!

> **Note:** This tutorial assumes that you have completed the previous tutorials: Moving in a Straight Line (/turtlesim/Tutorials/Moving%20in%20a%20Straight%20Line).
>
> Please ask about problems and questions regarding this tutorial on answers.ros.org (http://answers.ros.org). Don't forget to include in your question the link to this page, the versions of your OS & ROS, and also add appropriate tags.

Rotating Left/Right

Description: This tutorial is based on Turtlesim Video Tutorials (/turtlesim/Tutorials)

Tutorial Level: INTERMEDIATE

Next Tutorial: Moving to goal (/turtlesim/Tutorials/Go%20to%20Goal)

Contents

You can find the complete package at: https://github.com/clebercoutof/turtlesim_cleaner (https://github.com/clebercoutof/turtlesim_cleaner)

Now, we are going to rotate the turtle.

1. Preparing for work

Let's create our file **rotate.py**(or any name you want) and paste it in the source directory of our package, if you followed the past tutorial it will be: ***~/catkin_ws/src/turtlesim_cleaner/src***. Then , don't forget to make the node executable:

```
$ chmod u+x ~/catkin_ws/src/turtlesim_cleaner/src/rotate.py
```

2. Understanding the code

Similar as the past tutorial code, we will receive the speed, distance and a variable wich defines if the movement is clockwise or counter-clockwise. Since we can just publish a velocity to the topic ***/turtle1/cmd_vel***, our logic will have to calculate the distance specified, but in this case it will be an **angular velocity**.

3. The Code

Toggle line numbers

```
 1 #!/usr/bin/env python
 2 import rospy
 3 from geometry_msgs.msg import Twist
 4 PI = 3.1415926535897
 5
 6 def rotate():
 7     #Starts a new node
 8     rospy.init_node('robot_cleaner', anonymous=True)
 9     velocity_publisher = rospy.Publisher('/turtle1/cmd_vel', Twist, queue
_size=10)
10     vel_msg = Twist()
11
12     # Receiveing the user's input
13     print("Let's rotate your robot")
14     speed = input("Input your speed (degrees/sec):")
15     angle = input("Type your distance (degrees):")
16     clockwise = input("Clockwise?: ") #True or false
17
18     #Converting from angles to radians
19     angular_speed = speed*2*PI/360
20     relative_angle = angle*2*PI/360
21
22     #We wont use linear components
23     vel_msg.linear.x=0
24     vel_msg.linear.y=0
25     vel_msg.linear.z=0
26     vel_msg.angular.x = 0
27     vel_msg.angular.y = 0
28
29     # Checking if our movement is CW or CCW
30     if clockwise:
31         vel_msg.angular.z = -abs(angular_speed)
32     else:
33         vel_msg.angular.z = abs(angular_speed)
34     # Setting the current time for distance calculus
35     t0 = rospy.Time.now().to_sec()
36     current_angle = 0
37
38     while(current_angle < relative_angle):
39         velocity_publisher.publish(vel_msg)
40         t1 = rospy.Time.now().to_sec()
41         current_angle = angular_speed*(t1-t0)
42
43
44     #Forcing our robot to stop
45     vel_msg.angular.z = 0
46     velocity_publisher.publish(vel_msg)
47     rospy.spin()
48
49 if __name__ == '__main__':
```

```
50     try:
51         # Testing our function
52         rotate()
53     except rospy.ROSInterruptException:
54         pass
```

First we need to import the packages used on our script.The rospy library is the ros python library, it contains the basic functions, like creating a node, getting time and creating a publisher.The geometry_msgs contains the variable type Twist that will be used, and we define a constant PI that will be required:

Toggle line numbers

```
2 import rospy
3 from geometry_msgs.msg import Twist
4 PI = 3.1415926535897
```

Now we declare our function, initiate our node, our publisher and create the **Twist** variable.

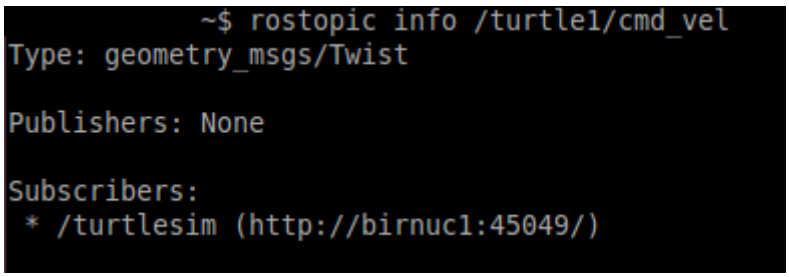
Toggle line numbers

```
6 def rotate():
7     #Starts a new node
8     rospy.init_node('robot_cleaner', anonymous=True)
9     velocity_publisher = rospy.Publisher('/turtle1/cmd_vel', Twist, queue
_size=10)
10     vel_msg = Twist()
```

The **Twist** is necessary because our topic **'/turtle1/cmd_vel'** uses the Twist message, you can check with the following command:

```
$ rostopic info /turtle1/cmd_vel
```

You should see the following screen:



The Twist message is composed by 3 linear components and 3 angular components,you can see the message description with the following command:

```
$ rosmsg show geometry_msgs/Twist
```

You should see the following screen:

```
          :~$ rosmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

The unit used by ROS is radians, so we have to convert the input from degrees to radians. Note that this **isn't necessary**, but for didactic purposes it's better to use the input on degrees, in the following statements the code does the conversion:

Toggle line numbers

```
18     #Converting from angles to radians
19     angular_speed = speed*2*PI/360
20     relative_angle = angle*2*PI/360
```

Since we are just rotating the turtle, we don't need the linear components, and , depending on the user's input we decide if the movement will be clockwise or counter-clockwise:

Toggle line numbers

```
23     vel_msg.linear.x=0
24     vel_msg.linear.y=0
25     vel_msg.linear.z=0
26     vel_msg.angular.x = 0
27     vel_msg.angular.y = 0
28
29     # Checking if our movement is CW or CCW
30     if clockwise:
31         vel_msg.angular.z = -abs(angular_speed)
32     else:
33         vel_msg.angular.z = abs(angular_speed)
```

Now , with the **rospy.Time.now().to_sec()**. we get the starting time **t0**, and the time **t1** to calculate the angular distance:

Toggle line numbers

```
34     # Setting the current time for distance calculus
35     t0 = rospy.Time.now().to_sec()
36     current_angle = 0
```

And while the actual distance is less than the user's input, it will keep publishing:

Toggle line numbers

```
38     while(current_angle < relative_angle):
39         velocity_publisher.publish(vel_msg)
40         t1 = rospy.Time.now().to_sec()
41         current_angle = angular_speed*(t1-t0)
```

After we get to the specified angle , we order our robot to stop:

Toggle line numbers

```
44     #Forcing our robot to stop
45     vel_msg.angular.z = 0
46     velocity_publisher.publish(vel_msg)
```

The following statement guarantees that if we press **ctrl+c** our code will stop:

Toggle line numbers

```
47     rospy.spin()
```

And then, we have our main loop which calls our function:

Toggle line numbers

```
49 if __name__ == '__main__':
50     try:
51         # Testing our function
52         rotate()
53     except rospy.ROSInterruptException:
54         pass
```

Now, you can test and move your robot.

4. Testing the code

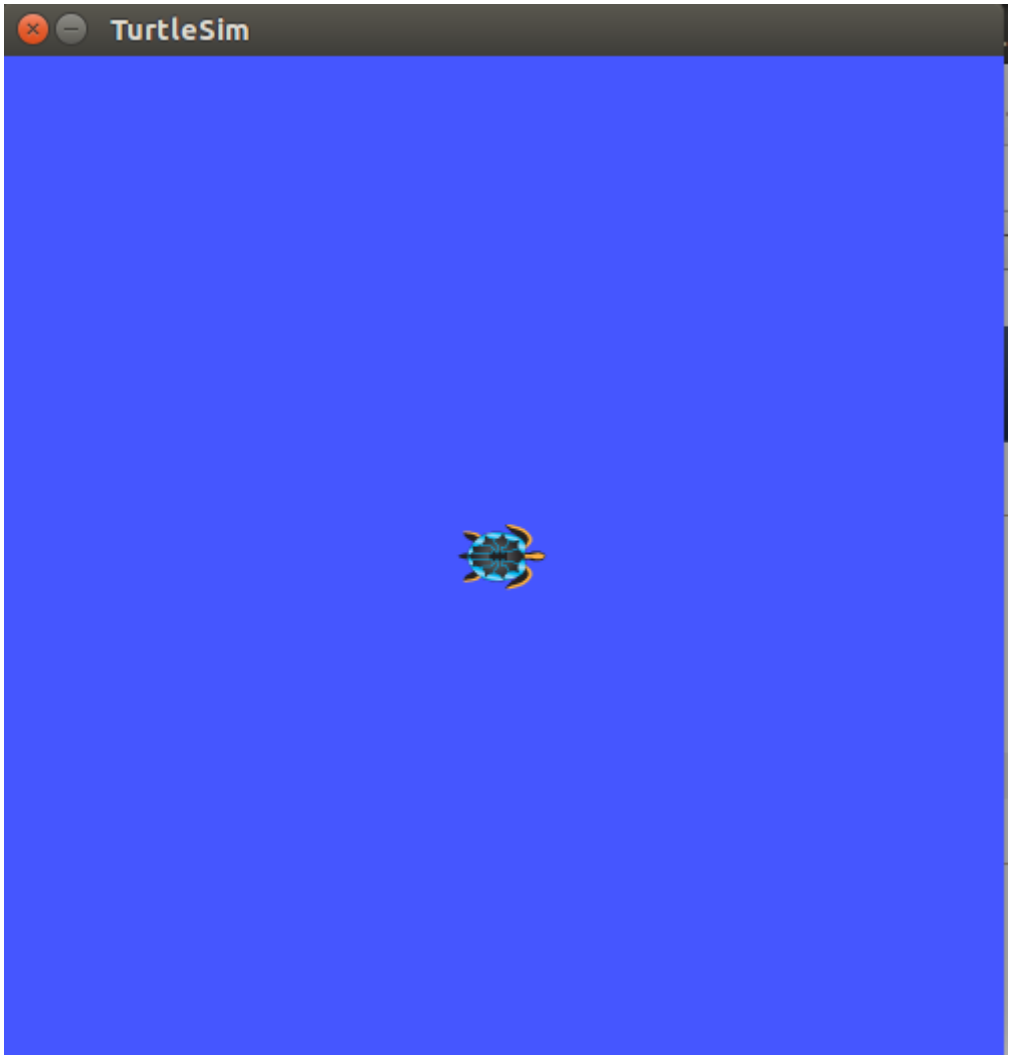
In a **new terminal**, run:

```
$ roscore
```

In a **new terminal**, run:

```
$ rosrun turtlesim turtlesim_node
```

The turtlesim window will open:



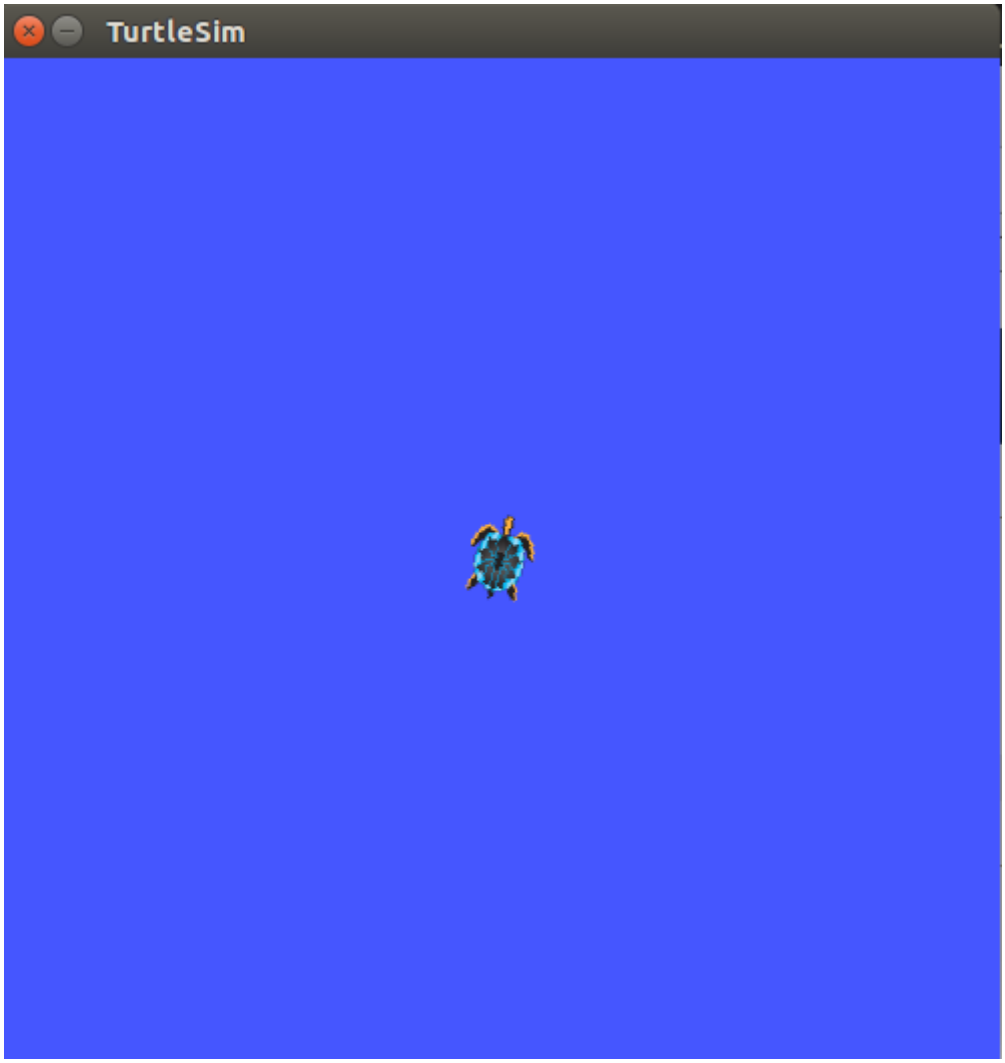
Now, in a **new terminal**, run our code:

```
$ rosrun turtlesim_cleaner rotate.py
```

Just type your inputs and the turtle will rotate! Here we have an example:

```
Let's rotate your robot
Input your speed (degrees/sec):30
Type your distance (degrees):75
Clowkise?: 0
```

The turtle will rotate like this:



Now you can go to the next tutorial! Learn how to move your to a specified goal (/turtlesim/Tutorials/Go%20to%20Goal).

Wiki: turtlesim/Tutorials/Rotating Left and Right (last edited 2017-11-16 14:58:36 by JuliaBibik (/JuliaBibik))