!

**Note:** This tutorial assumes that you have completed the previous tutorials: writing a simple publisher and subscriber (python) (/ROS/Tutorials/WritingPublisherSubscriber%28python%29).

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.

# Moving in a Straight Line

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

**Tutorial Level:** INTERMEDIATE

**Next Tutorial:** Rotating Left/Right (/turtlesim/Tutorials/Rotating%20Left%20and%20Right)

**Contents**

In this tutorial series, we will create python scripts to move our turtle, in order to practice the ROS basics.

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

## 1. Preparing for work

First of all, we have to create a new package.

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg turtlesim_cleaner geometry_msgs rospy
```

Now, build your workspace

```
#At your catkin workspace
$ cd ~/catkin_ws
$ catkin_make
```

And now, create a a src folder for your scripts

```
$ cd ~/catkin_ws/src/turtlesim_cleaner
$ mkdir src
$ catkin_make
```

## 2. Understanding the code

Our code will receive as inputs the desired speed, distance and a variable which defines if the movement is forwards or backwards. Since we can just publish a velocity to the topic ***/turtle1/cmd_vel***, our logic will have to calculate the distance specified.

## 3. The code

Create your move.py (or any name you want) file and save it in your ***~/catkin_ws/src/turtlesim_cleaner/src***, our code will look like this:

Toggle line numbers

```
#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist

def move():
    # Starts a new node
    rospy.init_node('robot_cleaner', anonymous=True)
    velocity_publisher = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
    vel_msg = Twist()

    #Receiveing the user's input
    print("Let's move your robot")
    speed = input("Input your speed:")
    distance = input("Type your distance:")
    isForward = input("Foward?: ")#True or False

    #Checking if the movement is forward or backwards
    if(isForward):
        vel_msg.linear.x = abs(speed)
    else:
        vel_msg.linear.x = -abs(speed)
    #Since we are moving just in x-axis
    vel_msg.linear.y = 0
    vel_msg.linear.z = 0
    vel_msg.angular.x = 0
    vel_msg.angular.y = 0
    vel_msg.angular.z = 0

    while not rospy.is_shutdown():

        #Setting the current time for distance calculus
        t0 = rospy.Time.now().to_sec()
        current_distance = 0

        #Loop to move the turtle in an specified distance
        while(current_distance < distance):
            #Publish the velocity
            velocity_publisher.publish(vel_msg)
            #Takes actual time to velocity calculus
            t1=rospy.Time.now().to_sec()
            #Calculates distancePoseStamped
            current_distance= speed*(t1-t0)
        #After the loop, stops the robot
        vel_msg.linear.x = 0
        #Force the robot to stop
        velocity_publisher.publish(vel_msg)

if __name__ == '__main__':
    try:
        #Testing our function
        move()
    except rospy.ROSInterruptException: pass
```

Don't forget to make your node executable:

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

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:

Toggle line numbers

```
import rospy
from geometry_msgs.msg import Twist
```

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

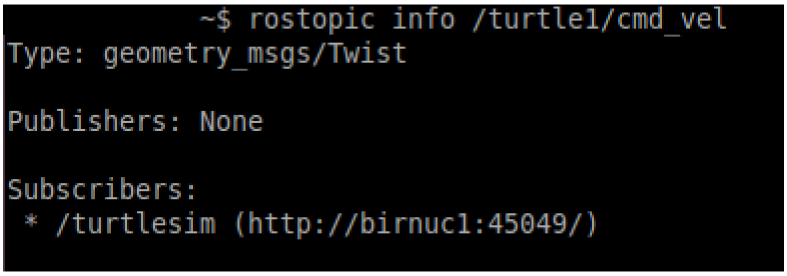
Toggle line numbers

```
def move():
    # Starts a new node
    rospy.init_node('robot_cleaner', anonymous=True)
    velocity_publisher = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=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:

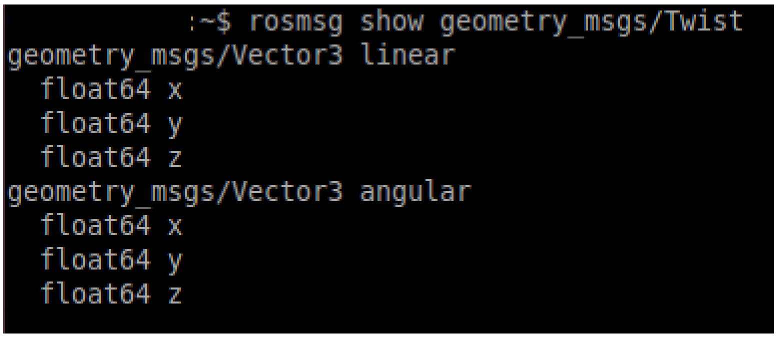


```
            ~$ rostopic info /turtle1/cmd_vel
Type: geometry_msgs/Twist

Publishers: None

Subscribers:
 * /turtlesim (http://birnuc1:45049/)
```

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
```

Since we are moving the turtle in a straight line, we just need the x component, and, depending on the user's input we decide if the movement is forwards or backwards.

Toggle line numbers

```
    if(isForward):
        vel_msg.linear.x = abs(speed)
    else:
        vel_msg.linear.x = -abs(speed)
    #Since we are moving just in x-axis
    vel_msg.linear.y = 0
    vel_msg.linear.z = 0
    vel_msg.angular.x = 0
    vel_msg.angular.y = 0
    vel_msg.angular.z = 0
```

The following statement guarentee that if we press **crtl + c** our code will stops

Toggle line numbers

```
    while not rospy.is_shutdown():
```

Now , with the **rospy.Time.now().to_sec()**. we get the starting time **t0**, and the time **t1** to calculate the distance and while the actual distance is less than the user's input, it will keep publishing:

Toggle line numbers

```
        while(current_distance < distance):
            #Publish the velocity
            velocity_publisher.publish(vel_msg)
            #Takes actual time to velocity calculus
            t1=rospy.Time.now().to_sec()
            #Calculates distancePoseStamped
            current_distance= speed*(t1-t0)
```

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

Toggle line numbers

```
        vel_msg.linear.x = 0
        #Force the robot to stop
        velocity_publisher.publish(vel_msg)
```

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

Toggle line numbers

```
if __name__ == '__main__':
    try:
        #Testing our function
        move()
    except rospy.ROSInterruptException: pass
```

Now , you can test and move your robot!

# 4. Testing the code

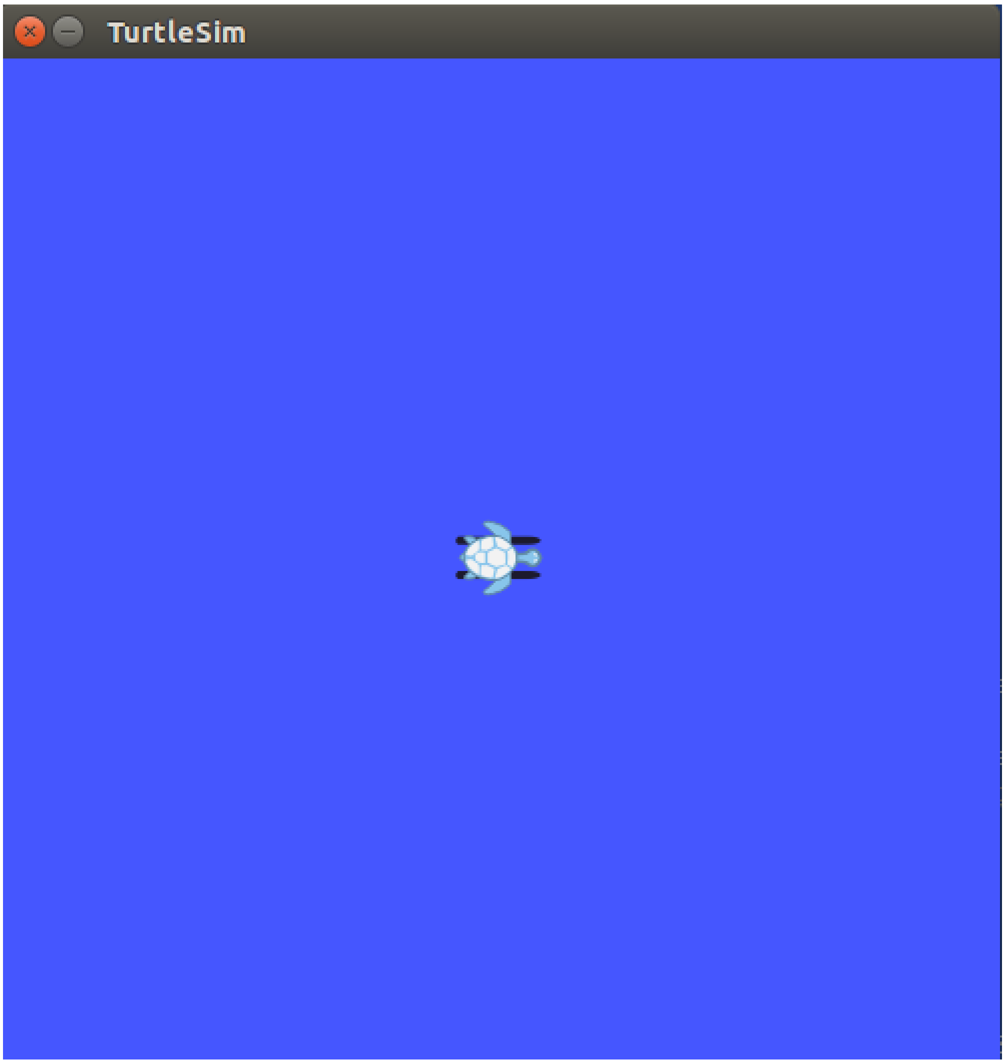
In a **new terminal**, run:

```
$ roscore
```

In a **new terminal**, run:

```
$ rosrun turtlesim turtlesim_node
```

The turtlesim window will open:



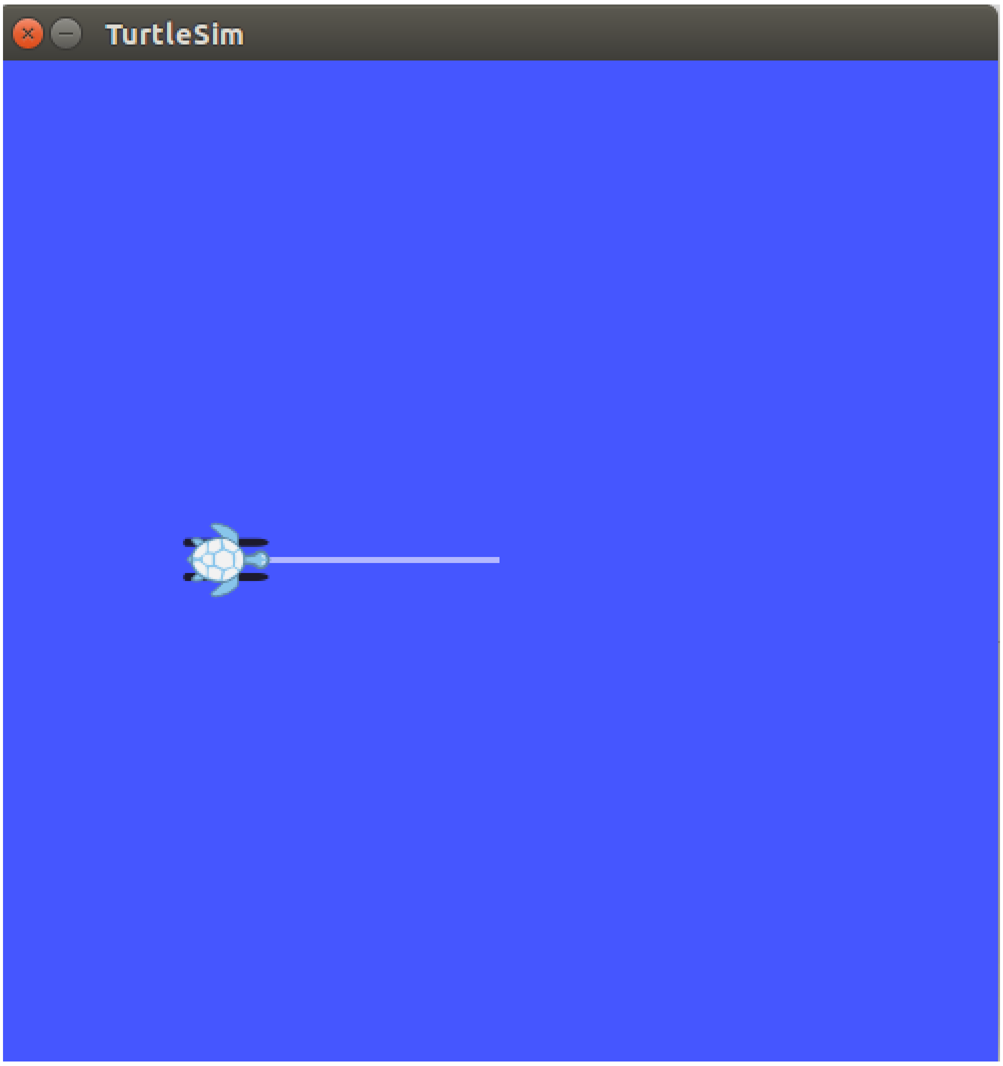
Now, in a **new terminal**, run our code:

```
$ rosrun turtlesim_cleaner move.py
```

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

```
Let's move your robot
Input your speed:1
Type your distance:3
Foward?: 0
```

The turtle will move like this:



Now you can go to the next tutorial! Learn how to rotate (/turtlesim/Tutorials/Rotating%20Left%20and%20Right) your turtle.

Wiki: turtlesim/Tutorials/Moving in a Straight Line (last edited 2018-02-22 18:10:55 by CleberCoutoFilho (/CleberCoutoFilho) )