Rviz Simulator Guidelines

The Rviz simulator is a method of testing python scripts and autonomous driving schemes, just like Gazebo. The difference is that Rviz is just an alternate visualization for simulating the car. Gazebo was the simulator used in the older versions of the F1tenth community, and Rviz is now the simulator used instead.

**Installation guidelines:**

1. Setup the new F1tenth workspace as directed here: <https://f1tenth.readthedocs.io/en/stable/getting_started/driving/drive_workspace.html#workspace-setup>
2. Install the simulation repository as directed here: <https://f1tenth.readthedocs.io/en/stable/going_forward/simulation/index.html>
3. Run the simulation test: $ roslaunch f110\_simulator simulator.launch

**Installation Notes:**

* The installation guidelines assume ROS and Rviz are installed.
* When the *catkin\_make* command is run, it must complete (it will show 100%) with no errors. If you encounter CMake errors with regard to packages, then some initial packages may not have been included in installation. Follow these steps to fix this (<https://answers.ros.org/question/288091/catkin_make-error-with-geometry2-package/> ):
  + Install rosdep: $ sudo apt-get install python-rosdep
  + In your workspace directory (f110\_ws) run this command to install missing packages: $ rosdep install --from-paths src --ignore-src -r -y
  + In the same directory, run: $ catkin\_make (this process should be completed successfully now)
  + Test the simulation with step 3 of the installation guidelines
* During catkin\_make you may also receive a message about duplicate folders (particularly the waypoint\_logger and cartographer\_config folders). Simply delete this folder from the directory as it is not important for the simulation to run.
* When the simulation is launched, IGNORE the following error that occurs:

[ERROR] [1589300465.064663919]: Couldn't open joystick /dev/input/js0. Will retry every second. This error occurs because we do not have a game controller (joystick) plugged in. The simulation will still work without this controller.

Once installed you can run the simulation and use keyboard input, game controller, or autonomous scripts to control the car, just follow the slide show here: <https://f1tenth.org/learn.html>

* For keyboard control:
  + Launch the simulation as in Step 3 of the installation guidelines
  + In the terminal used to launch, type “k” on the keyboard
  + Use the “wasd” keys on the keyboard to control the car and the car should start moving
  + If car collides with wall, simulation will pause and can be unpaused by moving the car to another position (use the 2D Pose Estimate tool in the top toolbar of RViz)

**Worlds**

In Gazebo, worlds are created using 3D models of walls and obstacles in \*.dae files. In Rviz, worlds are much simpler to create. Rviz worlds are 2D images in \*.pgm format. Refer to the examples in your simulator directory: /home/user/your\_ws/src/f1tenth\_labs/f110\_simulator/maps. You will see the following to note:

* An image in \*.pgm format
  + Simply a 2D drawing with black lines as walls and light gray/white for path
* A corresponding \*.yaml file sharing the same name
  + This file is used to call the image file and initial view settings of the world
  + **NOTE:** The “porto” track .yaml file needs to be fixed by changing the first line from map.pgm to porto.pgm. This will cause problems if you try to use this world and was likely a mistake by the creators of these example files.

What needs to be investigated: how to set the remaining parameters in the .yaml file for best viewing of the simulation.

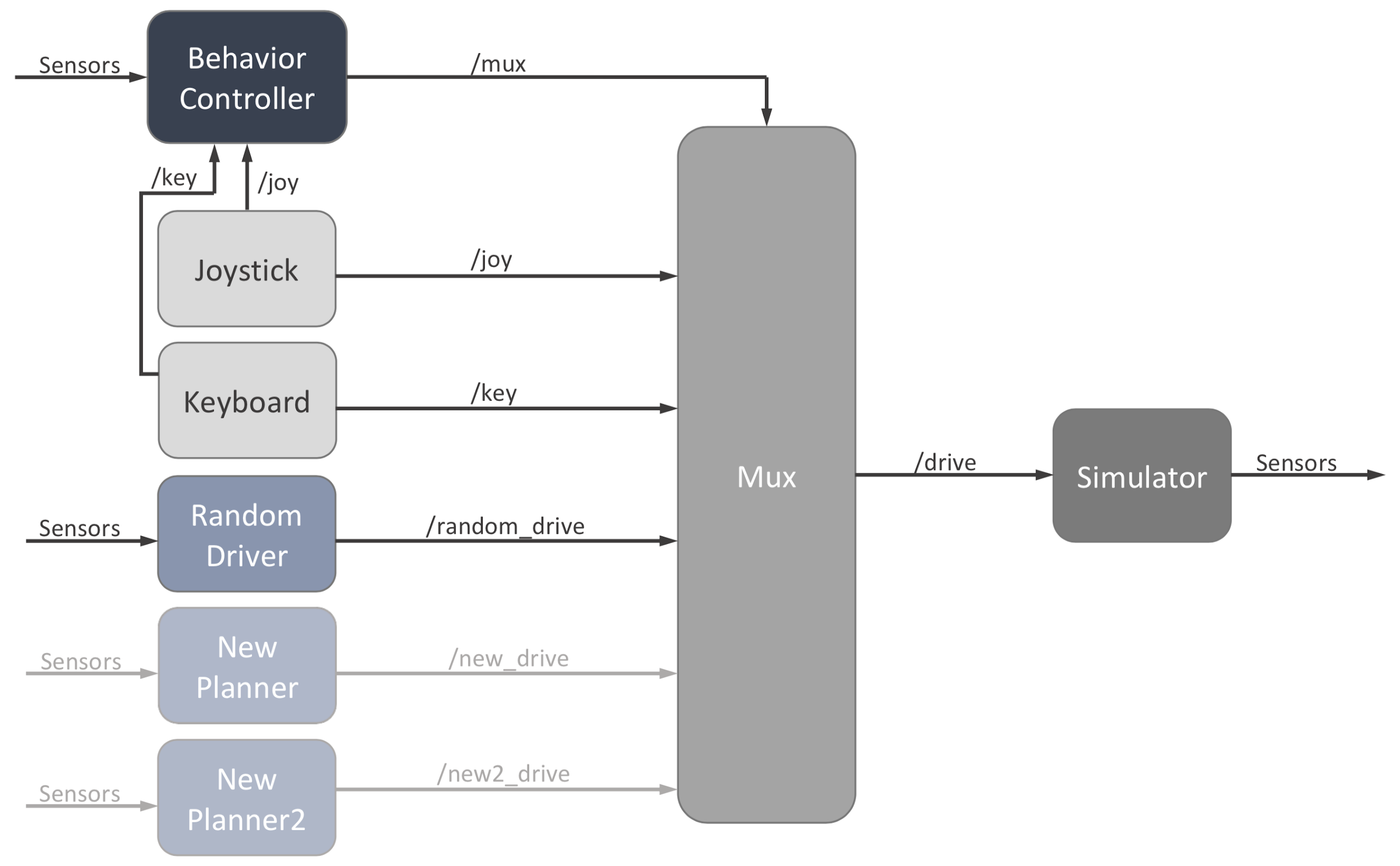
Run Examples:

If you want to open different worlds as an example, you need to edit the \*.yaml file name in the default arg field of the simulator.launch file: /home/user/your\_ws/src/f1tenth\_labs/f110\_simulator/launch.

**Configuring the Simulation.**

Please read and try to understand all of the modules and tutorials covered on the following webpage: <https://f1tenth.org/learn.html>. There, the layout and operation of the RViz simulator is laid out. Here, the same information is explained in less detail but in a different way that may be helpful.

Below is an image of how the default topics and nodes of the simulator are structured.



The column of nodes on the left are controllers and planners. The auxiliary controllers (joystick, and keyboard) send drive commands to the mux, as well as control commands to the behavior controller. The behavior controller is the node that tells the mux which drive message to pass to the simulator. The behavior controller listens to the keyboard and joystick to determine which drive message to pass. For example, when you start the simulation and press “k” on the keyboard, the behavior controller tells the mux to passed keyboard commands (WASD) to drive the car. This works because every node on the left is mapped to a keyboard and joystick input for you to be able to select them. The image above also shows sensor data being read by the behavior controller; this is because more complex algorithms can allow you to automatically choose different driving schemes based on what you see at the sensors.

The blue nodes in the image above are the drive planners (or driving schemes/algorithms). This is where the wall\_following, obstacle\_avoidance, pure\_pursuit, localization, etc. driving schemes will be. These nodes receive sensor data (like LiDAR) and publish their drive message to the MUX. If the Mux is told by the behavior controller to pass that node’s message, then that drive message will be used to operate the car.

The f1tenth site provides guidelines on how to implement a new planner. To add a new planner, the following files need to be edited in the /home/user/your\_ws/src/f1tenth\_labs/f110\_simulator/node path (the following files have comments indicating these required sections):

* Behavior\_controller.cpp
  + You will see a section where you need to add a new keyboard character that refers to your planner
* Mux\_controller.cpp
  + You will see a section where you need to add the planner node

And in the params.yaml file in the /home/user/your\_ws/src/f1tenth\_labs/f110\_simulator/ path, where you will see a line for adding a “new\_key\_char” which is the keyboard key for you new planner.

**Running your simulation code on the car**

As noted on the F1tenth website, you should be able to directly use your drive messages that are used for simulation on the real car. There is not much documentation available regarding this procedure but here are some ideas on how to do this:

* Utilize the teleop.launch file in the following path and implement your nodes there: /home/pguarner123/F1tenth\_dir/f110\_ws\_new/src/f110\_system/racecar/racecar/launch
  + Note that you will need to learn how the mux for this (which may be different from the simulation mux) works and how you need to edit it to make sure the drive messages are sent to the car.
* Make a new node to use instead of the simulator node to publish messages directly to the car from the simulator MUX.
* Make a new launch file that utilizes the teleop.launch and simulator MUX/nodes only.