World and Launch Documentation

Alexander Greus, Kenneth Thompson, Dylan Zemlin, Spencer Smith

September 26, 2025

World File

This file specifies bunch of information about the map that we made for this project. Most importantly it specifies the model that we created to simulate the given assignment map specification. The model section lists all the walls that make it up, their positions/'poses' and their size. It also lists the collision separately from their visuals, which though its probably not as important for our model, since its relatively simple, it probably is important to separate visual and colliding geometry to optimize their representations for each. As well it also specifies the ground plane model. We initially ran into a humorous bug when we first tried running the turtle bot on our map: we had forgotten to put a ground plane and immediately the turtle bot began plummeting into infinity. For all the models in our world file there also parameters that we didn't really mess with, such as friction or visual material. As well, there are parameters for the world that seem very interesting but that we did not change, such as gravity or magnetic field. There are also parameters for physics and rendering plugins, which we did not take advantage of. With regards to the actual model for the assigned map specification, we used Gazebo's building editor. We unfortunately were not able to find a way to change the units of the editor, so we converted all measurements in feet to meters.

Launch File

To launch our project:

```
roslaunch project1_turtlebot project1.launch
```

Our launch file ended up being pretty simple since we were able to reuse or 'include' the launch files from the other turtle bot modules. It begins with setting up the config arguments, most importantly the world file matching the world file we created. It also sets up some other turtle bot configurations, such as the base model, battery, etc. The launch file also includes a GUI argument, since we were having issues initially bringing up the GUI's for Gazebo and RVIZ initially. The launch file then starts up Gazebo with our world file. Next, we launch the gmapping package for the mapping capabilities. We then spawn our turtle bot into the gazebo world, using the arguments we discussed previously. The next launched node is the 'robot_state_publisher' which, as the name may suggest, publishes any transform state about the robot to the /tf2 topic, for use by other nodes. For visualizing the mapping the robot is doing, we also launch rvis with the view_navigation launch file. Moving onto nodes needed for the turtle bot's functionality, we begin with the laser scanning node: 'depthimage_to_laserscan'. This is not a real laser, merely a remap of the depth image sensor to a laser scan posted to /scan. We also provide some parameters to the laser node, specifying the scan height, the image from which we are remapping, and the minimum range for the scan. We use the laser scan as our obstacle avoidance sensor. Next we set up the keyboard teleop node. This allows manual, user operation of the robot, though it does make the console you launch from quite ugly, as it is not only reporting logging from all the nodes, but also the Teleop's input. We decided to go this route just so everything could be launched from one file, instead of requiring the user to launch the teleop node themselves. And finally, we start our own node that contains all of the programmed behavior, that is, to halt on collisions, to escape from symmetric and asymmetric obstacles, to turn randomly ever 1 for of forward movement, and to drive forward. The other behaviors, namely mapping and teleop, of the robot were not programmed by us but provided by nodes launched, alongside our node.