TurtleSim Adventures

GT MRG

Manuel Roglan

09/18/2022

Part 1: Running TurtleSim

Check out the TurtleSim tutorial:

https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Introducing-Turtlesim/Introducing-Turtlesim.html

You only need to work through step 3 for this challenge, though feel free to finish the full tutorial if interested.

In step 3, you are asked to run `ros2 node list` and `ros2 topic list`. Make sure that you see the `/turtlesim` node when you run the former and `/turtle1/cmd_vel` when you run the latter.

We are now leaving the TurtleSim tutorial.

With TurtleSim running, in a separate terminal run `ros2 topic info /turtle1/cmd_vel`. Running `ros2 topic info <topic>` gives you important information about the topic. As you can see, the type of message the topic can send and receive is `geometry_msgs/msg/Twist`.

A quick google search of a message type will give you information about it. Here is the documentation for `geometry_msgs/msg/Twist`:

http://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/Twist.html

Part 2: A Square

Now, we want to drive our turtle in a square (forever). The `/turtlesim` will listen to the topic `/turtle1/cmd_vel` and use data published there to move the turtle on the screen. So, if we want the turtle to move in a square, we need to change its velocity in a coordinated way.

We know from Part 1 that the data we need to send to `/turtle1/cmd_vel` is of type `geometry_msgs/msg/Twist`. We need to now create a node that will publish a velocity to this topic and thus drive the turtle. Create this node in a new package.

When you are ready to test your node, make sure to build the package and source it before trying to run the node. **You must build and source every time you make a change to the node.**

Good luck!

Hints:

1. You can import the Twist message in your code like this: from `geometry_msgs.msg import Twist`
2. The turtle won't keep going in the direction you send it forever, so you will need to constantly send it the velocity you want it to go at, maybe a timer would be useful?
3. Within your node, you can publish messages to the terminal by writing `self.get_logger().info("My message")`. This can be useful for debugging.
4. Make sure TurtleSim is running while you test your node, or else you'll just be sending a velocity to nobody!

Part 3: Counting Squares

We want to be able to know how many squares our turtle has done. While this could be done within the node we just created, let's do this in a separate node. Create a new node in the same package you are working in.

From this node, subscribe to the `/turtle1/cmd_vel` topic. You will need this information to determine when the turtle has completed a square. Publish how many squares the turtle has completed to a topic called `/square_count`.

Good luck!

Hints:

1. What data type should you publish to `/square_count`? Maybe one of these will work: https://docs.ros2.org/foxy/api/std_msgs/index-msg.html
2. Do we need to know the location of the turtle on the screen for this, or can we just assume it starts at some arbitrary point, like (0, 0)?
3. In a new terminal, you can run `ros2 topic echo /square_count` to see what you are publishing.

Part 4: All Together

Wow, so that took 3 terminals just to run all our nodes, and a 4th to see our square count data! We can use a launch file to run all these nodes at the same time, from one terminal.

Create a launch file that launches TurtleSim, your node from Part 2, and your node from Part 3.

Good luck!

Hints:

1. Launching nodes tutorial: https://docs.ros.org/en/foxy/Tutorials/Beginner-CLI-Tools/Launching-Multiple-Nodes/Launching-Multiple-Nodes.html