import rclpy

from rclpy.node import Node

from std\_msgs.msg import String

from nav\_msgs.msg import Odometry

from turtlesim.msg import Pose

from geometry\_msgs.msg import Twist

from math import sqrt

import math

class Image(Node):

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

super().\_\_init\_\_('image\_subscriber')

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

self.subscriber = self.create\_subscription(Pose, "/pose", self.sub\_cb, 10)

self.timer = self.create\_timer(0.5, self.pub\_cb)

self.coordtimer = self.create\_timer(0.5, self.coordinates)

self.coordinateArray = [[0, 0], [0, 1], [1, 1], [1, 0], [0, 0]]

self.currentPose = None

self.currentGoalIndex = 0

slef.tolerance = 0.1

def sub\_cb(self, msg):

self.x = msg.pose.pose.position.x

self.y = msg.pose.pose.position.y

self.currentPose = msg

if self.z == 0:

self.z = pos.pos.orientation.z

if self.initialx == 0:

self.initialx = msg.pose.pose.position.x

self.initialy = msg.pose.pose.position.y

def pub\_cb(self):

msg = Twist()

msg.linear.x = 0.0

msg.linear.y = 0.0

msg.linear.z = 0.

msg.angular.x = 0.

msg.angular.y = 0.

msg.angular.z = 0.

self.publisher\_.publish(msg)

if self.currentPose is None or self.currentGoalIndex >= len(self.coordinateArray):

return

goal\_x, goal\_y = self.coordinateArray[self.current\_goal\_index]

distance = sqrt((goal\_x - self.current\_pose.x) \*\* 2 + (goal\_y - self.current\_pose.y) \*\* 2)

if distance < self.tolerance:

self.current\_goal\_index += 1

if self.current\_goal\_index >= len(self.coordinateArray):

self.stop\_turtle()

return

angle\_to\_goal = atan2(goal\_y - self.current\_pose.y, goal\_x - self.current\_pose.x)

angle\_difference = angle\_to\_goal - self.current\_pose.theta

while angle\_difference > pi:

angle\_difference -= 2 \* pi

while angle\_difference < -pi:

angle\_difference += 2 \* pi

if abs(angle\_difference) > self.tolerance:

msg.angular.z = 1.0 \* angle\_difference

else:

msg.linear.x = 0.5 \* distance

def stop\_turtle(self):

msg = Twist()

msg.linear.x = 0.0

msg.angular.z = 0.0

self.publisher\_.publish(msg)

def main():

rclpy.init()

image\_subscription = Image()

rclpy.spin(image\_subscription)

rclpy.shutdown()

if \_\_name\_\_ == '\_\_main\_\_':

main()