ROS Assignment

Vasileios Papadopoulos

## Binaries

{

/opt/ros/noetic/bin

}

## Create catkin package

{

cd ~/desktop/ros-assignment/src

catkin\_create\_pkg ros\_assignment std\_msgs rospy roscpp

cd ~/desktop/ros-assignment

catkin\_make

. ~/desktop/ros-assignment/devel/setup.bash

rospack depends1 ros\_assignment

}

## Start ros core infrastructure

{

roscore

}

## Launch turtle bot

{

#rosrun turtlesim turtlesim\_node

roslaunch turtle\_tf turtle\_tf\_demo.launch

}

## Display topic messages

{

rostopic list

rostopic echo turtle1/cmd\_vel

}

## Launch teleoperation

{

rosrun turtlesim turtle\_teleop\_key

}

## Change background

In order to change the background color of turtlesim we use the rosparam command.

Particularly, we set RGB colors individually and then we use rosservice command to

apply the changes.

{

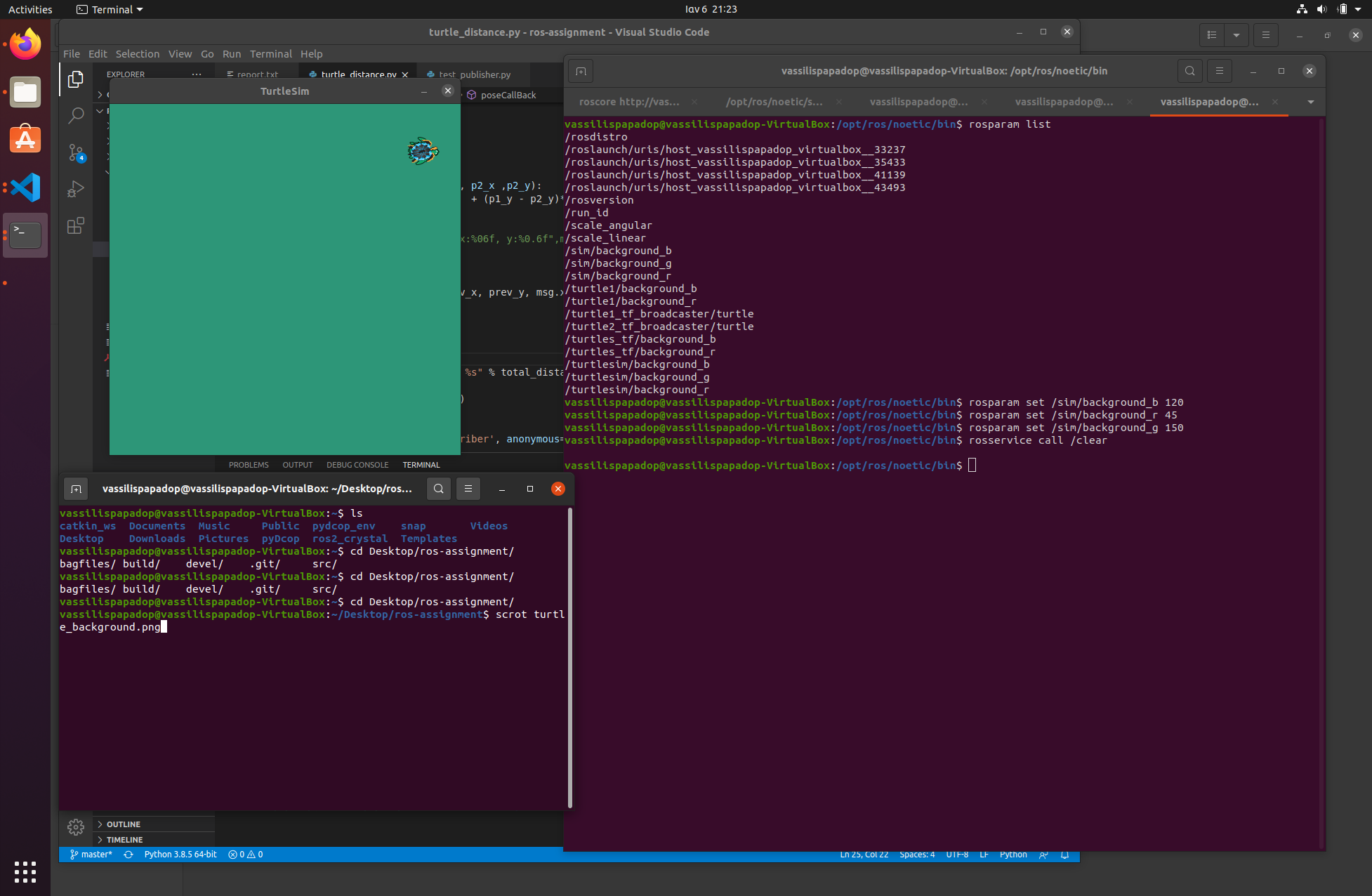
rosparam set /sim/background\_r 45

rosparam set /sim/background\_g 150

rosparam set /sim/background\_b 120

##apply changes

rosservice call /clear



}

## Inspect tf tree

In order to run tf view\_frames command I had to change line 89 in ***/opt/ros/noetic/lib/view\_frames*** to avoid **TypeError: cannot use a string pattern on a bytes-like object**

line 89 replaced with :

decoded = vstr.decode('utf-8')

m = r.search(decoded)

Distributor ID: Ubuntu

Description: Ubuntu 20.04.1 LTS

Release: 20.04

Codename: focal

{

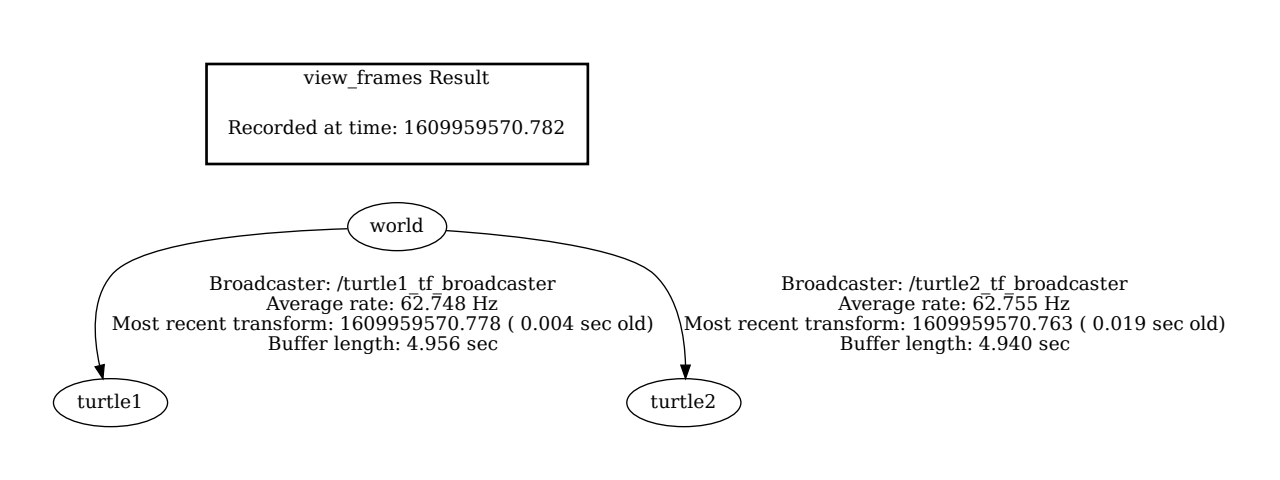
rosrun tf view\_frames

OR

rosrun rqt\_tf\_tree rqt\_tf\_tree

#view pdf

evince frames.pdf



}

## Record/Play playback

{

rosbag record -O subset /turtle1/cmd\_vel /turtle1/pose

rosbag play subset.bag

}

## Calculate Distance(subscribe/publish)

import rospy

from turtlesim.msg import Pose

from std\_msgs.msg import String

from math import sqrt

from threading import Thread, Lock

prev\_x = 0.0

prev\_y = 0.0

total\_distance = 0.0

# Euclidean distance

def calculate\_distance(p1\_x, p1\_y, p2\_x ,p2\_y):

return sqrt((p1\_x - p2\_x) \*\*2 + (p1\_y - p2\_y)\*\*2)

def poseCallBack(msg, publisher):

#rospy.loginfo("turtle pose: x:%06f, y:%0.6f",msg.x , msg.y)

global prev\_x

global prev\_y

global total\_distance

step = calculate\_distance(prev\_x, prev\_y, msg.x, msg.y)

total\_distance += step

prev\_x = msg.x

prev\_y = msg.y

#publish distance

publish\_msg = "Total distance %s" % total\_distance

rospy.loginfo(publish\_msg)

publisher.publish(publish\_msg)

def subscriber():

rospy.init\_node('turtle\_subscriber', anonymous=True)

publisher = rospy.Publisher('turtle\_publisher', String, queue\_size=10)

rospy.Subscriber('/turtle1/pose', Pose, poseCallBack, publisher)

rospy.spin()

# print

print('Total travelled distance', total\_distance)

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

subscriber()

## Test publisher

import rospy

from std\_msgs.msg import String

def callback(data):

rospy.loginfo(rospy.get\_caller\_id() + 'received %s', data.data)

def listener():

# In ROS, nodes are uniquely named. If two nodes with the same

# name are launched, the previous one is kicked off. The

# anonymous=True flag means that rospy will choose a unique

# name for our 'listener' node so that multiple listeners can

# run simultaneously.

rospy.init\_node('listener', anonymous=True)

rospy.Subscriber('turtle\_publisher', String, callback)

# spin() simply keeps python from exiting until this node is stopped

rospy.spin()

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

listener()