**Course 2 - Gazebo Basics**

## Gazebo Basics

### Introduction

Video 1.1

**Gazebo Basics Lesson Roadmap**

* What is Gazebo?
* Gazebo in the Workspace
* Gazebo Components
* Understanding the GUI
* Simulating your First Robot
* Writing a Plugin
* Exploring the Building Editor

A logo of a cube

AI-generated content may be incorrect.

### What is Gazebo?

A screenshot of a computer

AI-generated content may be incorrect.

Gazebo is a physics-based, high fidelity 3D simulator for robotics. Gazebo provides the ability to accurately simulate one or more robots in complex indoor and outdoor environments filled with static and dynamic objects, realistic lighting, and programmable interactions.

Gazebo facilitates robotic design, rapid prototyping, testing, and simulation of real-life scenarios. While Gazebo is platform agnostic and runs on Windows, Mac, and Linux, it is mostly used in conjunction with the Robotics Operating System (ROS) running on Linux systems. You will learn how to use ROS in the upcoming lessons. Simply put, Gazebo is an essential tool for every roboticist.

1. Gazebo Features

Gazebo has eight features that you can take advantage of:

1. **Dynamics Simulation:** Model a robot's dynamics with a high-performance physics engine.
2. **Advanced 3D Graphics:** Render your environment with high-fidelity graphics, including lighting, shadows, and textures.
3. **Sensors:** Add sensors to your robot, generate data, and simulate noise.
4. **Plugins:** Write a plugin to interact with your world, robot, or sensor.
5. **Model Database:** Download a robot or environment from Gazebo library or build your own through their engine.
6. **Socket-Based Communication:** Interact with Gazebo running on a remote server through [**socket-based(opens in a new tab)**](https://en.wikipedia.org/wiki/Network_socket) communication.
7. **Cloud Simulation:** Run Gazebo on a server and interact with it through a browser.
8. **Command Line Tools:** Control your simulated environment through the command line tools.

This lesson will focus on Gazebo’s four main features including **Dynamics Simulation**, **Advanced 3D Graphics**, **Plugins**, and **Model Database.** For more information on Gazebo and its history visit the [**Gazebo website(opens in a new tab)**](http://gazebosim.org/).

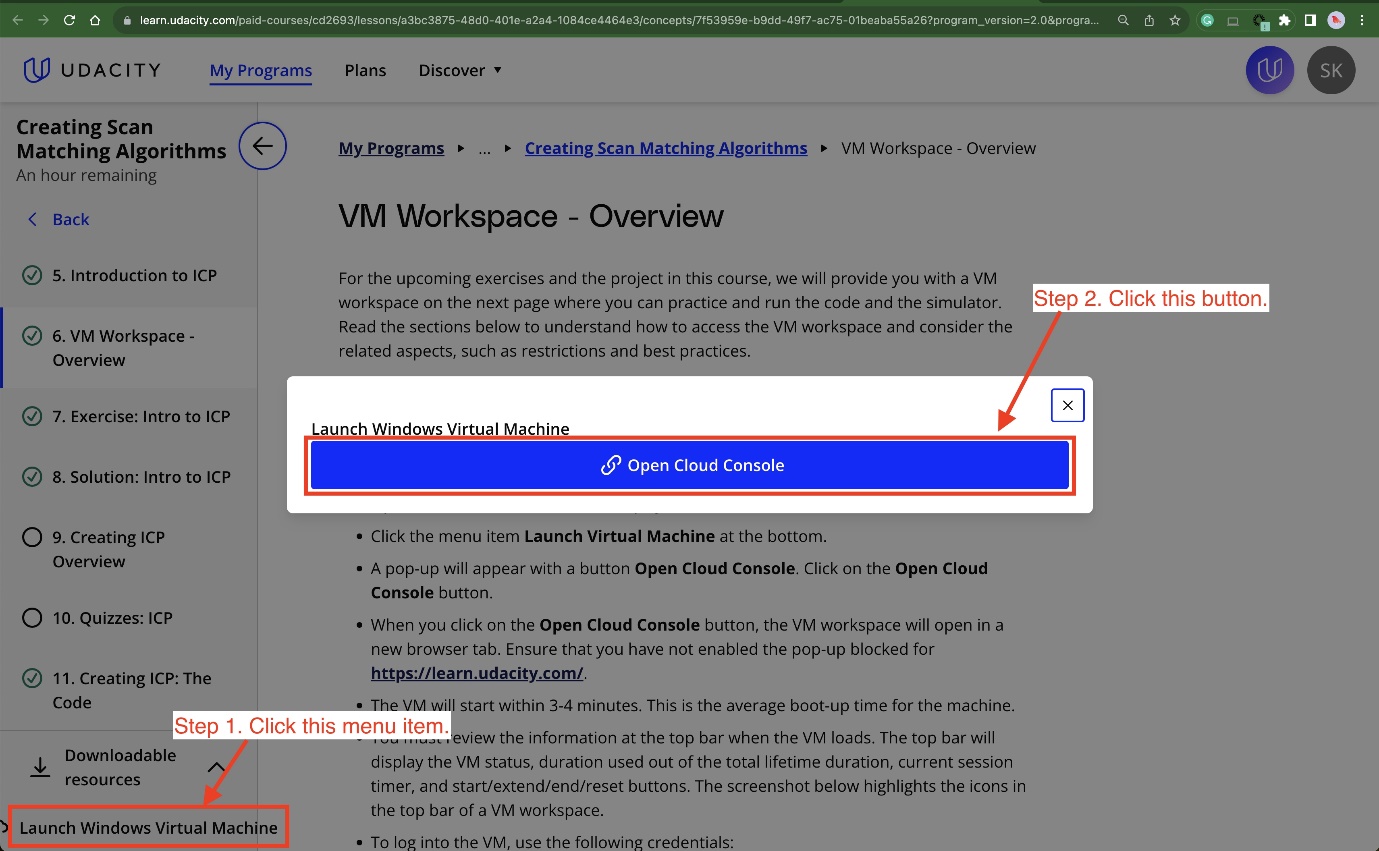
### Ubuntu VM Workspace – Overview

For the upcoming exercises/project in this course, we will provide you with a VM workspace. Read the sections below to understand how to access the VM workspace and consider the related aspects, such as restrictions and best practices.

**VM Workspace - Access**

Follow the instructions below to access your VM workspace.

* Open the left menu on the current page and scroll to the bottom of the menu.
* Click the menu item **Launch Virtual Machine** at the bottom.
* A pop-up will appear with a button **Open Cloud Console**. Click on the **Open Cloud Console** button.



A screenshot highlighting the steps to open a VM workspace.

* When you click on the **Open Cloud Console** button, the VM workspace will open in a new browser tab. Ensure that you have not enabled the pop-up blocked for [**https://learn.udacity.com/(opens in a new tab)**](https://learn.udacity.com/).
* The VM will start within 3-4 minutes. This is the average boot-up time for the machine.
* You must review the information at the top bar when the VM loads. The top bar will display the VM status, duration used out of the total lifetime duration, current session timer, and start/extend/end/reset buttons. The screenshot below highlights the icons in the top bar of a VM workspace.
* To log into the VM, use the following credentials:

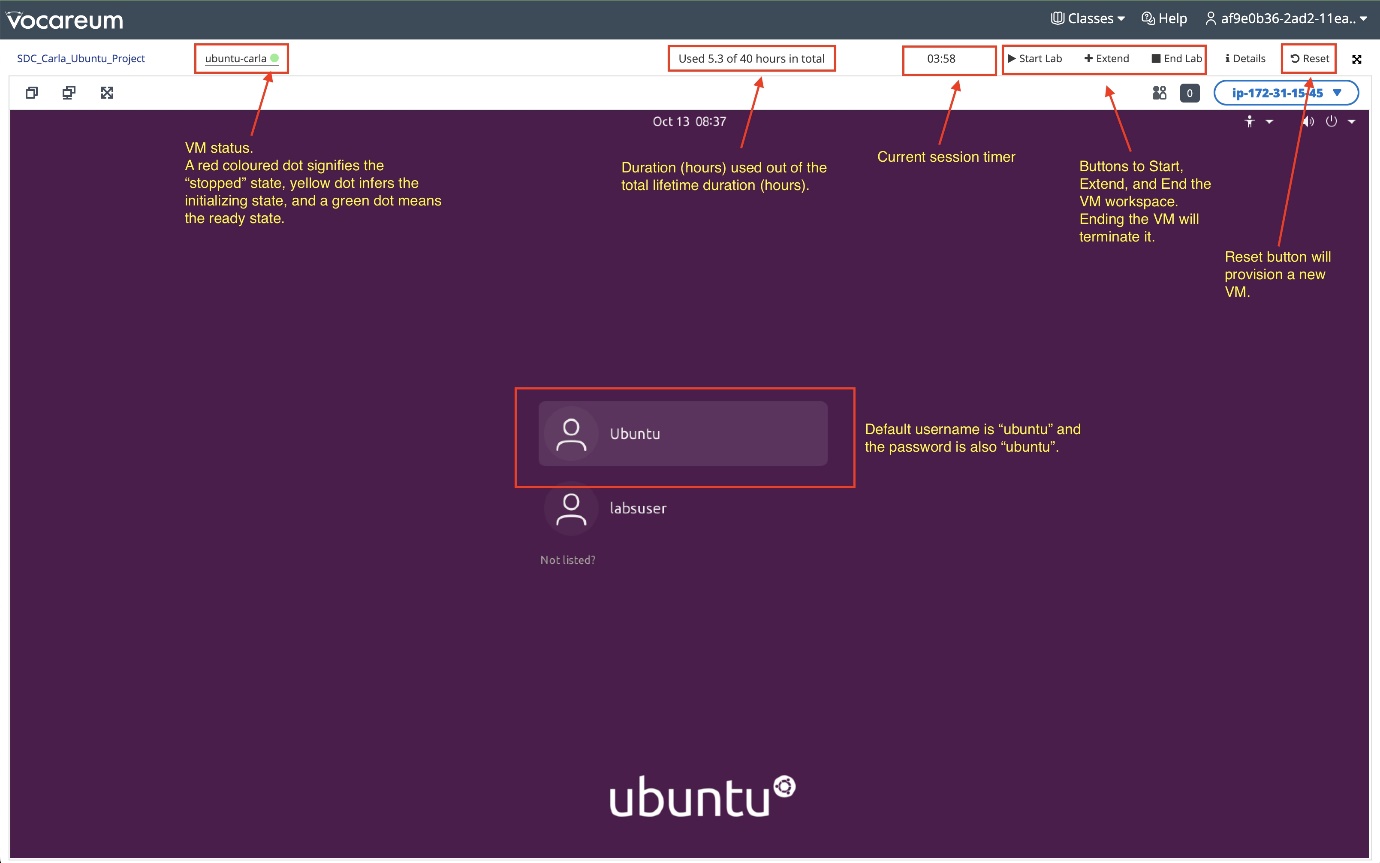
Username: ubuntu

Password: ubuntu

Also, an additional pair of credentials are:

Username: labsuser

Password: vocareum



A screenshot of a VM workspace

**VM Workspace - Overview**

The VM workspace is an Ubuntu 20.04 machine with NVIDIA Tesla T4 GPU support. It has all the necessary tools installed that you will need to run the exercises and the project in this course. For example, it has the following major tools and corresponding configurations:

* [**CARLA simulator 0.9.9.4(opens in a new tab)**](https://github.com/carla-simulator/carla/releases/tag/0.9.9)
* [**NICE DCV Server(opens in a new tab)**](https://docs.aws.amazon.com/dcv/latest/adminguide/setting-up-installing-linux-prereq.html). This includes the Nvidia drivers along with CUDA libraries for the underlying Tesla T4 GPU
* C++
* Git
* [**OpenCV(opens in a new tab)**](https://docs.opencv.org/4.x/d7/d9f/tutorial_linux_install.html)
* [**CMake(opens in a new tab)**](https://askubuntu.com/questions/161104/how-do-i-install-make) and Make
* [**VSCode(opens in a new tab)**](https://code.visualstudio.com/download)
* [**Eigen Library for C++(opens in a new tab)**](https://eigen.tuxfamily.org/index.php?title=Main_Page)
* [**Point Cloud Library(opens in a new tab)**](https://pointclouds.org/downloads/)
* Python3 and Pip
* ROS

**However, you can install more tools as necessary for the classroom exercises or the project.**

An important aspect is that the VM workspace has several restrictions, as explained below. You must read the restrictions and best practices below before you plan your work in the VM workspace.

**VM Workspace - Restrictions and Best Practices**

1. The VM workspace is a **Sandbox environment**. When you access the VM workspace, a session initiates for a limited duration.
   * After the session expires, the VM workspace will terminate.
   * You must reload the classroom page to access the VM workspace and thus start a new session.
   * Udacity will not preserve your work or any custom installations for your next session. You will get a fresh new VM workspace in each new attempt.
   * **Best practice**: You must save your edits to the starter/solution code and the Readme file in a Github repo so you can pull it again in your next session.
2. The VM workspace has a limited session duration and a fixed total lifetime duration. For example, each session can be four hours, and the total lifetime duration could be 40 hours.
   * You will find the session timer and the total lifetime duration at the top bar of your VM workspace.
   * When the VM workspace session is about to end, you will get a prompt to extend the session duration by another hour. You can make such extensions for up to four more hours. This means you will get the initial fixed session duration and the extended session duration in one session.

For example, if the initial fixed session duration is four hours, and you have made four extensions, you will get eight hours in a single session.

* + **Best practice**: You must save your work intermittently to avoid potential data loss if the VM workspace session ends abruptly.

1. **Best practice**: As Udacity pays for every minute of runtime, you must click the **End lab** button at the top to terminate the VM workspace when not in use. This step will ensure you have sufficient time remaining for the project.
2. Misusing the VM workspace can lead to **revoking access** or deactivating your Udacity account. Please follow the instructions described in the exercise/project instructions. Udacity monitors VM workspace usage to ensure the ethical use of resources for educational purposes only.

**VM Workspace - Getting Help**

If the VM workspace does not load for you, file a support ticket with Udacity. You will find the link for filing a support ticket at the bottom-right of your classroom page.

### Gazebo in the Workspace

**ToDo - Instructions**

Follow these simple steps to boot up Gazebo in your Workspace:

* Open Udacity's VM workspace (Ubuntu 20.04 LTS) or the local VM image (Ubuntu 18.04 LTS) running on your VMWare/VirtualBox to practice the current exercise. If you haven't already, review the overview and restrictions outlined in the **Ubuntu VM Workspace - Overview** page.
* Once you log into the VM, open a Terminal window.
* Update and upgrade the Workspace image to get the latest features of Gazebo. To do so, open a terminal, and write the following statement:

$ sudo apt-get update && sudo apt-get upgrade -y

**Note**: Remember to update and upgrade your image after each reboot since these updates(or any package that you install) are not permanent. Ignore any error you get while upgrading.

* Now launch Gazebo from the terminal by typing:

$ gazebo

Be advised that occasionally Gazebo gets stuck in a loading loop. If Gazebo does not fully load after about one minute, try closing it by pressing Ctrl+C while the Gazebo terminal is active. Then, try running the command again to restart Gazebo.

### Gazebo Components

There are six components involved in running an instance of a Gazebo simulation:

1. Gazebo Server
2. Gazebo Client
3. World Files
4. Model Files
5. Environment Variables
6. Plugins

**1- Gazebo Server**

The first main component involved in running an instance of a Gazebo simulation is the Gazebo Server or also known by **gzserver**.

**gzserver** performs most of the heavy-lifting for Gazebo. It is responsible for parsing the description files related to the scene we are trying to simulate, as well as the objects within. It then simulates the complete scene using a physics and sensor engine.

While the server can be launched independently by using the following command in a terminal:

$ gzserver

It does not have any GUI component. Running **gzserver** in a so-called headless mode can come in handy in certain situations, but we will talk more about that in future lessons.

**2- Gazebo Client**

The second main component involved in running an instance of a Gazebo simulation is the Gazebo Client or also known by **gzclient**.

**gzclient** on the other hand provides the very essential Graphical Client that connects to the **gzserver** and renders the simulation scene along with useful interactive tools. While you can technically run **gzclient** by itself using the following command:

$ gzclient

it does nothing at all (except consume your compute resources), since it does not have a **gzserver** to connect to and receive instructions from.

**Combining Gazebo Server and Gazebo Client**

It is a common practice to run **gzserver** first, followed by **gzclient**, allowing some time to initialize the simulation scene, objects within, and associated parameters before rendering it. To make our lives easier, there is a single intuitive command that necessarily launches both the components sequentially:

$ gazebo

**3- World Files**

A **world** file in Gazebo contains all the elements in the simulated environment. These elements are your robot model, its environment, lighting, sensors, and other objects. You have the ability to save your simulation to a world file that usually has a .world extension.

Gazebo can also read the content of a world file from your disk to generate the simulation. To launch the simulation from a **world** file, type:

$ gazebo <yourworld>.world

The **world** file is formatted using the **Simulation Description Format** or [**SDF(opens in a new tab)**](http://sdformat.org/spec?ver=1.6&elem=world) for short. Here’s the basic format of an **SDF** **world** file:

<?xml version="1.0" ?>

<sdf version="1.5">

  <world name="default">

    <physics type="ode">

      ...

    </physics>

    <scene>

      ...

    </scene>

    <model name="box">

      ...

    </model>

    <model name="sphere">

      ...

    </model>

    <light name="spotlight">

      ...

    </light>

  </world>

</sdf>

**4- Model Files**

For simplification, you must create a separate **SDF** file of your robot with exactly the same format as your **world** file. This **model** file should only represent a single model (ex: a robot) and can be imported by your **world** file. The reason why you need to keep your model in a separate file is to use it in other projects. To include a **model** file of a robot or any other model inside your **world** file, you can add the following code to the **world’s** **SDF** file:

<include>

  <uri>model://model\_file\_name</uri>

</include>

**5- Environment Variables**

There are many environment variables that Gazebo uses, primarily to locate files (world, model, …) and set up communications between **gzserver** and **gzclient**. While working on a robotic project, you’ll leave these variables as default. Here’s an example of a variable that Gazebo uses:

GAZEBO\_MODEL\_PATH: List of directories where Gazebo looks to populate a model file.

**6- Plugins**

To interact with a world, model, or sensor in Gazebo, you can write plugins. These plugins can be either loaded from the command line or added to your **SDF** world file. You’ll learn about World Plugins later in the lesson.

### Understanding the GUI

Video 1.6

The Gazebo GUI is divided into four major sections:

1. Scene
2. Side Panel
3. Toolbars
4. Menu

**Scene**

The scene is where you will be spending most of your time, whether creating a simulation or running one. While you can use a trackpad to navigate inside the scene, a mouse is highly recommended.

You can pan the scene by pressing the left mouse button and dragging. If you hold down SHIFT in addition, you can now rotate the view. You can zoom in and out by using the mouse scroll or pressing and dragging the RMB.

A computer mouse with text below

AI-generated content may be incorrect.

**Side Panel**

The side panel on the left consists of three tabs:

1. World
2. Insert
3. Layers

*World*

This tab displays the lights and models currently in the scene. By clicking on individual model, you can view or edit its basic parameters like position and orientation. In addition, you can also change the physics of the scene like gravity and magnetic field via the Physics option. The GUI option provides access to the default camera view angle and pose.

*Insert*

This is where you will find objects (models) to add to the simulation scene. Left click to expand or collapse a list of models. To place an object in the scene, simply left-click the object of interest under the Insert tab; this will bind the object to your mouse cursor, and now you can place it anywhere in the scene by left-clicking at that location.

Layers

This is an optional feature, so this tab will be empty in most cases. To learn more about Layers, click [**here(opens in a new tab)**](http://gazebosim.org/tutorials?tut=visual_layers&cat=build_robot).

**Top Toolbar**

Next, we have a toolbar at the top. It provides quick access to some cursor types, geometric shapes, and views.

Select mode

Select mode is the most commonly used cursor mode. It allows you to navigate the scene.

Translate mode

One way to change an object's position is to select the object in the world tab on the side panel and then change its pose via properties. This is cumbersome and also unnatural, the translate mode cursor allows you to change the position of any model in the scene. Simply select the cursor mode and then use the proper axis to drag the object to its desired location.

Rotate mode

Similar to translate mode, this cursor mode allows you to change the orientation of any given model.

Scale mode

Scale mode allows you to change the scale, and hence, overall size of any model.

Undo/Redo

Since we humans are best at making mistakes, the undo tool helps us revert our mistakes. On the other hand if you undid something that you did not intend to, the redo tool can come to the rescue.

Simple shapes

You can insert basic 3D models like cubes, spheres, or cylinders into the scene.

Lights

Add different light sources like a spotlight, point light, or directional light to the scene.

Copy/Paste

These tools let you copy/paste models in the scene. On the other hand, you can simply press Ctrl+C to copy and Ctrl+V to paste any model.

Align

This tool allows you to align one model with another along one of the three principal axes.

Change view

The change view tool lets you view the scene from different perspectives like top view, side view, front view, bottom view.

Bottom Toolbar

The Bottom Toolbar has a neat play and pause button. This button allows you to pause the simulation and conveniently move objects around. This toolbar also displays data about the simulation, such as the simulation time, the real time, and the relationship between the two. There is also an frames-per-second counter that can be found to gauge your system's performance for any given scene.

Menu

The Menu: Some of the menu options are duplicated in the Toolbars or as right-click context menu options in the Scene. If you click on Edit, you can switch to the Building Editor to design building or the Model Editor to build models. We will be working in both of these features in the upcoming concepts.

A screenshot of a computer

AI-generated content may be incorrect.

### Simulating your First Robot

**Create Directories**

Video 1.7

A screenshot of a computer

AI-generated content may be incorrect.

**Construct the Robot's Chassis**

Video 1.7.1

A screenshot of a computer

AI-generated content may be incorrect.

**Attach Wheels to the Robot's Chassis**

**Video 1.7.2**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

<?xml version="1.0" ?>

<sdf version="1.5">

    <world name="default">

      <physics type="ode">

        ...

      </physics>

      <scene>

        ...

      </scene>

      <model name="box">

        ...

      </model>

      <model name="sphere">

        ...

      </model>

      <light name="spotlight">

        ...

      </light>

    </world>

</sdf>

Next, you will learn how to write a Plugin to interact with your World in Gazebo.

### Writing a Plugin

Video 1.8

A screenshot of a computer

AI-generated content may be incorrect.

#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)

}

A screenshot of a computer program

AI-generated content may be incorrect.

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

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer error

AI-generated content may be incorrect.

**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.

### Exploring the Building Editor

Video 1.9

A blue line drawing of a letter u

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

### Outro

Video 1.10

**Tutorials**

Click [**here(opens in a new tab)**](http://gazebosim.org/tutorials) to access Gazebo official tutorials.

A white background with black text

AI-generated content may be incorrect.

## Project: Build My World

### Introduction

Video 2.1

### Sample Simulation World

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

A computer screen shot of a room

AI-generated content may be incorrect.

**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.

### Gazebo in the Workspace

A screenshot of a computer

AI-generated content may be incorrect.

### Project Cheat Sheet

Project Cheat Sheet

A close-up of a paper

AI-generated content may be incorrect.

**Supporting Materials**

* [**Robotics P1 Cheat Sheet**](https://video.udacity-data.com/topher/2019/October/5da11bf1_robotics-p1-lessons-cheat-sheet/robotics-p1-lessons-cheat-sheet.pdf)

### Project Rubric

**Project: Build My World**

**Basic Requirements**

| **Criteria** | **Submission Requirements** |
| --- | --- |
| Does the project include a **world** directory containing the Gazebo world file, a **model** directory containing a buidling and a robot model files, a **script** directory containing the C++ plugin code, and a **CMakeLists.txt** file? | The student submitted all required files specified in the criteria. |

**Building**

| **Criteria** | **Submission Requirements** |
| --- | --- |
| Does the project include a house with walls built with Gazebo's **Building Editor**? | The student designed a building and stored it in the model directory.  The building basic requirements:   * Building is different from the one shown in the sample simulation world. * Single floor. * Enough space for robots to navigate. * At least one feature. * At least one color. |

**Modeling**

| **Criteria** | **Submission Requirements** |
| --- | --- |
| Does the project include an robot built using Gazebo's **Model Editor**? | The student designed a robot and stored it in the model directory.  Model basic requirements:   * Robot is different than the one shown in the sample simulation world. * Robot links are connected through joints. |

**Gazebo World**

| **Criteria** | **Submission Requirements** |
| --- | --- |
| Doe the project contain a **Gazebo world** with multiple models? | The student created a Gazebo world and stored it in the world directory.  Gazebo World basic requirements:   * World is different than the one shown in the sample simulation world. * Contains the building model (e.g., house, restaurant, etc.). * Contains two instances of the robot model. * Contains one model from the Gazebo online library. |

**World Plugin**

| **Criteria** | **Submission Requirements** |
| --- | --- |
| Does the project contain a C++ **world plugin**? | The student created a C++ plugin and stored it in the script directory. Also, the student created a CMakeLists.txt file and stored in the main project directory.  World plugin basic requirements:   * The plugin C++ code should print “Welcome to ’s World!” message. * Do not include the build directory in your submission. |

**Suggestions to Make Your Project Stand Out**

Stand out submissions should have multiple models that interact through model plugins inside the Gazebo world.

### Submit Project

**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.

A screenshot of a checklist

AI-generated content may be incorrect.