Laboratory assignment 1

ROS: Introduction to Turtlesim

Name: Sérgio Miguel Rosa Estêvão **JMBAG:**

Preparation

- Review the ROS lecture slides.
- Read the documentation of the packages `turtlesim` and `teleop_twist_keyboard` on the ROS wiki.
- Remember that in order to use software based on ROS, either installed using the Ubuntu system package manager `apt` or built from source in a Catkin workspace, you need to run the appropriate environment setup script.

  The base ROS setup script `setup.bash` is located in `/opt/ros/<rosdistro>`. The distribution of ROS1 that we are using for this course is `noetic`.
- If you do not want to run the ROS environment setup script each time you open a new terminal, you can place it in the shell startup script (`.bashrc`) in your home directory.
- The application `terminator` (installable with the system package manager) can be useful in cases when there is a need to run many terminal-based applications at once. The application can split terminals horizontally and vertically, allowing you to organize them while having open as many terminals simultaneously as needed.

Assignments

Task: Introduction to Turtlesim

Besides answering the questions from the task themselves, make sure to include all commands that you used, in full.

a) Install the packages `turtlesim` and `teleop_twist_keyboard`. Which commands did you use to install the packages?

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

b) Start `roscore` and run the nodes `turtlesim_node` and `teleop_twist_keyboard.py` from the respective packages. (Note: *do not use* the `turtle_teleop_key` node for this assignment.)

When running these two nodes *without any additional options*, which topics do they publish and subscribe to? What are the topic names and types? (Remember to write all commands you used to examine this).

```
#Answer:

The /turtlesim node published the topics (<topic_name> [<topic_type>]):
 * /rosout [rosgraph_msgs/Log]
 * /turtle1/color_sensor [turtlesim/Color]
 * /turtle1/pose [turtlesim/Pose]
and subscribes to the topics:
 * /turtle1/cmd_vel [unknown type]

The /teleop_twist_keyboard node published the topics (<topic_name> [<topic_type>]):
```

c) As you might have noticed, pressing keys in `teleop_twist_keyboard.py` does not move the turtle, and the node is printing the message "`Waiting for subscriber to connect to...`".

What is wrong? How does the `turtlesim` node subscribe to velocity commands for the turtle, and how does `teleop_twist_keyboard.py` publish the velocity commands by default? What needs to be fixed?

Run `teleop_twist_keyboard.py` with the appropriate topic remapping so that controlling the turtle by keyboard works. Verify by driving the turtle around. Write the command you used in your answer.

#Answer:
The issue is that teleop_twist_keyboard.py is trying to publish velocity commands by default to the /cmd_vel topic, but there is no subscriber (listener) for that topic by default in the turtlesim environment, which typically subscribes to the topic /turtle1/cmd_vel.

To make the keyboard control work, the topic used by teleop_twist_keyboard.py should be remapped to the topic expected by turtlesim.

#Commands:

rosrun teleop_twist_keyboard teleop_twist_keyboard.py cmd_vel:=/turtle1/cmd_vel

d) Print and examine the contents of the velocity command messages published by `teleop_twist_keyboard.py` as you are driving the turtle.

What is the type of the velocity command message? Which ROS command can you use to *show* the full definition of this message type? Which component of the message contains the **forward** velocity, and which component contains the **angular** velocity?

#Answer:
The type of the velocity command message is geometry_msgs/Twist.

The command "rosmsg show <message_type>" can be used to show the full definition.

Considering this is the output of the command "rosmsg show geometry_msgs/Twist":
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x

e) In robotics, the *pose* of an object consists of the information about the object's position and object's orientation. Print and examine the contents of messages published by `turtlesim` which contain the current turtle pose.

What is the possible range of turtle's **position**? What are the values of the position coordinates at the corners of the simulator stage (lower left, lower right, upper left and upper left)?

#Answer:
Considering the default values of the turtlesim environment window, the possible range of turtle positions is the limits of the window, which are:
>lower-left: (0,0)
>lower-right: (10,0)
>upper-left: (0,10)
>upper-right: (10,10)

So, the turtle's possible positions are:
>x: [0, 10]
>y: [0, 10]

f) Inspect the ROS services made available by the `turtlesim` node.

The turtle has a *pen*, which is drawing a coloured trail as the turtle is moving. What is the name of the ROS service for configuring the turtle's pen?

Search the web for "color picker" to select your favourite color and obtain the RGB values for it. Which pair of commands can you use to disable turtle's pen, and then later re-enable it to make the turtle draw your favourite color?

Hint: by pressing the tab key during writing the `rosservice call` command, you can have ROS pre-fill the service call parameters message for you.

#Answer:
The ROS service for configuring the turtle's pen is typically named /turtle1/set_pen. You can use this service to change the pen color and other properties.

To disable the pen, can use the command "rosservice call /turtle1/set_pen 0 0 0 0 1", setting to True the "off" parameter of the call

To set the pen to a color, in this case dark blue, one should use "rosservice call /turtle1/set_pen 0 0 139 2 0"

#Commands:
##Inspect ROS services:

g) Using the commands from the previous subtask and `teleop_twist_keyboard.py`, try to draw your name. Save a screenshot called `myname.png`, like so:

Hint: there is also a ROS service you can use for clearing the background if you need to start again.

Hint: hold shift to strafe. Press any key (e.g. space) to stop. Angular velocity can be reduced; read the help text printed by `teleop_twist_keyboard.py`.

Hint: Use Alt + PrtSc to save a screenshot of a single window.

Assignment submission

Upload a zip file `lab1-yourjmbag.zip` containing `myname.png`, and this pdf file with the filled out answers, to Moodle.