-22AIE442-
ROBOTIC OPERATING SYSTEMS AND ROBOTIC SIMULATIONS

Labsheet – 5

Exploring Gazebo Plugins

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

Gazebo Plugins

0. Prerequisites

Gazebo plugins are essential for extending the functionalities of the Gazebo simulation environment, enabling control over various aspects of the simulation, such as the world environment, model behaviours, sensors, system parameters, and user interface. The different types of Gazebo plugins available: World, Model, Sensor, System, Visual, and GUI.

$ sudo apt install ros-noetic-gazebo-plugins

```
giriirig@DESKTOP-O7BKI3U:~$ sudo apt install ros-noetic-gazebo-plugins
[sudo] password for giriirig:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  ros-noetic-gazebo-plugins
1 upgraded, 0 newly installed, 0 to remove and 244 not upgraded.
Need to get 1717 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://packages.ros.org/ros/ubuntu focal/main amd64 ros-noetic-gazebo-plugins amd64 2.9.2-1focal.20240913.201633 [
1717 kB]
Fetched 1717 kB in 7s (231 kB/s)
(Reading database ... 149306 files and directories currently installed.)
Preparing to unpack .../ros-noetic-gazebo-plugins_2.9.2-1focal.20240913.201633_amd64.deb ...
Unpacking ros-noetic-gazebo-plugins (2.9.2-1focal.20240913.201633) over (2.9.2-1focal.20240111.182925) ...
Setting up ros-noetic-gazebo-plugins (2.9.2-1focal.20240913.201633) ...
giriirig@DESKTOP-O7BKI3U:~$
```

$ mkdir -p ~/GazeboPluginLab
$ cd ~/GazeboPluginLab
$ mkdir -p WorldPlugin/build
$ mkdir -p ModelPlugin/build
$ mkdir -p SensorPlugin/build
$ mkdir -p SystemPlugin/build
$ mkdir -p VisualPlugin/build
$ mkdir -p GUIPlugin/build
$ mkdir sdf

$ export GAZEBO_PLUGIN_PATH=~/GazeboPluginLab/WorldPlugin/build:~/GazeboPluginLab/ModelPlugin/build:~/GazeboPluginLab/SensorPlugin/build:~/GazeboPluginLab/SystemPlugin/build:~/GazeboPluginLab/VisualPlugin/build:~/GazeboPluginLab/GUIPlugin/build

```
giriirig@DESKTOP-O7BKI3U:~$ mkdir -p ~/GazeboPluginLab
giriirig@DESKTOP-O7BKI3U:~$ cd ~/GazeboPluginLab
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab$ mkdir -p WorldPlugin/build
build
mkdir -p SensorPlugin/build
mkdir -p SystemPlugin/build
mkdir -p VisualPlugin/build
mkdir -p GUIPlugin/build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab$ mkdir -p ModelPlugin/build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab$ mkdir -p SensorPlugin/build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab$ mkdir -p SystemPlugin/build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab$ mkdir -p VisualPlugin/build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab$ mkdir -p GUIPlugin/build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab$ ls
GUIPlugin  ModelPlugin  SensorPlugin  SystemPlugin  VisualPlugin  WorldPlugin
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab$ mkdir sdf
```

1. World Plugin

World plugins control properties and behaviours across the entire simulation world. They can adjust environmental factors such as gravity, lighting, and atmosphere.

$ cd ~/GazeboPluginLab/WorldPlugin

$ nano WorldPluginExample.cpp

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

$ nano CMakeLists.txt

```
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})
```

$ cd build

$ cmake ..

$ make

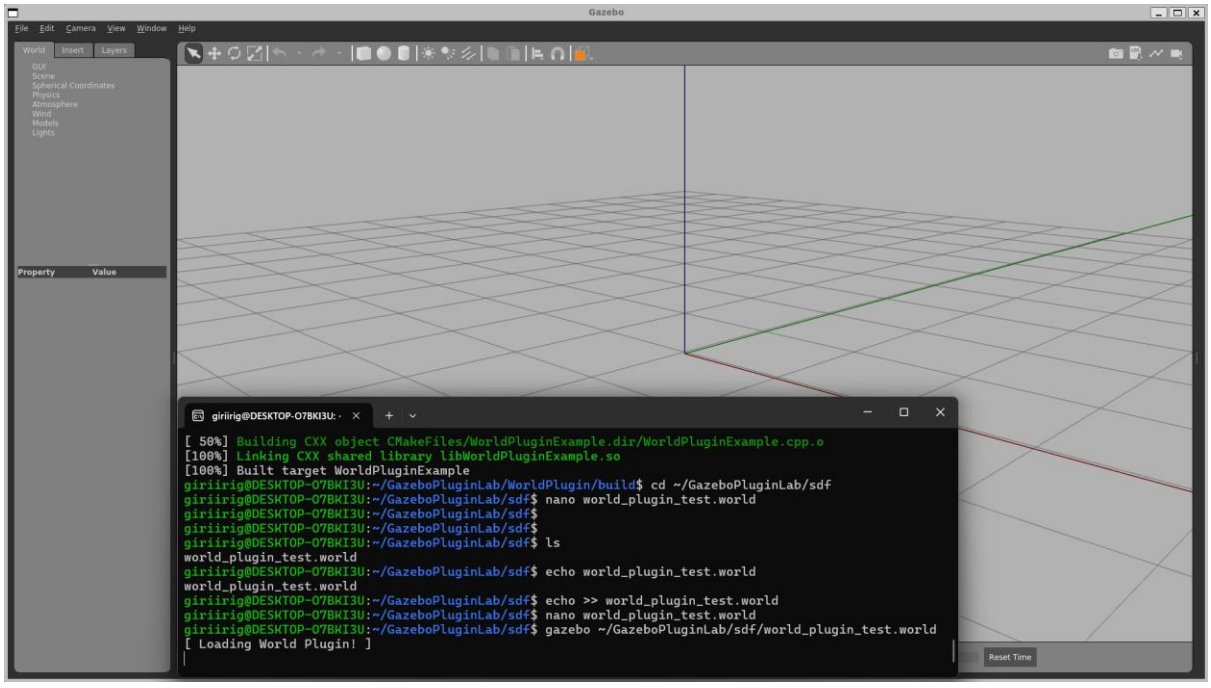
```
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab$ cd ~/GazeboPluginLab/WorldPlugin
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/WorldPlugin$ nano WorldPluginExample.cpp
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/WorldPlugin$ nano CMakeLists.txt
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/WorldPlugin$ cd build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/WorldPlugin/build$ cmake..
cmake..: command not found
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/WorldPlugin/build$ cmake ..
CMake Warning (dev) in CMakeLists.txt:
  No project() command is present.  The top-level CMakeLists.txt file must
  contain a literal, direct call to the project() command.  Add a line of
  code such as

    project(ProjectName)

  near the top of the file, but after cmake_minimum_required().

  CMake is pretending there is a "project(Project)" command on the first
  line.
This warning is for project developers.  Use -Wno-dev to suppress it.

-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
```

$ cd ~/GazeboPluginLab/sdf

$ nano world_plugin_test.world

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

$ gazebo ~/GazeboPluginLab/sdf/world_plugin_test.world



2. Model Plugin

Model plugins allow control and customization of individual models (e.g., robots, objects) within the simulation. They enable direct manipulation of model behaviours, such as applying forces, controlling joints, managing state changes, position, velocity, and behaviour.

$ cd ~/GazeboPluginLab/ModelPlugin

$ nano ModelPluginExample.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("[ Loaded Model Plugin ] : %s\n", _model->GetName().c_str());
      }
  };
  GZ_REGISTER_MODEL_PLUGIN(ModelPluginExample)
}
```

$ nano CMakeLists.txt

```
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})
```

$ cd build

$ cmake ..

$ make

```
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/sdf$ cd ~/GazeboPluginLab/ModelPlugin
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/ModelPlugin$ nano ModelPluginExample.cpp
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/ModelPlugin$ nano ModelPluginExample.cpp
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/ModelPlugin$ nano CMakeLists.txt
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/ModelPlugin$ cd build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/ModelPlugin/build$ cmake ..
CMake Warning (dev) in CMakeLists.txt:
  No project() command is present.  The top-level CMakeLists.txt file must
  contain a literal, direct call to the project() command.  Add a line of
  code such as

    project(ProjectName)

  near the top of the file, but after cmake_minimum_required().

  CMake is pretending there is a "project(Project)" command on the first
  line.
This warning is for project developers.  Use -Wno-dev to suppress it.

-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Checking for module 'bullet>=2.82'
--   Found bullet, version 2.88
```

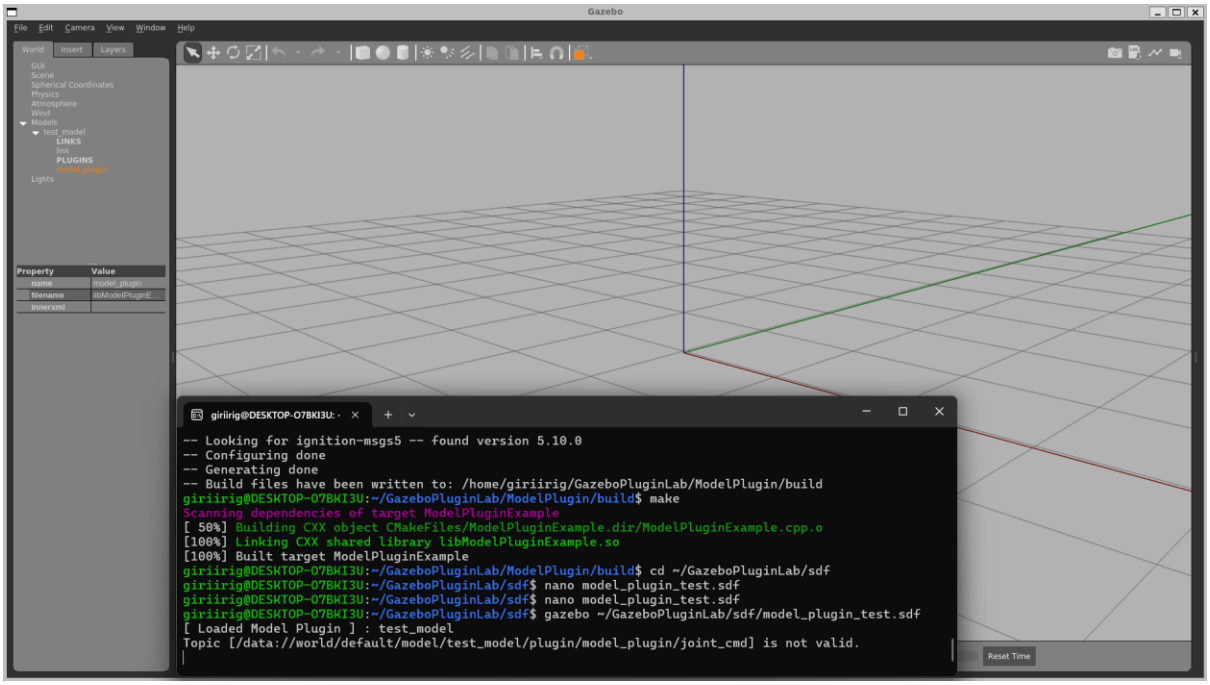
$ cd ~/GazeboPluginLab/sdf

$ nano model_plugin_test.sdf

```
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/ModelPlugin/build$ make
Scanning dependencies of target ModelPluginExample
[ 50%] Building CXX object CMakeFiles/ModelPluginExample.dir/ModelPluginExample.cpp.o
[100%] Linking CXX shared library libModelPluginExample.so
[100%] Built target ModelPluginExample
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/ModelPlugin/build$ cd ~/GazeboPluginLab/sdf
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/sdf$ nano model_plugin_test.sdf
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/sdf$
```

```
<?xml version="1.0" ?>
<sdf version="1.6">
  <world name="default">
    <model name="test_model">
      <link name="link">
        <visual name="visual">
          <geometry>
            <box>
              <size>1 1 1</size>
            </box>
          </geometry>
        </visual>
      </link>
      <plugin name="model_plugin" filename="libModelPluginExample.so"/>
    </model>
  </world>
</sdf>
```

$ gazebo ~/GazeboPluginLab/sdf/model_plugin_test.sdf



3. Sensor Plugin

The Sensor plugins provide customization for different sensors like cameras, lasers, LiDARs, and GPS devices. They process sensor data, customize output, and adjust sensor properties.

$ cd ~/GazeboPluginLab/SensorPlugin

$ nano SensorPluginExample.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("[ Loaded Sensor Plugin! ]\n");
      }
  };
  GZ_REGISTER_SENSOR_PLUGIN(SensorPluginExample)
}
```

$ nano CMakeLists.txt

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

$ cd build

$ cmake ..

$ make

```
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/sdf$ cd ~/GazeboPluginLab/SensorPlugin
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SensorPlugin$ nano SensorPluginExample.cpp
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SensorPlugin$ nano CMakeLists.txt
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SensorPlugin$ cd build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SensorPlugin/build$ cmake ..
CMake Warning (dev) in CMakeLists.txt:
  No project() command is present.  The top-level CMakeLists.txt file must
  contain a literal, direct call to the project() command.  Add a line of
  code such as

    project(ProjectName)

  near the top of the file, but after cmake_minimum_required().

  CMake is pretending there is a "project(Project)" command on the first
  line.
This warning is for project developers.  Use -Wno-dev to suppress it.

-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Checking for module 'bullet>=2.82'
--   Found bullet, version 2.88
```

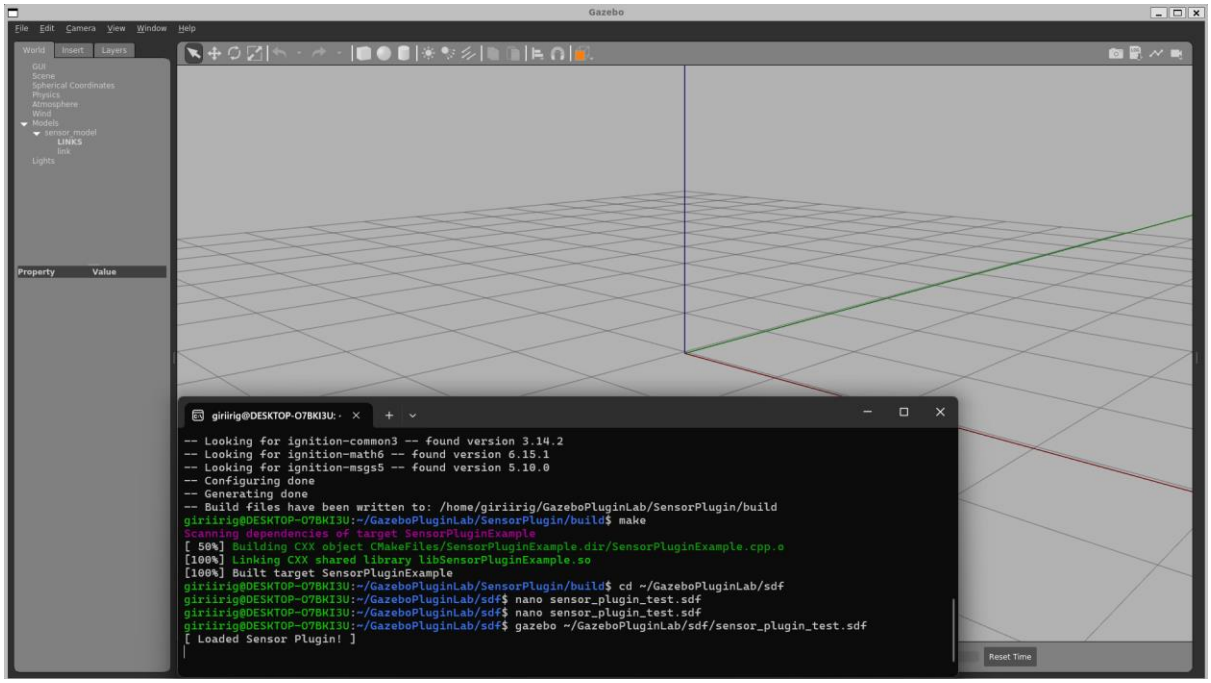
$ cd ~/GazeboPluginLab/sdf

$ nano sensor_plugin_test.sdf

```
-- Configuring done
-- Generating done
-- Build files have been written to: /home/giriirig/GazeboPluginLab/SensorPlugin/build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SensorPlugin/build$ make
Scanning dependencies of target SensorPluginExample
[ 50%] Building CXX object CMakeFiles/SensorPluginExample.dir/SensorPluginExample.cpp.o
[100%] Linking CXX shared library libSensorPluginExample.so
[100%] Built target SensorPluginExample
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SensorPlugin/build$ cd ~/GazeboPluginLab/sdf
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/sdf$ nano sensor_plugin_test.sdf
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/sdf$
```

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

$ gazebo ~/GazeboPluginLab/sdf/sensor_plugin_test.sdf



4. System Plugin

System plugins can control and access all aspects of the Gazebo server. They manage the simulation's state, add event handlers, and interface with external systems, enabling system-wide control and event handling.

$ cd ~/GazeboPluginLab/SystemPlugin

$ nano SystemPluginExample.cpp

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

$ nano CMakeLists.txt

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

$ cd build

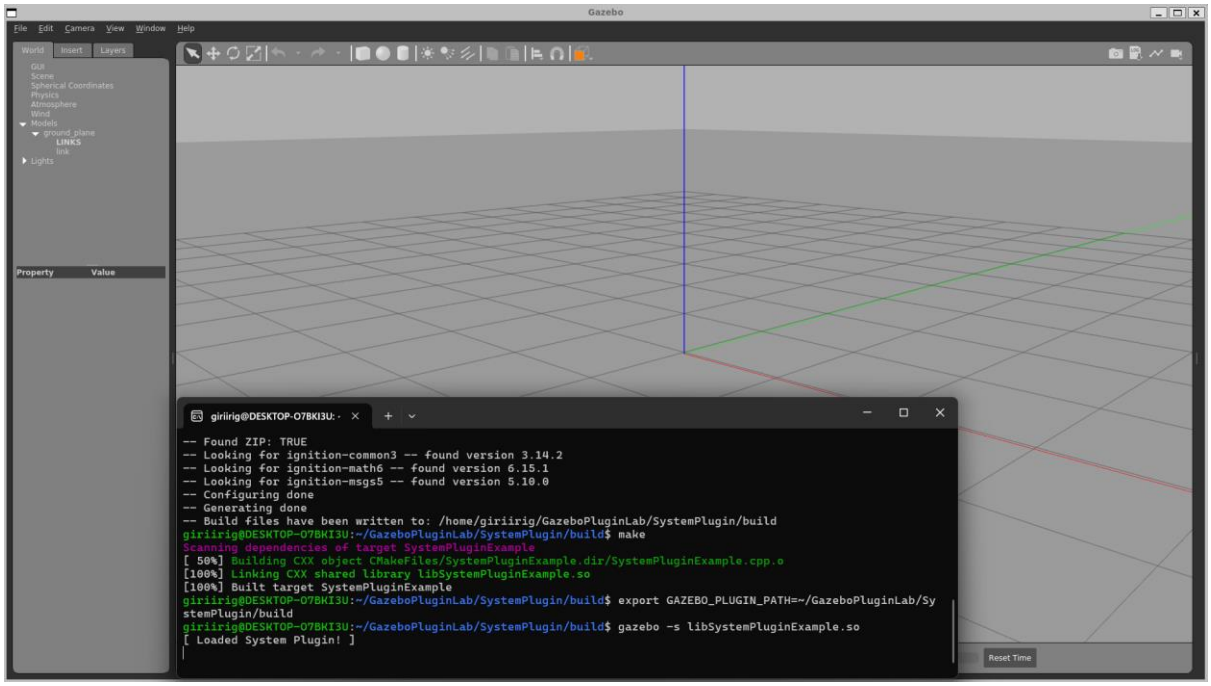
$ cmake ..

$ make

```
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/sdf$ cd ~/GazeboPluginLab/SystemPlugin
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SystemPlugin$ nano SystemPluginExample.cpp
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SystemPlugin$
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SystemPlugin$ nano CMakeLists.txt
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SystemPlugin$ cd build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SystemPlugin/build$ cmake ..
CMake Warning (dev) in CMakeLists.txt:
  No project() command is present.  The top-level CMakeLists.txt file must
  contain a literal, direct call to the project() command.  Add a line of
  code such as

    project(ProjectName)

  near the top of the file, but after cmake_minimum_required().

  CMake is pretending there is a "project(Project)" command on the first
  line.
This warning is for project developers.  Use -Wno-dev to suppress it.

-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Checking for module 'bullet>=2.82'
--   Found bullet, version 2.88
```

$ export GAZEBO_PLUGIN_PATH=~/GazeboPluginLab/SystemPlugin/build

$ gazebo -s libSystemPluginExample.so



5. Visual Plugin

Visual plugins affect the appearance of models allowing customization of appearance and visual effects and can add effects such as color changes, transparency, and other visual properties. Visual plugins are also used to add custom rendering to models.

$ cd ~/GazeboPluginLab/VisualPlugin

$ nano VisualPluginExample.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("[ Loaded Visual Plugin! ]\n");
      }
  };
  GZ_REGISTER_VISUAL_PLUGIN(VisualPluginExample)
}
```

$ nano CMakeLists.txt

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

$ cd build

$ cmake ..

$ make

```
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/SystemPlugin/build$ cd ~/GazeboPluginLab/VisualPlugin
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/VisualPlugin$ nano VisualPluginExample.cpp
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/VisualPlugin$ nano CMakeLists.txt
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/VisualPlugin$ cd build
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/VisualPlugin/build$ cmake ..
CMake Warning (dev) in CMakeLists.txt:
  No project() command is present.  The top-level CMakeLists.txt file must
  contain a literal, direct call to the project() command.  Add a line of
  code such as

    project(ProjectName)

  near the top of the file, but after cmake_minimum_required().

  CMake is pretending there is a "project(Project)" command on the first
  line.
This warning is for project developers.  Use -Wno-dev to suppress it.

-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Checking for module 'bullet>=2.82'
--   Found bullet, version 2.88
```

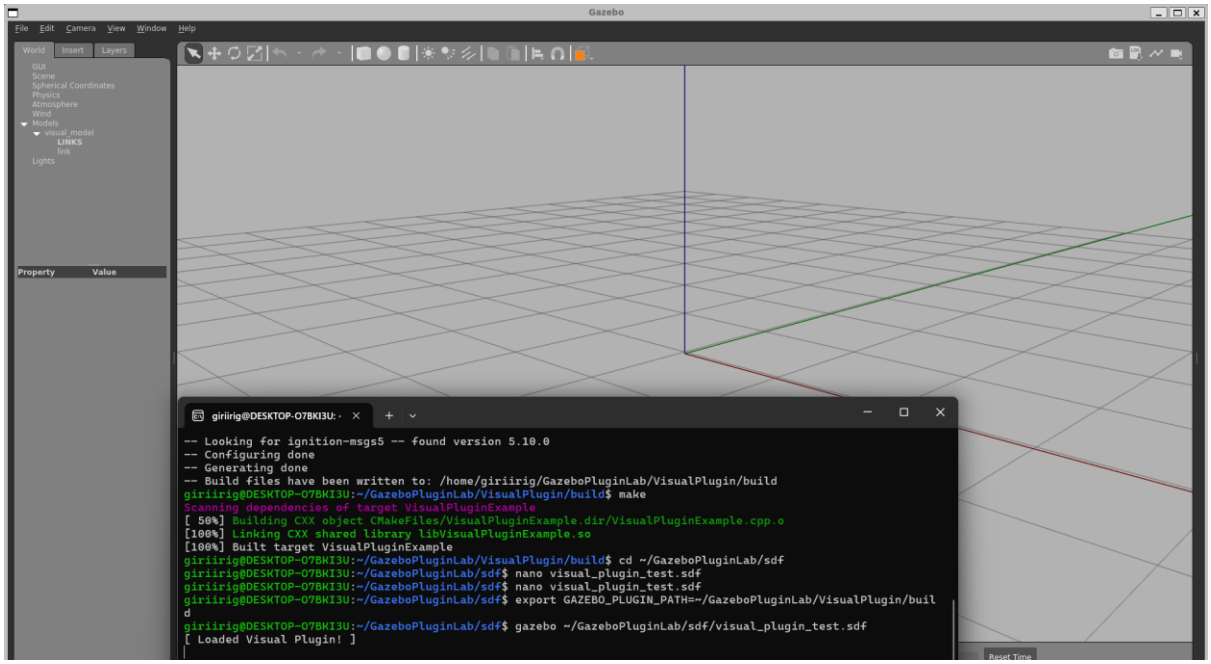
$ cd ~/GazeboPluginLab/sdf

$ nano visual_plugin_test.sdf

```
<?xml version="1.0" ?>
<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>
          <plugin name="visual_plugin" filename="libVisualPluginExample.so"/>
        </visual>
      </link>
    </model>
  </world>
</sdf>
```

$ export GAZEBO_PLUGIN_PATH=~/GazeboPluginLab/VisualPlugin/build

$ gazebo ~/GazeboPluginLab/sdf/visual_plugin_test.sdf



6. GUI Plugin

GUI plugins customize the Gazebo graphical user interface, provides additional display options and controls, allowing you to add custom panels, buttons, tools and other widgets that enhance the user experience. GUI plugins add custom UI elements or controls in Gazebo. These plugins are defined in the <gui> section of the SDF file.

$ cd ~/GazeboPluginLab/GUIPlugin

$ nano GUIExampleSpawnWidget.hh

```
#ifndef _GUI_EXAMPLE_SPAWN_WIDGET_HH_
#define _GUI_EXAMPLE_SPAWN_WIDGET_HH_

#include <gazebo/common/Plugin.hh>
#include <gazebo/gui/GuiPlugin.hh>
#include <gazebo/transport/transport.hh>

namespace gazebo
{
  class GUIExampleSpawnWidget : public GUIPlugin
  {
    Q_OBJECT
  public:
    GUIExampleSpawnWidget();
    virtual ~GUIExampleSpawnWidget();

  protected slots:
    void OnButton();

  private:
    unsigned int counter;
    transport::NodePtr node;
    transport::PublisherPtr factoryPub;
  };
}
#endif
```

$ nano GUIExampleSpawnWidget.cpp

```
#include "GUIExampleSpawnWidget.hh"
#include <gazebo/msgs/msgs.hh>
#include <QPushButton>
#include <QHBoxLayout>
#include <QVBoxLayout>

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));
  msgs::AddSphereLink(model, 1.0, 0.5);

  std::ostringstream newModelStr;
  newModelStr << "<sdf version='" << SDF_VERSION << "'>"
              << msgs::ModelToSDF(model)->ToString("")
              << "</sdf>";

  msgs::Factory msg;
  msg.set_sdf(newModelStr.str());
  this->factoryPub->Publish(msg);
}
```

$ nano CMakeLists.txt

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

$ cd build

$ cmake ..

$ make

```
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/sdf$ cd ~/GazeboPluginLab/GUIPlugin
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/GUIPlugin$ nano GUIExampleSpawnWidget.hh
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/GUIPlugin$ ^C
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/GUIPlugin$ nano GUIExampleSpawnWidget.cpp
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/GUIPlugin$ nano CMakeLists.txt
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/GUIPlugin$ cd build
.
make
giriirig@DESKTOP-O7BKI3U:~/GazeboPluginLab/GUIPlugin/build$ cmake ..
CMake Warning (dev) in CMakeLists.txt:
  No project() command is present.  The top-level CMakeLists.txt file must
  contain a literal, direct call to the project() command.  Add a line of
  code such as

    project(ProjectName)

  near the top of the file, but after cmake_minimum_required().

  CMake is pretending there is a "project(Project)" command on the first
  line.
This warning is for project developers.  Use -Wno-dev to suppress it.

-- The C compiler identification is GNU 9.4.0
-- The CXX compiler identification is GNU 9.4.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Checking for module 'bullet>=2.82'
--   Found bullet, version 2.88
```

$ cd ~/GazeboPluginLab/sdf

$ nano gui_plugin_test.world

```
<?xml version="1.0" ?>
<sdf version="1.5">
  <world name="default">
    <gui>
      <plugin name="sample" filename="libGUIExampleSpawnWidget.so"/>
    </gui>
    <include>
      <uri>model://sun</uri>
    </include>
    <include>
      <uri>model://ground_plane</uri>
    </include>
  </world>
</sdf>
```

$ export GAZEBO_PLUGIN_PATH=~/GazeboPluginLab/GUIPlugin/build

$ gazebo ~/GazeboPluginLab/sdf/gui_plugin_test.world