Back (/tutorials?cat=write_plugin)

System Plugin

Prerequisites

Overview / HelloWorld (http://gazebosim.org/tutorials?tut=plugins_hello_world) Plugin Tutorial

Overview

Source: gazebo/examples/plugins/system_gui_plugin (https://bitbucket.org/osrf/gazebo/src/gazebo5/examples/plugins/system_gui_plugin)

This tutorial will create a source file that is a system plugin for gzclient designed to save images into the directory `/tmp/gazebo_frames` .

Source code

We'll start with the source file. Create a file called `system_gui.cc` with the following contents:

```
$ cd ~/gazebo_plugin_tutorial
$ gedit system_gui.cc
```

Copy the following into `system_gui.cc`

```
#include <gazebo/math/Rand.hh>
#include <gazebo/gui/GuiIface.hh>
#include <gazebo/rendering/rendering.hh>
#include <gazebo/gazebo.hh>

namespace gazebo
{
  class SystemGUI : public SystemPlugin
  {
    ////////////////////////////////////////////
    /// \brief Destructor
    public: virtual ~SystemGUI()
    {
      this->connections.clear();
      if (this->userCam)
        this->userCam->EnableSaveFrame(false);
      this->userCam.reset();
    }

    ////////////////////////////////////////////
    /// \brief Called after the plugin has been constructed.
    public: void Load(int /*_argc*/, char ** /*_argv*/)
    {
      this->connections.push_back(
          event::Events::ConnectPreRender(
            boost::bind(&SystemGUI::Update, this)));
    }

    ////////////////////////////////////////////
    // \brief Called once after Load
    private: void Init()
    {
    }

    ////////////////////////////////////////////
    /// \brief Called every PreRender event. See the Load function.
    private: void Update()
    {
      if (!this->userCam)
      {
        // Get a pointer to the active user camera
        this->userCam = gui::get_active_camera();

        // Enable saving frames
        this->userCam->EnableSaveFrame(true);

        // Specify the path to save frames into
        this->userCam->SetSaveFramePathname("/tmp/gazebo_frames");
      }

      // Get scene pointer
      rendering::ScenePtr scene = rendering::get_scene();

      // Wait until the scene is initialized.
      if (!scene || !scene->Initialized())
        return;

      // Look for a specific visual by name.
      if (scene->GetVisual("ground_plane"))
        std::cout << "Has ground plane visual\n";
    }

    /// Pointer the user camera.
    private: rendering::UserCameraPtr userCam;

    /// All the event connections.
    private: std::vector<event::ConnectionPtr> connections;
  };

  // Register this plugin with the simulator
  GZ_REGISTER_SYSTEM_PLUGIN(SystemGUI)
}
```

Both the **`Load`** and **`Init`** functions must not block. The **`Load`** and **`Init`** functions are called at startup, before Gazebo is loaded.

On the first **`Update`**, we get a pointer to the user camera (the camera used in the graphical interface) and enable saving of frames.

1. Get the user camera

   ```
   this->userCam = gui::get_active_camera();
   ```

2. Enable save frames

   ```
   this->userCam->EnableSaveFrame(true);
   ```

3. Set the location to save frames

```
this->userCam->SetSaveFramePathname("/tmp/gazebo_frames");
```

Compiling Camera Plugin

Assuming the reader has gone through the Hello WorldPlugin tutorial (http://gazebosim.org/tutorials?tut=plugins_hello_world) all that needs to be done is to add the following lines to `~/gazebo_plugin_tutorial/` **`CMakeLists`**`.txt`

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

Rebuild, and you should end up with a libsystem_gui.so library.

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

Running Plugin

First start gzserver in the background:

```
$ gzserver &
```

Run the client with plugin:

```
$ gzclient -g libsystem_gui.so
```

Inside `/tmp/gazebo_frames` you should see many saved images from the current plugin.

Note: Remember to also terminate the background server process after you quit the client. In the same terminal, bring the process to foreground:

```
$ fg
```

and press **`Ctrl`**`-C` to abort the process. Alternatively, just kill the `gzserver` process:

```
$ killall gzserver
```


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

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