Gazebo Simulator

Unit 1: Introduction

- Summary -

Estimated time to completion: **4 hrs**

This unit presents the goal and subjects you will learn in this course. In addition, it briefly introduces Gazebo and the motivation for using this important simulator for the robotics community.

- End of Summary -

1.1 Overview

- Briefing -

What is Gazebo?

Gazebo is a simulator for robotics environments maintained by the **Open Source Robotics Foundation (OSRF)**. It allows you to test algorithms rapidly, design robots, train AI systems, and more.

Why work with Gazebo?

Gazebo is the default simulator for Robot Operating System (ROS). It provides dynamic simulations that you can choose among different physic engines, advanced 3D graphics, sensors and noises, plugins for robots, and ready-to-use robot models. Further, it runs over the TCP/IP transport layer, which allows easy network integration, as ROS does.

Visualization and debugging

Running a simulation provides all the information you need about the environment and the robot. You can even debug your software and simulation together to see which one is failing. If you want to test a specific case, you can force a situation and see your software repeatedly running until you get the necessary information.

"Fail fast, learn faster."

You do not need to test your software on a real robot to find issues and tune parameters. Failing on a simulated robot is faster and cheaper. It makes your software get better much faster than trying on a real robot.

Building a robot is too expensive!

Everyone wants to test algorithms on a real robot, but the truth is that having one available is too expensive! Not only for purchasing all the necessary components for building a robot, but it is also time-consuming to have everything mounted and working together. Then, you have the mechanical, electrical, and software parts.

Community

Gazebo is open-source and maintained by **OSRF**. It has a vibrant community that encourages use, improvement, and supporting other users.

- End of Briefing -

1.2 Gazebo Components

- Gazebo components -

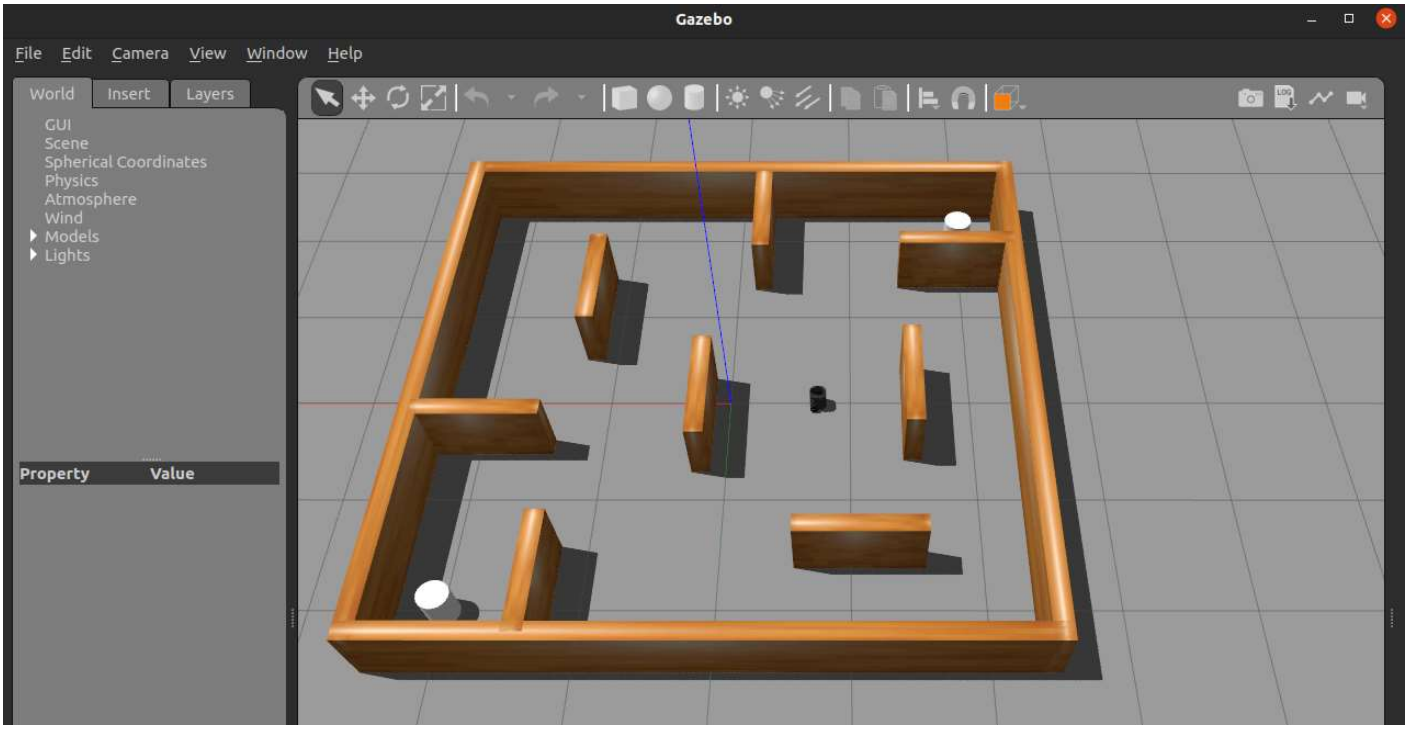
You know what Gazebo is and why it is relevant for robotics development. Now see what Gazebo looks like and how you interact with it! To open Gazebo, launch a ready-to-use simulation. Run the command below in one of the web shells:

▶ Execute

In [ ]:
```
export TURTLEBOT3_MODEL=burger
```

In [ ]:
```
roslaunch turtlebot3_gazebo turtlebot3_stage_4.launch
```

Wait until the simulation gets ready and Gazebo client is open. You must have the following window open in the blank space:

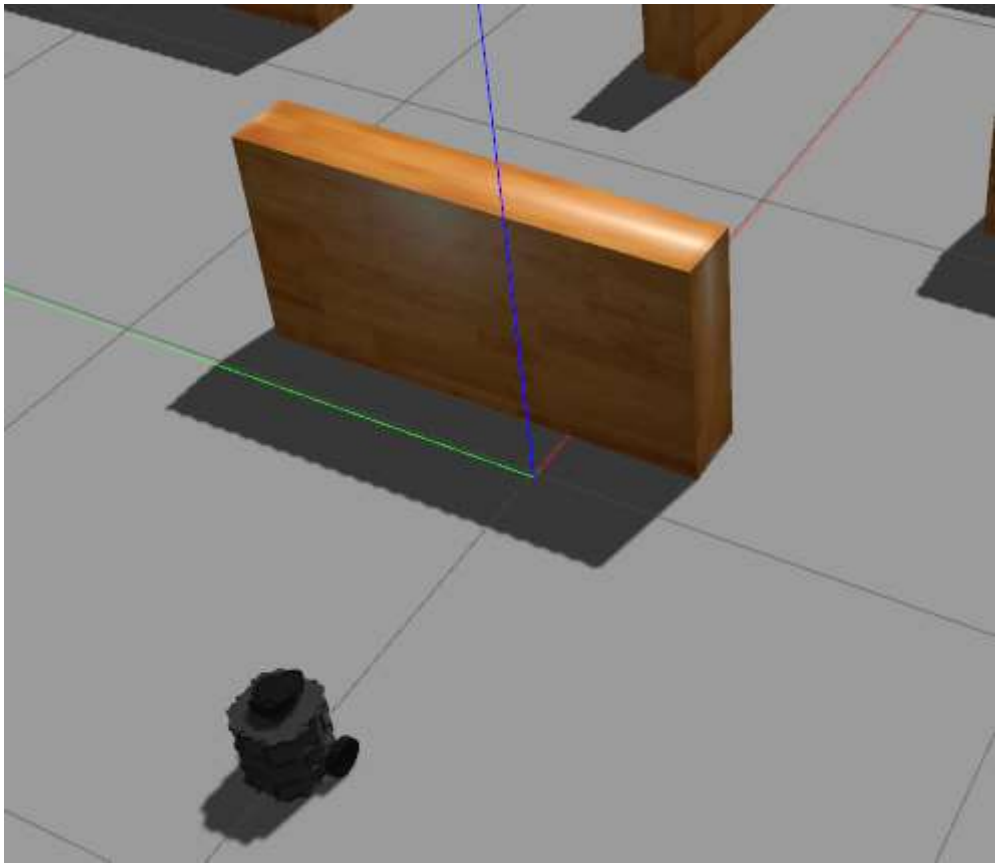


The **simulation** is composed of different components. Now understand each one of them.

World

The **world** is the simulation's whole environment. It contains everything and is described in a **.world** file. This file is formatted using **SDF (Simulation Description Format)** (http://sdformat.org/spec). You can see all the components of the world in the left menu, under the **World** tab.

Gazebo world frame follows the right-hand orientation - the red line for X-axis, green for Y-axis, and blue for Z-axis.



Models

Inside the world you launched, there are five objects you can explore using the World tab like below:


Models

All the models are composed of **links**. Some of them are even composed of other models. You will see how it is done in the next chapter.

Models are also described with **SDF**. They can be described inside a **world** file or in a separate file(**.sdf**) to be reused as often as needed.

Environment Variables

Gazebo uses environment variables to set up its runtime configuration. You can check them by running the following:

▶ Execute

In [ ]:
```
env | grep GAZEBO
```

```
user:~$ env | grep GAZEBO
GAZEBO_MASTER_URI=http://2_xterm:11345
GAZEBO_PLUGIN_PATH=/usr/lib/x86_64-linux-gnu/gazebo-11/plugins:/usr/li
b/x86_64-linux-gnu/gazebo-11/plugins:
GAZEBO_MODEL_DATABASE_URI=http://models.gazebosim.org
GAZEBO_RESOURCE_PATH=/usr/share/gazebo-11:/usr/share/gazebo-11:/usr/sh
are/gazebo
GAZEBO_MASTER_PORT=11345
GAZEBO_MASTER_HOSTNAME=2_xterm
GAZEBO_MODEL_PATH=/usr/share/gazebo-11/models:/usr/share/gazebo-11/mod
els:/usr/share/gazebo-9/models:/usr/share/gazebo/models_robox
```

The ones that end with: **_MASTER_URI**, **_MASTER_PORT**, and **_MASTER_HOSTNAME** are used to define where the Gazebo server is running. In that case, there are specific configurations for the platform. If you have Gazebo running on a local computer, you must have **localhost** as hostname, for example.

Gazebo has its own library of models to be used. Such a library is taken from the folders listed in the variable **GAZEBO_MODEL_PATH** and the online models from **GAZEBO_MODEL_DATABASE**.

These variables are important and make a difference when starting a Gazebo simulation.

Gzserver and Gzclient

Gazebo has both processes divided, meaning the simulation does not need the GUI to run. Do a quick test.

Check ROS and Gazebo Processes

Run the commands:

▶ Execute

In [ ]:
```
rostopic list
```

In [ ]:
```
ps faux | grep gzclient
```

In [ ]:
```
ps faux | grep gzserver
```

You must get some results similar to those below:

```
user:~$ ps faux | grep gzclient
user     22353  0.0  0.0   3980  3020 pts/2    S+   18:18   0:00  |   \_ /bin/bash /usr/local/bin/gzclient
user     22355  151  2.5 5104644 398428 pts/2  SLl+ 18:18   3:41  |       \_ /usr/bin/gzclient-11.5.1
user     23216  0.0  0.0   3312   728 pts/3    S+   18:21   0:00      \_ grep --color=auto gzclient
user:~$ ps faux | grep gzserver
user     20532  0.0  0.0   3980  3068 ?        Ss   18:13   0:00  |       \_ /bin/sh /opt/ros/noetic/lib/gazebo_ros/gzserver -e ode /home
/user/catkin_ws/src/turtlebot3_simulations/turtlebot3_gazebo/worlds/turtlebot3_stage_4.world __name:=gazebo __log:=/home/user/.ros/log/b9
94b202-1bd0-11ec-9190-0242ac150007/gazebo-2.log
user     20603 41.9  1.7 4867612 271928 ?      SLl  18:13   3:27  |           \_ gzserver -e ode /home/user/catkin_ws/src/turtlebot3_simu
lations/turtlebot3_gazebo/worlds/turtlebot3_stage_4.world -s /opt/ros/noetic/lib/libgazebo_ros_paths_plugin.so -s /opt/ros/noetic/lib/lib
gazebo_ros_api_plugin.so --verbose __name:=gazebo __log:=/home/user/.ros/log/b994b202-1bd0-11ec-9190-0242ac150007/gazebo-2.log
user     23232  0.0  0.0   3312   728 pts/3    S+   18:21   0:00      \_ grep --color=auto gzserver
```

Kill the current simulation using **Ctrl+C** in the web shell that has it running.

Check ROS and Gazebo Processes Again

Check the topics with `rostopic list` or processes with `ps faux | grep ...` commands.

You should not see anything running - just the grep process.

```
user:~$ ps faux | grep gzclient
user     24040  0.0  0.0   3312   728 pts/3    S+   18:25   0:00      \_ grep --color=auto gzclient
user:~$ ps faux | grep gzserver
user     24042  0.0  0.0   3312   732 pts/3    S+   18:25   0:00      \_ grep --color=auto gzserver
user:~$
```

Create a new launch file with the following content:

▶ Execute

In [ ]:
```
touch ~/test.launch
```

In [ ]:
```
vim test.launch
```

Copy and paste

In [ ]:

```
<launch>
  <arg name="model" default="burger" doc="model type [burger, waffle, waffle_p
  <arg name="x_pos" default="-0.7"/>
  <arg name="y_pos" default="0.0"/>
  <arg name="z_pos" default="0.0"/>

  <include file="$(find gazebo_ros)/launch/empty_world.launch">
    <arg name="world_name" value="$(find turtlebot3_gazebo)/worlds/turtlebot3_
    <arg name="paused" value="false"/>
    <arg name="use_sim_time" value="true"/>
    <arg name="gui" value="false"/>
    <arg name="headless" value="false"/>
    <arg name="debug" value="false"/>
  </include>


  <param name="robot_description" command="$(find xacro)/xacro --inorder $(fir

  <node name="spawn_urdf" pkg="gazebo_ros" type="spawn_model" args="-urdf -moc

</launch>
```

This is a copy of the previous launch file you ran, except for the line below:

```
<arg name="gui" value="false"/>
```

You are setting the Graphical User Interface **GUI** argument to **false**. It means the simulation will start without the window.

Relaunch the simulation using this file:

▶ Execute

In [ ]:

```
roslaunch ~/test.launch
```

Check ROS and Gazebo Processes

This time you must see ROS topics

▶ Execute

In [ ]:

```
rostopic list
```

But only see `gzserver` running

```
user:~$ ps faux | grep gzclient
user     24822  0.0  0.0   3312   728 pts/3    S+   18:28   0:00      \_ grep --color=auto gzclient
user:~$ ps faux | grep gzserver
user     24467  0.0  0.0   3980  3040 ?        Ss   18:28   0:00  |       \_ /bin/sh /opt/ros/noetic/lib/gazebo_ros/gzserver -e ode /home
/user/catkin_ws/src/turtlebot3_simulations/turtlebot3_gazebo/worlds/turtlebot3_stage_4.world __name:=gazebo __log:=/home/user/.ros/log/d6
8f928a-1bd2-11ec-97d1-0242ac150007/gazebo-2.log
user     24538 67.6  1.7 4933148 271708 ?      SLl  18:28   0:06  |           \_ gzserver -e ode /home/user/catkin_ws/src/turtlebot3_simu
lations/turtlebot3_gazebo/worlds/turtlebot3_stage_4.world -s /opt/ros/noetic/lib/libgazebo_ros_paths_plugin.so -s /opt/ros/noetic/lib/lib
gazebo_ros_api_plugin.so --verbose __name:=gazebo __log:=/home/user/.ros/log/d68f928a-1bd2-11ec-97d1-0242ac150007/gazebo-2.log
user     24824  0.0  0.0   3312   660 pts/3    S+   18:28   0:00      \_ grep --color=auto gzserver
user:~$
```

The simulation is running correctly, but you do not have any GUI. The client can be launched separately. Execute the command below:

▶ Execute

In [ ]:
```
gzclient
```

After a few seconds, you must have the simulation shown in the Graphical Tools again! In the same simulation, this time, the GUI was launched separately.

It is important to understand these two processes. Sometimes you may have Gazebo running but cannot see it. That is because the GUI may not have started, but `gzserver` did.

When working with ROS, it is not common to use such commands because you rely on **launch files** to start or kill a simulation. Although you may fall into errors during the development, that knowledge will help you start, stop or even kill the simulation separately when needed.

Plugins

Gazebo plugins are used to interface Gazebo with external programs. There are two ways of loading such plugins: **command line** or inside the **SDF** files.

When working with ROS, plugins are loaded in the **SDF** file since you use **launch files** to start a simulation. Your **launch files** include **URDF** robot models, which are converted to **SDF** when passed to Gazebo. Inside your robot models, you have plugin configurations. These will set up robot interfaces like motor controllers and sensors.

These plugins are in charge of interfacing Gazebo simulation and your ROS programs. You will use some of them in the course to create and set up robots!

- End of Gazebo components -

- Practice 1.2 -

Modify the file **test.launch** created to change the spawning position of the robot. Use the arguments **x_pos**, **y_pos**, and **z_pos** to do so.

This exercise does not have a solution, so use this time to practice and ensure you understand the world frame reference.

- End of Practice 1.2 -

1.3 Graphical User Interface

- Graphical User Interface -

You already know how Gazebo works, understand the difference between **gzserver** and **gzclient** and know about Gazebo components. Now explore the tools the Gazebo Graphical Interface provides.

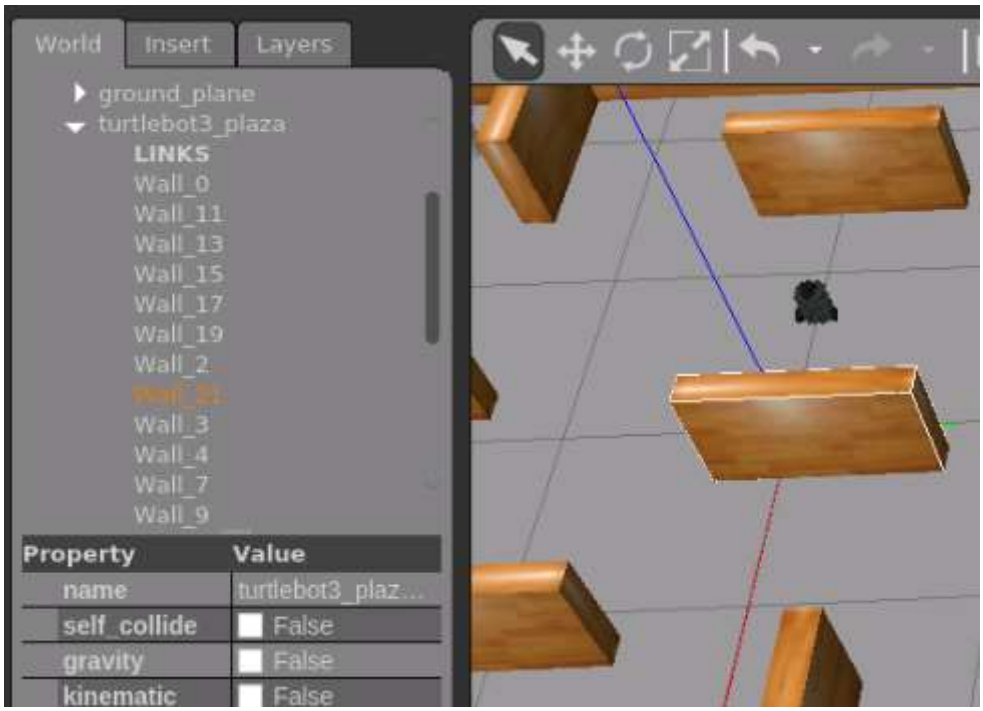
Toolbar

There is a toolbar at the top of the simulation window. Review its features one by one.

The first four options are useful for selecting, moving, rotating, and scaling models in the simulation.

Selector

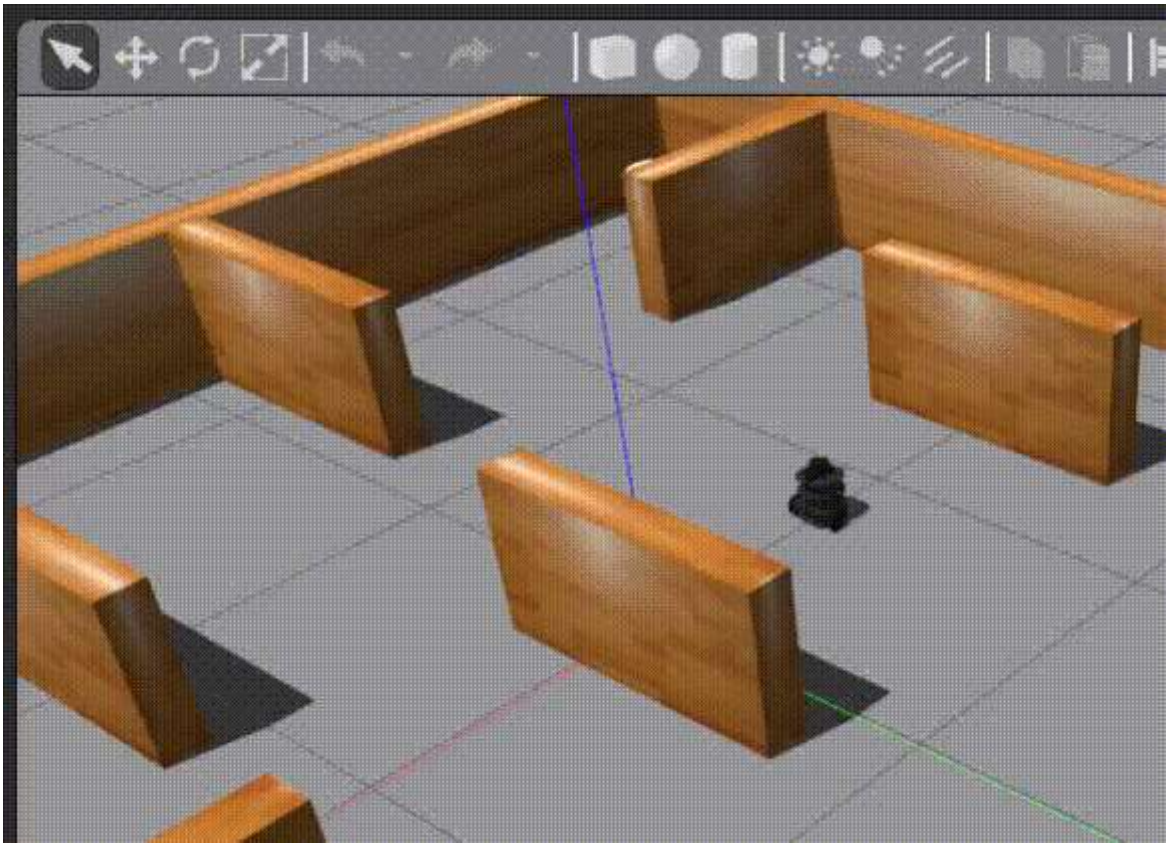
The first option is a simple selection mode. Its details are displayed on the left-side panel whenever you click over an object. You can even click over its children models, if needed. For example, the image below shows the details of a wall inside the **plaza** model.



Translation Mode

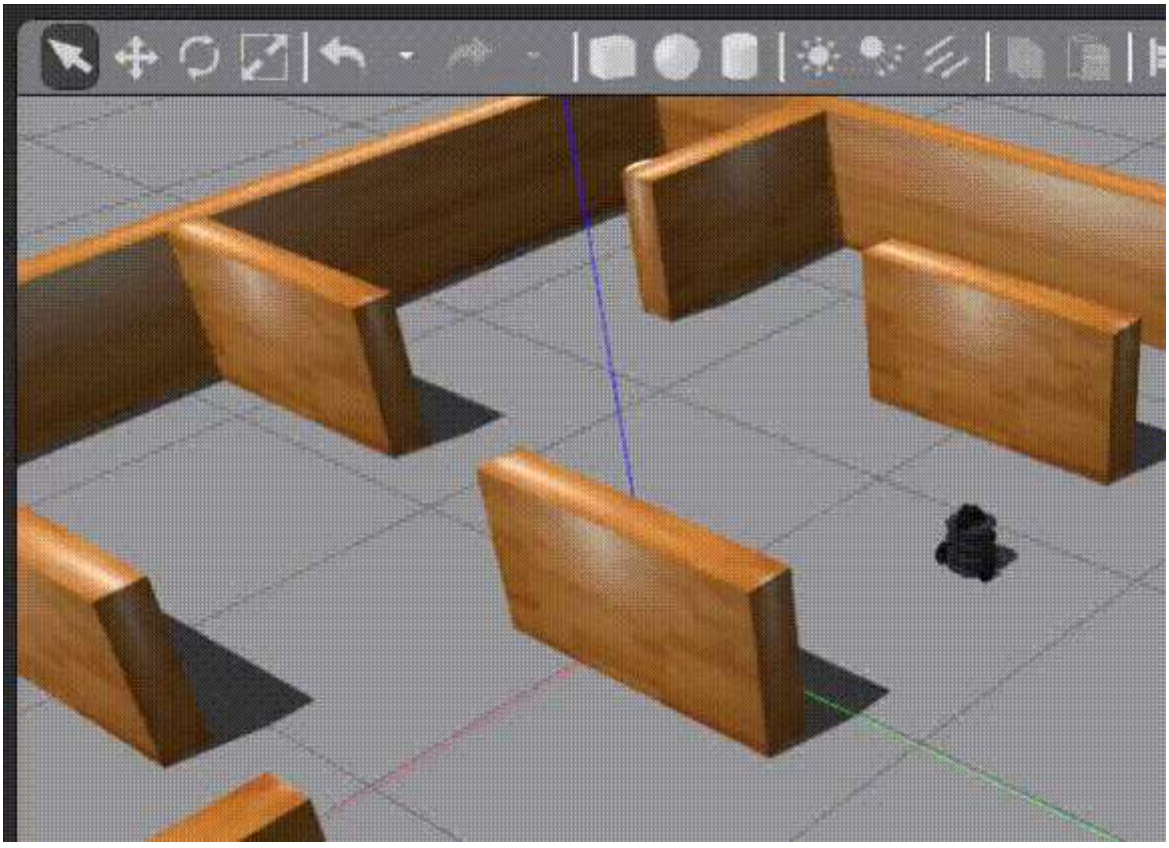
The second option is the translation mode. Select it and drag any model into the simulation. It will follow the mouse pointer if you keep the left button pressed.

It is also possible to click over the object once and use one of the axis arrows to move the model in that direction only.



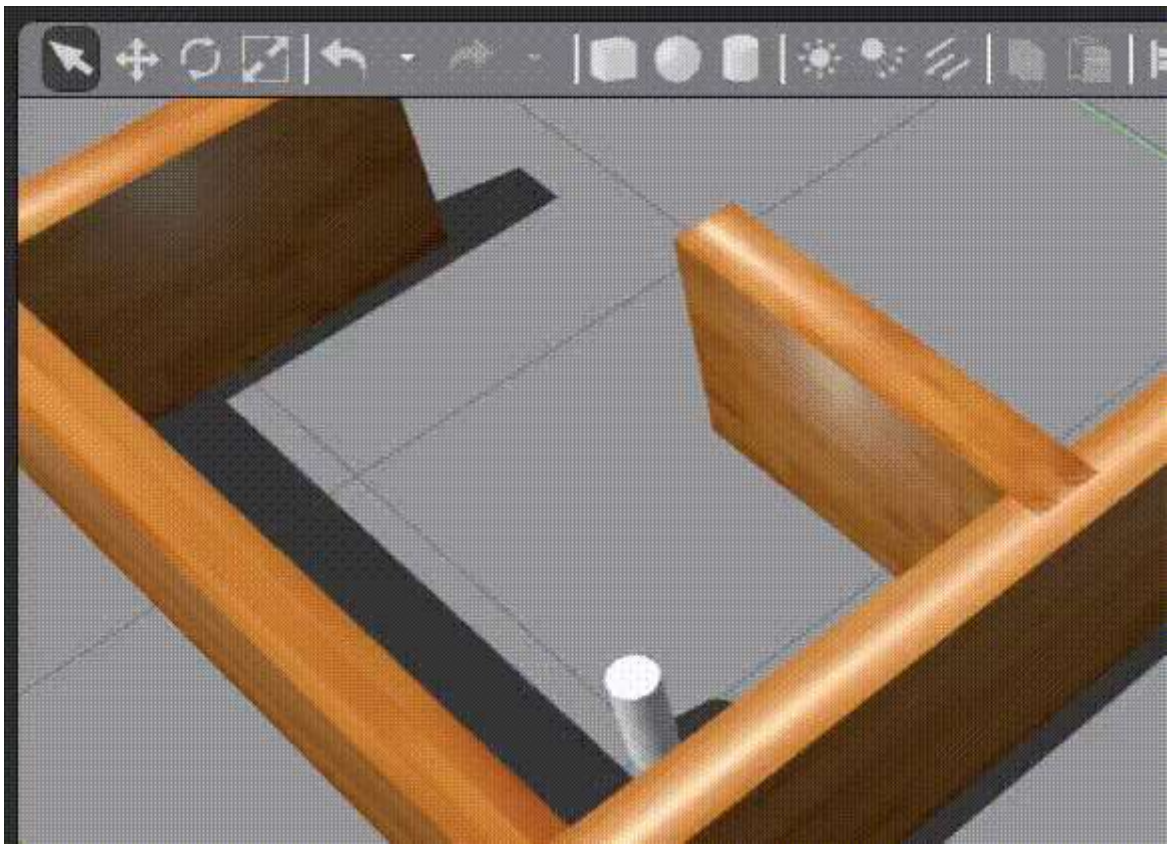
Rotation Mode

The third option is used to rotate the model. The three possible rotation angles will be displayed as circles whenever you select the model to be rotated. Drag one of them to rotate the model.



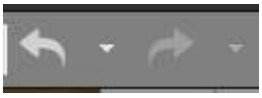
Scale Mode

The fourth option is used to scale models. Only the simple ones can be scaled. The ones composed of more than one link cannot be scaled.



Edit

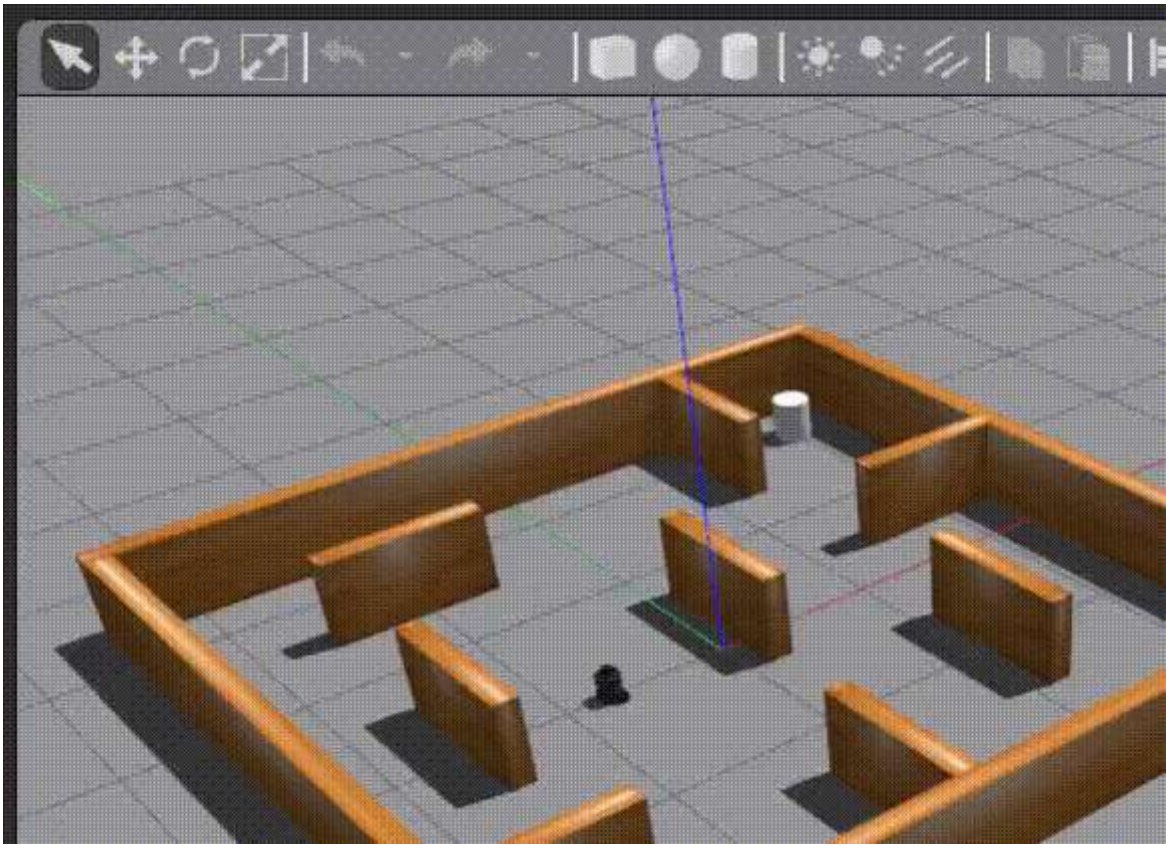
The following two options are used to **undo** or **redo** edit actions. The same actions are available in the **Edit** window tab.



Insert Models or Light

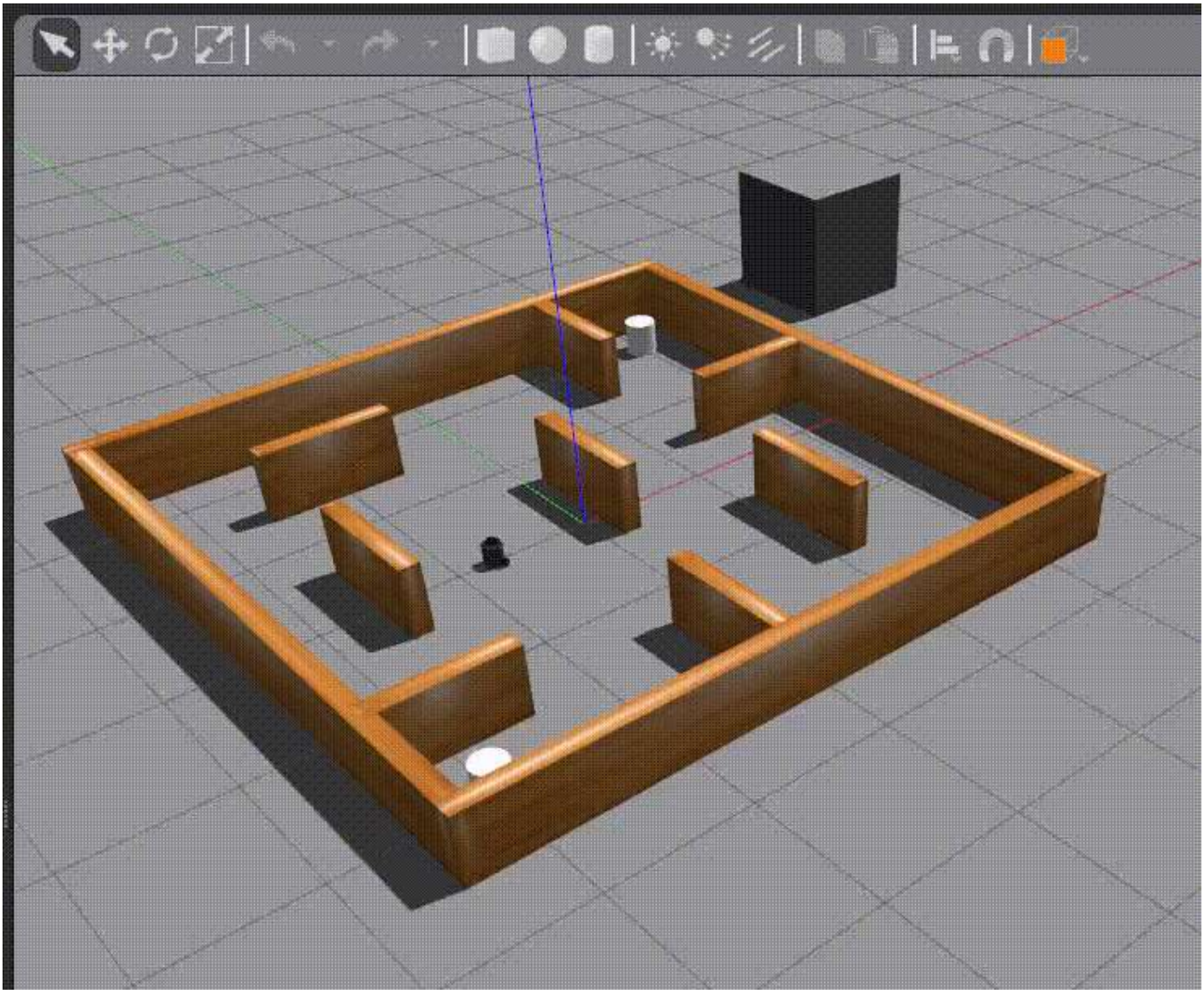
The following icons insert simple models or lights into the simulation.

You can try each one of them to understand how it works. It is a new object selected, and you can translate it to put it into the world. Notice how new lights are added to affect the luminosity and shadows.



Toolbar Copy and Paste

A very simple feature, but useful and worth it to be mentioned, copy and paste.



Toolbar Align and Snap

Different objects can be aligned using the X, Y, or Z-axis. Snap is another option to put objects together. Make a try of these options. They are very helpful in putting objects in the desired position when creating worlds.

Toolbar Copy and Paste

Toolbar Perspective

Finally, the last option is an easy way to change the camera's perspective. You have two sub-option per axis (both sides per axis) and the initial position configured in the world file.

You can even try the Orthographic view. Unfortunately, it does not render shadows or neither perspective visual effects. However, it is useful for debugging and also consumes fewer graphical resources.

Toolbar Copy and Paste

Mouse Control

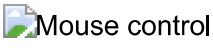
You might have tried exploring the world at this point. Let us clarify how you can do this using the mouse.

There are three actions. First, **move around**. You do so using the **left button** on the mouse.

With the **middle button**, it is possible to rotate around a certain point.

Finally, with the **right button**, zoom in and zoom out. The same can be done using the mouse wheel.

Every click will show a **yellow marker** for the translation, rotation, or zooming **reference**.

 Mouse control

- End of Graphical User Interface -

- Practice 1.3 -

Practicing what you have learned:

- Create new objects in the simulation
- Put new objects available in the toolbar
- Use the alignment tool to put them in a row
- Use the scale option to resize some of them.

This exercise does not have a right solution, so use this time to practice and ensure you understand how to use the toolbar.

- End of Practice 1.3 -