Fundamentals of ROS

Carmine Tommaso Recchiuto

Our first ROS package

- Please find the package developed during last week here: https://github.com/CarmineD8/turtlebot_controller

You will find here also the python script implementing the same thing

Geometry_msgs is not present among the dependencies: how can we add it? We can manually modify the files CMakelists.txt and package.xml

Exercise 1

Starting from the package, try to perform the following exercise:

- Develop a new node that:
  - Move turtle1 forward, until it reaches the wall (x>10.5)
  - When this happens, go backward
  - When it reaches the center again, rotate the turtle of 90 degrees, and move it forward again until it reaches the top (y>11.0)
  - When this happens, go backward

Repeat this loop continuously!

Modify the CMakeLists.txt file (if needed) so as to build the program

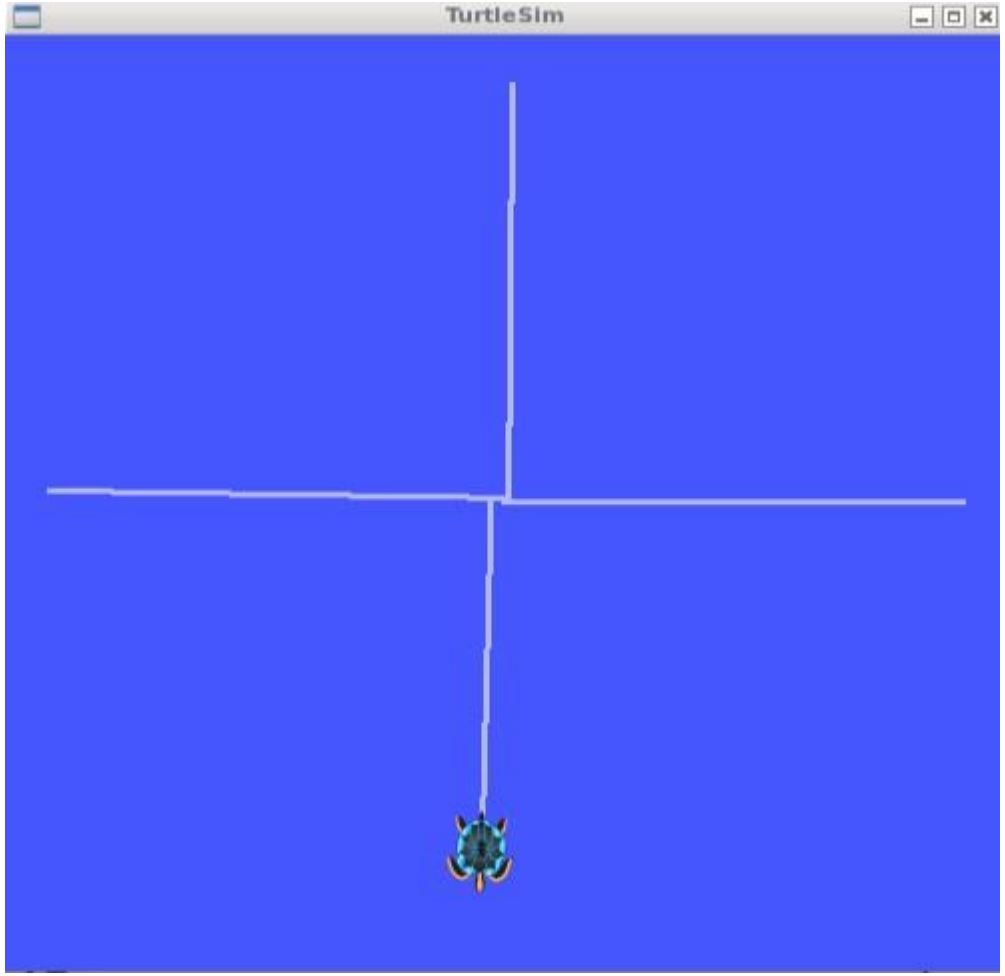
Exercise 1

Some hints:

- When you publish a message on cmd_vel, the command is executed in one second

- To make the robot rotate of 90 degrees, you can send an angular velocity of 1.57; however you will see that this is not precise (it’s however ok for the current implementation of the exercise)

- You can use some sleep functions in your code (in cpp, you will need to include unistd.h)



Services

- *The publish/subscribe model (i.e. topic/msgs)* is not always appropriate for RPC **request / reply interactions**, which are likely to occur in a distributed system

- *Request / reply* is done via a **Service,** which is defined with a pair of messages, one for the request and one for the reply.

Services

- A providing ROS node offers a service under a string name, and the client calls the service by sending a request message and awaiting the reply

- Practical usage: ***rosservice / rossrv***

- ***rosservice list*** *(list all the available services)*
- ***rosservice type*** *(gives info about the services)*

Services

- The details of the arguments needed from a service can be determined using ***rossrv show [service_type]***

- Example: in order to add a new turtle in a specific point in the environment call the **spawn** service

Example: *spawn* client

Objective: write a node that spawn a turtle in a certain position by using the service "spawn"

1) Check the type of the service (on the terminal ***rosservice type turtle1/spawn***)

2) In the .cpp file, Include the header "***turtlesim/Spawn.h***"

Example: *spawn* client

3)Create a client that sends a Spawn request to the /spawn service

```
ros::ServiceClient client1 =  nh.serviceClient<turtlesim::Spawn>(“/spawn”);
```

4) Check the structure of the service (on the terminal

```
rossrv show turtlesim/Spawn)
```

5) Define a Spawn service message

```
turtlesim::Spawn srv1;
```

Example: *spawn* client

6)Add the arguments of the message as defined in the structure of the service*:*

```
 srv1.request.x = 1.0;
srv1.request.y = 5.0;
srv1.request.theta = 0.0;
srv1.request.name = "my_turtle";
```

7) Call the service:

```
client1.call(srv1);
```

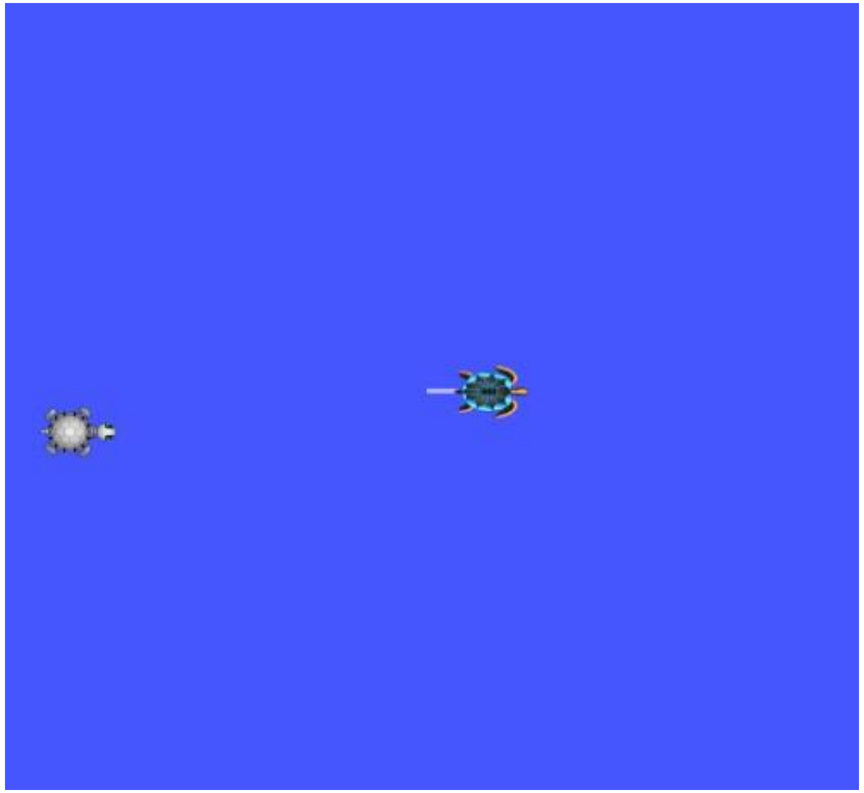
Example: *spawn* client in python

We can simply add the following lines to our code:

```
from turtlesim.srv import Spawn

…

client = rospy.ServiceProxy("/spawn", Spawn)
resp = client(1.0, 5.0, 0.0, "my_turtle")
```



Other ROS tools

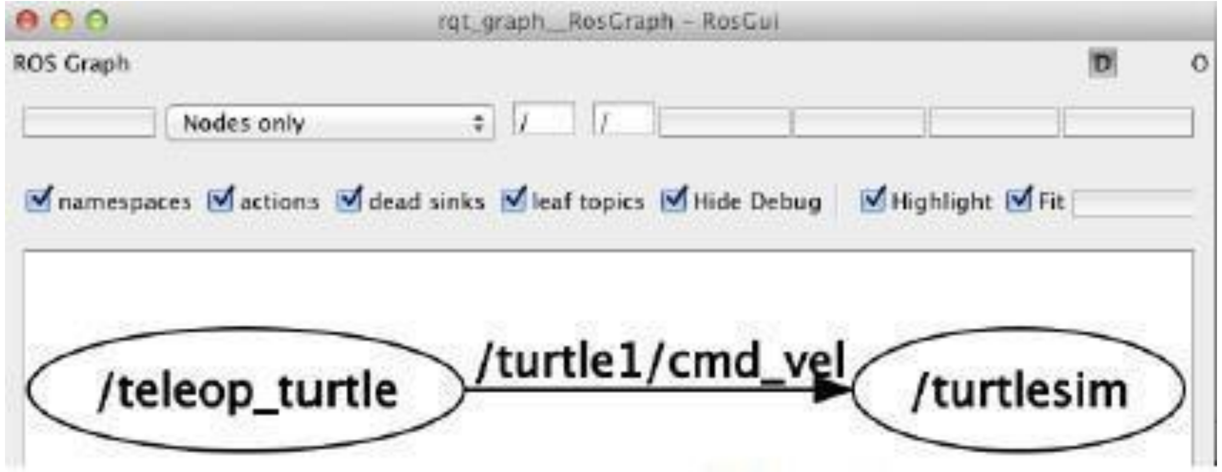
Topics and services can also be echoed and tested with the terminal before implementing the code:

- rqt
- rostopic echo / rostopic call
- rosservice call

Other ROS tools

- **Rqt_graph** shows a dynamic graph of what's going on in the system :

rosrun rqt_graph rqt_graph



rqt_graph__RosGraph - RosGui

ROS Graph

Nodes only

/ /

☑ namespaces ☑ actions ☑ dead sinks ☑ leaf topics ☑ Hide Debug ☑ Highlight ☑ Fit

Other ROS tools

- Record and play topics:

  rosbag record chatter

- It will create in the same folder a .bag file with all the messages published on that topic

- The info can be stored and played again in order to simulate the robot behaviour:

  rosbag play [filename].bag

Exercise 2

You can find a starting cpp file in the course material (Teams, Aulaweb) or download it from https://github.com/CarmineD8/ros_ex1

- ✓ Kill the turtle named **turtle1**
- ✓ Spawn a turtle named **rpr_turtle** in the position x = 2.0, y=1.0, theta=0.0
- ✓ Let the turtle move along x, until it reaches the end (x > 9.0)
- ✓ When x > 9.0 or x < 2.0, make it turn in a circular arc
- ✓ Continue until the turtle covers the whole area.
- ✓ Modify the CMakeLists.txt file (if needed) so as to build the program

Exercise 2