Back (/tutorials?cat=write_plugin)

World plugins

Prerequisites:

- Model Manipulation (http://gazebosim.org/tutorials/?tut=plugins_model)
- Plugin Tutorial (http://gazebosim.org/tutorials/?tut=plugins_hello_world)

Code:

Source code: gazebo/examples/plugins/factory (https://bitbucket.org/osrf/gazebo/src/gazebo7/examples/plugins/factory)

It can be useful to control what models exist in a running simulation, and when they should be inserted. This tutorial demonstrates how to insert predefined and custom models into Gazebo.

Use the `gazebo_plugin_tutorial` from the previous plugin tutorials

```
$ mkdir ~/gazebo_plugin_tutorial
$ cd ~/gazebo_plugin_tutorial
```

Create a new source file:

```
$ gedit factory.cc
```

Copy the following code into the `factory.cc` file:

```
#include <ignition/math/Pose3.hh>
#include "gazebo/physics/physics.hh"
#include "gazebo/common/common.hh"
#include "gazebo/gazebo.hh"

namespace gazebo
{
class Factory : public WorldPlugin
{
  public: void Load(physics::WorldPtr _parent, sdf::ElementPtr /*_sdf*/)
  {
    // Option 1: Insert model from file via function call.
    // The filename must be in the GAZEBO_MODEL_PATH environment variable.
    _parent->InsertModelFile("model://box");

    // Option 2: Insert model from string via function call.
    // Insert a sphere model from string
    sdf::SDF sphereSDF;
    sphereSDF.SetFromString(
       "<sdf version ='1.4'>\
          <model name ='sphere'>\
            <pose>1 0 0 0 0 0</pose>\
            <link name ='link'>\
              <pose>0 0 .5 0 0 0</pose>\
              <collision name ='collision'>\
                <geometry>\
                  <sphere><radius>0.5</radius></sphere>\
                </geometry>\
              </collision>\
              <visual name ='visual'>\
                <geometry>\
                  <sphere><radius>0.5</radius></sphere>\
                </geometry>\
              </visual>\
            </link>\
          </model>\
        </sdf>");
    // Demonstrate using a custom model name.
    sdf::ElementPtr model = sphereSDF.Root()->GetElement("model");
    model->GetAttribute("name")->SetFromString("unique_sphere");
    _parent->InsertModelSDF(sphereSDF);

    // Option 3: Insert model from file via message passing.
    {
      // Create a new transport node
      transport::NodePtr node(new transport::Node());

      // Initialize the node with the world name
      node->Init(_parent->GetName());

      // Create a publisher on the ~/factory topic
      transport::PublisherPtr factoryPub =
      node->Advertise<msgs::Factory>("~/factory");

      // Create the message
      msgs::Factory msg;

      // Model file to load
      msg.set_sdf_filename("model://cylinder");

      // Pose to initialize the model to
      msgs::Set(msg.mutable_pose(),
          ignition::math::Pose3d(
            ignition::math::Vector3d(1, -2, 0),
            ignition::math::Quaterniond(0, 0, 0)));

      // Send the message
      factoryPub->Publish(msg);
    }
  }
};

// Register this plugin with the simulator
GZ_REGISTER_WORLD_PLUGIN(Factory)
}
```

The Code Explained

The first part of the code creates a world plugin.

```
#include <ignition/math/Pose3.hh>
#include "gazebo/physics/physics.hh"
#include "gazebo/common/common.hh"
#include "gazebo/gazebo.hh"

namespace gazebo
{
class Factory : public WorldPlugin
{
  public: void Load(physics::WorldPtr _parent, sdf::ElementPtr /*_sdf*/)
```

Within the `Load` function are three different methods for model insertion.

The first method uses a World method to load a model based on a file in the resource path defined by the `GAZEBO_MODEL_PATH` environment variable.

```
    // Option 1: Insert model from file via function call.
    // The filename must be in the GAZEBO_MODEL_PATH environment variable.
    _parent->InsertModelFile("model://box");
```

The second method uses a World method to load a model based on string data.

```
    // Option 2: Insert model from string via function call.
    // Insert a sphere model from string
    sdf::SDF sphereSDF;
    sphereSDF.SetFromString(
       "<sdf version ='1.4'>\
          <model name ='sphere'>\
            <pose>1 0 0 0 0 0</pose>\
            <link name ='link'>\
              <pose>0 0 .5 0 0 0</pose>\
              <collision name ='collision'>\
                <geometry>\
                  <sphere><radius>0.5</radius></sphere>\
                </geometry>\
              </collision>\
              <visual name ='visual'>\
                <geometry>\
                  <sphere><radius>0.5</radius></sphere>\
                </geometry>\
              </visual>\
            </link>\
          </model>\
        </sdf>");
    // Demonstrate using a custom model name.
    sdf::ElementPtr model = sphereSDF.Root()->GetElement("model");
    model->GetAttribute("name")->SetFromString("unique_sphere");
    _parent->InsertModelSDF(sphereSDF);
```

The third method uses the message passing mechanism to insert a model. This method is most useful for stand alone applications that communicate with Gazebo over a network connection.

```
    // Option 3: Insert model from file via message passing.
    {
      // Create a new transport node
      transport::NodePtr node(new transport::Node());

      // Initialize the node with the world name
      node->Init(_parent->GetName());

      // Create a publisher on the ~/factory topic
      transport::PublisherPtr factoryPub =
      node->Advertise<msgs::Factory>("~/factory");

      // Create the message
      msgs::Factory msg;

      // Model file to load
      msg.set_sdf_filename("model://cylinder");

      // Pose to initialize the model to
      msgs::Set(msg.mutable_pose(),
          ignition::math::Pose3d(
            ignition::math::Vector3d(1, -2, 0),
            ignition::math::Quaterniond(0, 0, 0)));

      // Send the message
      factoryPub->Publish(msg);
```

Compile

Assuming the reader has gone through the Plugin Overview Tutorial (http://gazebosim.org/tutorials/?tut=plugins_hello_world), all that needs to be done in addition is save the above code as `~/gazebo_plugin_tutorial/factory.cc` and add the following lines to `~/gazebo_plugin_tutorial/CMakeLists.txt`

```
add_library(factory SHARED factory.cc)
target_link_libraries(factory
  ${GAZEBO_LIBRARIES}
)
```

Compiling this code will result in a shared library, `~/gazebo_plugin_tutorial/ build/libfactory .so` , that can be inserted in a Gazebo simulation.

```
$ mkdir ~/gazebo_plugin_tutorial/build
$ cd ~/gazebo_plugin_tutorial/build
$ cmake ../
$ make
```

Make the shapes

Make a models directory with a box and a cylinder inside

```
$ mkdir ~/gazebo_plugin_tutorial/models
$ cd ~/gazebo_plugin_tutorial/models
$ mkdir box cylinder
```

Create a box model

```
$ cd box
$ gedit model.sdf
```

Copy and paste the following into box/model.sdf

```
<?xml version='1.0'?>
<sdf version ='1.6'>
  <model name ='box'>
    <pose>1 2 0 0 0 0</pose>
    <link name ='link'>
      <pose>0 0 .5 0 0 0</pose>
      <collision name ='collision'>
        <geometry>
          <box><size>1 1 1</size></box>
        </geometry>
      </collision>
      <visual name ='visual'>
        <geometry>
          <box><size>1 1 1</size></box>
        </geometry>
      </visual>
    </link>
  </model>
</sdf>
```

Create a `model.config` file

```
$ gedit model.config
```

Copy the following into `model.config`

```
<?xml version='1.0'?>

<model>
  <name>box</name>
  <version>1.0</version>
  <sdf >model.sdf</sdf>

  <author>
    <name>me</name>
    <email>somebody@somewhere.com</email>
  </author>

  <description>
    A simple Box.
  </description>
</model>
```

Navigate to the cylinder directory, and create a new `model.sdf` file

```
$ cd ~/gazebo_plugin_tutorial/models/cylinder
$ gedit model.sdf
```

Copy the following into `model.sdf`

```
<?xml version='1.0'?>
<sdf version ='1.6'>
  <model name ='cylinder'>
    <pose>1 2 0 0 0 0</pose>
    <link name ='link'>
      <pose>0 0 .5 0 0 0</pose>
      <collision name ='collision'>
        <geometry>
          <cylinder><radius>0.5</radius><length>1</length></cylinder>
        </geometry>
      </collision>
      <visual name='visual'>
        <geometry>
          <cylinder><radius>0.5</radius><length>1</length></cylinder>
        </geometry>
      </visual>
    </link>
  </model>
</sdf>
```

Create a model.config file

```
$ gedit model.config
```

Copy the following into model.config

```
<?xml version='1.0'?>

<model>
  <name>cylinder</name>
  <version>1.0</version>
  <sdf>model.sdf</sdf>

  <author>
    <name>me</name>
    <email>somebody@somewhere.com</email>
  </author>

  <description>
    A simple cylinder.
  </description>
</model>
```

Run the code

Make sure your `$GAZEBO_MODEL_PATH` refers to your new models directory

```
$ export GAZEBO_MODEL_PATH=$HOME/gazebo_plugin_tutorial/models:$GAZEBO_MODEL_PATH
```

Add your library path to the `GAZEBO_PLUGIN_PATH` :

```
$ export GAZEBO_PLUGIN_PATH=$HOME/gazebo_plugin_tutorial/build:$GAZEBO_PLUGIN_PATH
```

Create a world SDF file called `~/gazebo_plugin_tutorial/ factory .world`

```
$ cd ~/gazebo_plugin_tutorial
$ gedit factory.world
```

Copy the following into the world

```
<?xml version="1.0"?>
<sdf version="1.4">
  <world name="default">
    <include>
      <uri>model://ground_plane</uri>
    </include>

    <include>
      <uri>model://sun</uri>
    </include>

    <plugin name="factory" filename="libfactory.so"/>
  </world>
</sdf>
```

Run Gazebo

```
$ gazebo ~/gazebo_plugin_tutorial/factory.world
```

The Gazebo window should show an environment with a sphere, box, and cylinder arranged in a row.

(//plus.google.com/u/0/115981436296571800301?prsrc=3)

(https://www.youtube.com/channel/UCJyqf9XJpDoM9