Lab Sheet 5

Exploring Gazebo Plugins

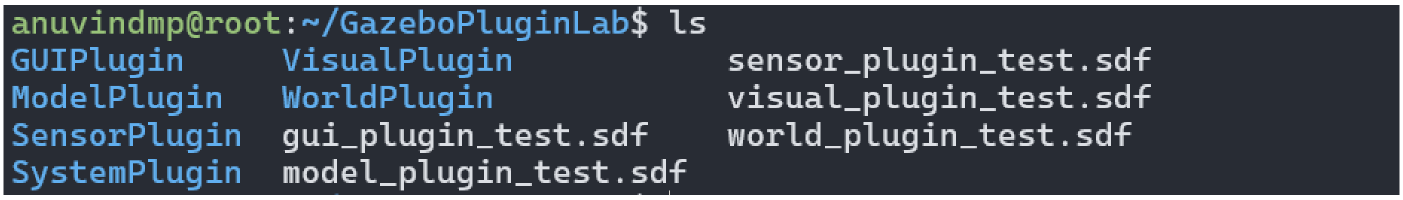
Name : Anuvind MP

Roll : AM.EN.U4AIE22010

Q) Investigate the Gazebo Plugins and make a report on the same with screen shots.

Gazebo Plugins

- The project directory, named **GazeboPluginLab**, contains everything except the GUI.
- **SDF files** and the **world file** for the world plugin are placed within the SDF project directory.
- Each plugin directory has its own **build directory** where the cmake command is executed to establish dependencies. This is also where .so files are generated after running cmake .. and make commands.
- The **CMakeLists.txt** file specifies all required dependencies, libraries, and settings needed for the plugin to function correctly.
- The **.cpp file** contains the actual code for the plugin.

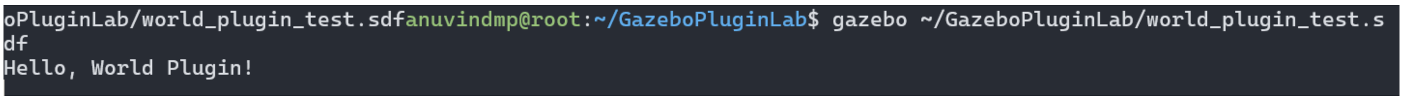


```
anuvindmp@root:~/GazeboPluginLab$ ls
GUIPlugin      VisualPlugin           sensor_plugin_test.sdf
ModelPlugin    WorldPlugin            visual_plugin_test.sdf
SensorPlugin   gui_plugin_test.sdf    world_plugin_test.sdf
SystemPlugin   model_plugin_test.sdf
```

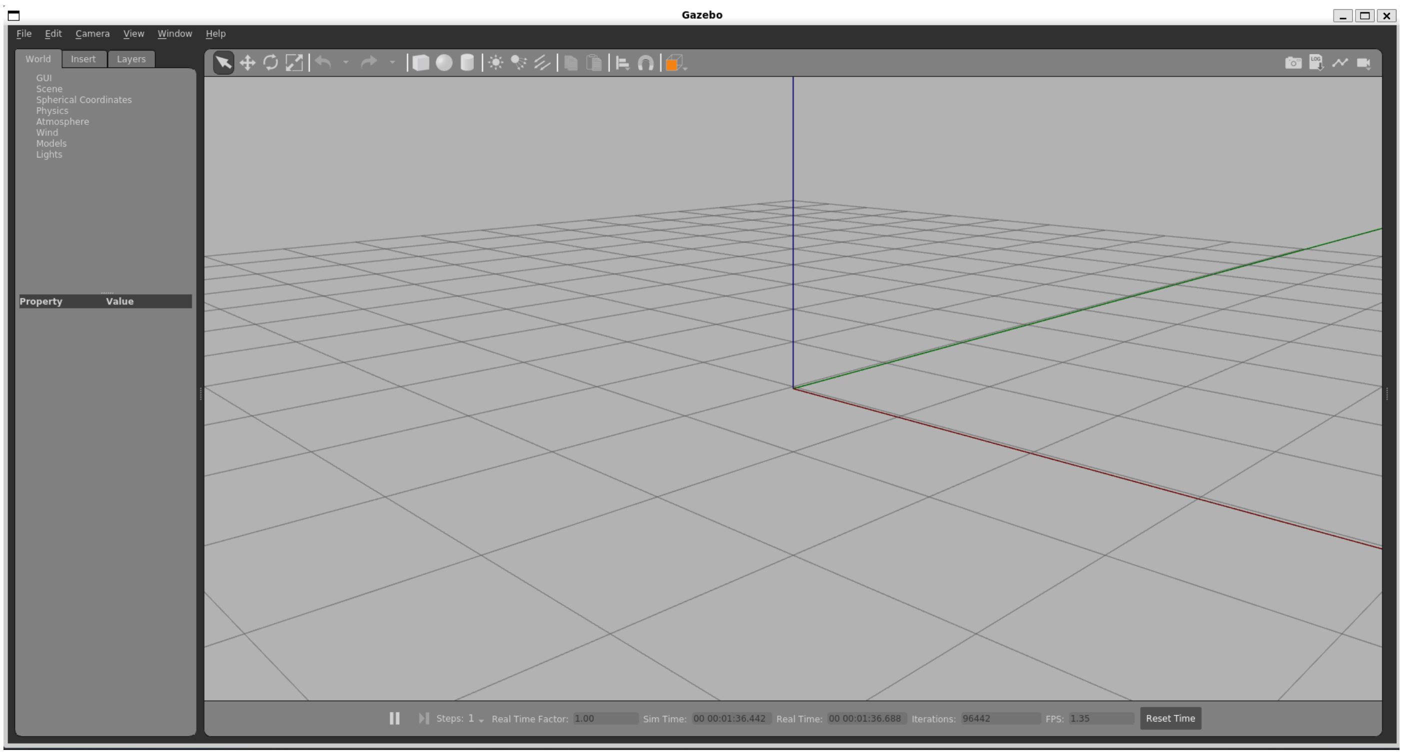
1. **World Plugin :**

- The **world plugin** is specifically designed to control elements within the simulation environment.
- It allows developers to add automation or scripted interactions directly within the Gazebo world.
- The **world configuration** and **elements**, such as objects, lighting, and layout, are defined using SDF files linked to the plugin.
- A successful load of the world plugin can be verified by a custom message in the terminal (e.g., "Hello, world Plugin!").
- The plugin's functionality can include custom behaviors, event handling, and real-time modifications to the simulation.

SCREENSHOTS :



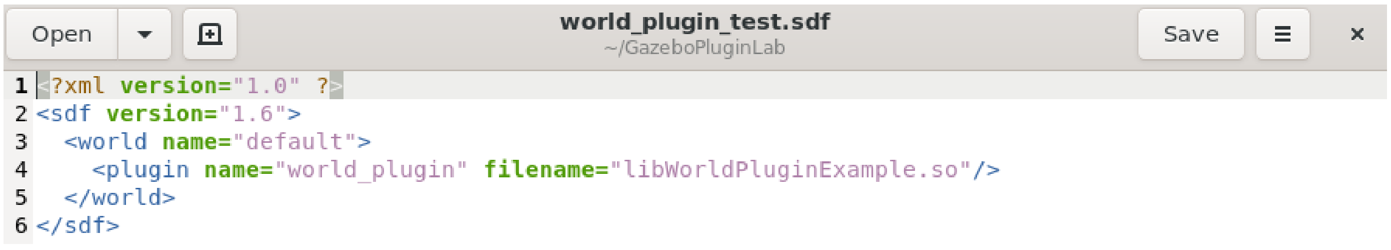
```
oPluginLab/world_plugin_test.sdfanuvindmp@root:~/GazeboPluginLab$ gazebo ~/GazeboPluginLab/world_plugin_test.s
df
Hello, World Plugin!
```



The world generated by the plugin is empty. In the terminal, the message "Hello, world Plugin!" appears, which was added in the C++ code to verify that the world file loaded correctly.

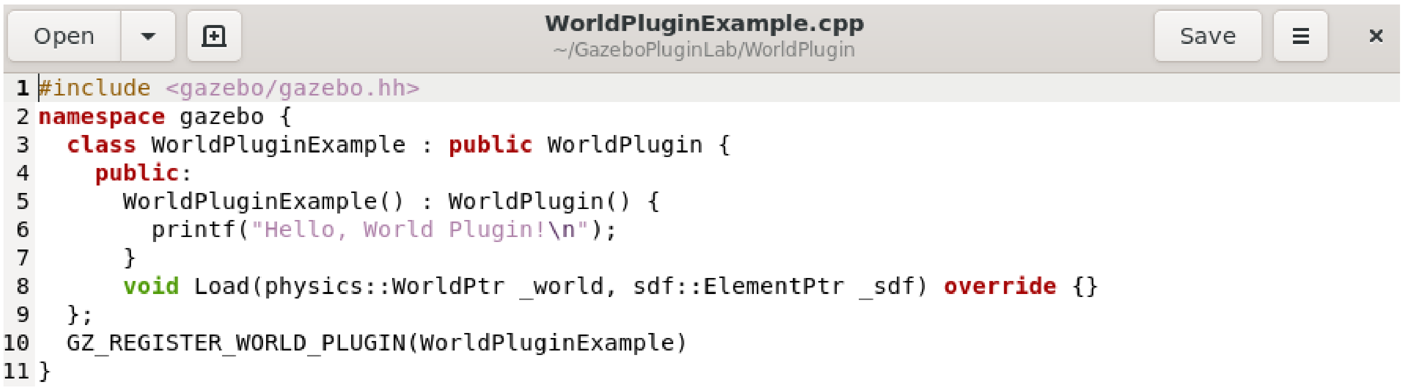
Code :

a. world_plugin_test.sdf



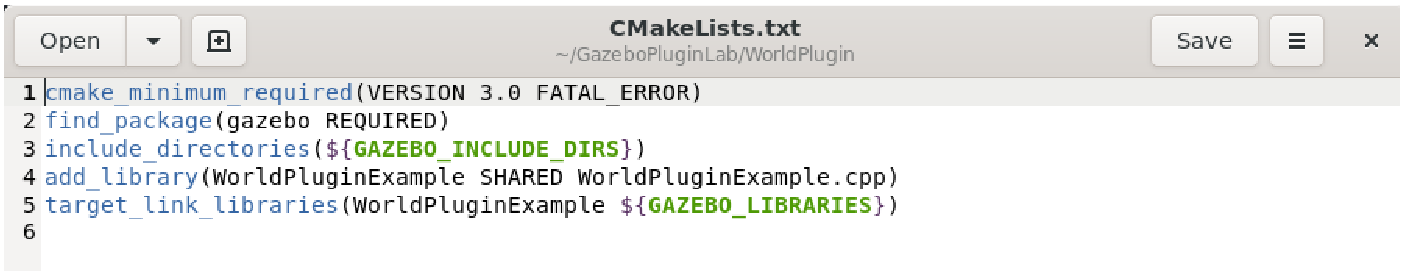
```
<?xml version="1.0" ?>
<sdf version="1.6">
  <world name="default">
    <plugin name="world_plugin" filename="libWorldPluginExample.so"/>
  </world>
</sdf>
```

b. WorldPluginExample.cpp



```cpp
#include <gazebo/gazebo.hh>
namespace gazebo {
  class WorldPluginExample : public WorldPlugin {
    public:
      WorldPluginExample() : WorldPlugin() {
        printf("Hello, World Plugin!\n");
      }
      void Load(physics::WorldPtr _world, sdf::ElementPtr _sdf) override {}
  };
  GZ_REGISTER_WORLD_PLUGIN(WorldPluginExample)
}
```

c. CMakeLists.txt

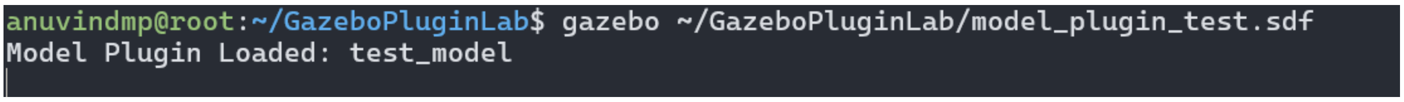


```cmake
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)
find_package(gazebo REQUIRED)
include_directories(${GAZEBO_INCLUDE_DIRS})
add_library(WorldPluginExample SHARED WorldPluginExample.cpp)
target_link_libraries(WorldPluginExample ${GAZEBO_LIBRARIES})
```

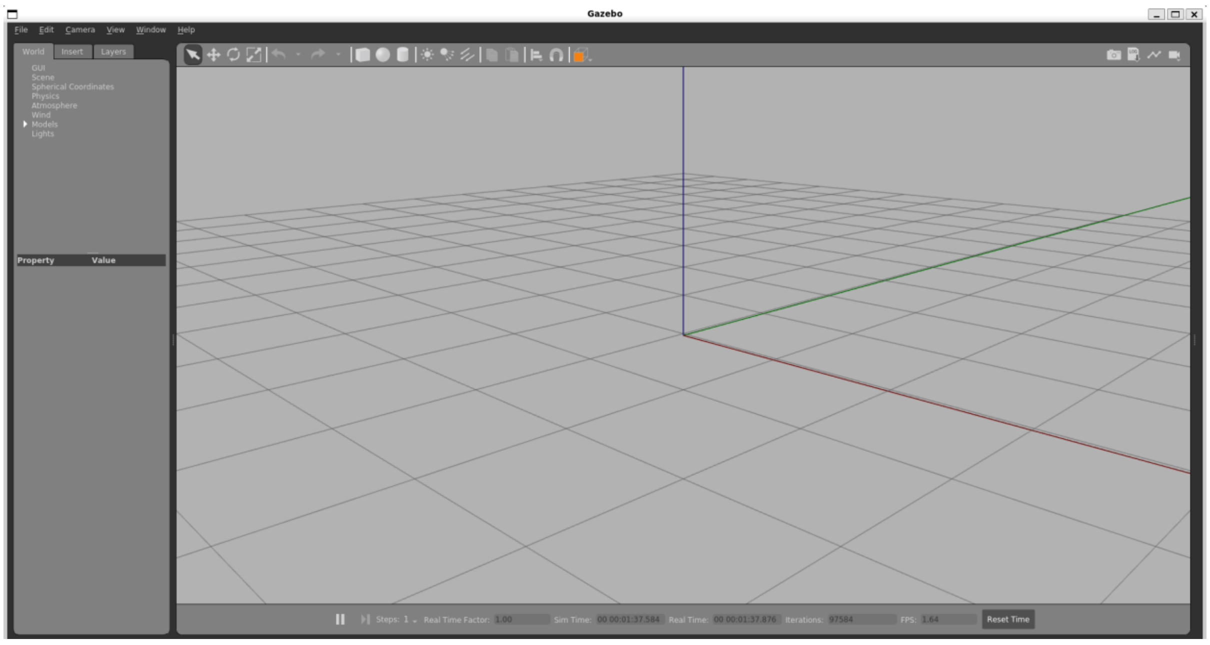
2. Model Plugin

- The **model plugin** is used to control and manipulate individual models within the Gazebo simulation.
- It provides a way to define specific behaviors for a model, such as motion, response to environmental changes, or interaction with other models.

SCREENSHOTS :

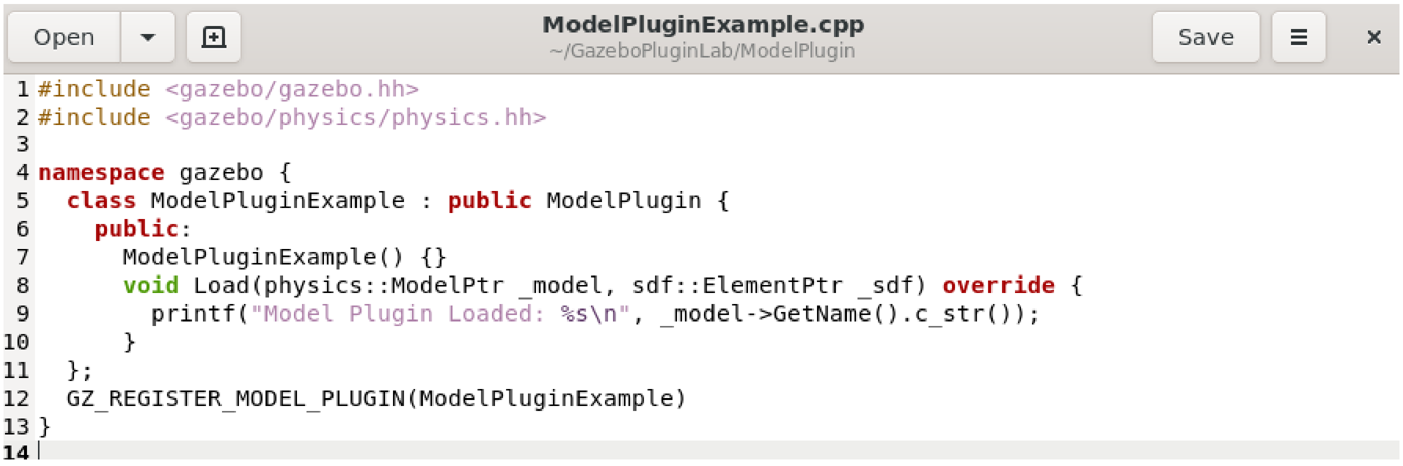


```
anuvindmp@root:~/GazeboPluginLab$ gazebo ~/GazeboPluginLab/model_plugin_test.sdf
Model Plugin Loaded: test_model
```



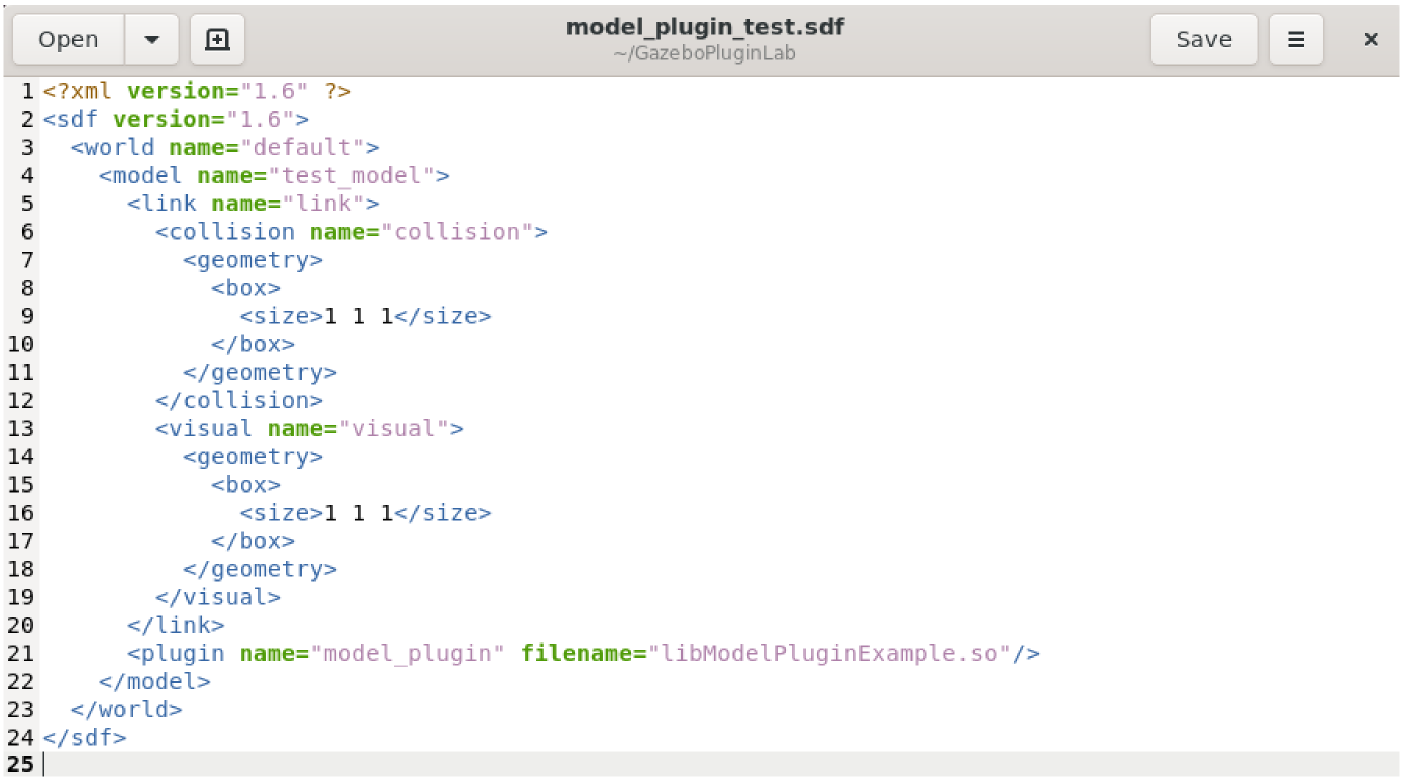
CODE :

a. ModelPluginExample.cpp



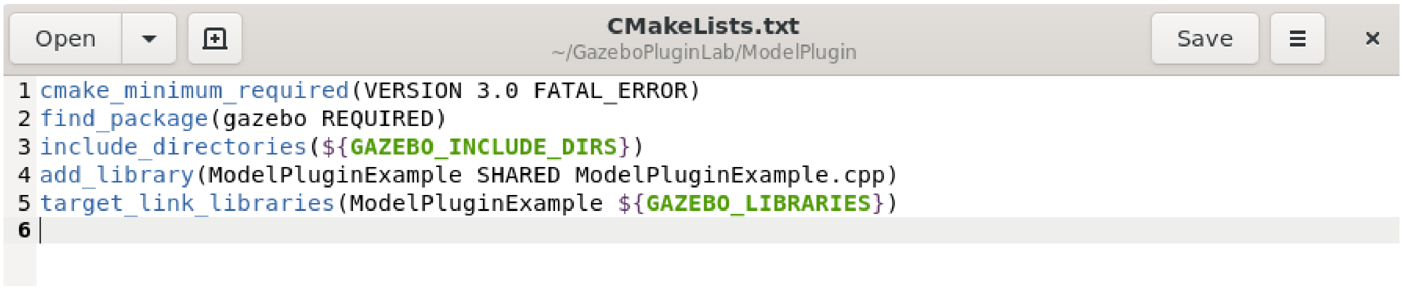
```cpp
#include <gazebo/gazebo.hh>
#include <gazebo/physics/physics.hh>

namespace gazebo {
  class ModelPluginExample : public ModelPlugin {
    public:
      ModelPluginExample() {}
      void Load(physics::ModelPtr _model, sdf::ElementPtr _sdf) override {
        printf("Model Plugin Loaded: %s\n", _model->GetName().c_str());
      }
  };
  GZ_REGISTER_MODEL_PLUGIN(ModelPluginExample)
}

```

b. Model_plugin_test.sdf



```xml
<?xml version="1.6" ?>
<sdf version="1.6">
  <world name="default">
    <model name="test_model">
      <link name="link">
        <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>
      <plugin name="model_plugin" filename="libModelPluginExample.so"/>
    </model>
  </world>
</sdf>

```

c. CMakeLists.txt

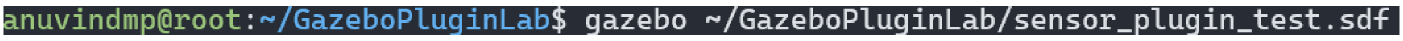


```cmake
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)
find_package(gazebo REQUIRED)
include_directories(${GAZEBO_INCLUDE_DIRS})
add_library(ModelPluginExample SHARED ModelPluginExample.cpp)
target_link_libraries(ModelPluginExample ${GAZEBO_LIBRARIES})

```

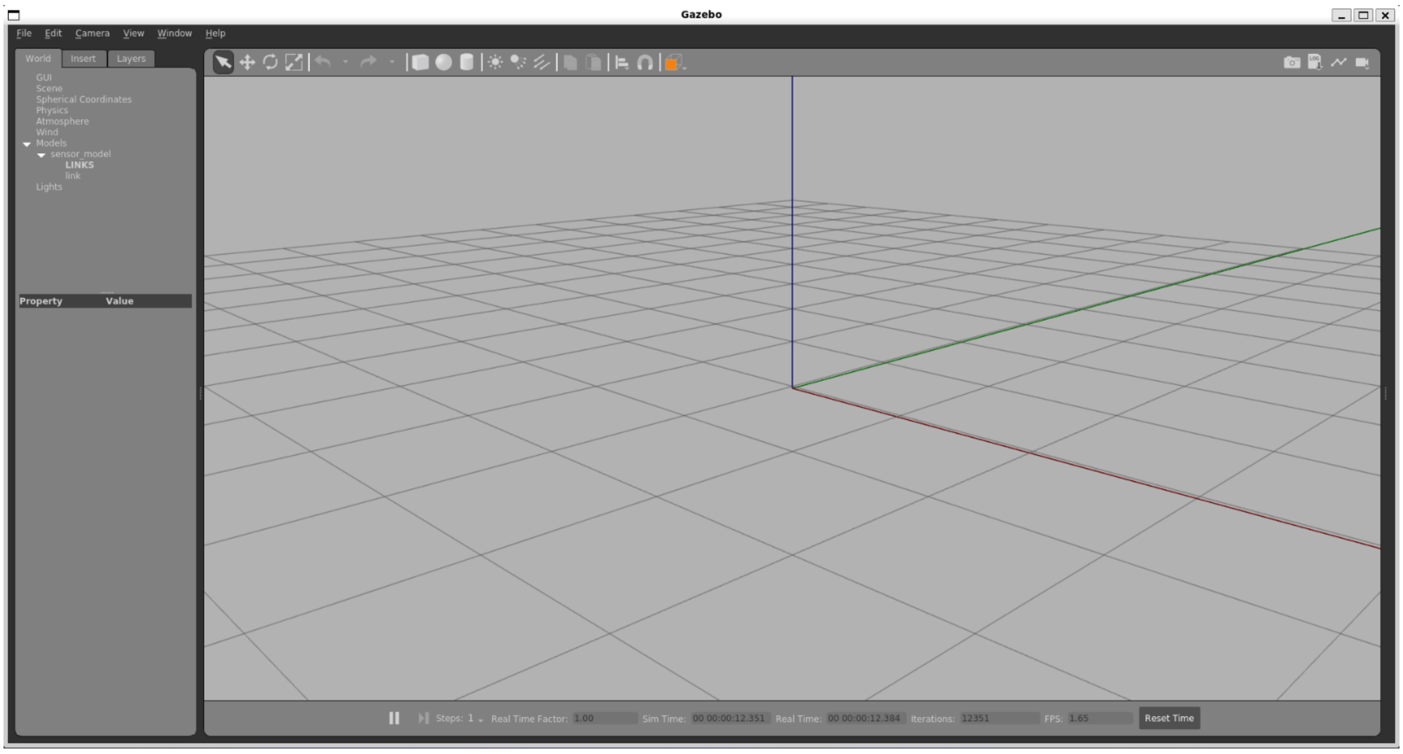
3. Sensor Plugin

- The **sensor plugin** is used to interface with and control sensors within the Gazebo simulation, such as cameras, lidar, or IMUs (Inertial Measurement Units).
- It allows for real-time data collection from sensors, enabling simulation of sensor readings and interactions with the environment.

SCREENSHOTS :



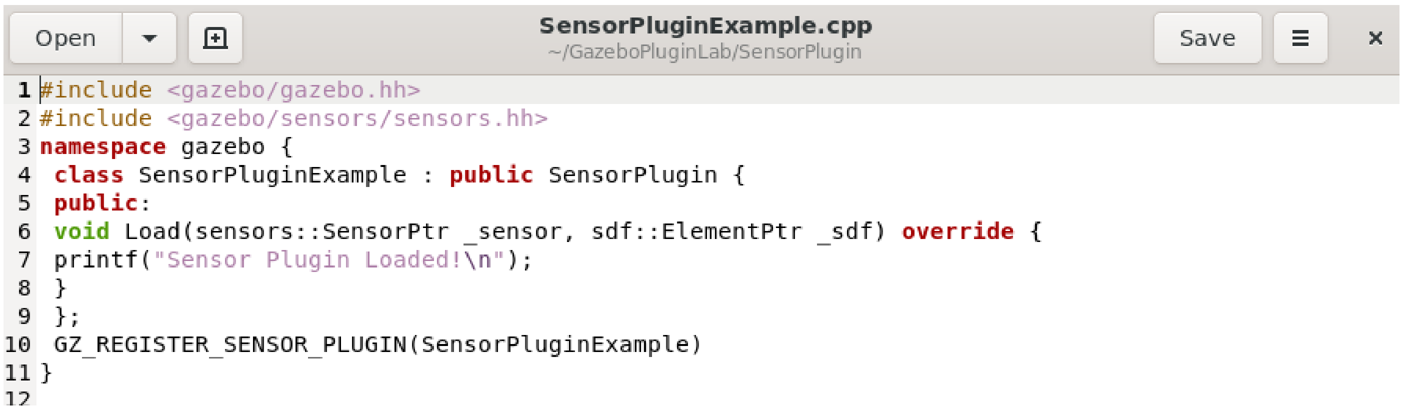
```
anuvindmp@root:~/GazeboPluginLab$ gazebo ~/GazeboPluginLab/sensor_plugin_test.sdf
```



Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:00:12.351 Real Time: 00 00:00:12.384 Iterations: 12351 FPS: 1.65

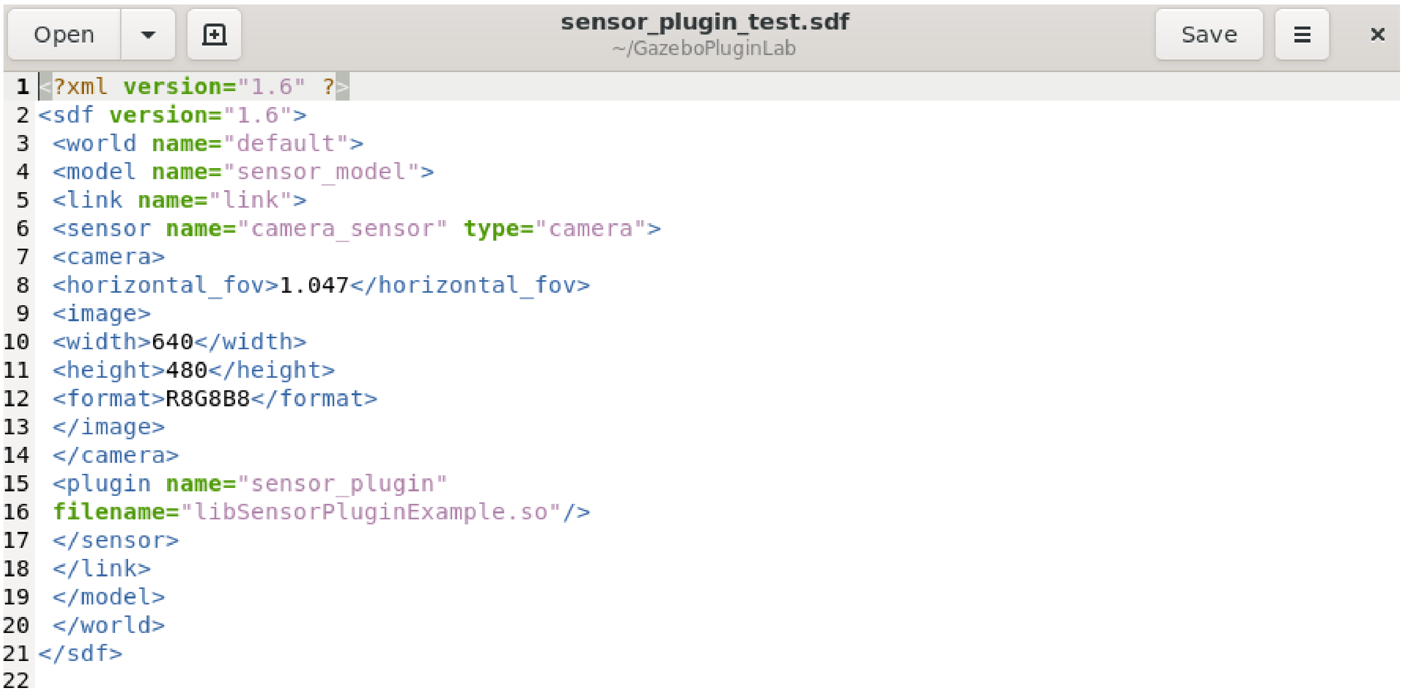
CODE :

a. SensorPluginExample.cpp



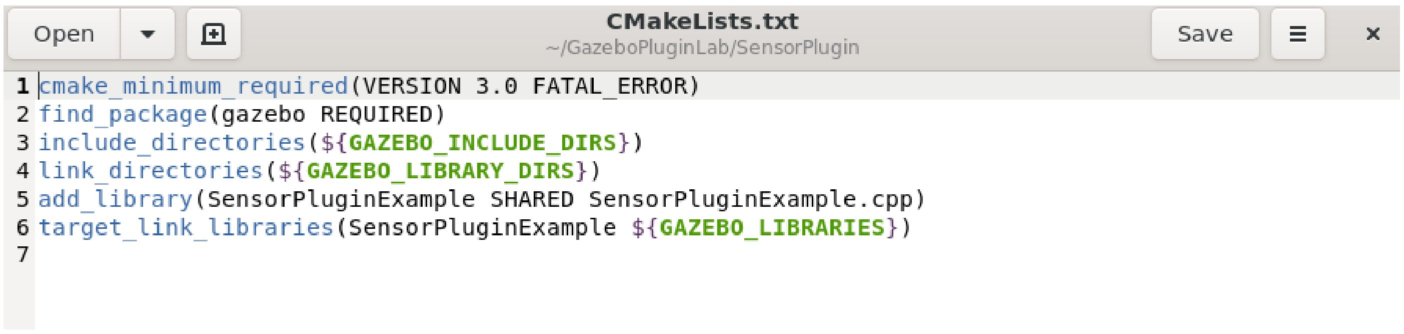
```cpp
#include <gazebo/gazebo.hh>
#include <gazebo/sensors/sensors.hh>
namespace gazebo {
 class SensorPluginExample : public SensorPlugin {
 public:
 void Load(sensors::SensorPtr _sensor, sdf::ElementPtr _sdf) override {
 printf("Sensor Plugin Loaded!\n");
 }
 };
 GZ_REGISTER_SENSOR_PLUGIN(SensorPluginExample)
}

```

b. Sensor_plugin_test.sdf



```xml
<?xml version="1.6" ?>
<sdf version="1.6">
 <world name="default">
 <model name="sensor_model">
 <link name="link">
 <sensor name="camera_sensor" type="camera">
 <camera>
 <horizontal_fov>1.047</horizontal_fov>
 <image>
 <width>640</width>
 <height>480</height>
 <format>R8G8B8</format>
 </image>
 </camera>
 <plugin name="sensor_plugin"
 filename="libSensorPluginExample.so"/>
 </sensor>
 </link>
 </model>
 </world>
</sdf>
```

c. CMakeLists.txt

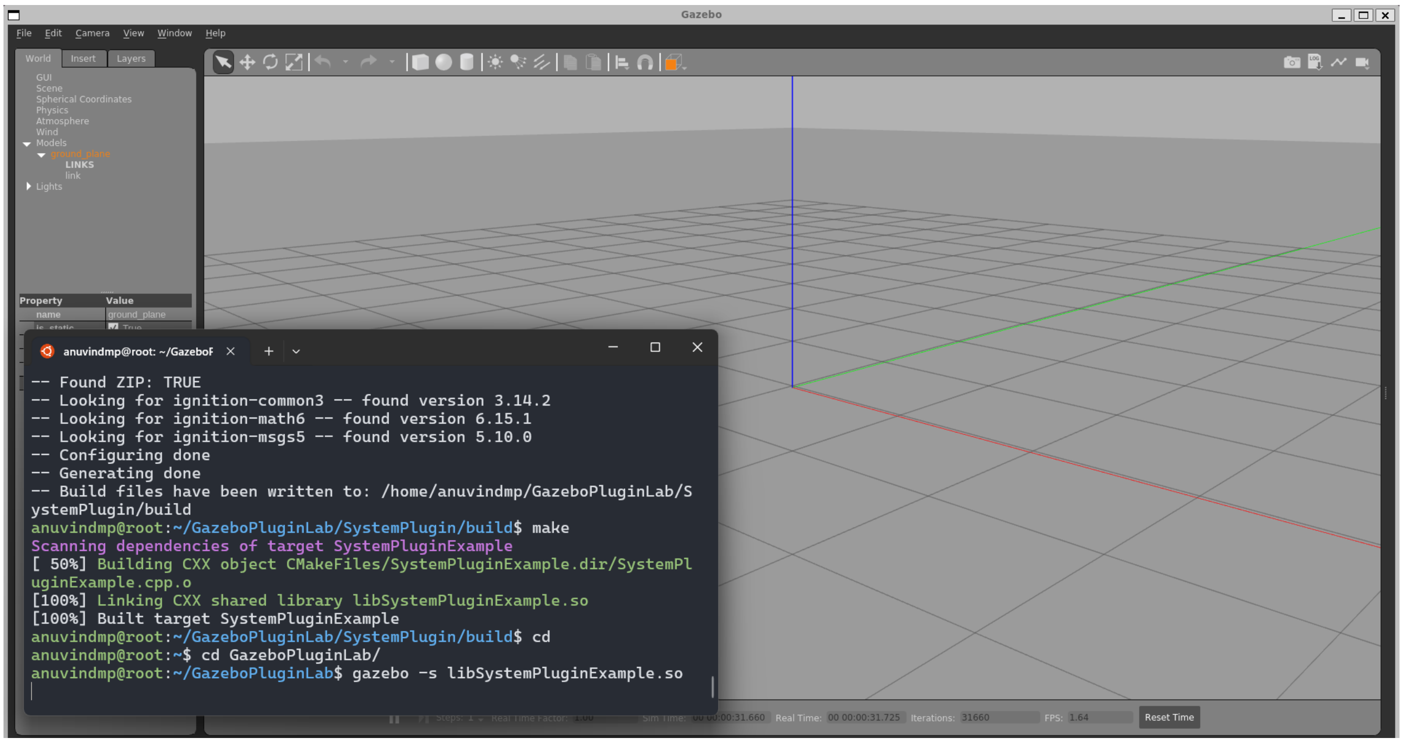


```
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)
find_package(gazebo REQUIRED)
include_directories(${GAZEBO_INCLUDE_DIRS})
link_directories(${GAZEBO_LIBRARY_DIRS})
add_library(SensorPluginExample SHARED SensorPluginExample.cpp)
target_link_libraries(SensorPluginExample ${GAZEBO_LIBRARIES})
```

4. System

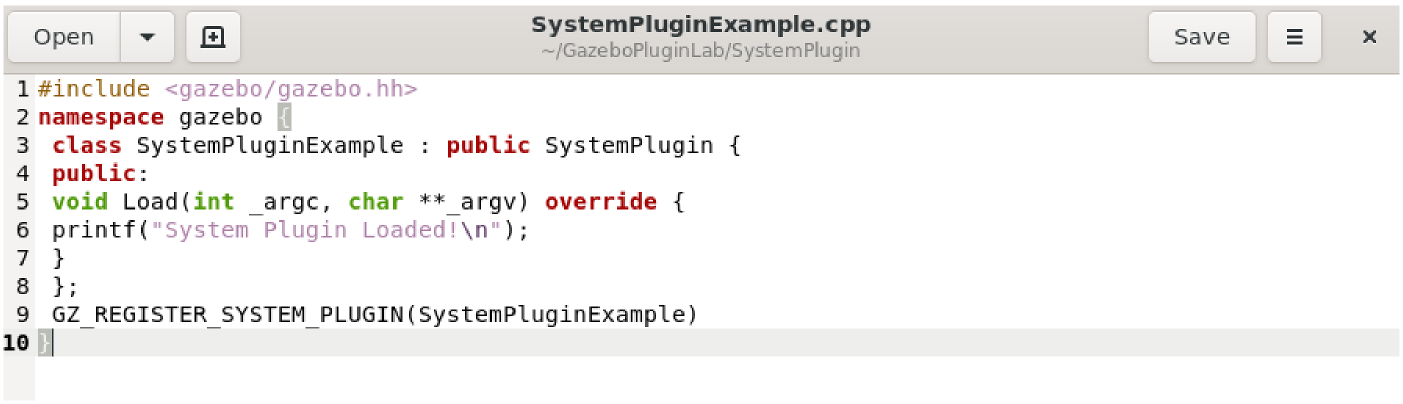
- The **system plugin** operates at the simulation level, allowing for control and customization of the entire Gazebo environment rather than individual models or sensors.
- It can be used to manage global settings, initialize multiple plugins, and handle overarching events or interactions across the simulation.
- Loaded at the start of the simulation, the system plugin provides a foundation for high-level control, ensuring seamless interaction and management of multiple components in the Gazebo world.

SCREENSHOTS:



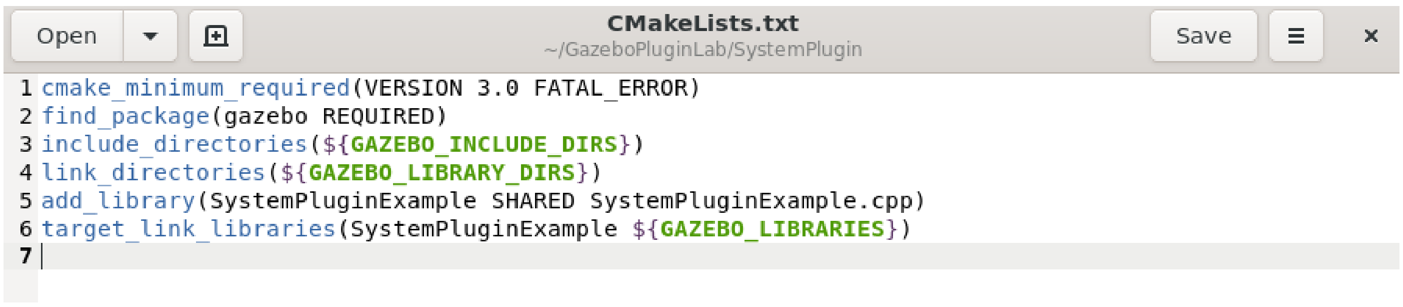
```
-- Found ZIP: TRUE
-- Looking for ignition-common3 -- found version 3.14.2
-- Looking for ignition-math6 -- found version 6.15.1
-- Looking for ignition-msgs5 -- found version 5.10.0
-- Configuring done
-- Generating done
-- Build files have been written to: /home/anuvindmp/GazeboPluginLab/S
ystemPlugin/build
anuvindmp@root:~/GazeboPluginLab/SystemPlugin/build$ make
Scanning dependencies of target SystemPluginExample
[ 50%] Building CXX object CMakeFiles/SystemPluginExample.dir/SystemPl
uginExample.cpp.o
[100%] Linking CXX shared library libSystemPluginExample.so
[100%] Built target SystemPluginExample
anuvindmp@root:~/GazeboPluginLab/SystemPlugin/build$ cd
anuvindmp@root:~$ cd GazeboPluginLab/
anuvindmp@root:~/GazeboPluginLab$ gazebo -s libSystemPluginExample.so
```

Code :



SystemPluginExample.cpp
~/GazeboPluginLab/SystemPlugin

```cpp
#include <gazebo/gazebo.hh>
namespace gazebo {
 class SystemPluginExample : public SystemPlugin {
 public:
 void Load(int _argc, char **_argv) override {
 printf("System Plugin Loaded!\n");
 }
 };
 GZ_REGISTER_SYSTEM_PLUGIN(SystemPluginExample)
}
```



CMakeLists.txt
~/GazeboPluginLab/SystemPlugin

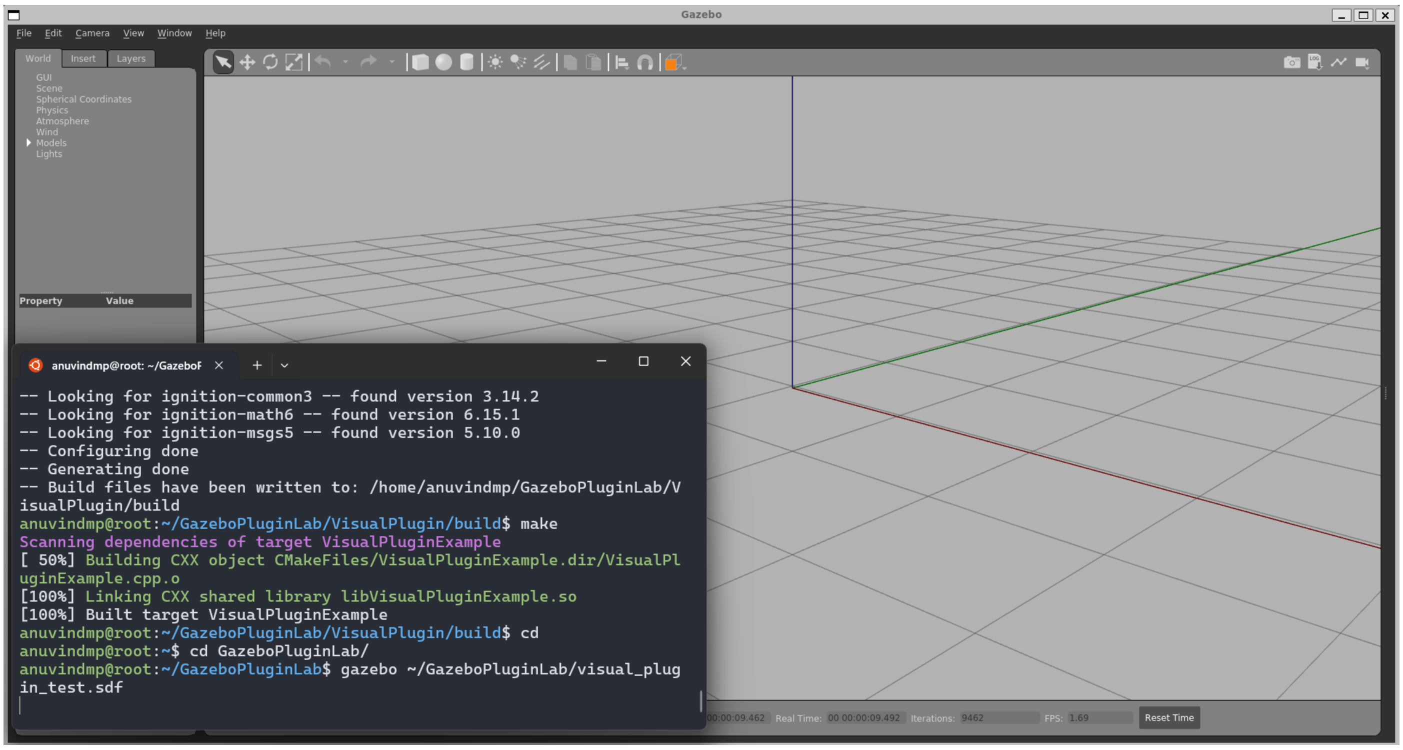
```cmake
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)
find_package(gazebo REQUIRED)
include_directories(${GAZEBO_INCLUDE_DIRS})
link_directories(${GAZEBO_LIBRARY_DIRS})
add_library(SystemPluginExample SHARED SystemPluginExample.cpp)
target_link_libraries(SystemPluginExample ${GAZEBO_LIBRARIES})
```

.sdf are not required for this

5. Visual Plugin

- The **visual plugin** is used to control and enhance the visual appearance of models and elements within the Gazebo simulation.
- It allows for modifications to rendering properties, such as **colors, textures, lighting effects,** and **animations,** providing more detailed and dynamic visuals.
- Once loaded, the visual plugin integrates with Gazebo's rendering engine, enhancing the visual elements and helping to create a more immersive simulation experience.

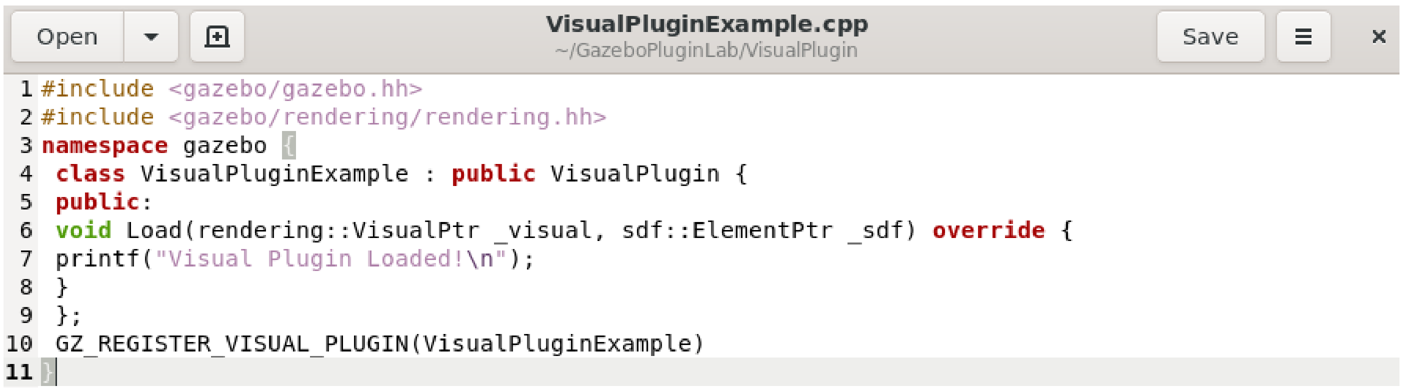
SCREENSHOT :



```
-- Looking for ignition-common3 -- found version 3.14.2
-- Looking for ignition-math6 -- found version 6.15.1
-- Looking for ignition-msgs5 -- found version 5.10.0
-- Configuring done
-- Generating done
-- Build files have been written to: /home/anuvindmp/GazeboPluginLab/V
isualPlugin/build
anuvindmp@root:~/GazeboPluginLab/VisualPlugin/build$ make
Scanning dependencies of target VisualPluginExample
[ 50%] Building CXX object CMakeFiles/VisualPluginExample.dir/VisualPl
uginExample.cpp.o
[100%] Linking CXX shared library libVisualPluginExample.so
[100%] Built target VisualPluginExample
anuvindmp@root:~/GazeboPluginLab/VisualPlugin/build$ cd
anuvindmp@root:~$ cd GazeboPluginLab/
anuvindmp@root:~/GazeboPluginLab$ gazebo ~/GazeboPluginLab/visual_plug
in_test.sdf
```

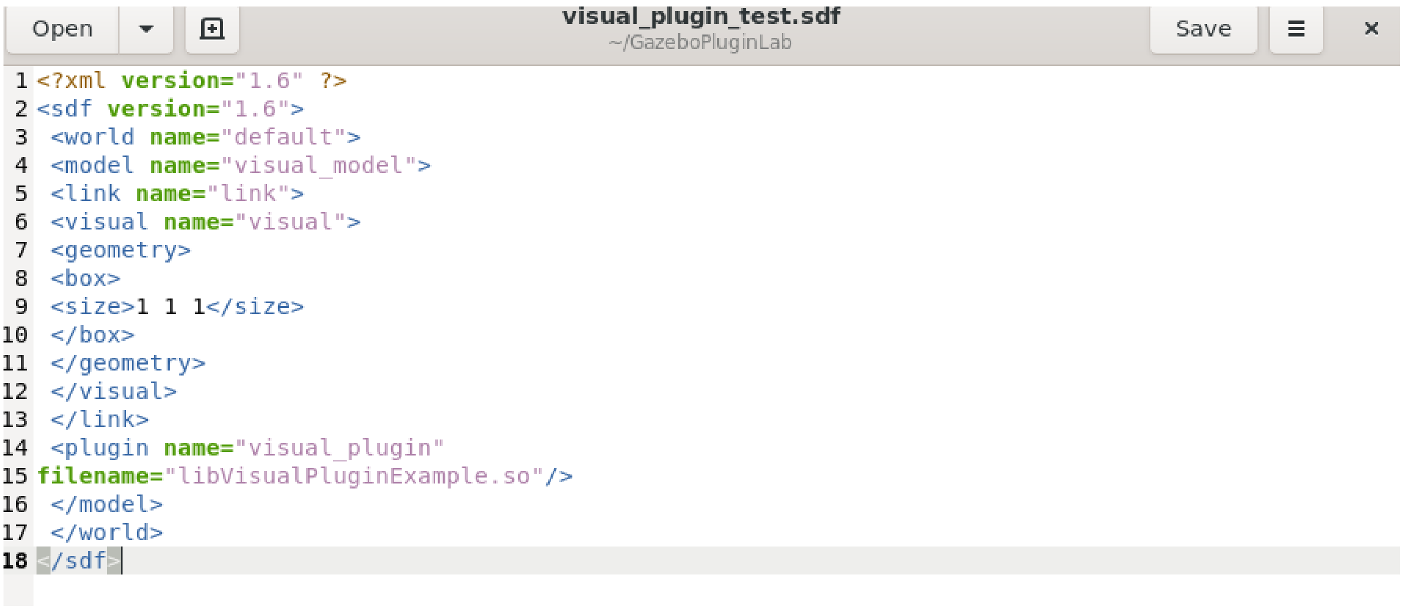
CODE :

a. VisualPluginExample.cpp



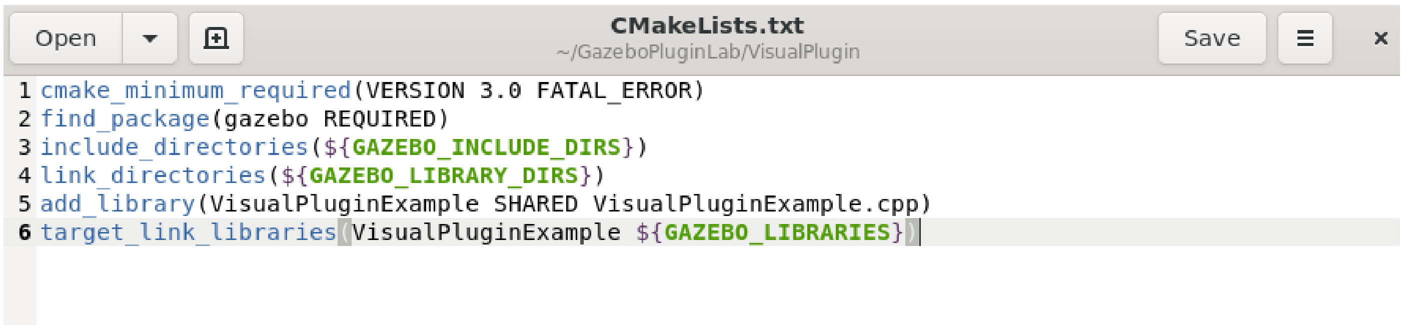
```cpp
#include <gazebo/gazebo.hh>
#include <gazebo/rendering/rendering.hh>
namespace gazebo {
 class VisualPluginExample : public VisualPlugin {
 public:
 void Load(rendering::VisualPtr _visual, sdf::ElementPtr _sdf) override {
 printf("Visual Plugin Loaded!\n");
 }
 };
 GZ_REGISTER_VISUAL_PLUGIN(VisualPluginExample)
}
```

b. visual_plugin_test.sdf



```xml
<?xml version="1.6" ?>
<sdf version="1.6">
 <world name="default">
 <model name="visual_model">
 <link name="link">
 <visual name="visual">
 <geometry>
 <box>
 <size>1 1 1</size>
 </box>
 </geometry>
 </visual>
 </link>
 <plugin name="visual_plugin"
filename="libVisualPluginExample.so"/>
 </model>
 </world>
</sdf>
```

c. CMakeLists.txt

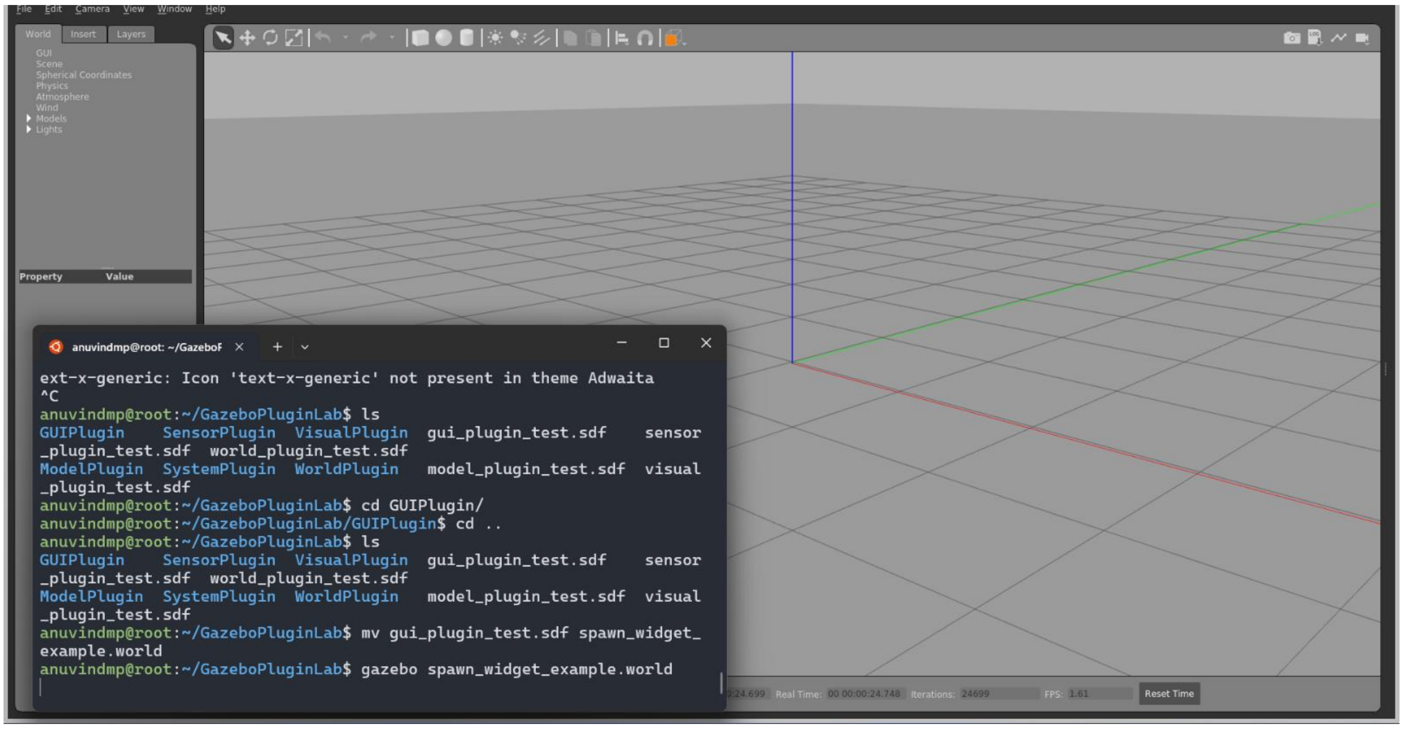


```cmake
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)
find_package(gazebo REQUIRED)
include_directories(${GAZEBO_INCLUDE_DIRS})
link_directories(${GAZEBO_LIBRARY_DIRS})
add_library(VisualPluginExample SHARED VisualPluginExample.cpp)
target_link_libraries(VisualPluginExample ${GAZEBO_LIBRARIES})
```

6. GUI plugin

- The **GUI plugin** is used to customize and extend the Gazebo user interface, allowing for added interactive controls, widgets, and visualization tools.
- It enables developers to create custom UI elements, such as **buttons, sliders, and panels**, that interact directly with the simulation and enhance user control.
- Loaded alongside the main GUI, the GUI plugin offers a powerful way to make the simulation more interactive and user-friendly, enabling custom functionality beyond the default Gazebo interface.

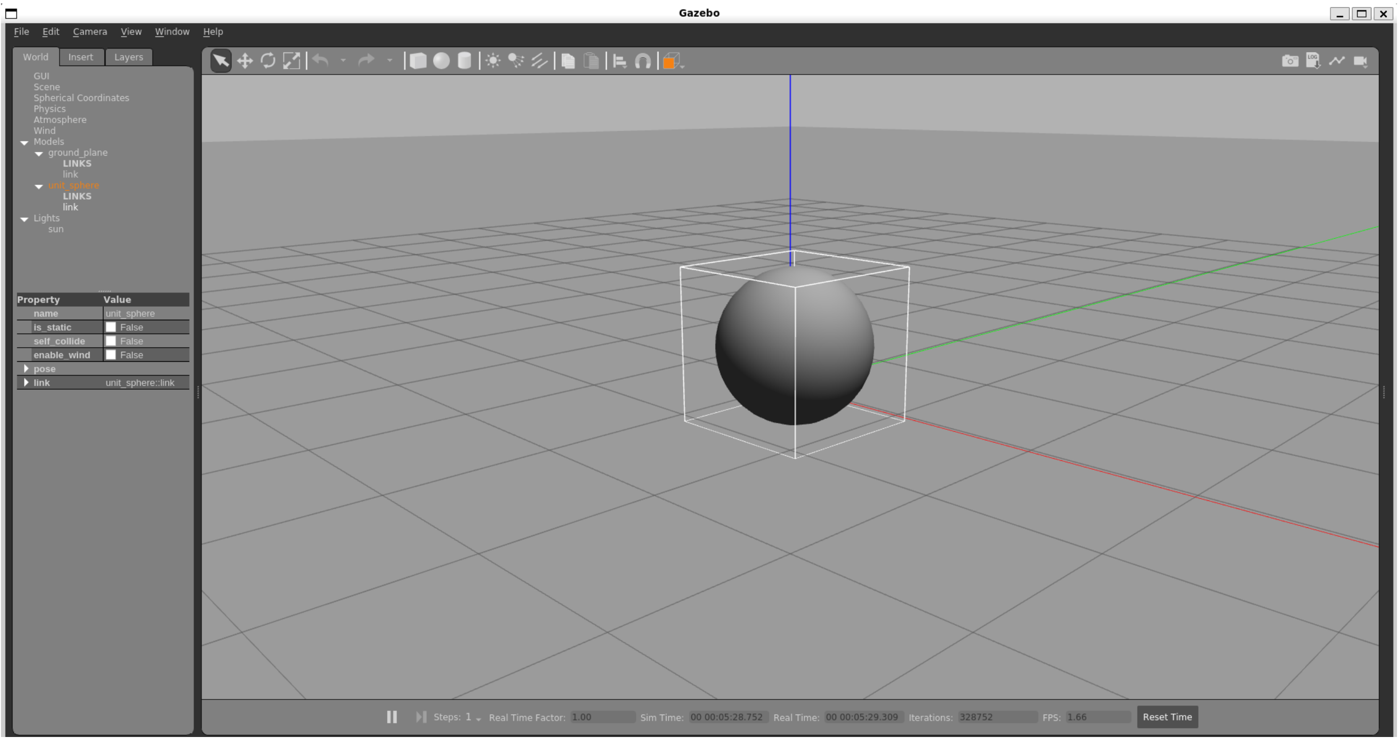
Screenshots :



```
ext-x-generic: Icon 'text-x-generic' not present in theme Adwaita
^C
anuvindmp@root:~/GazeboPluginLab$ ls
GUIPlugin    SensorPlugin  VisualPlugin  gui_plugin_test.sdf    sensor
_plugin_test.sdf  world_plugin_test.sdf
ModelPlugin  SystemPlugin  WorldPlugin   model_plugin_test.sdf  visual
_plugin_test.sdf
anuvindmp@root:~/GazeboPluginLab$ cd GUIPlugin/
anuvindmp@root:~/GazeboPluginLab/GUIPlugin$ cd ..
anuvindmp@root:~/GazeboPluginLab$ ls
GUIPlugin    SensorPlugin  VisualPlugin  gui_plugin_test.sdf    sensor
_plugin_test.sdf  world_plugin_test.sdf
ModelPlugin  SystemPlugin  WorldPlugin   model_plugin_test.sdf  visual
_plugin_test.sdf
anuvindmp@root:~/GazeboPluginLab$ mv gui_plugin_test.sdf spawn_widget_
example.world
anuvindmp@root:~/GazeboPluginLab$ gazebo spawn_widget_example.world
```

```
:24.699  Real Time: 00 00:00:24.748  Iterations: 24699  FPS: 1.61
```

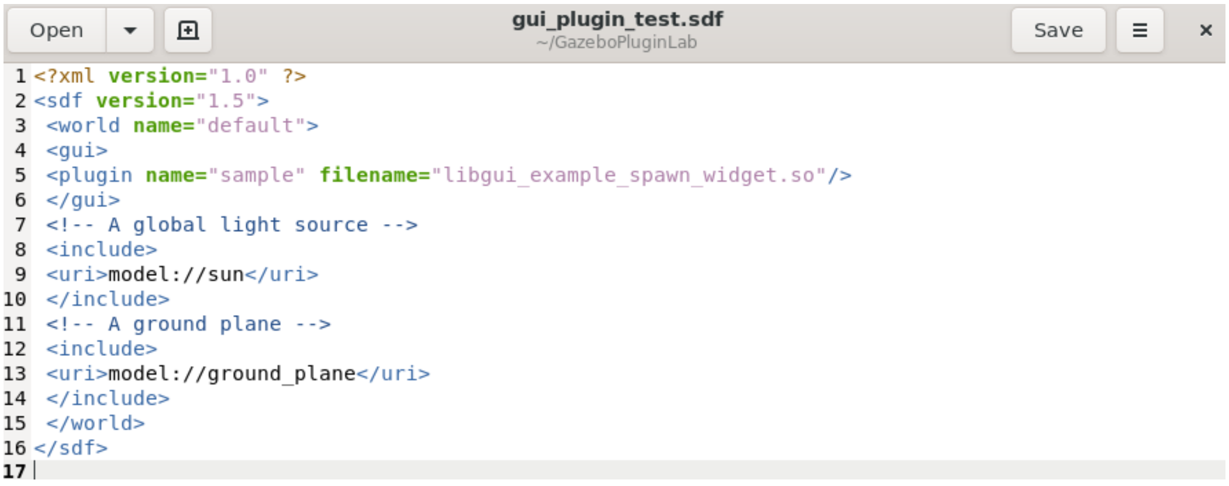
Sphere :



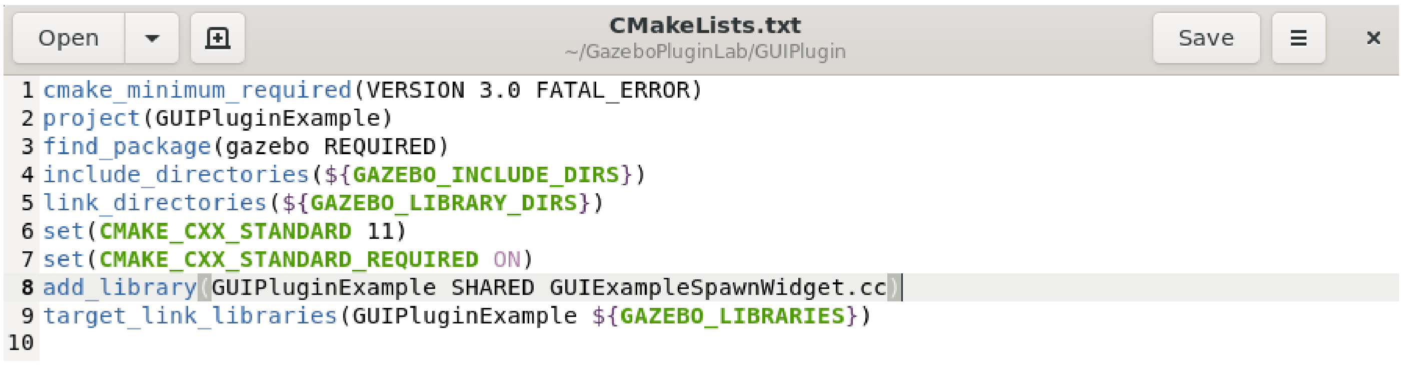
Property	Value
name	unit_sphere
is_static	False
self_collide	False
enable_wind	False
pose	
link	unit_sphere::link

```
Steps: 1  Real Time Factor: 1.00  Sim Time: 00 00:05:28.752  Real Time: 00 00:05:29.309  Iterations: 328752  FPS: 1.66
```

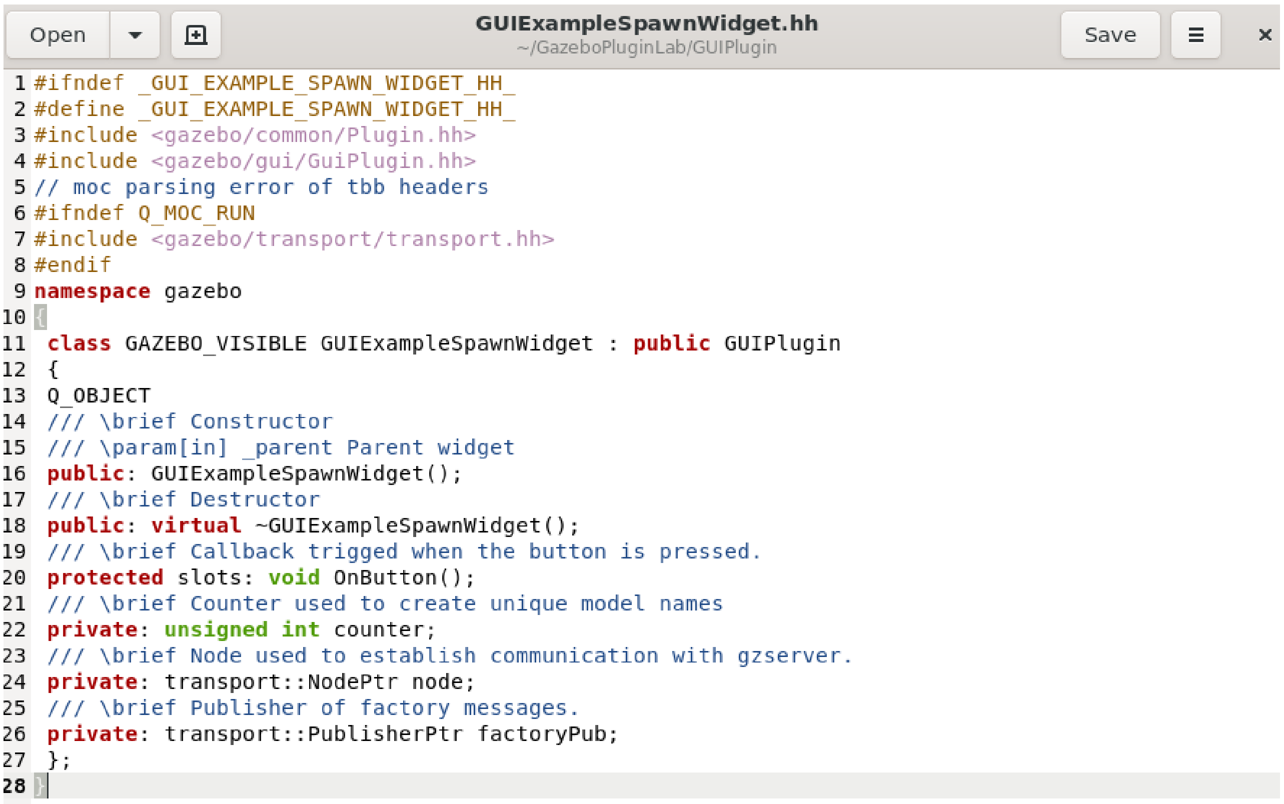
Code :



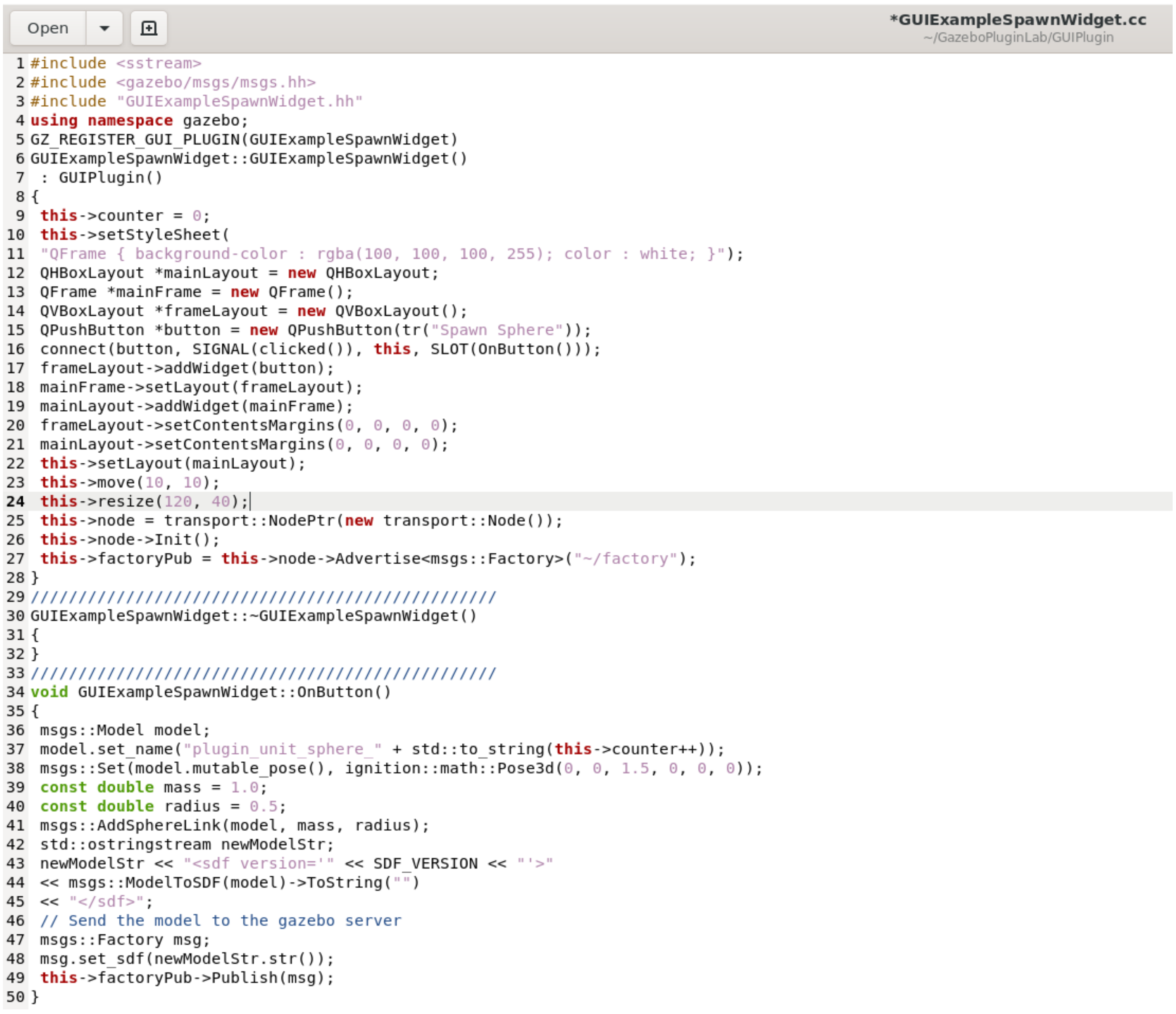
```
<?xml version="1.0" ?>
<sdf version="1.5">
 <world name="default">
 <gui>
 <plugin name="sample" filename="libgui_example_spawn_widget.so"/>
 </gui>
 <!-- A global light source -->
 <include>
 <uri>model://sun</uri>
 </include>
 <!-- A ground plane -->
 <include>
 <uri>model://ground_plane</uri>
 </include>
 </world>
</sdf>

```



```
cmake_minimum_required(VERSION 3.0 FATAL_ERROR)
project(GUIPluginExample)
find_package(gazebo REQUIRED)
include_directories(${GAZEBO_INCLUDE_DIRS})
link_directories(${GAZEBO_LIBRARY_DIRS})
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
add_library(GUIPluginExample SHARED GUIExampleSpawnWidget.cc)
target_link_libraries(GUIPluginExample ${GAZEBO_LIBRARIES})

```



```
#ifndef _GUI_EXAMPLE_SPAWN_WIDGET_HH_
#define _GUI_EXAMPLE_SPAWN_WIDGET_HH_
#include <gazebo/common/Plugin.hh>
#include <gazebo/gui/GuiPlugin.hh>
// moc parsing error of tbb headers
#ifndef Q_MOC_RUN
#include <gazebo/transport/transport.hh>
#endif
namespace gazebo
{
 class GAZEBO_VISIBLE GUIExampleSpawnWidget : public GUIPlugin
 {
 Q_OBJECT
 /// \brief Constructor
 /// \param[in] _parent Parent widget
 public: GUIExampleSpawnWidget();
 /// \brief Destructor
 public: virtual ~GUIExampleSpawnWidget();
 /// \brief Callback trigged when the button is pressed.
 protected slots: void OnButton();
 /// \brief Counter used to create unique model names
 private: unsigned int counter;
 /// \brief Node used to establish communication with gzserver.
 private: transport::NodePtr node;
 /// \brief Publisher of factory messages.
 private: transport::PublisherPtr factoryPub;
 };
}
```



```cpp
#include <sstream>
#include <gazebo/msgs/msgs.hh>
#include "GUIExampleSpawnWidget.hh"
using namespace gazebo;
GZ_REGISTER_GUI_PLUGIN(GUIExampleSpawnWidget)
GUIExampleSpawnWidget::GUIExampleSpawnWidget()
 : GUIPlugin()
{
 this->counter = 0;
 this->setStyleSheet(
 "QFrame { background-color : rgba(100, 100, 100, 255); color : white; }");
 QHBoxLayout *mainLayout = new QHBoxLayout;
 QFrame *mainFrame = new QFrame();
 QVBoxLayout *frameLayout = new QVBoxLayout();
 QPushButton *button = new QPushButton(tr("Spawn Sphere"));
 connect(button, SIGNAL(clicked()), this, SLOT(OnButton()));
 frameLayout->addWidget(button);
 mainFrame->setLayout(frameLayout);
 mainLayout->addWidget(mainFrame);
 frameLayout->setContentsMargins(0, 0, 0, 0);
 mainLayout->setContentsMargins(0, 0, 0, 0);
 this->setLayout(mainLayout);
 this->move(10, 10);
 this->resize(120, 40);
 this->node = transport::NodePtr(new transport::Node());
 this->node->Init();
 this->factoryPub = this->node->Advertise<msgs::Factory>("~/factory");
}
/////////////////////////////////////////////////
GUIExampleSpawnWidget::~GUIExampleSpawnWidget()
{
}
/////////////////////////////////////////////////
void GUIExampleSpawnWidget::OnButton()
{
 msgs::Model model;
 model.set_name("plugin_unit_sphere_" + std::to_string(this->counter++));
 msgs::Set(model.mutable_pose(), ignition::math::Pose3d(0, 0, 1.5, 0, 0, 0));
 const double mass = 1.0;
 const double radius = 0.5;
 msgs::AddSphereLink(model, mass, radius);
 std::ostringstream newModelStr;
 newModelStr << "<sdf version='" << SDF_VERSION << "'>"
 << msgs::ModelToSDF(model)->ToString("")
 << "</sdf>";
 // Send the model to the gazebo server
 msgs::Factory msg;
 msg.set_sdf(newModelStr.str());
 this->factoryPub->Publish(msg);
}
```