PythonControlOfTurtlesim

Contents

PoseTurtlesimWithRounding 1/24/2017

```
#!/usr/bin/env python
# poseTurtlesimWithRounding.py          round to 4 decimal places
import rospy

from turtlesim.msg import Pose

# self.pose_subscriber = rospy.Subscriber('/turtle1/pose', Pose, self.callback

class turtlesim():

    def __init__(self):
        #Creating our node,publisher and subscriber
        rospy.init_node('turtlebot_controller', anonymous=True)
# self.velocity_publisher = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
        self.pose_subscriber = rospy.Subscriber('/turtle1/pose', Pose, self.callback)
        self.pose = Pose()
        self.rate = rospy.Rate(.1)
        rospy.spin()

# Continuous printing   of time x and y
# self.rate.sleep()

# Callback function implementing the pose value received
    def callback(self, data):
        self.pose = data
        self.pose.x = round(self.pose.x, 4)
        self.pose.y = round(self.pose.y, 4)
        rospy.loginfo('self.pose.x: {}, self.pose.y: {}'.format(self.pose.x,self.pose.y))

#       rospy.spin()

if __name__ == '__main__':
    try:
        #Testing our function
        x = turtlesim()
 #       rospy.spin()
    except rospy.ROSInterruptException: pass

# [INFO] [1506903472.485273]: self.pose.x: 5.5444, self.pose.y: 5.5444
```

MAKE EXECUTABLE and Find Turtlesim pose

harman@D104-45931:~/Desktop$ chmod +x *.py
harman@D104-45931:~/Desktop$ ls –la

START TURTLESIM

$ roscore (NEW TAB)

$ rosrun turtlesim turtlesim_node

NEW TAB

$ python poseTurtlesimWithRounding.py

RESULT:

[INFO] [1516840025.712550]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.728667]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.744945]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.761138]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.776326]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.792574]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.808646]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.824945]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.840937]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.857207]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.872182]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.888495]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.904501]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.920830]: self.pose.x: 5.5444, self.pose.y: 5.5444
[INFO] [1516840025.936847]: self.pose.x: 5.5444, self.pose.y: 5.5444

PYTHON GO TO GOAL

```
#!/usr/bin/env python          gotogoal.py   turtlesim_cleaner/src/gotogoal.py GitHub

import rospy
from geometry_msgs.msg  import Twist
from turtlesim.msg import Pose
from math import pow,atan2,sqrt


class turtlebot():

   def __init__(self):

      #Creating our node,publisher and subscriber
      rospy.init_node('turtlebot_controller', anonymous=True)
      self.velocity_publisher = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
      self.pose_subscriber = rospy.Subscriber('/turtle1/pose', Pose, self.callback)
      self.pose = Pose()
      self.rate = rospy.Rate(10)


   #Callback function implementing the pose value received

   def callback(self, data):

      self.pose = data
      self.pose.x = round(self.pose.x, 4)
      self.pose.y = round(self.pose.y, 4)

   def get_distance(self, goal_x, goal_y):

      distance = sqrt(pow((goal_x - self.pose.x), 2) + pow((goal_y - self.pose.y), 2))

      return distance

   def move2goal(self):

      goal_pose = Pose()
      goal_pose.x = input("Set your x goal:")
      goal_pose.y = input("Set your y goal:")
      distance_tolerance = input("Set your tolerance:")
      vel_msg = Twist()

      while sqrt(pow((goal_pose.x - self.pose.x), 2) + pow((goal_pose.y - self.pose.y), 2)) >=
distance_tolerance:


         #Porportional Controller

         #linear velocity in the x-axis:
```

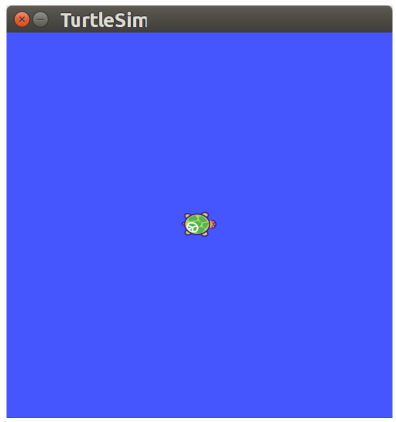
```
        vel_msg.linear.x = 1.5 * sqrt(pow((goal_pose.x - self.pose.x), 2) + pow((goal_pose.y - self.pose.y),
2))
        vel_msg.linear.y = 0
        vel_msg.linear.z = 0

        #angular velocity in the z-axis:
        vel_msg.angular.x = 0
        vel_msg.angular.y = 0
        vel_msg.angular.z = 4 * (atan2(goal_pose.y - self.pose.y, goal_pose.x - self.pose.x) -
self.pose.theta)

        #Publishing our vel_msg
        self.velocity_publisher.publish(vel_msg)
        self.rate.sleep()
      #Stopping our robot after the movement is over
      vel_msg.linear.x = 0
      vel_msg.angular.z =0
      self.velocity_publisher.publish(vel_msg)

      rospy.spin()


if __name__ == '__main__':

   try:

      #Testing our function
      x = turtlebot()
      x.move2goal()

   except rospy.ROSInterruptException: pass

#RESULT:
 #  harman@D104-45931:~/Desktop$ python gotogoal.py
# Set your x goal:1.0
# Set your y goal:1.0
# Set your tolerance:0.5
```

Start Turtlesim_node and Move TurtlesimToGoal

Terminal 1
harman@D104-45931:~$ **roscore**

Terminal 2
harman@D104-45931:/opt/ros/kinetic/share/turtlesim$ **rosrun turtlesim turtlesim_node**

[INFO] [1506798544.035333766]: Starting turtlesim with node name /turtlesim
[INFO] [1506798544.039981993]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]



Let Move Turtle to 1,1

$ **python gotogoal.py**

Set your x goal:1.0
Set your y goal:1.0
Set your tolerance:0.2