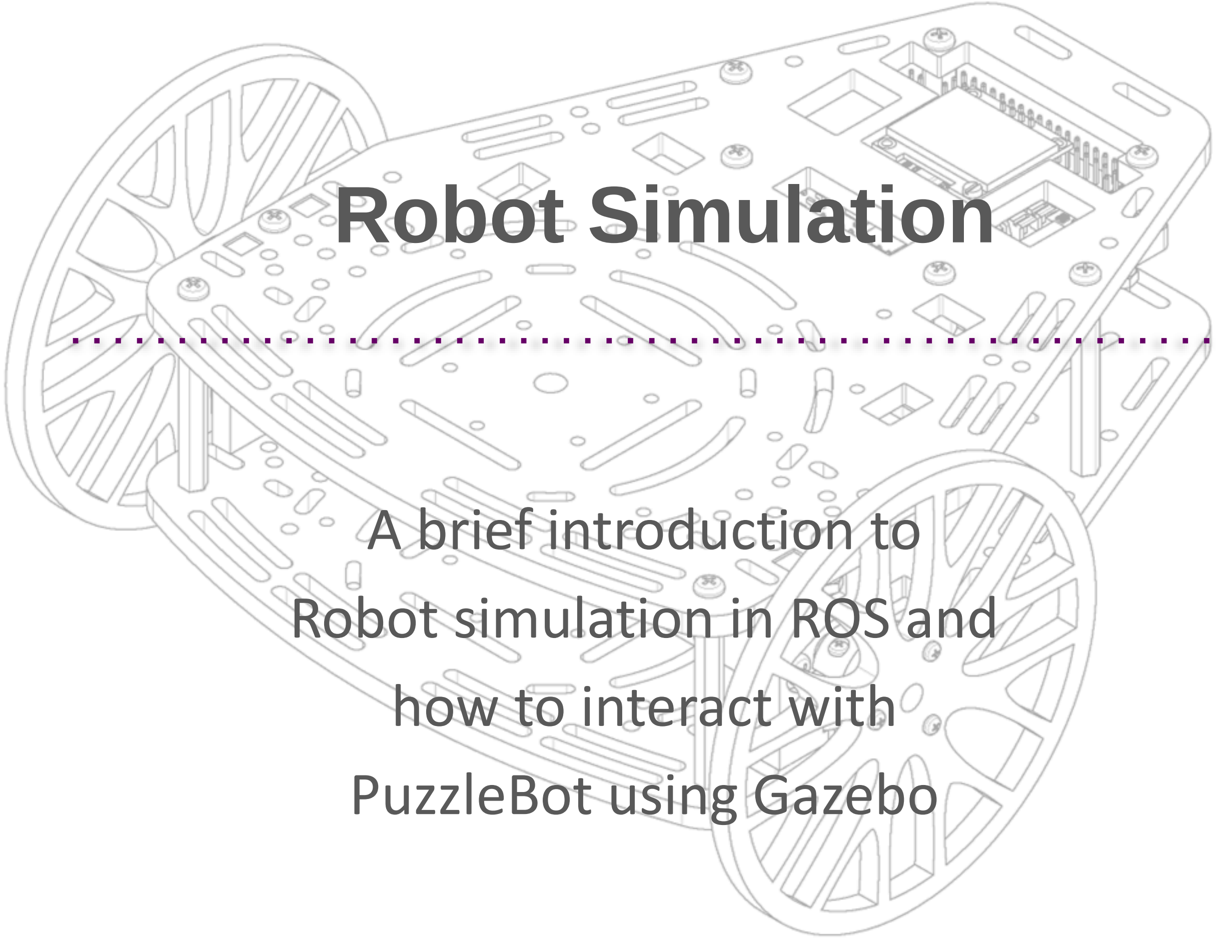


Robot Simulation

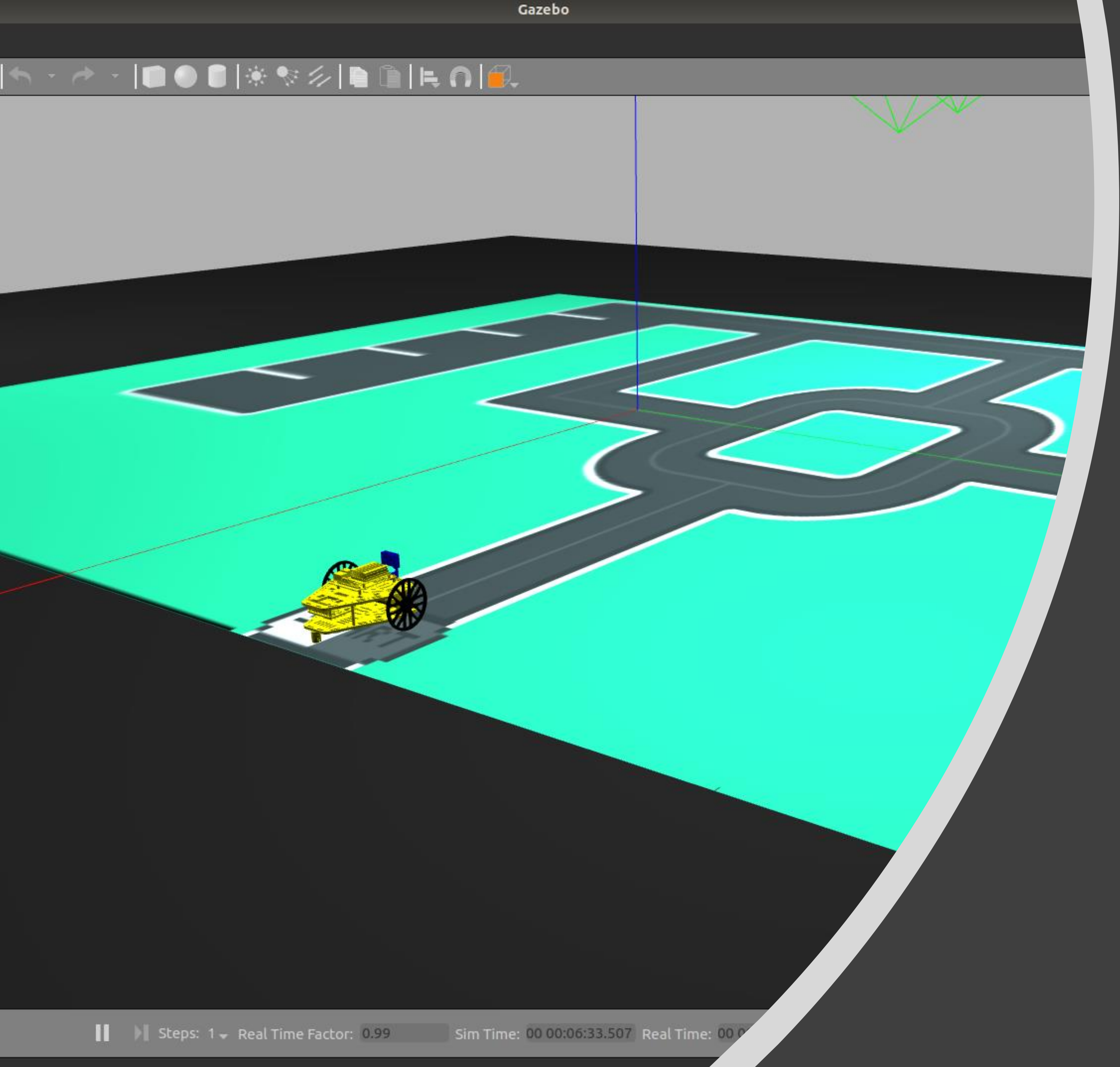
A brief introduction to
Robot simulation in ROS and
how to interact with
PuzzleBot using Gazebo

Session 1a: Robot Simulation.

I. ROS Visualisation Tools
- Robot Model for Simulation
- ROS Visualisation Interfaces
  - What is RVIZ and Gazebo
  - Comparison
- Gazebo
  - -Connection between ROS and Gazebo
  - -ROS Control

II. Walkthrough – using PuzzleBot in Gazebo
- Gazebo interface
- How to create and modify a world
- Spawn Puzzlebot in Gazebo

Requirements: Laptop, ROS preinstalled, Ubuntu preinstalled, Basic Knowledge of Python.


Gazebo

Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:06:33.507 Real Time: 00 0

UPY BIS

NVIDIA

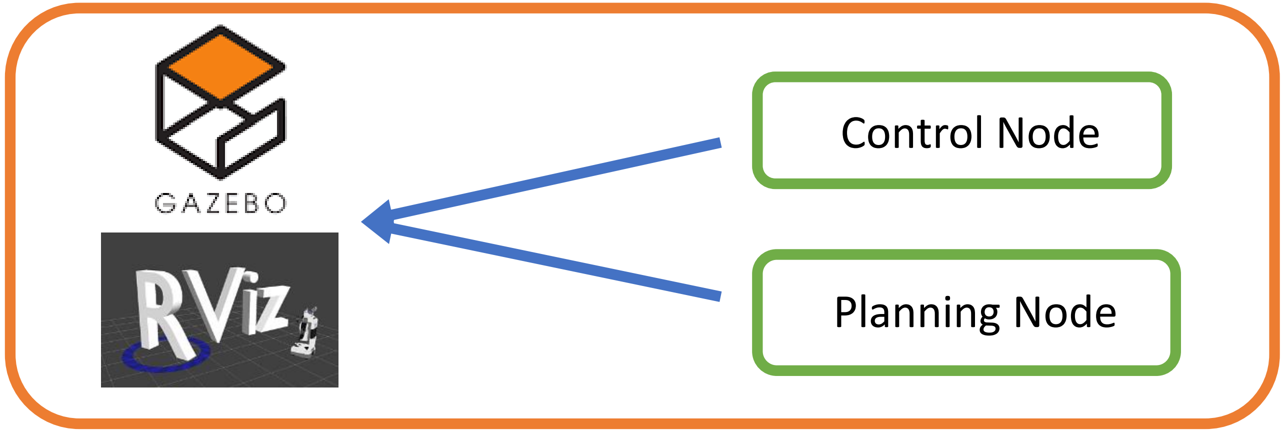
ROS visualisation and simulation tools

ROS tools

Visualization vs Simulation

In ROS, we can describe a robot and its behaviour. Some tools can help us to test our algorithms without the need of having a robot at hand.

The most common tools available are RVIZ and Gazebo, but they differ from each other. One will allow us to display information about my robot, and the other will enable simulation.



Control Node

Planning Node

Gazebo and Rviz Comparison

Key features

RVIZ

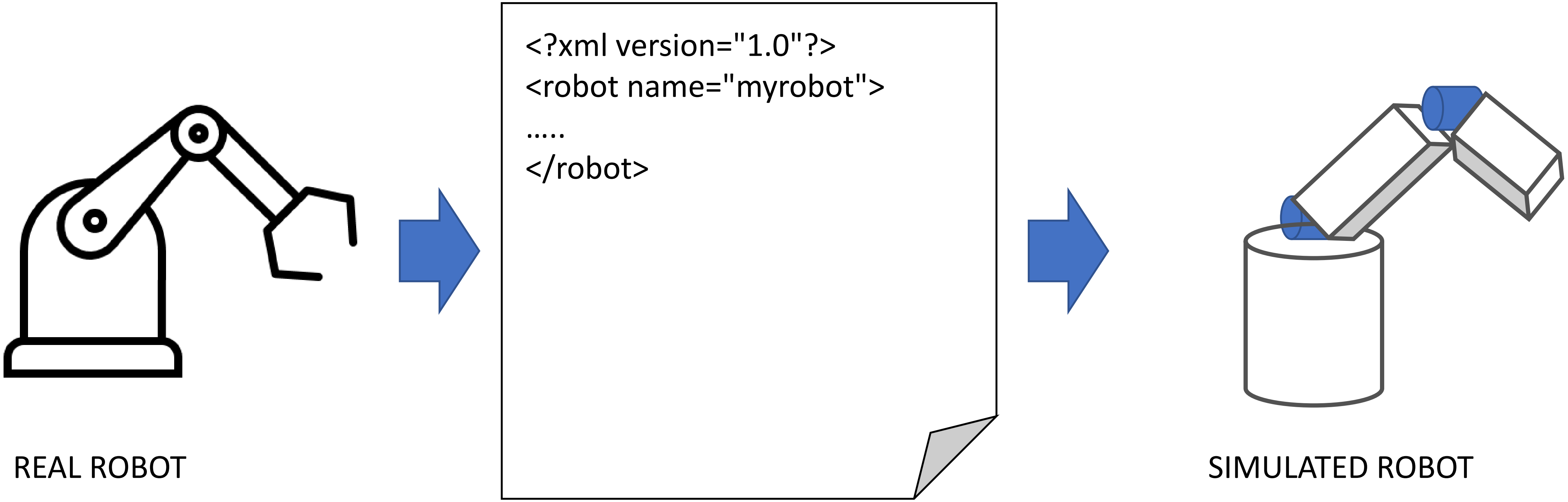
- 3D Visualisation tool
- Uses information to describe what the world around could be.
- It can display sensor information

Gazebo

- Robot simulator
- Interacts with the world as in reality (gravity, friction, etc)
- Can add plugins (describe sensor behaviours interacting with the world)

Robot modelling

Overview



```
<?xml version="1.0"?>
<robot name="myrobot">
.....
</robot>
```

REAL ROBOT

URDF file

SIMULATED ROBOT

Gazebo and Rviz Comparison

Models: URDF vs SDF Model

URDF Model

(Universal Robot Description Format)

- ROS compatible (RVIZ)
- Gazebo compatible with tags
- Xacro compatible
- Describes only robots

SDF Model

(Simulation Description format)

- Gazebo compatible only
- Can describe robots, 3D objects and 3D worlds
- Can add plugins (describe sensor behaviours)

Gazebo and Rviz Comparison

Models: URDF vs SDF Model

URDF Model

(Universal Robot Description Format)

- ROS compatible (RVIZ)
- Gazebo compatible with tags
- Xacro compatible
- Describes only robots

SDF Model

(Simulation Description format)

- Gazebo compatible only
- Can describe robots, 3D objects and 3D worlds
- Can add plugins (describe sensor behaviours)

Conclusion: You probably need both

Robot modelling

ROS Puzzlebot files.

Puzzlebot model files

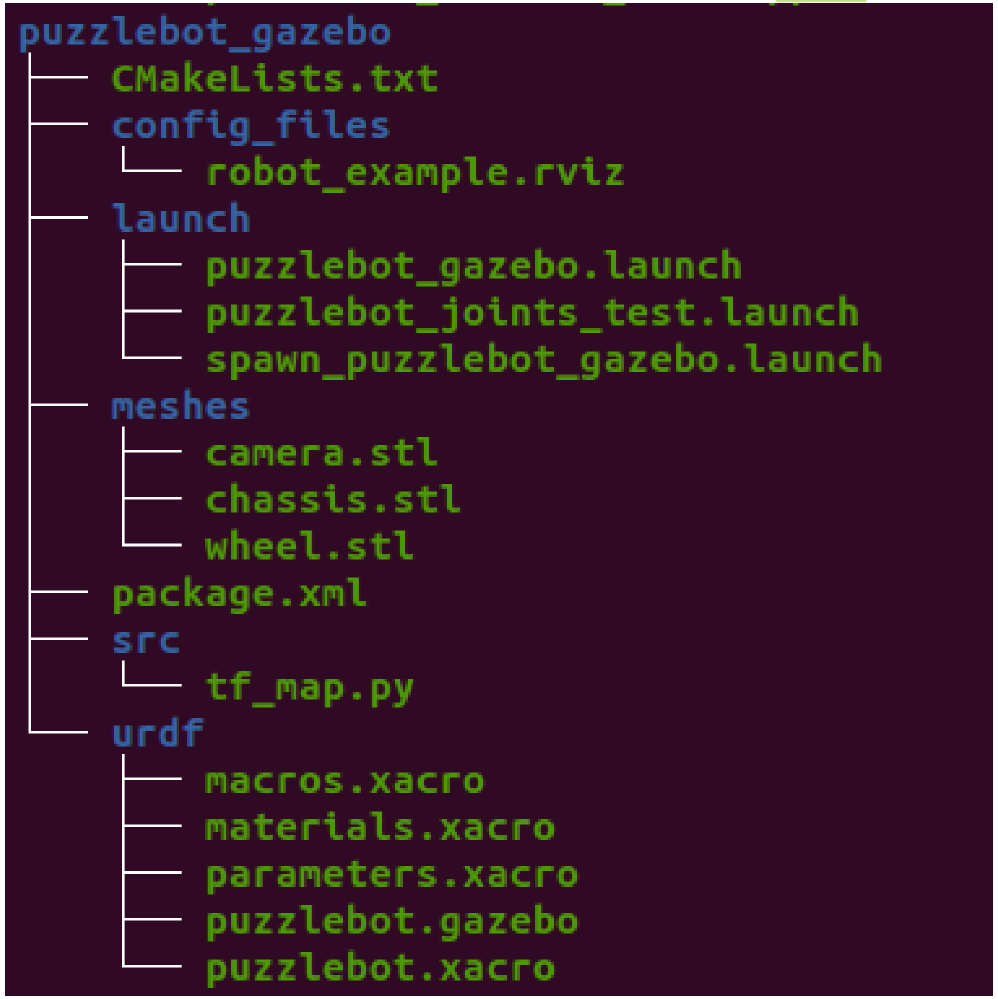
```
catkin_ws
    ├─ puzzlebot_gazebo – (or description) contains robot model description files.
    ├─ puzzlebot_world – contains files world description and models files
    ├─ puzzlebot_control – contains robot control files.
```

Robot modelling

ROS Puzzlebot files.

Puzzlebot model files

```
catkin_ws
    ├── puzzlebot_gazebo
    ├── puzzlebot_world
    └── puzzlebot_control
```



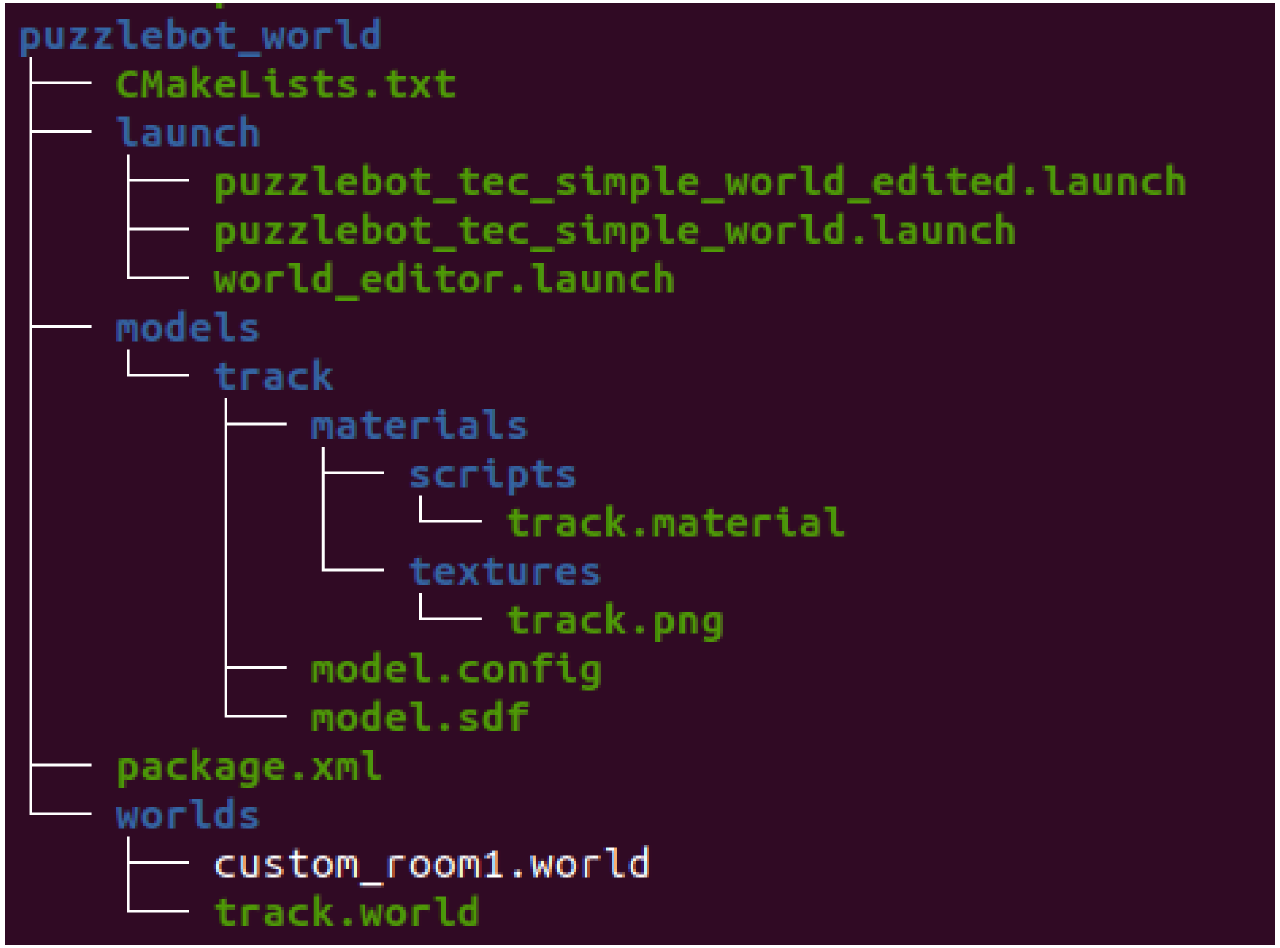
```
puzzlebot_gazebo
├── CMakeLists.txt
├── config_files
│   └── robot_example.rviz
├── launch
│   ├── puzzlebot_gazebo.launch
│   ├── puzzlebot_joints_test.launch
│   └── spawn_puzzlebot_gazebo.launch
├── meshes
│   ├── camera.stl
│   ├── chassis.stl
│   └── wheel.stl
├── package.xml
├── src
│   └── tf_map.py
└── urdf
    ├── macros.xacro
    ├── materials.xacro
    ├── parameters.xacro
    ├── puzzlebot.gazebo
    └── puzzlebot.xacro
```

Robot modelling

ROS Puzzlebot files.

Puzzlebot model files

```
catkin_ws
   ├── puzzlebot_gazebo
   ├── puzzlebot_world
   ├── puzzlebot_control
```

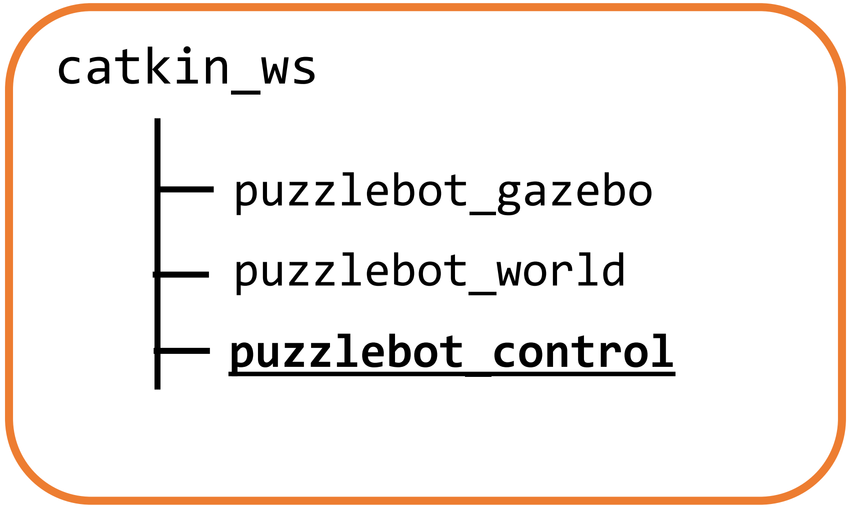


```
puzzlebot_world
├── CMakeLists.txt
├── launch
│   ├── puzzlebot_tec_simple_world_edited.launch
│   ├── puzzlebot_tec_simple_world.launch
│   └── world_editor.launch
├── models
│   └── track
│       ├── materials
│       │   ├── scripts
│       │   │   └── track.material
│       │   └── textures
│       │       └── track.png
│       ├── model.config
│       └── model.sdf
├── package.xml
└── worlds
    ├── custom_room1.world
    └── track.world
```

Robot modelling

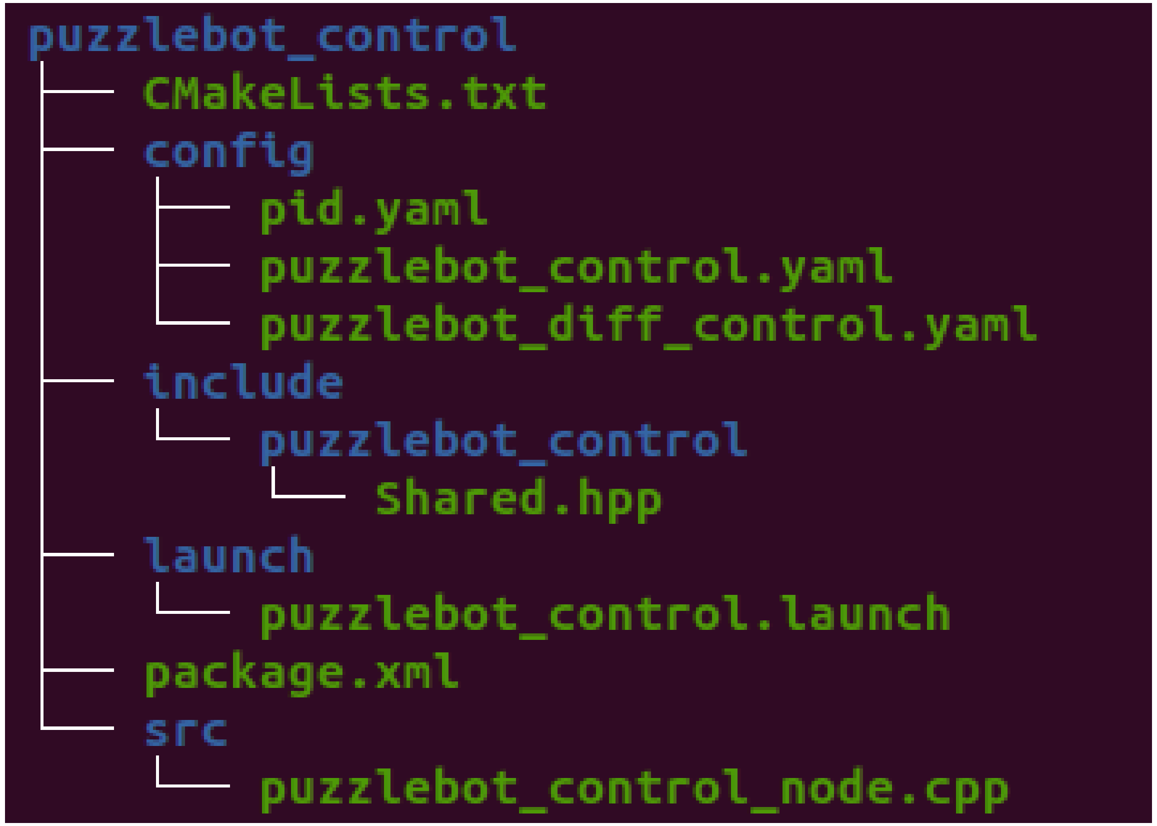
ROS Puzzlebot files.

Puzzlebot model files



```
catkin_ws
    ├── puzzlebot_gazebo
    ├── puzzlebot_world
    └── puzzlebot_control
```

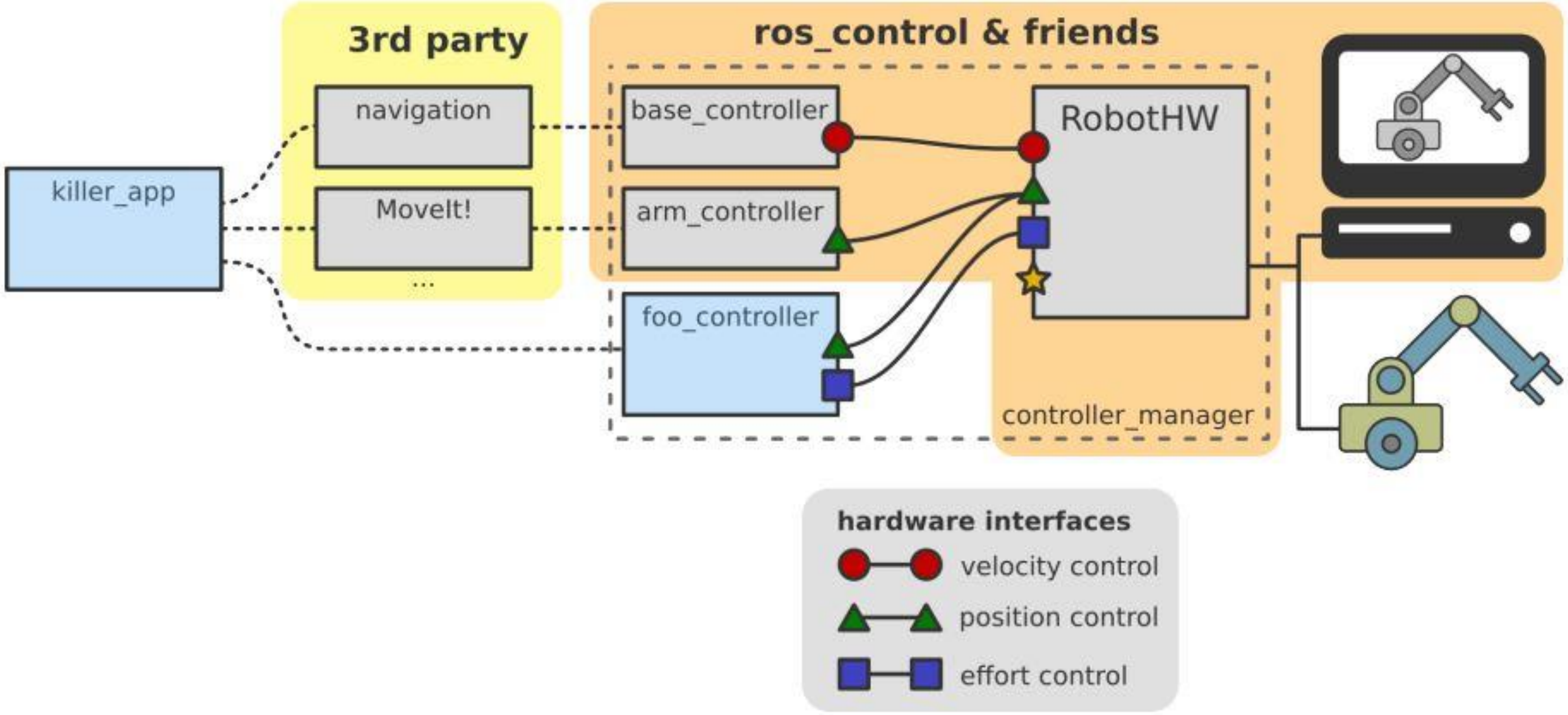
These files are here for reference only. They could vary depending on the controller used.



```
puzzlebot_control
├── CMakeLists.txt
├── config
│   ├── pid.yaml
│   ├── puzzlebot_control.yaml
│   └── puzzlebot_diff_control.yaml
├── include
│   └── puzzlebot_control
│       └── Shared.hpp
├── launch
│   └── puzzlebot_control.launch
├── package.xml
└── src
    └── puzzlebot_control_node.cpp
```

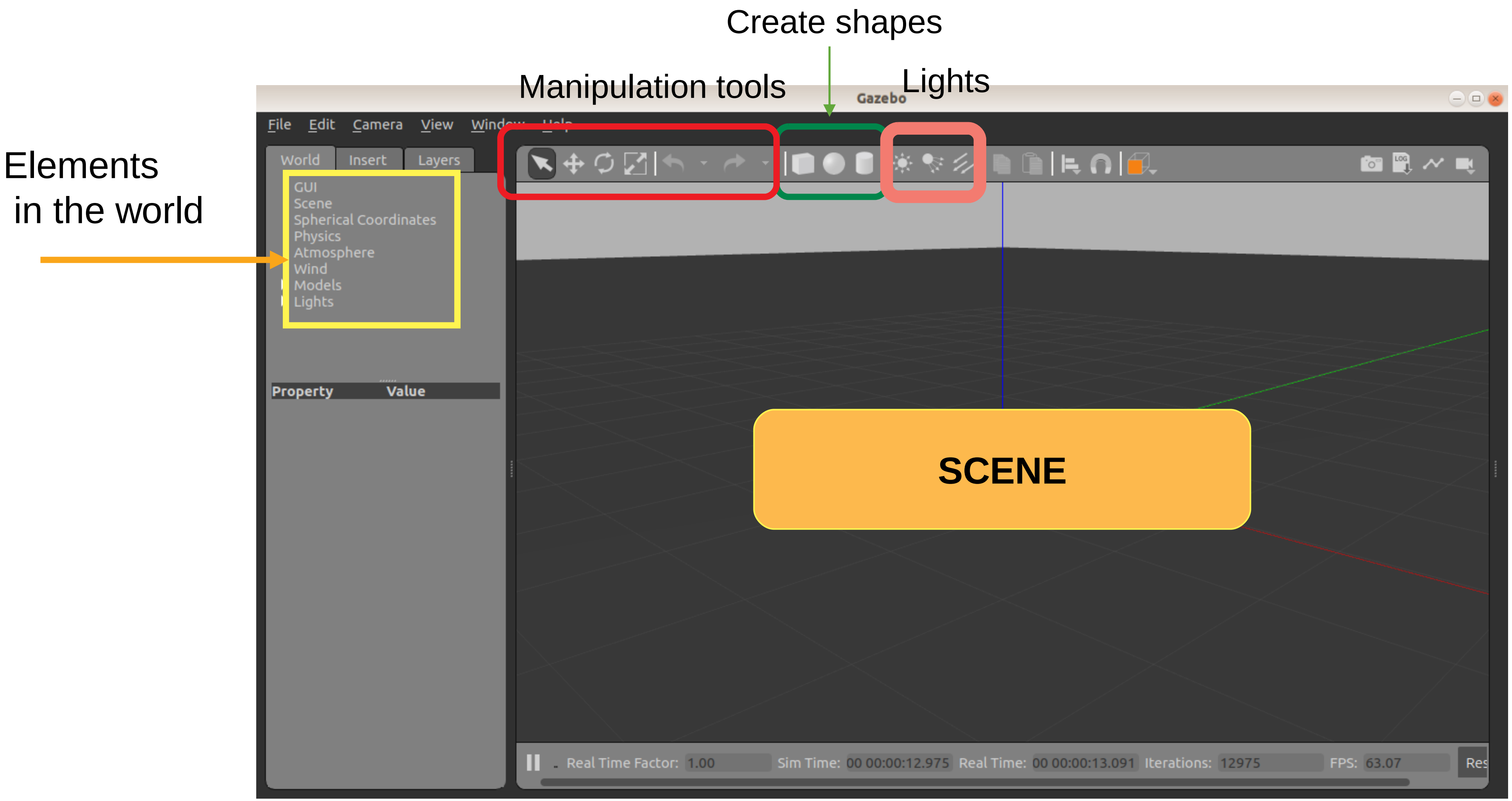
ROS Control

Overview



Gazebo

Interface Introduction



Create shapes

Manipulation tools

Lights

Elements in the world

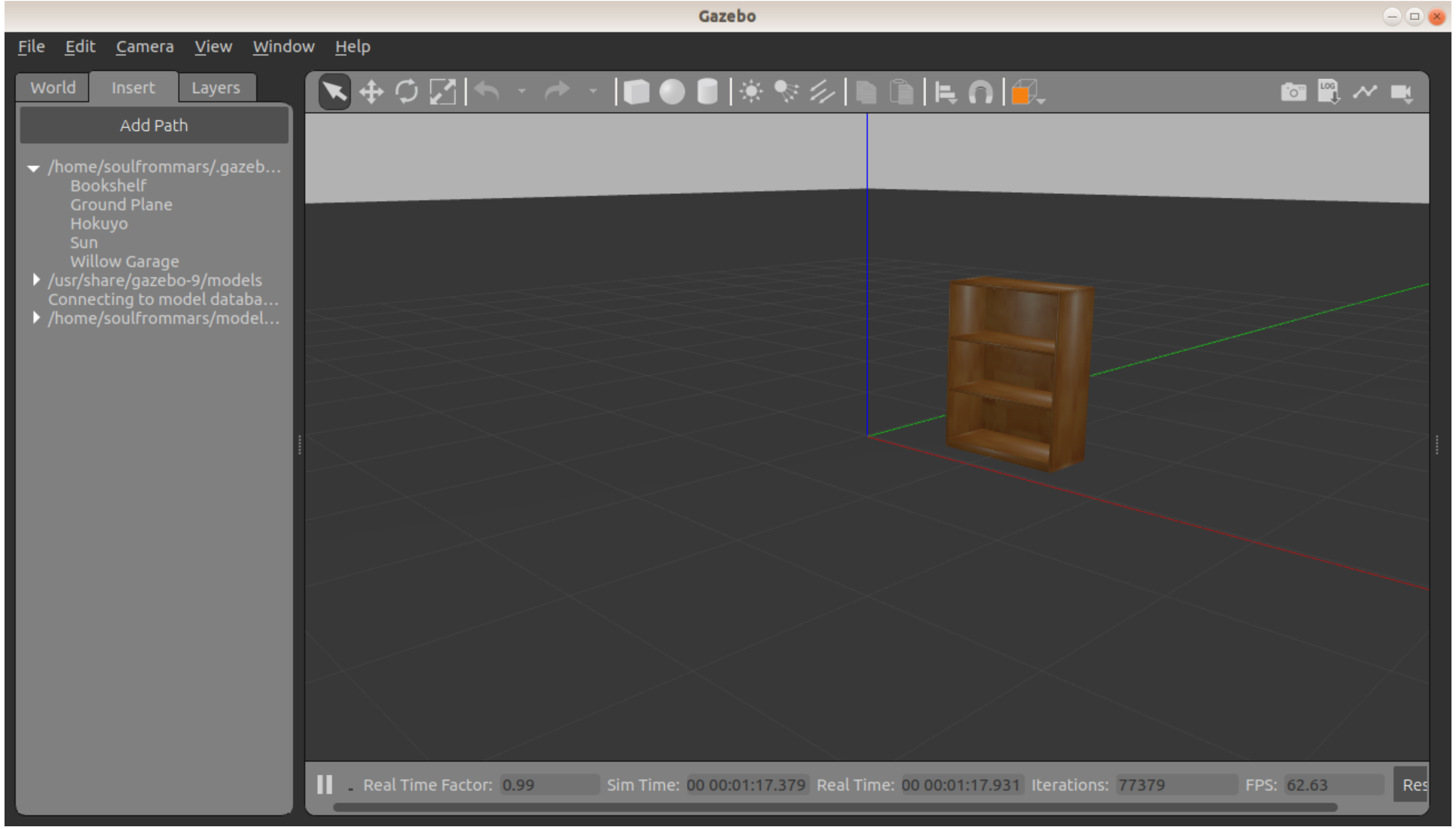
SCENE

Real Time Factor: 1.00 Sim Time: 00 00:00:12.975 Real Time: 00 00:00:13.091 Iterations: 12975 FPS: 63.07

More details in [here]

Gazebo

How to spawn a library model


Gazebo
File Edit Camera View Window Help
World
Insert
Layers
Add Path
/home/soulfrommars/.gazeb…
Bookshelf
Ground Plane
Hokuyo
Sun
Willow Garage
/usr/share/gazebo-9/models
Connecting to model databa…
/home/soulfrommars/model…
Res

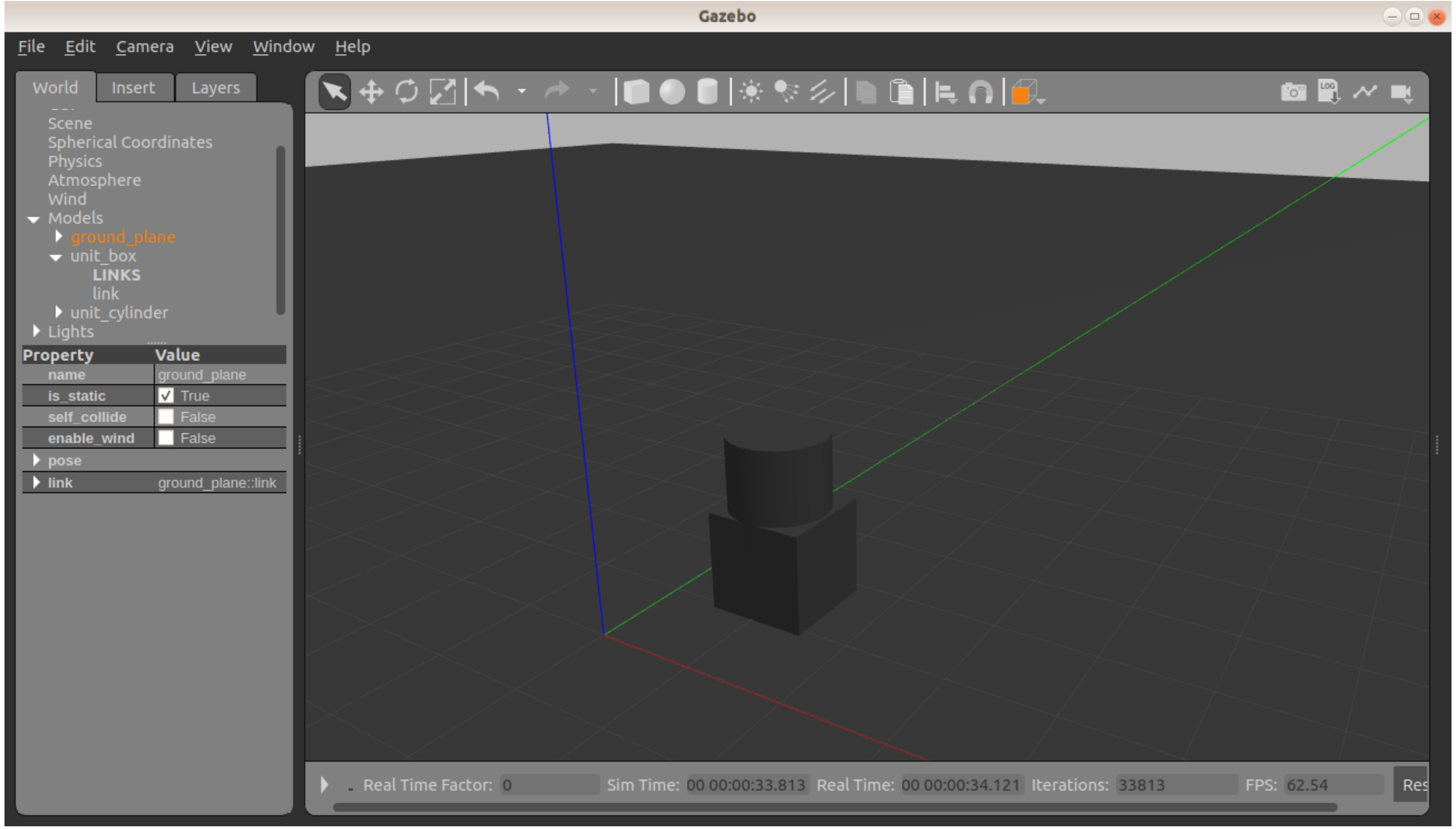
Real Time Factor: 0.99 Sim Time: 00 00:01:17.379 Real Time: 00 00:01:17.931 Iterations: 77379 FPS: 62.63

In the second tab, Objects from the gazebo Library are found.

Some other files can be added to a local library. Usually in home/.gazebo folder, but if required a model saved in the package can be loaded with a launch file.

Gazebo

How to create an object



Property	Value
name	ground_plane
is_static	☑ True
self_collide	□ False
enable_wind	□ False
pose	
link	ground_plane::link

Real Time Factor: 0 Sim Time: 00 00:00:33.813 Real Time: 00 00:00:34.121 Iterations: 33813 FPS: 62.54

It is possible to create models, links, or simple objects via the tools and Also the model editor.

The important part is to remember this is all saved in SDF (a gazebo format).

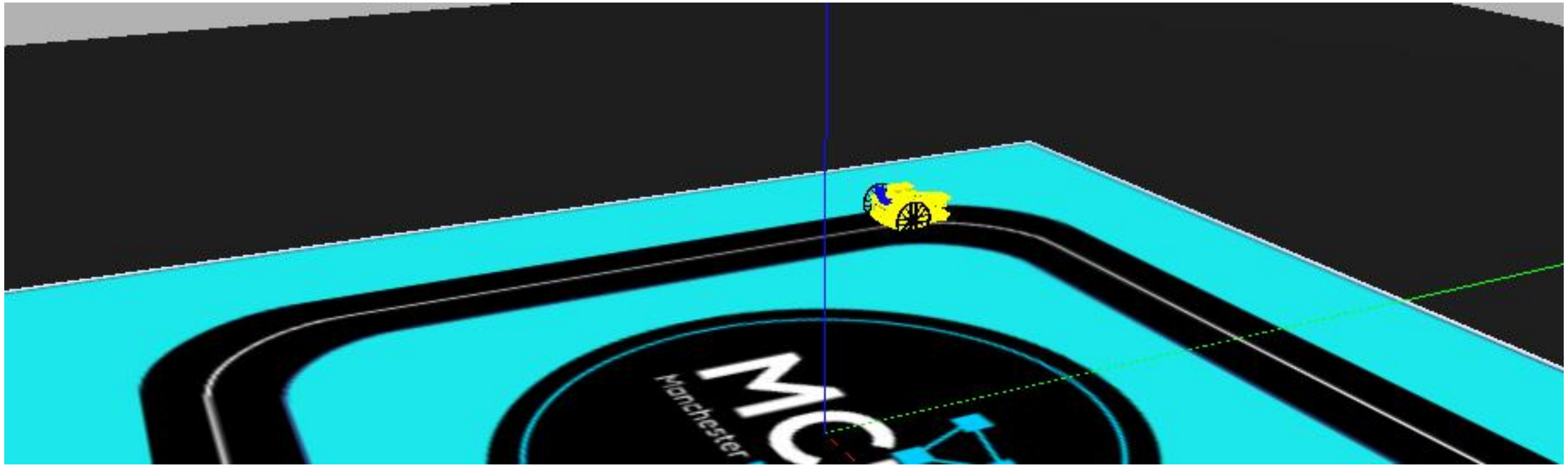
Also, the model editor cannot
Edit models or objects done Using URDF or Xacro.

Activity 1

Spawn Puzzlebot model in Gazebo.

- Download the folders from the GitHub repository for the lecture.
- Add files to your catkin workspace, and compile them.
- Use the launch file to launch your robot

```
roslaunch puzzlebot_world puzzlebot_tec_simple_world.launch
```



Activity 2

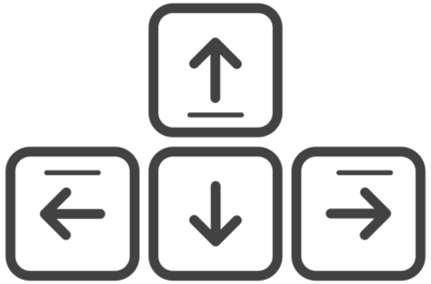
Teleoperate a PuzzleBot

- Install the teleop package using:

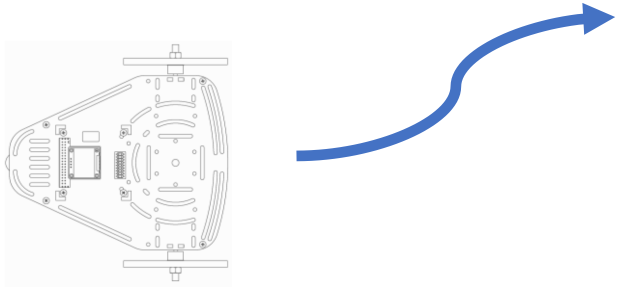
```
sudo apt-get install ros-noetic-teleop-twist-keyboard
```

- Run

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```



Input in the terminal the commands to move the robot



See how the robot traverses accordingly the environment.

Activity 3

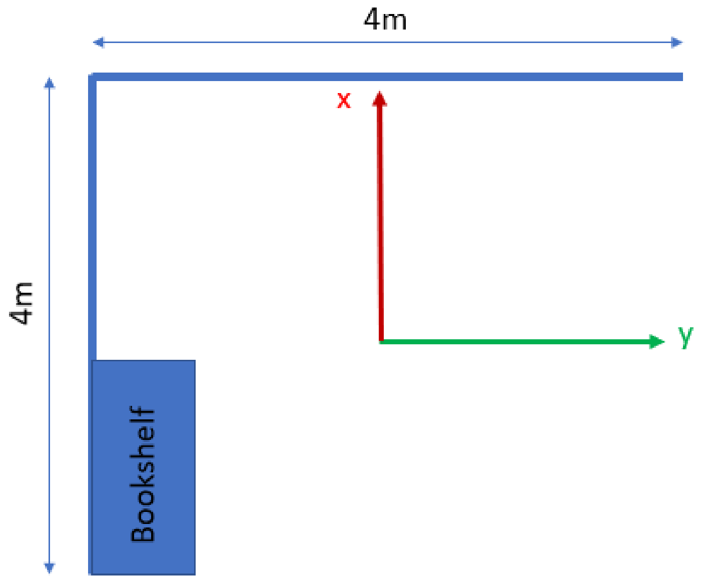
Spawn Puzzlebot model in your world.

- Open Gazebo using the command:

```
gazebo
```

- Using the interface, create a world with the information described in diagram 1.
- Save the world with the name “custom_room1.world”.
- Use the following command to spawn the puzzle in the world you saved.

```
roslaunch puzzlebot_world puzzlebot_tec_simple_world_edited.launch
```



4m
4m
x
y
Bookshelf