

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)	✓	⚡	✗	✗

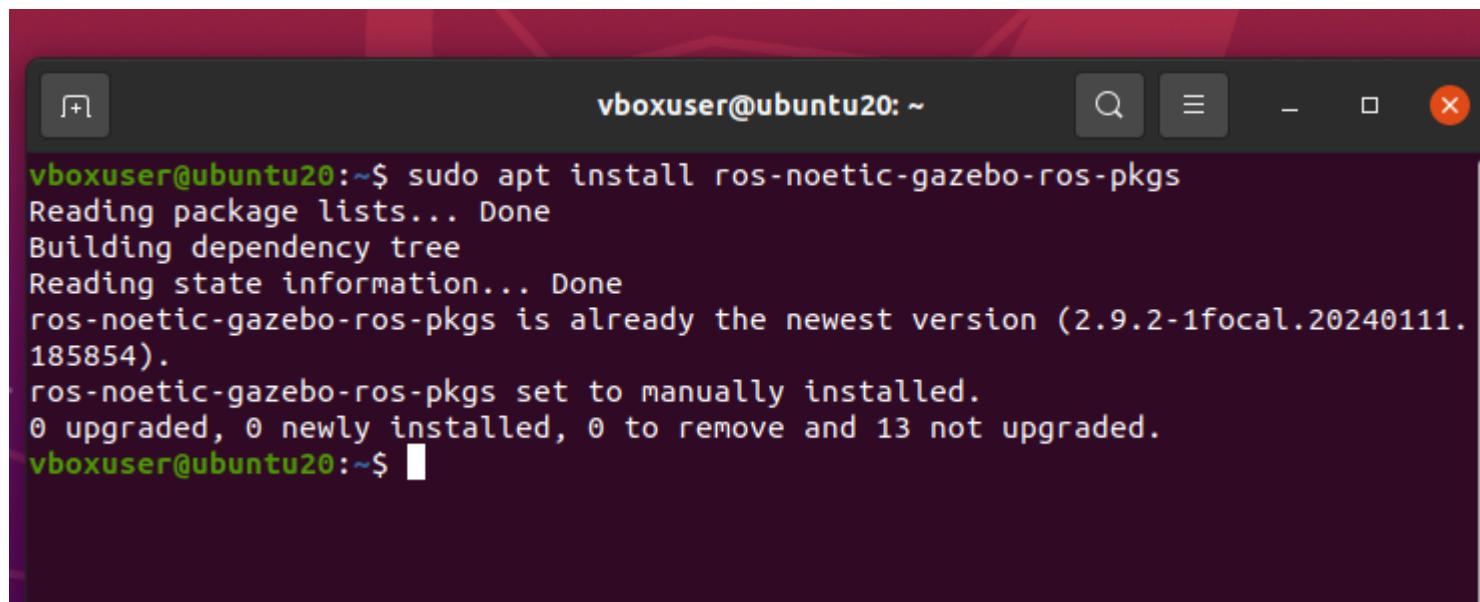
- - Recommended combination
- - Incompatible / not possible.
- - Possible, but use with caution. These combinations of ROS and Gazebo can be made to work together, but some effort is required.

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

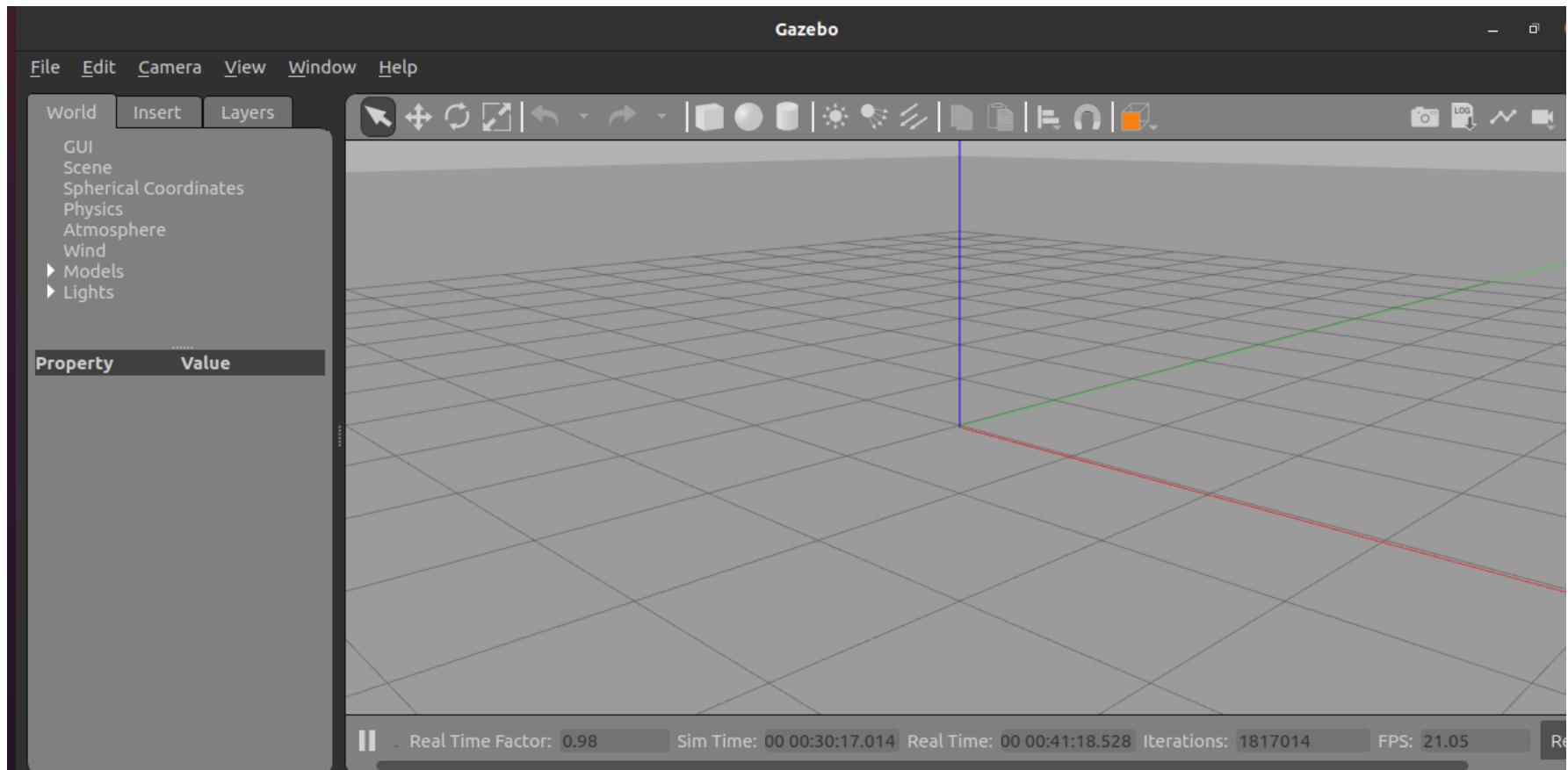


```
vboxuser@ubuntu20:~$ sudo apt install ros-noetic-gazebo-ros-pkgs
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-gazebo-ros-pkgs is already the newest version (2.9.2-1focal.20240111.185854).
ros-noetic-gazebo-ros-pkgs set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
vboxuser@ubuntu20:~$
```

Checking Gazebo



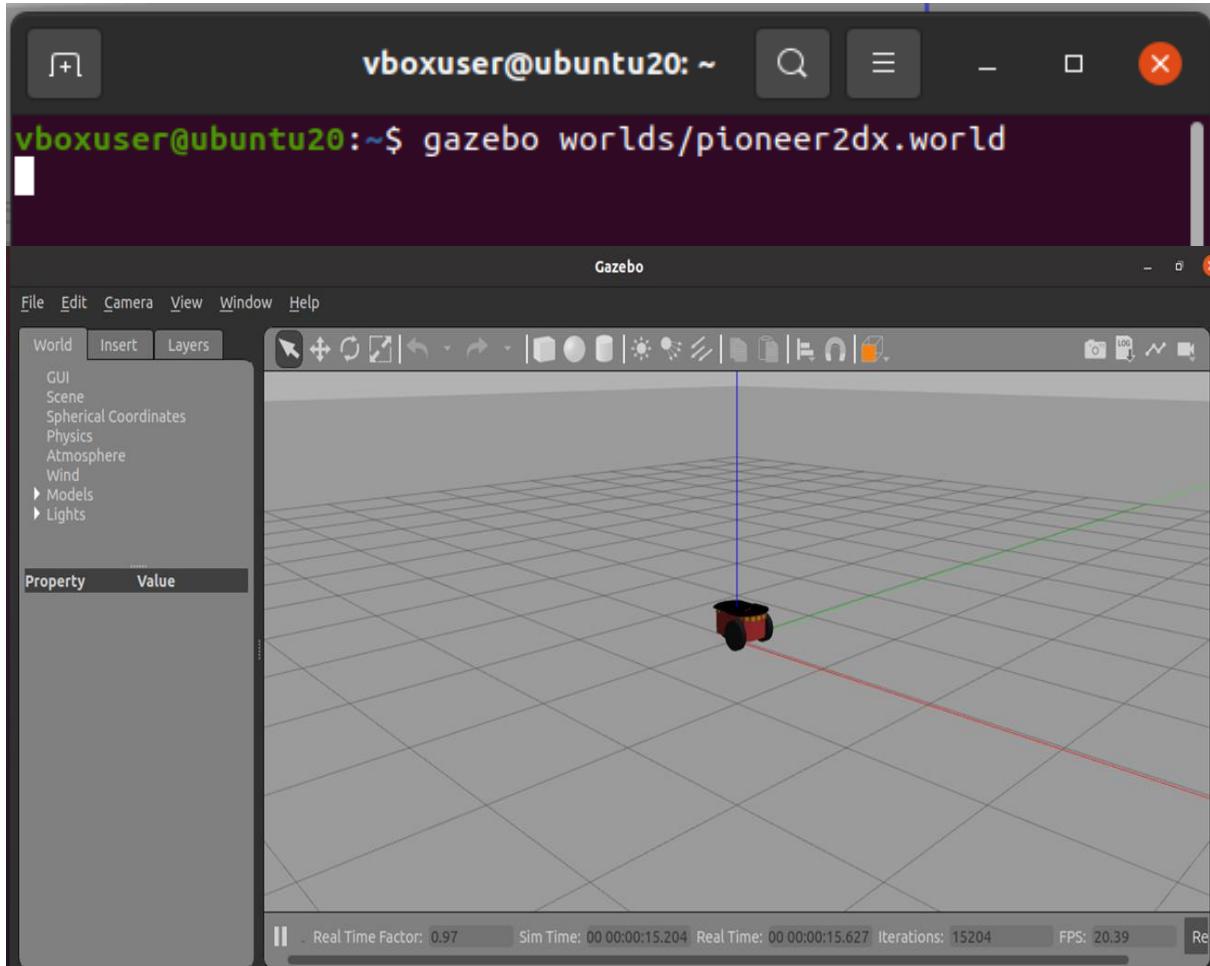
Checking Gazebo



Run Gazebo with a robot

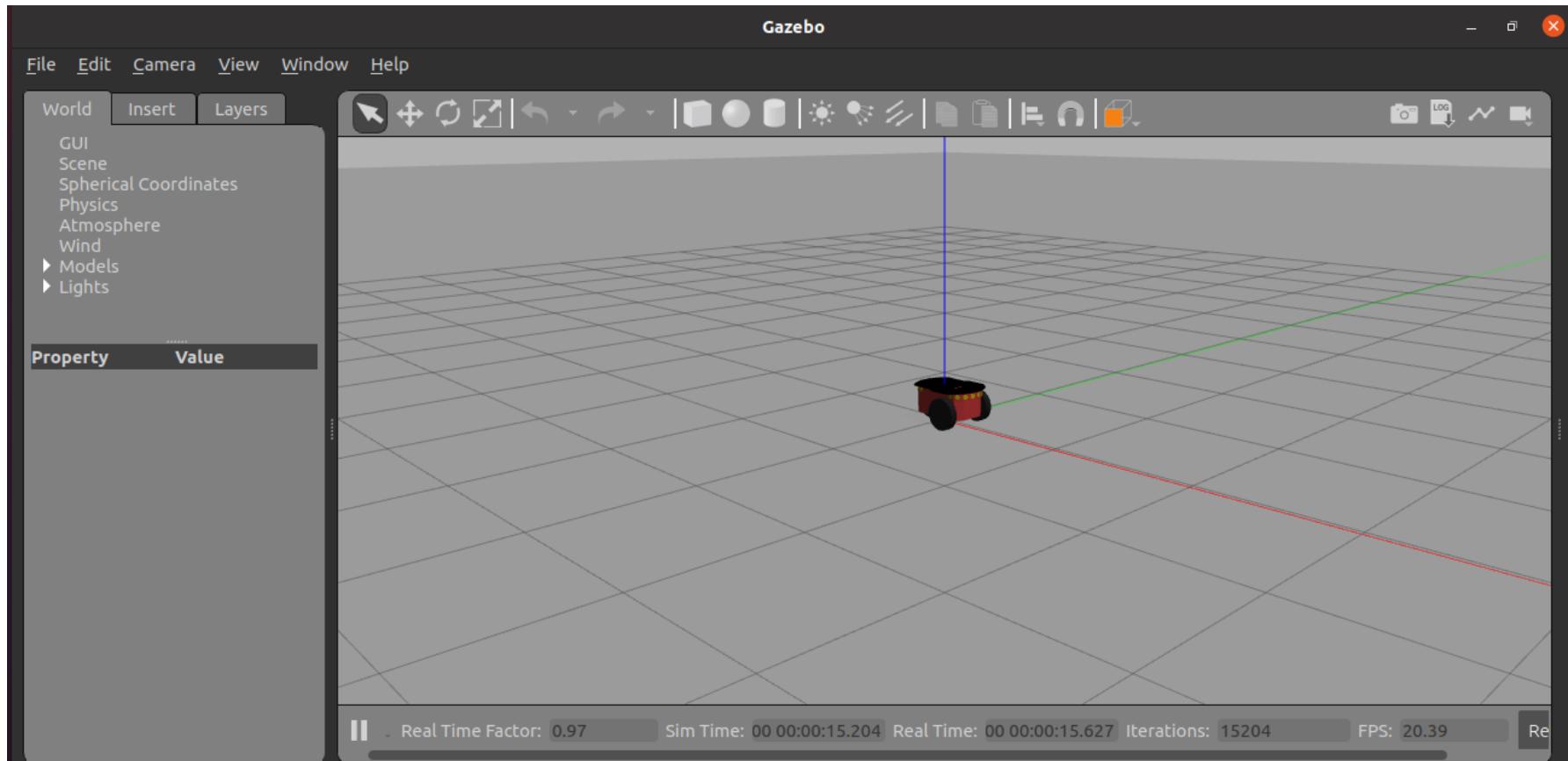
```
vboxuser@ubuntu20:~$ ls /usr/share/gazebo-11/worlds
actor_bvh.world          fuel_models.world
actor.world               gravity_compensation.world
animated_box.world        gripper.world
animation_tension.world   harness.world
attach_lights.world       heightmap_demo.world
blank.world               heightmap.world
blink_visual.world        hydra_demo.world
cafe.world                imu_demo.world
camera_intrinsics.world  initial_velocity.world
camera.world              init_joint_control.world
cart_demo.world           iris_arducopter_demo.world
cessna_demo.world         joint_damping_demo.world
collision_zero.world     joint_friction_demo.world
contact.world             joints.world
contain_plugin_demo.world joy_demo.world
contain_plugin_moving_demo.world keys_to_cmd_vel.world
deform_visual.world       led_plugin_demo.world
depth_camera2.world      lensflare_plugin.world
depth_camera.world        lift_drag_demo.world
elevator.world            lights.world
empty_1_0.world           linear_battery_demo.world
empty_bullet.world        logical_camera.world
empty_sky.world           lookat_demo.world
empty.world               magnetometer.world
everything.world          mass_on_rails.world
fiducial.world            misalignment_plugin_demo.world
flash_light_plugin_demo.world mud_bitmask.world
flocking.world            mud.world
force_torque_demo.world  multilink_shape.world
friction_demo.world       nested_model.world
friction_pyramid.world   nested_multilink_shape.world
openal.world               osrf_elevator.world
ortho.world               pioneer2dx_camera.world
pioneer2dx_laser_camera.world pioneer2dx_laser.world
pioneer2dx.world          pioneer2dx.world
plane_demo.world          plane_propeller_demo.world
plot3d.world              polyline.world
plugin.world              point_light_shadows_demo.world
population.world          polyline.world
pr2.world                 population.world
presentation.world        pr2.world
pressure_sensor.world    presentation.world
profiler.world            pressure_sensor.world
projector.world           profiler.world
random_velocity.world    projector.world
ray_cpu.world             random_velocity.world
ray_noise_plugin.world   ray_cpu.world
reflectance.world         ray_noise_plugin.world
roadTextures.world        ray_noisy_plugin.world
road.world                reflectance.world
robocup09_spl_field.world road.world
robocup14_spl_field.world roadTextures.world
robocup_3Dsim.world       rubble.world
robocup_3Dsim.world       seesaw.world
shapes_bitmask.world      rubble.world
shapes_layers.world       seesaw.world
shapes_shininess.world   shapes_bitmask.world
shapes.world              zephyr_demo.world
sim_events.world          shapes_layers.world
simple_arm_teleop.world  shapes_shininess.world
simple_arm.world          shapes.world
simple_gripper.world     sim_events.world
single_rotor_demo.world  simple_arm_te
sonar_demo.world          simple_gripper.world
sphere_atlas_demo.world  single_rotor_demo.world
spotlight_shadows_demo.world sonar_demo.world
ssao_plugin.world          sphere_atlas_demo.world
stacks.world              spotlight_shadows_demo.world
static_map_plugin.world  ssao_plugin.world
timer_gui.world           stacks.world
torsional_friction_demo.world static_map_plugin.world
touch_plugin_demo.world  timer_gui.world
tracked_vehicle_simple.world torsional_friction_demo.world
tracked_vehicle_wheeled.world touch_plugin_demo.world
transporter.world         tracked_vehicle_simple.world
trigger.world             tracked_vehicle_wheeled.world
twin_rotor_demo.world     transporter.world
underwater.world          trigger.world
variable_gearbox_plugin.world twin_rotor_demo.world
willowgarage.world         underwater.world
wind_demo.world           variable_gearbox_plugin.world
wireless_sensors.world    willowgarage.world
zephyr_demo.world          wind_demo.world
vboxuser@ubuntu20:~$
```

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:

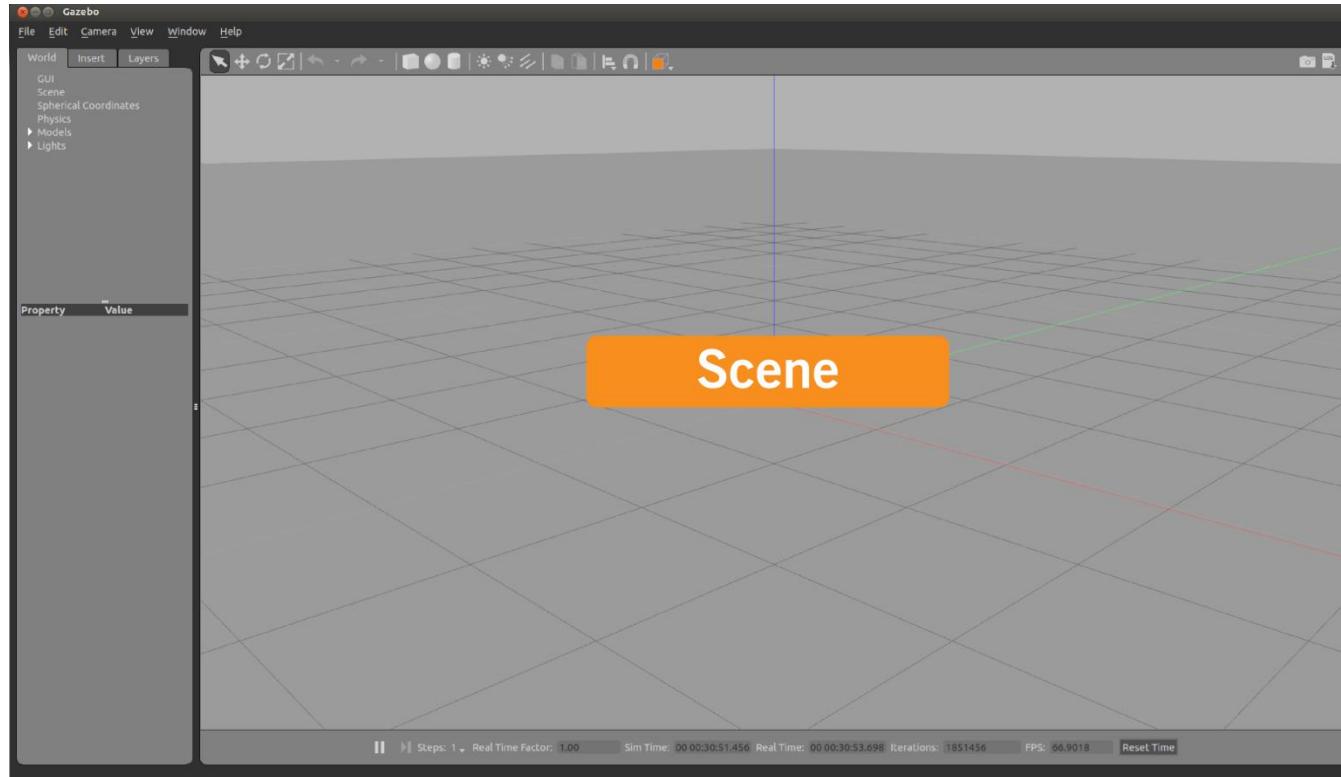
`gzclient`

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

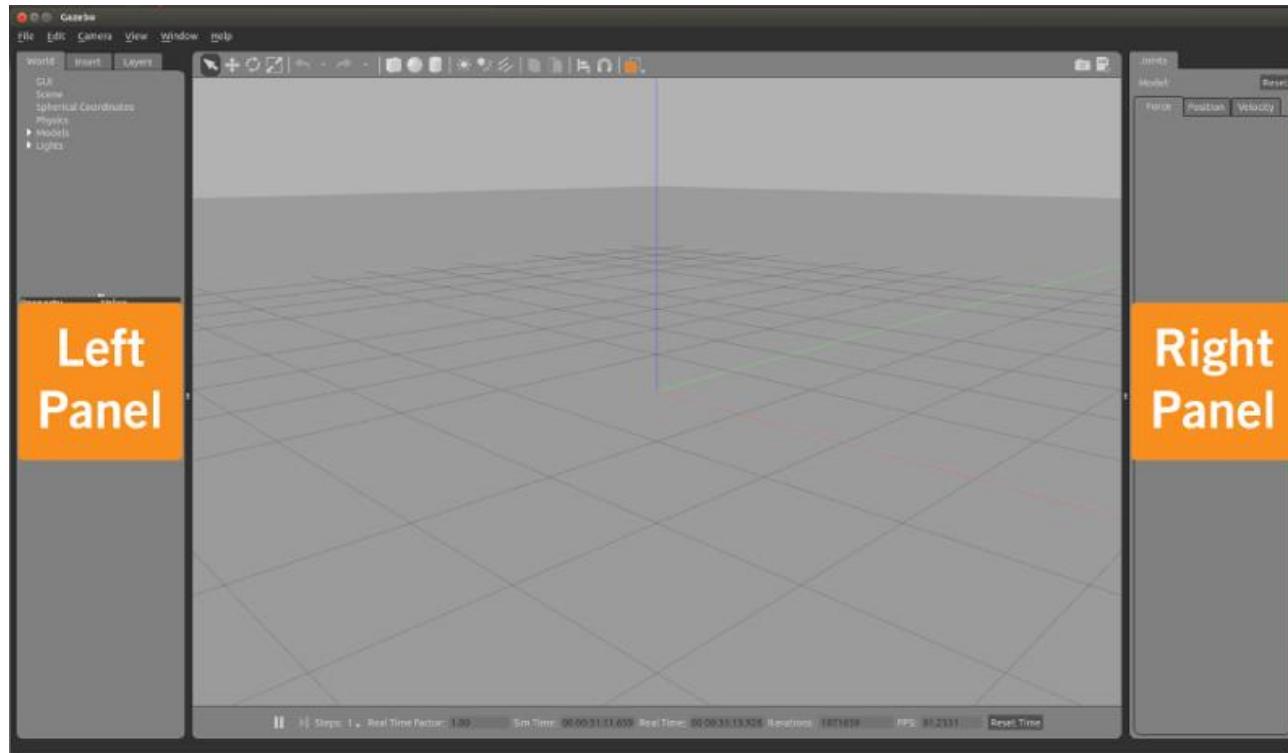
The Scene is the main part of the simulator. This is where the simulated objects are animated and you interact with the environment.



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.

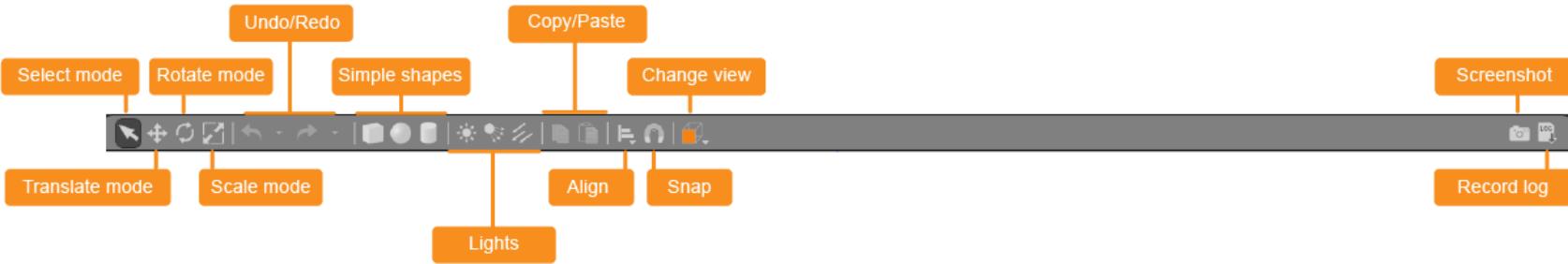
Right Panel (hidden by default)

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.

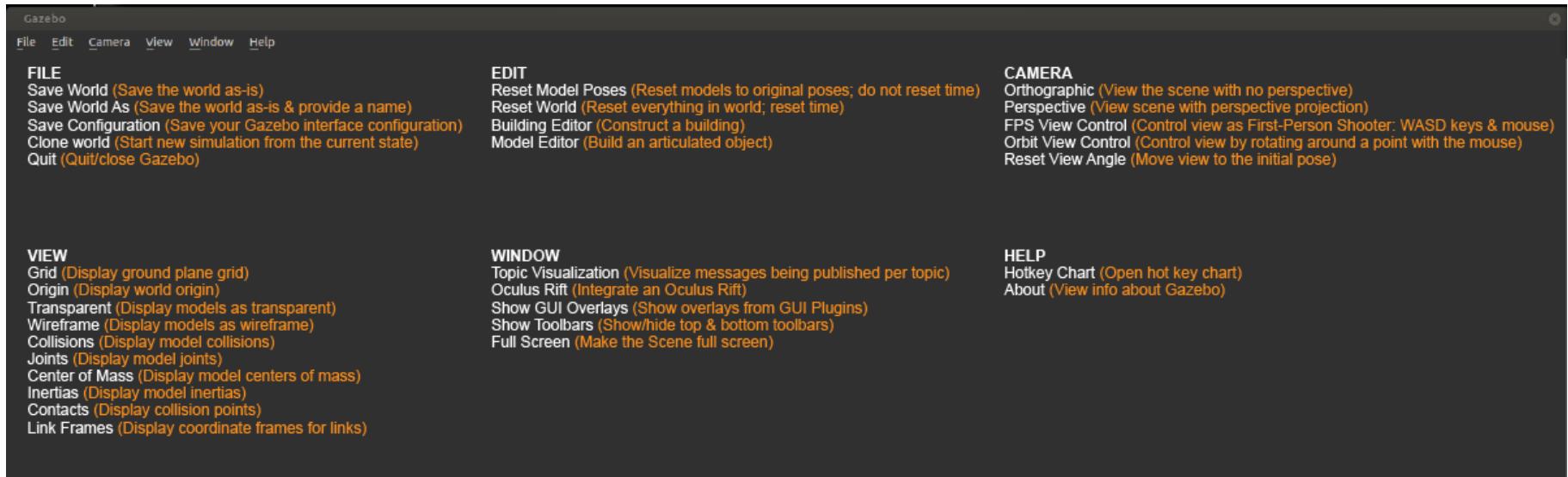
The state of the world in Gazebo is calculated once per iteration. You can see the number of iterations on the right side of the bottom toolbar. Each iteration advances simulation by a fixed number of seconds, called the step size. By default, the step size is 1 ms. You can press the pause button to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

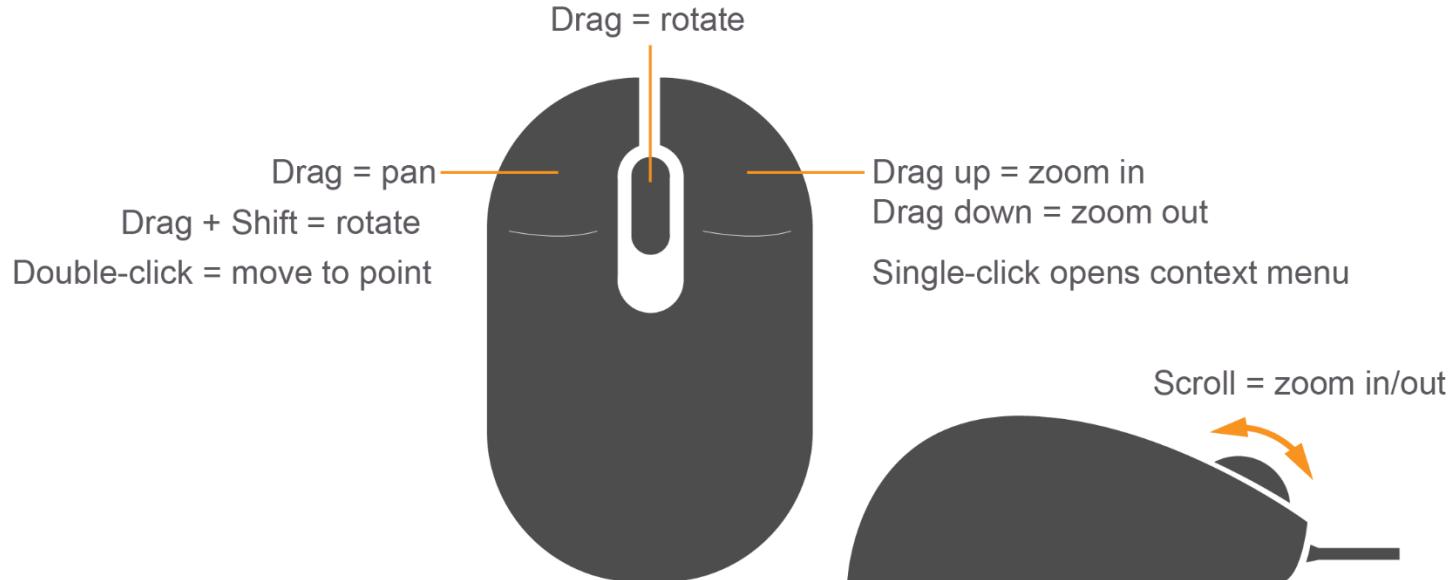


User Interface

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.

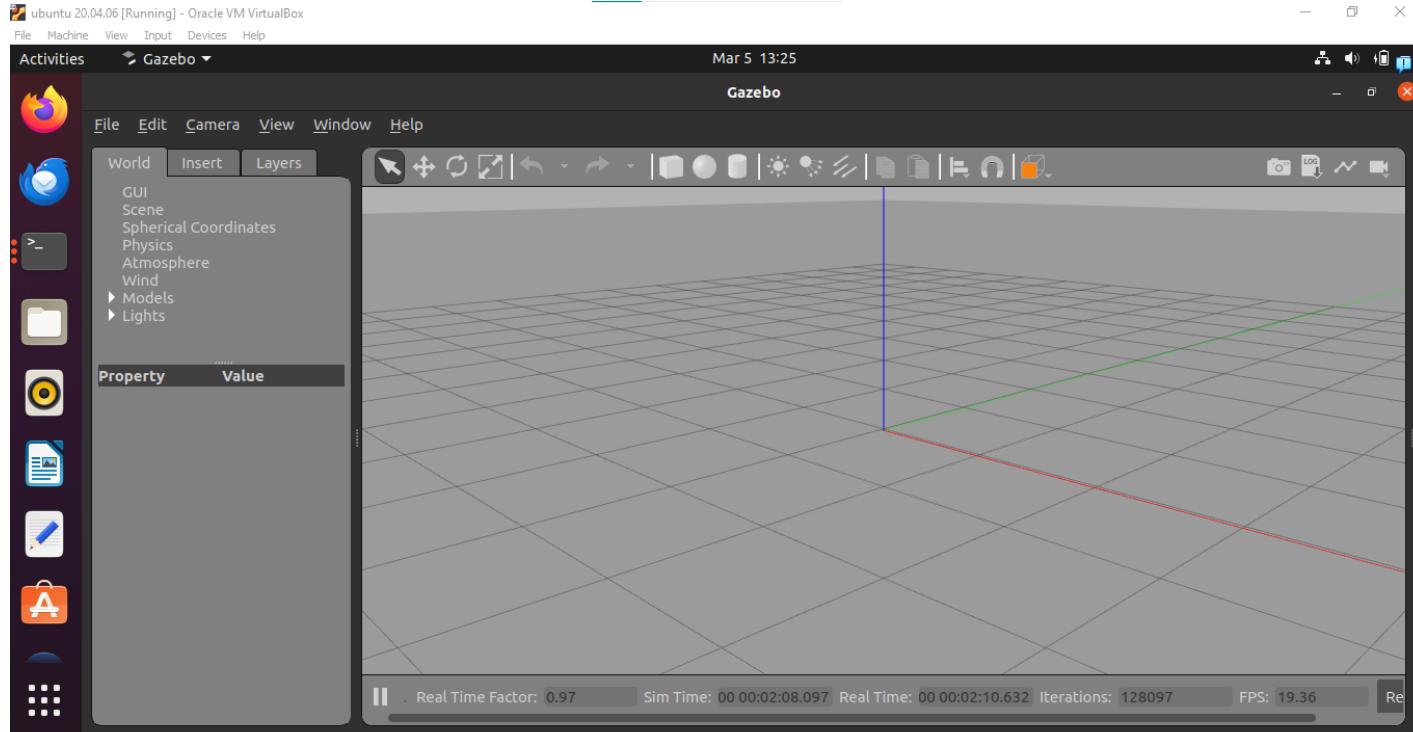
Right-clicking on models will open a context menu with various options. Right-click on a model now to see what's available.



Checking Gazebo

rosrun gazebo_ros gazebo

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



Type here to search



Right Ctrl

01:25 PM
05-03-2024
ENG

Checking Gazebo

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

rostopic list

rosservice list

The screenshot shows a terminal window with two tabs open. The left tab displays the output of the command `rostopic list`, and the right tab displays the output of the command `rosservice list`. Both tabs have the title bar "vboxuser@ubuntu20: ~".

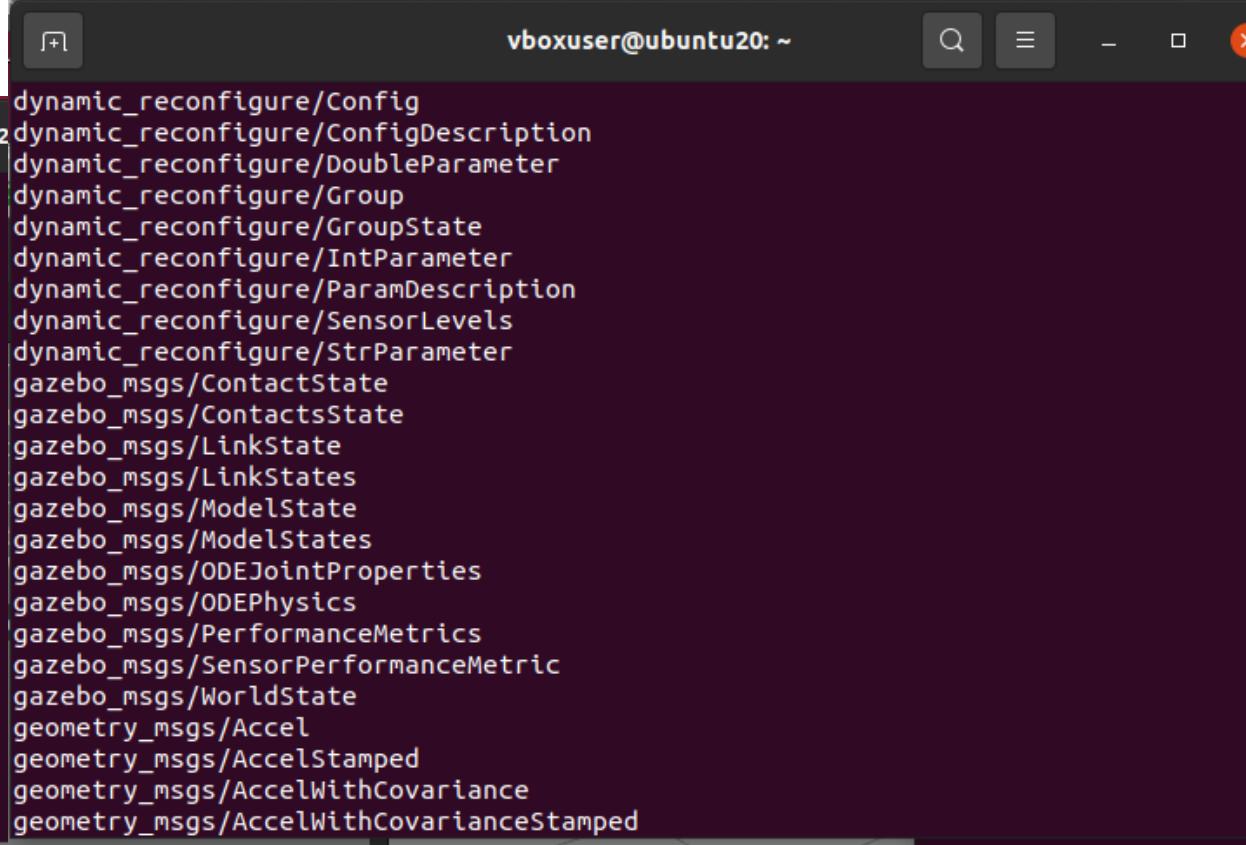
```
vboxuser@ubuntu20:~$ rostopic list
/clock
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/performance_metrics
/gazebo/set_link_state
/gazebo/set_model_state
/rosout
/rosout_agg
vboxuser@ubuntu20:~$
```

```
/boxuser@ubuntu20:~$ rostopic list
/clock
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/performance_metrics
/gazebo/set_link_state
/gazebo/set_model_state
/rosout
/rosout_agg
/boxuser@ubuntu20:~$ rosservice list
/gazebo/apply_body_wrench
/gazebo/apply_joint_effort
/gazebo/clear_body_wrenches
/gazebo/clear_joint_forces
/gazebo/delete_light
/gazebo/delete_model
/gazebo/get_joint_properties
/gazebo/get_light_properties
/gazebo/get_link_properties
/gazebo/get_link_state
/gazebo/get_loggers
/gazebo/get_model_properties
```

Checking Gazebo

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

rosmsg list



A terminal window titled "vboxuser@ubuntu20: ~" showing the output of the "rosmsg list" command. The window has a dark background and light-colored text. The title bar includes standard window controls (minimize, maximize, close) and a search icon.

```
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
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:~$
```

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

```
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_tutorials
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:~$
```

Trutlebot3 ?

Need to install

Turtlebot3



```
vboxuser@ubuntu20:~$ rosrun turtlebot3_gazebo turtlebot3_empty_world.launch
RLException: [turtlebot3_empty_world.launch] is neither a launch file in package [turtlebot3_gazebo] nor is [turtlebot3_gazebo] a launch file name
The traceback for the exception was written to the log file
vboxuser@ubuntu20:~$
```

```
roscore http://ubuntu20:11311/
```

SUMMARY
=====

PARAMETERS

- * /rosdistro: noetic
- * /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [3362]
ROS_MASTER_URI=http://ubuntu20:11311/

setting /run_id to 8db5e5f4-ded6-11ee-91ae-0b15314c22dd
process[rosout-1]: started with pid [3372]
started core service [/rosout]

rosrun 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
```

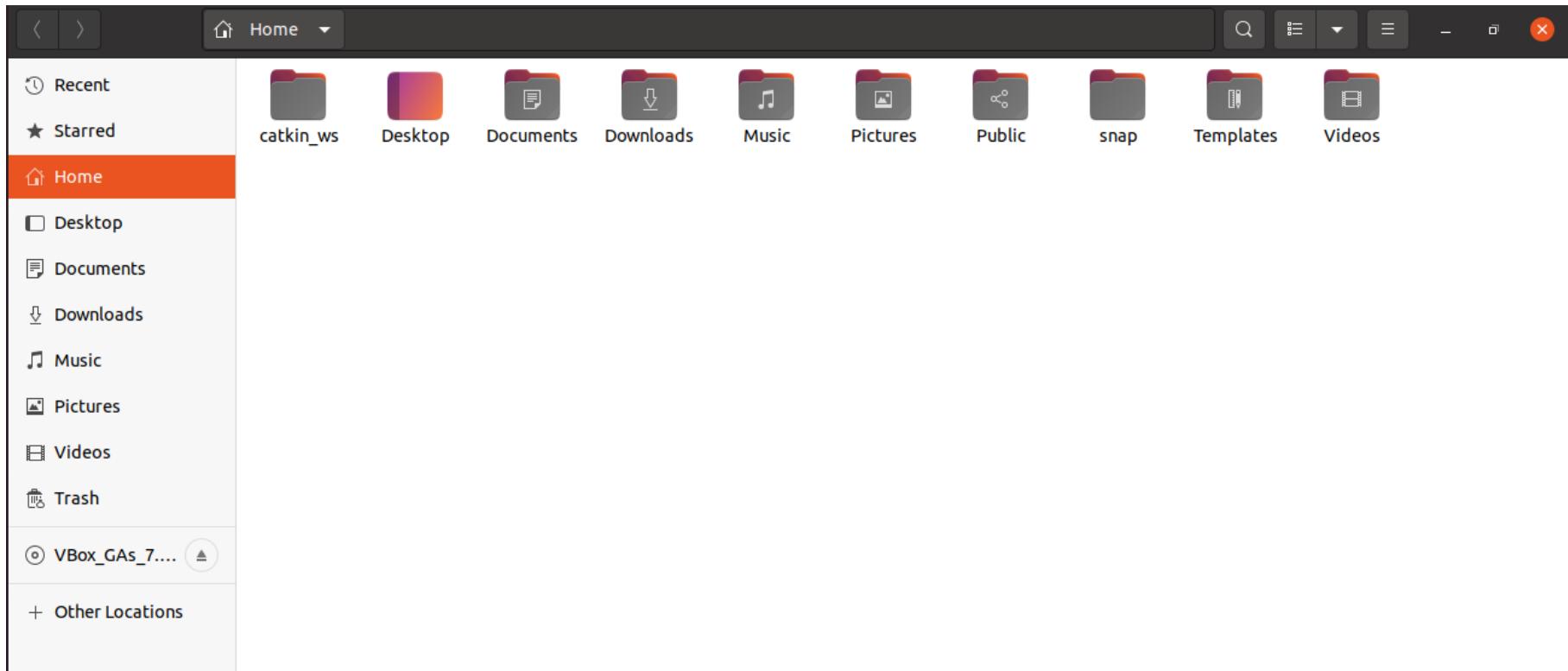
```
git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
```

```
cd ~/turtlebot_ws
```

```
catkin_make
```

```
source ~/turtlebot_ws/devel/setup.bash
```

Installation of Turtlebot3



Installation of Turtlebot3

The image shows two terminal windows side-by-side. The left window displays the command-line process of cloning three GitHub repositories into a workspace directory:

```
vboxuser@ubuntu20:~/turtlebot_ws/src$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
Cloning into 'turtlebot3'...
remote: Enumerating objects: 6636, done.
remote: Counting objects: 100% (1342/1342), done.
remote: Compressing objects: 100% (231/231), done.
remote: Total 6636 (delta 1166), reused 1159 (delta 1111), pack-reused 5294
Receiving objects: 100% (6636/6636), 119.99 MiB | 1.15 MiB/s, done.
Resolving deltas: 100% (4155/4155), done.
vboxuser@ubuntu20:~/turtlebot_ws/src$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
Cloning into 'turtlebot3_msgs'...
remote: Enumerating objects: 415, done.
remote: Counting objects: 100% (173/173), done.
remote: Compressing objects: 100% (58/58), done.
remote: Total 415 (delta 72), reused 156 (delta 61), pack-reused 242
Receiving objects: 100% (415/415), 91.62 KiB | 830.00 KiB/s, done.
Resolving deltas: 100% (173/173), done.
vboxuser@ubuntu20:~/turtlebot_ws/src$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
Cloning into 'turtlebot3_simulations'...
remote: Enumerating objects: 3167, done.
remote: Counting objects: 100% (969/969), done.
remote: Compressing objects: 100% (192/192), done.
remote: Total 3167 (delta 844), reused 778 (delta 777), pack-reused 2198
Receiving objects: 100% (3167/3167), 15.39 MiB | 1.87 MiB/s, done.
Resolving deltas: 100% (1881/1881), done.
vboxuser@ubuntu20:~/turtlebot_ws/src$ cd ..
vboxuser@ubuntu20:~/turtlebot_ws$ catkin_make
```

The right window shows the output of running the `roscore` command:

```
vboxuser@ubuntu20:~$ roscore
... logging to /home/vboxuser/.ros/log/89da67de-dee5-11ee-915f-970965d741e3/roslaunch-ubuntu20-3337.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu20:39157/
ros_comm version 1.16.0

SUMMARY
========
PARAMETERS
  * /rosdistro: noetic
  * /rosversion: 1.16.0
NODES

auto-starting new master
process[master]: started with pid [3513]
ROS_MASTER_URI=http://ubuntu20:11311/

setting /run_id to 89da67de-dee5-11ee-915f-970965d741e3
process[rosout-1]: started with pid [3525]
started core service [/rosout]
```

Installation of Turtlebot3

The image shows two terminal windows side-by-side. The left window, titled 'vboxuser@ubuntu20: ~/turtlebot_ws', displays the output of a CMake build process for a Turtlebot3 example. It shows various stages of generating C++ code from message files, building objects, and linking executables. The right window, titled 'roscore http://ubuntu20:11311', shows the output of starting the ROS master node. It includes a summary of parameters, a list of nodes, and environment variables like ROS_MASTER_URI.

```
vboxuser@ubuntu20:~/turtlebot_ws$ cd turtlebot3_simulations/turtlebot3_gazebo/CMakeFiles/turtlebot3_drive.dir/src/turtlebot3_drive.cpp.o
[ 85%] Generating C++ code from turtlebot3_example/Turtlebot3ActionFeedback.msg
[ 85%] Built target turtlebot3_example_generate_messages_eus
[ 87%] Generating C++ code from turtlebot3_example/Turtlebot3Goal.msg
Scanning dependencies of target turtlebot3_drive
[ 88%] Building CXX object turtlebot3_simulations/turtlebot3_gazebo/CMakeFiles/turtlebot3_drive.dir/src/turtlebot3_drive.cpp.o
[ 90%] Generating C++ code from turtlebot3_example/Turtlebot3Result.msg
[ 91%] Generating C++ code from turtlebot3_example/Turtlebot3Feedback.msg
[ 91%] Built target turtlebot3_example_generate_messages_cpp
Scanning dependencies of target turtlebot3_msgs_generate_messages
[ 91%] Built target turtlebot3_msgs_generate_messages
Scanning dependencies of target turtlebot3_diagnostics
[ 93%] Building CXX object turtlebot3/turtlebot3Bringup/CMakeFiles/turtlebot3_diagnostics.dir/src/turtlebot3_diagnostics.cpp.o
[ 95%] Linking CXX executable /home/vboxuser/turtlebot_ws/devel/lib/turtlebot3_gazebo/flat_world_imu_node
[ 95%] Built target flat_world_imu_node
Scanning dependencies of target turtlebot3_example_generate_messages
[ 95%] Built target turtlebot3_example_generate_messages
[ 96%] Linking CXX executable /home/vboxuser/turtlebot_ws/devel/lib/turtlebot3_gazebo/turtlebot3_drive
[ 98%] Linking CXX executable /home/vboxuser/turtlebot_ws/devel/lib/turtlebot3_fakes/turtlebot3_fake_node
[100%] Linking CXX executable /home/vboxuser/turtlebot_ws/devel/lib/turtlebot3_bringup/turtlebot3_diagnostics
[100%] Built target turtlebot3_fake_node
[100%] Built target turtlebot3_diagnostics
[100%] Built target turtlebot3_drive
vboxuser@ubuntu20:~/turtlebot_ws$
```

```
vboxuser@ubuntu20:~$ roscore
... logging to /home/vboxuser/.ros/log/89da67de-dee5-11ee-915f-970965d741e3/roslaunch-ubuntu20-3337.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu20:39157/
ros_comm version 1.16.0

SUMMARY
========
PARAMETERS
  * /rosdistro: noetic
  * /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [3513]
ROS_MASTER_URI=http://ubuntu20:11311/

setting /run_id to 89da67de-dee5-11ee-915f-970965d741e3
process[rosout-1]: started with pid [3525]
  started core service [/rosout]
```

Installation of Turtlebot3

The terminal window shows the build process of Turtlebot3 software:

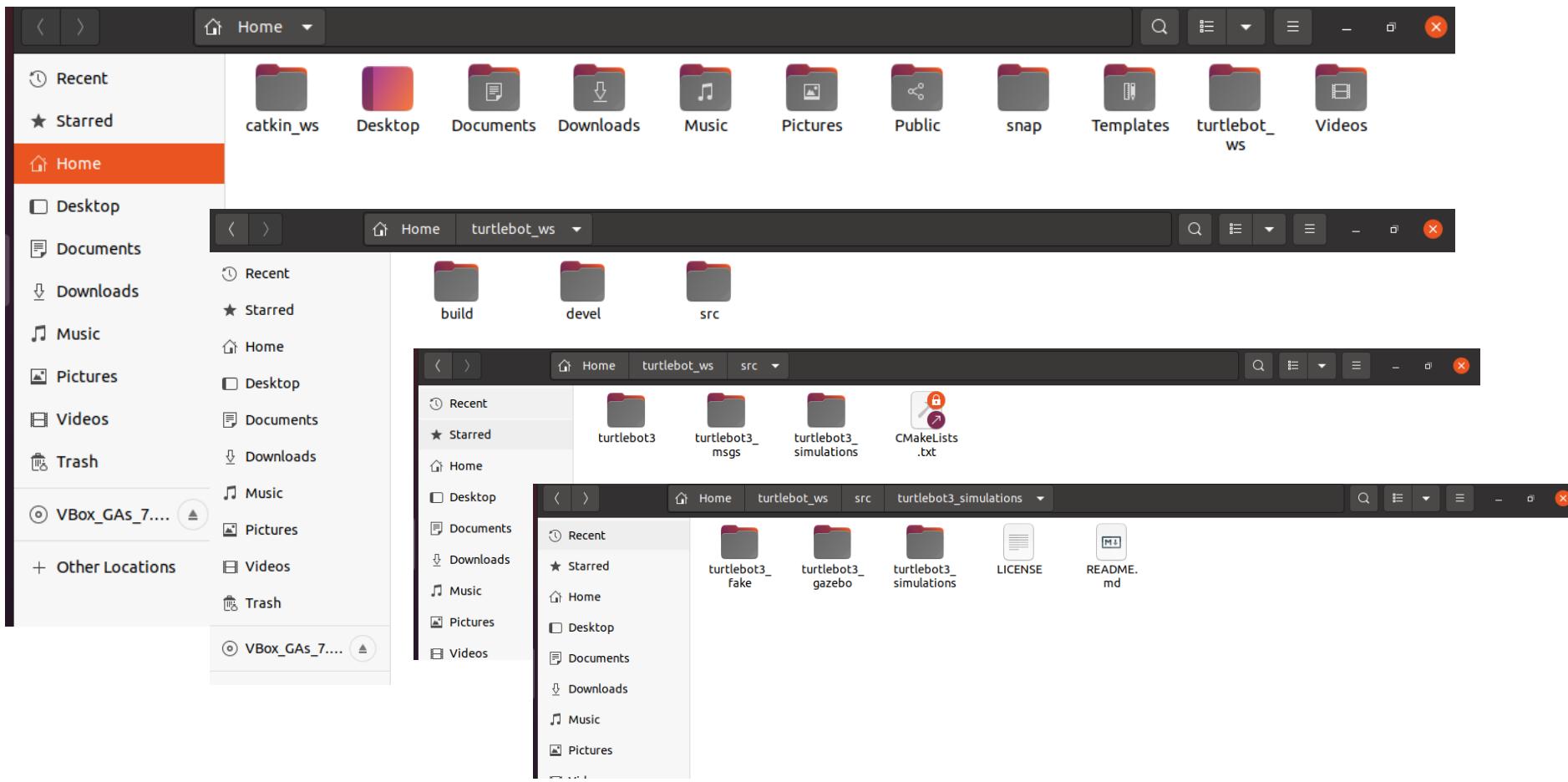
```
[ 87%] Generating C++ code from turtlebot3_example/Turtlebot3Goal.msg
Scanning dependencies of target turtlebot3_drive
[ 88%] Building CXX object turtlebot3_simulations/turtlebot3_gazebo/CMakeFiles/turtlebot3_drive.dir/src/turtlebot3_drive.cpp.o
[ 90%] Generating C++ code from turtlebot3_example/Turtlebot3Result.msg
[ 91%] Generating C++ code from turtlebot3_example/Turtlebot3Feedback.msg
[ 91%] Built target turtlebot3_example_generate_messages_cpp
Scanning dependencies of target turtlebot3_msgs_generate_messages
[ 91%] Built target turtlebot3_msgs_generate_messages
Scanning dependencies of target turtlebot3_diagnostics
[ 93%] Building CXX object turtlebot3/turtlebot3_bringup/CMakeFiles/turtlebot3_diagnostics.dir/src/turtlebot3_diagnostics.cpp.o
[ 95%] Linking CXX executable /home/vboxuser/turtlebot_ws/devel/lib/turtlebot3_simulations/flat_world_imu_node
[ 95%] Built target flat_world_imu_node
Scanning dependencies of target turtlebot3_example_generate_messages
[ 95%] Built target turtlebot3_example_generate_messages
[ 96%] Linking CXX executable /home/vboxuser/turtlebot_ws/devel/lib/turtlebot3_gazebo/turtlebot3_drive
[ 98%] Linking CXX executable /home/vboxuser/turtlebot_ws/devel/lib/turtlebot3_fakes/turtlebot3_fake_node
[100%] Linking CXX executable /home/vboxuser/turtlebot_ws/devel/lib/turtlebot3_bringup/turtlebot3_diagnostics
[100%] Built target turtlebot3_fake_node
[100%] Built target turtlebot3_diagnostics
[100%] Built target turtlebot3_drive
vboxuser@ubuntu20:~/turtlebot_ws$ source ~/turtlebot_ws/devel/setup.bash
vboxuser@ubuntu20:~/turtlebot_ws$ gedit ~/.bashrc
```

The code editor shows the .bashrc file with the following content:

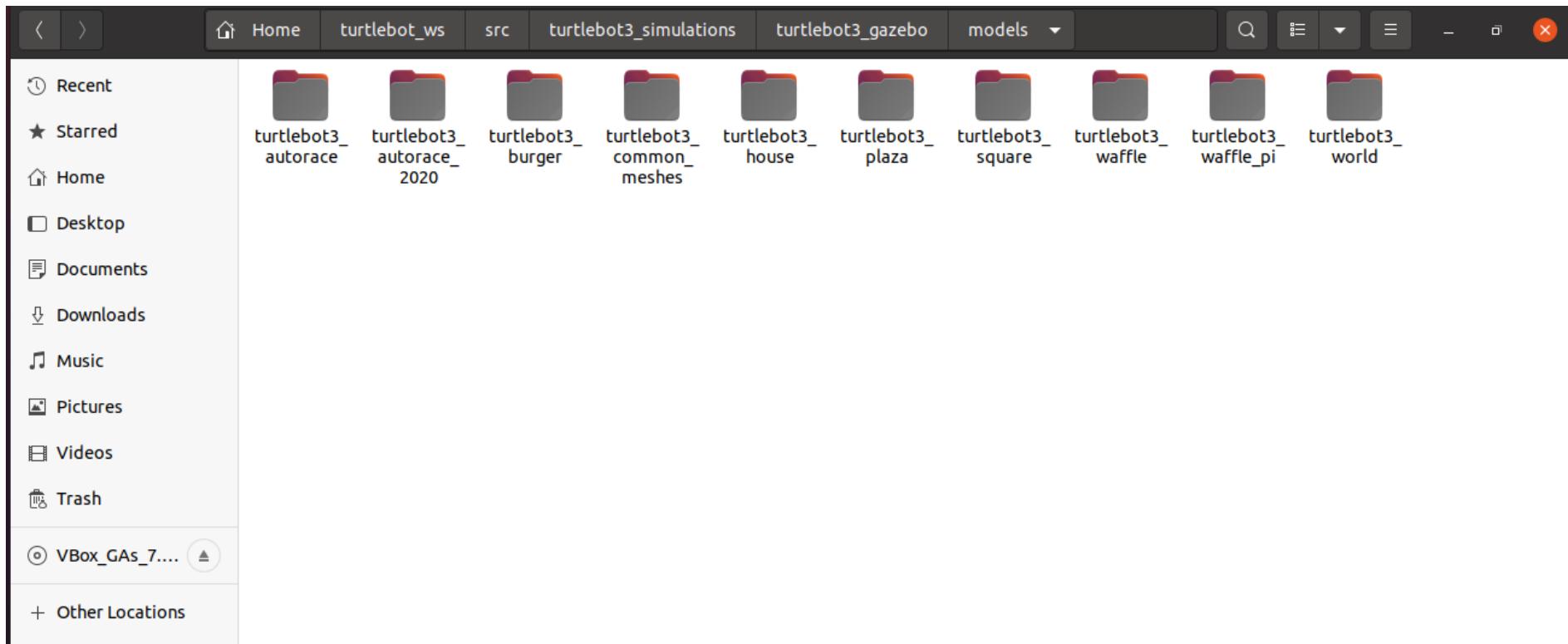
```
file like
101 # ~/.bash_aliases, instead of adding them here directly.
102 # See /usr/share/doc/bash-doc/examples in the bash-doc
103 # package.
104 if [ -f ~/.bash_aliases ]; then
105     . ~/.bash_aliases
106 fi
107
108 # enable programmable completion features (you don't need
109 # to enable
109 # this, if it's already enabled in /etc/bash.bashrc and /etc/profile
110 # sources /etc/bash.bashrc).
111 if ! shopt -oq posix; then
112     if [ -f /usr/share/bash-completion/bash_completion ]; then
113         . /usr/share/bash-completion/bash_completion
114     elif [ -f /etc/bash_completion ]; then
115         . /etc/bash_completion
116     fi
117 fi
118 source /opt/ros/noetic/setup.bash
119 source ~/catkin_ws/devel/setup.bash
120
121 source /opt/ros/noetic/setup.bash
122 source ~/catkin_ws/devel/setup.bash
123 source ~/turtlebot_ws/devel/setup.bash
124
```

Terminal status bar: sh ▾ Tab Width: 8 ▾ Ln 124, Col 1

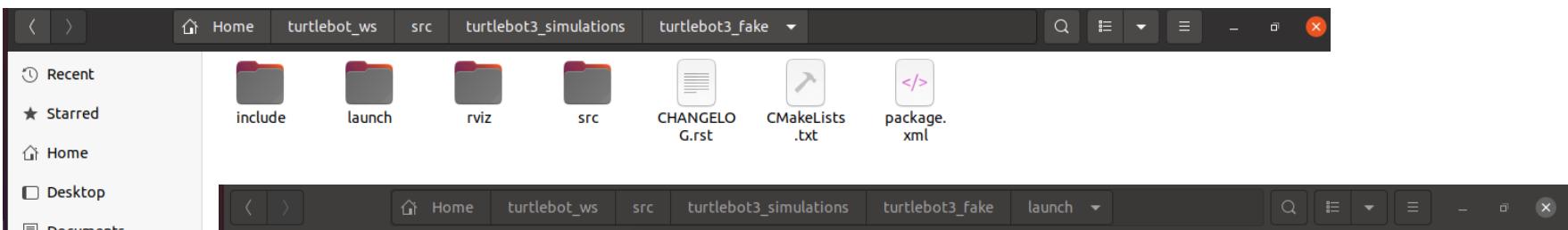
Installation of Turtlebot3



Installation of Turtlebot3



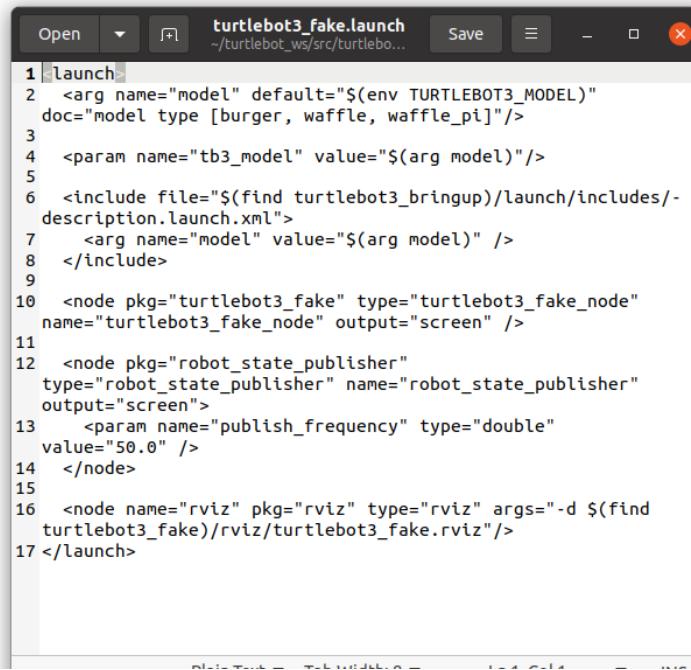
Installation of Turtlebot3



The screenshot shows a file manager window with the following directory structure:

```
Home
  turtlebot_ws
    src
      turtlebot3_simulations
        turtlebot3_fake
          launch
          include
          rviz
          src
          CHANGELOG.rst
          CMakeLists.txt
          package.xml
```

The `turtlebot3_fake.launch` file is selected and displayed in a code editor window.



```
1 <launch>
2   <arg name="model" default="$(env TURTLEBOT3_MODEL)"
3     doc="model type [burger, waffle, waffle_pi]"/>
4   <param name="tb3_model" value="$(arg model)"/>
5   <include file="$(find turtlebot3_bringup)/launch/includes/-description.launch.xml">
6     <arg name="model" value="$(arg model)" />
7   </include>
8
9
10  <node pkg="turtlebot3_fake" type="turtlebot3_fake_node"
11    name="turtlebot3_fake_node" output="screen" />
12
13  <node pkg="robot_state_publisher"
14    type="robot_state_publisher" name="robot_state_publisher"
15    output="screen">
16    <param name="publish_frequency" type="double"
17      value="50.0" />
18  </node>
19
20  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
21    turtlebot3_fake)/rviz/turtlebot3_fake.rviz"/>
22</launch>
```

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

```
vboxuser@ubuntu20:~/turtlebot_ws$ export TURTLEBOT3_MODEL=burger
vboxuser@ubuntu20:~/turtlebot_ws$
```

```
PARAMETERS
* /rosdistro: noetic
* /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [3513]
ROS_MASTER_URI=http://ubuntu20:11311/

setting /run_id to 89da67de-dee5-11ee-915f-970965d741e3
process[rosout-1]: started with pid [3525]
started core service [/rosout]
```

```
vboxuser@ubuntu20:~$
```

Simulate the robot motion in Rviz

Let's bring the Rviz simulation environment

`roslaunch turtlebot3_fake turtlebot3_fake.launch`

```
/home/vboxuser/turtlebot_ws/src/turtlebot3_simulations/turtlebot3_fake/launch/turtlebot3_fake.launch... /home/vboxuser/turtlebot_ws$ export TURTLEBOT3_MODEL=burger  
vboxuser@ubuntu20:~/turtlebot_ws$ roslaunch turtlebot3_fake turtlebot3_fake.launch  
... logging to /home/vboxuser/.ros/log/89da67de-dee5-11ee-915f-970965d741e3/roslaunch-ubuntu20-5594.log  
Checking log directory for disk usage. This may take a while.  
Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage is <1GB.  
  
xacro: in-order processing became default in ROS Melodic. You can drop the option.  
started roslaunch server http://ubuntu20:40727/  
  
SUMMARY  
=====
```

PARAMETERS

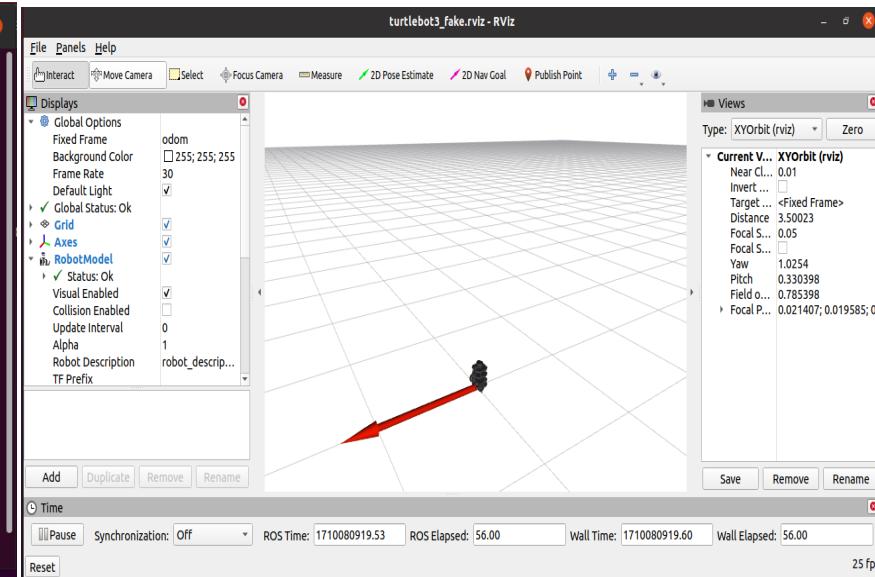
- * `/robot_description: <?xml version="1....`
- * `/robot_state_publisher/publish_frequency: 50.0`
- * `/rosdistro: noetic`
- * `/rosversion: 1.16.0`
- * `/tb3_model: burger`

NODES

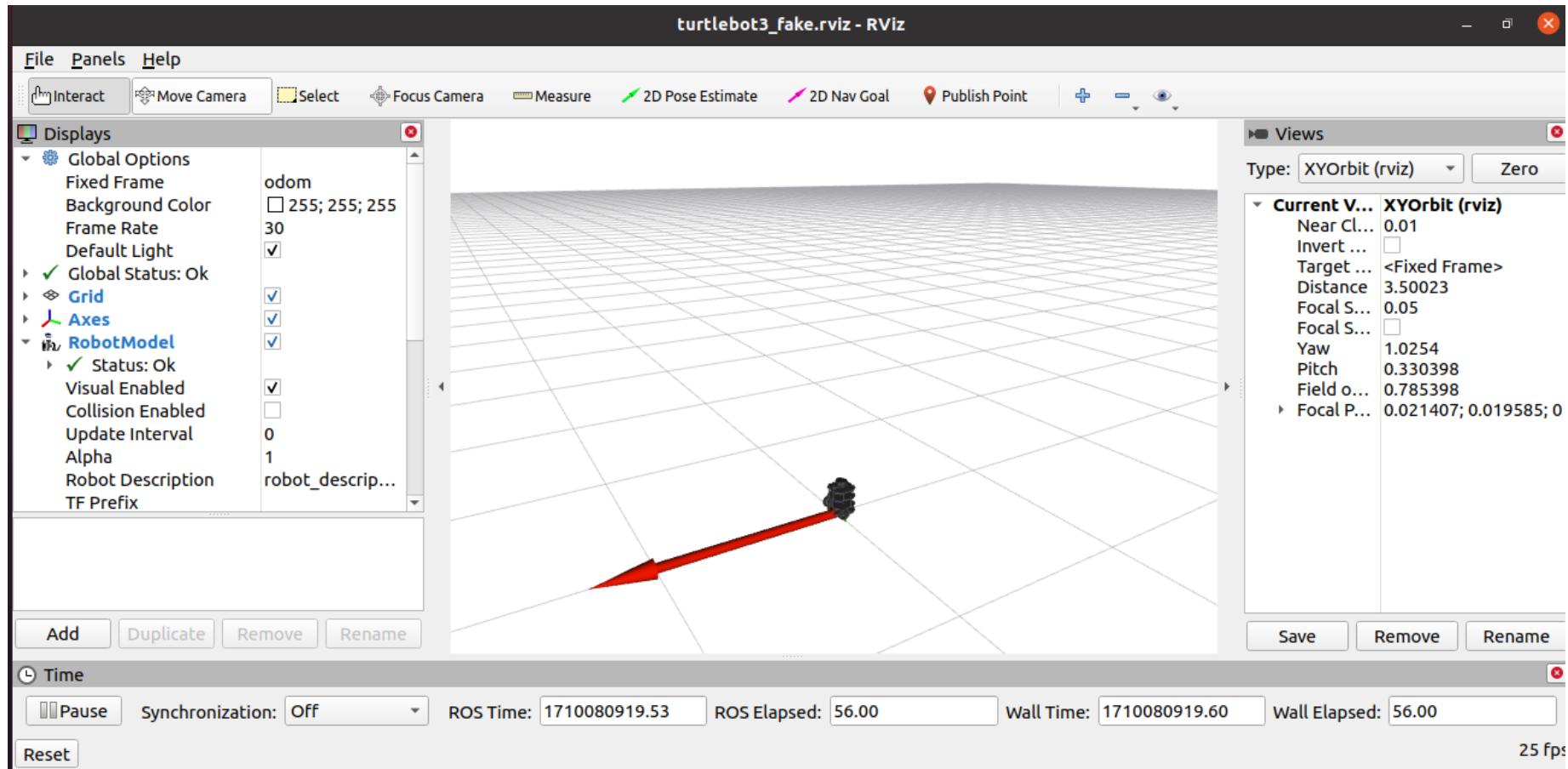
- /
 - `robot_state_publisher (robot_state_publisher/robot_state_publisher)`
 - `rviz (rviz/rviz)`
 - `turtlebot3_fake_node (turtlebot3_fake/turtlebot3_fake_node)`

ROS_MASTER_URI=<http://localhost:11311>

process[turtlebot3_fake_node-1]: started with pid [5610]
process[robot_state_publisher-2]: started with pid [5611]



Simulate the robot motion in Rviz



Simulate the robot motion in Rviz

```
vboxuser@ubuntu20:~$ rosrun turtlebot3 teleop turtlebot3_teleop_key.launch
... logging to /home/vboxuser/.ros/log/89da67de-dee5-11ee-915f-970965d741e3/roslaunch-ubuntu20-5662.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

RLEexception: Invalid <arg> tag: environment variable 'TURTLEBOT3_MODEL' is not set.

Arg xml is <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
The traceback for the exception was written to the log file
vboxuser@ubuntu20:~$
```

```
/home/vboxuser/turtlebot_ws/src/turtlebot3/turtlebot3_teleop/la...
vboxuser@ubuntu20:~$ export TURTLEBOT3_MODEL=burger
vboxuser@ubuntu20:~$ rosrun turtlebot3 teleop turtlebot3_teleop_key.launch
... logging to /home/vboxuser/.ros/log/89da67de-dee5-11ee-915f-970965d741e3/roslaunch-ubuntu20-5670.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ubuntu20:43997/
SUMMARY
=====
PARAMETERS
  * /model: burger
  * /rosdistro: noetic
  * /rosversion: 1.16.0
```

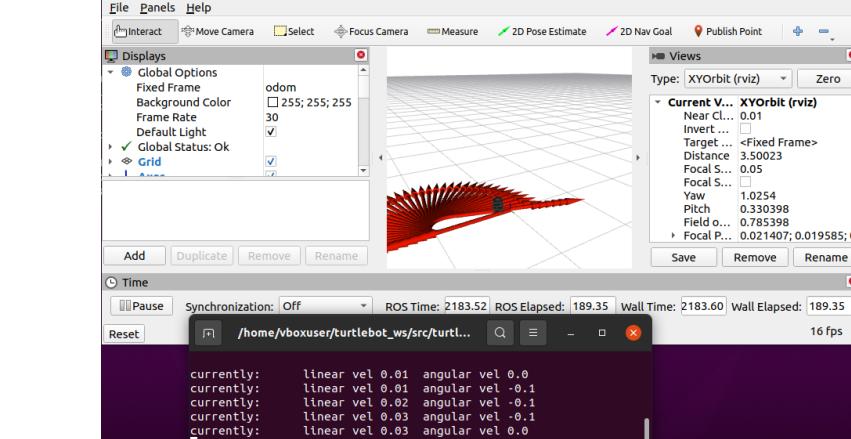
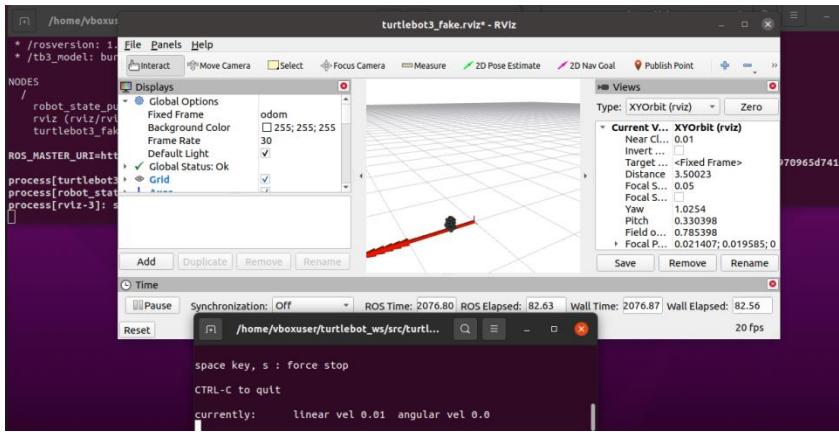
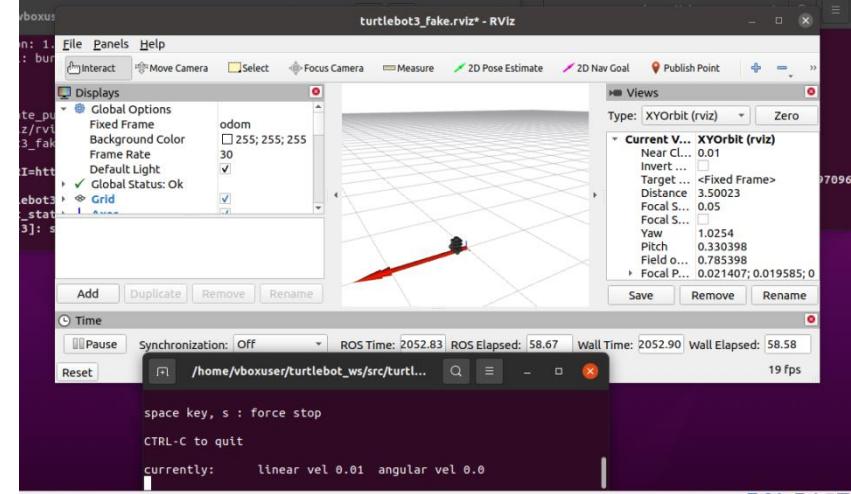
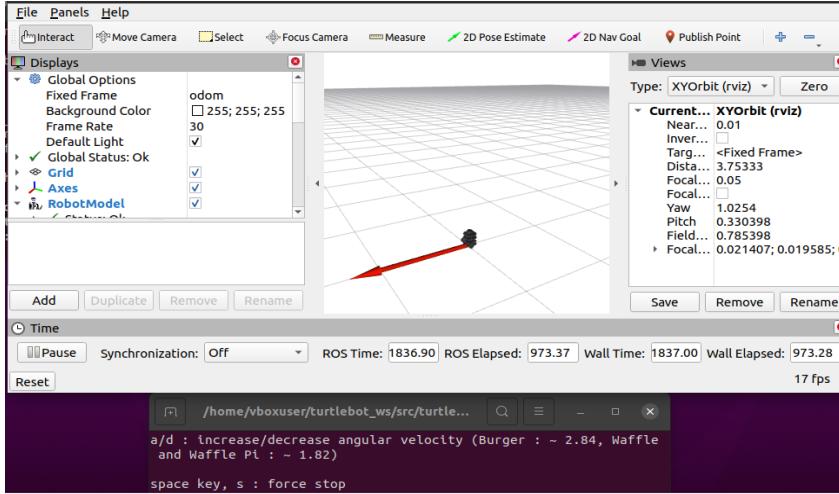
```
/home/vboxuser/turtlebot_ws/src/turtlebot3/turtlebot3_teleop/la...
Control Your TurtleBot3!
-----
Moving around:
      W
    a   s   d
      X

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.2
6)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.
82)

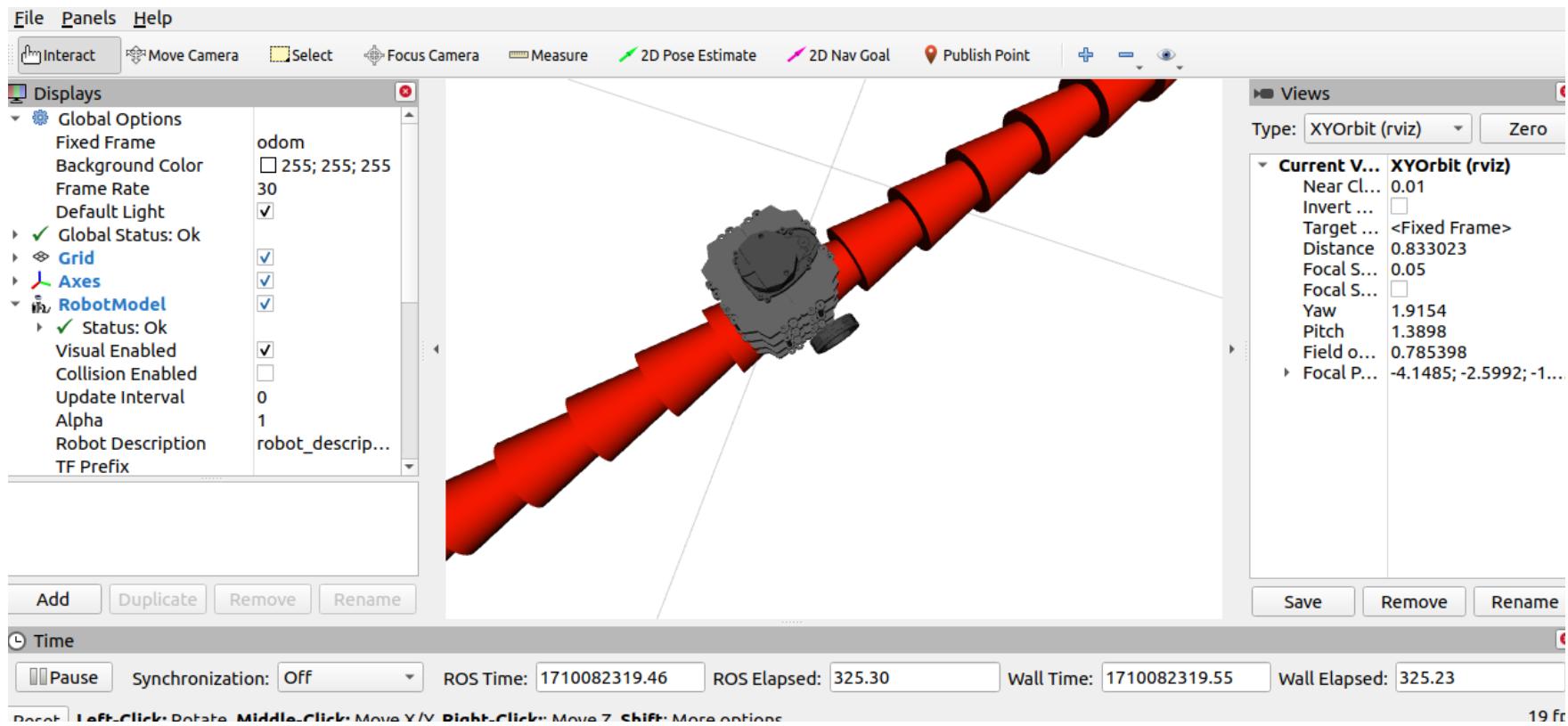
space key, s : force stop

CTRL-C to quit
```

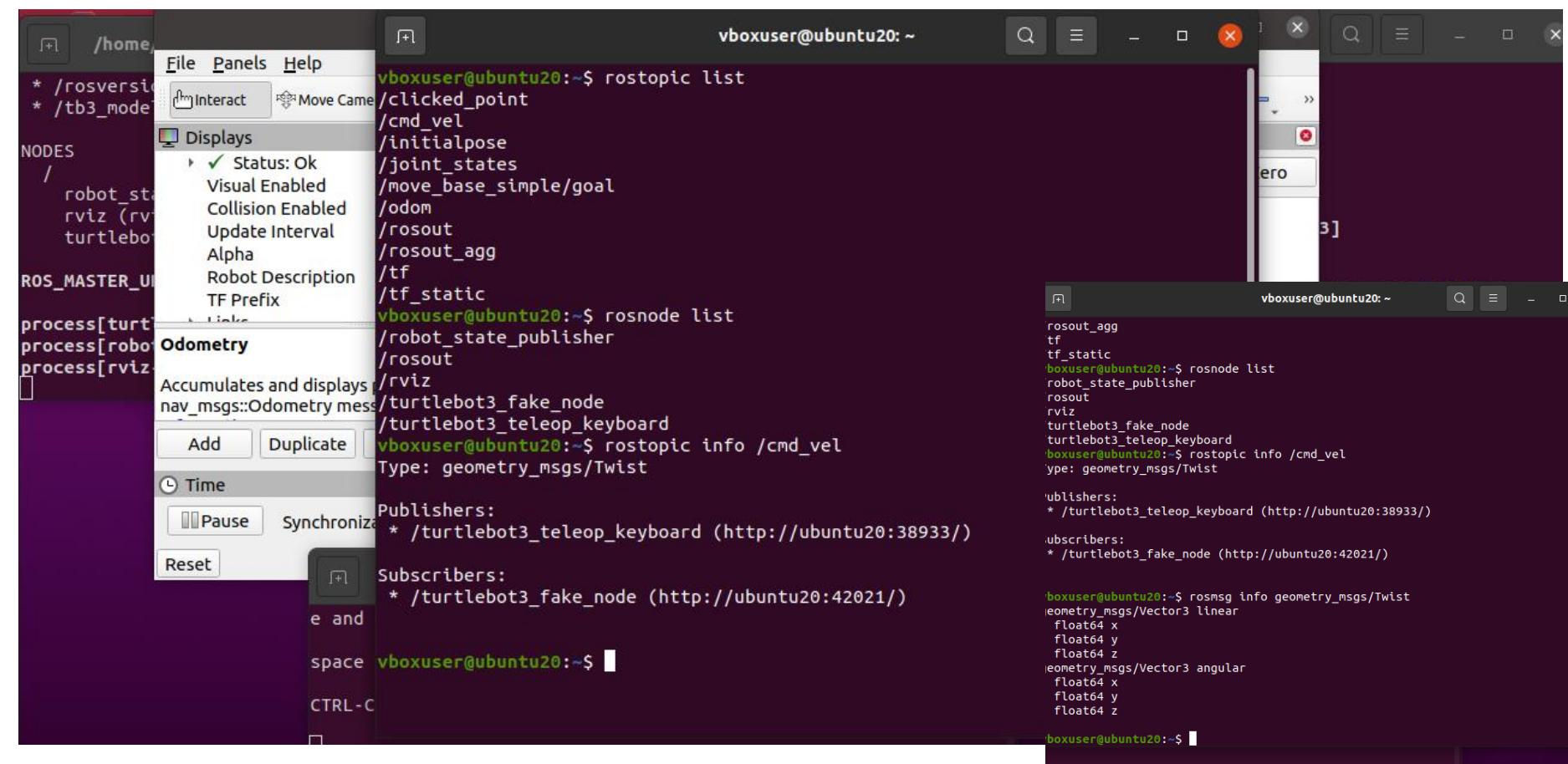
Simulate the robot motion in Rviz



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
```

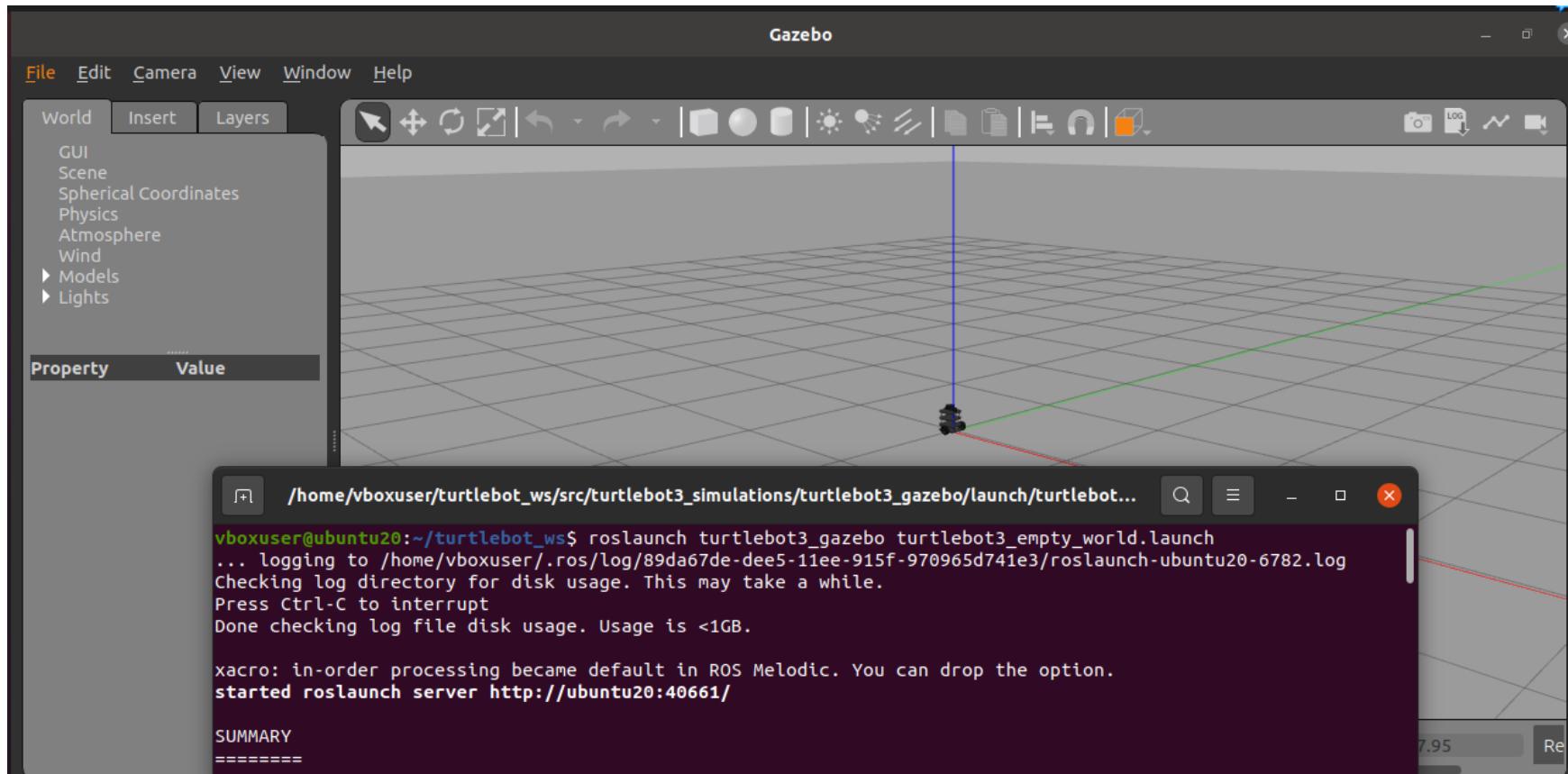
Open an another terminal

```
export TURTLEBOT3_MODEL=burger
```

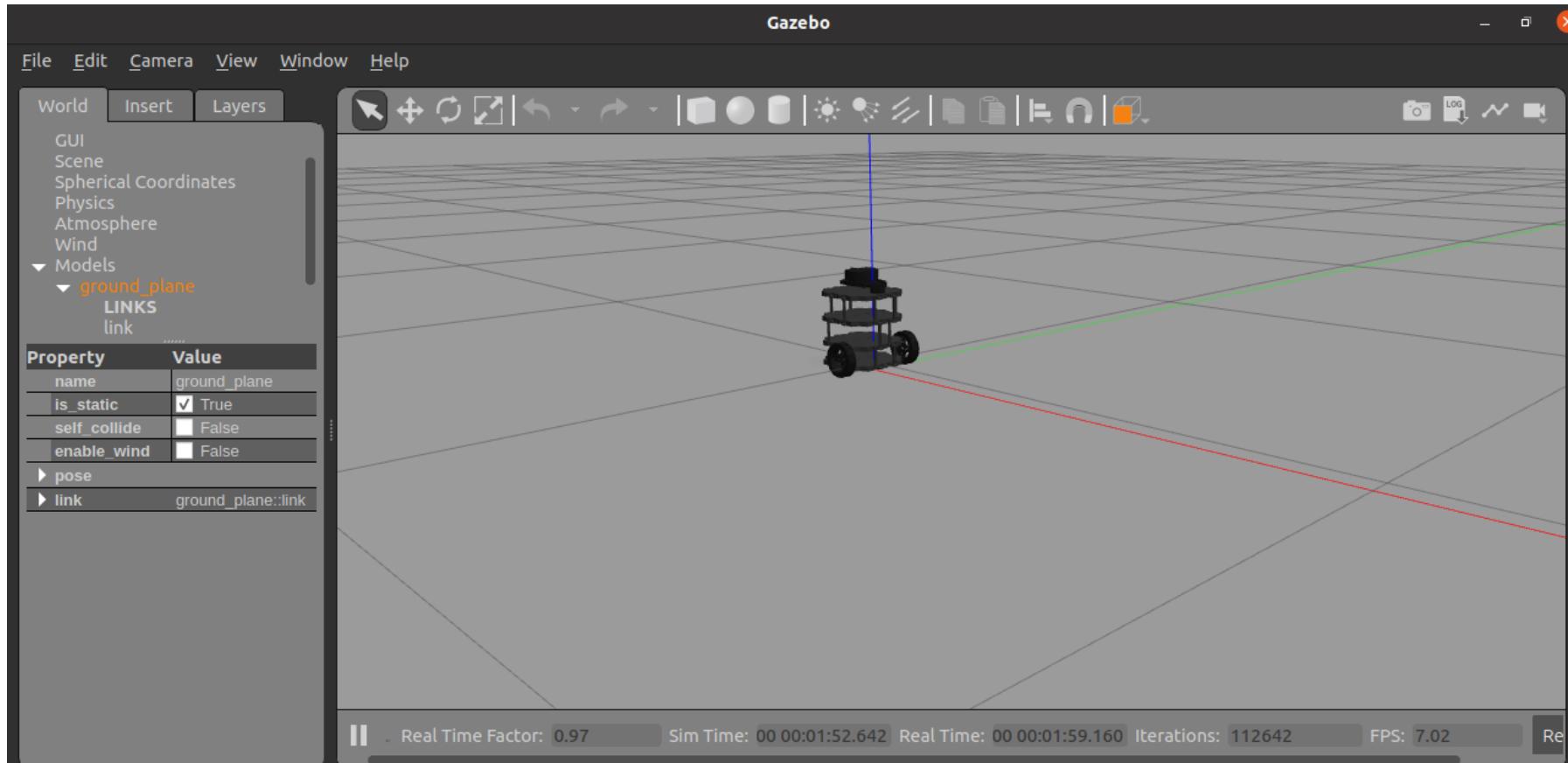
```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

rostopic list
rosnode list

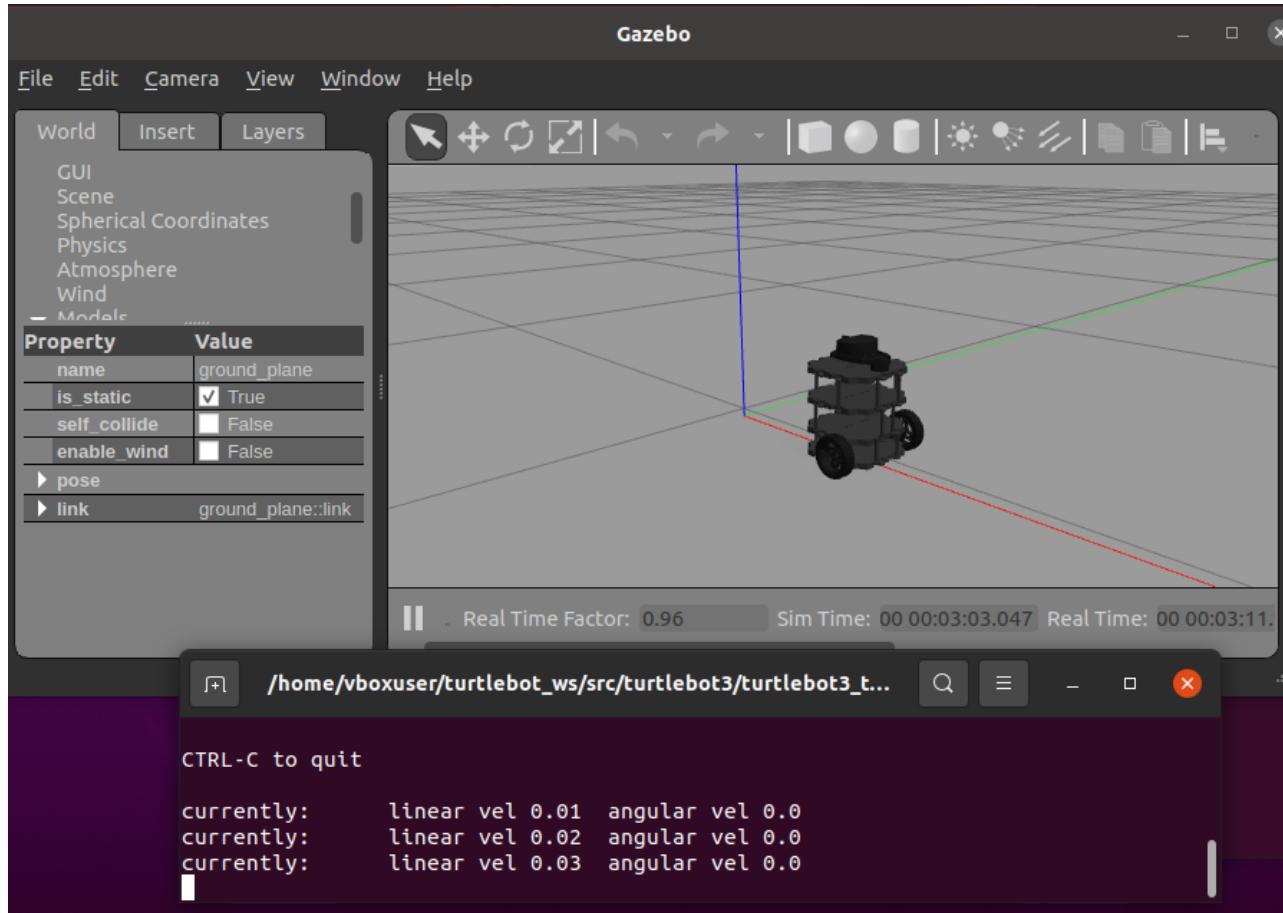
Simulate the robot motion in Gazebo



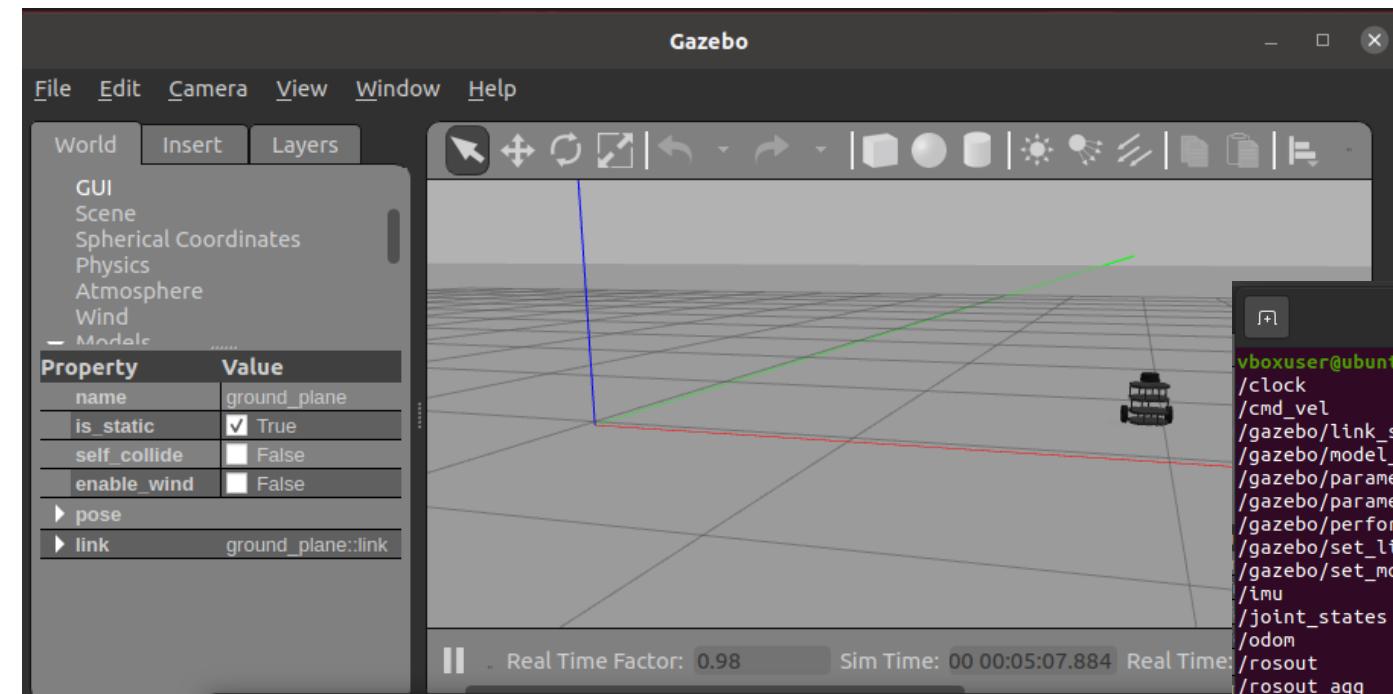
Simulate the robot motion in Gazebo



Simulate the robot motion in Gazebo



Simulate the robot motion in Gazebo

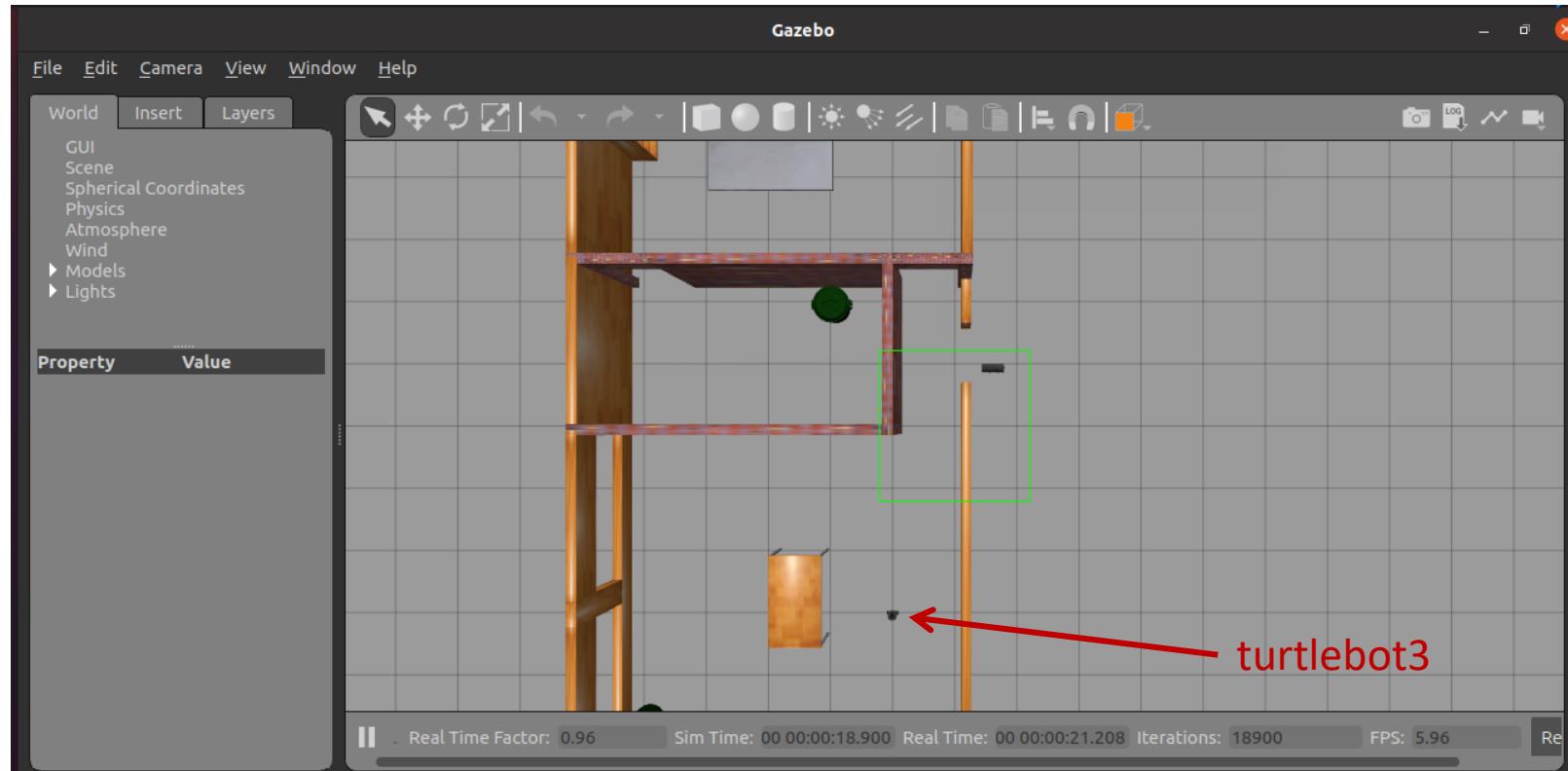


```
/home/vboxuser/turtlebot_ws/src/turtlebot3/turtlebot3_t...
currently: linear vel 0.0 angular vel 0.0
currently: linear vel 0.01 angular vel 0.0
currently: linear vel 0.02 angular vel 0.0
currently: linear vel 0.03 angular vel 0.0
currently: linear vel 0.04 angular vel 0.0
currently: linear vel 0.05 angular vel 0.0
```

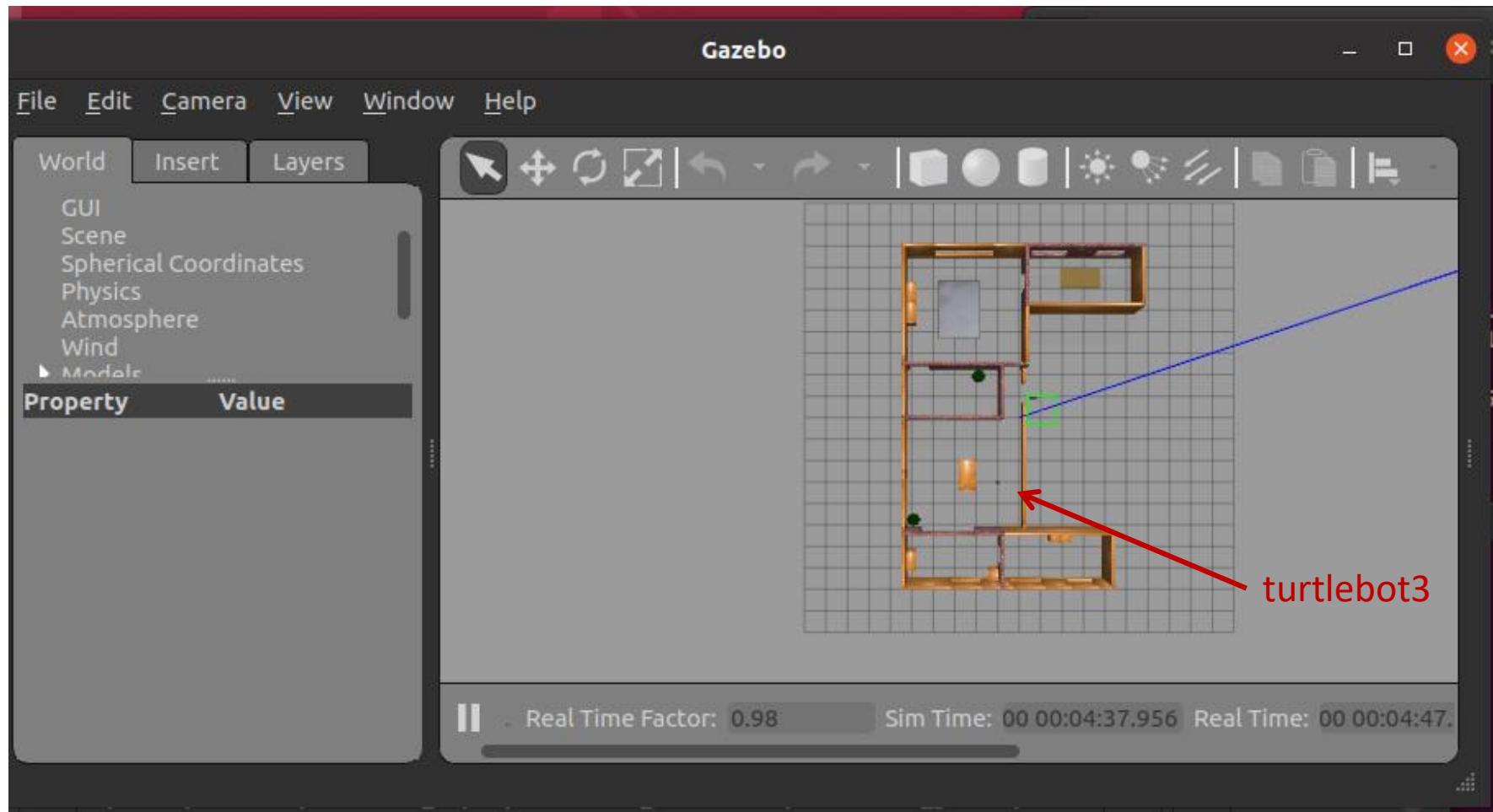
```
vboxuser@ubuntu20:~$ rostopic list
/clock
/cmd_vel
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/performance_metrics
/gazebo/set_link_state
/gazebo/set_model_state
/imu
/joint_states
/odom
/rosout
/rosout_agg
/scan
/tf
vboxuser@ubuntu20:~$ rosnode list
/gazebo
/gazebo_gui
/rosout
/turtlebot3_teleop_keyboard
vboxuser@ubuntu20:~$
```

Simulate the robot motion in Gazebo

Roslaunch turtlebot3_gazebo turtlebot3_house.launch



Simulate the robot motion in Gazebo



Simulate the robot motion in Gazebo

