Gazebo

1.0.1

Generated by Doxygen 1.8.2

Thu Nov 8 2012 10:31:42

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

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 7

Module Documentation

7.1 Common

Class used to load Collada mesh files.

Files

- file **CommonTypes.hh**

  Forward declarations for the common classes.

Classes

- class **ColladaLoader**
- class **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**
- class **Logger**
- class **Material**
- class **Mesh**
- struct **NodeAssignment**
- class **SubMesh**

  A child mesh.
- class **MeshLoader**
- class **MeshManager**
- class **MouseEvent**
- class **PluginT**< **T** >
- class **WorldPlugin**

- class **ModelPlugin**
- class **SensorPlugin**
- class **SystemPlugin**
- class **SingletonT< T >**
- class **Skeleton**
- class **SkeletonNode**

  A node.
- class **NodeTransform**
- class **STLLoader**
- class **SystemPaths**
- class **Time**
- class **Timer**

Macros

- #define **gzmsg** (**gazebo::common::Console::Instance**()->ColorMsg("Msg", 32))
- #define **gzdbg** (**gazebo::common::Console::Instance**()->ColorMsg("Dbg", 36))

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

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

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

  Log a message.
- #define **DIAG_TIMER**(name) DiagnosticManager::Instance()->CreateTimer(name);
- #define **gzthrow**(msg)

Typedefs

- typedef boost::shared_ptr
  < DiagnosticTimer > **DiagnosticTimerPtr**

Functions

- **ColladaLoader** ()

  Constructor.
- virtual ~**ColladaLoader** ()

  Destructor.
- virtual **Mesh** * **Load** (const std::string &filename)

  Load a mesh.

7.1.1 Detailed Description

Class used to load Collada mesh files. A timer class.

A **Time** (p. 355) class.

Functions to handle getting system paths.

Class used to load STL mesh files.

A skeleton.

Singleton class.

A class which all plugins must inherit from.

Generic description of a mouse event.

Maintains and manages all meshes.

Base class for loading meshes.

A 3D mesh.

Encapsulates a description of a material.

Handles logging of data to disk.

Encapsulates an image.

Throw an error.

Output a message.

Defines a color.

7.1.2 Macro Definition Documentation

7.1.2.1 #define gzerr

Value:

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

Output an error message.

Referenced by Connection::AsyncRead(), ScrewJoint< ODEJoint >::Load(), and ParamT< std::string >::Set().

7.1.2.2 #define gzlog

Value:

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

Log a message.

7.1.2.3 #define gzthrow(*msg*)

Value:

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

7.1.2.4 #define gzwarn

Value:

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

Output a warning message.

7.2 Events

Base class for all events.

Namespaces

- namespace **gazebo::event**

  Event (p. *148*) *namespace.*

Classes

- class **Event**
- class **Connection**

  A class that encapsulates a connection.
- class **EventT< T >**

  An class for event processing.

Functions

- template<typename T >
  static ConnectionPtr **ConnectPause** (T _subscriber)
- static void **DisconnectPause** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectStep** (T _subscriber)

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

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

  Connect a boost::slot the the world created signal.
- static void **DisconnectWorldCreated** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectCreateEntity** (T _subscriber)

  Connect a boost::slot the the add entity signal.
- static void **DisconnectCreateEntity** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectSetSelectedEntity** (T _subscriber)

  Connect a boost::slot the set selected entity.
- static void **DisconnectSetSelectedEntity** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectDeleteEntity** (T _subscriber)

  Connect a boost::slot the delete entity.
- static void **DisconnectDeleteEntity** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectAddEntity** (T _subscriber)

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

- static void **DisconnectAddEntity** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowLights** (T _subscriber)

  Connect a boost::slot the the show light source signal.
- static void **DisconnectShowLights** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowCameras** (T _subscriber)

  Connect a boost::slot the the show camera source signal.
- static void **DisconnectShowCameras** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowContacts** (T _subscriber)

  Connect a boost::slot the the show contacts signal.
- static void **DisconnectShowContacts** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowWireframe** (T _subscriber)

  Connect a boost::slot the the show wireframe signal.
- static void **DisconnectShowWireframe** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowPhysics** (T _subscriber)

  Connect a boost::slot the the show physics signal.
- static void **DisconnectShowPhysics** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowJoints** (T _subscriber)

  Connect a boost::slot the the show joints signal.
- static void **DisconnectShowJoints** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowBoundingBoxes** (T _subscriber)

  Connect a boost::slot the the show bounding boxes signal.
- static void **DisconnectShowBoundingBoxes** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectWorldUpdateStart** (T _subscriber)

  Connect a boost::slot the the world update start signal.
- static void **DisconnectWorldUpdateStart** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectWorldUpdateEnd** (T _subscriber)

  Connect a boost::slot the the world update end signal.
- static void **DisconnectWorldUpdateEnd** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectEntitySelected** (T _subscriber)

  Connect a boost::slot the the entity selected signal.
- static void **DisconnectEntitySelected** (ConnectionPtr _subscriber)

  Disconnect a boost::slot the the entity selected signal.
- template<typename T >
  static ConnectionPtr **ConnectPreRender** (T _subscriber)

  Render start signal.
- static void **DisconnectPreRender** (ConnectionPtr _subscriber)

  Disconnect a render start signal.
- template<typename T >
  static ConnectionPtr **ConnectRender** (T _subscriber)

Connect a boost::slot the render update signal.

- static void **DisconnectRender** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectPostRender** (T _subscriber)

  Connect a boost::slot the post render update signal.
- static void **DisconnectPostRender** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectDiagTimerStart** (T _subscriber)

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

  Connect a boost::slot the diagnostic timer stop signal.
- static void **DisconnectDiagTimerStop** (ConnectionPtr _subscriber)
- 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)

Variables

- static EventT< void(bool)> **pause**

  Pause signal.
- static EventT< void()> **step**

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

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

  A world has been created.
- static EventT< void(std::string)> **entityCreated**

  An entity has been created.
- static EventT< void(std::string)> **setSelectedEntity**

  An entity has been selected.
- 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(bool)> **showLights**

  Light *(*p. *191) visuals should be shown.*
- static EventT< void(bool)> **showJoints**

  Joint *(*p. *181) visuals should be shown.*
- static EventT< void(bool)> **showCameras**

  Camera visuals should be shown.
- static EventT< void(bool)> **showContacts**

  Contact visuals should be shown.
- static EventT< void(bool)> **wireframe**

Wireframe enable signal.

- static EventT< void(bool)> **showPhysics**

  Show masses.
- static EventT< void(bool)> **showBoundingBoxes**

  Show bounding boxes.
- static EventT< void(std::string)> **entitySelected**

  Entity has been selected.
- static EventT< void()> **worldUpdateStart**

  World *(p. 385) update has started.*
- static EventT< void()> **worldUpdateEnd**

  World *(p. 385) update has ended.*
- static EventT< void()> **preRender**

  Pre-render.
- static EventT< void()> **render**

  Render.
- static EventT< void()> **postRender**

  Post-Render.
- static EventT< void(std::string)> **diagTimerStart**

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

  Diagnostic timer stop.

7.2.1 Detailed Description

Base class for all events. Connect a boost::slot the the pause signal.

7.2.2 Function Documentation

7.2.2.1 ConnectionPtr Connect (const boost::function< T > & *_subscriber*)

Connect a callback to this event.

Returns

A **Connection** (p. 117) object, which will automatically call Disconnect when it goes out of scope

Referenced by Events::ConnectAddEntity(), Events::ConnectCreateEntity(), Events::ConnectDeleteEntity(), Events::ConnectDiagTimerStart(), Events::ConnectDiagTimerStop(), Link::ConnectEnabled(), Events::ConnectEntitySelected(), Joint::ConnectJointUpdate(), DepthCamera::ConnectNewDepthFrame(), Camera::ConnectNewImageFrame(), GpuLaser::ConnectNewLaserFrame(), MultiRayShape::ConnectNewLaserScans(), DepthCamera::ConnectNewRGBPointCloud(), Events::ConnectPostRender(), Events::ConnectPreRender(), Events::ConnectRender(), Events::ConnectSetSelectedEntity(), Events::ConnectShowBoundingBoxes(), Events::ConnectShowCameras(), Events::ConnectShowContacts(), Events::ConnectShowJoints(), Events::ConnectShowLights(), Events::ConnectShowPhysics(), Events::ConnectShowWireframe(), Events::ConnectStep(), Events::ConnectStop(), Events::ConnectWorldCreated(), Events::ConnectWorldUpdateEnd(), and Events::ConnectWorldUpdateStart().

7.3 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 **Angle**
- class **Box**
- class **Matrix3**
- class **Matrix4**
- class **Plane**
- class **Pose**
- class **Quaternion**
- class **Rand**
- class **Vector2d**
- class **Vector2i**
- class **Vector3**
- class **Vector4**

7.3.1 Detailed Description

A set of classes that encapsulate math related properties and functions. double Generic x, y, z, w vector

Generic double x, y, z vector.

Generic integer x, y vector.

Generic double x, y vector.

Random number generator class.

A quaternion class.

Encapsulates a position and rotation in three space.

A plane and related functions.

A 3x3 matrix class.

A box and related functions.

An angle and related functions.

7.4 Messages

All messages and helper functions.

Namespaces

- namespace **gazebo::msgs**

  Messages namespace.

Classes

- interface **Axis**

  Axis (p. *84) message.*
- interface **BoxGeom**

  Information about a box geometry.
- interface **CameraSensor**

  Information about a camera sensor element.
- interface **Collision**

  Information about a collision element.
- interface **Color**

  Color (p. *115) message.*
- interface **ContactSensor**

  Information about a contact sensor element.
- interface **CylinderGeom**

  Information about a cylinder geometry.
- interface **Factory**

  Message to create new model in gazebo.
- interface **Frction**

  Information about friction.
- interface **Geometry**

  Information about a geometry element.
- interface **GUI**

  *Message for a **GUI*** (p. *168).*
- interface **Header**

  General information included by many messages.
- interface **Heightmap**

  Message for a heightmap geometry.
- interface **Image**

  Message for an image.
- interface **Inertial**

  Information about inertia.
- interface **Int**

  Integer message.
- interface **Joint**

  Message for joints.
- interface **Light**

Message for a light.

- interface **Link**

  Information about a link.

- interface **Material**

  Information about a material.

- interface **Mesh**

  Message for a mesh geometry.

- interface **Model**

  Information about a model.

- interface **Entities**

  Information about all entities in a world.

- interface **Packet**

  Message that encapsulates another message with a type description.

- interface **PID**

  *Message for **PID** (p. 278).*

- interface **Plane**

  Message for a plane geometry.

- interface **Pose**

  Message for a pose.

- interface **Projector**

  Information about a projector.

- interface **Publish**

  Message that contains information about a publisher of data.

- interface **Publisher**

  A list of publishers.

- interface **Quaternion**

  A message for a quaternion.

- interface **RaySensor**

  Information about a ray sensor element.

- interface **Request**

  A message containing a string request.

- interface **Response**

  Message that encapsulates a respons message with a type description.

- interface **Scene**

  A message containing a description of a scene.

- interface **Selection**

  *A message for **GUI** (p. 168) selection data.*

- interface **Sensor**

  Information about a sensor element.

- interface **ServerControl**

  A message that allows for control of the server functions.

- interface **Shadows**

  A message for shadow data.

- interface **SphereGeom**

  Information about a sphere geometry.

- interface **String**

  A message for string data.

- interface **Subscribe**

  A message for subscription data.
- interface **Surface**

  Information about a surface element.
- interface **Time**

  A message for time data.
- interface **Topic**

  A message for topic information.
- interface **Track**

  *in the **GUI** (p. 168))*
- interface **Vector2d**

  Message for a vector2 double.
- interface **Vector3d**

  Message for a vector3 double.
- interface **Visual**

  A message containing visual information.
- interface **WorldControl**

  A message that allows for control of world functions.
- interface **World**

  A message that allows for modifying (open, close) worlds.
- interface **WorldStatistics**

  A message statiscs about a world.

Functions

- msgs::Request * **CreateRequest** (const std::string &_request, const std::string &_data="")
- void **Init** (google::protobuf::Message &_message, const std::string &_id="")

  Initialize a message.
- void **Stamp** (msgs::Header *_header)

  ***Time** (p. 359) stamp a header.*
- void **Stamp** (msgs::Time *_time)

  Set the time in a time message.
- msgs::Vector3d **Convert** (const math::Vector3 &_v)

  *Convert a **math::Vector3** (p. 373) to a msgs::Vector3d.*
- msgs::Quaternion **Convert** (const math::Quaternion &_q)

  *Convert a **math::Quaternion** (p. 295) to a msgs::Quaternion.*
- msgs::Pose **Convert** (const math::Pose &_p)

  *Convert a **math::Pose** (p. 284) to a msgs::Pose.*
- msgs::Color **Convert** (const common::Color &_c)

  *Convert a **common::Color** (p. 114) to a msgs::Color.*
- msgs::Time **Convert** (const common::Time &_t)

  *Convert a **common::Time** (p. 355) to a msgs::Time.*
- msgs::PlaneGeom **Convert** (const math::Plane &_p)

  *Convert a **math::Plane** (p. 279) 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. 295).*

- math::Pose **Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose** (p. 284).*
- void **Set** (common::Image &_img, const msgs::Image &_msg)

  *Convert a msgs::Image to a **common::Image** (p. 174).*
- common::Color **Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color** (p. 114).*
- common::Time **Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time** (p. 355).*
- math::Plane **Convert** (const msgs::PlaneGeom &_p)

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

  *Set a msgs::Image from a **common::Image** (p. 174).*
- void **Set** (msgs::Vector3d *_pt, const math::Vector3 &_v)

  *Set a msgs::Vector3d from a **math::Vector3** (p. 373).*
- void **Set** (msgs::Vector2d *_pt, const math::Vector2d &_v)

  *Set a msgs::Vector2d from a **math::Vector3** (p. 373).*
- void **Set** (msgs::Quaternion *_q, const math::Quaternion &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 295).*
- void **Set** (msgs::Pose *_p, const math::Pose &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 284).*
- void **Set** (msgs::Color *_c, const common::Color &_v)

  *Set a msgs::Color from a **common::Color** (p. 114).*
- void **Set** (msgs::Time *_t, const common::Time &_v)

  *Set a msgs::Time from a **common::Time** (p. 355).*
- void **Set** (msgs::PlaneGeom *_p, const math::Plane &_v)

  *Set a msgs::Plane from a **math::Plane** (p. 279).*
- msgs::TrackVisual **TrackVisualFromSDF** (sdf::ElementPtr _sdf)

  Create a msgs::TrackVisual from a track visual SDF element.
- msgs::GUI **GUIFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::GUI from a **GUI** (p. 168) SDF element.*
- msgs::Light **LightFromSDF** (sdf::ElementPtr _sdf)

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

  Create a msgs::Visual from a visual SDF element.
- msgs::Fog **FogFromSDF** (sdf::ElementPtr _sdf)

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

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

  Get the header from a protobuf message.

7.4.1 Detailed Description

All messages and helper functions. Create a request message

Parameters

_request	**Request** (p. 313) string
_data	Optional data string

Returns

A **Request** (p. 313) message

7.4.2 Function Documentation

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

Convert a **math::Vector3** (p. 373) to a msgs::Vector3d.

Parameters

_v	The vector to convert

Returns

A msgs::Vector3d object

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

Convert a **math::Quaternion** (p. 295) to a msgs::Quaternion.

Parameters

_q	The quaternion to convert

Returns

A msgs::Quaternion object

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

Convert a **math::Pose** (p. 284) to a msgs::Pose.

Parameters

_p	The pose to convert

Returns

A msgs::Pose object

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

Convert a **common::Color** (p. 114) to a msgs::Color.

Parameters

_c	The color to convert

Returns

A msgs::Color object

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

Convert a **common::Time** (p. 355) to a msgs::Time.

Parameters

_t	The time to convert

Returns

A msgs::Time object

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

Convert a **math::Plane** (p. 279) to a msgs::PlaneGeom.

Parameters

_p	The plane to convert

Returns

A msgs::PlaneGeom object

7.4.2.7 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. 373) object

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

Convert a msgs::Quaternion to a **math::Quaternion** (p. 295).

Parameters

_q	The quaternion to convert

Returns

A **math::Quaternion** (p. 295) object

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

Convert a msgs::Pose to a **math::Pose** (p. 284).

Parameters

_q	The pose to convert

Returns

A **math::Pose** (p. 284) object

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

Convert a msgs::Color to a **common::Color** (p. 114).

Parameters

_c	The color to convert

Returns

A **common::Color** (p. 114) object

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

Convert a msgs::Time to a **common::Time** (p. 355).

Parameters

_t	The time to convert

Returns

A **common::Time** (p. 355) object

7.4.2.12 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

7.4.2.13 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

7.4.2.14 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

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

Create a msgs::GUI from a **GUI** (p. 168) SDF element.

Parameters

_sdf	The sdf element

Returns

The new msgs::GUI object

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

Initialize a message.

Parameters

_message	Message to initialize
_id	Optional string id

7.4.2.17 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

7.4.2.18 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

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

Convert a msgs::Image to a **common::Image** (p. 174).

Parameters

_img	The **common::Image** (p. 174) container
_msg	The **Image** (p. 176) message to convert

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

Set a msgs::Image from a **common::Image** (p. 174).

Parameters

_msg	A msgs::Image pointer
_i	A **common::Image** (p. 174) reference

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

Set a msgs::Vector3d from a **math::Vector3** (p. 373).

Parameters

_pt	A msgs::Vector3d pointer
_v	A **math::Vector3** (p. 373) reference

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

Set a msgs::Vector2d from a **math::Vector3** (p. 373).

Parameters

_pt	A msgs::Vector2d pointer
_v	A **math::Vector2d** (p. 369) reference

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

Set a msgs::Quaternion from a **math::Quaternion** (p. 295).

Parameters

_q	A msgs::Quaternion pointer
_v	A **math::Quaternion** (p. 295) reference

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

Set a msgs::Pose from a **math::Pose** (p. 284).

Parameters

_p	A msgs::Pose pointer
_v	A **math::Pose** (p. 284) reference

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

Set a msgs::Color from a **common::Color** (p. 114).

Parameters

_p	A msgs::Color pointer
_v	A **common::Color** (p. 114) reference

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

Set a msgs::Time from a **common::Time** (p. 355).

Parameters

_p	A msgs::Time pointer
_v	A **common::Time** (p. 355) reference

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

Set a msgs::Plane from a **math::Plane** (p. 279).

Parameters

_p	A msgs::Plane pointer
_v	A **math::Plane** (p. 279) reference

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

Time (p. 359) stamp a header.

Parameters

_header	**Header** (p. 169) to stamp

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

Set the time in a time message.

Parameters

_time	A **Time** (p. 359) message

7.4.2.30 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

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

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

Parameters

_sdf	The sdf element

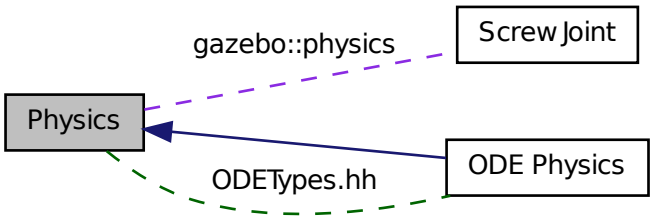
Returns

The new msgs::Visual object

7.5 Physics

An actor.

Collaboration diagram for Physics:



Modules

- **ODE Physics**

  An ODe ball joint.

Files

- file **ODETypes.hh**

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

  Physics forward declarations and type defines.

Namespaces

- namespace **gazebo::physics**

  Physics namespace.

Classes

- class **Actor**
- class **BallJoint**< **T** >
- class **Base**

  Base *(*p. *86) class for most physics classes.*
- class **BoxShape**
- class **Collision**
- class **Contact**
- class **CylinderShape**
- class **Entity**

- class **Gripper**

  A gripper abstraction.
- class **HeightmapShape**
- class **Hinge2Joint**< **T** >

  A two axis hinge joint.
- class **Inertial**
- class **Joint**
- class **JointFeedback**
- class **Link**
- class **MapShape**
- class **Model**
- class **MultiRayShape**
- class **PhysicsEngine**
- class **PhysicsFactory**

  The physics factory.
- class **PlaneShape**
- class **RayShape**
- class **Shape**
- class **SliderJoint**< **T** >
- class **SphereShape**
- class **SurfaceParams**
- class **TrimeshShape**
- class **UniversalJoint**< **T** >
- class **World**

  *The **World** (p. 386).*

Macros

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

  Static physics registration macro.

Typedefs

- typedef PhysicsEnginePtr(∗ **PhysicsFactoryFn**)(WorldPtr world)

Functions

- **Gripper** (ModelPtr _model)

  Constructor.
- virtual ∼**Gripper** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load.
- virtual void **Init** ()

  Initialize.
- bool **load** ()
- bool **fini** ()
- WorldPtr **create_world** (const std::string &name="")

- WorldPtr **get_world** (const std::string &name="")
- void **load_world** (WorldPtr world, sdf::ElementPtr _sdf)
- void **init_world** (WorldPtr world)
- void **run_world** (WorldPtr world)
- void **stop_world** (WorldPtr world)
- void **pause_world** (WorldPtr world, bool pause)
- void **load_worlds** (sdf::ElementPtr _sdf)
- void **init_worlds** ()
- void **run_worlds** ()
- void **stop_worlds** ()
- void **pause_worlds** (bool pause)
- void **remove_worlds** ()

7.5.1 Detailed Description

An actor. A universal joint.

Triangle mesh collision.

Surface (p. 352) params.

Sphere collision.

A slider joint.

Base (p. 86) class for all shapes.

Ray collision.

Collision (p. 110) for an infinite plane.

Base (p. 86) class for a physics engine.

Laser collision contains a set of ray-collisions, structured to simulate a laser range scanner.

A model.

Map collision.

Link (p. 192) class.

Feedback information from a joint.

Base (p. 86) class for all joints.

A class for inertial information about a link.

Height map collision.

Base (p. 86) class for all physics objects in Gazebo.

Cylinder collision.

A contact between two collisions.

Base (p. 86) class for all collision entities.

Box geometry.

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

A ball joint

Each contact can consist of a numnber of contact points

This collision is used primarily for ground planes. Note that while the plane in infinite, only the part near the camera is drawn.

7.5.2 Macro Definition Documentation

7.5.2.1 #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

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

7.5.3 Function Documentation

7.5.3.1 virtual void Load (sdf::ElementPtr *_sdf*) [virtual]

Load.

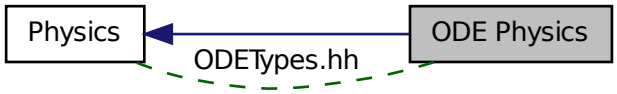
Parameters

_sdf	Shared point to an sdf element that contains the list of links in the gripper.

7.6 ODE Physics

An ODe ball joint.

Collaboration diagram for ODE Physics:



Files

- file **ODETypes.hh**

  ODE wrapper forward declarations and typedefs.

Classes

- class **ODEBallJoint**
- class **ODEBoxShape**
- class **ODECollision**
- class **ODECylinderShape**
- class **ODEHeightmapShape**
- class **ODEHinge2Joint**

  A two axis hinge joint.
- class **ODEHingeJoint**
- class **ODEJoint**
- class **ODELink**
- class **ODEMultiRayShape**
- class **ODEPhysics**
- class **ODEPlaneShape**
- class **ODERayShape**
- class **ODESliderJoint**
- class **ODESphereShape**
- class **ODESurfaceParams**
- class **ODETrimeshShape**
- class **ODEUniversalJoint**

Typedefs

- typedef boost::shared_ptr
  < ODEPhysics > **ODEPhysicsPtr**
- typedef boost::shared_ptr
  < ODESurfaceParams > **ODESurfaceParamsPtr**

7.6.1 Detailed Description

An ODe ball joint. A universal joint.

Triangle mesh collision.

Surface (p. 352) params.

A ODE sphere shape.

A slider joint.

Ray collision.

An ODE **Plane** (p. 280) shape.

ODE physics engine.

ODE specific version of **MultiRayShape** (p. 228).

ODE Physics wrapper.

ODE joint interface.

A single axis hinge joint.

ODE Height map collision.

ODE cylinder shape.

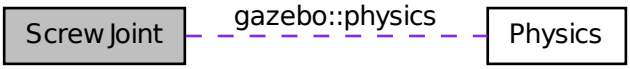
Base (p. 86) class for all ODE collisions.

ODE Box shape.

ODE **Link** (p. 192) class

7.7 Screw Joint

Collaboration diagram for Screw Joint:



Namespaces

- namespace **gazebo::physics**

  Physics namespace.

Classes

- class **ODEScrewJoint**

  A screw joint.
- class **ScrewJoint**< **T** >

  A screw joint.

7.7.1 Detailed Description

7.8 Rendering

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

Files

- file **RenderTypes.hh**

  Forward declarations and type defs for rendering.

Namespaces

- namespace **gazebo::rendering**

  Rendering namespace.

Classes

- class **Camera**
- class **Conversions**
- class **DepthCamera**
- class **DynamicLines**
- class **DynamicRenderable**
- class **FPSViewController**
- class **GpuLaser**
- class **Grid**

  Displays a grid of cells, drawn with lines.
- class **Heightmap**
- class **Light**
- class **Material**
- class **MovableText**
- class **OrbitViewController**
- class **RenderEngine**
- class **RTShaderSystem**
- class **Scene**
- class **SelectionObj**
- class **UserCamera**
- class **ViewController**
- class **Visual**
- class **WindowManager**

Functions

- bool **load** ()
- bool **init** ()
- bool **fini** ()
- rendering::ScenePtr **get_scene** (const std::string &_name)
- rendering::ScenePtr **create_scene** (const std::string &name, bool _enableVisualizations)
- void **remove_scene** (const std::string &name)
- **SelectionObj** (**Scene** ∗scene_)

Constructor.

- virtual ~**SelectionObj** ()

  Destructor.
- void **Init** ()
- void **Attach** (VisualPtr visual)

  Set the position of the node.
- void **Clear** ()
- 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.
- std::string **GetVisualName** () const

  Get the name of the visual the seleciton obj is attached to.
- void **SetHighlight** (const std::string &_mod)

  Highlight the selection object based on a modifier.

7.8.1 Detailed Description

A set of rendering related class, functions, and definitions. Class to mangage render windows.

A renderable object.

Base class for view controllers.

A camera used for user visualization of a scene.

A graphical selection object.

Representation of an entire scene graph.

Implements Ogre's Run-Time Shader system.

Adaptor to Ogre3d.

Orbit view controller.

Movable text.

Rendering material.

Wrapper around an ogre light source.

Height map geom.

First Person Shooter style view controller.

Abstract base class providing mechanisms for dynamically growing hardware buffers.

Class for drawing lines.

A set of utility function to convert between Gazebo and Ogre data types.

Basic camera sensor

This is the base class for all cameras.

Just a helper function for now

7.9 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 **CameraSensor**
- class **ContactSensor**
- class **DepthCameraSensor**
- class **GpuRaySensor**
- class **ImuSensor**
- class **RaySensor**
- class **Sensor**
- class **SensorFactory**
- class **SensorManager**

Macros

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

  Static sensor registration macro.

Functions

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

  Create a sensor using SDF.
- void **remove_sensor** (const std::string &_sensorName)

  Remove a sensor by name.
- void **run_once** (bool force=true)

  Run the sensor generation one step.
- void **run** ()

  Run sensor generation continuously. This is a blocking call.
- void **stop** ()

  Stop the sensor generation loop.
- bool **init** ()
- bool **fini** ()

- bool **remove_sensors** ()
- SensorPtr **get_sensor** (const std::string &_name)

  Get a sensor by name.

7.9.1 Detailed Description

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

Class to manage and update all sensors.

The sensor factory; the class is just for namespacing purposes.

Base class for sensors.

An IMU sensor.

Sensor (p. 329) with one or more rays.

Contact sensor.

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

This sensor detects and reports contacts between objects

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).

7.9.2 Macro Definition Documentation

7.9.2.1 #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** (p. 328) type name, as it appears in the world file.
classname	C++ class name for the sensor.

7.9.3 Function Documentation

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

Create a sensor using SDF.

Parameters

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

Returns

The name of the new sensor

7.9.3.2 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.

7.10 Transport

Handles transportation of messages.

Files

- file **TransportTypes.hh**

  Forward declarations for transport.

Namespaces

- namespace **gazebo::transport**

  Transport namespace.

Classes

- class **CallbackHelper**
- class **CallbackHelperT**< **M** >

  Callback helper Template.
- class **DebugCallbackHelper**
- class **Connection**
- class **ConnectionManager**
- class **IOManager**
- class **Node**
- class **Publication**
- class **PublicationTransport**
- class **Publisher**
- class **SubscribeOptions**
- class **Subscriber**
- class **SubscriptionTransport**

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

  Manages topics and their subscriptions.

Typedefs

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

Functions

- bool **get_master_uri** (std::string &master_host, unsigned int &master_port)
- bool **init** (const std::string &master_host="", unsigned int master_port=0)

  Initialize the transport system.
- void **run** ()

  Run the transport component.
- bool **is_stopped** ()

Return true if the transport system is stopped.

- void **stop** ()

  Stop the transport component from running.

- void **fini** ()

  Cleanup the transport component.

- void **clear_buffers** ()
- 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.

7.10.1 Detailed Description

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

A subscriber to a topic.

Options for a subscription.

A publisher of messages on a topic.

Reads data from a remote advertiser, and passes the data along to local subscribers.

A publication for a topic.

A node can advertise and subscribe topics, publish on advertised topics and listen to subscribed topics.

Managers boost::asio IO.

Manager of connections.

TCP/IP **Connection** (p. 117).

A helper class to handle callbacks when messages arrive

This facilitates transport of messages

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

7.10.2 Function Documentation

7.10.2.1 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.

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

Set to true to pause incoming messages.

They are still queued for later delivery

7.10.2.3 void gazebo::transport::run ( )

Run the transport component.

This starts message passing. This is a blocking call

Chapter 8

Namespace Documentation

8.1 gazebo::event Namespace Reference

Event (p. 148) namespace.

Classes

- class **Event**
- class **Connection**

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

  An class for event processing.
- class **Events**

Typedefs

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

8.1.1 Detailed Description

Event (p. 148) namespace.

8.2 gazebo::math Namespace Reference

Math namespace.

Classes

- class **Angle**

- class **Box**
- class **Matrix3**
- class **Matrix4**
- class **Plane**
- class **Pose**
- class **Quaternion**
- class **Rand**
- class **RotationSpline**
- class **Spline**
- class **Vector2d**
- class **Vector2i**
- class **Vector3**
- class **Vector4**

Typedefs

- typedef boost::mt19937 **GeneratorType**
- typedef boost::uniform_real < double > **UniformRealDist**
- typedef boost::normal_distribution < double > **NormalRealDist**
- typedef boost::uniform_int< int > **UniformIntDist**
- typedef boost::variate_generator < GeneratorType &, UniformRealDist > **URealGen**
- typedef boost::variate_generator < GeneratorType &, NormalRealDist > **NRealGen**
- typedef boost::variate_generator < GeneratorType &, UniformIntDist > **UIntGen**

Functions

- template<typename T > T **mean** (const std::vector< T > &_values)
- template<typename T > T **variance** (const std::vector< T > &_values)
- template<typename T > T **max** (const std::vector< T > &_values)
- template<typename T > T **min** (const std::vector< T > &_values)
- bool **equal** (const double &_a, const double &_b, const double &_epsilon=1e-6)
- bool **equal** (const float &_a, const float &_b, const float &_epsilon=1e-6)
- double **precision** (const double &_a, const unsigned int &_precision)
- float **precision** (const float &_a, const unsigned int &_precision)
- bool **isPowerOfTwo** (unsigned int _x)
- int **parseInt** (const std::string &input)
- double **parseFloat** (const std::string &input)

8.2.1 Detailed Description

Math namespace.

8.3 gazebo::msgs Namespace Reference

Messages namespace.

Functions

- msgs::Request * **CreateRequest** (const std::string &_request, const std::string &_data="")
- void **Init** (google::protobuf::Message &_message, const std::string &_id="")

  Initialize a message.
- void **Stamp** (msgs::Header *_header)

  Time *(p. 359) stamp a header.*
- void **Stamp** (msgs::Time *_time)

  Set the time in a time message.
- msgs::Vector3d **Convert** (const **math::Vector3** &_v)

  *Convert a **math::Vector3** (p. 373) to a msgs::Vector3d.*
- msgs::Quaternion **Convert** (const **math::Quaternion** &_q)

  *Convert a **math::Quaternion** (p. 295) to a msgs::Quaternion.*
- msgs::Pose **Convert** (const **math::Pose** &_p)

  *Convert a **math::Pose** (p. 284) to a msgs::Pose.*
- msgs::Color **Convert** (const **common::Color** &_c)

  *Convert a **common::Color** (p. 114) to a msgs::Color.*
- msgs::Time **Convert** (const **common::Time** &_t)

  *Convert a **common::Time** (p. 355) to a msgs::Time.*
- msgs::PlaneGeom **Convert** (const **math::Plane** &_p)

  *Convert a **math::Plane** (p. 279) 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. 295).*
- **math::Pose Convert** (const msgs::Pose &_p)

  *Convert a msgs::Pose to a **math::Pose** (p. 284).*
- void **Set** (**common::Image** &_img, const msgs::Image &_msg)

  *Convert a msgs::Image to a **common::Image** (p. 174).*
- **common::Color Convert** (const msgs::Color &_c)

  *Convert a msgs::Color to a **common::Color** (p. 114).*
- **common::Time Convert** (const msgs::Time &_t)

  *Convert a msgs::Time to a **common::Time** (p. 355).*
- **math::Plane Convert** (const msgs::PlaneGeom &_p)

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

  *Set a msgs::Image from a **common::Image** (p. 174).*
- void **Set** (msgs::Vector3d *_pt, const **math::Vector3** &_v)

*Set a msgs::Vector3d from a **math::Vector3** (p. 373).*

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

  *Set a msgs::Vector2d from a **math::Vector3** (p. 373).*
- void **Set** (msgs::Quaternion *_q, const **math::Quaternion** &_v)

  *Set a msgs::Quaternion from a **math::Quaternion** (p. 295).*
- void **Set** (msgs::Pose *_p, const **math::Pose** &_v)

  *Set a msgs::Pose from a **math::Pose** (p. 284).*
- void **Set** (msgs::Color *_c, const **common::Color** &_v)

  *Set a msgs::Color from a **common::Color** (p. 114).*
- void **Set** (msgs::Time *_t, const **common::Time** &_v)

  *Set a msgs::Time from a **common::Time** (p. 355).*
- void **Set** (msgs::PlaneGeom *_p, const **math::Plane** &_v)

  *Set a msgs::Plane from a **math::Plane** (p. 279).*
- msgs::TrackVisual **TrackVisualFromSDF** (sdf::ElementPtr _sdf)

  Create a msgs::TrackVisual from a track visual SDF element.
- msgs::GUI **GUIFromSDF** (sdf::ElementPtr _sdf)

  *Create a msgs::GUI from a **GUI** (p. 168) SDF element.*
- msgs::Light **LightFromSDF** (sdf::ElementPtr _sdf)

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

  Create a msgs::Visual from a visual SDF element.
- msgs::Fog **FogFromSDF** (sdf::ElementPtr _sdf)

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

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

  Get the header from a protobuf message.

8.3.1 Detailed Description

Messages namespace.

8.4 gazebo::physics Namespace Reference

Physics namespace.

Classes

- struct **TrajectoryInfo**
- class **Actor**
- class **BallJoint**
- class **Base**

  ***Base** (p. 86) class for most physics classes.*
- class **BoxShape**
- class **Collision**
- class **CollisionState**

- class **Contact**
- class **CylinderShape**
- class **Entity**
- class **Gripper**

  A gripper abstraction.
- class **HeightmapShape**
- class **Hinge2Joint**

  A two axis hinge joint.
- class **HingeJoint**

  A single axis hinge joint.
- class **Inertial**
- class **Joint**
- class **JointController**
- class **JointFeedback**
- class **JointState**
- class **Link**
- class **LinkState**
- class **MapShape**
- class **Model**
- class **ModelState**
- class **MultiRayShape**
- class **ODEBallJoint**
- class **ODEBoxShape**
- class **ODECollision**
- class **ODECylinderShape**
- class **ODEHeightmapShape**
- class **ODEHinge2Joint**

  A two axis hinge joint.
- class **ODEHingeJoint**
- class **ODEJoint**
- class **ODELink**
- class **ODEMultiRayShape**
- class **ContactFeedback**
- class **ODEPhysics**
- class **ODEPlaneShape**
- class **ODERayShape**
- class **ODEScrewJoint**

  A screw joint.
- class **ODESliderJoint**
- class **ODESphereShape**
- class **ODESurfaceParams**
- class **ODETrimeshShape**
- class **ODEUniversalJoint**
- class **PhysicsEngine**
- class **PhysicsFactory**

  The physics factory.
- class **PlaneShape**
- class **RayShape**
- class **ScrewJoint**

  A screw joint.

- class **Shape**
- class **SliderJoint**
- class **SphereShape**
- class **State**
- class **SurfaceParams**
- class **TrimeshShape**
- class **UniversalJoint**
- class **World**

  *The **World** (p. 386).*
- class **WorldState**

Typedefs

- typedef boost::shared_ptr < **ODEPhysics** > **ODEPhysicsPtr**
- typedef boost::shared_ptr < **ODESurfaceParams** > **ODESurfaceParamsPtr**
- typedef boost::shared_ptr < **ODECollision** > **ODECollisionPtr**
- typedef boost::shared_ptr < **ODELink** > **ODELinkPtr**
- typedef boost::shared_ptr < **ODERayShape** > **ODERayShapePtr**
- typedef PhysicsEnginePtr(∗ **PhysicsFactoryFn**)(WorldPtr world)
- typedef boost::shared_ptr< **Base** > **BasePtr**
- typedef boost::shared_ptr < **Contact** > **ContactPtr**
- typedef boost::shared_ptr< **Entity** > **EntityPtr**
- typedef boost::shared_ptr< **World** > **WorldPtr**
- typedef boost::shared_ptr< **Model** > **ModelPtr**
- typedef boost::shared_ptr< **Actor** > **ActorPtr**
- typedef boost::shared_ptr< **Link** > **LinkPtr**
- typedef boost::shared_ptr < **Collision** > **CollisionPtr**
- typedef boost::shared_ptr< **Joint** > **JointPtr**
- typedef boost::shared_ptr < **PhysicsEngine** > **PhysicsEnginePtr**
- typedef boost::shared_ptr< **Shape** > **ShapePtr**
- typedef boost::shared_ptr < **RayShape** > **RayShapePtr**
- typedef boost::shared_ptr < **MultiRayShape** > **MultiRayShapePtr**
- typedef boost::shared_ptr < **Inertial** > **InertialPtr**
- typedef boost::shared_ptr < **SurfaceParams** > **SurfaceParamsPtr**
- typedef boost::shared_ptr < **BoxShape** > **BoxShapePtr**
- typedef boost::shared_ptr < **CylinderShape** > **CylinderShapePtr**

- typedef boost::shared_ptr < **SphereShape** > **SphereShapePtr**
- typedef boost::shared_ptr < MeshShape > **MeshShapePtr**
- typedef std::vector< BasePtr > **Base_V**
- typedef std::vector< ModelPtr > **Model_V**
- typedef std::vector< ActorPtr > **Actor_V**
- typedef std::vector< JointPtr > **Joint_V**
- typedef std::vector< LinkPtr > **Link_V**
- typedef std::vector< CollisionPtr > **Collision_V**

Functions

- bool **load** ()
- bool **fini** ()
- WorldPtr **create_world** (const std::string &name="")
- WorldPtr **get_world** (const std::string &name="")
- void **load_world** (WorldPtr world, sdf::ElementPtr _sdf)
- void **init_world** (WorldPtr world)
- void **run_world** (WorldPtr world)
- void **stop_world** (WorldPtr world)
- void **pause_world** (WorldPtr world, bool pause)
- void **load_worlds** (sdf::ElementPtr _sdf)
- void **init_worlds** ()
- void **run_worlds** ()
- void **stop_worlds** ()
- void **pause_worlds** (bool pause)
- void **remove_worlds** ()

8.4.1 Detailed Description

Physics namespace.

8.5 gazebo::rendering Namespace Reference

Rendering namespace.

Classes

- class **ArrowVisual**
- class **AxisVisual**
- class **Camera**
- class **CameraVisual**
- class **COMVisual**
- class **ContactVisual**
- class **Conversions**
- class **DepthCamera**
- class **DynamicLines**

- class **DynamicRenderable**
- class **FPSViewController**
- class **GpuLaser**
- class **Grid**

  Displays a grid of cells, drawn with lines.
- class **GUIOverlay**
- class **Heightmap**
- class **JointVisual**
- class **LaserVisual**
- class **Light**
- class **Material**
- class **MovableText**
- class **OrbitViewController**
- class **Projector**

  A Bumper controller.
- class **RenderEngine**
- class **Events**
- class **RFIDTagVisual**
- class **RFIDVisual**
- class **RTShaderSystem**
- class **Scene**
- class **SelectionObj**
- class **UserCamera**
- class **ViewController**
- class **Visual**
- class **WindowManager**

Typedefs

- typedef boost::shared_ptr< **Scene** > **ScenePtr**
- typedef boost::shared_ptr< **Light** > **LightPtr**
- typedef boost::shared_ptr< **Camera** > **CameraPtr**
- typedef boost::shared_ptr < **UserCamera** > **UserCameraPtr**
- typedef boost::shared_ptr < **DepthCamera** > **DepthCameraPtr**
- typedef boost::shared_ptr < **GpuLaser** > **GpuLaserPtr**
- typedef boost::shared_ptr < **DynamicLines** > **DynamicLinesPtr**
- typedef boost::shared_ptr< **Visual** > **VisualPtr**
- typedef boost::shared_ptr < **LaserVisual** > **LaserVisualPtr**
- typedef boost::shared_ptr < **CameraVisual** > **CameraVisualPtr**
- typedef boost::shared_ptr < **JointVisual** > **JointVisualPtr**
- typedef boost::shared_ptr < **ContactVisual** > **ContactVisualPtr**
- typedef boost::shared_ptr < **ArrowVisual** > **ArrowVisualPtr**

- typedef boost::shared_ptr < **AxisVisual** > **AxisVisualPtr**
- typedef boost::shared_ptr < **COMVisual** > **COMVisualPtr**
- typedef boost::shared_ptr < **RFIDVisual** > **RFIDVisualPtr**
- typedef boost::shared_ptr < **RFIDTagVisual** > **RFIDTagVisualPtr**

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 }

Functions

- bool **load** ()
- bool **init** ()
- bool **fini** ()
- rendering::ScenePtr **get_scene** (const std::string &_name)
- rendering::ScenePtr **create_scene** (const std::string &name, bool _enableVisualizations)
- void **remove_scene** (const std::string &name)

8.5.1 Detailed Description

Rendering namespace.

8.5.2 Enumeration Type Documentation

8.5.2.1 enum RenderOpType

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.

8.6 gazebo::sensors Namespace Reference

Sensors namespace.

Classes

- class **CameraSensor**
- class **ContactSensor**
- class **DepthCameraSensor**
- class **GpuRaySensor**
- class **ImuSensor**
- class **RaySensor**
- class **RFIDSensor**
- class **RFIDTag**
- class **RFIDTagManager**
- class **Sensor**
- class **SensorFactory**
- class **SensorManager**

Typedefs

- typedef **Sensor** ∗(∗ **SensorFactoryFn**)()
- typedef boost::shared_ptr< **Sensor** > **SensorPtr**
- typedef boost::shared_ptr < **RaySensor** > **RaySensorPtr**
- typedef boost::shared_ptr < **CameraSensor** > **CameraSensorPtr**
- typedef boost::shared_ptr < **DepthCameraSensor** > **DepthCameraSensorPtr**
- typedef boost::shared_ptr < **ContactSensor** > **ContactSensorPtr**
- typedef boost::shared_ptr < **GpuRaySensor** > **GpuRaySensorPtr**
- typedef boost::shared_ptr < **RFIDSensor** > **RFIDSensorPtr**
- typedef boost::shared_ptr < **RFIDTag** > **RFIDTagPtr**
- typedef std::vector< SensorPtr > **Sensor_V**
- typedef std::vector< RaySensorPtr > **RaySensor_V**
- typedef std::vector < CameraSensorPtr > **CameraSensor_V**
- typedef std::vector < DepthCameraSensorPtr > **DepthCameraSensor_V**
- typedef std::vector < ContactSensorPtr > **ContactSensor_V**
- typedef std::vector < GpuRaySensorPtr > **GpuRaySensor_V**
- typedef std::vector< **RFIDSensor** > **RFIDSensor_V**
- typedef std::vector< **RFIDTag** > **RFIDTag_V**

Functions

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

  Create a sensor using SDF.
- void **remove_sensor** (const std::string &_sensorName)

  Remove a sensor by name.
- void **run_once** (bool force=true)

  Run the sensor generation one step.
- void **run** ()

  Run sensor generation continuously. This is a blocking call.
- void **stop** ()

  Stop the sensor generation loop.
- bool **init** ()
- bool **fini** ()
- bool **remove_sensors** ()
- SensorPtr **get_sensor** (const std::string &_name)

  Get a sensor by name.

8.6.1 Detailed Description

Sensors namespace.

8.7 gazebo::transport Namespace Reference

Transport namespace.

Classes

- class **CallbackHelper**
- class **CallbackHelperT**

  Callback helper Template.
- class **DebugCallbackHelper**
- class **Connection**
- class **ConnectionManager**
- class **IOManager**
- class **Node**
- class **Publication**
- class **PublicationTransport**
- class **Publisher**
- class **SubscribeOptions**
- class **Subscriber**
- class **SubscriptionTransport**

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

  Manages topics and their subscriptions.

Typedefs

- typedef boost::shared_ptr < **CallbackHelper** > **CallbackHelperPtr**
- typedef boost::shared_ptr < **Connection** > **ConnectionPtr**
- typedef boost::shared_ptr < **Publisher** > **PublisherPtr**
- typedef boost::shared_ptr < **Subscriber** > **SubscriberPtr**
- typedef boost::shared_ptr< **Node** > **NodePtr**
- typedef boost::shared_ptr < **Publication** > **PublicationPtr**
- typedef boost::shared_ptr < **PublicationTransport** > **PublicationTransportPtr**
- typedef boost::shared_ptr < **SubscriptionTransport** > **SubscriptionTransportPtr**

Functions

- bool **is_stopped** ()

  Return true if the transport system is stopped.
- bool **get_master_uri** (std::string &master_host, unsigned int &master_port)
- bool **init** (const std::string &master_host="", unsigned int master_port=0)

  Initialize the transport system.
- void **run** ()

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

  Stop the transport component from running.
- void **fini** ()

  Cleanup the transport component.
- void **clear_buffers** ()
- 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.

8.7.1 Detailed Description

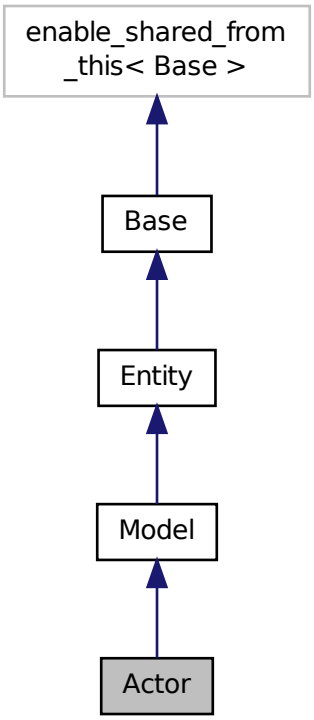
Transport namespace.

Chapter 9

Class Documentation

9.1 Actor Class Reference

Inheritance diagram for Actor:



Public Member Functions

- **Actor** (BasePtr **parent**)
  Constructor.
- virtual ∼**Actor** ()
  Destructor.
- void **Load** (sdf::ElementPtr _sdf)
  Load the actor.
- virtual void **Init** ()
  Initialize the actor.
- virtual void **Play** ()
  Start playing the script.
- virtual void **Stop** ()
  Stop playing the script.
- virtual bool **IsActive** ()
  Returns true when actor is playing animation.
- void **Update** ()
  Update the actor.
- virtual void **Fini** ()
  Finalize the actor.
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)
  update the parameters using new sdf values
- virtual const sdf::ElementPtr **GetSDF** ()
  Get the SDF values for the actor.

Protected Attributes

- const **common::Mesh** ∗ **mesh**
- **common::Skeleton** ∗ **skeleton**
- std::string **skinFile**
- double **skinScale**
- double **startDelay**
- double **scriptLength**
- double **lastScriptTime**
- bool **loop**
- bool **active**
- bool **autoStart**
- LinkPtr **mainLink**
- **common::Time prevFrameTime**
- **common::Time playStartTime**
- std::vector
  < **common::PoseAnimation** ∗ > **trajectories**
- std::vector< **TrajectoryInfo** > **trajInfo**
- std::map< std::string,
  common::SkeletonAnimation ∗ > **skelAnimation**
- std::map< std::string,
  std::map< std::string,
  std::string > > **skelNodesMap**
- std::map< std::string, bool > **interpolateX**
- std::string **visualName**
- transport::PublisherPtr **bonePosePub**
- std::string **oldAction**

Additional Inherited Members

9.1.1 Constructor & Destructor Documentation

9.1.1.1 Actor (BasePtr *parent*)

Constructor.

Parameters

parent	Parent object

9.1.2 Member Function Documentation

9.1.2.1 void Load (sdf::ElementPtr *_sdf*) [virtual]

Load the actor.

Parameters

_sdf	SDF parameters

Reimplemented from **Entity** (p. 146).

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

- Actor.hh

9.2 Angle Class Reference

Public Member Functions

- **Angle** ()
  Constructor.
- **Angle** (double _radian)
  Copy Constructor.
- **Angle** (const **Angle** &_angle)
  Copy constructor.
- virtual ~**Angle** ()
  Destructor.
- void **SetFromRadian** (double _radian)
  Set the value from an angle in radians.
- void **SetFromDegree** (double _degree)
  Set the value from an angle in degrees.
- double **GetAsRadian** () const
  Get the angle in radians.
- double **GetAsDegree** () const
  Get the angle in degrees.
- void **Normalize** ()
  Normalize the angle.

- double **operator*** () const

  Dereference operator.
- **Angle operator-** (const **Angle** &_angle) const

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

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

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

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

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

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

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

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

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

  Inequality.
- 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

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

  Greater equal.

Friends

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

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

  Istream operator.

9.2.1 Constructor & Destructor Documentation

9.2.1.1 Angle (double *_radian*)

Copy Constructor.

Parameters

_radian	Radians

9.2.1.2 Angle (const Angle & _angle)

Copy constructor.

Parameters

_angle	**Angle** (p. 75) to copy

9.2.2 Member Function Documentation

9.2.2.1 double GetAsDegree ( ) const

Get the angle in degrees.

Returns

Double containing the angle's degree value

9.2.2.2 double GetAsRadian ( ) const

Get the angle in radians.

Returns

Double containing the angle's radian value

9.2.2.3 bool operator!= (const Angle & _angle) const

Inequality.

Parameters

_angle	**Angle** (p. 75) to check for inequality

Returns

True if this != _angle

9.2.2.4 double operator∗ ( ) const [inline]

Dereference operator.

Returns

Double containing the angle's radian value

9.2.2.5 Angle operator∗ (const Angle & _angle) const

Multiplication, result = this ∗ _angle.

Parameters

_angle	**Angle** (p. 75) for multiplication

Returns

The new angle

9.2.2.6 Angle operator*= (const Angle & *_angle*)

Multiplication set, this = this * _angle.

Parameters

_angle	**Angle** (p. 75) for multiplication

Returns

This angle

9.2.2.7 Angle operator+ (const Angle & *_angle*) const

Addition, result = this + _angle.

Parameters

_angle	**Angle** (p. 75) for addition

Returns

The new angle

9.2.2.8 Angle operator+= (const Angle & *_angle*)

Addition set, this = this + _angle.

Parameters

_angle	**Angle** (p. 75) for addition

Returns

This angle

9.2.2.9 Angle operator- (const Angle & *_angle*) const

Substraction, result = this - _angle.

Parameters

_angle	**Angle** (p. 75) for substraction

Returns

The new angle

9.2.2.10 Angle operator-= (const Angle & *_angle*)

Subtraction set, this = this - _angle.

Parameters

_angle	**Angle** (p. 75) for subtraction

Returns

This angle

9.2.2.11 Angle operator/ (const Angle & *_angle*) const

Division, result = this / _angle.

Parameters

_angle	**Angle** (p. 75) for division

Returns

The new angle

9.2.2.12 Angle operator/= (const Angle & *_angle*)

Division set, this = this / _angle.

Parameters

_angle	**Angle** (p. 75) for division

Returns

This angle

9.2.2.13 bool operator< (const Angle & *_angle*) const

Less than operator.

Parameters

_angle	**Angle** (p. 75) to check

Returns

True if this < _angle

9.2.2.14 bool operator<= (const Angle & *_angle*) const

Less or equal operator.

Parameters

_angle	**Angle** (p. 75) to check

Returns

True if this <= _angle

9.2.2.15 bool operator== (const Angle & *_angle*) const

Equality operator, result = this == _angle.

Parameters

_angle	**Angle** (p. 75) to check for equality

Returns

True if this == _angle

9.2.2.16 bool operator> (const Angle & *_angle*) const

Greater than operator.

Parameters

_angle	**Angle** (p. 75) to check

Returns

True if this > _angle

9.2.2.17 bool operator>= (const Angle & *_angle*) const

Greater equal.

Parameters

_angle	**Angle** (p. 75) to check

Returns

True if this >= _angle

9.2.2.18 void SetFromDegree (double *_degree*)

Set the value from an angle in degrees.

Parameters

_degree	Degree value

9.2.2.19 void SetFromRadian (double *_radian*)

Set the value from an angle in radians.

Parameters

_radian	Radian value

9.2.3 Friends And Related Function Documentation

9.2.3.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::math::Angle & *_a*) [friend]

Ostream operator.

Outputs in degrees

Parameters

out	Ostream
pt	**Angle** (p. 75) to output

Returns

The Ostream

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

Istream operator.

Assumes input is in degrees

Parameters

in	Ostream
pt	**Angle** (p. 75) to read value into

Returns

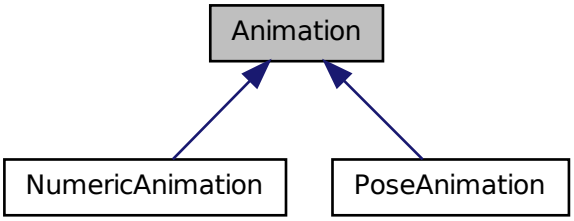
The istream

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

- Angle.hh

9.3 Animation Class Reference

Inheritance diagram for Animation:



Public Member Functions

- **Animation** (const std::string &_name, double _length, bool _loop)
- double **GetLength** () const
- void **SetLength** (double _len)
- void **SetTime** (double _time)
- void **AddTime** (double _time)
- double **GetTime** () const
- unsigned int **GetNumKeyFrames** () const
- **KeyFrame** ∗ **GetKeyFrame** (unsigned int _index) const

Protected Types

- typedef std::vector< **KeyFrame** ∗ > **KeyFrame_V**

Protected Member Functions

- double **GetKeyFramesAtTime** (double _time, **KeyFrame** ∗∗_kf1, **KeyFrame** ∗∗_kf2, unsigned int &_firstKeyIndex) const

Protected Attributes

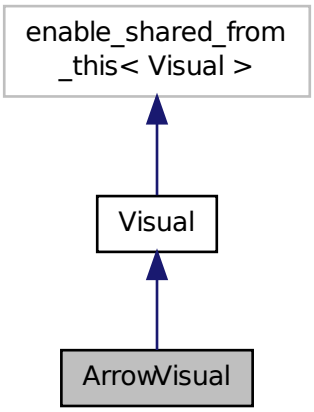
- std::string **name**
- double **length**
- double **timePos**
- bool **build**
- bool **loop**
- KeyFrame_V **keyFrames**

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

- Animation.hh

9.4 ArrowVisual Class Reference

Inheritance diagram for ArrowVisual:



Public Member Functions

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

  Load the visual with default parameters.
- void **ShowRotation** ()

Additional Inherited Members

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

- ArrowVisual.hh

9.5 Axis Interface Reference

Axis (p. 84) message.

9.5.1 Detailed Description

Axis (p. 84) message.

```
@verbatim
```

import "vector3d.proto";

message **Axis** (p. 84) { required **Vector3d** (p. 376) xyz = 1; required double limit_lower = 2; required double limit_upper = 3; required double limit_effort = 4; required double limit_velocity = 5; required double damping = 6; required double friction = 7;

}

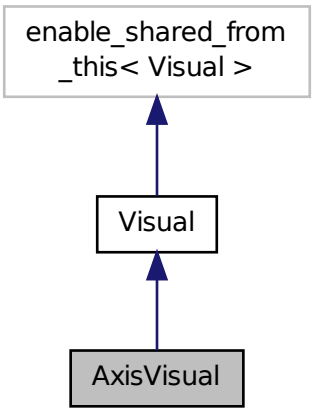
///

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

- axis.proto

9.6 AxisVisual Class Reference

Inheritance diagram for AxisVisual:



Public Member Functions

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

- virtual void **Load** ()

  Load the visual with default parameters.
- void **ShowRotation** (unsigned int _axis)
- void **ScaleXAxis** (const **math::Vector3** &_scale)
- void **ScaleYAxis** (const **math::Vector3** &_scale)
- void **ScaleZAxis** (const **math::Vector3** &_scale)
- void **SetAxisMaterial** (unsigned int _axis, const std::string &_material)

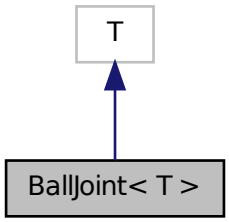
Additional Inherited Members

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

- AxisVisual.hh

9.7 BallJoint< T > Class Template Reference

Inheritance diagram for BallJoint< T >:



Public Member Functions

- **BallJoint** ()

  Constructor.
- virtual ~**BallJoint** ()

  Destructor.
- virtual void **SetAxis** (int, const **math::Vector3** &)

  Set the axis of rotation.
- virtual void **SetHighStop** (int, **math::Angle**)

  Set the high stop of an axis(index).
- virtual void **SetLowStop** (int, **math::Angle**)

  Set the low stop of an axis(index).
- virtual **math::Angle GetHighStop** (int)

  Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int)

  Get the low stop of an axis(index).

Protected Member Functions

- void **Load** (sdf::ElementPtr _sdf)

  Load the joint.

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

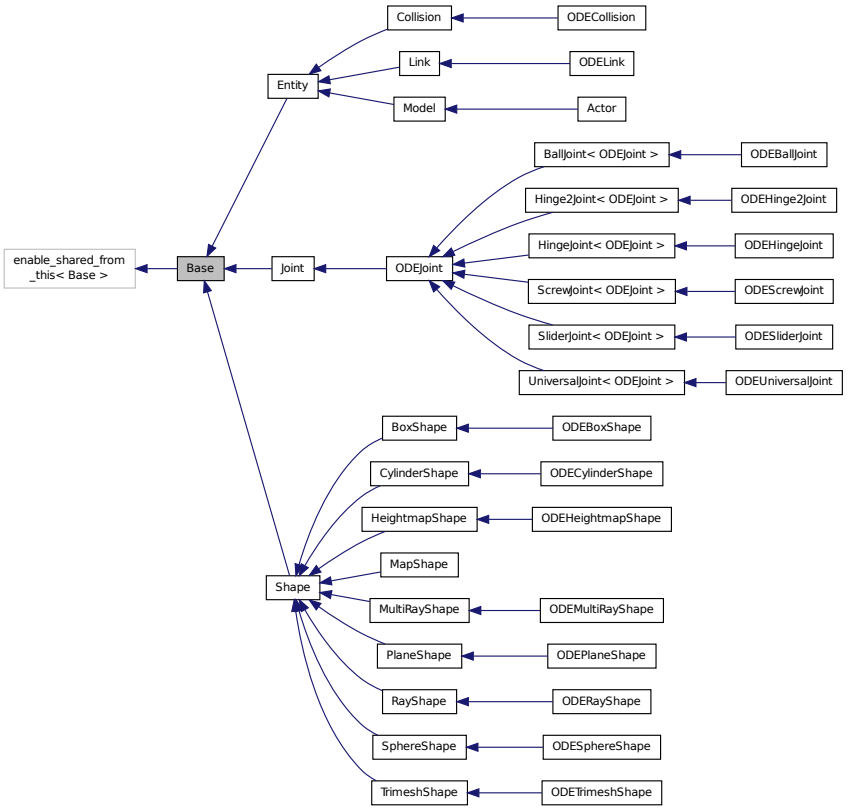
- BallJoint.hh

9.8 Base Class Reference

Base (p. 86) class for most physics classes.

`#include <Base.hh>`

Inheritance diagram for Base:



Public Types

- enum **EntityType** {
  BASE = 0x00000000, **ENTITY** = 0x00000001, **MODEL** = 0x00000002, **ACTOR** = 0x00000003,
  LINK = 0x00000004, **COLLISION** = 0x00000008, **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 }

Public Member Functions

- **Base** (BasePtr **parent**)

  Constructor.
- virtual ~**Base** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load.
- virtual void **Fini** ()

  Finialize the object.
- virtual void **Init** ()
- virtual void **Reset** ()
- virtual void **Update** ()
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  Update the parameters using new sdf values.
- virtual void **SetName** (const std::string &name)

  Set the name of the entity.
- std::string **GetName** () const

  Return the name of the entity.
- unsigned int **GetId** () const

  Return the ID of this entity.
- void **SetSaveable** (bool v)

  Set whether the object should be "saved", when the user selects to save the world to xml.
- bool **GetSaveable** () const

  Get whether the object should be "saved", when the user selects to save the world to xml.
- int **GetParentId** () const

  Return the ID of the parent.
- void **SetParent** (BasePtr **parent**)

  Set the parent.
- BasePtr **GetParent** () const

  Get the parent.
- void **AddChild** (BasePtr child)

  Add a child to this entity.
- virtual void **RemoveChild** (unsigned int id)

  Remove a child from this entity.
- void **RemoveChildren** ()

  Remove all children.
- unsigned int **GetChildCount** () const

  Get the number of children.

- BasePtr **GetById** (unsigned int _id) const

  Get 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.
- void **RemoveChild** (const std::string &_name)

  Remove a child by name.
- void **AddType** (EntityType _type)

  Add a type specifier.
- bool **HasType** (const EntityType &_t) const

  Get the type.
- unsigned int **GetType** () const

  Get a type by index.
- std::string **GetScopedName** () const

  Return the name of this entity with the model scope world::model1::...::modelN::entityName.
- std::string **GetCompleteScopedName** () const GAZEBO_DEPRECATED

  Return the name of this entity with the model+body+collision scope model1::...::modelN::bodyN::entityName.
- void **Print** (std::string prefix)
- bool **GetShowInGui** () const

  True == show parameters in the gui.
- void **SetShowInGui** (bool v)

  True == show parameters in the gui.
- virtual bool **SetSelected** (bool s)

  Set whether this entity has been selected by the user through.
- bool **IsSelected** () const

  True if the entity is selected by the user.
- bool **operator==** (const **Base** &ent) const

  Returns true if the entities are the same. Checks only the name.
- void **SetWorld** (const WorldPtr &_newWorld)

  Set the world this object belongs to.
- const WorldPtr & **GetWorld** () const

  Get the world this object is in.
- virtual const sdf::ElementPtr **GetSDF** ()

  Get the SDF values for the model.

Protected Attributes

- sdf::ElementPtr **sdf**
- BasePtr **parent**

  Parent of this entity.
- Base_V **children**

  Children of this entity.
- Base_V::iterator **childrenEnd**
- WorldPtr **world**

9.8.1 Detailed Description

Base (p. 86) class for most physics classes.

9.8.2 Constructor & Destructor Documentation

9.8.2.1 Base (BasePtr *parent*)

Constructor.

Parameters

parent	Parent of this object

9.8.3 Member Function Documentation

9.8.3.1 void AddChild (BasePtr *child*)

Add a child to this entity.

Parameters

child	Child entity

9.8.3.2 BasePtr GetByName (const std::string & *name*)

Get by name.

Parameters

name	Get a child (or self) object by name

Returns

A pointer to the object, NULL if not found

9.8.3.3 unsigned int GetChildCount ( ) const

Get the number of children.

Returns

The number of children

9.8.3.4 unsigned int GetId ( ) const

Return the ID of this entity.

This id is unique

Returns

Integer ID

9.8.3.5 std::string GetName ( ) const

Return the name of the entity.

Returns

Name of the entity

9.8.3.6 BasePtr GetParent ( ) const

Get the parent.

Returns

Pointer to the parent entity

9.8.3.7 int GetParentId ( ) const

Return the ID of the parent.

Returns

Integer ID

9.8.3.8 bool 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

9.8.3.9 virtual void Load (sdf::ElementPtr *_sdf*) [virtual]

Load.

Parameters

node	Pointer to an SDF parameters

Reimplemented in **Joint** (p. 182), **Actor** (p. 75), **Entity** (p. 146), **Model** (p. 221), **Link** (p. 196), **Collision** (p. 110), **MapShape** (p. 199), **BallJoint**< **ODEJoint** > (p. 86), **HeightmapShape** (p. 171), **ODECollision** (p. 238), **ScrewJoint**< **ODEJoint** > (p. 326), **ODEHinge2Joint** (p. 243), **ODEHingeJoint** (p. 245), **ODELink** (p. 251), **ODEScrewJoint** (p. 259), **ODETrimeshShape** (p. 264), **Hinge2Joint**< **ODEJoint** > (p. 172), **SliderJoint**< **ODEJoint** > (p. 341), **ODEJoint** (p. 246), **ODESliderJoint** (p. 261), **UniversalJoint**< **ODEJoint** > (p. 366), and **HingeJoint**< **ODEJoint** >

(p. 173).

9.8.3.10 virtual void RemoveChild (unsigned int *id*) [virtual]

Remove a child from this entity.

Parameters

child	Child to remove

9.8.3.11 virtual void SetName (const std::string & *name*) [virtual]

Set the name of the entity.

Parameters

name	**Link** (p. 192) name

Reimplemented in **Entity** (p. 146).

9.8.3.12 void SetParent (BasePtr *parent*)

Set the parent.

Parameters

parent	Parent object

9.8.3.13 void SetSaveable (bool *v*)

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

Parameters

v	Set to True if the object should be saved.

9.8.3.14 void SetWorld (const WorldPtr & *_newWorld*)

Set the world this object belongs to.

This will also set the world for all children

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

- Base.hh

9.9 Box Class Reference

Public Member Functions

- **Box** ()

  Default constructor.
- **Box** (const **Vector3** &_min, const **Vector3** &_max)

  Constructor.
- **Box** (const **Box** &_b)

  Copy Constructor.
- virtual ~**Box** ()

  Destructor.
- 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.
- **math::Vector3 GetSize** () const

  Get the size of the box.
- **math::Vector3 GetCenter** () const

  Get the box center.
- void **Merge** (const **Box** &_box)

  Merge a box with this box.
- **Box** & **operator=** (const **Box** &_b)

  Equal operator.
- **Box operator+** (const **Box** &_b) const

  Addition operator.
- const **Box** & **operator+=** (const **Box** &_b)

  Addition set operator.
- bool **operator==** (const **Box** &_b)

  Equality test operatoer.
- **Box operator-** (const **Vector3** &_v)

  Subtract a vector from the min and max values.

Public Attributes

- **Vector3 min**

  Minimum corner of the box.
- **Vector3 max**

  Maximum corner of the box.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Box** &_b)

  Output operator.

9.9.1 Constructor & Destructor Documentation

9.9.1.1 Box (const Vector3 & _min, const Vector3 & _max)

Constructor.

Parameters

_min	Minimum corner of the box
_max	Maximum corner of the box

9.9.1.2 Box (const Box & _b)

Copy Constructor.

Parameters

_b	**Box** (p. 91) to copy

9.9.2 Member Function Documentation

9.9.2.1 math::Vector3 GetCenter ( ) const

Get the box center.

Returns

The center position of the box

9.9.2.2 math::Vector3 GetSize ( ) const

Get the size of the box.

Returns

Size of the box

9.9.2.3 double GetXLength ( ) const

Get the length along the x dimension.

Returns

Double value of the length in the x dimension

9.9.2.4 double GetYLength ( ) const

Get the length along the y dimension.

Returns

Double value of the length in the y dimension

9.9.2.5 double GetZLength ( ) const

Get the length along the z dimension.

Returns

Double value of the length in the z dimension

9.9.2.6 void Merge (const Box & _box)

Merge a box with this box.

Parameters

_box	**Box** (p. 91) to add to this box

9.9.2.7 Box operator+ (const Box & _b) const

Addition operator.

result = this + _b

Parameters

_b	**Box** (p. 91) to add

Returns

The new box

9.9.2.8 const Box& operator+= (const Box & _b)

Addition set operator.

this = this + _b

Parameters

_b	**Box** (p. 91) to add

Returns

This new box

9.9.2.9 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

9.9.2.10 Box& operator= (const Box & _b)

Equal operator.

Set this box to the parameter

Parameters

_b	**Box** (p. 91) to copy

Returns

The new box.

9.9.2.11 bool operator== (const Box & _b)

Equality test operatoer.

Parameters

_b	**Box** (p. 91) to test

Returns

True if equal

9.9.3 Friends And Related Function Documentation

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

Output operator.

Parameters

_out	Output stream
_b	**Box** (p. 91) to output to the stream

Returns

The stream

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

- Box.hh

9.10 BoxGeom Interface Reference

Information about a box geometry.

9.10.1 Detailed Description

Information about a box geometry.

```
@verbatim
```

import "vector3d.proto";

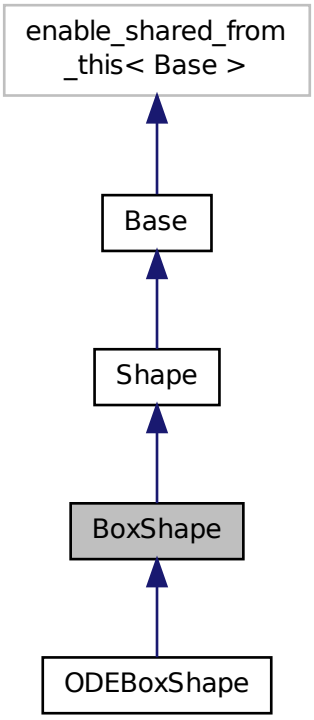
message **BoxGeom** (p. 96) { required **Vector3d** (p. 376) size = 1; } ///

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

- boxgeom.proto

9.11 BoxShape Class Reference

Inheritance diagram for BoxShape:



Public Member Functions

- **BoxShape** (CollisionPtr **parent**)

  Constructor.
- virtual ~**BoxShape** ()

  Destructor.
- virtual void **Init** ()

  Initialize the box.
- virtual void **SetSize** (const **math::Vector3** &size)

  Set the size of the box.
- **math::Vector3 GetSize** () const

  Get the size of the box.
- void **FillShapeMsg** (msgs::Geometry &_msg)
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
- virtual double **GetMass** (double _density) const

  Get the mass of a shape.

- virtual void **GetInertial** (double _mass, InertialPtr _inertial) const

  Get inertial for a shape.

Additional Inherited Members

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

- BoxShape.hh

9.12 BVHLoader Class Reference

Public Member Functions

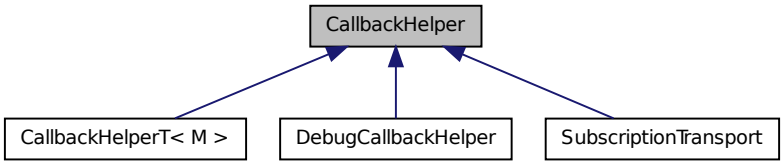
- **Skeleton** * **Load** (const std::string &filename, double scale)

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

- BVHLoader.hh

9.13 CallbackHelper Class Reference

Inheritance diagram for CallbackHelper:



Public Member Functions

- 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.
- bool **GetLatching** () const

Protected Attributes

- bool **latching**

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

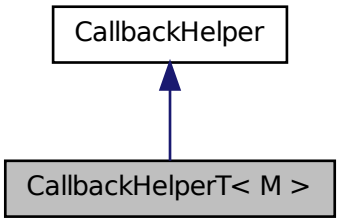
- CallbackHelper.hh

9.14 CallbackHelperT< M > Class Template Reference

Callback helper Template.

`#include <CallbackHelper.hh>`

Inheritance diagram for CallbackHelperT< M >:



Public Member Functions

- **CallbackHelperT** (const boost::function< void(const boost::shared_ptr< 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

9.14.1 Detailed Description

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

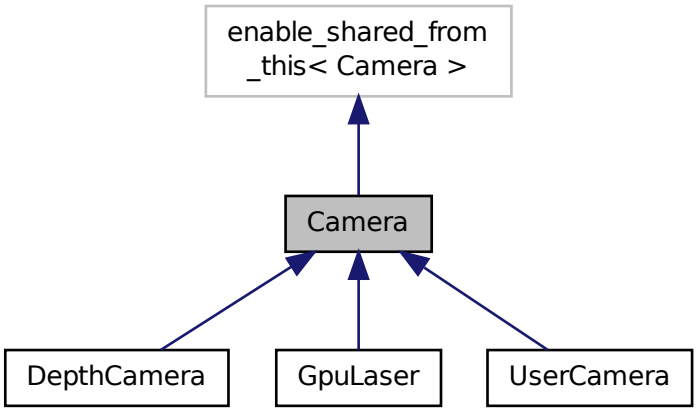
Callback helper Template.

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

- CallbackHelper.hh

9.15 Camera Class Reference

Inheritance diagram for Camera:



Public Member Functions

- **Camera** (const std::string &namePrefix, **Scene** *scene, bool _autoRender=true)

  Constructor.
- virtual ~**Camera** ()

  Destructor.
- void **Load** (sdf::ElementPtr _sdf)

  Load the camera with a set of parmeters.
- void **Load** ()

  Load the camera with default parmeters.
- void **Init** ()

  Initialize the camera.
- void **SetRenderRate** (double _hz)

  Set the render Hz rate.
- void **Render** ()

  Render the camera.
- virtual void **PostRender** ()

  Post render.
- virtual void **Update** ()

  Update the sensor information.
- void **Fini** ()

  Finalize the camera.
- bool **IsInitialized** () const
- void **SetWindowId** (unsigned int windowId)

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

- unsigned int **GetWindowId** () const

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

- void **SetScene** (**Scene** ∗scene)

  Set the scene this camera is viewing.

- **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.

- 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** &_quant)

  Set the world orientation.

- void **Translate** (const **math::Vector3** &direction)

  Translate the camera.

- void **RotateYaw** (float angle)

  Rotate the camera around the yaw axis.

- void **RotatePitch** (float angle)

  Rotate the camera around the pitch axis.

- void **SetClipDist** (float near, float far)

  Set the clip distances.

- void **SetClipDist** ()
- void **SetHFOV** (float radians)

  Set the camera FOV (horizontal)

- **math::Angle GetHFOV** () const

  Get the camera FOV (horizontal)

- **math::Angle GetVFOV** () const

  Get the camera FOV (vertical)

- void **SetImageSize** (unsigned int _w, unsigned int _h)

  Set the image size.

- void **SetImageWidth** (unsigned int _w)

  Set the image height.

- void **SetImageHeight** (unsigned int _h)

  Set the image height.

- unsigned int **GetImageWidth** () const

  Get the width of the image.

- unsigned int **GetTextureWidth** () const

  Get the width of the texture.

- unsigned int **GetImageHeight** () const

  Get the height of the image.

- unsigned int **GetImageDepth** () const

  Get the height of the image.

- std::string **GetImageFormat** () const

Get the height of the image.

- unsigned int **GetTextureHeight** () const

  Get the height of the texture.

- size_t **GetImageByteSize** () const

  Get the image size in bytes.

- double **GetZValue** (int x, int y)

  Get the Z-buffer value at the given image coordinate.

- double **GetNearClip** ()

  Get the near clip distance.

- double **GetFarClip** ()

  Get the far clip distance.

- void **EnableSaveFrame** (bool enable)

  Enable or disable saving.

- void **SetSaveFramePathname** (const std::string &pathname)

  Set the save frame pathname.

- bool **SaveFrame** (const std::string &_filename)

  Save the last frame to disk.

- Ogre::Camera * **GetOgreCamera** () const

  Get a pointer to the ogre camera.

- Ogre::Viewport * **GetViewport** () const
- unsigned int **GetViewportWidth** () const

  Get the viewport width in pixels.

- unsigned int **GetViewportHeight** () const

  Get the viewport height in pixels.

- **math::Vector3 GetUp** ()

  Get the viewport up vector.

- **math::Vector3 GetRight** ()

  Get the viewport right vector.

- virtual float **GetAvgFPS** ()

  Get the average FPS.

- virtual unsigned int **GetTriangleCount** ()

  Get the triangle count.

- void **SetAspectRatio** (float ratio)

  Set the aspect ratio.

- float **GetAspectRatio** () const

  Get the apect ratio.

- void **SetSceneNode** (Ogre::SceneNode *node)

  Set the camera's scene node.

- Ogre::SceneNode * **GetSceneNode** () const

  Get the camera's scene node.

- virtual const unsigned char * **GetImageData** (unsigned int i=0)

  Get a pointer to the image data.

- std::string **GetName** () const

  Get the camera's name.

- void **SetName** (const std::string &name)

  Set the camera's name.

- void **ToggleShowWireframe** ()

Toggle whether to view the world in wireframe.

- void **ShowWireframe** (bool s)

  Set whether to view the world in wireframe.

- 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.

- void **SetCaptureData** (bool _value)

  Set whether to capture data.

- void **CreateRenderTexture** (const std::string &textureName)

  Set the render target.

- **Scene** ∗ **GetScene** () const

  Get the scene this camera is in.

- bool **GetWorldPointOnPlane** (int _x, int _y, const **math::Plane** &_plane, **math::Vector3** &_result)

  Get point on a plane.

- virtual void **SetRenderTarget** (Ogre::RenderTarget ∗target)
- void **AttachToVisual** (const std::string &_visualName, bool _inheritOrientation, double _minDist=0.0, double _maxDist=0.0)

  Attach the camera to a scene node.

- void **TrackVisual** (const std::string &_visualName)

  Set the camera to track a scene node.

- Ogre::Texture ∗ **GetRenderTexture** () const

  Get the render texture.

- **math::Vector3 GetDirection** () const

  Get the camera's direction vector.

- template<typename T >
  event::ConnectionPtr **ConnectNewImageFrame** (T subscriber)

  Connect a to the add entity signal.

- void **DisconnectNewImageFrame** (event::ConnectionPtr &c)
- **common::Time GetLastRenderWallTime** ()

  Get the last time the camera was rendered.

- 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.

- bool **GetInitialized** () const

  Returns true if initialized.

- bool **MoveToPosition** (const **math::Vector3** &_end, double _pitch, double _yaw, double _time)
- bool **MoveToPositions** (const std::vector< **math::Pose** > &_pts, double _time, boost::function< void()> _onComplete=NULL)

Static Public Member Functions

- static size_t **GetImageByteSize** (unsigned int _width, unsigned int _height, const std::string &_format)
- static bool **SaveFrame** (const unsigned char ∗_image, unsigned int _width, unsigned int _height, int _depth, const std::string &_format, const std::string &_filename)

Protected Member Functions

- virtual void **RenderImpl** ()
- bool **TrackVisualImpl** (const std::string &_visualName)
- virtual bool **TrackVisualImpl** (VisualPtr _visual)

  Set the camera to track a scene node.
- virtual bool **AttachToVisualImpl** (const std::string &_name, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  Attach the camera to a visual.
- 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.

Protected Attributes

- sdf::ElementPtr **sdf**
- unsigned int **windowId**
- unsigned int **textureWidth**
- unsigned int **textureHeight**
- Ogre::Camera * **camera**
- Ogre::Viewport * **viewport**
- Ogre::SceneNode * **sceneNode**
- Ogre::SceneNode * **pitchNode**
- unsigned char * **saveFrameBuffer**
- unsigned char * **bayerFrameBuffer**
- unsigned int **saveCount**
- int **imageFormat**
- int **imageWidth**
- int **imageHeight**
- Ogre::RenderTarget * **renderTarget**
- Ogre::Texture * **renderTexture**
- bool **captureData**
- bool **newData**
- **common::Time lastRenderWallTime**
- **Scene** * **scene**
- **event::EventT**< void(const unsigned char *, unsigned int, unsigned int, unsigned int, const std::string &)> **newImageFrame**
- std::vector< event::ConnectionPtr > **connections**
- std::list< msgs::Request > **requests**
- bool **initialized**
- Ogre::AnimationState * **animState**
- **common::Time prevAnimTime**
- boost::function< void()> **onAnimationComplete**

9.15.1 Member Function Documentation

9.15.1.1 bool GetWorldPointOnPlane (int *_x,* int *_y,* const math::Plane & *_plane,* math::Vector3 & *_result*)

Get point on a plane.

Returns

True if a valid point was found

9.15.1.2 double GetZValue (int *x,* int *y*)

Get the Z-buffer value at the given image coordinate.

Parameters

x,y	**Image** (p. 176) coordinate; (0, 0) specifies the top-left corner.

Returns

Image (p. 176) z value; note that this is abitrarily scaled and is *not* the same as the depth value.

9.15.1.3 void Load (sdf::ElementPtr *_sdf*)

Load the camera with a set of parmeters.

Parameters

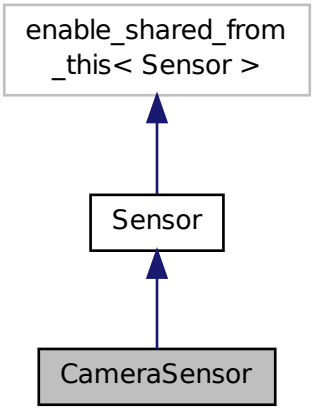
_sdf	The SDF camera info

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

- Camera.hh

9.16 CameraSensor Class Reference

Inheritance diagram for CameraSensor:



Public Member Functions

- **CameraSensor** ()

  Constructor.
- virtual ~**CameraSensor** ()

  Destructor.
- virtual void **SetParent** (const std::string &_name)

  Set the parent of the sensor.
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)

  Load the camera using parameter from an SDF element.
- virtual void **Load** (const std::string &_worldName)

  Load the camera using default parameters.
- virtual void **Init** ()

  Initialize the camera.
- virtual std::string **GetTopic** () const
- virtual void **SetActive** (bool value)

  Set whether the sensor is active or not.
- rendering::CameraPtr **GetCamera** () const
- unsigned int **GetImageWidth** () const
- unsigned int **GetImageHeight** () const
- const unsigned char * **GetImageData** ()
- bool **SaveFrame** (const std::string &_filename)

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.
- virtual void **Fini** ()

  Finalize the camera.

Additional Inherited Members

9.16.1 Member Function Documentation

9.16.1.1 virtual void Load (const std::string & _*worldName*, sdf::ElementPtr _*sdf*) [virtual]

Load the camera using parameter from an SDF element.

Parameters

_*sdf*	The SDF parameters

Reimplemented from **Sensor** (p. 331).

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

- CameraSensor.hh

9.17 CameraSensor Interface Reference

Information about a camera sensor element.

9.17.1 Detailed Description

Information about a camera sensor element.

```
@verbatim
```

import "vector2d.proto";

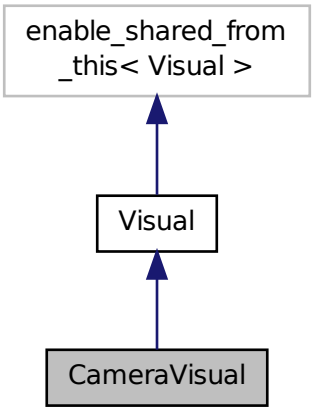
message **CameraSensor** (p. 107) { optional double horizontal_fov = 1; optional **Vector2d** (p. 368) image_size = 2; optional string image_format = 3; optional double near_clip = 4; optional double far_clip = 5; optional bool save_enabled = 6; optional string save_path = 7; } ///

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

- camerasensor.proto

9.18 CameraVisual Class Reference

Inheritance diagram for CameraVisual:



Public Member Functions

- **CameraVisual** (const std::string &_name, VisualPtr _vis)
- void **Load** (unsigned int _width, unsigned int _height)

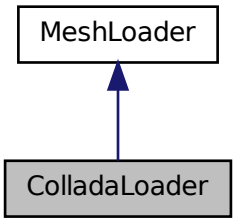
Additional Inherited Members

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

- CameraVisual.hh

9.19 ColladaLoader Class Reference

Inheritance diagram for ColladaLoader:



Public Member Functions

- **ColladaLoader** ()
  Constructor.
- virtual ∼**ColladaLoader** ()
  Destructor.
- virtual **Mesh** ∗ **Load** (const std::string &filename)
  Load a mesh.

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

- ColladaLoader.hh

9.20 Collision Interface Reference

Information about a collision element.

9.20.1 Detailed Description

Information about a collision element.

`@verbatim`

import "header.proto"; import "pose.proto"; import "geometry.proto"; import "surface.proto"; import "visual.proto";

message **Collision** (p. 109) { required uint32 id = 1; required string name = 2; optional double laser_retro = 3; optional double max_contacts = 4; optional **Pose** (p. 283) pose = 5; optional **Geometry** (p. 157) geometry = 6; optional **Surface** (p. 352) surface = 7;

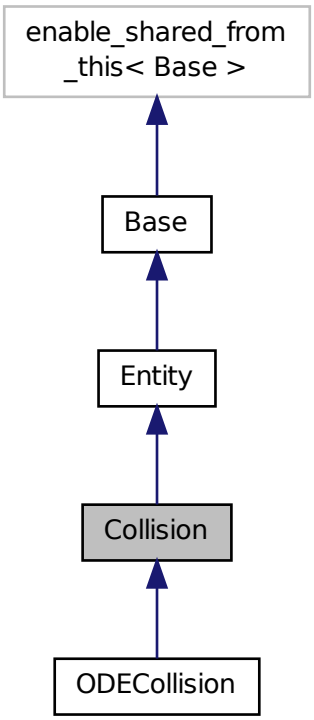
repeated **Visual** (p. 384) visual = 8; } ///

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

- collision.proto

9.21 Collision Class Reference

Inheritance diagram for Collision:



Public Member Functions

- **Collision** (LinkPtr _link)

  Constructor.
- virtual ~**Collision** ()

  Destructor.
- void **Fini** ()

  Finalize the collision.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the collision.
- virtual void **Init** ()

  Initialize the collision.
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

Update the parameters using new sdf values.

- void **SetCollision** (bool _placeable)

  Set the encapsulated collsion object.
- bool **IsPlaceable** () const

  Return whether this collision is placeable.
- 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 **SetLaserRetro** (float _retro)

  Set the laser retro reflectiveness.
- 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::Box GetBoundingBox** () const =0

  Get the bounding box for this collision.
- unsigned int **GetShapeType** ()

  Get the shape type.
- void **SetShape** (ShapePtr _shape)

  Set the shape for this collision.
- ShapePtr **GetShape** () const

  Get the attached shape.
- void **SetContactsEnabled** (bool _enable)

  Turn contact recording on or off.
- bool **GetContactsEnabled** () const

  Return true of contact recording is on.
- void **AddContact** (const **Contact** &_contact)

  Add an occurance of a contact to this collision.
- virtual **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the collision.
- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the collision in the world frame.
- virtual **math::Vector3 GetRelativeAngularVel** () const

  Get the angular velocity of the collision.
- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the collision in the world frame.
- virtual **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the collision.
- virtual **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the collision in the world frame.
- virtual **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the collision.
- virtual **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the collision in the world frame.

- **CollisionState GetState** ()

  Get the collision state.
- void **SetState** (const **CollisionState** &_state)

  Set the current collision state.
- template<typename T >
  event::ConnectionPtr **ConnectContact** (T _subscriber)
- void **DisconnectContact** (event::ConnectionPtr &_c)
- void **FillCollisionMsg** (msgs::Collision &_msg)

  Fill a collision message.
- void **ProcessMsg** (const msgs::Collision &_msg)

  Update parameters from a message.
- SurfaceParamsPtr **GetSurface** () const

  Get the surface parameters.
- const InertialPtr **GetInertial** () const

  Get the inertial properties.
- void **SetInertial** (InertialPtr _inertial)

  Set the inertial properties.

Public Attributes

- **event::EventT**< void(const
  std::string &, const **Contact** &)> **contact**

Protected Attributes

- LinkPtr **link**

  The link this collision belongs to.
- bool **placeable**
- ShapePtr **shape**

Additional Inherited Members

9.21.1 Member Function Documentation

9.21.1.1 const InertialPtr GetInertial ( ) const

Get the inertial properties.

Returns

A pointer to the inertial properties

9.21.1.2 virtual void SetCategoryBits (unsigned int *_bits*) [pure virtual]

Set the category bits, used during collision detection.

Parameters

bits	The bits

Implemented in **ODECollision** (p. 239).

9.21.1.3 virtual void SetCollideBits (unsigned int *_bits*) [pure virtual]

Set the collide bits, used during collision detection.

Parameters

bits	The bits

Implemented in **ODECollision** (p. 239).

9.21.1.4 void SetInertial (InertialPtr *_inertial*)

Set the inertial properties.

Parameters

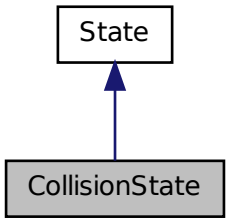
_inertial	The new inertial properties

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

- Collision.hh

9.22 CollisionState Class Reference

Inheritance diagram for CollisionState:



Public Member Functions

- **CollisionState** ()

  Default constructor.
- **CollisionState** (const CollisionPtr _collision)

  Constructor.

- virtual ~**CollisionState** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _elem)

  Load state from SDF element.
- **math::Pose GetPose** () const

  Get the collision pose.

Additional Inherited Members

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

- CollisionState.hh

9.23 Color Class Reference

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.
- 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.
- **math::Vector3 GetAsHSV** () const

  Get the color in HSV colorspace.
- void **SetFromHSV** (float _h, float _s, float _v)

  Set a color based on HSV values.
- **math::Vector3 GetAsYUV** () const

  Get the color in YUV colorspace.
- void **SetFromYUV** (float _y, float _u, float _v)

  Set from yuv.
- **Color** & **operator=** (const **Color** &_pt)

  Equal operator.
- float **operator[]** (unsigned int index)

  Array index operator.
- float **R** () const

  Get the red color.
- float **G** () const

  Get the green color.
- float **B** () const

Get the blue color.

- float **A** () const

  Get the alpha color.

- void **R** (float _r)

  Set the red color.

- void **G** (float _g)

  Set the green color.

- void **B** (float _b)

  Set the blue color.

- void **A** (float _a)

  Set the alpha color.

- **Color operator+** (const **Color** &pt) const
- **Color operator+** (const float &v) const
- const **Color** & **operator+=** (const **Color** &pt)
- **Color operator-** (const **Color** &pt) const
- **Color operator-** (const float &pt) const
- const **Color** & **operator-=** (const **Color** &pt)
- const **Color operator**/ (const float &i) const
- const **Color operator**/ (const **Color** &pt) const
- const **Color** & **operator/=** (const **Color** &pt)
- const **Color operator**∗ (const float &i) const
- const **Color operator**∗ (const **Color** &pt) const
- const **Color** & **operator∗=** (const **Color** &pt)
- bool **operator==** (const **Color** &pt) const
- bool **operator!=** (const **Color** &pt) const

Static Public Attributes

- static const **Color White**
- static const **Color Black**
- static const **Color Red**
- static const **Color Green**
- static const **Color Blue**
- static const **Color Yellow**

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **Color** &_pt)
- std::istream & **operator**>> (std::istream &in, **Color** &pt)

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

- Color.hh

9.24 Color Interface Reference

Color (p. 115) message.

9.24.1 Detailed Description

Color (p. 115) message.

```
@verbatim
```

message **Color** (p. 115) { required float r = 2; required float g = 3; required float b = 4; optional float a = 5 [default = 1.0]; }

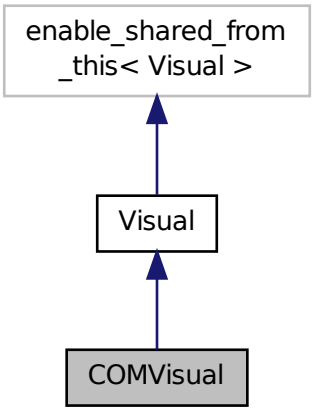
///

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

- color.proto

9.25 COMVisual Class Reference

Inheritance diagram for COMVisual:



Public Member Functions

- **COMVisual** (const std::string &_name, VisualPtr _vis)
- virtual void **Load** (ConstLinkPtr &_msg)

Additional Inherited Members

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

- COMVisual.hh

9.26 Connection Class Reference

A class that encapsulates a connection.

`#include <Event.hh>`

Public Member Functions

- **Connection** (**Event** *_e, int _i)
- int **GetId** () const
- int **GetUniqueId** () const

9.26.1 Detailed Description

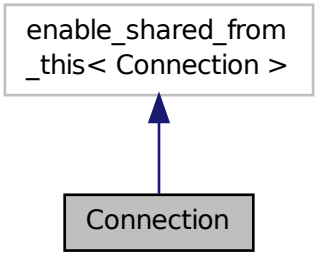
A class that encapsulates a connection.

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

- Event.hh

9.27 Connection Class Reference

Inheritance diagram for 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.
- bool **Connect** (const std::string &host, unsigned int port)

  Connect to a remote host.
- void **Listen** (unsigned int port, const AcceptCallback &accept_cb)

  Start a server that listens on a port.
- 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.
- void **Shutdown** ()

  Shutdown the socket.
- bool **IsOpen** () const

  Return true if the connection is open.
- void **Cancel** ()

  Cancel all async operations on an open socket.
- bool **Read** (std::string &data)

  Read data from the socket.
- void **EnqueueMsg** (const std::string &_buffer, bool _force=false)

  Write data to the socket.
- std::string **GetLocalURI** () const

  Get the local URI.
- std::string **GetRemoteURI** () const

  Get the remote URI.
- std::string **GetLocalAddress** () const

  Get the address of this connection.
- unsigned int **GetLocalPort** () const

  Get the port of this connection.
- std::string **GetRemoteAddress** () const

  Get the remote address.
- unsigned int **GetRemotePort** () const

  Get the remote port number.
- std::string **GetRemoteHostname** () const

  Get the remote hostname.
- std::string **GetLocalHostname** () const

  Get the local hostname.
- template<typename Handler >
  void **AsyncRead** (Handler handler)

  Peform and asyncronous read.
- event::ConnectionPtr **ConnectToShutdown** (boost::function< void()> subscriber_)
- void **DisconnectShutdown** (event::ConnectionPtr subscriber_)
- void **ProcessWriteQueue** ()

  Handle on write callbacks.

Public Attributes

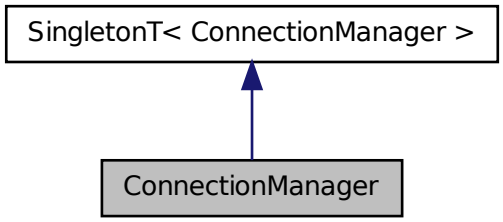
- unsigned int **id**
- unsigned int **writeCount**

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

- Connection.hh

9.28 ConnectionManager Class Reference

Inheritance diagram for ConnectionManager:



Public Member Functions

- bool **Init** (const std::string &master_host, unsigned int master_port)
- void **Run** ()

  Run the connection manager loop.
- bool **IsRunning** () const

  Return true if running (not stopped)
- void **Fini** ()

  Finalize the conneciton manager.
- void **Stop** ()

  Stop the conneciton manager.
- void **Subscribe** (const std::string &_topic, const std::string &_msgType, bool _latching)
- void **Unsubscribe** (const msgs::Subscribe &_sub)
- void **Unsubscribe** (const std::string &_topic, const std::string &_msgType)
- void **Advertise** (const std::string &topic, const std::string &msgType)
- void **Unadvertise** (const std::string &topic)
- void **GetAllPublishers** (std::list< msgs::Publish > &publishers)

  Explicitly update the publisher list.
- void **RemoveConnection** (ConnectionPtr &conn)

  Remove a connection.
- void **RegisterTopicNamespace** (const std::string &_name)

Register a new topic namespace.

- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

  Get all the topic namespaces.
- ConnectionPtr **ConnectToRemoteHost** (const std::string &host, unsigned int port)

  Connect to a remote server.
- void **RunUpdate** ()

Protected Attributes

- std::vector< event::ConnectionPtr > **eventConnections**

Additional Inherited Members

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

- ConnectionManager.hh

9.29 Console Class Reference

Message, error, warning, and logging functionality.

`#include <Console.hh>`

Public Member Functions

- void **Load** ()

  Load the message parameters.
- void **SetQuiet** (bool _q)

  Set quiet output.
- std::ostream & **ColorMsg** (const std::string &lbl, int color)

  Use this to output a message to the terminal.
- 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::ofstream & **Log** ()

  Use this to output a message to a log file.

Static Public Member Functions

- static **Console** * **Instance** ()

  Return an instance to this class.

9.29.1 Detailed Description

Message, error, warning, and logging functionality.

9.29.2 Member Function Documentation

9.29.2.1 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.

Parameters

level	Level of the message

9.29.2.2 std::ostream& ColorMsg (const std::string & *lbl,* int *color*)

Use this to output a message to the terminal.

Parameters

level	Level of the message

9.29.2.3 void SetQuiet (bool *_q*)

Set quiet output.

Parameters

q	True to prevent warning

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

- Console.hh

9.30 Contact Class Reference

Public Member Functions

- **Contact** ()

  Constructor.
- **Contact** (const **Contact** &c)

  Copy constructor.
- virtual ~**Contact** ()

  Destructor.
- **Contact Clone** () const

  Clone the contact.
- **Contact** & **operator=** (const **Contact** &contact)

  Operator =.
- void **Reset** ()

  Reset.

Public Attributes

- **Collision** * **collision1**
- **Collision** * **collision2**
- **JointFeedback forces** [20]
- **math::Vector3 positions** [20]
- **math::Vector3 normals** [20]
- double **depths** [20]
- int **count**
- **common::Time time**

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

- Contact.hh

9.31 ContactFeedback Class Reference

Public Attributes

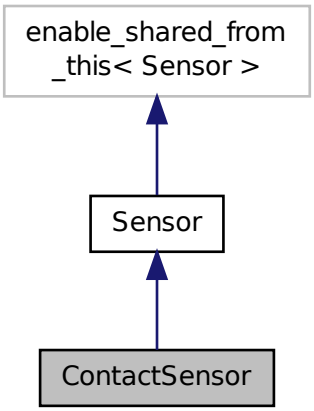
- **Contact contact**
- dJointFeedback **feedbacks** [MAX_CONTACT_JOINTS]
- int **feedbackCount**

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

- ODEPhysics.hh

9.32 ContactSensor Class Reference

Inheritance diagram for ContactSensor:



Public Member Functions

- **ContactSensor** ()

  Constructor.
- virtual ∼**ContactSensor** ()

  Destructor.
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)

  Load the contact sensor using parameter from an XMLConfig node.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.
- virtual void **Init** ()

  Initialize the sensor.
- 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.
- unsigned int **GetCollisionContactCount** (const std::string &_collisionName) const

  Return the number of contacts for an observed collision.
- **physics::Contact GetCollisionContact** (const std::string &_collisionName, unsigned int _index) const

  Get a contact for a collision by index.
- std::map< std::string, **physics::Contact** > **GetContacts** (const std::string &_collisionName)
- boost::mutex ∗ **GetUpdateMutex** () const

Protected Member Functions

- virtual void **UpdateImpl** (bool force)

  Update sensed values.
- virtual void **Fini** ()

  Finalize the sensor.

Additional Inherited Members

9.32.1 Constructor & Destructor Documentation

9.32.1.1 ContactSensor ( )

Constructor.

Parameters

body	The underlying collision test uses an ODE collision, so ray sensors must be attached to a body.

9.32.2 Member Function Documentation

9.32.2.1 virtual void Load (const std::string & *_worldName,* sdf::ElementPtr *_sdf*) [virtual]

Load the contact sensor using parameter from an XMLConfig node.

Parameters

node	The XMLConfig node

Reimplemented from **Sensor** (p. 331).

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

- ContactSensor.hh

9.33 ContactSensor Interface Reference

Information about a contact sensor element.

9.33.1 Detailed Description

Information about a contact sensor element.

```
@verbatim
```

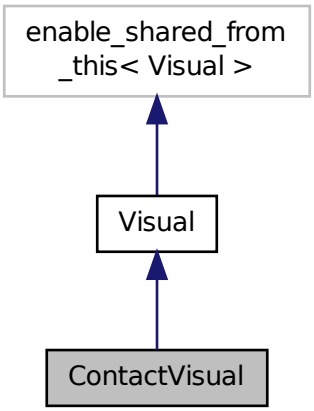
message **ContactSensor** (p. 124) { optional string collision_name = 1; } ///

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

- contactsensor.proto

9.34 ContactVisual Class Reference

Inheritance diagram for ContactVisual:



Public Member Functions

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

Additional Inherited Members

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

- ContactVisual.hh

9.35 Conversions Class Reference

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 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 quaternion.
- static **math::Quaternion Convert** (const Ogre::Quaternion &v)

  Ogre quaternion to Gazebo quaternion.

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

- Conversions.hh

9.36 CylinderGeom Interface Reference

Information about a cylinder geometry.

9.36.1 Detailed Description

Information about a cylinder geometry.

`@verbatim`

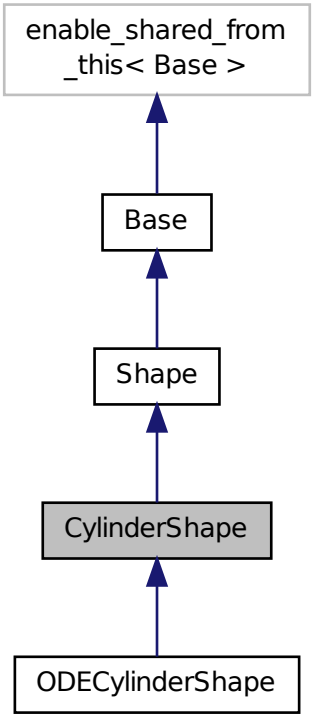
message **CylinderGeom** (p. 125) { required double radius = 1; required double length = 2; } ///

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

- cylindergeom.proto

9.37 CylinderShape Class Reference

Inheritance diagram for CylinderShape:



Public Member Functions

- **CylinderShape** (CollisionPtr **parent**)

  Constructor.
- virtual ~**CylinderShape** ()

  Destructor.
- void **Init** ()

  Initialize the cylinder.
- void **SetRadius** (const double &radius)

  Set radius.
- void **SetLength** (const double &length)

  Set length.
- double **GetRadius** () const

  Get radius.
- double **GetLength** () const

  Get length.

- virtual void **SetSize** (const double &radius, const double &length)

  Set the size of the cylinder.
- void **FillShapeMsg** (msgs::Geometry &_msg)
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
- virtual double **GetMass** (double _density) const

  Get the mass of a shape.
- virtual void **GetInertial** (double _mass, InertialPtr _inertial) const

  Get inertial for a shape.

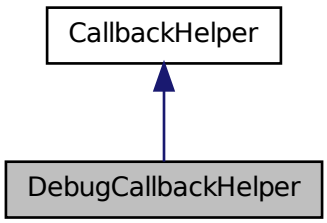
Additional Inherited Members

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

- CylinderShape.hh

9.38 DebugCallbackHelper Class Reference

Inheritance diagram for DebugCallbackHelper:



Public Member Functions

- **DebugCallbackHelper** (const boost::function< void(ConstStringPtr &)> &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.

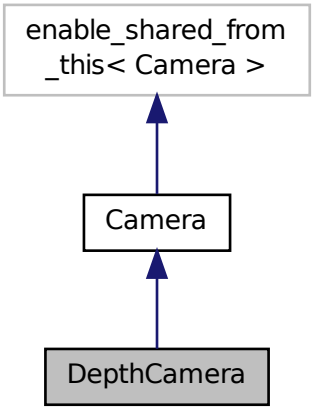
Additional Inherited Members

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

- CallbackHelper.hh

9.39 DepthCamera Class Reference

Inheritance diagram for DepthCamera:



Public Member Functions

- **DepthCamera** (const std::string &_namePrefix, **Scene** *_scene, bool _autoRender=true)

  Constructor.
- virtual ~**DepthCamera** ()

  Destructor.
- void **Load** (sdf::ElementPtr &_sdf)

  Load the camera with a set of parmeters.
- void **Load** ()

  Load the camera with default parmeters.
- void **Init** ()

  Initialize the camera.
- void **Fini** ()

  Finalize the camera.
- void **CreateDepthTexture** (const std::string &_textureName)
- virtual void **PostRender** ()

  Render the camera.
- virtual const float * **GetDepthData** ()
- template<typename T >
  event::ConnectionPtr **ConnectNewDepthFrame** (T subscriber)

  Connect a to the add entity signal.
- void **DisconnectNewDepthFrame** (event::ConnectionPtr &c)
- virtual void **SetDepthTarget** (Ogre::RenderTarget *target)
- template<typename T >
  event::ConnectionPtr **ConnectNewRGBPointCloud** (T subscriber)

Connect a to the add entity signal.

- void **DisconnectNewRGBPointCloud** (event::ConnectionPtr &c)

Protected Attributes

- Ogre::Texture ∗ **depthTexture**
- Ogre::RenderTarget ∗ **depthTarget**
- Ogre::Viewport ∗ **depthViewport**
- Ogre::Viewport ∗ **pcdViewport**
- Ogre::Texture ∗ **pcdTexture**
- Ogre::RenderTarget ∗ **pcdTarget**

Additional Inherited Members

9.39.1 Member Function Documentation

9.39.1.1 void Load (sdf::ElementPtr & *_sdf*)

Load the camera with a set of parmeters.

Parameters

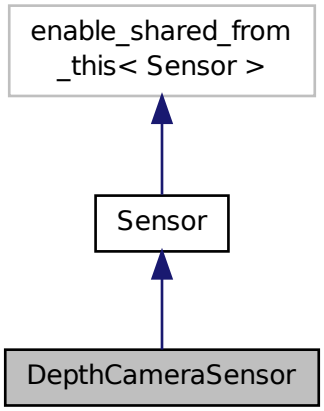
_sdf	The SDF camera info

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

- DepthCamera.hh

9.40 DepthCameraSensor Class Reference

Inheritance diagram for DepthCameraSensor:



Public Member Functions

- **DepthCameraSensor** ()

  Constructor.
- virtual ~**DepthCameraSensor** ()

  Destructor.
- virtual void **SetParent** (const std::string &_name)

  Set the parent of the sensor.
- virtual void **SetActive** (bool value)

  Set whether the sensor is active or not.
- rendering::DepthCameraPtr **GetDepthCamera** () const
- bool **SaveFrame** (const std::string &_filename)

Protected Member Functions

- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr &_sdf)

  Load the camera using parameter from an SDF element.
- virtual void **Load** (const std::string &_worldName)

  Load the camera using default parameters.
- virtual void **Init** ()

  Initialize the camera.
- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.
- virtual void **Fini** ()

  Finalize the camera.

Additional Inherited Members

9.40.1 Member Function Documentation

9.40.1.1 virtual void Load (const std::string & *_worldName,* sdf::ElementPtr & *_sdf*) [protected], [virtual]

Load the camera using parameter from an SDF element.

Parameters

_sdf	The SDF parameters

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

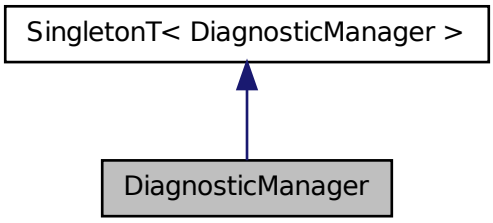
- DepthCameraSensor.hh

9.41 DiagnosticManager Class Reference

A diagnostic manager class.

`#include <Diagnostics.hh>`

Inheritance diagram for DiagnosticManager:



Public Member Functions

- DiagnosticTimerPtr **CreateTimer** (const std::string &_name)
- void **TimerStart** (**DiagnosticTimer** *_timer)

  A diagnostic timer has started.
- void **TimerStop** (**DiagnosticTimer** *_timer)

  A diagnostic timer has stoped.
- int **GetTimerCount** () const

  Get the number of timers.
- **Time GetTime** (int _index) const

  Get a specific time.
- **Time GetTime** (const std::string &_label) const

Get a time based on a label.

- std::string **GetLabel** (int _index) const

  Get a label for a timer.

- void **SetEnabled** (bool _e)
- bool **GetEnabled** () const

Additional Inherited Members

9.41.1 Detailed Description

A diagnostic manager class.

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

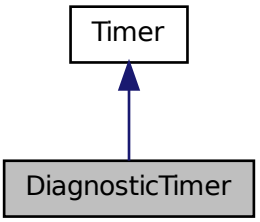
- Diagnostics.hh

9.42 DiagnosticTimer Class Reference

A timer designed for diagnostics.

`#include <Diagnostics.hh>`

Inheritance diagram for DiagnosticTimer:



Public Member Functions

- **DiagnosticTimer** (const std::string &_name)

  Constructor.

- virtual ~**DiagnosticTimer** ()

  Destructor.

- const std::string **GetName** () const

9.42.1 Detailed Description

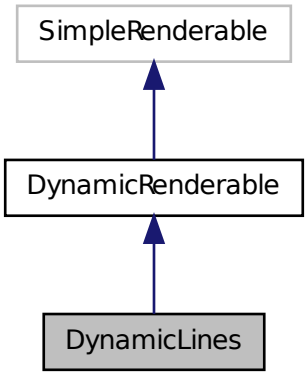
A timer designed for diagnostics.

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

- Diagnostics.hh

9.43 DynamicLines Class Reference

Inheritance diagram for DynamicLines:



Public Member Functions

- **DynamicLines** (**RenderOpType** opType=**RENDERING_LINE_STRIP**)

  Constructor.
- virtual ~**DynamicLines** ()

  Destructor.
- virtual const Ogre::String & **getMovableType** () const

  Returns "gazebo::ogredynamicslines".
- void **AddPoint** (const **math::Vector3** &pt)

  Add a point to the point list.
- void **SetPoint** (unsigned int index, const **math::Vector3** &value)

  Change the location of an existing point in the point list.
- 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 **Clear** ()

Remove all points from the point list.

- void **Update** ()

  Call this to update the hardware buffer after making changes.

Static Public Member Functions

- static std::string **GetMovableType** ()

Protected Member Functions

- virtual void **CreateVertexDeclaration** ()

  Implementation ***DynamicRenderable*** *(*p. *135), creates a simple vertex-only decl.*
- virtual void **FillHardwareBuffers** ()

  Implementation ***DynamicRenderable*** *(*p. *135), pushes point list out to hardware memory.*

Additional Inherited Members

9.43.1 Member Function Documentation

9.43.1.1 void AddPoint (const math::Vector3 & *pt*)

Add a point to the point list.

Parameters

pt	**math::Vector3** (p. 373) point

9.43.1.2 const math::Vector3& GetPoint (unsigned int *index*) const

Return the location of an existing point in the point list.

Parameters

index	Number of the point to return

Returns

math::Vector3 (p. 373) value of the point

9.43.1.3 unsigned int GetPointCount ( ) const

Return the total number of points in the point list.

Returns

Number of points

9.43.1.4 void SetPoint (unsigned int *index*, const math::Vector3 & *value*)

Change the location of an existing point in the point list.

Parameters

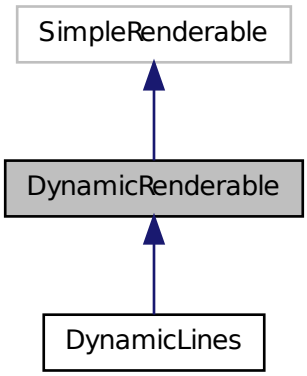
index	Index of the point to set
value	**math::Vector3** (p. 373) value to set the point to

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

- DynamicLines.hh

9.44 DynamicRenderable Class Reference

Inheritance diagram for DynamicRenderable:



Public Member Functions

- **DynamicRenderable** ()

  Constructor.
- virtual ~**DynamicRenderable** ()

  Virtual destructor.
- void **Init** (**RenderOpType** operationType, bool useIndices=false)

  Initializes the dynamic renderable.
- void **SetOperationType** (**RenderOpType** opType)

  Set the render operation type.
- **RenderOpType GetOperationType** () const

  Get the render operation type.
- virtual Ogre::Real **getBoundingRadius** (void) const

Implementation of Ogre::SimpleRenderable.

- virtual Ogre::Real **getSquaredViewDepth** (const Ogre::Camera *cam) const

  Implementation of Ogre::SimpleRenderable.

Protected Member Functions

- virtual void **CreateVertexDeclaration** ()=0

  Creates the vertex declaration.

- void **PrepareHardwareBuffers** (size_t vertexCount, size_t indexCount)

  Prepares the hardware buffers for the requested vertex and index counts.

- virtual void **FillHardwareBuffers** ()=0

  Fills the hardware vertex and index buffers with data.

Protected Attributes

- size_t **vertexBufferCapacity**

  Maximum capacity of the currently allocated vertex buffer.

- size_t **indexBufferCapacity**

  Maximum capacity of the currently allocated index buffer.

9.44.1 Member Function Documentation

9.44.1.1 virtual void 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 **DynamicLines** (p. 134).

9.44.1.2 virtual void 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 **DynamicLines** (p. 134).

9.44.1.3 void Init (RenderOpType *operationType,* 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. 136).

Parameters

operationType	The type of render operation to perform.
useIndices	Specifies whether to use indices to determine the vertices to use as input.

9.44.1.4 void 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

vertexCount	The number of vertices the buffer must hold.
indexCount	The number of indices the buffer must hold. This parameter is ignored if not using indices.

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

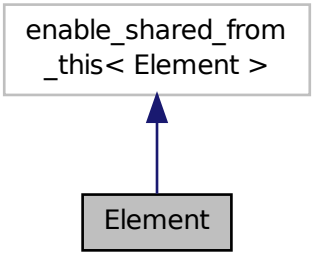
- DynamicRenderable.hh

9.45 Element Class Reference

SDF (p. 327) **Element** (p. 137) class.

`#include <SDF.hh>`

Inheritance diagram for Element:



Public Member Functions

- boost::shared_ptr< **Element** > **Clone** () const
- void **Copy** (const ElementPtr _elem)

  *Copy values from an **Element** (p. 137).*
- ElementPtr **GetParent** () const
- void **SetParent** (const ElementPtr _parent)
- void **SetName** (const std::string &_name)
- const std::string & **GetName** () const
- void **SetRequired** (const std::string &_req)
- const std::string & **GetRequired** () const
- void **SetCopyChildren** (bool _value)
- bool **GetCopyChildren** () const
- void **PrintDescription** (std::string _prefix)
- void **PrintValues** (std::string _prefix)
- std::string **ToString** (const std::string &_prefix) const
- void **AddAttribute** (const std::string &_key, const std::string &_type, const std::string &_defaultvalue, bool _required)
- void **AddValue** (const std::string &_type, const std::string &_defaultValue, bool _required)
- ParamPtr **GetAttribute** (const std::string &_key)

  Get the param of an attribute.
- unsigned int **GetAttributeCount** () const

  Get the number of attributes.
- ParamPtr **GetAttribute** (unsigned int _index) const

  Get an attribute using an index.
- unsigned int **GetElementDescriptionCount** () const

  Get the number of element descriptions.
- 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.
- bool **HasElementDescription** (const std::string &_name)

Return true if an element description exists.

- bool **HasAttribute** (const std::string &_key)
- bool **GetAttributeSet** (const std::string &_key)

  Return true if the attribute was set (i.e. not default value)
- ParamPtr **GetValue** ()

  Get the param of the elements value.
- bool **GetValueBool** (const std::string &_key="")
- int **GetValueInt** (const std::string &_key="")
- float **GetValueFloat** (const std::string &_key="")
- double **GetValueDouble** (const std::string &_key="")
- unsigned int **GetValueUInt** (const std::string &_key="")
- char **GetValueChar** (const std::string &_key="")
- std::string **GetValueString** (const std::string &_key="")
- **gazebo::math::Vector3 GetValueVector3** (const std::string &_key="")
- **gazebo::math::Vector2d GetValueVector2d** (const std::string &_key="")
- **gazebo::math::Quaternion GetValueQuaternion** (const std::string &_key="")
- **gazebo::math::Pose GetValuePose** (const std::string &_key="")
- **gazebo::common::Color GetValueColor** (const std::string &_key="")
- **gazebo::common::Time GetValueTime** (const std::string &_key="")
- 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)
- bool **HasElement** (const std::string &_name) const
- ElementPtr **GetElement** (const std::string &_name) const
- ElementPtr **GetFirstElement** () const
- ElementPtr **GetNextElement** (const std::string &_name="") const
- ElementPtr **GetOrCreateElement** (const std::string &_name)
- ElementPtr **AddElement** (const std::string &_name)
- void **InsertElement** (ElementPtr _elem)
- void **ClearElements** ()
- void **Update** ()
- void **Reset** ()
- void **SetInclude** (const std::string &_filename)
- std::string **GetInclude** () const
- void **AddElementDescription** (ElementPtr _elem)

  Add a new element description.

9.45.1 Detailed Description

SDF (p. 327) **Element** (p. 137) class.

9.45.2 Member Function Documentation

9.45.2.1 ParamPtr GetAttribute (const std::string & *_key*)

Get the param of an attribute.

Parameters

_key	the name of the attribute

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

- SDF.hh

9.46 Entities Interface Reference

Information about all entities in a world.

9.46.1 Detailed Description

Information about all entities in a world.

```
@verbatim
```

import "model.proto";

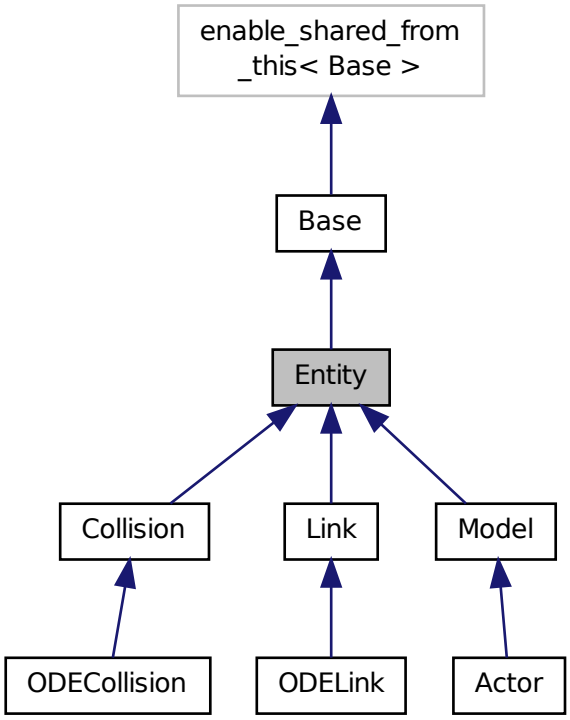
message Model_V { repeated **Model** (p. 222) models = 2; } ///

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

- model_v.proto

9.47 Entity Class Reference

Inheritance diagram for Entity:



Public Member Functions

- **Entity** (BasePtr **parent**)

  Constructor.
- virtual ~**Entity** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load.
- virtual void **Fini** ()

  Finalize the entity.
- virtual void **Reset** ()
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  Update the parameters using new sdf values.
- virtual void **SetName** (const std::string &name)

  Set the name of the entity.
- void **SetStatic** (const bool &s)

Set whether this entity is static: immovable.

- bool **IsStatic** () const

  Return whether this entity is static.

- void **SetInitialRelativePose** (const **math::Pose** &p)

  Set the initial pose.

- virtual **math::Box GetBoundingBox** () const

  Return the bounding box for the entity.

- const **math::Pose** & **GetWorldPose** () const

  Get the absolute pose of the entity.

- **math::Pose GetRelativePose** () const

  Get the pose of the entity relative to its parent.

- void **SetRelativePose** (const **math::Pose** &pose, bool notify=true)

  Set the pose of the entity relative to its parent.

- void **SetWorldPose** (const **math::Pose** &pose, bool notify=true)

  Set the world pose of the entity.

- virtual **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the entity.

- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the entity in the world frame.

- virtual **math::Vector3 GetRelativeAngularVel** () const

  Get the angular velocity of the entity.

- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the entity in the world frame.

- virtual **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the entity.

- virtual **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the entity in the world frame.

- virtual **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the entity.

- virtual **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the entity in the world frame.

- void **SetCanonicalLink** (bool _value)

  Set to true if this entity is a canonical link for a model.

- bool **IsCanonicalLink** () const

  A helper function that checks if this is a canonical body.

- void **SetAnimation** (const common::PoseAnimationPtr &_anim, boost::function< void()> _onComplete)

  Set an animation for this entity.

- void **SetAnimation** (common::PoseAnimationPtr _anim)

- virtual void **StopAnimation** ()

  Stop the current animation, if any.

- ModelPtr **GetParentModel** ()

  Get the parent model, if one exists.

- 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.

- void **GetNearestEntityBelow** (double &_distBelow, std::string &_entityName)

Get the distance to the nearest entity below (along the Z-axis) this entity.

- void **PlaceOnNearestEntityBelow** ()

  Move this entity to be ontop of the nearest entity below.
- void **PlaceOnEntity** (const std::string &_entityName)
- **math::Box GetCollisionBoundingBox** () const
- void **SetWorldTwist** (const **math::Vector3** &linear, const **math::Vector3** &angular, bool updateChildren=true)
- const **math::Pose** & **GetDirtyPose** () const

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

- EntityPtr **parentEntity**

  A helper that prevents numerous dynamic_casts.
- transport::NodePtr **node**
- transport::PublisherPtr **visPub**
- transport::PublisherPtr **requestPub**
- msgs::Visual ∗ **visualMsg**
- msgs::Pose ∗ **poseMsg**
- common::PoseAnimationPtr **animation**
- **common::Time prevAnimationTime**
- **math::Pose animationStartPose**
- std::vector< event::ConnectionPtr > **connections**
- event::ConnectionPtr **animationConnection**
- **math::Pose dirtyPose**

Additional Inherited Members

9.47.1 Constructor & Destructor Documentation

9.47.1.1 Entity (BasePtr *parent*)

Constructor.

Parameters

parent	Parent of the entity.

9.47.2 Member Function Documentation

9.47.2.1 void GetNearestEntityBelow (double & *_distBelow,* std::string & *_entityName*)

Get the distance to the nearest entity below (along the Z-axis) this entity.

Parameters

_distBelow	The distance to the nearest entity below
_entityName	The name of the nearest entity below

9.47.2.2 ModelPtr GetParentModel ( )

Get the parent model, if one exists.

Returns

Pointer to a model, or NULL if no parent model exists

9.47.2.3 virtual math::Vector3 GetRelativeAngularAccel ( ) const [inline], [virtual]

Get the angular acceleration of the entity.

Returns

A **math::Vector3** (p. 373) for the acceleration

Reimplemented in **Link** (p. 194), **Collision** (p. 111), and **Model** (p. 219).

9.47.2.4 virtual math::Vector3 GetRelativeAngularVel ( ) const [inline], [virtual]

Get the angular velocity of the entity.

Returns

A **math::Vector3** (p. 373) for the velocity

Reimplemented in **Link** (p. 193), **Collision** (p. 111), and **Model** (p. 219).

9.47.2.5 virtual math::Vector3 GetRelativeLinearAccel ( ) const [inline], [virtual]

Get the linear acceleration of the entity.

Returns

A **math::Vector3** (p. 373) for the acceleration

Reimplemented in **Link** (p. 194), **Collision** (p. 111), and **Model** (p. 219).

9.47.2.6 virtual math::Vector3 GetRelativeLinearVel ( ) const [inline], [virtual]

Get the linear velocity of the entity.

Returns

A **math::Vector3** (p. 373) for the linear velocity

Reimplemented in **Link** (p. 193), **Collision** (p. 111), and **Model** (p. 220).

9.47.2.7 virtual math::Vector3 GetWorldAngularAccel ( ) const [inline], [virtual]

Get the angular acceleration of the entity in the world frame.

Returns

A **math::Vector3** (p. 373) for the acceleration

Reimplemented in **Link** (p. 194), **Collision** (p. 111), and **Model** (p. 220).

9.47.2.8 virtual math::Vector3 GetWorldAngularVel ( ) const [inline], [virtual]

Get the angular velocity of the entity in the world frame.

Returns

A **math::Vector3** (p. 373) for the velocity

Reimplemented in **Collision** (p. 111), **Model** (p. 220), and **ODELink** (p. 250).

9.47.2.9 virtual math::Vector3 GetWorldLinearAccel ( ) const [inline], [virtual]

Get the linear acceleration of the entity in the world frame.

Returns

A **math::Vector3** (p. 373) for the acceleration

Reimplemented in **Link** (p. 194), **Collision** (p. 111), and **Model** (p. 220).

9.47.2.10 virtual math::Vector3 GetWorldLinearVel ( ) const [inline], [virtual]

Get the linear velocity of the entity in the world frame.

Returns

A **math::Vector3** (p. 373) for the linear velocity

Reimplemented in **Collision** (p. 111), **ODELink** (p. 249), and **Model** (p. 220).

9.47.2.11 bool IsStatic ( ) const

Return whether this entity is static.

Returns

bool True = static

9.47.2.12 virtual void Load (sdf::ElementPtr *_sdf*) [virtual]

Load.

Parameters

node	Pointer to an configuration node

Reimplemented from **Base** (p. 90).

Reimplemented in **Actor** (p. 75), **Model** (p. 221), **Link** (p. 196), **Collision** (p. 110), **ODECollision** (p. 238), and **ODELink** (p. 251).

9.47.2.13 void SetCanonicalLink (bool *_value*)

Set to true if this entity is a canonical link for a model.

Parameters

_value	True if the link is canonical.

9.47.2.14 void SetInitialRelativePose (const math::Pose & *p*)

Set the initial pose.

Parameters

p	The initial pose

9.47.2.15 virtual void SetName (const std::string & *name*) [virtual]

Set the name of the entity.

Parameters

name	The new name

Reimplemented from **Base** (p. 91).

9.47.2.16 void SetRelativePose (const math::Pose & *pose,* bool *notify* = true)

Set the pose of the entity relative to its parent.

Parameters

pose	The new pose
notify	True = tell children of the pose change

9.47.2.17 void SetStatic (const bool & *s*)

Set whether this entity is static: immovable.

Parameters

s	Bool, true = static

9.47.2.18 void SetWorldPose (const math::Pose & *pose*, bool *notify* = `true`)

Set the world pose of the entity.

Parameters

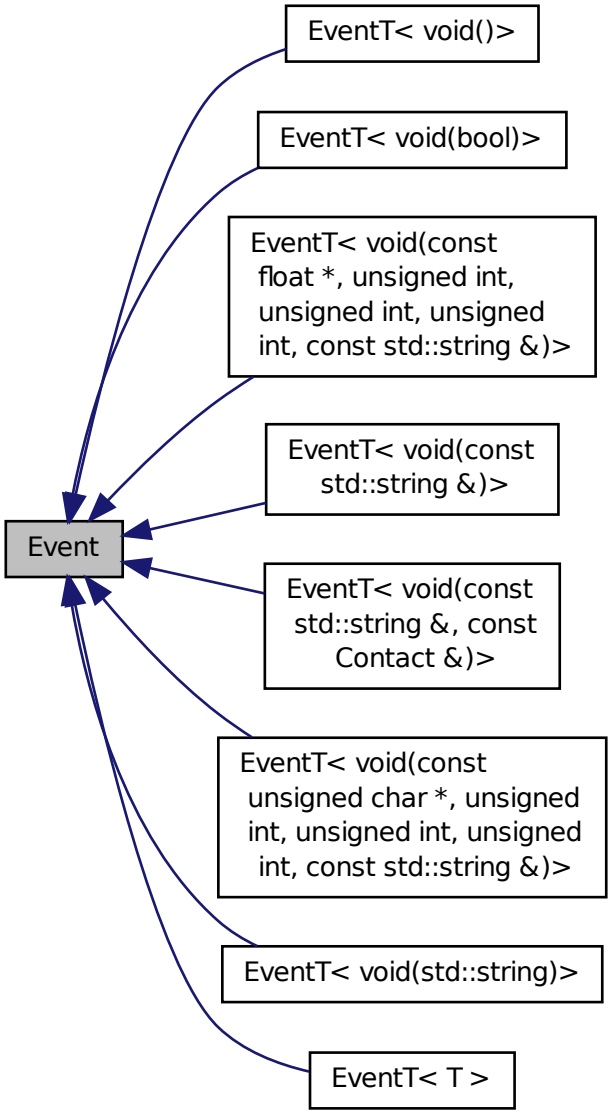
pose	The new world pose
notify	True = tell children of the pose change

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

- Entity.hh

9.48 Event Class Reference

Inheritance diagram for Event:



Public Member Functions

- virtual void **Disconnect** (ConnectionPtr _c)=0
- virtual void **Disconnect** (int _id)=0

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

- Event.hh

9.49 Events Class Reference

Static Public Member Functions

- template<typename T >
  static event::ConnectionPtr **ConnectCreateScene** (T subscriber)
- static void **DisconnectCreateScene** (event::ConnectionPtr subscriber)
- template<typename T >
  static event::ConnectionPtr **ConnectRemoveScene** (T subscriber)
- static void **DisconnectRemoveScene** (event::ConnectionPtr subscriber)

Static Public Attributes

- static **event::EventT**< void(const std::string &)> **createScene**
- static **event::EventT**< void(const std::string &)> **removeScene**

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

- RenderEvents.hh

9.50 Events Class Reference

Static Public Member Functions

- template<typename T >
  static ConnectionPtr **ConnectPause** (T _subscriber)
- static void **DisconnectPause** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectStep** (T _subscriber)

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

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

  Connect a boost::slot the the world created signal.
- static void **DisconnectWorldCreated** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectCreateEntity** (T _subscriber)

  Connect a boost::slot the the add entity signal.
- static void **DisconnectCreateEntity** (ConnectionPtr _subscriber)

- template<typename T >
  static ConnectionPtr **ConnectSetSelectedEntity** (T _subscriber)

  Connect a boost::slot the set selected entity.
- static void **DisconnectSetSelectedEntity** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectDeleteEntity** (T _subscriber)

  Connect a boost::slot the delete entity.
- static void **DisconnectDeleteEntity** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectAddEntity** (T _subscriber)

  Connect a boost::slot the the add entity signal.
- static void **DisconnectAddEntity** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowLights** (T _subscriber)

  Connect a boost::slot the the show light source signal.
- static void **DisconnectShowLights** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowCameras** (T _subscriber)

  Connect a boost::slot the the show camera source signal.
- static void **DisconnectShowCameras** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowContacts** (T _subscriber)

  Connect a boost::slot the the show contacts signal.
- static void **DisconnectShowContacts** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowWireframe** (T _subscriber)

  Connect a boost::slot the the show wireframe signal.
- static void **DisconnectShowWireframe** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowPhysics** (T _subscriber)

  Connect a boost::slot the the show physics signal.
- static void **DisconnectShowPhysics** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowJoints** (T _subscriber)

  Connect a boost::slot the the show joints signal.
- static void **DisconnectShowJoints** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectShowBoundingBoxes** (T _subscriber)

  Connect a boost::slot the the show bounding boxes signal.
- static void **DisconnectShowBoundingBoxes** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectWorldUpdateStart** (T _subscriber)

  Connect a boost::slot the the world update start signal.
- static void **DisconnectWorldUpdateStart** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectWorldUpdateEnd** (T _subscriber)

  Connect a boost::slot the the world update end signal.
- static void **DisconnectWorldUpdateEnd** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectEntitySelected** (T _subscriber)

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

- static void **DisconnectEntitySelected** (ConnectionPtr _subscriber)

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

- template<typename T >
  static ConnectionPtr **ConnectPreRender** (T _subscriber)

  Render start signal.

- static void **DisconnectPreRender** (ConnectionPtr _subscriber)

  Disconnect a render start signal.

- template<typename T >
  static ConnectionPtr **ConnectRender** (T _subscriber)

  Connect a boost::slot the render update signal.

- static void **DisconnectRender** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectPostRender** (T _subscriber)

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

- static void **DisconnectPostRender** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectDiagTimerStart** (T _subscriber)

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

- static void **DisconnectDiagTimerStart** (ConnectionPtr _subscriber)
- template<typename T >
  static ConnectionPtr **ConnectDiagTimerStop** (T _subscriber)

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

- static void **DisconnectDiagTimerStop** (ConnectionPtr _subscriber)

Static Public Attributes

- static **EventT**< void(bool)> **pause**

  Pause signal.

- static **EventT**< void()> **step**

  Step the simulation once signal.

- static **EventT**< void()> **stop**

  Simulation stop signal.

- static **EventT**< void(std::string)> **worldCreated**

  A world has been created.

- static **EventT**< void(std::string)> **entityCreated**

  An entity has been created.

- static **EventT**< void(std::string)> **setSelectedEntity**

  An entity has been selected.

- 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(bool)> **showLights**

  Light *(*p. *191) visuals should be shown.*

- static **EventT**< void(bool)> **showJoints**

  Joint *(*p. *181) visuals should be shown.*

- static **EventT**< void(bool)> **showCameras**

Camera visuals should be shown.

- static **EventT**< void(bool)> **showContacts**

  Contact visuals should be shown.
- static **EventT**< void(bool)> **wireframe**

  Wireframe enable signal.
- static **EventT**< void(bool)> **showPhysics**

  Show masses.
- static **EventT**< void(bool)> **showBoundingBoxes**

  Show bounding boxes.
- static **EventT**< void(std::string)> **entitySelected**

  Entity has been selected.
- static **EventT**< void()> **worldUpdateStart**

  World (p. *385) update has started.*
- static **EventT**< void()> **worldUpdateEnd**

  World (p. *385) update has ended.*
- static **EventT**< void()> **preRender**

  Pre-render.
- static **EventT**< void()> **render**

  Render.
- static **EventT**< void()> **postRender**

  Post-Render.
- static **EventT**< void(std::string)> **diagTimerStart**

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

  Diagnostic timer stop.

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

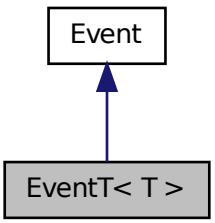
- Events.hh

9.51 EventT< T > Class Template Reference

An class for event processing.

`#include <Event.hh>`

Inheritance diagram for EventT< T >:



Public Member Functions

- 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)
- void **operator()** ()
- void **Signal** ()
- template<typename P >
  void **operator()** (const P &p)
- template<typename P1 , typename P2 >
  void **operator()** (const P1 &p1, const P2 &p2)
- template<typename P1 , typename P2 , typename P3 >
  void **operator()** (const P1 &p1, const P2 &p2, const P3 &p3)
- template<typename P1 , typename P2 , typename P3 , typename P4 >
  void **operator()** (const P1 &p1, const P2 &p2, const P3 &p3, const P4 &p4)
- 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)
- 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)
- 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)
- 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)
- 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)
- 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)
- template<typename P >
  void **Signal** (const P &p)
- template<typename P1 , typename P2 >
  void **Signal** (const P1 &p1, const P2 &p2)
- template<typename P1 , typename P2 , typename P3 >
  void **Signal** (const P1 &p1, const P2 &p2, const P3 &p3)
- template<typename P1 , typename P2 , typename P3 , typename P4 >
  void **Signal** (const P1 &p1, const P2 &p2, const P3 &p3, const P4 &p4)
- 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)
- 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)
- 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)
- 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)

- 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)
- 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)

9.51.1 Detailed Description

template<typename T>class gazebo::event::EventT< T >

An class for event processing.

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

- Event.hh

9.52 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)
  Ostream operator for Gazebo Error.

9.52.1 Detailed Description

Class for generating exceptions.

9.52.2 Constructor & Destructor Documentation

9.52.2.1 Exception (const char * *file*, int *line*, std::string *msg*)

Default constructor.

Parameters

file	File name
line	Line number where the error occurred
msg	Error message

9.52.3 Member Function Documentation

9.52.3.1 std::string GetErrorFile ( ) const

Return the error function.

Returns

The error function name

9.52.3.2 std::string GetErrorStr ( ) const

Return the error string.

Returns

The error string

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

- Exception.hh

9.53 Factory Interface Reference

Message to create new model in gazebo.

9.53.1 Detailed Description

Message to create new model in gazebo.

```
@verbatim
```

import "header.proto"; import "pose.proto";

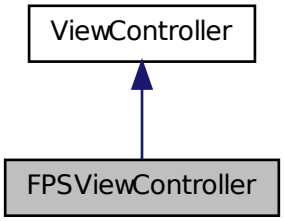
message **Factory** (p. 155) { optional string sdf = 1; optional string sdf_filename = 2; optional **Pose** (p. 283) pose = 3; optional string edit_name = 4; optional string clone_model_name = 5; } ///

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

- factory.proto

9.54 FPSViewController Class Reference

Inheritance diagram for FPSViewController:



Public Member Functions

- **FPSViewController** (**UserCamera** ∗camera)

  Constructor.
- virtual ∼**FPSViewController** ()

  Destructor.
- virtual void **Init** ()
- virtual void **Update** ()

  Update.
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &event)

  Handle a mouse event.

Static Public Member Functions

- static std::string **GetTypeString** ()

  Get the type name of this view controller.

Additional Inherited Members

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

- FPSViewController.hh

9.55 Frction Interface Reference

Information about friction.

9.55.1 Detailed Description

Information about friction.

```
@verbatim
```

import "vector3d.proto";

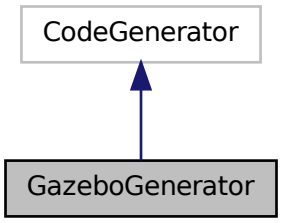
message Friction { optional double mu = 1; optional double mu2 = 2; optional **Vector3d** (p. 376) fdir1 = 3; optional double slip1 = 4; optional double slip2 = 5; } ///

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

- friction.proto

9.56 GazeboGenerator Class Reference

Inheritance diagram for GazeboGenerator:



Public Member Functions

- **GazeboGenerator** (const std::string &_name)
- virtual bool **Generate** (const FileDescriptor ∗file, const string &parameter, OutputDirectory ∗directory, string ∗error) const

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

- GazeboGenerator.hh

9.57 Geometry Interface Reference

Information about a geometry element.

9.57.1 Detailed Description

Information about a geometry element.

@verbatim

import "boxgeom.proto"; import "cylindergeom.proto"; import "spheregeom.proto"; import "planegeom.proto"; import "imagegeom.proto"; import "heightmapgeom.proto"; import "meshgeom.proto"; import "vector3d.proto";

message **Geometry** (p. 157) { enum Type { BOX = 1; CYLINDER = 2; SPHERE = 3; PLANE = 4; IMAGE = 5; HEIGHTMAP = 6; MESH = 7; TRIANGLE_FAN = 8; LINE_STRIP = 9; }

optional Type type = 1; optional **BoxGeom** (p. 96) box = 2; optional **CylinderGeom** (p. 125) cylinder = 3; optional PlaneGeom plane = 4; optional **SphereGeom** (p. 342) sphere = 5; optional ImageGeom image = 6; optional HeightmapGeom heightmap = 7; optional MeshGeom mesh = 8;

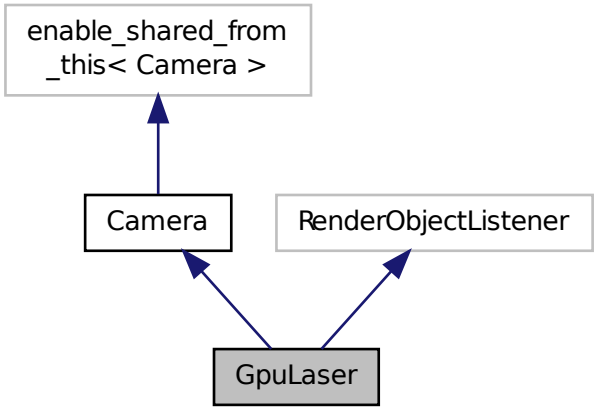
repeated **Vector3d** (p. 376) points = 9; } ///

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

- geometry.proto

9.58 GpuLaser Class Reference

Inheritance diagram for GpuLaser:



Public Member Functions

- **GpuLaser** (const std::string &_namePrefix, **Scene** *_scene, bool _autoRender=true)
  Constructor.
- virtual ~**GpuLaser** ()
  Destructor.

- void **Load** (sdf::ElementPtr &_sdf)

  Load the camera with a set of parmeters.
- void **Load** ()

  Load the camera with default parmeters.
- void **Init** ()

  Initialize the camera.
- void **Fini** ()

  Finalize the camera.
- void **CreateLaserTexture** (const std::string &_textureName)
- virtual void **PostRender** ()

  Render the camera.
- virtual const float * **GetLaserData** ()
- template<typename T >
  event::ConnectionPtr **ConnectNewLaserFrame** (T subscriber)

  Connect a to the add entity signal.
- void **DisconnectNewLaserFrame** (event::ConnectionPtr &c)
- void **SetRangeCount** (unsigned int _w, unsigned int _h=1)
- void **SetParentSensor** (**sensors::GpuRaySensor** *parent)
- virtual void **notifyRenderSingleObject** (Ogre::Renderable *rend, const Ogre::Pass *, const Ogre::AutoParamDataSource *, const Ogre::LightList *, bool)

Protected Member Functions

- virtual void **Set1stPassTarget** (Ogre::RenderTarget *target, unsigned int index)
- virtual void **Set2ndPassTarget** (Ogre::RenderTarget *target)

Protected Attributes

- Ogre::Texture * **_1stPassTextures** [3]
- Ogre::Texture * **_2ndPassTexture**
- Ogre::RenderTarget * **_1stPassTargets** [3]
- Ogre::RenderTarget * **_2ndPassTarget**
- Ogre::Viewport * **_1stPassViewports** [3]
- Ogre::Viewport * **_2ndPassViewport**
- unsigned int **_textureCount**
- double **cameraYaws** [4]
- Ogre::RenderTarget * **current_target**
- Ogre::Camera * **orthoCam**
- Ogre::SceneNode * **origParentNode_ortho**
- Ogre::SceneNode * **pitchNode_ortho**
- **common::Mesh** * **undist_mesh**
- Ogre::MovableObject * **object**
- VisualPtr **visual**
- unsigned int **w2nd**
- unsigned int **h2nd**
- **sensors::GpuRaySensor** * **parent_sensor**
- double **lastRenderDuration**

Additional Inherited Members

9.58.1 Member Function Documentation

9.58.1.1 void Load (sdf::ElementPtr & *_sdf*)

Load the camera with a set of parmeters.

Parameters

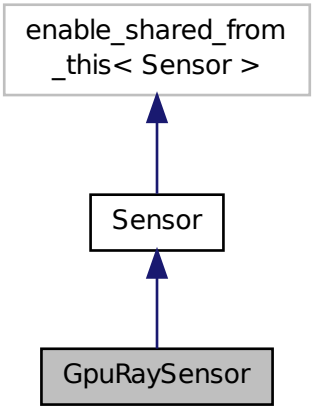
_sdf	The SDF camera info

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

- GpuLaser.hh

9.59 GpuRaySensor Class Reference

Inheritance diagram for GpuRaySensor:



Public Member Functions

- **GpuRaySensor** ()

  Constructor.
- virtual ~**GpuRaySensor** ()

  Destructor.
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr &_sdf)

  Load the ray using parameter from an SDF.
- virtual void **Load** (const std::string &_worldName)

Load the sensor with default parameters.

- virtual void **Init** ()

  Initialize the ray.

- rendering::GpuLaserPtr **GetLaserCamera** () const
- **math::Angle GetAngleMin** () const

  Get the minimum angle.

- void **SetAngleMin** (double angle)

  Set the scan minimum angle.

- **math::Angle GetAngleMax** () const

  Get the maximum angle.

- void **SetAngleMax** (double angle)

  Set the scan maximum angle.

- double **GetAngleResolution** () const

  Get radians between each range.

- double **GetRangeMin** () const

  Get the minimum range.

- double **GetRangeMax** () const

  Get the maximum range.

- double **GetRangeResolution** () const

  Get the range resolution.

- int **GetRayCount** () const

  Get the ray count.

- int **GetRangeCount** () const

  Get the range count.

- int **GetVerticalRayCount** () const

  Get the vertical scan line count.

- int **GetVerticalRangeCount** () const

  Get the vertical scan line count.

- **math::Angle GetVerticalAngleMin** () const

  Get the vertical scan bottom angle.

- void **SetVerticalAngleMin** (double angle)

  Set the vertical scan bottom angle.

- **math::Angle GetVerticalAngleMax** () const

  Get the vertical scan line top angle.

- void **SetVerticalAngleMax** (double angle)

  Set the vertical scan line top angle.

- double **GetRange** (int index)

  Get detected range for a ray.

- void **GetRanges** (std::vector< double > &_ranges)

  Get all the ranges.

- double **GetRetro** (int index)

  Get detected retro (intensity) value for a ray.

- int **GetFiducial** (int index)

  Get detected fiducial value for a ray.

- unsigned int **GetCameraCount** ()
- bool **IsHorizontal** ()
- double **Get1stRatio** ()

- double **Get2ndRatio** ()
- double **GetHFOV** ()
- double **GetCHFOV** ()
- double **GetVFOV** ()
- double **GetCVFOV** ()
- double **GetHAngle** ()
- double **GetVAngle** ()
- 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.
- void **DisconnectNewLaserFrame** (event::ConnectionPtr &c)

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.
- virtual void **Fini** ()

  Finalize the ray.

Protected Attributes

- **math::Vector3 offset**
- sdf::ElementPtr **rayElem**
- sdf::ElementPtr **scanElem**
- sdf::ElementPtr **horzElem**
- sdf::ElementPtr **vertElem**
- sdf::ElementPtr **rangeElem**
- sdf::ElementPtr **cameraElem**
- unsigned int **cameraCount**
- double **hfov**
- double **vfov**
- double **chfov**
- double **cvfov**
- double **hang**
- double **vang**
- double **near**
- double **far**
- unsigned int **width_1st**
- unsigned int **height_1st**
- unsigned int **width_2nd**
- unsigned int **height_2nd**
- double **ratio_1st**
- double **ratio_2nd**
- bool **isHorizontal**

9.59.1 Member Function Documentation

9.59.1.1 math::Angle GetAngleMax ( ) const

Get the maximum angle.

Returns

the maximum angle

9.59.1.2 math::Angle GetAngleMin ( ) const

Get the minimum angle.

Returns

The minimum angle

9.59.1.3 int 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 aceess 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).

9.59.1.4 double 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 aceess loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Returns

Returns DBL_MAX for no detection.

9.59.1.5 int GetRangeCount ( ) const

Get the range count.

Returns

The number of ranges

9.59.1.6 double GetRangeMax ( ) const

Get the maximum range.

Returns

The maximum range

9.59.1.7 double GetRangeMin ( ) const

Get the minimum range.

Returns

The minimum range

9.59.1.8 void GetRanges (std::vector< double > & *_ranges*)

Get all the ranges.

Parameters

_range	A vector that will contain all the range data

9.59.1.9 int GetRayCount ( ) const

Get the ray count.

Returns

The number of rays

9.59.1.10 double 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 aceess 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).

9.59.1.11 math::Angle GetVerticalAngleMax ( ) const

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

9.59.1.12 math::Angle GetVerticalAngleMin ( ) const

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

9.59.1.13 int GetVerticalRangeCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

9.59.1.14 int GetVerticalRayCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

9.59.1.15 virtual void Load (const std::string & *_worldName*, sdf::ElementPtr & *_sdf*) [virtual]

Load the ray using parameter from an SDF.

Parameters

node	The XMLConfig node

9.59.1.16 void SetAngleMax (double *angle*)

Set the scan maximum angle.

Parameters

The	maximum angle

9.59.1.17 void SetAngleMin (double *angle*)

Set the scan minimum angle.

Parameters

The	minimum angle

9.59.1.18 void SetVerticalAngleMax (double *angle*)

Set the vertical scan line top angle.

Parameters

The	Maximum angle of the scan block

9.59.1.19 void SetVerticalAngleMin (double *angle*)

Set the vertical scan bottom angle.

Parameters

The	minimum angle of the scan block

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

- GpuRaySensor.hh

9.60 Grid Class Reference

Displays a grid of cells, drawn with lines.

`#include <Grid.hh>`

Public Member Functions

- **Grid** (**Scene** *scene_, uint32_t cellCount_, float cellLength_, float lineWidth_, const **common::Color** &color_)

  Constructor.
- ~**Grid** ()

  Destructor.
- void **Init** ()
- Ogre::SceneNode * **GetSceneNode** ()

  Get the Ogre scene node associated with this grid.
- void **SetUserData** (const Ogre::Any &data_)

  Sets user data on all ogre objects we own.
- void **SetColor** (const **common::Color** &color_)
- **common::Color GetColor** ()
- void **SetCellCount** (uint32_t count_)
- float **GetCellCount** ()
- void **SetCellLength** (float len_)
- float **GetCellLength** ()
- void **SetLineWidth** (float width_)
- float **GetLineWidth** ()
- void **SetHeight** (uint32_t count_)
- uint32_t **GetHeight** ()

9.60.1 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 XZ plane.

9.60.2 Constructor & Destructor Documentation

9.60.2.1 Grid (Scene * *scene_*, uint32_t *cellCount_*, float *cellLength_*, float *lineWidth_*, const common::Color & *color_*)

Constructor.

Parameters

Scene *(p. 322)*	The scene this object is part of
cell_count	The number of cells to draw
cell_length	The size of each cell
r	Red color component, in the range [0, 1]
g	Green color component, in the range [0, 1]
b	Blue color component, in the range [0, 1]

9.60.3 Member Function Documentation

9.60.3.1 Ogre::SceneNode* GetSceneNode ( ) [inline]

Get the Ogre scene node associated with this grid.

Returns

The Ogre scene node associated with this grid

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

- Grid.hh

9.61 Gripper Class Reference

A gripper abstraction.

`#include <Gripper.hh>`

Public Member Functions

- **Gripper** (ModelPtr _model)
  Constructor.
- virtual ~**Gripper** ()
  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)
  Load.
- virtual void **Init** ()
  Initialize.

9.61.1 Detailed Description

A gripper abstraction.

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

- Gripper.hh

9.62 GUI Interface Reference

Message for a **GUI** (p. 168).

9.62.1 Detailed Description

Message for a **GUI** (p. 168).

Message for a gui overlay configuration.

Message for a **GUI** (p. 168) Camera.

```
@verbatim
```

import "gui_camera.proto";

message **GUI** (p. 168) { optional bool fullscreen = 1; optional GUICamera camera = 2; }

///

```
@verbatim
```

import "pose.proto"; import "track_visual.proto";

message GUICamera { required string name = 1; optional string view_controller = 2; optional **Pose** (p. 283) origin = 3; optional TrackVisual track = 4; }

///

```
@verbatim
```

message GUIOverlayConfig { required string layout_filename = 1; }

///

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

- gui.proto

9.63 GUIOverlay Class Reference

Public Member Functions

- void **Init** (Ogre::RenderTarget *_renderTarget)
- void **Hide** ()

- void **Show** ()
- void **Update** ()
- bool **IsInitialized** ()
- 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)
- bool **HandleMouseEvent** (const **common::MouseEvent** &_evt)
- bool **HandleKeyPressEvent** (const std::string &_key)
- bool **HandleKeyReleaseEvent** (const std::string &_key)
- void **LoadLayout** (const std::string &_filename)

  Load a CEGUI layout file.
- bool **AttachCameraToImage** (CameraPtr &_camera, const std::string &_windowName)
- bool **AttachCameraToImage** (DepthCameraPtr &_camera, const std::string &_windowName)
- void **Resize** (unsigned int _width, unsigned int _height)
- template<typename T >
  void **ButtonCallback** (const std::string &_buttonName, void(T::*_fp)(), T *_obj)

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

- GUIOverlay.hh

9.64 Header Interface Reference

General information included by many messages.

9.64.1 Detailed Description

General information included by many messages.

`@verbatim`

import "time.proto";

message **Header** (p. 169) { optional string str_id = 1; optional **Time** (p. 359) stamp = 2; optional int32 index = 3; } ///

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

- header.proto

9.65 Heightmap Class Reference

Public Member Functions

- **Heightmap** (ScenePtr scene)

  Constructor.
- virtual ~**Heightmap** ()

  Destructor.
- void **Load** ()

  Load the heightmap.

- void **LoadFromMsg** (ConstVisualPtr &_msg)

  Load the heightmap from a visual message.

- double **GetHeight** (double x, double y)

  Get the height at a location.

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

- Heightmap.hh

9.66 Heightmap Interface Reference

Message for a heightmap geometry.

9.66.1 Detailed Description

Message for a heightmap geometry.

```
@verbatim
```

import "image.proto"; import "vector3d.proto";

message HeightmapGeom { required **Image** (p. 176) image = 1; // The height data required **Vector3d** (p. 376) size = 2; // Size in meters optional **Vector3d** (p. 376) origin = 3; // Origin in world coordinate frame

message Texture { required string diffuse = 1; required string normal = 2; required double size = 3; }

message Blend { required double min_height = 1; required double fade_dist = 2; }

repeated Texture texture = 4; // List of textures repeated Blend blend = 5; // How to blend the textures }

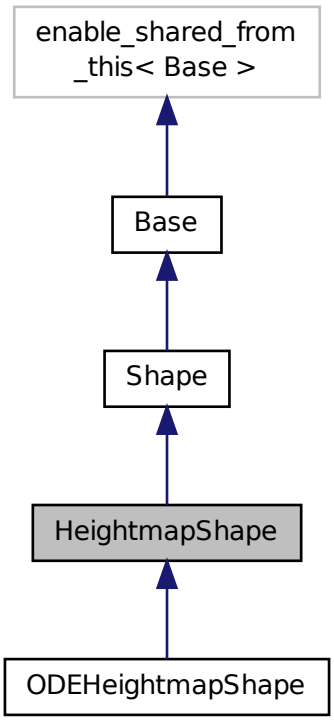
///

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

- heightmapgeom.proto

9.67 HeightmapShape Class Reference

Inheritance diagram for HeightmapShape:



Public Member Functions

- **HeightmapShape** (CollisionPtr **parent**)
  Constructor.
- virtual ~**HeightmapShape** ()
  Destructor.
- void **Update** ()
  Update function.
- virtual void **Load** (sdf::ElementPtr _sdf)
  Load the heightmap.
- virtual void **Init** ()
  Initialize the heightmap.
- std::string **GetFilename** () const
- **math::Vector3 GetSize** () const
- **math::Vector3 GetOrigin** () const
- void **FillShapeMsg** (msgs::Geometry &_msg)
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

Protected Attributes

- std::vector< float > **heights**
- **common::Image img**

Additional Inherited Members

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

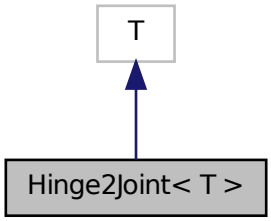
- HeightmapShape.hh

9.68 Hinge2Joint< T > Class Template Reference

A two axis hinge joint.

`#include <Hinge2Joint.hh>`

Inheritance diagram for Hinge2Joint< T >:



Public Member Functions

- **Hinge2Joint** ()

  Constructor.
- virtual ~**Hinge2Joint** ()

  Destructor.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the joint.

9.68.1 Detailed Description

template<class T>class gazebo::physics::Hinge2Joint< T >

A two axis hinge joint.

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

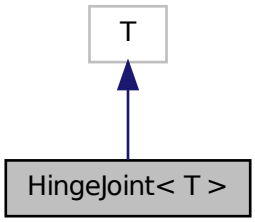
- Hinge2Joint.hh

9.69 HingeJoint< T > Class Template Reference

A single axis hinge joint.

`#include <HingeJoint.hh>`

Inheritance diagram for HingeJoint< T >:



Public Member Functions

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

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)
  Load joint.
- virtual void **Init** ()

9.69.1 Detailed Description

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

A single axis hinge joint.

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

- HingeJoint.hh

9.70 Image Class Reference

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** }

Public Member Functions

- **Image** (const std::string &_filename="")

  Constructor.
- virtual ∼**Image** ()

  Destructor.
- int **Load** (const std::string &_filename)

  Load an image. Return 0 on success.
- void **SetFromData** (const unsigned char ∗data, unsigned int width, unsigned int height, int scanline_bytes, unsigned int bpp) GAZEBO_DEPRECATED

  Set the image from raw data (R8G8B8)
- void **SetFromData** (const unsigned char ∗_data, unsigned int _width, unsigned int _height, Image::PixelFormat _format)

  Set the image from raw data.
- void **GetData** (unsigned char ∗∗_data, unsigned int &_count) const

  Get the image as a data array.
- unsigned int **GetWidth** () const

  Get the width.
- unsigned int **GetHeight** () const

  Get the height.
- unsigned int **GetBPP** () const

  Get the size of one pixel in bits.
- int **GetPitch** () const
- std::string **GetFilename** () const

  Get the full filename of the image.
- PixelFormat **GetPixelFormat** () const

  Get the pixel format.
- **Color GetPixel** (unsigned int _x, unsigned int _y)

  Get a pixel color value.
- **Color GetAvgColor** ()

  Get the average color.
- **Color GetMaxColor** ()

Get the max color.

- void **Rescale** (int _width, int _height)

  Rescale the image.

- bool **Valid** () const

  Render this image using opengl.

9.70.1 Member Function Documentation

9.70.1.1 void GetData (unsigned char ** *_data*, unsigned int & *_count*) const

Get the image as a data array.

Parameters

_data	Pointer to a NULL array of char.
_count	The resulting data array size

9.70.1.2 void SetFromData (const unsigned char * *data*, unsigned int *width*, unsigned int *height*, int *scanline_bytes*, unsigned int *bpp*)

Set the image from raw data (R8G8B8)

Parameters

data	Pointer to the raw image data
width	Width in pixels
height	Height in pixels
scanline_bytes	Size of a image row in bytes
bpp	Bits per pixels, aka depth

9.70.1.3 void SetFromData (const unsigned char * *_data*, unsigned int *_width*, unsigned int *_height*, Image::PixelFormat *_format*)

Set the image from raw data.

Parameters

_data	Pointer to the raw image data
_width	Width in pixels
_height	Height in pixels
_format	Pixel format of the provided data

9.70.1.4 bool Valid ( ) const

Render this image using opengl.

Returns whether this is a valid image

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

- Image.hh

9.71 Image Interface Reference

Message for an image.

9.71.1 Detailed Description

Message for an image.

Message for a image geometry.

Message for an image with a time.

```
@verbatim
```

message **Image** (p. 176) { required uint32 width = 1; // **Image** (p. 176) width (number of columns) required uint32 height = 2; // **Image** (p. 176) height (number of rows) required uint32 pixel_format = 3; // Corresponds to Image::PixelFormate enum required uint32 step = 4; // Full row length in bytes // repeated uint32 data = 5; // Actual data, size if (step ∗ rows) required bytes data = 5; // Actual data, size if (step ∗ rows) }

///

```
@verbatim
```

import "time.proto"; import "image.proto";

message ImageStamped { required **Time** (p. 359) time = 1; // **Time** (p. 359) when the data was captured required **Image** (p. 176) image = 2; }

///

```
@verbatim
```

message ImageGeom { required string filename = 1; optional double scale = 2; optional int32 threshold = 3 [default = 255]; optional double height = 4; optional int32 granularity = 5; }

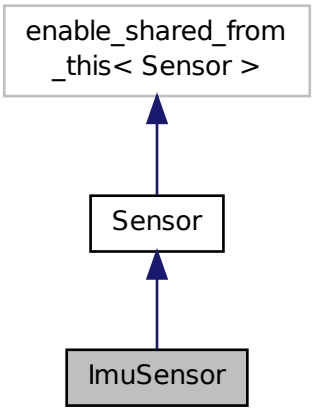
///

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

- image.proto

9.72 ImuSensor Class Reference

Inheritance diagram for ImuSensor:



Public Member Functions

- **ImuSensor** (Body *body)

  Constructor.
- virtual ~**ImuSensor** ()

  Destructor.
- **Pose GetVelocity** ()

Protected Member Functions

- virtual void **LoadChild** (XMLConfigNode *node)
- virtual void **SaveChild** (std::string &prefix, std::ostream &stream)

  Save the sensor info in XML format.
- virtual void **InitChild** ()

  Initialize the ray.
- virtual void **UpdateChild** ()

  Update sensed values.
- virtual void **FiniChild** ()

  Finalize the ray.

Additional Inherited Members

9.72.1 Constructor & Destructor Documentation

9.72.1.1 ImuSensor (Body * *body*)

Constructor.

Parameters

body	The IMU sensor must be attached to a body.

9.72.2 Member Function Documentation

9.72.2.1 virtual void LoadChild (XMLConfigNode * *node*) [protected], [virtual]

Parameters

node	The XMLConfig node

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

- ImuSensor.hh

9.73 Inertial Interface Reference

Information about inertia.

9.73.1 Detailed Description

Information about inertia.

@verbatim

import "pose.proto";

message **Inertial** (p. 178) { optional double mass = 1; optional **Pose** (p. 283) pose = 2; optional double ixx = 3; optional double ixy = 4; optional double ixz = 5; optional double iyy = 6; optional double iyz = 7; optional double izz = 8; } ///

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

- inertial.proto

9.74 Inertial Class Reference

Public Member Functions

- **Inertial** ()
  Default Constructor.
- **Inertial** (double mass)
  Constructor.
- **Inertial** (const **Inertial** &_inertial)
  Copy constructor.

- virtual ~**Inertial** ()

  Destructor.
- void **Load** (sdf::ElementPtr _sdf)
- void **UpdateParameters** (sdf::ElementPtr _sdf)

  update the parameters using new sdf values
- void **Reset** ()

  Reset all the mass properties.
- void **SetMass** (double m)

  Set the mass.
- double **GetMass** () const

  Get the mass.
- void **SetInertiaMatrix** (double ixx, double iyy, double izz, double ixy, double ixz, double iyz)

  Set the mass matrix.
- void **SetCoG** (double cx, double cy, double cz)

  Set the center of gravity.
- void **SetCoG** (const **math::Vector3** &c)

  Set the center of gravity.
- const **math::Vector3** & **GetCoG** () const

  Get the center of gravity.
- const **math::Pose GetPose** () const
- **math::Vector3 GetPrincipalMoments** () const

  Get the prinicpal moments of inertia (Ixx, Iyy, Izz)
- **math::Vector3 GetProductsofInertia** () const

  Get the products of inertia (Ixy, Ixy, Iyz)
- double **GetIXX** () const
- double **GetIYY** () const
- double **GetIZZ** () const
- double **GetIXY** () const
- double **GetIXZ** () const
- double **GetIYZ** () const
- void **SetIXX** (double _v)
- void **SetIYY** (double _v)
- void **SetIZZ** (double _v)
- void **SetIXY** (double _v)
- void **SetIXZ** (double _v)
- void **SetIYZ** (double _v)
- void **SetBoxDensity** (double _density, const **math::Vector3** &_size)

  Set the inertia matrix for a box based on a density.
- void **SetBoxMass** (double _mass, const **math::Vector3** &_size)

  Set the inertia matrix for a box based on a total mass.
- void **SetCylinderDensity** (double _density, double _radius, double _length)

  Set the inertia matrix for a cylinder based on a density.
- void **SetCylinderMass** (double _mass, double _radius, double _length)

  Set the inertia matrix for a cylinder based on a total mass.
- void **SetSphereDensity** (double _density, double _radius)

  Set the inertia matrix for a sphere based on a density.
- void **SetSphereMass** (double _mass, double _radius)

  Set the inertia matrix for a sphere based on a total mass.

- void **SetMeshDensity** (double _density, const std::string &_meshName)

  Set the inertia matrix for a mesh based on a density.
- void **SetMeshMass** (double _mass, const std::string &_meshName)

  Set the inertia matrix for a mesh based on a total mass.
- void **Rotate** (const **math::Quaternion** &rot)

  Rotate this mass.
- **Inertial** & **operator=** (const **Inertial** &_inertial)

  Equal operator.
- **Inertial operator+** (const **Inertial** &_inertial) const
- const **Inertial** & **operator+=** (const **Inertial** &_inertial)
- void **ProcessMsg** (const msgs::Inertial &_msg)

  Update parameters from a message.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::physics::Inertial** &_inertial)

9.74.1 Member Function Documentation

9.74.1.1 void ProcessMsg (const msgs::Inertial & *_msg*)

Update parameters from a message.

Parameters

_msg	Message to read

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

- Inertial.hh

9.75 Int Interface Reference

Integer message.

9.75.1 Detailed Description

Integer message.

```
@verbatim
```

message **Int** (p. 180) { /// Integer data required int32 data = 1; }

///

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

- int.proto

9.76 IOManager Class Reference

Public Member Functions

- boost::asio::io_service & **GetIO** ()
- void **IncCount** ()
- void **DecCount** ()
- unsigned int **GetCount** () const
- void **Stop** ()

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

- IOManager.hh

9.77 Joint Interface Reference

Message for joints.

9.77.1 Detailed Description

Message for joints.

Message for a model joint animation.

@verbatim

import "vector3d.proto"; import "axis.proto"; import "pose.proto";

message **Joint** (p. 181) { enum Type { REVOLUTE = 1; REVOLUTE2 = 2; PRISMATIC = 3; UNIVERSAL = 4; BALL = 5; SCREW = 6; }

required string name = 1; optional Type type = 2; optional string parent = 3; optional string child = 4; optional **Pose** (p. 283) pose = 5; optional **Axis** (p. 84) axis1 = 6; optional **Axis** (p. 84) axis2 = 7;

optional double cfm = 8; optional double bounce = 9; optional double velocity = 10; optional double fudge_factor = 11; optional double limit_cfm = 12; optional double limit_erp = 13; optional double suspension_cfm = 14; optional double suspension_erp = 15; } ///

@verbatim

import "pose.proto"; import "time.proto";

message JointAnimation { message **Joint** (p. 181) { repeated string name = 1; repeated double angle = 2; }

required string model_name = 1; repeated **Joint** (p. 181) joint = 2; repeated **Time** (p. 359) time = 3; }

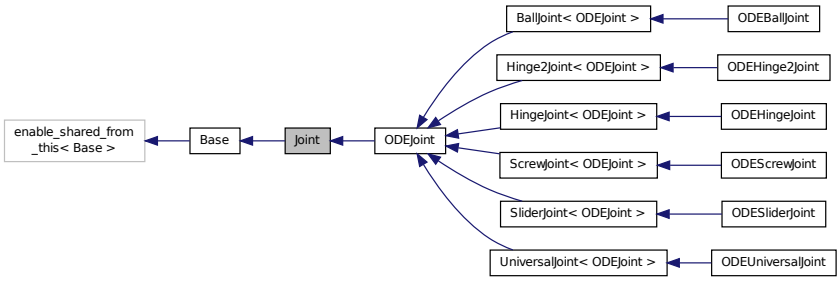
///

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

- joint.proto

9.78 Joint Class Reference

Inheritance diagram for Joint:



Public Types

- enum **Attribute** {
  FUDGE_FACTOR, **SUSPENSION_ERP**, **SUSPENSION_CFM**, **STOP_ERP**,
  STOP_CFM, **ERP**, **CFM**, **FMAX**,
  VEL, **HI_STOP**, **LO_STOP** }

  Type of joint.

Public Member Functions

- **Joint** ()

  Constructor.
- virtual ~**Joint** ()

  Destructor.
- void **Load** (LinkPtr _parent, LinkPtr _child, const **math::Pose** &_origin)

  Load a joint.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load a joint.
- virtual void **Init** ()

  Initialize a joint.
- void **Update** ()

  Update the joint.
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  update the parameters using new sdf values
- virtual void **Reset** ()

  Reset the joint.
- **JointState GetState** ()

  Get the joint state.
- void **SetState** (const **JointState** &_state)

  Set the joint state.
- void **SetModel** (ModelPtr model)

Set the model this joint belongs too.

- virtual LinkPtr **GetJointLink** (int index) const =0

  Get the link to which the joint is attached according the _index.
- virtual bool **AreConnected** (LinkPtr one, LinkPtr two) const =0

  Determines of the two bodies are connected by a joint.
- virtual void **Attach** (LinkPtr **parent**, LinkPtr child)

  Attach the two bodies with this joint.
- virtual void **Detach** ()=0

  Detach this joint from all bodies.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)=0

  Set the axis of rotation.
- virtual void **SetDamping** (int index, const double damping)=0

  Set the joint damping.
- template<typename T >
  event::ConnectionPtr **ConnectJointUpdate** (T subscriber)

  Connect a boost::slot the the joint update signal.
- void **DisconnectJointUpdate** (event::ConnectionPtr &c)

  Disconnect a boost::slot the the joint update signal.
- **math::Vector3 GetLocalAxis** (int _index) const

  Get the axis of rotation.
- virtual **math::Vector3 GetGlobalAxis** (int _index) const =0
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)=0

  Set the anchor point.
- virtual **math::Vector3 GetAnchor** (int index) const =0

  Get the anchor point.
- virtual void **SetHighStop** (int index, **math::Angle** angle)=0

  Set the high stop of an axis(index).
- virtual void **SetLowStop** (int index, **math::Angle** angle)=0

  Set the low stop of an axis(index).
- virtual **math::Angle GetHighStop** (int index)=0

  Get the high stop of an axis(index).
- virtual **math::Angle GetLowStop** (int index)=0

  Get the low stop of an axis(index).
- virtual void **SetVelocity** (int index, double v)=0

  Set the velocity of an axis(index).
- virtual double **GetVelocity** (int index) const =0

  Get the rotation rate of an axis(index)
- virtual void **SetForce** (int, double)

  Set the force applied to an axis.
- virtual double **GetForce** (int)

  Get the force applied to an axis.
- virtual void **SetMaxForce** (int index, double t)=0

  Set the max allowed force of an axis(index).
- virtual double **GetMaxForce** (int index)=0

  Get the max allowed force of an axis(index).
- **math::Angle GetAngle** (int index) const

  Get the angle of rotation of an axis(index)

- void **SetAngle** (int _index, **math::Angle** _angle)

  Set the angle of rotation.
- virtual **math::Vector3 GetLinkForce** (unsigned int index) const =0

  Get the force the joint applies to the first link.
- virtual **math::Vector3 GetLinkTorque** (unsigned int index) const =0

  Get the torque the joint applies to the first link.
- virtual void **SetAttribute** (**Attribute**, int index, double value)=0

  Set a parameter for the joint.
- LinkPtr **GetChild** () const

  Get the child link.
- LinkPtr **GetParent** () const

  Get the child link.
- void **FillJointMsg** (msgs::Joint &_msg)

  Fill a joint message.

Protected Member Functions

- virtual **math::Angle GetAngleImpl** (int _index) const =0

Protected Attributes

- LinkPtr **childLink**

  The first link this joint connects to.
- LinkPtr **parentLink**

  The second link this joint connects to.
- std::string **visual**
- std::string **line1**
- std::string **line2**
- bool **showJoints**
- ModelPtr **model**
- **math::Vector3 anchorPos**
- LinkPtr **anchorLink**
- double **damping_coefficient**
- transport::PublisherPtr **vis_pub**

9.78.1 Member Function Documentation

9.78.1.1 virtual math::Vector3 GetLinkForce (unsigned int *index*) const [pure virtual]

Get the force the joint applies to the first link.

Parameters

index	The index of the link(0 or 1)

Implemented in **ODEJoint** (p. 247).

9.78.1.2 virtual math::Vector3 GetLinkTorque (unsigned int *index*) const [pure virtual]

Get the torque the joint applies to the first link.

Parameters

index	The index of the link(0 or 1)

Implemented in **ODEJoint** (p. 247).

9.78.1.3 void SetAngle (int *_index*, math::Angle *_angle*)

Set the angle of rotation.

This will not move the joint. The purpose of this function is purely to support animation of static models.

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

- Joint.hh

9.79 JointController Class Reference

Public Member Functions

- **JointController** (ModelPtr _model)
- void **AddJoint** (JointPtr _joint)
- void **Update** ()
- void **Reset** ()

  Reset all commands.
- void **SetJointPosition** (const std::string &_name, double _position)

  Set the position of a joint.
- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)

  Set the positions of a set of joints.

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

- JointController.hh

9.80 JointFeedback Class Reference

Public Member Functions

- **JointFeedback** & **operator=** (const **JointFeedback** &f)

  Operator =.

Public Attributes

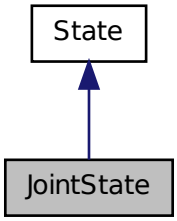
- **math::Vector3 body1Force**
- **math::Vector3 body2Force**
- **math::Vector3 body1Torque**
- **math::Vector3 body2Torque**

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

- JointFeedback.hh

9.81 JointState Class Reference

Inheritance diagram for JointState:



Public Member Functions

- **JointState** ()
  Default constructor.
- **JointState** (JointPtr _joint)
  Constructor.
- virtual ~**JointState** ()
  Destructor.
- virtual void **Load** (sdf::ElementPtr _elem)
  Load state from SDF element.
- unsigned int **GetAngleCount** () const
  Get the number of angles.
- **math::Angle GetAngle** (unsigned int _axis) const
  Get the joint angle.

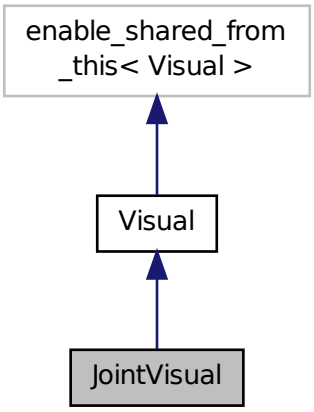
Additional Inherited Members

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

- JointState.hh

9.82 JointVisual Class Reference

Inheritance diagram for JointVisual:



Public Member Functions

- **JointVisual** (const std::string &_name, VisualPtr _vis)
- void **Load** (ConstJointPtr &_msg)

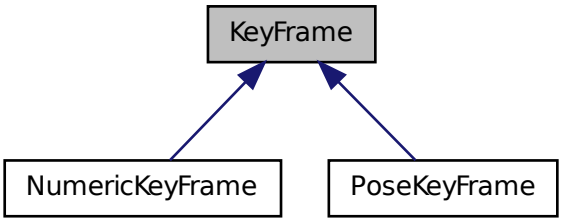
Additional Inherited Members

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

- JointVisual.hh

9.83 KeyFrame Class Reference

Inheritance diagram for KeyFrame:



Public Member Functions

- **KeyFrame** (double _time)
- double **GetTime** () const

Protected Attributes

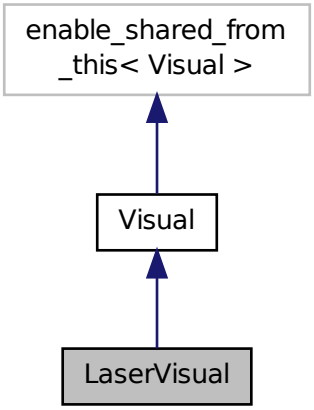
- double **time**

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

- KeyFrame.hh

9.84 LaserVisual Class Reference

Inheritance diagram for LaserVisual:



Public Member Functions

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

Additional Inherited Members

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

- LaserVisual.hh

9.85 Light Class Reference

Public Member Functions

- **Light** (**Scene** ∗scene)

  Constructor.
- virtual ∼**Light** ()

  Destructor.
- 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 **SetName** (const std::string &name)

  Set the name of the visual.
- std::string **GetName** () const

  Get the name of the visual.
- std::string **GetType** () const

  Get the type of the light.
- void **SetPosition** (const **math::Vector3** &p)

  Set the position of the light.
- virtual bool **SetSelected** (bool s)

  Set whether this entity has been selected by the user through the gui.
- void **ToggleShowVisual** ()
- void **ShowVisual** (bool s)

  Set whether to show the visual.
- void **SetLightType** (const std::string &type)

  Set the light type.
- void **SetDiffuseColor** (const **common::Color** &color)

  Set the diffuse.
- **common::Color GetDiffuseColor** () const

  Get the diffuse color.
- void **SetSpecularColor** (const **common::Color** &color)

  Set the specular color.
- void **SetDirection** (const **math::Vector3** &dir)

  Set the direction.
- **math::Vector3 GetDirection** () const

  Get the direction.
- void **SetAttenuation** (double constant, double linear, double quadratic)

  Set the attenuation.
- void **SetSpotInnerAngle** (const double &angle)

  Set the spot light inner angle.
- void **SetSpotOuterAngle** (const double &angle)

  Set the spot light outter angle.
- void **SetSpotFalloff** (const double &angle)

  Set the spot light falloff.
- void **SetRange** (const double &range)

  Set the range.
- void **SetCastShadows** (const bool &cast)

  Set cast shadowsj.

Protected Member Functions

- virtual void **OnPoseChange** ()

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

- Light.hh

9.86 Light Interface Reference

Message for a light.

9.86.1 Detailed Description

Message for a light.

@verbatim

import "header.proto"; import "pose.proto"; import "vector3d.proto"; import "color.proto";

message **Light** (p. 191) { required string name = 1; enum LightType { POINT = 1; SPOT = 2; DIRECTIONAL = 3; } optional LightType type = 2;

optional **Pose** (p. 283) pose = 3; optional **Color** (p. 115) diffuse = 4; optional **Color** (p. 115) specular = 5; optional float attenuation_constant = 6; optional float attenuation_linear = 7; optional float attenuation_quadratic = 8; optional **Vector3d** (p. 376) direction = 9; optional float range = 10; optional bool cast_shadows = 11; optional float spot_inner_angle = 12; optional float spot_outer_angle = 13; optional float spot_falloff = 14; }

///

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

- light.proto

9.87 Link Interface Reference

Information about a link.

9.87.1 Detailed Description

Information about a link.

@verbatim

import "header.proto"; import "inertial.proto"; import "collision.proto"; import "visual.proto"; import "sensor.proto"; import "projector.proto"; import "pose.proto";

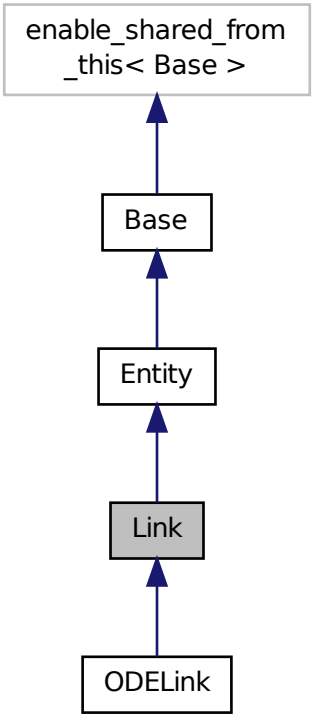
message **Link** (p. 191) { required uint32 id = 1; required string name = 2; optional bool self_collide = 3; optional bool gravity = 4; optional bool kinematic = 5; optional bool enabled = 6; optional **Inertial** (p. 178) inertial = 7; optional **Pose** (p. 283) pose = 8; repeated **Visual** (p. 384) visual = 9; repeated **Collision** (p. 109) collision = 10; repeated **Sensor** (p. 328) sensor = 11; repeated **Projector** (p. 291) projector = 12; } ///

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

- link.proto

9.88 Link Class Reference

Inheritance diagram for Link:



Public Member Functions

- **Link** (EntityPtr **parent**)

  Constructor.
- virtual ~**Link** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the body based on an SDF element.
- virtual void **Init** ()

  Initialize the body.
- void **Fini** ()

  Finalize the body.
- void **Reset** ()

  Reset the link.
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  Update the parameters using new sdf values.

- virtual void **Update** ()

  Update the body.
- virtual void **SetEnabled** (bool enable) const =0

  Set whether this body is enabled.
- virtual bool **GetEnabled** () const =0

  Get whether this body is enabled in the physics engine.
- virtual bool **SetSelected** (bool s)

  Set whether this entity has been selected by the user through the gui.
- virtual void **SetGravityMode** (bool mode)=0

  Set whether gravity affects this body.
- virtual bool **GetGravityMode** ()=0

  Get the gravity mode.
- virtual void **SetSelfCollide** (bool collide)=0

  Set whether this body will collide with others in the model.
- void **SetCollideMode** (const std::string &m)

  Set the collide mode of 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.
- void **SetLaserRetro** (float retro)

  Set the laser retro reflectiveness.
- virtual void **SetLinearVel** (const **math::Vector3** &vel)=0

  Set the linear velocity of the body.
- virtual void **SetAngularVel** (const **math::Vector3** &vel)=0

  Set the angular velocity of the body.
- void **SetLinearAccel** (const **math::Vector3** &accel)

  Set the linear acceleration of the body.
- void **SetAngularAccel** (const **math::Vector3** &accel)

  Set the angular acceleration of the body.
- virtual void **SetForce** (const **math::Vector3** &_force)=0

  Set the force applied to the body.
- virtual void **SetTorque** (const **math::Vector3** &_force)=0

  Set the torque applied to the body.
- virtual void **AddForce** (const **math::Vector3** &_force)=0

  Add a force to the body.
- 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 **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)=0

  Add a force to the body using a global position.
- 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 **AddTorque** (const **math::Vector3** &_torque)=0

  Add a torque to the body.
- 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.
- **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the body.
- **math::Vector3 GetRelativeAngularVel** () const

Get the angular velocity of the body.

- **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the body.
- **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the body in the world frame.
- **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the body.
- **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the body in the world frame.
- **math::Vector3 GetRelativeForce** () const

  Get the force applied to the body.
- virtual **math::Vector3 GetWorldForce** () const =0

  Get the force applied to the body in the world frame.
- **math::Vector3 GetRelativeTorque** () const

  Get the torque applied to the body.
- virtual **math::Vector3 GetWorldTorque** () const =0

  Get the torque applied to the body in the world frame.
- ModelPtr **GetModel** () const

  Get the model that this body belongs to.
- InertialPtr **GetInertial** () const

  Get the mass of the body.
- void **SetInertial** (const InertialPtr &_inertial)

  Set the mass of the body.
- CollisionPtr **GetCollisionById** (unsigned int _id) const

  Get a collision by id.
- CollisionPtr **GetCollision** (const std::string &name)

  accessor for collisions
- CollisionPtr **GetCollision** (unsigned int _index) const

  accessor for collisions
- virtual **math::Box GetBoundingBox** () const

  Get the size of the body.
- virtual void **SetLinearDamping** (double _damping)=0

  Set the linear damping factor.
- virtual void **SetAngularDamping** (double _damping)=0

  Set the angular damping factor.
- double **GetLinearDamping** () const

  Get the linear damping factor.
- double **GetAngularDamping** () const

  Get the angular damping factor.
- virtual void **SetKinematic** (const bool &)

  Set whether this body is in the kinematic state.
- virtual bool **GetKinematic** () const

  Get whether this body is in the kinematic state.
- unsigned int **GetSensorCount** () const

  Get sensor count.
- std::string **GetSensorName** (unsigned int _i) const

  Get sensor name.

- template<typename T >
  event::ConnectionPtr **ConnectEnabled** (T subscriber)

  Connect a to the add entity signal.
- void **DisconnectEnabled** (event::ConnectionPtr &c)
- void **FillLinkMsg** (msgs::Link &_msg)

  Fill a link message.
- void **ProcessMsg** (const msgs::Link &_msg)

  Update parameters from a message.
- void **AddChildJoint** (JointPtr joint)

  *Joints that have this **Link** (p. 192) as a parent **Link** (p. 192).*
- void **AddParentJoint** (JointPtr joint)

  *Joints that have this **Link** (p. 192) as a child **Link** (p. 192).*
- void **AttachStaticModel** (ModelPtr &_model, const **math::Pose** &_offset)

  Attach a static model to this link.
- void **DetachStaticModel** (const std::string &_modelName)

  Detach a static model from this link.
- void **DetachAllStaticModels** ()

  Detach all static models from this link.
- virtual void **OnPoseChange** ()

  This function is called when the entity's (or one of its parents) pose of the parent has changed.
- **LinkState GetState** ()

  Get the link state.
- void **SetState** (const **LinkState** &_state)

  Set the current link state.
- virtual void **UpdateMass** ()

  Update the mass matrix.
- virtual void **UpdateSurface** ()

  Update surface parameters.

Protected Attributes

- bool **isStatic**
- InertialPtr **inertial**
- std::vector< std::string > **cgVisuals**
- std::vector< std::string > **visuals**
- **math::Vector3 linearAccel**
- **math::Vector3 angularAccel**
- **math::Pose newPose**
- std::vector< **math::Pose** > **attachedModelsOffset**

Additional Inherited Members

9.88.1 Member Function Documentation

9.88.1.1 virtual void 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.

Implemented in **ODELink** (p. 249).

9.88.1.2 virtual void AddRelativeForce (const math::Vector3 & _force) [pure virtual]

Add a force to the body, components are relative to the body's own frame of reference.

Implemented in **ODELink** (p. 249).

9.88.1.3 virtual void AddRelativeTorque (const math::Vector3 & _torque) [pure virtual]

Add a torque to the body, components are relative to the body's own frame of reference.

Implemented in **ODELink** (p. 249).

9.88.1.4 void FillLinkMsg (msgs::Link & _msg)

Fill a link message.

Parameters

_msg	Message to fill

9.88.1.5 CollisionPtr GetCollisionById (unsigned int _id) const

Get a collision by id.

Returns

Pointer to the collision

9.88.1.6 bool GetSelfCollide ( )

Get Self-Collision Flag, if this is true, this body will collide with other bodies even if they share the same parent.

9.88.1.7 virtual void Load (sdf::ElementPtr _sdf) [virtual]

Load the body based on an SDF element.

Parameters

_sdf	SDF parameters

Reimplemented from **Entity** (p. 146).

Reimplemented in **ODELink** (p. 251).

9.88.1.8 void ProcessMsg (const msgs::Link & _msg)

Update parameters from a message.

Parameters

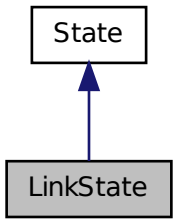
_msg	Message to read

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

- Link.hh

9.89 LinkState Class Reference

Inheritance diagram for LinkState:



Public Member Functions

- **LinkState** ()

  Default constructor.
- **LinkState** (const LinkPtr _link)

  Constructor.
- virtual ~**LinkState** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _elem)

  Load state from SDF element.
- **math::Pose GetPose** () const

  Get the link pose.
- unsigned int **GetCollisionStateCount** () const

  Get the number of link states.
- **CollisionState GetCollisionState** (unsigned int _index) const

  Get a link state.
- **CollisionState GetCollisionState** (const std::string &_collisionName) const

  Get a link state by link name.
- void **FillStateSDF** (sdf::ElementPtr _elem)

  *Fill a **State** (p. 346) SDF element with state info.*
- void **UpdateLinkSDF** (sdf::ElementPtr _elem)

  *Update a **Link** (p. 192) SDF element with this state info.*

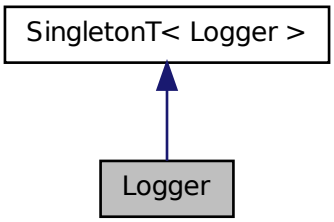
Additional Inherited Members

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

- LinkState.hh

9.90 Logger Class Reference

Inheritance diagram for Logger:



Public Member Functions

- **Logger** ()

  Constructor.
- virtual ~**Logger** ()

  Destructor.
- void **AddLog** (const std::string &_model, const std::string &_filename)

  Add a log file.
- void **RemoveLog** (const std::string &_entity)

  Remove a log.
- void **Update** ()

  Update the log files.

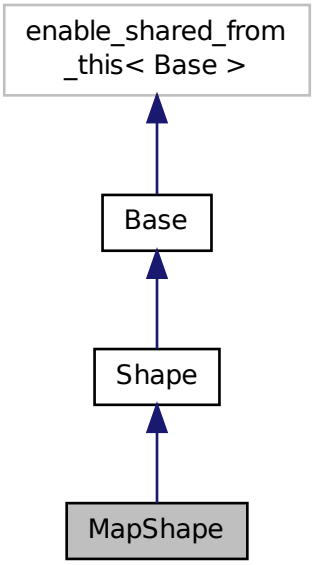
Additional Inherited Members

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

- Logger.hh

9.91 MapShape Class Reference

Inheritance diagram for MapShape:



Public Member Functions

- **MapShape** (CollisionPtr **parent**)

  Constructor.
- virtual ~**MapShape** ()

  Destructor.
- void **Update** ()

  Update function.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the map.
- virtual void **Init** ()

  Init the map.
- void **FillShapeMsg** (msgs::Geometry &_msg)
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
- std::string **GetFilename** () const
- double **GetScale** () const
- int **GetThreshold** () const
- double **GetHeight** () const
- int **GetGranularity** () const

Additional Inherited Members

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

- MapShape.hh

9.92 Master Class Reference

Public Member Functions

- **Master** ()
  Constructor.
- virtual ∼**Master** ()
  Destructor.
- void **Init** (uint16_t _port)
  Initialize.
- void **Run** ()
- void **RunThread** ()
  Run the master in a new thread.
- void **RunOnce** ()
  Run the master one iteration.
- void **Stop** ()
  Stop the master.
- void **Fini** ()
  Finalize the master.

9.92.1 Member Function Documentation

9.92.1.1 void Init (uint16_t *_port*)

Initialize.

Parameters

_port	The master's port

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

- Master.hh

9.93 Material Class Reference

Static Public Member Functions

- static void **CreateMaterials** ()
- static void **Update** (const **gazebo::common::Material** ∗_mat)

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

- rendering/Material.hh

9.94 Material Class Reference

Public Types

- enum **ShadeMode** {
  FLAT, **GOURAUD**, **PHONG**, **BLINN**,
  SHADE_COUNT }
- enum **BlendMode** { **ADD**, **MODULATE**, **REPLACE**, **BLEND_COUNT** }

Public Member Functions

- **Material** ()

  Constructor.
- virtual ~**Material** ()

  Destructor.
- **Material** (const **Color** &_clr)

  Create a material with a default color.
- std::string **GetName** () const

  Get the name of the material.
- void **SetTextureImage** (const std::string &_tex)

  Set a texture image.
- void **SetTextureImage** (const std::string &tex, const std::string &resource_path)

  Set a texture image.
- std::string **GetTextureImage** () const

  Get a thie texture image.
- void **SetAmbient** (const **Color** &_clr)

  Set the ambient color.
- **Color GetAmbient** () const

  Get the ambient color.
- void **SetDiffuse** (const **Color** &_clr)

  Set the diffuse color.
- **Color GetDiffuse** () const

  Get the diffuse color.
- void **SetSpecular** (const **Color** &_clr)

  Set the specular color.
- **Color GetSpecular** () const

  Get the specular color.
- void **SetEmissive** (const **Color** &_clr)

  Set the emissive color.
- **Color GetEmissive** () const

  Get the emissive color.
- void **SetTransparency** (double _t)

  Set the transparency percentage (0..1)
- double **SetTransparency** () const

Get the transparency percentage (0..1)

- void **SetShininess** (double _t)

  Set the shininess.
- double **GetShininess** () const

  Get the shininess.
- double **GetTransparency** () const

  Get the shininess.
- void **SetBlendFactors** (double _srcFactor, double _dstFactor)

  Set the blende factors.
- void **GetBlendFactors** (double &_srcFactor, double &_dstFactor)

  Get the blend factors.
- void **SetBlendMode** (BlendMode _b)

  Set the blending mode.
- BlendMode **GetBlendMode** () const

  Get the blending mode.
- void **SetShadeMode** (ShadeMode _b)

  Set the shading mode.
- ShadeMode **GetShadeMode** () const

  Get the shading mode.
- void **SetPointSize** (double _size)

  Set the point size.
- double **GetPointSize** () const

  Get the point size.
- void **SetDepthWrite** (bool _value)

  Set depth write.
- bool **GetDepthWrite** () const

  Get depth write.
- void **SetLighting** (bool _value)

  Set lighting enabled.
- bool **GetLighting** () const

  Get lighting enabled.

Static Public Attributes

- static std::string **ShadeModeStr** [SHADE_COUNT]
- static std::string **BlendModeStr** [BLEND_COUNT]

Protected Attributes

- std::string **name**
- std::string **texImage**
- **Color ambient**
- **Color diffuse**
- **Color specular**
- **Color emissive**
- double **transparency**
- double **shininess**
- double **pointSize**
- BlendMode **blendMode**
- ShadeMode **shadeMode**

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Material** &_m)

  Ostream operator.

9.94.1 Member Function Documentation

9.94.1.1 void SetBlendFactors (double *_srcFactor,* double *_dstFactor*)

Set the blende factors.

Will be interpreted as: (texture ∗ _srcFactor) + (scene_pixel ∗ _dstFactor)

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

- common/Material.hh

9.95 Material Interface Reference

Information about a material.

9.95.1 Detailed Description

Information about a material.

```
@verbatim
```

import "color.proto";

message **Material** (p. 203) { enum ShaderType { VERTEX = 1; PIXEL = 2; NORMAL_MAP_OBJECT_SPACE = 3; NORMAL_MAP_TANGENT_SPACE = 4; }

optional string script = 1; optional ShaderType shader_type = 2; optional string normal_map = 3; optional **Color** (p. 115) ambient = 4; optional **Color** (p. 115) diffuse = 5; optional **Color** (p. 115) specular = 6; optional **Color** (p. 115) emissive = 7; } ///

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

- material.proto

9.96 Matrix3 Class Reference

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.
- void **SetFromAxes** (const **Vector3** &_xAxis, const **Vector3** &_yAxis, const **Vector3** &_zAxis)

  Set the matrix from three axis.
- void **SetFromAxis** (const **Vector3** &_axis, double _angle)

  Set the matrix from an axis and angle.
- void **SetCol** (unsigned int _c, const **Vector3** &_v)

  Set a column.
- bool **operator==** (const **Matrix3** &_m) const

  Equality test operatoer.
- const double * **operator[]** (size_t _row) const
- double * **operator[]** (size_t _row)

Protected Attributes

- double **m** [3][3]

  the 3x3 matrix

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Matrix3** &_m)

  Output operator.

9.96.1 Constructor & Destructor Documentation

9.96.1.1 Matrix3 (const Matrix3 & *_m*)

Copy constructor.

Parameters

_m	Matrix to copy

9.96.1.2 Matrix3 (double *_v00*, double *_v01*, double *_v02*, double *_v10*, double *_v11*, double *_v12*, double *_v20*, double *_v21*, double *_v22*)

Constructor.

Parameters

_v00	Row 0, Col 0 value
_v01	Row 0, Col 1 value
_v02	Row 0, Col 2 value
_v10	Row 1, Col 0 value
_v11	Row 1, Col 1 value
_v12	Row 1, Col 2 value
_v20	Row 2, Col 0 value
_v21	Row 2, Col 1 value

_v22	Row 2, Col 2 value

9.96.2 Member Function Documentation

9.96.2.1 bool operator== (const Matrix3 & _m) const

Equality test operatoer.

Parameters

_m	**Matrix3** (p. 203) to test

Returns

True if equal

9.96.2.2 void SetCol (unsigned int _c, const Vector3 & _v)

Set a column.

Parameters

_c	The colum index (0, 1, 2)
_v	The value to set in each row of the column

9.96.2.3 void SetFromAxes (const Vector3 & _xAxis, const Vector3 & _yAxis, const Vector3 & _zAxis)

Set the matrix from three axis.

Parameters

_xAxis	The x axis
_yAxis	The y axis
_zAxis	The z axis

9.96.3 Friends And Related Function Documentation

9.96.3.1 std::ostream& operator<< (std::ostream & _out, const gazebo::math::Matrix3 & _m) [friend]

Output operator.

Parameters

_out	Output stream
_m	Matrix to output

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

- Matrix3.hh

9.97 Matrix4 Class Reference

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.
- 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)

  Set.
- void **SetTranslate** (const **Vector3** &_t)

  Set the translational values [(0, 3) (1, 3) (2, 3)].
- **Vector3 GetTranslation** () const

  *Get the translational values as a **Vector3** (*p. *373)**.*
- **Quaternion GetRotation** () const

  Get the rotation as a quaternion.
- **Vector3 GetEulerRotation** (unsigned int solution_number=1) const

  Get the rotation as a Euler angles.
- **math::Pose GetAsPose** () const

  *Get the transformation as **math::Pose** (*p. *284)**.*
- void **SetScale** (const **Vector3** &_s)

  Set the scale.
- bool **IsAffine** () const

  Return true if the matrix is affine.
- **Vector3 TransformAffine** (const **Vector3** &_v) const

  Perform an affine transformation.
- **Matrix4 Inverse** () const

  Return the inverse matrix.
- **Matrix4** & **operator=** (const **Matrix4** &_mat)

  Equal operator.
- const **Matrix4** & **operator=** (const **Matrix3** &_mat)

  Equal operator for 3x3 matrix.
- **Matrix4 operator**∗ (const **Matrix4** &_mat) const

  Multiplication operator.
- **Matrix4 operator**∗ (const **Matrix3** &_mat) const

  Multiplication operator.
- **Vector3 operator**∗ (const **Vector3** &_vec) const

  Multiplication operator.
- double ∗ **operator[]** (size_t _row)
- const double ∗ **operator[]** (size_t _row) const
- bool **operator==** (const **Matrix4** &_m) const

  Equality test operatoer.

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)

  Output operator.

9.97.1 Constructor & Destructor Documentation

9.97.1.1 Matrix4 (const Matrix4 & _m)

Copy constructor.

Parameters

_m	Matrix to copy

9.97.1.2 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

_v00	Row 0, Col 0 value
_v01	Row 0, Col 1 value
_v02	Row 0, Col 2 value
_v03	Row 0, Col 3 value
_v10	Row 1, Col 0 value
_v11	Row 1, Col 1 value
_v12	Row 1, Col 2 value
_v13	Row 1, Col 3 value
_v20	Row 2, Col 0 value
_v21	Row 2, Col 1 value
_v22	Row 2, Col 2 value
_v23	Row 2, Col 3 value
_v30	Row 3, Col 0 value
_v31	Row 3, Col 1 value
_v32	Row 3, Col 2 value
_v33	Row 3, Col 3 value

9.97.2 Member Function Documentation

9.97.2.1 bool IsAffine ( ) const

Return true if the matrix is affine.

Returns

True if the matrix is affine

9.97.2.2 Matrix4 operator∗ (const Matrix4 & *_mat*) const

Multiplication operator.

Parameters

_mat	Incoming matrix

Returns

This matrix ∗ _mat

9.97.2.3 Matrix4 operator∗ (const Matrix3 & *_mat*) const

Multiplication operator.

Parameters

_mat	Incoming matrix

Returns

This matrix ∗ _mat

9.97.2.4 Vector3 operator∗ (const Vector3 & *_vec*) const

Multiplication operator.

Parameters

_vec	**Vector3** (p. 373)

Returns

Resulting vector from multiplication

9.97.2.5 Matrix4& operator= (const Matrix4 & *_mat*)

Equal operator.

this = _mat

Parameters

_mat	Incoming matrix

Returns

The resulting matrix

9.97.2.6 const Matrix4& operator= (const Matrix3 & _mat)

Equal operator for 3x3 matrix.

Parameters

_mat	Incoming matrix

Returns

The resulting matrix

9.97.2.7 bool operator== (const Matrix4 & _m) const

Equality test operatoer.

Parameters

_m	**Matrix3** (p. 203) to test

Returns

True if equal

9.97.2.8 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)

Set.

Parameters

_v00	Row 0, Col 0 value
_v01	Row 0, Col 1 value
_v02	Row 0, Col 2 value
_v03	Row 0, Col 3 value
_v10	Row 1, Col 0 value
_v11	Row 1, Col 1 value
_v12	Row 1, Col 2 value
_v13	Row 1, Col 3 value
_v20	Row 2, Col 0 value
_v21	Row 2, Col 1 value

_v22	Row 2, Col 2 value
_v23	Row 2, Col 3 value
_v30	Row 3, Col 0 value
_v31	Row 3, Col 1 value
_v32	Row 3, Col 2 value
_v33	Row 3, Col 3 value

9.97.2.9 void SetScale (const Vector3 & *_s*)

Set the scale.

Parameters

_s	scale

9.97.2.10 void SetTranslate (const Vector3 & *_t*)

Set the translational values [(0, 3) (1, 3) (2, 3)].

Parameters

_t	Values to set

9.97.2.11 Vector3 TransformAffine (const Vector3 & *_v*) const

Perform an affine transformation.

Parameters

_v	**Vector3** (p. 373) value for the transformation

Returns

The result of the transformation

9.97.3 Friends And Related Function Documentation

9.97.3.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::math::Matrix4 & *_m*) [friend]

Output operator.

Parameters

_out	Output stream
_m	Matrix to output

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

- Matrix4.hh

9.98 Mesh Class Reference

Public Member Functions

- **Mesh** ()

  Constructor.
- virtual ∼**Mesh** ()

  Destructor.
- void **SetPath** (const std::string &_path)

  Set the path which contains the mesh resource.
- std::string **GetPath** () const

  Get the path which contains the mesh resource.
- void **SetName** (const std::string &_n)

  Set the name of this mesh.
- std::string **GetName** () const

  Get the name of this mesh.
- **math::Vector3 GetMax** () const

  Get the maximun X, Y, Z values.
- **math::Vector3 GetMin** () const

  Get the minimum X, Y, Z values.
- unsigned int **GetVertexCount** () const

  Return the number of vertices.
- unsigned int **GetNormalCount** () const

  Return the number of normals.
- unsigned int **GetIndexCount** () const

  Return the number of indicies.
- unsigned int **GetTexCoordCount** () const

  Return the number of texture coordinates.
- void **AddSubMesh** (**SubMesh** ∗_child)

  Add a submesh mesh.
- unsigned int **GetSubMeshCount** () const

  Get the number of children.
- unsigned int **AddMaterial** (**Material** ∗_mat)

  Add a material to the mesh.
- unsigned int **GetMaterialCount** () const

  Get the number of materials.
- const **Material** ∗ **GetMaterial** (int index) const

  Get a material.
- const **SubMesh** ∗ **GetSubMesh** (unsigned int i) const

  Get a child.
- void **FillArrays** (float ∗∗_vertArr, int ∗∗_indArr) const

  Put all the data into flat arrays.
- void **RecalculateNormals** ()

  Recalculate all the normals.
- void **GetAABB** (**math::Vector3** &_center, **math::Vector3** &_min_xyz, **math::Vector3** &_max_xyz) const

  Get AABB coordinate.
- void **GenSphericalTexCoord** (const **math::Vector3** &_center)

Generate texture coordinates using spherical projection from center.

- **Skeleton** * **GetSkeleton** () const

  Get the skeleton to which this mesh is attached.

- void **SetSkeleton** (**Skeleton** *_skel)

  Set the mesh skeleton.

- bool **HasSkeleton** () const

  Return true if mesh is attached to a skeleton.

9.98.1 Member Function Documentation

9.98.1.1 unsigned int AddMaterial (**Material** * *_mat*)

Add a material to the mesh.

Returns

Index of this material

9.98.1.2 **Skeleton*** GetSkeleton ( ) const

Get the skeleton to which this mesh is attached.

Returns

pointer to skeleton, or NULL if none is present.

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

- Mesh.hh

9.99 Mesh Interface Reference

Message for a mesh geometry.

9.99.1 Detailed Description

Message for a mesh geometry.

```
@verbatim
```

import "vector3d.proto";

message MeshGeom { required string filename = 1; optional **Vector3d** (p. 376) scale = 2; }

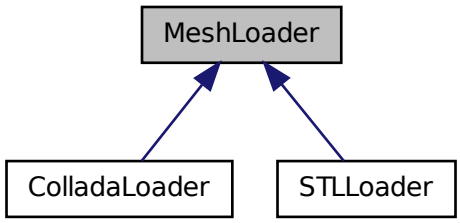
///

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

- meshgeom.proto

9.100 MeshLoader Class Reference

Inheritance diagram for MeshLoader:



Public Member Functions

- **MeshLoader** ()

  Constructor.
- virtual ~**MeshLoader** ()

  Destructor.
- virtual **Mesh** * **Load** (const std::string &filename)=0

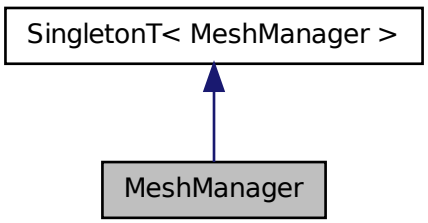
  Load a 3D mesh.

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

- MeshLoader.hh

9.101 MeshManager Class Reference

Inheritance diagram for MeshManager:



Public Member Functions

- const **Mesh** ∗ **Load** (const std::string &filename)

  Load a mesh from a file.

- bool **IsValidFilename** (const std::string &_filename)

  Return true if the file extension is loadable.

- void **GetMeshAABB** (const **Mesh** ∗mesh, **math::Vector3** &center, **math::Vector3** &min_xyz, **math::Vector3** &max_xyz)

  get mesh aabb and center

- void **GenSphericalTexCoord** (const **Mesh** ∗_mesh, **math::Vector3** _center)

  generate spherical texture coordinates

- void **AddMesh** (**Mesh** ∗_mesh)

  Add a mesh to the manager.

- const **Mesh** ∗ **GetMesh** (const std::string &name) const

  Get a mesh by name.

- bool **HasMesh** (const std::string &_name) const

  Return true if the mesh exists.

- void **CreateSphere** (const std::string &name, float radius, int rings, int segments)

  Create a sphere mesh.

- void **CreateBox** (const std::string &name, const **math::Vector3** &sides, const **math::Vector2d** &uvCoords)

  Create a Box mesh.

- void **CreateCylinder** (const std::string &name, float radius, float height, int rings, int segments)

  Create a cylinder mesh.

- void **CreateCone** (const std::string &name, float radius, float height, int rings, int segments)

  Create a cone mesh.

- void **CreateTube** (const std::string &name, float innerRadius, float outterRadius, float height, int rings, int segments)

  Create a tube mesh.

- void **CreatePlane** (const std::string &name, const **math::Plane** &plane, const **math::Vector2d** &segments, const **math::Vector2d** &uvTile)
- void **CreatePlane** (const std::string &name, const **math::Vector3** &normal, double d, const **math::Vector2d** &size, const **math::Vector2d** &segments, const **math::Vector2d** &uvTile)
- void **CreateCamera** (const std::string &_name, float _scale)

  Create a Camera mesh.

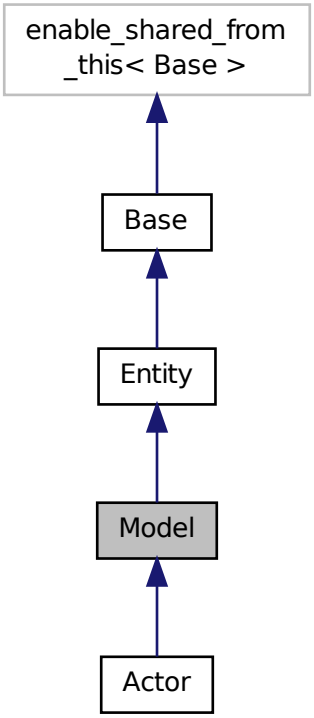
Additional Inherited Members

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

- MeshManager.hh

9.102 Model Class Reference

Inheritance diagram for Model:



Public Member Functions

- **Model** (BasePtr **parent**)

  Constructor.
- virtual ~**Model** ()

  Destructor.
- void **Load** (sdf::ElementPtr _sdf)

  Load the model.
- virtual void **Init** ()

  Initialize the model.
- void **Update** ()

  Update the model.
- virtual void **Fini** ()

  Finalize the model.
- virtual void **UpdateParameters** (sdf::ElementPtr _sdf)

  update the parameters using new sdf values

- virtual const sdf::ElementPtr **GetSDF** ()

  Get the SDF values for the model.
- virtual void **RemoveChild** (EntityPtr child)

  Remove a child.
- void **Reset** ()

  Reset the model.
- void **SetLinearVel** (const **math::Vector3** &vel)

  Set the linear velocity of the model.
- void **SetAngularVel** (const **math::Vector3** &vel)

  Set the angular velocity of the model.
- void **SetLinearAccel** (const **math::Vector3** &vel)

  Set the linear acceleration of the model.
- void **SetAngularAccel** (const **math::Vector3** &vel)

  Set the angular acceleration of the model.
- virtual **math::Vector3 GetRelativeLinearVel** () const

  Get the linear velocity of the entity.
- virtual **math::Vector3 GetWorldLinearVel** () const

  Get the linear velocity of the entity in the world frame.
- virtual **math::Vector3 GetRelativeAngularVel** () const

  Get the angular velocity of the entity.
- virtual **math::Vector3 GetWorldAngularVel** () const

  Get the angular velocity of the entity in the world frame.
- virtual **math::Vector3 GetRelativeLinearAccel** () const

  Get the linear acceleration of the entity.
- virtual **math::Vector3 GetWorldLinearAccel** () const

  Get the linear acceleration of the entity in the world frame.
- virtual **math::Vector3 GetRelativeAngularAccel** () const

  Get the angular acceleration of the entity.
- virtual **math::Vector3 GetWorldAngularAccel** () const

  Get the angular acceleration of the entity in the world frame.
- virtual **math::Box GetBoundingBox** () const

  Get the size of the bounding box.
- unsigned int **GetJointCount** () const

  Get the number of joints.
- JointPtr **GetJoint** (unsigned int index) const

  Get a joing by index.
- JointPtr **GetJoint** (const std::string &name)

  Get a joint.
- LinkPtr **GetLinkById** (unsigned int _id) const

  Get a link by id.
- 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.
- void **SetGravityMode** (const bool &v)

  Set the gravity mode of the model.
- void **SetCollideMode** (const std::string &m)

Set the collide mode of the model.

- void **SetLaserRetro** (const float &retro)

  Set the laser retro reflectiveness of the model.
- void **FillModelMsg** (msgs::Model &_msg)

  Fill a model message.
- void **ProcessMsg** (const msgs::Model &_msg)

  Update parameters from a model message.
- void **SetJointPositions** (const std::map< std::string, double > &_jointPositions)

  Set the positions of a set of joints.
- void **SetJointAnimation** (const std::map< std::string, common::NumericAnimationPtr > anim, boost::function< void()> _onComplete=NULL)
- virtual void **StopAnimation** ()

  Stop the current animations.
- void **AttachStaticModel** (ModelPtr &_model, **math::Pose** _offset)

  Attach a static model to this model.
- void **DetachStaticModel** (const std::string &_model)
- **ModelState GetState** ()

  Get the current model state.
- void **SetState** (const **ModelState** &_state)

  Set the current model state.
- void **SetEnabled** (bool _enabled)

  Enable all the links in all the models.

Protected Member Functions

- virtual void **OnPoseChange** ()

  This function is called when the entity's (or one of its parents) pose of the parent has changed.

Protected Attributes

- std::vector< ModelPtr > **attachedModels**
- std::vector< **math::Pose** > **attachedModelsOffset**

Additional Inherited Members

9.102.1 Constructor & Destructor Documentation

9.102.1.1 Model (BasePtr *parent*)

Constructor.

Parameters

parent	Parent object

9.102.2 Member Function Documentation

9.102.2.1 void FillModelMsg (msgs::Model & *_msg*)

Fill a model message.

Parameters

_msg	Message to fill

9.102.2.2 virtual math::Box GetBoundingBox ( ) const [virtual]

Get the size of the bounding box.

Returns

The bounding box

Reimplemented from **Entity** (p. 142).

9.102.2.3 JointPtr GetJoint (unsigned int *index*) const

Get a joing by index.

Parameters

index	Index of the joint

Returns

A pointer to the joint

9.102.2.4 JointPtr GetJoint (const std::string & *name*)

Get a joint.

Parameters

name	The name of the joint, specified in the world file

Returns

Pointer to the joint

9.102.2.5 unsigned int GetJointCount ( ) const

Get the number of joints.

Returns

Get the number of joints

9.102.2.6 LinkPtr GetLink (const std::string & *name* = "canonical") const

Get a link by name.

Returns

Pointer to the link

9.102.2.7 LinkPtr GetLink (unsigned int *_index*) const

Get a child link by index.

Returns

Point to the link

9.102.2.8 LinkPtr GetLinkById (unsigned int *_id*) const

Get a link by id.

Returns

Pointer to the link

9.102.2.9 virtual math::Vector3 GetRelativeAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity.

Returns

math::Vector3 (p. 373), set to 0, 0, 0 if the model has no body

Reimplemented from **Entity** (p. 144).

9.102.2.10 virtual math::Vector3 GetRelativeAngularVel ( ) const [virtual]

Get the angular velocity of the entity.

Returns

math::Vector3 (p. 373), set to 0, 0, 0 if the model has no body

Reimplemented from **Entity** (p. 144).

9.102.2.11 virtual math::Vector3 GetRelativeLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity.

Returns

math::Vector3 (p. 373), set to 0, 0, 0 if the model has no body

Reimplemented from **Entity** (p. 144).

9.102.2.12 virtual math::Vector3 GetRelativeLinearVel ( ) const [virtual]

Get the linear velocity of the entity.

Returns

math::Vector3 (p. 373), set to 0, 0, 0 if the model has no body

Reimplemented from **Entity** (p. 144).

9.102.2.13 virtual math::Vector3 GetWorldAngularAccel ( ) const [virtual]

Get the angular acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 373), set to 0, 0, 0 if the model has no body

Reimplemented from **Entity** (p. 145).

9.102.2.14 virtual math::Vector3 GetWorldAngularVel ( ) const [virtual]

Get the angular velocity of the entity in the world frame.

Returns

math::Vector3 (p. 373), set to 0, 0, 0 if the model has no body

Reimplemented from **Entity** (p. 145).

9.102.2.15 virtual math::Vector3 GetWorldLinearAccel ( ) const [virtual]

Get the linear acceleration of the entity in the world frame.

Returns

math::Vector3 (p. 373), set to 0, 0, 0 if the model has no body

Reimplemented from **Entity** (p. 145).

9.102.2.16 virtual math::Vector3 GetWorldLinearVel ( ) const [virtual]

Get the linear velocity of the entity in the world frame.

Returns

math::Vector3 (p. 373), set to 0, 0, 0 if the model has no body

Reimplemented from **Entity** (p. 145).

9.102.2.17 void Load (sdf::ElementPtr _sdf) [virtual]

Load the model.

Parameters

_sdf	SDF parameters

Reimplemented from **Entity** (p. 146).

9.102.2.18 virtual void RemoveChild (EntityPtr *child*) [virtual]

Remove a child.

Parameters

child	Remove a child entity

9.102.2.19 void SetAngularAccel (const math::Vector3 & *vel*)

Set the angular acceleration of the model.

Parameters

vel	The new angular acceleration

9.102.2.20 void SetAngularVel (const math::Vector3 & *vel*)

Set the angular velocity of the model.

Parameters

vel	The new angular velocity

9.102.2.21 void SetCollideMode (const std::string & *m*)

Set the collide mode of the model.

Parameters

m	The collision mode

9.102.2.22 void SetLaserRetro (const float & *retro*)

Set the laser retro reflectiveness of the model.

Parameters

retro	Retro reflectance value

9.102.2.23 void SetLinearAccel (const math::Vector3 & *vel*)

Set the linear acceleration of the model.

Parameters

vel	The new linear acceleration

9.102.2.24 void SetLinearVel (const math::Vector3 & *vel*)

Set the linear velocity of the model.

Parameters

vel	The new linear velocity

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

- Model.hh

9.103 Model Interface Reference

Information about a model.

9.103.1 Detailed Description

Information about a model.

```
@verbatim
```

import "joint.proto"; import "link.proto"; import "pose.proto"; import "visual.proto";

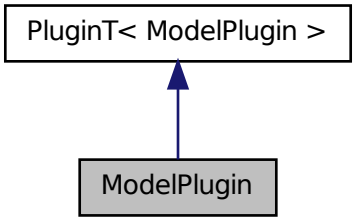
message **Model** (p. 222) { required string name = 1; optional uint32 id = 2; optional bool is_static = 3; optional **Pose** (p. 283) pose = 4; repeated **Joint** (p. 181) joint = 5; repeated **Link** (p. 191) link = 6; optional bool deleted = 7; repeated **Visual** (p. 384) visual = 8; } ///

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

- model.proto

9.104 ModelPlugin Class Reference

Inheritance diagram for ModelPlugin:



Public Member Functions

- virtual void **Load** (physics::ModelPtr _model, sdf::ElementPtr _sdf)=0
  Load function.
- virtual void **Init** ()
- virtual void **Reset** ()

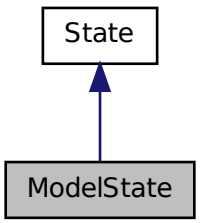
Additional Inherited Members

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

- common/Plugin.hh

9.105 ModelState Class Reference

Inheritance diagram for ModelState:



Public Member Functions

- **ModelState** ()

  Default constructor.
- **ModelState** (ModelPtr _model)

  Constructor.
- virtual ~**ModelState** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _elem)

  Load state from SDF element.
- **math::Pose GetPose** () const

  Get the model pose.
- unsigned int **GetLinkStateCount** () const

  Get the number of link states.
- **LinkState GetLinkState** (unsigned int _index) const

  Get a link state.
- **LinkState GetLinkState** (const std::string &_modelName) const

  Get a link state by model name.
- unsigned int **GetJointStateCount** () const

  Get the number of joint states.
- **JointState GetJointState** (unsigned int _index) const

  Get a model state.
- **JointState GetJointState** (const std::string &_jointName) const

  Get a model state by model name.
- void **FillStateSDF** (sdf::ElementPtr _elem)

  *Fill a **State** (p. 346) SDF element with state info.*
- void **UpdateModelSDF** (sdf::ElementPtr _elem)

  *Update a **Model** (p. 215) SDF element with this state info.*

Additional Inherited Members

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

- ModelState.hh

9.106 MouseEvent Class Reference

Public Types

- enum **Buttons** { **NO_BUTTON** = 0x0, **LEFT** = 0x1, **MIDDLE** = 0x2, **RIGHT** = 0x4 }
- enum **EventType** {
  NO_EVENT, **MOVE**, **PRESS**, **RELEASE**,
  SCROLL }

Public Attributes

- **math::Vector2i pos**
- **math::Vector2i prevPos**
- **math::Vector2i pressPos**
- **math::Vector2i scroll**
- float **moveScale**
- bool **dragging**
- EventType **type**
- unsigned int **button**
- unsigned int **buttons**

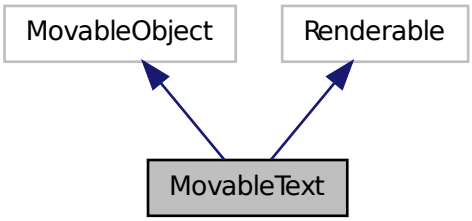
  State of the buttons when the event was generated.
- bool **shift**
- bool **control**

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

- MouseEvent.hh

9.107 MovableText Class Reference

Inheritance diagram for 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.
- 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 **SetFontName** (const std::string &font)

  Set the font.
- const std::string & **GetFont** () const

  Get the font.
- void **SetText** (const std::string &caption)

  Set the text to display.
- const std::string & **GetText** () const

  Get the displayed text.
- void **SetColor** (const **common::Color** &_color)

  Set the text color.
- const **common::Color** & **GetColor** () const

  Get the text color.
- void **SetCharHeight** (float height)

  Set the height of a character.
- float **GetCharHeight** () const

  Set the height of a characters.
- void **SetSpaceWidth** (float width)

  Set the width of a space.
- float **GetSpaceWidth** () const

  Get the width of a space.
- void **SetTextAlignment** (const **HorizAlign** &hAlign, const **VertAlign** &vAlign)

  Set the alignment of the text.
- void **SetBaseline** (float height)

  Set the baseline height of the text.
- float **GetBaseline** () const

  Get the baseline height.
- void **SetShowOnTop** (bool show)

  True = text always is displayed ontop.
- bool **GetShowOnTop** () const

  True = text is displayed on top.
- **math::Box GetAABB** ()

  Get the axis aligned bounding box of the text.
- void **Update** ()

  Update the text.
- virtual void **visitRenderables** (Ogre::Renderable::Visitor *visitor, bool debug=false)

  Method to allow a caller to abstractly iterate over the.

Protected Member Functions

- void **_setupGeometry** ()

  *setup the geometry (from **MovableText** (p. 225))*

- void **_updateColors** ()

  update the color(from MovableText)

- void **getWorldTransforms** (Ogre::Matrix4 *xform) const

  Get the world transform (from MovableObject)

- float **getBoundingRadius** () const

  Get the bounding radiu (from MovableObject)

- float **getSquaredViewDepth** (const Ogre::Camera *cam) const

  Get the squared view depth (from MovableObject)

- void **getRenderOperation** (Ogre::RenderOperation &op)

  Get the render operation.

- const Ogre::MaterialPtr & **getMaterial** (void) const

  Get the material.

- const Ogre::LightList & **getLights** (void) const

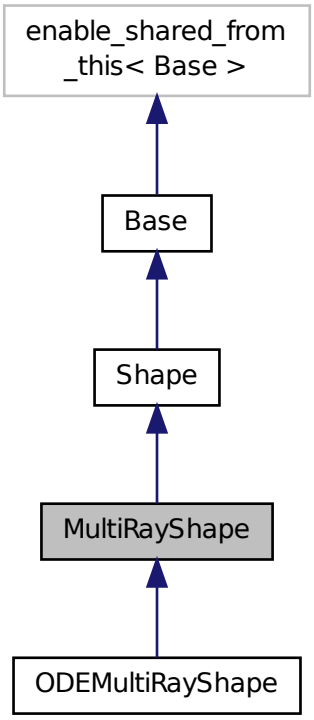
  Get the lights.

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

- MovableText.hh

9.108 MultiRayShape Class Reference

Inheritance diagram for MultiRayShape:



Public Member Functions

- **MultiRayShape** (CollisionPtr **parent**)

  Constructor.
- virtual ~**MultiRayShape** ()

  Destructor.
- virtual void **Init** ()

  Init the shape.
- void **SetDisplayType** (const std::string &type)
- double **GetRange** (int index)

  Get detected range for a ray.
- double **GetRetro** (int index)

  Get detected retro (intensity) value for a ray.
- int **GetFiducial** (int index)

  Get detected fiducial value for a ray.
- double **GetMinRange** () const

Get the minimum range.

- double **GetMaxRange** () const

  Get the maximum range.
- double **GetResRange** () const

  Get the range resolution.
- int **GetSampleCount** () const

  Get the horizontal sample count.
- double **GetScanResolution** () const

  Get the range resolution.
- **math::Angle GetMinAngle** () const

  Get the minimum angle.
- **math::Angle GetMaxAngle** () const

  Get the maximum angle.
- int **GetVerticalSampleCount** () const

  Get the vertical sample count.
- double **GetVerticalScanResolution** () const

  Get the vertical range resolution.
- **math::Angle GetVerticalMinAngle** () const

  Get the vertical min angle.
- **math::Angle GetVerticalMaxAngle** () const

  Get the vertical max angle.
- void **Update** ()

  Update the collision.
- void **FillShapeMsg** (msgs::Geometry &)
- virtual void **ProcessMsg** (const msgs::Geometry &)
- template<typename T >
  event::ConnectionPtr **ConnectNewLaserScans** (T subscriber)

  Connect a to the add entity signal.
- void **DisconnectNewLaserScans** (event::ConnectionPtr &c)

Protected Member Functions

- virtual void **UpdateRays** ()=0

  Physics engine specific method for updating the rays.
- virtual void **AddRay** (const **math::Vector3** &start, const **math::Vector3** &end)

  Add a ray to the collision.

Protected Attributes

- std::vector< RayShapePtr > **rays**

  Ray data.
- **math::Vector3 offset**
- sdf::ElementPtr **rayElem**
- sdf::ElementPtr **scanElem**
- sdf::ElementPtr **horzElem**
- sdf::ElementPtr **vertElem**
- sdf::ElementPtr **rangeElem**
- **event::EventT**< void()> **newLaserScans**

Additional Inherited Members

9.108.1 Member Function Documentation

9.108.1.1 double GetRange (int *index*)

Get detected range for a ray.

Returns

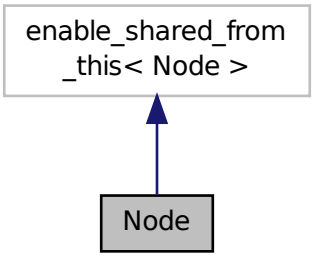
Returns DBL_MAX for no detection.

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

- MultiRayShape.hh

9.109 Node Class Reference

Inheritance diagram for Node:



Public Member Functions

- **Node** ()
  Constructor.
- virtual ~**Node** ()
  Destructor.
- void **Init** (const std::string &_space="")
  Init the node.
- void **Fini** ()
- std::string **GetTopicNamespace** () const
  Get the topic namespace for this node.
- std::string **DecodeTopicName** (const std::string &topic)
  Decode a topic name.

- std::string **EncodeTopicName** (const std::string &topic)

  Encode a topic name.
- unsigned int **GetId** () const

  Get the unique ID of the node.
- void **ProcessPublishers** ()

  Process all publishers, which has each publisher send it's most recent message over the wire.
- void **ProcessIncoming** ()
- template<typename M >
  transport::PublisherPtr **Advertise** (const std::string &topic, unsigned int _queueLimit=10, bool _latch=false)

  Adverise a topic.
- template<typename M , typename T >
  SubscriberPtr **Subscribe** (const std::string &topic, void(T::∗fp)(const boost::shared_ptr< M const > &), T ∗obj, bool _latching=false)

  Subscribe *(*p. *350) to a topic, and return data on the callback.*
- template<typename M >
  SubscriberPtr **Subscribe** (const std::string &topic, void(∗fp)(const boost::shared_ptr< M const > &), bool _latching=false)

  Subscribe *(*p. *350) to a topic, and return data on the callback.*
- bool **HandleData** (const std::string &_topic, const std::string &_msg)
- std::string **GetMsgType** (const std::string &_topic) const

  Get the message type for a topic.

9.109.1 Member Function Documentation

9.109.1.1 std::string GetTopicNamespace ( ) const

Get the topic namespace for this node.

Returns

The namespace

9.109.1.2 void 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. 200)

9.109.1.3 void ProcessPublishers ( )

Process all publishers, which has each publisher send it's most recent message over the wire.

This is for internal use only

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

- Node.hh

9.110 NodeAnimation Class Reference

Public Member Functions

- **NodeAnimation** (const std::string &_name)
- void **SetName** (const std::string &_name)
- std::string **GetName** () const
- void **AddKeyFrame** (const double _time, const **math::Matrix4** _trans)
- void **AddKeyFrame** (const double _time, const **math::Pose** _pose)
- unsigned int **GetFrameCount** () const
- void **GetKeyFrame** (const unsigned int _i, double &_time, **math::Matrix4** &_trans) const
- std::pair< double, **math::Matrix4** > **GetKeyFrame** (const unsigned int _i) const
- double **GetLength** () const
- **math::Matrix4 GetFrameAt** (double _time, bool _loop=true) const
- void **Scale** (const double _scale)
- double **GetTimeAtX** (const double _x) const

Protected Attributes

- std::string **name**
- std::map< double, **math::Matrix4** > **keyFrames**
- double **length**

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

- SkeletonAnimation.hh

9.111 NodeAssignment Struct Reference

Public Attributes

- unsigned int **vertexIndex**
- unsigned int **nodeIndex**
- float **weight**

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

- Mesh.hh

9.112 NodeTransform Class Reference

Public Types

- enum **TransformType** { **TRANSLATE**, **ROTATE**, **SCALE**, **MATRIX** }

Public Member Functions

- **NodeTransform** (TransformType _type=MATRIX)
- **NodeTransform** (**math::Matrix4** _mat, std::string _sid="_default_", TransformType _type=MATRIX)
- void **Set** (**math::Matrix4** _mat)
- void **SetType** (TransformType _type)
- void **SetSID** (std::string _sid)
- **math::Matrix4 Get** ()
- TransformType **GetType** ()
- std::string **GetSID** ()
- void **SetComponent** (unsigned int _idx, double _value)
- void **SetSourceValues** (**math::Matrix4** _mat)
- void **SetSourceValues** (**math::Vector3** _vec)
- void **SetSourceValues** (**math::Vector3** _axis, double _angle)
- void **RecalculateMatrix** ()
- void **PrintSource** ()
- **math::Matrix4 operator()** ()
- **math::Matrix4 operator*** (**NodeTransform** _t)
- **math::Matrix4 operator*** (**math::Matrix4** _m)

Protected Attributes

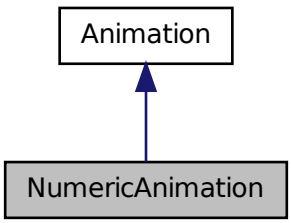
- std::string **sid**
- TransformType **type**
- **math::Matrix4 transform**
- std::vector< double > **source**

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

- Skeleton.hh

9.113 NumericAnimation Class Reference

Inheritance diagram for NumericAnimation:



Public Member Functions

- **NumericAnimation** (const std::string &_name, double _length, bool _loop)
- **NumericKeyFrame** * **CreateKeyFrame** (double _time)
- void **GetInterpolatedKeyFrame** (**NumericKeyFrame** &_kf) const

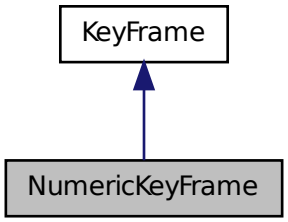
Additional Inherited Members

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

- Animation.hh

9.114 NumericKeyFrame Class Reference

Inheritance diagram for NumericKeyFrame:



Public Member Functions

- **NumericKeyFrame** (double _time)
- void **SetValue** (const double &_value)
- const double & **GetValue** () const

Protected Attributes

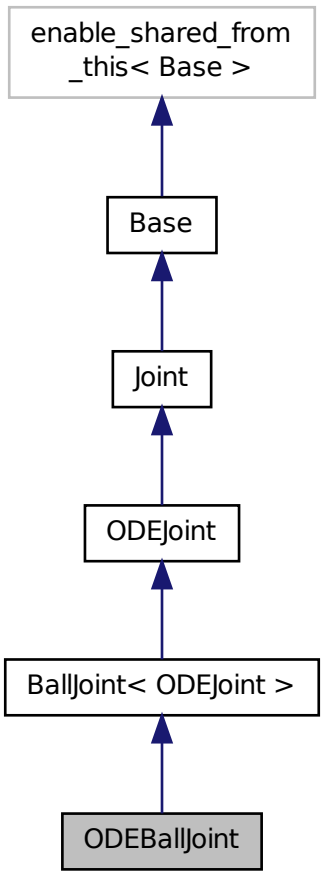
- double **value**

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

- KeyFrame.hh

9.115 ODEBallJoint Class Reference

Inheritance diagram for ODEBallJoint:



Public Member Functions

- **ODEBallJoint** (dWorldID worldId)

  Constructor.
- virtual ~**ODEBallJoint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (int index) const

  Get joint's anchor point.
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)

  Set joint's anchor point.
- virtual **math::Vector3 GetGlobalAxis** (int) const

  Get the axis of rotation.

- virtual void **SetDamping** (int index, const double damping)

  Set joint damping, not yet implemented.

- virtual void **SetVelocity** (int, double)

  Set the velocity of an axis(index).

- virtual double **GetVelocity** (int) const

  Get the rotation rate of an axis(index)

- virtual double **GetMaxForce** (int)

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

- virtual void **SetMaxForce** (int, double)

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

- virtual **math::Angle GetAngleImpl** (int) const

  Get the angle of rotation of an axis(index)

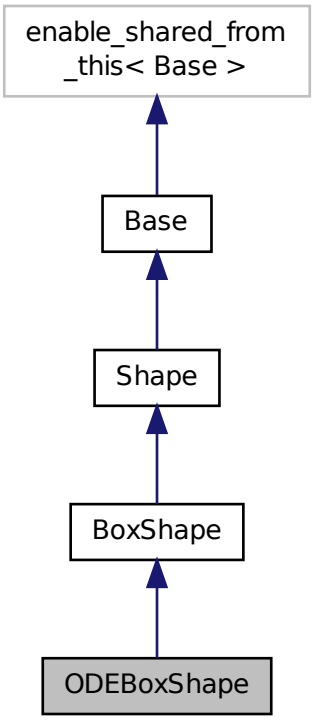
Additional Inherited Members

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

- ODEBallJoint.hh

9.116 ODEBoxShape Class Reference

Inheritance diagram for ODEBoxShape:



Public Member Functions

- **ODEBoxShape** (ODECollisionPtr _parent)
- virtual void **SetSize** (const **math::Vector3** &size)

  Set the size of the box.

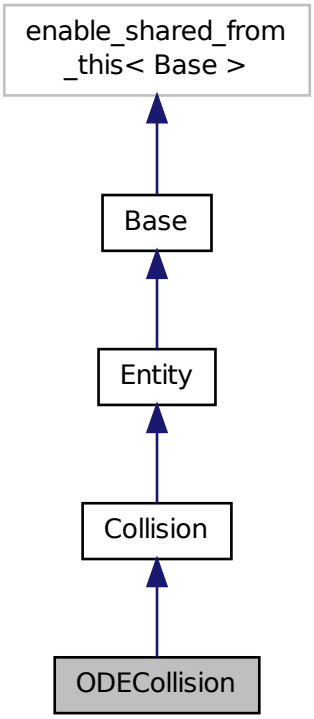
Additional Inherited Members

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

- ODEBoxShape.hh

9.117 ODECollision Class Reference

Inheritance diagram for ODECollision:



Public Member Functions

- **ODECollision** (LinkPtr **link**)

  Constructor.
- virtual ~**ODECollision** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the collision.
- void **Fini** ()

  Finalize the collision.
- void **SetCollision** (dGeomID **collisionId**, bool placeable)

  Set the encapsulated geometry object.
- dGeomID **GetCollisionId** () const

  Return the collision id.
- int **GetCollisionClass** () const

  Get the ODE collision class.

- 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.
- virtual **math::Box GetBoundingBox** () const

  Get the bounding box, defined by the physics engine.
- dSpaceID **GetSpaceId** () const

  Get the collision's space ID.
- void **SetSpaceId** (dSpaceID spaceid)

  Set the collision's space ID.

Protected Attributes

- dSpaceID **spaceId**
- dGeomID **collisionId**

  ID for the sub-collision.

Additional Inherited Members

9.117.1 Member Function Documentation

9.117.1.1 dGeomID GetCollisionId ( ) const

Return the collision id.

Returns

The collision id

9.117.1.2 virtual void SetCategoryBits (unsigned int *bits*) [virtual]

Set the category bits, used during collision detection.

Parameters

bits	The bits

Implements **Collision** (p. 112).

9.117.1.3 virtual void SetCollideBits (unsigned int *bits*) [virtual]

Set the collide bits, used during collision detection.

Parameters

bits	The bits

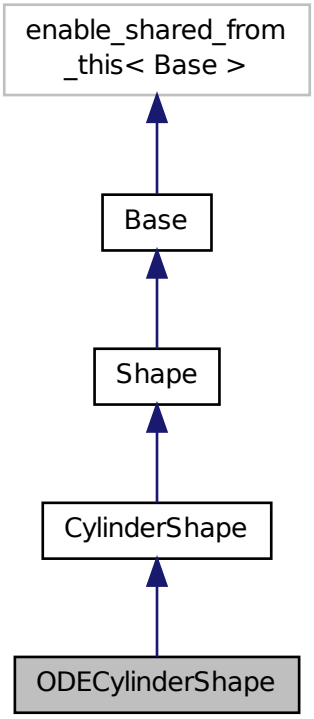
Implements **Collision** (p. 113).

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

- ODECollision.hh

9.118 ODECylinderShape Class Reference

Inheritance diagram for ODECylinderShape:



Public Member Functions

- **ODECylinderShape** (CollisionPtr _parent)
- void **SetSize** (const double &_radius, const double &_length)

  Set the size of the cylinder.

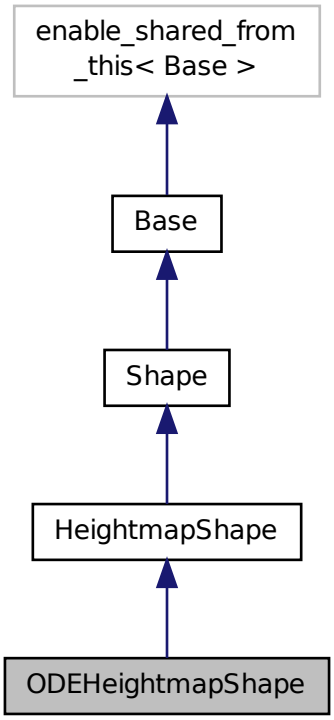
Additional Inherited Members

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

- ODECylinderShape.hh

9.119 ODEHeightmapShape Class Reference

Inheritance diagram for ODEHeightmapShape:



Public Member Functions

- **ODEHeightmapShape** (CollisionPtr _parent)

  Constructor.
- virtual ~**ODEHeightmapShape** ()

  Destructor.
- virtual void **Init** ()

  Load the heightmap.

Additional Inherited Members

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

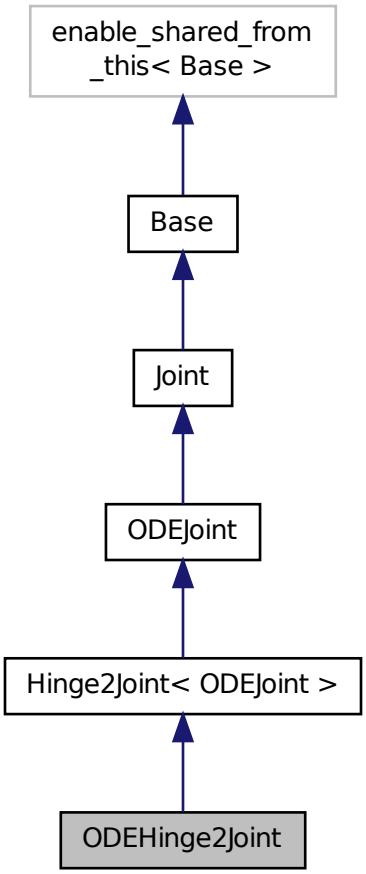
- ODEHeightmapShape.hh

9.120 ODEHinge2Joint Class Reference

A two axis hinge joint.

`#include <ODEHinge2Joint.hh>`

Inheritance diagram for ODEHinge2Joint:



Public Member Functions

- **ODEHinge2Joint** (dWorldID worldId)

  Constructor.
- virtual ~**ODEHinge2Joint** ()

  Destructor.
- 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, const double damping)

  Set joint damping, not yet implemented.

- virtual **math::Vector3 GetAnchor** (int index) const

  Get anchor point.

- virtual **math::Vector3 GetGlobalAxis** (int index) const

  Get first axis of rotation.

- virtual **math::Angle GetAngleImpl** (int index) const

  Get angle of rotation about first axis.

- virtual double **GetVelocity** (int index) const

  Get rate of rotation about first axis.

- virtual void **SetVelocity** (int index, double angle)

  Set the velocity of an axis(index).

- virtual void **SetMaxForce** (int index, double t)

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

- virtual double **GetMaxForce** (int index)

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

- virtual void **SetForce** (int index, double torque)

  Set the torque.

- virtual double **GetParam** (int parameter) const

  Get the specified parameter.

- virtual void **SetParam** (int parameter, double value)

  Set _parameter with _value.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the joint.

Additional Inherited Members

9.120.1 Detailed Description

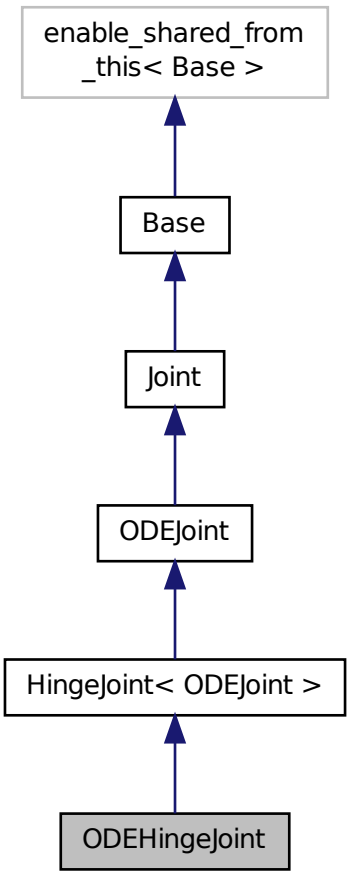
A two axis hinge joint.

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

- ODEHinge2Joint.hh

9.121 ODEHingeJoint Class Reference

Inheritance diagram for ODEHingeJoint:



Public Member Functions

- **ODEHingeJoint** (dWorldID worldId)

  Constructor.
- virtual ~**ODEHingeJoint** ()

  Destructor.
- virtual **math::Vector3 GetAnchor** (int index) const

  Get the anchor point.
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)

  Set the anchor point.
- virtual **math::Vector3 GetGlobalAxis** (int index) const

  Get the axis of rotation.

- virtual void **SetAxis** (int index, const **math::Vector3** &axis)

  Set the axis of rotation.

- virtual void **SetDamping** (int index, const double damping)

  Set the joint damping.

- void **ApplyDamping** ()

  callback to apply damping force to joint

- virtual **math::Angle GetAngleImpl** (int index) const

  Get the angle of rotation.

- virtual void **SetVelocity** (int index, double angle)

  Set the velocity of an axis(index).

- virtual double **GetVelocity** (int index) const

  Get the rotation rate of an axis(index)

- virtual void **SetMaxForce** (int index, double t)

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

- virtual double **GetMaxForce** (int index)

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

- virtual void **SetForce** (int index, double torque)

  Set the torque of a joint.

- virtual double **GetParam** (int parameter) const

  Get the specified parameter.

- virtual void **SetParam** (int parameter, double value)

  Set the parameter to value.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load joint.

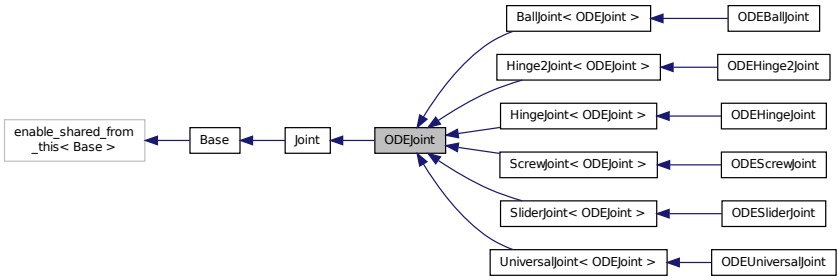
Additional Inherited Members

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

- ODEHingeJoint.hh

9.122 ODEJoint Class Reference

Inheritance diagram for ODEJoint:



Public Member Functions

- **ODEJoint** ()

  Constructor.
- virtual ∼**ODEJoint** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load a joint.
- virtual void **Reset** ()

  Reset the joint.
- virtual LinkPtr **GetJointLink** (int _index) const

  Get the link to which the joint is attached according the _index.
- virtual bool **AreConnected** (LinkPtr _one, LinkPtr _two) const

  Determines of the two bodies are connected by a joint.
- virtual double **GetParam** (int _parameter) const

  The default function does nothing.
- virtual void **Attach** (LinkPtr **parent**, LinkPtr child)

  Attach the two bodies with this joint.
- virtual void **Detach** ()

  Detach this joint from all bodies.
- virtual void **SetParam** (int _parameter, double _value)

  By default this does nothing.
- void **SetERP** (double newERP)

  Set the ERP of this joint.
- double **GetERP** ()

  Get the ERP of this joint.
- void **SetCFM** (double newCFM)

  Set the CFM of this joint.
- double **GetCFM** ()

  Get the ERP of this joint.
- dJointFeedback ∗ **GetFeedback** ()

Get the feedback data structure for this joint, if set.

- virtual void **SetHighStop** (int index, **math::Angle** angle)

  Set the high stop of an axis(index).
- virtual void **SetLowStop** (int index, **math::Angle** angle)

  Set the low stop of an axis(index).
- 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 **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 void **SetAttribute** (**Attribute**, int index, double value)

  Set a parameter for the joint.

Protected Attributes

- dJointID **jointId**

  This is our id.

Additional Inherited Members

9.122.1 Member Function Documentation

9.122.1.1 virtual math::Vector3 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 **Joint** (p. 184).

9.122.1.2 virtual math::Vector3 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 **Joint** (p. 185).

9.122.1.3 virtual double GetParam (int *_parameter*) const [virtual]

The default function does nothing.

This should be overriden in the child classes where appropriate

Reimplemented in **ODEHingeJoint** (p. 245), **ODEHinge2Joint** (p. 243), **ODEScrewJoint** (p. 259), and **ODESliderJoint** (p. 261).

9.122.1.4 virtual void SetParam (int *_parameter,* double *_value*) [virtual]

By default this does nothing.

It should be overridden in child classes where appropriate

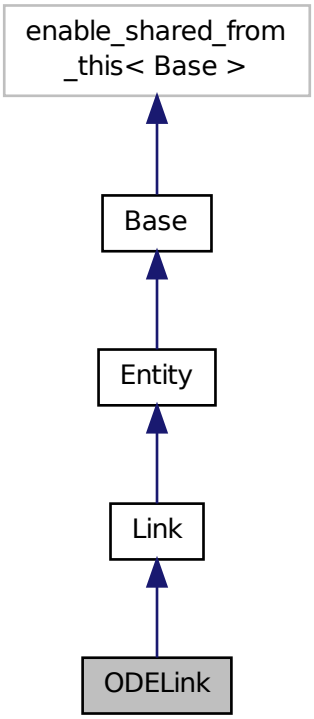
Reimplemented in **ODEHingeJoint** (p. 245), **ODEHinge2Joint** (p. 243), **ODEScrewJoint** (p. 259), **ODESliderJoint** (p. 261), and **ODEUniversalJoint** (p. 266).

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

- ODEJoint.hh

9.123 ODELink Class Reference

Inheritance diagram for ODELink:



Public Member Functions

- **ODELink** (EntityPtr _parent)

  Constructor.
- virtual ~**ODELink** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the link based on SDF parameters.
- virtual void **Init** ()

  Initialize the link.
- virtual void **Fini** ()

  Finalize the link.
- virtual void **Update** ()

  Update the link.
- virtual void **OnPoseChange** ()

  Called when the pose of the entity (or one of its parents) has changed.
- dBodyID **GetODEId** () const

  Return the ID of this link.
- virtual void **SetEnabled** (bool enable) const

  Set whether this link is enabled.
- virtual bool **GetEnabled** () const

  Get whether this link is enabled in the physics engine.
- virtual void **UpdateMass** ()

  Update the mass matrix.
- virtual void **UpdateSurface** ()

  Update other parameters for ODE.
- virtual void **SetLinearVel** (const **math::Vector3** &vel)

  Set the linear velocity of the link.
- virtual void **SetAngularVel** (const **math::Vector3** &vel)

  Set the angular velocity of the link.
- virtual void **SetForce** (const **math::Vector3** &_force)

  Set the force applied to the link.
- virtual void **SetTorque** (const **math::Vector3** &_torque)

  Set the torque applied to the link.
- virtual void **AddForce** (const **math::Vector3** &_force)

  Add a force to the body.
- virtual void **AddRelativeForce** (const **math::Vector3** &_force)

  Add a force to the body.
- virtual void **AddForceAtWorldPosition** (const **math::Vector3** &_force, const **math::Vector3** &_pos)

  Add a force to the body using a global position.
- virtual void **AddForceAtRelativePosition** (const **math::Vector3** &_force, const **math::Vector3** &_relpos)

  Set the force applied to the body (add by Stefano)
- virtual void **AddTorque** (const **math::Vector3** &_torque)

  Add a torque to the body.
- virtual void **AddRelativeTorque** (const **math::Vector3** &_torque)

  Add a torque to the body relative frame.
- virtual **math::Vector3 GetWorldLinearVel** () const

Get the linear velocity of the link in the world frame.

- 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 GetWorldTorque** () const

  Get the torque applied to the link in the world frame.

- virtual void **SetGravityMode** (bool mode)

  Set whether gravity affects this link.

- virtual bool **GetGravityMode** ()

  Get the gravity mode.

- void **SetSelfCollide** (bool collide)

  Set whether this link will collide with others in the model.

- dSpaceID **GetSpaceId** () const

  Get the link's space ID.

- void **SetSpaceId** (dSpaceID spaceid)

  Set the link's space ID.

- virtual void **SetLinearDamping** (double damping)

  Set the linear damping factor.

- virtual void **SetAngularDamping** (double damping)

  Set the angular damping factor.

- virtual void **SetKinematic** (const bool &state)

  Set whether this link is in the kinematic state.

- virtual bool **GetKinematic** () const

  Get whether this link is in the kinematic state.

Static Public Member Functions

- static void **DisabledCallback** (dBodyID _id)
- static void **MoveCallback** (dBodyID id)

Protected Attributes

- **math::Pose pose**

Additional Inherited Members

9.123.1 Member Function Documentation

9.123.1.1 dBodyID GetODEId ( ) const

Return the ID of this link.

Returns

ODE link id

9.123.1.2 virtual void Load (sdf::ElementPtr *_sdf*) [virtual]

Load the link based on SDF parameters.

Parameters

_sdf	the sdf parameters

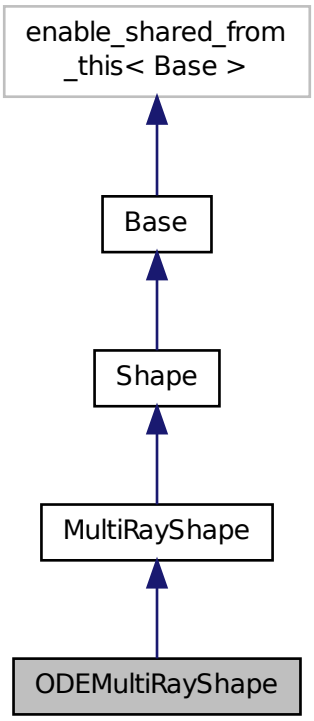
Reimplemented from **Link** (p. 196).

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

- ODELink.hh

9.124 ODEMultiRayShape Class Reference

Inheritance diagram for 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.

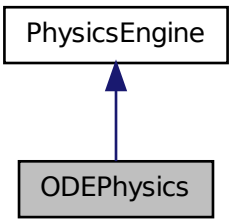
Additional Inherited Members

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

- ODEMultiRayShape.hh

9.125 ODEPhysics Class Reference

Inheritance diagram for ODEPhysics:



Public Member Functions

- **ODEPhysics** (WorldPtr world)

  Constructor.

- virtual ~**ODEPhysics** ()

  Destructor.

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the ODE engine.

- virtual void **Init** ()

  Initialize the ODE engine.

- void **Reset** ()

- virtual void **InitForThread** ()

  Init the engine for threads.
- virtual void **UpdateCollision** ()

  Update the ODE collision.
- virtual void **UpdatePhysics** ()

  Update the ODE engine.
- virtual void **Fini** ()

  Finilize the ODE engine.
- void **SetStepTime** (double _value)

  Set the step time.
- virtual double **GetStepTime** ()

  Get the simulation step time.
- virtual LinkPtr **CreateLink** (ModelPtr _parent)

  Create a new body.
- virtual CollisionPtr **CreateCollision** (const std::string &_shapeType, LinkPtr _parent)

  Create a collision.
- virtual ShapePtr **CreateShape** (const std::string &_shapeType, CollisionPtr _collision)
- virtual JointPtr **CreateJoint** (const std::string &type)

  Create a new joint.
- dSpaceID **GetSpaceId** () const

  Return the space id.
- dWorldID **GetWorldId** ()

  Get the world id.
- virtual std::string **GetStepType** () const

  Get the step type.
- virtual void **SetStepType** (const std::string &_type)

  Set the step type.
- virtual void **SetGravity** (const **gazebo::math::Vector3** &gravity)

  Set the gavity vector.
- **math::Vector3 GetGravity** () const

  Get gravity vector.
- void **SetWorldCFM** (double cfm)

  access functions to set ODE parameters
- void **SetWorldERP** (double erp)

  access functions to set ODE parameters
- void **SetSORPGSPreconIters** (unsigned int iters)

  access functions to set ODE parameters
- void **SetSORPGSIters** (unsigned int iters)

  access functions to set ODE parameters
- void **SetSORPGSW** (double w)

  access functions to set ODE parameters
- void **SetContactMaxCorrectingVel** (double vel)

  access functions to set ODE parameters
- void **SetContactSurfaceLayer** (double layer_depth)

  access functions to set ODE parameters
- void **SetMaxContacts** (unsigned int max_contacts)

  access functions to set ODE parameters

- double **GetWorldCFM** ()

  access functions to set ODE parameters
- double **GetWorldERP** ()

  access functions to set ODE parameters
- int **GetSORPGSPreconIters** ()

  access functions to set ODE parameters
- int **GetSORPGSIters** ()

  access functions to set ODE parameters
- double **GetSORPGSW** ()

  access functions to set ODE parameters
- double **GetContactMaxCorrectingVel** ()

  access functions to set ODE parameters
- double **GetContactSurfaceLayer** ()

  access functions to set ODE parameters
- int **GetMaxContacts** ()

  access functions to set ODE parameters
- void **CreateContact** (**ODECollision** *collision1, **ODECollision** *collision2)
- void **Collide** (**ODECollision** *collision1, **ODECollision** *collision2, dContactGeom *contactCollisions)

  Collide two collisions.
- void **ProcessContactFeedback** (**ContactFeedback** *feedback, msgs::Contact *_msg)
- virtual void **DebugPrint** () const

  Debug print out of the physic engine state.

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 **OnRequest** (ConstRequestPtr &)
- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &)

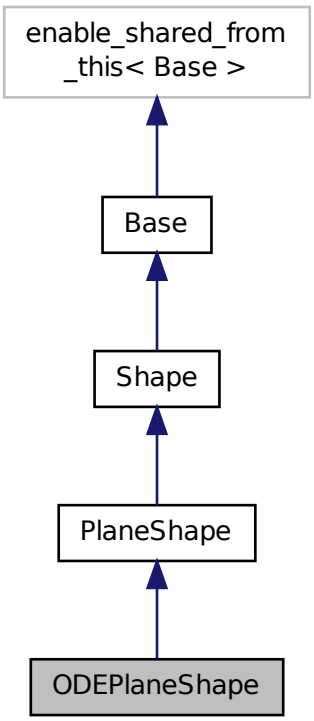
Additional Inherited Members

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

- ODEPhysics.hh

9.126 ODEPlaneShape Class Reference

Inheritance diagram for 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.

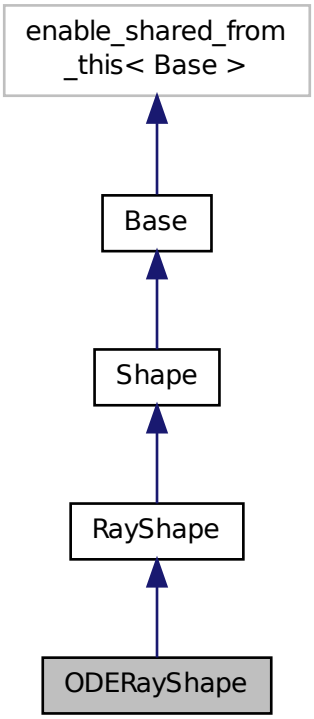
Additional Inherited Members

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

- ODEPlaneShape.hh

9.127 ODERayShape Class Reference

Inheritance diagram for ODERayShape:



Public Member Functions

- **ODERayShape** (PhysicsEnginePtr _physicsEngine)

  Constructor for a global ray.
- **ODERayShape** (CollisionPtr collision)

  Constructor.
- virtual ~**ODERayShape** ()

  Destructor.
- virtual void **Update** ()

  Update the tay collision.
- 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.

Additional Inherited Members

9.127.1 Constructor & Destructor Documentation

9.127.1.1 ODERayShape (CollisionPtr *collision*)

Constructor.

Parameters

body	**Link** (p. 192) the ray is attached to

9.127.2 Member Function Documentation

9.127.2.1 virtual void 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 **RayShape** (p. 311).

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

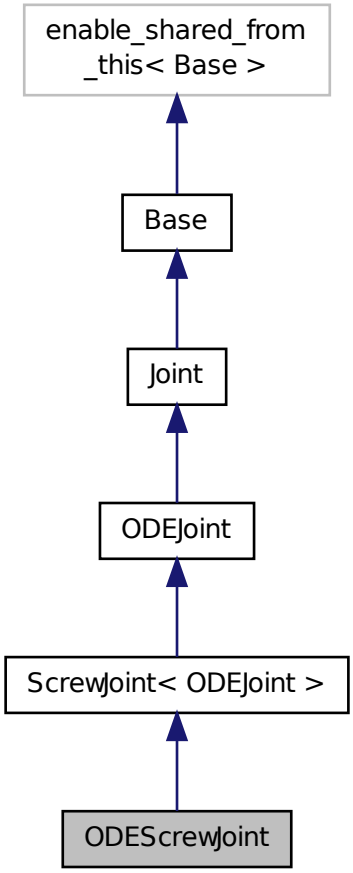
- ODERayShape.hh

9.128 ODEScrewJoint Class Reference

A screw joint.

`#include <ODEScrewJoint.hh>`

Inheritance diagram for ODEScrewJoint:



Public Member Functions

- **ODEScrewJoint** (dWorldID worldId)

  Constructor.
- virtual ~**ODEScrewJoint** ()

  Destructor.
- virtual **math::Vector3 GetGlobalAxis** (int index) const

  Get the axis of rotation.
- virtual 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 **SetThreadPitch** (int index, const double _thread_pitch)

  Set screw joint thread pitch.

- void **ApplyDamping** ()

  callback to apply damping force to joint

- virtual **math::Angle GetAngleImpl** (int index) const

  Get the position of the joint.

- virtual double **GetVelocity** (int index) const

  Get the rate of change.

- virtual void **SetVelocity** (int index, double angle)

  Set the velocity of an axis(index).

- 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 double **GetMaxForce** (int index)

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

- virtual double **GetParam** (int parameter) const

  Get the _parameter.

- virtual void **SetParam** (int parameter, double value)

  Set the _parameter.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the joint.

Additional Inherited Members

9.128.1 Detailed Description

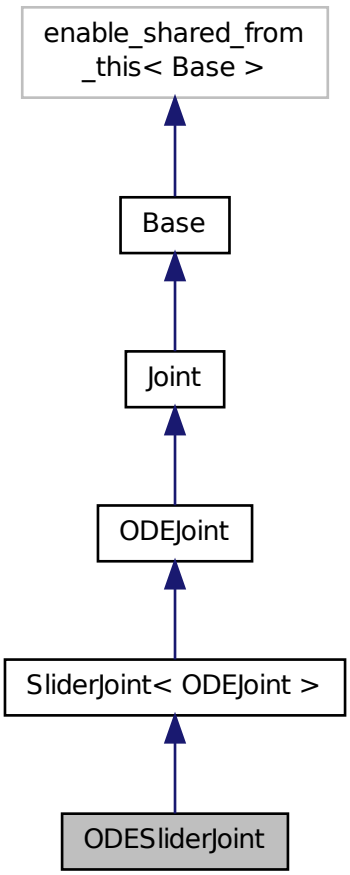
A screw joint.

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

- ODEScrewJoint.hh

9.129 ODESliderJoint Class Reference

Inheritance diagram for ODESliderJoint:



Public Member Functions

- **ODESliderJoint** (dWorldID worldId)

  Constructor.
- virtual ~**ODESliderJoint** ()

  Destructor.
- virtual **math::Vector3 GetGlobalAxis** (int index) const

  Get the axis of rotation.
- virtual 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.

- void **ApplyDamping** ()

  callback to apply damping force to joint

- virtual **math::Angle GetAngleImpl** (int index) const

  Get the position of the joint.

- virtual double **GetVelocity** (int index) const

  Get the rate of change.

- virtual void **SetVelocity** (int index, double angle)

  Set the velocity of an axis(index).

- 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 double **GetMaxForce** (int index)

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

- virtual double **GetParam** (int parameter) const

  Get the _parameter.

- virtual void **SetParam** (int parameter, double value)

  Set the _parameter.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the joint.

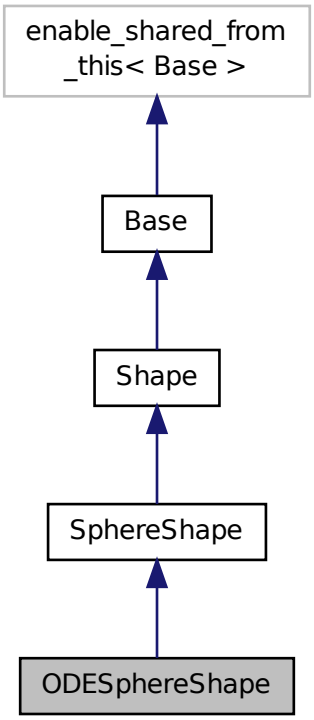
Additional Inherited Members

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

- ODESliderJoint.hh

9.130 ODESphereShape Class Reference

Inheritance diagram for ODESphereShape:



Public Member Functions

- **ODESphereShape** (ODECollisionPtr _parent)

  Constructor.
- virtual ~**ODESphereShape** ()

  Destructor.
- void **SetRadius** (const double &_radius)

  Set the radius.

Additional Inherited Members

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

- ODESphereShape.hh

9.131 ODESurfaceParams Class Reference

Public Member Functions

- **ODESurfaceParams** ()

  Constructor.
- virtual ~**ODESurfaceParams** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)

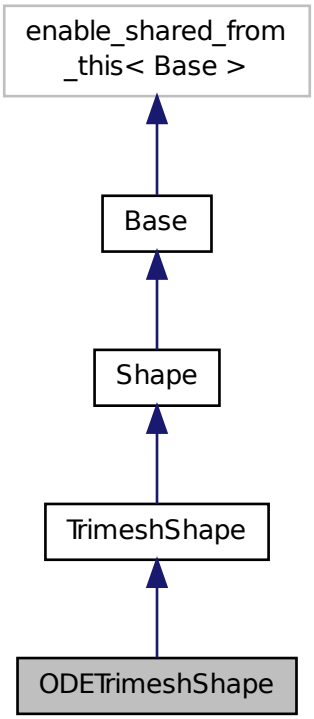
  Load the contact params.

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

- ODESurfaceParams.hh

9.132 ODETrimeshShape Class Reference

Inheritance diagram for ODETrimeshShape:



Public Member Functions

- **ODETrimeshShape** (CollisionPtr _parent)

  Constructor.

- virtual ~**ODETrimeshShape** ()

  Destructor.

- void **Update** ()

  Update function.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the trimesh.

- virtual void **Init** ()

  Init the trimesh shape.

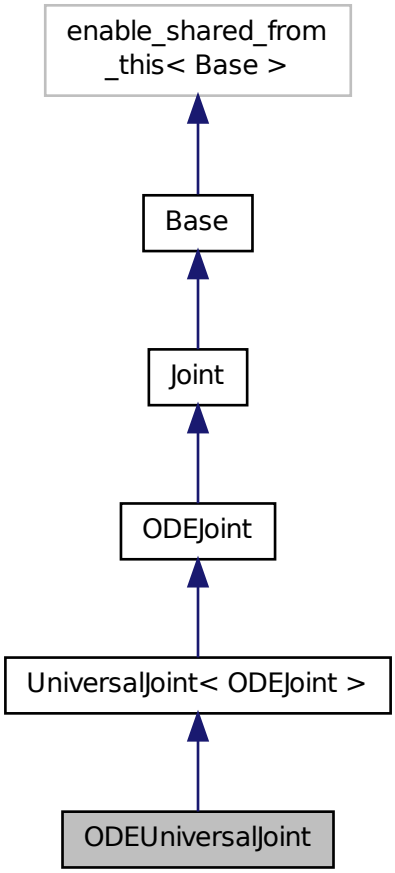
Additional Inherited Members

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

- ODETrimeshShape.hh

9.133 ODEUniversalJoint Class Reference

Inheritance diagram for ODEUniversalJoint:



Public Member Functions

- **ODEUniversalJoint** (dWorldID worldId)

  Constructor.
- virtual ~**ODEUniversalJoint** ()

  Destuctor.
- virtual **math::Vector3 GetAnchor** (int index) const

  Get the anchor point.
- virtual void **SetAnchor** (int index, const **math::Vector3** &anchor)

  Set the anchor point.
- virtual void **SetDamping** (int index, const double damping)

  Set joint damping, not yet implemented.

- virtual **math::Vector3 GetGlobalAxis** (int index) const

  Get the first axis of rotation.
- virtual void **SetAxis** (int index, const **math::Vector3** &axis)

  Set the first axis of rotation.
- virtual **math::Angle GetAngleImpl** (int index) const

  Get the angle of axis.
- virtual double **GetVelocity** (int index) const

  Get the angular rate of an axis.
- virtual void **SetVelocity** (int index, double angle)

  Set the velocity of an axis(index).
- 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 double **GetMaxForce** (int index)

  Get the max allowed force of an axis(index).
- virtual void **SetParam** (int parameter, double value)

  Set the parameter to value.

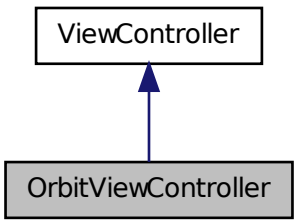
Additional Inherited Members

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

- ODEUniversalJoint.hh

9.134 OrbitViewController Class Reference

Inheritance diagram for OrbitViewController:



Public Member Functions

- **OrbitViewController** (**UserCamera** ∗camera)

  Constructor.
- virtual ∼**OrbitViewController** ()

  Destructor.
- virtual void **Init** ()
- virtual void **Init** (const **math::Vector3** &_focalPoint)
- void **SetDistanceRange** (double _minDist, double _maxDist)

  Set the min and max distance from the focal point.
- virtual void **Update** ()

  Update.
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &event)

  Handle a mouse event.
- void **SetDistance** (float _d)

  Set the distance to the focal point.
- void **SetFocalPoint** (const **math::Vector3** &_fp)

  Set the focal point.
- **math::Vector3 GetFocalPoint** () const

  Get the focal point.
- void **SetYaw** (double _yaw)
- void **SetPitch** (double _pitch)

Static Public Member Functions

- static std::string **GetTypeString** ()

  Get the type name of this view controller.

Additional Inherited Members

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

- OrbitViewController.hh

9.135 Packet Interface Reference

Message that encapsulates another message with a type description.

9.135.1 Detailed Description

Message that encapsulates another message with a type description.

@verbatim

import "time.proto";

message **Packet** (p. 267) { required **Time** (p. 359) stamp = 1; required string type = 2; required bytes serialized_data = 3; }

///

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

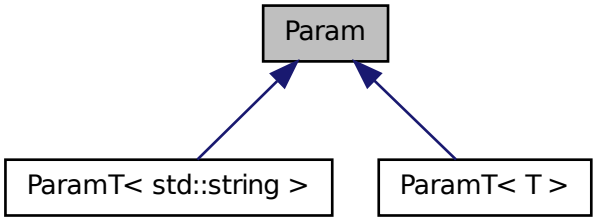
- packet.proto

9.136 Param Class Reference

A parameter class.

`#include <Param.hh>`

Inheritance diagram for Param:



Public Member Functions

- **Param** (**Param** ∗_newParam)

  Constructor.
- virtual ∼**Param** ()

  Destructor.
- virtual std::string **GetAsString** () const

  Get the type.
- virtual std::string **GetDefaultAsString** () const
- virtual bool **SetFromString** (const std::string &)

  Set the parameter value from a string.
- virtual void **Reset** ()=0

  Reset the parameter.
- const std::string & **GetKey** () const
- std::string **GetTypeName** () const
- bool **GetRequired** () const
- bool **GetSet** () const

  Return true if the parameter has been set.

- virtual boost::shared_ptr< **Param** > **Clone** () const =0
- template<typename T >
  void **SetUpdateFunc** (T _updateFunc)

  Update function.
- virtual void **Update** ()=0
- bool **IsBool** () const
- bool **IsInt** () const
- bool **IsUInt** () const
- bool **IsFloat** () const
- bool **IsDouble** () const
- bool **IsChar** () const
- bool **IsStr** () const
- bool **IsVector3** () const
- bool **IsVector2i** () const
- bool **IsVector2d** () const
- bool **IsQuaternion** () const
- bool **IsPose** () const
- bool **IsColor** () const
- bool **IsTime** () const
- 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)
- 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)

Protected Attributes

- std::string **key**
- bool **required**
- bool **set**
- std::string **typeName**
- boost::function< boost::any()> **updateFunc**

9.136.1 Detailed Description

A parameter class.

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

- Param.hh

9.137 ParamT< T > Class Template Reference

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

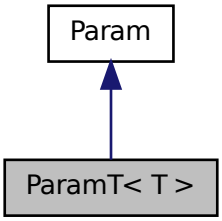
- **CommonTypes.hh**

9.138 ParamT< T > Class Template Reference

Templatized parameter class.

`#include <Param.hh>`

Inheritance diagram for ParamT< T >:



Public Member Functions

- **ParamT** (const std::string &_key, const std::string &_default, bool _required)

  Constructor.
- virtual ~**ParamT** ()

Destructor.

- virtual void **Update** ()
- virtual std::string **GetAsString** () const

  Get the parameter value as a string.
- virtual bool **SetFromString** (const std::string &_value)

  Set the parameter value from a string.
- virtual std::string **GetDefaultAsString** () const
- virtual bool **Set** (const std::string &_str)

  Set the parameter value from a string.
- T **GetValue** () const

  Get the value.
- void **SetValue** (const T &_value)

  Set the value of the parameter.
- virtual void **Reset** ()

  Reset to default value.
- virtual boost::shared_ptr< **Param** > **Clone** () const
- T **operator*** () const

Protected Attributes

- T **value**
- T **defaultValue**

Friends

- std::ostream & **operator<<** (std::ostream &_out, const **ParamT**< T > &_p)

9.138.1 Detailed Description

template<typename T>class sdf::ParamT< T >

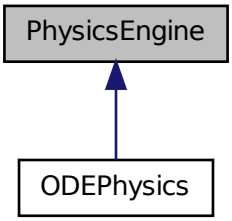
Templatized parameter class.

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

- Param.hh

9.139 PhysicsEngine Class Reference

Inheritance diagram for PhysicsEngine:



Public Member Functions

- **PhysicsEngine** (WorldPtr _world)

  Default constructor.
- virtual ~**PhysicsEngine** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the physics engine.
- virtual void **Init** ()=0

  Initialize the physics engine.
- virtual void **Fini** ()

  Finilize the physics engine.
- virtual void **Reset** ()
- virtual void **InitForThread** ()=0

  Init the engine for threads.
- virtual void **UpdateCollision** ()=0

  Update the physics engine collision.
- void **SetUpdateRate** (double _value)

  Set the simulation update rate.
- double **GetUpdateRate** ()

  Get the simulation update rate.
- double **GetUpdatePeriod** ()
- virtual void **SetStepTime** (double _value)=0

  Set the simulation step time.
- virtual double **GetStepTime** ()=0

  Get the simulation step time.
- virtual void **UpdatePhysics** ()

  Update the physics engine.
- virtual LinkPtr **CreateLink** (ModelPtr _parent)=0

Create a new body.

- 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 ShapePtr **CreateShape** (const std::string &_shapeType, CollisionPtr _collision)=0
- virtual JointPtr **CreateJoint** (const std::string &_type)=0

  Create a new joint.
- **math::Vector3 GetGravity** () const

  Return the gavity vector.
- virtual void **SetGravity** (const **gazebo::math::Vector3** &gravity)=0

  Set the gavity vector.
- void **ShowContacts** (const bool &show)

  Set whether to show contacts.
- virtual void **SetWorldCFM** (double)

  access functions to set ODE parameters
- virtual void **SetWorldERP** (double)

  access functions to set ODE parameters
- virtual void **SetAutoDisableFlag** (bool)

  access functions to set ODE parameters
- virtual void **SetSORPGSPreconIters** (unsigned int)

  access functions to set ODE parameters
- virtual void **SetSORPGSIters** (unsigned int)

  access functions to set ODE parameters
- virtual void **SetSORPGSW** (double)

  access functions to set ODE parameters
- virtual void **SetContactMaxCorrectingVel** (double)

  access functions to set ODE parameters
- virtual void **SetContactSurfaceLayer** (double)

  access functions to set ODE parameters
- virtual void **SetMaxContacts** (double)

  access functions to set ODE parameters
- virtual double **GetWorldCFM** ()

  access functions to set ODE parameters
- virtual double **GetWorldERP** ()

  access functions to set ODE parameters
- virtual bool **GetAutoDisableFlag** ()

  access functions to set ODE parameters
- virtual int **GetSORPGSPreconIters** ()

  access functions to set ODE parameters
- virtual int **GetSORPGSIters** ()

  access functions to set ODE parameters
- virtual double **GetSORPGSW** ()

  access functions to set ODE parameters
- virtual double **GetContactMaxCorrectingVel** ()

  access functions to set ODE parameters
- virtual double **GetContactSurfaceLayer** ()

access functions to set ODE parameters

- virtual int **GetMaxContacts** ()

  access functions to set ODE parameters

- virtual void **DebugPrint** () const =0

  Debug print out of the physic engine state.

- boost::recursive_mutex * **GetPhysicsUpdateMutex** () const

Protected Member Functions

- virtual void **OnRequest** (ConstRequestPtr &)
- virtual void **OnPhysicsMsg** (ConstPhysicsPtr &)

Protected Attributes

- WorldPtr **world**
- sdf::ElementPtr **sdf**
- transport::NodePtr **node**
- transport::PublisherPtr **responsePub**
- transport::PublisherPtr **contactPub**
- transport::SubscriberPtr **physicsSub**
- transport::SubscriberPtr **requestSub**
- boost::recursive_mutex * **physicsUpdateMutex**

9.139.1 Constructor & Destructor Documentation

9.139.1.1 PhysicsEngine (WorldPtr *_world*)

Default constructor.

Parameters

world	Pointer to the world

9.139.2 Member Function Documentation

9.139.2.1 math::Vector3 GetGravity ( ) const

Return the gavity vector.

Returns

The gavity vector

9.139.2.2 virtual void Load (sdf::ElementPtr *_sdf*) [virtual]

Load the physics engine.

Parameters

_sdf	Pointer to the SDF parameters

Reimplemented in **ODEPhysics** (p. 252).

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

- PhysicsEngine.hh

9.140 PhysicsFactory Class Reference

The physics factory.

`#include <PhysicsFactory.hh>`

Static Public Member Functions

- static void **RegisterAll** ()

  Register everything.
- static void **RegisterPhysicsEngine** (std::string classname, PhysicsFactoryFn factoryfn)

  Register a physics class.
- static PhysicsEnginePtr **NewPhysicsEngine** (const std::string &classname, WorldPtr world)

  Create a new instance of a physics engine.

9.140.1 Detailed Description

The physics factory.

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

- PhysicsFactory.hh

9.141 PID Class Reference

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.
- 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].
- void **SetPGain** (double _p)

  Set the proportional Gain.
- void **SetIGain** (double _i)

Set the integral Gain.

- void **SetDGain** (double _d)

  Set the derivtive Gain.

- void **SetIMax** (double _i)

  Set the integral upper limit.

- void **SetIMin** (double _i)

  Set the integral lower limit.

- void **SetCmdMax** (double _c)

  Set the maximum value for the command.

- void **SetCmdMin** (double _c)

  Set the maximum value for the command.

- double **Update** (double _error, **common::Time** _dt)

  Update the Pid loop with nonuniform time step size.

- void **SetCmd** (double _cmd)

  *Set current command for this **PID** (p. 275) controller.*

- double **GetCmd** ()

  *Return current command for this **PID** (p. 275) controller.*

- void **GetErrors** (double &_pe, double &_ie, double &_de)

  *Return **PID** (p. 275) error terms for the controller.*

- **PID** & **operator=** (const **PID** &_p)

- void **Reset** ()

  Reset the errors and command.

9.141.1 Constructor & Destructor Documentation

9.141.1.1 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

_p	The proportional gain.
_i	The integral gain.
_d	The derivative gain.
_imax	The integral upper limit.
_imin	The integral lower limit.

9.141.2 Member Function Documentation

9.141.2.1 void GetErrors (double & *_pe*, double & *_ie*, double & *_de*)

Return **PID** (p. 275) error terms for the controller.

Parameters

_pe	The proportional error.
_ie	The integral error.
_de	The derivative error.

9.141.2.2 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].

Parameters

_p	The proportional gain.
_i	The integral gain.
_d	The derivative gain.
_imax	The integral upper limit.
_imin	The integral lower limit.

9.141.2.3 void SetCmd (double _cmd)

Set current command for this **PID** (p. 275) controller.

Parameters

_cmd	New command

9.141.2.4 void SetCmdMax (double _c)

Set the maximum value for the command.

Parameters

_c	The maximum value

9.141.2.5 void SetCmdMin (double _c)

Set the maximum value for the command.

Parameters

_c	The maximum value

9.141.2.6 void SetDGain (double _d)

Set the derivtive Gain.

Parameters

_p	dertivative gain value

9.141.2.7 void SetIGain (double _i)

Set the integral Gain.

Parameters

_p	integral gain value

9.141.2.8 void SetIMax (double _i)

Set the integral upper limit.

Parameters

_p	integral upper limit value

9.141.2.9 void SetIMin (double _i)

Set the integral lower limit.

Parameters

_p	integral lower limit value

9.141.2.10 void SetPGain (double _p)

Set the proportional Gain.

Parameters

_p	proportional gain value

9.141.2.11 double Update (double _error, common::Time _dt)

Update the Pid loop with nonuniform time step size.

Parameters

_error	Error since last call (p_state-p_target)
_dt	Change in time since last call

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

- PID.hh

9.142 PID Interface Reference

Message for **PID** (p. 278).

9.142.1 Detailed Description

Message for **PID** (p. 278).

```
@verbatim
```

message **PID** (p. 278) { optional double target = 1[default=0.0]; optional double p_gain = 2[default=0.0]; optional double i_gain = 3[default=0.0]; optional double d_gain = 4[default=0.0]; optional double i_max = 5[default=0.0]; optional double i_min = 6[default=0.0]; optional double limit = 7[default=0.0]; } ///

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

- pid.proto

9.143 Plane Class Reference

Public Member Functions

- **Plane** ()

  Constructor.
- **Plane** (const **Vector3** &_normal, double _offset=0.0)

  Constructor.
- **Plane** (const **Vector3** &_normal, const **Vector2d** &_size, double _offset)

  Constructor.
- virtual ~**Plane** ()

  Destructor.
- void **Set** (const **Vector3** &_normal, const **Vector2d** &_size, double offset)

  Set the plane.
- 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.

Public Attributes

- **Vector3 normal**

  Plane *(*p. *279) normal.*
- **Vector2d size**

  Plane *(*p. *279) size.*
- double **d**

  Plane *(*p. *279) offset.*

9.143.1 Constructor & Destructor Documentation

9.143.1.1 Plane (const Vector3 & *_normal,* double *_offset* = 0.0)

Constructor.

Parameters

_normal	The plane normal
_offset	Offset along the normal

9.143.1.2 Plane (const Vector3 & _normal, const Vector2d & _size, double _offset)

Constructor.

Parameters

_normal	The plane normal
_size	Size of the plane
_offset	Offset along the normal

9.143.2 Member Function Documentation

9.143.2.1 Plane& operator= (const Plane & _p)

Equal operator.

Parameters

_p	**Plane** (p. 279) values

9.143.2.2 void Set (const Vector3 & _normal, const Vector2d & _size, double offset)

Set the plane.

Parameters

_normal	The plane normal
_size	Size of the plane
_offset	Offset along the normal

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

- Plane.hh

9.144 Plane Interface Reference

Message for a plane geometry.

9.144.1 Detailed Description

Message for a plane geometry.

```
@verbatim
```

import "vector3d.proto"; import "vector2d.proto";

message PlaneGeom { required **Vector3d** (p. 376) normal = 1; optional **Vector2d** (p. 368) size = 2; optional double d = 3 [default = 0]; }

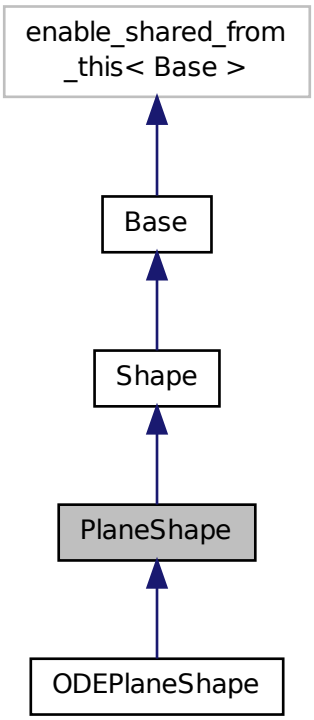
///

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

- planegeom.proto

9.145 PlaneShape Class Reference

Inheritance diagram for PlaneShape:



Public Member Functions

- **PlaneShape** (CollisionPtr **parent**)

  Constructor.
- virtual ~**PlaneShape** ()

  Destructor.
- virtual void **Init** ()

  Initialize the plane.
- virtual void **CreatePlane** ()

  Create the plane.
- virtual void **SetAltitude** (const **math::Vector3** &pos)

  Set the altitude of the plane.
- void **SetNormal** (const **math::Vector3** &norm)

Set the normal.

- **math::Vector3 GetNormal** () const
- void **FillShapeMsg** (msgs::Geometry &_msg)
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

Additional Inherited Members

9.145.1 Constructor & Destructor Documentation

9.145.1.1 PlaneShape (CollisionPtr *parent*)

Constructor.

Parameters

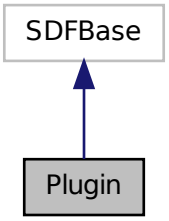
body	**Link** (p. 192) to which we are attached.

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

- PlaneShape.hh

9.146 Plugin Class Reference

Inheritance diagram for Plugin:



Public Member Functions

- void **Clear** ()
- void **Print** (const std::string &prefix)

Public Attributes

- **ParamT**< std::string > **name**
- **ParamT**< std::string > **filename**

- std::vector< **ParamT**
  < std::string > > **data**

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

- sdf/interface/Plugin.hh

9.147 PluginT< T > Class Template Reference

Public Types

- typedef boost::shared_ptr< T > **TPtr**

Public Member Functions

- std::string **GetFilename** () const

  Get the name of the handler.
- std::string **GetHandle** () const

  Get the short name of the handler.

Static Public Member Functions

- static TPtr **Create** (const std::string &_filename, const std::string &_handle)

Protected Attributes

- std::string **filename**
- std::string **handle**

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

- common/Plugin.hh

9.148 Pose Interface Reference

Message for a pose.

9.148.1 Detailed Description

Message for a pose.

Message for a model pose animation.

```
@verbatim
```

import "vector3d.proto"; import "quaternion.proto";

message **Pose** (p. 283) { optional string name = 1; required **Vector3d** (p. 376) position = 2; required **Quaternion** (p. 294) orientation = 3; }

///

```
@verbatim
```

import "pose.proto"; import "time.proto";

message PoseAnimation { required string model_name = 1; repeated **Pose** (p. 283) pose = 2; repeated **Time** (p. 359) time = 3; }

///

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

- pose.proto

9.149 Pose Class Reference

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.
- void **Set** (const **Vector3** &_pos, const **Quaternion** &_rot)
- bool **IsFinite** () const

  See if a pose is finite (e.g., not nan)
- void **Correct** ()

  Fix any nan values.
- **Pose GetInverse** () const

  Get the inverse of this pose.
- **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.

- bool **operator!=** (const **Pose** &_pose) const

  Inequality operator.
- **Pose operator*** (const **Pose** &pose)

  Multiplication operator.
- **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.
- **Pose CoordPoseSolve** (const **Pose** &_b) const

  Find the inverse of a pose; i.e., if b = this + a, given b and this, find a.
- 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.

Public Attributes

- **Vector3 pos**

  The position.
- **Quaternion rot**

  The rotation.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Pose** &_pose)

  Ostream operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Pose** &_pose)

9.149.1 Constructor & Destructor Documentation

9.149.1.1 Pose (const Vector3 & *_pos*, const Quaternion & *_rot*)

Constructor.

Parameters

pos	A position
rot	A rotation

9.149.1.2 Pose (const Pose & _pose)

Copy constructor.

Parameters

pose	**Pose** (p. 284) to copy

9.149.2 Member Function Documentation

9.149.2.1 Vector3 CoordPositionAdd (const Vector3 & _pos) const

Add one point to a vector: result = this + pos.

Parameters

pos	Position to add to this pose

Returns

The resulting position

9.149.2.2 Vector3 CoordPositionAdd (const Pose & _pose) const

Add one point to another: result = this + pose.

Parameters

pose	The **Pose** (p. 284) to add

Returns

The resulting position

9.149.2.3 Vector3 CoordPositionSub (const Pose & _pose) const [inline]

Subtract one position from another: result = this - pose.

Parameters

pose	**Pose** (p. 284) to subtract

Returns

The resulting position

References Quaternion::GetInverse(), Pose::pos, Pose::rot, Vector3::x, Quaternion::x, Vector3::y, Quaternion::y, Vector3::z, and Quaternion::z.

Referenced by Pose::operator-().

9.149.2.4 Quaternion CoordRotationAdd (const Quaternion & *_rot*) const

Add one rotation to another: result = this->rot + rot.

Parameters

rot	Rotation to add

Returns

The resulting rotation

9.149.2.5 Quaternion CoordRotationSub (const Quaternion & *_rot*) const [inline]

Subtract one rotation from another: result = this->rot - rot.

Parameters

rot	The rotation to subtract

Returns

The resulting rotation

References Quaternion::GetInverse(), Quaternion::Normalize(), and Pose::rot.

Referenced by Pose::operator-().

9.149.2.6 bool operator!= (const Pose & *_pose*) const

Inequality operator.

Parameters

_pose	**Pose** (p. 284) for comparison

Returns

True if not equal

9.149.2.7 Pose operator+ (const Pose & *pose*) const

Addition operator.

Parameters

pose	**Pose** (p. 284) to add to this pose

Returns

The resulting pose

9.149.2.8 const Pose& operator+= (const Pose & *pose*)

Add-Equals operator.

Parameters

pose	**Pose** (p. 284) to add to this pose

Returns

The resulting pose

9.149.2.9 Pose operator- (const Pose & *_pose*) const [inline]

Subtraction operator.

Parameters

pose	**Pose** (p. 284) to subtract from this one

Returns

The resulting pose

References Pose::CoordPositionSub(), Pose::CoordRotationSub(), Pose::Pose(), and Pose::rot.

9.149.2.10 const Pose& operator-= (const Pose & *_pose*)

Subtraction operator.

Parameters

pose	**Pose** (p. 284) to subtract from this one

Returns

The resulting pose

9.149.2.11 bool operator== (const Pose & *_pose*) const

Equality operator.

Parameters

_pose	**Pose** (p. 284) for comparison

Returns

True if equal

9.149.3 Friends And Related Function Documentation

9.149.3.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::math::Pose & *_pose*) [friend]

Ostream operator.

Parameters

out	Ostream
pose	**Pose** (p. 284) to output

Returns

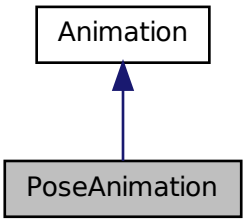
the Ostream

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

- Pose.hh

9.150 PoseAnimation Class Reference

Inheritance diagram for PoseAnimation:



Public Member Functions

- **PoseAnimation** (const std::string &_name, double _length, bool _loop)
- **PoseKeyFrame** * **CreateKeyFrame** (double _time)
- void **GetInterpolatedKeyFrame** (**PoseKeyFrame** &_kf) const

Protected Member Functions

- void **GetInterpolatedKeyFrame** (double _time, **PoseKeyFrame** &_kf) const
- void **BuildInterpolationSplines** () const

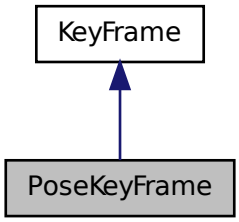
Additional Inherited Members

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

- Animation.hh

9.151 PoseKeyFrame Class Reference

Inheritance diagram for PoseKeyFrame:



Public Member Functions

- **PoseKeyFrame** (double _time)
- void **SetTranslation** (const **math::Vector3** &_trans)
- const **math::Vector3** & **GetTranslation** () const
- void **SetRotation** (const **math::Quaternion** &_rot)
- const **math::Quaternion** & **GetRotation** () const

Protected Attributes

- **math::Vector3 translate**
- **math::Quaternion rotate**

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

- KeyFrame.hh

9.152 Projector Class Reference

A Bumper controller.

`#include <Projector.hh>`

Public Member Functions

- **Projector** (VisualPtr _parent)

  Constructor.
- virtual ∼**Projector** ()

  Destructor.
- void **Load** (sdf::ElementPtr _sdf)

  Load from an sdf pointer.
- void **Load** (const msgs::Projector &_msg)

  Load from a message.
- void **Load** (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)
- void **SetTexture** (const std::string &_textureName)

  Load a texture into the projector.
- void **Toggle** ()

  Toggle the activation of the projector.

9.152.1 Detailed Description

A Bumper controller.

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

- Projector.hh

9.153 Projector Interface Reference

Information about a projector.

9.153.1 Detailed Description

Information about a projector.

```
@verbatim
```

import "pose.proto";

message **Projector** (p. 291) { required string name = 1; required string texture = 2; optional **Pose** (p. 283) pose = 3; optional double fov = 4[default=0.785]; optional double near_clip = 5[default=0.1]; optional double far_clip = 6[default=10.0]; } ///

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

- projector.proto

9.154 Publication Class Reference

Public Member Functions

- **Publication** (const std::string &topic, const std::string &msgType)

  Constructor.
- virtual ~**Publication** ()

  Destructor.
- std::string **GetMsgType** () const

  Get the type of message.
- void **AddSubscription** (const CallbackHelperPtr &callback)
- void **AddSubscription** (const NodePtr &_node)
- 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)
- unsigned int **GetTransportCount** () const
- unsigned int **GetCallbackCount** () const
- unsigned int **GetNodeCount** () const
- unsigned int **GetRemoteSubscriptionCount** ()
- bool **GetLocallyAdvertised** () const

  Return true if the topic has been advertised from this process.
- void **SetLocallyAdvertised** (bool _value)

  Set whether this topic has been advertised from this process.
- void **LocalPublish** (const std::string &data)

  Publish *(p. 293) data.*
- void **Publish** (const google::protobuf::Message &msg, const boost::function< void()> &cb=NULL)
- void **AddTransport** (const PublicationTransportPtr &publink)
- bool **HasTransport** (const std::string &_host, unsigned int _port)
- void **AddPublisher** (PublisherPtr _pub)

9.154.1 Member Function Documentation

9.154.1.1 bool GetLocallyAdvertised ( ) const

Return true if the topic has been advertised from this process.

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

- Publication.hh

9.155 PublicationTransport Class Reference

Public Member Functions

- **PublicationTransport** (const std::string &topic, const std::string &msgType)
- void **Init** (const ConnectionPtr &conn)

- void **Fini** ()
- void **AddCallback** (const boost::function< void(const std::string &)> &cb)
- const ConnectionPtr **GetConnection** () const
- std::string **GetTopic** () const
- std::string **GetMsgType** () const

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

- PublicationTransport.hh

9.156 Publish Interface Reference

Message that contains information about a publisher of data.

9.156.1 Detailed Description

Message that contains information about a publisher of data.

```
@verbatim
```

message **Publish** (p. 293) { required string topic = 1; required string msg_type = 2; required string host = 3; required uint32 port = 4; }

///

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

- publish.proto

9.157 Publisher Interface Reference

A list of publishers.

9.157.1 Detailed Description

A list of publishers.

```
@verbatim
```

import "publish.proto";

message Publishers { repeated **Publish** (p. 293) publisher = 1; }

///

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

- publishers.proto

9.158 Publisher Class Reference

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 **HasConnections** () const
- void **WaitForConnection** () const

  Block until a connection has been established with this publisher.
- void **SetPublication** (PublicationPtr &_publication, int _i)
- void **Publish** (const google::protobuf::Message &_message, bool _block=false)

  Publish *(*p. *293) a message on the topic.*
- template<typename M >
  void **Publish** (M _message, bool _block=false)
- unsigned int **GetOutgoingCount** () const
- std::string **GetTopic** () const

  Get the topic name.
- std::string **GetMsgType** () const

  Get the message type.
- void **SendMessage** ()

  Send latest message over the wire. For internal use only.
- bool **GetLatching** () const
- std::string **GetPrevMsg** () const

9.158.1 Constructor & Destructor Documentation

9.158.1.1 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

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

- Publisher.hh

9.159 Quaternion Interface Reference

A message for a quaternion.

9.159.1 Detailed Description

A message for a quaternion.

```
@verbatim
```

message **Quaternion** (p. 294) { required double x = 2; required double y = 3; required double z = 4; required double w = 5; }

///

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

- quaternion.proto

9.160 Quaternion Class Reference

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.
- **Quaternion** & **operator=** (const **Quaternion** &qt)

  Equal operator.
- void **Invert** ()

  Invert the quaternion.
- **Quaternion GetInverse** () const

  Get the inverse of this quaternion.
- void **SetToIdentity** ()

  Set the quatern to the identity.
- **Quaternion GetLog** () const
- **Quaternion GetExp** () const
- void **Normalize** ()

  Normalize the quaternion.
- 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 **Set** (double _u, double _x, double _y, double _z)

  Set this quaternion from another.
- void **SetFromEuler** (const **Vector3** &_vec)

Set the quaternion from Euler angles.

- **Vector3 GetAsEuler** () const

  Return the rotation in Euler angles.

- double **GetRoll** ()

  Get the Euler roll angle in radians.

- double **GetPitch** ()

  Get the Euler pitch angle in radians.

- double **GetYaw** ()

  Get the Euler yaw angle in radians.

- void **GetAsAxis** (**Vector3** &_axis, double &_angle) const

  Return rotation as axis and angle.

- void **Scale** (double _scale)

  Scale a Quaternionion.

- **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 **Quaternion** &qt)

  Substraction operator.

- **Quaternion operator*** (const **Quaternion** &_q) const

  Multiplication operator.

- **Quaternion operator*** (const double &_f) const

  Multipcation operator.

- **Quaternion operator*=** (const **Quaternion** &qt)

  Multiplication operator.

- **Vector3 operator*** (const **Vector3** &v) const

  Vector3 *(*p. *373) multiplication operator.*

- bool **operator==** (const **Quaternion** &_qt) const

  Equality operator.

- bool **operator!=** (const **Quaternion** &_qt) const

  Inequality operator.

- **Quaternion operator-** () const

  Negate 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.

- bool **IsFinite** () const

  See if a quatern is finite (e.g., not nan)

- void **Correct** ()

  Correct any nan.

- **Matrix3 GetAsMatrix3** () const

  Get the quaternion as a 3x3 matrix.

- **Matrix4 GetAsMatrix4** () const

  Get the quaternion as a 4x4 matrix.

- **Vector3 GetXAxis** () const
- **Vector3 GetYAxis** () const
- **Vector3 GetZAxis** () const
- void **Round** (int _precision)

  Round all values to _precision decimal places.
- double **Dot** (const **Quaternion** &_q) const

  Dot product.

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 Squad** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkA, const **Quaternion** &_rkB, const **Quaternion** &_rkQ, bool _shortestPath=false)

  Spherical quadratic interpolation.
- static **Quaternion Slerp** (double _fT, const **Quaternion** &_rkP, const **Quaternion** &_rkQ, bool _shortestPath=false)

  Spherical linear interpolation.

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)

  Ostream operator.
- std::istream & **operator**>> (std::istream &in, **gazebo::math::Quaternion** &q)

  Istream operator.

9.160.1 Constructor & Destructor Documentation

9.160.1.1 Quaternion (const double & *w*, const double & *x*, const double & *y*, const double & *z*)

Constructor.

Parameters

w	W param
x	X param
y	Y param
z	Z param

9.160.1.2 Quaternion (const Quaternion & _qt)

Copy constructor.

Parameters

qt	**Quaternion** (p. 295) to copy

9.160.2 Member Function Documentation

9.160.2.1 Vector3 GetAsEuler ( ) const

Return the rotation in Euler angles.

Returns

This quaternion as an Euler vector

9.160.2.2 Quaternion GetInverse ( ) const [inline]

Get the inverse of this quaternion.

Returns

Inverse quarenion

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

Referenced by Pose::CoordPositionSub(), Pose::CoordRotationSub(), and Quaternion::RotateVector().

9.160.2.3 bool IsFinite ( ) const

See if a quatern is finite (e.g., not nan)

Returns

True if quatern is finite

9.160.2.4 bool operator!= (const Quaternion & _qt) const

Inequality operator.

Parameters

_qt	**Quaternion** (p. 295) for comparison

Returns

True if not equal

9.160.2.5 Quaternion operator∗ (const Quaternion & _q) const [inline]

Multiplication operator.

Parameters

qt	**Quaternion** (p. 295) for multiplication

Returns

This quatern multiplied by the parameter

References Quaternion::Quaternion(), Quaternion::w, Quaternion::x, Quaternion::y, and Quaternion::z.

9.160.2.6 Quaternion operator∗ (const double & _f) const

Multipcation operator.

Parameters

_f	Double factor

Returns

Quaternion (p. 295) multiplied by _f

9.160.2.7 Quaternion operator∗= (const Quaternion & qt)

Multiplication operator.

Parameters

qt	**Quaternion** (p. 295) for multiplication

Returns

This quatern multiplied by the parameter

9.160.2.8 Quaternion operator+ (const Quaternion & qt) const

Addition operator.

Parameters

qt	**Quaternion** (p. 295) for addition

Returns

This quatern + qt

9.160.2.9 Quaternion operator+= (const Quaternion & *qt*)

Addition operator.

Parameters

qt	**Quaternion** (p. 295) for addition

Returns

This quatern + qt

9.160.2.10 Quaternion operator- (const Quaternion & *qt*) const

Substraction operator.

Parameters

qt	**Quaternion** (p. 295) for substraction

Returns

This quatern - qt

9.160.2.11 Quaternion operator-= (const Quaternion & *qt*)

Substraction operator.

Parameters

qt	**Quaternion** (p. 295) for substraction

Returns

This quatern - qt

9.160.2.12 Quaternion& operator= (const Quaternion & *qt*)

Equal operator.

Parameters

qt	**Quaternion** (p. 295) to copy

9.160.2.13 bool operator== (const Quaternion & _qt) const

Equality operator.

Parameters

_qt	**Quaternion** (p. 295) for comparison

Returns

True if equal

9.160.2.14 Vector3 RotateVector (const Vector3 & _vec) const [inline]

Rotate a vector using the quaternion.

Returns

The rotated vector

References Quaternion::GetInverse(), Vector3::x, Quaternion::x, Vector3::y, Quaternion::y, Vector3::z, and Quaternion::z.

9.160.2.15 void Scale (double _scale)

Scale a Quaternionion.

Parameters

scale	Amount to scale this rotation

9.160.2.16 void SetFromAxis (double _x, double _y, double _z, double _a)

Set the quaternion from an axis and angle.

Parameters

x	X axis
y	Y axis
z	Z axis
a	**Angle** (p. 75) in radians

9.160.2.17 void SetFromAxis (const Vector3 & _axis, double _a)

Set the quaternion from an axis and angle.

Parameters

_axis	**Axis** (p. 84)
_a	**Angle** (p. 75) in radians

9.160.2.18 void SetFromEuler (const Vector3 & *_vec*)

Set the quaternion from Euler angles.

Parameters

vec	Euler angle

9.160.3 Friends And Related Function Documentation

9.160.3.1 std::ostream& operator<< (std::ostream & *out,* const gazebo::math::Quaternion & *q*) [friend]

Ostream operator.

Parameters

out	Ostream
q	**Quaternion** (p. 295) to output

Returns

The ostream

9.160.3.2 std::istream& operator>> (std::istream & *in,* gazebo::math::Quaternion & *q*) [friend]

Istream operator.

Parameters

in	Ostream
q	**Quaternion** (p. 295) to read values into

Returns

The istream

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

- Quaternion.hh

9.161 Rand Class Reference

Static Public Member Functions

- static double **GetDblUniform** (double _min=0, double _max=1)

Get a double from a uniform distribution.

- static double **GetDblNormal** (double _mean=0, double _sigma=1)

  Get a double from a normal distribution.

- static int **GetIntUniform** (int _min, int _max)

  Get a integer from a uniform distribution.

- static int **GetIntNormal** (int _mean, int _sigma)

  Get a double from a normal distribution.

9.161.1 Member Function Documentation

9.161.1.1 static double GetDblNormal (double *_mean* = 0, double *_sigma* = 1) [static]

Get a double from a normal distribution.

Parameters

mean	Mean value for the distribution
sigma	Sigma value for the distribution

9.161.1.2 static double GetDblUniform (double *_min* = 0, double *_max* = 1) [static]

Get a double from a uniform distribution.

Parameters

min	Minimum bound for the random number
max	Maximum bound for the random number

9.161.1.3 static int GetIntNormal (int *_mean*, int *_sigma*) [static]

Get a double from a normal distribution.

Parameters

mean	Mean value for the distribution
sigma	Sigma value for the distribution

9.161.1.4 static int GetIntUniform (int *_min*, int *_max*) [static]

Get a integer from a uniform distribution.

Parameters

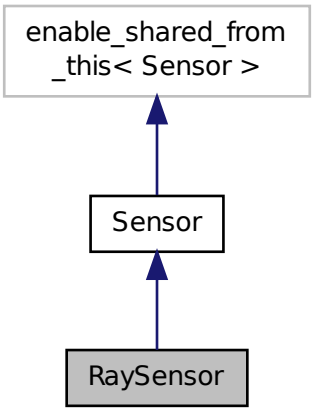
min	Minimum bound for the random number
max	Maximum bound for the random number

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

- Rand.hh

9.162 RaySensor Class Reference

Inheritance diagram for RaySensor:



Public Member Functions

- **RaySensor** ()

  Constructor.
- virtual ~**RaySensor** ()

  Destructor.
- virtual void **Load** (const std::string &_worldName, sdf::ElementPtr _sdf)

  Load the ray using parameter from an SDF.
- virtual void **Load** (const std::string &_worldName)

  Load the sensor with default parameters.
- virtual void **Init** ()

  Initialize the ray.
- virtual std::string **GetTopic** () const
- **math::Angle GetAngleMin** () const

  Get the minimum angle.
- **math::Angle GetAngleMax** () const

  Get the maximum angle.
- double **GetAngleResolution** () const

  Get radians between each range.
- double **GetRangeMin** () const

  Get the minimum range.
- double **GetRangeMax** () const

  Get the maximum range.
- double **GetRangeResolution** () const

Get the range resolution.

- int **GetRayCount** () const

  Get the ray count.
- int **GetRangeCount** () const

  Get the range count.
- int **GetVerticalRayCount** () const

  Get the vertical scan line count.
- int **GetVerticalRangeCount** () const

  Get the vertical scan line count.
- **math::Angle GetVerticalAngleMin** () const

  Get the vertical scan bottom angle.
- **math::Angle GetVerticalAngleMax** () const

  Get the vertical scan line top angle.
- double **GetRange** (int index)

  Get detected range for a ray.
- void **GetRanges** (std::vector< double > &_ranges)

  Get all the ranges.
- double **GetRetro** (int index)

  Get detected retro (intensity) value for a ray.
- int **GetFiducial** (int index)

  Get detected fiducial value for a ray.
- physics::MultiRayShapePtr **GetLaserShape** () const

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

  Update the sensor information.
- virtual void **Fini** ()

  Finalize the ray.

Additional Inherited Members

9.162.1 Member Function Documentation

9.162.1.1 math::Angle GetAngleMax ( ) const

Get the maximum angle.

Returns

the maximum angle

9.162.1.2 math::Angle GetAngleMin ( ) const

Get the minimum angle.

Returns

The minimum angle

9.162.1.3 int 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 aceess 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).

9.162.1.4 double 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 aceess loop. This means some data will come from one
scan, and some from another scan. You can solve this
problem by using SetActive(false) <your accessor loop>
SetActive(true).
```

Returns

Returns DBL_MAX for no detection.

9.162.1.5 int GetRangeCount ( ) const

Get the range count.

Returns

The number of ranges

9.162.1.6 double GetRangeMax ( ) const

Get the maximum range.

Returns

The maximum range

9.162.1.7 double GetRangeMin ( ) const

Get the minimum range.

Returns

The minimum range

9.162.1.8 void GetRanges (std::vector< double > & *_ranges*)

Get all the ranges.

Parameters

_range	A vector that will contain all the range data

9.162.1.9 int GetRayCount ( ) const

Get the ray count.

Returns

The number of rays

9.162.1.10 double 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 aceess 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).

9.162.1.11 math::Angle GetVerticalAngleMax ( ) const

Get the vertical scan line top angle.

Returns

The Maximum angle of the scan block

9.162.1.12 math::Angle GetVerticalAngleMin ( ) const

Get the vertical scan bottom angle.

Returns

The minimum angle of the scan block

9.162.1.13 int GetVerticalRangeCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

9.162.1.14 int GetVerticalRayCount ( ) const

Get the vertical scan line count.

Returns

The number of scan lines vertically

9.162.1.15 virtual void Load (const std::string & *_worldName,* sdf::ElementPtr *_sdf*) [virtual]

Load the ray using parameter from an SDF.

Parameters

node	The XMLConfig node

Reimplemented from **Sensor** (p. 331).

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

- RaySensor.hh

9.163 RaySensor Interface Reference

Information about a ray sensor element.

9.163.1 Detailed Description

Information about a ray sensor element.

@verbatim

message **RaySensor** (p. 308) { optional bool display_scan = 1; optional int32 horizontal_samples = 2; optional double horizontal_resolution = 3; optional double horizontal_min_angle = 4; optional double horizontal_max_angle = 5;

optional int32 vertical_samples = 6; optional double vertical_resolution = 7; optional double vertical_min_angle = 8; optional double vertical_max_angle = 9;

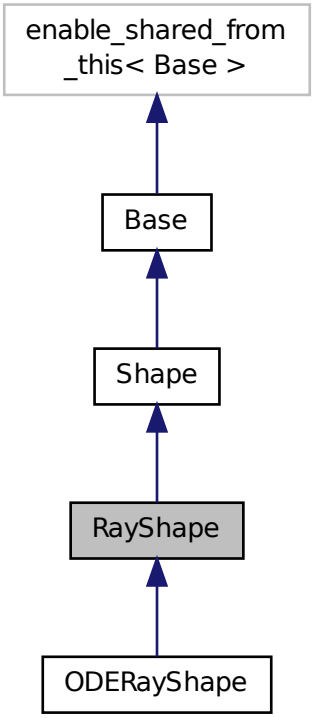
optional double range_min = 10; optional double range_max = 11; optional double range_resolution = 12; } ///

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

- raysensor.proto

9.164 RayShape Class Reference

Inheritance diagram for RayShape:



Public Member Functions

- **RayShape** (PhysicsEnginePtr _physicsEngine)

  Constructor for a global ray.
- **RayShape** (CollisionPtr **parent**)

  Constructor.
- virtual ~**RayShape** ()

  Destructor.
- 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 **GetRelativePoints** (**math::Vector3** &posA, **math::Vector3** &posB)

  Get the relative starting and ending points.
- virtual void **GetGlobalPoints** (**math::Vector3** &posA, **math::Vector3** &posB)

  Get the global starting and ending points.
- virtual void **SetLength** (double len)

  Set the length of the ray.

- double **GetLength** () const

  Get the length of the ray.
- virtual void **Update** ()=0

  Update the ray collision.
- virtual void **GetIntersection** (double &_dist, std::string &_entity)=0

  Get the nearest intersection.
- void **SetRetro** (float retro)

  Set the retro-reflectivness detected by this ray.
- float **GetRetro** () const

  Get the retro-reflectivness detected by this ray.
- void **SetFiducial** (int fid)

  Set the fiducial id detected by this ray.
- int **GetFiducial** () const

  Get the fiducial id detected by this ray.
- virtual void **Init** ()

  In the ray.
- void **FillShapeMsg** (msgs::Geometry &)
- virtual void **ProcessMsg** (const msgs::Geometry &)

Protected Attributes

- double **contactLen**

  Contact *(*p. *121) information; this is filled out during collision detection.*
- double **contactRetro**
- int **contactFiducial**
- **math::Vector3 relativeStartPos**

  Start and end positions of the ray, relative to the body.
- **math::Vector3 relativeEndPos**
- **math::Vector3 globalStartPos**

  Start and end positions of the ray in global cs.
- **math::Vector3 globalEndPos**

Additional Inherited Members

9.164.1 Constructor & Destructor Documentation

9.164.1.1 RayShape (CollisionPtr *parent*)

Constructor.

Parameters

body	**Link** (p. 192) the ray is attached to
displayRays	Indicates if the rays should be displayed when rendering images

9.164.2 Member Function Documentation

9.164.2.1 virtual void 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

9.164.2.2 virtual void 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

9.164.2.3 virtual void SetLength (double *len*) [virtual]

Set the length of the ray.

Parameters

len	Length of the array

9.164.2.4 virtual void 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 **ODERayShape** (p. 257).

9.164.3 Member Data Documentation

9.164.3.1 double contactLen [protected]

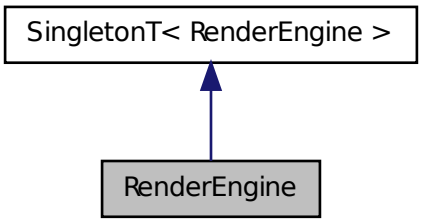
Contact (p. 121) information; this is filled out during collision detection.

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

- RayShape.hh

9.165 RenderEngine Class Reference

Inheritance diagram for RenderEngine:



Public Member Functions

- void **Load** ()
  Load the parameters for Ogre.
- void **Init** ()
  Initialize ogre.
- void **Fini** ()
  Finalize.
- void **Save** (std::string &prefix, std::ostream &stream)
  Save Ogre settings.
- double **GetUpdateRate** ()
  Get the desired update rate.
- ScenePtr **CreateScene** (const std::string &name, bool _enableVisualizations)
  Create a scene.
- void **RemoveScene** (const std::string &name)
  Remove a scene.
- ScenePtr **GetScene** (const std::string &_name)
  Get a scene.
- ScenePtr **GetScene** (unsigned int index)
  Get a scene manager.
- unsigned int **GetSceneCount** () const
  Get the number of scene managers.
- bool **HasGLSL** ()
  Returns true if the graphics card support GLSL.
- void **AddResourcePath** (const std::string &_path)

Public Attributes

- Ogre::Root ∗ **root**
  Pointer to the root scene node.

Protected Attributes

- uint64_t **dummyWindowId**

  ID for a dummy window. Used for gui-less operation.
- void * **dummyDisplay**

  Pointer to the dummy display.Used for gui-less operation.
- void * **dummyContext**

  GLX context used to render the scenes.Used for gui-less operation.

Additional Inherited Members

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

- RenderEngine.hh

9.166 Request Interface Reference

A message containing a string request.

9.166.1 Detailed Description

A message containing a string request.

@verbatim

message **Request** (p. 313) { required string request = 1; required int32 id = 2; optional string data = 3; optional double dbl_data = 4; }

///

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

- request.proto

9.167 Response Interface Reference

Message that encapsulates a respons message with a type description.

9.167.1 Detailed Description

Message that encapsulates a respons message with a type description.

@verbatim

message **Response** (p. 313) { required int32 id = 1; required string request = 2; required string response = 3; optional string type = 4; optional bytes serialized_data = 5; }

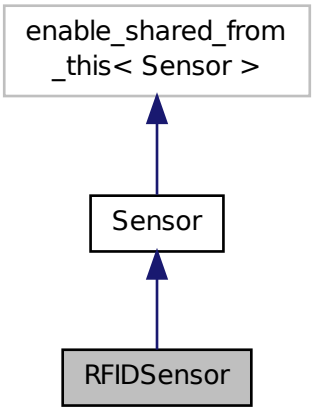
///

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

- response.proto

9.168 RFIDSensor Class Reference

Inheritance diagram for RFIDSensor:



Public Member Functions

- **RFIDSensor** ()

  Constructor.
- virtual ~**RFIDSensor** ()

  Destructor.
- 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 **Init** ()

  Initialize the sensor.
- virtual void **Fini** ()

  Finalize the sensor.

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

Additional Inherited Members

9.168.1 Member Function Documentation

9.168.1.1 virtual void Load (const std::string & *_worldName*, sdf::ElementPtr *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

_sdf	SDF **Sensor** (p. 329) parameteres

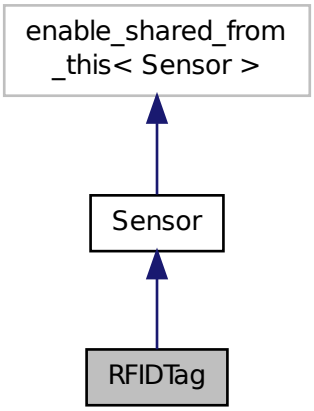
Reimplemented from **Sensor** (p. 331).

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

- RFIDSensor.hh

9.169 RFIDTag Class Reference

Inheritance diagram for RFIDTag:



Public Member Functions

- **RFIDTag** ()

  Constructor.
- virtual ~**RFIDTag** ()

  Destructor.
- 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 **Init** ()

  Initialize the sensor.
- virtual void **Fini** ()

  Finalize the sensor.
- **math::Pose GetTagPose** ()

Protected Member Functions

- virtual void **UpdateImpl** (bool _force)

Additional Inherited Members

9.169.1 Member Function Documentation

9.169.1.1 virtual void Load (const std::string & *_worldName*, sdf::ElementPtr & *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

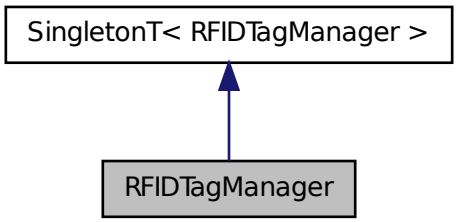
_sdf	SDF **Sensor** (p. 329) parameteres

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

- RFIDTag.hh

9.170 RFIDTagManager Class Reference

Inheritance diagram for RFIDTagManager:



Public Member Functions

- void **AddTaggedModel** (**RFIDTag** *_model)
- std::vector< **RFIDTag** * > **GetTags** ()

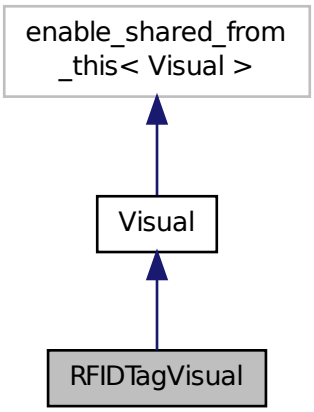
Additional Inherited Members

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

- RFIDTagManager.hh

9.171 RFIDTagVisual Class Reference

Inheritance diagram for RFIDTagVisual:



Public Member Functions

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

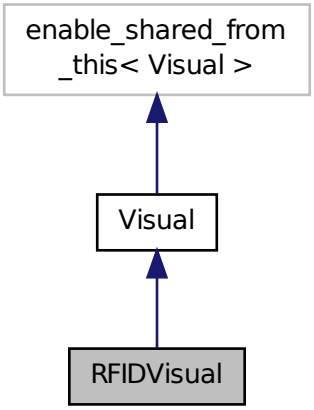
Additional Inherited Members

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

- RFIDTagVisual.hh

9.172 RFIDVisual Class Reference

Inheritance diagram for RFIDVisual:



Public Member Functions

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

Additional Inherited Members

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

- RFIDVisual.hh

9.173 RotationSpline Class Reference

Public Member Functions

- void **AddPoint** (const **Quaternion** &_p)

  Adds a control point to the end of the spline.
- const **Quaternion** & **GetPoint** (unsigned int _index) const

  Gets the detail of one of the control points of the spline.
- unsigned int **GetNumPoints** () const

  Gets the number of control points in the spline.
- void **Clear** ()

  Clears all the points in the spline.
- void **UpdatePoint** (unsigned int _index, const **Quaternion** &_value)

Updates a single point in 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 **SetAutoCalculate** (bool _autoCalc)

  Tells the spline whether it should automatically calculate tangents on demand as points are added.
- void **RecalcTangents** ()

  Recalculates the tangents associated with this spline.

Protected Attributes

- bool **autoCalc**
- std::vector< **Quaternion** > **points**
- std::vector< **Quaternion** > **tangents**

9.173.1 Member Function Documentation

9.173.1.1 Quaternion 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

t	Parametric value.
useShortestPath	Defines if rotation should take the shortest possible path

9.173.1.2 Quaternion Interpolate (unsigned int _fromIndex, double _t, bool _useShortestPath = true)

Interpolates a single segment of the spline given a parametric value.

Parameters

fromIndex	The point index to treat as t = 0. fromIndex + 1 is deemed to be t = 1
t	Parametric value
useShortestPath	Defines if rotation should take the shortest possible path

9.173.1.3 void 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.

9.173.1.4 void 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

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.

9.173.1.5 void UpdatePoint (unsigned int *_index,* const Quaternion & *_value*)

Updates a single point in the spline.

Remarks

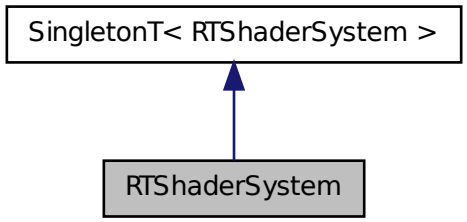
This point must already exist in the spline.

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

- RotationSpline.hh

9.174 RTShaderSystem Class Reference

Inheritance diagram for RTShaderSystem:



Public Types

- enum **LightingModel** { **SSLM_PerVertexLighting**, **SSLM_PerPixelLighting**, **SSLM_NormalMapLighting-TangentSpace**, **SSLM_NormalMapLightingObjectSpace** }

Public Member Functions

- void **Init** ()

  Init the run time shader system.
- void **Fini** ()

  Finalize the shader system.
- void **Clear** ()
- void **AddScene** (**Scene** ∗_scene)

  Add a scene manager.
- void **RemoveScene** (**Scene** ∗scene)

  Remove a scene.
- void **UpdateShaders** ()

  Update the shaders.
- void **AttachEntity** (**Visual** ∗vis)

  Set an Ogre::Entity to use RT shaders.
- void **DetachEntity** (**Visual** ∗vis)

  Remove and entity.
- void **SetPerPixelLighting** (bool s)

  Set the lighting model to per pixel or per vertex.
- void **GenerateShaders** (**Visual** ∗vis)

  Generate shaders for an entity.
- void **ApplyShadows** (**Scene** ∗scene)
- void **RemoveShadows** (**Scene** ∗_scene)

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)

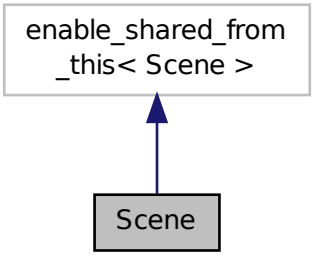
Additional Inherited Members

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

- RTShaderSystem.hh

9.175 Scene Class Reference

Inheritance diagram for Scene:



Public Types

- enum **SceneType** { **BSP**, **GENERIC** }

Public Member Functions

- **Scene** (const std::string &_name, bool _enableVisualizations=false)

  Constructor.
- virtual ∼**Scene** ()

  Destructor.
- void **Load** (sdf::ElementPtr _scene)

  Load the scene from a set of parameters.
- void **Load** ()

  Load the scene with default parameters.
- void **Init** ()

  Init.

- void **PreRender** ()

  Process all received messages.
- Ogre::SceneManager * **GetManager** () const

  Get the OGRE scene manager.
- std::string **GetName** () const

  Get the name of the scene.
- void **SetAmbientColor** (const **common::Color** &color)

  Set the ambient color.
- **common::Color GetAmbientColor** () const

  Get the ambient color.
- void **SetBackgroundColor** (const **common::Color** &color)

  Set the background color.
- **common::Color GetBackgroundColor** () const

  Get the background color.
- void **CreateGrid** (uint32_t cell_count, float cell_length, float line_width, const **common::Color** &color)

  Create a grid.
- **Grid** * **GetGrid** (uint32_t index) const

  Get the grid.
- 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 visual laser.
- uint32_t **GetCameraCount** () const

  Get the number of cameras in this scene.
- CameraPtr **GetCamera** (uint32_t index) const

  Get a camera.
- CameraPtr **GetCamera** (const std::string &_name) const

  Get a camera by name.
- UserCameraPtr **CreateUserCamera** (const std::string &name)

  Create a user camera.
- uint32_t **GetUserCameraCount** () const

  Get the number of user cameras in this scene.
- UserCameraPtr **GetUserCamera** (uint32_t index) const

  Get a user camera.
- LightPtr **GetLight** (const std::string &_name) const

  Get a light by name.
- unsigned int **GetLightCount** () const

  Get the count of the lights.
- LightPtr **GetLight** (unsigned int _index) const

  Get a light based on an index.
- VisualPtr **CreateVisual** (const std::string &_name)

  Create a visual.
- VisualPtr **GetVisual** (const std::string &_name) const

  Get a visual by name.
- VisualPtr **SelectVisualAt** (CameraPtr camera, **math::Vector2i** mousePos)

- void **SelectVisual** (const std::string &_name) const

  Select a visual by name.
- VisualPtr **GetVisualAt** (CameraPtr camera, **math::Vector2i** mousePos, std::string &mod)

  Get an entity at a pixel location using a camera.
- void **SnapVisualToNearestBelow** (const std::string &_visualName)
- VisualPtr **GetVisualAt** (CameraPtr _camera, **math::Vector2i** _mousePos)

  Get a visual at a mouse position.
- VisualPtr **GetModelVisualAt** (CameraPtr _camera, **math::Vector2i** _mousePos)

  Get a model's 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.
- **math::Vector3 GetFirstContact** (CameraPtr camera, **math::Vector2i** mousePos)

  Get the world pos of a the first contact at a pixel location.
- void **PrintSceneGraph** ()
- void **SetVisible** (const std::string &name, bool visible)

  Hide a visual.
- void **DrawLine** (const **math::Vector3** &start, const **math::Vector3** &end, const std::string &name)

  Draw a named line.
- void **SetFog** (const std::string &_type, const **common::Color** &_color, double _density, double _start, double _end)
- uint32_t **GetId** () const
- std::string **GetIdString** () const
- std::string **GetUniqueName** (const std::string &_prefix)

  Get a unique scene node node.
- **SelectionObj** * **GetSelectionObj** () const

  Get the selection object.
- void **SetSky** (const std::string &_material)
- void **SetShadowsEnabled** (bool _value)

  Set whether shadows are on or off.
- bool **GetShadowsEnabled** () const

  Get whether shadows are on or off.
- void **AddVisual** (VisualPtr &_vis)

  Add a visual to the scene.
- void **RemoveVisual** (VisualPtr _vis)

  Remove a visual from the scene.
- void **SetGrid** (bool _enabled)

  Set the grid on or off.
- VisualPtr **GetWorldVisual** () const
- VisualPtr **CloneVisual** (const std::string &_visualName, const std::string &_newName)

  Clone a visual.
- std::string **StripSceneName** (const std::string &_name) const
- **Heightmap** * **GetHeightmap** () const
- void **Clear** ()

9.175.1 Member Function Documentation

9.175.1.1 VisualPtr GetVisualAt (CameraPtr *camera,* math::Vector2i *mousePos,* std::string & *mod*)

Get an entity at a pixel location using a camera.

Used for mouse picking.

Parameters

camera	The ogre camera, used to do mouse picking
mousePos	The position of the mouse in screen coordinates
_mod	Used for object manipulation

Returns

The selected entity, or NULL

9.175.1.2 void SetGrid (bool *_enabled*)

Set the grid on or off.

Parameters

_enabled	Set to true to turn on the grid

9.175.1.3 void SetShadowsEnabled (bool *_value*)

Set whether shadows are on or off.

Parameters

_value	True to enable shadows, False to disable

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

- Scene.hh

9.176 Scene Interface Reference

A message containing a description of a scene.

9.176.1 Detailed Description

A message containing a description of a scene.

`@verbatim`

import "header.proto"; import "color.proto"; import "fog.proto"; import "shadows.proto"; import "visual.proto"; import "pose.proto"; import "light.proto"; import "joint.proto"; import "model.proto";

message **Scene** (p. 325) { required string name = 1; optional **Color** (p. 115) ambient = 2; optional **Color** (p. 115) background = 3; optional string sky_material = 4; optional bool shadows = 5 [default = true]; optional Fog fog = 6; optional bool grid = 7;

repeated **Model** (p. 222) model = 8; repeated **Light** (p. 191) light = 9; repeated **Joint** (p. 181) joint = 10; }

///

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

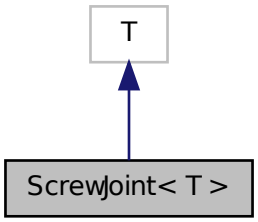
- scene.proto

9.177 ScrewJoint< T > Class Template Reference

A screw joint.

`#include <ScrewJoint.hh>`

Inheritance diagram for ScrewJoint< T >:



Public Member Functions

- **ScrewJoint** ()

  Constructor.
- virtual ~**ScrewJoint** ()

  Destructor.
- virtual void **SetAnchor** (int, const **math::Vector3** &anchor)

  Set the anchor.
- virtual **math::Vector3 GetAnchor** (int) const

  Get the anchor.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the joint.

Protected Attributes

- **math::Vector3 fakeAnchor**
- double **threadPitch**

9.177.1 Detailed Description

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

A screw joint.

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

- ScrewJoint.hh

9.178 SDF Class Reference

Base **SDF** (p. 327) class.

`#include <SDF.hh>`

Public Member Functions

- void **PrintDescription** ()
- void **PrintValues** ()
- void **Write** (const std::string &_filename)
- std::string **ToString** () const
- void **SetFromString** (const std::string &_sdfData)

  *Set **SDF** (p. 327) values from a string.*

Public Attributes

- ElementPtr **root**

9.178.1 Detailed Description

Base **SDF** (p. 327) class.

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

- SDF.hh

9.179 Selection Interface Reference

A message for **GUI** (p. 168) selection data.

9.179.1 Detailed Description

A message for **GUI** (p. 168) selection data.

```
@verbatim
```

import "header.proto";

message **Selection** (p. 327) { required uint32 id = 1; required string name = 2; optional bool selected = 3 [default = false]; }

///

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

- selection.proto

9.180 SelectionObj Class Reference

Public Member Functions

- **SelectionObj** (**Scene** *scene_)

  Constructor.
- virtual ~**SelectionObj** ()

  Destructor.
- void **Init** ()
- void **Attach** (VisualPtr visual)

  Set the position of the node.
- void **Clear** ()
- 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.
- std::string **GetVisualName** () const

  Get the name of the visual the seleciton obj is attached to.
- void **SetHighlight** (const std::string &_mod)

  Highlight the selection object based on a modifier.

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

- SelectionObj.hh

9.181 Sensor Interface Reference

Information about a sensor element.

9.181.1 Detailed Description

Information about a sensor element.

@verbatim

import "pose.proto"; import "camerasensor.proto"; import "raysensor.proto"; import "contactsensor.proto";

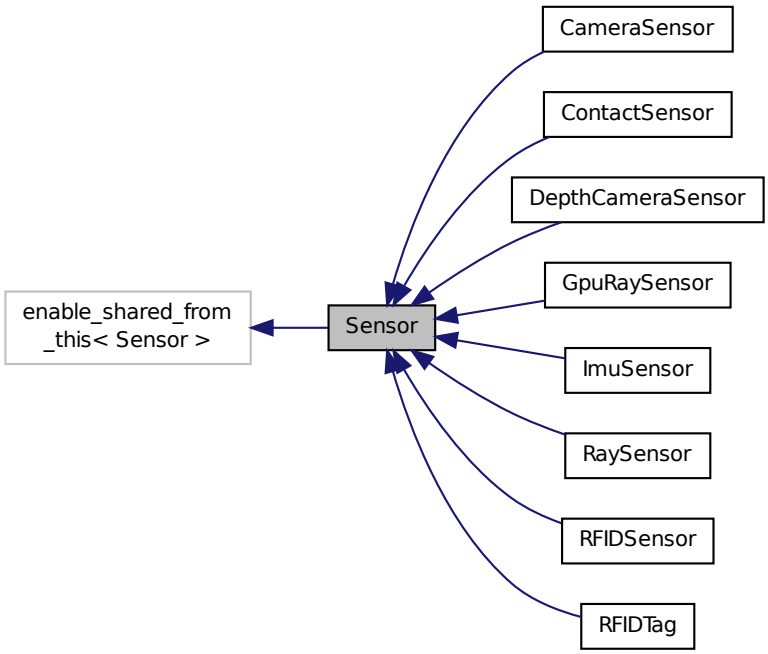
message **Sensor** (p. 328) { required string name = 1; required string parent = 2; required string type = 3; optional bool always_on = 4; optional double update_rate = 5; optional **Pose** (p. 283) pose = 6; optional **CameraSensor** (p. 107) camera = 7; optional **RaySensor** (p. 308) ray = 8; optional **ContactSensor** (p. 124) contact = 9; optional bool visualize = 10; optional string topic = 11; } ///

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

- sensor.proto

9.182 Sensor Class Reference

Inheritance diagram for Sensor:



Public Member Functions

- **Sensor** ()

  Constructor.

- virtual ~**Sensor** ()

  Destructor.
- 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 **Init** ()

  Initialize the sensor.
- virtual void **SetParent** (const std::string &_name)

  Set the parent of the sensor.
- std::string **GetParentName** () const
- void **Update** (bool _force)

  Update the sensor.
- void **SetUpdateRate** (double _hz)

  Set the update rate of the sensor.
- virtual void **Fini** ()

  Finalize the sensor.
- std::string **GetName** () const

  Get name.
- virtual **math::Pose GetPose** () const

  Get the current pose.
- virtual void **SetActive** (bool value)

  Set whether the sensor is active or not.
- bool **IsActive** ()
- std::string **GetType** () const

  Get sensor type.
- **common::Time GetLastUpdateTime** ()

  return last update time
- bool **GetVisualize** () const
- virtual std::string **GetTopic** () const
- void **FillMsg** (msgs::Sensor &_msg)
- std::string **GetWorldName** () const

Protected Member Functions

- virtual void **UpdateImpl** (bool)

Protected Attributes

- bool **active**

  True if active.
- sdf::ElementPtr **sdf**
- **math::Pose pose**
- std::vector< event::ConnectionPtr > **connections**
- transport::NodePtr **node**
- transport::SubscriberPtr **poseSub**
- std::string **parentName**

- std::vector< SensorPluginPtr > **plugins**
- gazebo::physics::WorldPtr **world**
- **common::Time updatePeriod**
- **common::Time lastUpdateTime**

9.182.1 Member Function Documentation

9.182.1.1 virtual void Load (const std::string & *_worldName,* sdf::ElementPtr *_sdf*) [virtual]

Load the sensor with SDF parameters.

Parameters

_sdf	SDF **Sensor** (p. 329) parameteres

Reimplemented in **RaySensor** (p. 308), **ContactSensor** (p. 123), **CameraSensor** (p. 107), and **RFIDSensor** (p. 315).

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

- Sensor.hh

9.183 SensorFactory Class Reference

Static Public Member Functions

- static void **RegisterAll** ()
- static void **RegisterSensor** (const std::string &classname, SensorFactoryFn factoryfn)

  Register a sensor class (called by sensor registration function).
- static SensorPtr **NewSensor** (const std::string &classname)

  Create a new instance of a sensor.
- static void **GetSensorTypes** (std::vector< std::string > &_types)

  Get all the sensor types.

9.183.1 Member Function Documentation

9.183.1.1 static SensorPtr NewSensor (const std::string & *classname*) [static]

Create a new instance of a sensor.

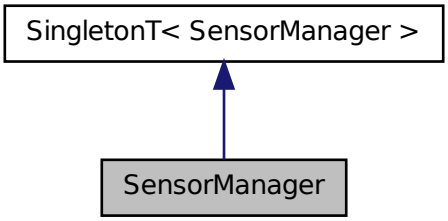
Used by the world when reading the world file.

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

- SensorFactory.hh

9.184 SensorManager Class Reference

Inheritance diagram for SensorManager:



Public Member Functions

- **SensorManager** ()

  Constructor.
- virtual ∼**SensorManager** ()

  Destructor.
- void **Update** (bool force=false)

  Update all the sensors.
- void **Init** ()

  Init all the sensors.
- void **Run** ()

  Run the sensor manager update in a new thread.
- void **Stop** ()

  Stop the run thread.
- void **Fini** ()

  Finalize all the sensors.
- void **GetSensorTypes** (std::vector< std::string > &_types) const

  Get all the sensor types.
- std::string **CreateSensor** (sdf::ElementPtr _elem, const std::string &_worldName, const std::string &_parentName)

  Add a sensor from an SDF element.
- SensorPtr **GetSensor** (const std::string &_name)

  Get a sensor.
- void **RemoveSensor** (const std::string &_name)

  Remove a sensor.
- void **RemoveSensors** ()

  Remove all sensors.

Additional Inherited Members

9.184.1 Member Function Documentation

9.184.1.1 std::string 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

_elem	The SDF element that describes the sensor
_worldName	Name of the world in which to create the sensor
_parentName	The name of the parent link which the sensor is attached to.

Returns

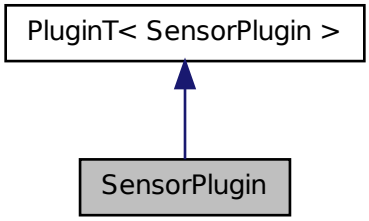
The name of the sensor

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

- SensorManager.hh

9.185 SensorPlugin Class Reference

Inheritance diagram for SensorPlugin:



Public Member Functions

- virtual void **Load** (sensors::SensorPtr _sensor, sdf::ElementPtr _sdf)=0

  Load function.
- virtual void **Init** ()
- virtual void **Reset** ()

Additional Inherited Members

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

- common/Plugin.hh

9.186 Server Class Reference

Public Member Functions

- void **PrintUsage** ()
- bool **ParseArgs** (int argc, char ∗∗argv)
- bool **Load** (const std::string &_filename="worlds/empty.world")
- void **Init** ()
- void **Run** ()
- void **Stop** ()
- void **Fini** ()
- void **SetParams** (const common::StrStr_M &params)
- bool **GetInitialized** () const

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

- Server.hh

9.187 ServerControl Interface Reference

A message that allows for control of the server functions.

9.187.1 Detailed Description

A message that allows for control of the server functions.

```
@verbatim
```

import "header.proto";

message **ServerControl** (p. 334) { optional string save_world_name = 1; optional string save_filename = 2; optional string open_filename = 3; optional bool new_world = 4; }

///

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

- server_control.proto

9.188 Shadows Interface Reference

A message for shadow data.

9.188.1 Detailed Description

A message for shadow data.

```
@verbatim
```

import "color.proto";

message **Shadows** (p. 334) { enum ShadowType { STENCIL_ADDITIVE = 1; STENCIL_MODULATIVE = 2; TEXTURE_ADDITIVE = 3; TEXTURE_MODULATIVE = 4; } optional ShadowType type = 5; optional **Color** (p. 115) color = 6; }

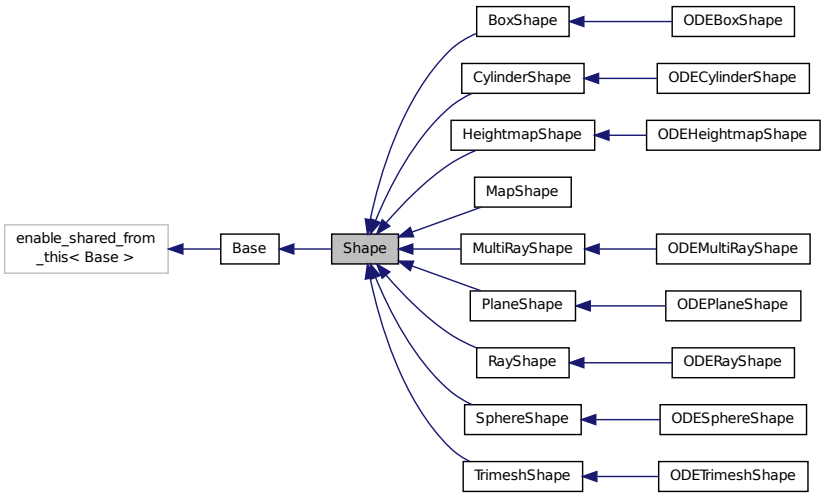
///

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

- shadows.proto

9.189 Shape Class Reference

Inheritance diagram for Shape:



Public Member Functions

- **Shape** (CollisionPtr p)

  Constructor.
- virtual ~**Shape** ()

  Destructor.
- virtual void **Init** ()=0

  Initialize the shape.
- virtual double **GetMass** (double _density) const

Get the mass of a shape.

- virtual void **GetInertial** (double _mass, InertialPtr _inertial) const

Get inertial for a shape.

- virtual void **FillShapeMsg** (msgs::Geometry &_msg)=0

- virtual void **ProcessMsg** (const msgs::Geometry &_msg)=0

Public Attributes

- CollisionPtr **collisionParent**

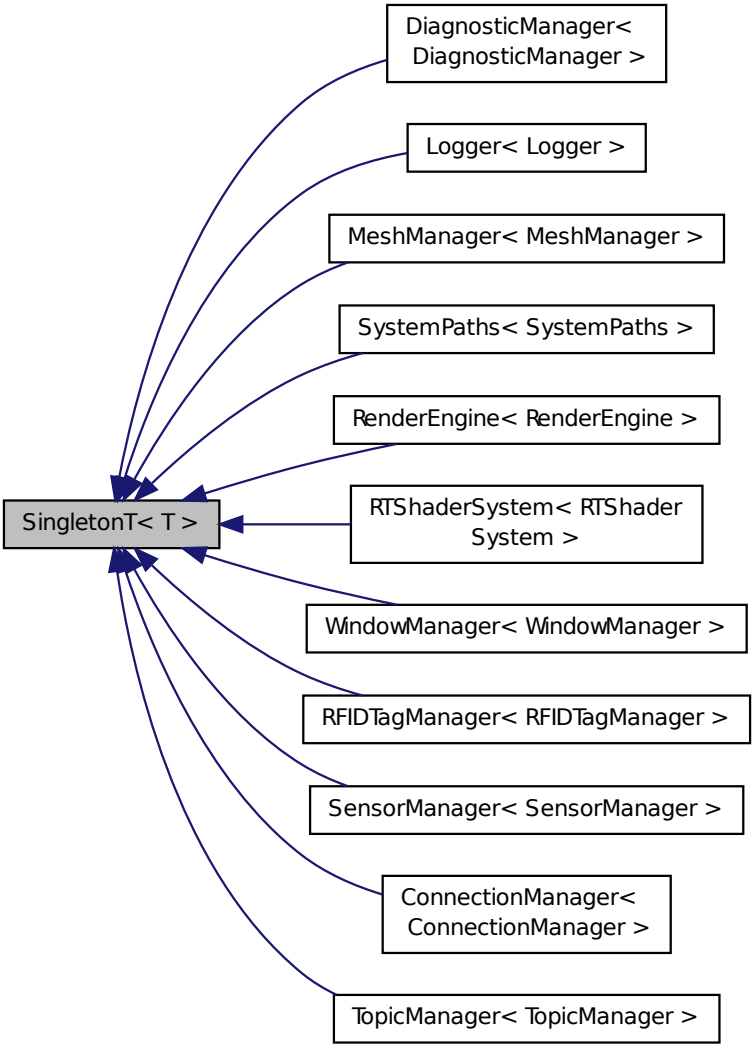
Additional Inherited Members

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

- Shape.hh

9.190 SingletonT< T > Class Template Reference

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.

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

- SingletonT.hh

9.191 Skeleton Class Reference

Public Member Functions

- **Skeleton** ()

  Constructor.
- **Skeleton** (**SkeletonNode** *_root)
- virtual ~**Skeleton** ()

  Destructor.
- void **SetRootNode** (**SkeletonNode** *_node)
- **SkeletonNode** * **GetRootNode** ()
- **SkeletonNode** * **GetNodeByName** (std::string _name)
- **SkeletonNode** * **GetNodeById** (std::string _id)
- **SkeletonNode** * **GetNodeByHandle** (unsigned int _handle)
- unsigned int **GetNumNodes** ()
- unsigned int **GetNumJoints** ()
- void **Scale** (double _scale)
- void **SetBindShapeTransform** (**math::Matrix4** _trans)
- **math::Matrix4 GetBindShapeTransform** ()
- void **PrintTransforms** ()
- NodeMap **GetNodes** ()
- void **SetNumVertAttached** (unsigned int _vertices)
- void **AddVertNodeWeight** (unsigned int _vertex, std::string _node, double _weight)
- unsigned int **GetNumVertNodeWeights** (unsigned int _vertex)
- std::pair< std::string, double > **GetVertNodeWeight** (unsigned int _v, unsigned int _i)
- unsigned int **GetNumAnimations** ()
- **SkeletonAnimation** * **GetAnimation** (const unsigned int i)
- void **AddAnimation** (**SkeletonAnimation** *_anim)

Protected Member Functions

- void **BuildNodeMap** ()

Protected Attributes

- **SkeletonNode** * **root**
- NodeMap **nodes**
- **math::Matrix4 bindShapeTransform**
- RawNodeWeights **rawNW**
- std::vector< **SkeletonAnimation** * > **anims**

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

- Skeleton.hh

9.192 SkeletonAnimation Class Reference

Public Member Functions

- **SkeletonAnimation** (const std::string &_name)
- void **SetName** (const std::string &_name)
- std::string **GetName** () const
- unsigned int **GetNodeCount** () const
- bool **HasNode** (const std::string &_node) const
- void **AddKeyFrame** (const std::string &_node, const double _time, const **math::Matrix4** _mat)
- void **AddKeyFrame** (const std::string &_node, const double _time, const **math::Pose** _pose)
- **math::Matrix4 GetNodePoseAt** (const std::string &_node, const double _time, const bool _loop=true)
- std::map< std::string, **math::Matrix4** > **GetPoseAt** (const double _time, const bool _loop=true) const
- std::map< std::string, **math::Matrix4** > **GetPoseAtX** (const double _x, const std::string &_node, const bool _loop=true) const
- void **Scale** (const double _scale)
- double **GetLength** () const

Protected Attributes

- std::string **name**
- double **length**
- std::map< std::string, **NodeAnimation** * > **animations**

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

- SkeletonAnimation.hh

9.193 SkeletonNode Class Reference

A node.

`#include <Skeleton.hh>`

Public Types

- enum **SkeletonNodeType** { **NODE**, **JOINT** }

Public Member Functions

- **SkeletonNode** (**SkeletonNode** *_parent)

  Constructor.
- **SkeletonNode** (**SkeletonNode** *_parent, std::string _name, std::string _id, SkeletonNodeType _type=JOINT)
- virtual ~**SkeletonNode** ()

  Destructor.
- void **SetName** (std::string _name)
- std::string **GetName** ()
- void **SetId** (std::string _id)
- std::string **GetId** ()
- void **SetType** (SkeletonNodeType _type)
- bool **IsJoint** ()
- void **SetTransform** (**math::Matrix4** _trans, bool _updateChildren=true)
- void **SetModelTransform** (**math::Matrix4** _trans, bool _updateChildren=true)
- void **UpdateChildrenTransforms** ()
- void **SetInitialTransform** (**math::Matrix4** _tras)
- void **Reset** (bool resetChildren)
- **math::Matrix4 GetTransform** ()

  Get transform relative to parent.
- void **SetParent** (**SkeletonNode** *_parent)
- **SkeletonNode** * **GetParent** ()
- bool **IsRootNode** ()
- void **AddChild** (**SkeletonNode** *_child)
- unsigned int **GetChildCount** ()
- **SkeletonNode** * **GetChild** (unsigned int _index)
- **SkeletonNode** * **GetChildByName** (std::string _name)

  Get child by name.
- **SkeletonNode** * **GetChildById** (std::string _id)

  Get child by id.
- void **SetHandle** (unsigned int _h)
- unsigned int **GetHandle** ()
- void **SetInverseBindTransform** (**math::Matrix4** _invBM)
- **math::Matrix4 GetInverseBindTransform** ()
- **math::Matrix4 GetModelTransform** ()
- std::vector< **NodeTransform** > **GetRawTransforms** ()
- unsigned int **GetNumRawTrans** ()
- **NodeTransform GetRawTransform** (unsigned int _i)
- void **AddRawTransform** (**NodeTransform** _t)
- std::vector< **NodeTransform** > **GetTransforms** ()

Protected Attributes

- std::string **name**
- std::string **id**
- SkeletonNodeType **type**
- **math::Matrix4 transform**
- **math::Matrix4 initialTransform**
- **math::Matrix4 modelTransform**
- **math::Matrix4 invBindTransform**

- **SkeletonNode** * **parent**
- std::vector< **SkeletonNode** * > **children**
- unsigned int **handle**
- std::vector< **NodeTransform** > **rawTransforms**

9.193.1 Detailed Description

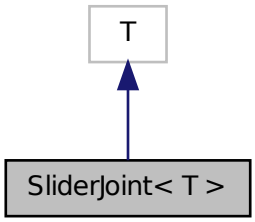
A node.

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

- Skeleton.hh

9.194 SliderJoint< T > Class Template Reference

Inheritance diagram for SliderJoint< T >:



Public Member Functions

- **SliderJoint** ()

  Constructor.
- virtual ~**SliderJoint** ()

  Destructor.
- virtual void **SetAnchor** (int, const **math::Vector3** &_anchor)

  Set the anchor.
- virtual **math::Vector3 GetAnchor** (int) const

  Get the anchor.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

  Load the joint.

Protected Attributes

- **math::Vector3 fakeAnchor**

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

- SliderJoint.hh

9.195 SphereGeom Interface Reference

Information about a sphere geometry.

9.195.1 Detailed Description

Information about a sphere geometry.

```
@verbatim
```

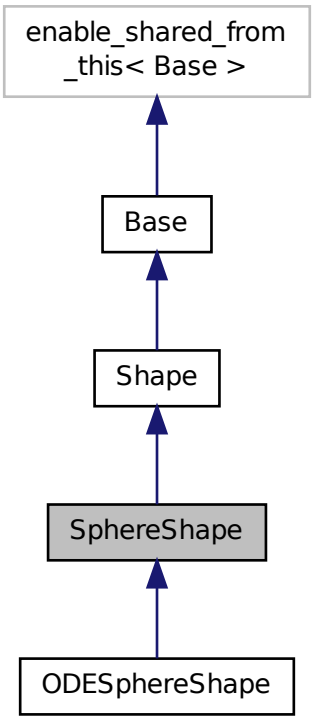
message **SphereGeom** (p. 342) { required double radius = 1; } ///

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

- spheregeom.proto

9.196 SphereShape Class Reference

Inheritance diagram for SphereShape:



Public Member Functions

- **SphereShape** (CollisionPtr **parent**)
  Constructor.
- virtual ~**SphereShape** ()
  Destructor.
- virtual void **Init** ()
  Initialize the sphere.
- virtual void **SetRadius** (const double &radius)
  Set the size.
- double **GetRadius** () const
- virtual void **FillShapeMsg** (msgs::Geometry &_msg)
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)
- virtual double **GetMass** (double _density) const
  Get the mass of a shape.
- virtual void **GetInertial** (double _mass, InertialPtr _inertial) const
  Get inertial for a shape.

Additional Inherited Members

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

- SphereShape.hh

9.197 Spline Class Reference

Public Member Functions

- void **AddPoint** (const **Vector3** &_pt)

  Adds a control point to the end of the spline.
- const **Vector3** & **GetPoint** (unsigned int _index) const

  Gets the detail of one of the control points of the spline.
- unsigned int **GetNumPoints** () const

  Gets the number of control points in the spline.
- void **Clear** ()

  Clears all the points in the spline.
- void **UpdatePoint** (unsigned int _index, const **Vector3** &_value)

  Updates a single point in the spline.
- **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 **SetAutoCalculate** (bool _autoCalc)

  Tells the spline whether it should automatically calculate tangents on demand as points are added.
- void **RecalcTangents** ()

  Recalculates the tangents associated with this spline.

Protected Attributes

- bool **autoCalc**
- std::vector< **Vector3** > **points**
- std::vector< **Vector3** > **tangents**
- **Matrix4 coeffs**

  Matrix of coefficients.

9.197.1 Member Function Documentation

9.197.1.1 Vector3 Interpolate (double _t) const

Returns an interpolated point based on a parametric value over the whole series.

9.197.1.2 Vector3 Interpolate (unsigned int _fromIndex, double _t) const

Interpolates a single segment of the spline given a parametric value.

Parameters

_fromIndex	The point index to treat as t = 0. fromIndex + 1 is deemed to be t = 1
_t	Parametric value

9.197.1.3 void 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.

9.197.1.4 void 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

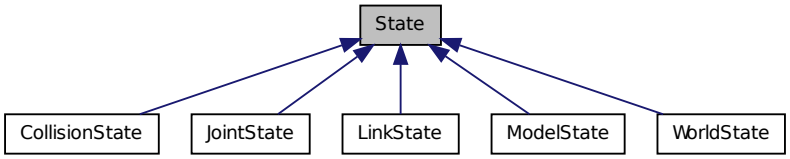
_autoCalc	If true, tangents are calculated for you whenever a point changes. If false, you must call reclacTangents to recalculate them when it best suits.

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

- Spline.hh

9.198 State Class Reference

Inheritance diagram for State:



Public Member Functions

- **State** ()

  Default constructor.
- **State** (const std::string &_name, const **common::Time** &_realTime, const **common::Time** &_simTime)

  Constructor.
- virtual ~**State** ()

  Destructor.
- virtual void **Load** (sdf::ElementPtr _elem)=0

  Load state from SDF element.
- std::string **GetName** () const

  Get the name of the state.
- **common::Time GetWallTime** () const

  Get the wall time when this state was generated.
- **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.

Protected Attributes

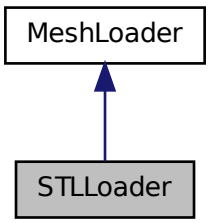
- std::string **name**
- **common::Time wallTime**
- **common::Time realTime**
- **common::Time simTime**

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

- State.hh

9.199 STLLoader Class Reference

Inheritance diagram for STLLoader:



Public Member Functions

- **STLLoader** ()
  Constructor.
- virtual ∼**STLLoader** ()
  Destructor.
- virtual **Mesh** ∗ **Load** (const std::string &filename)
  Load a mesh.

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

- STLLoader.hh

9.200 String Interface Reference

A message for string data.

9.200.1 Detailed Description

A message for string data.

A message for a vector of string data.

```
@verbatim
```

message **String** (p. 347) { required string data = 1; }

///

```
@verbatim
```

message String_V { repeated string data = 1; }

///

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

- string.proto

9.201 SubMesh Class Reference

A child mesh.

`#include <Mesh.hh>`

Public Types

- enum **PrimitiveType** {
  POINTS, **LINES**, **LINESTRIPS**, **TRIANGLES**,
  TRIFANS, **TRISTRIPS** }

Public Member Functions

- **SubMesh** ()
  Constructor.
- virtual ~**SubMesh** ()
  Destructor.
- void **SetPrimitiveType** (PrimitiveType _type)
  Set the primitive type.
- PrimitiveType **GetPrimitiveType** () const
  Get the primitive type.
- void **CopyVertices** (const std::vector< **math::Vector3** > &_verts)
  Copy vertices from a vector.
- void **CopyNormals** (const std::vector< **math::Vector3** > &_norms)
  Copy normals from a vector.
- void **SetVertexCount** (unsigned int _count)
  Resize the vertex array.
- void **SetIndexCount** (unsigned int _count)
  Resize the index array.
- void **SetNormalCount** (unsigned int _count)
  Resize the normal array.
- void **SetTexCoordCount** (unsigned int _count)
  Resize the texture coordinate array.
- void **AddIndex** (unsigned int _i)
  Add an index 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 **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 **AddNodeAssignment** (unsigned int _vertex, unsigned int _node, float _weight)

  Add a vertex - skeleton node assignment.
- **math::Vector3 GetVertex** (unsigned int _i) const

  Get a vertex.
- void **SetVertex** (unsigned int _i, const **math::Vector3** &_v)

  Set a vertex.
- **math::Vector3 GetNormal** (unsigned int _i) const

  Get a normal.
- void **SetNormal** (unsigned int _i, const **math::Vector3** &_n)

  Set a normal.
- **math::Vector2d GetTexCoord** (unsigned int _i) const

  Get a tex coord.
- **NodeAssignment GetNodeAssignment** (unsigned int _i) const

  Get a vertex - skeleton node assignment.
- void **SetTexCoord** (unsigned int _i, const **math::Vector2d** &_t)

  Set a tex coord.
- unsigned int **GetIndex** (unsigned int _i) const

  Get an index.
- **math::Vector3 GetMax** () const

  Get the maximun X, Y, Z values.
- **math::Vector3 GetMin** () const

  Get the minimum X, Y, Z values.
- unsigned int **GetVertexCount** () const

  Return the number of vertices.
- unsigned int **GetNormalCount** () const

  Return the number of normals.
- unsigned int **GetIndexCount** () const

  Return the number of indicies.
- unsigned int **GetTexCoordCount** () const

  Return the number of texture coordinates.
- unsigned int **GetNodeAssignmentsCount** () const

  Return the number of vertex - skeleton node assignments.
- unsigned int **GetMaxIndex** () const

  Get the highest index value.
- void **SetMaterialIndex** (unsigned int _index)

  Set the material index.
- unsigned int **GetMaterialIndex** () const

  Get the material index.
- bool **HasVertex** (const **math::Vector3** &_v) const

  Return true if this submesh has the vertex.
- unsigned int **GetVertexIndex** (const **math::Vector3** &_v) const

Get the index of the vertex.

- void **FillArrays** (float **_vertArr, int **_indArr) const

  Put all the data into flat arrays.

- void **RecalculateNormals** ()

  Recalculate all the normals.

- void **SetSubMeshCenter** (**math::Vector3** _center)

  Reset mesh center to geometric center.

- void **GenSphericalTexCoord** (const **math::Vector3** &_center)

  Generate texture coordinates using spherical projection from center.

9.201.1 Detailed Description

A child mesh.

9.201.2 Member Function Documentation

9.201.2.1 void SetMaterialIndex (unsigned int *_index*)

Set the material index.

Relates to the parent mesh material list

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

- Mesh.hh

9.202 Subscribe Interface Reference

A message for subscription data.

9.202.1 Detailed Description

A message for subscription data.

```
@verbatim
```

message **Subscribe** (p. 350) { required string topic = 1; required string host = 2; required uint32 port = 3; required string msg_type = 4; optional bool latching = 5 [default=true]; }

///

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

- subscribe.proto

9.203 SubscribeOptions Class Reference

Public Member Functions

- template<class M >
  void **Init** (const std::string &_topic, NodePtr _node, bool _latching)
- NodePtr **GetNode** () const
- std::string **GetTopic** () const
- std::string **GetMsgType** () const
- bool **GetLatching** () const

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

- SubscribeOptions.hh

9.204 Subscriber Class Reference

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.

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

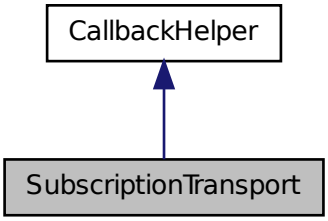
- Subscriber.hh

9.205 SubscriptionTransport Class Reference

Handles sending data over the wire to remote subscribers.

`#include <SubscriptionTransport.hh>`

Inheritance diagram for SubscriptionTransport:



Public Member Functions

- **SubscriptionTransport** ()

  Constructor.
- virtual ~**SubscriptionTransport** ()

  Destructor.
- void **Init** (const ConnectionPtr &conn, bool _latching)

  Initialize the publication link.
- virtual bool **HandleData** (const std::string &newdata)

  Output a message to a connection.
- const ConnectionPtr & **GetConnection** () const

  Get the connection.
- virtual bool **IsLocal** () const

  Return true if the callback is local, false if the callback is tied to a remote connection.

Additional Inherited Members

9.205.1 Detailed Description

Handles sending data over the wire to remote subscribers.

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

- SubscriptionTransport.hh

9.206 Surface Interface Reference

Information about a surface element.

9.206.1 Detailed Description

Information about a surface element.

```
@verbatim
```

import "friction.proto";

message **Surface** (p. 352) { optional Friction friction = 1; optional double restitution_coefficient = 2; optional double bounce_threshold = 3; optional double soft_cfm = 4; optional double soft_erp = 5; optional double kp = 6; optional double kd = 7; optional double max_vel = 8; optional double min_depth = 9; } ///

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

- surface.proto

9.207 SurfaceParams Class Reference

Public Member Functions

- **SurfaceParams** ()
  Constructor.
- virtual ~**SurfaceParams** ()
  Constructor.
- virtual void **Load** (sdf::ElementPtr _sdf)
  Load the contact params.
- void **FillSurfaceMsg** (msgs::Surface &_msg)
  Fill in a surface message.
- virtual void **ProcessMsg** (const msgs::Surface &_msg)

Public Attributes

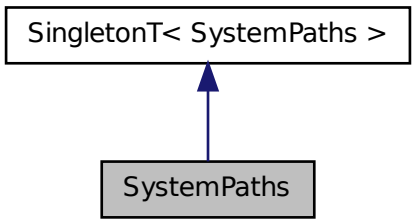
- double **bounce**
  0..1, 0 = no bounciness
- double **bounceThreshold**
  bounce vel
- double **kp**
- double **kd**
- double **cfm**
- double **erp**
- double **maxVel**
- double **minDepth**
- double **mu1**
- double **mu2**
- double **slip1**
- double **slip2**
- **math::Vector3 fdir1**

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

- SurfaceParams.hh

9.208 SystemPaths Class Reference

Inheritance diagram for SystemPaths:



Public Member Functions

- std::string **GetLogPath** () const
  Get the log path.
- const std::list< std::string > & **GetGazeboPaths** ()
  Get the gazebo install paths.
- const std::list< std::string > & **GetOgrePaths** ()
  Get the ogre install paths.
- const std::list< std::string > & **GetPluginPaths** ()
  Get the plugin paths.
- std::string **GetModelPathExtension** ()
  Get the model path extension.
- std::string **GetWorldPathExtension** ()
  Get the world path extension.
- std::string **FindFileWithGazeboPaths** (const std::string &_filename)
- void **AddGazeboPaths** (std::string _path)
  Add colon delimited paths to Gazebo install.
- void **AddOgrePaths** (std::string _path)
  Add colon delimited paths to ogre install.
- void **AddPluginPaths** (std::string _path)
  Add colon delimited paths to plugins.
- void **ClearGazeboPaths** ()
- void **ClearOgrePaths** ()
- void **ClearPluginPaths** ()

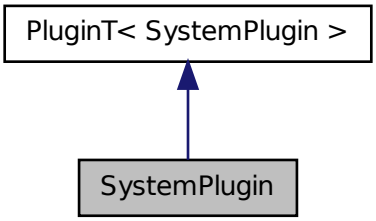
Additional Inherited Members

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

- SystemPaths.hh

9.209 SystemPlugin Class Reference

Inheritance diagram for SystemPlugin:



Public Member Functions

- virtual void **Load** ()=0
  Load function.
- virtual void **Init** ()
- virtual void **Reset** ()

Additional Inherited Members

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

- common/Plugin.hh

9.210 Time Class Reference

Public Member Functions

- **Time** ()
  Constructors.
- **Time** (const **Time** &_time)
  Copy constructor.
- **Time** (const struct timeval &_tv)
  Constructor.
- **Time** (int32_t _sec, int32_t _nsec)
  Constructor.
- **Time** (double _time)
  Constuctor.
- virtual ~**Time** ()
  Destructor.

- void **SetToWallTime** ()

  Set the time to the wall time.
- void **Set** (int32_t _sec, int32_t _nsec)

  Set to sec and nsec.
- void **Set** (double _seconds)

  Set to seconds.
- double **Double** () const

  Get the time as a double.
- float **Float** () const

  Get the time as a float.
- **Time** & **operator=** (const struct timeval &tv)

  Equal opeator.
- **Time** & **operator=** (const **Time** &time)

  Equal opeator.
- **Time operator+** (const struct timeval &tv) const

  Addition operators.
- const **Time** & **operator+=** (const struct timeval &tv)

  Addition operators.
- **Time operator+** (const **Time** &time) const

  Addition operators.
- const **Time** & **operator+=** (const **Time** &time)

  Addition operators.
- **Time operator-** (const struct timeval &tv) const

  Subtraction operator.
- const **Time** & **operator-=** (const struct timeval &tv)

  Subtraction operator.
- **Time operator-** (const **Time** &time) const

  Subtraction operator.
- const **Time** & **operator-=** (const **Time** &time)

  Subtraction operator.
- **Time operator**∗ (const struct timeval &tv) const

  Multiplication operators.
- const **Time** & **operator**∗**=** (const struct timeval &tv)

  Multiplication operators.
- **Time operator**∗ (const **Time** &time) const

  Multiplication operators.
- const **Time** & **operator**∗**=** (const **Time** &time)

  Multiplication operators.
- **Time operator**/ (const struct timeval &tv) const

  Division operators.
- const **Time** & **operator**/**=** (const struct timeval &tv)

  Division operators.
- **Time operator**/ (const **Time** &time) const

  Division operators.
- const **Time** & **operator**/**=** (const **Time** &time)

  Division operators.
- bool **operator==** (const struct timeval &tv) const

Equality operators.

- bool **operator==** (const **Time** &time) const

  Equality operators.
- bool **operator==** (double time) const

  Equality operators.
- bool **operator!=** (const struct timeval &tv) const

  Equality operators.
- bool **operator!=** (const **Time** &time) const

  Equality operators.
- bool **operator!=** (double time) const

  Equality operators.
- bool **operator**< (const struct timeval &tv) const

  Equality operators.
- bool **operator**< (const **Time** &time) const

  Equality operators.
- bool **operator**< (double time) const

  Equality operators.
- bool **operator**<**=** (const struct timeval &tv) const

  Equality operators.
- bool **operator**<**=** (const **Time** &time) const

  Equality operators.
- bool **operator**<**=** (double time) const

  Equality operators.
- bool **operator**> (const struct timeval &tv) const

  Equality operators.
- bool **operator**> (const **Time** &time) const

  Equality operators.
- bool **operator**> (double time) const

  Equality operators.
- bool **operator**>**=** (const struct timeval &tv) const

  Equality operators.
- bool **operator**>**=** (const **Time** &time) const

  Equality operators.
- bool **operator**>**=** (double time) const

  Equality operators.

Static Public Member Functions

- static const **Time** & **GetWallTime** ()

  Get the wall time.
- static **Time MSleep** (unsigned int _ms)

  Millisecond sleep.
- static double **SecToNano** (double _sec)

  Convert seconds to nanoseconds.
- static double **MilToNano** (double _ms)

  Convert milliseconds to nanoseconds.
- static double **MicToNano** (double _ms)

  Convert microseconds to nanoseconds.

Public Attributes

- int32_t **sec**

  Seconds.
- int32_t **nsec**

  Microseconds.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::common::Time** &_time)

  Stream operators.
- std::istream & **operator**>> (std::istream &_in, **gazebo::common::Time** &_time)

9.210.1 Constructor & Destructor Documentation

9.210.1.1 Time (const Time & *_time*)

Copy constructor.

Parameters

time	**Time** (p. 355) to copy

9.210.1.2 Time (const struct timeval & *_tv*)

Constructor.

Parameters

tv	**Time** (p. 355) to initialize to

9.210.1.3 Time (int32_t *_sec*, int32_t *_nsec*)

Constructor.

Parameters

sec	Seconds
nsec	Microseconds

9.210.1.4 Time (double *_time*)

Constuctor.

Parameters

time	**Time** (p. 355) in double format sec.nsec

9.210.2 Member Function Documentation

9.210.2.1 double Double ( ) const

Get the time as a double.

Returns

Time (p. 355) as a double

9.210.2.2 float Float ( ) const

Get the time as a float.

Returns

Time (p. 355) as a float

9.210.2.3 void Set (int32_t *_sec*, int32_t *_nsec*)

Set to sec and nsec.

Parameters

sec	Seconds
nsec	micro seconds

9.210.2.4 void Set (double *_seconds*)

Set to seconds.

Parameters

seconds	Number of seconds

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

- Time.hh

9.211 Time Interface Reference

A message for time data.

9.211.1 Detailed Description

A message for time data.

```
@verbatim
```

message **Time** (p. 359) { required int32 sec = 1; required int32 nsec = 2; }

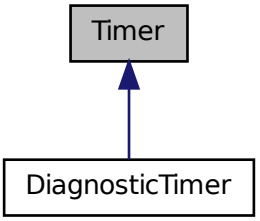
///

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

- time.proto

9.212 Timer Class Reference

Inheritance diagram for Timer:



Public Member Functions

- **Timer** ()
  Constructor.
- virtual ~**Timer** ()
  Destructor.
- void **Start** ()
  Start the timer.
- **Time GetElapsed** () const
  Get the elapsed itme.

Friends

- std::ostream & **operator**<< (std::ostream &out, const **gazebo::common::Timer** &t)

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

- Timer.hh

9.213 Topic Interface Reference

A message for topic information.

9.213.1 Detailed Description

A message for topic information.

@verbatim

import "publish.proto"; import "subscribe.proto";

message TopicInfo { required string msg_type = 1; repeated **Publish** (p. 293) publisher = 2; repeated **Subscribe** (p. 350) subscriber = 3; }

///

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

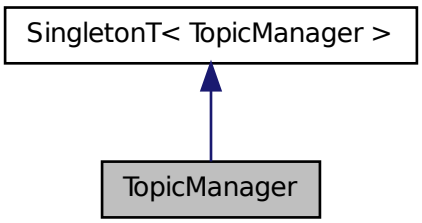
- topic_info.proto

9.214 TopicManager Class Reference

Manages topics and their subscriptions.

`#include <TopicManager.hh>`

Inheritance diagram for TopicManager:



Public Types

- typedef std::map< std::string, std::list< NodePtr > > **SubNodeMap**

Public Member Functions

- void **Init** ()
- void **Fini** ()
- PublicationPtr **FindPublication** (const std::string &topic)
- void **AddNode** (NodePtr _node)
- void **RemoveNode** (unsigned int _id)

- void **ProcessNodes** ()
- bool **IsAdvertised** (const std::string &_topic)

  Returns true if the topic has been advertised.
- SubscriberPtr **Subscribe** (const **SubscribeOptions** &options)

  Subscribe *(p. 350) to a topic.*
- void **Unsubscribe** (const std::string &_topic, const NodePtr &_sub)

  Unsubscribe from a topic.
- template<typename M >
  PublisherPtr **Advertise** (const std::string &_topic, unsigned int _queueLimit, bool _latch)

  Advertise on a topic.
- void **Unadvertise** (const std::string &topic)

  Stop advertising on a topic.
- void **Publish** (const std::string &topic, const google::protobuf::Message &message, const boost::function< void()> &cb=NULL)

  Send a message.
- void **ConnectPubToSub** (const std::string &topic, const SubscriptionTransportPtr &sublink)

  Connection *(p. 117) a local* ***Publisher*** *(p. 293) to a remote* ***Subscriber*** *(p. 351).*
- void **ConnectSubToPub** (const msgs::Publish &_pub)

  Connect a local ***Subscriber*** *(p. 351) to a remote* ***Publisher*** *(p. 293).*
- 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.
- void **ConnectSubscribers** (const std::string &topic)

  Connect all subscribers on a topic to known publishers.
- PublicationPtr **UpdatePublications** (const std::string &topic, const std::string &msgType)

  Update our list of advertised topics.
- void **RegisterTopicNamespace** (const std::string &_name)

  Register a new topic namespace.
- void **GetTopicNamespaces** (std::list< std::string > &_namespaces)

  Get all the topic namespaces.
- void **ClearBuffers** ()
- void **PauseIncoming** (bool _pause)

Additional Inherited Members

9.214.1 Detailed Description

Manages topics and their subscriptions.

9.214.2 Member Function Documentation

9.214.2.1 PublisherPtr Advertise (const std::string & *_topic,* unsigned int *_queueLimit,* bool *_latch*) [inline]

Advertise on a topic.

Parameters

topic	The name of the topic

References SingletonT< T >::Instance(), and TopicManager::UpdatePublications().

9.214.2.2 bool 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

9.214.2.3 void Publish (const std::string & *topic*, const google::protobuf::Message & *message*, const boost::function< void() > & *cb* = NULL)

Send a message.

Use a **Publisher** (p. 293) 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.

9.214.2.4 void Unsubscribe (const std::string & *_topic*, const NodePtr & *_sub*)

Unsubscribe from a topic.

Use a **Subscriber** (p. 351) rather than calling this function directly

9.214.2.5 PublicationPtr 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 TopicManager::Advertise().

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

- TopicManager.hh

9.215 Track Interface Reference

in the **GUI** (p. 168))

9.215.1 Detailed Description

in the **GUI** (p. 168))
Message for a trackking a visual with a camera

```
message TrackVisual
{
  required string name              = 1;
  optional bool inherit_orientation = 2;
  optional double min_dist          = 3;
  optional double max_dist          = 4;
}

///
```

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

- track_visual.proto

9.216 TrajectoryInfo Struct Reference

Public Attributes

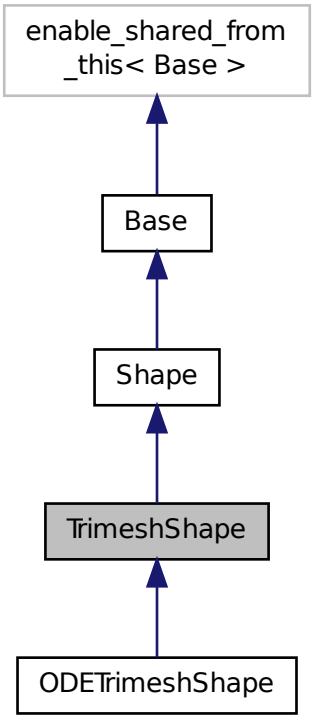
- unsigned int **id**
- std::string **type**
- double **duration**
- double **startTime**
- double **endTime**

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

- Actor.hh

9.217 TrimeshShape Class Reference

Inheritance diagram for TrimeshShape:



Public Member Functions

- **TrimeshShape** (CollisionPtr **parent**)

  Constructor.
- virtual ~**TrimeshShape** ()

  Destructor.
- virtual void **Update** ()
- virtual void **Init** ()

  Init the trimesh shape.
- virtual **math::Vector3 GetSize** () const
- std::string **GetFilename** () const

  Get the filename of the mesh data.
- void **SetFilename** (const std::string &_filename)
- void **SetScale** (const **math::Vector3** &_scale)
- void **FillShapeMsg** (msgs::Geometry &_msg)
- virtual void **ProcessMsg** (const msgs::Geometry &_msg)

- virtual double **GetMass** (double _density) const

  Get the mass of a shape.

Protected Attributes

- const **common::Mesh** ∗ **mesh**

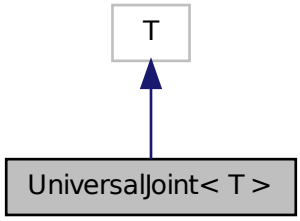
Additional Inherited Members

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

- TrimeshShape.hh

9.218 UniversalJoint< T > Class Template Reference

Inheritance diagram for UniversalJoint< T >:



Public Member Functions

- **UniversalJoint** ()

  Constructor.
- virtual ∼**UniversalJoint** ()

  Destuctor.

Protected Member Functions

- virtual void **Load** (sdf::ElementPtr _sdf)

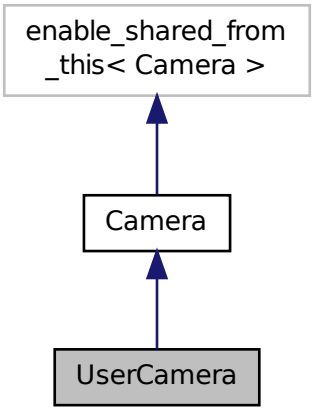
  Load the joint.

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

- UniversalJoint.hh

9.219 UserCamera Class Reference

Inheritance diagram for UserCamera:



Public Member Functions

- **UserCamera** (const std::string &_name, **Scene** *_scene)

  Constructor.
- virtual ~**UserCamera** ()

  Destructor.
- void **Load** (sdf::ElementPtr _sdf)

  Load the user camera.
- void **Load** ()
- void **Init** ()

  Initialize.
- virtual void **Update** ()

  Render the camera.
- virtual void **PostRender** ()

  Post render.
- void **Fini** ()

  Finialize.
- virtual void **SetWorldPose** (const **math::Pose** &_pose)

  Set the global pose of the camera.
- void **HandleMouseEvent** (const **common::MouseEvent** &_evt)

  Hande a mouse event.
- void **HandleKeyPressEvent** (const std::string &_key)
- void **HandleKeyReleaseEvent** (const std::string &_key)
- void **SetViewController** (const std::string &_type)

Set view controller.

- void **SetViewController** (const std::string &_type, const **math::Vector3** &_pos)

  Set view controller.
- void **Resize** (unsigned int _w, unsigned int _h)

  Resize the camera.
- void **SetViewportDimensions** (float _x, float _y, float _w, float _h)

  Set the dimensions of the viewport.
- float **GetAvgFPS** () const

  Get the average frames per second.
- float **GetTriangleCount** () const

  Get the triangle count.
- void **MoveToVisual** (VisualPtr _visual)

  Move the camera to focus on a scene node.
- void **MoveToVisual** (const std::string &_visualName)
- virtual void **SetRenderTarget** (Ogre::RenderTarget *_target)

  Set to true to enable rendering.
- **GUIOverlay** * **GetGUIOverlay** ()

  *Get the **GUI** (p. 168) overlay.*
- void **EnableViewController** (bool _value) const

  Set whether the view controller is enabled.

Protected Member Functions

- virtual bool **AttachToVisualImpl** (VisualPtr _visual, bool _inheritOrientation, double _minDist=0, double _maxDist=0)

  Set the camera to be attached to a scene node.
- virtual bool **TrackVisualImpl** (VisualPtr _visual)

  Set the camera to track a scene node.

Additional Inherited Members

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

- UserCamera.hh

9.220 Vector2d Interface Reference

Message for a vector2 double.

9.220.1 Detailed Description

Message for a vector2 double.

@verbatim

import "header.proto";

message **Vector2d** (p. 368) { required double x = 1; required double y = 2; }

///

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

- vector2d.proto

9.221 Vector2d Class Reference

Public Member Functions

- **Vector2d** ()
  Constructor.
- **Vector2d** (const double &_x, const double &_y)
  Constructor.
- **Vector2d** (const **Vector2d** &_pt)
  Constructor.
- virtual ~**Vector2d** ()
  Destructor.
- double **Distance** (const **Vector2d** &_pt) const
  Calc distance to the given point.
- void **Normalize** ()
  Normalize the vector length.
- void **Set** (double _x, double _y)
  Set the contents of the vector.
- **Vector2d GetCrossProd** (const **Vector2d** &_pt) const
  Return the cross product of this vector and pt.
- **Vector2d** & **operator=** (const **Vector2d** &pt)
  Equal operator.
- const **Vector2d** & **operator=** (double value)
  Equal operator.
- **Vector2d operator+** (const **Vector2d** &pt) const
  Addition operator.
- const **Vector2d** & **operator+=** (const **Vector2d** &pt)
  Addition operator.
- **Vector2d operator-** (const **Vector2d** &pt) const
  Subtraction operators.
- const **Vector2d** & **operator-=** (const **Vector2d** &pt)
  Subtraction operators.
- const **Vector2d operator**/ (const **Vector2d** &pt) const
  Division operators.
- const **Vector2d** & **operator**/**=** (const **Vector2d** &pt)
  Division operators.
- const **Vector2d operator**/ (double v) const
  Division operators.
- const **Vector2d** & **operator**/**=** (double v)

Division operators.

- const **Vector2d operator**∗ (const **Vector2d** &pt) const

  Multiplication operators.
- const **Vector2d** & **operator∗=** (const **Vector2d** &pt)

  Multiplication operators.
- const **Vector2d operator**∗ (double v) const

  Multiplication operators.
- const **Vector2d** & **operator∗=** (double v)

  Multiplication operators.
- bool **operator==** (const **Vector2d** &pt) const

  Equality operators.
- bool **operator!=** (const **Vector2d** &pt) const

  Equality operators.
- bool **IsFinite** () const

  See if a point is finite (e.g., not nan)
- double **operator[]** (unsigned int index) const

  [] operator

Public Attributes

- double **x**

  x data
- double **y**

  y data

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector2d** &_pt)

  Ostream operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector2d** &_pt)

  Istream operator.

9.221.1 Friends And Related Function Documentation

9.221.1.1 std::ostream& operator<< (std::ostream & *_out*, const gazebo::math::Vector2d & *_pt*) [friend]

Ostream operator.

Parameters

out	Ostream
pt	**Vector2d** (p. 369) to output

Returns

The Ostream

9.221.1.2 std::istream& operator>> (std::istream & *in*, gazebo::math::Vector2d & *pt*) [friend]

Istream operator.

Parameters

in	Ostream
pt	**Vector3** (p. 373) to read values into

Returns

The istream

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

- Vector2d.hh

9.222 Vector2i Class Reference

Public Member Functions

- **Vector2i** ()
  Constructor.
- **Vector2i** (const int &_x, const int &_y)
  Constructor.
- **Vector2i** (const **Vector2i** &_pt)
  Constructor.
- virtual ~**Vector2i** ()
  Destructor.
- int **Distance** (const **Vector2i** &_pt) const
  Calc distance to the given point.
- void **Normalize** ()
  Normalize the vector length.
- void **Set** (int _x, int _y)
  Set the contents of the vector.
- **Vector2i GetCrossProd** (const **Vector2i** &_pt) const
  Return the cross product of this vector and pt.
- **Vector2i** & **operator=** (const **Vector2i** &pt)
  Equal operator.
- const **Vector2i** & **operator=** (int value)
  Equal operator.
- **Vector2i operator+** (const **Vector2i** &pt) const
  Addition operator.
- const **Vector2i** & **operator+=** (const **Vector2i** &pt)
  Addition operator.
- **Vector2i operator-** (const **Vector2i** &pt) const
  Subtraction operators.
- const **Vector2i** & **operator-=** (const **Vector2i** &pt)

Subtraction operators.

- const **Vector2i operator**/ (const **Vector2i** &pt) const

  Division operators.
- const **Vector2i** & **operator/=** (const **Vector2i** &pt)

  Division operators.
- const **Vector2i operator**/ (int v) const

  Division operators.
- const **Vector2i** & **operator/=** (int v)

  Division operators.
- const **Vector2i operator**∗ (const **Vector2i** &pt) const

  Multiplication operators.
- const **Vector2i** & **operator∗=** (const **Vector2i** &pt)

  Multiplication operators.
- const **Vector2i operator**∗ (int v) const

  Multiplication operators.
- const **Vector2i** & **operator∗=** (int v)

  Multiplication operators.
- bool **operator==** (const **Vector2i** &pt) const

  Equality operators.
- bool **operator!=** (const **Vector2i** &pt) const

  Equality operators.
- bool **IsFinite** () const

  See if a point is finite (e.g., not nan)
- int **operator[]** (unsigned int index) const

  [] operator

Public Attributes

- int **x**

  x data
- int **y**

  y data

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector2i** &_pt)

  Ostream operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector2i** &_pt)

  Istream operator.

9.222.1 Friends And Related Function Documentation

9.222.1.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::math::Vector2i & *_pt*) [friend]

Ostream operator.

Parameters

out	Ostream
pt	**Vector2i** (p. 371) to output

Returns

The Ostream

9.222.1.2 std::istream& operator>> (std::istream & *_in,* gazebo::math::Vector2i & *_pt*) [friend]

Istream operator.

Parameters

in	Ostream
pt	**Vector3** (p. 373) to read values into

Returns

The istream

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

- Vector2i.hh

9.223 Vector3 Class Reference

Public Member Functions

- **Vector3** ()

  Constructor.
- **Vector3** (const double &_x, const double &_y, const double &_z)

  Constructor.
- **Vector3** (const **Vector3** &_pt)

  Constructor.
- virtual ~**Vector3** ()

  Destructor.
- double **GetSum** () const

  Return the sum of the values.
- 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 **GetLength** () const

  Returns the length (magnitude) of the vector.
- double **GetSquaredLength** () const

  Return the square of the length (magnitude) of the vector.

- void **Normalize** ()

  Normalize the vector length.
- **Vector3 Round** ()

  Round to near whole number, return the result.
- **Vector3 GetRounded** () const

  Get a rounded version of this vector.
- void **Set** (double _x=0, double _y=0, double _z=0)

  Set the contents of the vector.
- **Vector3 GetCrossProd** (const **Vector3** &_pt) const

  Return the cross product of this vector and pt.
- double **GetDotProd** (const **Vector3** &_pt) const

  Return the dot product of this vector and pt.
- **Vector3 GetAbs** () const

  Get the absolute value of the vector
- **Vector3 GetPerpendicular** () const

  Return a vector that is perpendicular to this one.
- double **GetDistToLine** (const **Vector3** &_pt1, const **Vector3** &_pt2)

  Get distance to a line.
- 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.
- double **GetMax** () const

  Get the maximum value in the vector.
- double **GetMin** () const

  Get the minimum value in the vector.
- **Vector3** & **operator=** (const **Vector3** &_pt)

  Equal operator.
- const **Vector3** & **operator=** (double _value)

  Equal operator.
- **Vector3 operator+** (const **Vector3** &_pt) const

  Addition operator.
- const **Vector3** & **operator+=** (const **Vector3** &_pt)

  Addition 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 operators.
- const **Vector3** & **operator**/**=** (const **Vector3** &_pt)

  Division operators.
- const **Vector3 operator**/ (double _v) const

  Division operators.
- const **Vector3** & **operator**/**=** (double _v)

  Division operators.
- **Vector3 operator**∗ (const **Vector3** &_pt) const

Multiplication operators.

- const **Vector3** & **operator∗=** (const **Vector3** &_pt)

  Multiplication operators.

- **Vector3 operator∗** (double _v) const

  Multiplication operators.

- const **Vector3** & **operator∗=** (double _v)

  Multiplication operators.

- bool **operator==** (const **Vector3** &_pt) const

  Equality operators.

- bool **operator!=** (const **Vector3** &_pt) const

  Equality operators.

- bool **IsFinite** () const

  See if a point is finite (e.g., not nan)

- void **Correct** ()

  Corrects any nan values.

- double **operator[ ]** (unsigned int index) const

  [] operator

- void **Round** (int _precision)

  Round all values to _precision decimal places.

- bool **Equal** (const **Vector3** &_v) const

  Returns true if the two vectors are exactly equal.

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)

  Ostream operator.

- std::istream & **operator>>** (std::istream &_in, **gazebo::math::Vector3** &_pt)

  Istream operator.

9.223.1 Friends And Related Function Documentation

9.223.1.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::math::Vector3 & *_pt*) [friend]

Ostream operator.

Parameters

out	Ostream
pt	**Vector3** (p. 373) to output

Returns

The Ostream

9.223.1.2 std::istream& operator>> (std::istream & *_in,* gazebo::math::Vector3 & *_pt*) [friend]

Istream operator.

Parameters

in	Ostream
pt	**Vector3** (p. 373) to read values into

Returns

The istream

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

- Vector3.hh

9.224 Vector3d Interface Reference

Message for a vector3 double.

9.224.1 Detailed Description

Message for a vector3 double.

@verbatim

import "header.proto";

message **Vector3d** (p. 376) { required double x = 2; required double y = 3; required double z = 4; }

///

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

- vector3d.proto

9.225 Vector4 Class Reference

Public Member Functions

- **Vector4** ()

  Constructor.
- **Vector4** (const double &_x, const double &_y, const double &_z, const double &_w)

  Constructor.
- **Vector4** (const **Vector4** &_pt)

  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.
- void **Normalize** ()

  Normalize the vector length.
- void **Set** (double _x=0, double _y=0, double _z=0, double _w=0)

  Set the contents of the vector.
- **Vector4** & **operator=** (const **Vector4** &pt)

  Equal operator.
- const **Vector4** & **operator=** (double value)

  Equal operator.
- **Vector4 operator+** (const **Vector4** &pt) const

  Addition operator.
- const **Vector4** & **operator+=** (const **Vector4** &pt)

  Addition operator.
- **Vector4 operator-** (const **Vector4** &pt) const

  Subtraction operators.
- const **Vector4** & **operator-=** (const **Vector4** &pt)

  Subtraction operators.
- const **Vector4 operator**/ (const **Vector4** &pt) const

  Division operators.
- const **Vector4** & **operator**/**=** (const **Vector4** &pt)

  Division operators.
- const **Vector4 operator**/ (double v) const

  Division operators.
- const **Vector4** & **operator**/**=** (double v)

  Division operators.
- const **Vector4 operator**∗ (const **Vector4** &pt) const

  Multiplication operators.
- const **Vector4 operator**∗ (const **Matrix4** &_m) const
- const **Vector4** & **operator**∗**=** (const **Vector4** &pt)

  Multiplication operators.

- const **Vector4 operator**∗ (double v) const

  Multiplication operators.
- const **Vector4** & **operator∗=** (double v)

  Multiplication operators.
- bool **operator==** (const **Vector4** &pt) const

  Equality operators.
- bool **operator!=** (const **Vector4** &pt) const

  Equality operators.
- bool **IsFinite** () const

  See if a point is finite (e.g., not nan)
- double **operator[ ]** (unsigned int index) const

  [] operator

Public Attributes

- double **x**

  X value.
- double **y**

  Y value.
- double **z**

  Z value.
- double **w**

  W value.

Friends

- std::ostream & **operator**<< (std::ostream &_out, const **gazebo::math::Vector4** &_pt)

  Ostream operator.
- std::istream & **operator**>> (std::istream &_in, **gazebo::math::Vector4** &_pt)

  Istream operator.

9.225.1 Friends And Related Function Documentation

9.225.1.1 std::ostream& operator<< (std::ostream & *_out,* const gazebo::math::Vector4 & *_pt*) [friend]

Ostream operator.

Parameters

out	Ostream
pt	**Vector4** (p. 377) to output

Returns

The Ostream

9.225.1.2 std::istream& operator>> (std::istream & *in*, gazebo::math::Vector4 & *pt*) [friend]

Istream operator.

Parameters

in	Ostream
pt	**Vector4** (p. 377) to read values into

Returns

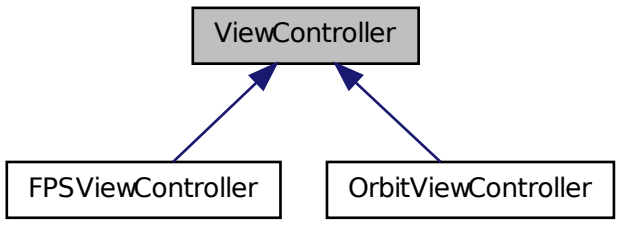
The istream

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

- Vector4.hh

9.226 ViewController Class Reference

Inheritance diagram for ViewController:



Public Member Functions

- **ViewController** (**UserCamera** *camera)

  Constructor.
- virtual ~**ViewController** ()

  Destructor.
- virtual void **Init** ()=0
- virtual void **Init** (const **math::Vector3** &)
- virtual void **Update** ()=0
- void **SetEnabled** (bool _value)

  Set whether the controller is enabled.
- virtual void **HandleMouseEvent** (const **common::MouseEvent** &_event)=0

  Handle a mouse event.
- std::string **GetTypeString** () const

Protected Attributes

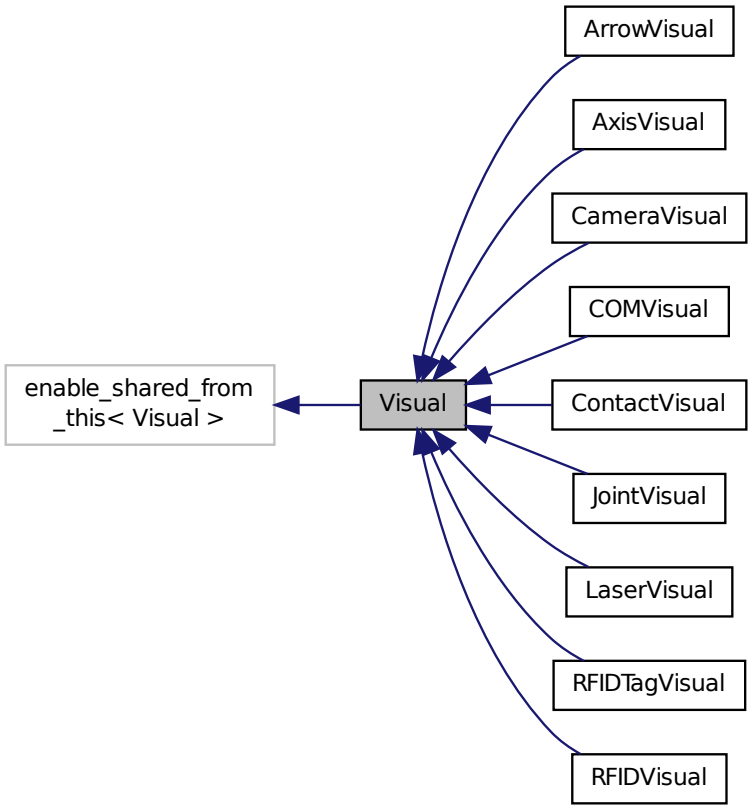
- **UserCamera** ∗ **camera**
- bool **enabled**
- std::string **typeString**

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

- ViewController.hh

9.227 Visual Class Reference

Inheritance diagram for 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 **Init** ()

  Helper for the contructor.
- void **Fini** ()
- VisualPtr **Clone** (const std::string &_name, VisualPtr _newParent)

  Clone the visual with a new name.
- void **LoadFromMsg** (ConstVisualPtr &_msg)

  Load from a message.
- void **Load** (sdf::ElementPtr sdf)

  Load the visual with a set of parameters.
- virtual void **Load** ()

  Load the visual with default parameters.
- void **Update** ()

  Update the visual.
- void **SetName** (const std::string &name)

  Set the name of the visual.
- std::string **GetName** () const

  Get the name of the visual.
- void **AttachVisual** (VisualPtr _vis)

  Attach a visual.
- void **DetachVisual** (VisualPtr _vis)

  Detach a visual.
- void **DetachVisual** (const std::string &_name)
- void **AttachObject** (Ogre::MovableObject ∗obj)

  Attach a renerable object to the visual.
- bool **HasAttachedObject** (const std::string &_name)

  Returns true if an object with _name is attached.
- void **DetachObjects** ()

  Detach all objects.
- unsigned int **GetChildCount** ()

  Get the number of attached visuals.
- VisualPtr **GetChild** (unsigned int _num)

  Get an attached visual.
- Ogre::MovableObject ∗ **AttachMesh** (const std::string &_meshName, const std::string &_objName="")

  Attach a mesh to this visual by name.
- void **SetScale** (const **math::Vector3** &scale)

  Set the scale.
- **math::Vector3 GetScale** ()

  Get the scale.
- void **SetMaterial** (const std::string &materialName, bool _unique=true)

  Set the material.
- void **SetAmbient** (const **common::Color** &_color)

  Set the ambient color of the visual.
- void **SetDiffuse** (const **common::Color** &_color)

Set the diffuse color of the visual.

- void **SetSpecular** (const **common::Color** &_color)

  Set the specular color of the visual.
- void **AttachAxes** ()
- void **SetTransparency** (float trans)

  Set the transparency.
- float **GetTransparency** ()

  Get the transparency.
- void **SetEmissive** (const **common::Color** &_color)

  Set the emissive value.
- void **SetCastShadows** (const bool &shadows)

  Set whether the visual should cast shadows.
- void **SetVisible** (bool visible, bool cascade=true)

  Set whether the visual is visible.
- void **ToggleVisible** ()

  Toggle whether this visual is visible.
- bool **GetVisible** () const

  Get whether the visual is visible.
- void **SetPosition** (const **math::Vector3** &pos)

  Set the position of the visual.
- void **SetRotation** (const **math::Quaternion** &rot)

  Set the rotation of the visual.
- void **SetPose** (const **math::Pose** &pose)

  Set the pose of the visual.
- **math::Vector3 GetPosition** () const

  Get the position of the visual.
- **math::Quaternion GetRotation** () const

  Get the rotation of the visual.
- **math::Pose GetPose** () const

  Get the pose of the visual.
- **math::Pose GetWorldPose** () const

  Get the global pose of the node.
- void **SetWorldPose** (const **math::Pose** _pose)

  Set the world pose of the visual.
- void **SetWorldPosition** (const **math::Vector3** &_pos)
- void **SetWorldRotation** (const **math::Quaternion** &_q)
- Ogre::SceneNode * **GetSceneNode** () const

  Return the scene Node of this visual entity.
- void **MakeStatic** ()

  Make the visual objects static renderables.
- bool **IsStatic** () const

  Return true if the visual is a static geometry.
- void **EnableTrackVisual** (**Visual** *vis)

  Set one visual to track/follow another.
- void **DisableTrackVisual** ()

  Disable tracking of a visual.
- std::string **GetNormalMap** () const

Get the normal map.

- void **SetNormalMap** (const std::string &nmap)

  Set the normal map.

- void **SetRibbonTrail** (bool value, const **common::Color** &_initialColor, const **common::Color** &_changeColor)

  True on or off a ribbon trail.

- **math::Box GetBoundingBox** () const

  Get the bounding box for the visual.

- **DynamicLines** ∗ **CreateDynamicLine** (**RenderOpType** type=**RENDERING_LINE_STRIP**)

  Add a line to the visual.

- void **DeleteDynamicLine** (**DynamicLines** ∗line)

  Delete a dynamic line.

- void **AttachLineVertex** (**DynamicLines** ∗_line, unsigned int _index)

  Attach a vertex of a line to the position of the visual.

- std::string **GetMaterialName** () const

  Get the name of the material.

- void **InsertMesh** (const std::string &_meshName)

  Insert a mesh into Ogre.

- void **UpdateFromMsg** (const boost::shared_ptr< msgs::Visual const > &msg)

  Update a visual based on a message.

- bool **IsPlane** () const

  Return true if the visual is a plane.

- VisualPtr **GetParent** () const

  Get the parent visual, if one exists.

- std::string **GetShaderType** () const

  Get the shader type.

- void **SetShaderType** (const std::string &_type)

  Set the shader type.

- void **MoveToPosition** (const **math::Vector3** &_end, double _pitch, double _yaw, double _time)
- void **MoveToPositions** (const std::vector< **math::Pose** > &_pts, double _time, boost::function< void()> _onComplete=NULL)
- void **SetVisibilityFlags** (uint32_t _flags)

  Set visibility flags for this visual and all children.

- void **ShowBoundingBox** ()
- void **ShowCollision** (bool _show)
- void **SetScene** (ScenePtr _scene)
- ScenePtr **GetScene** () const
- void **ShowJoints** (bool _show)
- void **ShowCOM** (bool _show)
- void **SetSkeletonPose** (const msgs::PoseAnimation &_pose)
- void **ClearParent** ()

Static Public Member Functions

- static void **InsertMesh** (const **common::Mesh** ∗mesh)

  Insert a mesh into Ogre.

Public Attributes

- VisualPtr **parent**
- std::vector< VisualPtr > **children**

Protected Attributes

- Ogre::SceneNode * **sceneNode**
- ScenePtr **scene**

9.227.1 Member Function Documentation

9.227.1.1 void SetVisible (bool *visible,* bool *cascade* = `true`)

Set whether the visual is visible.

Parameters

visible	set this node visible
cascade	setting this parameter in children too

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

- Visual.hh

9.228 Visual Interface Reference

A message containing visual information.

9.228.1 Detailed Description

A message containing visual information.

`@verbatim`

import "pose.proto"; import "geometry.proto"; import "material.proto";

message **Visual** (p. 384) { required string name = 1; optional string parent_name = 2; optional bool cast_shadows = 3; optional double transparency = 4; optional double laser_retro = 5; optional **Pose** (p. 283) pose = 6; optional **Geometry** (p. 157) geometry = 7; optional **Material** (p. 203) material = 8;

optional bool visible = 9; optional bool delete_me = 11; optional bool is_static = 12; }

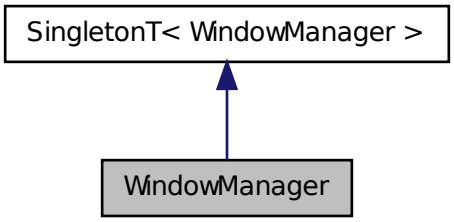
///

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

- visual.proto

9.229 WindowManager Class Reference

Inheritance diagram for WindowManager:



Public Member Functions

- void **Fini** ()

  Shutdown all the windows.
- int **CreateWindow** (const std::string &ogreHandle, unsigned int width, unsigned int height)
- void **GetAttribute** (unsigned int id, const std::string &attr, void ∗data)
- void **SetCamera** (int windowId, CameraPtr camera)

  Attach a camera to a window.
- void **Resize** (unsigned int id, int width, int height)

  Resize a window.
- void **Moved** (unsigned int id)
- float **GetAvgFPS** (unsigned int windowId)

  Get the average FPS.
- unsigned int **GetTriangleCount** (unsigned int windotId)

  Get the triangle count.

Additional Inherited Members

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

- WindowManager.hh

9.230 World Interface Reference

A message that allows for modifying (open, close) worlds.

9.230.1 Detailed Description

A message that allows for modifying (open, close) worlds.

@verbatim

message WorldModify { required string world_name = 1; optional bool remove = 2; optional bool create = 3; }

///

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

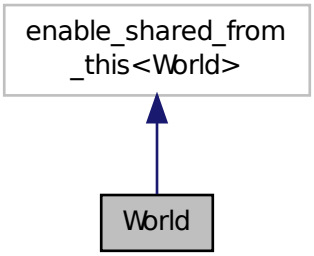
- world_modify.proto

9.231 World Class Reference

The **World** (p. 386).

`#include <World.hh>`

Inheritance diagram for World:



Public Member Functions

- **World** (const std::string &_name="")
  Private constructor.
- ~**World** ()
  Private destructor.
- void **Load** (sdf::ElementPtr _sdf)
  Load the world using SDF parameters.
- void **Save** (const std::string &_filename)
- void **Init** ()
  Initialize the world.
- void **Run** ()
  Run the world in a thread.

- void **Stop** ()

  Stop the world.
- void **Fini** ()

  Finilize the world.
- void **Clear** ()

  Remove all entities from the world.
- std::string **GetName** () const

  Get the name of the world.
- unsigned int **GetParamCount** () const

  Get the number of parameters.
- common::Param * **GetParam** (unsigned int index) const

  Get a param.
- PhysicsEnginePtr **GetPhysicsEngine** () const

  Return the physics engine.
- unsigned int **GetModelCount** () const

  Get the number of models.
- ModelPtr **GetModel** (unsigned int index)

  Get a model based on an index.
- void **Reset** (bool _resetTime=true)

  Reset the simulation to the initial settings.
- EntityPtr **GetSelectedEntity** () const

  Get the selected entity.
- void **PrintEntityTree** ()

  Print entity tree.
- **common::Time GetSimTime** () const

  Get the simulation time.
- void **SetSimTime** (**common::Time** t)

  Set the sim time.
- **common::Time GetPauseTime** () const

  Get the pause time.
- **common::Time GetStartTime** () const

  Get the start time.
- **common::Time GetRealTime** () const

  Get the real time (elapsed time)
- bool **IsPaused** () const

  Returns the state of the simulation true if paused.
- void **SetPaused** (bool p)

  Set whether the simulation is paused.
- BasePtr **GetByName** (const std::string &name)

  Get an element by name.
- ModelPtr **GetModelById** (unsigned int _id)

  Get a model by id.
- ModelPtr **GetModelByName** (const std::string &name) GAZEBO_DEPRECATED

  Get a model by name DEPRECATED.
- ModelPtr **GetModel** (const std::string &_name)

  Get a model by name.
- EntityPtr **GetEntityByName** (const std::string &_name) GAZEBO_DEPRECATED

Get a pointer to a entity based on a name.

- EntityPtr **GetEntity** (const std::string &_name)

  Get a pointer to a entity based on a name.

- ModelPtr **GetModelBelowPoint** (const **math::Vector3** &_pt)

  Get the nearest model below a point.

- EntityPtr **GetEntityBelowPoint** (const **math::Vector3** &_pt)

  Get the nearest entity below a point.

- **WorldState GetState** ()

  Get the current world state.

- void **SetState** (const **WorldState** &_state)

  Set the current world state.

- void **InsertModel** (const std::string &_sdfFilename)

  Insert a model from an SDF file.

- void **InsertModel** (const **sdf::SDF** &_sdf)

  Insert a model using SDF.

- std::string **StripWorldName** (const std::string &_name) const

  Return a version of the name with "<world_name>::" removed.

- void **LoadPlugin** (const std::string &_filename, const std::string &_name, sdf::ElementPtr _sdf)

  Load a plugin.

- void **RemovePlugin** (const std::string &_name)

  Remove a running plugin.

- void **EnableAllModels** ()

  Enable all links in all the models.

- void **DisableAllModels** ()

  Disable all links in all the models.

- void **StepWorld** (int _steps)

  Step callback.

Public Attributes

- boost::recursive_mutex * **setWorldPoseMutex**

  TODO: Add an accessor for this, and make it private lock all pose updates when worldPose is being updated for a model.

- std::list< **Entity** * > **dirtyPoses**

Protected Attributes

- common::Param_V **parameters**

  List of all the parameters.

9.231.1 Detailed Description

The **World** (p. 386).

9.231.2 Member Function Documentation

9.231.2.1 common::Time GetPauseTime ( ) const

Get the pause time.

Returns

The pause time

9.231.2.2 PhysicsEnginePtr GetPhysicsEngine ( ) const

Return the physics engine.

Returns

Pointer to the physics engine

9.231.2.3 common::Time GetRealTime ( ) const

Get the real time (elapsed time)

Returns

The real time

9.231.2.4 common::Time GetSimTime ( ) const

Get the simulation time.

Returns

The simulation time

9.231.2.5 common::Time GetStartTime ( ) const

Get the start time.

Returns

The start time

9.231.2.6 void Load (sdf::ElementPtr *_sdf*)

Load the world using SDF parameters.

Parameters

_sdf	SDF parameters

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

- World.hh

9.232 WorldControl Interface Reference

A message that allows for control of world functions.

9.232.1 Detailed Description

A message that allows for control of world functions.

```
@verbatim
```

import "header.proto";

message **WorldControl** (p. 390) { optional bool pause = 1; optional bool step = 2; optional bool reset_time = 3; optional bool reset_world = 4; }

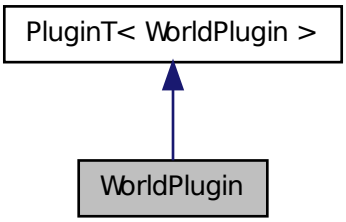
///

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

- world_control.proto

9.233 WorldPlugin Class Reference

Inheritance diagram for WorldPlugin:



Public Member Functions

- virtual void **Load** (physics::WorldPtr _world, sdf::ElementPtr _sdf)=0

  Load function.
- virtual void **Init** ()
- virtual void **Reset** ()

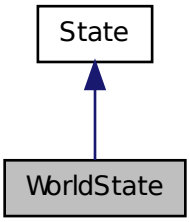
Additional Inherited Members

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

- common/Plugin.hh

9.234 WorldState Class Reference

Inheritance diagram for WorldState:



Public Member Functions

- **WorldState** ()
  Default constructor.
- **WorldState** (WorldPtr _world)
  Constructor.
- virtual ~**WorldState** ()
  Destructor.
- virtual void **Load** (sdf::ElementPtr _elem)
  Load state from SDF element.
- unsigned int **GetModelStateCount** () const
  Get the number of model states.
- const sdf::ElementPtr & **GetSDF** () const
  Get the state in SDF format.
- **ModelState GetModelState** (unsigned int _index) const
  Get a model state.
- **ModelState GetModelState** (const std::string &_modelName) const
  Get a model state by model name.

Additional Inherited Members

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

- WorldState.hh

9.235 WorldStatistics Interface Reference

A message statiscs about a world.

9.235.1 Detailed Description

A message statiscs about a world.

`@verbatim`

import "header.proto"; import "time.proto";

message **WorldStatistics** (p. 392) { required **Time** (p. 359) sim_time = 2; required **Time** (p. 359) pause_time = 3; required **Time** (p. 359) real_time = 4; required bool paused = 5; optional int32 model_count = 6; }

///

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

- world_stats.proto

Chapter 10

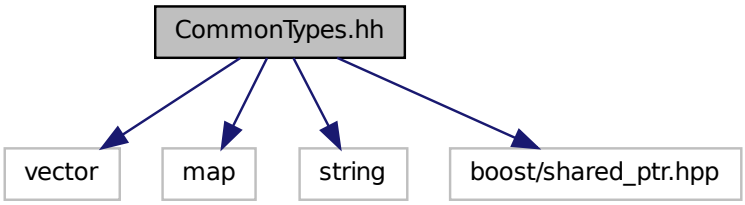
File Documentation

10.1 CommonTypes.hh File Reference

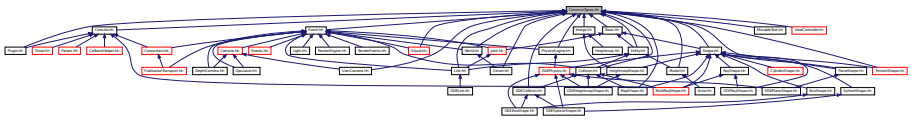
Forward declarations for the common classes.

```
#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::event**

  Event *(p. 148) namespace.*

Macros

- #define **NULL** 0
- #define **GAZEBO_DEPRECATED**
- #define **GAZEBO_FORCEINLINE**

Typedefs

- typedef boost::shared_ptr < WorldPlugin > **WorldPluginPtr**
- typedef boost::shared_ptr < ModelPlugin > **ModelPluginPtr**
- typedef boost::shared_ptr < SensorPlugin > **SensorPluginPtr**
- typedef boost::shared_ptr < GUIPlugin > **GUIPluginPtr**
- typedef boost::shared_ptr < SystemPlugin > **SystemPluginPtr**
- typedef std::vector < common::Param * > **Param_V**
- typedef std::map< std::string, std::string > **StrStr_M**
- typedef boost::shared_ptr < Animation > **AnimationPtr**
- typedef boost::shared_ptr < PoseAnimation > **PoseAnimationPtr**
- typedef boost::shared_ptr < NumericAnimation > **NumericAnimationPtr**
- typedef boost::shared_ptr < Connection > **ConnectionPtr**
- typedef std::vector < ConnectionPtr > **Connection_V**

10.1.1 Detailed Description

Forward declarations for the common classes.

10.2 MathTypes.hh File Reference

Forward declarations for the math classes.

This graph shows which files directly or indirectly include this file:

Namespaces

- namespace **gazebo::math**

  Math namespace.

10.2.1 Detailed Description

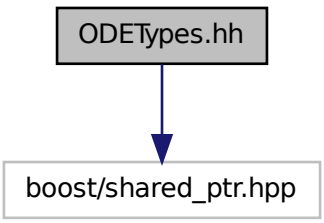
Forward declarations for the math classes.

10.3 ODETypes.hh File Reference

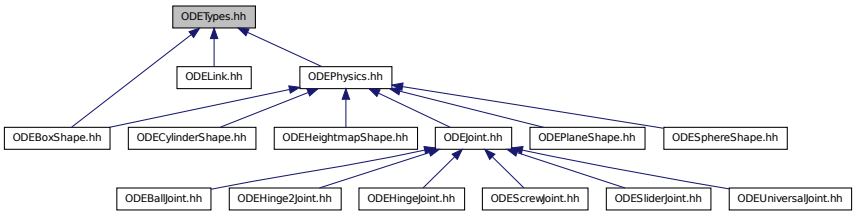
ODE wrapper forward declarations and typedefs.

```
#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::physics**

  Physics namespace.

Typedefs

- typedef boost::shared_ptr
  < ODECollision > **ODECollisionPtr**
- typedef boost::shared_ptr
  < ODELink > **ODELinkPtr**
- typedef boost::shared_ptr
  < ODERayShape > **ODERayShapePtr**

10.3.1 Detailed Description

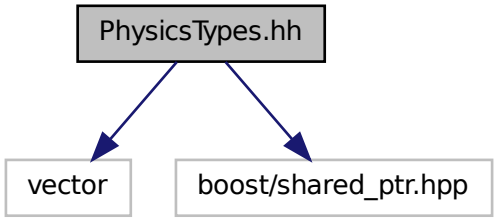
ODE wrapper forward declarations and typedefs.

10.4 PhysicsTypes.hh File Reference

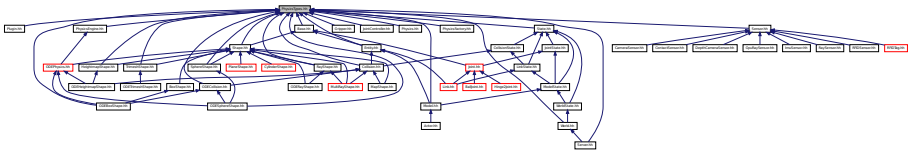
Physics forward declarations and type defines.

```
#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::physics**

  Physics namespace.

Macros

- #define **GZ_ALL_COLLIDE** 0x0FFFFFFF
- #define **GZ_NONE_COLLIDE** 0x00000000
- #define **GZ_FIXED_COLLIDE** 0x00000001
- #define **GZ_SENSOR_COLLIDE** 0x00000003
- #define **GZ_GHOST_COLLIDE** 0x10000000

Typedefs

- typedef boost::shared_ptr< Base > **BasePtr**
- typedef boost::shared_ptr < Contact > **ContactPtr**
- typedef boost::shared_ptr< Entity > **EntityPtr**
- typedef boost::shared_ptr< **World** > **WorldPtr**
- typedef boost::shared_ptr< **Model** > **ModelPtr**
- typedef boost::shared_ptr< Actor > **ActorPtr**
- typedef boost::shared_ptr< **Link** > **LinkPtr**

- typedef boost::shared_ptr < **Collision** > **CollisionPtr**
- typedef boost::shared_ptr< **Joint** > **JointPtr**
- typedef boost::shared_ptr < PhysicsEngine > **PhysicsEnginePtr**
- typedef boost::shared_ptr< Shape > **ShapePtr**
- typedef boost::shared_ptr < RayShape > **RayShapePtr**
- typedef boost::shared_ptr < MultiRayShape > **MultiRayShapePtr**
- typedef boost::shared_ptr < **Inertial** > **InertialPtr**
- typedef boost::shared_ptr < SurfaceParams > **SurfaceParamsPtr**
- typedef boost::shared_ptr < BoxShape > **BoxShapePtr**
- typedef boost::shared_ptr < CylinderShape > **CylinderShapePtr**
- typedef boost::shared_ptr < SphereShape > **SphereShapePtr**
- typedef boost::shared_ptr < MeshShape > **MeshShapePtr**
- typedef std::vector< BasePtr > **Base_V**
- typedef std::vector< ModelPtr > **Model_V**
- typedef std::vector< ActorPtr > **Actor_V**
- typedef std::vector< JointPtr > **Joint_V**
- typedef std::vector< LinkPtr > **Link_V**
- typedef std::vector< CollisionPtr > **Collision_V**

10.4.1 Detailed Description

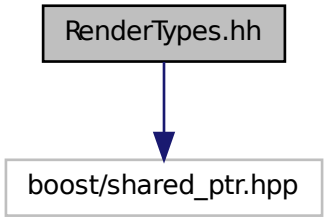
Physics forward declarations and type defines.

10.5 RenderTypes.hh File Reference

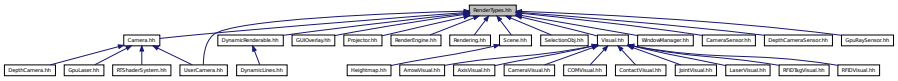
Forward declarations and type defs for rendering.

```
#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::rendering**

  Rendering namespace.

Macros

- #define **GZ_VISIBILITY_ALL** 0xFFFFFFFF
- #define **GZ_VISIBILITY_GUI** 0x00000001

Typedefs

- typedef boost::shared_ptr< **Scene** > **ScenePtr**
- typedef boost::shared_ptr< **Light** > **LightPtr**
- typedef boost::shared_ptr< Camera > **CameraPtr**
- typedef boost::shared_ptr < UserCamera > **UserCameraPtr**
- typedef boost::shared_ptr < DepthCamera > **DepthCameraPtr**
- typedef boost::shared_ptr < GpuLaser > **GpuLaserPtr**
- typedef boost::shared_ptr < DynamicLines > **DynamicLinesPtr**
- typedef boost::shared_ptr< **Visual** > **VisualPtr**

- typedef boost::shared_ptr < LaserVisual > **LaserVisualPtr**
- typedef boost::shared_ptr < CameraVisual > **CameraVisualPtr**
- typedef boost::shared_ptr < JointVisual > **JointVisualPtr**
- typedef boost::shared_ptr < ContactVisual > **ContactVisualPtr**
- typedef boost::shared_ptr < ArrowVisual > **ArrowVisualPtr**
- typedef boost::shared_ptr < AxisVisual > **AxisVisualPtr**
- typedef boost::shared_ptr < COMVisual > **COMVisualPtr**
- typedef boost::shared_ptr < RFIDVisual > **RFIDVisualPtr**
- typedef boost::shared_ptr < RFIDTagVisual > **RFIDTagVisualPtr**

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 }

10.5.1 Detailed Description

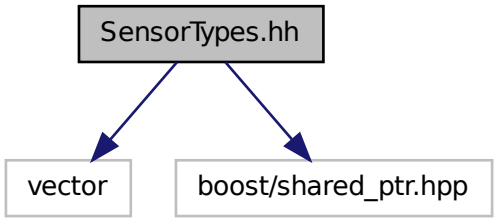
Forward declarations and type defs for rendering.

10.6 SensorTypes.hh File Reference

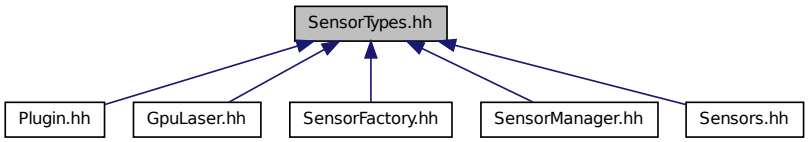
Forward declarations and typedefs for sensors.

```
#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::sensors**

  Sensors namespace.

Typedefs

- typedef boost::shared_ptr< **Sensor** > **SensorPtr**
- typedef boost::shared_ptr < **RaySensor** > **RaySensorPtr**
- typedef boost::shared_ptr < **CameraSensor** > **CameraSensorPtr**
- typedef boost::shared_ptr < DepthCameraSensor > **DepthCameraSensorPtr**
- typedef boost::shared_ptr < **ContactSensor** > **ContactSensorPtr**
- typedef boost::shared_ptr < GpuRaySensor > **GpuRaySensorPtr**
- typedef boost::shared_ptr < RFIDSensor > **RFIDSensorPtr**
- typedef boost::shared_ptr < RFIDTag > **RFIDTagPtr**

- typedef std::vector< SensorPtr > **Sensor_V**
- typedef std::vector< RaySensorPtr > **RaySensor_V**
- typedef std::vector < CameraSensorPtr > **CameraSensor_V**
- typedef std::vector < DepthCameraSensorPtr > **DepthCameraSensor_V**
- typedef std::vector < ContactSensorPtr > **ContactSensor_V**
- typedef std::vector < GpuRaySensorPtr > **GpuRaySensor_V**
- typedef std::vector< RFIDSensor > **RFIDSensor_V**
- typedef std::vector< RFIDTag > **RFIDTag_V**

10.6.1 Detailed Description

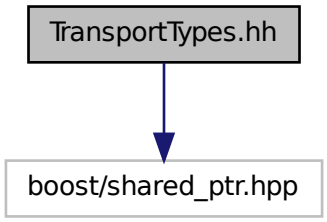
Forward declarations and typedefs for sensors.

10.7 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::transport**

Transport namespace.

Typedefs

- typedef boost::shared_ptr < **Publisher** > **PublisherPtr**
- typedef boost::shared_ptr < Subscriber > **SubscriberPtr**
- typedef boost::shared_ptr< Node > **NodePtr**
- typedef boost::shared_ptr < Publication > **PublicationPtr**
- typedef boost::shared_ptr < PublicationTransport > **PublicationTransportPtr**
- typedef boost::shared_ptr < SubscriptionTransport > **SubscriptionTransportPtr**

10.7.1 Detailed Description

Forward declarations for transport.

Index