RFIDSensor** ()

  Destructor.
- virtual void **Fini** ()

  Finalize the sensor.
- virtual void **Init** ()

  Initialize the sensor.
- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** _sdf)

  Load the sensor with SDF parameters.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  This gets overwritten by derived sensor types.

Additional Inherited Members

10.157.1 Detailed Description

Sensor (p. 765) class for RFID type of sensor.

10.157.2 Constructor & Destructor Documentation

10.157.2.1 gazebo::sensors::RFIDSensor::RFIDSensor ( )

Constructor.

10.157.2.2 virtual gazebo::sensors::RFIDSensor::~RFIDSensor ( ) [virtual]

Destructor.

10.157.3 Member Function Documentation

10.157.3.1 virtual void gazebo::sensors::RFIDSensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

10.157.3.2 virtual void gazebo::sensors::RFIDSensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.157.3.3 virtual void gazebo::sensors::RFIDSensor::Load (const std::string & *_worldName*, sdf::ElementPtr *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 765) parameters
in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.157.3.4 virtual void gazebo::sensors::RFIDSensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.157.3.5 virtual void gazebo::sensors::RFIDSensor::UpdateImpl (bool) [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 771).

The documentation for this class was generated from the following file:

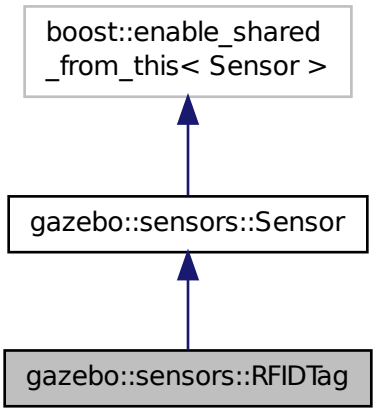
- **RFIDSensor.hh**

10.158 gazebo::sensors::RFIDTag Class Reference

RFIDTag (p. 729) to interact with RFIDTagSensors Nate check.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::RFIDTag:



Public Member Functions

- **RFIDTag** ()

Constructor.

- virtual ∼**RFIDTag** ()

  Destructor.

- virtual void **Fini** ()

  Finalize the sensor.

- **math::Pose GetTagPose** ()

  returns pose of tag in world coordinate

- virtual void **Init** ()

  Initialize the sensor.

- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** &_sdf)

  Load the sensor with SDF parameters.

- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.

Additional Inherited Members

10.158.1 Detailed Description

RFIDTag (p. 729) to interact with RFIDTagSensors Nate check.

10.158.2 Constructor & Destructor Documentation

10.158.2.1 gazebo::sensors::RFIDTag::RFIDTag ( )

Constructor.

10.158.2.2 virtual gazebo::sensors::RFIDTag::∼RFIDTag ( ) [virtual]

Destructor.

10.158.3 Member Function Documentation

10.158.3.1 virtual void gazebo::sensors::RFIDTag::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 768).

10.158.3.2 math::Pose gazebo::sensors::RFIDTag::GetTagPose ( ) [inline]

returns pose of tag in world coordinate

Returns

Pose of object

References gazebo::physics::Entity::GetWorldPose().

10.158.3.3 virtual void gazebo::sensors::RFIDTag::Init ( ) [virtual]

Initialize the sensor.

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.158.3.4 virtual void gazebo::sensors::RFIDTag::Load (const std::string & *_worldName*, sdf::ElementPtr & *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 765) parameters
in	*_worldName*	Name of world to load from

10.158.3.5 virtual void gazebo::sensors::RFIDTag::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented from **gazebo::sensors::Sensor** (p. 770).

10.158.3.6 virtual void gazebo::sensors::RFIDTag::UpdateImpl (bool *_force*) [protected], [virtual]

Update the sensor information.

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented from **gazebo::sensors::Sensor** (p. 771).

The documentation for this class was generated from the following file:

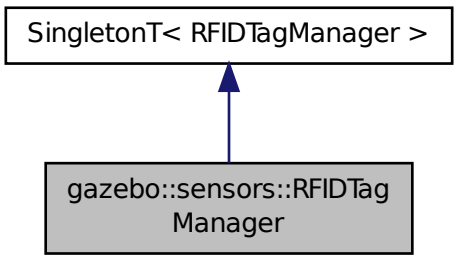
- **RFIDTag.hh**

10.159 gazebo::sensors::RFIDTagManager Class Reference

Nate fill in

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::RFIDTagManager:



Public Member Functions

- void **AddTaggedModel** (**RFIDTag** *_model)
  Adds tag model.
- std::vector< **RFIDTag** * > **GetTags** ()
  Gets vector of tagged models.

Additional Inherited Members

10.159.1 Detailed Description

Nate fill in

10.159.2 Member Function Documentation

10.159.2.1 void gazebo::sensors::RFIDTagManager::AddTaggedModel (RFIDTag * *_model*)

Adds tag model.

Parameters

_model	Tagged model to add Nate check

10.159.2.2 std::vector<RFIDTag*> gazebo::sensors::RFIDTagManager::GetTags ( ) [inline]

Gets vector of tagged models.

Returns

vector of **RFIDTag** (p. 729), the tagged models Nate check

The documentation for this class was generated from the following file:

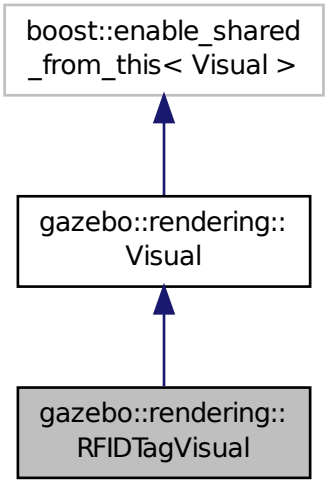
- **RFIDTagManager.hh**

10.160 gazebo::rendering::RFIDTagVisual Class Reference

Visualization for RFID tags sensor.

`#include <RFIDTagVisual.hh>`

Inheritance diagram for gazebo::rendering::RFIDTagVisual:



Public Member Functions

- **RFIDTagVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

  Constructor.
- virtual ~**RFIDTagVisual** ()

  Destructor.

Additional Inherited Members

10.160.1 Detailed Description

Visualization for RFID tags sensor.

10.160.2 Constructor & Destructor Documentation

10.160.2.1 gazebo::rendering::RFIDTagVisual::RFIDTagVisual (const std::string & *_name,* 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. 727)

10.160.2.2 virtual gazebo::rendering::RFIDTagVisual::~RFIDTagVisual ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

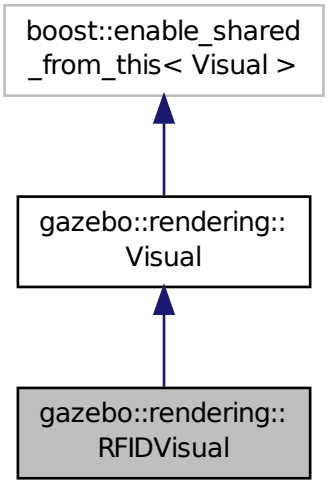
- **RFIDTagVisual.hh**

10.161 gazebo::rendering::RFIDVisual Class Reference

Visualization for RFID sensor.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RFIDVisual:



Public Member Functions

- **RFIDVisual** (const std::string &_name, **VisualPtr** _vis, const std::string &_topicName)

  Constructor.
- virtual ~**RFIDVisual** ()

  Destructor.

Additional Inherited Members

10.161.1 Detailed Description

Visualization for RFID sensor.

10.161.2 Constructor & Destructor Documentation

10.161.2.1 gazebo::rendering::RFIDVisual::RFIDVisual (const std::string & *_name*, VisualPtr *_vis*, const std::string & *_topicName*)

Constructor.

Parameters

in	*_name*	Name of the **Visual** (p. 931).
in	*_vis*	Parent **Visual** (p. 931).
in	*_topicName*	Name of the topic which publishes RFID data.

10.161.2.2 virtual gazebo::rendering::RFIDVisual::~RFIDVisual ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

- **RFIDVisual.hh**

10.162 Road Class Reference

Used to render a strip of road.

`#include <rendering/rendering.hh>`

10.162.1 Detailed Description

Used to render a strip of road.

The documentation for this class was generated from the following file:

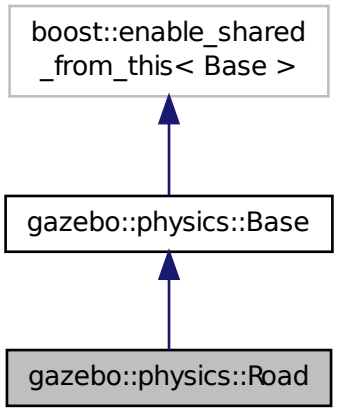
- **Road2d.hh**

10.163 gazebo::physics::Road Class Reference

for building a **Road** (p. 736) from SDF

`#include <Road.hh>`

Inheritance diagram for gazebo::physics::Road:



Public Member Functions

- **Road** (**BasePtr** _parent)
  Constructor.
- virtual ~**Road** ()
  Destructor.
- virtual void **Init** ()
  Initialize the object.
- void **Load** (**sdf::ElementPtr** _elem)
  Load the road from SDF.

Additional Inherited Members

10.163.1 Detailed Description

for building a **Road** (p. 736) from SDF

10.163.2 Constructor & Destructor Documentation

10.163.2.1 gazebo::physics::Road::Road (BasePtr _parent)

Constructor.

10.163.2.2 virtual gazebo::physics::Road::~Road ( ) [virtual]

Destructor.

10.163.3 Member Function Documentation

10.163.3.1 virtual void gazebo::physics::Road::Init ( ) [virtual]

Initialize the object.

Reimplemented from **gazebo::physics::Base** (p. 153).

10.163.3.2 void gazebo::physics::Road::Load (sdf::ElementPtr _elem) [virtual]

Load the road from SDF.

Reimplemented from **gazebo::physics::Base** (p. 154).

The documentation for this class was generated from the following file:

- **Road.hh**

10.164 gazebo::rendering::Road2d Class Reference

`#include <Road2d.hh>`

Public Member Functions

- **Road2d** ()

  Constructor.
- virtual ~**Road2d** ()

  Destructor.
- void **Load** (**VisualPtr** _parent)

  Load the visual using a parent visual.

10.164.1 Constructor & Destructor Documentation

10.164.1.1 gazebo::rendering::Road2d::Road2d ( )

Constructor.

10.164.1.2 virtual gazebo::rendering::Road2d::~Road2d ( ) [virtual]

Destructor.

10.164.2 Member Function Documentation

10.164.2.1 void gazebo::rendering::Road2d::Load (VisualPtr *_parent*)

Load the visual using a parent visual.

Parameters

in	*_parent*	Pointer to the parent visual.

The documentation for this class was generated from the following file:

- **Road2d.hh**

10.165 gazebo::math::RotationSpline Class Reference

Spline (p. 808) for rotations.

`#include <RotationSpline.hh>`

Public Member Functions

- **RotationSpline** ()

  Constructor. Sets the autoCalc to true.
- ~**RotationSpline** ()

  Destructor. Nothing is done.
- void **AddPoint** (const **Quaternion** &_p)

  Adds a control point to the end of the spline.
- void **Clear** ()

Clears all the points in the spline.

- unsigned int **GetNumPoints** () const

  Gets the number of control points in the spline.
- const **Quaternion** & **GetPoint** (unsigned int _index) const

  Gets the detail of one of the control points of the spline.
- **Quaternion Interpolate** (double _t, bool _useShortestPath=true)

  Returns an interpolated point based on a parametric value over the whole series.
- **Quaternion Interpolate** (unsigned int _fromIndex, double _t, bool _useShortestPath=true)

  Interpolates a single segment of the spline given a parametric value.
- void **RecalcTangents** ()

  Recalculates the tangents associated with this spline.
- void **SetAutoCalculate** (bool _autoCalc)

  Tells the spline whether it should automatically calculate tangents on demand as points are added.
- void **UpdatePoint** (unsigned int _index, const **Quaternion** &_value)

  Updates a single point in the spline.

Protected Attributes

- bool **autoCalc**

  Automatic recalcultation of tangeants when control points are updated.
- std::vector< **Quaternion** > **points**

  the control points
- std::vector< **Quaternion** > **tangents**

  the tangents

10.165.1 Detailed Description

Spline (p. 808) for rotations.

10.165.2 Constructor & Destructor Documentation

10.165.2.1 gazebo::math::RotationSpline::RotationSpline ( )

Constructor. Sets the autoCalc to true.

10.165.2.2 gazebo::math::RotationSpline::~RotationSpline ( )

Destructor. Nothing is done.

10.165.3 Member Function Documentation

10.165.3.1 void gazebo::math::RotationSpline::AddPoint (const Quaternion & _p)

Adds a control point to the end of the spline.

Parameters

in	*_p*	control point

10.165.3.2 void gazebo::math::RotationSpline::Clear ( )

Clears all the points in the spline.

10.165.3.3 unsigned int gazebo::math::RotationSpline::GetNumPoints ( ) const

Gets the number of control points in the spline.

Returns

the count

10.165.3.4 const Quaternion& gazebo::math::RotationSpline::GetPoint (unsigned int *_index*) const

Gets the detail of one of the control points of the spline.

Parameters

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)

10.165.3.5 Quaternion gazebo::math::RotationSpline::Interpolate (double *_t*, bool *_useShortestPath* = `true`)

Returns an interpolated point based on a parametric value over the whole series.

Remarks

Given a t value between 0 and 1 representing the parametric distance along the whole length of the spline, this method returns an interpolated point.

Parameters

in	*_t*	Parametric value.
in	*_useShortest-Path*	Defines if rotation should take the shortest possible path

Returns

the rotation

10.165.3.6 Quaternion gazebo::math::RotationSpline::Interpolate (unsigned int *_fromIndex,* double *_t,* bool *_useShortestPath =* `true`)

Interpolates a single segment of the spline given a parametric value.

Parameters

in	*_fromIndex*	The point index to treat as t = 0. _fromIndex + 1 is deemed to be t = 1
in	*_t*	Parametric value
in	*_useShortest-Path*	Defines if rotation should take the shortest possible path

Returns

the rotation

10.165.3.7 void gazebo::math::RotationSpline::RecalcTangents ( )

Recalculates the tangents associated with this spline.

Remarks

If you tell the spline not to update on demand by calling setAutoCalculate(false) then you must call this after completing your updates to the spline points.

10.165.3.8 void gazebo::math::RotationSpline::SetAutoCalculate (bool *_autoCalc*)

Tells the spline whether it should automatically calculate tangents on demand as points are added.

Remarks

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the 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.

10.165.3.9 void gazebo::math::RotationSpline::UpdatePoint (unsigned int *_index,* const Quaternion & *_value*)

Updates a single point in the spline.

Remarks

This point must already exist in the spline.

Parameters

in	*_index*	index
in	*_value*	the new control point value

10.165.4 Member Data Documentation

10.165.4.1 bool gazebo::math::RotationSpline::autoCalc [protected]

Automatic recalcultation of tangeants when control points are updated.

10.165.4.2 std::vector<Quaternion> gazebo::math::RotationSpline::points [protected]

the control points

10.165.4.3 std::vector<Quaternion> gazebo::math::RotationSpline::tangents [protected]

the tangents

The documentation for this class was generated from the following file:

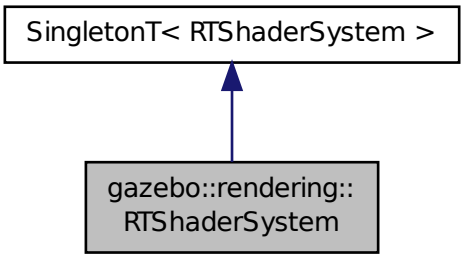
- **RotationSpline.hh**

10.166 gazebo::rendering::RTShaderSystem Class Reference

Implements **Ogre** (p. 118)'s Run-Time Shader system.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::RTShaderSystem:



Public Types

- enum **LightingModel** { **SSLM_PerVertexLighting**, **SSLM_PerPixelLighting**, **SSLM_NormalMapLighting-TangentSpace**, **SSLM_NormalMapLightingObjectSpace** }

Public Member Functions

- void **AddScene** (**Scene** ∗_scene)

  Add a scene manager.
- void **ApplyShadows** (**Scene** ∗_scene)

  Apply shadows to a scene.
- void **AttachEntity** (**Visual** ∗vis)

  Set an Ogre::Entity to use RT shaders.
- void **Clear** ()

  Clear the shader system.
- void **DetachEntity** (**Visual** ∗_vis)

  Remove and entity.
- void **Fini** ()

  Finalize the shader system.
- void **GenerateShaders** (**Visual** ∗_vis)

  Generate shaders for an entity.
- void **Init** ()

  Init the run time shader system.
- void **RemoveScene** (**Scene** ∗_scene)

  Remove a scene.
- void **RemoveShadows** (**Scene** ∗_scene)

  Remove shadows from a scene.
- void **SetPerPixelLighting** (bool _set)

  Set the lighting model to per pixel or per vertex.
- void **UpdateShaders** ()

  Update the shaders. This should not be called frequently.

Static Public Member Functions

- static void **AttachViewport** (Ogre::Viewport ∗_viewport, **Scene** ∗_scene)

  Set a viewport to use shaders.
- static void **DetachViewport** (Ogre::Viewport ∗_viewport, **Scene** ∗_scene)

  Set a viewport to not use shaders.

Additional Inherited Members

10.166.1 Detailed Description

Implements **Ogre** (p. 118)'s Run-Time Shader system.

This class allows Gazebo to generate per-pixel shaders for every material at run-time.

10.166.2 Member Enumeration Documentation

10.166.2.1 enum gazebo::rendering::RTShaderSystem::LightingModel

The type of lighting.

Enumerator:

SSLM_PerVertexLighting Per-Vertex lighting: best performance.

SSLM_PerPixelLighting Per-Pixel lighting: best look.

SSLM_NormalMapLightingTangentSpace Normal Map lighting: lighting calculations have been stored in a light map (texture) using tangent space.

SSLM_NormalMapLightingObjectSpace Normal Map lighting: lighting calculations have been stored in a light map (texture) using object space.

10.166.3 Member Function Documentation

10.166.3.1 void gazebo::rendering::RTShaderSystem::AddScene (Scene * _scene)

Add a scene manager.

Parameters

in	*_scene*	The scene to process

10.166.3.2 void gazebo::rendering::RTShaderSystem::ApplyShadows (Scene * _scene)

Apply shadows to a scene.

Parameters

in	*_scene*	The scene to receive shadows.

10.166.3.3 void gazebo::rendering::RTShaderSystem::AttachEntity (Visual * *vis*)

Set an Ogre::Entity to use RT shaders.

Parameters

in	*_vis*	**Visual** (p. 931) that will use the **RTShaderSystem** (p. 742).

10.166.3.4 static void gazebo::rendering::RTShaderSystem::AttachViewport (Ogre::Viewport * _viewport, Scene * _scene) [static]

Set a viewport to use shaders.

Parameters

in	*_viewport*	The viewport to add.
in	*_scene*	The scene that the viewport uses.

10.166.3.5 void gazebo::rendering::RTShaderSystem::Clear ( )

Clear the shader system.

10.166.3.6 void gazebo::rendering::RTShaderSystem::DetachEntity (Visual * _vis)

Remove and entity.

Parameters

in	*_vis*	Remove this visual.

10.166.3.7 static void gazebo::rendering::RTShaderSystem::DetachViewport (Ogre::Viewport * _viewport, Scene * _scene) [static]

Set a viewport to not use shaders.

Parameters

in	*_viewport*	The viewport to remove.
in	*_scene*	The scene that the viewport uses.

10.166.3.8 void gazebo::rendering::RTShaderSystem::Fini ( )

Finalize the shader system.

10.166.3.9 void gazebo::rendering::RTShaderSystem::GenerateShaders (Visual * _vis)

Generate shaders for an entity.

Parameters

in	*_vis*	The visual to generate shaders for.

10.166.3.10 void gazebo::rendering::RTShaderSystem::Init ( )

Init the run time shader system.

10.166.3.11 void gazebo::rendering::RTShaderSystem::RemoveScene (Scene * _scene)

Remove a scene.

Parameters

in	*The*	scene to remove

10.166.3.12 void gazebo::rendering::RTShaderSystem::RemoveShadows (Scene * _scene)

Remove shadows from a scene.

Parameters

in	_scene	The scene to remove shadows from.

10.166.3.13 void gazebo::rendering::RTShaderSystem::SetPerPixelLighting (bool _set)

Set the lighting model to per pixel or per vertex.

Parameters

in	_set	True means to use per-pixel shaders.

10.166.3.14 void gazebo::rendering::RTShaderSystem::UpdateShaders ( )

Update the shaders. This should not be called frequently.

The documentation for this class was generated from the following file:

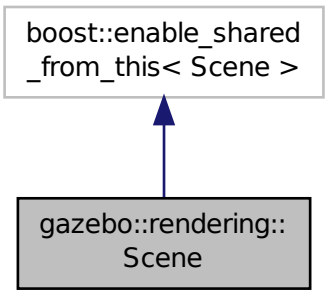
- **RTShaderSystem.hh**

10.167 gazebo::rendering::Scene Class Reference

Representation of an entire scene graph.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::Scene:



Public Member Functions

- **Scene** (const std::string &_name, bool _enableVisualizations=false)

  Constructor.
- virtual ~**Scene** ()

  Destructor.
- void **AddVisual** (**VisualPtr** _vis)

  Add a visual to the scene.
- void **Clear** ()

  *Clear **rendering::Scene** (p. 746).*
- **VisualPtr CloneVisual** (const std::string &_visualName, const std::string &_newName)

  Clone a visual.
- **CameraPtr CreateCamera** (const std::string &_name, bool _autoRender=true)

  Create a camera.
- **DepthCameraPtr CreateDepthCamera** (const std::string &_name, bool _autoRender=true)

  Create depth camera.
- **GpuLaserPtr CreateGpuLaser** (const std::string &_name, bool _autoRender=true)

  Create laser that generates data from rendering.
- void **CreateGrid** (uint32_t _cellCount, float _cellLength, float _lineWidth, const **common::Color** &_color)

  Create a square grid of cells.
- **UserCameraPtr CreateUserCamera** (const std::string &_name)

  Create a user camera.
- void **DrawLine** (const **math::Vector3** &_start, const **math::Vector3** &_end, const std::string &_name)

  Draw a named line.
- **common::Color GetAmbientColor** () const

  Get the ambient color.
- **common::Color GetBackgroundColor** () const

  Get the background color.
- **CameraPtr GetCamera** (uint32_t _index) const

  Get a camera based on an index.
- **CameraPtr GetCamera** (const std::string &_name) const

  Get a camera by name.
- uint32_t **GetCameraCount** () const

  Get the number of cameras in this scene.
- **math::Vector3 GetFirstContact** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos)

  Get the world pos of a the first contact at a pixel location.
- **Grid** * **GetGrid** (uint32_t _index) const

  Get a grid based on an index.
- uint32_t **GetGridCount** () const

  Get the number of grids.
- **Heightmap** * **GetHeightmap** () const

  Get a pointer to the heightmap.
- uint32_t **GetId** () const

  Get the scene ID.
- std::string **GetIdString** () const

  Get the scene Id as a string.
- **LightPtr GetLight** (const std::string &_name) const

Get a light by name.

- **LightPtr GetLight** (uint32_t _index) const

  Get a light based on an index.

- uint32_t **GetLightCount** () const

  Get the count of the lights.

- Ogre::SceneManager * **GetManager** () const

  Get the OGRE scene manager.

- **VisualPtr GetModelVisualAt** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos)

  Get a model's visual at a mouse position.

- std::string **GetName** () const

  Get the name of the scene.

- **VisualPtr GetSelectedVisual** () const

  Get the currently selected visual.

- bool **GetShadowsEnabled** () const

  Get whether shadows are on or off.

- **UserCameraPtr GetUserCamera** (uint32_t _index) const

  Get a user camera by index.

- uint32_t **GetUserCameraCount** () const

  Get the number of user cameras in this scene.

- **VisualPtr GetVisual** (const std::string &_name) const

  Get a visual by name.

- **VisualPtr GetVisualAt** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos, std::string &mod)

  Get an entity at a pixel location using a camera.

- **VisualPtr GetVisualAt** (**CameraPtr** _camera, const **math::Vector2i** &_mousePos)

  Get a visual at a mouse position.

- **VisualPtr GetVisualBelow** (const std::string &_visualName)

  Get the closest visual below a given visual.

- void **GetVisualsBelowPoint** (const **math::Vector3** &_pt, std::vector< **VisualPtr** > &_visuals)

  Get a visual directly below a point.

- **VisualPtr GetWorldVisual** () const

  Get the top level world visual.

- void **Init** ()

  *Init **rendering::Scene*** (p. 746).

- void **Load** (**sdf::ElementPtr** _scene)

  Load the scene from a set of parameters.

- void **Load** ()

  Load the scene with default parameters.

- void **PreRender** ()

  Process all received messages.

- void **PrintSceneGraph** ()

  Print the scene graph to std_out.

- void **RemoveVisual** (**VisualPtr** _vis)

  Remove a visual from the scene.

- void **SelectVisual** (const std::string &_name, const std::string &_mode)

  Select a visual by name.

- void **SetAmbientColor** (const **common::Color** &_color)

  Set the ambient color.

- void **SetBackgroundColor** (const **common::Color** &_color)

  Set the background color.
- void **SetFog** (const std::string &_type, const **common::Color** &_color, double _density, double _start, double _end)

  Set the fog parameters.
- void **SetGrid** (bool _enabled)

  Set the grid on or off.
- void **SetShadowsEnabled** (bool _value)

  Set whether shadows are on or off.
- void **SetVisible** (const std::string &_name, bool _visible)

  Hide or show a visual.
- void **SnapVisualToNearestBelow** (const std::string &_visualName)

  Move the visual to be ontop of the nearest visual below it.
- std::string **StripSceneName** (const std::string &_name) const

  Remove the name of scene from a string.

Public Attributes

- SkyX::SkyX ∗ **skyx**

  Pointer to the sky.

10.167.1 Detailed Description

Representation of an entire scene graph.

Maintains all the Visuals, Lights, and Cameras for a World.

10.167.2 Constructor & Destructor Documentation

10.167.2.1 gazebo::rendering::Scene::Scene (const std::string & *_name,* bool *_enableVisualizations* = `false`)

Constructor.

Parameters

in	*_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.

10.167.2.2 virtual gazebo::rendering::Scene::∼Scene ( ) [virtual]

Destructor.

10.167.3 Member Function Documentation

10.167.3.1 void gazebo::rendering::Scene::AddVisual (VisualPtr *_vis*)

Add a visual to the scene.

Parameters

in	*_vis*	**Visual** (p. 931) to add.

10.167.3.2 void gazebo::rendering::Scene::Clear ( )

Clear **rendering::Scene** (p. 746).

10.167.3.3 VisualPtr gazebo::rendering::Scene::CloneVisual (const std::string & *_visualName,* const std::string & *_newName*)

Clone a visual.

Parameters

in	*_visualName*	Name of the visual to clone.
in	*_newName*	New name of the visual.

Returns

Pointer to the cloned visual.

10.167.3.4 CameraPtr gazebo::rendering::Scene::CreateCamera (const std::string & *_name,* bool *_autoRender* = `true`)

Create a camera.

Parameters

in	*_name*	Name of the new camera.
in	*_autoRender*	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.167.3.5 DepthCameraPtr gazebo::rendering::Scene::CreateDepthCamera (const std::string & *_name,* bool *_autoRender* = `true`)

Create depth camera.

Parameters

in	*_name*	Name of the new camera.
in	*_autoRender*	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new camera.

10.167.3.6 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	*_autoRender*	True to allow Gazebo to automatically render the camera. This should almost always be true.

Returns

Pointer to the new laser.

10.167.3.7 void gazebo::rendering::Scene::CreateGrid (uint32_t _cellCount, float _cellLength, float _lineWidth, const common::Color & _color)

Create a square grid of cells.

Parameters

in	*_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.

10.167.3.8 UserCameraPtr gazebo::rendering::Scene::CreateUserCamera (const std::string & _name)

Create a user camera.

A user camera is one design for use with a GUI.

Parameters

in	*_name*	Name of the **UserCamera** (p. 878).

Returns

A pointer to the new **UserCamera** (p. 878).

10.167.3.9 void gazebo::rendering::Scene::DrawLine (const math::Vector3 & _start, const math::Vector3 & _end, const std::string & _name)

Draw a named line.

Parameters

in	*_start*	Start position of the line.
in	*_end*	End position of the line.
in	*_name*	Name of the line.

10.167.3.10 common::Color gazebo::rendering::Scene::GetAmbientColor ( ) const

Get the ambient color.

Returns

The scene's ambient color.

10.167.3.11 common::Color gazebo::rendering::Scene::GetBackgroundColor ( ) const

Get the background color.

Returns

The background color.

10.167.3.12 CameraPtr gazebo::rendering::Scene::GetCamera (uint32_t *_index*) const

Get a camera based on an index.

Index must be between 0 and **Scene::GetCameraCount** (p. 752).

Parameters

in	*_index*	Index of the camera to get.

Returns

Pointer to the camera. Or NULL if the index is invalid.

10.167.3.13 CameraPtr gazebo::rendering::Scene::GetCamera (const std::string & *_name*) const

Get a camera by name.

Parameters

in	*_name*	Name of the camera.

Returns

Pointer to the camera. Or NULL if the name is invalid.

10.167.3.14 uint32_t gazebo::rendering::Scene::GetCameraCount ( ) const

Get the number of cameras in this scene.

Returns

Number of lasers.

10.167.3.15 math::Vector3 gazebo::rendering::Scene::GetFirstContact (CameraPtr _camera, const math::Vector2i & _mousePos)

Get the world pos of a the first contact at a pixel location.

Parameters

in	*_camera*	Pointer to the camera.
in	*_mousePos*	2D position of the mouse in pixels.

Returns

3D position of the first contact point.

10.167.3.16 Grid* gazebo::rendering::Scene::GetGrid (uint32_t _index) const

Get a grid based on an index.

Index must be between 0 and **Scene::GetGridCount** (p. 753).

Parameters

in	*_index*	Index of the grid.

10.167.3.17 uint32_t gazebo::rendering::Scene::GetGridCount ( ) const

Get the number of grids.

Returns

The number of grids.

10.167.3.18 Heightmap* gazebo::rendering::Scene::GetHeightmap ( ) const

Get a pointer to the heightmap.

Returns

Pointer to the heightmap, NULL if no heightmap.

10.167.3.19 uint32_t gazebo::rendering::Scene::GetId ( ) const

Get the scene ID.

Returns

The ID of the scene.

10.167.3.20 std::string gazebo::rendering::Scene::GetIdString ( ) const

Get the scene Id as a string.

Returns

The ID as a string.

10.167.3.21 LightPtr gazebo::rendering::Scene::GetLight (const std::string & *_name*) const

Get a light by name.

Parameters

in	*_name*	Name of the light to get.

Returns

Pointer to the light, or NULL if the light was not found.

10.167.3.22 LightPtr gazebo::rendering::Scene::GetLight (uint32_t *_index*) const

Get a light based on an index.

The index must be between 0 and **Scene::GetLightCount** (p. 754).

Parameters

in	*_index*	Index of the light.

Returns

Pointer to the **Light** (p. 448) or NULL if index was invalid.

10.167.3.23 uint32_t gazebo::rendering::Scene::GetLightCount ( ) const

Get the count of the lights.

Returns

The number of lights.

10.167.3.24 Ogre::SceneManager* gazebo::rendering::Scene::GetManager ( ) const

Get the OGRE scene manager.

Returns

Pointer to the **Ogre** (p. 118) SceneManager.

10.167.3.25 VisualPtr gazebo::rendering::Scene::GetModelVisualAt (CameraPtr *_camera*, const math::Vector2i & *_mousePos*)

Get a model's visual at a mouse position.

Parameters

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.

10.167.3.26 std::string gazebo::rendering::Scene::GetName ( ) const

Get the name of the scene.

Returns

Name of the scene.

10.167.3.27 VisualPtr gazebo::rendering::Scene::GetSelectedVisual ( ) const

Get the currently selected visual.

Returns

Pointer to the currently selected visual, or NULL if nothing is selected.

10.167.3.28 bool gazebo::rendering::Scene::GetShadowsEnabled ( ) const

Get whether shadows are on or off.

Returns

True if shadows are enabled.

10.167.3.29 UserCameraPtr gazebo::rendering::Scene::GetUserCamera (uint32_t *_index*) const

Get a user camera by index.

The index value must be between 0 and **Scene::GetUserCameraCount** (p. 756).

Parameters

in	*_index*	Index of the **UserCamera** (p. 878) to get.

Returns

Pointer to the **UserCamera** (p. 878), or NULL if the index was invalid.

10.167.3.30 uint32_t gazebo::rendering::Scene::GetUserCameraCount ( ) const

Get the number of user cameras in this scene.

Returns

The number of user cameras.

10.167.3.31 VisualPtr gazebo::rendering::Scene::GetVisual (const std::string & *_name*) const

Get a visual by name.

10.167.3.32 VisualPtr gazebo::rendering::Scene::GetVisualAt (CameraPtr *_camera,* const math::Vector2i & *_mousePos,* std::string & *mod*)

Get an entity at a pixel location using a camera.

Used for mouse picking.

Parameters

in	*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

10.167.3.33 VisualPtr gazebo::rendering::Scene::GetVisualAt (CameraPtr *_camera,* const math::Vector2i & *_mousePos*)

Get a visual at a mouse position.

Parameters

in	*_camera*	Pointer to the camera used to project the mouse position.
in	*_mousePos*	The 2d position of the mouse in pixels.

Returns

Pointer to the visual, NULL if none found.

10.167.3.34 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.

10.167.3.35 void gazebo::rendering::Scene::GetVisualsBelowPoint (const math::Vector3 & *_pt*, std::vector< VisualPtr > & *_visuals*)

Get a visual directly below a point.

Parameters

in	*_pt*	3D point to get the visual below.
out	*_visuals*	The visuals below the point order in proximity.

10.167.3.36 VisualPtr gazebo::rendering::Scene::GetWorldVisual ( ) const

Get the top level world visual.

Returns

Pointer to the world visual.

10.167.3.37 void gazebo::rendering::Scene::Init ( )

Init **rendering::Scene** (p. 746).

10.167.3.38 void gazebo::rendering::Scene::Load (sdf::ElementPtr *_scene*)

Load the scene from a set of parameters.

Parameters

in	*_scene*	SDF scene element to load.

10.167.3.39 void gazebo::rendering::Scene::Load ( )

Load the scene with default parameters.

10.167.3.40 void gazebo::rendering::Scene::PreRender ( )

Process all received messages.

10.167.3.41 void gazebo::rendering::Scene::PrintSceneGraph ( )

Print the scene graph to std_out.

10.167.3.42 void gazebo::rendering::Scene::RemoveVisual (VisualPtr *_vis*)

Remove a visual from the scene.

Parameters

in	*_vis*	**Visual** (p. 931) to remove.

10.167.3.43 void gazebo::rendering::Scene::SelectVisual (const std::string & *_name*, const std::string & *_mode*)

Select a visual by name.

Parameters

in	*_name*	Name of the visual to select.
in	*_mode*	Selection mode (normal, or move).

10.167.3.44 void gazebo::rendering::Scene::SetAmbientColor (const common::Color & *_color*)

Set the ambient color.

Parameters

in	*_color*	The ambient color to use.

10.167.3.45 void gazebo::rendering::Scene::SetBackgroundColor (const common::Color & *_color*)

Set the background color.

Parameters

in	*_color*	The background color.

10.167.3.46 void gazebo::rendering::Scene::SetFog (const std::string & *_type*, const common::Color & *_color*, double *_density*, double *_start*, double *_end*)

Set the fog parameters.

Parameters

in	*_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.

10.167.3.47 void gazebo::rendering::Scene::SetGrid (bool _enabled)

Set the grid on or off.

Parameters

in	*_enabled*	Set to true to turn on the grid

10.167.3.48 void gazebo::rendering::Scene::SetShadowsEnabled (bool _value)

Set whether shadows are on or off.

Parameters

in	*_value*	True to enable shadows, False to disable

10.167.3.49 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.

10.167.3.50 void gazebo::rendering::Scene::SnapVisualToNearestBelow (const std::string & _visualName)

Move the visual to be ontop of the nearest visual below it.

Parameters

in	*_visualName*	Name of the visual to move.

10.167.3.51 std::string gazebo::rendering::Scene::StripSceneName (const std::string & _name) const

Remove the name of scene from a string.

Parameters

in	*_name*	Name to string the scene name from.

Returns

The stripped name.

10.167.4 Member Data Documentation

10.167.4.1 SkyX::SkyX* gazebo::rendering::Scene::skyx

Pointer to the sky.

The documentation for this class was generated from the following file:

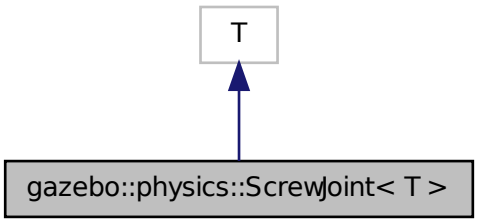
- **Scene.hh**

10.168 gazebo::physics::ScrewJoint< T > Class Template Reference

A screw joint.

`#include <ScrewJoint.hh>`

Inheritance diagram for gazebo::physics::ScrewJoint< T >:



Public Member Functions

- **ScrewJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**ScrewJoint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (int) const

  Get the anchor.
- virtual void **SetAnchor** (int, const **math::Vector3** &anchor)

  Set the anchor.
- virtual void **SetThreadPitch** (int _index, double _threadPitch)=0

  Set screw joint thread pitch.

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load a ScreJoint.

Protected Attributes

- **math::Vector3 fakeAnchor**
- double **threadPitch**

10.168.1 Detailed Description

template<class T>class gazebo::physics::ScrewJoint< T >

A screw joint.

10.168.2 Constructor & Destructor Documentation

10.168.2.1 template<class T> gazebo::physics::ScrewJoint< T >::ScrewJoint (BasePtr *_parent*) [inline]

Constructor.

10.168.2.2 template<class T> virtual gazebo::physics::ScrewJoint< T >::~ScrewJoint ( ) [inline], [virtual]

Destructor.

10.168.3 Member Function Documentation

10.168.3.1 template<class T> virtual math::Vector3 gazebo::physics::ScrewJoint< T >::GetAnchor (int) const [inline], [virtual]

Get the anchor.

10.168.3.2 template<class T> virtual void gazebo::physics::ScrewJoint< T >::Load (sdf::ElementPtr *_sdf*) [inline], [protected], [virtual]

Load a ScreJoint.

Reimplemented in **gazebo::physics::BulletScrewJoint** (p. 216), and **gazebo::physics::ODEScrewJoint** (p. 624).

10.168.3.3 template<class T> virtual void gazebo::physics::ScrewJoint< T >::SetAnchor (int , const math::Vector3 & *anchor*) [inline], [virtual]

Set the anchor.

10.168.3.4 template<class T> virtual void gazebo::physics::ScrewJoint< T >::SetThreadPitch (int *_index*, double *_threadPitch*) [pure virtual]

Set screw joint thread pitch.

This must be implemented in a child class

Parameters

in	*_index*	Index of the axis.
in	*_threadPitch*	Thread pitch value.

Implemented in **gazebo::physics::ODEScrewJoint** (p. 625).

10.168.4 Member Data Documentation

10.168.4.1 template<class T> math::Vector3 gazebo::physics::ScrewJoint< T >::fakeAnchor [protected]

Referenced by gazebo::physics::ScrewJoint< BulletJoint >::GetAnchor(), and gazebo::physics::ScrewJoint< BulletJoint >::SetAnchor().

10.168.4.2 template<class T> double gazebo::physics::ScrewJoint< T >::threadPitch [protected]

Referenced by gazebo::physics::ScrewJoint< BulletJoint >::Load().

The documentation for this class was generated from the following file:

- **ScrewJoint.hh**

10.169 sdf::SDF Class Reference

Base **SDF** (p. 762) class.

`#include <SDF.hh>`

Public Member Functions

- **SDF** ()
- void **PrintDescription** ()
- void **PrintDoc** ()
- void **PrintValues** ()
- void **PrintWiki** ()
- void **SetFromString** (const std::string &_sdfData)

  *Set **SDF** (p. 762) values from a string.*
- std::string **ToString** () const
- void **Write** (const std::string &_filename)

Public Attributes

- **ElementPtr root**

Static Public Attributes

- static std::string **version**

10.169.1 Detailed Description

Base **SDF** (p. 762) class.

10.169.2 Constructor & Destructor Documentation

10.169.2.1 sdf::SDF::SDF ( )

10.169.3 Member Function Documentation

10.169.3.1 void sdf::SDF::PrintDescription ( )

10.169.3.2 void sdf::SDF::PrintDoc ( )

10.169.3.3 void sdf::SDF::PrintValues ( )

10.169.3.4 void sdf::SDF::PrintWiki ( )

10.169.3.5 void sdf::SDF::SetFromString (const std::string & *_sdfData*)

Set **SDF** (p. 762) values from a string.

10.169.3.6 std::string sdf::SDF::ToString ( ) const

10.169.3.7 void sdf::SDF::Write (const std::string & *_filename*)

10.169.4 Member Data Documentation

10.169.4.1 ElementPtr sdf::SDF::root

10.169.4.2 std::string sdf::SDF::version [static]

The documentation for this class was generated from the following file:

- **SDF.hh**

10.170 gazebo::rendering::SelectionObj Class Reference

A graphical selection object.

`#include <rendering/rendering.hh>`

Public Member Functions

- **SelectionObj** (**Scene** *_scene)

Constructor.

- virtual ~**SelectionObj** ()

  Destructor.
- void **Attach** (**VisualPtr** _visual)

  Set the position of the node.
- void **Clear** ()

  *Clear the **rendering::SelectionObj** (p. 763) object.*
- std::string **GetVisualName** () const

  Get the name of the visual the selection obj is attached to.
- void **Init** ()

  *Initialize the **rendering::SelectionObj** (p. 763) object.*
- bool **IsActive** () const

  Return true if the user is move the selection obj.
- void **SetActive** (bool _active)

  Set true if the user is moving the selection obj.
- void **SetHighlight** (const std::string &_mod)

  Highlight the selection object based on a modifier.

10.170.1 Detailed Description

A graphical selection object.

Used to draw a visual around a selected object.

10.170.2 Constructor & Destructor Documentation

10.170.2.1 gazebo::rendering::SelectionObj::SelectionObj (Scene * *_scene*)

Constructor.

Parameters

in	*_scene*	**Scene** (p. 746) to use.

10.170.2.2 virtual gazebo::rendering::SelectionObj::~SelectionObj ( ) [virtual]

Destructor.

10.170.3 Member Function Documentation

10.170.3.1 void gazebo::rendering::SelectionObj::Attach (VisualPtr *_visual*)

Set the position of the node.

Parameters

in	*This*	draws the selection object around the passed in visual.

10.170.3.2 void gazebo::rendering::SelectionObj::Clear ( )

Clear the **rendering::SelectionObj** (p. 763) object.

10.170.3.3 std::string gazebo::rendering::SelectionObj::GetVisualName ( ) const

Get the name of the visual the selection obj is attached to.

Returns

Name of the selected visual.

10.170.3.4 void gazebo::rendering::SelectionObj::Init ( )

Initialize the **rendering::SelectionObj** (p. 763) object.

10.170.3.5 bool gazebo::rendering::SelectionObj::IsActive ( ) const

Return true if the user is move the selection obj.

Returns

True if something is selected.

10.170.3.6 void gazebo::rendering::SelectionObj::SetActive (bool *_active*)

Set true if the user is moving the selection obj.

Parameters

in	*_active*	True if the user is interacting with the selection object.

10.170.3.7 void gazebo::rendering::SelectionObj::SetHighlight (const std::string & *_mod*)

Highlight the selection object based on a modifier.

Parameters

in	*_mod*	Modifier used when highlighting the selection object.

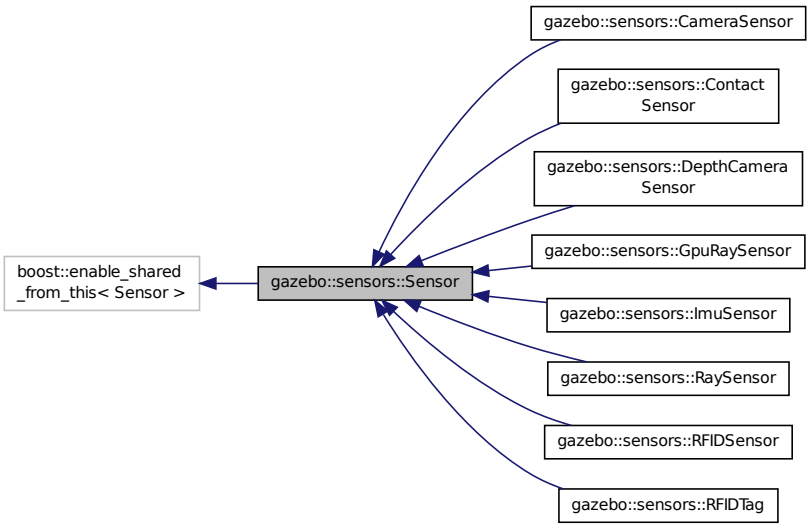
The documentation for this class was generated from the following file:

- **SelectionObj.hh**

10.171 gazebo::sensors::Sensor Class Reference

`#include <Sensor.hh>`

Inheritance diagram for gazebo::sensors::Sensor:



Public Member Functions

- **Sensor** ()

  Constructor.
- virtual ~**Sensor** ()

  Destructor.
- void **FillMsg** (msgs::Sensor &_msg)

  fills a msgs::Sensor message.
- virtual void **Fini** ()

  Finalize the sensor.
- **common::Time GetLastUpdateTime** ()

  return last update time
- std::string **GetName** () const

  Get name.
- std::string **GetParentName** () const

  Returns the name of the sensor parent.
- virtual **math::Pose GetPose** () const

  Get the current pose.
- std::string **GetScopedName** () const

  Get fully scoped name of the sensor.
- virtual std::string **GetTopic** () const

  Returns the topic name as set in SDF.
- std::string **GetType** () const

  Get sensor type.
- bool **GetVisualize** () const

return true if user requests the sensor to be visualized via tag: <visualize>true</visualize> in SDF

- std::string **GetWorldName** () const

  Returns the name of the world the sensor is in.

- virtual void **Init** ()

  Initialize the sensor.

- bool **IsActive** ()

  Returns true if sensor generation is active.

- virtual void **Load** (const std::string &_worldName, **sdf::ElementPtr** _sdf)

  Load the sensor with SDF parameters.

- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.

- virtual void **SetActive** (bool _value)

  Set whether the sensor is active or not.

- virtual void **SetParent** (const std::string &_name)

  Set the parent of the sensor.

- void **SetUpdateRate** (double _hz)

  Set the update rate of the sensor.

- void **Update** (bool _force)

  Update the sensor.

Protected Member Functions

- virtual void **UpdateImpl** (bool)

  This gets overwritten by derived sensor types.

Protected Attributes

- bool **active**

  True if active.

- std::vector< **event::ConnectionPtr** > **connections**
- **common::Time lastUpdateTime**
- **transport::NodePtr node**
- std::string **parentName**
- std::vector< **SensorPluginPtr** > **plugins**
- **math::Pose pose**
- **transport::SubscriberPtr poseSub**
- **sdf::ElementPtr sdf**
- **common::Time updatePeriod**
- **gazebo::physics::WorldPtr world**

10.171.1 Constructor & Destructor Documentation

10.171.1.1 gazebo::sensors::Sensor::Sensor ( )

Constructor.

10.171.1.2 virtual gazebo::sensors::Sensor::~Sensor ( ) [virtual]

Destructor.

10.171.2 Member Function Documentation

10.171.2.1 void gazebo::sensors::Sensor::FillMsg (msgs::Sensor & *_msg*)

fills a msgs::Sensor message.

Parameters

_msg	Message to fill

10.171.2.2 virtual void gazebo::sensors::Sensor::Fini ( ) [virtual]

Finalize the sensor.

Reimplemented in **gazebo::sensors::CameraSensor** (p. 257), **gazebo::sensors::RaySensor** (p. 713), **gazebo::sensors::ContactSensor** (p. 301), **gazebo::sensors::GpuRaySensor** (p. 382), **gazebo::sensors::DepthCameraSensor** (p. 318), **gazebo::sensors::RFIDSensor** (p. 728), and **gazebo::sensors::RFIDTag** (p. 730).

10.171.2.3 common::Time gazebo::sensors::Sensor::GetLastUpdateTime ( )

return last update time

Returns

Time of last update

10.171.2.4 std::string gazebo::sensors::Sensor::GetName ( ) const

Get name.

Returns

name of sensor

10.171.2.5 std::string gazebo::sensors::Sensor::GetParentName ( ) const

Returns the name of the sensor parent.

The parent name is set by **Sensor::SetParent** (p. 771).

Returns

Name of Parent

10.171.2.6 virtual math::Pose gazebo::sensors::Sensor::GetPose ( ) const [virtual]

Get the current pose.

Returns

Current pose of the sensor

10.171.2.7 std::string gazebo::sensors::Sensor::GetScopedName ( ) const

Get fully scoped name of the sensor.

Returns

world_name::parent_name::sensor_name

10.171.2.8 virtual std::string gazebo::sensors::Sensor::GetTopic ( ) const [virtual]

Returns the topic name as set in SDF.

Returns

Topic name

Reimplemented in **gazebo::sensors::RaySensor** (p. 716), and **gazebo::sensors::CameraSensor** (p. 258).

10.171.2.9 std::string gazebo::sensors::Sensor::GetType ( ) const

Get sensor type.

Returns

Type of sensor

10.171.2.10 bool gazebo::sensors::Sensor::GetVisualize ( ) const

return true if user requests the sensor to be visualized via tag: <visualize>true</visualize> in SDF

Returns

True if visualized, false if not

10.171.2.11 std::string gazebo::sensors::Sensor::GetWorldName ( ) const

Returns the name of the world the sensor is in.

Returns

Name of the world

10.171.2.12 virtual void gazebo::sensors::Sensor::Init ( ) [virtual]

Initialize the sensor.

Reimplemented in **gazebo::sensors::RaySensor** (p. 717), **gazebo::sensors::ContactSensor** (p. 303), **gazebo::sensors::CameraSensor** (p. 258), **gazebo::sensors::GpuRaySensor** (p. 386), **gazebo::sensors::DepthCameraSensor** (p. 319), **gazebo::sensors::RFIDSensor** (p. 728), and **gazebo::sensors::RFIDTag** (p. 731).

10.171.2.13 bool gazebo::sensors::Sensor::IsActive ( )

Returns true if sensor generation is active.

Returns

True if active, false if not

10.171.2.14 virtual void gazebo::sensors::Sensor::Load (const std::string & *_worldName*, sdf::ElementPtr *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

in	*_sdf*	SDF **Sensor** (p. 765) parameters
in	*_worldName*	Name of world to load from

Reimplemented in **gazebo::sensors::RaySensor** (p. 717), **gazebo::sensors::ContactSensor** (p. 303), **gazebo::sensors::CameraSensor** (p. 258), and **gazebo::sensors::RFIDSensor** (p. 728).

10.171.2.15 virtual void gazebo::sensors::Sensor::Load (const std::string & *_worldName*) [virtual]

Load the sensor with default parameters.

Parameters

in	*_worldName*	Name of world to load from

Reimplemented in **gazebo::sensors::RaySensor** (p. 717), **gazebo::sensors::ContactSensor** (p. 303), **gazebo::sensors::CameraSensor** (p. 258), **gazebo::sensors::GpuRaySensor** (p. 387), **gazebo::sensors::DepthCameraSensor** (p. 319), **gazebo::sensors::RFIDSensor** (p. 728), and **gazebo::sensors::RFIDTag** (p. 731).

10.171.2.16 virtual void gazebo::sensors::Sensor::SetActive (bool *_value*) [virtual]

Set whether the sensor is active or not.

Parameters

in	*value*	True if active, false if not

Reimplemented in **gazebo::sensors::CameraSensor** (p. 259), and **gazebo::sensors::DepthCameraSensor** (p. 319).

10.171.2.17 virtual void gazebo::sensors::Sensor::SetParent (const std::string & _name) [virtual]

Set the parent of the sensor.

Reimplemented in **gazebo::sensors::CameraSensor** (p. 259), and **gazebo::sensors::DepthCameraSensor** (p. 320).

10.171.2.18 void gazebo::sensors::Sensor::SetUpdateRate (double _hz)

Set the update rate of the sensor.

Parameters

in	*_hz*	update rate of sensor

10.171.2.19 void gazebo::sensors::Sensor::Update (bool _force)

Update the sensor.

Parameters

in	*_force*	True to force update, false otherwise

10.171.2.20 virtual void gazebo::sensors::Sensor::UpdateImpl (bool) [inline], [protected], [virtual]

This gets overwritten by derived sensor types.

```
This function is called during Sensor::Update.
And in turn, Sensor::Update is called by
SensorManager::Update
```

Parameters

in	*_force*	True if update is forced, false if not

Reimplemented in **gazebo::sensors::CameraSensor** (p. 259), **gazebo::sensors::RaySensor** (p. 718), **gazebo::sensors::ContactSensor** (p. 303), **gazebo::sensors::GpuRaySensor** (p. 388), **gazebo::sensors::DepthCameraSensor** (p. 320), **gazebo::sensors::RFIDSensor** (p. 729), and **gazebo::sensors::RFIDTag** (p. 731).

10.171.3 Member Data Documentation

10.171.3.1 bool gazebo::sensors::Sensor::active [protected]

True if active.

10.171.3.2 std::vector<event::ConnectionPtr> gazebo::sensors::Sensor::connections [protected]

10.171.3.3 common::Time gazebo::sensors::Sensor::lastUpdateTime [protected]

10.171.3.4 transport::NodePtr gazebo::sensors::Sensor::node [protected]

10.171.3.5 std::string gazebo::sensors::Sensor::parentName [protected]

10.171.3.6 std::vector<SensorPluginPtr> gazebo::sensors::Sensor::plugins [protected]

10.171.3.7 math::Pose gazebo::sensors::Sensor::pose [protected]

10.171.3.8 transport::SubscriberPtr gazebo::sensors::Sensor::poseSub [protected]

10.171.3.9 sdf::ElementPtr gazebo::sensors::Sensor::sdf [protected]

10.171.3.10 common::Time gazebo::sensors::Sensor::updatePeriod [protected]

10.171.3.11 gazebo::physics::WorldPtr gazebo::sensors::Sensor::world [protected]

The documentation for this class was generated from the following file:

- **Sensor.hh**

10.172 SensorFactor Class Reference

The sensor factory; the class is just for namespacing purposes.

`#include <sensors/sensors.hh>`

10.172.1 Detailed Description

The sensor factory; the class is just for namespacing purposes.

The documentation for this class was generated from the following file:

- **SensorFactory.hh**

10.173 gazebo::sensors::SensorFactory Class Reference

`#include <SensorFactory.hh>`

Static Public Member Functions

- static void **GetSensorTypes** (std::vector< std::string > &_types)

  Get all the sensor types.
- static **SensorPtr NewSensor** (const std::string &_classname)

  Create a new instance of a sensor.
- static void **RegisterAll** ()

  Register all known sensors.
- static void **RegisterSensor** (const std::string &_classname, **SensorFactoryFn** _factoryfn)

  Register a sensor class (called by sensor registration function).

10.173.1 Member Function Documentation

10.173.1.1 static void gazebo::sensors::SensorFactory::GetSensorTypes (std::vector< std::string > & *_types*) [static]

Get all the sensor types.

Parameters

_types	Vector of strings of the sensor types, populated by function

10.173.1.2 static SensorPtr gazebo::sensors::SensorFactory::NewSensor (const std::string & *_classname*) [static]

Create a new instance of a sensor.

Used by the world when reading the world file.

Parameters

in	*Name*	of sensor class

Returns

Pointer to **Sensor** (p. 765)

10.173.1.3 static void gazebo::sensors::SensorFactory::RegisterAll ( ) [static]

Register all known sensors.

- **sensors::CameraSensor** (p. 255)
- **sensors::DepthCameraSensor** (p. 317)
- **sensors::GpuRaySensor** (p. 378)
- **sensors::RaySensor** (p. 711)
- **sensors::ContactSensor** (p. 300)
- **sensors::RFIDSensor** (p. 727)
- **sensors::RFIDTag** (p. 729)

10.173.1.4 static void gazebo::sensors::SensorFactory::RegisterSensor (const std::string & *_classname,* SensorFactoryFn *_factoryfn*) [static]

Register a sensor class (called by sensor registration function).

Parameters

in	*_classname*	Name of class of sensor to register
	_factoryfn	Function handle for registration Nate check

The documentation for this class was generated from the following file:

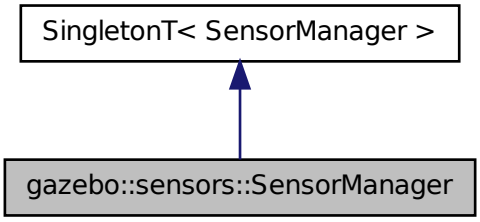
- **SensorFactory.hh**

10.174 gazebo::sensors::SensorManager Class Reference

Class to manage and update all sensors.

`#include <sensors/sensors.hh>`

Inheritance diagram for gazebo::sensors::SensorManager:



Public Member Functions

- std::string **CreateSensor** (**sdf::ElementPtr** _elem, const std::string &_worldName, const std::string &_parentName)

  Add a sensor from an SDF element.
- void **Fini** ()

  Finalize all the sensors.
- **SensorPtr GetSensor** (const std::string &_name)

  Get a sensor.
- void **GetSensorTypes** (std::vector< std::string > &_types) const

  Get all the sensor types.
- void **Init** ()

  Init all the sensors.
- void **RemoveSensor** (const std::string &_name)

  Remove a sensor.
- void **RemoveSensors** ()

  Remove all sensors.
- void **Run** ()

  Run the sensor manager update in a new thread.
- bool **SensorsInitialized** ()

  True if SensorManager::initSensors queue is empty i.e.
- void **Stop** ()

Stop the run thread.

- void **Update** (bool _force=false)

  Update all the sensors.

Additional Inherited Members

10.174.1 Detailed Description

Class to manage and update all sensors.

10.174.2 Member Function Documentation

10.174.2.1 std::string gazebo::sensors::SensorManager::CreateSensor (sdf::ElementPtr *_elem,* const std::string & *_worldName,* const std::string & *_parentName*)

Add a sensor from an SDF element.

This function will also Load and Init the sensor.

Parameters

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

10.174.2.2 void gazebo::sensors::SensorManager::Fini ( )

Finalize all the sensors.

10.174.2.3 SensorPtr gazebo::sensors::SensorManager::GetSensor (const std::string & *_name*)

Get a sensor.

Parameters

in	*_name*	The name of a sensor to find.

Returns

A pointer to the sensor. NULL if not found.

10.174.2.4 void gazebo::sensors::SensorManager::GetSensorTypes (std::vector< std::string > & *_types*) const

Get all the sensor types.

Parameters

out	*All*	the sensor types.

10.174.2.5 void gazebo::sensors::SensorManager::Init ( )

Init all the sensors.

10.174.2.6 void gazebo::sensors::SensorManager::RemoveSensor (const std::string & *_name*)

Remove a sensor.

Parameters

in	*_name*	The name of the sensor to remove.

10.174.2.7 void gazebo::sensors::SensorManager::RemoveSensors ( )

Remove all sensors.

10.174.2.8 void gazebo::sensors::SensorManager::Run ( )

Run the sensor manager update in a new thread.

10.174.2.9 bool gazebo::sensors::SensorManager::SensorsInitialized ( )

True if SensorManager::initSensors queue is empty i.e.

all sensors managed by **SensorManager** (p. 774) have been initialized

10.174.2.10 void gazebo::sensors::SensorManager::Stop ( )

Stop the run thread.

10.174.2.11 void gazebo::sensors::SensorManager::Update (bool *_force* = `false`)

Update all the sensors.

Checks to see if any sensor need to be initialized first, then updates all sensors once.

Parameters

in	*_force*	True force update, false if not

The documentation for this class was generated from the following file:

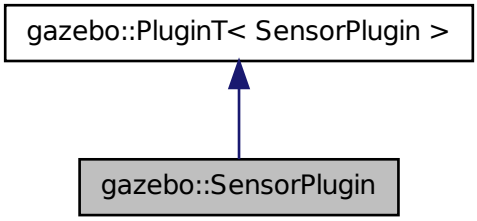
- **SensorManager.hh**

10.175 gazebo::SensorPlugin Class Reference

A plugin with access to physics::Sensor.

`#include <Plugin.hh>`

Inheritance diagram for gazebo::SensorPlugin:



Public Member Functions

- **SensorPlugin** ()

  Constructor.
- virtual ∼**SensorPlugin** ()

  Destructor.
- virtual void **Init** ()

  Override this method for custom plugin initialization behavior.
- virtual void **Load** (**sensors::SensorPtr** _sensor, **sdf::ElementPtr** _sdf)=0

  Load function.
- virtual void **Reset** ()

  Override this method for custom plugin reset behavior.

Additional Inherited Members

10.175.1 Detailed Description

A plugin with access to physics::Sensor.

See `reference`.

10.175.2 Constructor & Destructor Documentation

10.175.2.1 gazebo::SensorPlugin::SensorPlugin ( ) [inline]

Constructor.

References gazebo::SENSOR_PLUGIN, and gazebo::PluginT< SensorPlugin >::type.

10.175.2.2 virtual gazebo::SensorPlugin::~SensorPlugin ( ) [inline], [virtual]

Destructor.

10.175.3 Member Function Documentation

10.175.3.1 virtual void gazebo::SensorPlugin::Init ( ) [inline], [virtual]

Override this method for custom plugin initialization behavior.

10.175.3.2 virtual void gazebo::SensorPlugin::Load (sensors::SensorPtr *_sensor*, sdf::ElementPtr *_sdf*) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	*_sensor*	Pointer the Sensor.
in	*_sdf*	Pointer the the SDF element of the plugin.

10.175.3.3 virtual void gazebo::SensorPlugin::Reset ( ) [inline], [virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

- **common/Plugin.hh**

10.176 gazebo::Server Class Reference

`#include <Server.hh>`

Public Member Functions

- **Server** ()
- virtual ~**Server** ()
- void **Fini** ()
- bool **GetInitialized** () const
- void **Init** ()
- bool **Load** (const std::string &_filename="worlds/empty.world")
- bool **ParseArgs** (int argc, char **argv)
- void **PrintUsage** ()
- void **Run** ()
- void **SetParams** (const **common::StrStr_M** &params)
- void **Stop** ()

Public Attributes

- int **systemPluginsArgc**
- char ** **systemPluginsArgv**

10.176.1 Constructor & Destructor Documentation

10.176.1.1 gazebo::Server::Server ( )

10.176.1.2 virtual gazebo::Server::~Server ( ) [virtual]

10.176.2 Member Function Documentation

10.176.2.1 void gazebo::Server::Fini ( )

10.176.2.2 bool gazebo::Server::GetInitialized ( ) const

10.176.2.3 void gazebo::Server::Init ( )

10.176.2.4 bool gazebo::Server::Load (const std::string & *_filename* = "worlds/empty.world")

10.176.2.5 bool gazebo::Server::ParseArgs (int *argc*, char ** *argv*)

10.176.2.6 void gazebo::Server::PrintUsage ( )

10.176.2.7 void gazebo::Server::Run ( )

10.176.2.8 void gazebo::Server::SetParams (const common::StrStr_M & *params*)

10.176.2.9 void gazebo::Server::Stop ( )

10.176.3 Member Data Documentation

10.176.3.1 int gazebo::Server::systemPluginsArgc

10.176.3.2 char** gazebo::Server::systemPluginsArgv

The documentation for this class was generated from the following file:

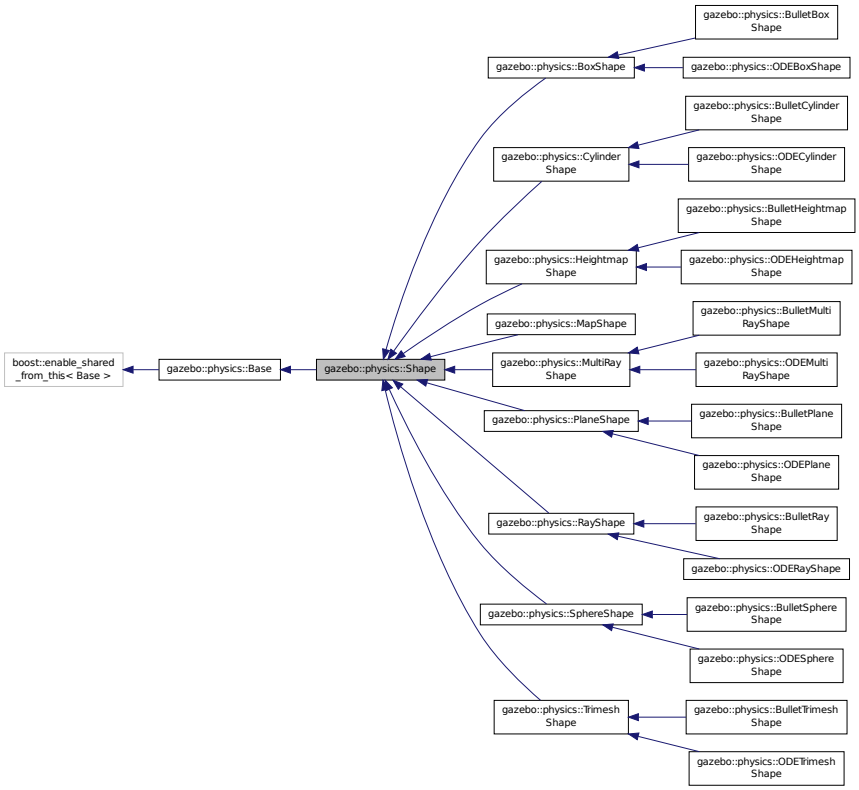
- **Server.hh**

10.177 gazebo::physics::Shape Class Reference

Base (p. 145) class for all shapes.

#include <Shape.hh>

Inheritance diagram for gazebo::physics::Shape:



Public Member Functions

- **Shape** (**CollisionPtr** p)

  Constructor.
- virtual ~**Shape** ()

  Destructor.
- virtual void **FillMsg** (msgs::Geometry &_msg)=0
- virtual void **FillShapeMsg** (msgs::Geometry &_msg) **GAZEBO_DEPRECATED**
- virtual void **GetInertial** (double _mass, **InertialPtr** _inertial) const

  Get inertial for a shape.
- virtual double **GetMass** (double _density) const

  Get the mass of a shape.
- virtual void **Init** ()=0

  Initialize the shape.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)=0

Public Attributes

- **CollisionPtr collisionParent**

Additional Inherited Members

10.177.1 Detailed Description

Base (p. 145) class for all shapes.

10.177.2 Constructor & Destructor Documentation

10.177.2.1 gazebo::physics::Shape::Shape (CollisionPtr *p*)

Constructor.

10.177.2.2 virtual gazebo::physics::Shape::~Shape ( ) [virtual]

Destructor.

10.177.3 Member Function Documentation

10.177.3.1 virtual void gazebo::physics::Shape::FillMsg (msgs::Geometry & *_msg*) [pure virtual]

Implemented in **gazebo::physics::MultiRayShape** (p. 552), **gazebo::physics::RayShape** (p. 720), **gazebo::physics::HeightmapShape** (p. 401), **gazebo::physics::CylinderShape** (p. 309), **gazebo::physics::MapShape** (p. 485), **gazebo::physics::PlaneShape** (p. 673), **gazebo::physics::TrimeshShape** (p. 867), **gazebo::physics::BoxShape** (p. 164), and **gazebo::physics::SphereShape** (p. 807).

10.177.3.2 virtual void gazebo::physics::Shape::FillShapeMsg (msgs::Geometry & *_msg*) [virtual]

Reimplemented in **gazebo::physics::BoxShape** (p. 164).

10.177.3.3 virtual void gazebo::physics::Shape::GetInertial (double *_mass,* InertialPtr *_inertial*) const [virtual]

Get inertial for a shape.

Reimplemented in **gazebo::physics::CylinderShape** (p. 309), **gazebo::physics::BoxShape** (p. 164), and **gazebo::physics::SphereShape** (p. 807).

10.177.3.4 virtual double gazebo::physics::Shape::GetMass (double *_density*) const [inline], [virtual]

Get the mass of a shape.

Reimplemented in **gazebo::physics::CylinderShape** (p. 310), **gazebo::physics::BoxShape** (p. 165), **gazebo::physics::TrimeshShape** (p. 868), and **gazebo::physics::SphereShape** (p. 808).

10.177.3.5 virtual void gazebo::physics::Shape::Init ( ) [pure virtual]

Initialize the shape.

Reimplemented from **gazebo::physics::Base** (p. 153).

Implemented in **gazebo::physics::RayShape** (p. 721), **gazebo::physics::MapShape** (p. 486), **gazebo::physics::-HeightmapShape** (p. 403), **gazebo::physics::TrimeshShape** (p. 868), **gazebo::physics::BulletHeightmapShape** (p. 178), **gazebo::physics::MultiRayShape** (p. 555), **gazebo::physics::ODETrimeshShape** (p. 634), **gazebo-::physics::ODEHeightmapShape** (p. 586), **gazebo::physics::PlaneShape** (p. 673), **gazebo::physics::Bullet-TrimeshShape** (p. 224), **gazebo::physics::BoxShape** (p. 165), **gazebo::physics::CylinderShape** (p. 310), and **gazebo::physics::SphereShape** (p. 808).

10.177.3.6 virtual void gazebo::physics::Shape::ProcessMsg (const msgs::Geometry & *_msg*) [pure virtual]

Implemented in **gazebo::physics::MultiRayShape** (p. 555), **gazebo::physics::RayShape** (p. 721), **gazebo-::physics::HeightmapShape** (p. 403), **gazebo::physics::CylinderShape** (p. 310), **gazebo::physics::MapShape** (p. 487), **gazebo::physics::PlaneShape** (p. 673), **gazebo::physics::BoxShape** (p. 165), **gazebo::physics::Trimesh-Shape** (p. 868), and **gazebo::physics::SphereShape** (p. 808).

10.177.4 Member Data Documentation

10.177.4.1 CollisionPtr gazebo::physics::Shape::collisionParent

Referenced by gazebo::physics::ODEPlaneShape::CreatePlane(), gazebo::physics::BulletPlaneShape::CreatePlane(), gazebo::physics::ODEPlaneShape::SetAltitude(), gazebo::physics::ODESphereShape::SetRadius(), gazebo::physics::-BulletSphereShape::SetRadius(), gazebo::physics::ODECylinderShape::SetSize(), gazebo::physics::ODEBoxShape::-SetSize(), gazebo::physics::BulletBoxShape::SetSize(), and gazebo::physics::BulletCylinderShape::SetSize().

The documentation for this class was generated from the following file:

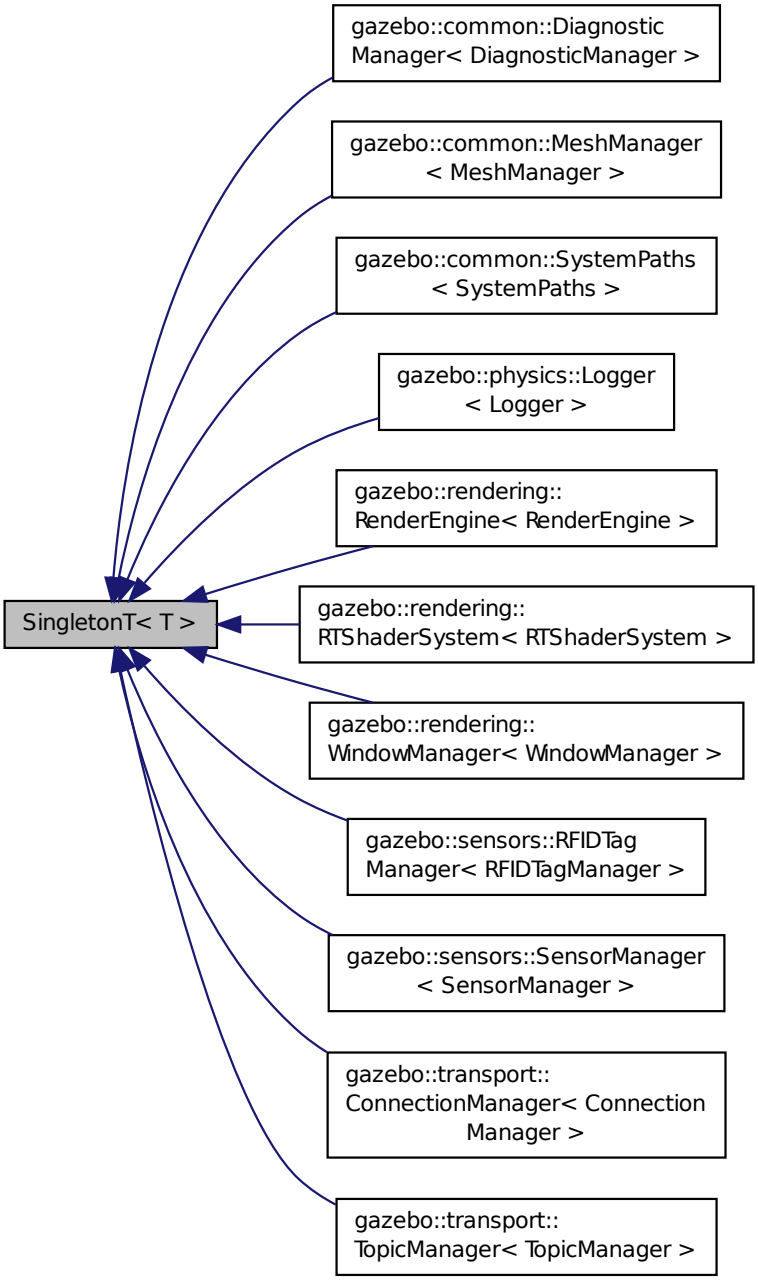
- **Shape.hh**

10.178 SingletonT< T > Class Template Reference

Singleton template class.

`#include <SingletonT.hh>`

Inheritance diagram for SingletonT< T >:



Static Public Member Functions

- static T * **Instance** ()

  Get an instance of the singleton.

Protected Member Functions

- **SingletonT** ()

  Constructor.
- virtual ~**SingletonT** ()

  Destructor.

10.178.1 Detailed Description

template<class T>class SingletonT< T >

Singleton template class.

10.178.2 Constructor & Destructor Documentation

10.178.2.1 template<class T> SingletonT< T >::SingletonT ( ) [inline], [protected]

Constructor.

10.178.2.2 template<class T> virtual SingletonT< T >::~SingletonT ( ) [inline], [protected], [virtual]

Destructor.

10.178.3 Member Function Documentation

10.178.3.1 template<class T> static T* SingletonT< T >::Instance ( ) [inline], [static]

Get an instance of the singleton.

Referenced by gazebo::transport::Node::Advertise(), gazebo::transport::TopicManager::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), and gazebo::transport::Node::Subscribe().

The documentation for this class was generated from the following file:

- **SingletonT.hh**

10.179 gazebo::common::Skeleton Class Reference

A skeleton.

`#include <Skeleton.hh>`

Public Member Functions

- **Skeleton** ()

  Constructor.
- **Skeleton** (**SkeletonNode** *_root)

  Constructor.

- virtual ~**Skeleton** ()

  Destructor.
- void **AddAnimation** (**SkeletonAnimation** *_anim)

  Add an animation.
- void **AddVertNodeWeight** (unsigned int _vertex, std::string _node, double _weight)

  Add a new weight to a node (bone)
- **SkeletonAnimation** * **GetAnimation** (const unsigned int _i)

  Find animation.
- **math::Matrix4 GetBindShapeTransform** ()

  Return bind pose skeletal transform.
- **SkeletonNode** * **GetNodeByHandle** (unsigned int _handle)

  Find or create node with handle.
- **SkeletonNode** * **GetNodeById** (std::string _id)

  Find node by index.
- **SkeletonNode** * **GetNodeByName** (std::string _name)

  Find a node.
- **NodeMap GetNodes** ()

  Get a copy or the node dictionary.
- unsigned int **GetNumAnimations** ()

  Returns the number of animations.
- unsigned int **GetNumJoints** ()

  Returns the number of joints.
- unsigned int **GetNumNodes** ()

  Returns the node count.
- unsigned int **GetNumVertNodeWeights** (unsigned int _vertex)

  Returns the number of bone weights for a vertex.
- **SkeletonNode** * **GetRootNode** ()

  Return the root.
- std::pair< std::string, double > **GetVertNodeWeight** (unsigned int _v, unsigned int _i)

  Weight of a bone for a vertex.
- void **PrintTransforms** ()

  Outputs the transforms to std::err stream.
- void **Scale** (double _scale)

  Scale all nodes, transforms and animation data.
- void **SetBindShapeTransform** (**math::Matrix4** _trans)

  Set the bind pose skeletal transform.
- void **SetNumVertAttached** (unsigned int _vertices)

  Resizes the raw node weight array.
- void **SetRootNode** (**SkeletonNode** *_node)

  Change the root node.

Protected Member Functions

- void **BuildNodeMap** ()

  Initializes the hande numbers for each node in the map using breath first traversal.

Protected Attributes

- std::vector< **SkeletonAnimation** * > **anims**

  the array of animations
- **math::Matrix4 bindShapeTransform**

  the bind pose skeletal transform
- **NodeMap nodes**

  The dictionary of nodes, indexed by name.
- **RawNodeWeights rawNW**

  the node weight table
- **SkeletonNode** * **root**

  the root node

10.179.1 Detailed Description

A skeleton.

10.179.2 Constructor & Destructor Documentation

10.179.2.1 gazebo::common::Skeleton::Skeleton ( )

Constructor.

10.179.2.2 gazebo::common::Skeleton::Skeleton (**SkeletonNode** * *_root*)

Constructor.

Parameters

in	*_root*	node

10.179.2.3 virtual gazebo::common::Skeleton::~Skeleton ( ) [virtual]

Destructor.

10.179.3 Member Function Documentation

10.179.3.1 void gazebo::common::Skeleton::AddAnimation (**SkeletonAnimation** * *_anim*)

Add an animation.

The skeleton does not take ownership of the animation

Parameters

_anim	the animation

10.179.3.2 void gazebo::common::Skeleton::AddVertNodeWeight (unsigned int _vertex, std::string _node, double _weight)

Add a new weight to a node (bone)

Parameters

in	*_vertex*	index of the vertex
in	*_node*	name of the bone
in	*_weight*	the new weight (range 0 to 1)

10.179.3.3 void gazebo::common::Skeleton::BuildNodeMap ( ) [protected]

Initializes the hande numbers for each node in the map using breath first traversal.

10.179.3.4 SkeletonAnimation* gazebo::common::Skeleton::GetAnimation (const unsigned int _i)

Find animation.

Parameters

_i	the animation index

Returns

the animation, or NULL if _i is out of bounds

10.179.3.5 math::Matrix4 gazebo::common::Skeleton::GetBindShapeTransform ( )

Return bind pose skeletal transform.

Returns

a matrix

10.179.3.6 SkeletonNode* gazebo::common::Skeleton::GetNodeByHandle (unsigned int _handle)

Find or create node with handle.

Parameters

in	*_handle*	

Returns

the node. A new node is created if it didn't exist

10.179.3.7 SkeletonNode* gazebo::common::Skeleton::GetNodeById (std::string _id)

Find node by index.

Parameters

in	*_id*	the index

Returns

the node, or NULL if not found

10.179.3.8 SkeletonNode* gazebo::common::Skeleton::GetNodeByName (std::string *_name*)

Find a node.

Parameters

in	*_name*	the name of the node to look for

Returns

the node, or NULL if not found

10.179.3.9 NodeMap gazebo::common::Skeleton::GetNodes ( )

Get a copy or the node dictionary.

10.179.3.10 unsigned int gazebo::common::Skeleton::GetNumAnimations ( )

Returns the number of animations.

Returns

the count

10.179.3.11 unsigned int gazebo::common::Skeleton::GetNumJoints ( )

Returns the number of joints.

Returns

the count

10.179.3.12 unsigned int gazebo::common::Skeleton::GetNumNodes ( )

Returns the node count.

Returns

the count

10.179.3.13 unsigned int gazebo::common::Skeleton::GetNumVertNodeWeights (unsigned int _vertex)

Returns the number of bone weights for a vertex.

Parameters

in	*the*	index of the vertex

Returns

the count

10.179.3.14 SkeletonNode* gazebo::common::Skeleton::GetRootNode ( )

Return the root.

Returns

the root

10.179.3.15 std::pair<std::string, double> gazebo::common::Skeleton::GetVertNodeWeight (unsigned int _v, unsigned int _i)

Weight of a bone for a vertex.

Parameters

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

10.179.3.16 void gazebo::common::Skeleton::PrintTransforms ( )

Outputs the transforms to std::err stream.

10.179.3.17 void gazebo::common::Skeleton::Scale (double _scale)

Scale all nodes, transforms and animation data.

Parameters

in	*the*	scaling factor

10.179.3.18 void gazebo::common::Skeleton::SetBindShapeTransform (math::Matrix4 _trans)

Set the bind pose skeletal transform.

Parameters

in	*_trans*	the transform

10.179.3.19 void gazebo::common::Skeleton::SetNumVertAttached (unsigned int *_vertices*)

Resizes the raw node weight array.

Parameters

in	*_vertices*	the new size

10.179.3.20 void gazebo::common::Skeleton::SetRootNode (SkeletonNode * *_node*)

Change the root node.

Parameters

in	*_node*	the new node

10.179.4 Member Data Documentation

10.179.4.1 std::vector<SkeletonAnimation*> gazebo::common::Skeleton::anims [protected]

the array of animations

10.179.4.2 math::Matrix4 gazebo::common::Skeleton::bindShapeTransform [protected]

the bind pose skeletal transform

10.179.4.3 NodeMap gazebo::common::Skeleton::nodes [protected]

The dictionary of nodes, indexed by name.

10.179.4.4 RawNodeWeights gazebo::common::Skeleton::rawNW [protected]

the node weight table

10.179.4.5 SkeletonNode* gazebo::common::Skeleton::root [protected]

the root node

The documentation for this class was generated from the following file:

- **Skeleton.hh**

10.180 gazebo::common::SkeletonAnimation Class Reference

Skeleton (p. 784) animation.

`#include <SkeletonAnimation.hh>`

Public Member Functions

- **SkeletonAnimation** (const std::string &_name)

  The Constructor.
- ∼**SkeletonAnimation** ()

  The destructor.
- void **AddKeyFrame** (const std::string &_node, const double _time, const **math::Matrix4** _mat)

  Adds or replaces a named key frame at a specific time.
- void **AddKeyFrame** (const std::string &_node, const double _time, const **math::Pose** _pose)

  Adds or replaces a named key frame at a specific time.
- double **GetLength** () const

  Returns the duration of the animations.
- std::string **GetName** () const

  Returns the name.
- unsigned int **GetNodeCount** () const

  Returns the number of animation nodes.
- **math::Matrix4 GetNodePoseAt** (const std::string &_node, const double _time, const bool _loop=true)

  Returns the key frame transformation for a named animation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.
- std::map< std::string, **math::Matrix4** > **GetPoseAt** (const double _time, const bool _loop=true) const

  Returns a dictionary of transformations indexed by name at a specific time if a node does not exist at that specific time (with tolerance of 1e-6 sec), the transformation is interpolated.
- std::map< std::string, **math::Matrix4** > **GetPoseAtX** (const double _x, const std::string &_node, const bool _loop=true) const

  Returns a dictionary of transformations indexed by name where a named node transformation's translational value along the X axis is equal to _x.
- bool **HasNode** (const std::string &_node) const

  Looks for a node with a specific name in the animations.
- void **Scale** (const double _scale)

  Scales every animation in the animations list.
- void **SetName** (const std::string &_name)

  Changes the name.

Protected Attributes

- std::map< std::string, **NodeAnimation** ∗ > **animations**

  a dictionary of node animations
- double **length**

  the duration of the longest animation
- std::string **name**

  the node name

10.180.1 Detailed Description

Skeleton (p. 784) animation.

10.180.2 Constructor & Destructor Documentation

10.180.2.1 gazebo::common::SkeletonAnimation::SkeletonAnimation (const std::string & *_name*)

The Constructor.

Parameters

in	*_name*	the name of the animation

10.180.2.2 gazebo::common::SkeletonAnimation::~SkeletonAnimation ( )

The destructor.

Clears the list without destroying the animations

10.180.3 Member Function Documentation

10.180.3.1 void gazebo::common::SkeletonAnimation::AddKeyFrame (const std::string & *_node*, const double *_time*, const **math::Matrix4** *_mat*)

Adds or replaces a named key frame at a specific time.

Parameters

in	*_node*	the name of the new or existing node
in	*_time*	the time
in	*_mat*	the key frame transformation

10.180.3.2 void gazebo::common::SkeletonAnimation::AddKeyFrame (const std::string & *_node*, const double *_time*, const **math::Pose** *_pose*)

Adds or replaces a named key frame at a specific time.

Parameters

in	*_node*	the name of the new or existing node
in	*_time*	the time
in	*_pose*	the key frame transformation as a **math::Pose** (p. 677)

10.180.3.3 double gazebo::common::SkeletonAnimation::GetLength ( ) const

Returns the duration of the animations.

Returns

the duration in seconds

10.180.3.4 std::string gazebo::common::SkeletonAnimation::GetName ( ) const

Returns the name.

Returns

the name

10.180.3.5 unsigned int gazebo::common::SkeletonAnimation::GetNodeCount ( ) const

Returns the number of animation nodes.

Returns

the count

10.180.3.6 math::Matrix4 gazebo::common::SkeletonAnimation::GetNodePoseAt (const std::string & *_node,* const double *_time,* const bool *_loop* = `true`)

Returns the key frame transformation for a named animation at a specific time if a node does not exist at that time (with tolerance of 1e-6 sec), the transformation is interpolated.

Parameters

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

10.180.3.7 std::map<std::string, math::Matrix4> gazebo::common::SkeletonAnimation::GetPoseAt (const double *_time,* const bool *_loop* = `true`) const

Returns a dictionary of transformations indexed by name at a specific time if a node does not exist at that specific time (with tolerance of 1e-6 sec), the transformation is interpolated.

Parameters

in	*_time*	the time
in	*_loop*	when true, the time is divided by the duration (see GetLength)

Returns

the transformation for every node

10.180.3.8 std::map<std::string, math::Matrix4> gazebo::common::SkeletonAnimation::GetPoseAtX (const double _x, const std::string & _node, const bool _loop = true) const

Returns a dictionary of transformations indexed by name where a named node transformation's translational value along the X axis is equal to _x.

Parameters

in	*_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)

10.180.3.9 bool gazebo::common::SkeletonAnimation::HasNode (const std::string & _node) const

Looks for a node with a specific name in the animations.

Parameters

in	*_node*	the name of the node

Returns

true if the node exits

10.180.3.10 void gazebo::common::SkeletonAnimation::Scale (const double _scale)

Scales every animation in the animations list.

Parameters

in	*_scale*	the scaling factor

10.180.3.11 void gazebo::common::SkeletonAnimation::SetName (const std::string & _name)

Changes the name.

Parameters

in	*_name*	the new name

10.180.4 Member Data Documentation

10.180.4.1 std::map<std::string, NodeAnimation*> gazebo::common::SkeletonAnimation::animations [protected]

a dictionary of node animations

10.180.4.2 double gazebo::common::SkeletonAnimation::length [protected]

the duration of the longest animation

10.180.4.3 std::string gazebo::common::SkeletonAnimation::name [protected]

the node name

The documentation for this class was generated from the following file:

- **SkeletonAnimation.hh**

10.181 gazebo::common::SkeletonNode Class Reference

A skeleton node.

`#include <Skeleton.hh>`

Public Types

- enum **SkeletonNodeType** { **NODE**, **JOINT** }
  enumeration of node types

Public Member Functions

- **SkeletonNode** (**SkeletonNode** *_parent)
  Constructor.
- **SkeletonNode** (**SkeletonNode** *_parent, std::string _name, std::string _id, **SkeletonNodeType** _type=**JOINT**)
  Constructor.
- virtual ~**SkeletonNode** ()
  Destructor.
- void **AddChild** (**SkeletonNode** *_child)
  Add a new child.
- void **AddRawTransform** (**NodeTransform** _t)
  Add a raw transform.
- **SkeletonNode** * **GetChild** (unsigned int _index)
  Find a child by index.
- **SkeletonNode** * **GetChildById** (std::string _id)
  Get child by string id.
- **SkeletonNode** * **GetChildByName** (std::string _name)
  Get child by name.
- unsigned int **GetChildCount** ()
  Returns the children count.
- unsigned int **GetHandle** ()
  Get the handle index.
- std::string **GetId** ()
  Returns the index.

- **math::Matrix4 GetInverseBindTransform** ()

  Retrieve the inverse of the bind pose skeletal transform.
- **math::Matrix4 GetModelTransform** ()

  Retrieve the model transform.
- std::string **GetName** ()

  Returns the name.
- unsigned int **GetNumRawTrans** ()

  Return the raw transformations count.
- **SkeletonNode** ∗ **GetParent** ()

  Returns the parent node.
- **NodeTransform GetRawTransform** (unsigned int _i)

  Find a raw transformation.
- std::vector< **NodeTransform** > **GetRawTransforms** ()

  Retrieve the raw transformations.
- **math::Matrix4 GetTransform** ()

  Get transform relative to parent.
- std::vector< **NodeTransform** > **GetTransforms** ()

  Returns a copy of the array of transformations.
- bool **IsJoint** ()

  Is a joint query.
- bool **IsRootNode** ()

  Queries wether a node has no parent parent.
- void **Reset** (bool _resetChildren)

  Reset the transformation to the initial transformation.
- void **SetHandle** (unsigned int _h)

  Assign a handle number.
- void **SetId** (std::string _id)

  Change the id string.
- void **SetInitialTransform** (**math::Matrix4** _tras)

  Sets the initial transformation.
- void **SetInverseBindTransform** (**math::Matrix4** _invBM)

  Assign the inverse of the bind pose skeletal transform.
- void **SetModelTransform** (**math::Matrix4** _trans, bool _updateChildren=true)

  Set the model transformation.
- void **SetName** (std::string _name)

  Change the name.
- void **SetParent** (**SkeletonNode** ∗_parent)

  Set the parent node.
- void **SetTransform** (**math::Matrix4** _trans, bool _updateChildren=true)

  Set a transformation.
- void **SetType** (**SkeletonNodeType** _type)

  Change the skeleton node type.
- void **UpdateChildrenTransforms** ()

  Apply model transformations in order for each node in the tree.

Protected Attributes

- std::vector< **SkeletonNode** * > **children**

  the children nodes
- unsigned int **handle**

  handle index number
- std::string **id**

  a string identifier
- **math::Matrix4 initialTransform**

  the initial transformation
- **math::Matrix4 invBindTransform**

  the inverse of the bind pose skeletal transform
- **math::Matrix4 modelTransform**

  the model transformation
- std::string **name**

  the name of the skeletal node
- **SkeletonNode** * **parent**

  the parent node
- std::vector< **NodeTransform** > **rawTransforms**

  the raw transformation
- **math::Matrix4 transform**

  the transform
- **SkeletonNodeType type**

  the type fo node

10.181.1 Detailed Description

A skeleton node.

10.181.2 Member Enumeration Documentation

10.181.2.1 enum gazebo::common::SkeletonNode::SkeletonNodeType

enumeration of node types

Enumerator:

NODE

JOINT

10.181.3 Constructor & Destructor Documentation

10.181.3.1 gazebo::common::SkeletonNode::SkeletonNode (SkeletonNode * *_parent*)

Constructor.

10.181.3.2 gazebo::common::SkeletonNode::SkeletonNode (SkeletonNode * _parent, std::string _name, std::string _id, SkeletonNodeType _type = JOINT)

Constructor.

Parameters

in	*_parent*	the parent node
in	*__name*	name of node

10.181.3.3 virtual gazebo::common::SkeletonNode::~SkeletonNode ( ) [virtual]

Destructor.

10.181.4 Member Function Documentation

10.181.4.1 void gazebo::common::SkeletonNode::AddChild (SkeletonNode * _child)

Add a new child.

Parameters

in	*_child*	a child

10.181.4.2 void gazebo::common::SkeletonNode::AddRawTransform (NodeTransform _t)

Add a raw transform.

Parameters

in	*_t*	the transform

10.181.4.3 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

10.181.4.4 SkeletonNode* gazebo::common::SkeletonNode::GetChildById (std::string _id)

Get child by string id.

Parameters

in	*_id*	the string id

Returns

the child skeleton or NULL if not found

10.181.4.5 SkeletonNode* gazebo::common::SkeletonNode::GetChildByName (std::string *_name*)

Get child by name.

Parameters

in	*_name*	the name of the child skeleton

Returns

the skeleton, or NULL if not found

10.181.4.6 unsigned int gazebo::common::SkeletonNode::GetChildCount ( )

Returns the children count.

Returns

the count

10.181.4.7 unsigned int gazebo::common::SkeletonNode::GetHandle ( )

Get the handle index.

Returns

the handle index

10.181.4.8 std::string gazebo::common::SkeletonNode::GetId ( )

Returns the index.

Returns

the id string

10.181.4.9 math::Matrix4 gazebo::common::SkeletonNode::GetInverseBindTransform ( )

Retrieve the inverse of the bind pose skeletal transform.

Returns

the transform

10.181.4.10 math::Matrix4 gazebo::common::SkeletonNode::GetModelTransform ( )

Retrieve the model transform.

Returns

the transform

10.181.4.11 std::string gazebo::common::SkeletonNode::GetName ( )

Returns the name.

Returns

the name

10.181.4.12 unsigned int gazebo::common::SkeletonNode::GetNumRawTrans ( )

Return the raw transformations count.

Returns

the count

10.181.4.13 SkeletonNode* gazebo::common::SkeletonNode::GetParent ( )

Returns the parent node.

Returns

the parent

10.181.4.14 NodeTransform gazebo::common::SkeletonNode::GetRawTransform (unsigned int _i)

Find a raw transformation.

Parameters

in	_i	the index of the transformation

Returns

the node transform. NO BOUNDS CHECKING PERFORMED

10.181.4.15 std::vector<NodeTransform> gazebo::common::SkeletonNode::GetRawTransforms ( )

Retrieve the raw transformations.

Returns

an array of transformations

10.181.4.16 math::Matrix4 gazebo::common::SkeletonNode::GetTransform ( )

Get transform relative to parent.

10.181.4.17 std::vector<NodeTransform> gazebo::common::SkeletonNode::GetTransforms ( )

Returns a copy of the array of transformations.

Returns

the array of transform (These are the same as the raw trans)

10.181.4.18 bool gazebo::common::SkeletonNode::IsJoint ( )

Is a joint query.

Returns

true if the skeleton type is a joint, false otherwise

10.181.4.19 bool gazebo::common::SkeletonNode::IsRootNode ( )

Queries wether a node has no parent parent.

Returns

true if the node has no parent, fasle otherwise

10.181.4.20 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

10.181.4.21 void gazebo::common::SkeletonNode::SetHandle (unsigned int *_h*)

Assign a handle number.

Parameters

in	*_h*	the handle

10.181.4.22 void gazebo::common::SkeletonNode::SetId (std::string *_id*)

Change the id string.

Parameters

in	*_id*	the new id string

10.181.4.23 void gazebo::common::SkeletonNode::SetInitialTransform (math::Matrix4 *_tras*)

Sets the initial transformation.

Parameters

in	*_tras*	the transfromation matrix

10.181.4.24 void gazebo::common::SkeletonNode::SetInverseBindTransform (math::Matrix4 *_invBM*)

Assign the inverse of the bind pose skeletal transform.

Parameters

in	*_invBM*	the transform

10.181.4.25 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

10.181.4.26 void gazebo::common::SkeletonNode::SetName (std::string *_name*)

Change the name.

Parameters

in	*_name*	the new name

10.181.4.27 void gazebo::common::SkeletonNode::SetParent (SkeletonNode * *_parent*)

Set the parent node.

Parameters

in	*_parent*	the new parent

10.181.4.28 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

10.181.4.29 void gazebo::common::SkeletonNode::SetType (SkeletonNodeType _type)

Change the skeleton node type.

Parameters

in	*_type*	the new type

10.181.4.30 void gazebo::common::SkeletonNode::UpdateChildrenTransforms ( )

Apply model transformations in order for each node in the tree.

10.181.5 Member Data Documentation

10.181.5.1 std::vector<SkeletonNode*> gazebo::common::SkeletonNode::children [protected]

the children nodes

10.181.5.2 unsigned int gazebo::common::SkeletonNode::handle [protected]

handle index number

10.181.5.3 std::string gazebo::common::SkeletonNode::id [protected]

a string identifier

10.181.5.4 math::Matrix4 gazebo::common::SkeletonNode::initialTransform [protected]

the initial transformation

10.181.5.5 math::Matrix4 gazebo::common::SkeletonNode::invBindTransform [protected]

the inverse of the bind pose skeletal transform

10.181.5.6 math::Matrix4 gazebo::common::SkeletonNode::modelTransform [protected]

the model transformation

10.181.5.7 std::string gazebo::common::SkeletonNode::name [protected]

the name of the skeletal node

10.181.5.8 SkeletonNode* gazebo::common::SkeletonNode::parent [protected]

the parent node

10.181.5.9 std::vector<NodeTransform> gazebo::common::SkeletonNode::rawTransforms [protected]

the raw transformation

10.181.5.10 math::Matrix4 gazebo::common::SkeletonNode::transform [protected]

the transform

10.181.5.11 SkeletonNodeType gazebo::common::SkeletonNode::type [protected]

the type fo node

The documentation for this class was generated from the following file:

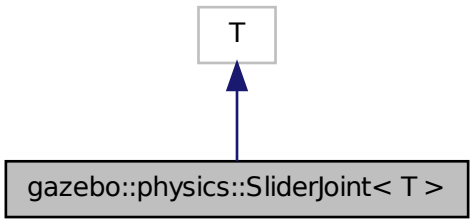
- **Skeleton.hh**

10.182 gazebo::physics::SliderJoint< T > Class Template Reference

A slider joint.

`#include <SliderJoint.hh>`

Inheritance diagram for gazebo::physics::SliderJoint< T >:



Public Member Functions

- **SliderJoint** (**BasePtr** _parent)

Constructor.

- virtual ~**SliderJoint** ()

  Destructor.

- virtual **math::Vector3 GetAnchor** (int) const

  Get the anchor.

- virtual void **SetAnchor** (int, const **math::Vector3** &_anchor)

  Set the anchor.

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load a **SliderJoint** (p. 804).*

Protected Attributes

- **math::Vector3 fakeAnchor**

10.182.1 Detailed Description

template<class T>class gazebo::physics::SliderJoint< T >

A slider joint.

10.182.2 Constructor & Destructor Documentation

10.182.2.1 template<class T> gazebo::physics::SliderJoint< T >::SliderJoint (BasePtr *_parent*) [inline]

Constructor.

10.182.2.2 template<class T> virtual gazebo::physics::SliderJoint< T >::~SliderJoint ( ) [inline], [virtual]

Destructor.

10.182.3 Member Function Documentation

10.182.3.1 template<class T> virtual math::Vector3 gazebo::physics::SliderJoint< T >::GetAnchor (int) const [inline], [virtual]

Get the anchor.

10.182.3.2 template<class T> virtual void gazebo::physics::SliderJoint< T >::Load (sdf::ElementPtr *_sdf*) [inline], [protected], [virtual]

Load a **SliderJoint** (p. 804).

Reimplemented in **gazebo::physics::BulletSliderJoint** (p. 220), and **gazebo::physics::ODESliderJoint** (p. 628).

10.182.3.3 template<class T> virtual void gazebo::physics::SliderJoint< T >::SetAnchor (int , const math::Vector3 & _anchor) [inline], [virtual]

Set the anchor.

10.182.4 Member Data Documentation

10.182.4.1 template<class T> math::Vector3 gazebo::physics::SliderJoint< T >::fakeAnchor [protected]

Referenced by gazebo::physics::SliderJoint< BulletJoint >::GetAnchor(), and gazebo::physics::SliderJoint< BulletJoint >::SetAnchor().

The documentation for this class was generated from the following file:

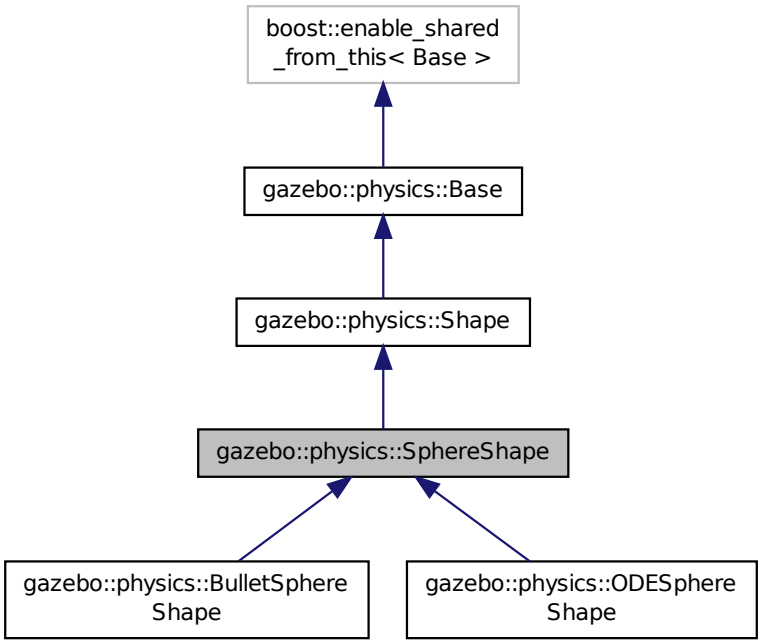
- **SliderJoint.hh**

10.183 gazebo::physics::SphereShape Class Reference

Sphere collision.

`#include <SphereShape.hh>`

Inheritance diagram for gazebo::physics::SphereShape:



Public Member Functions

- **SphereShape** (**CollisionPtr parent**)

  Constructor.
- virtual ~**SphereShape** ()

  Destructor.
- virtual void **FillMsg** (msgs::Geometry &_msg)
- virtual void **GetInertial** (double _mass, **InertialPtr** _inertial) const

  Get inertial for a shape.
- virtual double **GetMass** (double _density) const

  Get the mass of a shape.
- double **GetRadius** () const
- virtual void **Init** ()

  Initialize the sphere.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
- virtual void **SetRadius** (double _radius)

  Set the size.

Additional Inherited Members

10.183.1 Detailed Description

Sphere collision.

10.183.2 Constructor & Destructor Documentation

10.183.2.1 gazebo::physics::SphereShape::SphereShape (CollisionPtr *parent*)

Constructor.

10.183.2.2 virtual gazebo::physics::SphereShape::~SphereShape ( ) [virtual]

Destructor.

10.183.3 Member Function Documentation

10.183.3.1 virtual void gazebo::physics::SphereShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Implements **gazebo::physics::Shape** (p. 781).

10.183.3.2 virtual void gazebo::physics::SphereShape::GetInertial (double *_mass,* InertialPtr *_inertial*) const [virtual]

Get inertial for a shape.

Reimplemented from **gazebo::physics::Shape** (p. 781).

10.183.3.3 virtual double gazebo::physics::SphereShape::GetMass (double *_density*) const [virtual]

Get the mass of a shape.

Reimplemented from **gazebo::physics::Shape** (p. 781).

10.183.3.4 double gazebo::physics::SphereShape::GetRadius ( ) const

10.183.3.5 virtual void gazebo::physics::SphereShape::Init ( ) [virtual]

Initialize the sphere.

Implements **gazebo::physics::Shape** (p. 781).

10.183.3.6 virtual void gazebo::physics::SphereShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Implements **gazebo::physics::Shape** (p. 782).

10.183.3.7 virtual void gazebo::physics::SphereShape::SetRadius (double *_radius*) [virtual]

Set the size.

Reimplemented in **gazebo::physics::ODESphereShape** (p. 631).

Referenced by gazebo::physics::ODESphereShape::SetRadius(), and gazebo::physics::BulletSphereShape::SetRadius().

The documentation for this class was generated from the following file:

- **SphereShape.hh**

10.184 gazebo::math::Spline Class Reference

Splines.

`#include <Spline.hh>`

Public Member Functions

- **Spline** ()

  constructor
- ~**Spline** ()

  destructor
- void **AddPoint** (const **Vector3** &_pt)

  Adds a control point to the end of the spline.
- void **Clear** ()

  Clears all the points in the spline.
- **Vector3 GetPoint** (unsigned int _index) const

  Gets the detail of one of the control points of the spline.
- unsigned int **GetPointCount** () const

Gets the number of control points in the spline.

- **Vector3 GetTangent** (unsigned int _index) const

  Get the tangent value for a point.

- double **GetTension** () const

  Get the tension value.

- **Vector3 Interpolate** (double _t) const

  Returns an interpolated point based on a parametric value over the whole series.

- **Vector3 Interpolate** (unsigned int _fromIndex, double _t) const

  Interpolates a single segment of the spline given a parametric value.

- void **RecalcTangents** ()

  Recalculates the tangents associated with this spline.

- void **SetAutoCalculate** (bool _autoCalc)

  Tells the spline whether it should automatically calculate tangents on demand as points are added.

- void **SetTension** (double _t)

  Set the tension parameter.

- void **UpdatePoint** (unsigned int _index, const **Vector3** &_value)

  Updates a single point in the spline.

Protected Attributes

- bool **autoCalc**

  when true, the tangents are recalculated when the control point change

- **Matrix4 coeffs**

  Matrix of coefficients.

- std::vector< **Vector3** > **points**

  control points

- std::vector< **Vector3** > **tangents**

  tangents

- double **tension**

  Tension of 0 = Catmull-Rom spline, otherwise a Cardinal spline.

10.184.1 Detailed Description

Splines.

10.184.2 Constructor & Destructor Documentation

10.184.2.1 gazebo::math::Spline::Spline ( )

constructor

10.184.2.2 gazebo::math::Spline::~Spline ( )

destructor

10.184.3 Member Function Documentation

10.184.3.1 void gazebo::math::Spline::AddPoint (const Vector3 & _pt)

Adds a control point to the end of the spline.

Parameters

in	*_pt*	point to add

10.184.3.2 void gazebo::math::Spline::Clear ( )

Clears all the points in the spline.

10.184.3.3 Vector3 gazebo::math::Spline::GetPoint (unsigned int _index) const

Gets the detail of one of the control points of the spline.

Parameters

in	*_index*	the control point index

Returns

the control point, or [0,0,0] and a message on the error stream

10.184.3.4 unsigned int gazebo::math::Spline::GetPointCount ( ) const

Gets the number of control points in the spline.

Returns

the count

10.184.3.5 Vector3 gazebo::math::Spline::GetTangent (unsigned int _index) const

Get the tangent value for a point.

Parameters

in	*_index*	the control point index

10.184.3.6 double gazebo::math::Spline::GetTension ( ) const

Get the tension value.

Returns

The value of the tension, which is between 0.0 and 1.0

10.184.3.7 Vector3 gazebo::math::Spline::Interpolate (double _t) const

Returns an interpolated point based on a parametric value over the whole series.

Parameters

in	*_t*	parameter (range 0 to 1)

10.184.3.8 Vector3 gazebo::math::Spline::Interpolate (unsigned int *_fromIndex*, double *_t*) const

Interpolates a single segment of the spline given a parametric value.

Parameters

in	*_fromIndex*	The point index to treat as t = 0. fromIndex + 1 is deemed to be t = 1
in	*_t*	Parametric value

10.184.3.9 void gazebo::math::Spline::RecalcTangents ( )

Recalculates the tangents associated with this spline.

Remarks

If you tell the spline not to update on demand by calling setAutoCalculate(false) then you must call this after completing your updates to the spline points.

10.184.3.10 void gazebo::math::Spline::SetAutoCalculate (bool *_autoCalc*)

Tells the spline whether it should automatically calculate tangents on demand as points are added.

Remarks

The spline calculates tangents at each point automatically based on the input points. Normally it does this every time a point changes. However, if you have a lot of points to add in one go, you probably don't want to incur this overhead and would prefer to defer the calculation until you are finished setting all the points. You can do this by calling this method with a parameter of 'false'. Just remember to manually call the recalcTangents method when you are done.

Parameters

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.

10.184.3.11 void gazebo::math::Spline::SetTension (double _t)

Set the tension parameter.

A value of 0 = Catmull-Rom spline.

Parameters

in	*_t*	Tension value between 0.0 and 1.0

10.184.3.12 void gazebo::math::Spline::UpdatePoint (unsigned int _index, const Vector3 & _value)

Updates a single point in the spline.

Remarks

an error to the error stream is printed when the index is out of bounds

Parameters

in	*_index*	the control point index
in	*_value*	the new position

10.184.4 Member Data Documentation

10.184.4.1 bool gazebo::math::Spline::autoCalc [protected]

when true, the tangents are recalculated when the control point change

10.184.4.2 Matrix4 gazebo::math::Spline::coeffs [protected]

Matrix of coefficients.

10.184.4.3 std::vector<Vector3> gazebo::math::Spline::points [protected]

control points

10.184.4.4 std::vector<Vector3> gazebo::math::Spline::tangents [protected]

tangents

10.184.4.5 double gazebo::math::Spline::tension [protected]

Tension of 0 = Catmull-Rom spline, otherwise a Cardinal spline.

The documentation for this class was generated from the following file:

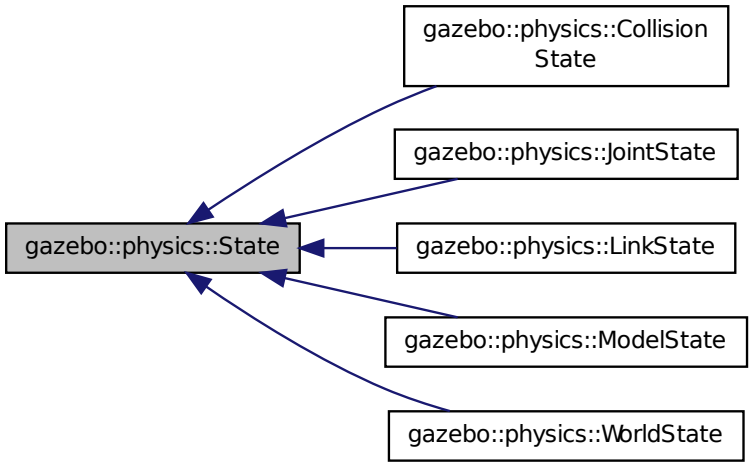
- **Spline.hh**

10.185 gazebo::physics::State Class Reference

State (p. 813) of an entity.

`#include <physics/State.hh>`

Inheritance diagram for gazebo::physics::State:



Public Member Functions

- **State** ()

  Default constructor.
- **State** (const std::string &_name, const **common::Time** &_realTime, const **common::Time** &_simTime)

  Constructor.
- virtual ~**State** ()

  Destructor.
- std::string **GetName** () const

  *Get the name associated with this **State** (p. 813).*
- **common::Time GetRealTime** () const

  Get the real time when this state was generated.
- **common::Time GetSimTime** () const

  Get the sim time when this state was generated.
- **common::Time GetWallTime** () const

  Get the wall time when this state was generated.
- virtual void **Load** (**sdf::ElementPtr** _elem)=0

  Load state from SDF element.

Protected Attributes

- std::string **name**

  *Name associated with this **State** (p. 813).*
- **common::Time realTime**
- **common::Time simTime**
- **common::Time wallTime**

  Times for the state data.

10.185.1 Detailed Description

State (p. 813) of an entity.

This is the base class for all **State** (p. 813) information.

10.185.2 Constructor & Destructor Documentation

10.185.2.1 gazebo::physics::State::State ( )

Default constructor.

10.185.2.2 gazebo::physics::State::State (const std::string & *_name,* const **common::Time** & *_realTime,* const **common::Time** & *_simTime*)

Constructor.

Construct a **State** (p. 813) object using some basic information.

Parameters

_name	Name associated with the **State** (p. 813) information. This is typically the name of an **Entity** (p. 338). _realTime Clock time since simulation started.
_simTime	Simulation time associated with this **State** (p. 813) info.

10.185.2.3 virtual gazebo::physics::State::~State ( ) [virtual]

Destructor.

10.185.3 Member Function Documentation

10.185.3.1 std::string gazebo::physics::State::GetName ( ) const

Get the name associated with this **State** (p. 813).

Returns

Name associated with this state information. Typically a name of an **Entity** (p. 338).

10.185.3.2 common::Time gazebo::physics::State::GetRealTime ( ) const

Get the real time when this state was generated.

Returns

Clock time since simulation was stated.

10.185.3.3 common::Time gazebo::physics::State::GetSimTime ( ) const

Get the sim time when this state was generated.

Returns

Simulation time when the data was recorded.

10.185.3.4 common::Time gazebo::physics::State::GetWallTime ( ) const

Get the wall time when this state was generated.

Returns

The absolute clock time when the **State** (p. 813) data was recorded.

10.185.3.5 virtual void gazebo::physics::State::Load (sdf::ElementPtr *_elem*) [pure virtual]

Load state from SDF element.

Populates the **State** (p. 813) information from data stored in an SDF::Element

Parameters

_elem	Pointer to the SDF::Element

Implemented in **gazebo::physics::LinkState** (p. 477), **gazebo::physics::ModelState** (p. 539), **gazebo::physics::WorldState** (p. 969), **gazebo::physics::CollisionState** (p. 274), and **gazebo::physics::JointState** (p. 443).

10.185.4 Member Data Documentation

10.185.4.1 std::string gazebo::physics::State::name [protected]

Name associated with this **State** (p. 813).

10.185.4.2 common::Time gazebo::physics::State::realTime [protected]

10.185.4.3 common::Time gazebo::physics::State::simTime [protected]

10.185.4.4 common::Time gazebo::physics::State::wallTime [protected]

Times for the state data.

The documentation for this class was generated from the following file:

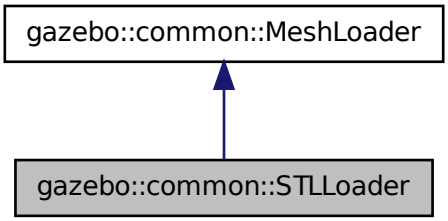
- **State.hh**

10.186 gazebo::common::STLLoader Class Reference

Class used to load STL mesh files.

`#include <STLLoader.hh>`

Inheritance diagram for gazebo::common::STLLoader:



Public Member Functions

- **STLLoader** ()

  Constructor.
- virtual ~**STLLoader** ()

  Destructor.
- virtual **Mesh** * **Load** (const std::string &filename)

  Creates a new mesh and loads the data from a file.

10.186.1 Detailed Description

Class used to load STL mesh files.

10.186.2 Constructor & Destructor Documentation

10.186.2.1 gazebo::common::STLLoader::STLLoader ( )

Constructor.

10.186.2.2 virtual gazebo::common::STLLoader::~STLLoader ( ) [virtual]

Destructor.

10.186.3 Member Function Documentation

10.186.3.1 virtual Mesh* gazebo::common::STLLoader::Load (const std::string & *filename*) [virtual]

Creates a new mesh and loads the data from a file.

Parameters

in	*filename*	the mesh file

Implements **gazebo::common::MeshLoader** (p. 515).

The documentation for this class was generated from the following file:

- **STLLoader.hh**

10.187 gazebo::common::SubMesh Class Reference

A child mesh.

`#include <Mesh.hh>`

Public Types

- enum **PrimitiveType** {
  POINTS, **LINES**, **LINESTRIPS**, **TRIANGLES**,
  TRIFANS, **TRISTRIPS** }

  An enumeration of the geometric mesh primitives.

Public Member Functions

- **SubMesh** ()

  Constructor.
- virtual ~**SubMesh** ()

  Destructor.
- void **AddIndex** (unsigned int _i)

  Add an index to the mesh.
- void **AddNodeAssignment** (unsigned int _vertex, unsigned int _node, float _weight)

  Add a vertex - skeleton node assignment.
- void **AddNormal** (const **math::Vector3** &_n)

  Add a normal to the mesh.
- void **AddNormal** (double _x, double _y, double _z)

  Add a normal to the mesh.
- void **AddTexCoord** (double _u, double _v)

  Add a texture coord to the mesh.

- void **AddVertex** (const **math::Vector3** &_v)

  Add a vertex to the mesh.
- void **AddVertex** (double _x, double _y, double _z)

  Add a vertex to the mesh.
- void **CopyNormals** (const std::vector< **math::Vector3** > &_norms)

  Copy normals from a vector.
- void **CopyVertices** (const std::vector< **math::Vector3** > &_verts)

  Copy vertices from a vector.
- void **FillArrays** (float **_vertArr, int **_indArr) const

  Put all the data into flat arrays.
- void **GenSphericalTexCoord** (const **math::Vector3** &_center)

  Generate texture coordinates using spherical projection from center.
- unsigned int **GetIndex** (unsigned int _i) const

  Get an index.
- unsigned int **GetIndexCount** () const

  Return the number of indicies.
- unsigned int **GetMaterialIndex** () const

  Get the material index.
- **math::Vector3 GetMax** () const

  Get the maximun X, Y, Z values.
- unsigned int **GetMaxIndex** () const

  Get the highest index value.
- **math::Vector3 GetMin** () const

  Get the minimum X, Y, Z values.
- **NodeAssignment GetNodeAssignment** (unsigned int _i) const

  Get a vertex - skeleton node assignment.
- unsigned int **GetNodeAssignmentsCount** () const

  Return the number of vertex - skeleton node assignments.
- **math::Vector3 GetNormal** (unsigned int _i) const

  Get a normal.
- unsigned int **GetNormalCount** () const

  Return the number of normals.
- **PrimitiveType GetPrimitiveType** () const

  Get the primitive type.
- **math::Vector2d GetTexCoord** (unsigned int _i) const

  Get a tex coord.
- unsigned int **GetTexCoordCount** () const

  Return the number of texture coordinates.
- **math::Vector3 GetVertex** (unsigned int _i) const

  Get a vertex.
- unsigned int **GetVertexCount** () const

  Return the number of vertices.
- unsigned int **GetVertexIndex** (const **math::Vector3** &_v) const

  Get the index of the vertex.
- bool **HasVertex** (const **math::Vector3** &_v) const

  Return true if this submesh has the vertex.
- void **RecalculateNormals** ()

Recalculate all the normals.

- void **Scale** (double _factor)

  Scale all vertices by _factor.
- void **SetIndexCount** (unsigned int _count)

  Resize the index array.
- void **SetMaterialIndex** (unsigned int _index)

  Set the material index.
- void **SetNormal** (unsigned int _i, const **math::Vector3** &_n)

  Set a normal.
- void **SetNormalCount** (unsigned int _count)

  Resize the normal array.
- void **SetPrimitiveType** (**PrimitiveType** _type)

  Set the primitive type.
- void **SetSubMeshCenter** (**math::Vector3** _center)

  Reset mesh center to geometric center.
- void **SetTexCoord** (unsigned int _i, const **math::Vector2d** &_t)

  Set a tex coord.
- void **SetTexCoordCount** (unsigned int _count)

  Resize the texture coordinate array.
- void **SetVertex** (unsigned int _i, const **math::Vector3** &_v)

  Set a vertex.
- void **SetVertexCount** (unsigned int _count)

  Resize the vertex array.

10.187.1 Detailed Description

A child mesh.

10.187.2 Member Enumeration Documentation

10.187.2.1 enum gazebo::common::SubMesh::PrimitiveType

An enumeration of the geometric mesh primitives.

Enumerator:

POINTS

LINES

LINESTRIPS

TRIANGLES

TRIFANS

TRISTRIPS

10.187.3 Constructor & Destructor Documentation

10.187.3.1 gazebo::common::SubMesh::SubMesh ( )

Constructor.

10.187.3.2 virtual gazebo::common::SubMesh::~SubMesh ( ) [virtual]

Destructor.

10.187.4 Member Function Documentation

10.187.4.1 void gazebo::common::SubMesh::AddIndex (unsigned int _i)

Add an index to the mesh.

Parameters

in	*_i*	the new vertex index

10.187.4.2 void gazebo::common::SubMesh::AddNodeAssignment (unsigned int _vertex, unsigned int _node, float _weight)

Add a vertex - skeleton node assignment.

Parameters

in	*_vertex*	the vertex index
in	*_node*	the node index
in	*_weight*	the weight (between 0 and 1)

10.187.4.3 void gazebo::common::SubMesh::AddNormal (const math::Vector3 & _n)

Add a normal to the mesh.

Parameters

in	*_n*	the normal

10.187.4.4 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

10.187.4.5 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

10.187.4.6 void gazebo::common::SubMesh::AddVertex (const math::Vector3 & _v)

Add a vertex to the mesh.

Parameters

in	*_v*	the new position

10.187.4.7 void gazebo::common::SubMesh::AddVertex (double _x, double _y, double _z)

Add a vertex to the mesh.

Parameters

in	*_x*	position along x
in	*_y*	position along y
in	*_z*	position along z

10.187.4.8 void gazebo::common::SubMesh::CopyNormals (const std::vector< math::Vector3 > & _norms)

Copy normals from a vector.

Parameters

in	*_norms*	to copy from

10.187.4.9 void gazebo::common::SubMesh::CopyVertices (const std::vector< math::Vector3 > & _verts)

Copy vertices from a vector.

Parameters

in	*_verts*	the vertices to copy from

10.187.4.10 void gazebo::common::SubMesh::FillArrays (float ** _vertArr, int ** _indArr) const

Put all the data into flat arrays.

Parameters

in	*_verArr*	
in	*_indArr*	

10.187.4.11 void gazebo::common::SubMesh::GenSphericalTexCoord (const math::Vector3 & _center)

Generate texture coordinates using spherical projection from center.

Parameters

in	*_center*	

10.187.4.12 unsigned int gazebo::common::SubMesh::GetIndex (unsigned int *_i*) const

Get an index.

Parameters

in	*_i*	

10.187.4.13 unsigned int gazebo::common::SubMesh::GetIndexCount ( ) const

Return the number of indicies.

10.187.4.14 unsigned int gazebo::common::SubMesh::GetMaterialIndex ( ) const

Get the material index.

10.187.4.15 math::Vector3 gazebo::common::SubMesh::GetMax ( ) const

Get the maximun X, Y, Z values.

Returns

10.187.4.16 unsigned int gazebo::common::SubMesh::GetMaxIndex ( ) const

Get the highest index value.

10.187.4.17 math::Vector3 gazebo::common::SubMesh::GetMin ( ) const

Get the minimum X, Y, Z values.

Returns

10.187.4.18 NodeAssignment gazebo::common::SubMesh::GetNodeAssignment (unsigned int *_i*) const

Get a vertex - skeleton node assignment.

Parameters

in	*_i*	the index of the assignment

10.187.4.19 unsigned int gazebo::common::SubMesh::GetNodeAssignmentsCount ( ) const

Return the number of vertex - skeleton node assignments.

10.187.4.20 math::Vector3 gazebo::common::SubMesh::GetNormal (unsigned int _i) const

Get a normal.

Parameters

in	_i	the normal index

Returns

the orientation of the normal, or throws an exception

10.187.4.21 unsigned int gazebo::common::SubMesh::GetNormalCount ( ) const

Return the number of normals.

10.187.4.22 PrimitiveType gazebo::common::SubMesh::GetPrimitiveType ( ) const

Get the primitive type.

Returns

the primitive type

10.187.4.23 math::Vector2d gazebo::common::SubMesh::GetTexCoord (unsigned int _i) const

Get a tex coord.

Parameters

in	_i	the texture index

Returns

the texture coordinates

10.187.4.24 unsigned int gazebo::common::SubMesh::GetTexCoordCount ( ) const

Return the number of texture coordinates.

10.187.4.25 math::Vector3 gazebo::common::SubMesh::GetVertex (unsigned int _i) const

Get a vertex.

Parameters

in	*_i*	the vertex index

Returns

the position or throws an exception

10.187.4.26 unsigned int gazebo::common::SubMesh::GetVertexCount ( ) const

Return the number of vertices.

10.187.4.27 unsigned int gazebo::common::SubMesh::GetVertexIndex (const math::Vector3 & *_v*) const

Get the index of the vertex.

Parameters

in	*_v*	

10.187.4.28 bool gazebo::common::SubMesh::HasVertex (const math::Vector3 & *_v*) const

Return true if this submesh has the vertex.

Parameters

in	*_v*	

10.187.4.29 void gazebo::common::SubMesh::RecalculateNormals ( )

Recalculate all the normals.

10.187.4.30 void gazebo::common::SubMesh::Scale (double *_factor*)

Scale all vertices by _factor.

Parameters

in	*_factor*	Scaling factor

10.187.4.31 void gazebo::common::SubMesh::SetIndexCount (unsigned int *_count*)

Resize the index array.

Parameters

in	*_count*	the new size of the array

10.187.4.32 void gazebo::common::SubMesh::SetMaterialIndex (unsigned int _index)

Set the material index.

Relates to the parent mesh material list

Parameters

in	*_index*	

10.187.4.33 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

10.187.4.34 void gazebo::common::SubMesh::SetNormalCount (unsigned int _count)

Resize the normal array.

Parameters

in	*_count*	the new size of the array

10.187.4.35 void gazebo::common::SubMesh::SetPrimitiveType (PrimitiveType _type)

Set the primitive type.

Parameters

in	*_type*	the type

10.187.4.36 void gazebo::common::SubMesh::SetSubMeshCenter (math::Vector3 _center)

Reset mesh center to geometric center.

Parameters

in	*_center*	

10.187.4.37 void gazebo::common::SubMesh::SetTexCoord (unsigned int _i, const math::Vector2d & _t)

Set a tex coord.

Parameters

in	*_i*	
in	*_t*	

10.187.4.38 void gazebo::common::SubMesh::SetTexCoordCount (unsigned int *_count*)

Resize the texture coordinate array.

Parameters

in	*_count*	

10.187.4.39 void gazebo::common::SubMesh::SetVertex (unsigned int *_i*, const **math::Vector3** & *_v*)

Set a vertex.

Parameters

in	*_i*	the index
in	*_v*	the position

10.187.4.40 void gazebo::common::SubMesh::SetVertexCount (unsigned int *_count*)

Resize the vertex array.

Parameters

in	*_count*	the new size of the array

The documentation for this class was generated from the following file:

- **Mesh.hh**

10.188 gazebo::transport::SubscribeOptions Class Reference

Options for a subscription.

`#include <SubscribeOptions.hh>`

Public Member Functions

- **SubscribeOptions** ()
- bool **GetLatching** () const
- std::string **GetMsgType** () const
- **NodePtr GetNode** () const
- std::string **GetTopic** () const
- template<class M >
  void **Init** (const std::string &_topic, **NodePtr** _node, bool _latching)

10.188.1 Detailed Description

Options for a subscription.

10.188.2 Constructor & Destructor Documentation

10.188.2.1 gazebo::transport::SubscribeOptions::SubscribeOptions ( ) [inline]

10.188.3 Member Function Documentation

10.188.3.1 bool gazebo::transport::SubscribeOptions::GetLatching ( ) const [inline]

10.188.3.2 std::string gazebo::transport::SubscribeOptions::GetMsgType ( ) const [inline]

10.188.3.3 NodePtr gazebo::transport::SubscribeOptions::GetNode ( ) const [inline]

10.188.3.4 std::string gazebo::transport::SubscribeOptions::GetTopic ( ) const [inline]

10.188.3.5 template<class M > void gazebo::transport::SubscribeOptions::Init (const std::string & *_topic,* NodePtr *_node,* bool *_latching*) [inline]

References gzthrow, and NULL.

The documentation for this class was generated from the following file:

- **SubscribeOptions.hh**

10.189 gazebo::transport::Subscriber Class Reference

A subscriber to a topic.

`#include <Subscriber.hh>`

Public Member Functions

- **Subscriber** (const std::string &topic, **NodePtr** _node)

  Constructor.
- virtual ~**Subscriber** ()

  Destructor.
- std::string **GetTopic** () const

  Get the topic name.
- void **Unsubscribe** () const

  Unsubscribe from the topic.

10.189.1 Detailed Description

A subscriber to a topic.

10.189.2 Constructor & Destructor Documentation

10.189.2.1 gazebo::transport::Subscriber::Subscriber (const std::string & *topic,* NodePtr *_node*)

Constructor.

10.189.2.2 virtual gazebo::transport::Subscriber::~Subscriber ( ) [virtual]

Destructor.

10.189.3 Member Function Documentation

10.189.3.1 std::string gazebo::transport::Subscriber::GetTopic ( ) const

Get the topic name.

10.189.3.2 void gazebo::transport::Subscriber::Unsubscribe ( ) const

Unsubscribe from the topic.

The documentation for this class was generated from the following file:

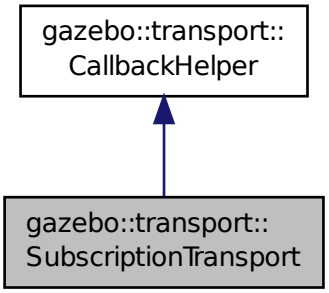
- **Subscriber.hh**

10.190 gazebo::transport::SubscriptionTransport Class Reference

Handles sending data over the wire to remote subscribers.

`#include <SubscriptionTransport.hh>`

Inheritance diagram for gazebo::transport::SubscriptionTransport:



Public Member Functions

- **SubscriptionTransport** ()

  Constructor.
- virtual ~**SubscriptionTransport** ()

  Destructor.
- const **ConnectionPtr** & **GetConnection** () const

  Get the connection.
- virtual bool **HandleData** (const std::string &newdata)

  Output a message to a connection.
- void **Init** (const **ConnectionPtr** &conn, bool _latching)

  Initialize the publication link.
- virtual bool **IsLocal** () const

  Return true if the callback is local, false if the callback is tied to a remote connection.

Additional Inherited Members

10.190.1 Detailed Description

Handles sending data over the wire to remote subscribers.

10.190.2 Constructor & Destructor Documentation

10.190.2.1 gazebo::transport::SubscriptionTransport::SubscriptionTransport ( )

Constructor.

10.190.2.2 virtual gazebo::transport::SubscriptionTransport::~SubscriptionTransport ( ) [virtual]

Destructor.

10.190.3 Member Function Documentation

10.190.3.1 const ConnectionPtr& gazebo::transport::SubscriptionTransport::GetConnection ( ) const

Get the connection.

10.190.3.2 virtual bool gazebo::transport::SubscriptionTransport::HandleData (const std::string & *newdata*) [virtual]

Output a message to a connection.

Implements **gazebo::transport::CallbackHelper** (p. 231).

10.190.3.3 void gazebo::transport::SubscriptionTransport::Init (const ConnectionPtr & *conn,* bool *_latching*)

Initialize the publication link.

10.190.3.4 virtual bool gazebo::transport::SubscriptionTransport::IsLocal ( ) const [virtual]

Return true if the callback is local, false if the callback is tied to a remote connection.

Implements **gazebo::transport::CallbackHelper** (p. 231).

The documentation for this class was generated from the following file:

- **SubscriptionTransport.hh**

10.191 gazebo::physics::SurfaceParams Class Reference

SurfaceParams (p. 830) defines various Surface contact parameters.

`#include <SurfaceParams.hh>`

Public Member Functions

- **SurfaceParams** ()

  Constructor.
- virtual ~**SurfaceParams** ()

  Constructor.
- void **FillSurfaceMsg** (msgs::Surface &_msg)

  Fill in a surface message.
- virtual void **Load** (**sdf::ElementPtr** _sdf)

  Load the contact params.
- virtual void **ProcessMsg** (const msgs::Surface &_msg)

Public Attributes

- double **bounce**

  bounce restitution coefficient [0,1], with 0 being inelastic, and 1 being perfectly elastic.
- double **bounceThreshold**

  minimum contact velocity for bounce to take effect, otherwise the collision is treated as an inelastic collision.
- double **cfm**

  Constraint Force Mixing parameter.
- double **erp**

  Error Reduction Parameter.
- **math::Vector3 fdir1**

  *Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 833)) of the friction pyramid.*
- double **kd**

  *spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 832) and **SurfaceParams::erp** (p. 832).*
- double **kp**

  *spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 832) and **SurfaceParams::erp** (p. 832).*
- double **maxVel**

  Maximum interpenetration error correction velocity.
- double **minDepth**

Minimum depth before ERP takes effect.

- double **mu1**

  Dry friction coefficient in the primary friction direction as defined by the friction pyramid.

- double **mu2**

  Dry friction coefficient in the second friction direction as defined by the friction pyramid.

- double **slip1**

  Artificial contact slip in the primary friction direction.

- double **slip2**

  Artificial contact slip in the secondary friction dirction.

10.191.1 Detailed Description

SurfaceParams (p. 830) defines various Surface contact parameters.

These parameters defines the properties of a **physics::Contact** (p. 296) constraint.

10.191.2 Constructor & Destructor Documentation

10.191.2.1 gazebo::physics::SurfaceParams::SurfaceParams ( )

Constructor.

10.191.2.2 virtual gazebo::physics::SurfaceParams::~SurfaceParams ( ) [virtual]

Constructor.

10.191.3 Member Function Documentation

10.191.3.1 void gazebo::physics::SurfaceParams::FillSurfaceMsg (msgs::Surface & *_msg*)

Fill in a surface message.

10.191.3.2 virtual void gazebo::physics::SurfaceParams::Load (sdf::ElementPtr *_sdf*) [virtual]

Load the contact params.

10.191.3.3 virtual void gazebo::physics::SurfaceParams::ProcessMsg (const msgs::Surface & *_msg*) [virtual]

10.191.4 Member Data Documentation

10.191.4.1 double gazebo::physics::SurfaceParams::bounce

bounce restitution coefficient [0,1], with 0 being inelastic, and 1 being perfectly elastic.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.191.4.2 double gazebo::physics::SurfaceParams::bounceThreshold

minimum contact velocity for bounce to take effect, otherwise the collision is treated as an inelastic collision.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.191.4.3 double gazebo::physics::SurfaceParams::cfm

Constraint Force Mixing parameter.

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_0 for more details.

10.191.4.4 double gazebo::physics::SurfaceParams::erp

Error Reduction Parameter.

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_0 for more details.

10.191.4.5 math::Vector3 gazebo::physics::SurfaceParams::fdir1

Primary friction direction for dry friction coefficient (**SurfaceParams::mu1** (p. 833)) of the friction pyramid.

```
If undefined, a vector consstrained to be perpendicular
to the contact normal in the global y-z plane is used.
```

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.191.4.6 double gazebo::physics::SurfaceParams::kd

spring damping constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 832) and **SurfaceParams::erp** (p. 832).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.191.4.7 double gazebo::physics::SurfaceParams::kp

spring constant equivalents of a contact as a function of **SurfaceParams::cfm** (p. 832) and **SurfaceParams::erp** (p. 832).

See Also

See for example http://www.ode.org/ode-latest-userguide.html#sec_3_8_2 for more details.

10.191.4.8 double gazebo::physics::SurfaceParams::maxVel

Maximum interpenetration error correction velocity.

If set to 0, two objects interpenetrating each other will not be pushed apart.

See Also

See dWroldSetContactMaxCorrectingVel (http://www.ode.org/ode-latest-userguide.html#sec_5_2_0)

10.191.4.9 double gazebo::physics::SurfaceParams::minDepth

Minimum depth before ERP takes effect.

See Also

See dWorldSetContactSurfaceLayer (http://www.ode.org/ode-latest-userguide.html#sec_5_2_0)

10.191.4.10 double gazebo::physics::SurfaceParams::mu1

Dry friction coefficient in the primary friction direction as defined by the friction pyramid.

This is fdir1 if defined, otherwise, a vector consstrained to be perpendicular to the contact normal in the global y-z plane is used.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.191.4.11 double gazebo::physics::SurfaceParams::mu2

Dry friction coefficient in the second friction direction as defined by the friction pyramid.

This is fdir1 if defined, otherwise, a vector consstrained to be perpendicular to the contact normal in the global y-z plane is used.

See Also

http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.191.4.12 double gazebo::physics::SurfaceParams::slip1

Artificial contact slip in the primary friction direction.

See Also

See dContactSlip1 in http://www.ode.org/ode-latest-userguide.html#sec_7_3_7

10.191.4.13 double gazebo::physics::SurfaceParams::slip2

Artificial contact slip in the secondary friction dirction.

See Also

See dContactSlip2 in `http://www.ode.org/ode-latest-userguide.html#sec_7_3_7`

The documentation for this class was generated from the following file:

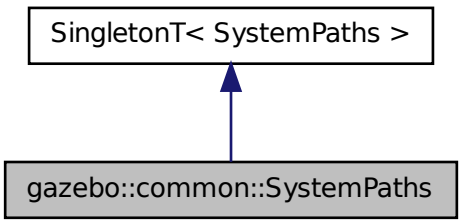
- **SurfaceParams.hh**

10.192 gazebo::common::SystemPaths Class Reference

Functions to handle getting system paths, keeps track of:

`#include <SystemPaths.hh>`

Inheritance diagram for gazebo::common::SystemPaths:



Public Member Functions

- void **AddGazeboPaths** (const std::string &_path)

  Add colon delimited paths to Gazebo install.
- void **AddOgrePaths** (const std::string &_path)

  Add colon delimited paths to ogre install.
- void **AddPluginPaths** (const std::string &_path)

  Add colon delimited paths to plugins.
- void **AddSearchPathSuffix** (const std::string &_suffix)

  add _suffix to the list of path search sufixes
- void **ClearGazeboPaths** ()

  clear out SystemPaths::gazeboPaths
- void **ClearOgrePaths** ()

  clear out SystemPaths::ogrePaths
- void **ClearPluginPaths** ()

clear out SystemPaths::pluginPaths

- std::string **FindFile** (const std::string &_filename, bool _searchLocalPath=true)

  Find a file in the gazebo paths.
- std::string **FindFileURI** (const std::string &_uri)

  Find a file or path using a URI.
- const std::list< std::string > & **GetGazeboPaths** ()

  Get the gazebo install paths.
- std::string **GetLogPath** () const

  Get the log path.
- const std::list< std::string > & **GetModelPaths** ()

  Get the model paths.
- const std::list< std::string > & **GetOgrePaths** ()

  Get the ogre install paths.
- const std::list< std::string > & **GetPluginPaths** ()

  Get the plugin paths.
- std::string **GetWorldPathExtension** ()

  Returns the world path extension.

Public Attributes

- bool **gazeboPathsFromEnv**

  *if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 837)*
- bool **modelPathsFromEnv**

  *if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 837)*
- bool **ogrePathsFromEnv**

  *if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 837)*
- bool **pluginPathsFromEnv**

  *if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 837)*

Additional Inherited Members

10.192.1 Detailed Description

Functions to handle getting system paths, keeps track of:

- SystemPaths::gazeboPaths - media paths containing worlds, models, sdf descriptions, material scripts, textures.
- SystemPaths::ogrePaths - ogre library paths. Should point to **Ogre** (p. 118) RenderSystem_GL.so et. al.
- SystemPaths::pluginPaths - plugin library paths for common::WorldPlugin

10.192.2 Member Function Documentation

10.192.2.1 void gazebo::common::SystemPaths::AddGazeboPaths (const std::string & *_path*)

Add colon delimited paths to Gazebo install.

Parameters

in	*_path*	the directory to add

10.192.2.2 void gazebo::common::SystemPaths::AddOgrePaths (const std::string & *_path*)

Add colon delimited paths to ogre install.

Parameters

in	*_path*	the directory to add

10.192.2.3 void gazebo::common::SystemPaths::AddPluginPaths (const std::string & *_path*)

Add colon delimited paths to plugins.

Parameters

in	*_path*	the directory to add

10.192.2.4 void gazebo::common::SystemPaths::AddSearchPathSuffix (const std::string & *_suffix*)

add _suffix to the list of path search sufixes

10.192.2.5 void gazebo::common::SystemPaths::ClearGazeboPaths ( )

clear out SystemPaths::gazeboPaths

10.192.2.6 void gazebo::common::SystemPaths::ClearOgrePaths ( )

clear out SystemPaths::ogrePaths

10.192.2.7 void gazebo::common::SystemPaths::ClearPluginPaths ( )

clear out SystemPaths::pluginPaths

10.192.2.8 std::string gazebo::common::SystemPaths::FindFile (const std::string & *_filename,* bool *_searchLocalPath* = `true`)

Find a file in the gazebo paths.

Parameters

in	*_filename*	Name of the file to find.
in	*_searchLocal-Path*	True to search in the current working directory.

10.192.2.9 std::string gazebo::common::SystemPaths::FindFileURI (const std::string & _uri)

Find a file or path using a URI.

Parameters

in	*_uri*	the uniform resource identifier

10.192.2.10 const std::list<std::string>& gazebo::common::SystemPaths::GetGazeboPaths ( )

Get the gazebo install paths.

Returns

a list of paths

10.192.2.11 std::string gazebo::common::SystemPaths::GetLogPath ( ) const

Get the log path.

Returns

the path

10.192.2.12 const std::list<std::string>& gazebo::common::SystemPaths::GetModelPaths ( )

Get the model paths.

Returns

a list of paths

10.192.2.13 const std::list<std::string>& gazebo::common::SystemPaths::GetOgrePaths ( )

Get the ogre install paths.

Returns

a list of paths

10.192.2.14 const std::list<std::string>& gazebo::common::SystemPaths::GetPluginPaths ( )

Get the plugin paths.

Returns

a list of paths

10.192.2.15 std::string gazebo::common::SystemPaths::GetWorldPathExtension ( )

Returns the world path extension.

Returns

Right now, it just returns "/worlds"

10.192.3 Member Data Documentation

10.192.3.1 bool gazebo::common::SystemPaths::gazeboPathsFromEnv

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 837)

10.192.3.2 bool gazebo::common::SystemPaths::modelPathsFromEnv

if true, call UpdateGazeboPaths() within **GetGazeboPaths()** (p. 837)

10.192.3.3 bool gazebo::common::SystemPaths::ogrePathsFromEnv

if true, call UpdateOgrePaths() within **GetOgrePaths()** (p. 837)

10.192.3.4 bool gazebo::common::SystemPaths::pluginPathsFromEnv

if true, call UpdatePluginPaths() within **GetPluginPaths()** (p. 837)

The documentation for this class was generated from the following file:

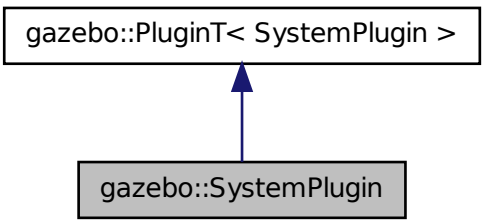
- **SystemPaths.hh**

10.193 gazebo::SystemPlugin Class Reference

A plugin loaded within the gzserver on startup.

`#include <Plugin.hh>`

Inheritance diagram for gazebo::SystemPlugin:



Public Member Functions

- **SystemPlugin** ()

  Constructor.
- virtual ∼**SystemPlugin** ()

  Destructor.
- virtual void **Init** ()

  Initialize the plugin.
- virtual void **Load** (int _argc=0, char ∗∗_argv=**NULL**)=0

  Load function.
- virtual void **Reset** ()

  Override this method for custom plugin reset behavior.

Additional Inherited Members

10.193.1 Detailed Description

A plugin loaded within the gzserver on startup.

See `reference`.

Todo how to make doxygen reference to the file gazebo.cc::g_plugins?

10.193.2 Constructor & Destructor Documentation

10.193.2.1 gazebo::SystemPlugin::SystemPlugin ( ) [inline]

Constructor.

References gazebo::SYSTEM_PLUGIN, and gazebo::PluginT< SystemPlugin >::type.

10.193.2.2 virtual gazebo::SystemPlugin::∼SystemPlugin ( ) [inline], [virtual]

Destructor.

10.193.3 Member Function Documentation

10.193.3.1 virtual void gazebo::SystemPlugin::Init ( ) [inline], [virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.193.3.2 virtual void gazebo::SystemPlugin::Load (int *_argc* = 0, char ∗∗ *_argv* = **NULL**) [pure virtual]

Load function.

Called before Gazebo is loaded. Must not block.

Parameters

_argc	Number of command line arguments.
_argv	Array of command line arguments.

10.193.3.3 virtual void gazebo::SystemPlugin::Reset ( ) [inline], [virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

- **common/Plugin.hh**

10.194 gazebo::common::Time Class Reference

A **Time** (p. 840) class, can be used to hold wall- or sim-time.

`#include <Time.hh>`

Public Member Functions

- **Time** ()
  Constructors.
- **Time** (const **Time** &_time)
  Copy constructor.
- **Time** (const struct timeval &_tv)
  Constructor.
- **Time** (const struct timespec &_tv)
  Constructor.
- **Time** (int32_t _sec, int32_t _nsec)
  Constructor.
- **Time** (double _time)
  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 double **MicToNano** (double _ms)

  Convert microseconds to nanoseconds.

- static double **MilToNano** (double _ms)

  Convert milliseconds to nanoseconds.

- static **Time MSleep** (unsigned int _ms)

  Millisecond sleep.

- static **Time NSleep** (unsigned int _ns)

  Nano sleep.

- static **Time NSleep** (**Time** _time)

  Nano sleep.

- static double **SecToNano** (double _sec)

  Convert seconds to nanoseconds.

Public Attributes

- int32_t **nsec**

  Microseconds.

- int32_t **sec**

  Seconds.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Time** &_time)

  Stream insertion operator.

- std::istream & **operator**>> (std::istream &_in, **gazebo::common::Time** &_time)

  Stream extraction operator.

10.194.1 Detailed Description

A **Time** (p. 840) class, can be used to hold wall- or sim-time.

stored as sec and nano-sec.

10.194.2 Constructor & Destructor Documentation

10.194.2.1 gazebo::common::Time::Time ( )

Constructors.

10.194.2.2 gazebo::common::Time::Time (const Time & *_time*)

Copy constructor.

Parameters

in	*time*	**Time** (p. 840) to copy

10.194.2.3 gazebo::common::Time::Time (const struct timeval & *_tv*)

Constructor.

Parameters

in	*tv*	**Time** (p. 840) to initialize to

10.194.2.4 gazebo::common::Time::Time (const struct timespec & *_tv*)

Constructor.

Parameters

in	*tv*	**Time** (p. 840) to initialize to

10.194.2.5 gazebo::common::Time::Time (int32_t *_sec*, int32_t *_nsec*)

Constructor.

Parameters

in	*sec*	Seconds
in	*nsec*	Microseconds

10.194.2.6 gazebo::common::Time::Time (double *_time*)

Constuctor.

Parameters

in	*time*	**Time** (p. 840) in double format sec.nsec

10.194.2.7 virtual gazebo::common::Time::~Time ( ) [virtual]

Destructor.

10.194.3 Member Function Documentation

10.194.3.1 double gazebo::common::Time::Double ( ) const

Get the time as a double.

Returns

Time (p. 840) as a double in seconds

10.194.3.2 float gazebo::common::Time::Float ( ) const

Get the time as a float.

Returns

Time (p. 840) as a float in seconds

10.194.3.3 static const Time& gazebo::common::Time::GetWallTime ( ) [static]

Get the wall time.

Returns

the current time

10.194.3.4 static double gazebo::common::Time::MicToNano (double *_ms*) [inline], [static]

Convert microseconds to nanoseconds.

Parameters

_ms	microseconds

Returns

nanoseconds

10.194.3.5 static double gazebo::common::Time::MilToNano (double *_ms*) [inline], [static]

Convert milliseconds to nanoseconds.

Parameters

in	*_ms*	milliseconds

Returns

nanoseconds

10.194.3.6 static Time gazebo::common::Time::MSleep (unsigned int *_ms*) [static]

Millisecond sleep.

Parameters

in	*_ms*	milliseconds

10.194.3.7 static Time gazebo::common::Time::NSleep (unsigned int *_ns*) [static]

Nano sleep.

Parameters

in	*_ns*	nanoseconds

10.194.3.8 static Time gazebo::common::Time::NSleep (Time *_time*) [static]

Nano sleep.

Parameters

in	*_time*	is a **Time** (p. 840)

10.194.3.9 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

10.194.3.10 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

10.194.3.11 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

10.194.3.12 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

10.194.3.13 Time gazebo::common::Time::operator* (const struct timeval & *tv*) const

Multiplication operator.

Returns

Time (p. 840) instance

10.194.3.14 Time gazebo::common::Time::operator* (const struct timespec & *tv*) const

Multiplication operator.

Returns

Time (p. 840) instance

10.194.3.15 Time gazebo::common::Time::operator* (const Time & *time*) const

Multiplication operators.

Parameters

in	*time*	the scaling factor

Returns

a scaled **Time** (p. 840) instance

10.194.3.16 const Time& gazebo::common::Time::operator*= (const struct timeval & *tv*)

Multiplication assignment operator.

Parameters

in	*tv*	the scaling duration

Returns

a reference to this instance

10.194.3.17 const Time& gazebo::common::Time::operator*= (const struct timespec & *tv*)

Multiplication assignment operator.

Parameters

in	*tv*	the scaling duration

Returns

a reference to this instance

10.194.3.18 const Time& gazebo::common::Time::operator*= (const Time & *time*)

Multiplication operators.

Parameters

in	*time*	scale factor

Returns

a scaled **Time** (p. 840) instance

10.194.3.19 Time gazebo::common::Time::operator+ (const struct timeval & *tv*) const

Addition operators.

Parameters

in	*tv*	the time to add

Returns

a **Time** (p. 840) instance

10.194.3.20 Time gazebo::common::Time::operator+ (const struct timespec & *tv*) const

Addition operators.

Parameters

in	*tv*	the time to add

Returns

a **Time** (p. 840) instance

10.194.3.21 Time gazebo::common::Time::operator+ (const Time & *time*) const

Addition operators.

Returns

a **Time** (p. 840) instance

10.194.3.22 const Time& gazebo::common::Time::operator+= (const struct timeval & *tv*)

Addition assignment operator.

Parameters

in	*tv*	

Returns

a reference to this instance

10.194.3.23 const Time& gazebo::common::Time::operator+= (const struct timespec & *tv*)

Addition assignment operator.

Parameters

in	*tv*	

Returns

a reference to this instance

10.194.3.24 const Time& gazebo::common::Time::operator+= (const Time & *time*)

Addition assignemtn operator.

Returns

a **Time** (p. 840) instance

10.194.3.25 Time gazebo::common::Time::operator- (const struct timeval & *tv*) const

Subtraction operator.

Returns

a **Time** (p. 840) instance

10.194.3.26 Time gazebo::common::Time::operator- (const struct timespec & *tv*) const

Subtraction operator.

Returns

a **Time** (p. 840) instance

10.194.3.27 Time gazebo::common::Time::operator- (const Time & *time*) const

Subtraction operator.

Returns

a **Time** (p. 840) instance

10.194.3.28 const Time& gazebo::common::Time::operator-= (const struct timeval & *tv*)

Subtraction assignment operator.

Returns

a **Time** (p. 840) instance

10.194.3.29 const Time& gazebo::common::Time::operator-= (const struct timespec & *tv*)

Subtraction assignment operator.

Returns

a **Time** (p. 840) instance

10.194.3.30 const Time& gazebo::common::Time::operator-= (const Time & *time*)

Subtraction assignment operator.

Returns

a reference to this instance

10.194.3.31 Time gazebo::common::Time::operator/ (const struct timeval & *tv*) const

Division operator.

Parameters

in	*tv*	a timeval divisor

Returns

a **Time** (p. 840) instance

10.194.3.32 Time gazebo::common::Time::operator/ (const struct timespec & *tv*) const

Division operator.

Parameters

in	*tv*	a timespec divisor

Returns

a **Time** (p. 840) instance

10.194.3.33 Time gazebo::common::Time::operator/ (const Time & *time*) const

Division operator.

Parameters

in	*time*	the divisor

Returns

a **Time** (p. 840) instance

10.194.3.34 const Time& gazebo::common::Time::operator/= (const struct timeval & *tv*)

Division assignment operator.

Parameters

in	*tv*	a divisor

Returns

a **Time** (p. 840) instance

10.194.3.35 const Time& gazebo::common::Time::operator/= (const struct timespec & *tv*)

Division assignment operator.

Parameters

in	*tv*	a divisor

Returns

a **Time** (p. 840) instance

10.194.3.36 const Time& gazebo::common::Time::operator/= (const Time & *time*)

Division assignment operator.

Parameters

in	*time*	the divisor

Returns

a **Time** (p. 840) instance

10.194.3.37 bool gazebo::common::Time::operator< (const struct timeval & *tv*) const

Less than operator.

Parameters

in	*tv*	the time to compare with

Returns

true if tv is shorter than this, false otherwise

10.194.3.38 bool gazebo::common::Time::operator< (const struct timespec & *tv*) const

Less than operator.

Parameters

in	*tv*	the time to compare with

Returns

true if tv is shorter than this, false otherwise

10.194.3.39 bool gazebo::common::Time::operator< (const Time & *time*) const

Less than operator.

Parameters

in	*time*	the time to compare with

Returns

true if time is shorter than this, false otherwise

10.194.3.40 bool gazebo::common::Time::operator< (double *time*) const

Less than operator.

Parameters

in	*time*	the time to compare with

Returns

true if time is shorter than this, false otherwise

10.194.3.41 bool gazebo::common::Time::operator<= (const struct timeval & *tv*) const

Less than or equal to operator.

Parameters

in	*tv*	the time to compare with

Returns

true if tv is shorter than or equal to this, false otherwise

10.194.3.42 bool gazebo::common::Time::operator<= (const struct timespec & *tv*) const

Less than or equal to operator.

Parameters

in	*tv*	the time to compare with

Returns

true if tv is shorter than or equal to this, false otherwise

10.194.3.43 bool gazebo::common::Time::operator<= (const Time & *time*) const

Less than or equal to operator.

Parameters

in	*time*	the time to compare with

Returns

true if time is shorter than or equal to this, false otherwise

10.194.3.44 bool gazebo::common::Time::operator<= (double *time*) const

Less than or equal to operator.

Parameters

in	*time*	the time to compare with

Returns

true if time is shorter than or equal to this, false otherwise

10.194.3.45 Time& gazebo::common::Time::operator= (const struct timeval & *tv*)

Assignment operator.

Parameters

in	*tv*	the new time

Returns

a reference to this instance

10.194.3.46 Time& gazebo::common::Time::operator= (const struct timespec & *tv*)

Assignment operator.

Parameters

in	*tv*	the new time

Returns

a reference to this instance

10.194.3.47 Time& gazebo::common::Time::operator= (const Time & *time*)

Assignment operator.

Parameters

in	*time*	the new time

Returns

a reference to this instance

10.194.3.48 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

10.194.3.49 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

10.194.3.50 bool gazebo::common::Time::operator== (const Time & *time*) const

Equal to operator.

Parameters

in	*tv*	the time to compare to

Returns

true if values are the same, false otherwise

10.194.3.51 bool gazebo::common::Time::operator== (double *time*) const

Equal to operator.

Parameters

in	*time*	the time to compare to

Returns

true if values are the same, false otherwise

10.194.3.52 bool gazebo::common::Time::operator> (const struct timeval & *tv*) const

Greater than operator.

Parameters

in	*time*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.194.3.53 bool gazebo::common::Time::operator> (const struct timespec & *tv*) const

Greater than operator.

Parameters

in	*time*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.194.3.54 bool gazebo::common::Time::operator> (const Time & *time*) const

Greater than operator.

Parameters

in	*time*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.194.3.55 bool gazebo::common::Time::operator> (double *time*) const

Greater than operator.

Parameters

in	*time*	the time to compare with

Returns

true if time is greater than this, false otherwise

10.194.3.56 bool gazebo::common::Time::operator>= (const struct timeval & *tv*) const

Greater than or equal operator.

Parameters

in	*tv*	the time to compare with

Returns

true if tv is greater than or equal to this, false otherwise

10.194.3.57 bool gazebo::common::Time::operator>= (const struct timespec & *tv*) const

Greater than or equal operator.

Parameters

in	*tv*	the time to compare with

Returns

true if tv is greater than or equal to this, false otherwise

10.194.3.58 bool gazebo::common::Time::operator>= (const Time & *time*) const

Greater than or equal operator.

Parameters

in	*time*	the time to compare with

Returns

true if time is greater than or equal to this, false otherwise

10.194.3.59 bool gazebo::common::Time::operator>= (double *time*) const

Greater than or equal operator.

Parameters

in	*time*	the time to compare with

Returns

true if time is greater than or equal to this, false otherwise

10.194.3.60 static double gazebo::common::Time::SecToNano (double *_sec*) [inline], [static]

Convert seconds to nanoseconds.

Parameters

in	*_sec*	duration in seconds

Returns

nanoseconds

10.194.3.61 void gazebo::common::Time::Set (int32_t *_sec*, int32_t *_nsec*)

Set to sec and nsec.

Parameters

in	*sec*	Seconds
in	*nsec*	micro seconds

10.194.3.62 void gazebo::common::Time::Set (double *_seconds*)

Set to seconds.

Parameters

in	*seconds*	Number of seconds

10.194.3.63 void gazebo::common::Time::SetToWallTime ( )

Set the time to the wall time.

10.194.4 Friends And Related Function Documentation

10.194.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::common::Time & *_time*) [friend]

Stream insertion operator.

Parameters

in	*_out*	the output stream
in	*_time*	time to write to the stream

Returns

the output stream

10.194.4.2 std::istream& operator>> (std::istream & *_in,* gazebo::common::Time & *_time*) [friend]

Stream extraction operator.

Parameters

in	*_in*	the input stream
in	*_time*	time to read from to the stream

Returns

the input stream

10.194.5 Member Data Documentation

10.194.5.1 int32_t gazebo::common::Time::nsec

Microseconds.

10.194.5.2 int32_t gazebo::common::Time::sec

Seconds.

The documentation for this class was generated from the following file:

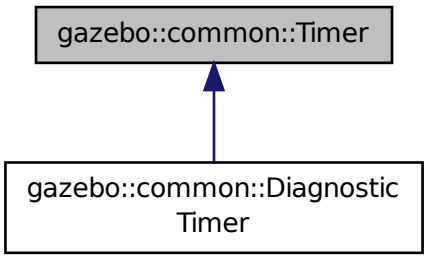
- **Time.hh**

10.195 gazebo::common::Timer Class Reference

A timer class, used to time things in real world walltime.

`#include <Timer.hh>`

Inheritance diagram for gazebo::common::Timer:



Public Member Functions

- **Timer** ()
  Constructor.
- virtual ~**Timer** ()
  Destructor.
- **Time GetElapsed** () const
  Get the elapsed time.
- void **Start** ()
  Start the timer.

Friends

- std::ostream & **operator**<< (std::ostream &out, const **gazebo::common::Timer** &t)
  stream operator friendly

10.195.1 Detailed Description

A timer class, used to time things in real world walltime.

10.195.2 Constructor & Destructor Documentation

10.195.2.1 gazebo::common::Timer::Timer ( )

Constructor.

10.195.2.2 virtual gazebo::common::Timer::~Timer ( ) [virtual]

Destructor.

10.195.3 Member Function Documentation

10.195.3.1 Time gazebo::common::Timer::GetElapsed ( ) const

Get the elapsed time.

10.195.3.2 void gazebo::common::Timer::Start ( )

Start the timer.

Referenced by gazebo::common::DiagnosticTimer::DiagnosticTimer().

10.195.4 Friends And Related Function Documentation

10.195.4.1 std::ostream& operator<< (std::ostream & *out,* const gazebo::common::Timer & *t*) [friend]

stream operator friendly

The documentation for this class was generated from the following file:

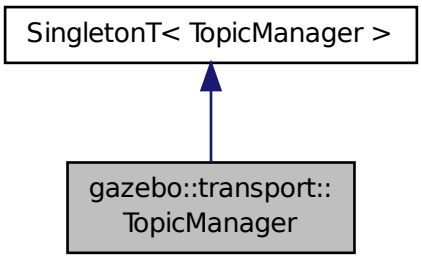
- **Timer.hh**

10.196 gazebo::transport::TopicManager Class Reference

Manages topics and their subscriptions.

`#include <TopicManager.hh>`

Inheritance diagram for gazebo::transport::TopicManager:



Public Types

- typedef std::map< std::string,
  std::list< **NodePtr** > > **SubNodeMap**

Public Member Functions

- void **AddNode** (**NodePtr** _node)
- template<typename M >
  PublisherPtr Advertise (const std::string &_topic, unsigned int _queueLimit, bool _latch)

  Advertise on a topic.
- void **ClearBuffers** ()
- void **ConnectPubToSub** (const std::string &topic, const **SubscriptionTransportPtr** &sublink)

  Connection *(*p. *288) a local* ***Publisher*** *(*p. *695) to a remote* ***Subscriber*** *(*p. *827).*
- void **ConnectSubscribers** (const std::string &topic)

  Connect all subscribers on a topic to known publishers.
- void **ConnectSubToPub** (const msgs::Publish &_pub)

  Connect a local ***Subscriber*** *(*p. *827) to a remote* ***Publisher*** *(*p. *695).*
- void **DisconnectPubFromSub** (const std::string &topic, const std::string &host, unsigned int port)

  Disconnect a local publisher from a remote subscriber.
- void **DisconnectSubFromPub** (const std::string &topic, const std::string &host, unsigned int port)

  Disconnection all local subscribers from a remote publisher.
- **PublicationPtr FindPublication** (const std::string &topic)
- void **Fini** ()
- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

  Get all the topic namespaces.
- void **Init** ()
- bool **IsAdvertised** (const std::string &_topic)

  Returns true if the topic has been advertised.
- void **PauseIncoming** (bool _pause)
- void **ProcessNodes** (bool _onlyOut=false)
- void **Publish** (const std::string &topic, const google::protobuf::Message &message, const boost::function< void()> &cb=**NULL**)

  Send a message.
- void **RegisterTopicNamespace** (const std::string &_name)

  Register a new topic namespace.
- void **RemoveNode** (unsigned int _id)
- **SubscriberPtr Subscribe** (const **SubscribeOptions** &options)

  Subscribe to a topic.
- void **Unadvertise** (const std::string &topic)

  Stop advertising on a topic.
- void **Unsubscribe** (const std::string &_topic, const **NodePtr** &_sub)

  Unsubscribe from a topic.
- **PublicationPtr UpdatePublications** (const std::string &topic, const std::string &msgType)

  Update our list of advertised topics.

Additional Inherited Members

10.196.1 Detailed Description

Manages topics and their subscriptions.

10.196.2 Member Typedef Documentation

10.196.2.1 typedef std::map<std::string, std::list<NodePtr> > gazebo::transport::TopicManager::SubNodeMap

10.196.3 Member Function Documentation

10.196.3.1 void gazebo::transport::TopicManager::AddNode (NodePtr *_node*)

10.196.3.2 template<typename M > PublisherPtr gazebo::transport::TopicManager::Advertise (const std::string & *_topic,* unsigned int *_queueLimit,* bool *_latch*) [inline]

Advertise on a topic.

Parameters

topic	The name of the topic

References gazebo::transport::Publication::AddPublisher(), gazebo::transport::Publication::AddSubscription(), FindPublication(), gazebo::transport::Publication::GetLocallyAdvertised(), gzthrow, SingletonT< T >::Instance(), NULL, gazebo::transport::Publication::SetLocallyAdvertised(), and UpdatePublications().

10.196.3.3 void gazebo::transport::TopicManager::ClearBuffers ( )

10.196.3.4 void gazebo::transport::TopicManager::ConnectPubToSub (const std::string & *topic,* const SubscriptionTransportPtr & *sublink*)

Connection (p. 288) a local **Publisher** (p. 695) to a remote **Subscriber** (p. 827).

10.196.3.5 void gazebo::transport::TopicManager::ConnectSubscribers (const std::string & *topic*)

Connect all subscribers on a topic to known publishers.

10.196.3.6 void gazebo::transport::TopicManager::ConnectSubToPub (const msgs::Publish & *_pub*)

Connect a local **Subscriber** (p. 827) to a remote **Publisher** (p. 695).

10.196.3.7 void gazebo::transport::TopicManager::DisconnectPubFromSub (const std::string & *topic,* const std::string & *host,* unsigned int *port*)

Disconnect a local publisher from a remote subscriber.

10.196.3.8 void gazebo::transport::TopicManager::DisconnectSubFromPub (const std::string & *topic,* const std::string & *host,* unsigned int *port*)

Disconnection all local subscribers from a remote publisher.

10.196.3.9 PublicationPtr gazebo::transport::TopicManager::FindPublication (const std::string & *topic*)

Referenced by Advertise().

10.196.3.10 void gazebo::transport::TopicManager::Fini ( )

10.196.3.11 void gazebo::transport::TopicManager::GetTopicNamespaces (std::list< std::string > & *_namespaces*)

Get all the topic namespaces.

10.196.3.12 void gazebo::transport::TopicManager::Init ( )

10.196.3.13 bool gazebo::transport::TopicManager::IsAdvertised (const std::string & *_topic*)

Returns true if the topic has been advertised.

Parameters

_topic	The name of the topic to check

Returns

True if the topic has been advertised

10.196.3.14 void gazebo::transport::TopicManager::PauseIncoming (bool *_pause*)

10.196.3.15 void gazebo::transport::TopicManager::ProcessNodes (bool *_onlyOut* = `false`)

Parameters

_onlyOut	True means only outbound messages on nodes will be sent. False means nodes process both outbound and inbound messages

10.196.3.16 void gazebo::transport::TopicManager::Publish (const std::string & *topic,* const google::protobuf::Message & *message,* const boost::function< void() > & *cb* = **NULL**)

Send a message.

Use a **Publisher** (p. 695) instead of calling this function directly.

Parameters

topic	Name of the topic
message	The message to send.
cb	Callback, used when the publish is completed.

10.196.3.17 void gazebo::transport::TopicManager::RegisterTopicNamespace (const std::string & *_name*)

Register a new topic namespace.

10.196.3.18 void gazebo::transport::TopicManager::RemoveNode (unsigned int *_id*)

10.196.3.19 SubscriberPtr gazebo::transport::TopicManager::Subscribe (const SubscribeOptions & *options*)

Subscribe to a topic.

10.196.3.20 void gazebo::transport::TopicManager::Unadvertise (const std::string & *topic*)

Stop advertising on a topic.

10.196.3.21 void gazebo::transport::TopicManager::Unsubscribe (const std::string & *_topic,* const NodePtr & *_sub*)

Unsubscribe from a topic.

Use a **Subscriber** (p. 827) rather than calling this function directly

10.196.3.22 PublicationPtr gazebo::transport::TopicManager::UpdatePublications (const std::string & *topic,* const std::string & *msgType*)

Update our list of advertised topics.

Returns

True if the provided params define a new publisher.

Referenced by Advertise().

The documentation for this class was generated from the following file:

- **TopicManager.hh**

10.197 gazebo::physics::TrajectoryInfo Struct Reference

`#include <Actor.hh>`

Public Attributes

- double **duration**
- double **endTime**
- unsigned int **id**
- double **startTime**
- bool **translated**
- std::string **type**

10.197.1 Member Data Documentation

10.197.1.1 double gazebo::physics::TrajectoryInfo::duration

10.197.1.2 double gazebo::physics::TrajectoryInfo::endTime

10.197.1.3 unsigned int gazebo::physics::TrajectoryInfo::id

10.197.1.4 double gazebo::physics::TrajectoryInfo::startTime

10.197.1.5 bool gazebo::physics::TrajectoryInfo::translated

10.197.1.6 std::string gazebo::physics::TrajectoryInfo::type

The documentation for this struct was generated from the following file:

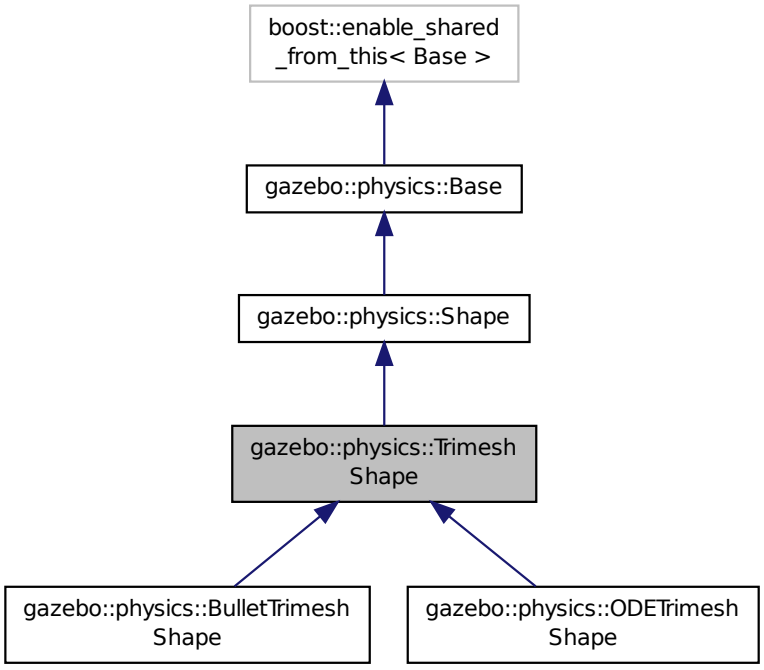
- **Actor.hh**

10.198 gazebo::physics::TrimeshShape Class Reference

Triangle mesh collision shape.

`#include <TrimeshShape.hh>`

Inheritance diagram for gazebo::physics::TrimeshShape:



Public Member Functions

- **TrimeshShape** (**CollisionPtr parent**)

  Constructor.

- virtual ~**TrimeshShape** ()

  Destructor.
- void **FillMsg** (msgs::Geometry &_msg)
- std::string **GetFilename** () const

  Get the filename of the mesh data.
- virtual double **GetMass** (double _density) const

  Get the mass of a shape.
- virtual **math::Vector3 GetSize** () const
- virtual void **Init** ()

  Init the trimesh shape.
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
- void **SetFilename** (const std::string &_filename)
- void **SetScale** (const **math::Vector3** &_scale)
- virtual void **Update** ()

  Update the object.

Protected Attributes

- const **common::Mesh** * **mesh**

Additional Inherited Members

10.198.1 Detailed Description

Triangle mesh collision shape.

10.198.2 Constructor & Destructor Documentation

10.198.2.1 gazebo::physics::TrimeshShape::TrimeshShape (CollisionPtr *parent*)

Constructor.

10.198.2.2 virtual gazebo::physics::TrimeshShape::~TrimeshShape ( ) [virtual]

Destructor.

10.198.3 Member Function Documentation

10.198.3.1 void gazebo::physics::TrimeshShape::FillMsg (msgs::Geometry & *_msg*) [virtual]

Implements **gazebo::physics::Shape** (p. 781).

10.198.3.2 std::string gazebo::physics::TrimeshShape::GetFilename ( ) const

Get the filename of the mesh data.

10.198.3.3 virtual double gazebo::physics::TrimeshShape::GetMass (double *_density*) const [virtual]

Get the mass of a shape.

Reimplemented from **gazebo::physics::Shape** (p. 781).

10.198.3.4 virtual math::Vector3 gazebo::physics::TrimeshShape::GetSize ( ) const [virtual]

10.198.3.5 virtual void gazebo::physics::TrimeshShape::Init ( ) [virtual]

Init the trimesh shape.

Implements **gazebo::physics::Shape** (p. 781).

Reimplemented in **gazebo::physics::ODETrimeshShape** (p. 634), and **gazebo::physics::BulletTrimeshShape** (p. 224).

10.198.3.6 virtual void gazebo::physics::TrimeshShape::ProcessMsg (const msgs::Geometry & *_msg*) [virtual]

Implements **gazebo::physics::Shape** (p. 782).

10.198.3.7 void gazebo::physics::TrimeshShape::SetFilename (const std::string & *_filename*)

10.198.3.8 void gazebo::physics::TrimeshShape::SetScale (const math::Vector3 & *_scale*)

10.198.3.9 virtual void gazebo::physics::TrimeshShape::Update ( ) [inline], [virtual]

Update the object.

Reimplemented from **gazebo::physics::Base** (p. 157).

Reimplemented in **gazebo::physics::ODETrimeshShape** (p. 634).

10.198.4 Member Data Documentation

10.198.4.1 const common::Mesh* gazebo::physics::TrimeshShape::mesh [protected]

The documentation for this class was generated from the following file:

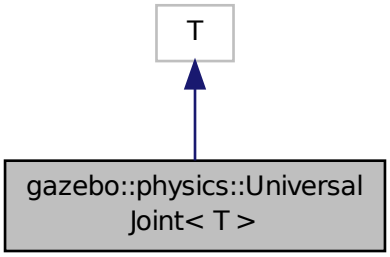
- **TrimeshShape.hh**

10.199 gazebo::physics::UniversalJoint< T > Class Template Reference

A universal joint.

`#include <UniversalJoint.hh>`

Inheritance diagram for gazebo::physics::UniversalJoint< T >:



Public Member Functions

- **UniversalJoint** (**BasePtr** _parent)

  Constructor.
- virtual ~**UniversalJoint** ()

  Destuctor.

Protected Member Functions

- virtual void **Load** (**sdf::ElementPtr** _sdf)

  *Load a **UniversalJoint** (p. 868).*

10.199.1 Detailed Description

template<class T>class gazebo::physics::UniversalJoint< T >

A universal joint.

10.199.2 Constructor & Destructor Documentation

10.199.2.1 template<class T> gazebo::physics::UniversalJoint< T >::UniversalJoint (BasePtr *_parent*) [inline]

Constructor.

10.199.2.2 template<class T> virtual gazebo::physics::UniversalJoint< T >::~UniversalJoint ( ) [inline], [virtual]

Destuctor.

10.199.3 Member Function Documentation

10.199.3.1 template<class T> virtual void gazebo::physics::UniversalJoint< T >::Load (sdf::ElementPtr _sdf) [inline],[protected],[virtual]

Load a **UniversalJoint** (p. 868).

The documentation for this class was generated from the following file:

- **UniversalJoint.hh**

10.200 urdf2gazebo::URDF2Gazebo Class Reference

`#include <parser_urdf.hh>`

Public Member Functions

- **URDF2Gazebo** ()
- ~**URDF2Gazebo** ()
- void **addKeyValue** (TiXmlElement *elem, const std::string &key, const std::string &value)

  append key value pair to the end of the xml element
- void **addTransform** (TiXmlElement *elem, const ::**gazebo::math::Pose** &transform)

  append transform (pose) to the end of the xml element
- **gazebo::math::Pose copyPose** (urdf::Pose pose)

  reduced fixed joints: utility to copy between urdf::Pose and math::Pose
- urdf::Pose **copyPose** (**gazebo::math::Pose** pose)

  reduced fixed joints: utility to copy between urdf::Pose and math::Pose
- void **createCollision** (TiXmlElement *elem, const urdf::Link *link, urdf::Collision *collision, std::string old_link_name=std::string(""))

  create SDF Collision block based on URDF
- void **createCollisions** (TiXmlElement *elem, const urdf::Link *link)

  create collision blocks from urdf collisions
- void **createGeometry** (TiXmlElement *elem, urdf::Geometry *geometry)

  create SDF geometry block based on URDF
- void **createInertial** (TiXmlElement *elem, const urdf::Link *link)

  create SDF Inertial block based on URDF
- void **createJoint** (TiXmlElement *root, const urdf::Link *link, **gazebo::math::Pose** &currentTransform)

  create SDF Joint block based on URDF
- void **createLink** (TiXmlElement *root, const urdf::Link *link, **gazebo::math::Pose** &currentTransform)

  create SDF Link block based on URDF
- void **createSDF** (TiXmlElement *root, const urdf::Link *link, const **gazebo::math::Pose** &transform)

  create SDF from URDF link
- void **createVisual** (TiXmlElement *elem, const urdf::Link *link, urdf::Visual *visual, std::string old_link_name=std::string(""))

  create SDF Visual block based on URDF
- void **createVisuals** (TiXmlElement *elem, const urdf::Link *link)

  create visual blocks from urdf visuals
- std::string **getGeometryBoundingBox** (urdf::Geometry *geometry, double *sizeVals)

- std::string **getKeyValueAsString** (TiXmlElement ∗elem)

  get value from <key value="..."> pair and return it as string
- TiXmlDocument **initModelDoc** (TiXmlDocument ∗_xmlDoc)
- TiXmlDocument **initModelFile** (std::string filename)
- TiXmlDocument **initModelString** (std::string urdf_str)
- TiXmlDocument **initModelString** (std::string urdf_str, bool _enforce_limits)
- void **insertGazeboExtensionCollision** (TiXmlElement ∗elem, std::string link_name)

  insert extensions into collision geoms
- void **insertGazeboExtensionJoint** (TiXmlElement ∗elem, std::string joint_name)

  insert extensions into joints
- void **insertGazeboExtensionLink** (TiXmlElement ∗elem, std::string link_name)

  insert extensions into links
- void **insertGazeboExtensionRobot** (TiXmlElement ∗elem)

  insert extensions into model
- void **insertGazeboExtensionVisual** (TiXmlElement ∗elem, std::string link_name)

  insert extensions into visuals
- **gazebo::math::Pose inverseTransformToParentFrame** (**gazebo::math::Pose** transform_in_link_frame, urdf::Pose parent_to_link_transform)

  reduced fixed joints: transform to parent frame
- void **listGazeboExtensions** ()

  list extensions for debugging
- void **listGazeboExtensions** (std::string reference)

  list extensions for debugging
- void **parseGazeboExtension** (TiXmlDocument &urdf_xml)

  things that do not belong in urdf but should be mapped into sdf
- urdf::Vector3 **parseVector3** (TiXmlNode ∗key, double scale=1.0)

  parser xml for vector 3
- void **printCollisionGroups** (urdf::Link ∗link)

  print collision groups for debugging purposes
- void **printMass** (urdf::Link ∗link)

  print mass for link for debugging
- void **printMass** (std::string link_name, dMass mass)

  print mass for link for debugging
- void **reduceCollisionsToParent** (urdf::Link ∗link)

  reduce fixed joints: lump collisions to parent link
- void **reduceCollisionToParent** (urdf::Link ∗link, std::string group_name, urdf::Collision ∗collision)

  reduce fixed joints: lump collision when reducing fixed joints
- void **reduceFixedJoints** (TiXmlElement ∗root, urdf::Link ∗link)

  reduce fixed joints by lumping inertial, visual and
- void **reduceGazeboExtensionContactSensorFrameReplace** (std::vector< TiXmlElement ∗ >::iterator blob_it, urdf::Link ∗link)

  reduced fixed joints: apply appropriate frame updates in urdf extensions when doing fixed joint reduction
- void **reduceGazeboExtensionFrameReplace** (**GazeboExtension** ∗ge, urdf::Link ∗link)

  reduced fixed joints: apply appropriate frame updates in urdf extensions when doing fixed joint reduction
- void **reduceGazeboExtensionGripperFrameReplace** (std::vector< TiXmlElement ∗ >::iterator blob_it, urdf::Link ∗link)

  reduced fixed joints: apply appropriate frame updates in gripper inside urdf extensions when doing fixed joint reduction

- void **reduceGazeboExtensionJointFrameReplace** (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link *link)

  reduced fixed joints: apply appropriate frame updates in joint inside urdf extensions when doing fixed joint reduction
- void **reduceGazeboExtensionPluginFrameReplace** (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link *link, std::string plugin_name, std::string element_name, **gazebo::math::Pose** reduction_transform)

  reduced fixed joints: apply appropriate frame updates in plugins inside urdf extensions when doing fixed joint reduction
- void **reduceGazeboExtensionProjectorFrameReplace** (std::vector< TiXmlElement * >::iterator blob_it, urdf::Link *link)

  reduced fixed joints: apply appropriate frame updates in projector inside urdf extensions when doing fixed joint reduction
- void **reduceGazeboExtensionProjectorTransformReduction** (std::vector< TiXmlElement * >::iterator blob_it, **gazebo::math::Pose** reduction_transform)

  reduced fixed joints: apply transform reduction for projectors in extensions when doing fixed joint reduction
- void **reduceGazeboExtensionSensorTransformReduction** (std::vector< TiXmlElement * >::iterator blob_it, **gazebo::math::Pose** reduction_transform)

  reduced fixed joints: apply transform reduction for ray sensors in extensions when doing fixed joint reduction
- void **reduceGazeboExtensionsTransformReduction** (**GazeboExtension** *ge)

  reduced fixed joints: apply transform reduction to extensions when doing fixed joint reduction
- void **reduceGazeboExtensionToParent** (urdf::Link *link)

  reduced fixed joints: apply appropriate updates to urdf extensions when doing fixed joint reduction
- void **reduceInertialToParent** (urdf::Link *link)

  reduce fixed joints: lump inertial to parent link
- void **reduceJointsToParent** (urdf::Link *link)

  reduce fixed joints: lump joints to parent link
- void **reduceVisualsToParent** (urdf::Link *link)

  reduce fixed joints: lump visuals to parent link
- void **reduceVisualToParent** (urdf::Link *link, std::string group_name, urdf::Visual *visual)

  reduce fixed joints: lump visuals when reducing fixed joints
- urdf::Pose **transformToParentFrame** (urdf::Pose transform_in_link_frame, urdf::Pose parent_to_link_transform)

  reduced fixed joints: transform to parent frame
- **gazebo::math::Pose transformToParentFrame** (**gazebo::math::Pose** transform_in_link_frame, urdf::Pose parent_to_link_transform)

  reduced fixed joints: transform to parent frame
- **gazebo::math::Pose transformToParentFrame** (**gazebo::math::Pose** transform_in_link_frame, **gazebo::math::Pose** parent_to_link_transform)

  reduced fixed joints: transform to parent frame
- std::string **values2str** (unsigned int count, const double *values)

  convert values to string
- std::string **vector32str** (const urdf::Vector3 vector)

  convert Vector3 to string

Public Attributes

- std::map< std::string, std::vector< **GazeboExtension** * > > **gazebo_extensions_**

10.200.1 Constructor & Destructor Documentation

10.200.1.1 urdf2gazebo::URDF2Gazebo::URDF2Gazebo ( )

10.200.1.2 urdf2gazebo::URDF2Gazebo::~URDF2Gazebo ( )

10.200.2 Member Function Documentation

10.200.2.1 void urdf2gazebo::URDF2Gazebo::addKeyValue (TiXmlElement * *elem,* const std::string & *key,* const std::string & *value*)

append key value pair to the end of the xml element

10.200.2.2 void urdf2gazebo::URDF2Gazebo::addTransform (TiXmlElement * *elem,* const ::gazebo::math::Pose & *transform*)

append transform (pose) to the end of the xml element

10.200.2.3 gazebo::math::Pose urdf2gazebo::URDF2Gazebo::copyPose (urdf::Pose *pose*)

reduced fixed joints: utility to copy between urdf::Pose and math::Pose

10.200.2.4 urdf::Pose urdf2gazebo::URDF2Gazebo::copyPose (gazebo::math::Pose *pose*)

reduced fixed joints: utility to copy between urdf::Pose and math::Pose

10.200.2.5 void urdf2gazebo::URDF2Gazebo::createCollision (TiXmlElement * *elem,* const urdf::Link * *link,* urdf::Collision * *collision,* std::string *old_link_name* = `std::string("")`)

create SDF Collision block based on URDF

10.200.2.6 void urdf2gazebo::URDF2Gazebo::createCollisions (TiXmlElement * *elem,* const urdf::Link * *link*)

create collision blocks from urdf collisions

10.200.2.7 void urdf2gazebo::URDF2Gazebo::createGeometry (TiXmlElement * *elem,* urdf::Geometry * *geometry*)

create SDF geometry block based on URDF

10.200.2.8 void urdf2gazebo::URDF2Gazebo::createInertial (TiXmlElement * *elem,* const urdf::Link * *link*)

create SDF Inertial block based on URDF

10.200.2.9 void urdf2gazebo::URDF2Gazebo::createJoint (TiXmlElement * *root,* const urdf::Link * *link,* gazebo::math::Pose & *currentTransform*)

create SDF Joint block based on URDF

10.200.2.10 void urdf2gazebo::URDF2Gazebo::createLink (TiXmlElement * *root,* const urdf::Link * *link,* gazebo::math::Pose & *currentTransform*)

create SDF Link block based on URDF

10.200.2.11 void urdf2gazebo::URDF2Gazebo::createSDF (TiXmlElement * *root,* const urdf::Link * *link,* const gazebo::math::Pose & *transform*)

create SDF from URDF link

10.200.2.12 void urdf2gazebo::URDF2Gazebo::createVisual (TiXmlElement * *elem,* const urdf::Link * *link,* urdf::Visual * *visual,* std::string *old_link_name* = `std::string("")`)

create SDF Visual block based on URDF

10.200.2.13 void urdf2gazebo::URDF2Gazebo::createVisuals (TiXmlElement * *elem,* const urdf::Link * *link*)

create visual blocks from urdf visuals

10.200.2.14 std::string urdf2gazebo::URDF2Gazebo::getGeometryBoundingBox (urdf::Geometry * *geometry,* double * *sizeVals*)

10.200.2.15 std::string urdf2gazebo::URDF2Gazebo::getKeyValueAsString (TiXmlElement * *elem*)

get value from <key value="..."> pair and return it as string

10.200.2.16 TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelDoc (TiXmlDocument * *_xmlDoc*)

10.200.2.17 TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelFile (std::string *filename*)

10.200.2.18 TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelString (std::string *urdf_str*)

10.200.2.19 TiXmlDocument urdf2gazebo::URDF2Gazebo::initModelString (std::string *urdf_str,* bool *_enforce_limits*)

10.200.2.20 void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionCollision (TiXmlElement * *elem,* std::string *link_name*)

insert extensions into collision geoms

10.200.2.21 void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionJoint (TiXmlElement * *elem,* std::string *joint_name*)

insert extensions into joints

10.200.2.22 void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionLink (TiXmlElement * *elem,* std::string *link_name*)

insert extensions into links

10.200.2.23 void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionRobot (TiXmlElement * *elem*)

insert extensions into model

10.200.2.24 void urdf2gazebo::URDF2Gazebo::insertGazeboExtensionVisual (TiXmlElement * *elem,* std::string *link_name*)

insert extensions into visuals

10.200.2.25 gazebo::math::Pose urdf2gazebo::URDF2Gazebo::inverseTransformToParentFrame (gazebo::math::Pose *transform_in_link_frame,* urdf::Pose *parent_to_link_transform*)

reduced fixed joints: transform to parent frame

10.200.2.26 void urdf2gazebo::URDF2Gazebo::listGazeboExtensions ( )

list extensions for debugging

10.200.2.27 void urdf2gazebo::URDF2Gazebo::listGazeboExtensions (std::string *reference*)

list extensions for debugging

10.200.2.28 void urdf2gazebo::URDF2Gazebo::parseGazeboExtension (TiXmlDocument & *urdf_xml*)

things that do not belong in urdf but should be mapped into sdf

Todo : do this using sdf definitions, not hard coded stuff

10.200.2.29 urdf::Vector3 urdf2gazebo::URDF2Gazebo::parseVector3 (TiXmlNode * *key,* double *scale* = 1.0)

parser xml for vector 3

10.200.2.30 void urdf2gazebo::URDF2Gazebo::printCollisionGroups (urdf::Link * *link*)

print collision groups for debugging purposes

10.200.2.31 void urdf2gazebo::URDF2Gazebo::printMass (urdf::Link * *link*)

print mass for link for debugging

10.200.2.32 void urdf2gazebo::URDF2Gazebo::printMass (std::string *link_name,* dMass *mass*)

print mass for link for debugging

10.200.2.33 void urdf2gazebo::URDF2Gazebo::reduceCollisionsToParent (urdf::Link * *link*)

reduce fixed joints: lump collisions to parent link

10.200.2.34 void urdf2gazebo::URDF2Gazebo::reduceCollisionToParent (urdf::Link * *link*, std::string *group_name*, urdf::Collision * *collision*)

reduce fixed joints: lump collision when reducing fixed joints

10.200.2.35 void urdf2gazebo::URDF2Gazebo::reduceFixedJoints (TiXmlElement * *root*, urdf::Link * *link*)

reduce fixed joints by lumping inertial, visual and

10.200.2.36 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionContactSensorFrameReplace (std::vector< TiXmlElement * >::iterator *blob_it*, urdf::Link * *link*)

reduced fixed joints: apply appropriate frame updates in urdf extensions when doing fixed joint reduction

10.200.2.37 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionFrameReplace (GazeboExtension * *ge*, urdf::Link * *link*)

reduced fixed joints: apply appropriate frame updates in urdf extensions when doing fixed joint reduction

10.200.2.38 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionGripperFrameReplace (std::vector< TiXmlElement * >::iterator *blob_it*, urdf::Link * *link*)

reduced fixed joints: apply appropriate frame updates in gripper inside urdf extensions when doing fixed joint reduction

10.200.2.39 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionJointFrameReplace (std::vector< TiXmlElement * >::iterator *blob_it*, urdf::Link * *link*)

reduced fixed joints: apply appropriate frame updates in joint inside urdf extensions when doing fixed joint reduction

10.200.2.40 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionPluginFrameReplace (std::vector< TiXmlElement * >::iterator *blob_it*, urdf::Link * *link*, std::string *plugin_name*, std::string *element_name*, gazebo::math::Pose *reduction_transform*)

reduced fixed joints: apply appropriate frame updates in plugins inside urdf extensions when doing fixed joint reduction

10.200.2.41 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionProjectorFrameReplace (std::vector< TiXmlElement * >::iterator *blob_it*, urdf::Link * *link*)

reduced fixed joints: apply appropriate frame updates in projector inside urdf extensions when doing fixed joint reduction

10.200.2.42 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionProjectorTransformReduction (std::vector< TiXmlElement * >::iterator *blob_it*, gazebo::math::Pose *reduction_transform*)

reduced fixed joints: apply transform reduction for projectors in extensions when doing fixed joint reduction

10.200.2.43 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionSensorTransformReduction (std::vector< TiXmlElement * >::iterator *blob_it,* gazebo::math::Pose *reduction_transform*)

reduced fixed joints: apply transform reduction for ray sensors in extensions when doing fixed joint reduction

10.200.2.44 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionsTransformReduction (GazeboExtension * *ge*)

reduced fixed joints: apply transform reduction to extensions when doing fixed joint reduction

10.200.2.45 void urdf2gazebo::URDF2Gazebo::reduceGazeboExtensionToParent (urdf::Link * *link*)

reduced fixed joints: apply appropriate updates to urdf extensions when doing fixed joint reduction

10.200.2.46 void urdf2gazebo::URDF2Gazebo::reduceInertialToParent (urdf::Link * *link*)

reduce fixed joints: lump inertial to parent link

10.200.2.47 void urdf2gazebo::URDF2Gazebo::reduceJointsToParent (urdf::Link * *link*)

reduce fixed joints: lump joints to parent link

10.200.2.48 void urdf2gazebo::URDF2Gazebo::reduceVisualsToParent (urdf::Link * *link*)

reduce fixed joints: lump visuals to parent link

10.200.2.49 void urdf2gazebo::URDF2Gazebo::reduceVisualToParent (urdf::Link * *link,* std::string *group_name,* urdf::Visual * *visual*)

reduce fixed joints: lump visuals when reducing fixed joints

10.200.2.50 urdf::Pose urdf2gazebo::URDF2Gazebo::transformToParentFrame (urdf::Pose *transform_in_link_frame,* urdf::Pose *parent_to_link_transform*)

reduced fixed joints: transform to parent frame

10.200.2.51 gazebo::math::Pose urdf2gazebo::URDF2Gazebo::transformToParentFrame (gazebo::math::Pose *transform_in_link_frame,* urdf::Pose *parent_to_link_transform*)

reduced fixed joints: transform to parent frame

10.200.2.52 gazebo::math::Pose urdf2gazebo::URDF2Gazebo::transformToParentFrame (gazebo::math::Pose *transform_in_link_frame,* gazebo::math::Pose *parent_to_link_transform*)

reduced fixed joints: transform to parent frame

10.200.2.53 std::string urdf2gazebo::URDF2Gazebo::values2str (unsigned int *count,* const double * *values*)

convert values to string

10.200.2.54 std::string urdf2gazebo::URDF2Gazebo::vector32str (const urdf::Vector3 *vector*)

convert Vector3 to string

10.200.3 Member Data Documentation

10.200.3.1 std::map<std::string, std::vector<GazeboExtension*> > urdf2gazebo::URDF2Gazebo::gazebo_extensions_

The documentation for this class was generated from the following file:

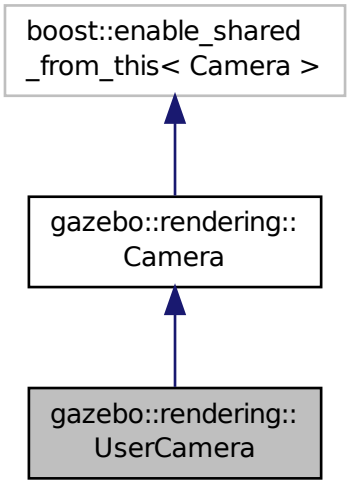
- **parser_urdf.hh**

10.201 gazebo::rendering::UserCamera Class Reference

A camera used for user visualization of a scene.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::UserCamera:



Public Member Functions

- **UserCamera** (const std::string &_name, **Scene** *_scene)

Constructor.

- virtual ~**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.
- float **GetTriangleCount** () const

  Get the triangle count.
- **VisualPtr GetVisual** (const **math::Vector2i** &mousePos, std::string &mod)

  Get an entity at a pixel location using a camera.
- **VisualPtr GetVisual** (const **math::Vector2i** &_mousePos) const

  Get a visual at a mouse position.
- void **HandleKeyPressEvent** (const std::string &_key)

  Handle a key press.
- void **HandleKeyReleaseEvent** (const std::string &_key)

  Handle a key release.
- void **HandleMouseEvent** (const **common::MouseEvent** &_evt)

  Handle a mouse event.
- void **Init** ()

  Initialize.
- void **Load** (**sdf::ElementPtr** _sdf)

  Load the user camera.
- void **Load** ()

  Generic load function.
- virtual bool **MoveToPosition** (const **math::Pose** &_pose, double _time)

  Move the camera to a position.
- void **MoveToVisual** (**VisualPtr** _visual)

  Move the camera to focus on a visual.
- void **MoveToVisual** (const std::string &_visualName)

  Move the camera to focus on a visual.
- virtual void **PostRender** ()

  Post render.
- void **Resize** (unsigned int _w, unsigned int _h)

  Resize the camera.
- void **SetFocalPoint** (const **math::Vector3** &_pt)

  Set the point the camera should orbit around.
- virtual void **SetRenderTarget** (Ogre::RenderTarget *_target)

  Set to true to enable rendering.
- void **SetViewController** (const std::string &_type)

  Set view controller.
- void **SetViewController** (const std::string &_type, const **math::Vector3** &_pos)

  Set view controller.

- void **SetViewportDimensions** (float _x, float _y, float _w, float _h)

  Set the dimensions of the viewport.
- virtual void **SetWorldPose** (const **math::Pose** &_pose)

  Set the pose in the world coordinate frame.
- virtual void **Update** ()

  Render the camera.

Protected Member Functions

- virtual bool **AttachToVisualImpl** (**VisualPtr** _visual, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  Set the camera to be attached to a visual.
- virtual bool **TrackVisualImpl** (**VisualPtr** _visual)

  Set the camera to track a scene node.

Additional Inherited Members

10.201.1 Detailed Description

A camera used for user visualization of a scene.

10.201.2 Constructor & Destructor Documentation

10.201.2.1 gazebo::rendering::UserCamera::UserCamera (const std::string & *_name,* Scene * *_scene*)

Constructor.

Parameters

in	*_name*	Name of the camera.
in	*_scene*	**Scene** (p. 746) to put the camera in.

10.201.2.2 virtual gazebo::rendering::UserCamera::~UserCamera ( ) [virtual]

Destructor.

10.201.3 Member Function Documentation

10.201.3.1 virtual bool gazebo::rendering::UserCamera::AttachToVisualImpl (VisualPtr *_visual,* bool *_inheritOrientation,* double *_minDist* = 0, double *_maxDist* = 0) [protected], [virtual]

Set the camera to be attached to a visual.

This causes the camera to move in relation to the specified visual.

Parameters

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. 239).

10.201.3.2 void gazebo::rendering::UserCamera::EnableViewController (bool *_value*) const

Set whether the view controller is enabled.

The view controller is used to handle user camera movements.

Parameters

in	*_value*	True to enable viewcontroller, False to disable.

10.201.3.3 void gazebo::rendering::UserCamera::Fini ( )

Finialize.

10.201.3.4 float gazebo::rendering::UserCamera::GetAvgFPS ( ) const

Get the average frames per second.

Returns

The average rendering frames per second

10.201.3.5 GUIOverlay* gazebo::rendering::UserCamera::GetGUIOverlay ( )

Get the GUI overlay.

An overlay allows you to draw 2D elements on the viewport.

Returns

Pointer to the **GUIOverlay** (p. 394).

10.201.3.6 float gazebo::rendering::UserCamera::GetTriangleCount ( ) const

Get the triangle count.

Returns

The number of triangles currently being rendered.

10.201.3.7 VisualPtr gazebo::rendering::UserCamera::GetVisual (const math::Vector2i & *mousePos,* std::string & *mod*)

Get an entity at a pixel location using a camera.

Used for mouse picking.

Parameters

in	*_mousePos*	The position of the mouse in screen coordinates
out	*_mod*	Used for object manipulation

Returns

The selected entity, or NULL

10.201.3.8 VisualPtr gazebo::rendering::UserCamera::GetVisual (const math::Vector2i & *_mousePos*) const

Get a visual at a mouse position.

Parameters

in	*_mousePos*	2D position of the mouse in pixels.

10.201.3.9 void gazebo::rendering::UserCamera::HandleKeyPressEvent (const std::string & *_key*)

Handle a key press.

Parameters

in	*_key*	The key pressed.

10.201.3.10 void gazebo::rendering::UserCamera::HandleKeyReleaseEvent (const std::string & *_key*)

Handle a key release.

Parameters

in	*_key*	The key released.

10.201.3.11 void gazebo::rendering::UserCamera::HandleMouseEvent (const common::MouseEvent & *_evt*)

Handle a mouse event.

Parameters

in	*_evt*	The mouse event.

10.201.3.12 void gazebo::rendering::UserCamera::Init ( )

Initialize.

10.201.3.13 void gazebo::rendering::UserCamera::Load (sdf::ElementPtr *_sdf*)

Load the user camera.

Parameters

in	*_sdf*	Parameters for the camera.

10.201.3.14 void gazebo::rendering::UserCamera::Load ( )

Generic load function.

10.201.3.15 virtual bool gazebo::rendering::UserCamera::MoveToPosition (const math::Pose & *_pose*, double *_time*) [virtual]

Move the camera to a position.

This is an animated motion

Parameters

in	*_pose*	End position of the camera
in	*_time*	Duration of the camera's movement

Reimplemented from **gazebo::rendering::Camera** (p. 248).

10.201.3.16 void gazebo::rendering::UserCamera::MoveToVisual (VisualPtr *_visual*)

Move the camera to focus on a visual.

Parameters

in	*_visual*	**Visual** (p. 931) to move the camera to.

10.201.3.17 void gazebo::rendering::UserCamera::MoveToVisual (const std::string & *_visualName*)

Move the camera to focus on a visual.

Parameters

in	*_visualName*	Name of the visual to move the camera to.

10.201.3.18 virtual void gazebo::rendering::UserCamera::PostRender ( ) [virtual]

Post render.

Reimplemented from **gazebo::rendering::Camera** (p. 248).

10.201.3.19 void gazebo::rendering::UserCamera::Resize (unsigned int *_w,* unsigned int *_h*)

Resize the camera.

Parameters

in	*_w*	Width of the camera image.
in	*_h*	Height of the camera image.

10.201.3.20 void gazebo::rendering::UserCamera::SetFocalPoint (const math::Vector3 & *_pt*)

Set the point the camera should orbit around.

Parameters

in	*_pt*	The focal point

10.201.3.21 virtual void gazebo::rendering::UserCamera::SetRenderTarget (Ogre::RenderTarget * *_target*) [virtual]

Set to true to enable rendering.

Use this only if you really know what you're doing.

Parameters

in	*_target*	The new rendering target.

Reimplemented from **gazebo::rendering::Camera** (p. 251).

10.201.3.22 void gazebo::rendering::UserCamera::SetViewController (const std::string & *_type*)

Set view controller.

Parameters

in	*_type*	The type of view controller: "orbit", "fps"

10.201.3.23 void gazebo::rendering::UserCamera::SetViewController (const std::string & *_type,* const math::Vector3 & *_pos*)

Set view controller.

Parameters

in	*_type*	The type of view controller: "orbit", "fps"
in	*_pos*	The initial pose of the camera.

10.201.3.24 void gazebo::rendering::UserCamera::SetViewportDimensions (float _x, float _y, float _w, float _h)

Set the dimensions of the viewport.

Parameters

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.

10.201.3.25 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. 252).

10.201.3.26 virtual bool gazebo::rendering::UserCamera::TrackVisualImpl (VisualPtr _visual) [protected], [virtual]

Set the camera to track a scene node.

Tracking just causes the camera to rotate to follow the visual.

Parameters

in	*_visual*	**Visual** (p. 931) to track.

Returns

True if the camera is now tracking the visual.

Reimplemented from **gazebo::rendering::Camera** (p. 253).

10.201.3.27 virtual void gazebo::rendering::UserCamera::Update ( ) [virtual]

Render the camera.

Reimplemented from **gazebo::rendering::Camera** (p. 254).

The documentation for this class was generated from the following file:

- **UserCamera.hh**

10.202 gazebo::math::Vector2d Class Reference

Generic double x, y vector.

`#include <Vector2d.hh>`

Public Member Functions

- **Vector2d** ()

  Constructor.
- **Vector2d** (const double &_x, const double &_y)

  Constructor.
- **Vector2d** (const **Vector2d** &_v)

  Copy constructor.
- virtual ∼**Vector2d** ()

  Destructor.
- **Vector2d Cross** (const **Vector2d** &_v) const

  Return the cross product of this vector and _v.
- double **Distance** (const **Vector2d** &_pt) const

  Calc distance to the given point.
- bool **IsFinite** () const

  See if a point is finite (e.g., not nan)
- void **Normalize** ()

  Normalize the vector length.
- bool **operator!=** (const **Vector2d** &_v) const

  Not equal to operator.
- const **Vector2d operator**∗ (const **Vector2d** &_v) const

  Multiplication operators.
- const **Vector2d operator**∗ (double _v) const

  Multiplication operators.
- const **Vector2d** & **operator**∗**=** (const **Vector2d** &_v)

  Multiplication assignment operator.
- const **Vector2d** & **operator**∗**=** (double _v)

  Multiplication assignment operator.
- **Vector2d operator+** (const **Vector2d** &_v) const

  Addition operator.
- const **Vector2d** & **operator+=** (const **Vector2d** &_v)

  Addition assignment operator.
- **Vector2d operator-** (const **Vector2d** &_v) const

  Subtraction operator.
- const **Vector2d** & **operator-=** (const **Vector2d** &_v)

  Subtraction assignment operator.
- const **Vector2d operator**/ (const **Vector2d** &_v) const

  Division operator.
- const **Vector2d operator**/ (double _v) const

  Division operator.
- const **Vector2d** & **operator**/**=** (const **Vector2d** &_v)

  Division operator.
- const **Vector2d** & **operator**/**=** (double _v)

  Division operator.
- **Vector2d** & **operator=** (const **Vector2d** &_v)

  Assignment operator.
- const **Vector2d** & **operator=** (double _v)

Assignment operator.

- bool **operator==** (const **Vector2d** &_v) const

  Equal to operator.

- double **operator[]** (unsigned int _index) const

  Array subscript operator.

- void **Set** (double _x, double _y)

  Set the contents of the vector.

Public Attributes

- double **x**

  x data

- double **y**

  y data

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector2d** &_pt)

  Stream extraction operator.

- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector2d** &_pt)

  Stream extraction operator.

10.202.1 Detailed Description

Generic double x, y vector.

10.202.2 Constructor & Destructor Documentation

10.202.2.1 gazebo::math::Vector2d::Vector2d ( )

Constructor.

10.202.2.2 gazebo::math::Vector2d::Vector2d (const double & *_x*, const double & *_y*)

Constructor.

Parameters

in	*_x*	value along x
in	*_y*	value along y

10.202.2.3 gazebo::math::Vector2d::Vector2d (const Vector2d & *_v*)

Copy constructor.

Parameters

in	*_v*	the value

10.202.2.4 virtual gazebo::math::Vector2d::~Vector2d ( ) [virtual]

Destructor.

10.202.3 Member Function Documentation

10.202.3.1 Vector2d gazebo::math::Vector2d::Cross (const Vector2d & *_v*) const

Return the cross product of this vector and _v.

Parameters

in	*_v*	the vector

Returns

the cross product

10.202.3.2 double gazebo::math::Vector2d::Distance (const Vector2d & *_pt*) const

Calc distance to the given point.

Returns

the distance

10.202.3.3 bool gazebo::math::Vector2d::IsFinite ( ) const

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.202.3.4 void gazebo::math::Vector2d::Normalize ( )

Normalize the vector length.

10.202.3.5 bool gazebo::math::Vector2d::operator!= (const Vector2d & *_v*) const

Not equal to operator.

Returns

true if elements are of diffent values (tolerence 1e-6)

10.202.3.6 const Vector2d gazebo::math::Vector2d::operator∗ (const Vector2d & _v) const

Multiplication operators.

Parameters

in	_v	the vector

Returns

the result

10.202.3.7 const Vector2d gazebo::math::Vector2d::operator∗ (double _v) const

Multiplication operators.

Parameters

in	_v	the scaling factor

Returns

a scaled vector

10.202.3.8 const Vector2d& gazebo::math::Vector2d::operator∗= (const Vector2d & _v)

Multiplication assignment operator.

Remarks

this is an element wise multiplication

Parameters

in	_v	the vector

Returns

this

10.202.3.9 const Vector2d& gazebo::math::Vector2d::operator∗= (double _v)

Multiplication assignment operator.

Parameters

in	_v	the scaling factor

Returns

a scaled vector

10.202.3.10 Vector2d gazebo::math::Vector2d::operator+ (const Vector2d & _v) const

Addition operator.

Parameters

in	_v	vector to add

Returns

sum vector

10.202.3.11 const Vector2d& gazebo::math::Vector2d::operator+= (const Vector2d & _v)

Addition assignment operator.

Parameters

in	_v	the vector to add

10.202.3.12 Vector2d gazebo::math::Vector2d::operator- (const Vector2d & _v) const

Subtraction operator.

Parameters

in	_v	the vector to substract

Returns

the subtracted vector

10.202.3.13 const Vector2d& gazebo::math::Vector2d::operator-= (const Vector2d & _v)

Subtraction assignment operator.

Parameters

in	_v	the vector to substract

Returns

this

10.202.3.14 const Vector2d gazebo::math::Vector2d::operator/ (const Vector2d & _v) const

Division operator.

Remarks

this is an element wise division

Parameters

in	_v	a vector

Returns

a result

10.202.3.15 const Vector2d gazebo::math::Vector2d::operator/ (double _v) const

Division operator.

Parameters

in	_v	the value

Returns

a vector

10.202.3.16 const Vector2d& gazebo::math::Vector2d::operator/= (const Vector2d & _v)

Division operator.

Remarks

this is an element wise division

Parameters

in	_v	a vector

Returns

this

10.202.3.17 const Vector2d& gazebo::math::Vector2d::operator/= (double _v)

Division operator.

Parameters

in	_v	the divisor

Returns

a vector

10.202.3.18 Vector2d& gazebo::math::Vector2d::operator= (const Vector2d & _v)

Assignment operator.

Parameters

in	_v	a value for x and y element

Returns

this

10.202.3.19 const Vector2d& gazebo::math::Vector2d::operator= (double _v)

Assignment operator.

Parameters

in	_v	the value for x and y element

Returns

this

10.202.3.20 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)

10.202.3.21 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

10.202.3.22 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

10.202.4 Friends And Related Function Documentation

10.202.4.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Vector2d & *_pt*) [friend]

Stream extraction operator.

Parameters

in	*_out*	output stream
in	*_pt*	**Vector2d** (p. 885) to output

Returns

The stream

10.202.4.2 std::istream& operator>> (std::istream & *_in*, gazebo::math::Vector2d & *_pt*) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_pt*	**Vector3** (p. 902) to read values into

Returns

The stream

10.202.5 Member Data Documentation

10.202.5.1 double gazebo::math::Vector2d::x

x data

10.202.5.2 double gazebo::math::Vector2d::y

y data

The documentation for this class was generated from the following file:

- **Vector2d.hh**

10.203 gazebo::math::Vector2i Class Reference

Generic integer x, y vector.

`#include <Vector2i.hh>`

Public Member Functions

- **Vector2i** ()
  Constructor.
- **Vector2i** (const int &_x, const int &_y)
  Constructor.
- **Vector2i** (const **Vector2i** &_pt)
  Copy onstructor.
- virtual ∼**Vector2i** ()
  Destructor.
- **Vector2i Cross** (const **Vector2i** &_pt) const
  Return the cross product of this vector and pt.
- int **Distance** (const **Vector2i** &_pt) const
  Calc distance to the given point.
- bool **IsFinite** () const
  See if a point is finite (e.g., not nan)
- void **Normalize** ()
  Normalize the vector length.
- bool **operator!=** (const **Vector2i** &_v) const
  Equality operators.
- const **Vector2i operator**∗ (const **Vector2i** &_v) const
  Multiplication operator.
- const **Vector2i operator**∗ (int _v) const
  Multiplication operator.
- const **Vector2i** & **operator**∗**=** (const **Vector2i** &_v)
  Multiplication operators.
- const **Vector2i** & **operator**∗**=** (int _v)
  Multiplication operator.
- **Vector2i operator+** (const **Vector2i** &_v) const
  Addition operator.
- const **Vector2i** & **operator+=** (const **Vector2i** &_v)
  Addition assignment operator.
- **Vector2i operator-** (const **Vector2i** &_v) const
  Subtraction operator.

- const **Vector2i** & **operator-=** (const **Vector2i** &_v)

  Subtraction operators.
- const **Vector2i operator**/ (const **Vector2i** &_v) const

  Division operator.
- const **Vector2i operator**/ (int _v) const

  Division operator.
- const **Vector2i** & **operator**/= (const **Vector2i** &_v)

  Division operator.
- const **Vector2i** & **operator**/= (int _v)

  Division operator.
- **Vector2i** & **operator=** (const **Vector2i** &_v)

  Assignment operator.
- const **Vector2i** & **operator=** (int _value)

  Assignment operator.
- bool **operator==** (const **Vector2i** &_v) const

  Equality operator.
- int **operator[]** (unsigned int _index) const

  Array subscript operator.
- void **Set** (int _x, int _y)

  Set the contents of the vector.

Public Attributes

- int **x**

  x data
- int **y**

  y data

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector2i** &_pt)

  Stream insertion operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector2i** &_pt)

  Stream extraction operator.

10.203.1 Detailed Description

Generic integer x, y vector.

10.203.2 Constructor & Destructor Documentation

10.203.2.1 gazebo::math::Vector2i::Vector2i ( )

Constructor.

10.203.2.2 gazebo::math::Vector2i::Vector2i (const int & _x, const int & _y)

Constructor.

Parameters

in	*_x*	value along x
in	*_y*	value along y

10.203.2.3 gazebo::math::Vector2i::Vector2i (const Vector2i & _pt)

Copy onstructor.

Parameters

in	*_pt*	a point

10.203.2.4 virtual gazebo::math::Vector2i::~Vector2i ( ) [virtual]

Destructor.

10.203.3 Member Function Documentation

10.203.3.1 Vector2i gazebo::math::Vector2i::Cross (const Vector2i & _pt) const

Return the cross product of this vector and pt.

Parameters

in	*the*	other vector

Returns

the product

10.203.3.2 int gazebo::math::Vector2i::Distance (const Vector2i & _pt) const

Calc distance to the given point.

Parameters

in	*_pt*	a point

Returns

the distance

10.203.3.3 bool gazebo::math::Vector2i::IsFinite ( ) const

See if a point is finite (e.g., not nan)

Returns

the result

10.203.3.4 void gazebo::math::Vector2i::Normalize ( )

Normalize the vector length.

10.203.3.5 bool gazebo::math::Vector2i::operator!= (const Vector2i & _v) const

Equality operators.

Parameters

_v	the vector to compare with

Returns

true if component have different values, false otherwise

10.203.3.6 const Vector2i gazebo::math::Vector2i::operator* (const Vector2i & _v) const

Multiplication operator.

Remarks

this is an element wise multiplication

Parameters

in	_v	the vector

Returns

the result

10.203.3.7 const Vector2i gazebo::math::Vector2i::operator* (int _v) const

Multiplication operator.

Parameters

in	_v	the scaling factor

Returns

the result

10.203.3.8 const Vector2i& gazebo::math::Vector2i::operator*= (const Vector2i & _v)

Multiplication operators.

Remarks

this is an element wise multiplication

Parameters

in	_v	the vector

Returns

this

10.203.3.9 const Vector2i& gazebo::math::Vector2i::operator*= (int _v)

Multiplication operator.

Parameters

in	_v	scaling factor

Returns

this

10.203.3.10 Vector2i gazebo::math::Vector2i::operator+ (const Vector2i & _v) const

Addition operator.

Parameters

in	_v	the vector to add

Returns

the sum vector

10.203.3.11 const Vector2i& gazebo::math::Vector2i::operator+= (const Vector2i & _v)

Addition assignment operator.

Parameters

in	_v	the vector to add

Returns

this

10.203.3.12 Vector2i gazebo::math::Vector2i::operator- (const Vector2i & _v) const

Subtraction operator.

Parameters

in	_v	the vector to substract

Returns

the result vector

10.203.3.13 const Vector2i& gazebo::math::Vector2i::operator-= (const Vector2i & _v)

Subtraction operators.

Parameters

in	_v	the vector to substract

Returns

this

10.203.3.14 const Vector2i gazebo::math::Vector2i::operator/ (const Vector2i & _v) const

Division operator.

Remarks

this is an element wise division.

Parameters

in	_v	the vector to divide

Returns

the result

10.203.3.15 const Vector2i gazebo::math::Vector2i::operator/ (int _v) const

Division operator.

Remarks

this is an element wise division.

Parameters

in	_v	the vector to divide

Returns

the result

10.203.3.16 const Vector2i& gazebo::math::Vector2i::operator/= (const Vector2i & _v)

Division operator.

Remarks

this is an element wise division.

Parameters

in	_v	the vector to divide

Returns

this

10.203.3.17 const Vector2i& gazebo::math::Vector2i::operator/= (int _v)

Division operator.

Remarks

this is an element wise division.

Parameters

in	_v	the vector to divide

Returns

this

10.203.3.18 Vector2i& gazebo::math::Vector2i::operator= (const Vector2i & _v)

Assignment operator.

Parameters

in	_v	the value

Returns

this

10.203.3.19 const Vector2i& gazebo::math::Vector2i::operator= (int _value)

Assignment operator.

Parameters

in	_value	the value for x and y

Returns

this

10.203.3.20 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

10.203.3.21 int gazebo::math::Vector2i::operator[] (unsigned int _index) const

Array subscript operator.

Parameters

in	_index	the array index

10.203.3.22 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

10.203.4 Friends And Related Function Documentation

10.203.4.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::math::Vector2i & *_pt*) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*pt*	**Vector2i** (p. 894) to output

Returns

the stream

10.203.4.2 std::istream& operator>> (std::istream & *_in,* gazebo::math::Vector2i & *_pt*) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_pt*	**Vector3** (p. 902) to read values into

Returns

The stream

10.203.5 Member Data Documentation

10.203.5.1 int gazebo::math::Vector2i::x

x data

10.203.5.2 int gazebo::math::Vector2i::y

y data

The documentation for this class was generated from the following file:

- **Vector2i.hh**

10.204 gazebo::math::Vector3 Class Reference

The **Vector3** (p. 902) class represents the generic vector containing 3 elements.

`#include <Vector3.hh>`

Public Member Functions

- **Vector3** ()

  Constructor.
- **Vector3** (const double &_x, const double &_y, const double &_z)

  Constructor.
- **Vector3** (const **Vector3** &_v)

  Copy constructor.
- virtual ~**Vector3** ()

  Destructor.
- void **Correct** ()

  Corrects any nan values.
- **Vector3 Cross** (const **Vector3** &_pt) const

  Return the cross product of this vector and pt.
- double **Distance** (const **Vector3** &_pt) const

  Calc distance to the given point.
- double **Distance** (double _x, double _y, double _z) const

  Calc distance to the given point.
- double **Dot** (const **Vector3** &_pt) const

  Return the dot product of this vector and pt.
- bool **Equal** (const **Vector3** &_v) const

  Equality test.
- **Vector3 GetAbs** () const

  Get the absolute value of the vector.
- double **GetDistToLine** (const **Vector3** &_pt1, const **Vector3** &_pt2)

  Get distance to a line.
- double **GetLength** () const

  Returns the length (magnitude) of the vector \ return the length.
- double **GetMax** () const

  Get the maximum value in the vector.
- double **GetMin** () const

  Get the minimum value in the vector.
- **Vector3 GetPerpendicular** () const

  Return a vector that is perpendicular to this one.
- **Vector3 GetRounded** () const

  Get a rounded version of this vector.
- double **GetSquaredLength** () const

  Return the square of the length (magnitude) of the vector.
- double **GetSum** () const

  Return the sum of the values.
- bool **IsFinite** () const

  See if a point is finite (e.g., not nan)
- **Vector3 Normalize** ()

  Normalize the vector length.
- bool **operator!=** (const **Vector3** &_v) const

Not equal to operator.

- **Vector3 operator**∗ (const **Vector3** &_p) const

  Multiplication operator.
- **Vector3 operator**∗ (double _v) const

  Multiplication operators.
- const **Vector3** & **operator∗=** (const **Vector3** &_v)

  Multiplication operators.
- const **Vector3** & **operator∗=** (double _v)

  Multiplication operator.
- **Vector3 operator+** (const **Vector3** &_v) const

  Addition operator.
- const **Vector3** & **operator+=** (const **Vector3** &_v)

  Addition assignment operator.
- **Vector3 operator-** (const **Vector3** &_pt) const

  Subtraction operators.
- const **Vector3** & **operator-=** (const **Vector3** &_pt)

  Subtraction operators.
- const **Vector3 operator**/ (const **Vector3** &_pt) const

  Division operator.
- const **Vector3 operator**/ (double _v) const

  Division operator.
- const **Vector3** & **operator**/**=** (const **Vector3** &_pt)

  Division assignment operator.
- const **Vector3** & **operator**/**=** (double _v)

  Division operator.
- **Vector3** & **operator=** (const **Vector3** &_v)

  Assignment operator.
- **Vector3** & **operator=** (double _value)

  Assignment operator.
- bool **operator==** (const **Vector3** &_pt) const

  Equal to operator.
- double **operator[ ]** (unsigned int index) const

  [] operator
- **Vector3 Round** ()

  Round to near whole number, return the result.
- void **Round** (int _precision)

  Round all values to _precision decimal places.
- void **Set** (double _x=0, double _y=0, double _z=0)

  Set the contents of the vector.
- void **SetToMax** (const **Vector3** &_v)

  Set this vector's components to the maximum of itself and the passed in vector.
- void **SetToMin** (const **Vector3** &_v)

  Set this vector's components to the minimum of itself and the passed in vector.

Static Public Member Functions

- static **Vector3 GetNormal** (const **Vector3** &_v1, const **Vector3** &_v2, const **Vector3** &_v3)

  Get a normal vector to a triangle.

Public Attributes

- double **x**

  X location.
- double **y**

  Y location.
- double **z**

  Z location.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector3** &_pt)

  Stream insertion operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector3** &_pt)

  Stream extraction operator.

10.204.1 Detailed Description

The **Vector3** (p. 902) class represents the generic vector containing 3 elements.

Since it's commonly used to keep coordinate system related information, its elements are labeled by x, y, z.

10.204.2 Constructor & Destructor Documentation

10.204.2.1 gazebo::math::Vector3::Vector3 ( )

Constructor.

Referenced by operator-().

10.204.2.2 gazebo::math::Vector3::Vector3 (const double & _x, const double & _y, const double & _z)

Constructor.

Parameters

in	*_x*	value along x
in	*_y*	value along y
in	*_z*	value along z

10.204.2.3 gazebo::math::Vector3::Vector3 (const Vector3 & _v)

Copy constructor.

Parameters

in	*_v*	a vector

10.204.2.4 virtual gazebo::math::Vector3::~Vector3 ( ) [virtual]

Destructor.

10.204.3 Member Function Documentation

10.204.3.1 void gazebo::math::Vector3::Correct ( ) [inline]

Corrects any nan values.

References x, y, and z.

Referenced by gazebo::math::Pose::Correct().

10.204.3.2 Vector3 gazebo::math::Vector3::Cross (const Vector3 & _pt) const

Return the cross product of this vector and pt.

Returns

the product

10.204.3.3 double gazebo::math::Vector3::Distance (const Vector3 & _pt) const

Calc distance to the given point.

Parameters

in	*_pt*	the point

Returns

the distance

10.204.3.4 double gazebo::math::Vector3::Distance (double _x, double _y, double _z) const

Calc distance to the given point.

Parameters

in	*_x*	value along x
in	*_y*	value along y
in	*_z*	value along z

Returns

the distance

10.204.3.5 double gazebo::math::Vector3::Dot (const Vector3 & _pt) const

Return the dot product of this vector and pt.

Returns

the product

10.204.3.6 bool gazebo::math::Vector3::Equal (const Vector3 & _v) const

Equality test.

Remarks

This is equivalent to the == operator

Parameters

in	_v	the other vector

Returns

true if the 2 vectors have the same values, false otherwise

10.204.3.7 Vector3 gazebo::math::Vector3::GetAbs ( ) const

Get the absolute value of the vector.

Returns

a vector with positive elements

10.204.3.8 double gazebo::math::Vector3::GetDistToLine (const Vector3 & _pt1, const Vector3 & _pt2)

Get distance to a line.

Parameters

in	_pt1	first point on the line
in	_pt2	second point on the line

Returns

the minimum distance from this point to the line

10.204.3.9 double gazebo::math::Vector3::GetLength ( ) const

Returns the length (magnitude) of the vector \ return the length.

10.204.3.10 double gazebo::math::Vector3::GetMax ( ) const

Get the maximum value in the vector.

Returns

the maximum element

10.204.3.11 double gazebo::math::Vector3::GetMin ( ) const

Get the minimum value in the vector.

Returns

the minimum element

10.204.3.12 static Vector3 gazebo::math::Vector3::GetNormal (const Vector3 & *_v1*, const Vector3 & *_v2*, const Vector3 & *_v3*) [static]

Get a normal vector to a triangle.

Parameters

in	*_v1*	first vertex of the triangle
in	*_v2*	second vertex
in	*_v3*	third vertex

Returns

the normal

10.204.3.13 Vector3 gazebo::math::Vector3::GetPerpendicular ( ) const

Return a vector that is perpendicular to this one.

Returns

an orthogonal vector

10.204.3.14 Vector3 gazebo::math::Vector3::GetRounded ( ) const

Get a rounded version of this vector.

Returns

a rounded vector

10.204.3.15 double gazebo::math::Vector3::GetSquaredLength ( ) const

Return the square of the length (magnitude) of the vector.

Returns

the lenght

10.204.3.16 double gazebo::math::Vector3::GetSum ( ) const

Return the sum of the values.

Returns

the sum

10.204.3.17 bool gazebo::math::Vector3::IsFinite ( ) const

See if a point is finite (e.g., not nan)

10.204.3.18 Vector3 gazebo::math::Vector3::Normalize ( )

Normalize the vector length.

Returns

unit length vector

10.204.3.19 bool gazebo::math::Vector3::operator!= (const Vector3 & _v) const

Not equal to operator.

Parameters

in	_v	a vector

Returns

true if each component is equal withing a default tolerence (1e-6), false otherwise

10.204.3.20 Vector3 gazebo::math::Vector3::operator* (const Vector3 & _p) const

Multiplication operator.

Remarks

this is an element wise multiplication, not a cross product

Parameters

in	_v	

10.204.3.21 Vector3 gazebo::math::Vector3::operator∗ (double _v) const

Multiplication operators.

Parameters

in	_v	the scaling factor

Returns

a scaled vector

10.204.3.22 const Vector3& gazebo::math::Vector3::operator∗= (const Vector3 & _v)

Multiplication operators.

Remarks

this is an element wise multiplication, not a cross product

Parameters

in	_v	a vector

Returns

this

10.204.3.23 const Vector3& gazebo::math::Vector3::operator∗= (double _v)

Multiplication operator.

Parameters

in	_v	scaling factor

Returns

this

10.204.3.24 Vector3 gazebo::math::Vector3::operator+ (const Vector3 & _v) const

Addition operator.

Parameters

in	*_v*	vector to add

Returns

the sum vector

10.204.3.25 const Vector3& gazebo::math::Vector3::operator+= (const Vector3 & _v)

Addition assignment operator.

Parameters

in	*_v*	vector to add

10.204.3.26 Vector3 gazebo::math::Vector3::operator- (const Vector3 & _pt) const [inline]

Subtraction operators.

Parameters

in	*_pt*	a vector to substract

Returns

a vector

References Vector3(), x, y, and z.

10.204.3.27 const Vector3& gazebo::math::Vector3::operator-= (const Vector3 & _pt)

Subtraction operators.

10.204.3.28 const Vector3 gazebo::math::Vector3::operator/ (const Vector3 & _pt) const

Division operator.

Remarks

this is an element wise division

Returns

a vector

10.204.3.29 const Vector3 gazebo::math::Vector3::operator/ (double _v) const

Division operator.

Remarks

this is an element wise division

Returns

a vector

10.204.3.30 const Vector3& gazebo::math::Vector3::operator/= (const Vector3 & _pt)

Division assignment operator.

Remarks

this is an element wise division

Returns

a vector

10.204.3.31 const Vector3& gazebo::math::Vector3::operator/= (double _v)

Division operator.

Remarks

this is an element wise division

Returns

this

10.204.3.32 Vector3& gazebo::math::Vector3::operator= (const Vector3 & _v)

Assignment operator.

Parameters

in	_v	a new value

Returns

this

10.204.3.33 Vector3& gazebo::math::Vector3::operator= (double _value)

Assignment operator.

Parameters

in	*_value*	assigned to all elements

Returns

this

10.204.3.34 bool gazebo::math::Vector3::operator== (const Vector3 & *_pt*) const

Equal to operator.

Returns

true if each component is equal withing a default tolerence (1e-6), false otherwise

10.204.3.35 double gazebo::math::Vector3::operator[] (unsigned int *index*) const

[] operator

10.204.3.36 Vector3 gazebo::math::Vector3::Round ( )

Round to near whole number, return the result.

Returns

the result

10.204.3.37 void gazebo::math::Vector3::Round (int *_precision*)

Round all values to _precision decimal places.

Parameters

in	*_precision*	the decimal places

10.204.3.38 void gazebo::math::Vector3::Set (double *_x* = 0, double *_y* = 0, double *_z* = 0) [inline]

Set the contents of the vector.

Parameters

in	*_x*	value along x
in	*_y*	value along y
in	*_z*	value aling z

References x, y, and z.

10.204.3.39 void gazebo::math::Vector3::SetToMax (const Vector3 & _v)

Set this vector's components to the maximum of itself and the passed in vector.

Parameters

in	*_v*	the maximum clamping vector

10.204.3.40 void gazebo::math::Vector3::SetToMin (const Vector3 & _v)

Set this vector's components to the minimum of itself and the passed in vector.

Parameters

in	*the*	minimumn clamping vector

10.204.4 Friends And Related Function Documentation

10.204.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector3 & _pt) [friend]

Stream insertion operator.

Parameters

_out	output stream
_pt	**Vector3** (p. 902) to output

Returns

the stream

10.204.4.2 std::istream& operator>> (std::istream & _in, gazebo::math::Vector3 & _pt) [friend]

Stream extraction operator.

Parameters

_in	input stream
_pt	vector3 to read values into

Returns

the stream

10.204.5 Member Data Documentation

10.204.5.1 double gazebo::math::Vector3::x

X location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), gazebo::physics::ODEPlaneShape::CreatePlane(),

operator-(), gazebo::math::Quaternion::RotateVector(), Set(), gazebo::physics::ODEPlaneShape::SetAltitude(), gazebo::physics::ODEBoxShape::SetSize(), and gazebo::physics::BulletBoxShape::SetSize().

10.204.5.2 double gazebo::math::Vector3::y

Y location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), gazebo::physics::ODEPlaneShape::CreatePlane(), operator-(), gazebo::math::Quaternion::RotateVector(), Set(), gazebo::physics::ODEPlaneShape::SetAltitude(), gazebo::physics::ODEBoxShape::SetSize(), and gazebo::physics::BulletBoxShape::SetSize().

10.204.5.3 double gazebo::math::Vector3::z

Z location.

Referenced by gazebo::math::Pose::CoordPositionSub(), Correct(), gazebo::physics::ODEPlaneShape::CreatePlane(), operator-(), gazebo::math::Quaternion::RotateVector(), Set(), gazebo::physics::ODEPlaneShape::SetAltitude(), gazebo::physics::ODEBoxShape::SetSize(), and gazebo::physics::BulletBoxShape::SetSize().

The documentation for this class was generated from the following file:

- **Vector3.hh**

10.205 gazebo::math::Vector4 Class Reference

double Generic x, y, z, w vector

`#include <Vector4.hh>`

Public Member Functions

- **Vector4** ()

  Constructor.
- **Vector4** (const double &_x, const double &_y, const double &_z, const double &_w)

  Constructor with component values.
- **Vector4** (const **Vector4** &_v)

  Copy constructor.
- virtual ~**Vector4** ()

  Destructor.
- double **Distance** (const **Vector4** &_pt) const

  Calc distance to the given point.
- double **GetLength** () const

  Returns the length (magnitude) of the vector.
- double **GetSquaredLength** () const

  Return the square of the length (magnitude) of the vector.
- bool **IsFinite** () const

  See if a point is finite (e.g., not nan)
- void **Normalize** ()

  Normalize the vector length.

- bool **operator!=** (const **Vector4** &_pt) const

  Not equal to operator.
- const **Vector4 operator**∗ (const **Vector4** &_pt) const

  Multiplication operator.
- const **Vector4 operator**∗ (const **Matrix4** &_m) const

  Matrix multiplication operator.
- const **Vector4 operator**∗ (double _v) const

  Multiplication operators.
- const **Vector4** & **operator∗=** (const **Vector4** &_pt)

  Multiplication assignment operator.
- const **Vector4** & **operator∗=** (double _v)

  Multiplication assignment operator.
- **Vector4 operator+** (const **Vector4** &_v) const

  Addition operator.
- const **Vector4** & **operator+=** (const **Vector4** &_v)

  Addition operator.
- **Vector4 operator-** (const **Vector4** &_v) const

  Subtraction operator.
- const **Vector4** & **operator-=** (const **Vector4** &_v)

  Subtraction 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.

10.205.1 Detailed Description

double Generic x, y, z, w vector

10.205.2 Constructor & Destructor Documentation

10.205.2.1 gazebo::math::Vector4::Vector4 ( )

Constructor.

10.205.2.2 gazebo::math::Vector4::Vector4 (const double & *_x*, const double & *_y*, const double & *_z*, const double & *_w*)

Constructor with component values.

Parameters

in	*_x*	value along x axis
in	*_y*	value along y axis
in	*_z*	value along z axis
in	*_w*	value along w axis

10.205.2.3 gazebo::math::Vector4::Vector4 (const Vector4 & *_v*)

Copy constructor.

Parameters

in	*_v*	vector

10.205.2.4 virtual gazebo::math::Vector4::~Vector4 ( ) [virtual]

Destructor.

10.205.3 Member Function Documentation

10.205.3.1 double gazebo::math::Vector4::Distance (const Vector4 & *_pt*) const

Calc distance to the given point.

Parameters

in	*_pt*	the point

Returns

the distance

10.205.3.2 double gazebo::math::Vector4::GetLength ( ) const

Returns the length (magnitude) of the vector.

10.205.3.3 double gazebo::math::Vector4::GetSquaredLength ( ) const

Return the square of the length (magnitude) of the vector.

Returns

the length

10.205.3.4 bool gazebo::math::Vector4::IsFinite ( ) const

See if a point is finite (e.g., not nan)

Returns

true if finite, false otherwise

10.205.3.5 void gazebo::math::Vector4::Normalize ( )

Normalize the vector length.

10.205.3.6 bool gazebo::math::Vector4::operator!= (const Vector4 & *_pt*) const

Not equal to operator.

Parameters

in	*_pt*	the other vector

Returns

true if each component is equal withing a default tolerence (1e-6), false otherwise

10.205.3.7 const Vector4 gazebo::math::Vector4::operator* (const Vector4 & *_pt*) const

Multiplication operator.

Remarks

Performs element wise multiplication, which has limited use.

Parameters

in	*_pt*	another vector

Returns

result vector

10.205.3.8 const Vector4 gazebo::math::Vector4::operator∗ (const Matrix4 & *_m*) const

Matrix multiplication operator.

Parameters

in	*_m*	matrix

Returns

the vector multiplied by _m

10.205.3.9 const Vector4 gazebo::math::Vector4::operator∗ (double *_v*) const

Multiplication operators.

Parameters

in	*_v*	scaling factor

Returns

a scaled vector

10.205.3.10 const Vector4& gazebo::math::Vector4::operator∗= (const Vector4 & *_pt*)

Multiplication assignment operator.

Remarks

Performs element wise multiplication, which has limited use.

Parameters

in	*_pt*	a vector

Returns

this

10.205.3.11 const Vector4& gazebo::math::Vector4::operator*= (double _v)

Multiplication assignment operator.

Parameters

in	_v	scaling factor

Returns

this

10.205.3.12 Vector4 gazebo::math::Vector4::operator+ (const Vector4 & _v) const

Addition operator.

Parameters

in	_v	the vector to add

Returns

a sum vector

10.205.3.13 const Vector4& gazebo::math::Vector4::operator+= (const Vector4 & _v)

Addition operator.

Parameters

in	_v	the vector to add

Returns

this vector

10.205.3.14 Vector4 gazebo::math::Vector4::operator- (const Vector4 & _v) const

Subtraction operator.

Parameters

in	_v	the vector to substract

Returns

a vector

10.205.3.15 const Vector4& gazebo::math::Vector4::operator-= (const Vector4 & _v)

Subtraction assigment operators.

Parameters

in	_v	the vector to substract

Returns

this vector

10.205.3.16 const Vector4 gazebo::math::Vector4::operator/ (const Vector4 & _v) const

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

in	_v	the vector to perform element wise division with

Returns

a result vector

10.205.3.17 const Vector4 gazebo::math::Vector4::operator/ (double _v) const

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

in	_pt	another vector

Returns

a result vector

10.205.3.18 const Vector4& gazebo::math::Vector4::operator/= (const Vector4 & _v)

Division assignment operator.

Remarks

Performs element wise division, which has limited use.

Parameters

in	_v	the vector to perform element wise division with

Returns

this

10.205.3.19 const Vector4& gazebo::math::Vector4::operator/= (double _v)

Division operator.

Parameters

in	_v	scaling factor

Returns

a vector

10.205.3.20 Vector4& gazebo::math::Vector4::operator= (const Vector4 & _v)

Assignment operator.

Parameters

in	_v	the vector

Returns

a reference to this vector

10.205.3.21 Vector4& gazebo::math::Vector4::operator= (double _value)

Assignment operator.

Parameters

in	_value	

10.205.3.22 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

10.205.3.23 double gazebo::math::Vector4::operator[] (unsigned int _index) const

Array subscript operator.

Parameters

in	*_index*	

10.205.3.24 void gazebo::math::Vector4::Set (double _x = 0, double _y = 0, double _z = 0, double _w = 0)

Set the contents of the vector.

Parameters

in	*_x*	value along x axis
in	*_y*	value along y axis
in	*_z*	value along z axis
in	*_w*	value along w axis

10.205.4 Friends And Related Function Documentation

10.205.4.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Vector4 & _pt) [friend]

Stream insertion operator.

Parameters

in	*_out*	output stream
in	*_pt*	**Vector4** (p. 915) to output

Returns

The stream

10.205.4.2 std::istream& operator>> (std::istream & _in, gazebo::math::Vector4 & _pt) [friend]

Stream extraction operator.

Parameters

in	*_in*	input stream
in	*_pt*	**Vector4** (p. 915) to read values into

Returns

the stream

10.205.5 Member Data Documentation

10.205.5.1 double gazebo::math::Vector4::w

W value.

10.205.5.2 double gazebo::math::Vector4::x

X value.

10.205.5.3 double gazebo::math::Vector4::y

Y value.

10.205.5.4 double gazebo::math::Vector4::z

Z value.

The documentation for this class was generated from the following file:

- **Vector4.hh**

10.206 gazebo::common::Video Class Reference

Handle video encoding and decoding using libavcodec.

`#include <Video.hh>`

Public Member Functions

- **Video** ()

  Constructor.
- virtual ∼**Video** ()

  Destructor.
- int **GetHeight** () const

  Get the height of the video in pixels.
- bool **GetNextFrame** (unsigned char ∗∗_buffer)

  Get the next frame of the video.
- int **GetWidth** () const

Get the width of the video in pixels.

- bool **Load** (const std::string &_filename)

  Load a video file.

10.206.1 Detailed Description

Handle video encoding and decoding using libavcodec.

10.206.2 Constructor & Destructor Documentation

10.206.2.1 gazebo::common::Video::Video ( )

Constructor.

10.206.2.2 virtual gazebo::common::Video::~Video ( ) [virtual]

Destructor.

10.206.3 Member Function Documentation

10.206.3.1 int gazebo::common::Video::GetHeight ( ) const

Get the height of the video in pixels.

Returns

the height

10.206.3.2 bool gazebo::common::Video::GetNextFrame (unsigned char ** *_buffer*)

Get the next frame of the video.

Parameters

out	*_img*	**Image** (p. 407) in which the frame is stored

Returns

false if HAVE_FFMPEG is not defined, true otherwise

10.206.3.3 int gazebo::common::Video::GetWidth ( ) const

Get the width of the video in pixels.

Returns

the width

10.206.3.4 bool gazebo::common::Video::Load (const std::string & _filename)

Load a video file.

Parameters

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:

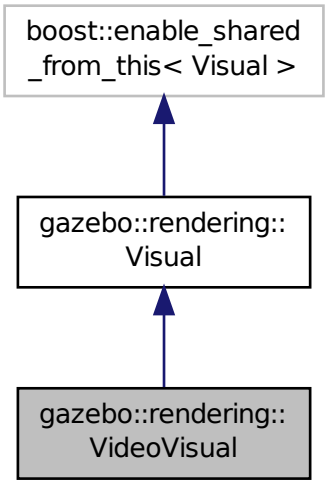
- **Video.hh**

10.207 gazebo::rendering::VideoVisual Class Reference

A visual element that displays a video as a texture.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::VideoVisual:



Public Member Functions

- **VideoVisual** (const std::string &_name, **VisualPtr** _parent)

  Constructor.
- virtual ~**VideoVisual** ()

  Destructor.

Additional Inherited Members

10.207.1 Detailed Description

A visual element that displays a video as a texture.

10.207.2 Constructor & Destructor Documentation

10.207.2.1 gazebo::rendering::VideoVisual::VideoVisual (const std::string & *_name,* VisualPtr *_parent*)

Constructor.

Parameters

in	*_name*	Name of the video visual.
in	*_parent*	Parent of the video visual.

10.207.2.2 virtual gazebo::rendering::VideoVisual::~VideoVisual ( ) [virtual]

Destructor.

The documentation for this class was generated from the following file:

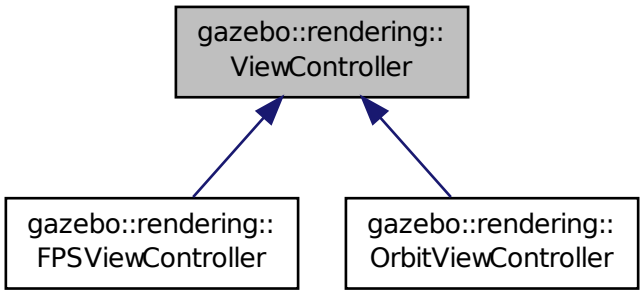
- **VideoVisual.hh**

10.208 gazebo::rendering::ViewController Class Reference

Base class for view controllers.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::ViewController:



Public Member Functions

- **ViewController** (**UserCameraPtr** _camera)

  Constructor.
- virtual ~**ViewController** ()

  Destructor.
- std::string **GetTypeString** () const

  Get the type of view controller.
- virtual void **HandleKeyPressEvent** (const std::string &_key)=0

  Handle a key press event.
- virtual void **HandleKeyReleaseEvent** (const std::string &_key)=0

  Handle a key release event.
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)=0

  Handle a mouse event.
- virtual void **Init** ()=0

  Initialize the view controller.
- virtual void **Init** (const **math::Vector3** &_focalPoint)

  Initialize with a focus point.
- void **SetEnabled** (bool _value)

  Set whether the controller is enabled.
- virtual void **Update** ()=0

  *Update the controller, which should update the position of the **Camera*** (p. 233).

Protected Attributes

- **UserCameraPtr camera**

  Pointer to the camera to control.
- bool **enabled**

  True if enabled.
- std::string **typeString**

  Type of view controller.

10.208.1 Detailed Description

Base class for view controllers.

10.208.2 Constructor & Destructor Documentation

10.208.2.1 gazebo::rendering::ViewController::ViewController (UserCameraPtr *_camera*)

Constructor.

Parameters

in	*_camera*	The user camera to controll.

10.208.2.2 virtual gazebo::rendering::ViewController::~ViewController ( ) [virtual]

Destructor.

10.208.3 Member Function Documentation

10.208.3.1 std::string gazebo::rendering::ViewController::GetTypeString ( ) const

Get the type of view controller.

Returns

The view controller type string.

10.208.3.2 virtual void gazebo::rendering::ViewController::HandleKeyPressEvent (const std::string & *_key*) [pure virtual]

Handle a key press event.

Parameters

in	*_key*	The key that was pressed.

Implemented in **gazebo::rendering::OrbitViewController** (p. 640), and **gazebo::rendering::FPSViewController** (p. 368).

10.208.3.3 virtual void gazebo::rendering::ViewController::HandleKeyReleaseEvent (const std::string & *_key*) [pure virtual]

Handle a key release event.

Parameters

in	*_key*	The key that was released.

Implemented in **gazebo::rendering::OrbitViewController** (p. 640), and **gazebo::rendering::FPSViewController** (p. 368).

10.208.3.4 virtual void gazebo::rendering::ViewController::HandleMouseEvent (const common::MouseEvent & *_event*) [pure virtual]

Handle a mouse event.

Parameters

in	*_event*	The mouse position.

Implemented in **gazebo::rendering::OrbitViewController** (p. 640), and **gazebo::rendering::FPSViewController** (p. 369).

10.208.3.5 virtual void gazebo::rendering::ViewController::Init ( ) [pure virtual]

Initialize the view controller.

Implemented in **gazebo::rendering::OrbitViewController** (p. 640), and **gazebo::rendering::FPSViewController** (p. 369).

10.208.3.6 virtual void gazebo::rendering::ViewController::Init (const math::Vector3 & *_focalPoint*) [virtual]

Initialize with a focus point.

Parameters

in	*_focalPoint*	The point to look at.

Reimplemented in **gazebo::rendering::OrbitViewController** (p. 641).

10.208.3.7 void gazebo::rendering::ViewController::SetEnabled (bool *_value*)

Set whether the controller is enabled.

Parameters

in	*_value*	True if the controller is enabled.

10.208.3.8 virtual void gazebo::rendering::ViewController::Update ( ) [pure virtual]

Update the controller, which should update the position of the **Camera** (p. 233).

Implemented in **gazebo::rendering::OrbitViewController** (p. 642), and **gazebo::rendering::FPSViewController** (p. 369).

10.208.4 Member Data Documentation

10.208.4.1 UserCameraPtr gazebo::rendering::ViewController::camera [protected]

Pointer to the camera to control.

10.208.4.2 bool gazebo::rendering::ViewController::enabled [protected]

True if enabled.

10.208.4.3 std::string gazebo::rendering::ViewController::typeString [protected]

Type of view controller.

The documentation for this class was generated from the following file:

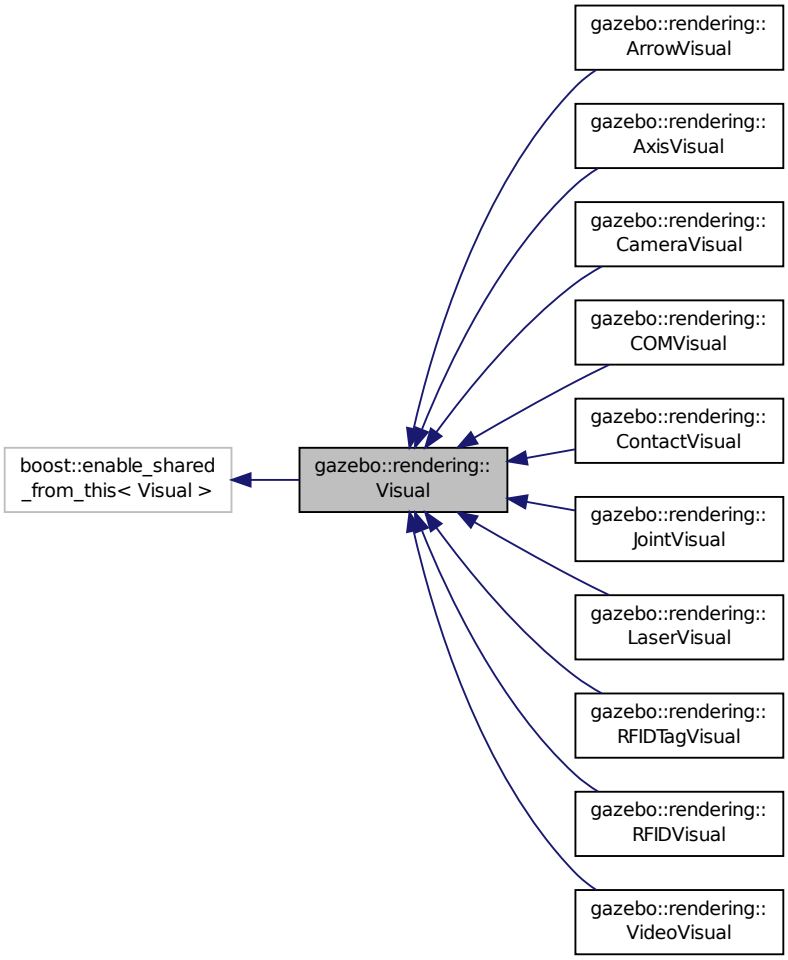
- **ViewController.hh**

10.209 gazebo::rendering::Visual Class Reference

A renderable object.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::Visual:



Public Member Functions

- **Visual** (const std::string &_name, **VisualPtr** _parent, bool _useRTShader=true)

  Constructor.
- **Visual** (const std::string &_name, **ScenePtr** _scene, bool _useRTShader=true)

  Constructor.
- virtual ~**Visual** ()

  Destructor.
- void **AttachAxes** ()

Attach visualization axes.

- void **AttachLineVertex** (**DynamicLines** *_line, unsigned int _index)

  Attach a vertex of a line to the position of the visual.

- Ogre::MovableObject * **AttachMesh** (const std::string &_meshName, const std::string &_objName="")

  Attach a mesh to this visual by name.

- void **AttachObject** (Ogre::MovableObject *_obj)

  Attach a renerable object to the visual.

- void **AttachVisual** (**VisualPtr** _vis)

  Attach a visual to this visual.

- void **ClearParent** ()

  Clear parents.

- **VisualPtr Clone** (const std::string &_name, **VisualPtr** _newParent)

  Clone the visual with a new name.

- **DynamicLines** * **CreateDynamicLine** (**RenderOpType** _type=**RENDERING_LINE_STRIP**)

  Add a line to the visual.

- void **DeleteDynamicLine** (**DynamicLines** *_line)

  Delete a dynamic line.

- void **DetachObjects** ()

  Detach all objects.

- void **DetachVisual** (**VisualPtr** _vis)

  Detach a visual.

- void **DetachVisual** (const std::string &_name)

  Detach a visual.

- void **DisableTrackVisual** ()

  Disable tracking of a visual.

- void **EnableTrackVisual** (**VisualPtr** _vis)

  Set one visual to track/follow another.

- void **Fini** ()

  Helper for the destructor.

- unsigned int **GetAttachedObjectCount** () const

  Return the number of attached movable objects.

- **math::Box GetBoundingBox** () const

  Get the bounding box for the visual.

- **VisualPtr GetChild** (unsigned int _index)

  Get an attached visual based on an index.

- unsigned int **GetChildCount** ()

  Get the number of attached visuals.

- std::string **GetMaterialName** () const

  Get the name of the material.

- std::string **GetName** () const

  Get the name of the visual.

- std::string **GetNormalMap** () const

  Get the normal map.

- **VisualPtr GetParent** () const

  Get the parent visual, if one exists.

- **math::Pose GetPose** () const

  Get the pose of the visual.

- **math::Vector3 GetPosition** () const

  Get the position of the visual.
- **VisualPtr GetRootVisual** ()

  Get the root visual.
- **math::Quaternion GetRotation** () const

  Get the rotation of the visual.
- **math::Vector3 GetScale** ()

  Get the scale.
- **ScenePtr GetScene** () const

  Get current.
- Ogre::SceneNode * **GetSceneNode** () const

  Return the scene Node of this visual entity.
- std::string **GetShaderType** () const

  Get the shader type.
- float **GetTransparency** ()

  Get the transparency.
- uint32_t **GetVisibilityFlags** ()

  Get visibility flags for this visual and all children.
- bool **GetVisible** () const

  Get whether the visual is visible.
- **math::Pose GetWorldPose** () const

  Get the global pose of the node.
- bool **HasAttachedObject** (const std::string &_name)

  Returns true if an object with _name is attached.
- void **Init** ()

  Helper for the contructor.
- void **InsertMesh** (const std::string &_meshName)

  *Insert a mesh into **Ogre** (p. 118).*
- bool **IsPlane** () const

  Return true if the visual is a plane.
- bool **IsStatic** () const

  Return true if the visual is a static geometry.
- void **Load** (**sdf::ElementPtr** _sdf)

  Load the visual with a set of parameters.
- virtual void **Load** ()

  Load the visual with default parameters.
- void **LoadFromMsg** (ConstVisualPtr &_msg)

  Load from a message.
- void **LoadPlugin** (const std::string &_filename, const std::string &_name, **sdf::ElementPtr** _sdf)

  Load a plugin.
- void **MakeStatic** ()

  Make the visual objects static renderables.
- void **MoveToPosition** (const **math::Pose** &_pose, double _time)

  Move to a pose and over a given time.
- void **MoveToPositions** (const std::vector< **math::Pose** > &_pts, double _time, boost::function< void()> _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)
- void **SetMaterial** (const std::string &_materialName, bool _unique=true)

  Set the material.
- void **SetName** (const std::string &_name)

  Set the name of the visual.
- void **SetNormalMap** (const std::string &nmap)

  Set the normal map.
- void **SetPose** (const **math::Pose** &_pose)

  Set the pose of the visual.
- void **SetPosition** (const **math::Vector3** &_pos)

  Set the position of the visual.
- void **SetRibbonTrail** (bool _value, const **common::Color** &_initialColor, const **common::Color** &_changeColor)

  True on or off a ribbon trail.
- void **SetRotation** (const **math::Quaternion** &_rot)

  Set the rotation of the visual.
- void **SetScale** (const **math::Vector3** &_scale)

  Set the scale.
- void **SetScene** (**ScenePtr** _scene)

  Set current scene.
- void **SetShaderType** (const std::string &_type)

  Set the shader type for the visual's material.
- void **SetSkeletonPose** (const msgs::PoseAnimation &_pose)

  Set animation skeleton pose.
- void **SetSpecular** (const **common::Color** &_color)

  Set the specular color of the visual.
- void **SetTransparency** (float _trans)

  Set the transparency.
- void **SetVisibilityFlags** (uint32_t _flags)

  Set visibility flags for this visual and all children.
- void **SetVisible** (bool _visible, bool _cascade=true)

  Set whether the visual is visible.
- void **SetWorldPose** (const **math::Pose** _pose)

  Set the world pose of the visual.
- void **SetWorldPosition** (const **math::Vector3** &_pos)

  Set the world linear position of the visual.
- void **SetWorldRotation** (const **math::Quaternion** &_rot)

Set the world orientation of the visual.

- void **ShowBoundingBox** ()

  Display the bounding box visual.
- void **ShowCollision** (bool _show)

  Display the collision visuals.
- void **ShowCOM** (bool _show)

  Display Center of Mass visuals.
- void **ShowJoints** (bool _show)

  Display joint visuals.
- void **ShowSkeleton** (bool _show)

  Display the skeleton visuals.
- void **ToggleVisible** ()

  Toggle whether this visual is visible.
- void **Update** ()

  Update the visual.
- void **UpdateFromMsg** (ConstVisualPtr &_msg)

  Update a visual based on a message.

Static Public Member Functions

- static void **InsertMesh** (const **common::Mesh** *_mesh)

  *Insert a mesh into **Ogre** (p. 118).*

Protected Attributes

- **VisualPtr parent**

  Parent visual.
- **ScenePtr scene**

  Pointer to the visual's scene.
- Ogre::SceneNode * **sceneNode**

  *Pointer to the visual's scene node in **Ogre** (p. 118).*

10.209.1 Detailed Description

A renderable object.

10.209.2 Constructor & Destructor Documentation

10.209.2.1 gazebo::rendering::Visual::Visual (const std::string & *_name*, VisualPtr *_parent*, bool *_useRTShader* = `true`)

Constructor.

Parameters

in	*_name*	Name of the visual.
in	*_parent*	Parent of the visual.
in	*_useRTShader*	True if the visual should use the real-time shader system (RTShader).

10.209.2.2 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. 746) containing the visual.
in	*_useRTShader*	True if the visual should use the real-time shader system (RTShader).

10.209.2.3 virtual gazebo::rendering::Visual::~Visual ( ) [virtual]

Destructor.

10.209.3 Member Function Documentation

10.209.3.1 void gazebo::rendering::Visual::AttachAxes ( )

Attach visualization axes.

10.209.3.2 void gazebo::rendering::Visual::AttachLineVertex (DynamicLines * *_line,* unsigned int *_index*)

Attach a vertex of a line to the position of the visual.

Parameters

in	*_line*	Line to attach to this visual.
in	*_index*	Index of the line vertex to attach.

10.209.3.3 Ogre::MovableObject* gazebo::rendering::Visual::AttachMesh (const std::string & *_meshName,* const std::string & *_objName* = " ")

Attach a mesh to this visual by name.

Parameters

in	*_meshName*	Name of the mesh.
in	*_objName*	Name of the attached Object to put the mesh onto.

10.209.3.4 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.

10.209.3.5 void gazebo::rendering::Visual::AttachVisual (VisualPtr _vis)

Attach a visual to this visual.

Parameters

in	*_vis*	**Visual** (p. 931) to attach.

10.209.3.6 void gazebo::rendering::Visual::ClearParent ( )

Clear parents.

10.209.3.7 VisualPtr gazebo::rendering::Visual::Clone (const std::string & _name, VisualPtr _newParent)

Clone the visual with a new name.

Parameters

in	*_name*	Name of the cloned **Visual** (p. 931).
in	*_newParent*	Parent of the cloned **Visual** (p. 931).

Returns

The visual.

10.209.3.8 DynamicLines* gazebo::rendering::Visual::CreateDynamicLine (RenderOpType _type = RENDERING_LINE_STRIP)

Add a line to the visual.

Parameters

in	*_type*	The type of line to make.

Returns

A pointer to the new dynamic line.

10.209.3.9 void gazebo::rendering::Visual::DeleteDynamicLine (DynamicLines * _line)

Delete a dynamic line.

Parameters

in	*_line*	Pointer to the line to delete.

10.209.3.10 void gazebo::rendering::Visual::DetachObjects ( )

Detach all objects.

10.209.3.11 void gazebo::rendering::Visual::DetachVisual (VisualPtr _vis)

Detach a visual.

Parameters

in	*_vis*	**Visual** (p. 931) to detach.

10.209.3.12 void gazebo::rendering::Visual::DetachVisual (const std::string & _name)

Detach a visual.

Parameters

in	*_name*	Name of the visual to detach.

10.209.3.13 void gazebo::rendering::Visual::DisableTrackVisual ( )

Disable tracking of a visual.

10.209.3.14 void gazebo::rendering::Visual::EnableTrackVisual (VisualPtr _vis)

Set one visual to track/follow another.

Parameters

in	*_vis*	**Visual** (p. 931) to track.

10.209.3.15 void gazebo::rendering::Visual::Fini ( )

Helper for the destructor.

10.209.3.16 unsigned int gazebo::rendering::Visual::GetAttachedObjectCount ( ) const

Return the number of attached movable objects.

Returns

The number of attached movable objects.

10.209.3.17 math::Box gazebo::rendering::Visual::GetBoundingBox ( ) const

Get the bounding box for the visual.

Returns

The bounding box in world coordinates.

10.209.3.18 VisualPtr gazebo::rendering::Visual::GetChild (unsigned int _index)

Get an attached visual based on an index.

Index should be between 0 and **Visual::GetChildCount** (p. 939).

Parameters

in	*_index*	Index of the child to retreive.

Returns

Pointer to the child visual, NULL if index is invalid.

10.209.3.19 unsigned int gazebo::rendering::Visual::GetChildCount ( )

Get the number of attached visuals.

Returns

The number of children.

10.209.3.20 std::string gazebo::rendering::Visual::GetMaterialName ( ) const

Get the name of the material.

Returns

The name of the visual applied to this visual.

10.209.3.21 std::string gazebo::rendering::Visual::GetName ( ) const

Get the name of the visual.

Returns

The name of the visual.

10.209.3.22 std::string gazebo::rendering::Visual::GetNormalMap ( ) const

Get the normal map.

Returns

The name of the normal map material.

10.209.3.23 VisualPtr gazebo::rendering::Visual::GetParent ( ) const

Get the parent visual, if one exists.

Returns

Pointer to the parent visual, NULL if no parent.

10.209.3.24 math::Pose gazebo::rendering::Visual::GetPose ( ) const

Get the pose of the visual.

Returns

The **Visual** (p. 931)'s pose.

10.209.3.25 math::Vector3 gazebo::rendering::Visual::GetPosition ( ) const

Get the position of the visual.

Returns

The visual's position.

10.209.3.26 VisualPtr gazebo::rendering::Visual::GetRootVisual ( )

Get the root visual.

Returns

The root visual, which is one level below the world visual.

10.209.3.27 math::Quaternion gazebo::rendering::Visual::GetRotation ( ) const

Get the rotation of the visual.

Returns

The visual's rotation.

10.209.3.28 math::Vector3 gazebo::rendering::Visual::GetScale ( )

Get the scale.

Returns

The scaling factor.

10.209.3.29 ScenePtr gazebo::rendering::Visual::GetScene ( ) const

Get current.

Returns

Pointer to the scene.

10.209.3.30 Ogre::SceneNode* gazebo::rendering::Visual::GetSceneNode ( ) const

Return the scene Node of this visual entity.

Returns

The **Ogre** (p. 118) scene node.

10.209.3.31 std::string gazebo::rendering::Visual::GetShaderType ( ) const

Get the shader type.

Returns

String of the shader type: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".

10.209.3.32 float gazebo::rendering::Visual::GetTransparency ( )

Get the transparency.

Returns

The transparency.

10.209.3.33 uint32_t gazebo::rendering::Visual::GetVisibilityFlags ( )

Get visibility flags for this visual and all children.

Returns

The visiblity flags.

See Also

GZ_VISIBILITY_ALL (p. 1129)
GZ_VISIBILITY_GUI (p. 1129)
GZ_VISIBILITY_NOT_SELECTABLE (p. 1130)

10.209.3.34 bool gazebo::rendering::Visual::GetVisible ( ) const

Get whether the visual is visible.

Returns

True if the visual is visible.

10.209.3.35 math::Pose gazebo::rendering::Visual::GetWorldPose ( ) const

Get the global pose of the node.

Returns

The pose in the world coordinate frame.

10.209.3.36 bool gazebo::rendering::Visual::HasAttachedObject (const std::string & _name)

Returns true if an object with _name is attached.

Parameters

in	*_name*	Name of an object to find.

10.209.3.37 void gazebo::rendering::Visual::Init ( )

Helper for the contructor.

10.209.3.38 void gazebo::rendering::Visual::InsertMesh (const std::string & _meshName)

Insert a mesh into **Ogre** (p. 118).

Parameters

in	*_meshName*	Name of the mesh to insert.

10.209.3.39 static void gazebo::rendering::Visual::InsertMesh (const common::Mesh * _mesh) [static]

Insert a mesh into **Ogre** (p. 118).

Parameters

in	*_mesh*	Pointer to the mesh to insert.

10.209.3.40 bool gazebo::rendering::Visual::IsPlane ( ) const

Return true if the visual is a plane.

Returns

True if a plane.

10.209.3.41 bool gazebo::rendering::Visual::IsStatic ( ) const

Return true if the visual is a static geometry.

Returns

True if the visual is static.

10.209.3.42 void gazebo::rendering::Visual::Load (sdf::ElementPtr _sdf)

Load the visual with a set of parameters.

Parameters

in	*_sdf*	Load from an SDF element.

10.209.3.43 virtual void gazebo::rendering::Visual::Load ( ) [virtual]

Load the visual with default parameters.

Reimplemented in **gazebo::rendering::ArrowVisual** (p. 140), and **gazebo::rendering::AxisVisual** (p. 142).

10.209.3.44 void gazebo::rendering::Visual::LoadFromMsg (ConstVisualPtr & _msg)

Load from a message.

Parameters

in	*_msg*	A visual message.

10.209.3.45 void gazebo::rendering::Visual::LoadPlugin (const std::string & _filename, const std::string & _name, sdf::ElementPtr _sdf)

Load a plugin.

Parameters

_filename	The filename of the plugin
_name	A unique name for the plugin
_sdf	The SDF to pass into the plugin.

10.209.3.46 void gazebo::rendering::Visual::MakeStatic ( )

Make the visual objects static renderables.

10.209.3.47 void gazebo::rendering::Visual::MoveToPosition (const math::Pose & _pose, double _time)

Move to a pose and over a given time.

Parameters

in	*_pose*	Pose the visual will end at.
in	*_time*	Time it takes the visual to move to the pose.

10.209.3.48 void gazebo::rendering::Visual::MoveToPositions (const std::vector< math::Pose > & _pts, double _time, boost::function< void() > _onComplete = NULL)

Move to a series of pose and over a given time.

Parameters

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.

10.209.3.49 void gazebo::rendering::Visual::RemovePlugin (const std::string & _name)

Remove a running plugin.

Parameters

_name	The unique name of the plugin to remove

10.209.3.50 void gazebo::rendering::Visual::SetAmbient (const common::Color & _color)

Set the ambient color of the visual.

Parameters

in	*_color*	The ambient color.

10.209.3.51 void gazebo::rendering::Visual::SetCastShadows (bool _shadows)

Set whether the visual should cast shadows.

Parameters

in	*_shadows*	True to enable shadows.

10.209.3.52 void gazebo::rendering::Visual::SetDiffuse (const common::Color & _color)

Set the diffuse color of the visual.

Parameters

in	*_color*	Set the diffuse color.

10.209.3.53 virtual void gazebo::rendering::Visual::SetEmissive (const common::Color & _color) [virtual]

Set the emissive value.

Parameters

in	*_color*	The emissive color.

Reimplemented in **gazebo::rendering::LaserVisual** (p. 448).

10.209.3.54 void gazebo::rendering::Visual::SetHighlighted (bool *_highlighted*)

10.209.3.55 void gazebo::rendering::Visual::SetMaterial (const std::string & *_materialName*, bool *_unique* = `true`)

Set the material.

Parameters

in	*_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.

10.209.3.56 void gazebo::rendering::Visual::SetName (const std::string & *_name*)

Set the name of the visual.

Parameters

in	*_name*	Name of the visual

10.209.3.57 void gazebo::rendering::Visual::SetNormalMap (const std::string & *nmap*)

Set the normal map.

Parameters

in	*_nmap*	Name of the normal map material.

10.209.3.58 void gazebo::rendering::Visual::SetPose (const math::Pose & *_pose*)

Set the pose of the visual.

Parameters

in	*_pose*	The new pose of the visual.

10.209.3.59 void gazebo::rendering::Visual::SetPosition (const math::Vector3 & *_pos*)

Set the position of the visual.

Parameters

in	*_pos*	The position to set the visual to.

10.209.3.60 void gazebo::rendering::Visual::SetRibbonTrail (bool *_value,* const common::Color & *_initialColor,* const common::Color & *_changeColor*)

True on or off a ribbon trail.

Parameters

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.

10.209.3.61 void gazebo::rendering::Visual::SetRotation (const math::Quaternion & *_rot*)

Set the rotation of the visual.

Parameters

in	*_rot*	The rotation of the visual.

10.209.3.62 void gazebo::rendering::Visual::SetScale (const math::Vector3 & *_scale*)

Set the scale.

Parameters

in	*_scale*	The scaling factor for the visual.

10.209.3.63 void gazebo::rendering::Visual::SetScene (ScenePtr *_scene*)

Set current scene.

Parameters

in	*_scene*	Pointer to the scene.

10.209.3.64 void gazebo::rendering::Visual::SetShaderType (const std::string & *_type*)

Set the shader type for the visual's material.

Parameters

in	*_type*	Shader type string: "vertex", "pixel", "normal_map_object_space", "normal_map_tangent_space".

10.209.3.65 void gazebo::rendering::Visual::SetSkeletonPose (const msgs::PoseAnimation & *_pose*)

Set animation skeleton pose.

Parameters

in	*_pose*	Skelton message

10.209.3.66 void gazebo::rendering::Visual::SetSpecular (const common::Color & *_color*)

Set the specular color of the visual.

Parameters

in	*_color*	Specular color.

10.209.3.67 void gazebo::rendering::Visual::SetTransparency (float *_trans*)

Set the transparency.

Parameters

in	*_trans*	The transparency, between 0 and 1 where 0 is no transparency.

10.209.3.68 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. 1129)
GZ_VISIBILITY_GUI (p. 1129)
GZ_VISIBILITY_NOT_SELECTABLE (p. 1130)

10.209.3.69 void gazebo::rendering::Visual::SetVisible (bool *_visible,* bool *_cascade* = `true`)

Set whether the visual is visible.

Parameters

in	*_visible*	set this node visible.
in	*_cascade*	setting this parameter in children too.

10.209.3.70 void gazebo::rendering::Visual::SetWorldPose (const math::Pose *_pose*)

Set the world pose of the visual.

Parameters

in	*_pose*	Pose of the visual in the world coordinate frame.

10.209.3.71 void gazebo::rendering::Visual::SetWorldPosition (const math::Vector3 & *_pos*)

Set the world linear position of the visual.

Parameters

in	*_pose*	Position in the world coordinate frame.

10.209.3.72 void gazebo::rendering::Visual::SetWorldRotation (const math::Quaternion & *_rot*)

Set the world orientation of the visual.

Parameters

in	*_rot*	Rotation in the world coordinate frame.

10.209.3.73 void gazebo::rendering::Visual::ShowBoundingBox ( )

Display the bounding box visual.

10.209.3.74 void gazebo::rendering::Visual::ShowCollision (bool *_show*)

Display the collision visuals.

Parameters

in	*_show*	True to show visuals labeled as collision objects.

10.209.3.75 void gazebo::rendering::Visual::ShowCOM (bool *_show*)

Display Center of Mass visuals.

Parameters

in	*_show*	True to show center of mass visualizations.

10.209.3.76 void gazebo::rendering::Visual::ShowJoints (bool *_show*)

Display joint visuals.

Parameters

in	*_show*	True to show joint visualizations.

10.209.3.77 void gazebo::rendering::Visual::ShowSkeleton (bool *_show*)

Display the skeleton visuals.

Parameters

in	*_show*	True to show skeleton visuals.

10.209.3.78 void gazebo::rendering::Visual::ToggleVisible ( )

Toggle whether this visual is visible.

10.209.3.79 void gazebo::rendering::Visual::Update ( )

Update the visual.

10.209.3.80 void gazebo::rendering::Visual::UpdateFromMsg (ConstVisualPtr & *_msg*)

Update a visual based on a message.

Parameters

in	*_msg*	The visual message.

10.209.4 Member Data Documentation

10.209.4.1 VisualPtr gazebo::rendering::Visual::parent [protected]

Parent visual.

10.209.4.2 ScenePtr gazebo::rendering::Visual::scene [protected]

Pointer to the visual's scene.

10.209.4.3 Ogre::SceneNode* gazebo::rendering::Visual::sceneNode [protected]

Pointer to the visual's scene node in **Ogre** (p. 118).

The documentation for this class was generated from the following file:

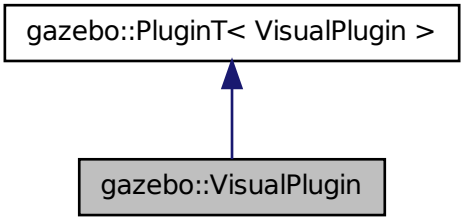
- **Visual.hh**

10.210 gazebo::VisualPlugin Class Reference

A plugin loaded within the gzserver on startup.

`#include <Plugin.hh>`

Inheritance diagram for gazebo::VisualPlugin:



Public Member Functions

- **VisualPlugin** ()
- virtual void **Init** ()

  Initialize the plugin.
- virtual void **Load** (**rendering::VisualPtr** _visual, **sdf::ElementPtr** _sdf)=0

  Load function.
- virtual void **Reset** ()

  Override this method for custom plugin reset behavior.

Additional Inherited Members

10.210.1 Detailed Description

A plugin loaded within the gzserver on startup.

See `reference`.

10.210.2 Constructor & Destructor Documentation

10.210.2.1 gazebo::VisualPlugin::VisualPlugin ( ) [inline]

References gazebo::PluginT< VisualPlugin >::type, and gazebo::VISUAL_PLUGIN.

10.210.3 Member Function Documentation

10.210.3.1 virtual void gazebo::VisualPlugin::Init ( ) [inline], [virtual]

Initialize the plugin.

Called after Gazebo has been loaded. Must not block.

10.210.3.2 virtual void gazebo::VisualPlugin::Load (rendering::VisualPtr _visual, sdf::ElementPtr _sdf) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	*_visual*	Pointer the Visual Object.
in	*_sdf*	Pointer the the SDF element of the plugin.

10.210.3.3 virtual void gazebo::VisualPlugin::Reset ( ) [inline], [virtual]

Override this method for custom plugin reset behavior.

The documentation for this class was generated from the following file:

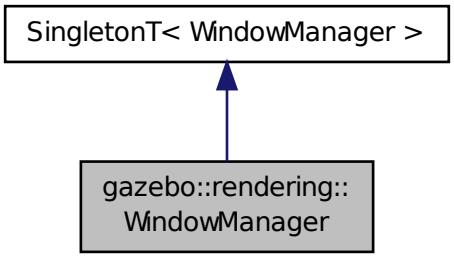
- **common/Plugin.hh**

10.211 gazebo::rendering::WindowManager Class Reference

Class to mangage render windows.

`#include <rendering/rendering.hh>`

Inheritance diagram for gazebo::rendering::WindowManager:



Public Member Functions

- int **CreateWindow** (const std::string &_ogreHandle, uint32_t _width, uint32_t _height)

  Create a window.
- void **Fini** ()

  Shutdown all the windows.
- float **GetAvgFPS** (uint32_t _id)

Get the average FPS.

- uint32_t **GetTriangleCount** (uint32_t _id)

  Get the triangle count.

- Ogre::RenderWindow * **GetWindow** (uint32_t _id)

  Get the render window associated with the given id.

- void **Moved** (uint32_t _id)

  *Tells **Ogre*** (p. *118) the window has moved, and needs updating.*

- void **Resize** (uint32_t _id, int _width, int _height)

  Resize a window.

- void **SetCamera** (int _windowId, **CameraPtr** _camera)

  Attach a camera to a window.

Additional Inherited Members

10.211.1 Detailed Description

Class to mangage render windows.

10.211.2 Member Function Documentation

10.211.2.1 int gazebo::rendering::WindowManager::CreateWindow (const std::string & *_ogreHandle*, uint32_t *_width*, uint32_t *_height*)

Create a window.

Parameters

in	*_ogreHandle*	String representing the ogre window handle.
in	*_width*	With of the window in pixels.
in	*_height*	Height of the window in pixels.

10.211.2.2 void gazebo::rendering::WindowManager::Fini ( )

Shutdown all the windows.

10.211.2.3 float gazebo::rendering::WindowManager::GetAvgFPS (uint32_t *_id*)

Get the average FPS.

Parameters

in	*_id*	ID of the window.

Returns

The frames per second.

10.211.2.4 uint32_t gazebo::rendering::WindowManager::GetTriangleCount (uint32_t _id)

Get the triangle count.

Parameters

in	*_id*	ID of the window.

Returns

The triangle count.

10.211.2.5 Ogre::RenderWindow* gazebo::rendering::WindowManager::GetWindow (uint32_t _id)

Get the render window associated with the given id.

Parameters

in	*_id*	ID of the window.

Returns

Pointer to the render window, NULL if the id is invalid.

10.211.2.6 void gazebo::rendering::WindowManager::Moved (uint32_t _id)

Tells **Ogre** (p. 118) the window has moved, and needs updating.

Parameters

in	*_id*	ID of the window.

10.211.2.7 void gazebo::rendering::WindowManager::Resize (uint32_t _id, int _width, int _height)

Resize a window.

Parameters

in	*_id*	Id of the window to resize.
in	*_width*	New width of the window.
in	*_height*	New height of the window.

10.211.2.8 void gazebo::rendering::WindowManager::SetCamera (int _windowId, CameraPtr _camera)

Attach a camera to a window.

Parameters

in	*_windowId*	Id of the window to add the camera to.
in	*_camera*	Pointer to the camera to attach.

The documentation for this class was generated from the following file:

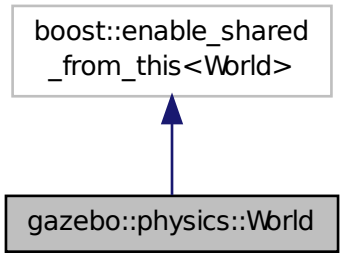
- **WindowManager.hh**

10.212 gazebo::physics::World Class Reference

The world provides access to all other object within a simulated environment.

`#include <physics/World.hh>`

Inheritance diagram for gazebo::physics::World:



Public Member Functions

- **World** (const std::string &_name="")
  Constructor.
- ~**World** ()
  Destructor.
- void **Clear** ()
  Remove all entities from the world.
- void **DisableAllModels** ()
  Disable all links in all the models.
- void **EnableAllModels** ()
  Enable all links in all the models.
- void **EnablePhysicsEngine** (bool _enable)
  enable/disable physics engine during World::Update
- void **Fini** ()
  Finilize the world.
- **BasePtr GetByName** (const std::string &_name)
  Get an element by name.
- bool **GetEnablePhysicsEngine** ()
  check if physics engine is enabled/disabled.
- **EntityPtr GetEntity** (const std::string &_name)

*Get a pointer to an **Entity** (p. 338) based on a name.*

- **EntityPtr GetEntityBelowPoint** (const **math::Vector3** &_pt)

  Get the nearest entity below a point.
- **EntityPtr GetEntityByName** (const std::string &_name) **GAZEBO_DEPRECATED**

  Get a pointer to a entity based on a name.
- **ModelPtr GetModel** (unsigned int _index) const

  Get a model based on an index.
- **ModelPtr GetModel** (const std::string &_name)

  Get a model by name.
- **ModelPtr GetModelBelowPoint** (const **math::Vector3** &_pt)

  Get the nearest model below a point.
- **ModelPtr GetModelByName** (const std::string &name) **GAZEBO_DEPRECATED**

  Get a model by name DEPRECATED.
- unsigned int **GetModelCount** () const

  Get the number of models.
- std::list< **ModelPtr** > **GetModels** () const

  Get a list of all the models.
- std::string **GetName** () const

  Get the name of the world.
- **common::Time GetPauseTime** () const

  Get the amount of time simulation has been paused.
- **PhysicsEnginePtr GetPhysicsEngine** () const

  Return the physics engine.
- **common::Time GetRealTime** () const

  Get the real time (elapsed time)
- **EntityPtr GetSelectedEntity** () const

  *Get the selected **Entity** (p. 338).*
- boost::mutex * **GetSetWorldPoseMutex** () const

  Get the set world pose mutex.
- **common::Time GetSimTime** () const

  *Get the world simulation time, note if you want the PC wall clock call **World::GetRealTime** (p. 961).*
- **common::Time GetStartTime** () const

  Get the wall time simulation was started.
- **WorldState GetState** ()

  Get the current world state.
- void **Init** ()

  Initialize the world.
- void **InsertModelFile** (const std::string &_sdfFilename)

  Insert a model from an SDF file.
- void **InsertModelSDF** (const **sdf::SDF** &_sdf)

  Insert a model using SDF.
- void **InsertModelString** (const std::string &_sdfString)

  Insert a model from an SDF string.
- bool **IsPaused** () const

  Returns the state of the simulation true if paused.
- void **Load** (**sdf::ElementPtr** _sdf)

  Load the world using SDF parameters.

- void **LoadPlugin** (const std::string &_filename, const std::string &_name, **sdf::ElementPtr** _sdf)

  Load a plugin.
- void **PrintEntityTree** ()

  *Print **Entity** (p. 338) tree.*
- void **RemovePlugin** (const std::string &_name)

  Remove a running plugin.
- void **Reset** ()

  Reset time and model poses, configurations in simulation.
- void **ResetEntities** (**Base::EntityType** _type=**Base::BASE**)

  Reset with options.
- void **ResetTime** ()

  Reset simulation time back to zero.
- void **Run** ()

  Run the world in a thread.
- void **Save** (const std::string &_filename)

  Save a world to a file.
- void **SetPaused** (bool _p)

  Set whether the simulation is paused.
- void **SetSimTime** (**common::Time** _t)

  Set the sim time.
- void **SetState** (const **WorldState** &_state)

  Set the current world state.
- void **StepWorld** (int _steps)

  Step callback.
- void **Stop** ()

  Stop the world.
- std::string **StripWorldName** (const std::string &_name) const

  Return a version of the name with "<world_name>::" removed.

Public Attributes

- std::list< **Entity** * > **dirtyPoses**

  *when physics engine makes an update and changes a link pose, this flag is set to trigger **Entity::SetWorldPose** (p. 348) on the **physics::Link** (p. 454) in World::Update.*

10.212.1 Detailed Description

The world provides access to all other object within a simulated environment.

The **World** (p. 954) is the container for all models and their components (links, joints, sensors, plugins, etc), and **WorldPlugin** (p. 965) instances. Many core function are also handled in the **World** (p. 954), including physics update, model updates, and message processing.

10.212.2 Constructor & Destructor Documentation

10.212.2.1 gazebo::physics::World::World (const std::string & *_name* = " ")

Constructor.

Constructor for the **World** (p. 954). Must specify a unique name.

Parameters

_name	Name of the world

10.212.2.2 gazebo::physics::World::~World ( )

Destructor.

10.212.3 Member Function Documentation

10.212.3.1 void gazebo::physics::World::Clear ( )

Remove all entities from the world.

10.212.3.2 void gazebo::physics::World::DisableAllModels ( )

Disable all links in all the models.

Disable is a physics concept. Disabling means that the physics engine should not update an entity.

10.212.3.3 void gazebo::physics::World::EnableAllModels ( )

Enable all links in all the models.

Enable is a physics concept. Enabling means that the physics engine should update an entity.

10.212.3.4 void gazebo::physics::World::EnablePhysicsEngine (bool *_enable*) [inline]

enable/disable physics engine during World::Update

Parameters

in	*_enable*	True to enable the physics engine.

10.212.3.5 void gazebo::physics::World::Fini ( )

Finilize the world.

Call this function to tear-down the world.

10.212.3.6 BasePtr gazebo::physics::World::GetByName (const std::string & *_name*)

Get an element by name.

Searches the list of entities, and return a pointer to the model with a matching _name.

Parameters

in	*_name*	The name of the **Model** (p. 521) to find.

Returns

A pointer to the entity, or NULL if no entity was found.

10.212.3.7 bool gazebo::physics::World::GetEnablePhysicsEngine ( ) [inline]

check if physics engine is enabled/disabled.

Parameters

True	if the physics engine is enabled.

10.212.3.8 EntityPtr gazebo::physics::World::GetEntity (const std::string & *_name*)

Get a pointer to an **Entity** (p. 338) based on a name.

This function is the same as GetByName, but limits the search to only Entities.

Parameters

in	*_name*	The name of the **Entity** (p. 338) to find.

Returns

A pointer to the **Entity** (p. 338), or NULL if no **Entity** (p. 338) was found.

10.212.3.9 EntityPtr gazebo::physics::World::GetEntityBelowPoint (const math::Vector3 & *_pt*)

Get the nearest entity below a point.

Projects a Ray down (-Z axis) starting at the given point. The first entity hit by the Ray is returned.

Parameters

in	*_pt*	The 3D point to search below

Returns

A pointer to nearest **Entity** (p. 338), NULL if none is found.

10.212.3.10 EntityPtr gazebo::physics::World::GetEntityByName (const std::string & *_name*)

Get a pointer to a entity based on a name.

Parameters

in	*_name*	Name of the entity.

10.212.3.11 ModelPtr gazebo::physics::World::GetModel (unsigned int *_index*) const

Get a model based on an index.

Get a **Model** (p. 521) using an index, where index must be greater than zero and less than **World::GetModelCount()** (p. 960)

Parameters

in	*_index*	The index of the model [0..GetModelCount)

Returns

A pointer to the **Model** (p. 521). NULL if _index is invalid.

10.212.3.12 ModelPtr gazebo::physics::World::GetModel (const std::string & *_name*)

Get a model by name.

This function is the same as GetByName, but limits the search to only models.

Parameters

in	*_name*	The name of the **Model** (p. 521) to find.

Returns

A pointer to the **Model** (p. 521), or NULL if no model was found.

10.212.3.13 ModelPtr gazebo::physics::World::GetModelBelowPoint (const math::Vector3 & *_pt*)

Get the nearest model below a point.

This function makes use of **World::GetEntityBelowPoint** (p. 958).

Parameters

in	*_pt*	The 3D point to search below

Returns

A pointer to nearest **Model** (p. 521), NULL if none is found.

10.212.3.14 ModelPtr gazebo::physics::World::GetModelByName (const std::string & *name*)

Get a model by name DEPRECATED.

10.212.3.15 unsigned int gazebo::physics::World::GetModelCount ( ) const

Get the number of models.

Returns

The number of models in the **World** (p. 954).

10.212.3.16 std::list<ModelPtr> gazebo::physics::World::GetModels ( ) const

Get a list of all the models.

Returns

A list of all the Models in the world

10.212.3.17 std::string gazebo::physics::World::GetName ( ) const

Get the name of the world.

Returns

The name of the world.

10.212.3.18 common::Time gazebo::physics::World::GetPauseTime ( ) const

Get the amount of time simulation has been paused.

Returns

The pause time

10.212.3.19 PhysicsEnginePtr gazebo::physics::World::GetPhysicsEngine ( ) const

Return the physics engine.

Get a pointer to the physics engine used by the world.

Returns

Pointer to the physics engine

10.212.3.20 common::Time gazebo::physics::World::GetRealTime ( ) const

Get the real time (elapsed time)

Returns

The real time

10.212.3.21 EntityPtr gazebo::physics::World::GetSelectedEntity ( ) const

Get the selected **Entity** (p. 338).

The selected entity is set via the GUI.

Returns

A point to the **Entity** (p. 338), NULL if nothing is selected

10.212.3.22 boost::mutex* gazebo::physics::World::GetSetWorldPoseMutex ( ) const [inline]

Get the set world pose mutex.

Returns

Pointer to the mutex.

10.212.3.23 common::Time gazebo::physics::World::GetSimTime ( ) const

Get the world simulation time, note if you want the PC wall clock call **World::GetRealTime** (p. 961).

Returns

The current simulation time

10.212.3.24 common::Time gazebo::physics::World::GetStartTime ( ) const

Get the wall time simulation was started.

Returns

The start time

10.212.3.25 WorldState gazebo::physics::World::GetState ( )

Get the current world state.

Returns

A object that contains the entire state of the **World** (p. 954)

10.212.3.26 void gazebo::physics::World::Init ( )

Initialize the world.

This is called after Load.

10.212.3.27 void gazebo::physics::World::InsertModelFile (const std::string & *_sdfFilename*)

Insert a model from an SDF file.

Spawns a model into the world base on and SDF file

Parameters

in	*_sdfFilename*	The name of the SDF file (including path).

10.212.3.28 void gazebo::physics::World::InsertModelSDF (const **sdf::SDF** & *_sdf*)

Insert a model using SDF.

Spawns a model into the world base on and SDF object

Parameters

in	*_sdf*	A reference to an SDF object

10.212.3.29 void gazebo::physics::World::InsertModelString (const std::string & *_sdfString*)

Insert a model from an SDF string.

Spawns a model into the world base on and SDF string

Parameters

in	*_sdfString*	A string containing valid SDF markup.

10.212.3.30 bool gazebo::physics::World::IsPaused ( ) const

Returns the state of the simulation true if paused.

Returns

True if paused.

10.212.3.31 void gazebo::physics::World::Load (**sdf::ElementPtr** *_sdf*)

Load the world using SDF parameters.

Load a world from and SDF pointer

Parameters

_sdf	SDF parameters

10.212.3.32 void gazebo::physics::World::LoadPlugin (const std::string & *_filename,* const std::string & *_name,* sdf::ElementPtr *_sdf*)

Load a plugin.

Parameters

in	*_filename*	The filename of the plugin
in	*_name*	A unique name for the plugin
in	*_sdf*	The SDF to pass into the plugin.

10.212.3.33 void gazebo::physics::World::PrintEntityTree ( )

Print **Entity** (p. 338) tree.

Prints alls the entities to stdout

10.212.3.34 void gazebo::physics::World::RemovePlugin (const std::string & *_name*)

Remove a running plugin.

Parameters

in	*_name*	The unique name of the plugin to remove

10.212.3.35 void gazebo::physics::World::Reset ( )

Reset time and model poses, configurations in simulation.

10.212.3.36 void gazebo::physics::World::ResetEntities (Base::EntityType *_type* = Base::BASE)

Reset with options.

The _type parameter specifies which type of eneities to reset. See **Base::EntityType** (p. 149).

Parameters

in	*_type*	The type of reset.

10.212.3.37 void gazebo::physics::World::ResetTime ( )

Reset simulation time back to zero.

10.212.3.38 void gazebo::physics::World::Run ( )

Run the world in a thread.

Run the update loop.

10.212.3.39 void gazebo::physics::World::Save (const std::string & *_filename*)

Save a world to a file.

Save the current world and its state to a file

Parameters

_filename	Name of the file to save into

10.212.3.40 void gazebo::physics::World::SetPaused (bool *_p*)

Set whether the simulation is paused.

Parameters

in	*_p*	True pauses the simulation. False runs the simulation.

10.212.3.41 void gazebo::physics::World::SetSimTime (common::Time *_t*)

Set the sim time.

Parameters

in	*_t*	The new simulation time

10.212.3.42 void gazebo::physics::World::SetState (const WorldState & *_state*)

Set the current world state.

Parameters

_state	The state to set the **World** (p. 954) to.

10.212.3.43 void gazebo::physics::World::StepWorld (int *_steps*)

Step callback.

Parameters

in	*_steps*	The number of steps the **World** (p. 954) should take

10.212.3.44 void gazebo::physics::World::Stop ( )

Stop the world.

Stop the update loop.

10.212.3.45 std::string gazebo::physics::World::StripWorldName (const std::string & *_name*) const

Return a version of the name with "<world_name>::" removed.

Parameters

in	*_name*	Usually the name of an entity.

Returns

The stripped world name

10.212.4 Member Data Documentation

10.212.4.1 std::list<Entity*> gazebo::physics::World::dirtyPoses

when physics engine makes an update and changes a link pose, this flag is set to trigger **Entity::SetWorldPose** (p. 348) on the **physics::Link** (p. 454) in World::Update.

The documentation for this class was generated from the following file:

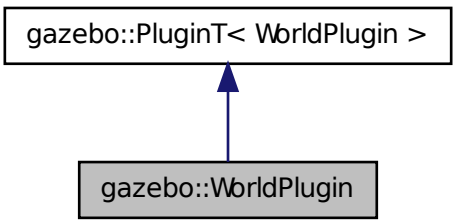
- **World.hh**

10.213 gazebo::WorldPlugin Class Reference

A plugin with access to **physics::World** (p. 954).

`#include <Plugin.hh>`

Inheritance diagram for gazebo::WorldPlugin:



Public Member Functions

- **WorldPlugin** ()

  Constructor.
- virtual ~**WorldPlugin** ()

  Destructor.
- virtual void **Init** ()
- virtual void **Load** (**physics::WorldPtr** _world, **sdf::ElementPtr** _sdf)=0

  Load function.
- virtual void **Reset** ()

Additional Inherited Members

10.213.1 Detailed Description

A plugin with access to **physics::World** (p. 954).

See `reference`.

10.213.2 Constructor & Destructor Documentation

10.213.2.1 gazebo::WorldPlugin::WorldPlugin ( ) [inline]

Constructor.

References gazebo::PluginT< WorldPlugin >::type, and gazebo::WORLD_PLUGIN.

10.213.2.2 virtual gazebo::WorldPlugin::~WorldPlugin ( ) [inline], [virtual]

Destructor.

10.213.3 Member Function Documentation

10.213.3.1 virtual void gazebo::WorldPlugin::Init ( ) [inline], [virtual]

10.213.3.2 virtual void gazebo::WorldPlugin::Load (**physics::WorldPtr** *_world,* **sdf::ElementPtr** *_sdf*) [pure virtual]

Load function.

Called when a Plugin is first created, and after the World has been loaded. This function should not be blocking.

Parameters

in	*_world*	Pointer the World
in	*_sdf*	Pointer the the SDF element of the plugin.

10.213.3.3 virtual void gazebo::WorldPlugin::Reset ( ) [inline], [virtual]

The documentation for this class was generated from the following file:

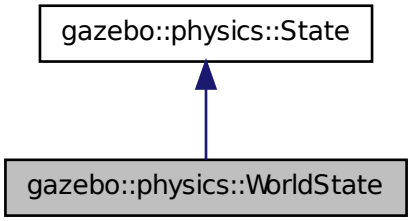
- **common/Plugin.hh**

10.214 gazebo::physics::WorldState Class Reference

Store state information of a **physics::World** (p. 954) object.

`#include <physics/WorldState.hh>`

Inheritance diagram for gazebo::physics::WorldState:



Public Member Functions

- **WorldState** ()
  Default constructor.
- **WorldState** (**WorldPtr** _world)
  Constructor.
- virtual ~**WorldState** ()
  Destructor.
- **ModelState GetModelState** (unsigned int _index) const
  Get a model state.
- **ModelState GetModelState** (const std::string &_modelName) const
  Get a model state by model name.
- unsigned int **GetModelStateCount** () const
  Get the number of model states.
- const **sdf::ElementPtr** & **GetSDF** () const
  Get the state in SDF format.
- virtual void **Load** (**sdf::ElementPtr** _elem)
  Load state from SDF element.

Additional Inherited Members

10.214.1 Detailed Description

Store state information of a **physics::World** (p. 954) object.

Instances of this class contain the state of a **World** (p. 954) at a specific time. **World** (p. 954) state includes the state of all models, and their children.

10.214.2 Constructor & Destructor Documentation

10.214.2.1 gazebo::physics::WorldState::WorldState ( )

Default constructor.

10.214.2.2 gazebo::physics::WorldState::WorldState (WorldPtr *_world*)

Constructor.

Generate a **WorldState** (p. 967) from an instance of a **World** (p. 954).

Parameters

_world	Pointer to a world

10.214.2.3 virtual gazebo::physics::WorldState::~WorldState ( ) [virtual]

Destructor.

10.214.3 Member Function Documentation

10.214.3.1 ModelState gazebo::physics::WorldState::GetModelState (unsigned int *_index*) const

Get a model state.

Get the state of a **Model** (p. 521) based on an index. The min index is and the max is **WorldState::GetModelStateCount()** (p. 968)

Parameters

_index	Index of the model

Returns

State (p. 813) of the requested **Model** (p. 521)

10.214.3.2 ModelState gazebo::physics::WorldState::GetModelState (const std::string & *_modelName*) const

Get a model state by model name.

10.214.3.3 unsigned int gazebo::physics::WorldState::GetModelStateCount ( ) const

Get the number of model states.

Returns the number of models in this instance.

Returns

Number of models

10.214.3.4 const sdf::ElementPtr& gazebo::physics::WorldState::GetSDF ( ) const

Get the state in SDF format.

Returns a pointer to the SDF representation of the **WorldState** (p. 967)

Returns

Pointer to the SDF representation of the **WorldState** (p. 967)

10.214.3.5 virtual void gazebo::physics::WorldState::Load (sdf::ElementPtr *_elem*) [virtual]

Load state from SDF element.

Set a **WorldState** (p. 967) from an SDF element containing **WorldState** (p. 967) info.

Parameters

_elem	Pointer to the **WorldState** (p. 967) SDF element.

Implements **gazebo::physics::State** (p. 815).

The documentation for this class was generated from the following file:

- **WorldState.hh**

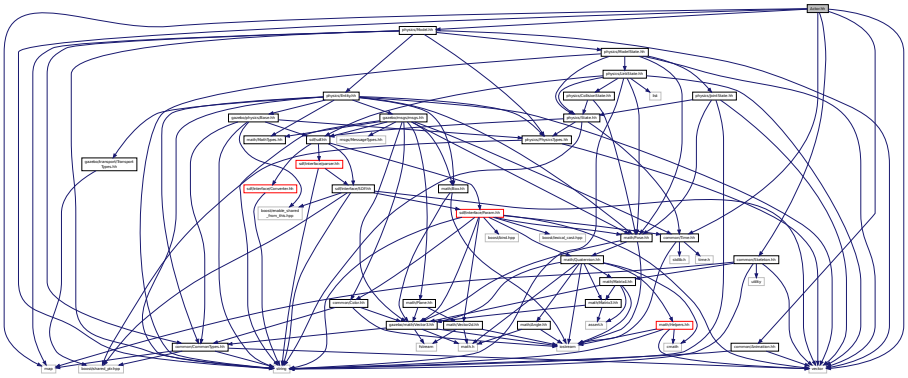
Chapter 11

File Documentation

11.1 Actor.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "physics/Model.hh"
#include "common/Time.hh"
#include "common/Skeleton.hh"
#include "common/Animation.hh"
```

Include dependency graph for Actor.hh:



Classes

- class **gazebo::physics::Actor**

  Actor (p. *121) class enables GPU based mesh model / skeleton scriptable animation.*
- struct **gazebo::physics::TrajectoryInfo**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::common**

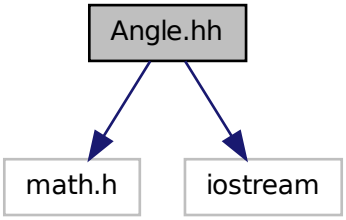
  Common namespace.
- namespace **gazebo::physics**

  namespace for physics

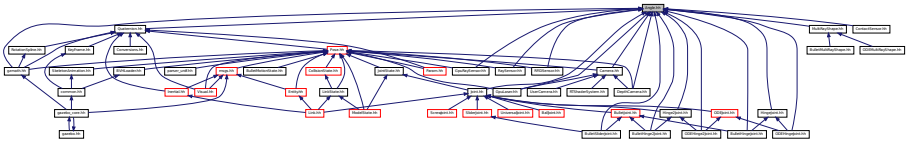
11.2 Angle.hh File Reference

```
#include <math.h>
#include <iostream>
```

Include dependency graph for Angle.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Angle**

  An angle and related functions.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

Macros

- #define **GZ_DTOR**(d) ((d) ∗ M_PI / 180)

  Converts degrees to radians.
- #define **GZ_NORMALIZE**(a) (atan2(sin(a), cos(a)))

  Macro tha normalizes an angle in the range -Pi to Pi.
- #define **GZ_RTOD**(r) ((r) ∗ 180 / M_PI)

  Macro that converts radians to degrees.

11.2.1 Macro Definition Documentation

11.2.1.1 #define GZ_DTOR(*d*) ((d) ∗ M_PI / 180)

Converts degrees to radians.

Parameters

in	*degrees*	

Returns

radians

11.2.1.2 #define GZ_NORMALIZE(*a*) (atan2(sin(a), cos(a)))

Macro tha normalizes an angle in the range -Pi to Pi.

Parameters

in	*angle*	

Returns

the angle, in range

11.2.1.3 #define GZ_RTOD(*r*) ((r) ∗ 180 / M_PI)

Macro that converts radians to degrees.

Parameters

in	*radians*	

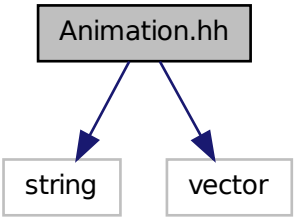
Returns

degrees

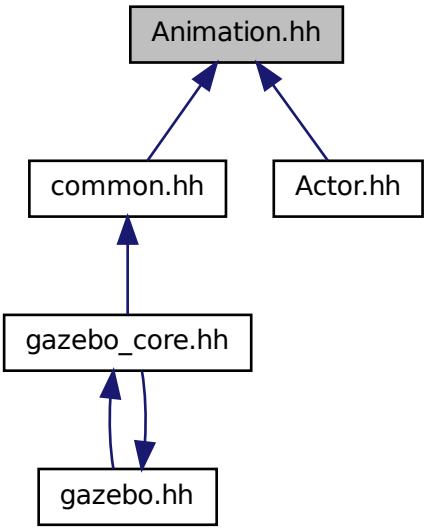
11.3 Animation.hh File Reference

```
#include <string>
#include <vector>
```

Include dependency graph for Animation.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Animation**

  Manages an animation, which is a collection of keyframes and the ability to interpolate between the keyframes.
- class **gazebo::common::NumericAnimation**

  A numeric animation.
- class **gazebo::common::PoseAnimation**

  A pose animation.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

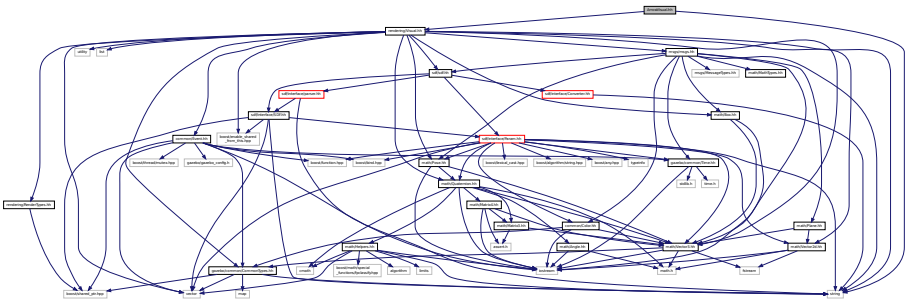
  Common namespace.
- namespace **gazebo::math**

  Math namespace.

11.4 ArrowVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
```

Include dependency graph for ArrowVisual.hh:



Classes

- class **gazebo::rendering::ArrowVisual**

  Basic arrow visualization.

Namespaces

- namespace **gazebo**

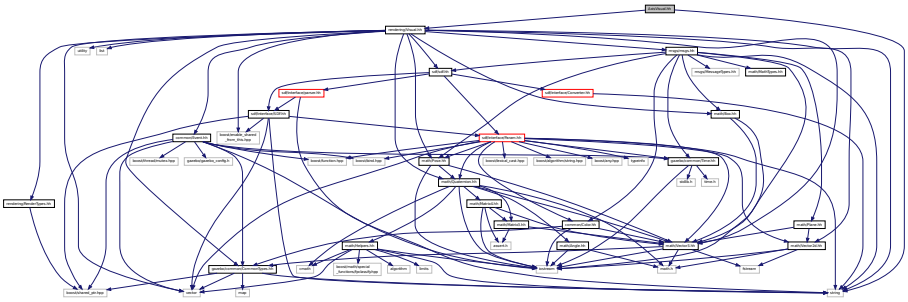
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **ogre**

11.5 AxisVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
```

Include dependency graph for AxisVisual.hh:



Classes

- class **gazebo::rendering::AxisVisual**

  Basic axis visualization.

Namespaces

- namespace **gazebo**

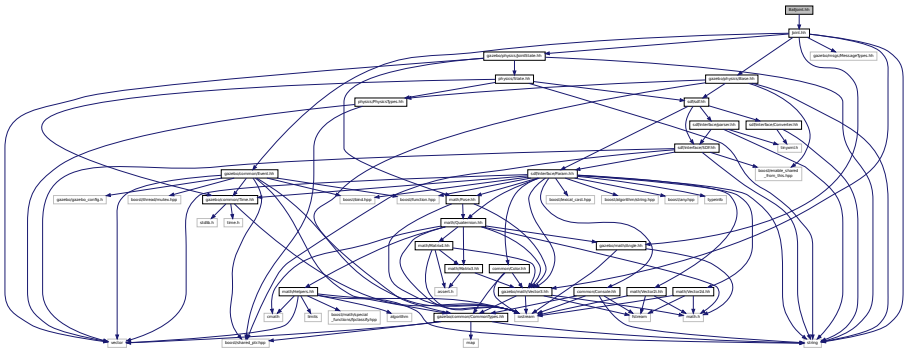
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

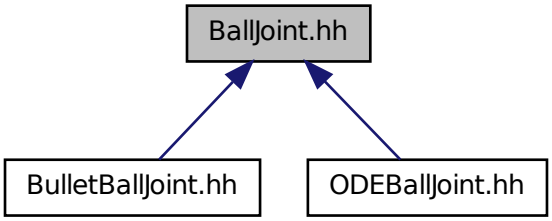
11.6 BallJoint.hh File Reference

```
#include "Joint.hh"
```

Include dependency graph for BallJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::BallJoint**< **T** >

  Base (p. *145*) *class for a ball joint.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

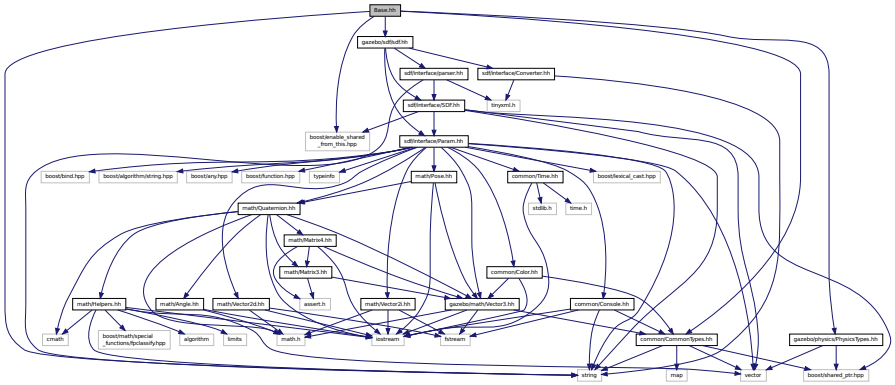
- namespace **gazebo::physics**

  namespace for physics

11.7 Base.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include "gazebo/sdf/sdf.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/physics/PhysicsTypes.hh"
```

Include dependency graph for Base.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::physics::Base**

  Base *(p. 145) class for most physics classes.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Variables

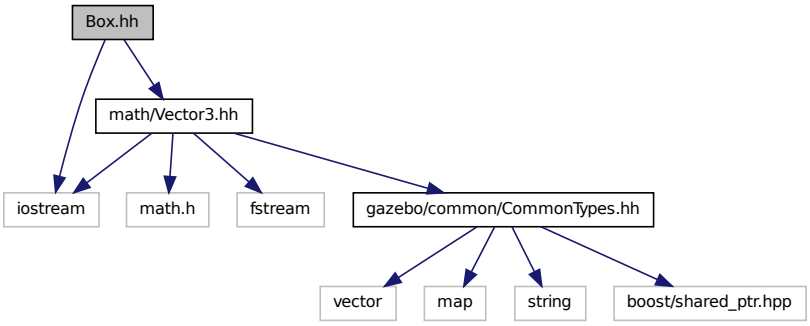
- static std::string **gazebo::physics::EntityTypename** [ ]

  String names for the different entity types.

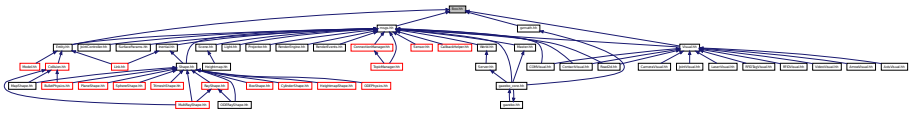
11.8 Box.hh File Reference

```
#include <iostream>
#include "math/Vector3.hh"
```

Include dependency graph for Box.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Box**

  Mathematical representation of a box and related functions.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

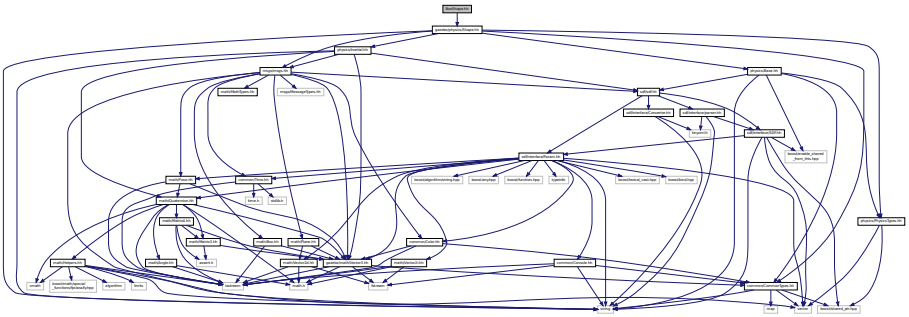
- namespace **gazebo::math**

  Math namespace.

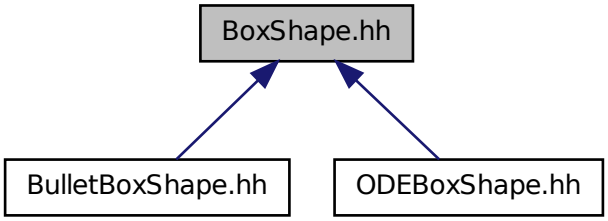
11.9 BoxShape.hh File Reference

```
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for BoxShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::BoxShape**

  Box geometry primitive.

Namespaces

- namespace **gazebo**

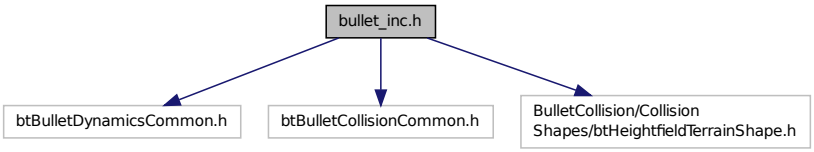
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

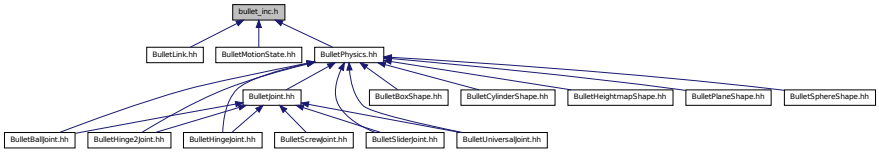
11.10 bullet_inc.h File Reference

```
#include <btBulletDynamicsCommon.h>
#include <btBulletCollisionCommon.h>
#include <BulletCollision/CollisionShapes/btHeightfieldTerrainShape.h>
```

Include dependency graph for bullet_inc.h:



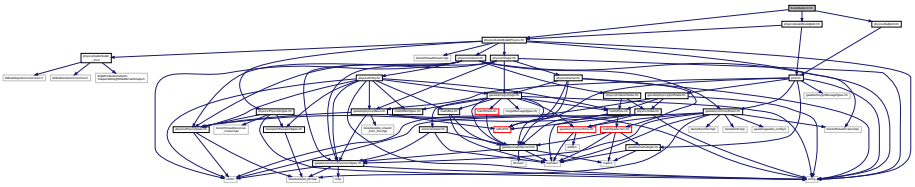
This graph shows which files directly or indirectly include this file:



11.11 BulletBallJoint.hh File Reference

```
#include "physics/BallJoint.hh"
#include "physics/bullet/BulletJoint.hh"
#include "physics/bullet/BulletPhysics.hh"
```

Include dependency graph for BulletBallJoint.hh:



Classes

- class **gazebo::physics::BulletBallJoint**

  BulletBallJoint *(p. 166) class models a ball joint in Bullet.*

Namespaces

- namespace **gazebo**

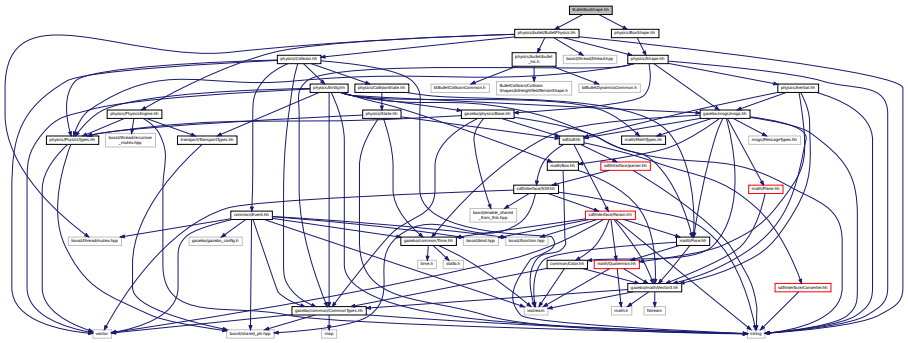
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.12 BulletBoxShape.hh File Reference

```
#include "physics/bullet/BulletPhysics.hh"
#include "physics/BoxShape.hh"
```

Include dependency graph for BulletBoxShape.hh:



Classes

- class **gazebo::physics::BulletBoxShape**

  Bullet box collision.

Namespaces

- namespace **gazebo**

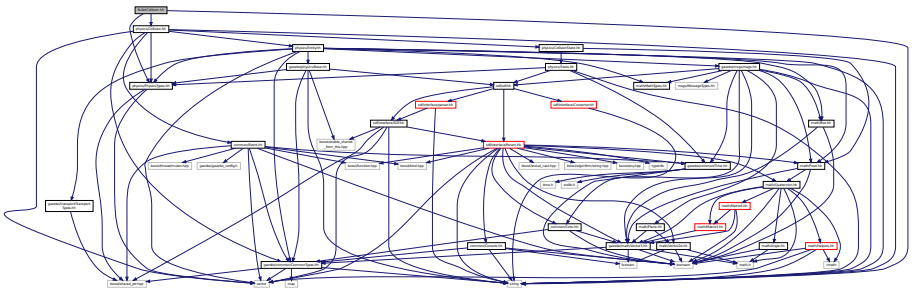
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.13 BulletCollision.hh File Reference

```
#include <string>
#include "physics/PhysicsTypes.hh"
#include "physics/Collision.hh"
```

Include dependency graph for BulletCollision.hh:



Classes

- class **gazebo::physics::BulletCollision**

  Bullet collisions.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

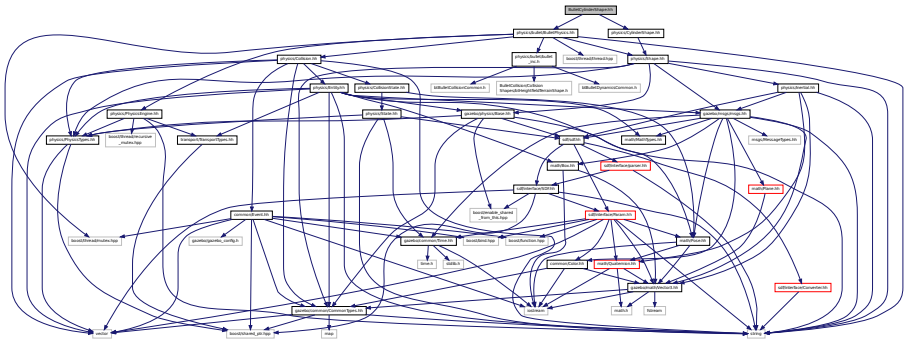
- namespace **gazebo::physics**

  namespace for physics

11.14 BulletCylinderShape.hh File Reference

```
#include "physics/bullet/BulletPhysics.hh"
#include "physics/CylinderShape.hh"
```

Include dependency graph for BulletCylinderShape.hh:



Classes

- class **gazebo::physics::BulletCylinderShape**

  Cylinder collision.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

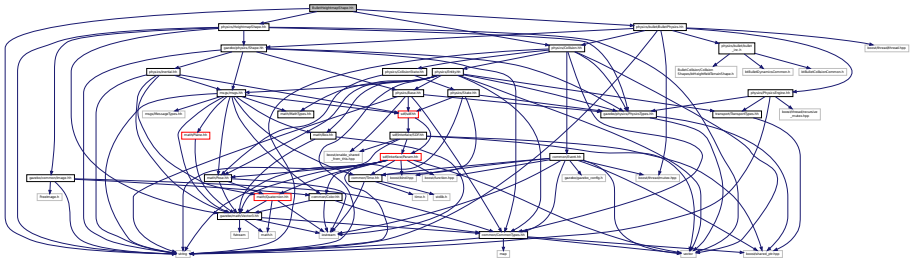
- namespace **gazebo::physics**

  namespace for physics

11.15 BulletHeightmapShape.hh File Reference

```
#include <string>
#include "physics/HeightmapShape.hh"
#include "physics/bullet/BulletPhysics.hh"
#include "physics/Collision.hh"
```

Include dependency graph for BulletHeightmapShape.hh:



Classes

- class **gazebo::physics::BulletHeightmapShape**

  Height map collision.

Namespaces

- namespace **gazebo**

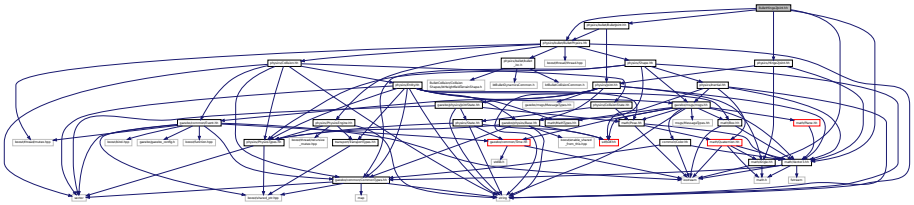
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.16 BulletHinge2Joint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/Hinge2Joint.hh"
#include "physics/bullet/BulletJoint.hh"
#include "physics/bullet/BulletPhysics.hh"
```

Include dependency graph for BulletHinge2Joint.hh:



Classes

- class **gazebo::physics::BulletHinge2Joint**

  A two axis hinge joint.

Namespaces

- namespace **gazebo**

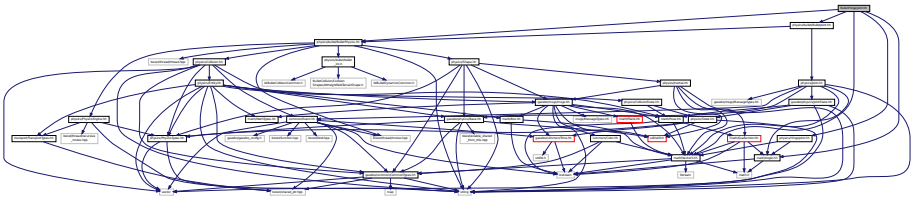
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.17 BulletHingeJoint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/HingeJoint.hh"
#include "physics/bullet/BulletJoint.hh"
#include "physics/bullet/BulletPhysics.hh"
```

Include dependency graph for BulletHingeJoint.hh:



Classes

- class **gazebo::physics::BulletHingeJoint**

  A single axis hinge joint.

Namespaces

- namespace **gazebo**

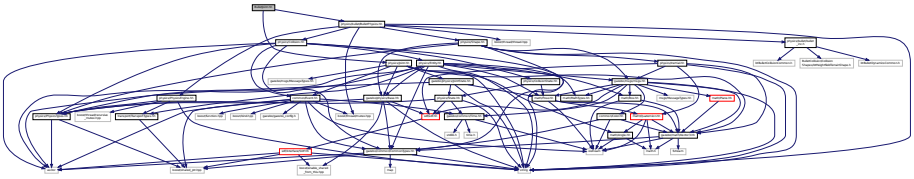
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

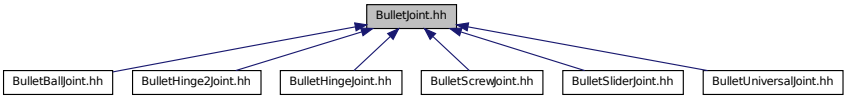
11.18 BulletJoint.hh File Reference

```
#include "physics/bullet/BulletPhysics.hh"
#include "physics/Joint.hh"
```

Include dependency graph for BulletJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::BulletJoint**

  Base *(p. 145) class for all joints.*

Namespaces

- namespace **gazebo**

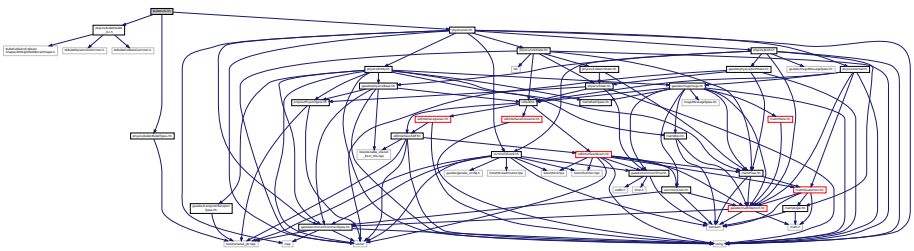
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.19 BulletLink.hh File Reference

```
#include "physics/bullet/bullet_inc.h"
#include "physics/bullet/BulletTypes.hh"
#include "physics/Link.hh"
```

Include dependency graph for BulletLink.hh:



Classes

- class **gazebo::physics::BulletLink**

  *Bullet **Link** (p. 454) class.*

Namespaces

- namespace **gazebo**

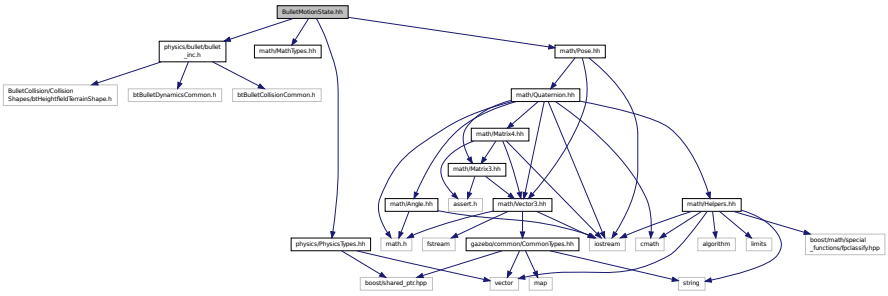
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.20 BulletMotionState.hh File Reference

```
#include "physics/bullet/bullet_inc.h"
#include "math/MathTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "math/Pose.hh"
```

Include dependency graph for BulletMotionState.hh:



Classes

- class **gazebo::physics::BulletMotionState**

  Bullet btMotionState encapsulation.

Namespaces

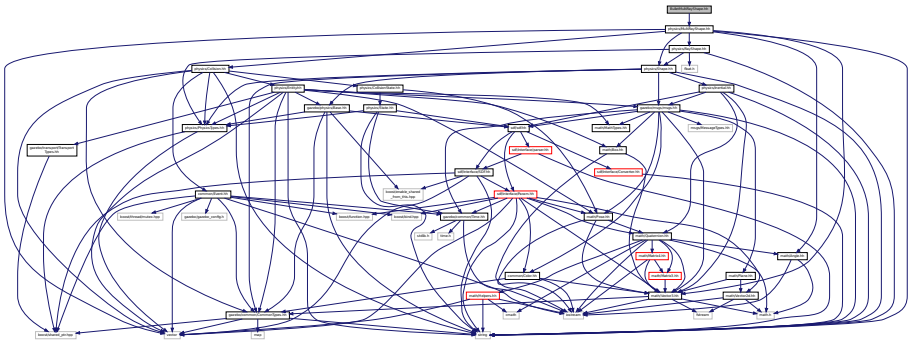
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.21 BulletMultiRayShape.hh File Reference

```
#include "physics/MultiRayShape.hh"
```
Include dependency graph for BulletMultiRayShape.hh:



Classes

- class **gazebo::physics::BulletMultiRayShape**

  Bullet specific version of ***MultiRayShape*** *(*p. *549).*

Namespaces

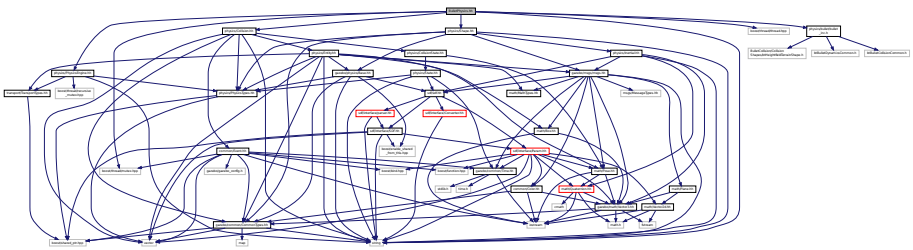
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

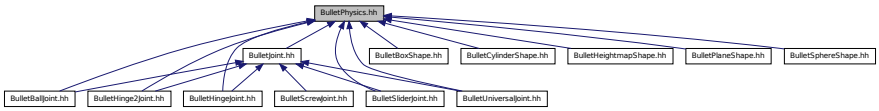
  namespace for physics

11.22 BulletPhysics.hh File Reference

```
#include <string>
#include <boost/thread/thread.hpp>
#include <boost/thread/mutex.hpp>
#include "physics/bullet/bullet_inc.h"
#include "physics/PhysicsEngine.hh"
#include "physics/Collision.hh"
#include "physics/Shape.hh"
```
Include dependency graph for BulletPhysics.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::BulletPhysics**

  Bullet physics engine.

Namespaces

- namespace **gazebo**

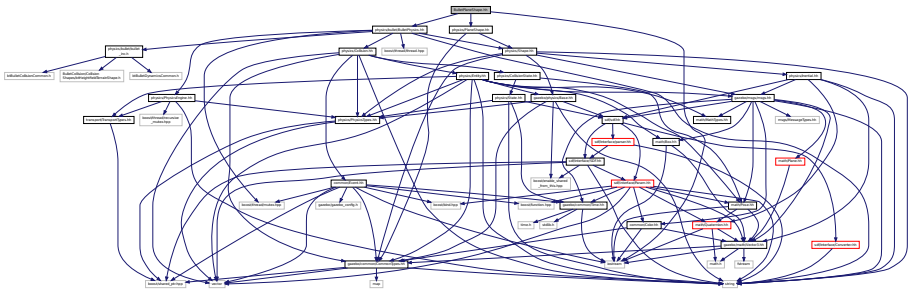
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.23 BulletPlaneShape.hh File Reference

```
#include <iostream>
#include "physics/bullet/BulletPhysics.hh"
#include "physics/PlaneShape.hh"
```

Include dependency graph for BulletPlaneShape.hh:



Classes

- class **gazebo::physics::BulletPlaneShape**

  Bullet collision for an infinite plane.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

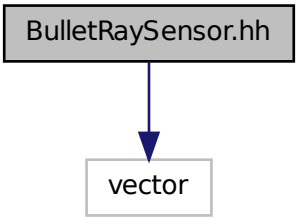
- namespace **gazebo::physics**

  namespace for physics

11.24 BulletRaySensor.hh File Reference

```
#include <vector>
```

Include dependency graph for BulletRaySensor.hh:



Classes

- class **gazebo::physics::BulletRaySensor**

  An Bullet Ray sensor.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

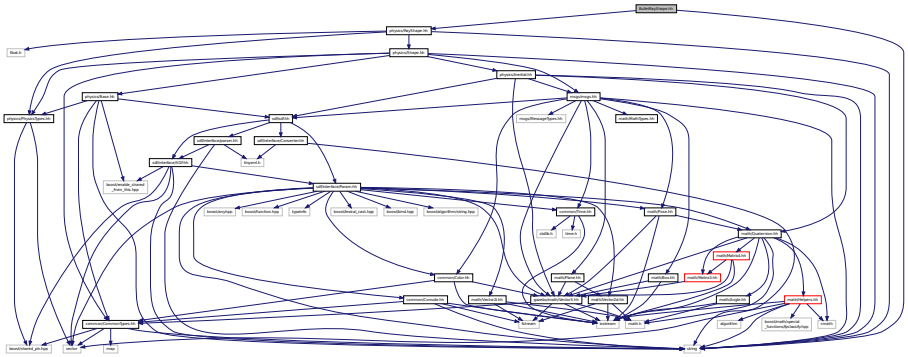
- namespace **gazebo::physics**

  namespace for physics

11.25 BulletRayShape.hh File Reference

```
#include <string>
#include "physics/RayShape.hh"
```

Include dependency graph for BulletRayShape.hh:



Classes

- class **gazebo::physics::BulletRayShape**

  Ray shape for bullet.

Namespaces

- namespace **gazebo**

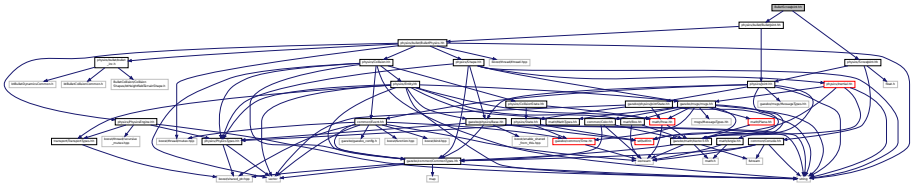
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.26 BulletScrewJoint.hh File Reference

```
#include "physics/bullet/BulletJoint.hh"
#include "physics/ScrewJoint.hh"
```

Include dependency graph for BulletScrewJoint.hh:



Classes

- class **gazebo::physics::BulletScrewJoint**

  A screw joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

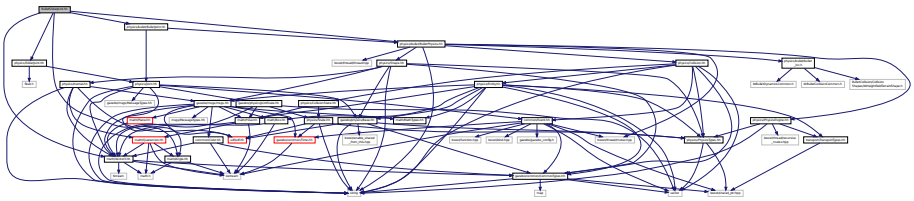
- namespace **gazebo::physics**

  namespace for physics

11.27 BulletSliderJoint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/bullet/BulletJoint.hh"
#include "physics/SliderJoint.hh"
#include "physics/bullet/BulletPhysics.hh"
```

Include dependency graph for BulletSliderJoint.hh:



Classes

- class **gazebo::physics::BulletSliderJoint**

  A slider joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

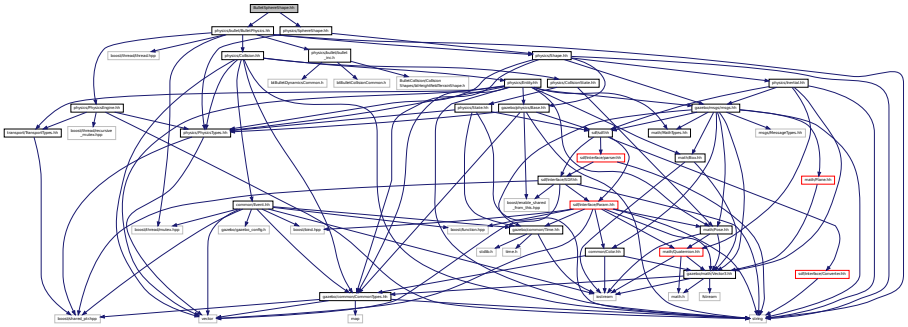
- namespace **gazebo::physics**

  namespace for physics

11.28 BulletSphereShape.hh File Reference

```
#include "physics/bullet/BulletPhysics.hh"
#include "physics/SphereShape.hh"
```

Include dependency graph for BulletSphereShape.hh:



Classes

- class **gazebo::physics::BulletSphereShape**

  Bullet sphere collision.

Namespaces

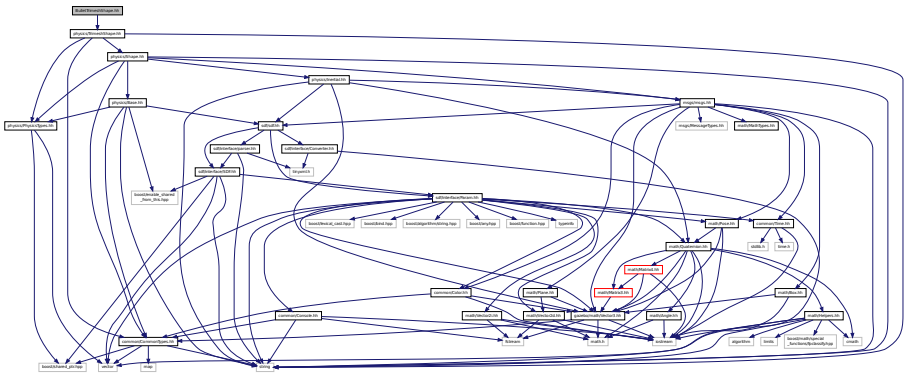
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.29 BulletTrimeshShape.hh File Reference

`#include "physics/TrimeshShape.hh"`
Include dependency graph for BulletTrimeshShape.hh:



Classes

- class **gazebo::physics::BulletTrimeshShape**

Triangle mesh collision.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

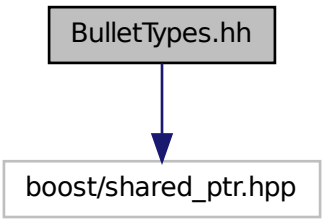
  namespace for physics

11.30 BulletTypes.hh File Reference

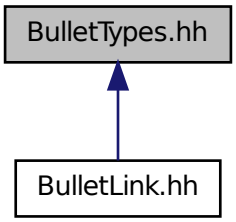
Bullet wrapper forward declarations and typedefs.

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for BulletTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Typedefs

- typedef BulletCollision ∗ **gazebo::physics::BulletCollisionPtr**
- typedef BulletLink ∗ **gazebo::physics::BulletLinkPtr**
- typedef BulletPhysics ∗ **gazebo::physics::BulletPhysicsPtr**
- typedef BulletRayShape ∗ **gazebo::physics::BulletRayShapePtr**

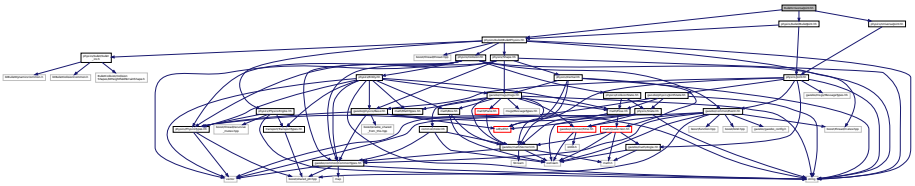
11.30.1 Detailed Description

Bullet wrapper forward declarations and typedefs.

11.31 BulletUniversalJoint.hh File Reference

```
#include "physics/UniversalJoint.hh"
#include "physics/bullet/BulletJoint.hh"
#include "physics/bullet/BulletPhysics.hh"
```

Include dependency graph for BulletUniversalJoint.hh:



Classes

- class **gazebo::physics::BulletUniversalJoint**

  A bullet universal joint class.

Namespaces

- namespace **gazebo**

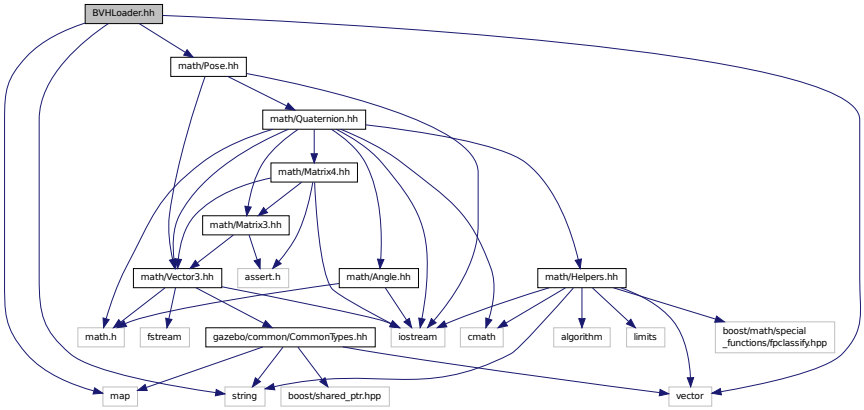
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

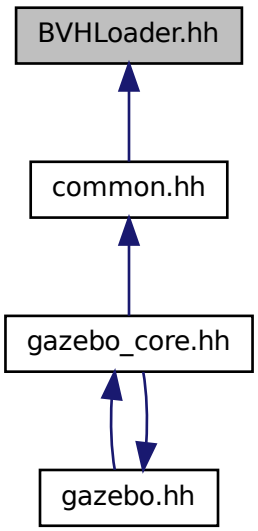
11.32 BVHLoader.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include "math/Pose.hh"
```

Include dependency graph for BVHLoader.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::BVHLoader**

  Handles loading BVH animation files.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Macros

- #define **X_POSITION** 0
- #define **X_ROTATION** 3
- #define **Y_POSITION** 1
- #define **Y_ROTATION** 4
- #define **Z_POSITION** 2
- #define **Z_ROTATION** 5

11.32.1 Macro Definition Documentation

11.32.1.1 #define X_POSITION 0

11.32.1.2 #define X_ROTATION 3

11.32.1.3 #define Y_POSITION 1

11.32.1.4 #define Y_ROTATION 4

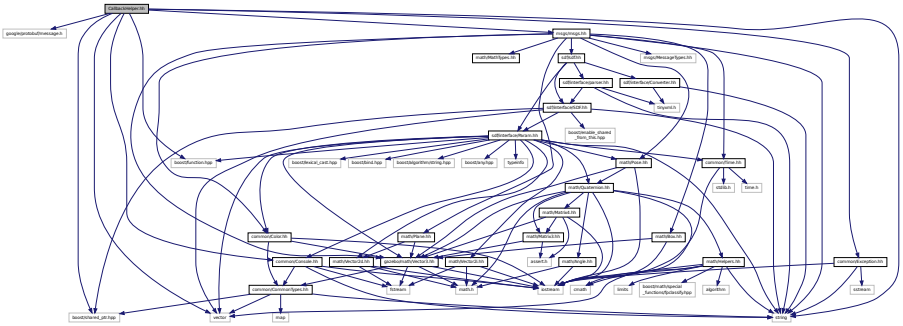
11.32.1.5 #define Z_POSITION 2

11.32.1.6 #define Z_ROTATION 5

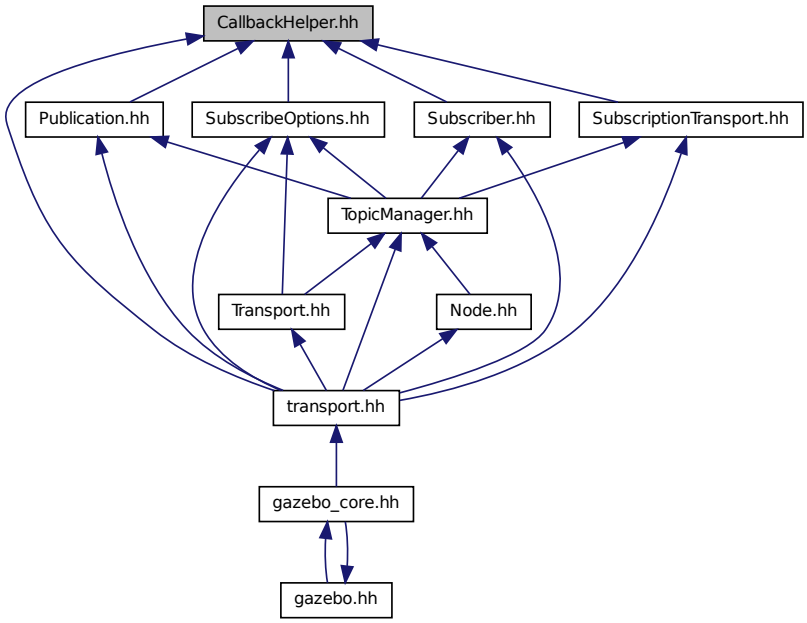
11.33 CallbackHelper.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <vector>
#include <string>
#include "common/Console.hh"
#include "msgs/msgs.hh"
#include "common/Exception.hh"
```

Include dependency graph for CallbackHelper.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::CallbackHelper**

  A helper class to handle callbacks when messages arrive.

- class **gazebo::transport::CallbackHelperT< M >**

  Callback helper Template.

- class **gazebo::transport::DebugCallbackHelper**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::transport**

  Transport namespace.

Typedefs

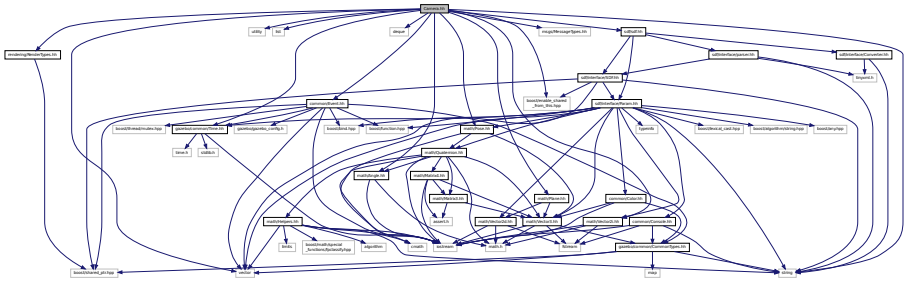
- typedef CallbackHelper ∗ **gazebo::transport::CallbackHelperPtr**

  *boost shared pointer to **transport::CallbackHelper** (p. 230)*

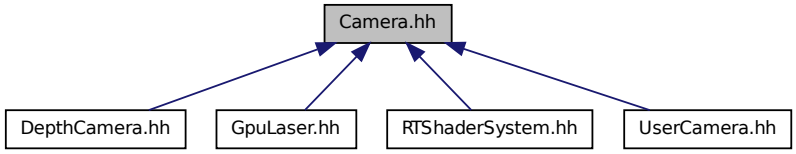
11.34 Camera.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <list>
#include <vector>
#include <deque>
#include "common/Event.hh"
#include "common/Time.hh"
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "math/Plane.hh"
#include "math/Vector2i.hh"
#include "msgs/MessageTypes.hh"
#include "rendering/RenderTypes.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for Camera.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Camera**

  Basic camera sensor.

Namespaces

- namespace **gazebo**

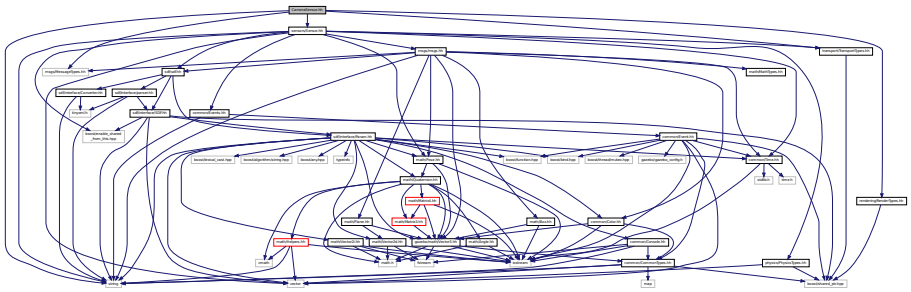
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.35 CameraSensor.hh File Reference

```
#include <string>
#include "sensors/Sensor.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for CameraSensor.hh:



Classes

- class **gazebo::sensors::CameraSensor**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

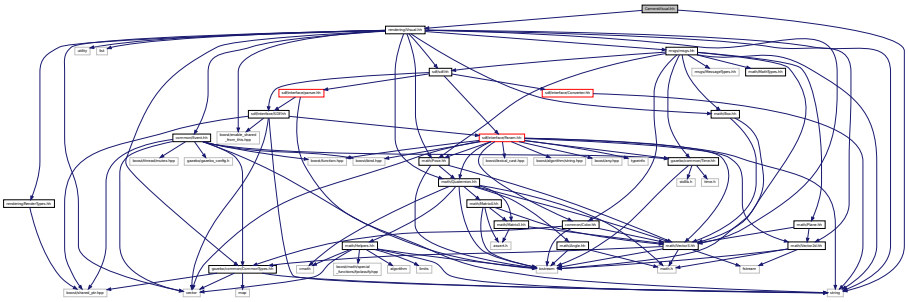
- namespace **gazebo::sensors**

  Sensors namespace.

11.36 CameraVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
```

Include dependency graph for CameraVisual.hh:



Classes

- class **gazebo::rendering::CameraVisual**

  Basic camera visualization.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

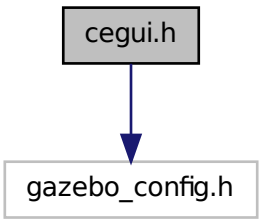
- namespace **gazebo::rendering**

  Rendering namespace.

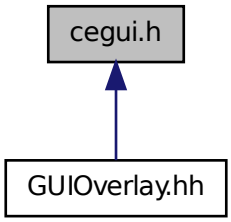
11.37 cegui.h File Reference

```
#include "gazebo_config.h"
```

Include dependency graph for cegui.h:



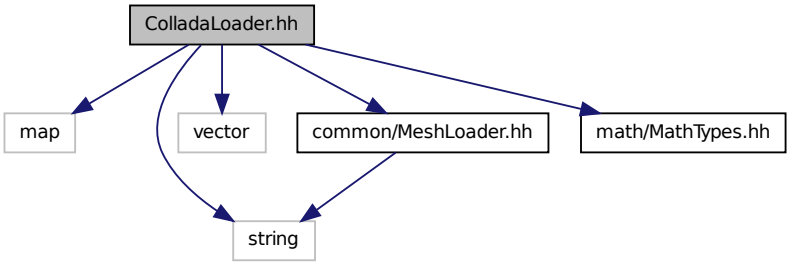
This graph shows which files directly or indirectly include this file:



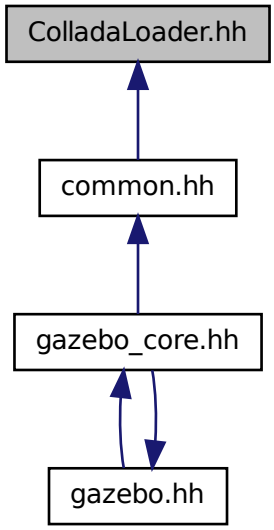
11.38 ColladaLoader.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include "common/MeshLoader.hh"
#include "math/MathTypes.hh"
```

Include dependency graph for ColladaLoader.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::ColladaLoader**

  Class used to load Collada mesh files.

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

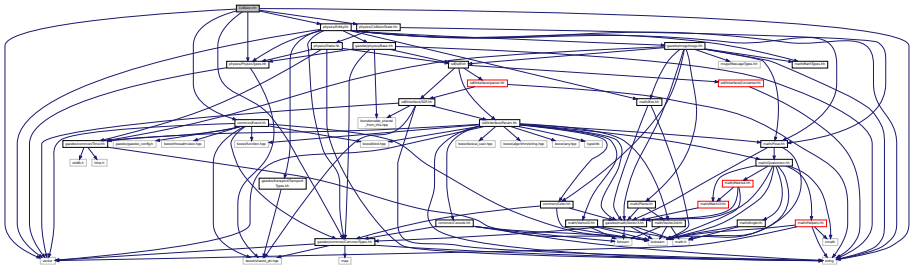
- namespace **gazebo::common**

  Common namespace.

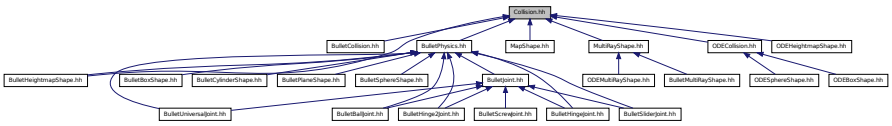
11.39 Collision.hh File Reference

```
#include <string>
#include <vector>
#include "common/Event.hh"
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/CollisionState.hh"
#include "physics/Entity.hh"
```

Include dependency graph for Collision.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Collision**

  Base *(p. 145) class for all collision entities.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

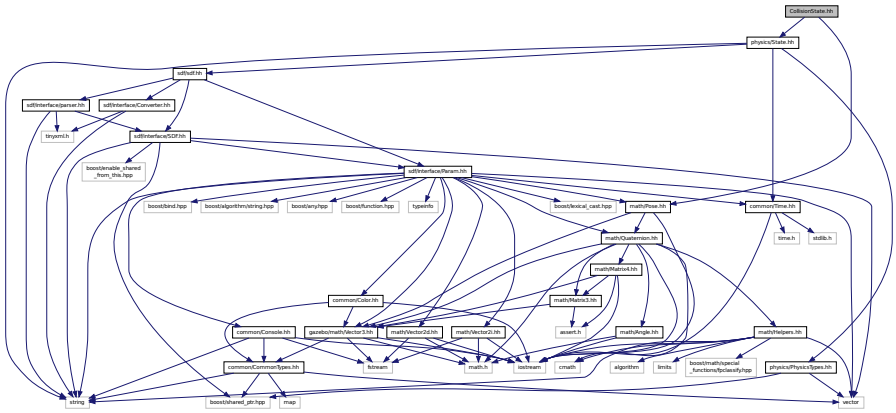
- namespace **gazebo::physics**

  namespace for physics

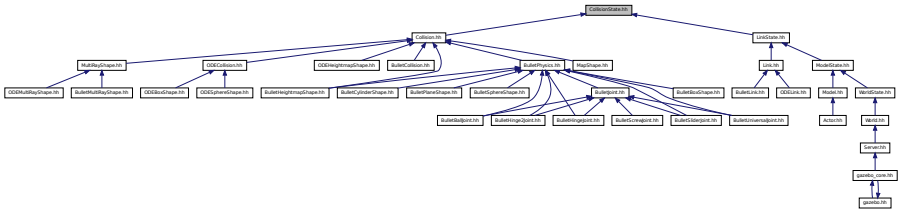
11.40 CollisionState.hh File Reference

```
#include "physics/State.hh"
#include "math/Pose.hh"
```

Include dependency graph for CollisionState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::CollisionState**

  *Store state information of a **physics::Collision*** (p. *262*) *object.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

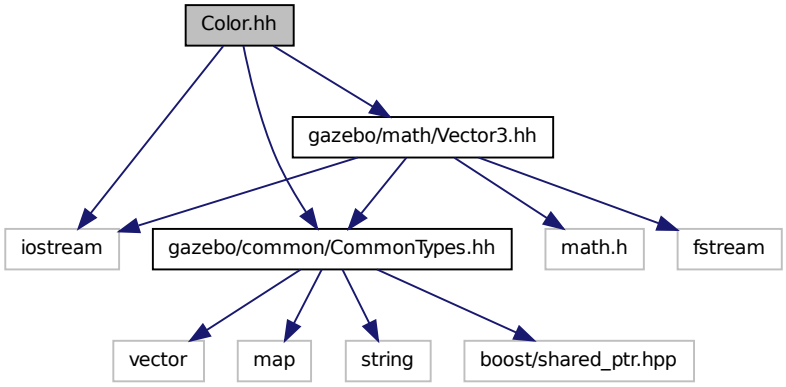
  namespace for physics

11.41 Color.hh File Reference

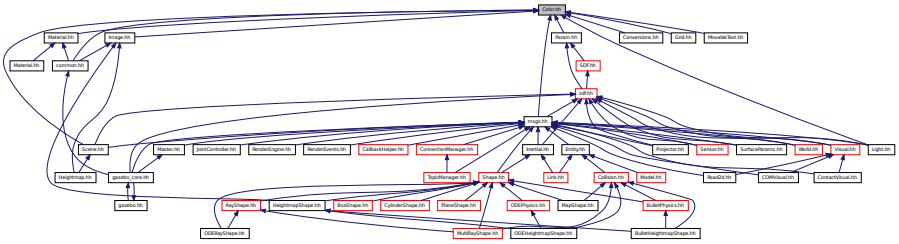
```
#include <iostream>
```

```
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/math/Vector3.hh"
```

Include dependency graph for Color.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Color**

  Defines a color.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

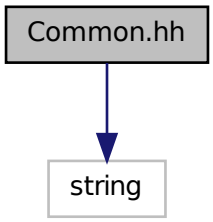
- namespace **gazebo::common**

  Common namespace.

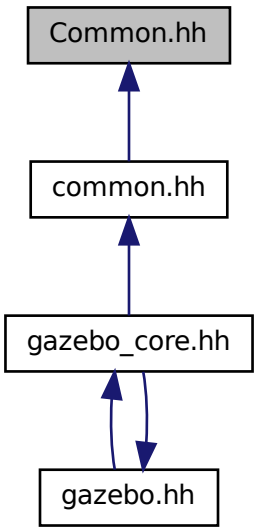
11.42 Common.hh File Reference

```
#include <string>
```

Include dependency graph for Common.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

Common namespace.

Functions

- void **gazebo::common::add_search_path_suffix** (const std::string &_suffix)

  *add path prefix to **common::SystemPaths** (p. 834)*
- std::string **gazebo::common::find_file** (const std::string &_file, bool _searchLocalPath=true)

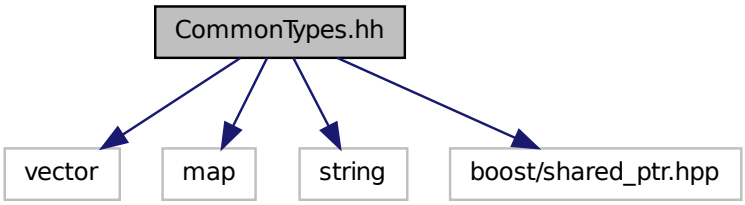
  *search for file in **common::SystemPaths** (p. 834)*
- std::string **gazebo::common::find_file_path** (const std::string &_file)

  *search for a file in **common::SystemPaths** (p. 834)*

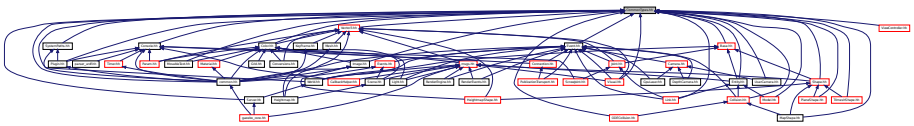
11.43 CommonTypes.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for CommonTypes.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **ParamT**< **T** >

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

- namespace **gazebo::event**

  Event *(p. 350) namespace.*

Macros

- #define **GAZEBO_DEPRECATED**
- #define **GAZEBO_FORCEINLINE**
- #define **NULL** 0

Typedefs

- typedef Animation * **gazebo::common::AnimationPtr**
- typedef std::vector < ConnectionPtr > **gazebo::event::Connection_V**
- typedef Connection * **gazebo::event::ConnectionPtr**
- typedef GUIPlugin * **gazebo::GUIPluginPtr**
- typedef ModelPlugin * **gazebo::ModelPluginPtr**
- typedef NumericAnimation * **gazebo::common::NumericAnimationPtr**
- typedef std::vector < common::Param * > **gazebo::common::Param_V**
- typedef PoseAnimation * **gazebo::common::PoseAnimationPtr**
- typedef SensorPlugin * **gazebo::SensorPluginPtr**
- typedef std::map< std::string, std::string > **gazebo::common::StrStr_M**
- typedef SystemPlugin * **gazebo::SystemPluginPtr**
- typedef VisualPlugin * **gazebo::VisualPluginPtr**
- typedef WorldPlugin * **gazebo::WorldPluginPtr**

11.43.1 Macro Definition Documentation

11.43.1.1 #define GAZEBO_DEPRECATED

11.43.1.2 #define GAZEBO_FORCEINLINE

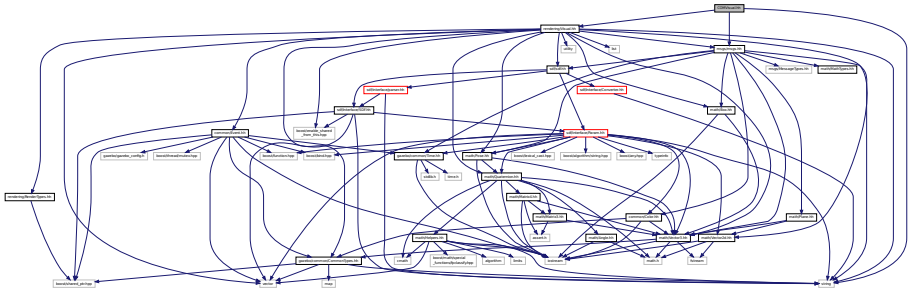
11.43.1.3 #define NULL 0

Referenced by gazebo::transport::TopicManager::Advertise(), gazebo::PluginT< ModelPlugin >::Create(), gazebo::physics::ODEPlaneShape::CreatePlane(), gazebo::event::EventT< T >::Disconnect(), gazebo::transport::CallbackHelperT< M >::GetMsgType(), gazebo::transport::SubscribeOptions::Init(), gazebo::physics::ODESphereShape::SetRadius(), gazebo::physics::ODECylinderShape::SetSize(), and gazebo::physics::ODEBoxShape::SetSize().

11.44 COMVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/msgs.hh"
```

Include dependency graph for COMVisual.hh:



Classes

- class **gazebo::rendering::COMVisual**

  Basic Center of Mass visualization.

Namespaces

- namespace **gazebo**

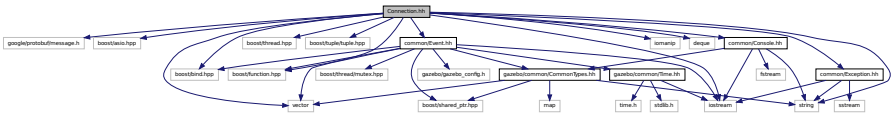
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **ogre**

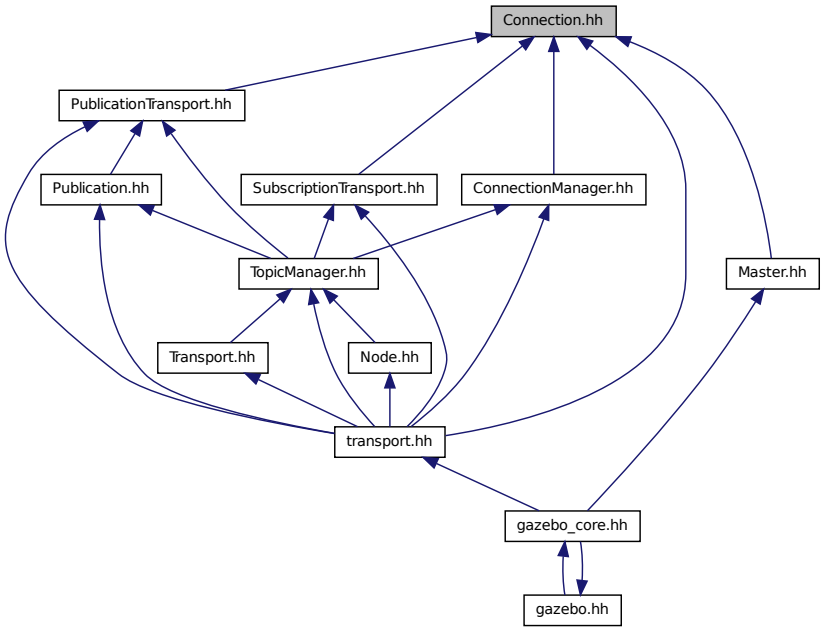
11.45 Connection.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/asio.hpp>
#include <boost/bind.hpp>
#include <boost/function.hpp>
#include <boost/thread.hpp>
#include <boost/tuple/tuple.hpp>
#include <string>
#include <vector>
#include <iostream>
#include <iomanip>
#include <deque>
#include "common/Event.hh"
#include "common/Console.hh"
#include "common/Exception.hh"
```

Include dependency graph for Connection.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Connection**

  *TCP/IP **Connection** (p. 288).*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::transport**

  Transport namespace.

Macros

- #define **HEADER_LENGTH** 8

Typedefs

- typedef Connection ∗ **gazebo::transport::ConnectionPtr**

Functions

- bool **gazebo::transport::is_stopped** ()

  Return true if the transport system is stopped.

11.45.1 Macro Definition Documentation

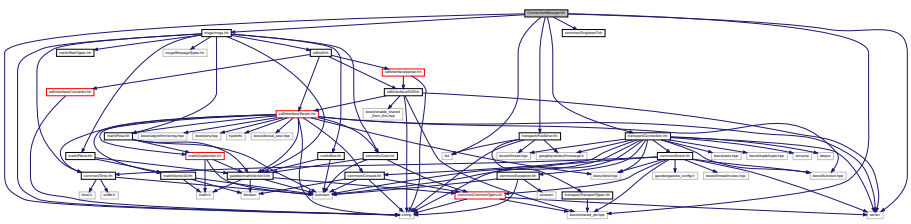
11.45.1.1 #define HEADER_LENGTH 8

Referenced by gazebo::transport::Connection::AsyncRead().

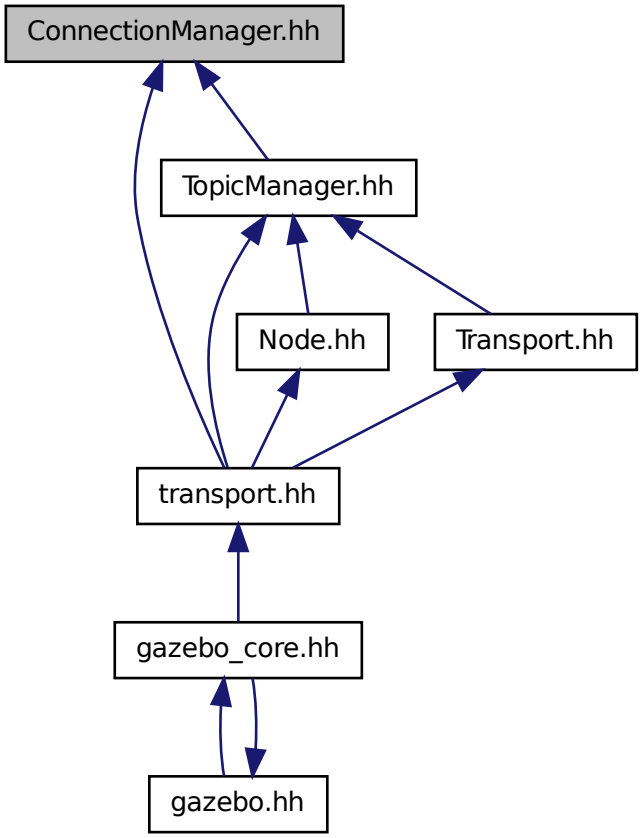
11.46 ConnectionManager.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include <list>
#include <vector>
#include "msgs/msgs.hh"
#include "common/SingletonT.hh"
#include "transport/Publisher.hh"
#include "transport/Connection.hh"
```

Include dependency graph for ConnectionManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::ConnectionManager**

  Manager of connections.

Namespaces

- namespace **gazebo**

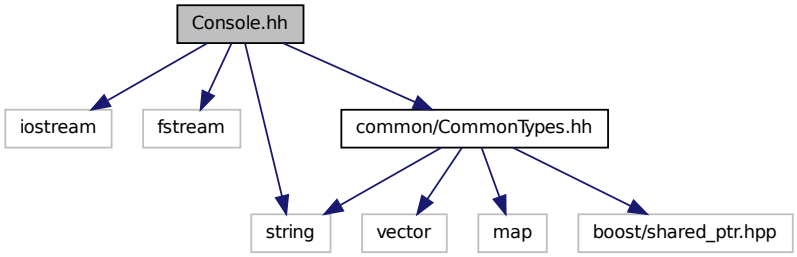
  Forward declarations for the common classes.
- namespace **gazebo::transport**

  Transport namespace.

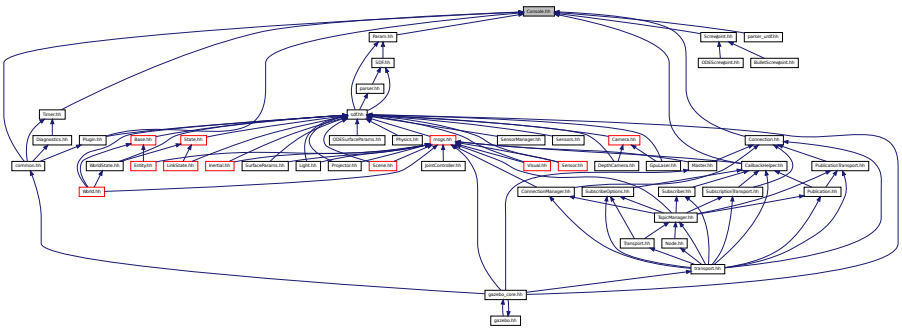
11.47 Console.hh File Reference

```
#include <iostream>
#include <fstream>
#include <string>
#include "common/CommonTypes.hh"
```

Include dependency graph for Console.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Console**

  Message, error, warning, and logging functionality.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

Macros

- #define **gzclr_end** "\033[0m"

  end marker

- #define **gzclr_start**(clr) "\033[1;33m"

  start marker

- #define **gzdbg** (**gazebo::common::Console::Instance**()->ColorMsg("Dbg", 36))

  Output a debug message.

- #define **gzerr**

  Output an error message.

- #define **gzlog**

  Log a message.

- #define **gzmsg** (**gazebo::common::Console::Instance**()->ColorMsg("Msg", 32))

  Output a message.

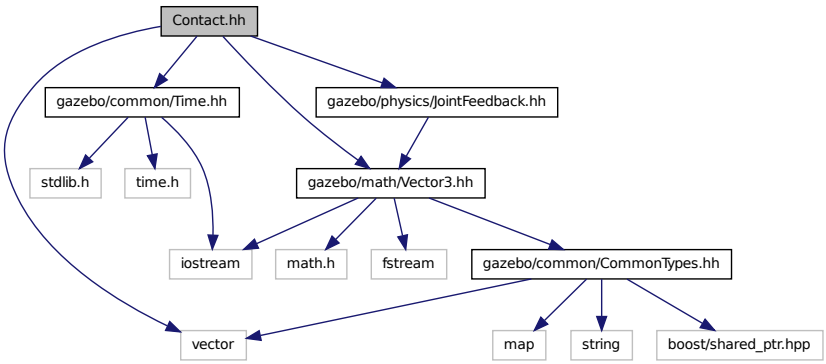
- #define **gzwarn**

  Output a warning message.

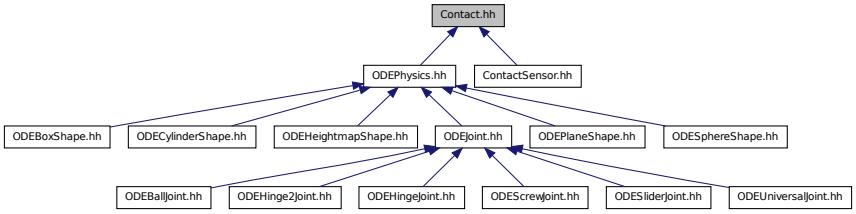
11.48 Contact.hh File Reference

```
#include <vector>
#include "gazebo/common/Time.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/JointFeedback.hh"
```

Include dependency graph for Contact.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Contact**

  A contact between two collisions.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Macros

- #define **MAX_COLLIDE_RETURNS** 250
- #define **MAX_CONTACT_JOINTS** 32

11.48.1 Macro Definition Documentation

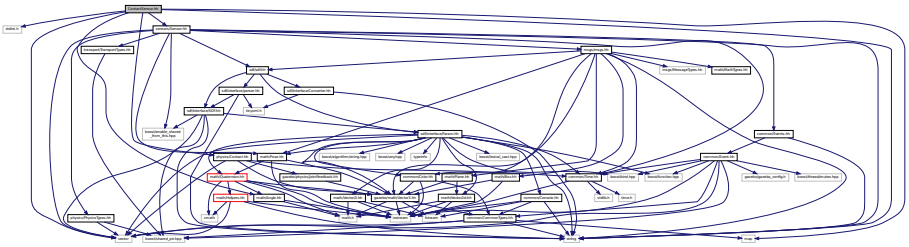
11.48.1.1 #define MAX_COLLIDE_RETURNS 250

11.48.1.2 #define MAX_CONTACT_JOINTS 32

11.49 ContactSensor.hh File Reference

```
#include <stdint.h>
#include <vector>
#include <map>
#include <string>
#include "math/Angle.hh"
#include "sensors/Sensor.hh"
#include "physics/Contact.hh"
```

Include dependency graph for ContactSensor.hh:



Classes

- class **gazebo::sensors::ContactSensor**

Namespaces

- namespace **gazebo**

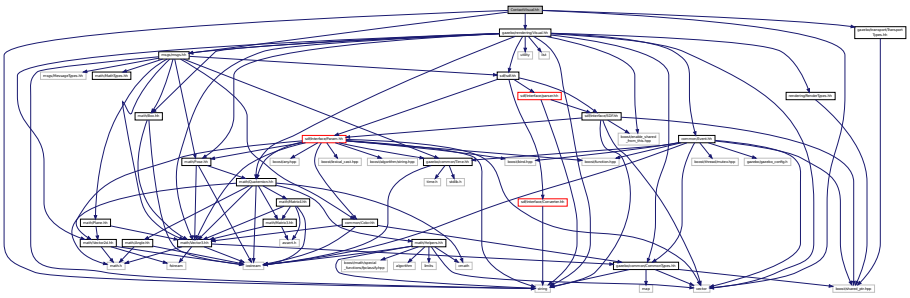
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.50 ContactVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
```

Include dependency graph for ContactVisual.hh:



Classes

- class **gazebo::rendering::ContactVisual**

  Contact visualization.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

  Rendering namespace.

- namespace **Ogre**

11.51 Conversions.hh File Reference

```
#include "rendering/ogre_gazebo.h"
#include "common/Color.hh"
#include "math/Vector3.hh"
#include "math/Quaternion.hh"
```

Include dependency graph for Conversions.hh:

Classes

- class **gazebo::rendering::Conversions**

  Conversions (p. *305*) ***Conversions.hh*** (p. *1018*) ***rendering/Conversions.hh*** (p. *1018*).

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

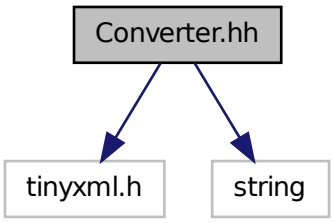
- namespace **gazebo::rendering**

  Rendering namespace.

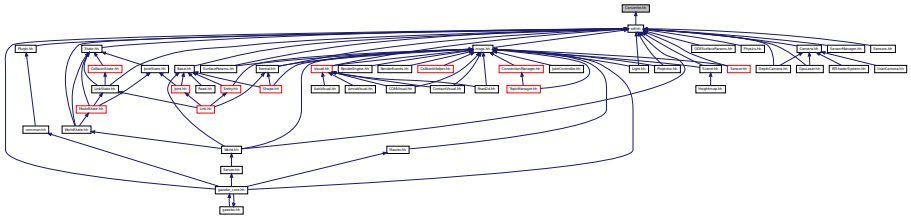
11.52 Converter.hh File Reference

```
#include <tinyxml.h>
#include <string>
```

Include dependency graph for Converter.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Converter**

  *Convert from one version of **SDF** (p. 762) to another.*

Namespaces

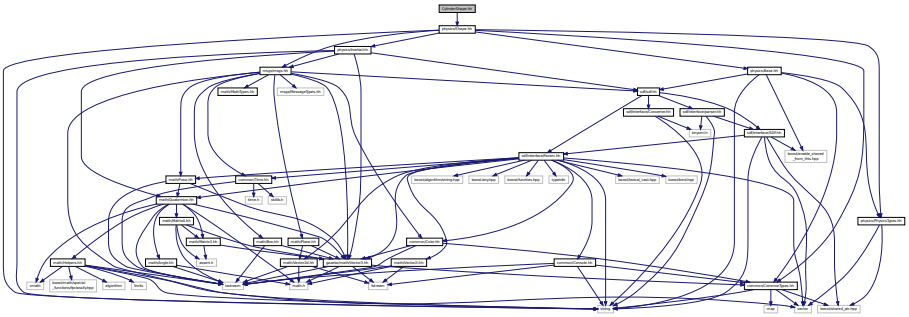
- namespace **sdf**

  namespace for Simulation Description Format parser

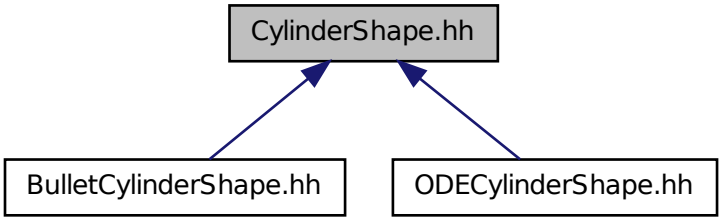
11.53 CylinderShape.hh File Reference

```
#include "physics/Shape.hh"
```

Include dependency graph for CylinderShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::CylinderShape**

  Cylinder collision.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

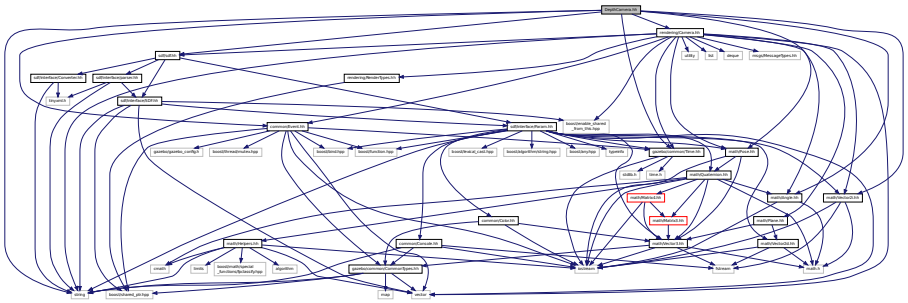
  namespace for physics

11.54 DepthCamera.hh File Reference

```
#include <string>
```

```
#include "common/Event.hh"
#include "common/Time.hh"
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "math/Vector2i.hh"
#include "sdf/sdf.hh"
#include "rendering/Camera.hh"
```

Include dependency graph for DepthCamera.hh:



Classes

- class **gazebo::rendering::DepthCamera**

  Depth camera used to render depth data into an image buffer.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

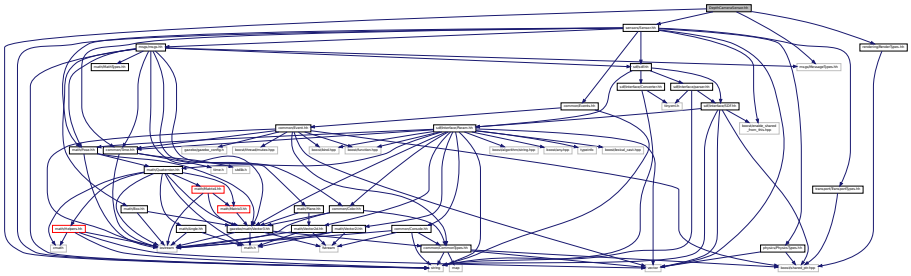
  Rendering namespace.

- namespace **Ogre**

11.55 DepthCameraSensor.hh File Reference

```
#include <string>
#include "sensors/Sensor.hh"
#include "msgs/MessageTypes.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for DepthCameraSensor.hh:



Classes

- class **gazebo::sensors::DepthCameraSensor**

Namespaces

- namespace **gazebo**

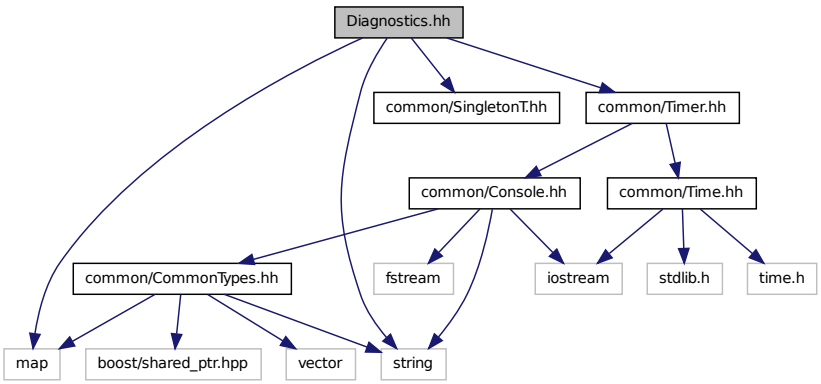
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

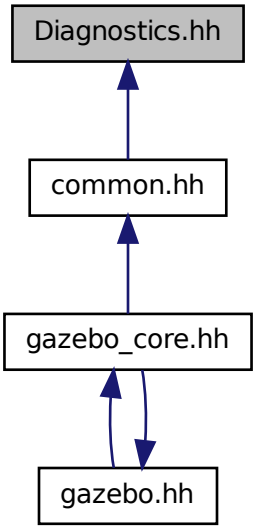
11.56 Diagnostics.hh File Reference

```
#include <map>
#include <string>
#include "common/SingletonT.hh"
#include "common/Timer.hh"
```

Include dependency graph for Diagnostics.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::DiagnosticManager**

  A diagnostic manager class.
- class **gazebo::common::DiagnosticTimer**

  A timer designed for diagnostics.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Macros

- #define **DIAG_TIMER**(name) DiagnosticManager::Instance()->CreateTimer(name);

  Create an instance of common::DiagnosticManager.

Typedefs

- typedef DiagnosticTimer * **gazebo::common::DiagnosticTimerPtr**

11.57 DynamicLines.hh File Reference

```
#include <vector>
#include <string>
#include "math/Vector3.hh"
#include "rendering/DynamicRenderable.hh"
```

Include dependency graph for DynamicLines.hh:

Classes

- class **gazebo::rendering::DynamicLines**

  Class for drawing lines that can change.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

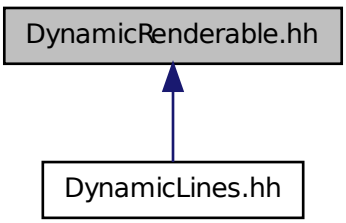
  Rendering namespace.

11.58 DynamicRenderable.hh File Reference

```
#include "rendering/ogre_gazebo.h"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for DynamicRenderable.hh:

This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::DynamicRenderable**

  Abstract base class providing mechanisms for dynamically growing hardware buffers.

Namespaces

- namespace **gazebo**

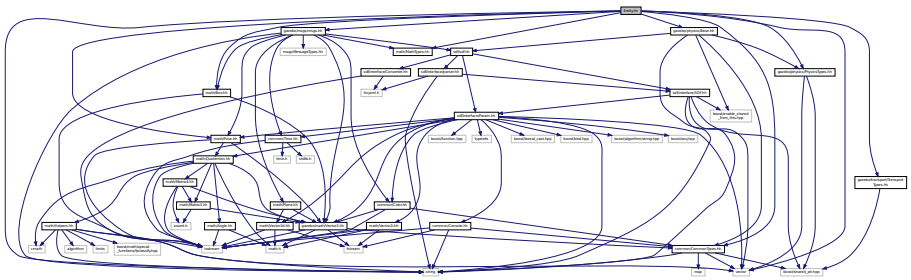
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

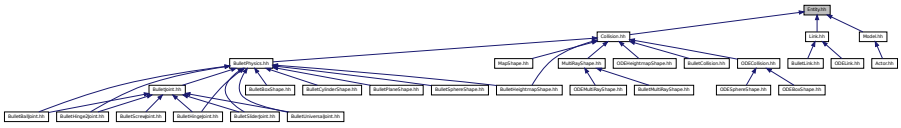
11.59 Entity.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/common/CommonTypes.hh"
#include "gazebo/math/MathTypes.hh"
#include "gazebo/math/Box.hh"
#include "gazebo/math/Pose.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Base.hh"
```

Include dependency graph for Entity.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Entity**

  Base *(p. 145) class for all physics objects in Gazebo.*

Namespaces

- namespace **boost**
- namespace **gazebo**

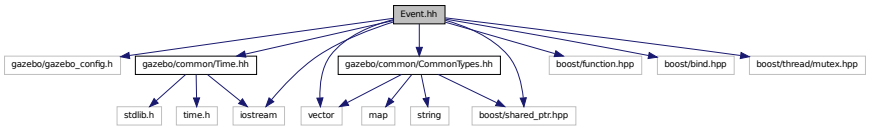
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.60 Event.hh File Reference

```
#include <gazebo/gazebo_config.h>
#include <gazebo/common/Time.hh>
#include <gazebo/common/CommonTypes.hh>
#include <boost/function.hpp>
#include <boost/bind.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/thread/mutex.hpp>
#include <iostream>
#include <vector>
```

Include dependency graph for Event.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::event::Connection**

  A class that encapsulates a connection.
- class **gazebo::event::Event**

  Base class for all events.
- class **gazebo::event::EventT< T >**

  A class for event processing.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

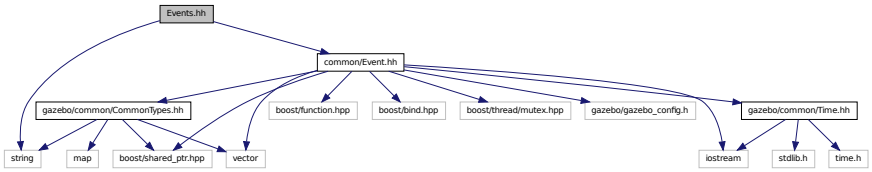
- namespace **gazebo::event**

  Event *(p. 350) namespace.*

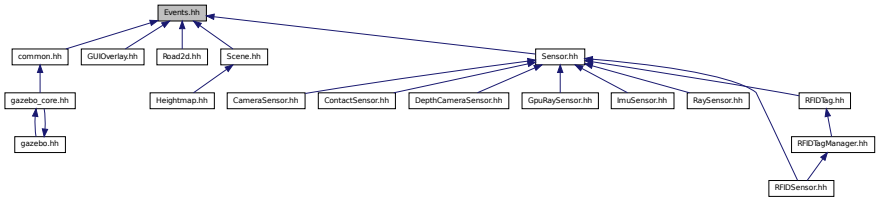
11.61 Events.hh File Reference

```
#include <string>
#include "common/Event.hh"
```

Include dependency graph for Events.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::event::Events**

  An ***Event*** *(p. 350) class to get notifications for simulator events.*

Namespaces

- namespace **gazebo**

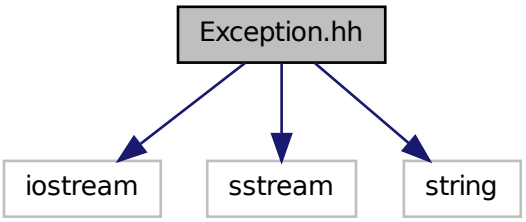
  Forward declarations for the common classes.
- namespace **gazebo::event**

  Event *(p. 350) namespace.*

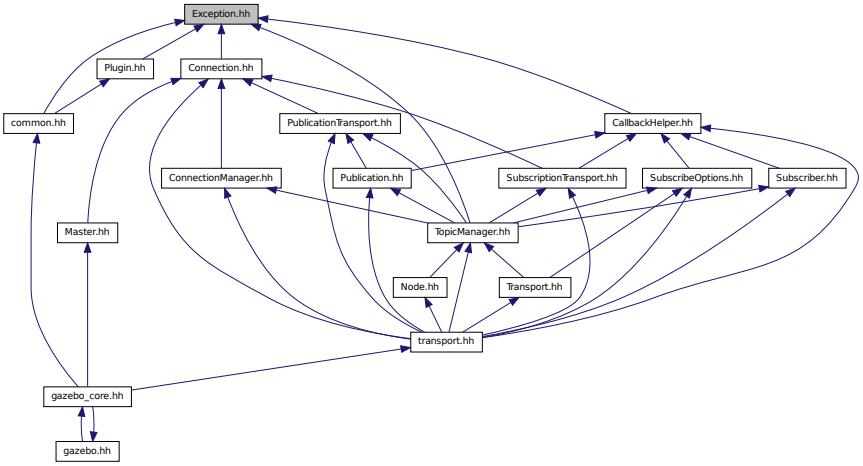
11.62 Exception.hh File Reference

```
#include <iostream>
#include <sstream>
#include <string>
```

Include dependency graph for Exception.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Exception**

  Class for generating exceptions.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Macros

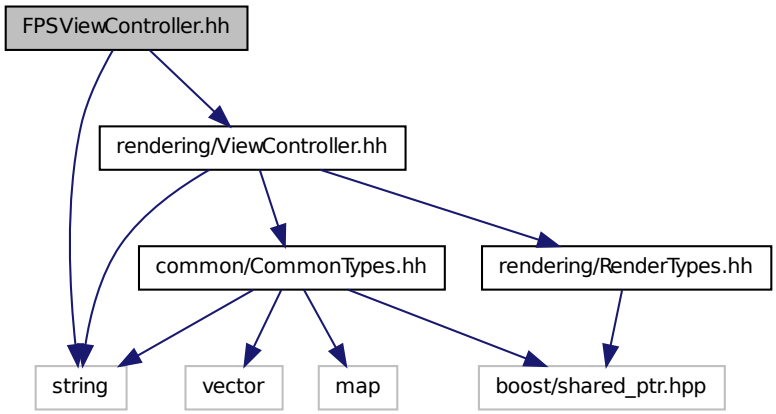
- #define **gzthrow**(msg)

  This macro logs an error to the throw stream and throws an exception that contains the file name and line number.

11.63 FPSViewController.hh File Reference

```
#include <string>
#include "rendering/ViewController.hh"
```

Include dependency graph for FPSViewController.hh:



Classes

- class **gazebo::rendering::FPSViewController**

  First Person Shooter style view controller.

Namespaces

- namespace **gazebo**

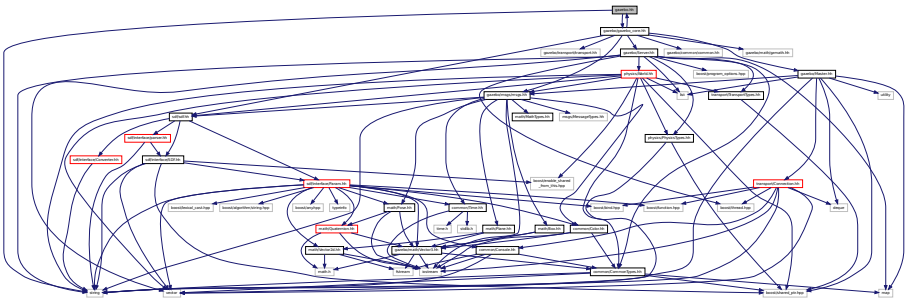
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

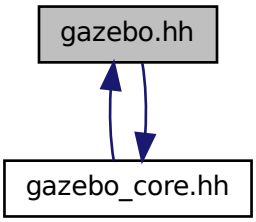
11.64 gazebo.hh File Reference

```
#include <gazebo/gazebo_core.hh>
#include <string>
```

Include dependency graph for gazebo.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

Functions

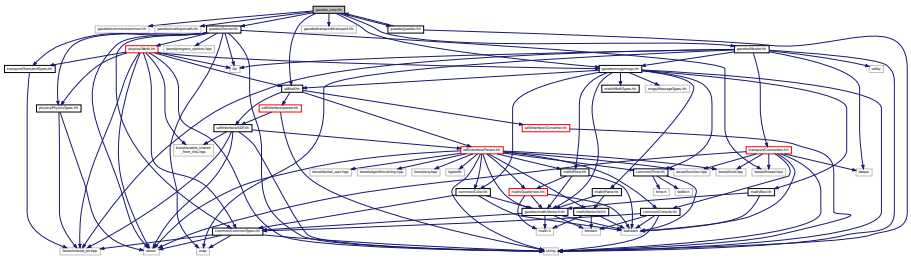
- void **gazebo::add_plugin** (const std::string &_filename)
- std::string **gazebo::find_file** (const std::string &_file)

  Find a file in the gazebo search paths.
- void **gazebo::fini** ()
- bool **gazebo::init** ()
- bool **gazebo::load** (int argc=0, char ∗∗argv=0)
- void **gazebo::print_version** ()
- void **gazebo::run** ()
- void **gazebo::stop** ()

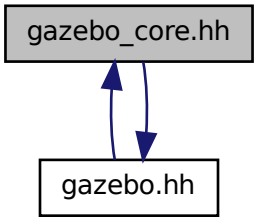
11.65 gazebo_core.hh File Reference

```
#include <gazebo/common/common.hh>
#include <gazebo/math/gzmath.hh>
#include <gazebo/msgs/msgs.hh>
#include <gazebo/sdf/sdf.hh>
#include <gazebo/transport/transport.hh>
#include <gazebo/Server.hh>
#include <gazebo/Master.hh>
#include <gazebo/gazebo.hh>
```

Include dependency graph for gazebo_core.hh:



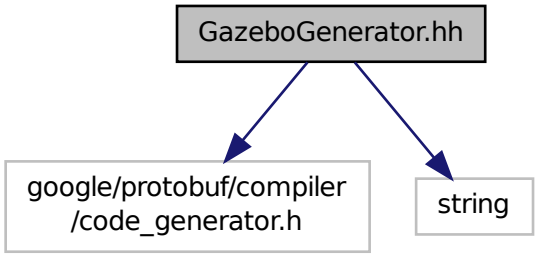
This graph shows which files directly or indirectly include this file:



11.66 GazeboGenerator.hh File Reference

```
#include <google/protobuf/compiler/code_generator.h>
#include <string>
```

Include dependency graph for GazeboGenerator.hh:



Classes

- class **google::protobuf::compiler::cpp::GazeboGenerator**

  *Google protobuf message generator for **gazebo::msgs*** (p. *101*).

Namespaces

- namespace **google**
- namespace **google::protobuf**
- namespace **google::protobuf::compiler**
- namespace **google::protobuf::compiler::cpp**

11.67 GpuLaser.hh File Reference

```
#include <string>
#include <vector>
#include "rendering/ogre_gazebo.h"
#include "rendering/Camera.hh"
#include "sensors/SensorTypes.hh"
#include "common/Event.hh"
#include "common/Time.hh"
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "math/Vector2i.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for GpuLaser.hh:

Classes

- class **gazebo::rendering::GpuLaser**

  GPU based laser distance sensor.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

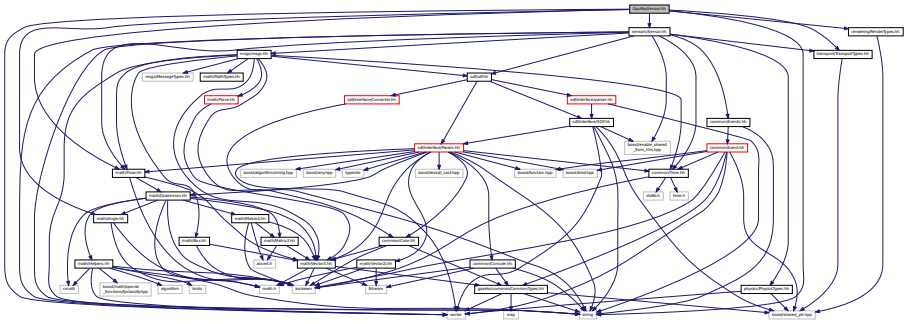
  Common namespace.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.68 GpuRaySensor.hh File Reference

```
#include <vector>
#include <string>
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "transport/TransportTypes.hh"
#include "sensors/Sensor.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for GpuRaySensor.hh:



Classes

- class **gazebo::sensors::GpuRaySensor**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.69 Grid.hh File Reference

```
#include <stdint.h>
#include <vector>
#include <string>
#include "rendering/ogre_gazebo.h"
#include "common/Color.hh"
```

Include dependency graph for Grid.hh:

Classes

- class **gazebo::rendering::Grid**

  Displays a grid of cells, drawn with lines.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

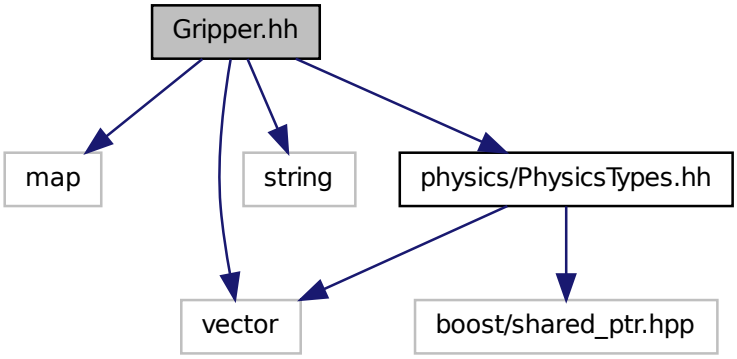
  Rendering namespace.

- namespace **Ogre**

11.70 Gripper.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#include "physics/PhysicsTypes.hh"
```

Include dependency graph for Gripper.hh:



Classes

- class **gazebo::physics::Gripper**

  A gripper abstraction.

Namespaces

- namespace **gazebo**

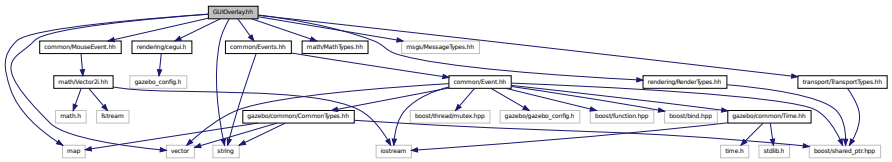
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.71 GUIOverlay.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "rendering/cegui.h"
#include "common/MouseEvent.hh"
#include "common/Events.hh"
#include "math/MathTypes.hh"
#include "rendering/RenderTypes.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for GUIOverlay.hh:



Classes

- class **gazebo::rendering::GUIOverlay**

  A class that creates a CEGUI overlay on a render window.

Namespaces

- namespace **gazebo**

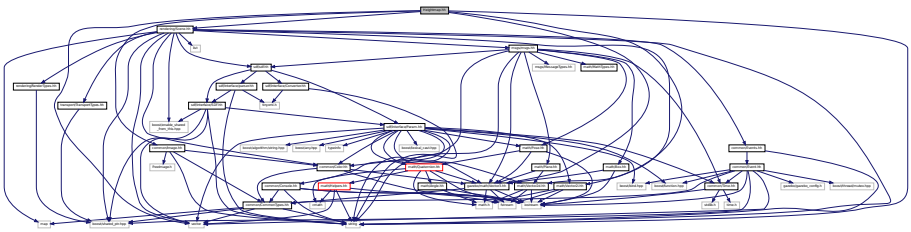
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.72 Heightmap.hh File Reference

```
#include <string>
#include <vector>
#include "common/Image.hh"
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
#include "rendering/Scene.hh"
```

Include dependency graph for Heightmap.hh:



Classes

- class **gazebo::rendering::Heightmap**

  Rendering a terrain using heightmap information.

Namespaces

- namespace **gazebo**

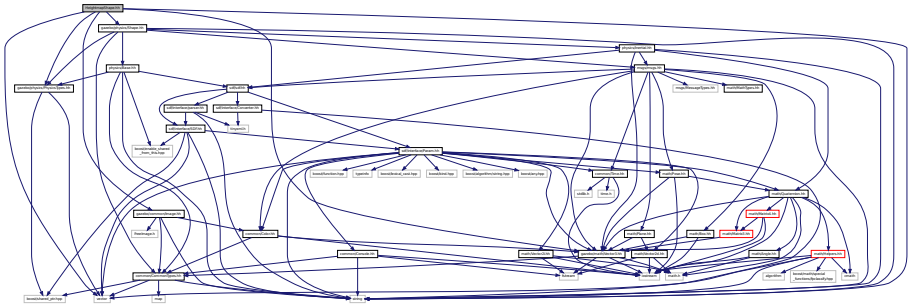
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

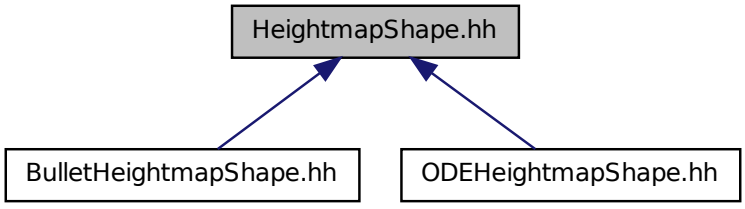
11.73 HeightmapShape.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/common/Image.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/physics/Shape.hh"
```

Include dependency graph for HeightmapShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::HeightmapShape**

  HeightmapShape *(p. 399) collision shape builds a heightmap from an image.*

Namespaces

- namespace **gazebo**

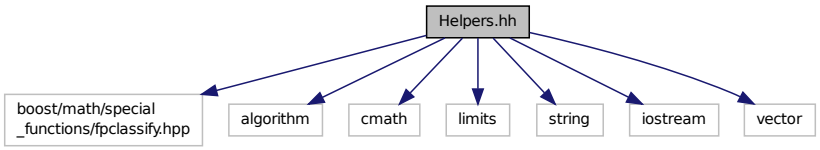
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

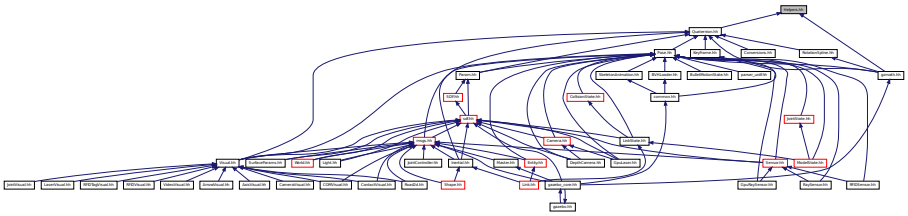
11.74 Helpers.hh File Reference

```
#include <boost/math/special_functions/fpclassify.hpp>
#include <algorithm>
#include <cmath>
#include <limits>
#include <string>
#include <iostream>
#include <vector>
```

Include dependency graph for Helpers.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

Macros

- #define **GZ_DBL_MAX** std::numeric_limits<double>::max()

- #define **GZ_DBL_MIN** std::numeric_limits<double>::min()
- #define **GZ_FLT_MAX** std::numeric_limits<float>::max()
- #define **GZ_FLT_MIN** std::numeric_limits<float>::min()

Functions

- template<typename T >
  T **gazebo::math::clamp** (T _v, T _min, T _max)

  simple clamping function
- template<typename T >
  bool **gazebo::math::equal** (const T &_a, const T &_b, const T &_epsilon=1e-6)

  check if two values are equal, within a tolerance
- bool **gazebo::math::isnan** (float _v)

  check if a float is NaN
- bool **gazebo::math::isnan** (double _v)

  check if a double is NaN
- bool **gazebo::math::isPowerOfTwo** (unsigned int _x)

  is this a power of 2?
- template<typename T >
  T **gazebo::math::max** (const std::vector< T > &_values)

  get the maximum value of vector of values
- template<typename T >
  T **gazebo::math::mean** (const std::vector< T > &_values)

  get mean of vector of values
- template<typename T >
  T **gazebo::math::min** (const std::vector< T > &_values)

  get the minimum value of vector of values
- double **gazebo::math::parseFloat** (const std::string &_input)

  parse string into float
- int **gazebo::math::parseInt** (const std::string &_input)

  parse string into an integer
- template<typename T >
  T **gazebo::math::precision** (const T &_a, const unsigned int &_precision)

  get value at a specified precision
- template<typename T >
  T **gazebo::math::variance** (const std::vector< T > &_values)

  get variance of vector of values

Variables

- static const double **gazebo::math::NAN_D** = std::numeric_limits<double>::quiet_NaN()

  Not a number.
- static const double **gazebo::math::NAN_I** = std::numeric_limits<int>::quiet_NaN()

  TODO Nate: type int has no quiet_NaN ... what does this 0 mean?

11.74.1 Macro Definition Documentation

11.74.1.1 #define GZ_DBL_MAX std::numeric_limits<double>::max()

11.74.1.2 #define GZ_DBL_MIN std::numeric_limits<double>::min()

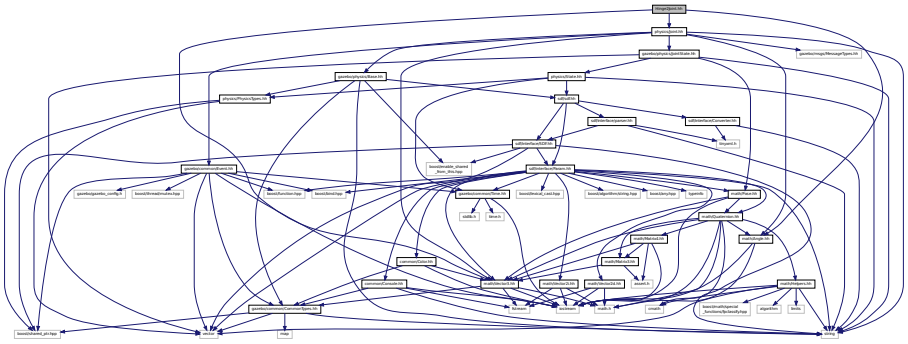
11.74.1.3 #define GZ_FLT_MAX std::numeric_limits<float>::max()

11.74.1.4 #define GZ_FLT_MIN std::numeric_limits<float>::min()

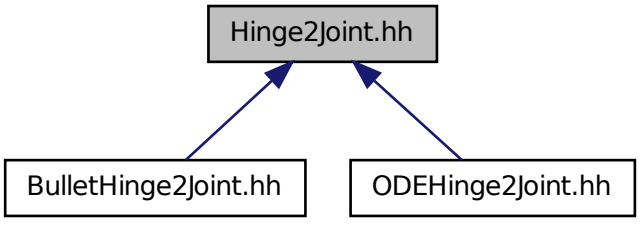
11.75 Hinge2Joint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/Joint.hh"
```

Include dependency graph for Hinge2Joint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Hinge2Joint**< **T** >

  A two axis hinge joint.

Namespaces

- namespace **gazebo**

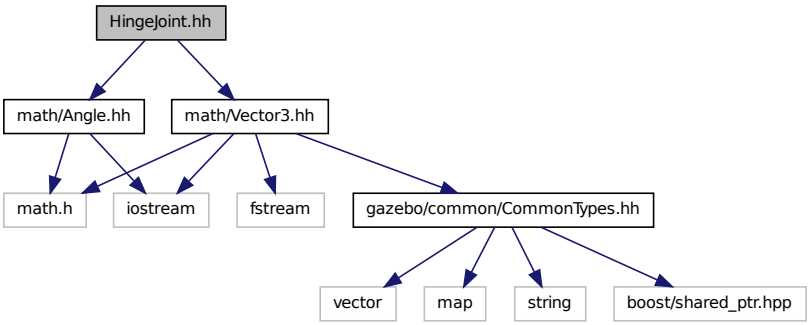
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

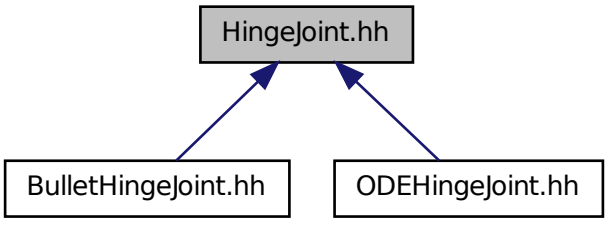
11.76 HingeJoint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
```

Include dependency graph for HingeJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::HingeJoint**< **T** >

  A single axis hinge joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

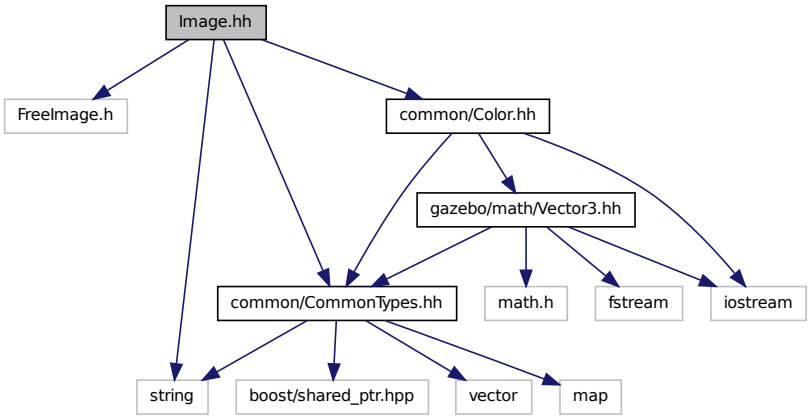
- namespace **gazebo::physics**

  namespace for physics

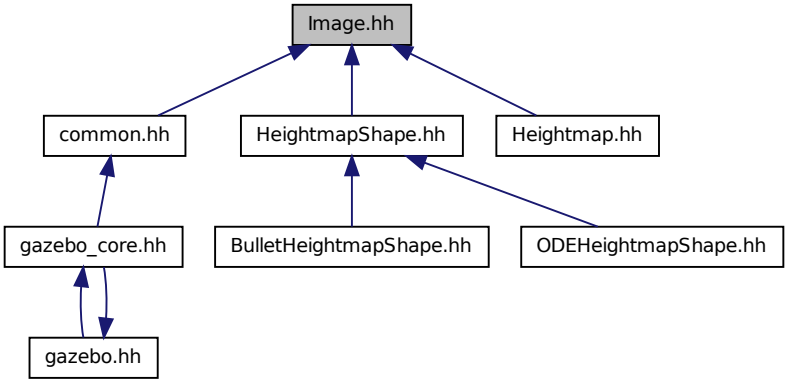
11.77 Image.hh File Reference

```
#include <FreeImage.h>
#include <string>
#include "common/CommonTypes.hh"
#include "common/Color.hh"
```

Include dependency graph for Image.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Image**

  Encapsulates an image.

Namespaces

- namespace **gazebo**

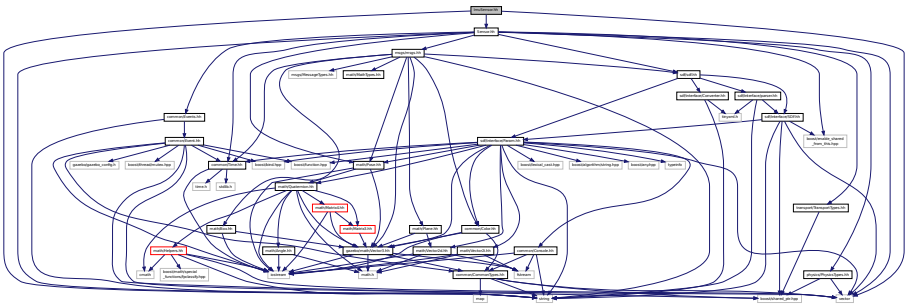
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.78 ImuSensor.hh File Reference

```
#include <vector>
#include <string>
#include "Sensor.hh"
```

Include dependency graph for ImuSensor.hh:



Classes

- class **gazebo::sensors::ImuSensor**

  An IMU sensor.

Namespaces

- namespace **gazebo**

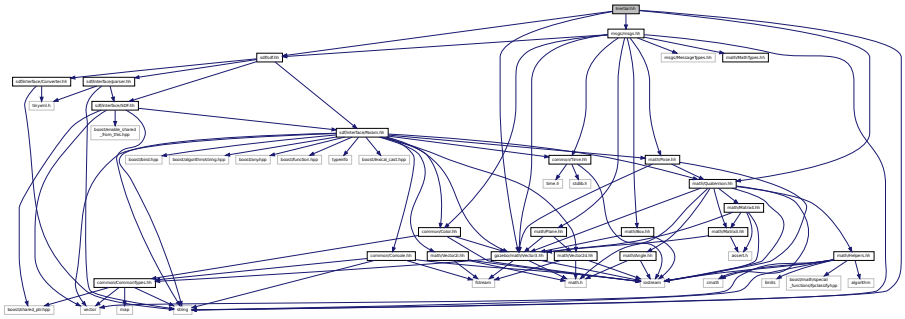
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

11.79 Inertial.hh File Reference

```
#include <string>
#include "msgs/msgs.hh"
#include "sdf/sdf.hh"
#include "math/Quaternion.hh"
#include "math/Vector3.hh"
```

Include dependency graph for Inertial.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::physics::Inertial**

  A class for inertial information about a link.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

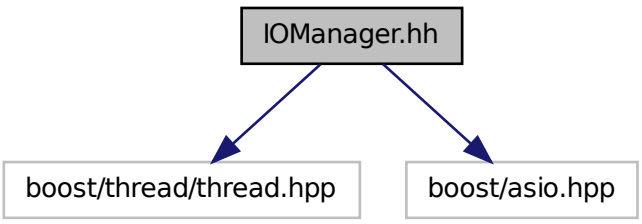
- namespace **gazebo::physics**

  namespace for physics

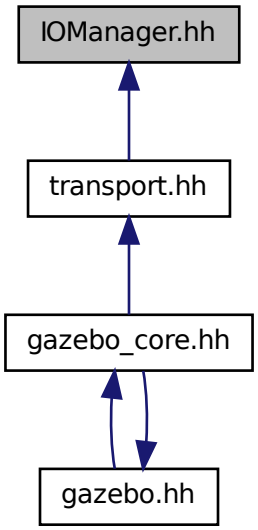
11.80 IOManager.hh File Reference

```
#include <boost/thread/thread.hpp>
#include <boost/asio.hpp>
```

Include dependency graph for IOManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::IOManager**

  Managers boost::asio IO.

Namespaces

- namespace **gazebo**

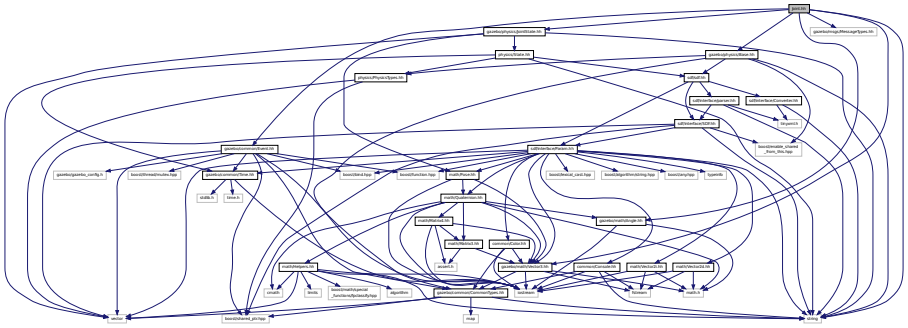
  Forward declarations for the common classes.
- namespace **gazebo::transport**

  Transport namespace.

11.81 Joint.hh File Reference

```
#include <string>
#include "gazebo/common/Event.hh"
#include "gazebo/math/Angle.hh"
#include "gazebo/math/Vector3.hh"
#include "gazebo/msgs/MessageTypes.hh"
#include "gazebo/physics/JointState.hh"
#include "gazebo/physics/Base.hh"
```

Include dependency graph for Joint.hh:



This graph shows which files directly or indirectly include this file:

Classes

- class **gazebo::physics::Joint**

  Base *(*p. *145) class for all joints.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

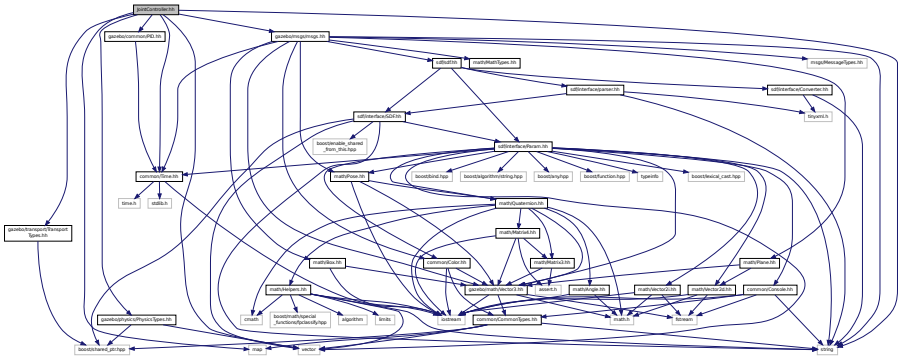
- namespace **gazebo::physics**

  namespace for physics

11.82 JointController.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include "gazebo/common/PID.hh"
#include "gazebo/common/Time.hh"
#include "gazebo/physics/PhysicsTypes.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/msgs/msgs.hh"
```

Include dependency graph for JointController.hh:



Classes

- class **gazebo::physics::JointController**

  *A class for manipulating **physics::Joint*** (p. *423*).

Namespaces

- namespace **gazebo**

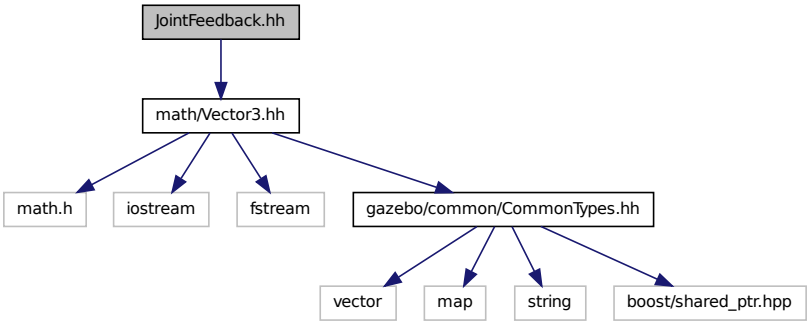
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

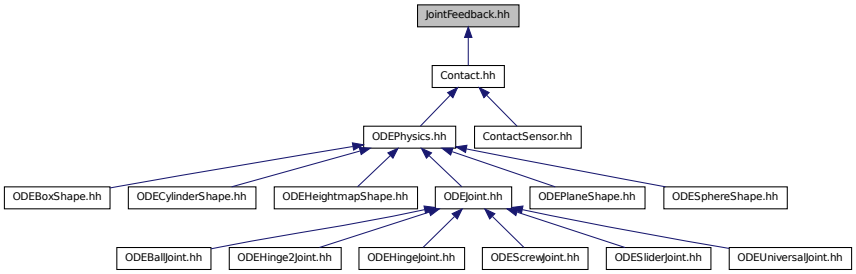
11.83 JointFeedback.hh File Reference

```
#include "math/Vector3.hh"
```

Include dependency graph for JointFeedback.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::JointFeedback**

  Feedback information from a joint.

Namespaces

- namespace **gazebo**

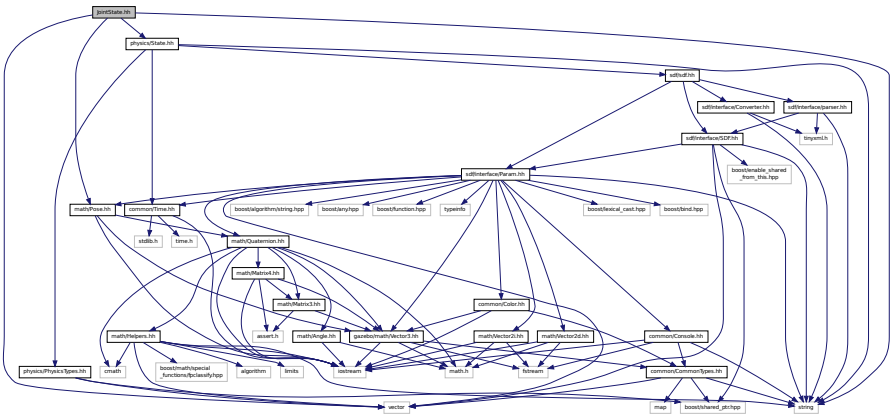
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

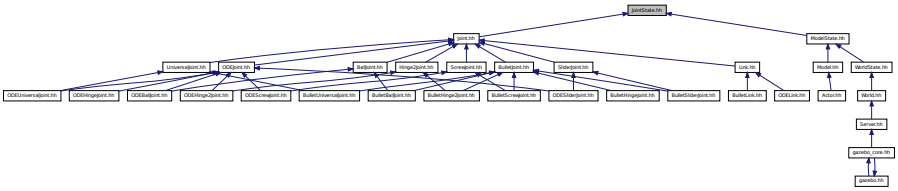
11.84 JointState.hh File Reference

```
#include <vector>
#include <string>
#include "physics/State.hh"
#include "math/Pose.hh"
```

Include dependency graph for JointState.hh:



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. 423)

Namespaces

- namespace **gazebo**

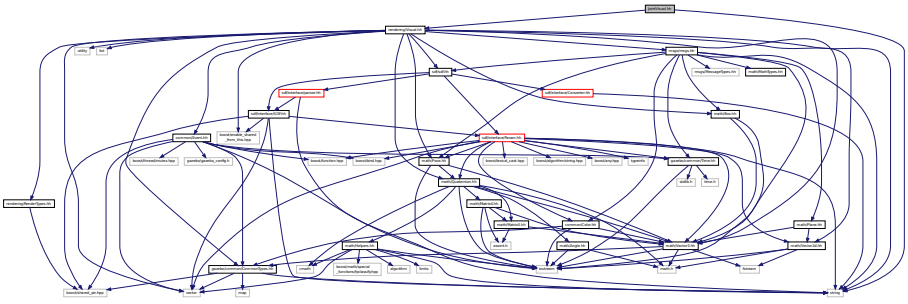
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.85 JointVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
```

Include dependency graph for JointVisual.hh:



Classes

- class **gazebo::rendering::JointVisual**

  Visualization for joints.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

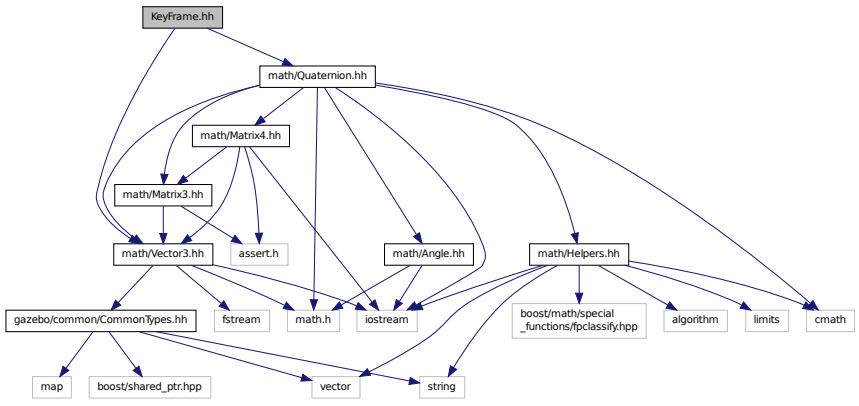
- namespace **gazebo::rendering**

  Rendering namespace.

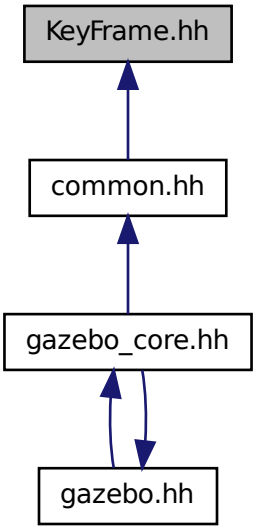
11.86 KeyFrame.hh File Reference

```
#include "math/Vector3.hh"
#include "math/Quaternion.hh"
```

Include dependency graph for KeyFrame.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::KeyFrame**

  A key frame in an animation.
- class **gazebo::common::NumericKeyFrame**

  *A keyframe for a **NumericAnimation** (p. 569).*
- class **gazebo::common::PoseKeyFrame**

  *A keyframe for a **PoseAnimation** (p. 685).*

Namespaces

- namespace **gazebo**

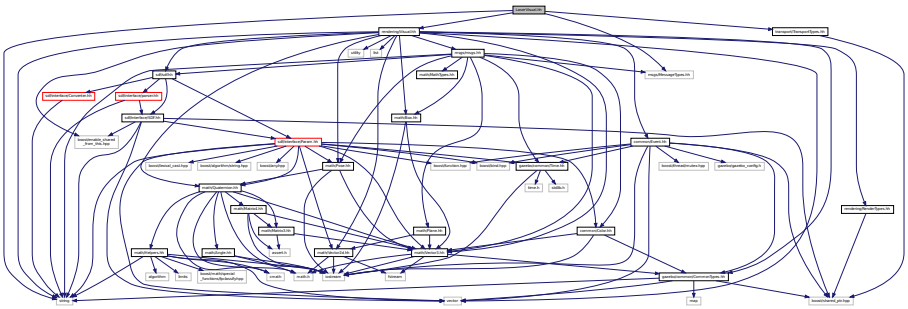
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.87 LaserVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for LaserVisual.hh:



Classes

- class **gazebo::rendering::LaserVisual**

  Visualization for laser data.

Namespaces

- namespace **gazebo**

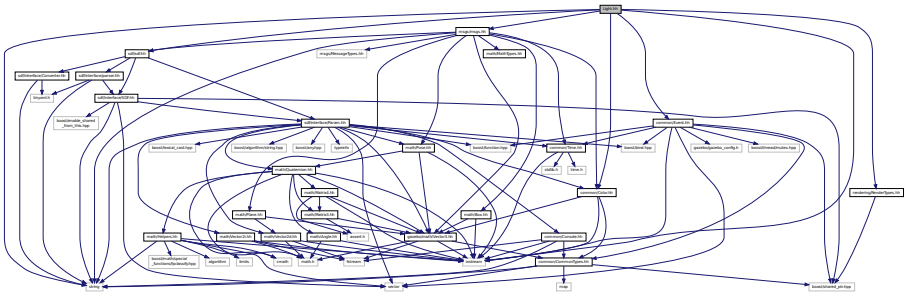
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.88 Light.hh File Reference

```
#include <string>
#include <iostream>
#include "msgs/msgs.hh"
#include "rendering/RenderTypes.hh"
#include "common/Event.hh"
#include "common/Color.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for Light.hh:



Classes

- class **gazebo::rendering::Light**

  A light source.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

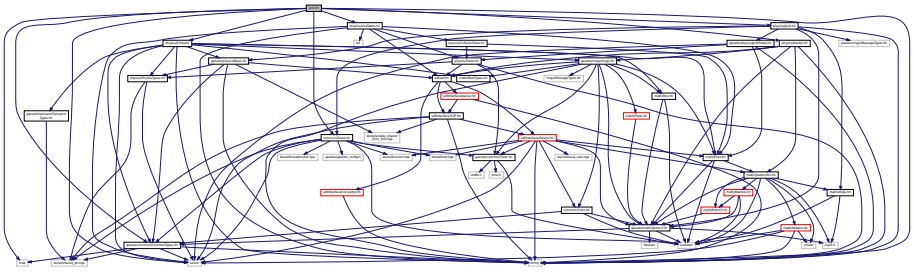
  Rendering namespace.

- namespace **Ogre**

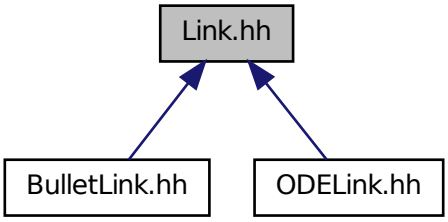
11.89 Link.hh File Reference

```
#include <map>
#include <vector>
#include <string>
#include "common/Event.hh"
#include "common/CommonTypes.hh"
#include "physics/LinkState.hh"
#include "physics/Entity.hh"
#include "physics/Inertial.hh"
#include "physics/Joint.hh"
```

Include dependency graph for Link.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Link**

  Link (p. *454*) *class defines a rigid body entity, containing information on inertia, visual and collision properties of a rigid body.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

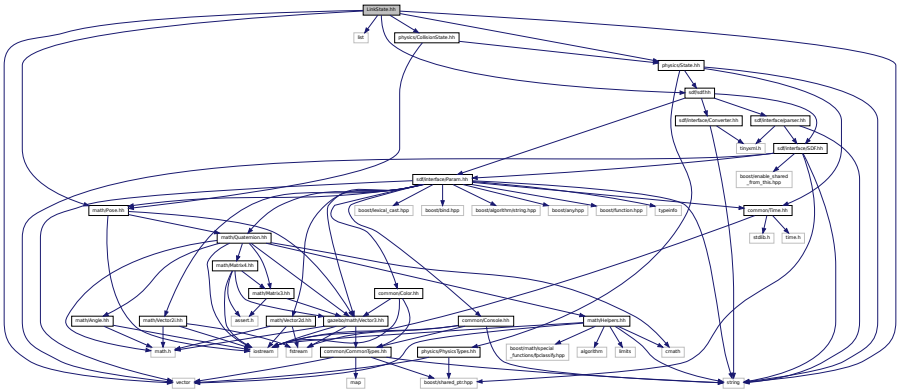
- namespace **gazebo::physics**

  namespace for physics

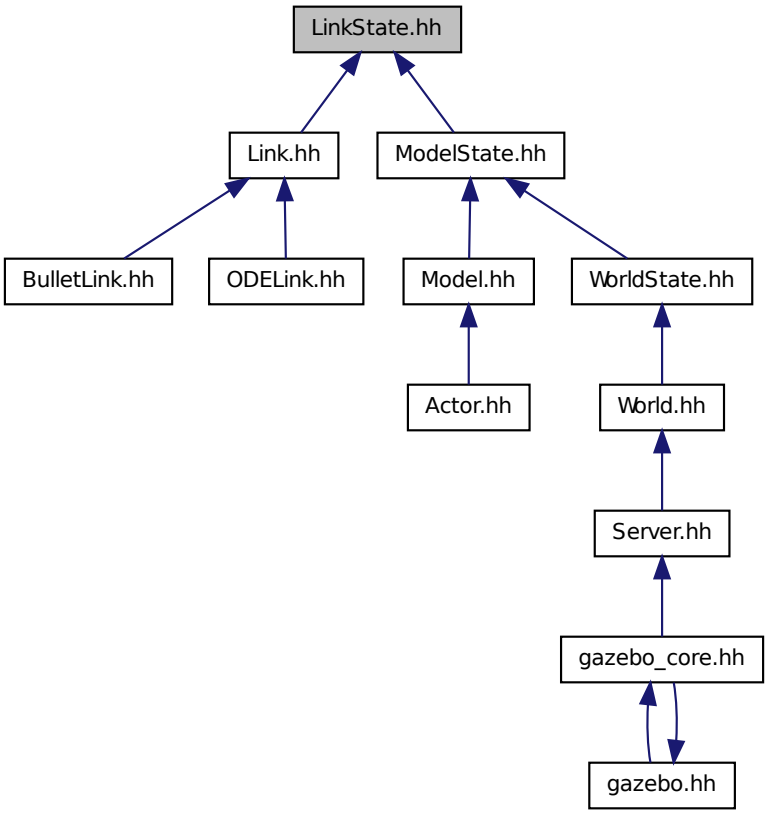
11.90 LinkState.hh File Reference

```
#include <vector>
#include <string>
#include <list>
#include "sdf/sdf.hh"
#include "physics/State.hh"
#include "physics/CollisionState.hh"
#include "math/Pose.hh"
```

Include dependency graph for LinkState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::LinkState**

  *Store state information of a **physics::Link** (p. 454) object.*

Namespaces

- namespace **gazebo**

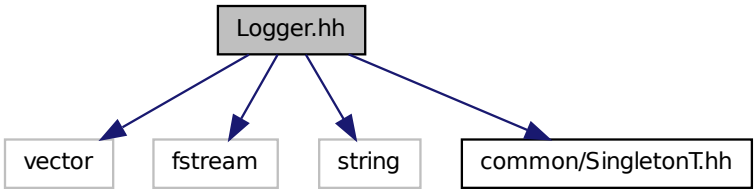
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.91 Logger.hh File Reference

```
#include <vector>
#include <fstream>
#include <string>
#include "common/SingletonT.hh"
```

Include dependency graph for Logger.hh:



Classes

- class **gazebo::physics::Logger**

  Handles logging of data to disk.

Namespaces

- namespace **gazebo**

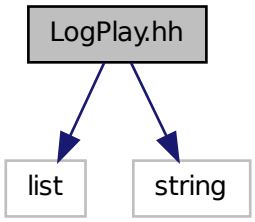
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.92 LogPlay.hh File Reference

```
#include <list>
#include <string>
```

Include dependency graph for LogPlay.hh:



Classes

- class **gazebo::physics::Logplay**

  *Open and playback log files that were recorded using **Logger** (p. 477).*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::physics**

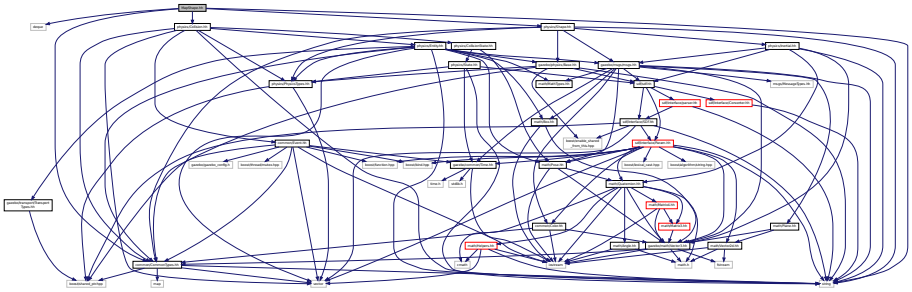
  namespace for physics

11.93 mainpage.html File Reference

11.94 MapShape.hh File Reference

```
#include <deque>
#include <string>
#include "common/CommonTypes.hh"
#include "physics/Collision.hh"
#include "physics/Shape.hh"
```

Include dependency graph for MapShape.hh:



Classes

- class **gazebo::physics::MapShape**

  Creates box extrusions based on an image.

Namespaces

- namespace **gazebo**

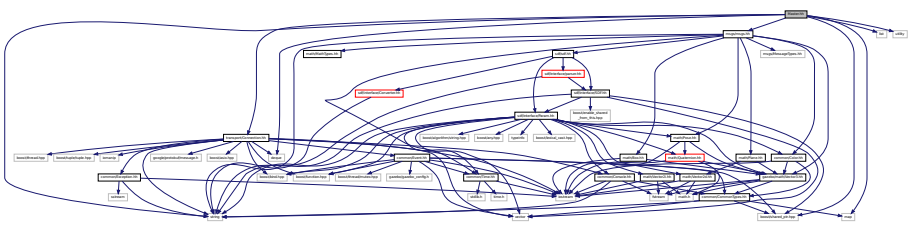
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

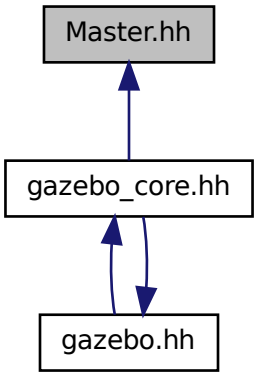
11.95 Master.hh File Reference

```
#include <string>
#include <list>
#include <deque>
#include <utility>
#include <map>
#include <boost/shared_ptr.hpp>
#include "msgs/msgs.hh"
#include "transport/Connection.hh"
```

Include dependency graph for Master.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Master**

  A ROS Master-like manager that directs gztopic connections, enables each gazebo network client to locate one another for peer-to-peer communication.

Namespaces

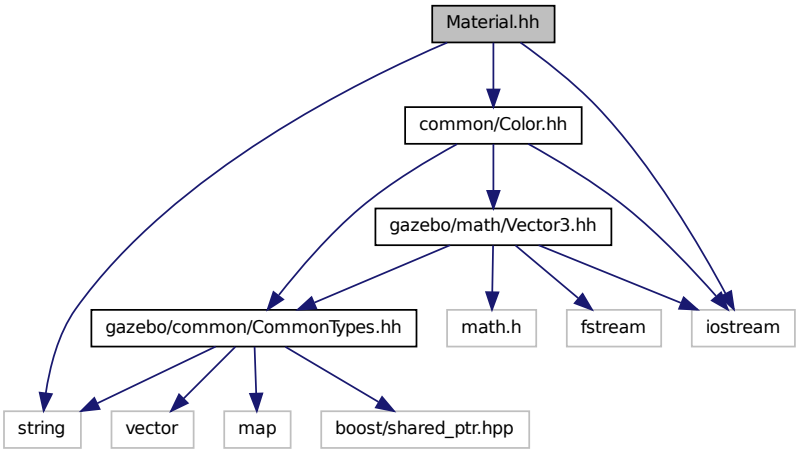
- namespace **gazebo**

  Forward declarations for the common classes.

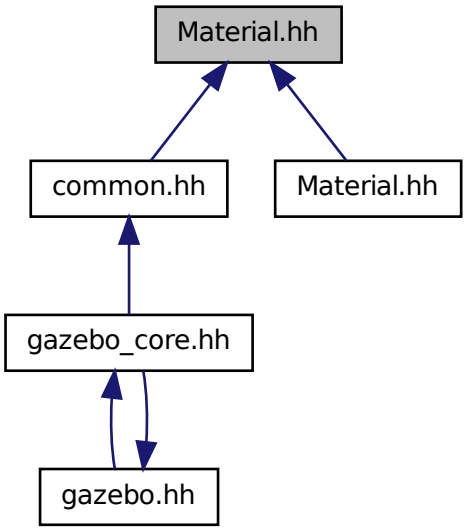
11.96 Material.hh File Reference

```
#include <string>
#include <iostream>
#include "common/Color.hh"
```

Include dependency graph for common/Material.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Material**

  Encapsulates description of a material.

Namespaces

- namespace **gazebo**

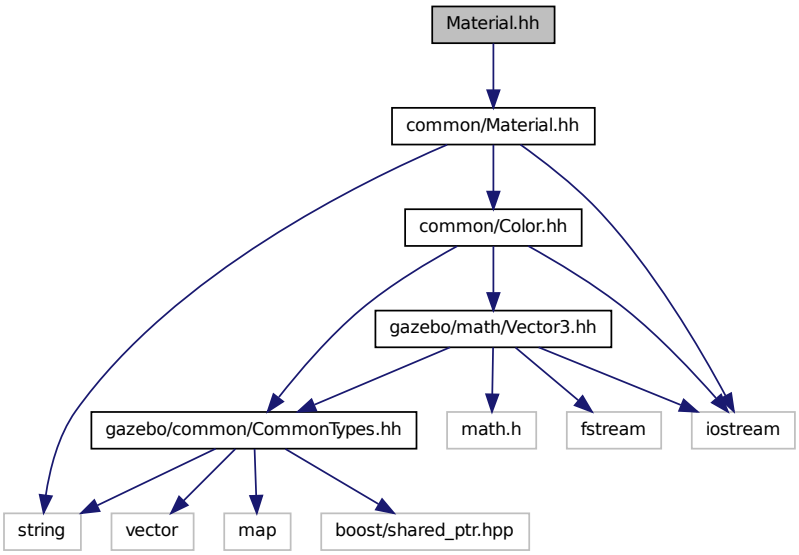
  Forward declarations for the common classes.

- namespace **gazebo::common**

  Common namespace.

11.97 Material.hh File Reference

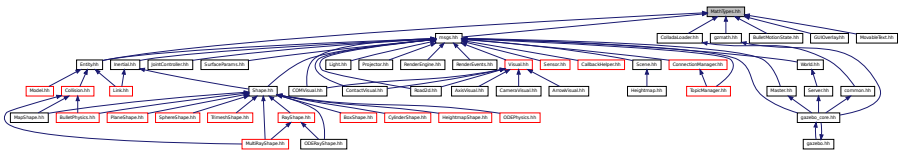
`#include "common/Material.hh"`
Include dependency graph for rendering/Material.hh:



11.98 MathTypes.hh File Reference

Forward declarations for the math classes.

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

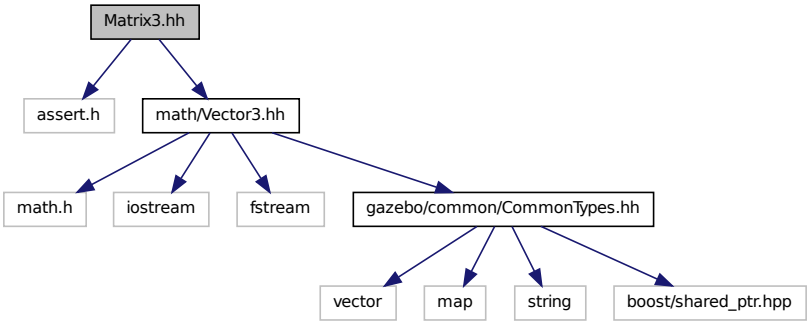
11.98.1 Detailed Description

Forward declarations for the math classes.

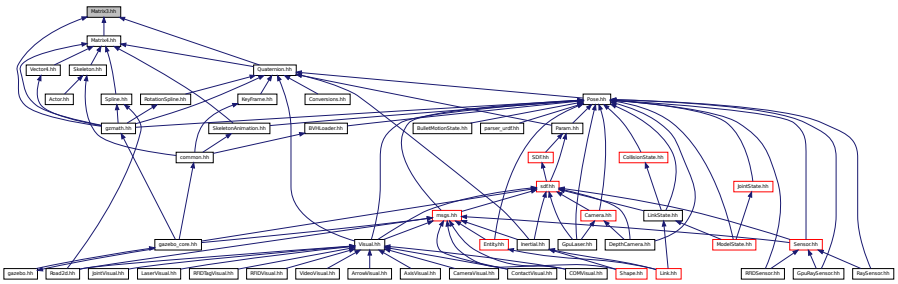
11.99 Matrix3.hh File Reference

```
#include <assert.h>
#include "math/Vector3.hh"
```

Include dependency graph for Matrix3.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Matrix3**

  A 3x3 matrix class.

Namespaces

- namespace **gazebo**

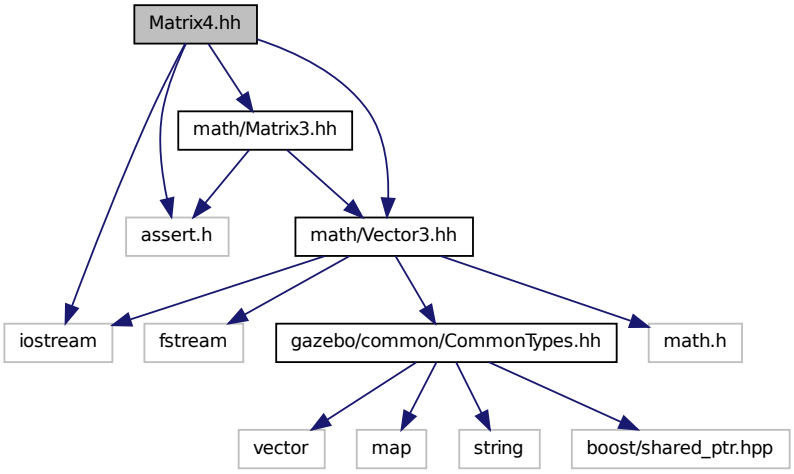
  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

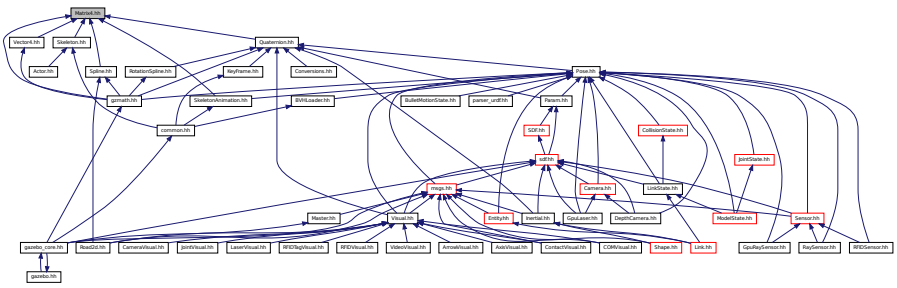
11.100 Matrix4.hh File Reference

```
#include <assert.h>
#include <iostream>
#include "math/Vector3.hh"
#include "math/Matrix3.hh"
```

Include dependency graph for Matrix4.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Matrix4**

  A 3x3 matrix class.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

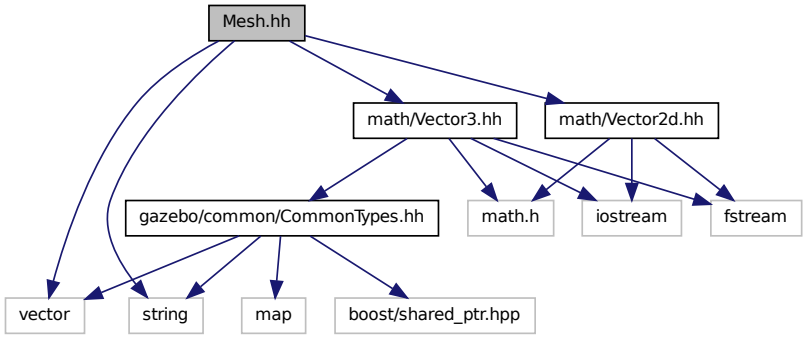
- namespace **gazebo::math**

  Math namespace.

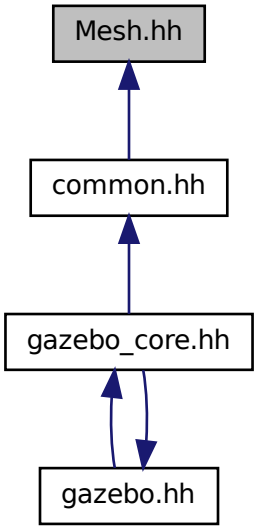
11.101 Mesh.hh File Reference

```
#include <vector>
#include <string>
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
```

Include dependency graph for Mesh.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Mesh**

  A 3D mesh.
- struct **gazebo::common::NodeAssignment**

  Vertex to node weighted assignement for skeleton animation visualization.
- class **gazebo::common::SubMesh**

  A child mesh.

Namespaces

- namespace **gazebo**

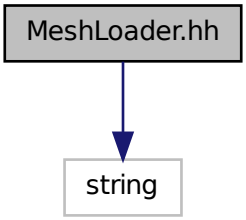
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

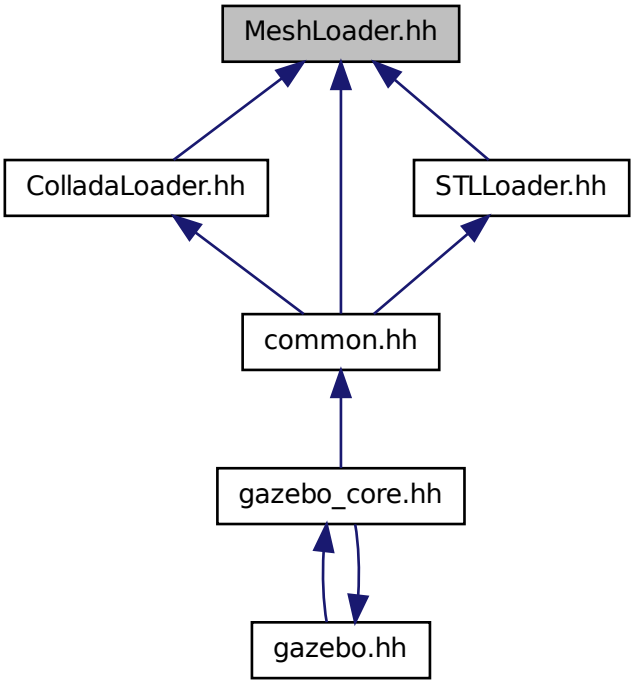
11.102 MeshLoader.hh File Reference

```
#include <string>
```

Include dependency graph for MeshLoader.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MeshLoader**

  Base class for loading meshes.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

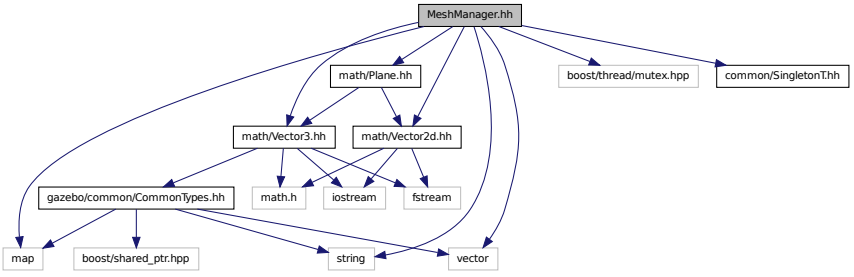
- namespace **gazebo::common**

  Common namespace.

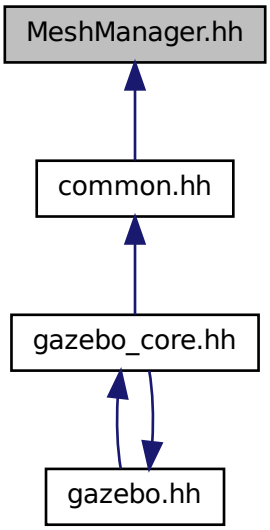
11.103 MeshManager.hh File Reference

```
#include <map>
#include <string>
#include <vector>
#include <boost/thread/mutex.hpp>
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
#include "math/Plane.hh"
#include "common/SingletonT.hh"
```

Include dependency graph for MeshManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MeshManager**

  Maintains and manages all meshes.

Namespaces

- namespace **gazebo**

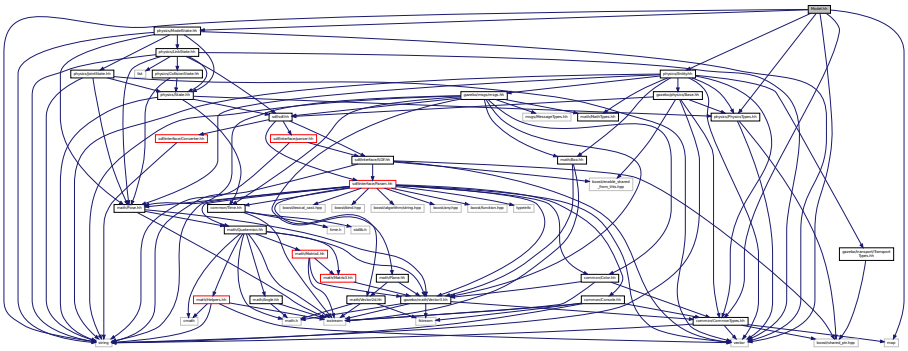
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

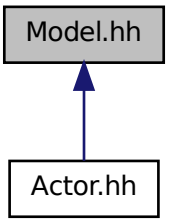
11.104 Model.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/ModelState.hh"
#include "physics/Entity.hh"
```

Include dependency graph for Model.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Model**

  A model is a collection of links, joints, and plugins.

Namespaces

- namespace **boost**
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

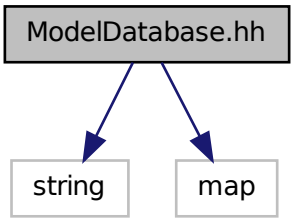
  namespace for physics

11.105 ModelDatabase.hh File Reference

```
#include <string>
```

```
#include <map>
```

Include dependency graph for ModelDatabase.hh:



Classes

- class **gazebo::common::ModelDatabase**

  Connects to model database, and has utility functions to find models.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

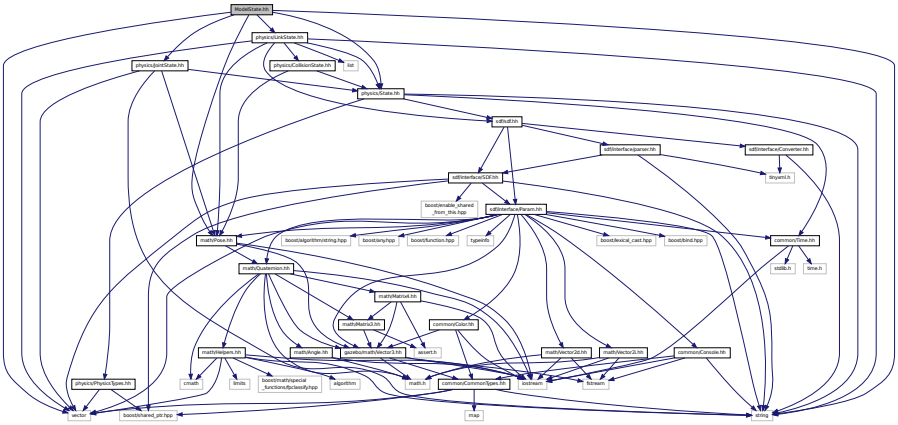
- namespace **gazebo::common**

  Common namespace.

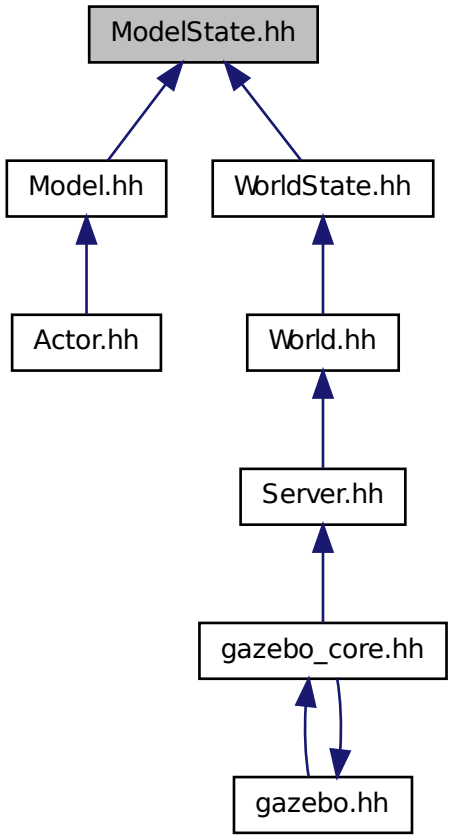
11.106 ModelState.hh File Reference

```
#include <vector>
#include <string>
#include "physics/State.hh"
#include "physics/LinkState.hh"
#include "physics/JointState.hh"
#include "math/Pose.hh"
```

Include dependency graph for ModelState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ModelState**

  *Store state information of a **physics::Model** (p. 521) object.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

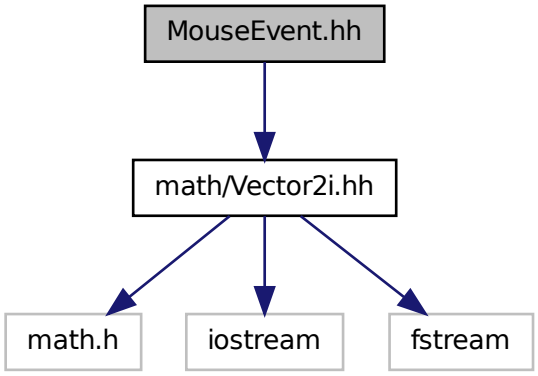
- namespace **gazebo::physics**

  namespace for physics

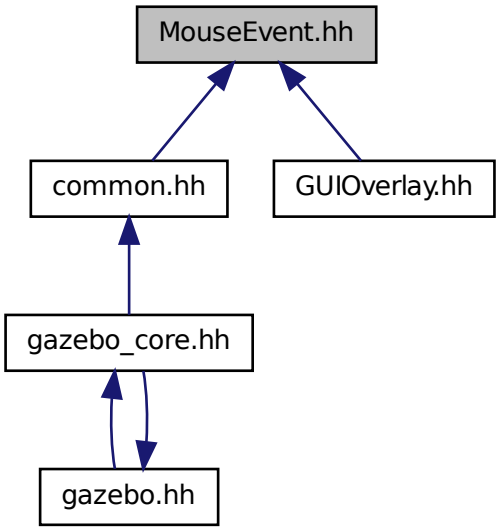
11.107 MouseEvent.hh File Reference

```
#include "math/Vector2i.hh"
```

Include dependency graph for MouseEvent.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::MouseEvent**

  Generic description of a mouse event.

Namespaces

- namespace **gazebo**

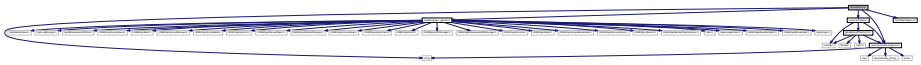
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

11.108 MovableText.hh File Reference

```
#include <string>
#include "rendering/ogre_gazebo.h"
#include "common/CommonTypes.hh"
#include "common/Color.hh"
#include "math/MathTypes.hh"
```

Include dependency graph for MovableText.hh:



Classes

- class **gazebo::rendering::MovableText**

  Movable text.

Namespaces

- namespace **boost**
- namespace **gazebo**

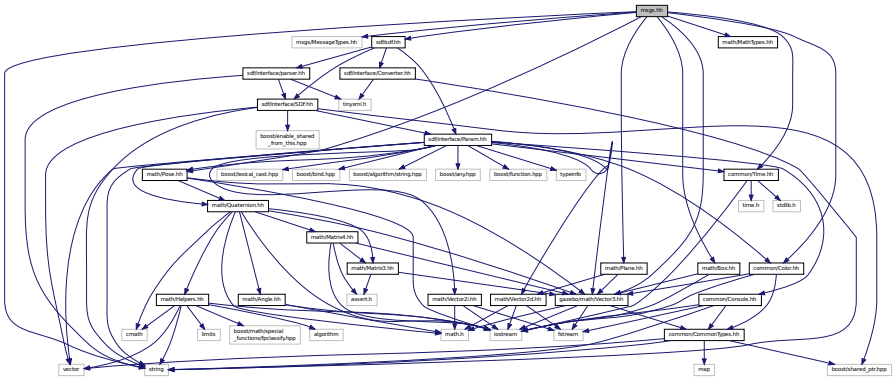
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.109 msgs.hh File Reference

```
#include <string>
#include "msgs/MessageTypes.hh"
#include "sdf/sdf.hh"
#include "math/MathTypes.hh"
#include "math/Vector3.hh"
#include "math/Pose.hh"
#include "math/Plane.hh"
#include "math/Box.hh"
#include "common/Color.hh"
#include "common/Time.hh"
```

Include dependency graph for msgs.hh:



This graph shows which files directly or indirectly include this file:

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::msgs**

  Messages namespace.

Functions

- msgs::Vector3d **gazebo::msgs::Convert** (const math::Vector3 &_v)

  *Convert a **math::Vector3** (p. 902) to a msgs::Vector3d.*
- msgs::Quaternion **gazebo::msgs::Convert** (const math::Quaternion &_q)

  *Convert a **math::Quaternion** (p. 697) to a msgs::Quaternion.*
- msgs::Pose **gazebo::msgs::Convert** (const math::Pose &_p)

  *Convert a **math::Pose** (p. 677) to a msgs::Pose.*
- msgs::Color **gazebo::msgs::Convert** (const common::Color &_c)

  *Convert a **common::Color** (p. 274) to a msgs::Color.*
- msgs::Time **gazebo::msgs::Convert** (const common::Time &_t)

  *Convert a **common::Time** (p. 840) to a msgs::Time.*
- msgs::PlaneGeom **gazebo::msgs::Convert** (const math::Plane &_p)

  *Convert a **math::Plane** (p. 668) to a msgs::PlaneGeom.*
- math::Vector3 **gazebo::msgs::Convert** (const msgs::Vector3d &_v)

  Convert a msgs::Vector3d to a math::Vector.
- math::Quaternion **gazebo::msgs::Convert** (const msgs::Quaternion &_q)

  *Convert a msgs::Quaternion to a **math::Quaternion** (p. 697).*
- math::Pose **gazebo::msgs::Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose** (p. 677).*
- common::Color **gazebo::msgs::Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color** (p. 274).*
- common::Time **gazebo::msgs::Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time** (p. 840).*
- math::Plane **gazebo::msgs::Convert** (const msgs::PlaneGeom &_p)

  Convert a msgs::PlaneGeom to a common::Plane.
- msgs::Request * **gazebo::msgs::CreateRequest** (const std::string &_request, const std::string &_data="")

  Create a request message.
- msgs::Fog **gazebo::msgs::FogFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Fog from a fog SDF element.
- msgs::Header * **gazebo::msgs::GetHeader** (google::protobuf::Message &_message)

  Get the header from a protobuf message.

- msgs::GUI **gazebo::msgs::GUIFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::GUI from a GUI SDF element.
- void **gazebo::msgs::Init** (google::protobuf::Message &_message, const std::string &_id="")

  Initialize a message.
- msgs::Light **gazebo::msgs::LightFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Light from a light SDF element.
- msgs::Scene **gazebo::msgs::SceneFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Scene from a scene SDF element.
- void **gazebo::msgs::Set** (common::Image &_img, const msgs::Image &_msg)

  *Convert a msgs::Image to a **common::Image** (p. 407).*
- void **gazebo::msgs::Set** (msgs::Image *_msg, const common::Image &_i)

  *Set a msgs::Image from a **common::Image** (p. 407).*
- void **gazebo::msgs::Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 902).*
- void **gazebo::msgs::Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 902).*
- void **gazebo::msgs::Set** (msgs::Quaternion *_q, const math::Quaternion &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 697).*
- void **gazebo::msgs::Set** (msgs::Pose *_p, const math::Pose &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 677).*
- void **gazebo::msgs::Set** (msgs::Color *_c, const common::Color &_v)

  *Set a msgs::Color from a **common::Color** (p. 274).*
- void **gazebo::msgs::Set** (msgs::Time *_t, const common::Time &_v)

  *Set a msgs::Time from a **common::Time** (p. 840).*
- void **gazebo::msgs::Set** (msgs::PlaneGeom *_p, const math::Plane &_v)

  *Set a msgs::Plane from a **math::Plane** (p. 668).*
- void **gazebo::msgs::Stamp** (msgs::Header *_header)

  Time stamp a header.
- void **gazebo::msgs::Stamp** (msgs::Time *_time)

  Set the time in a time message.
- msgs::TrackVisual **gazebo::msgs::TrackVisualFromSDF** (**sdf::ElementPtr** _sdf)

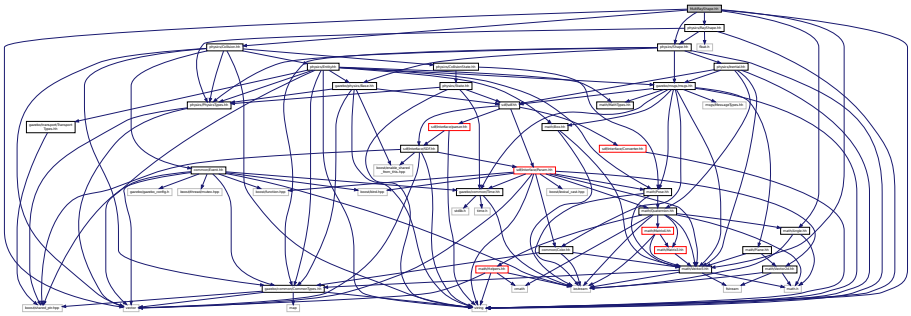
  Create a msgs::TrackVisual from a track visual SDF element.
- msgs::Visual **gazebo::msgs::VisualFromSDF** (**sdf::ElementPtr** _sdf)

  Create a msgs::Visual from a visual SDF element.

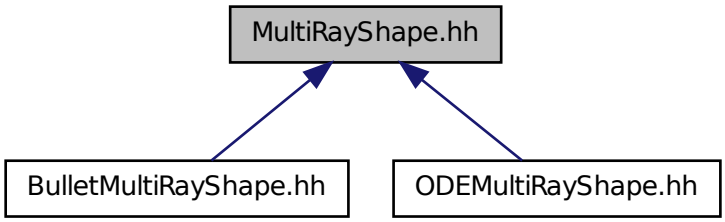
11.110 MultiRayShape.hh File Reference

```
#include <vector>
#include <string>
#include "math/Vector3.hh"
#include "math/Angle.hh"
#include "physics/Collision.hh"
#include "physics/Shape.hh"
#include "physics/RayShape.hh"
```

Include dependency graph for MultiRayShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::MultiRayShape**

  Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

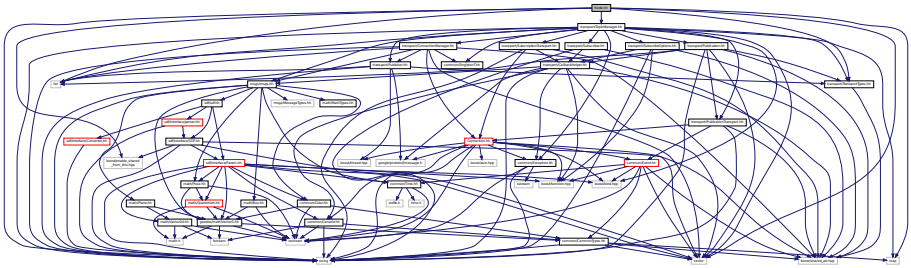
  namespace for physics

11.111 Node.hh File Reference

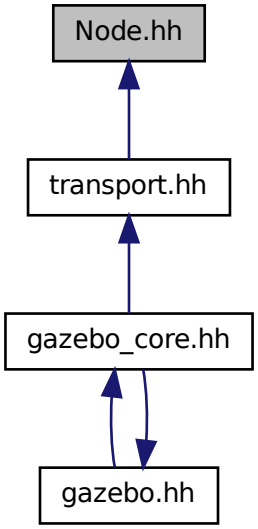
```
#include <boost/enable_shared_from_this.hpp>
```

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include "transport/TransportTypes.hh"
#include "transport/TopicManager.hh"
```

Include dependency graph for Node.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Node**

  A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

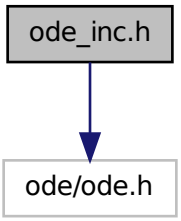
- namespace **gazebo::transport**

  Transport namespace.

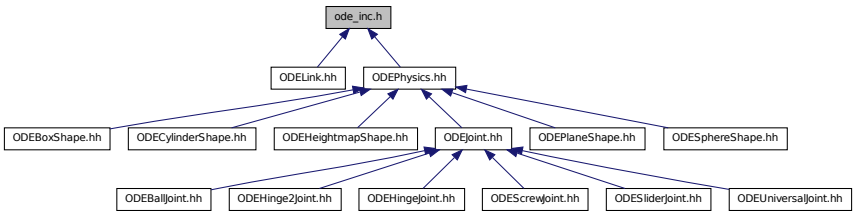
11.112 ode_inc.h File Reference

```
#include <ode/ode.h>
```

Include dependency graph for ode_inc.h:



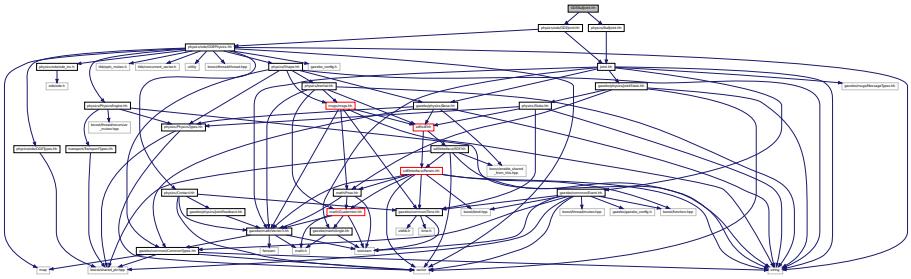
This graph shows which files directly or indirectly include this file:



11.113 ODEBallJoint.hh File Reference

```
#include "physics/BallJoint.hh"
#include "physics/ode/ODEJoint.hh"
```

Include dependency graph for ODEBallJoint.hh:



Classes

- class **gazebo::physics::ODEBallJoint**

  *An **ODEBallJoint** (p. 573).*

Namespaces

- namespace **gazebo**

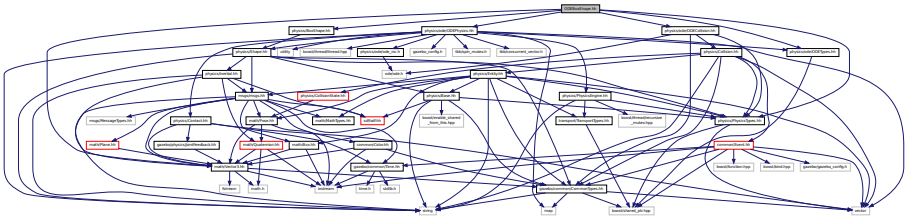
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.114 ODEBoxShape.hh File Reference

```
#include "math/Vector3.hh"
#include "physics/ode/ODEPhysics.hh"
#include "physics/ode/ODETypes.hh"
#include "physics/ode/ODECollision.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/BoxShape.hh"
```

Include dependency graph for ODEBoxShape.hh:



Classes

- class **gazebo::physics::ODEBoxShape**

  ODE Box shape.

Namespaces

- namespace **gazebo**

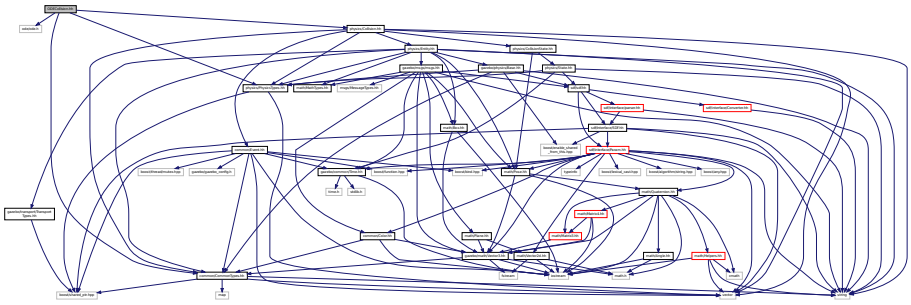
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

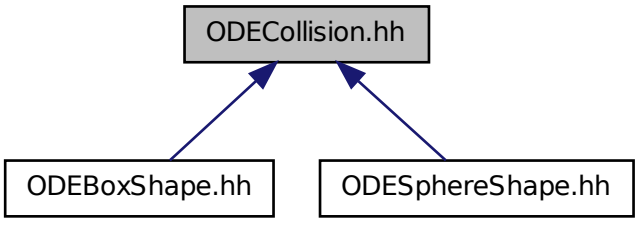
11.115 ODECollision.hh File Reference

```
#include "ode/ode.h"
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/Collision.hh"
```

Include dependency graph for ODECollision.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ODECollision**

  Base *(p. 145) class for all ODE collisions.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

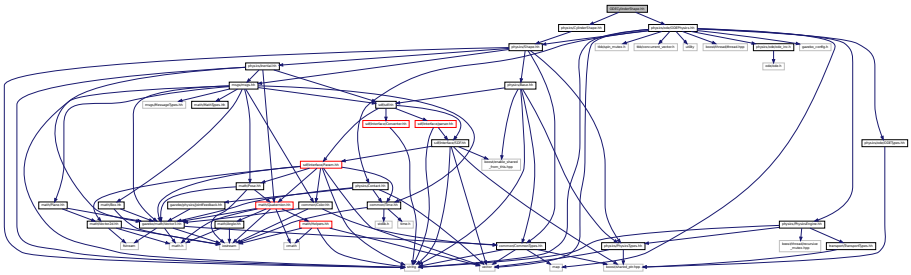
- namespace **gazebo::physics**

  namespace for physics

11.116 ODECylinderShape.hh File Reference

```
#include "physics/CylinderShape.hh"
#include "physics/ode/ODEPhysics.hh"
```

Include dependency graph for ODECylinderShape.hh:



Classes

- class **gazebo::physics::ODECylinderShape**

  ODE cylinder shape.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

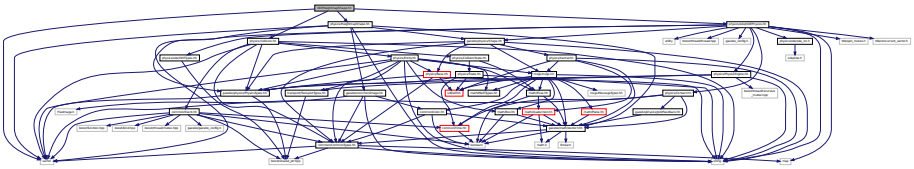
- namespace **gazebo::physics**

  namespace for physics

11.117 ODEHeightmapShape.hh File Reference

```
#include <vector>
#include "physics/HeightmapShape.hh"
#include "physics/ode/ODEPhysics.hh"
#include "physics/Collision.hh"
```

Include dependency graph for ODEHeightmapShape.hh:



Classes

- class **gazebo::physics::ODEHeightmapShape**

  ODE Height map collision.

Namespaces

- namespace **gazebo**

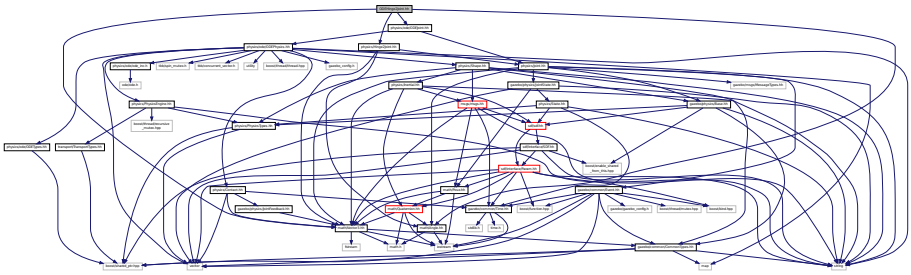
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.118 ODEHinge2Joint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/Hinge2Joint.hh"
#include "physics/ode/ODEJoint.hh"
```

Include dependency graph for ODEHinge2Joint.hh:



Classes

- class **gazebo::physics::ODEHinge2Joint**

  A two axis hinge joint.

Namespaces

- namespace **gazebo**

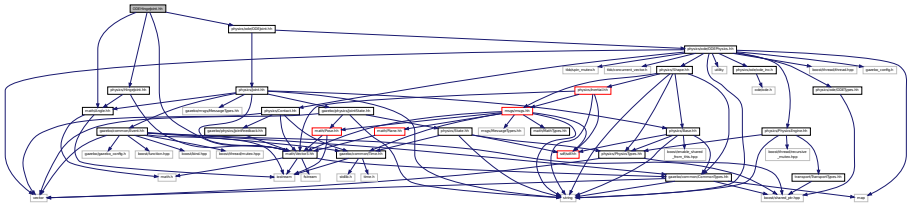
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.119 ODEHingeJoint.hh File Reference

```
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "physics/HingeJoint.hh"
#include "physics/ode/ODEJoint.hh"
```

Include dependency graph for ODEHingeJoint.hh:



Classes

- class **gazebo::physics::ODEHingeJoint**

  A single axis hinge joint.

Namespaces

- namespace **gazebo**

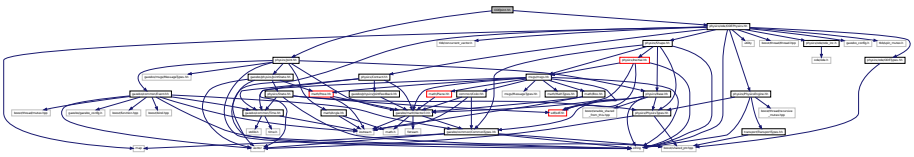
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

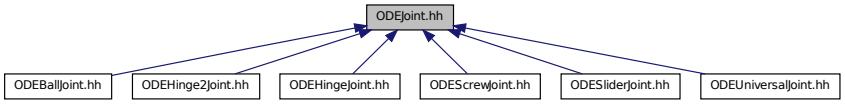
11.120 ODEJoint.hh File Reference

```
#include "physics/ode/ODEPhysics.hh"
#include "physics/Joint.hh"
```

Include dependency graph for ODEJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ODEJoint**

  ODE joint interface.

Namespaces

- namespace **gazebo**

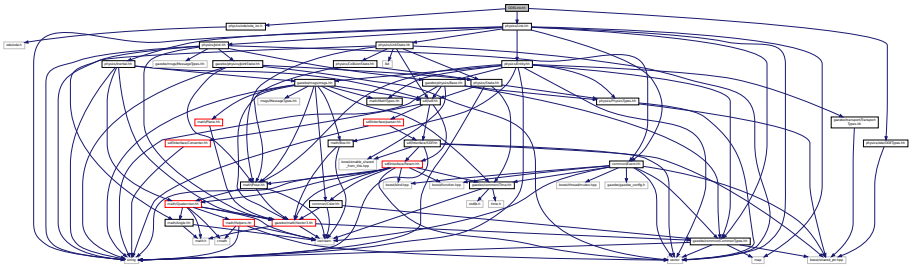
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.121 ODELink.hh File Reference

```
#include "physics/ode/ode_inc.h"
#include "physics/ode/ODETypes.hh"
#include "physics/Link.hh"
```

Include dependency graph for ODELink.hh:



Classes

- class **gazebo::physics::ODELink**

  *ODE **Link** (p. 454) class.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

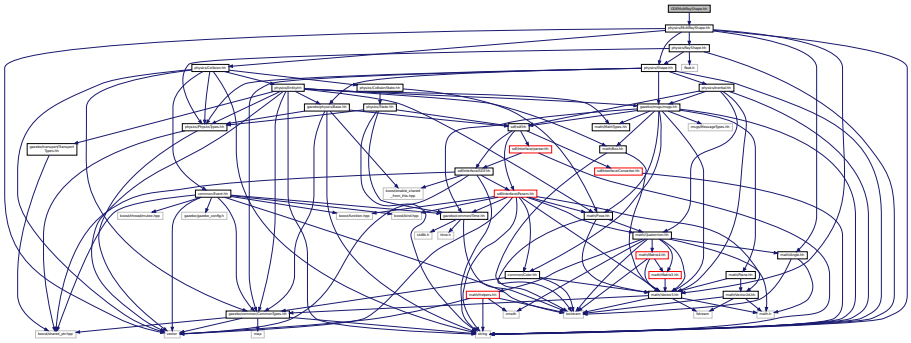
- namespace **gazebo::physics**

  namespace for physics

11.122 ODEMultiRayShape.hh File Reference

```
#include "physics/MultiRayShape.hh"
```

Include dependency graph for ODEMultiRayShape.hh:



Classes

- class **gazebo::physics::ODEMultiRayShape**

  *ODE specific version of **MultiRayShape** (p. 549).*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

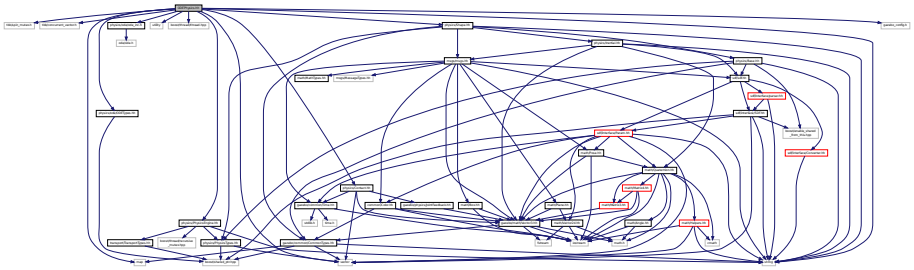
- namespace **gazebo::physics**

  namespace for physics

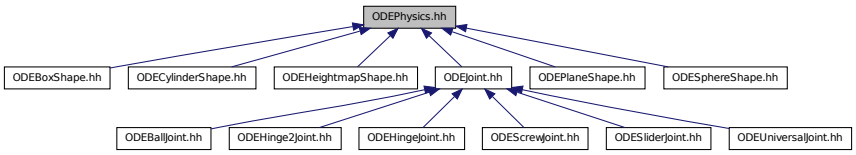
11.123 ODEPhysics.hh File Reference

```
#include <tbb/spin_mutex.h>
#include <tbb/concurrent_vector.h>
#include <map>
#include <string>
#include <vector>
#include <utility>
#include <boost/thread/thread.hpp>
#include "physics/ode/ode_inc.h"
#include "physics/ode/ODETypes.hh"
#include "physics/PhysicsEngine.hh"
#include "physics/Contact.hh"
#include "physics/Shape.hh"
#include "gazebo_config.h"
```

Include dependency graph for ODEPhysics.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ContactFeedback**

  data structure for contact feedbacks
- class **gazebo::physics::ODEPhysics**

  ODE physics engine.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

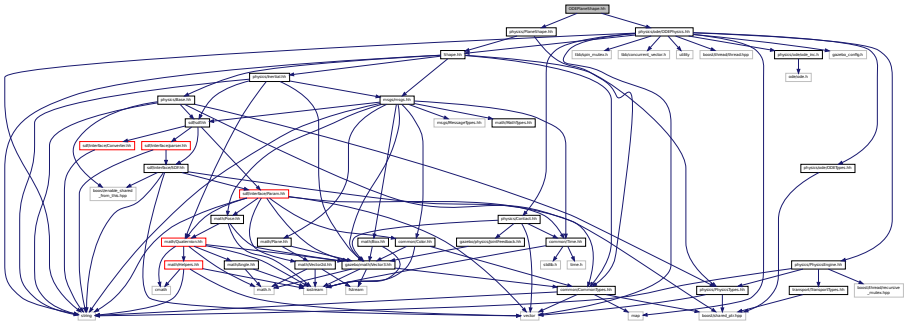
Typedefs

- typedef ODEPhysics ∗ **gazebo::physics::ODEPhysicsPtr**

11.124 ODEPlaneShape.hh File Reference

```
#include "physics/PlaneShape.hh"
#include "physics/ode/ODEPhysics.hh"
```

Include dependency graph for ODEPlaneShape.hh:



Classes

- class **gazebo::physics::ODEPlaneShape**

  An ODE Plane shape.

Namespaces

- namespace **gazebo**

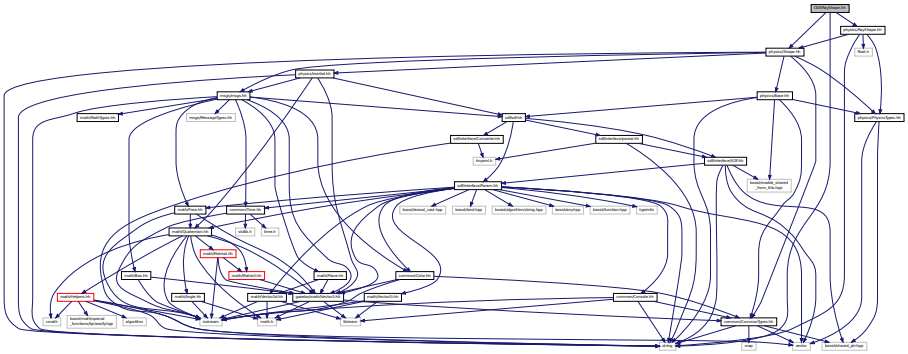
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.125 ODERayShape.hh File Reference

```
#include <string>
#include "physics/RayShape.hh"
#include "physics/Shape.hh"
```

Include dependency graph for ODERayShape.hh:



Classes

- class **gazebo::physics::ODERayShape**

  Ray collision.

Namespaces

- namespace **gazebo**

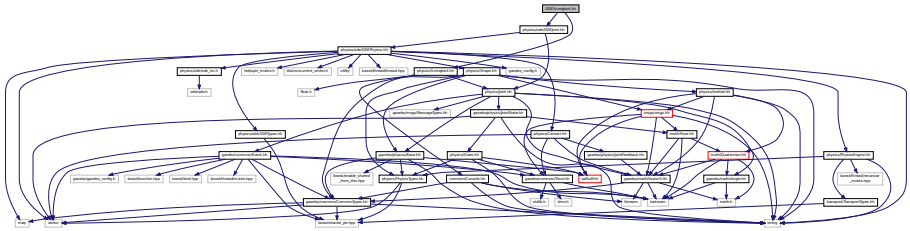
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.126 ODEScrewJoint.hh File Reference

```
#include "physics/ScrewJoint.hh"
#include "physics/ode/ODEJoint.hh"
```

Include dependency graph for ODEScrewJoint.hh:



Classes

- class **gazebo::physics::ODEScrewJoint**

  A screw joint.

Namespaces

- namespace **gazebo**

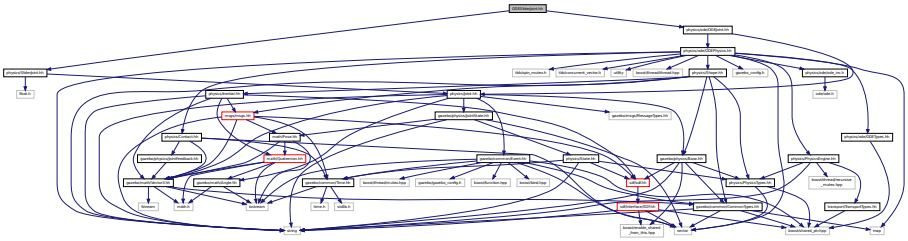
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.127 ODESliderJoint.hh File Reference

```
#include "physics/SliderJoint.hh"
#include "physics/ode/ODEJoint.hh"
```

Include dependency graph for ODESliderJoint.hh:



Classes

- class **gazebo::physics::ODESliderJoint**

  A slider joint.

Namespaces

- namespace **gazebo**

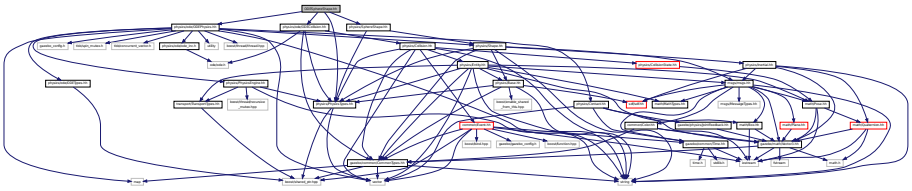
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.128 ODESphereShape.hh File Reference

```
#include "physics/ode/ODEPhysics.hh"
#include "physics/ode/ODECollision.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/SphereShape.hh"
```

Include dependency graph for ODESphereShape.hh:



Classes

- class **gazebo::physics::ODESphereShape**

  A ODE sphere shape.

Namespaces

- namespace **gazebo**

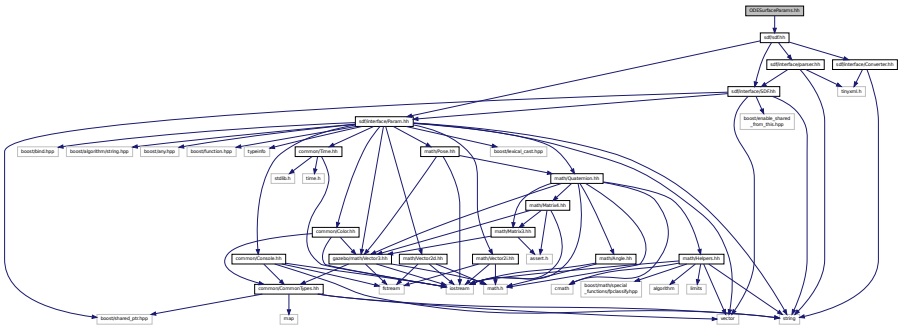
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.129 ODESurfaceParams.hh File Reference

```
#include "sdf/sdf.hh"
```

Include dependency graph for ODESurfaceParams.hh:



Classes

- class **gazebo::physics::ODESurfaceParams**

  Surface params.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

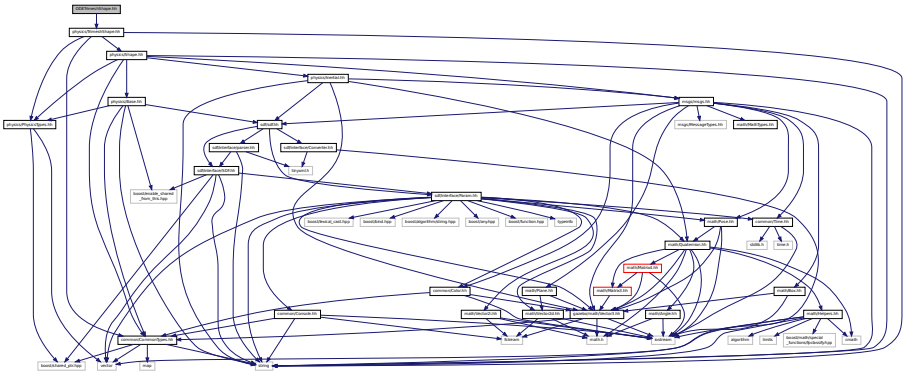
Typedefs

- typedef ODESurfaceParams ∗ **gazebo::physics::ODESurfaceParamsPtr**

11.130 ODETrimeshShape.hh File Reference

```
#include "physics/TrimeshShape.hh"
```

Include dependency graph for ODETrimeshShape.hh:



Classes

- class **gazebo::physics::ODETrimeshShape**

  Triangle mesh collision.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

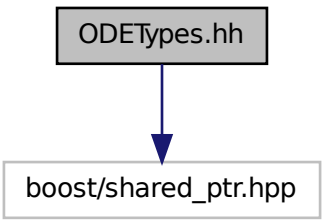
  namespace for physics

11.131 ODETypes.hh File Reference

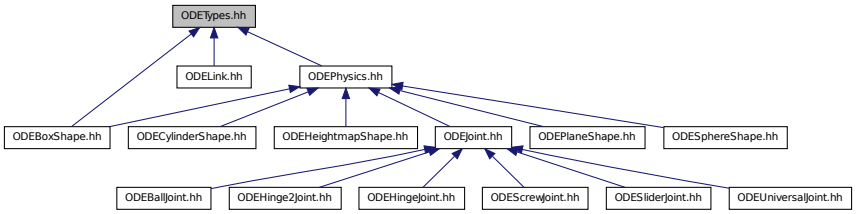
ODE wrapper forward declarations and typedefs.

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for ODETypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Typedefs

- typedef ODECollision ∗ **gazebo::physics::ODECollisionPtr**
- typedef ODELink ∗ **gazebo::physics::ODELinkPtr**
- typedef ODERayShape ∗ **gazebo::physics::ODERayShapePtr**

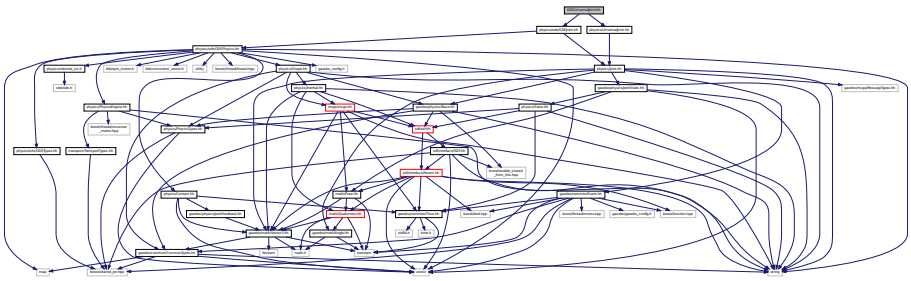
11.131.1 Detailed Description

ODE wrapper forward declarations and typedefs.

11.132 ODEUniversalJoint.hh File Reference

```
#include "physics/UniversalJoint.hh"
#include "physics/ode/ODEJoint.hh"
```

Include dependency graph for ODEUniversalJoint.hh:



Classes

- class **gazebo::physics::ODEUniversalJoint**

  A universal joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::physics**

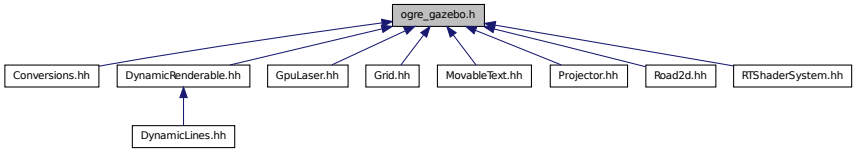
  namespace for physics

11.133 ogre_gazebo.h File Reference

```
#include <Ogre.h>
#include <OgreImageCodec.h>
#include <OGRE/OgreMovableObject.h>
#include <OGRE/OgreRenderable.h>
#include <OgrePlugin.h>
#include <OgreDataStream.h>
#include <OgreLogManager.h>
#include <OgreWindowEventUtilities.h>
#include <OGRE/OgreSceneQuery.h>
#include <OGRE/OgreRoot.h>
#include <OGRE/OgreSceneManager.h>
#include <OGRE/OgreSceneNode.h>
#include <OGRE/OgreVector3.h>
#include <OGRE/OgreManualObject.h>
#include <OGRE/OgreMaterialManager.h>
#include <OGRE/OgreColourValue.h>
#include <OGRE/OgreQuaternion.h>
#include <OGRE/OgreMesh.h>
#include <OGRE/OgreFontManager.h>
#include <OGRE/OgreHardwareBufferManager.h>
#include <OGRE/OgreCamera.h>
#include <OGRE/OgreNode.h>
#include <OGRE/OgreSimpleRenderable.h>
#include <OGRE/OgreFrameListener.h>
#include <OGRE/OgreTexture.h>
#include <OGRE/OgreRenderObjectListener.h>
```

Include dependency graph for ogre_gazebo.h:

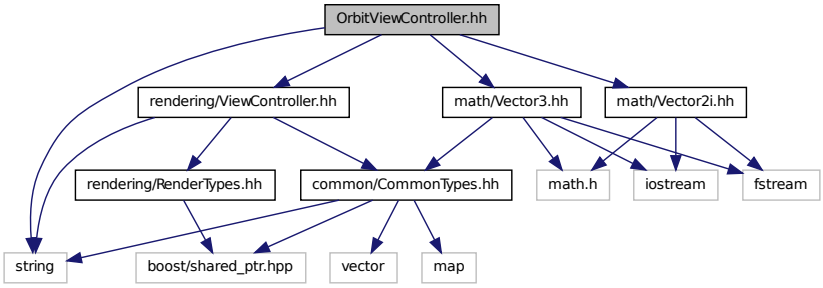
This graph shows which files directly or indirectly include this file:



11.134 OrbitViewController.hh File Reference

```
#include <string>
#include "rendering/ViewController.hh"
#include "math/Vector3.hh"
#include "math/Vector2i.hh"
```

Include dependency graph for OrbitViewController.hh:



Classes

- class **gazebo::rendering::OrbitViewController**

  Orbit view controller.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

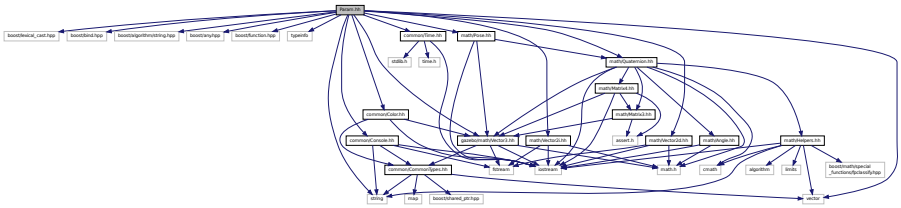
  Rendering namespace.

11.135 Param.hh File Reference

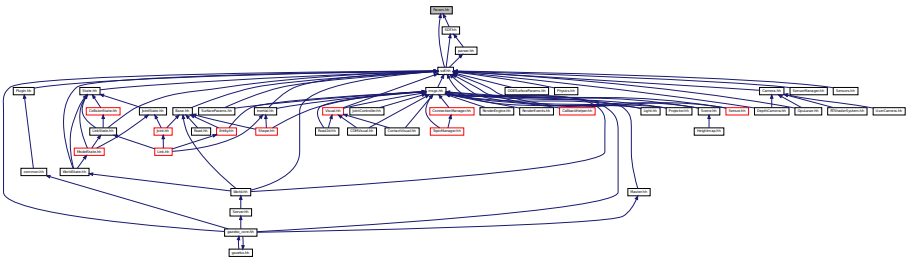
```
#include <boost/lexical_cast.hpp>
```

```
#include <boost/bind.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/any.hpp>
#include <boost/function.hpp>
#include <typeinfo>
#include <string>
#include <vector>
#include "common/Console.hh"
#include "common/Color.hh"
#include "common/Time.hh"
#include "math/Vector3.hh"
#include "math/Vector2i.hh"
#include "math/Vector2d.hh"
#include "math/Pose.hh"
#include "math/Quaternion.hh"
```

Include dependency graph for Param.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Param**

  A parameter class.

- class **sdf::ParamT**< **T** >

  Templatized parameter class.

Namespaces

- namespace **sdf**

  namespace for Simulation Description Format parser

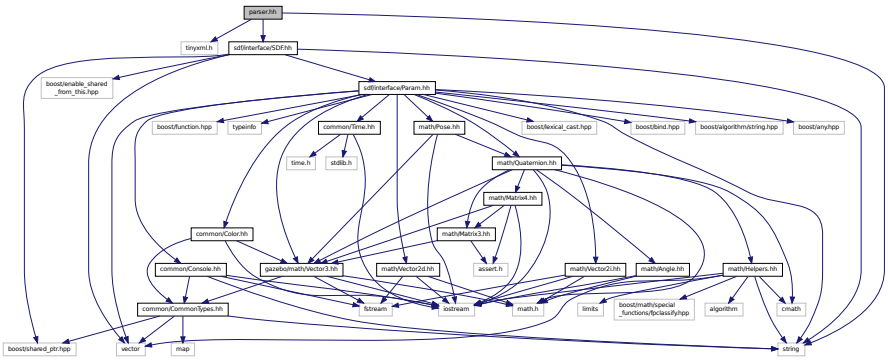
Typedefs

- typedef std::vector< ParamPtr > **sdf::Param_V**
- typedef Param * **sdf::ParamPtr**

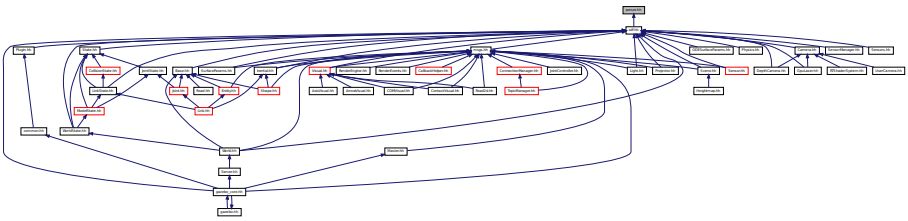
11.136 parser.hh File Reference

```
#include <tinyxml.h>
#include <string>
#include "sdf/interface/SDF.hh"
```

Include dependency graph for parser.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **sdf**

  namespace for Simulation Description Format parser

Functions

- void **sdf::addNestedModel** (ElementPtr _sdf, ElementPtr _includeSDF)
- void **sdf::copyChildren** (ElementPtr _sdf, TiXmlElement *_xml)
- bool **sdf::init** (SDFPtr _sdf)

  Init based on the installed sdf_format.xml file.
- bool **sdf::initDoc** (TiXmlDocument *_xmlDoc, SDFPtr _sdf)

- bool **sdf::initDoc** (TiXmlDocument *_xmlDoc, ElementPtr _sdf)
- bool **sdf::initFile** (const std::string &_filename, SDFPtr _sdf)
- bool **sdf::initFile** (const std::string &_filename, ElementPtr _sdf)
- bool **sdf::initString** (const std::string &_xmlString, SDFPtr _sdf)
- bool **sdf::initXml** (TiXmlElement *_xml, ElementPtr _sdf)
- bool **sdf::readDoc** (TiXmlDocument *_xmlDoc, SDFPtr _sdf, const std::string &_source)

  *Populate the **SDF** (p. 762) values from a TinyXML document.*
- bool **sdf::readDoc** (TiXmlDocument *_xmlDoc, ElementPtr _sdf, const std::string &_source)
- bool **sdf::readFile** (const std::string &_filename, SDFPtr _sdf)

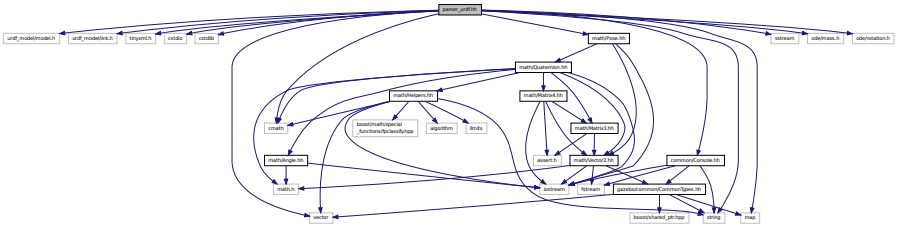
  *Populate the **SDF** (p. 762) values from a file.*
- bool **sdf::readString** (const std::string &_xmlString, SDFPtr _sdf)

  *Populate the **SDF** (p. 762) values from a string.*
- bool **sdf::readString** (const std::string &_xmlString, ElementPtr _sdf)
- bool **sdf::readXml** (TiXmlElement *_xml, ElementPtr _sdf)

11.137 parser_urdf.hh File Reference

```
#include <urdf_model/model.h>
#include <urdf_model/link.h>
#include <tinyxml.h>
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include <vector>
#include <string>
#include <sstream>
#include <map>
#include "ode/mass.h"
#include "ode/rotation.h"
#include "math/Pose.hh"
#include "common/Console.hh"
```

Include dependency graph for parser_urdf.hh:



Classes

- class **urdf2gazebo::GazeboExtension**
- class **urdf2gazebo::URDF2Gazebo**

Namespaces

- namespace **urdf2gazebo**

  namespace for URDF to SDF parser

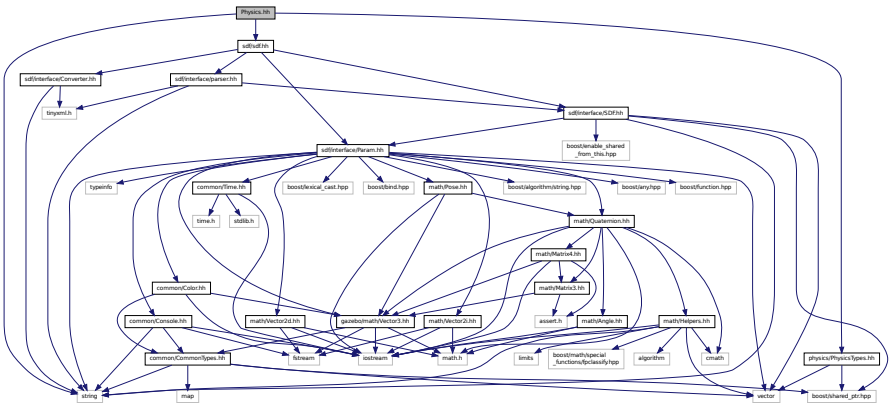
Typedefs

- typedef urdf::Collision ∗ **urdf2gazebo::CollisionPtr**
- typedef urdf::Visual ∗ **urdf2gazebo::VisualPtr**

11.138 Physics.hh File Reference

```
#include <string>
#include "physics/PhysicsTypes.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for Physics.hh:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Functions

- WorldPtr **gazebo::physics::create_world** (const std::string &_name="")

  Create a world given a name.
- bool **gazebo::physics::fini** ()

  *Finalize transport by calling **gazebo::transport::fini** (p. 90).*
- WorldPtr **gazebo::physics::get_world** (const std::string &_name="")

  Returns a pointer to a world by name.

- void **gazebo::physics::init_world** (WorldPtr _world)

  Init world given a pointer to it.
- void **gazebo::physics::init_worlds** ()

  initialize multiple worlds stored in static variable gazebo::g_worlds
- bool **gazebo::physics::load** ()

  *Setup **gazebo::SystemPlugin** (p. 838)'s and call **gazebo::transport::init** (p. 90).*
- void **gazebo::physics::load_world** (WorldPtr _world, **sdf::ElementPtr** _sdf)

  *Load world from **sdf::Element** (p. 332) pointer.*
- void **gazebo::physics::load_worlds** (**sdf::ElementPtr** _sdf)

  *load multiple worlds from single **sdf::Element** (p. 332) pointer*
- void **gazebo::physics::pause_world** (WorldPtr _world, bool _pause)

  *Pause world by calling **World::SetPaused** (p. 964).*
- void **gazebo::physics::pause_worlds** (bool pause)

  pause multiple worlds stored in static variable gazebo::g_worlds
- void **gazebo::physics::remove_worlds** ()

  remove multiple worlds stored in static variable gazebo::g_worlds
- void **gazebo::physics::run_world** (WorldPtr _world)

  *Run world by calling **World::Run()** (p. 964) given a pointer to it.*
- void **gazebo::physics::run_worlds** ()

  run multiple worlds stored in static variable gazebo::g_worlds
- void **gazebo::physics::stop_world** (WorldPtr _world)

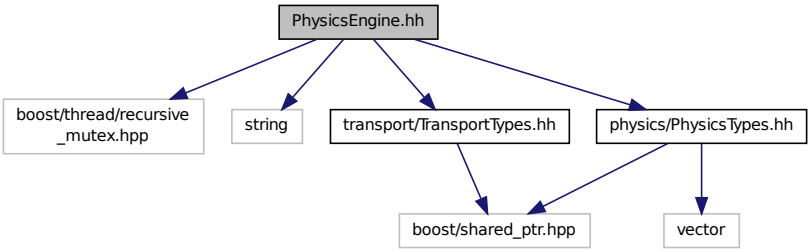
  *Stop world by calling **World::Stop()** (p. 965) given a pointer to it.*
- void **gazebo::physics::stop_worlds** ()

  stop multiple worlds stored in static variable gazebo::g_worlds

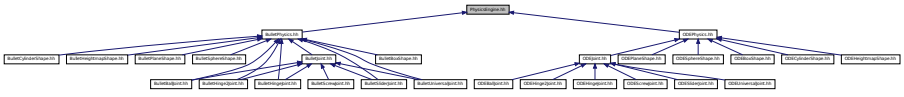
11.139 PhysicsEngine.hh File Reference

```
#include <boost/thread/recursive_mutex.hpp>
#include <string>
#include "transport/TransportTypes.hh"
#include "physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsEngine.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::PhysicsEngine**

  Base *(p. 145) class for a physics engine.*

Namespaces

- namespace **gazebo**

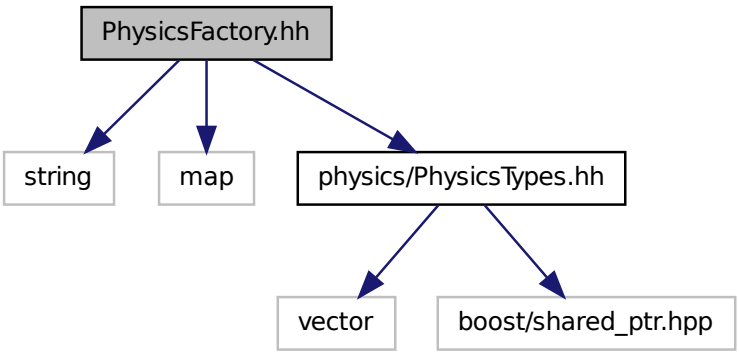
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.140 PhysicsFactory.hh File Reference

```
#include <string>
#include <map>
#include "physics/PhysicsTypes.hh"
```

Include dependency graph for PhysicsFactory.hh:



Classes

- class **gazebo::physics::PhysicsFactory**

  The physics factory instantiates different physics engines.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Macros

- #define **GZ_REGISTER_PHYSICS_ENGINE**(name, classname)

  Static physics registration macro.

Typedefs

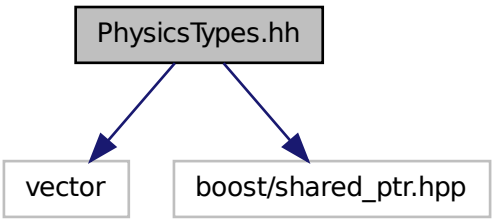
- typedef PhysicsEnginePtr(∗ **gazebo::physics::PhysicsFactoryFn**)(WorldPtr world)

11.141 PhysicsTypes.hh File Reference

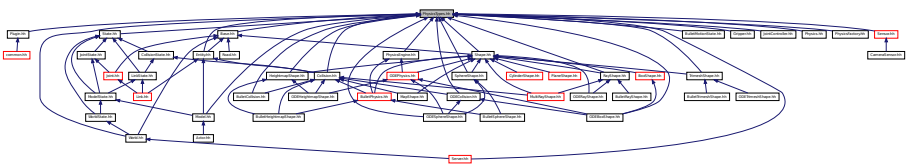
default namespace for gazebo

```
#include <vector>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for PhysicsTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Macros

- #define **GZ_ALL_COLLIDE** 0x0FFFFFFF

  Collides with everything.
- #define **GZ_FIXED_COLLIDE** 0x00000001

  Collides with everything else but other fixed.
- #define **GZ_GHOST_COLLIDE** 0x10000000

  Collides with everything else but other ghost.
- #define **GZ_NONE_COLLIDE** 0x00000000

  Collides with nothing.
- #define **GZ_SENSOR_COLLIDE** 0x00000003

  Collides with everything else but other sensors.

Typedefs

- typedef std::vector< ActorPtr > **gazebo::physics::Actor_V**
- typedef Actor * **gazebo::physics::ActorPtr**
- typedef std::vector< BasePtr > **gazebo::physics::Base_V**
- typedef Base * **gazebo::physics::BasePtr**
- typedef BoxShape * **gazebo::physics::BoxShapePtr**
- typedef std::vector< CollisionPtr > **gazebo::physics::Collision_V**
- typedef Collision * **gazebo::physics::CollisionPtr**
- typedef Contact * **gazebo::physics::ContactPtr**
- typedef CylinderShape * **gazebo::physics::CylinderShapePtr**
- typedef Entity * **gazebo::physics::EntityPtr**
- typedef HeightmapShape * **gazebo::physics::HeightmapShapePtr**
- typedef Inertial * **gazebo::physics::InertialPtr**
- typedef std::vector< JointPtr > **gazebo::physics::Joint_V**
- typedef Joint * **gazebo::physics::JointPtr**
- typedef std::vector< LinkPtr > **gazebo::physics::Link_V**
- typedef Link * **gazebo::physics::LinkPtr**
- typedef MeshShape * **gazebo::physics::MeshShapePtr**
- typedef std::vector< ModelPtr > **gazebo::physics::Model_V**
- typedef Model * **gazebo::physics::ModelPtr**
- typedef MultiRayShape * **gazebo::physics::MultiRayShapePtr**
- typedef PhysicsEngine * **gazebo::physics::PhysicsEnginePtr**
- typedef RayShape * **gazebo::physics::RayShapePtr**
- typedef **Road** * **gazebo::physics::RoadPtr**
- typedef Shape * **gazebo::physics::ShapePtr**
- typedef SphereShape * **gazebo::physics::SphereShapePtr**
- typedef SurfaceParams * **gazebo::physics::SurfaceParamsPtr**
- typedef World * **gazebo::physics::WorldPtr**

11.141.1 Detailed Description

default namespace for gazebo

11.141.2 Macro Definition Documentation

11.141.2.1 #define GZ_ALL_COLLIDE 0x0FFFFFFF

Collides with everything.

11.141.2.2 #define GZ_FIXED_COLLIDE 0x00000001

Collides with everything else but other fixed.

11.141.2.3 #define GZ_GHOST_COLLIDE 0x10000000

Collides with everything else but other ghost.

11.141.2.4 #define GZ_NONE_COLLIDE 0x00000000

Collides with nothing.

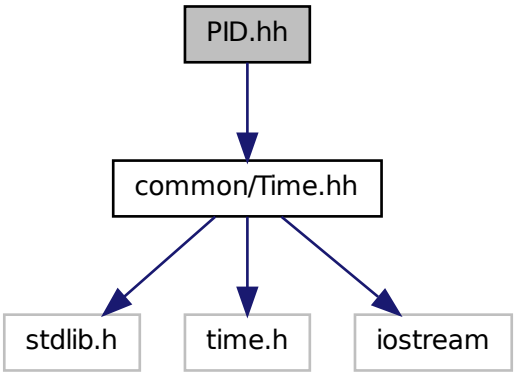
11.141.2.5 #define GZ_SENSOR_COLLIDE 0x00000003

Collides with everything else but other sensors.

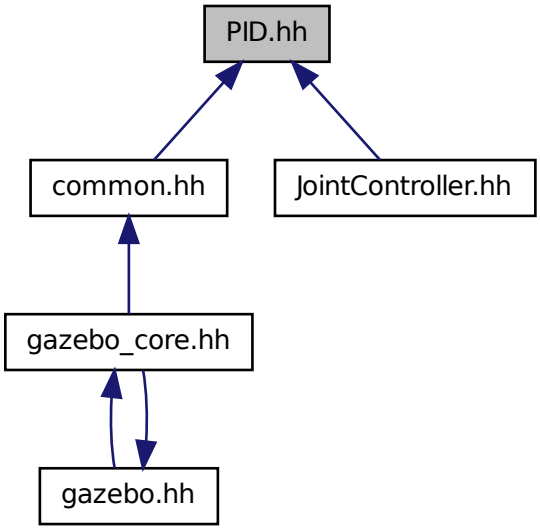
11.142 PID.hh File Reference

```
#include "common/Time.hh"
```

Include dependency graph for PID.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::PID**

  *Generic **PID** (p. 664) controller class.*

Namespaces

- namespace **gazebo**

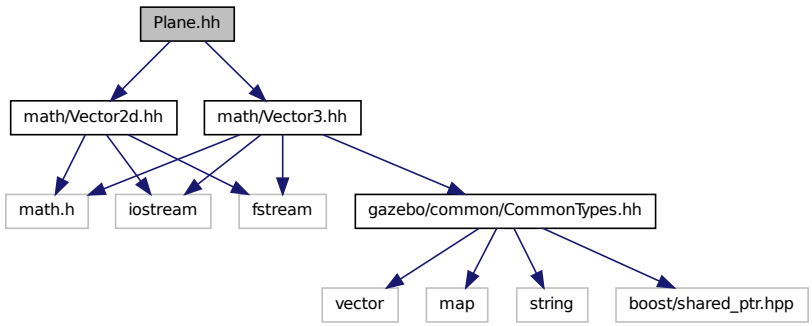
  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

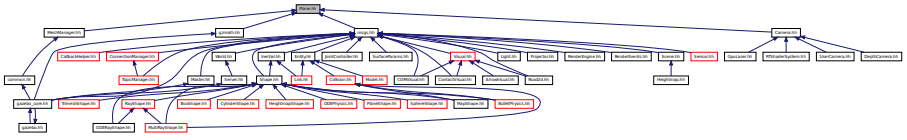
11.143 Plane.hh File Reference

```
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
```

Include dependency graph for Plane.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Plane**

  A plane and related functions.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

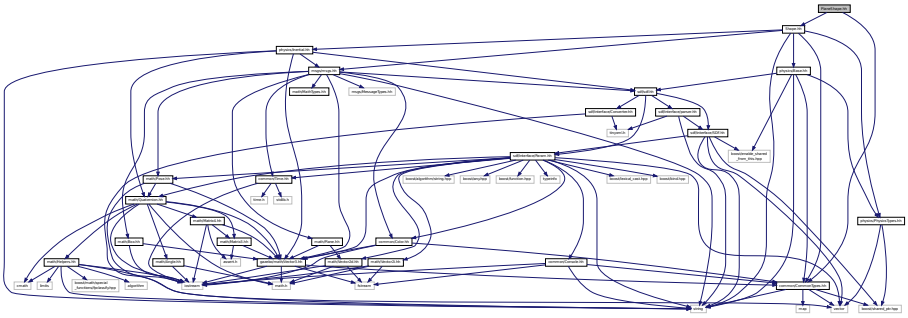
- namespace **gazebo::math**

  Math namespace.

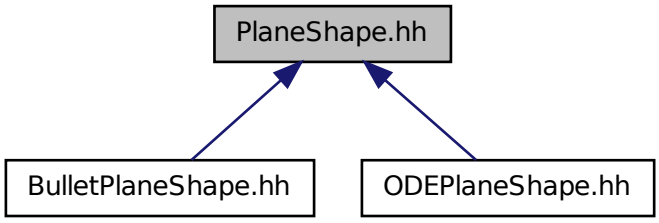
11.144 PlaneShape.hh File Reference

```
#include "common/CommonTypes.hh"
#include "Shape.hh"
```

Include dependency graph for PlaneShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::PlaneShape**

  Collision *(p. 262) for an infinite plane.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

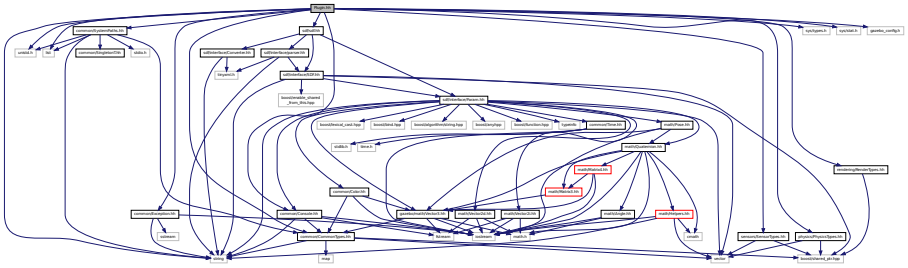
- namespace **gazebo::physics**

  namespace for physics

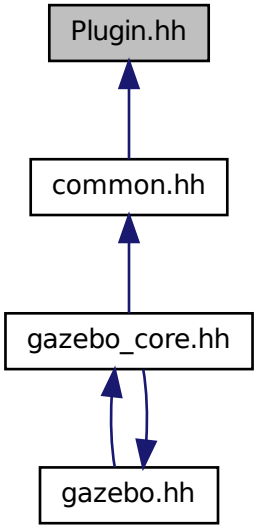
11.145 Plugin.hh File Reference

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <gazebo_config.h>
#include <list>
#include <string>
#include "common/CommonTypes.hh"
#include "common/SystemPaths.hh"
#include "common/Console.hh"
#include "common/Exception.hh"
#include "physics/PhysicsTypes.hh"
#include "sensors/SensorTypes.hh"
#include "sdf/sdf.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for common/Plugin.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::ModelPlugin**

  *A plugin with access to **physics::Model*** (p. *521*).
- class **gazebo::PluginT< T >**

  A class which all plugins must inherit from.
- class **gazebo::SensorPlugin**

  A plugin with access to physics::Sensor.
- class **gazebo::SystemPlugin**

  A plugin loaded within the gzserver on startup.
- class **gazebo::VisualPlugin**

  A plugin loaded within the gzserver on startup.
- class **gazebo::WorldPlugin**

  *A plugin with access to **physics::World*** (p. *954*).

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

Macros

- #define **GZ_REGISTER_MODEL_PLUGIN**(classname)

  Plugin registration function for model plugin.
- #define **GZ_REGISTER_SENSOR_PLUGIN**(classname)

  Plugin registration function for sensors.
- #define **GZ_REGISTER_SYSTEM_PLUGIN**(classname)

  Plugin registration function for system plugin.
- #define **GZ_REGISTER_VISUAL_PLUGIN**(classname)

  Plugin registration function for visual plugin.
- #define **GZ_REGISTER_WORLD_PLUGIN**(classname)

  Plugin registration function for world plugin.

Enumerations

- enum **gazebo::PluginType** {
  gazebo::WORLD_PLUGIN, **gazebo::MODEL_PLUGIN**, **gazebo::SENSOR_PLUGIN**, **gazebo::SYSTEM_PLUGIN**,
  gazebo::VISUAL_PLUGIN }

  Used to specify the type of plugin.

11.145.1 Macro Definition Documentation

11.145.1.1 #define GZ_REGISTER_MODEL_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::ModelPlugin *RegisterPlugin();    gazebo::ModelPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for model plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.145.1.2 #define GZ_REGISTER_SENSOR_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::SensorPlugin *RegisterPlugin();    gazebo::SensorPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for sensors.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.145.1.3 #define GZ_REGISTER_SYSTEM_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::SystemPlugin *RegisterPlugin();    gazebo::SystemPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for system plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.145.1.4 #define GZ_REGISTER_VISUAL_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::VisualPlugin *RegisterPlugin();    gazebo::VisualPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for visual plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

Returns

the name of the registered plugin

11.145.1.5 #define GZ_REGISTER_WORLD_PLUGIN(*classname*)

Value:

```
extern "C" gazebo::WorldPlugin *RegisterPlugin();    gazebo::WorldPlugin *RegisterPlugin() \
  {\
    return new classname();\
  }
```

Plugin registration function for world plugin.

Part of the shared object interface. This function is called when loading the shared library to add the plugin to the registered list.

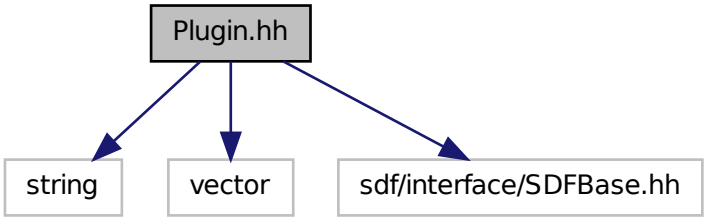
Returns

the name of the registered plugin

11.146 Plugin.hh File Reference

```
#include <string>
#include <vector>
#include "sdf/interface/SDFBase.hh"
```

Include dependency graph for sdf/interface/Plugin.hh:



Classes

- class **sdf::Plugin**

Namespaces

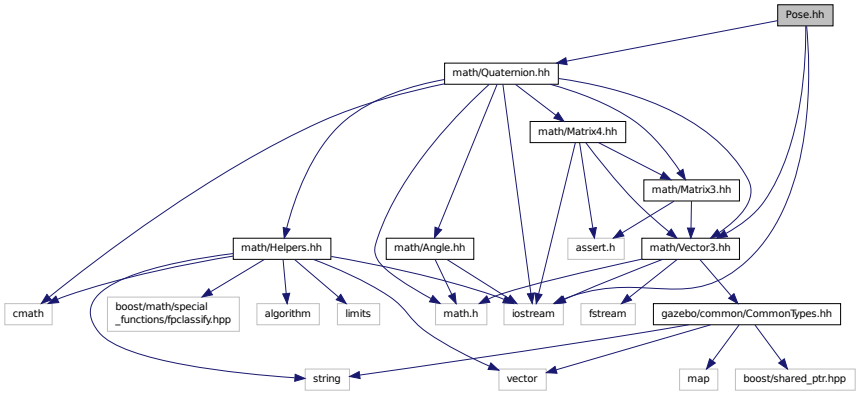
- namespace **sdf**

  namespace for Simulation Description Format parser

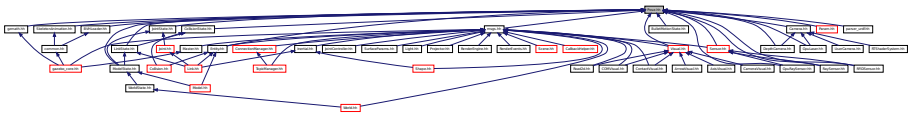
11.147 Pose.hh File Reference

```
#include <iostream>
#include "math/Vector3.hh"
#include "math/Quaternion.hh"
```

Include dependency graph for Pose.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Pose**

  Encapsulates a position and rotation in three space.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

11.148 Projector.hh File Reference

```
#include <string>
#include <map>
#include <list>
#include "rendering/ogre_gazebo.h"
#include "msgs/msgs.hh"
#include "sdf/sdf.hh"
#include "transport/transport.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for Projector.hh:

Classes

- class **gazebo::rendering::Projector**

  Projects a material onto surface, light a light projector.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

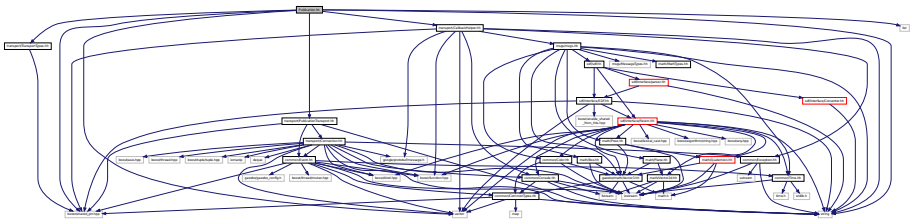
- namespace **gazebo::rendering**

  Rendering namespace.

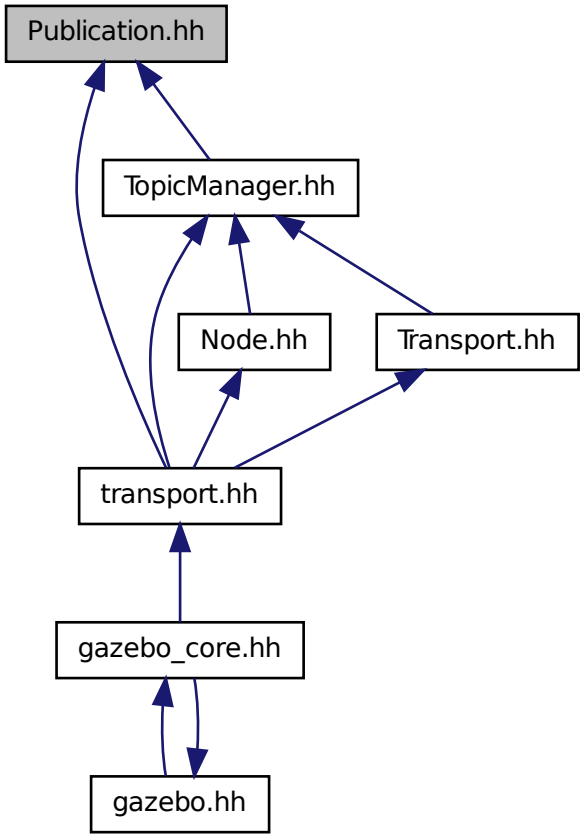
11.149 Publication.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <list>
#include <string>
#include <vector>
#include "transport/CallbackHelper.hh"
#include "transport/TransportTypes.hh"
#include "transport/PublicationTransport.hh"
```

Include dependency graph for Publication.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Publication**

  A publication for a topic.

Namespaces

- namespace **gazebo**

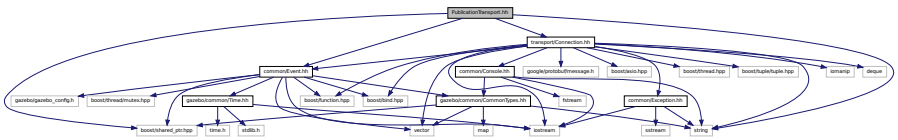
  Forward declarations for the common classes.
- namespace **gazebo::transport**

  Transport namespace.

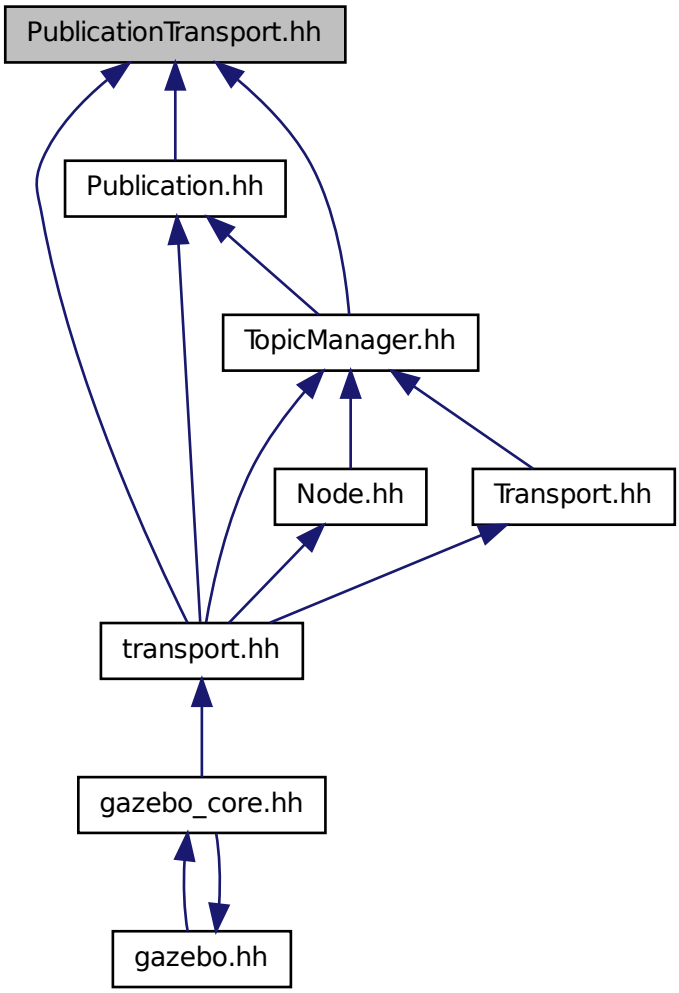
11.150 PublicationTransport.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include "transport/Connection.hh"
#include "common/Event.hh"
```

Include dependency graph for PublicationTransport.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::PublicationTransport**

  Reads data from a remote advertiser, and passes the data along to local subscribers.

Namespaces

- namespace **gazebo**

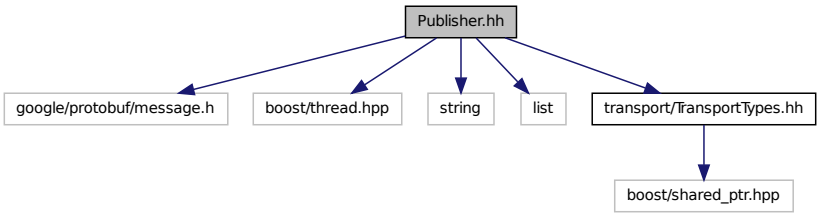
  Forward declarations for the common classes.
- namespace **gazebo::transport**

  Transport namespace.

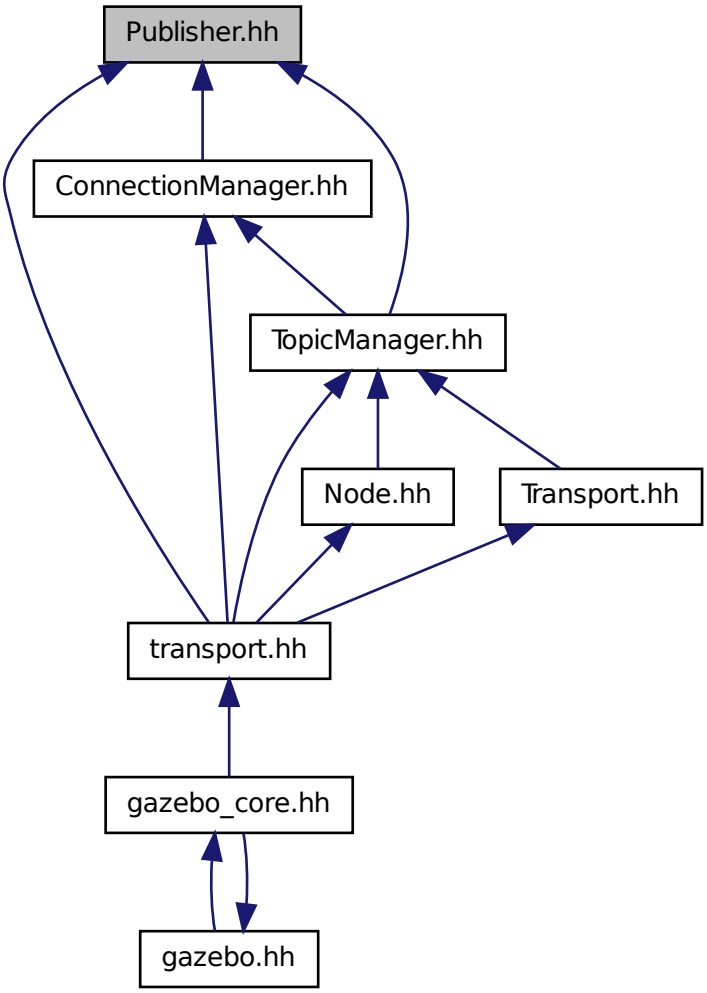
11.151 Publisher.hh File Reference

```
#include <google/protobuf/message.h>
#include <boost/thread.hpp>
#include <string>
#include <list>
#include "transport/TransportTypes.hh"
```

Include dependency graph for Publisher.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Publisher**

  A publisher of messages on a topic.

Namespaces

- namespace **gazebo**

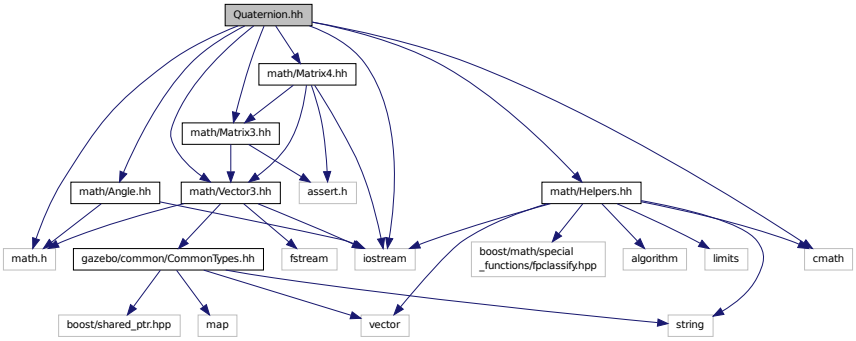
  Forward declarations for the common classes.
- namespace **gazebo::transport**

  Transport namespace.

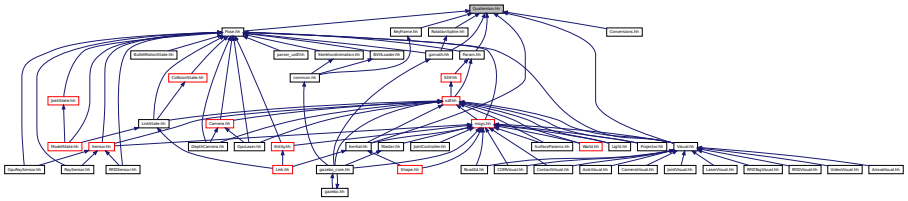
11.152 Quaternion.hh File Reference

```
#include <math.h>
#include <iostream>
#include <cmath>
#include "math/Helpers.hh"
#include "math/Angle.hh"
#include "math/Vector3.hh"
#include "math/Matrix3.hh"
#include "math/Matrix4.hh"
```

Include dependency graph for Quaternion.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Quaternion**

  A quaternion class.

Namespaces

- namespace **gazebo**

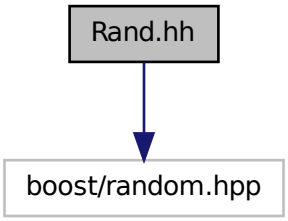
  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

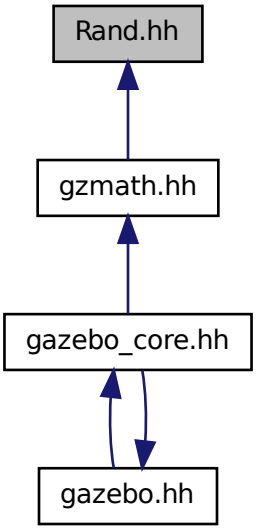
11.153 Rand.hh File Reference

```
#include <boost/random.hpp>
```

Include dependency graph for Rand.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Rand**

  Random number generator class.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

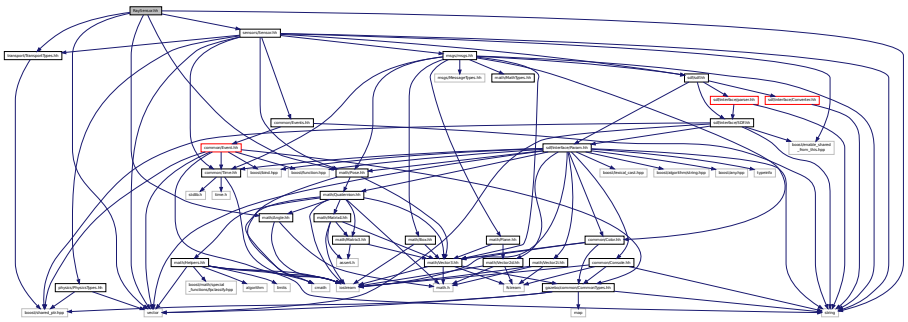
Typedefs

- typedef boost::mt19937 **gazebo::math::GeneratorType**
- typedef boost::normal_distribution < double > **gazebo::math::NormalRealDist**
- typedef boost::variate_generator < GeneratorType &, NormalRealDist > **gazebo::math::NRealGen**
- typedef boost::variate_generator < GeneratorType &, UniformIntDist > **gazebo::math::UIntGen**
- typedef boost::uniform_int< int > **gazebo::math::UniformIntDist**
- typedef boost::uniform_real < double > **gazebo::math::UniformRealDist**
- typedef boost::variate_generator < GeneratorType &, UniformRealDist > **gazebo::math::URealGen**

11.154 RaySensor.hh File Reference

```
#include <vector>
#include <string>
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "transport/TransportTypes.hh"
#include "sensors/Sensor.hh"
```

Include dependency graph for RaySensor.hh:



Classes

- class **gazebo::sensors::RaySensor**

Namespaces

- namespace **gazebo**

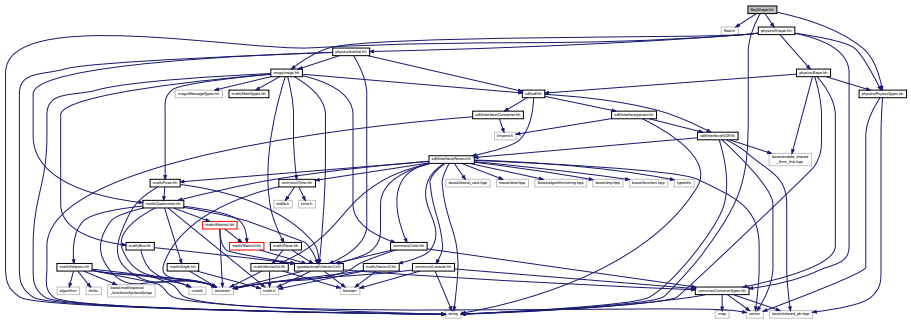
  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

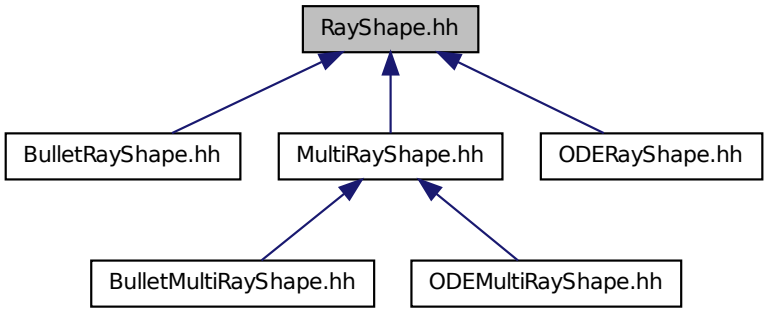
11.155 RayShape.hh File Reference

```
#include <float.h>
#include <string>
#include "physics/PhysicsTypes.hh"
#include "physics/Shape.hh"
```

Include dependency graph for RayShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::RayShape**

  Base *(p. 145) class for Ray collision geometry.*

Namespaces

- namespace **gazebo**

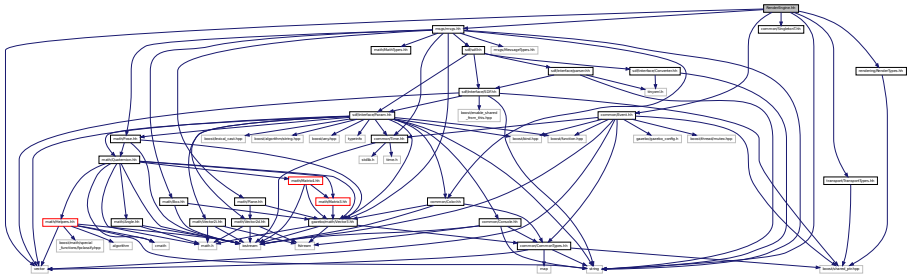
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.156 RenderEngine.hh File Reference

```
#include <vector>
#include <string>
#include "msgs/msgs.hh"
#include "common/SingletonT.hh"
#include "common/Event.hh"
#include "transport/TransportTypes.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for RenderEngine.hh:



Classes

- class **gazebo::rendering::RenderEngine**

  Adaptor to Ogre3d.

Namespaces

- namespace **gazebo**

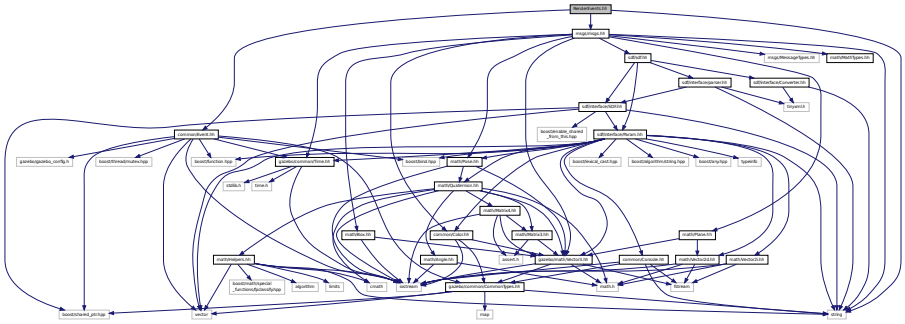
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

11.157 RenderEvents.hh File Reference

```
#include <string>
#include "common/Event.hh"
#include "msgs/msgs.hh"
```

Include dependency graph for RenderEvents.hh:



Classes

- class **gazebo::rendering::Events**

  Base class for rendering events.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

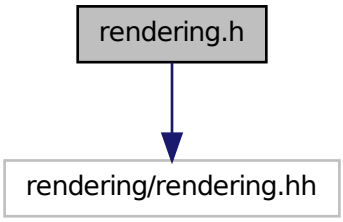
- namespace **gazebo::rendering**

  Rendering namespace.

11.158 rendering.h File Reference

```
#include "rendering/rendering.hh"
```

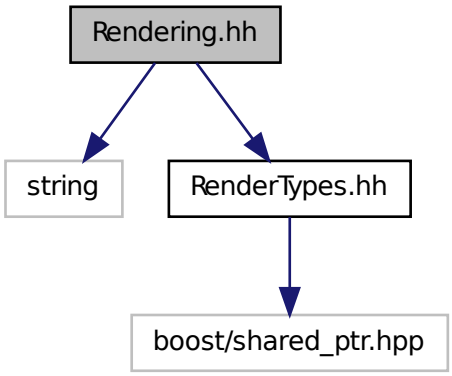
Include dependency graph for rendering.h:



11.159 Rendering.hh File Reference

```
#include <string>
#include "RenderTypes.hh"
```

Include dependency graph for Rendering.hh:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

Functions

- rendering::ScenePtr **gazebo::rendering::create_scene** (const std::string &_name, bool _enableVisualizations)

  *create **rendering::Scene*** (p. *746*) *by name.*
- bool **gazebo::rendering::fini** ()

  teardown rendering engine.
- rendering::ScenePtr **gazebo::rendering::get_scene** (const std::string &_name)

  *get pointer to **rendering::Scene*** (p. *746*) *by name.*
- bool **gazebo::rendering::init** ()

  init rendering engine.
- bool **gazebo::rendering::load** ()

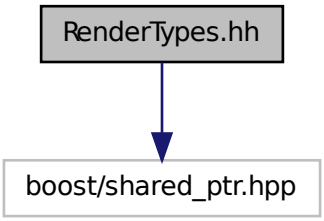
  load rendering engine.
- void **gazebo::rendering::remove_scene** (const std::string &_name)

  *remove a **rendering::Scene*** (p. *746*) *by name*

11.160 RenderTypes.hh File Reference

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for RenderTypes.hh:



This graph shows which files directly or indirectly include this file:

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

Macros

- #define **GZ_VISIBILITY_ALL** 0xFFFFFFFF

  Render everything visibility mask.
- #define **GZ_VISIBILITY_GUI** 0x00000001

  Render GUI visuals mask.
- #define **GZ_VISIBILITY_NOT_SELECTABLE** 0x00000002

  Render visuals that are not selectable mask.

Typedefs

- typedef ArrowVisual ∗ **gazebo::rendering::ArrowVisualPtr**
- typedef AxisVisual ∗ **gazebo::rendering::AxisVisualPtr**
- typedef Camera ∗ **gazebo::rendering::CameraPtr**
- typedef CameraVisual ∗ **gazebo::rendering::CameraVisualPtr**
- typedef COMVisual ∗ **gazebo::rendering::COMVisualPtr**
- typedef ContactVisual ∗ **gazebo::rendering::ContactVisualPtr**
- typedef DepthCamera ∗ **gazebo::rendering::DepthCameraPtr**
- typedef DynamicLines ∗ **gazebo::rendering::DynamicLinesPtr**
- typedef GpuLaser ∗ **gazebo::rendering::GpuLaserPtr**
- typedef JointVisual ∗ **gazebo::rendering::JointVisualPtr**
- typedef LaserVisual ∗ **gazebo::rendering::LaserVisualPtr**
- typedef Light ∗ **gazebo::rendering::LightPtr**
- typedef RFIDTagVisual ∗ **gazebo::rendering::RFIDTagVisualPtr**
- typedef RFIDVisual ∗ **gazebo::rendering::RFIDVisualPtr**
- typedef Scene ∗ **gazebo::rendering::ScenePtr**
- typedef UserCamera ∗ **gazebo::rendering::UserCameraPtr**
- typedef Visual ∗ **gazebo::rendering::VisualPtr**

Enumerations

- enum **gazebo::rendering::RenderOpType** {
  gazebo::rendering::RENDERING_POINT_LIST = 0, **gazebo::rendering::RENDERING_LINE_LIST** = 1, **gazebo::rendering::RENDERING_LINE_STRIP** = 2, **gazebo::rendering::RENDERING_TRIANGLE_LIST** = 3,
  gazebo::rendering::RENDERING_TRIANGLE_STRIP = 4, **gazebo::rendering::RENDERING_TRIANGLE_FAN** = 5, **gazebo::rendering::RENDERING_MESH_RESOURCE** = 6 }

  Type of render operation for a drawable.

11.160.1 Macro Definition Documentation

11.160.1.1 #define GZ_VISIBILITY_ALL 0xFFFFFFFF

Render everything visibility mask.

11.160.1.2 #define GZ_VISIBILITY_GUI 0x00000001

Render GUI visuals mask.

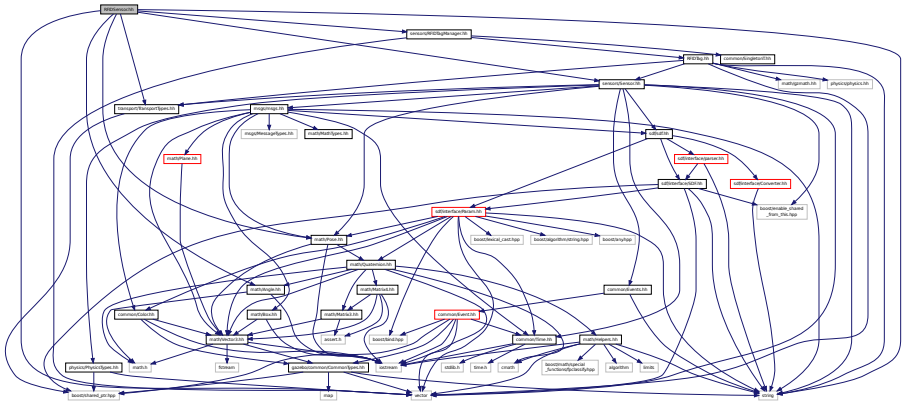
11.160.1.3 #define GZ_VISIBILITY_NOT_SELECTABLE 0x00000002

Render visuals that are not selectable mask.

11.161 RFIDSensor.hh File Reference

```
#include <vector>
#include <string>
#include "transport/TransportTypes.hh"
#include "math/Angle.hh"
#include "math/Pose.hh"
#include "sensors/Sensor.hh"
#include "sensors/RFIDTagManager.hh"
```

Include dependency graph for RFIDSensor.hh:



Classes

- class **gazebo::sensors::RFIDSensor**

  Sensor (p. *765*) *class for RFID type of sensor.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

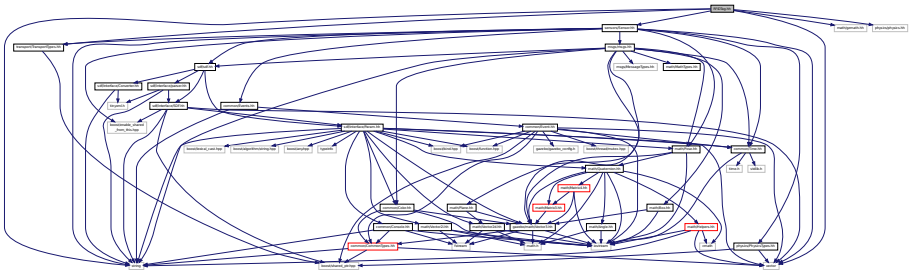
- namespace **gazebo::sensors**

  Sensors namespace.

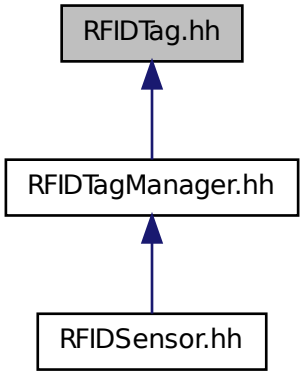
11.162 RFIDTag.hh File Reference

```
#include <vector>
#include <string>
#include "transport/TransportTypes.hh"
#include "sensors/Sensor.hh"
#include "math/gzmath.hh"
#include "physics/physics.hh"
```

Include dependency graph for RFIDTag.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::RFIDTag**

  RFIDTag (p. *729) to interact with RFIDTagSensors Nate check.*

Namespaces

- namespace **gazebo**

Forward declarations for the common classes.

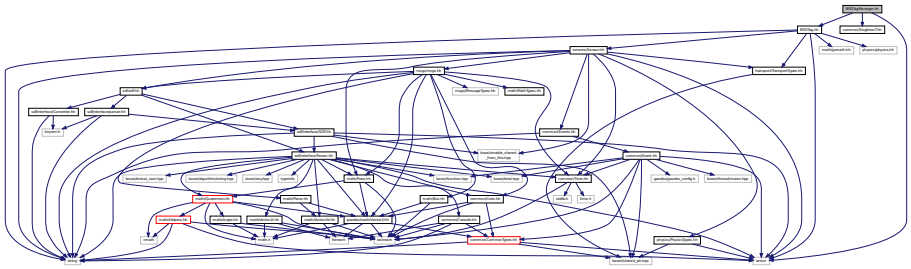
- namespace **gazebo::sensors**

  Sensors namespace.

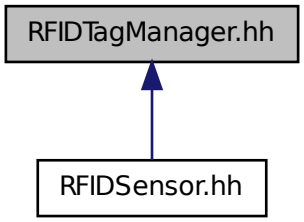
11.163 RFIDTagManager.hh File Reference

```
#include <vector>
#include "common/SingletonT.hh"
#include "RFIDTag.hh"
```

Include dependency graph for RFIDTagManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::RFIDTagManager**

  Nate fill in

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

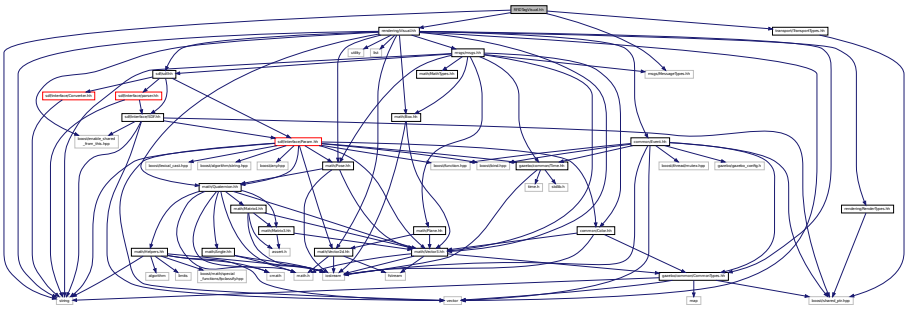
- namespace **gazebo::sensors**

  Sensors namespace.

11.164 RFIDTagVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for RFIDTagVisual.hh:



Classes

- class **gazebo::rendering::RFIDTagVisual**

  Visualization for RFID tags sensor.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

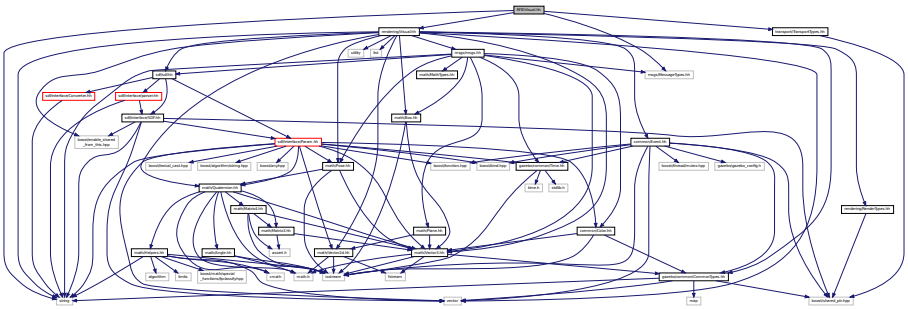
- namespace **gazebo::rendering**

  Rendering namespace.

11.165 RFIDVisual.hh File Reference

```
#include <string>
#include "rendering/Visual.hh"
#include "msgs/MessageTypes.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for RFIDVisual.hh:



Classes

- class **gazebo::rendering::RFIDVisual**

  Visualization for RFID sensor.

Namespaces

- namespace **gazebo**

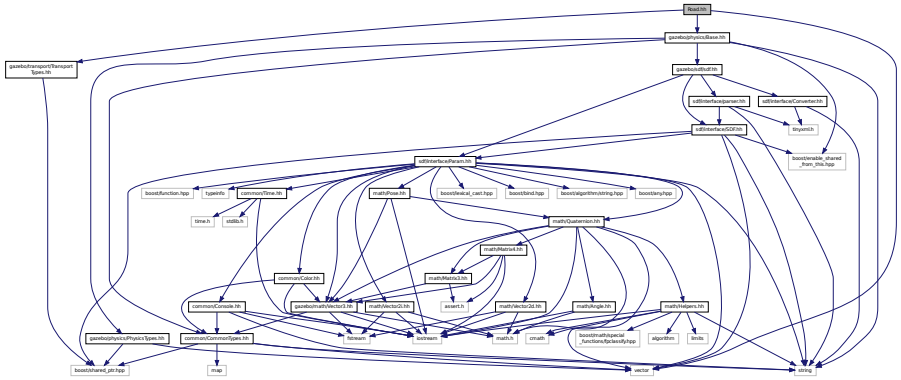
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

11.166 Road.hh File Reference

```
#include <vector>
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/physics/Base.hh"
```

Include dependency graph for Road.hh:



Classes

- class **gazebo::physics::Road**

  *for building a **Road** (p. 736) from SDF*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.167 Road2d.hh File Reference

```
#include <string>
#include <vector>
#include <list>
#include "gazebo/msgs/msgs.hh"
#include "gazebo/common/Events.hh"
#include "gazebo/transport/TransportTypes.hh"
#include "gazebo/rendering/ogre_gazebo.h"
#include "gazebo/math/Vector3.hh"
#include "gazebo/math/Spline.hh"
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for Road2d.hh:

Classes

- class **gazebo::rendering::Road2d**

Namespaces

- namespace **gazebo**

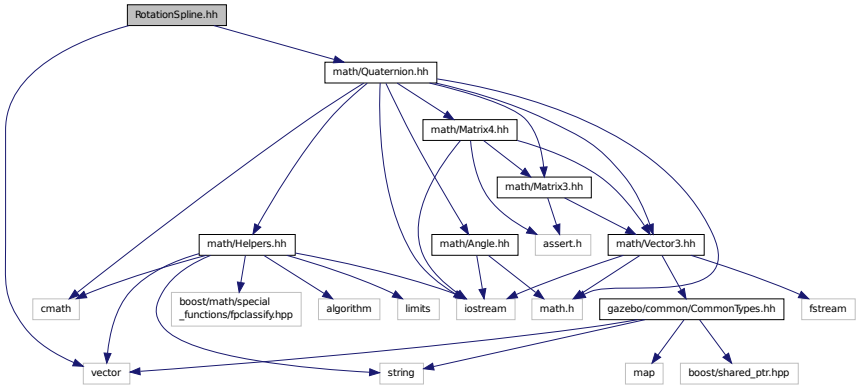
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

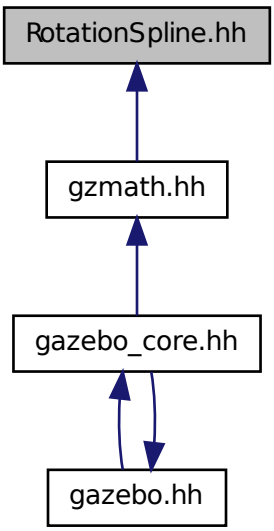
11.168 RotationSpline.hh File Reference

```
#include <vector>
#include "math/Quaternion.hh"
```

Include dependency graph for RotationSpline.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::RotationSpline**

  Spline *(p. 808) for rotations.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

11.169 RTShaderSystem.hh File Reference

```
#include <list>
#include <string>
#include <vector>
#include "rendering/ogre_gazebo.h"
#include "gazebo_config.h"
#include "rendering/Camera.hh"
#include "common/SingletonT.hh"
```

Include dependency graph for RTShaderSystem.hh:

Classes

- class **gazebo::rendering::RTShaderSystem**

  *Implements **Ogre*** (p. *118)'s Run-Time Shader system.*

Namespaces

- namespace **gazebo**

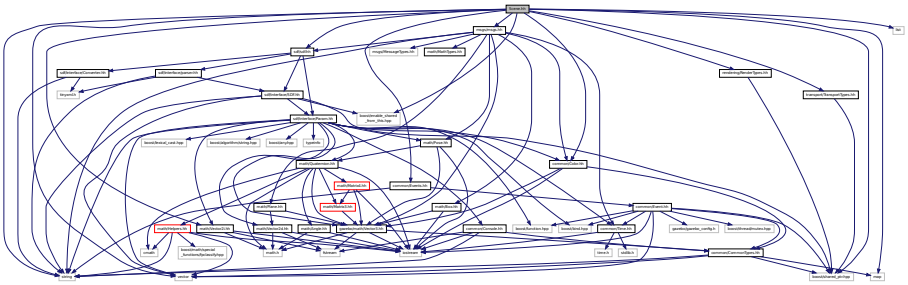
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

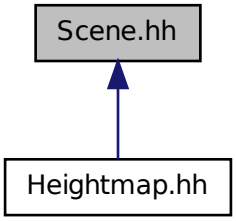
11.170 Scene.hh File Reference

```
#include <vector>
#include <map>
#include <string>
#include <list>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include "sdf/sdf.hh"
#include "msgs/msgs.hh"
#include "rendering/RenderTypes.hh"
#include "transport/TransportTypes.hh"
#include "common/Events.hh"
#include "common/Color.hh"
#include "math/Vector2i.hh"
```

Include dependency graph for Scene.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Scene**

  Representation of an entire scene graph.

Namespaces

- namespace **boost**
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::rendering**

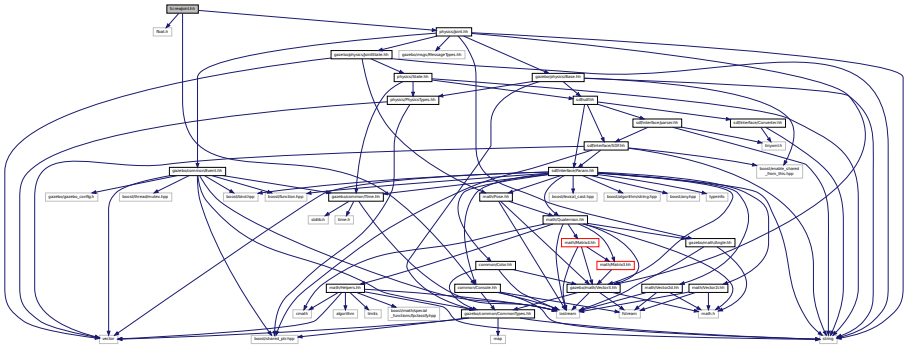
  Rendering namespace.
- namespace **Ogre**
- namespace **SkyX**

11.171 ScrewJoint.hh File Reference

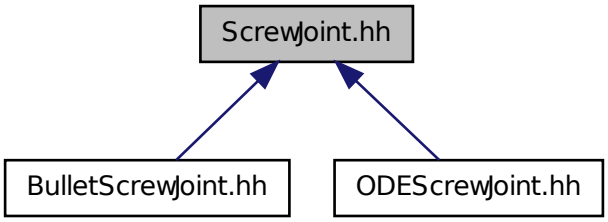
```
#include <float.h>
```

```
#include "physics/Joint.hh"
#include "gazebo/common/Console.hh"
```

Include dependency graph for ScrewJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::ScrewJoint**< **T** >

  A screw joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

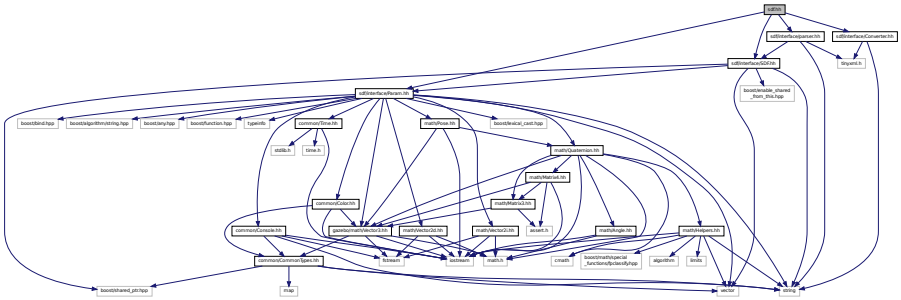
  namespace for physics

11.172 sdf.hh File Reference

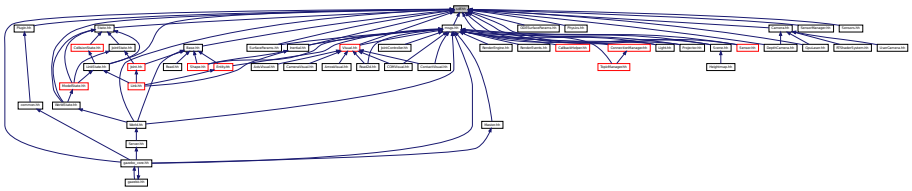
```
#include "sdf/interface/SDF.hh"
```

```
#include "sdf/interface/Param.hh"
#include "sdf/interface/parser.hh"
#include "sdf/interface/Converter.hh"
```

Include dependency graph for sdf.hh:



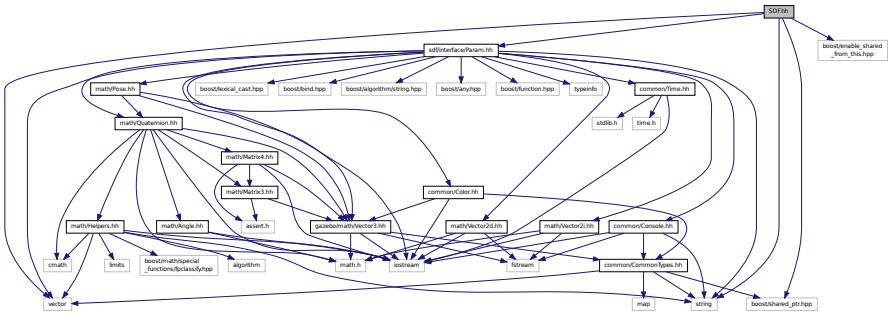
This graph shows which files directly or indirectly include this file:



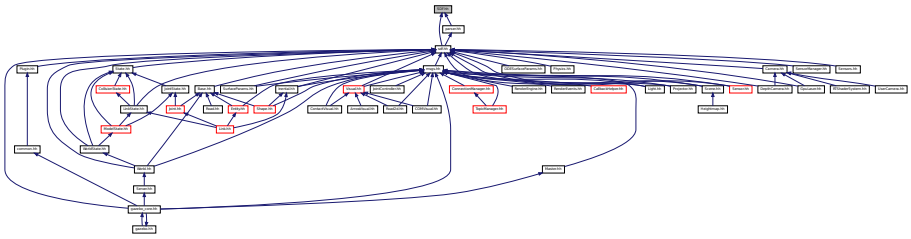
11.173 SDF.hh File Reference

```
#include <vector>
#include <string>
#include <boost/shared_ptr.hpp>
#include <boost/enable_shared_from_this.hpp>
#include "sdf/interface/Param.hh"
```

Include dependency graph for SDF.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **sdf::Element**

  SDF (p. *762*) ***Element*** (p. *332*) *class.*

- class **sdf::SDF**

  Base ***SDF*** (p. *762*) *class.*

Namespaces

- namespace **sdf**

  namespace for Simulation Description Format parser

Macros

- #define **SDF_VERSION** "1.2"

Typedefs

- typedef Element ∗ **sdf::ElementPtr**
- typedef std::vector< ElementPtr > **sdf::ElementPtr_V**
- typedef SDF ∗ **sdf::SDFPtr**

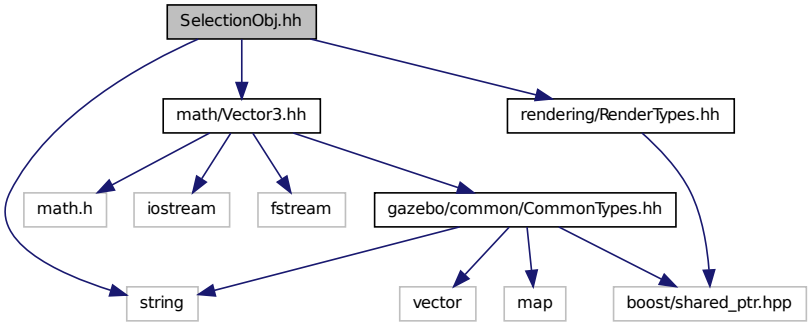
11.173.1 Macro Definition Documentation

11.173.1.1 #define SDF_VERSION "1.2"

11.174 SelectionObj.hh File Reference

```
#include <string>
#include "math/Vector3.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for SelectionObj.hh:



Classes

- class **gazebo::rendering::SelectionObj**

  A graphical selection object.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

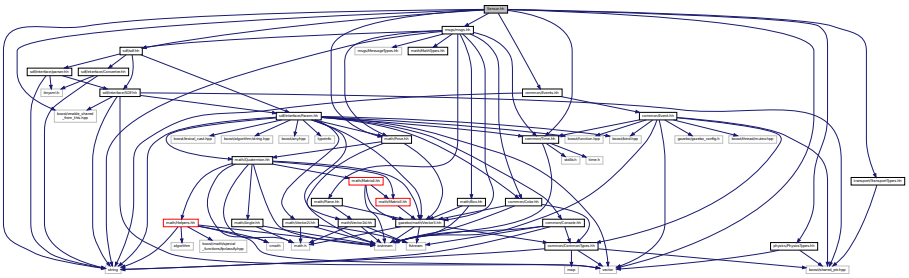
- namespace **gazebo::rendering**

  Rendering namespace.

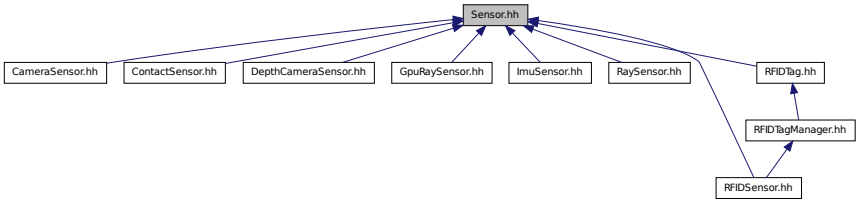
11.175 Sensor.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <vector>
#include <string>
#include "sdf/sdf.hh"
#include "physics/PhysicsTypes.hh"
#include "msgs/msgs.hh"
#include "common/Events.hh"
#include "common/Time.hh"
#include "math/Pose.hh"
#include "transport/TransportTypes.hh"
```

Include dependency graph for Sensor.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::sensors::Sensor**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

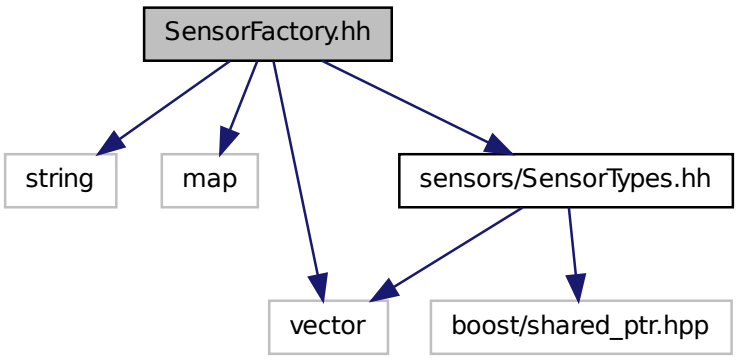
- namespace **gazebo::sensors**

  Sensors namespace.

11.176 SensorFactory.hh File Reference

```
#include <string>
#include <map>
#include <vector>
#include "sensors/SensorTypes.hh"
```

Include dependency graph for SensorFactory.hh:



Classes

- class **gazebo::sensors::SensorFactory**

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

Macros

- #define **GZ_REGISTER_STATIC_SENSOR**(name, classname)

  Static sensor registration macro.

Typedefs

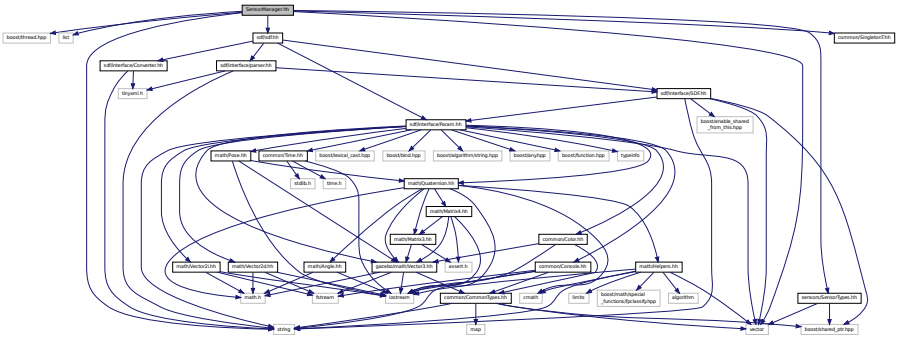
- typedef Sensor ∗(∗ **gazebo::sensors::SensorFactoryFn**)()

11.177 SensorManager.hh File Reference

```
#include <boost/thread.hpp>
```

```
#include <list>
#include <string>
#include <vector>
#include "common/SingletonT.hh"
#include "sensors/SensorTypes.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for SensorManager.hh:



Classes

- class **gazebo::sensors::SensorManager**

  Class to manage and update all sensors.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

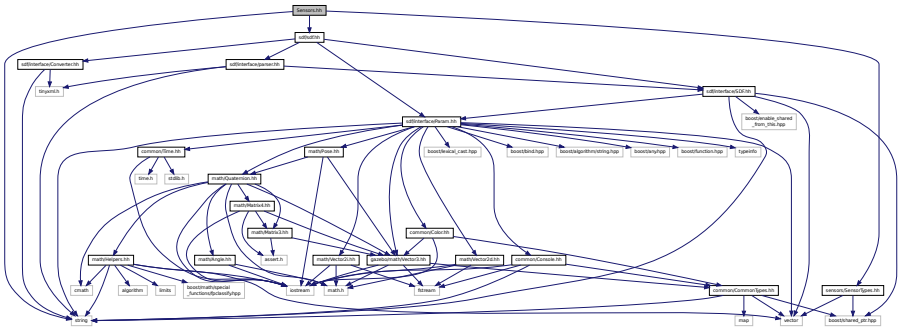
- namespace **gazebo::sensors**

  Sensors namespace.

11.178 Sensors.hh File Reference

```
#include <string>
#include "sdf/sdf.hh"
#include "sensors/SensorTypes.hh"
```

Include dependency graph for Sensors.hh:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::sensors**

  Sensors namespace.

Functions

- std::string **gazebo::sensors::create_sensor** (**sdf::ElementPtr** _elem, const std::string &_worldName, const std::string &_parentName)

  Create a sensor using SDF.
- bool **gazebo::sensors::fini** ()

  shutdown the sensor generation loop.
- SensorPtr **gazebo::sensors::get_sensor** (const std::string &_name)

  Get a sensor using by name.
- bool **gazebo::sensors::init** ()

  initialize the sensor generation loop.
- bool **gazebo::sensors::load** ()
- void **gazebo::sensors::remove_sensor** (const std::string &_sensorName)

  Remove a sensor by name.
- bool **gazebo::sensors::remove_sensors** ()

  Remove all sensors.
- void **gazebo::sensors::run** ()

  Run sensor generation continuously. This is a blocking call.
- void **gazebo::sensors::run_once** (bool _force=true)

  Run the sensor generation one step.
- void **gazebo::sensors::stop** ()

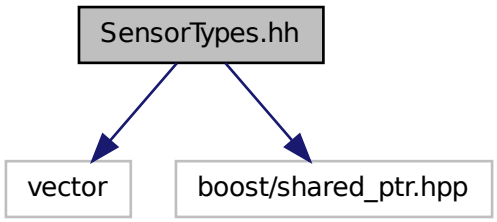
  Stop the sensor generation loop.

11.179 SensorTypes.hh File Reference

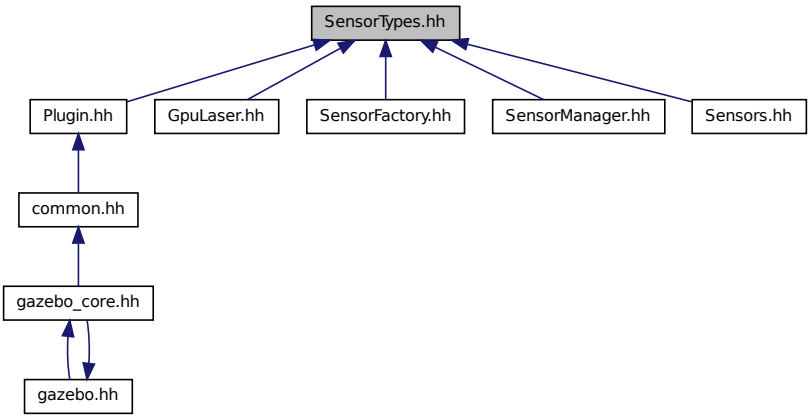
Forward declarations and typedefs for sensors.

```
#include <vector>
#include <boost/shared_ptr.hpp>
```

Include dependency graph for SensorTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::sensors**

  Sensors namespace.

Typedefs

- typedef std::vector < CameraSensorPtr > **gazebo::sensors::CameraSensor_V**
- typedef CameraSensor * **gazebo::sensors::CameraSensorPtr**
- typedef std::vector < ContactSensorPtr > **gazebo::sensors::ContactSensor_V**
- typedef ContactSensor * **gazebo::sensors::ContactSensorPtr**
- typedef std::vector < DepthCameraSensorPtr > **gazebo::sensors::DepthCameraSensor_V**
- typedef DepthCameraSensor * **gazebo::sensors::DepthCameraSensorPtr**
- typedef std::vector < GpuRaySensorPtr > **gazebo::sensors::GpuRaySensor_V**
- typedef GpuRaySensor * **gazebo::sensors::GpuRaySensorPtr**
- typedef std::vector< RaySensorPtr > **gazebo::sensors::RaySensor_V**
- typedef RaySensor * **gazebo::sensors::RaySensorPtr**
- typedef std::vector< RFIDSensor > **gazebo::sensors::RFIDSensor_V**
- typedef RFIDSensor * **gazebo::sensors::RFIDSensorPtr**
- typedef std::vector< RFIDTag > **gazebo::sensors::RFIDTag_V**
- typedef RFIDTag * **gazebo::sensors::RFIDTagPtr**
- typedef std::vector< SensorPtr > **gazebo::sensors::Sensor_V**
- typedef Sensor * **gazebo::sensors::SensorPtr**

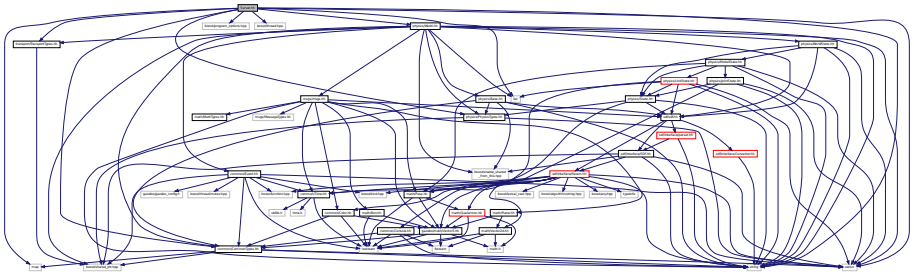
11.179.1 Detailed Description

Forward declarations and typedefs for sensors.

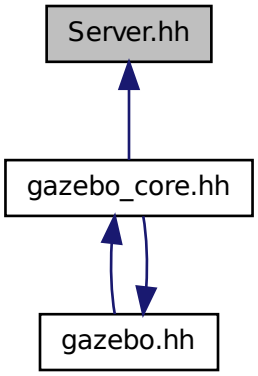
11.180 Server.hh File Reference

```
#include <string>
#include <vector>
#include <list>
#include <map>
#include <boost/program_options.hpp>
#include <boost/thread.hpp>
#include "transport/TransportTypes.hh"
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/World.hh"
```

Include dependency graph for Server.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::Server**

Namespaces

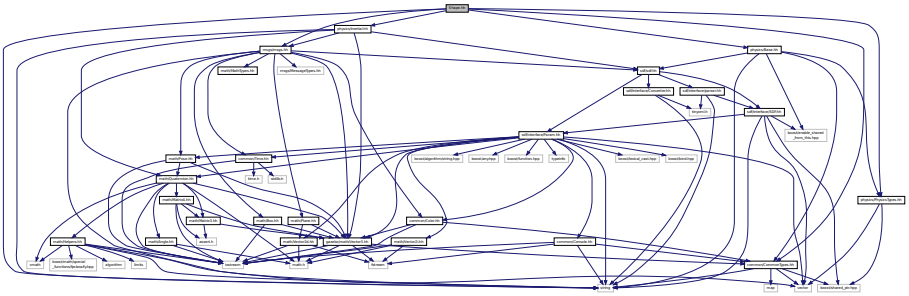
- namespace **boost**
- namespace **gazebo**

  Forward declarations for the common classes.

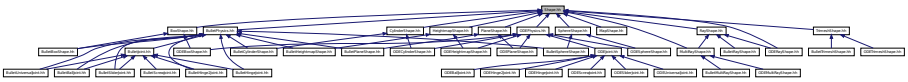
11.181 Shape.hh File Reference

```
#include <string>
#include "msgs/msgs.hh"
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/Inertial.hh"
#include "physics/Base.hh"
```

Include dependency graph for Shape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::Shape**

  Base *(p. 145) class for all shapes.*

Namespaces

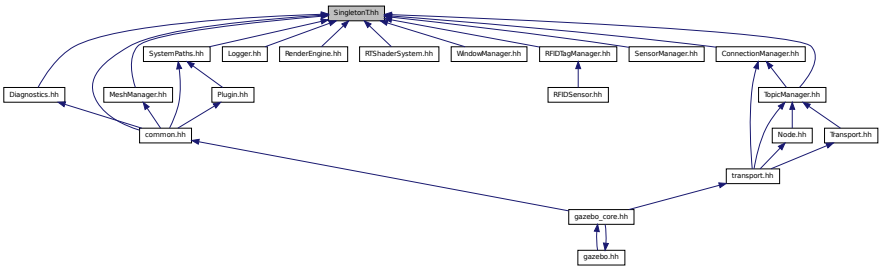
- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

11.182 SingletonT.hh File Reference

This graph shows which files directly or indirectly include this file:



Classes

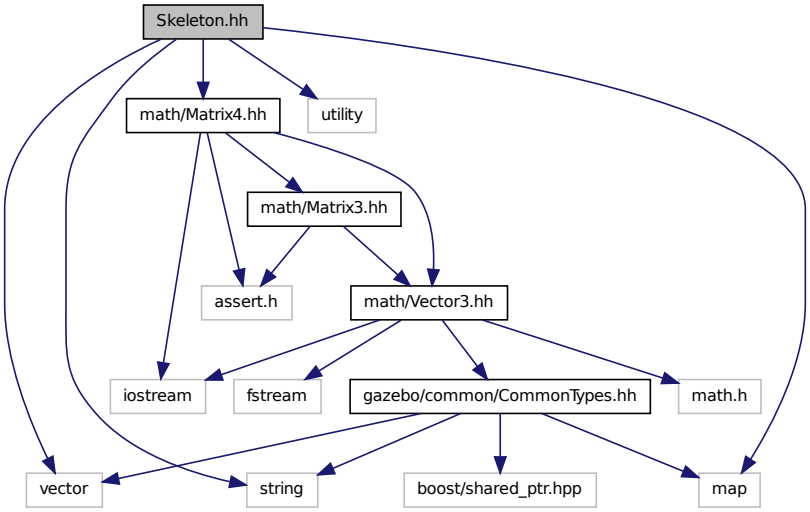
- class **SingletonT< T >**

  Singleton template class.

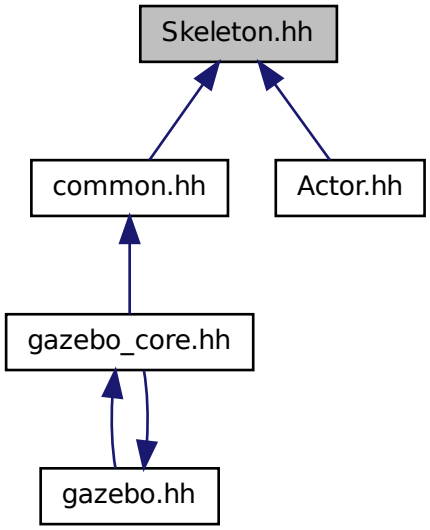
11.183 Skeleton.hh File Reference

```
#include <vector>
#include <string>
#include <map>
#include <utility>
#include "math/Matrix4.hh"
```

Include dependency graph for Skeleton.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::NodeTransform**

  A transformation node.
- class **gazebo::common::Skeleton**

  A skeleton.
- class **gazebo::common::SkeletonNode**

  A skeleton node.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Typedefs

- typedef std::map< unsigned int, SkeletonNode * > **gazebo::common::NodeMap**
- typedef std::map< unsigned int, SkeletonNode * >::iterator **gazebo::common::NodeMapIter**

- typedef std::map< double,
  std::vector< NodeTransform > > **gazebo::common::RawNodeAnim**

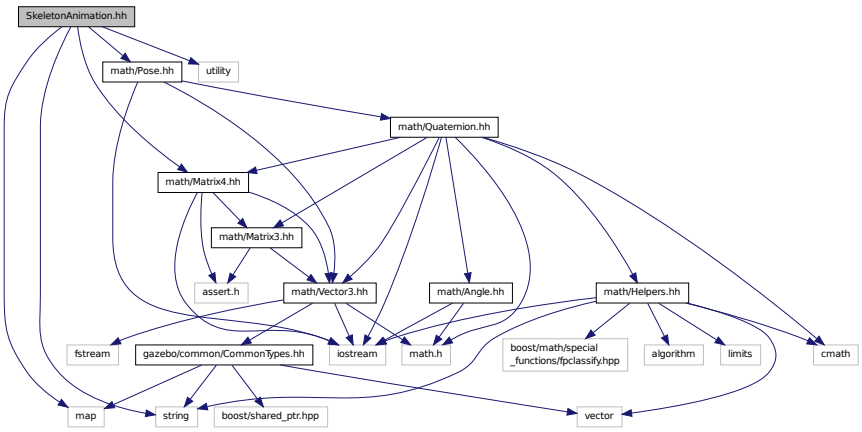
- typedef std::vector
  < std::vector< std::pair
  < std::string, double > > > **gazebo::common::RawNodeWeights**

- typedef std::map< std::string,
  RawNodeAnim > **gazebo::common::RawSkeletonAnim**

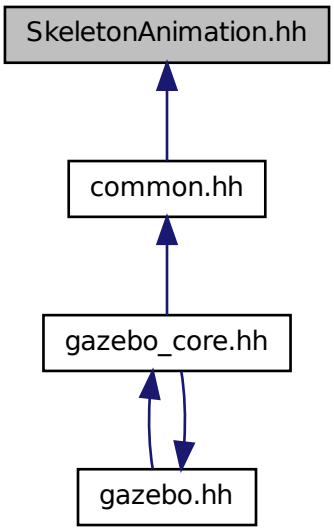
11.184 SkeletonAnimation.hh File Reference

```
#include <math/Matrix4.hh>
#include <math/Pose.hh>
#include <map>
#include <utility>
#include <string>
```

Include dependency graph for SkeletonAnimation.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::NodeAnimation**

  Node animation.

- class **gazebo::common::SkeletonAnimation**

  Skeleton *(*p. *784) animation.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

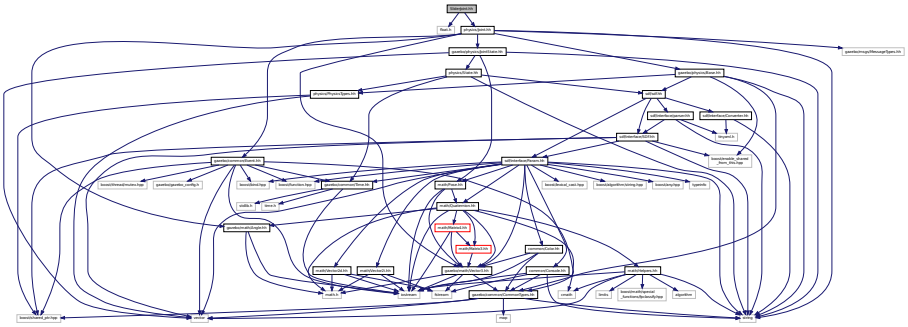
- namespace **gazebo::common**

  Common namespace.

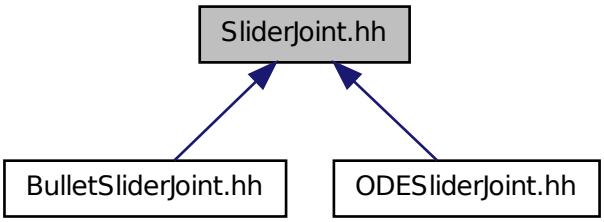
11.185 SliderJoint.hh File Reference

```
#include <float.h>
#include "physics/Joint.hh"
```

Include dependency graph for SliderJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SliderJoint**< **T** >

  A slider joint.

Namespaces

- namespace **gazebo**

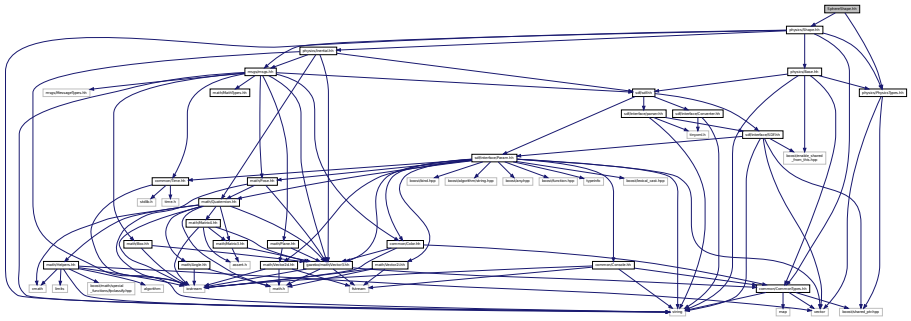
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

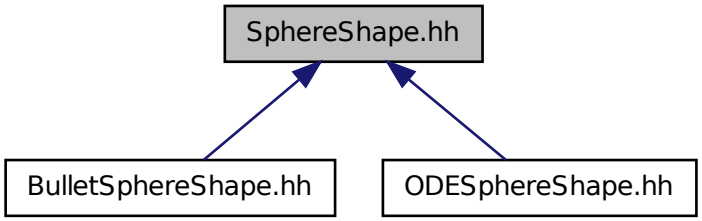
11.186 SphereShape.hh File Reference

```
#include "physics/Shape.hh"
#include "physics/PhysicsTypes.hh"
```

Include dependency graph for SphereShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::SphereShape**

  Sphere collision.

Namespaces

- namespace **gazebo**

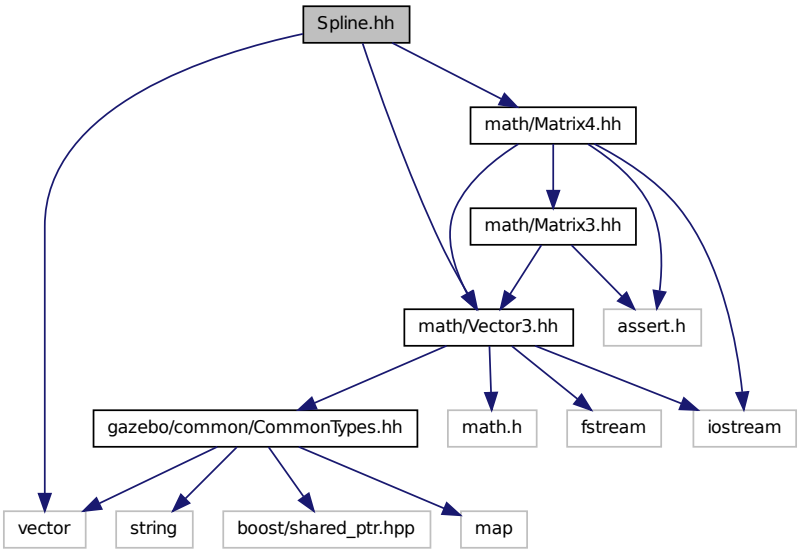
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

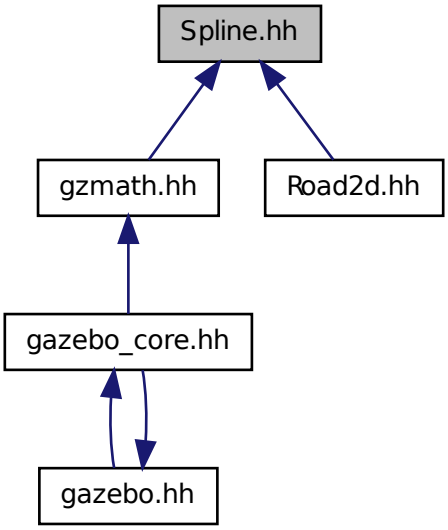
11.187 Spline.hh File Reference

```
#include <vector>
#include "math/Vector3.hh"
#include "math/Matrix4.hh"
```

Include dependency graph for Spline.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Spline**

  Splines.

Namespaces

- namespace **gazebo**

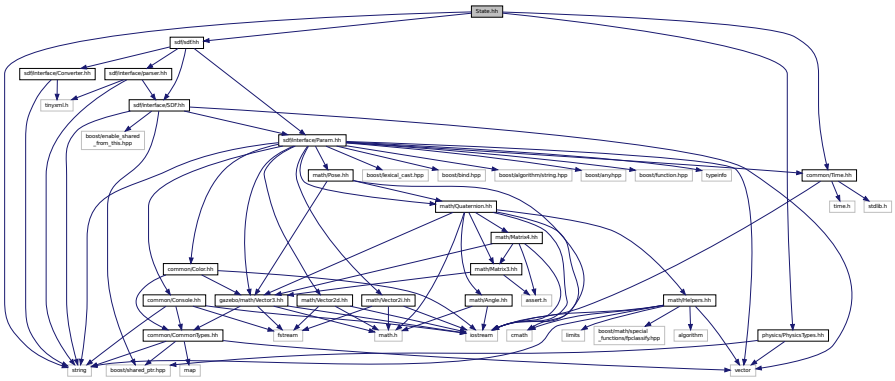
  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

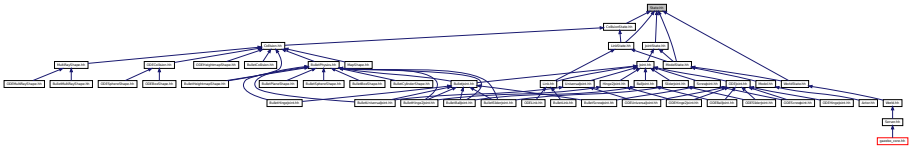
11.188 State.hh File Reference

```
#include <string>
#include "sdf/sdf.hh"
#include "physics/PhysicsTypes.hh"
#include "common/Time.hh"
```

Include dependency graph for State.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::State**

  State *(p. 813) of an entity.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

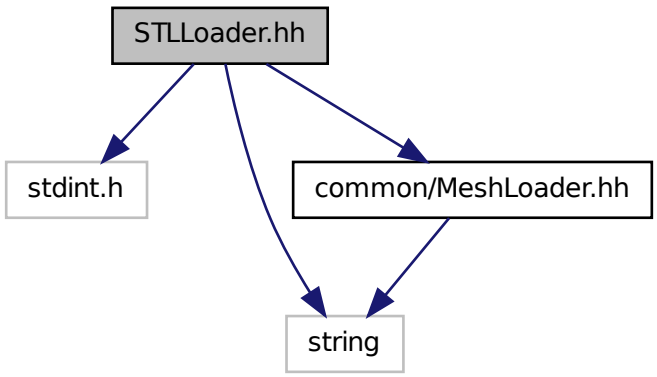
- namespace **gazebo::physics**

  namespace for physics

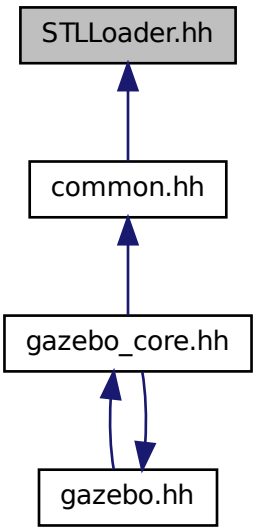
11.189 STLLoader.hh File Reference

```
#include <stdint.h>
#include <string>
#include "common/MeshLoader.hh"
```

Include dependency graph for STLLoader.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::STLLoader**

  Class used to load STL mesh files.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Macros

- #define **COR3_MAX** 200000
- #define **FACE_MAX** 200000
- #define **LINE_MAX_LEN** 256
- #define **ORDER_MAX** 10

11.189.1 Macro Definition Documentation

11.189.1.1 #define COR3_MAX 200000

11.189.1.2 #define FACE_MAX 200000

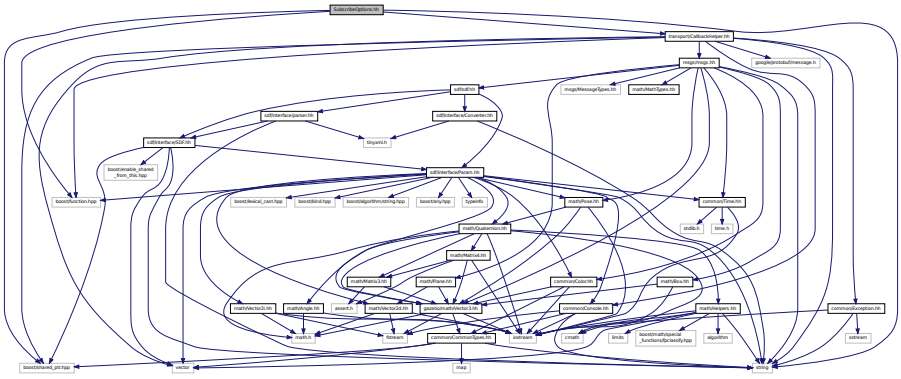
11.189.1.3 #define LINE_MAX_LEN 256

11.189.1.4 #define ORDER_MAX 10

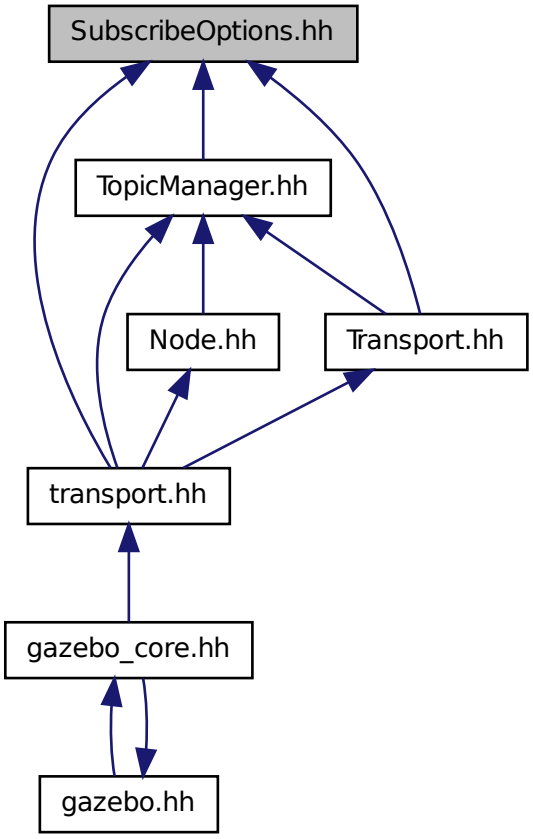
11.190 SubscribeOptions.hh File Reference

```
#include <boost/function.hpp>
#include <boost/shared_ptr.hpp>
#include <string>
#include "transport/CallbackHelper.hh"
```

Include dependency graph for SubscribeOptions.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::SubscribeOptions**

  Options for a subscription.

Namespaces

- namespace **gazebo**

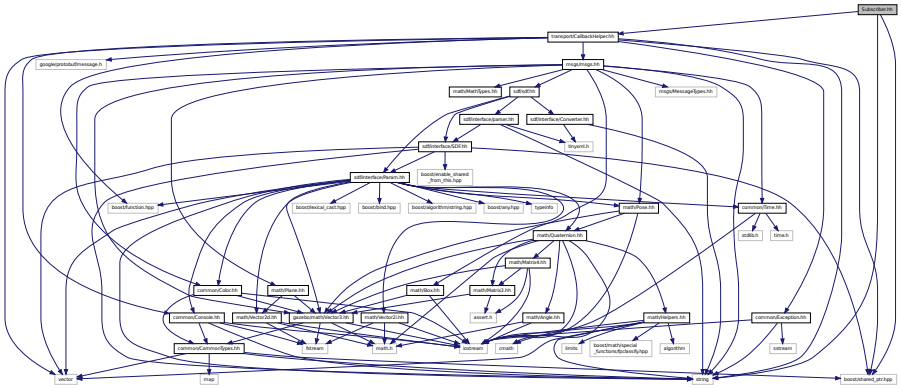
  Forward declarations for the common classes.
- namespace **gazebo::transport**

  Transport namespace.

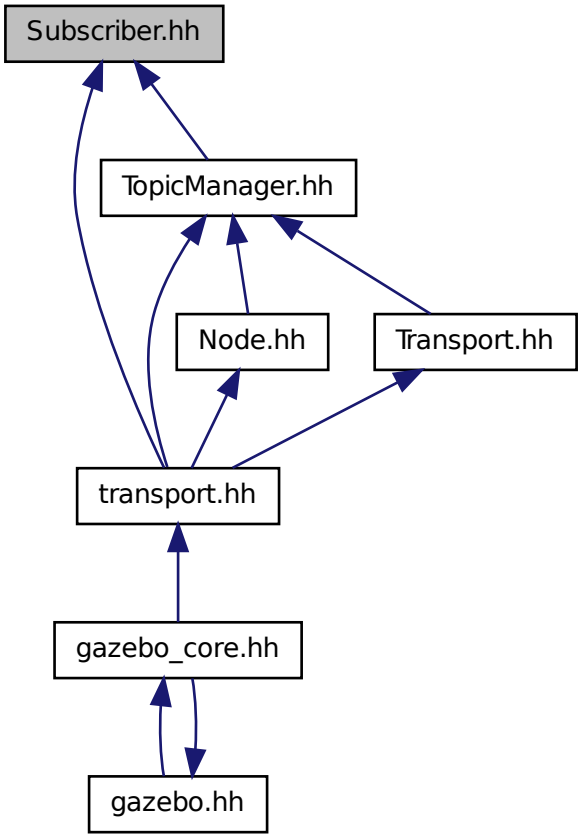
11.191 Subscriber.hh File Reference

```
#include <string>
#include <boost/shared_ptr.hpp>
#include "transport/CallbackHelper.hh"
```

Include dependency graph for Subscriber.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::Subscriber**

  A subscriber to a topic.

Namespaces

- namespace **gazebo**

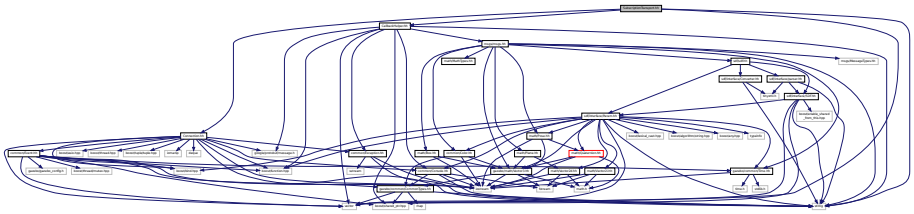
  Forward declarations for the common classes.
- namespace **gazebo::transport**

  Transport namespace.

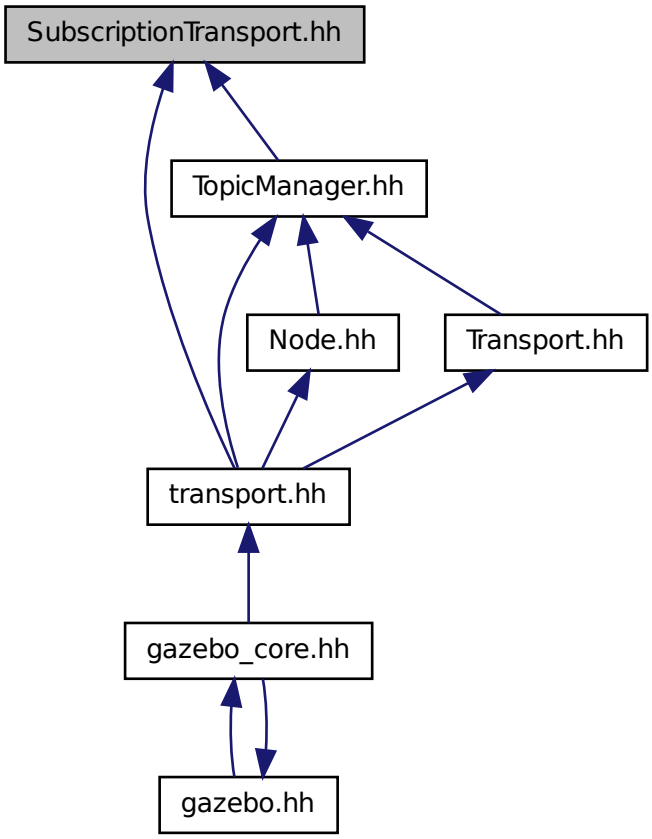
11.192 SubscriptionTransport.hh File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include "Connection.hh"
#include "CallbackHelper.hh"
```

Include dependency graph for SubscriptionTransport.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::SubscriptionTransport**

  Handles sending data over the wire to remote subscribers.

Namespaces

- namespace **gazebo**

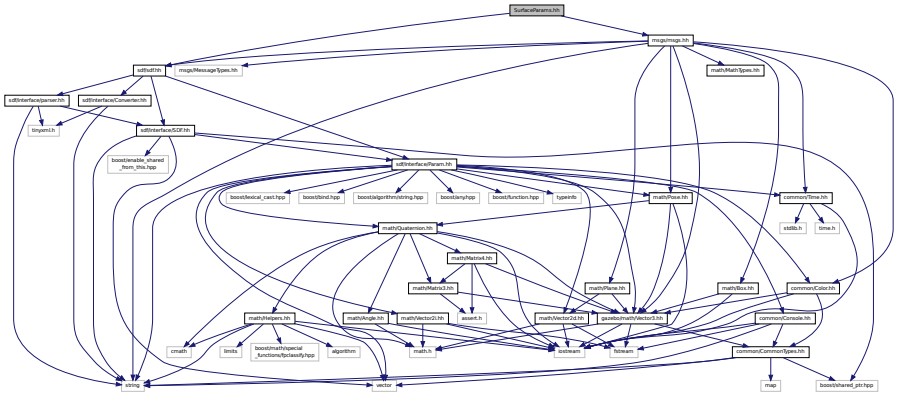
  Forward declarations for the common classes.
- namespace **gazebo::transport**

  Transport namespace.

11.193 SurfaceParams.hh File Reference

```
#include "msgs/msgs.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for SurfaceParams.hh:



Classes

- class **gazebo::physics::SurfaceParams**

  SurfaceParams *(*p. *830**) defines various Surface contact parameters.*

Namespaces

- namespace **gazebo**

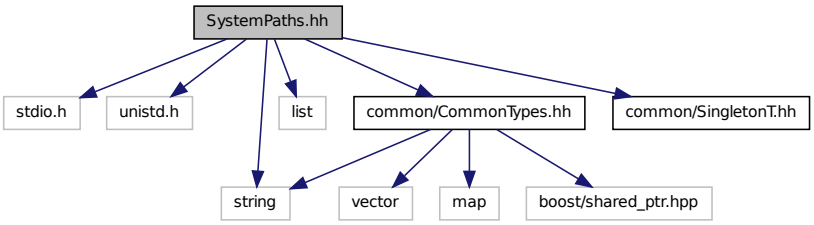
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

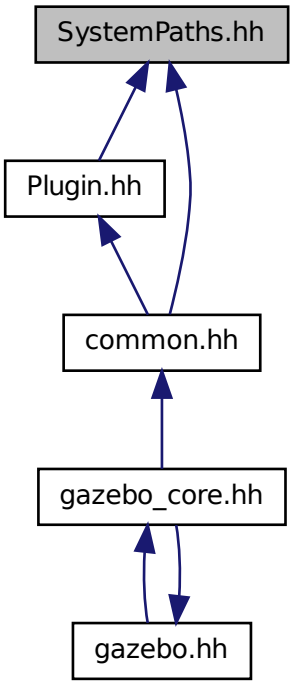
11.194 SystemPaths.hh File Reference

```
#include <stdio.h>
#include <unistd.h>
#include <string>
#include <list>
#include "common/CommonTypes.hh"
#include "common/SingletonT.hh"
```

Include dependency graph for SystemPaths.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::SystemPaths**

  Functions to handle getting system paths, keeps track of:

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

  Common namespace.

Macros

- #define **GetCurrentDir** getcwd
- #define **LINUX**

11.194.1 Macro Definition Documentation

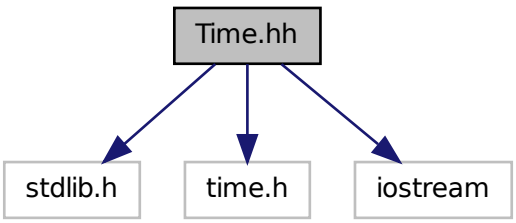
11.194.1.1 #define GetCurrentDir getcwd

11.194.1.2 #define LINUX

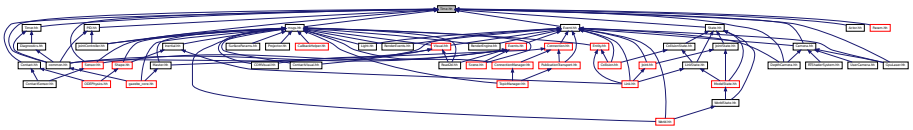
11.195 Time.hh File Reference

```
#include <stdlib.h>
#include <time.h>
#include <iostream>
```

Include dependency graph for Time.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Time**

  *A **Time** (p. 840) class, can be used to hold wall- or sim-time.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

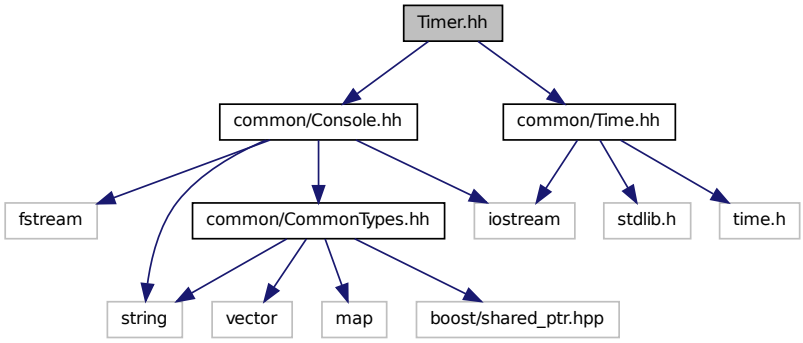
- namespace **gazebo::common**

  Common namespace.

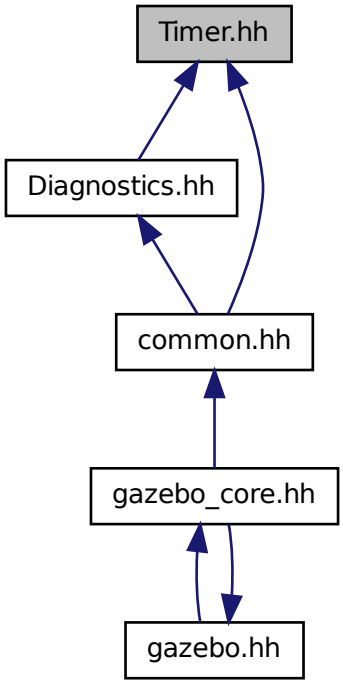
11.196 Timer.hh File Reference

```
#include "common/Console.hh"
#include "common/Time.hh"
```

Include dependency graph for Timer.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Timer**

  A timer class, used to time things in real world walltime.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

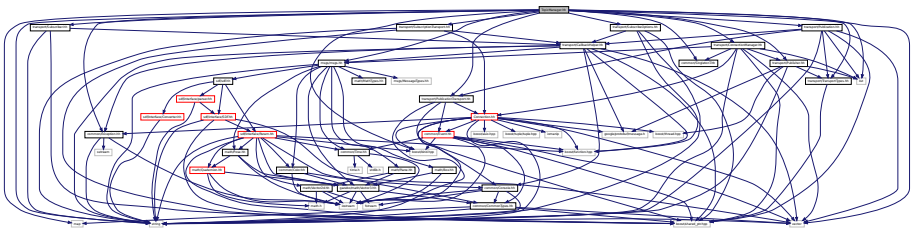
  Common namespace.

11.197 TopicManager.hh File Reference

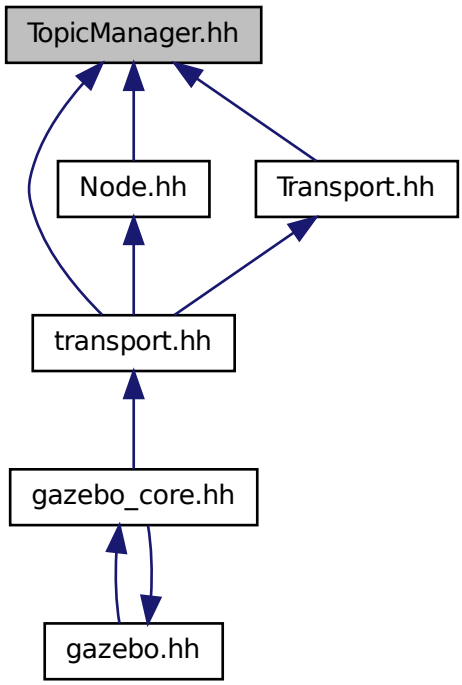
```
#include <boost/bind.hpp>
```

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include "common/Exception.hh"
#include "msgs/msgs.hh"
#include "common/SingletonT.hh"
#include "transport/TransportTypes.hh"
#include "transport/SubscribeOptions.hh"
#include "transport/SubscriptionTransport.hh"
#include "transport/PublicationTransport.hh"
#include "transport/ConnectionManager.hh"
#include "transport/Publisher.hh"
#include "transport/Publication.hh"
#include "transport/Subscriber.hh"
```

Include dependency graph for TopicManager.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::transport::TopicManager**

  Manages topics and their subscriptions.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::transport**

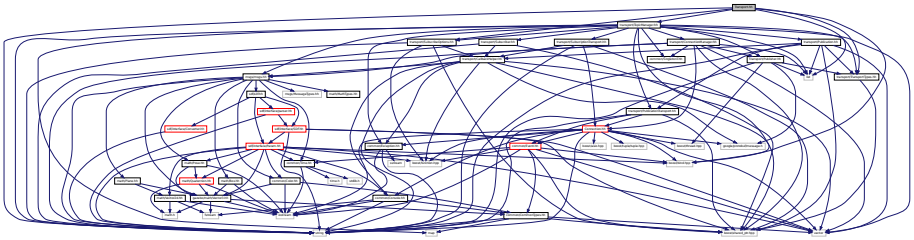
  Transport namespace.

11.198 Transport.hh File Reference

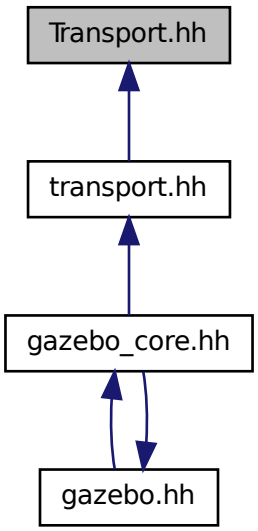
```
#include <boost/bind.hpp>
```

```
#include <string>
#include <list>
#include "transport/TransportTypes.hh"
#include "transport/SubscribeOptions.hh"
#include "transport/TopicManager.hh"
```

Include dependency graph for Transport.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::transport**

  Transport namespace.

Functions

- void **gazebo::transport::clear_buffers** ()

  clear any remaining communication buffers
- void **gazebo::transport::fini** ()

  Cleanup the transport component.
- bool **gazebo::transport::get_master_uri** (std::string &master_host, unsigned int &master_port)

  Get the hostname and port of the master from the GAZEBO_MASTER_URI environment variable.
- void **gazebo::transport::get_topic_namespaces** (std::list< std::string > &_namespaces)

  Return all the namespace (world names) on the master.
- bool **gazebo::transport::init** (const std::string &master_host="", unsigned int master_port=0)

  Initialize the transport system.
- bool **gazebo::transport::is_stopped** ()

  Return true if the transport system is stopped.
- void **gazebo::transport::pause_incoming** (bool _pause)

  Set to true to pause incoming messages.
- msgs::Response **gazebo::transport::request** (const std::string &_worldName, const msgs::Request &_request)

  Send a request, and receive a response.
- void **gazebo::transport::run** ()

  Run the transport component.
- void **gazebo::transport::stop** ()

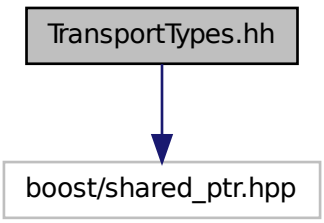
  Stop the transport component from running.

11.199 TransportTypes.hh File Reference

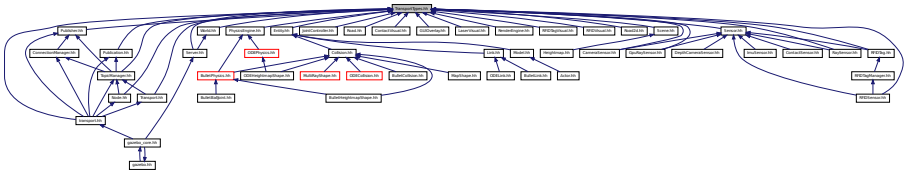
Forward declarations for transport.

```
#include <boost/shared_ptr.hpp>
```

Include dependency graph for TransportTypes.hh:



This graph shows which files directly or indirectly include this file:



Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::transport**

  Transport namespace.

Typedefs

- typedef Node ∗ **gazebo::transport::NodePtr**
- typedef Publication ∗ **gazebo::transport::PublicationPtr**
- typedef PublicationTransport ∗ **gazebo::transport::PublicationTransportPtr**
- typedef Publisher ∗ **gazebo::transport::PublisherPtr**
- typedef Subscriber ∗ **gazebo::transport::SubscriberPtr**
- typedef SubscriptionTransport ∗ **gazebo::transport::SubscriptionTransportPtr**

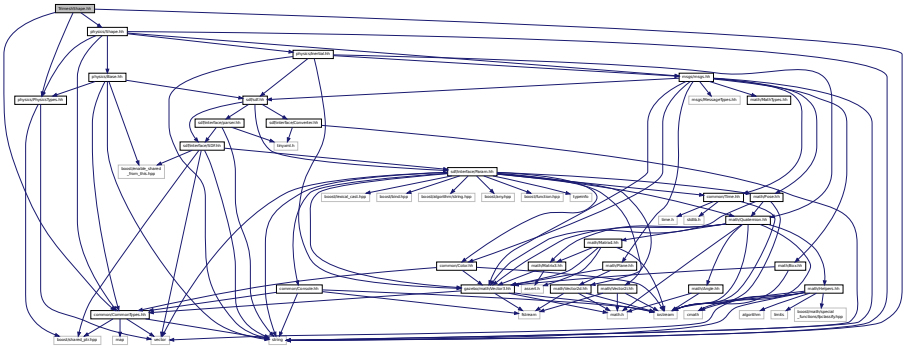
11.199.1 Detailed Description

Forward declarations for transport.

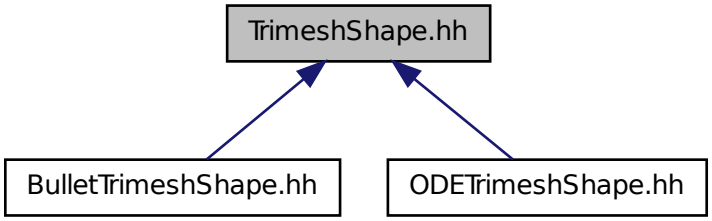
11.200 TrimeshShape.hh File Reference

```
#include <string>
#include "common/CommonTypes.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/Shape.hh"
```

Include dependency graph for TrimeshShape.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::TrimeshShape**

  Triangle mesh collision shape.

Namespaces

- namespace **gazebo**

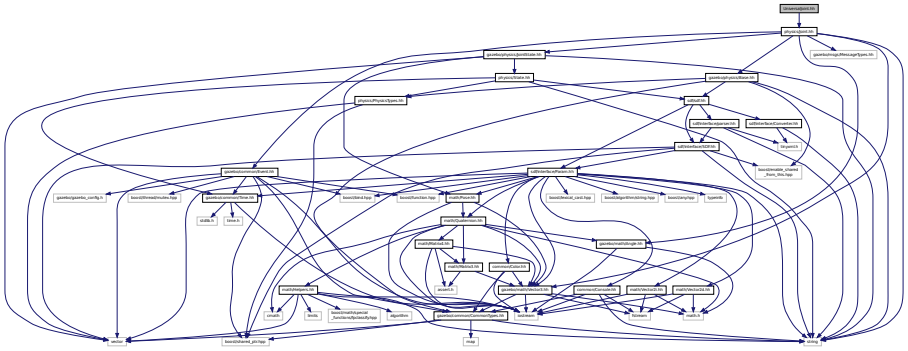
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

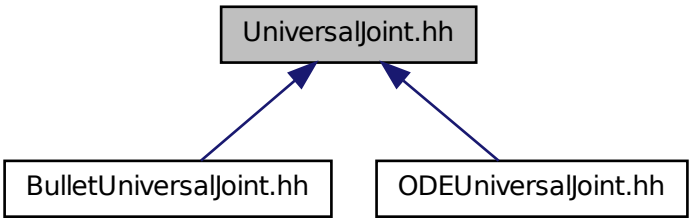
11.201 UniversalJoint.hh File Reference

```
#include "physics/Joint.hh"
```

Include dependency graph for UniversalJoint.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::UniversalJoint**< **T** >

  A universal joint.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

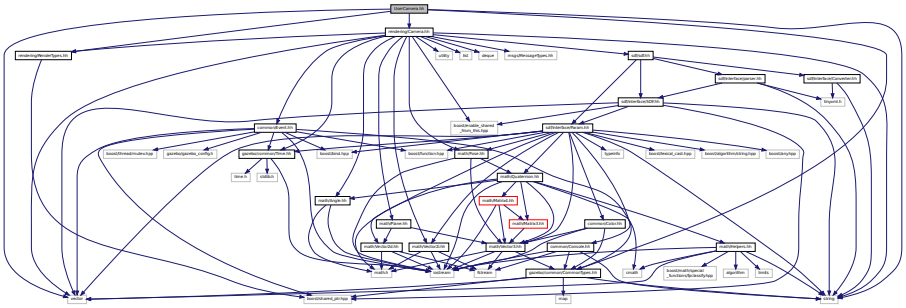
  namespace for physics

11.202 UserCamera.hh File Reference

```
#include <string>
```

```
#include <vector>
#include "rendering/Camera.hh"
#include "rendering/RenderTypes.hh"
#include "common/CommonTypes.hh"
```

Include dependency graph for UserCamera.hh:



Classes

- class **gazebo::rendering::UserCamera**

  A camera used for user visualization of a scene.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

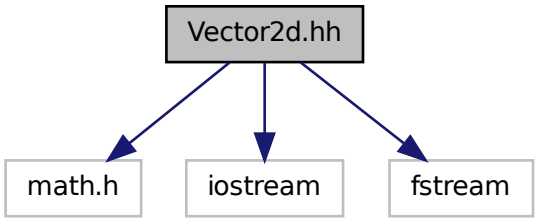
- namespace **gazebo::rendering**

  Rendering namespace.

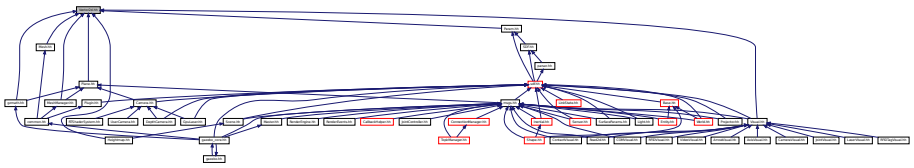
11.203 Vector2d.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
```

Include dependency graph for Vector2d.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector2d**

  Generic double x, y vector.

Namespaces

- namespace **gazebo**

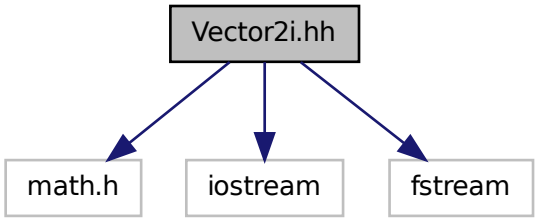
  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

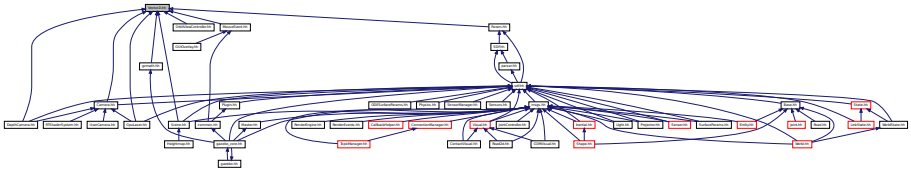
11.204 Vector2i.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
```

Include dependency graph for Vector2i.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector2i**

  Generic integer x, y vector.

Namespaces

- namespace **gazebo**

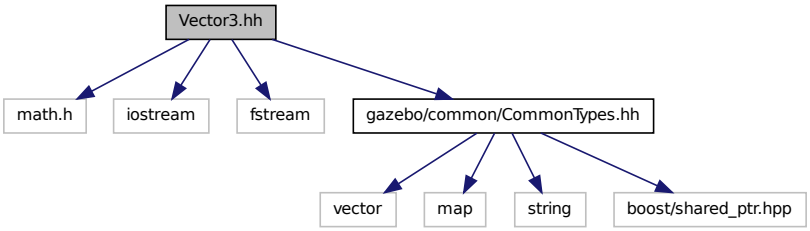
  Forward declarations for the common classes.
- namespace **gazebo::math**

  Math namespace.

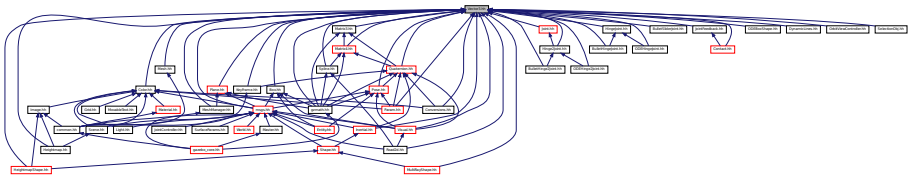
11.205 Vector3.hh File Reference

```
#include <math.h>
#include <iostream>
#include <fstream>
#include "gazebo/common/CommonTypes.hh"
```

Include dependency graph for Vector3.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector3**

  *The **Vector3** (p. 902) class represents the generic vector containing 3 elements.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

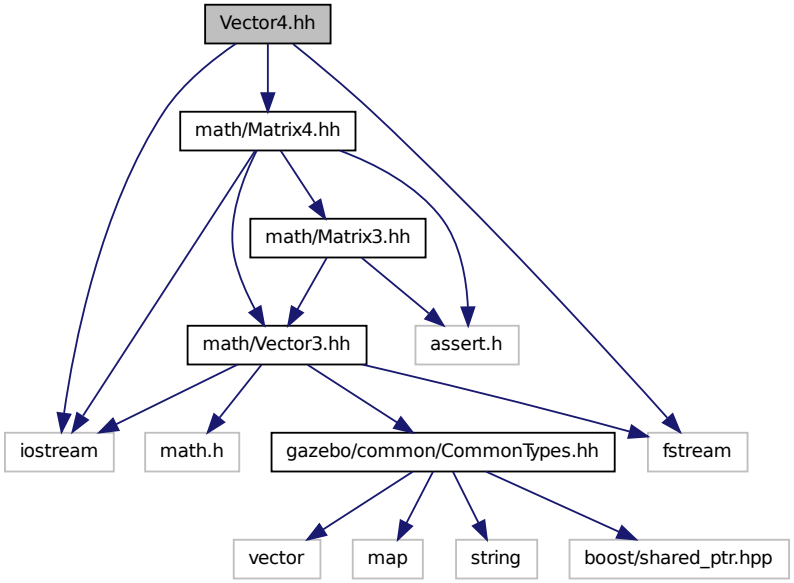
- namespace **gazebo::math**

  Math namespace.

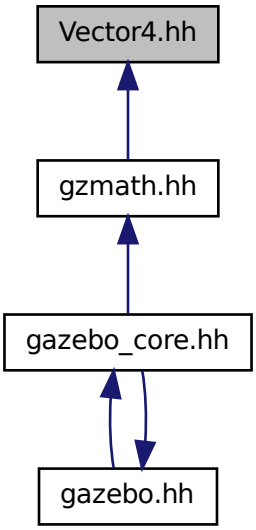
11.206 Vector4.hh File Reference

```
#include <iostream>
#include <fstream>
#include "math/Matrix4.hh"
```

Include dependency graph for Vector4.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::math::Vector4**

  double Generic x, y, z, w vector

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

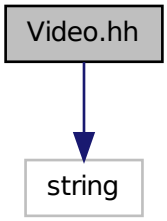
- namespace **gazebo::math**

  Math namespace.

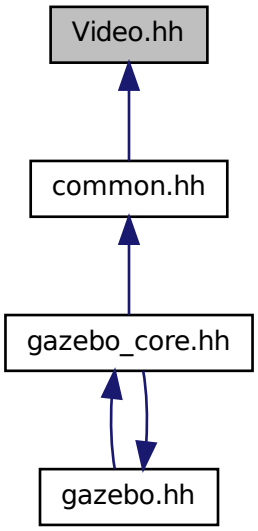
11.207 Video.hh File Reference

```
#include <string>
```

Include dependency graph for Video.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::common::Video**

  Handle video encoding and decoding using libavcodec.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

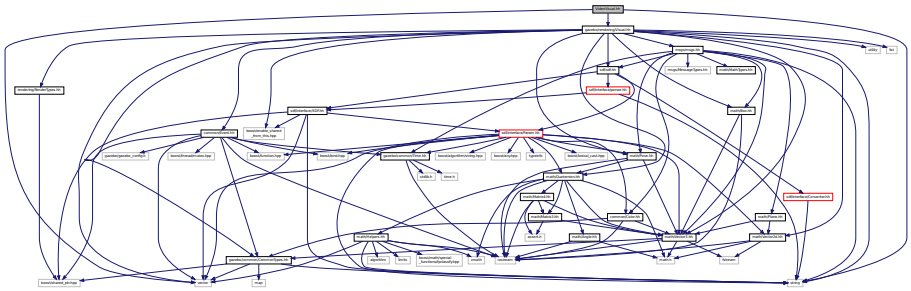
- namespace **gazebo::common**

  Common namespace.

11.208 VideoVisual.hh File Reference

```
#include <string>
#include <vector>
#include "gazebo/rendering/Visual.hh"
```

Include dependency graph for VideoVisual.hh:



Classes

- class **gazebo::rendering::VideoVisual**

  A visual element that displays a video as a texture.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::common**

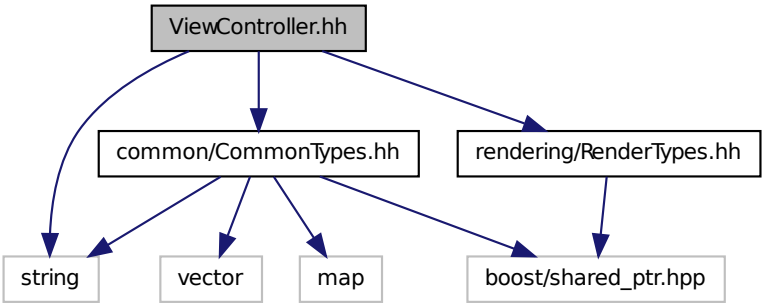
  Common namespace.
- namespace **gazebo::rendering**

  Rendering namespace.

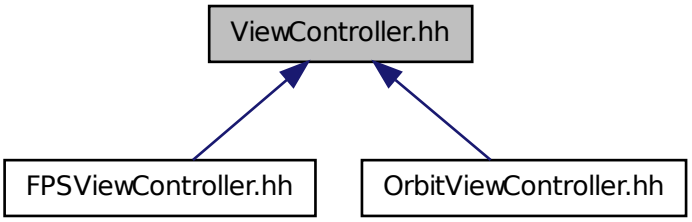
11.209 ViewController.hh File Reference

```
#include <string>
#include "common/CommonTypes.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for ViewController.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::ViewController**

  Base class for view controllers.

Namespaces

- namespace **gazebo**

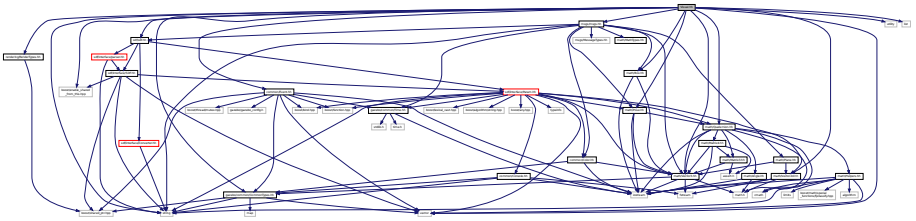
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.

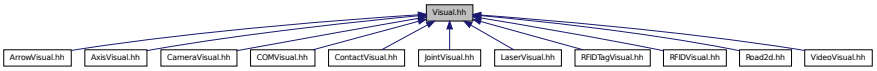
11.210 Visual.hh File Reference

```
#include <boost/enable_shared_from_this.hpp>
#include <string>
#include <utility>
#include <list>
#include <vector>
#include "common/Event.hh"
#include "math/Box.hh"
#include "math/Pose.hh"
#include "math/Quaternion.hh"
#include "math/Vector3.hh"
#include "math/Vector2d.hh"
#include "sdf/sdf.hh"
#include "msgs/msgs.hh"
#include "rendering/RenderTypes.hh"
#include "common/CommonTypes.hh"
```

Include dependency graph for Visual.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::rendering::Visual**

  A renderable object.

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.

- namespace **gazebo::rendering**

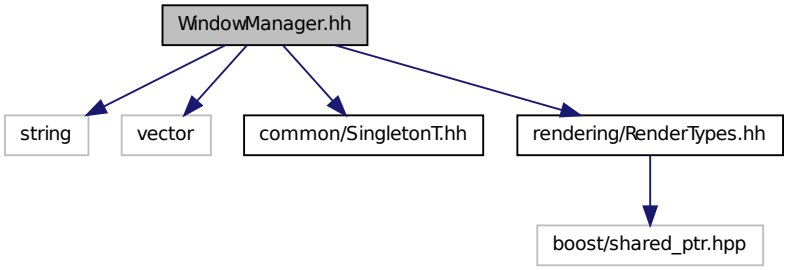
  Rendering namespace.

- namespace **Ogre**

11.211 WindowManager.hh File Reference

```
#include <string>
#include <vector>
#include "common/SingletonT.hh"
#include "rendering/RenderTypes.hh"
```

Include dependency graph for WindowManager.hh:



Classes

- class **gazebo::rendering::WindowManager**

  Class to mangage render windows.

Namespaces

- namespace **gazebo**

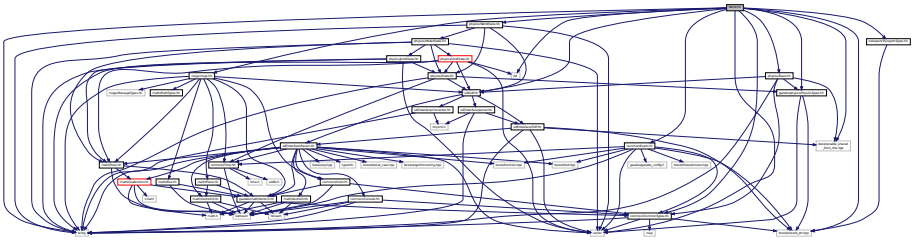
  Forward declarations for the common classes.
- namespace **gazebo::rendering**

  Rendering namespace.
- namespace **Ogre**

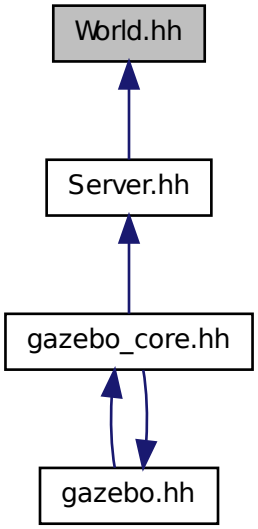
11.212 World.hh File Reference

```
#include <vector>
#include <list>
#include <string>
#include <boost/enable_shared_from_this.hpp>
#include <boost/shared_ptr.hpp>
#include "transport/TransportTypes.hh"
#include "msgs/msgs.hh"
#include "common/CommonTypes.hh"
#include "common/Event.hh"
#include "physics/Base.hh"
#include "physics/PhysicsTypes.hh"
#include "physics/WorldState.hh"
#include "sdf/sdf.hh"
```

Include dependency graph for World.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::World**

  The world provides access to all other object within a simulated environment.

Namespaces

- namespace **boost**
- namespace **gazebo**

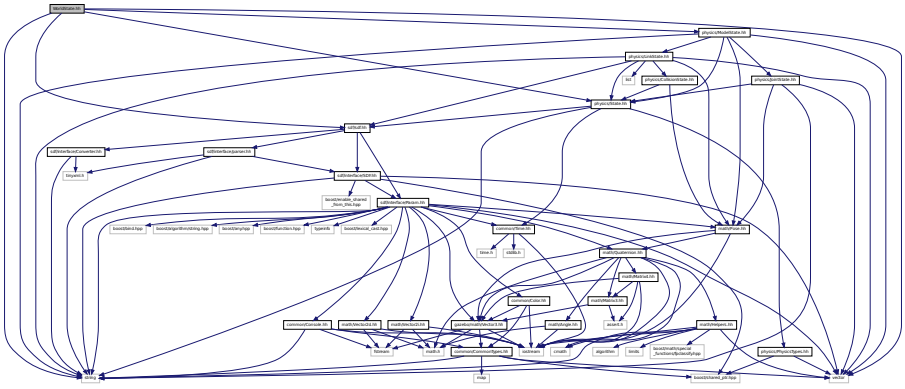
  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

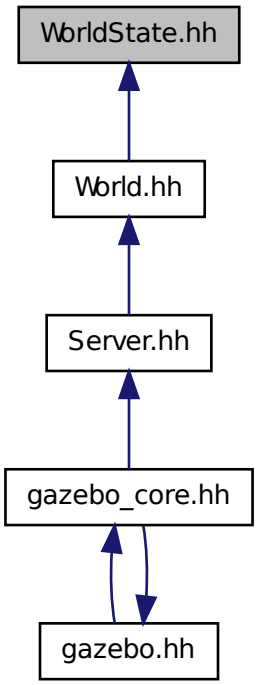
11.213 WorldState.hh File Reference

```
#include <string>
#include <vector>
#include "sdf/sdf.hh"
#include "physics/State.hh"
#include "physics/ModelState.hh"
```

Include dependency graph for WorldState.hh:



This graph shows which files directly or indirectly include this file:



Classes

- class **gazebo::physics::WorldState**

  *Store state information of a **physics::World*** (p. 954) *object.*

Namespaces

- namespace **gazebo**

  Forward declarations for the common classes.
- namespace **gazebo::physics**

  namespace for physics

Index