Module:4

ROS Programming

Credit hours:4 Hours

Dr. Abhishek Rudra Pal
Assistant Professor (Senior Grade-1)
SMEC
Vellore Institute of Technology - Chennai Campus
Email: abhishek.rudrapal@vit.ac.in
Mobile No: +91-9043534478;+91-9051728314(Whatsapp)

What is Gazebo?

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

Installing Gazebo with ROS

Summary of Compatible ROS and Gazebo Combinations

This table includes all currently supported versions of ROS and Gazebo. All other ROS and Gazebo releases are end of life and we do not recommend their continued use.

	GZ Citadel (LTS)	GZ Fortress (LTS)	GZ Garden	GZ Harmonic (LTS)
ROS 2 Jazzy (LTS)[1]	❌	❌	⚡	✅
ROS 2 Rolling	❌	✅	⚡	⚡
ROS 2 Iron	❌	✅	⚡	⚡
ROS 2 Humble (LTS)	❌	✅	⚡	⚡
ROS 2 Foxy (LTS)	✅	❌	❌	❌
ROS 1 Noetic (LTS)	✅	⚡	❌	❌

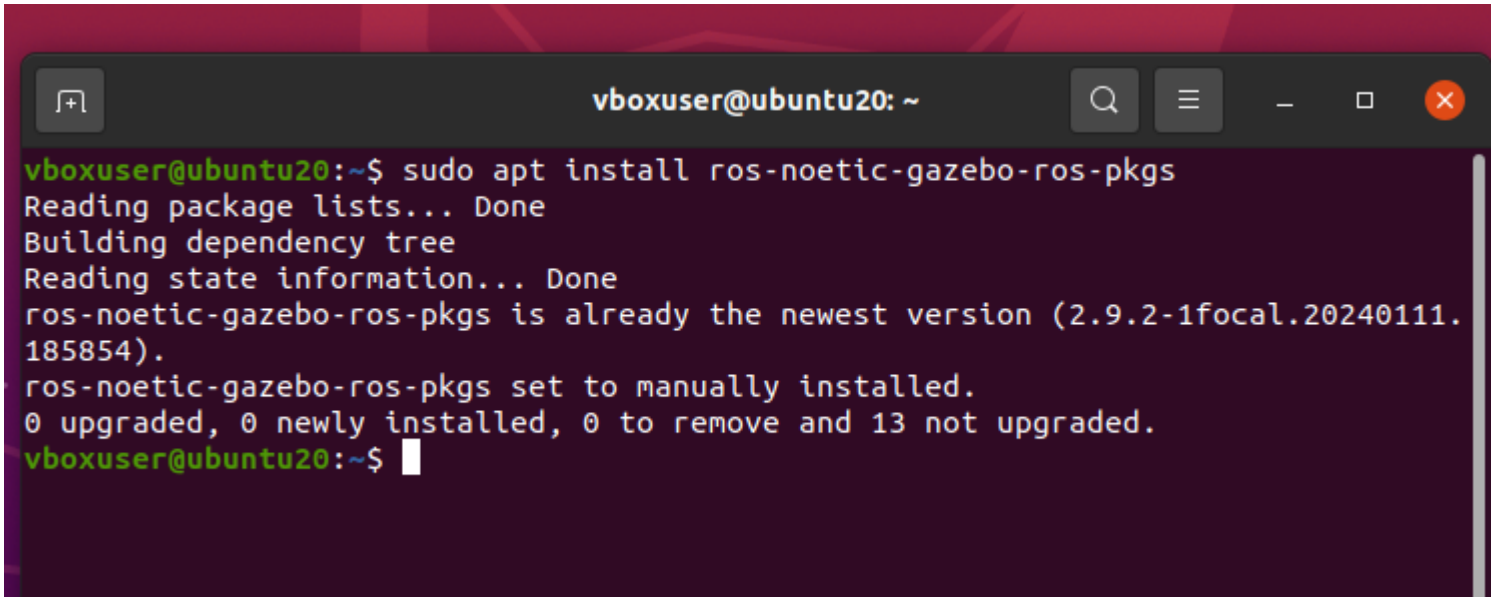
- ✅ - Recommended combination
- ❌ - Incompatible / not possible.
- ⚡ - Possible, *but use with caution*. These combinations of ROS and Gazebo can be made to work together, but some effort is required.

[1] When ROS 2 Jazzy is released, it will be paired with Gazebo Harmonic.

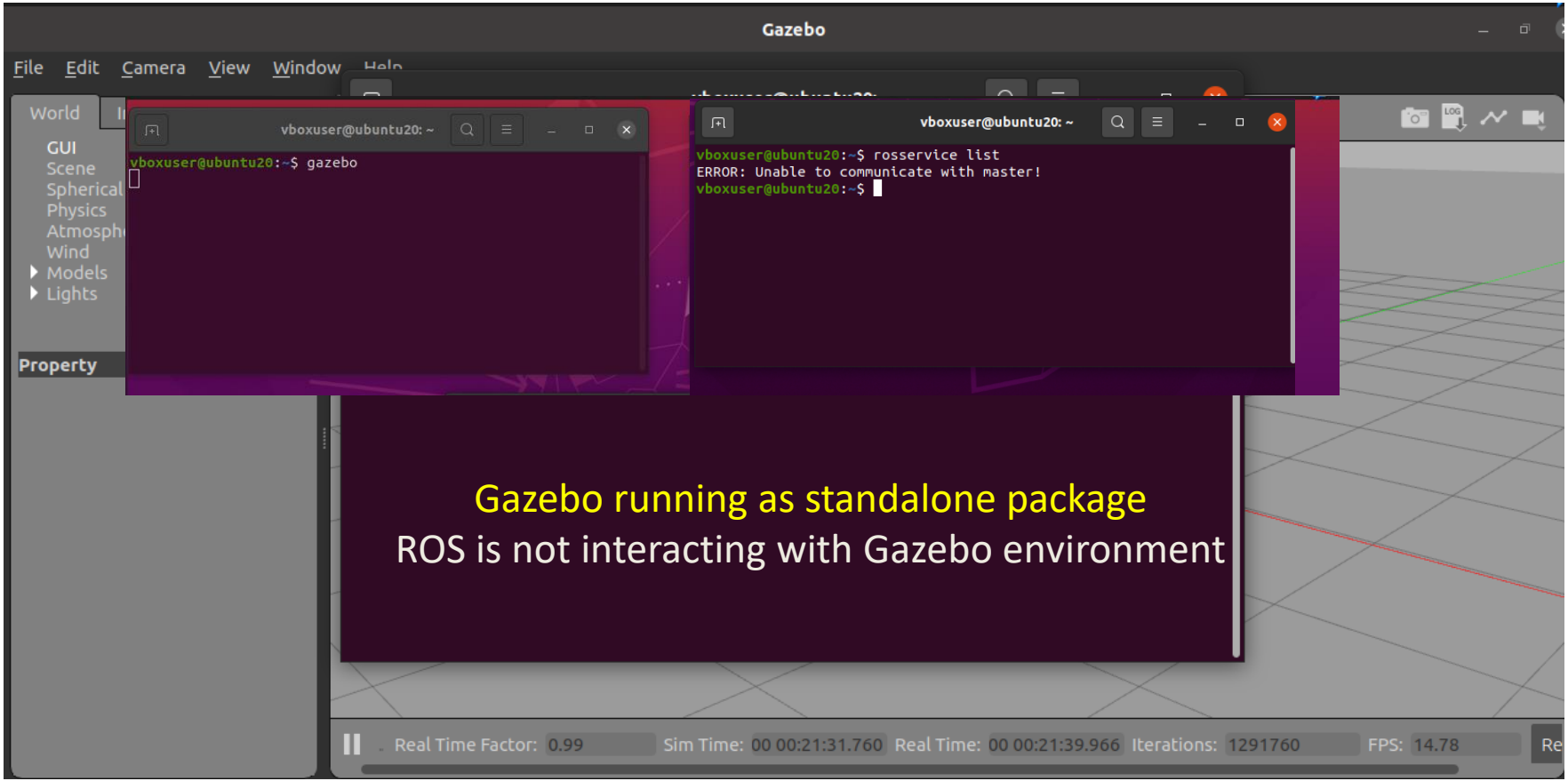
Installing Gazebo with ROS

With ROS already installed, the easiest way to install Gazebo with the correct dependencies and plugins to get it working with ROS is with the following command:

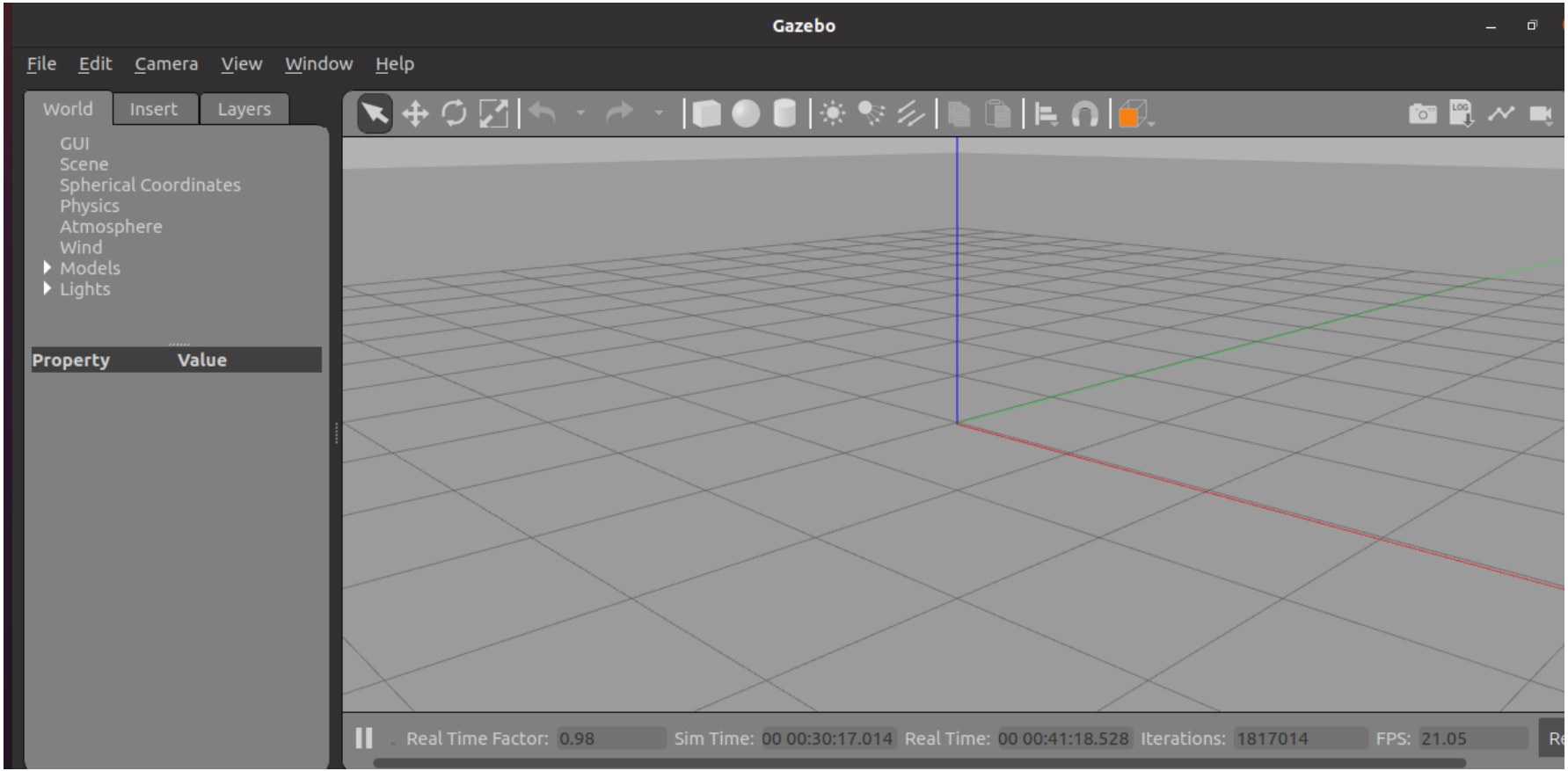
sudo apt install ros-noetic-gazebo-ros-pkgs



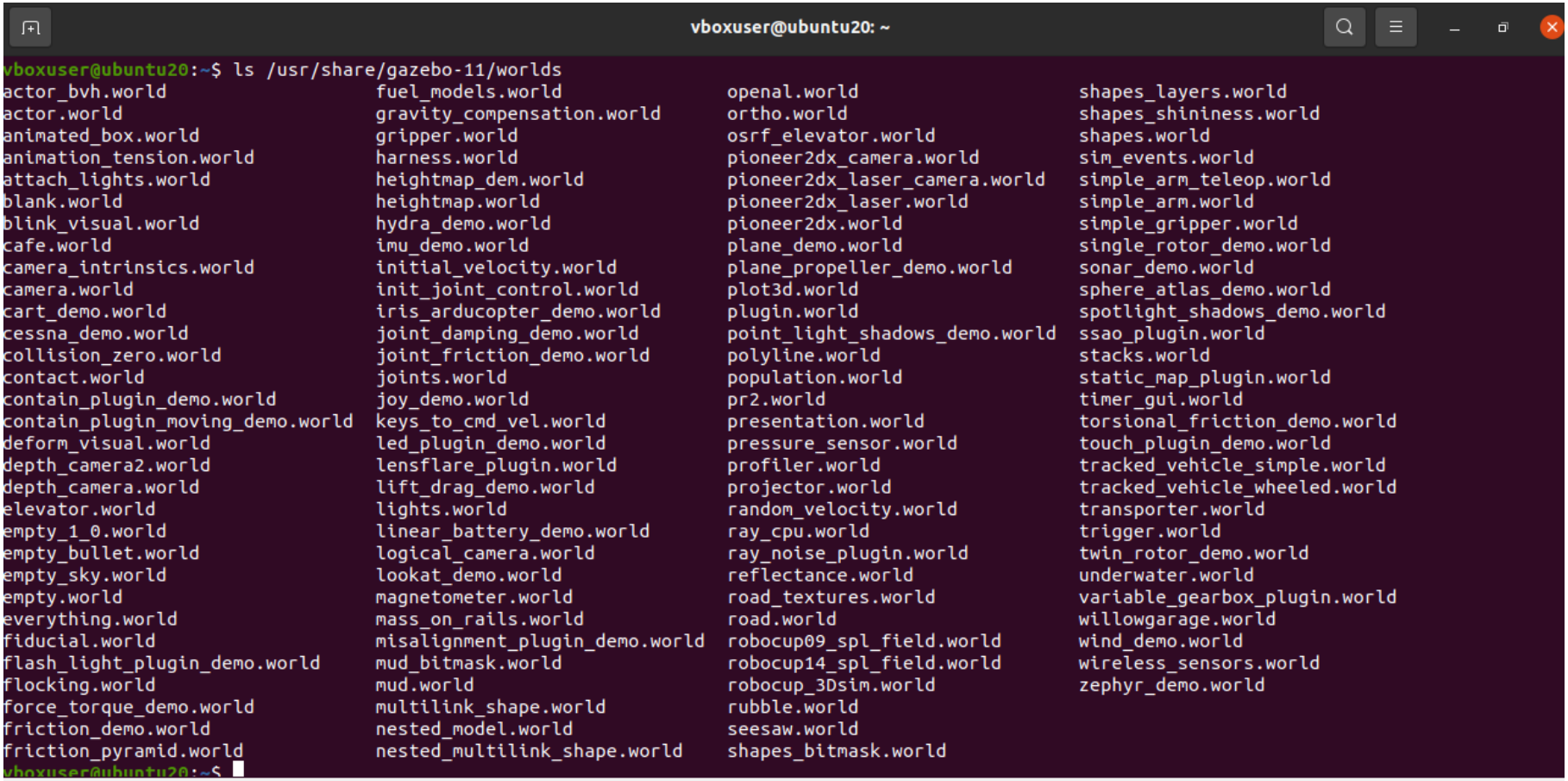
Checking Gazebo



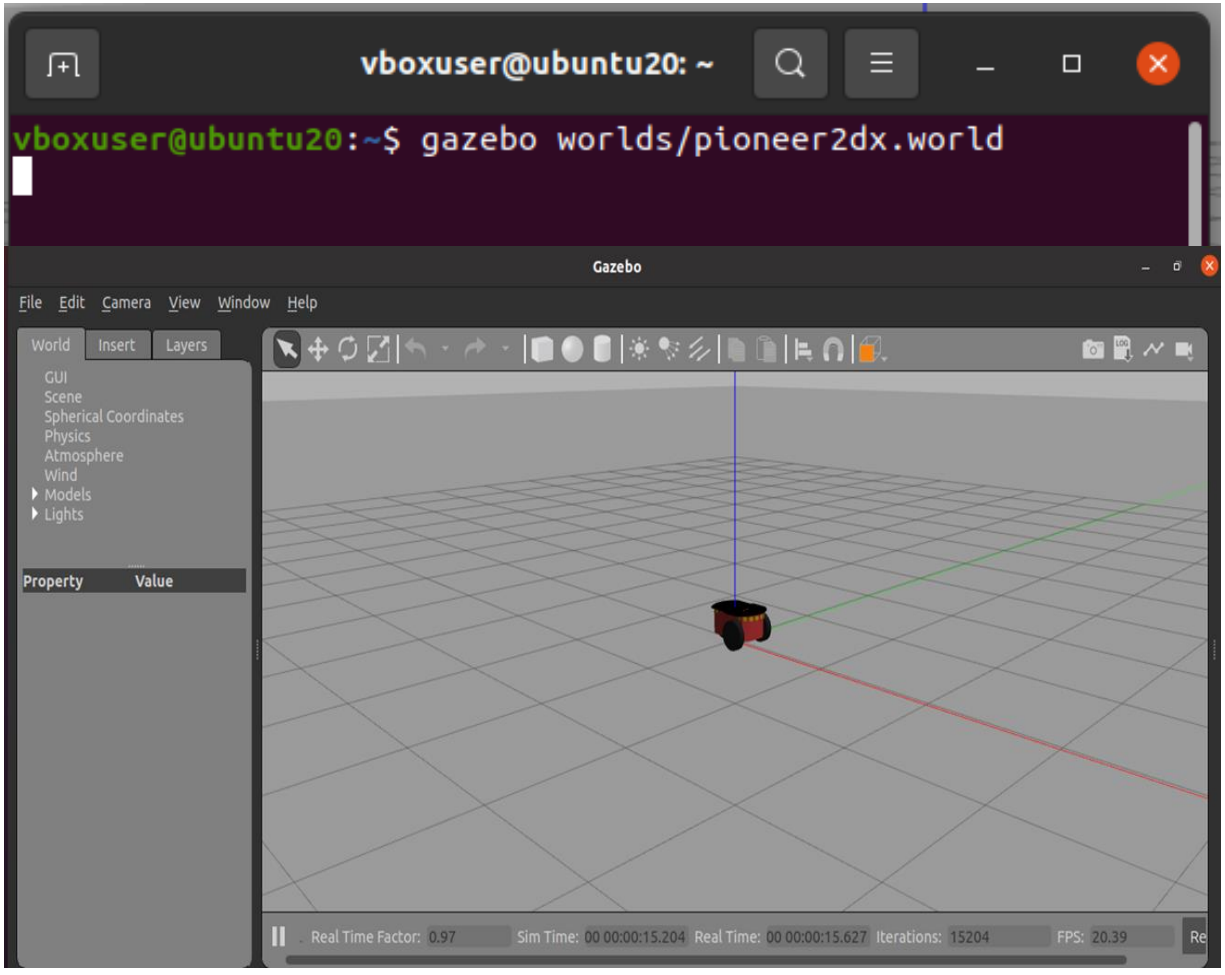
Checking Gazebo



Run Gazebo with a robot

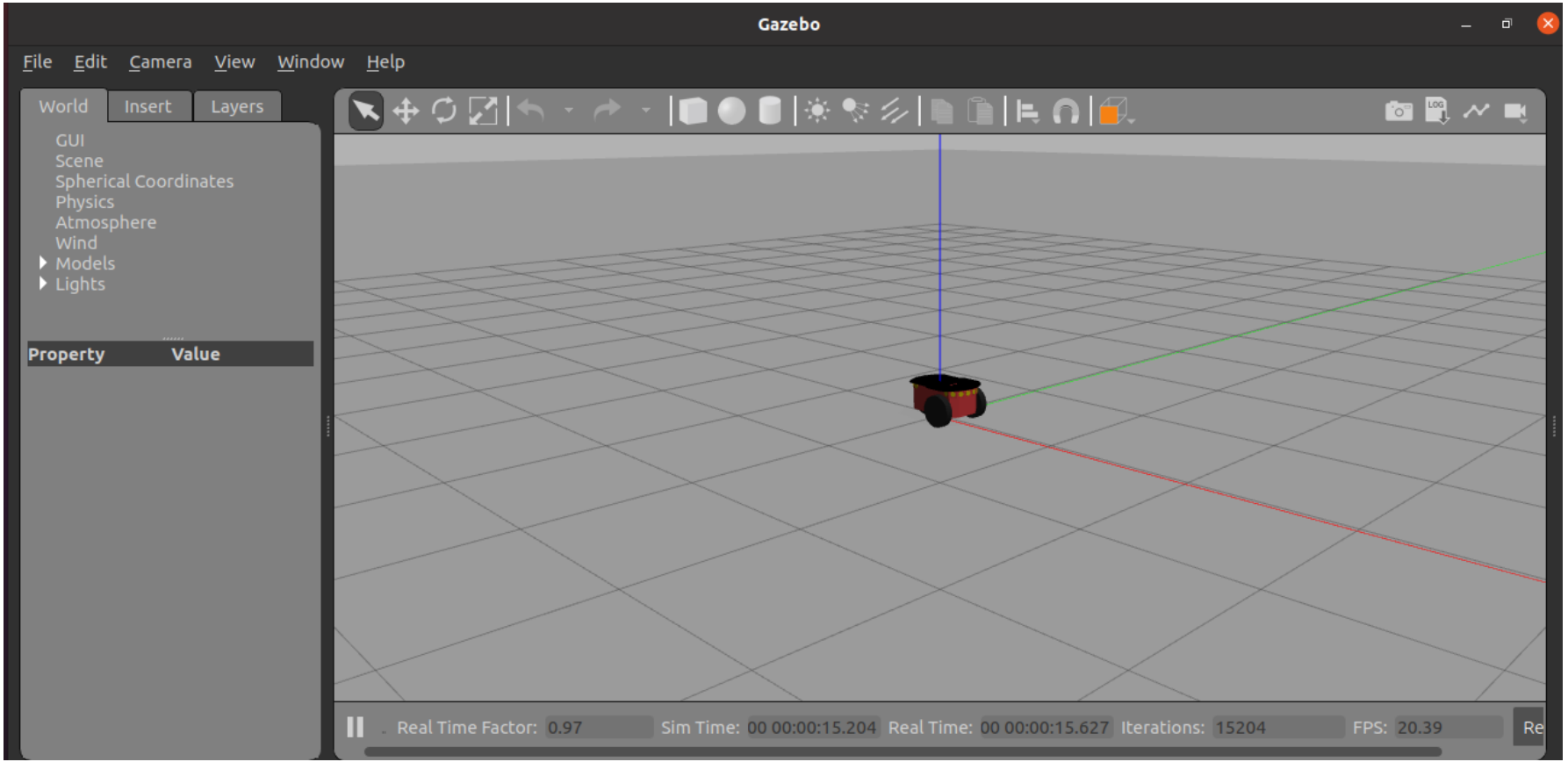


Run Gazebo with a robot



Note: If you don't have the pioneer2dx model already, Gazebo will download it from the online model database which may take some time.

Run Gazebo with a robot



Client and server separation

The **gazebo** command actually runs two different executables for you. 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 visualization of simulation, and convenient controls over various simulation properties.

Try running each of these executables. Open a terminal and run the server:

gzserver

Open another terminal and run the graphical client:

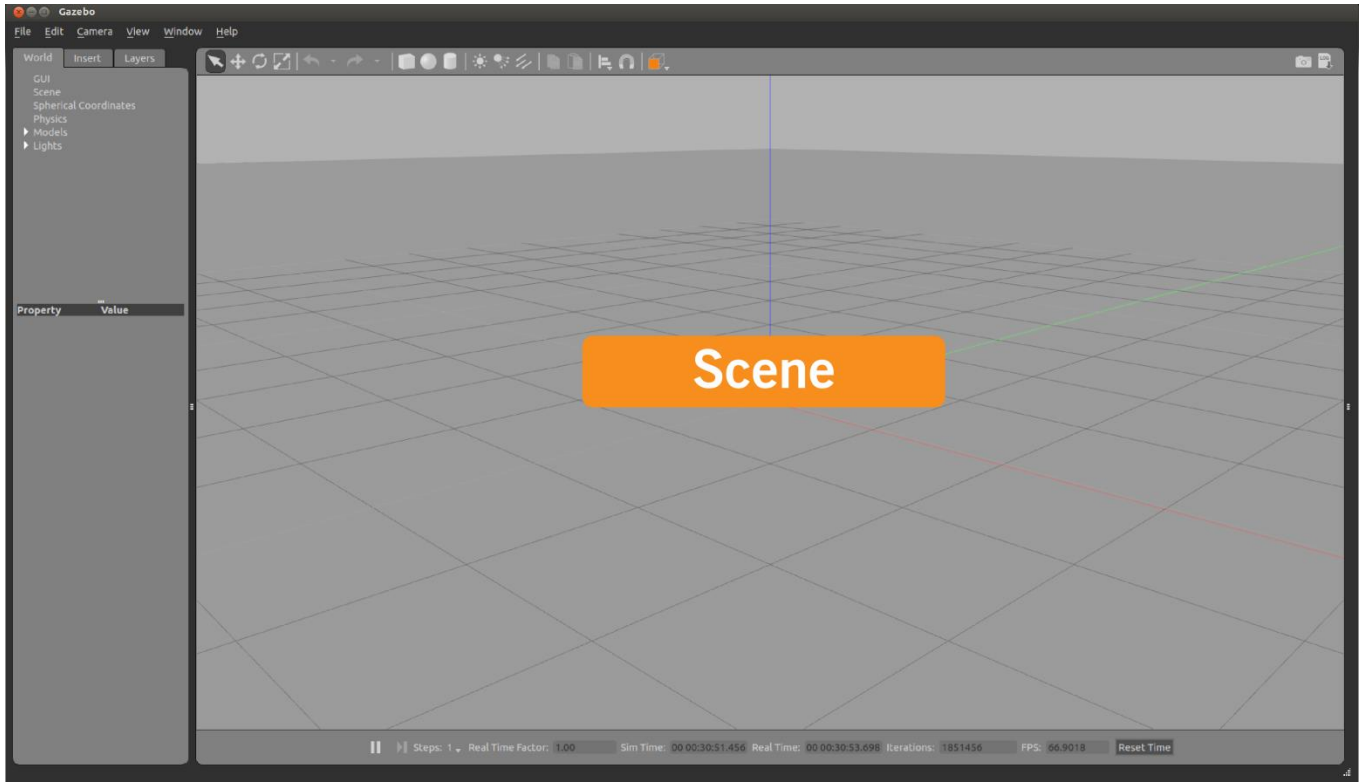
gzclient

At this point you should see the Gazebo user interface. You restart the gzclient application as often as you want, and even run multiple interfaces.

User Interface

The Scene

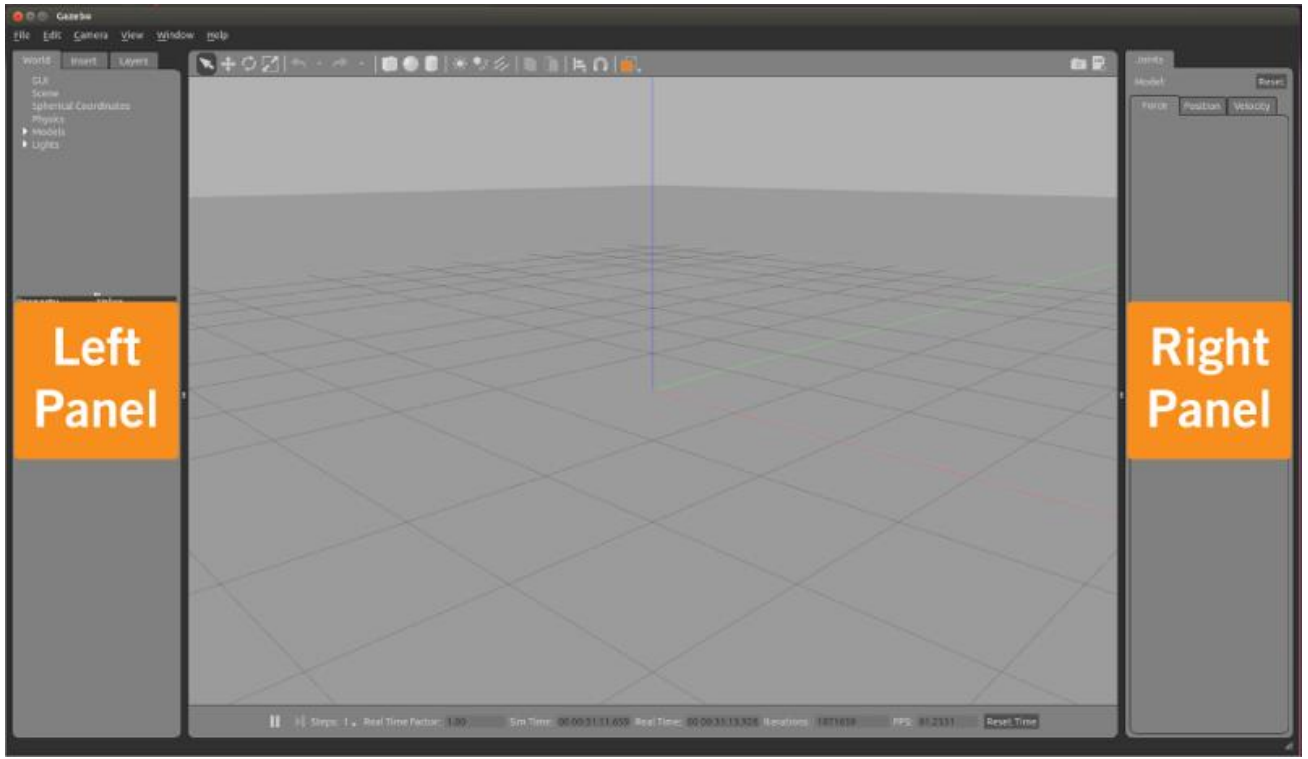
The Scene is the main part of the simulator. This is where the simulated objects are animated and you interact with the environment.



User Interface

The Panels

Both side panels—right and left—can be displayed, hidden or resized by dragging the bar that separates them from the Scene



User Interface

Left Panel

The left panel appears by default when you launch Gazebo. There are three tabs in the panel:

•**WORLD**: The World tab displays the models that are currently in the scene, and allows you to view and modify model parameters, like their pose. You can also change the camera view angle by expanding the "GUI" option and tweaking the camera pose.

•**INSERT**: The Insert tab is where you add new objects (models) to the simulation. To see the model list, you may need to click the arrow to expand the folder. Click (and release) on the model you want to insert, and click again in the Scene to add it.

•**LAYERS**: The Layers tab organizes and displays the different visualization groups that are available in the simulation, if any. A layer may contain one or more models. Toggling a layer on or off will display or hide the models in that layer.

•This is an optional feature, so this tab will be empty in most cases.

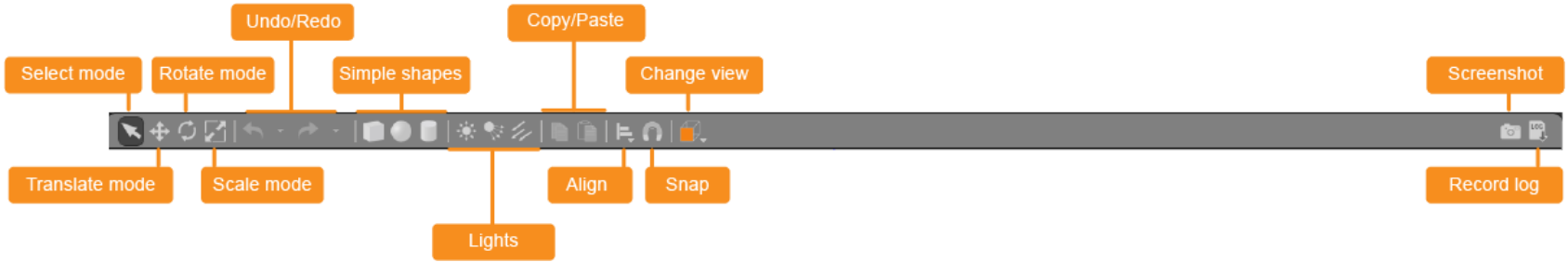
Right Panel (hidden by default)

The right panel is hidden by default. Click and drag the bar to open it. The right panel can be used to interact with the mobile parts of a selected model (the joints). If there are no models selected in the Scene, the panel does not display any information.

User Interface

The Toolbars

The Gazebo interface has two Toolbars. One is located just above the Scene, and the other is just below.



Upper Toolbar

The main Toolbar includes some of the most-used options for interacting with the simulator, such as buttons to: select, move, rotate, and scale objects; create simple shapes (e.g. cube, sphere, cylinder); and copy/paste. Go ahead and play around with each button to see how it behaves.

Select mode: Navigate in the scene
Translate mode: Select models you want to move
Rotate mode: Select models you want to rotate
Scale mode: Select models you want to scale
Undo/Redo: Undo/redo actions in the scene
Simple shapes: Insert simple shapes into the scene
Lights: Add lights to the scene
Copy/paste: Copy/paste models in the scene
Align: Align models to one another
Snap: Snap one model to another
Change view: View the scene from various angles

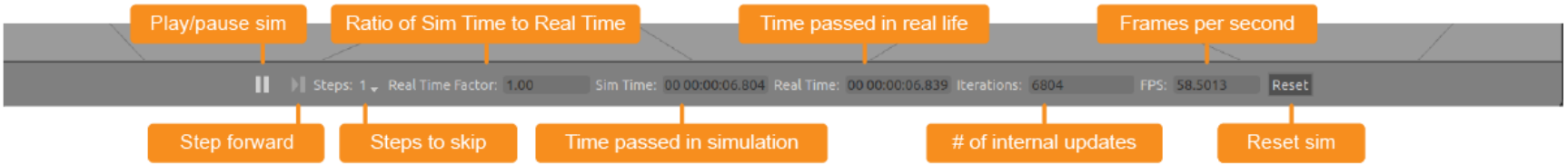
User Interface

Bottom Toolbar

The Bottom Toolbar displays data about the simulation, like the simulation time and its relationship to real-life time. "Simulation time" refers to how quickly time is passing in the simulator when a simulation is running. Simulation can be slower or faster than real time, depending on how much computation is required to run the simulation.

"Real time" refers to the actual time that is passing in real life as the simulator runs. The relationship between the simulation time and real time is known as the "real time factor" (RTF). It's the ratio of simulation time to real time. The RTF is a measure of how fast or slow your simulation is running compared to real time.

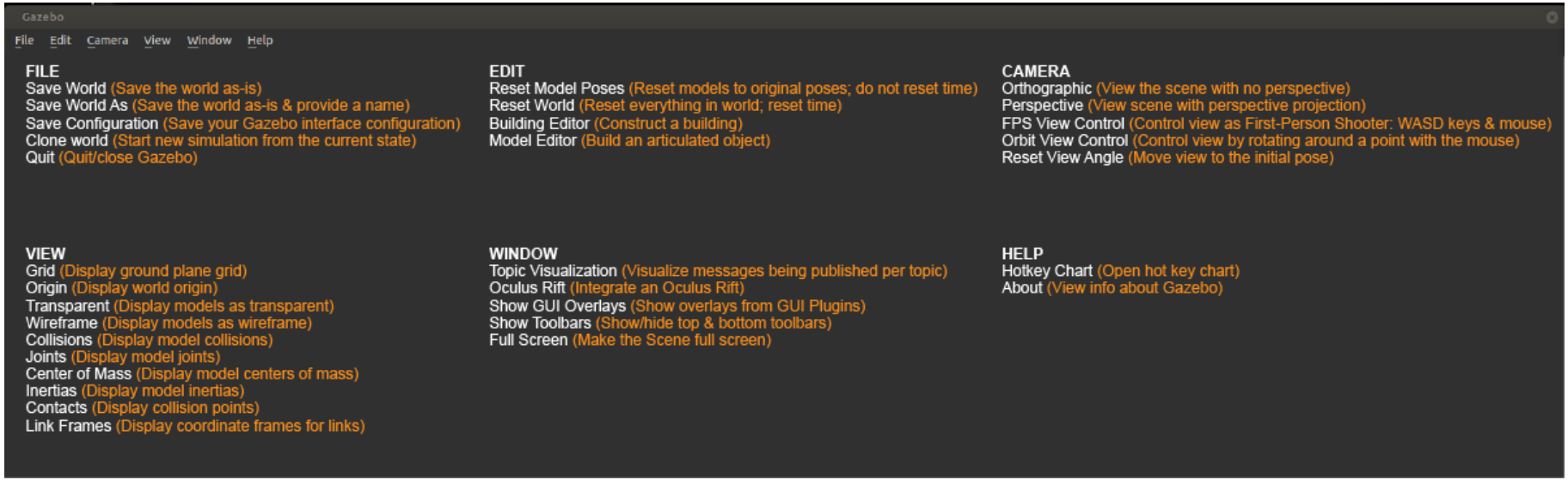
The state of the world in Gazebo is calculated once per iteration. You can see the number of iterations on the right side of the bottom toolbar. Each iteration advances simulation by a fixed number of seconds, called the step size. By default, the step size is 1 ms. You can press the pause button to pause the simulation and step through a few steps at a time using the step button.



User Interface

The Menu

Like most applications, Gazebo has an application menu up top. Some of the menu options are duplicated in the Toolbars or as right-click context menu options in the Scene. Check out the various menus to familiarize yourself.

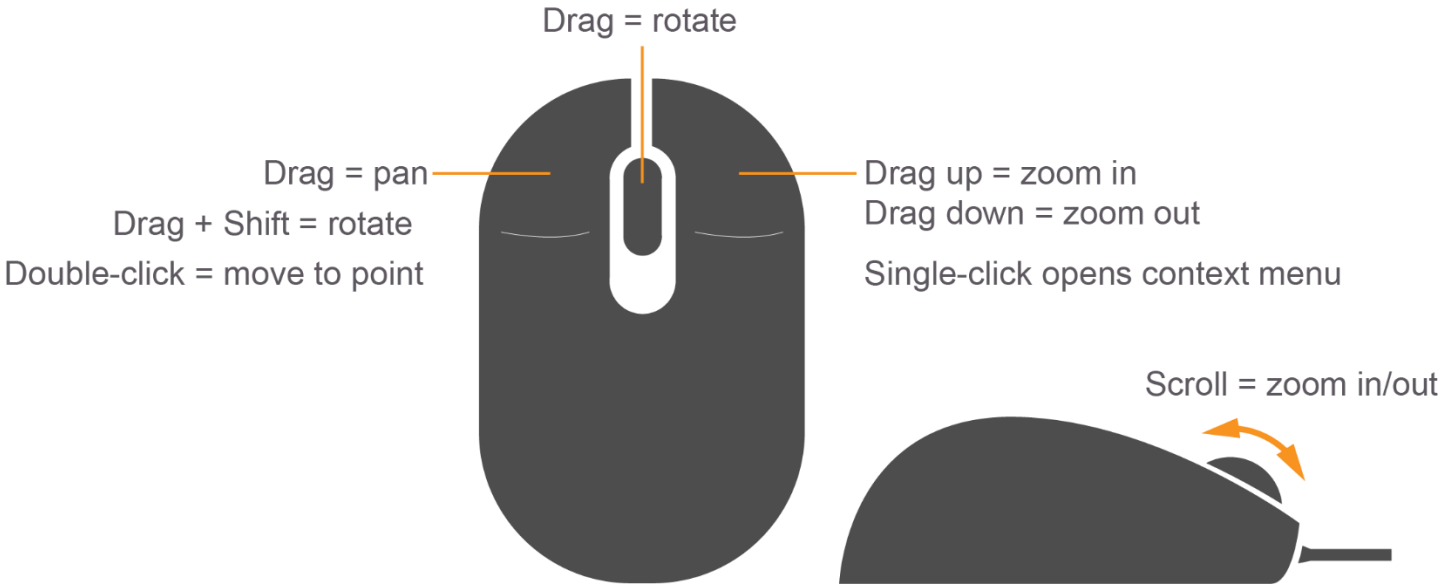


User Interface

Mouse Controls

The mouse is very useful when navigating in the Scene. We highly recommend using a mouse with a scroll wheel. Below are the basic mouse operations for navigating in the Scene and changing the view angle.

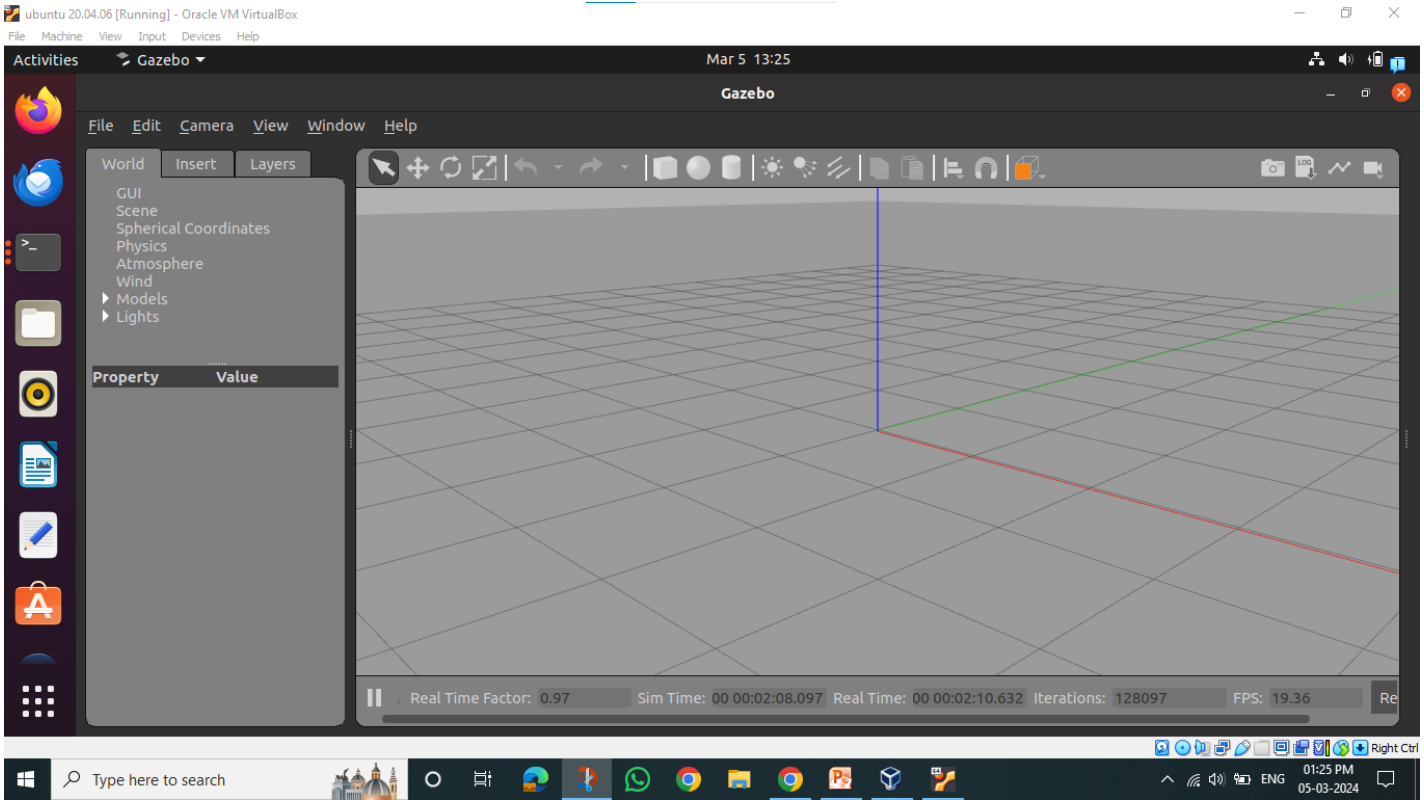
Right-clicking on models will open a context menu with various options. Right-click on a model now to see what's available.



Checking Gazebo

rosrun gazebo_ros gazebo

The Gazebo GUI should appear with nothing inside the viewing window.

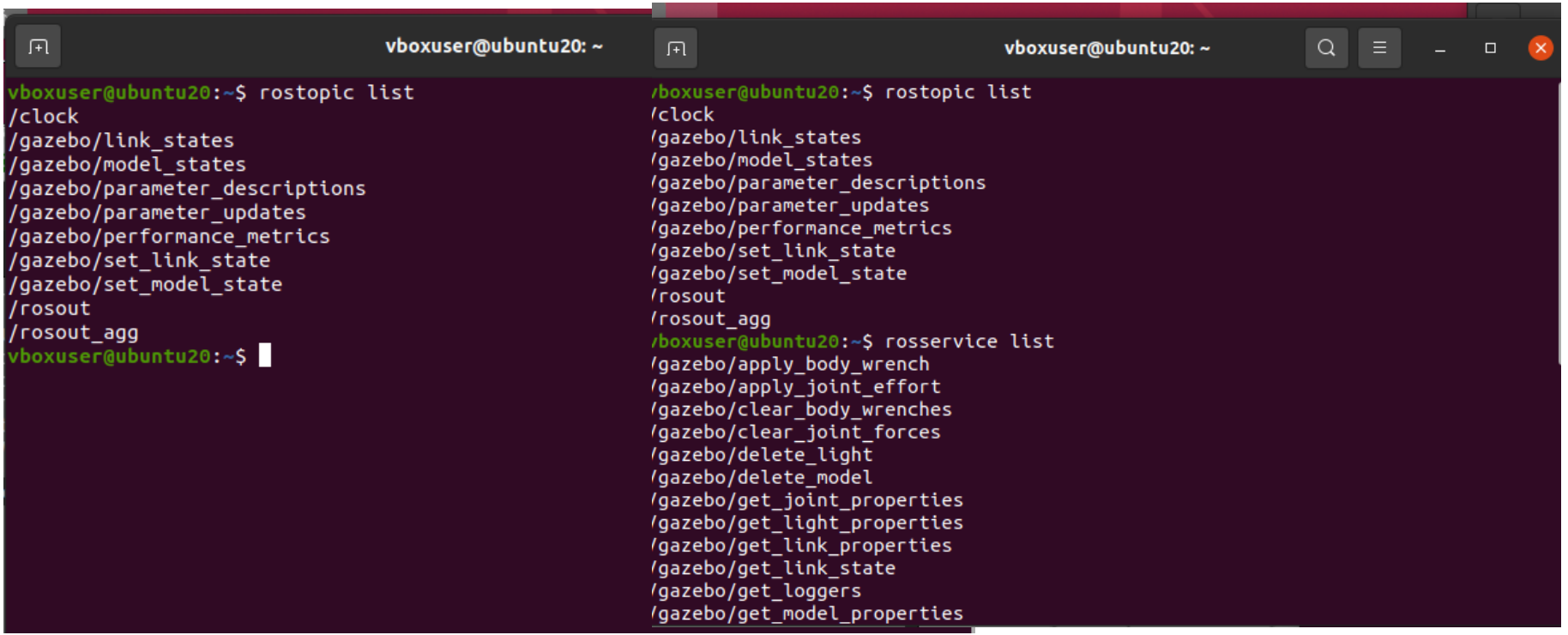


Checking Gazebo

To verify that the proper ROS connections are setup, view the available ROS topics:

rostopic list

rosservice list



Checking Gazebo

To verify that the proper ROS connections are setup, view the available ROS topics:

rosmsg list

```
vboxuser@ubuntu20:~$ rosmsg list
actionlib/TestAction
actionlib/TestActionFeedback
actionlib/TestActionGoal
actionlib/TestActionResult
actionlib/TestFeedback
actionlib/TestGoal
actionlib/TestRequestAction
actionlib/TestRequestActionFeedback
actionlib/TestRequestActionGoal
actionlib/TestRequestActionResult
actionlib/TestRequestFeedback
actionlib/TestRequestGoal
actionlib/TestRequestResult
actionlib/TestResult
actionlib/TwoIntsAction
actionlib/TwoIntsActionFeedback
actionlib/TwoIntsActionGoal
actionlib/TwoIntsActionResult
actionlib/TwoIntsFeedback
actionlib/TwoIntsGoal
actionlib/TwoIntsResult
actionlib_msgs/GoalID
actionlib_msgs/GoalStatus
```

vboxuser@ubuntu20: ~

```
dynamic_reconfigure/Config
dynamic_reconfigure/ConfigDescription
dynamic_reconfigure/DoubleParameter
dynamic_reconfigure/Group
dynamic_reconfigure/GroupState
dynamic_reconfigure/IntParameter
dynamic_reconfigure/ParamDescription
dynamic_reconfigure/SensorLevels
dynamic_reconfigure/StrParameter
gazebo_msgs/ContactState
gazebo_msgs/ContactsState
gazebo_msgs/LinkState
gazebo_msgs/LinkStates
gazebo_msgs/ModelState
gazebo_msgs/ModelStates
gazebo_msgs/ODEJointProperties
gazebo_msgs/ODEPhysics
gazebo_msgs/PerformanceMetrics
gazebo_msgs/SensorPerformanceMetric
gazebo_msgs/WorldState
geometry_msgs/Accel
geometry_msgs/AccelStamped
geometry_msgs/AccelWithCovariance
geometry_msgs/AccelWithCovarianceStamped
```

Checking Gazebo

Other ROS Ways To Start Gazebo

There are several `rosrun` commands for starting Gazebo:

- Launch both the server and client together

```
rosrun gazebo_ros gazebo
```

- Launch the Gazebo server only

```
rosrun gazebo_ros gzserver
```

- Launch the Gazebo client only

```
rosrun gazebo_ros gzclient
```

- Launches the Gazebo server only, in debug mode using GDB

```
rosrun gazebo_ros debug
```

- Additionally, you can start Gazebo using `roslaunch`

```
roslaunch gazebo_ros empty_world.launch
```

Gazebo Components

World Files

Model Files

Environment Variables

Gazebo Server

Graphical Client

Server + Graphical Client in one

Plugins

Gazebo Components

World Files

The world description file contains all the elements in a simulation, including robots, lights, sensors, and static objects.

This file is formatted using SDF (Simulation Description Format), and typically has a .world extension.

The Gazebo server (gzserver) reads this file to generate and populate a world.

A number of example worlds are shipped with Gazebo. These worlds are installed in <install_path>/share/gazebo-<version>/worlds;

Gazebo Components

What is SDFormat

SDFormat (Simulation Description Format), sometimes abbreviated as SDF, is an XML format that describes objects and environments for robot simulators, visualization, and control.

Originally developed as part of the Gazebo robot simulator, SDFormat was designed with scientific robot applications in mind.

Over the years, SDFormat has become a stable, robust, and extensible format capable of describing all aspects of robots, static and dynamic objects, lighting, terrain, and even physics.

Gazebo Components

Environment Variables

Gazebo uses a number of environment variables to locate files, and set up communications between the server and clients. Default values that work for most cases are compiled in. This means you don't need to set any variables.

Here are the variables:

GAZEBO_MODEL_PATH: colon-separated set of directories where Gazebo will search for models

GAZEBO_RESOURCE_PATH: colon-separated set of directories where Gazebo will search for other resources such as world and media files.

GAZEBO_MASTER_URI: URI of the Gazebo master. This specifies the IP and port where the server will be started and tells the clients where to connect to.

GAZEBO_PLUGIN_PATH: colon-separated set of directories where Gazebo will search for the plugin shared libraries at runtime.

GAZEBO_MODEL_DATABASE_URI: URI of the online model database where Gazebo will download models from.

These defaults are also included in a shell script:

source <install_path>/share/gazebo/setup.sh

If you want to modify Gazebo's behavior, e.g., by extending the path it searches for models, you should first source the shell script listed above, then modify the variables that it sets.

Gazebo Components

Gazebo Server

The server is the workhorse of Gazebo. It parses a world description file given on the command line, and then simulates the world using a physics and sensor engine.

The server can be started using the following command. Note that the server does not include any graphical interface; it's meant to run headless.

gzserver <world_filename>

The <world_filename> can be:

1.relative to the current directory,

2.an absolute path, or

3.relative to a path component in GAZEBO_RESOURCE_PATH.

4.worlds/<world_name>, where <world_name> is a world that is installed with Gazebo

For example, to use the empty_sky.world which is shipped with Gazebo, use the following command

gzserver worlds/empty_sky.world

Gazebo Components

Graphical Client

The graphical client connects to a running ***gzserver*** and visualizes the elements. This is also a tool which allows you to modify the running simulation.

The graphical client is run using:

gzclient

Server + Graphical Client in one

The gazebo command combines server and client in one executable. Instead of running gzserver worlds/empty.world and then gzclient, you can do this:

gazebo worlds/empty_sky.world

Gazebo Components

Plugins

Plugins provide a simple and convenient mechanism to interface with Gazebo. Plugins can either be loaded on the command line, or specified in an SDF file (see the SDF format).

Plugins specified on the command line are loaded first, then plugins specified in the SDF files are loaded. Some plugins are loaded by the server, such as plugins which affect physics properties, while other plugins are loaded by the graphical client to facilitate custom GUI generation.

Example of loading a system plugin via the command line:

gzserver -s <plugin_filename>

The -s flag indicates it is a system plugin, and <plugin_filename> is the name of a shared library found in GAZEBO_PLUGIN_PATH. For example, to load the RestWebPlugin that ships with Gazebo:

gzserver --verbose -s libRestWebPlugin.so

The same mechanism is used by the graphical client, the supported command line flags are the following:

For example, to load the TimerGUIPlugin:

gzclient --gui-client-plugin libTimerGUIPlugin.so

Turtlebot3

```
vboxuser@ubuntu20:~$ rospack list
```

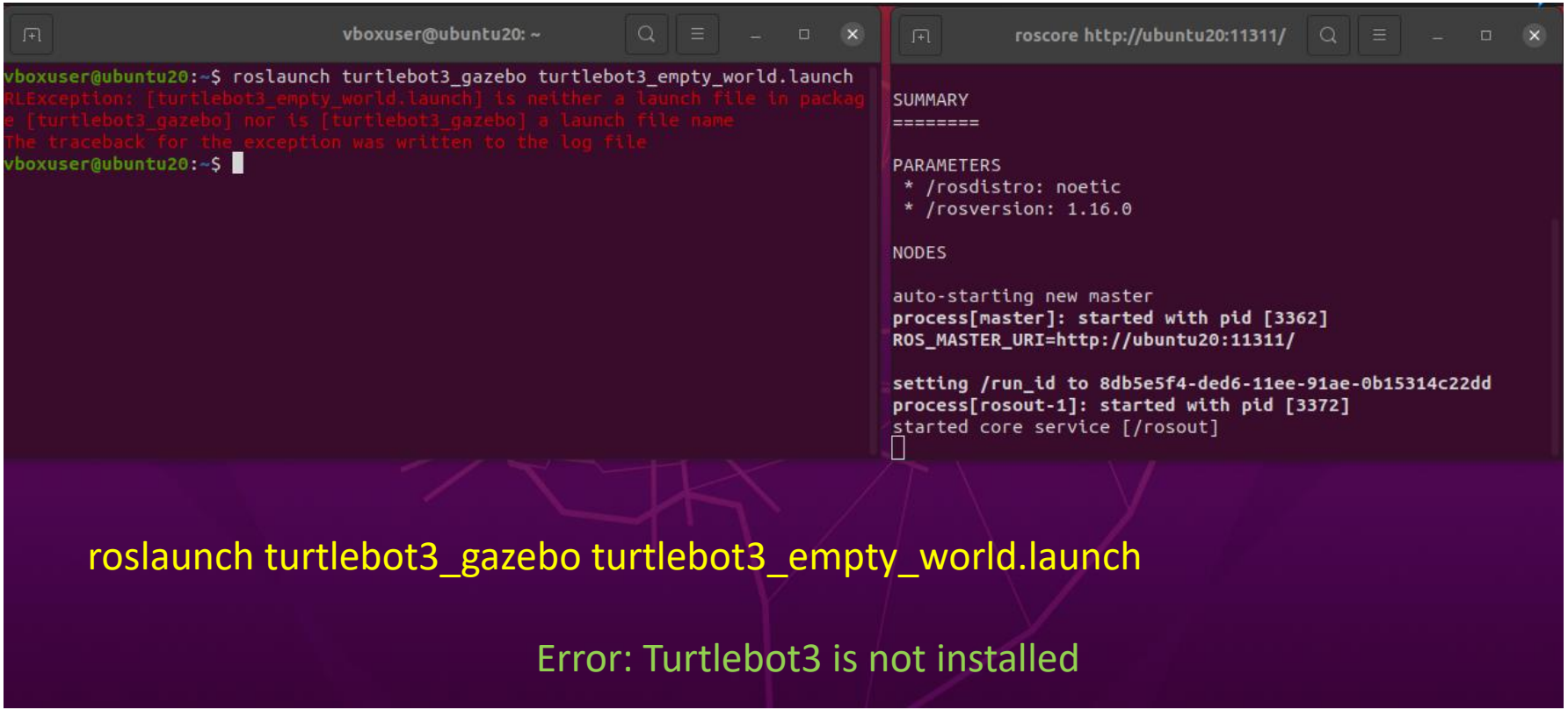
```
vboxuser@ubuntu20:~$ rospack find turtlebot3
[rospack] Error: package 'turtlebot3' not found
vboxuser@ubuntu20:~$
```

Trutlebot3 ?

Need to install

```
stereo_image_proc /opt/ros/noetic/share/stereo_image_proc
stereo_msgs /opt/ros/noetic/share/stereo_msgs
tf /opt/ros/noetic/share/tf
tf2 /opt/ros/noetic/share/tf2
tf2_eigen /opt/ros/noetic/share/tf2_eigen
tf2_geometry_msgs /opt/ros/noetic/share/tf2_geometry_msgs
tf2_kdl /opt/ros/noetic/share/tf2_kdl
tf2_msgs /opt/ros/noetic/share/tf2_msgs
tf2_py /opt/ros/noetic/share/tf2_py
tf2_ros /opt/ros/noetic/share/tf2_ros
tf_conversions /opt/ros/noetic/share/tf_conversions
theora_image_transport /opt/ros/noetic/share/theora_image_transport
topic_tools /opt/ros/noetic/share/topic_tools
trajectory_msgs /opt/ros/noetic/share/trajectory_msgs
transmission_interface /opt/ros/noetic/share/transmission_interface
turtle_actionlib /opt/ros/noetic/share/turtle_actionlib
turtle_tf /opt/ros/noetic/share/turtle_tf
turtle_tf2 /opt/ros/noetic/share/turtle_tf2
turtlesim /opt/ros/noetic/share/turtlesim
urdf /opt/ros/noetic/share/urdf
urdf_parser_plugin /opt/ros/noetic/share/urdf_parser_plugin
urdf_sim_tutorial /opt/ros/noetic/share/urdf_sim_tutorial
urdf_tutorial /opt/ros/noetic/share/urdf_tutorial
visualization_marker_tutorials /opt/ros/noetic/share/visualization_marker_tuto
rials
visualization_msgs /opt/ros/noetic/share/visualization_msgs
webkit_dependency /opt/ros/noetic/share/webkit_dependency
xacro /opt/ros/noetic/share/xacro
xmlrpcpp /opt/ros/noetic/share/xmlrpcpp
vboxuser@ubuntu20:~$
```

Turtlebot3



roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch

Error: Turtlebot3 is not installed

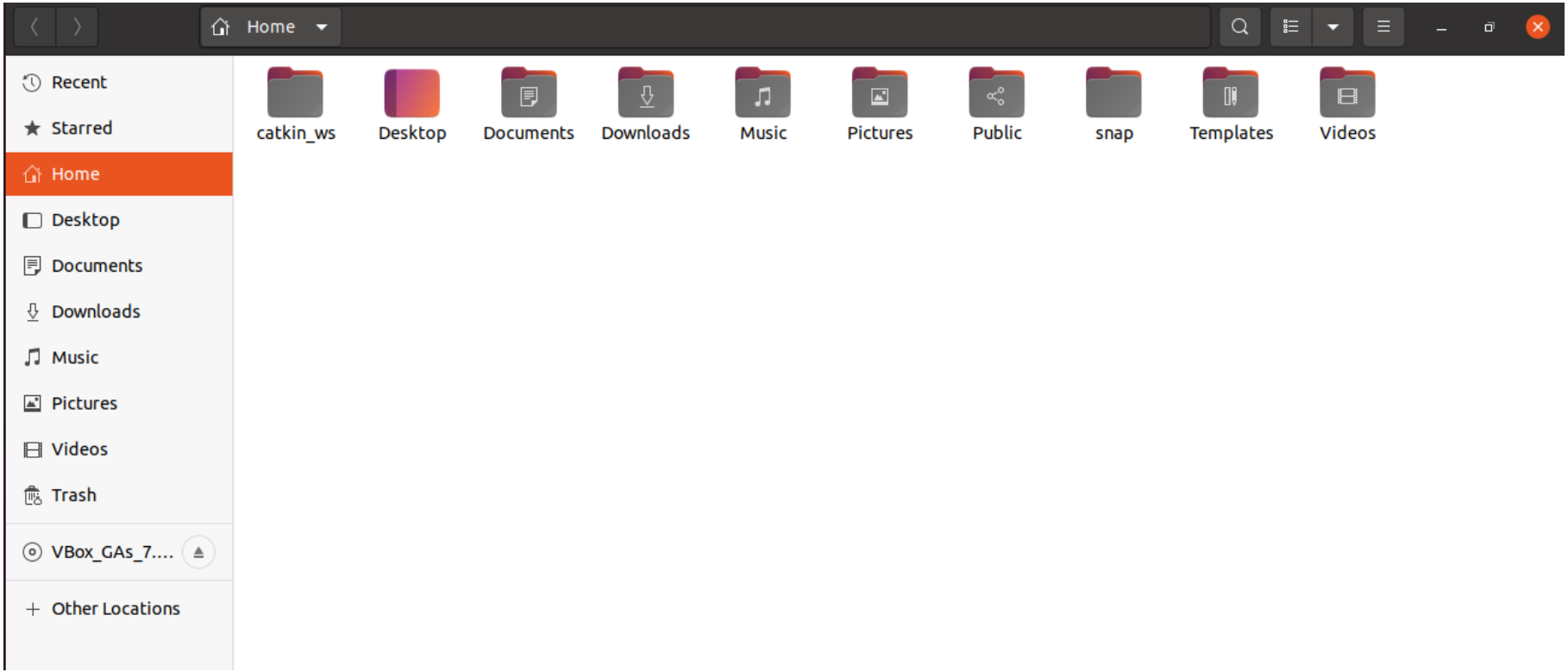
Installation of Turtlebot3

Lets create a new workspace for turtlebot tutorials named as **turtlebot_ws** and clone the necessary turtlebot packages from gitup

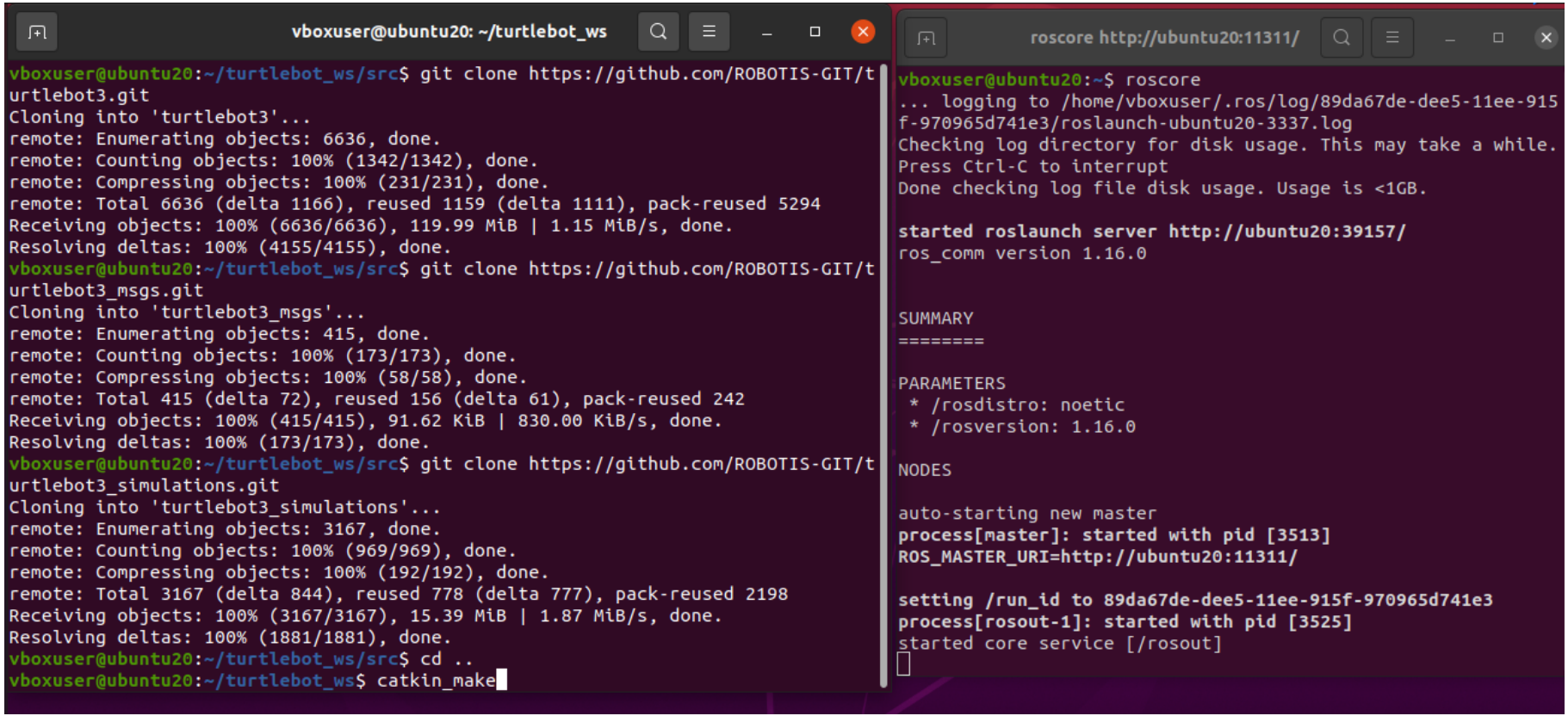
```
mkdir -p ~/turtlebot_ws/src
cd ~/turtlebot_ws/src
git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
cd ~/turtlebot_ws
catkin_make
```

source ~/turtlebot_ws/devel/setup.bash

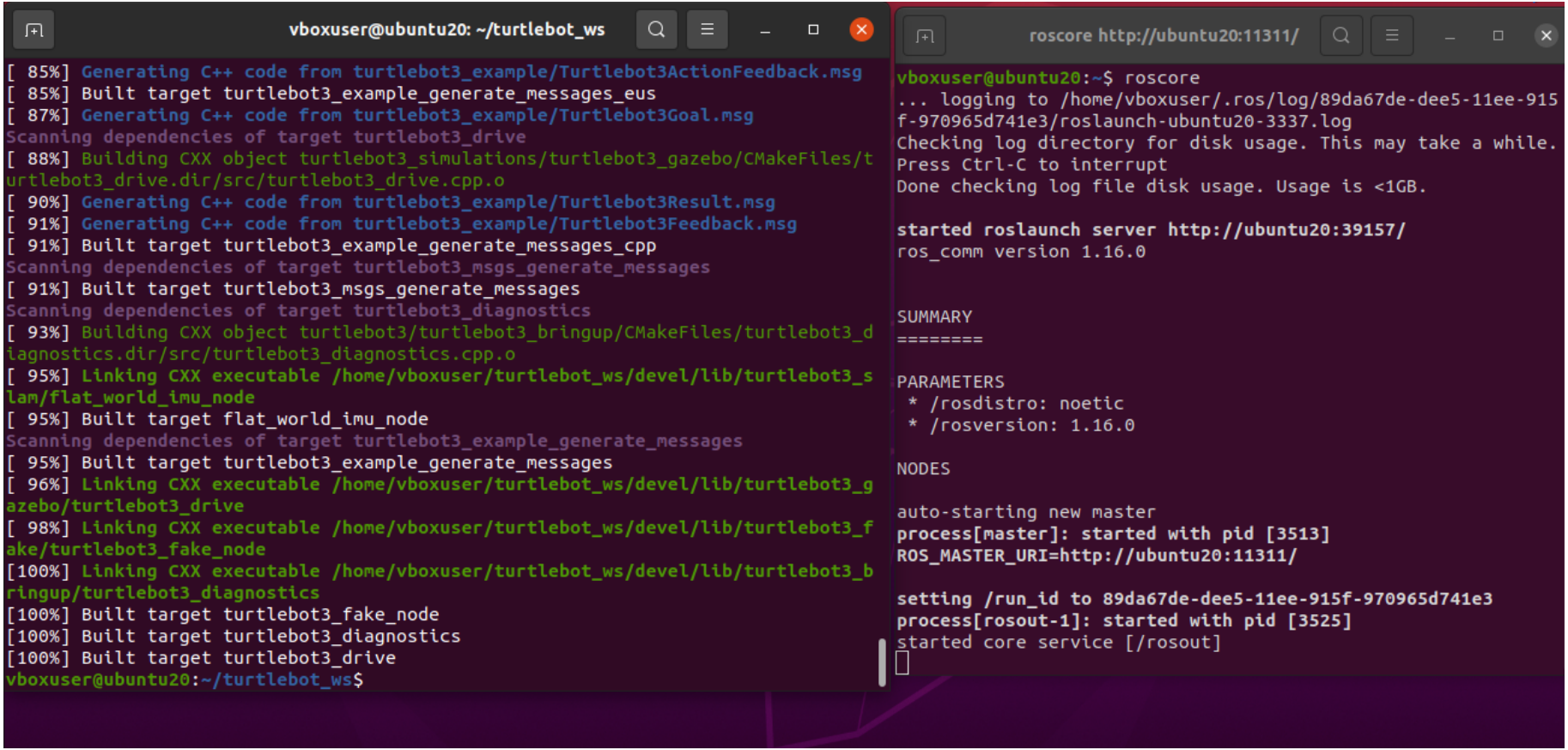
Installation of Turtlebot3



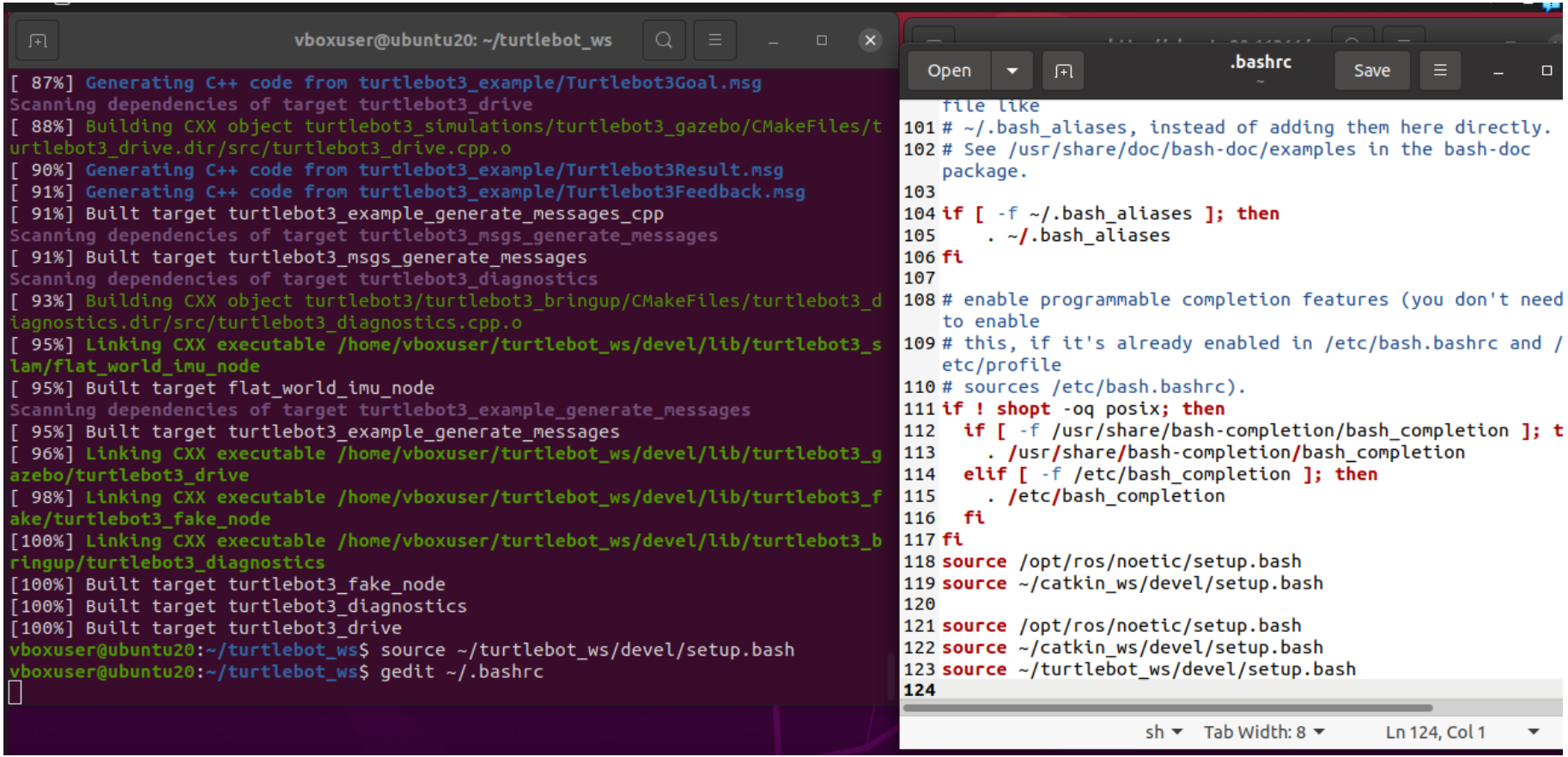
Installation of Turtlebot3



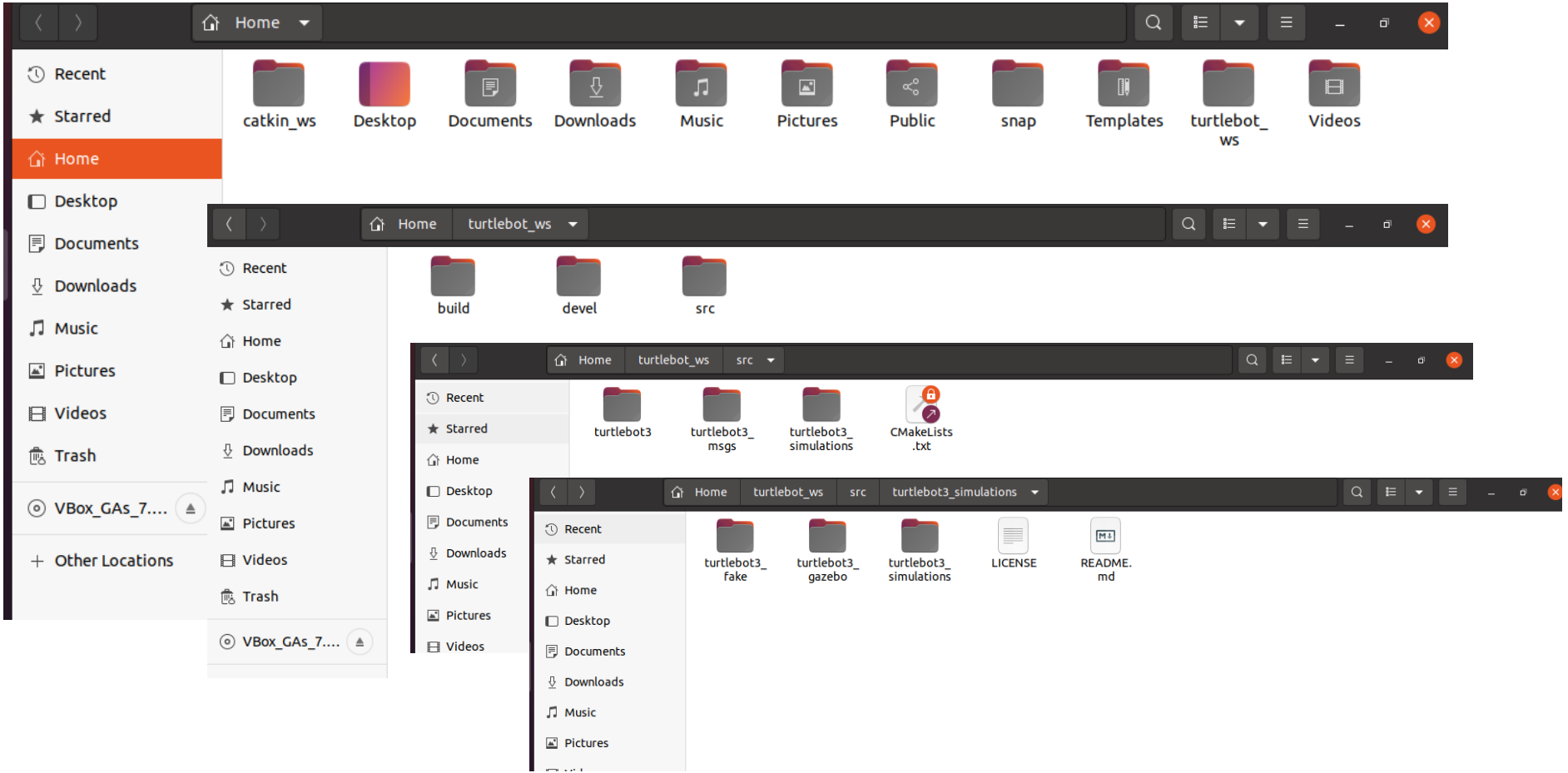
Installation of Turtlebot3



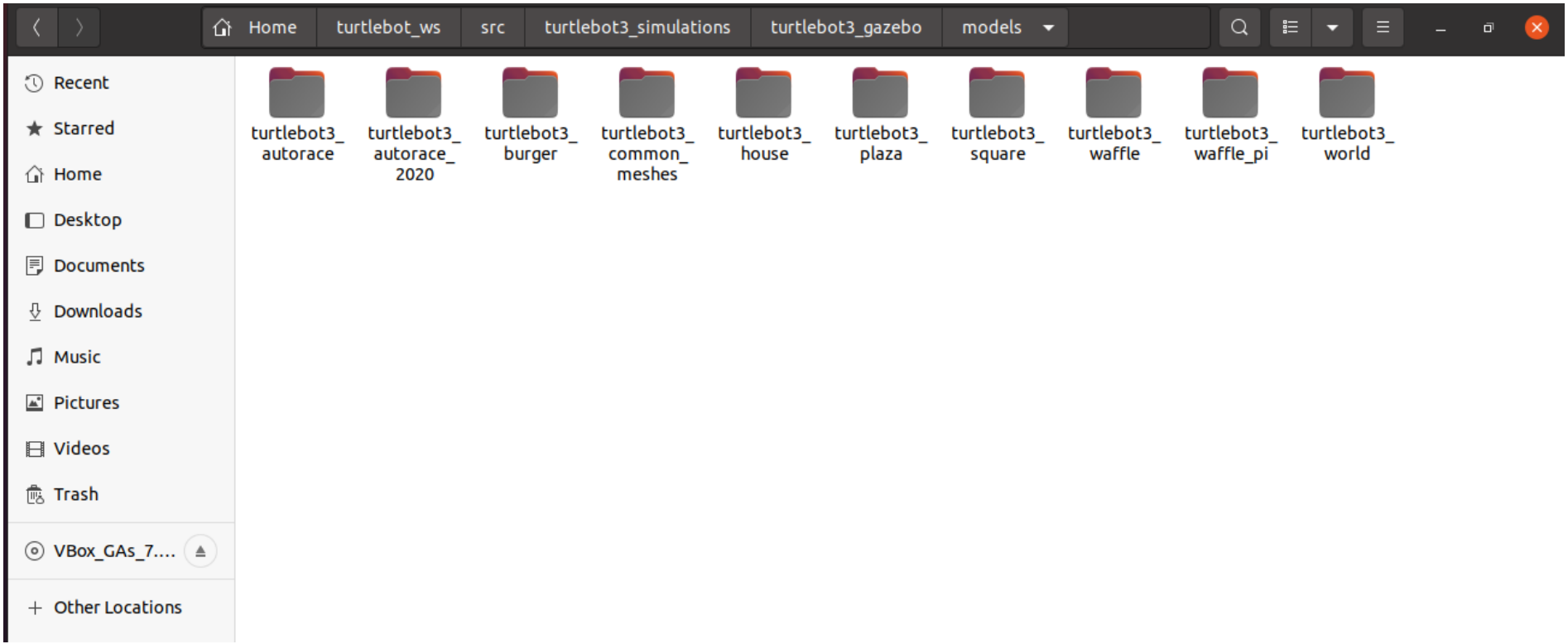
Installation of Turtlebot3



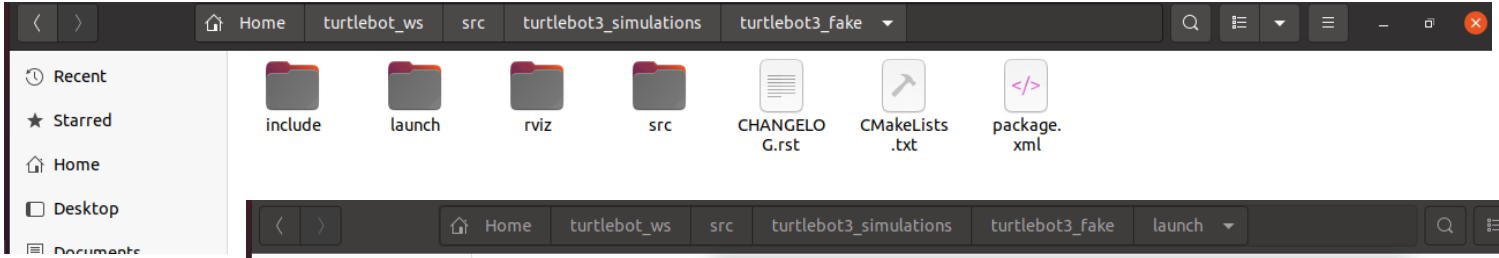
Installation of Turtlebot3

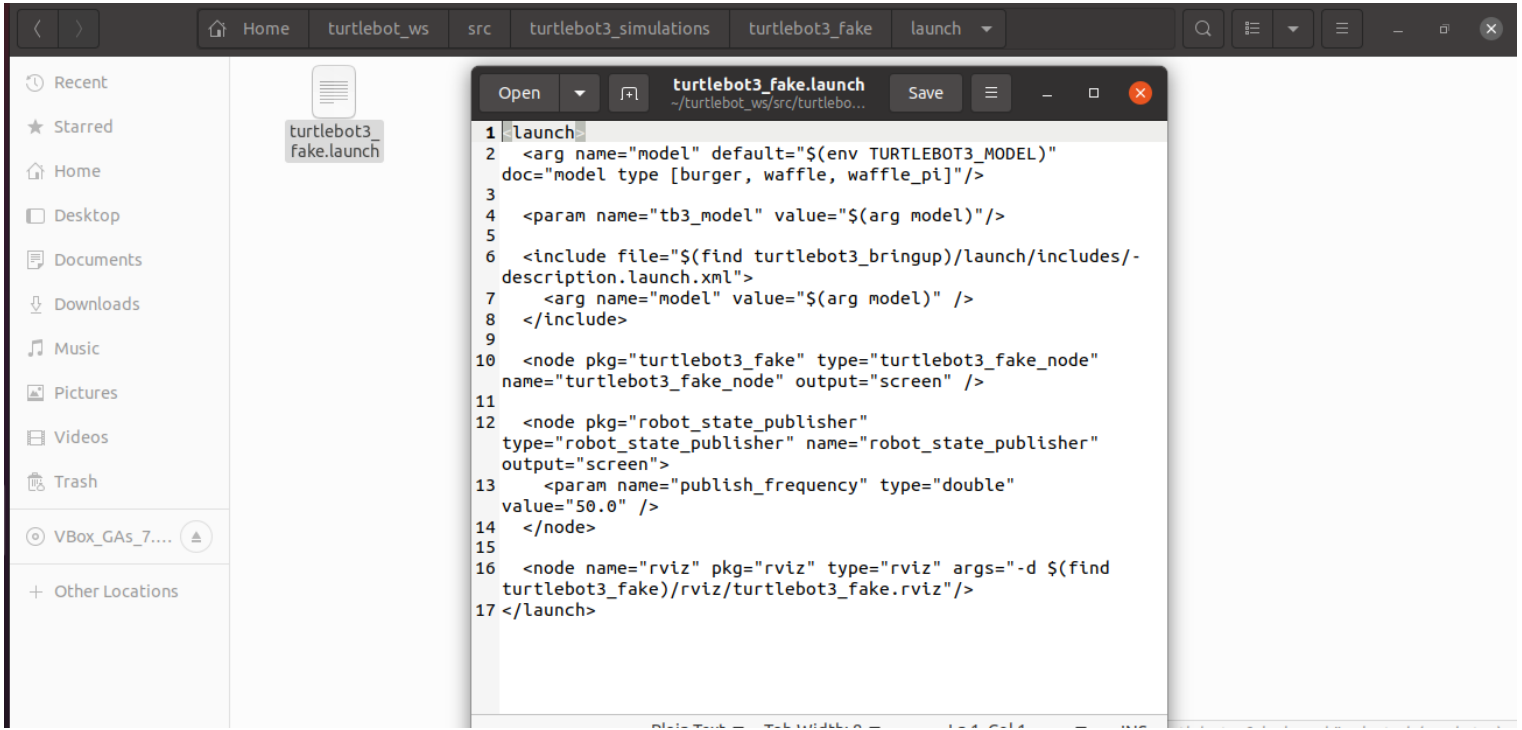


Installation of Turtlebot3



Installation of Turtlebot3





Simulate the robot motion in Rviz

https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#overview

You have 3 robot options: burger, waffle and waffle Pi

Let's select the robot by typing in the terminal
export TURTLEBOT3_MODEL=burger

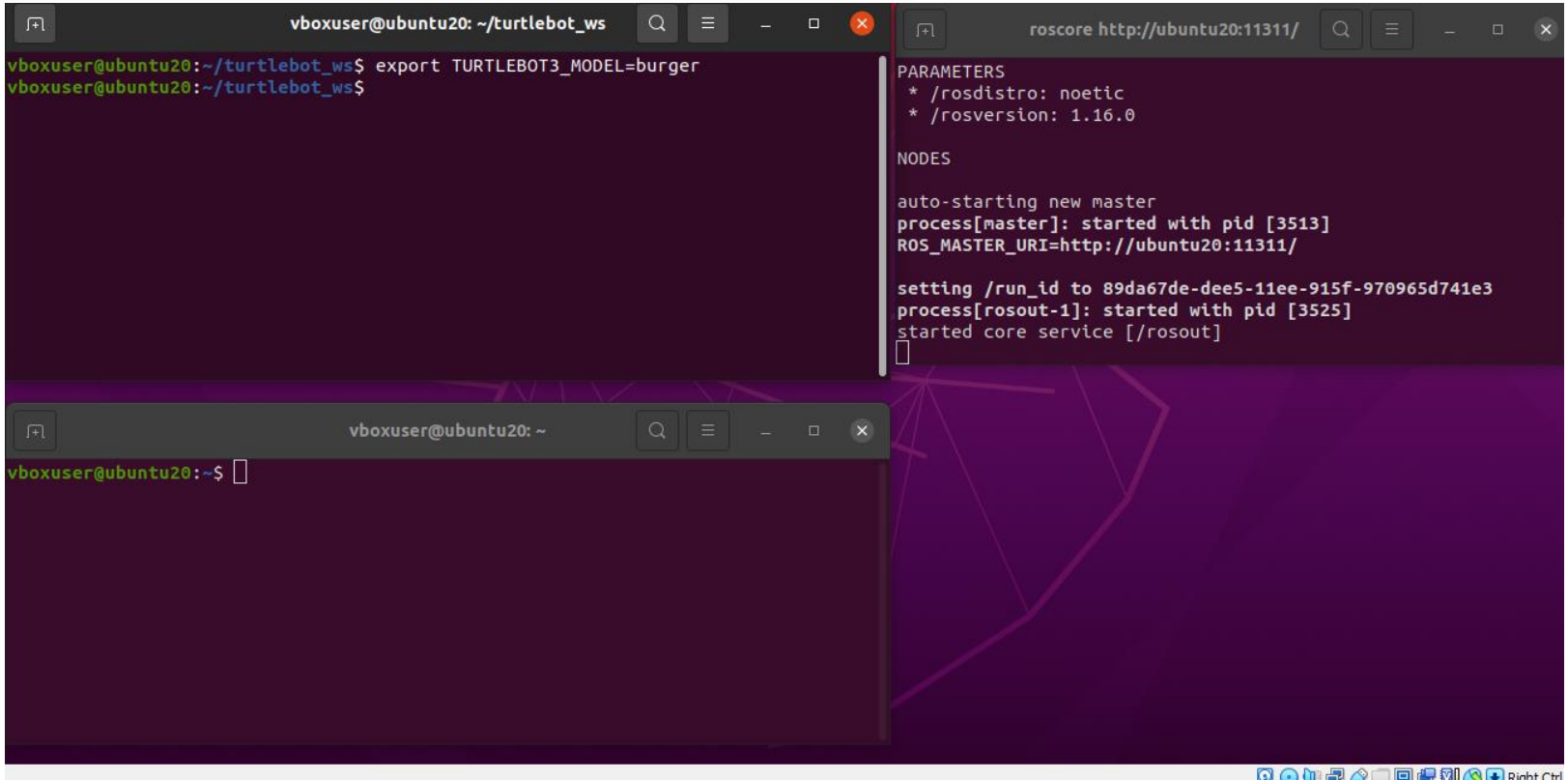
Let's bring the Rviz simulation environment
roslaunch turtlebot3_fake turtlebot3_fake.launch

Open an another terminal
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

rostopic list
rosnode list

Simulate the robot motion in Rviz

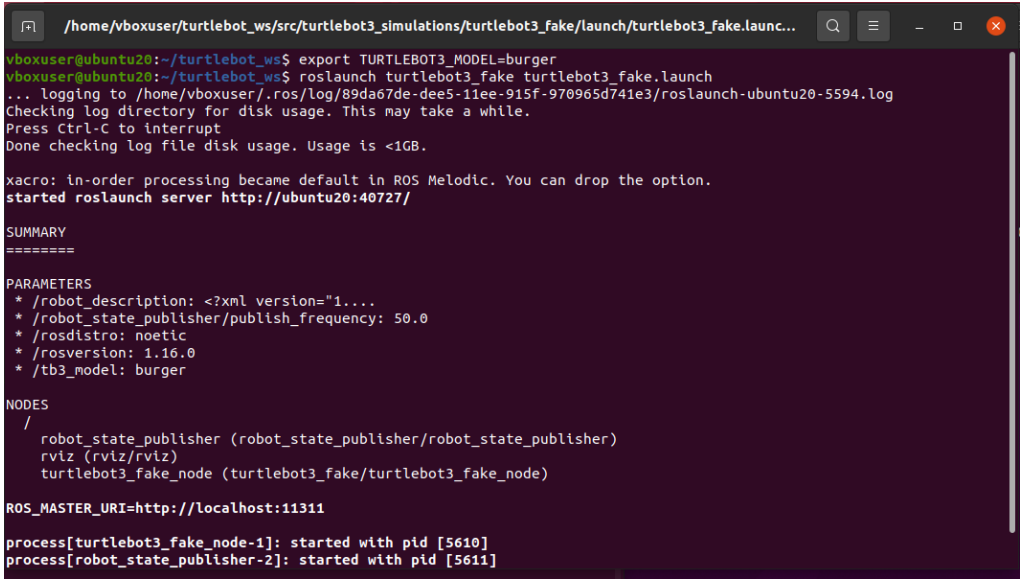
Let’s select the robot by typing in the terminal
export TURTLEBOT3_MODEL=burger

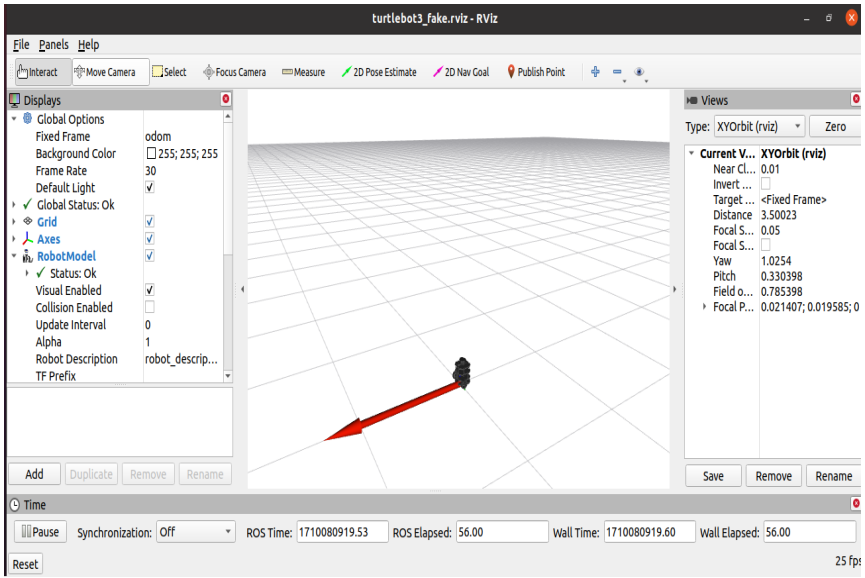


Simulate the robot motion in Rviz

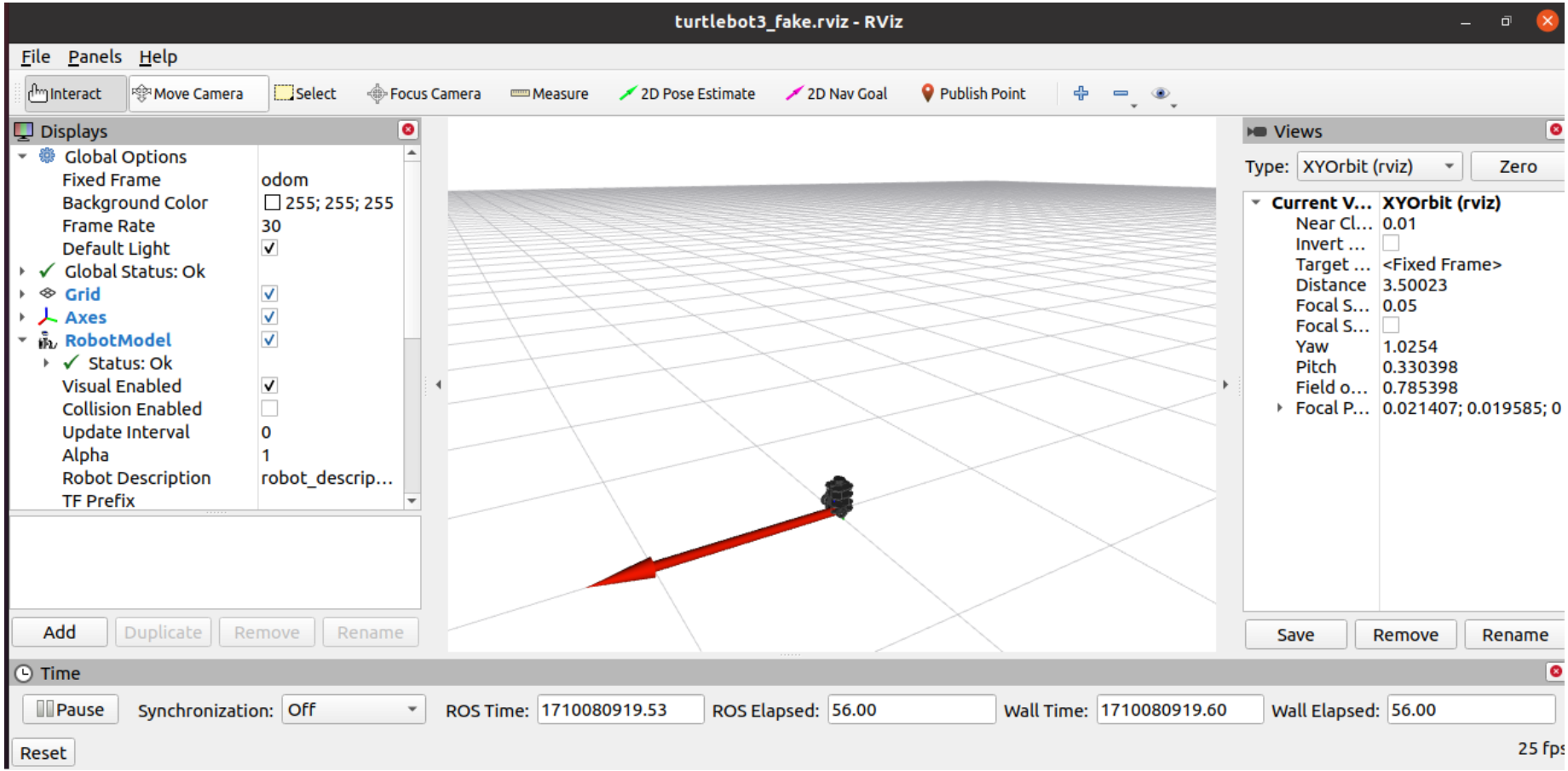
Let’s bring the Rviz simulation environment

roslaunch turtlebot3_fake turtlebot3_fake.launch

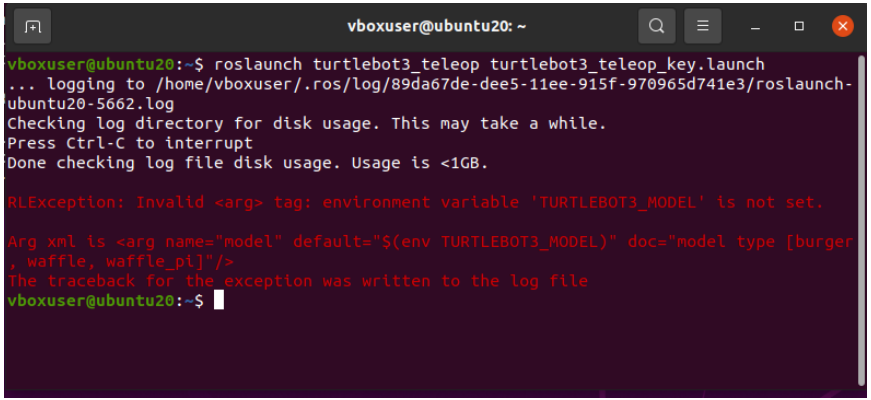


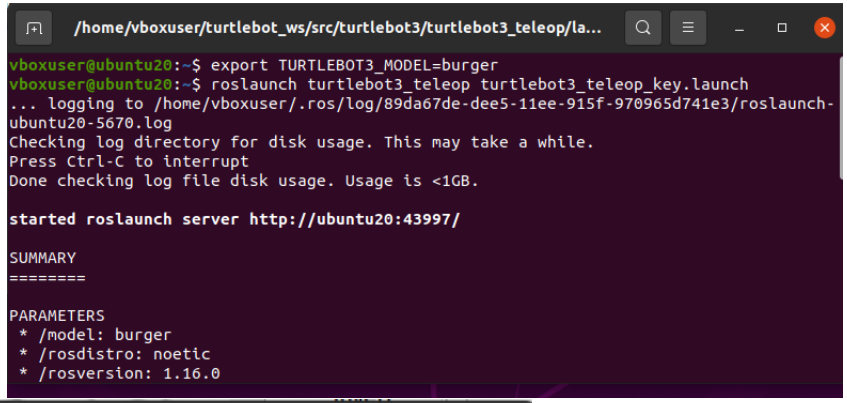


Simulate the robot motion in Rviz

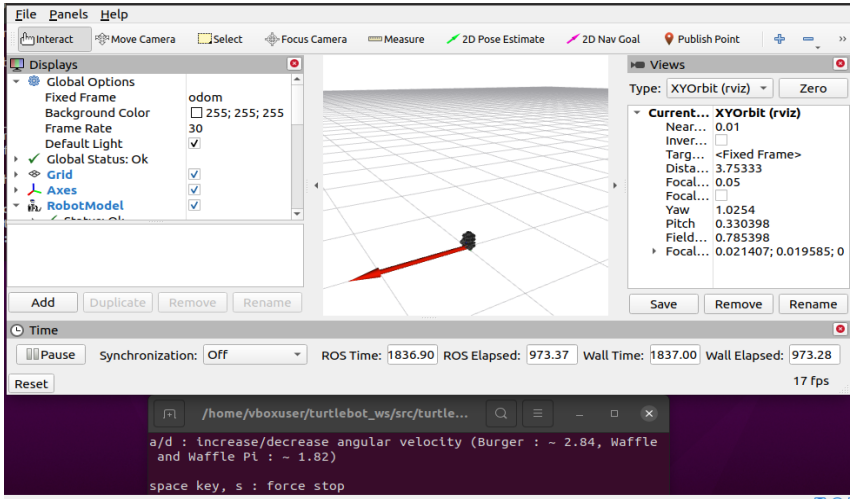


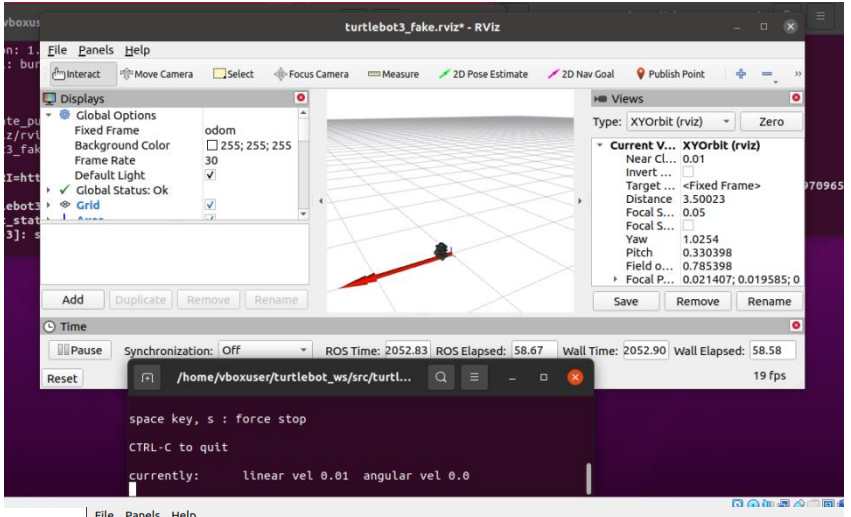
Simulate the robot motion in Rviz

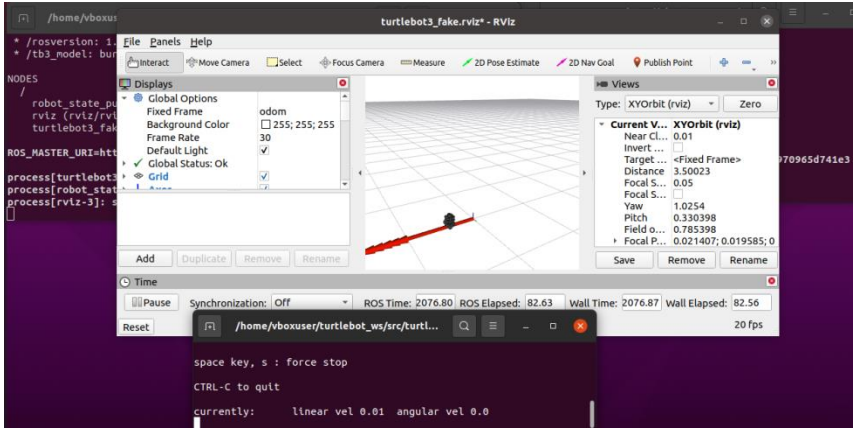


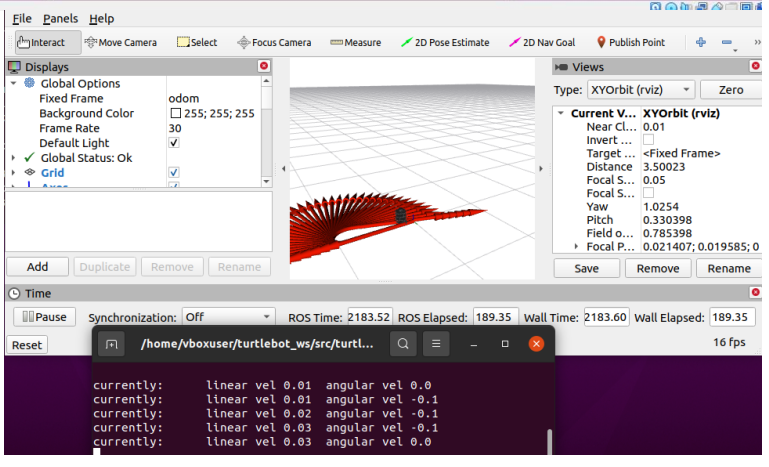


Simulate the robot motion in Rviz

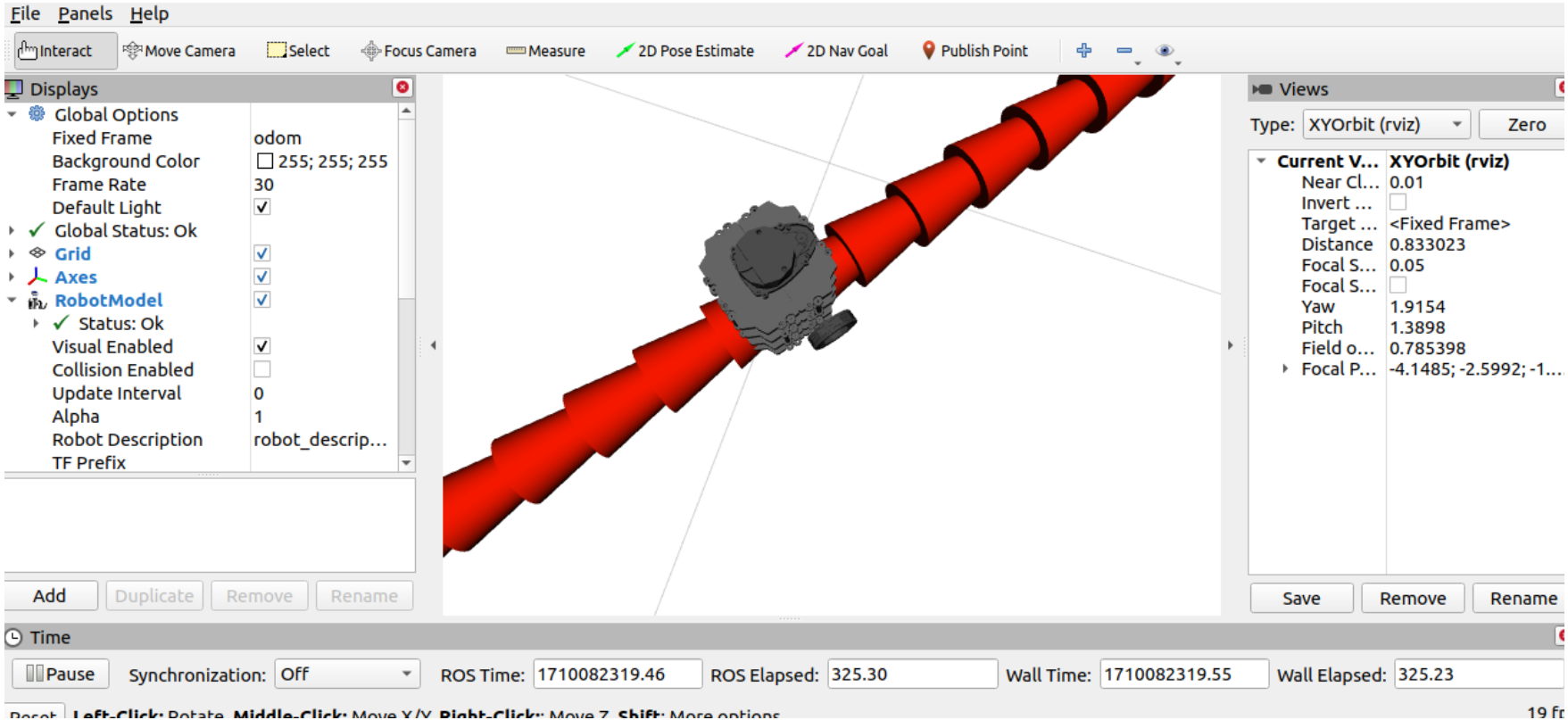




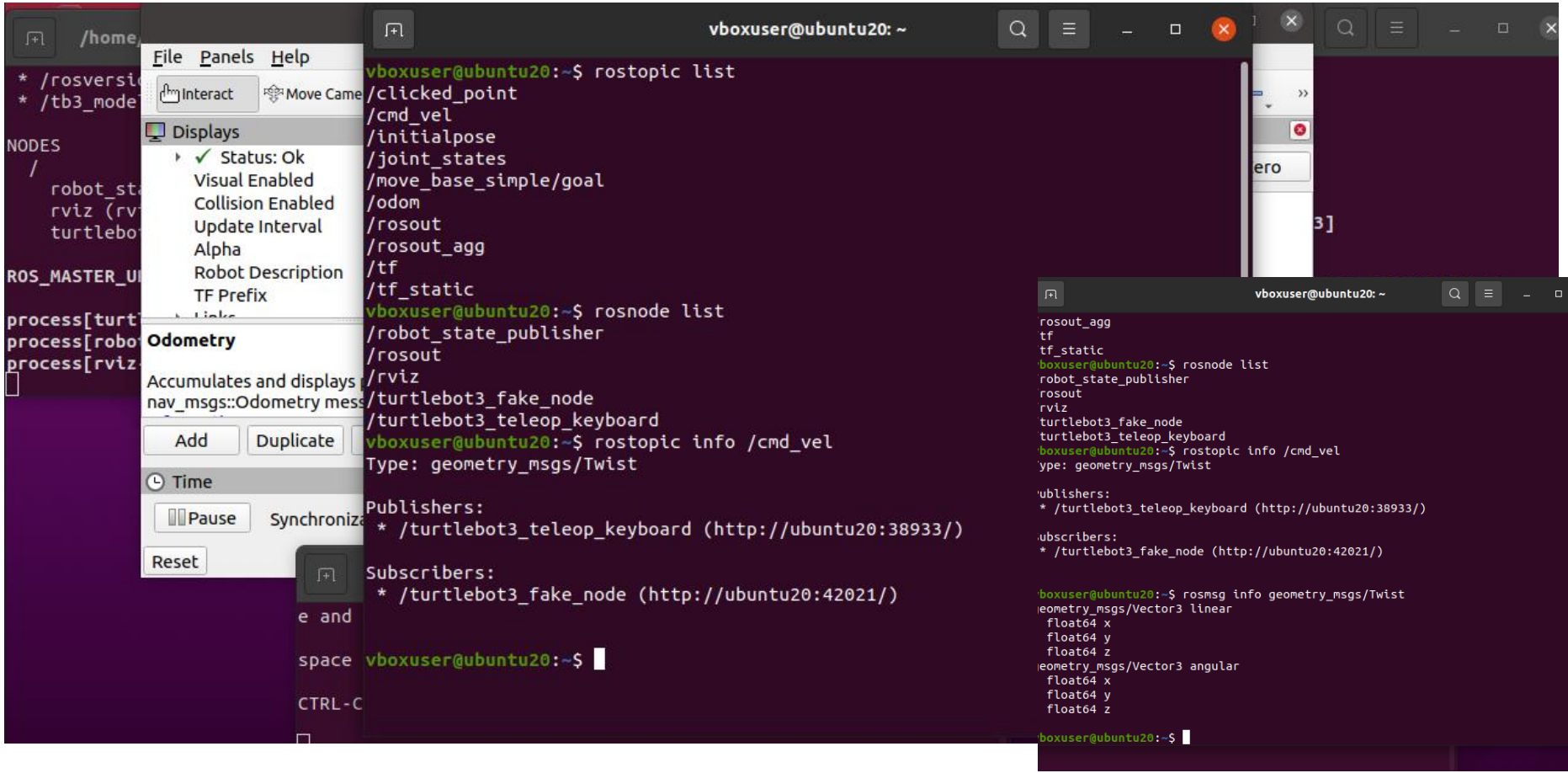




Simulate the robot motion in Rviz



Simulate the robot motion in Rviz



Simulate the robot motion in Gazebo

https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#overview

Turtlebot simulation stack is installed
sudo apt-get install ros-noetic-turtlebot-gazebo

Let's select the robot by typing in the terminal
export TURTLEBOT3_MODEL=burger

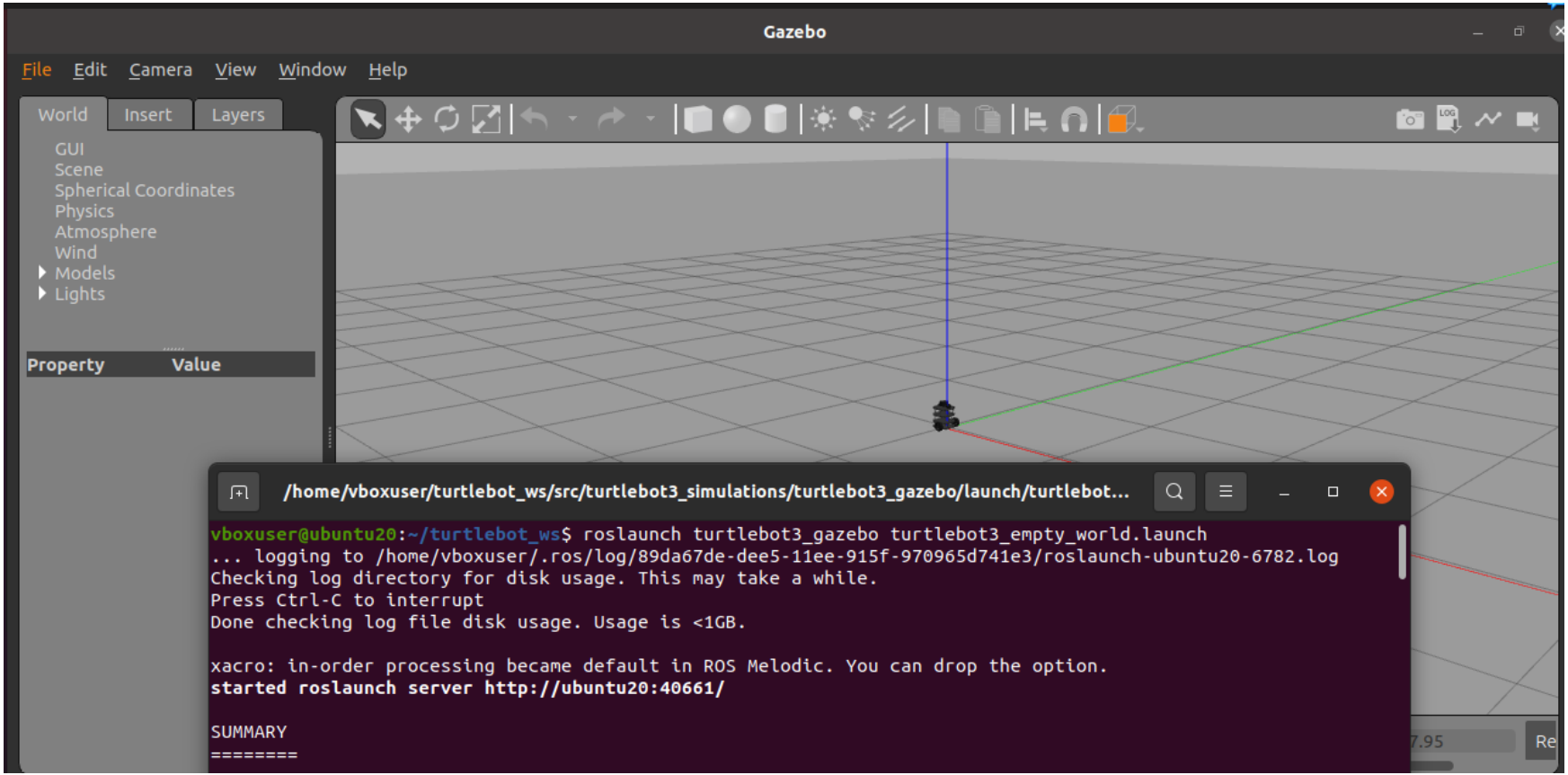
You have 3 robot options:
burger, waffle and waffle Pi

Let's bring the gazebo simulation environment
roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch

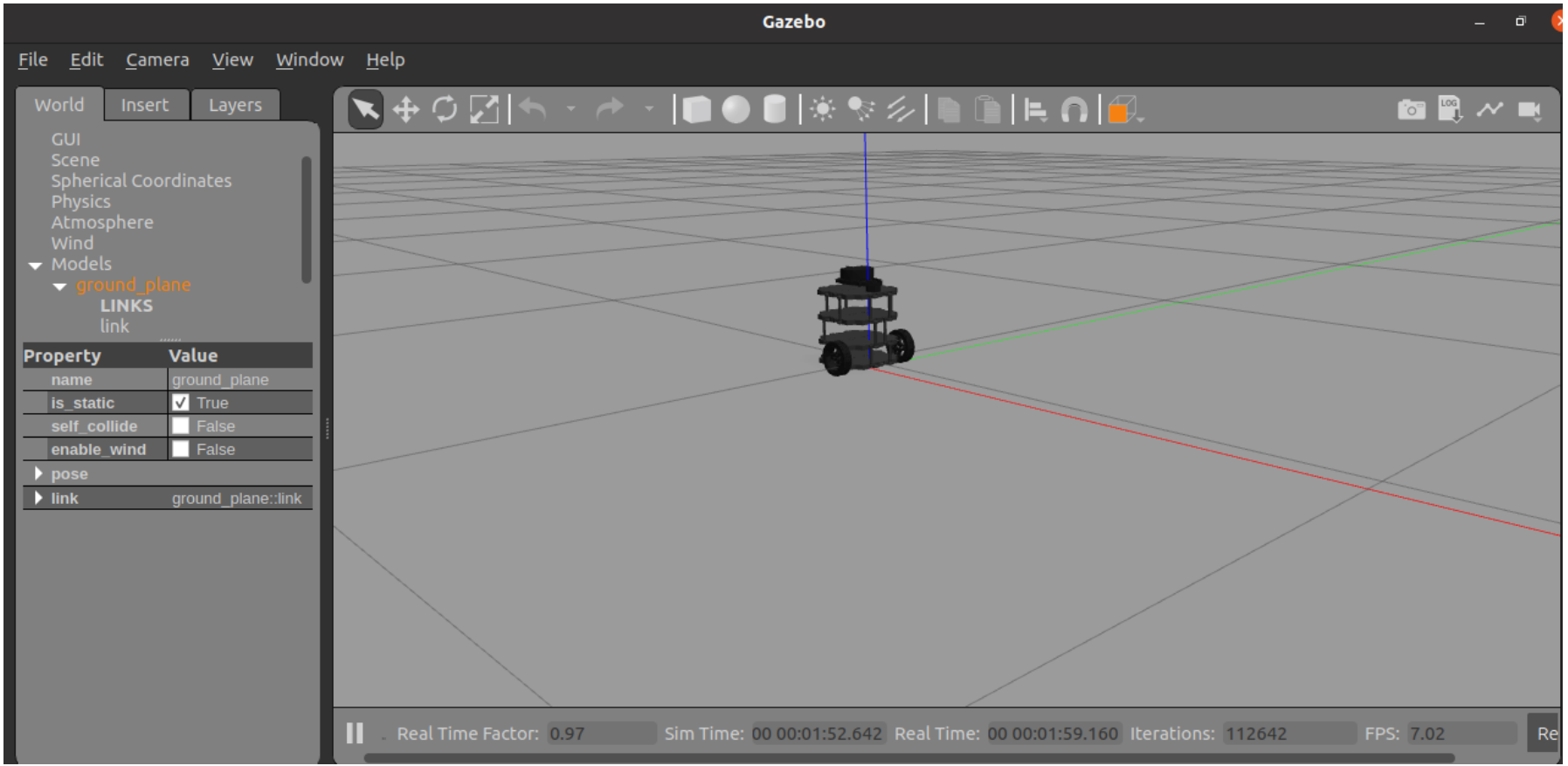
Open an another terminal
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

rostopic list
rosnode list

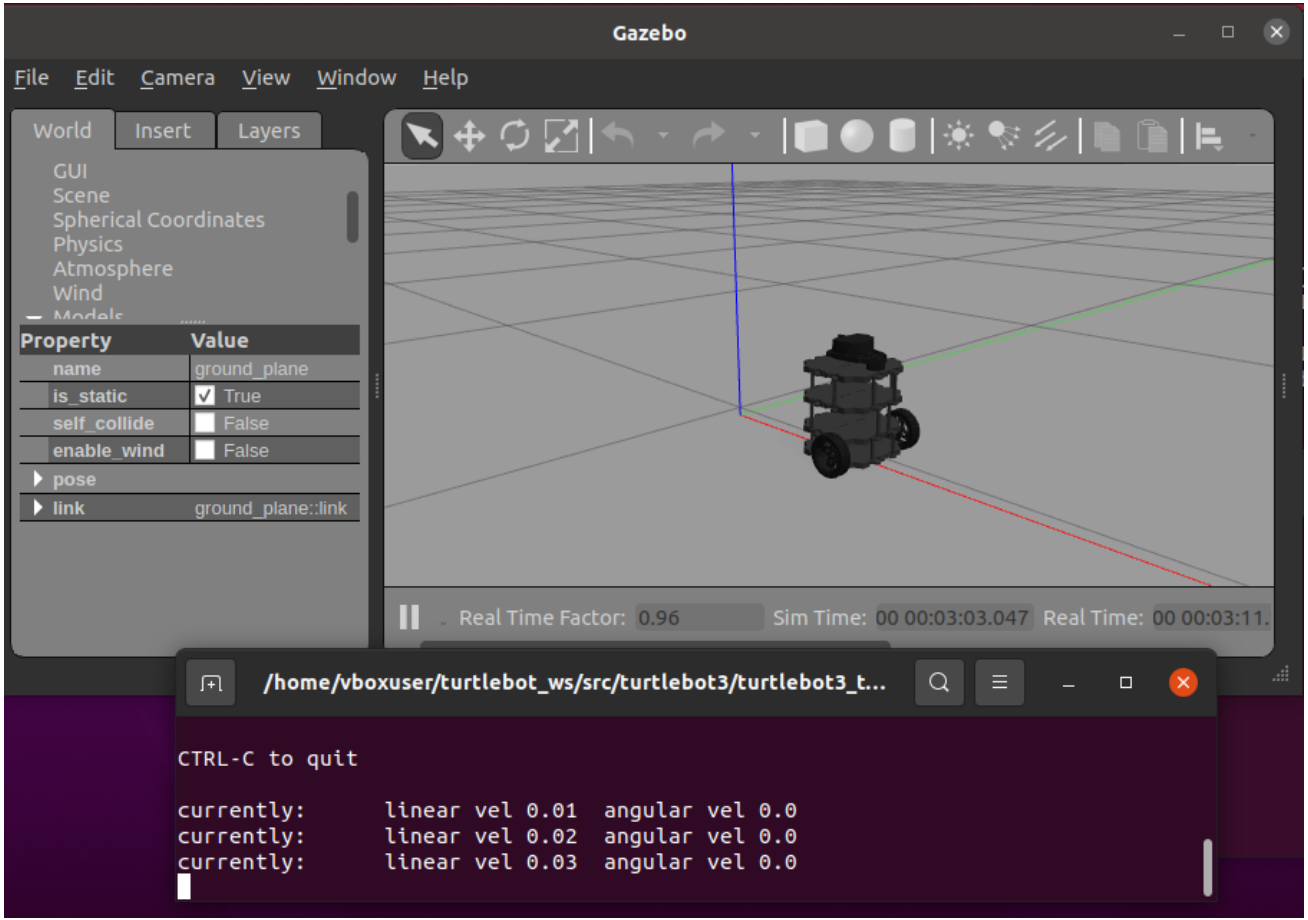
Simulate the robot motion in Gazebo



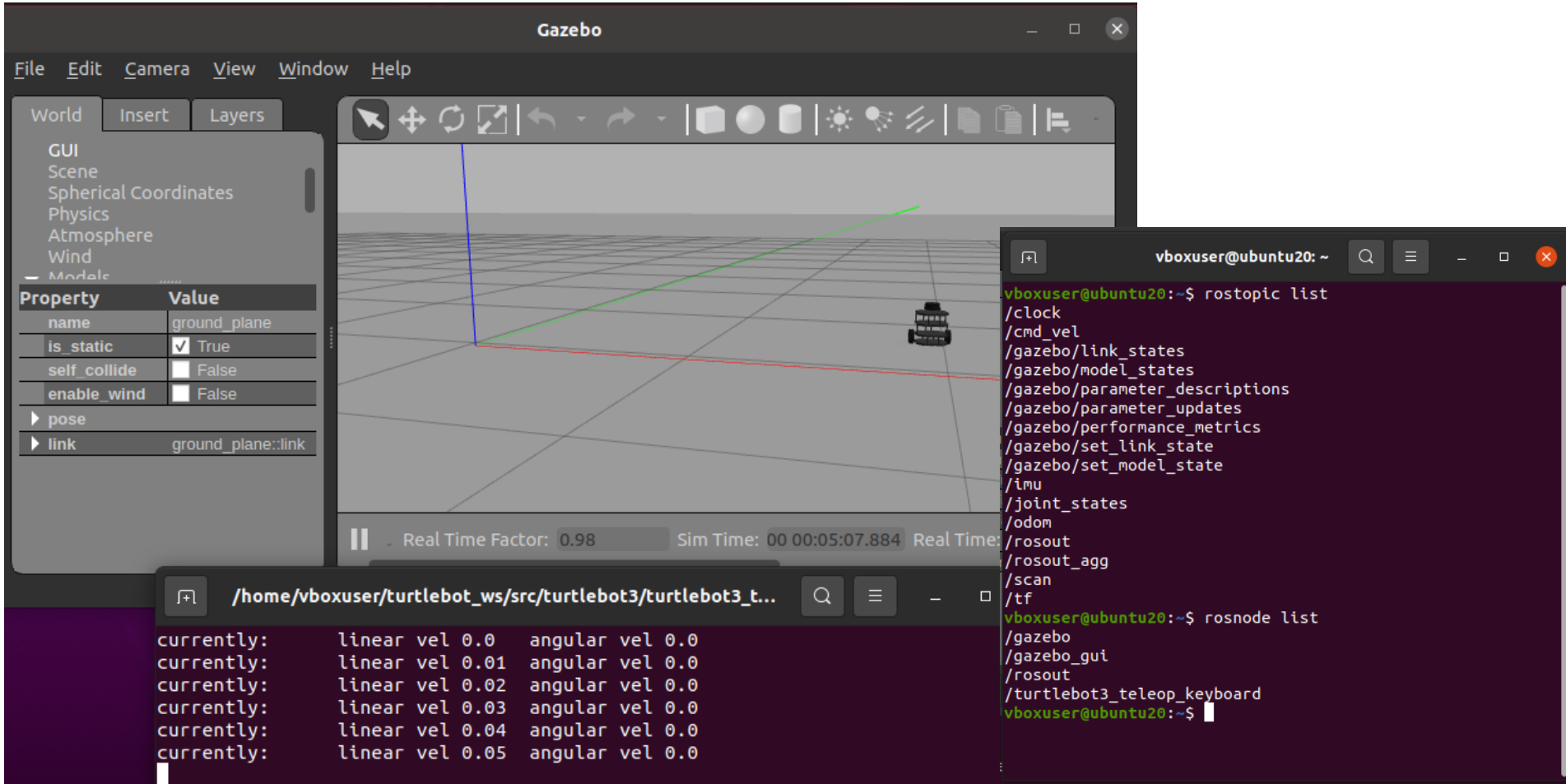
Simulate the robot motion in Gazebo



Simulate the robot motion in Gazebo

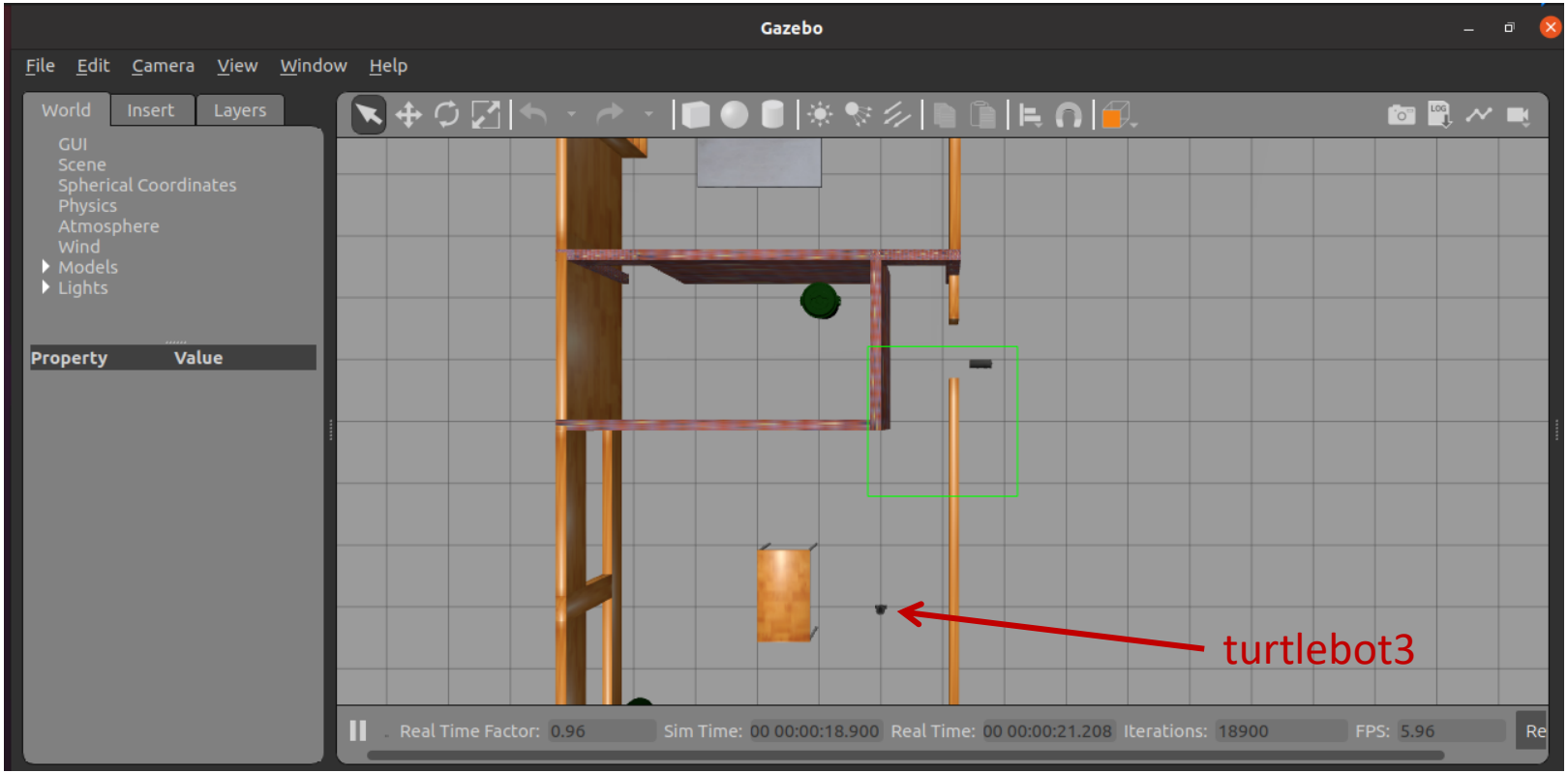


Simulate the robot motion in Gazebo

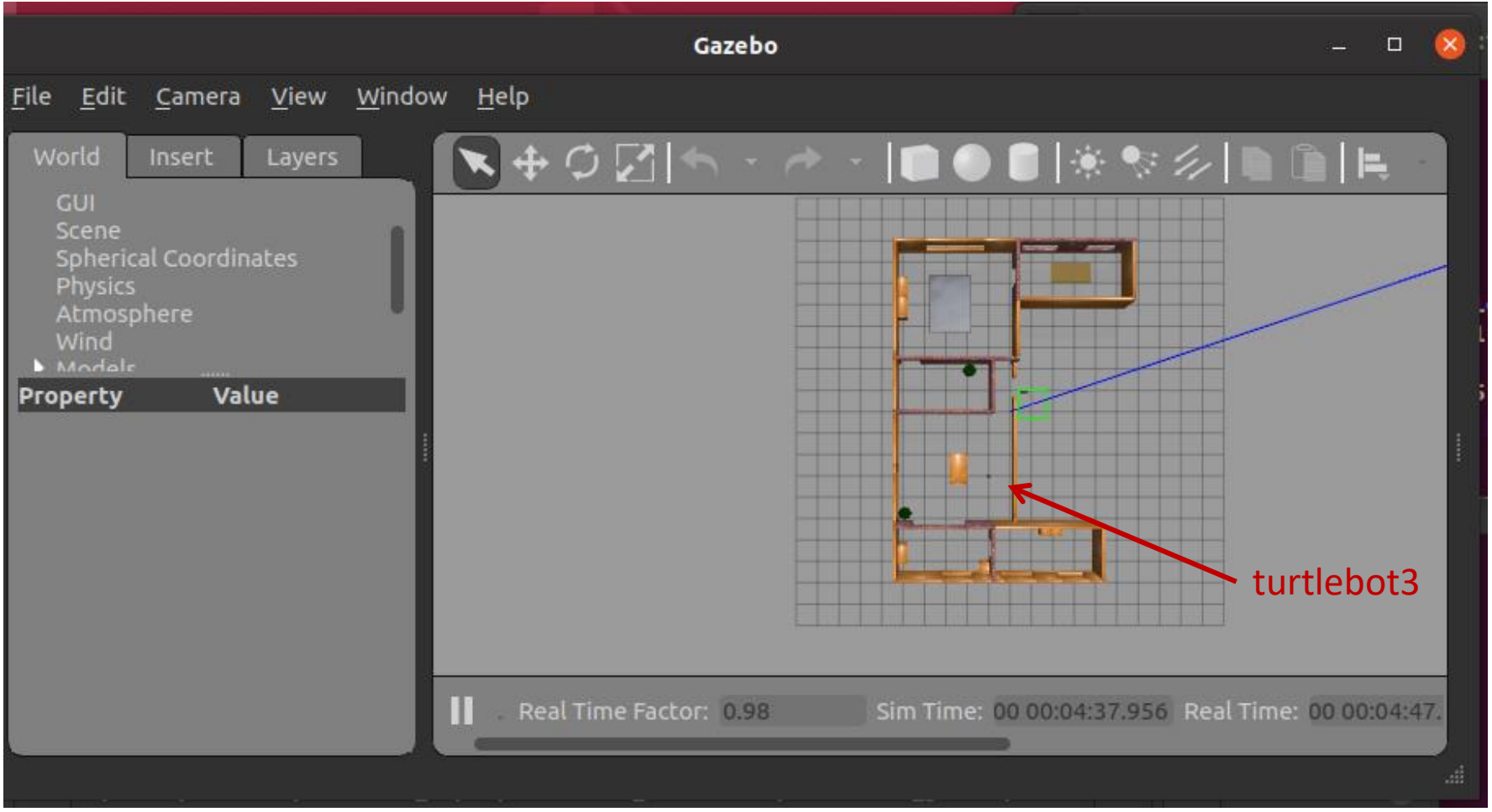


Simulate the robot motion in Gazebo

Roslaunch turtlebot3_gazebo turtlebot3_house.launch



Simulate the robot motion in Gazebo



Simulate the robot motion in Gazebo