Simple steps to interact with a World in Gazebo through a World plugin

**1- Create a directory for scripts inside “myrobot” to store a hello.cpp file**

$ cd /home/workspace/myrobot

$ mkdir script

$ cd script

$ gedit hello.cpp

Inside hello.cpp, include this code:

TODO: Put in better formatting

#include <gazebo/gazebo.hh> *// Include the main Gazebo header file*

namespace gazebo

{

*// Define a new class WorldPluginMyRobot tat inherits from the WorldPlugin class*

**class** **WorldPluginMyRobot** : **public** WorldPlugin

{

**public**:

*// Constructor for WorldPluginMyRobot*

**WorldPluginMyRobot**() : **WorldPlugin**()

{

**printf**("Hello World!\n");

}

**public**:

*// Load function is called by Gazebo when loading the plugin*

**void** **Load**(physics::WorldPtr \_world, sdf::ElementPtr \_sdf)

{

}

};

*// Register this plugin with Gazebo,*

*// making it discoverable at runtime*

GZ\_REGISTER\_WORLD\_PLUGIN(WorldPluginMyRobot)

}

**2- Create a CMakeLists.txt file**

CMake is a cross-platform tool that helps generate build systems for compiling software. It helps automate the process by reading instructions from a CMakeLists.txt file. This file defines the build configurations, dependencies, and the instructions for building specific targets, like libraries or executables.

$ cd /home/workspace/myrobot

$ gedit CMakeLists.txt

Inside, CMakeLists.txt, include the following:

**cmake\_minimum\_required**(VERSION 2.8 FATAL\_ERROR)

# Find and link to Gazebo libraries

**find\_package**(gazebo REQUIRED)

**include\_directories**(${GAZEBO\_INCLUDE\_DIRS})

**link\_directories**(${GAZEBO\_LIBRARY\_DIRS})

**list**(APPEND CMAKE\_CXX\_FLAGS "${GAZEBO\_CXX\_FLAGS}")

# Add the plugin library and link it to Gazebo

**add\_library**(hello SHARED script/hello.cpp)

**target\_link\_libraries**(hello ${GAZEBO\_LIBRARIES})

**3- Create a build directory and compile the code**

$ cd /home/workspace/myrobot$

$ mkdir build

$ cd build

$ cmake ..

$ make # You might **get** errors **if** your system is not up to date!

$ **export** GAZEBO\_PLUGIN\_PATH=${GAZEBO\_PLUGIN\_PATH}:/home/workspace/myrobot/build

**New Command Explanation:**

* cmake ..: This command instructs CMake to read CMakeLists.txt in the parent directory and generate the necessary compilation files in the build directory.
* make: Compiles the hello.cpp plugin into a shared library file called libhello.so. This shared library can be dynamically loaded at runtime by Gazebo to run your plugin. This library will be located in the **build** directory.
* export GAZEBO\_PLUGIN\_PATH=...: Adds the plugin path to the Gazebo environment so Gazebo can load the plugin.

**4- Open your world file and attach the plugin to it**

$ cd /home/workspace/myrobot/world/

$ gedit myworld

Copy this code:

<plugin name="hello" filename="libhello.so"/>

and paste it under:

<world name="default">

**5- Launch the world file in Gazebo to load both the world and the plugin**

$ cd /home/workspace/myrobot/world/

$ gazebo myworld

6- Visualize the output

A Hello World! message is printed in the terminal. This message interacts with the Gazebo World that includes your two-wheeled robot.

Troubleshooting

In case your plugins failed to load, you'll have to check and troubleshoot your error. The best way to troubleshoot errors with Gazebo is to launch it with the verbose as such:

$ gazebo myworld --verbose

When using --verbose, Gazebo provides detailed error messages. Reading error messages provides hints where to look to fix the issue. Check theses messages to locate issues, such as missing libraries or incorrect file paths, and make sure your GAZEBO\_PLUGIN\_PATH includes the directory of the compiled plugin.

GitHub Repo

I hope you followed along with these instructions, built your first model in Gazebo, included in an empty world, and interacted with the world through a World Plugin. You can always clone this lab from GitHub by clicking [**here(opens in a new tab)**](https://github.com/udacity/RoboND-myrobot). Follow the instructions in the Readme file to launch the world.

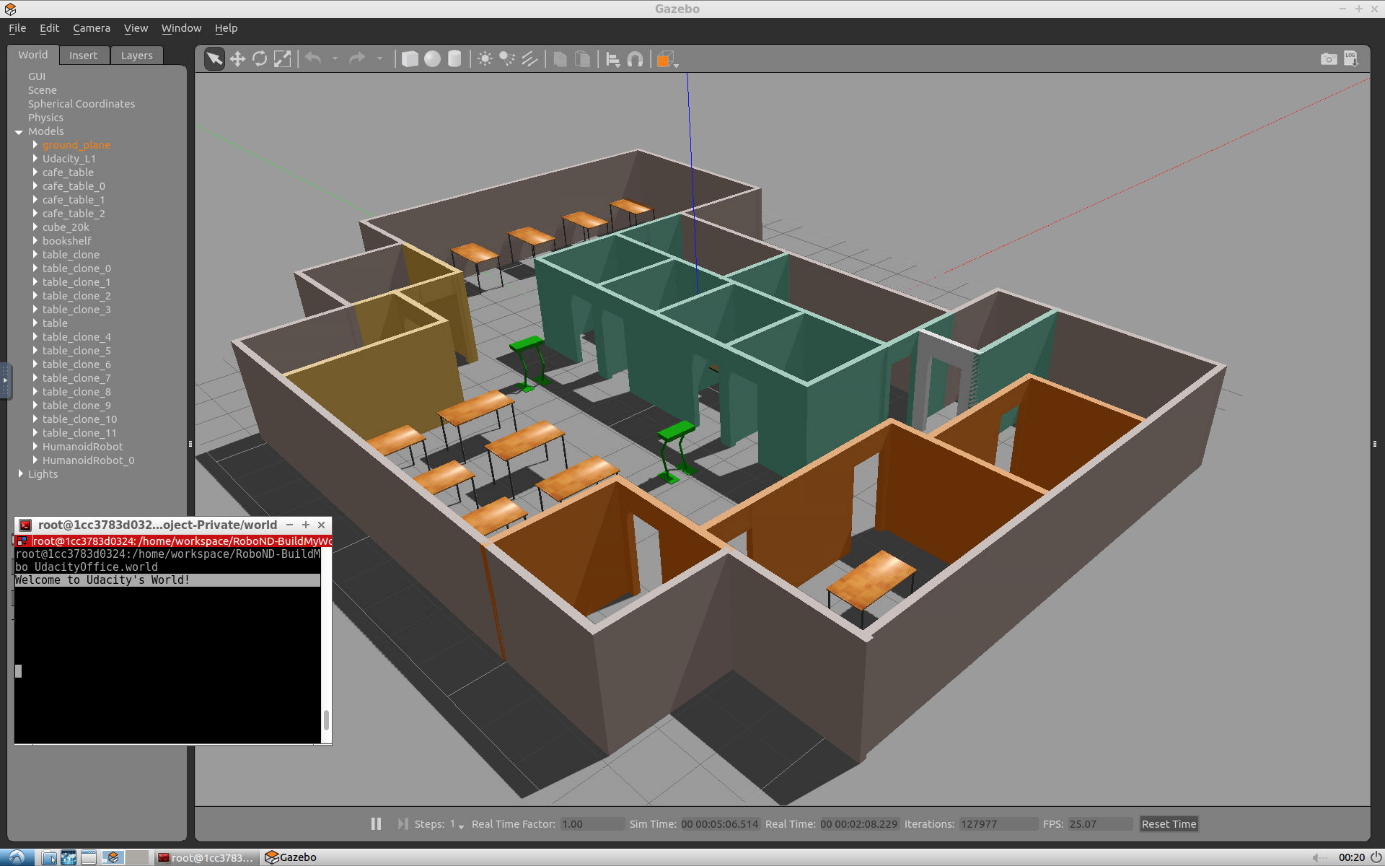
Sample Simulation World

LessonDownloadsCloud Resources

To help you get started, I’ve included a sample simulation. Review the files structure below. Reviewing the sample, its aspects, and the directory structure will help you build your own world in Gazebo.

Please note that you have only visual access to the sample simulation as it's not provided to you in the Udacity Workspace or anywhere else.

Output Image



Project Aspects

Inside the Gazebo world you can identify:

* **Udacity Office**: A building model designed on the Building Editor tool of Gazebo. The structure contains features, and colors.
* **Green humanoid robot**: Two instances of a model designed on the Model Editor tool of Gazebo.
* **Tables**: A model imported from the Gazebo online library.
* **Terminal**: A welcome message generated from a world plugin and printed to the terminal.

Directory Structure

The sample simulation world folder has the following directory structure:

.Project1 *# Build My World Project*

├── model *# Model files*

│ ├── Building

│ │ ├── model.config

│ │ ├── model.sdf

│ ├── HumanoidRobot

│ │ ├── model.config

│ │ ├── model.sdf

├── **script** *# Gazebo World plugin C++ script*

│ ├── welcome\_message.cpp

├── world *# Gazebo main World containing models*

│ ├── UdacityOffice.world

├── CMakeLists.txt *# Link libraries*

└──

Your task

Note that your project should have the same aspects and should follow the same directory structure as the sample project! Pick a place where you want to deploy your robot, for example: your apartment, your office, or your favorite restaurant. In addition, create any model of your choice and import a model from the Gazebo library.

Project: Build My World

**Past Due on March 18, 2025**

Summary of Tasks

Let’s summarize what you should do in this project to create a simulation world for all your projects in this Robotics Software Engineer Nanodegree Program.

1. Build a single floor building using the **Building Editor** tool in Gazebo. Apply at least one feature, one color, and optionally one texture to your structure. Make sure there's enough space between the walls for a robot to navigate.
2. Model any robot of your choice using the **Model Editor** tool in Gazebo. Your robot model links should be connected with joints.
3. Import your building and **two instances** of your robot model inside an empty **Gazebo World**.
4. Import at least one model from the **Gazebo online library** and implement it in your existing Gazebo world.
5. Write a C++ **World Plugin** to interact with your world. Your code should display “Welcome to ’s World!” message as soon as you launch the Gazebo world file.

These tasks are just the basic requirements for you to pass the project! Feel free to have fun designing and importing multiple models.

Evaluation

Once you finish designing your world in Gazebo, check the [**Project Rubric(opens in a new tab)**](https://review.udacity.com/#!/rubrics/2346/view) to see if your world meets the specifications. If so, then you are ready to submit!

Submission Folder

Here are the files that you need to include in your project submission folder:

* **model** folder:
* Any robot or model designed in the Model Editor tool of Gazebo
* A single floor building designed in the Building Editor tool of Gazebo
* **world** folder:
* Gazebo world file that includes the models
* **script** folder:
* Gazebo world plugin C++ script
* **CMakeLists.txt** file to link the C++ code to libraries

**Note:** Remove the **build** directory from your submission folder, as it contains environment-specific files.

Ready to submit your project?

Click on the "Submit Project" button and follow the instructions to submit!