Department of Electrical and Electronic Engineering

ELEN90066 Embedded System Design

Workshop Three (W03) - (Remote)

Modelling an environment

Welcome to Workshop 3 for Embedded System Design. In this workshop, you will be performing several exercises to gain familiarity with modelling an environment in Gazebo and creating your own Gazebo worlds for simulation. It is essential you understand the steps in this workshop as you will be using Gazebo for your robot simulation during the semester.

Note: You might be wondering what happened to ROS and Simulink from Workshops 1 and 2? Don't worry, you'll be seeing them again shortly. The goal of this workshop is to ensure you are familiar with Gazebo before integrating them for a complete robot control and simulation suite.

There are two options for using the software in this workshop:

1. **Locally on your own computer**

   You will need to have the following software installed before the workshop class:

   - VirtualBox for your host computer operating system with:
     - Embedded System Design Virtual Machine (ESD VM)

   IMPORTANT: Note that Gazebo can be computationally intensive. It is recommended that you adjust the following settings of the ESD VM (when it is switched off) before running Gazebo to suit your host computer's hardware :

   - System→Processor→Processor(s)
   - System→Motherboard→Base Memory

   It is recommended that you choose values that are as large as possible without being in the 'red areas' of the slider as this may cause your computer to become unresponsive.

2. **Remotely via MyUniApps**

   You will need to log in to MyUniApps and select the "MSE Desktop EDS12". Note that this option is ONLY available to students enrolled in Embedded System Design for this semester.

   Once logged in, open the File Explorer from the Start menu and navigate to `c:\Gazebo`. Double click on the "Launch Ubuntu 16.04.6 LTS ESD" link to launch the ESD VM.

Warning: While you have access to your profile files and folders on the MSE Desktop EDS12, any files created within the ESD VM will not be saved. It is recommended that you make a copy of any files that you create as part of this workshop. If you are using the Citrix receiver application you should be able to also access the local file system on your computer.

Note: Please sign out (Ctrl-Alt-Delete) after you have finished the workshop.

Pre-workshop background

Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. While similar to game engines, Gazebo offers physics simulation at a much higher degree of fidelity, a suite of sensors, and interfaces for both users and programs.

Typical uses of Gazebo include:

- testing robotics algorithms,
- designing robots,
- performing regression testing with realistic scenarios

A few key features of Gazebo include:

- multiple physics engines,
- a rich library of robot models and environments,
- a wide variety of sensors,
- convenient programmatic and graphical interfaces.

The `gazebo` command actually runs two different executables when involked. The first is called `gzserver`, and the second `gzclient`.

- The `gzserver` executable runs the physics update-loop and sensor data generation. This is the core of Gazebo, and can be used independently of a graphical interface. You may see the phrase "run headless" thrown about. This phrase equates to running only the `gzserver`. An example use case would involve running `gzserver` on a cloud computer where a user interface is not needed.
- The `gzclient` executable runs a QT based user interface. This application provides a nice visualisation of simulation, and convenient controls over various simulation properties.

You can run each of these executables separately by running the server `gzserver` in one terminal and the client `gzclient` in another. You can restart the `gzclient` application as often as you want, and even run multiple interfaces.

Pre-workshop questions

1. Briefly describe how you could use a simulator such as Gazebo in the context of developing an embedded system.

2. Why might a user decide to run Gazebo in "headless" mode?

3. Which version of Gazebo is included in the ESD VM?
   Hint: Open a terminal window in the ESD VM and run the command `gazebo` to find out.

Workshop Exercises

In this workshop you will do the following

- Add simple shapes and models from the Gazebo database to create a Gazebo world. (TASK 1 - 5 marks)
- Use the physics engine to simulate environment dynamics and interactions between models. (TASK 2 - 10 marks)
- Create a ROS package to integrate ROS with Gazebo. (TASK 3 -10 marks)

There are 25 marks available for successful completion of all tasks in this workshop, worth 1% of your final mark in the subject.

Gazebo basics

Startup Gazebo in the ESD VM by opening a new terminal window and running

```
$ gazebo
```

You should be greeted with a default Gazebo screen as shown in Figure 1.

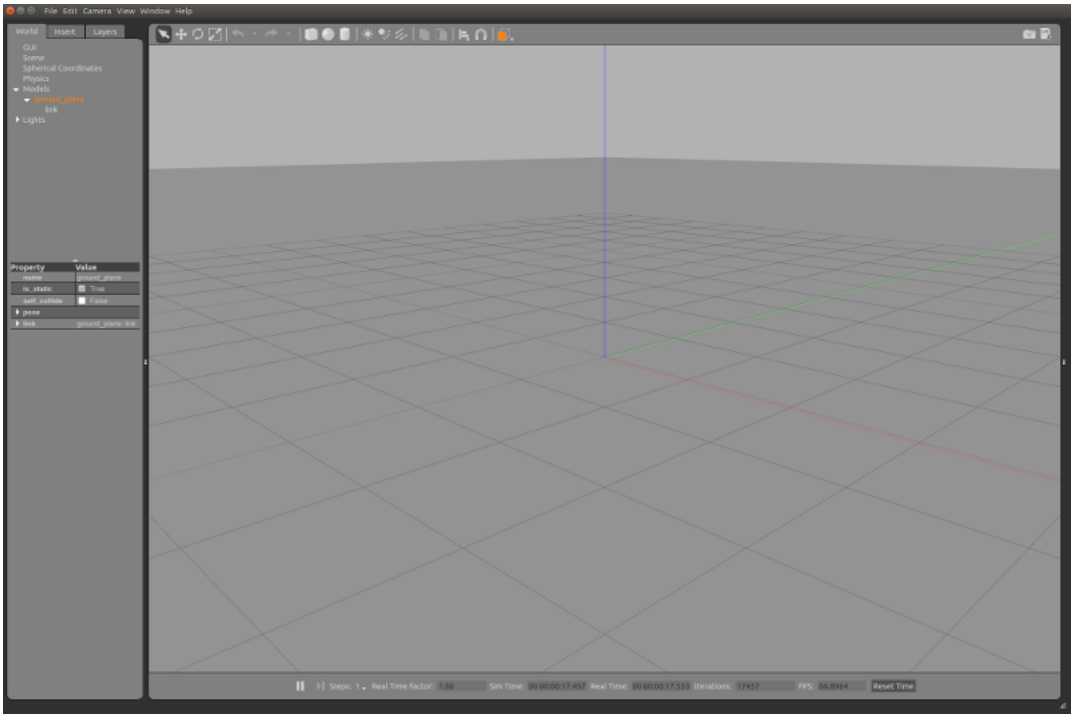


Figure 1: Default Gazebo screen

For a handy reference for the Gazebo User Interface, see the following link http://gazebosim.org/tutorials?cat=guided_b&tut=guided_b2

Adding Simple Shapes

Boxes, spheres, and cylinders may be added to the world by clicking on the appropriate icon above the render window, shown in Figure 2.

Figure 2: Simple shapes

Each shape is of unit size:

- Box: $1 \times 1 \times 1$ metre
- Sphere: 1 metre diameter
- Cylinder: 1 metre diameter, 1 metre length

To insert, select the desired icon, and then move your mouse onto the render window. You will see a shape that moves with your mouse. Left click when you are happy with the position of the shape.

Adding a Model from the Model Database

Gazebo's model database is a repository of all types of models including robots, tables, and buildings.

Select the Insert tab in the upper left hand corner to access the model database.

The list of models are divided into sections according to their current location. Each section is labelled with a path or URI. Selecting an object located on a remote server will cause the model to be downloaded and stored in /.gazebo/models . Be patient as it may take some time to download.

Positioning Models

The *pose* of each model may be altered through the translate and rotate tools shown in Figure 3.

Figure 3: Translate, rotate and scale tools

Translation

The translate tool allows you to move the object along the x, y, and z axes. Select this tool (or press t) and click on the object you want to move. A three axes visual marker will appear over the object, which allows you to move the object in x, y, and z directions.

- You can also just click on the object itself and drag it to move on the x-y plane. You may control which axis the object moves along by pressing and holding the x, y, or z key while dragging the object.
- You can hold the Ctrl key to snap the movement to a 1 metre grid.
- If the object is not aligned with the world (for example after you use the rotate tool explained next), you can hold the Shift key so the visual markers show up aligned with the world, and you can translate in world coordinates.

Rotation

The rotate tool allows you to orient a model around the x, y, and z axes. Select this tool (or press r) and click on the object you want to move. Three ring-shaped visual markers will appear over the object, which allows you to rotate the object around the x, y, and z axes.

- You can also just click on the object itself and hold the x, y, or z keys while dragging it to constrain the motion to one of these axes.
- You can hold the Ctrl key to snap the movement to 45 degree increments.
- If the object is not aligned with the world, you can hold the Shift key so the visual markers show up aligned with the world, and you can rotate about the world axes.

Scale

The scale tool allows you to resize a model in the x, y, and z directions. Currently the scale tool only works with simple shapes, i.e. box, cylinder and sphere. Select this tool (or press s) and click on a simple shape. A three axes visual marker will appear over the object, which allows you to scale the x, y, and z dimensions of the object.

- You can also just click on the object itself and hold the x, y, or z keys while dragging it to constrain the scaling to one of these axes.
- You can hold the Ctrl key to scale in 1 metre increments.

TASK 1 Insert three Jersey Barriers and a unit sphere (radius = 0.5m) at the positions shown in Figure 4. Note that all objects are resting on the ground plane and the sphere is located at `(x = 5, y = 0, z = 0.5)`.

Once you are happy with your world it can be save through the File menu. Select the File menu now, and choose Save World As.

A pop-up will appear asking you to enter a new filename. Enter `task1.world` and click ok. A saved world may be loaded on the command line:

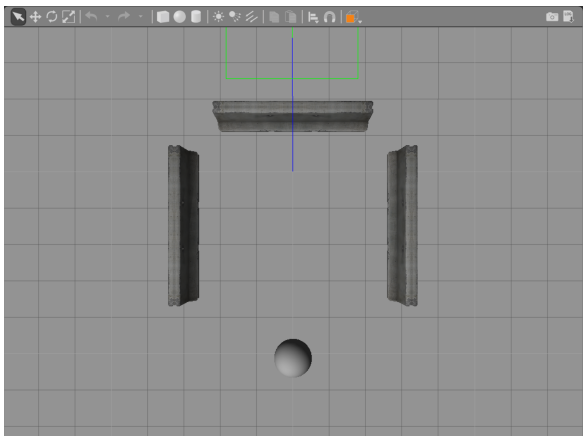
```
gazebo task1.world
```

The filename must be in the current working directory, or you must specify the complete path.

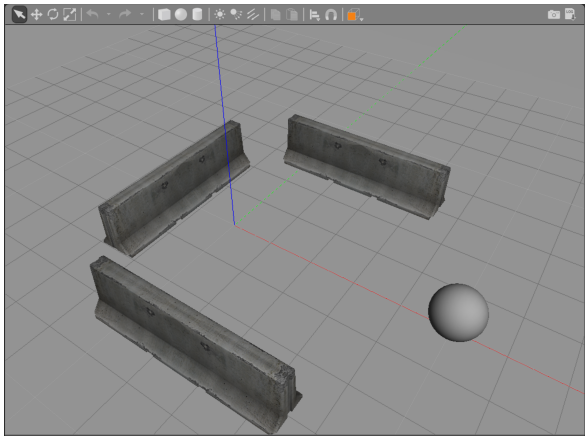
Hint: for precise positioning, go to the list of Models, choose a model and expand the `pose` property. You should be able to directly enter values in the fields for `x`, `y`, `z`, `roll`, `pitch` and `yaw`.

Note : Your demonstrator will assess this task once you have finished it.

There are 5/25 marks for successful completion of this task.



(a) Object positions for Task 1, viewed from above.



(b) Object positions for Task 1, viewed from front.

Figure 4: Object positions for Task 1

Simulating using the physics engine

Models in Gazebo can have forces or torques applied to them and the resulting dynamics can be simulated.

In the World tab, select the physics item. A list of physics properties will be displayed in the list box below. You may need to adjust some of these properties if your computer is struggling under computational load.

- The `enable physics` check-box can be used to disable physics while allowing plugins and sensors to continue running.
- The `real time update rate` parameter specifies in Hz the number of physics updates that will be attempted per second. If this number is set to zero, it will run as fast as it can. Note that the product of `real time update rate` and `max step size` represents the `target real time factor`, or ratio of simulation time to real-time.
- The `max step size` specifies the time duration in seconds of each physics update step.

To apply a force to an model, select it with the mouse, right click and choosing "Apply Force/-Torque". Using your world from Task 1, do this for the sphere model. This will bring up a dialogue giving you various options. You can enter values in the boxes or use the mouse to interact with the visual markers that appear in the render window.

If your simulation was running when you opened the 'Apply Force/Torque" dialogue, clicking "Apply Force" will apply it instantly. Otherwise, if your simulation was paused, you need to resume it to see the effect of applying the force. It's a good idea to have the simulation paused when you apply forces so that the dialogue is not in the way when you resume it to see the effect.

To get the sphere back to its original position, pause the simulation and change the `pose` property back to its original values.

Now for some fun!

TASK 2 In your world from Task 1, insert a cylinder with a radius of 0.16 and height of 3 at the coordinates `(x = 1, y = 0, z = 1.5)`.

Hint: You can check the size of the cylinder by expanding its link→visual→geometry property.

Apply a force of sufficient strength (at least several kN) in the appropriate direction to send the sphere crashing into the cylinder and knocking it over!

Note : Your will need to demonstrate the sphere knocking over the cylinder to your demonstrator.

There are 10/25 marks for successful completion of this task.

Integrating Gazebo with ROS

There are many ways to start Gazebo, open world models and spawn robot models into the simulated environment. We will be using `roslaunch` to do so as it will also instantiate a ROS master for communication with any robot models that are spawned into the Gazebo world.

For example, to start an empty Gazebo world using `roslaunch` simply run

```
$ roslaunch gazebo_ros empty_world.launch
```

This command takes a ROS package name (in this case `gazebo_ros`) and a filename to launch Gazebo with that is within the package (in this case `empty_world.launch`). The launch file sets a number of options and loads a selected world into Gazebo, including spawning any robot models you might be testing. The `gazebo_ros` package contains a number of example Gazebo launch files and worlds.

Creating your own Gazebo ROS Package

Before spawning robots into Gazebo, you need to understand the file hierarchy standards for using ROS with Gazebo.

Everything concerning a robot's model and description is located, as per ROS standards, in a package named `/myrobot_description` and all the world files and launch files used with Gazebo is located in a ROS package named `/myrobot_gazebo`. Normally you replace `myrobot` with the name of your robot (must be in lower case letters), but for this workshop we will replace it with 'ws03'.

With these two packages, a typical hierarchy under a workspace folder called `ws03_ws` would be as follows:

```
../ws03_ws/src
    /myrobot_description
        package.xml
        CMakeLists.txt
         /urdf
            myrobot.urdf
        /meshes
            mesh1.dae
            mesh2.dae
            ...
            /materials
            /cad
    /myrobot_gazebo
```

```
/launch
    myrobot.launch
/worlds
    myrobot.world
/models
    world_object1.dae
    world_object2.stl
    world_object3.urdf
/materials
/plugins
```

We are now going to build a new ROS package with this hierarchy by first creating a catkin workspace[1].

Change to your home directory in the VM, create a source folder called `ws03_ws/src` and create and build a catkin workspace called `ws03_ws`

```
$ cd
$ mkdir -p ws03_ws/src
$ cd ws03_ws
$ catkin_make
```

The `catkin_make` command is a convenience tool for working with catkin workspaces. Running it the first time in your workspace, it will create a `CMakeLists.txt` link in your 'src' folder.

Additionally, if you look in your current directory you should now have a 'build' and 'devel' folder. Inside the 'devel' folder you can see that there are now several setup.*sh files. Sourcing any of these files will overlay this workspace on top of your environment.

Now, to create the folder that will contain the launch and world files, run the commands

```
$ cd src
$ catkin_create_pkg ws03_gazebo
```

Notice that the `ws03_gazebo` folder has been created. We do not need a `/ws03_description` folder for the moment as we don't have a robot to put into the simulation.

Creating a Custom World File

You can create custom .world files within your own ROS packages that are specific to your robots and packages.

To start off, have a quick look at the world file you created for Task 1 by using the `nano` editor

[1] A catkin workspace is a folder where you modify, build, and install catkin packages. Catkin is a collection of CMake macros and associated code used to build packages used in ROS.

```
$ nano ~/task1.world
```

All of the information required to build your world is in this file, coded in XML format. You can easily change any of the model or simulation parameters by editing this file without having to load Gazebo.

Now, it's time to build your own world within a ROS package.

Assuming you are still in the `src` folder, create a `launch` folder within the `ws03_gazebo` package, and then create a `ws03.launch` file in that folder:

```
$ mkdir -p ws03_gazebo/launch
$ cd ws03_gazebo/launch
$ nano ws03.launch
```

with the following contents (default arguments excluded):

```
<launch>
  <!-- We inherit the logic in empty_world.launch, changing only the name of
     the world to be launched -->
  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find ws03_gazebo)/worlds/ws03.world"/>
    <!-- more default parameters can be changed here -->
  </include>
</launch>
```

Create a `worlds` folder within the `ws03_gazebo` package and then a `ws03.world` file in that folder:

```
$ cd ..
$ mkdir worlds
$ cd worlds
$ nano ws03.world
```

with the following contents:

```
<?xml version="1.0" ?>
<sdf version="1.4">
  <world name="default">
    <include>
      <uri>model://ground_plane</uri>
    </include>
    <include>
      <uri>model://sun</uri>
    </include>
    <include>
      <uri>model://gas_station</uri>
      <name>gas_station</name>
      <pose>-2.0 7.0 0 0 0 0</pose>
    </include>
  </world>
</sdf>
```

You should now be able to launch your custom world (with a gas station - awesome!) into Gazebo using the following commands in the same terminal window:

```
$ . ~/ws03_ws/devel/setup.bash
$ roslaunch ws03_gazebo ws03.launch
```

Please be patient as the VM will need to download the gas station model [2].

The script on the first line registers your package with ROS. If you open a new terminal window, close the terminal window, or shut down the VM you will need to run it again to register your package with ROS. Note that once you have registered your package with ROS you only need to run the second command to launch your world again, as long as you are within the same terminal window.

You can always check that your package is recognised by ROS by running the following command

```
$ rospack list
```

Congratulations! You have now created a ROS package to integrate with Gazebo.

TASK 3 Show your demonstrator the successful launching of Gazebo and the world file via the `roslaunch` command, using your newly created ROS package.

There are 10/25 marks for successful completion of this task.

[2] If you are impatient, use the `jersey_barrier` model instead.