#!/usr/bin/env python3

import rclpy

from rclpy.node import Node

from turtlesim.msg import Pose

from geometry\_msgs.msg import Twist

import math

import time

# from turtle\_control import TurtleControllerNode

class traingle(Node):

def \_\_init\_\_(self):

super().\_\_init\_\_("turtle\_controller")

self.length\_y = 2.0

self.length\_x=2.0

self.theta = math.pi

self.pose\_ = None

self.cmd\_vel\_publisher\_ = self.create\_publisher(Twist, "turtle1/cmd\_vel", 10)

self.control\_loop\_timer\_ = self.create\_timer(0.01, self.control\_loop)

def control\_loop(self):

msg = Twist()

msg.linear.x = 0.0

msg.angular.z = (math.pi/3)

if msg.angular.z == math.pi/3:

self.cmd\_vel\_publisher\_.publish(msg)

time.sleep(2)

msg.angular.z = 0.0

msg.linear.x = self.length\_x

# msg.linear.y = self.length\_y

self.cmd\_vel\_publisher\_.publish(msg)

time.sleep(1)

msg.angular.z = math.pi/1.525

msg.linear.x = 0.0

self.cmd\_vel\_publisher\_.publish(msg)

time.sleep(2)

if msg.angular.z == math.pi/1.525:

msg.angular.z = 0.0

msg.linear.x = self.length\_x

self.cmd\_vel\_publisher\_.publish(msg)

time.sleep(1)

msg.angular.z = math.pi/1.525

msg.linear.x = 0.0

self.cmd\_vel\_publisher\_.publish(msg)

time.sleep(2)

if msg.angular.z == math.pi/1.525:

msg.angular.z=0.0

msg.linear.x = self.length\_x

self.cmd\_vel\_publisher\_.publish(msg)

time.sleep(2)

def main(args=None):

rclpy.init(args=args)

tri = traingle()

rclpy.spin(tri)

rclpy.shutdown()

if \_\_name\_\_=="\_\_main\_\_":

main()