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
}
```

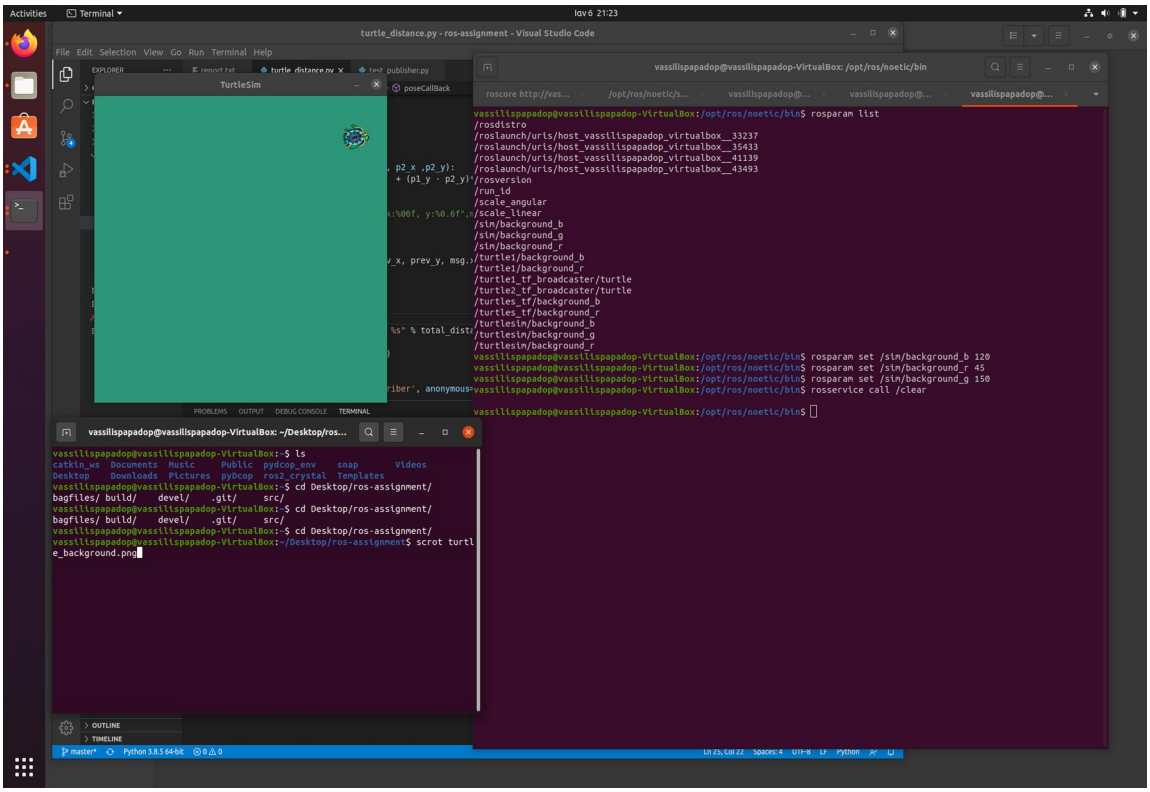
Change background

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 :

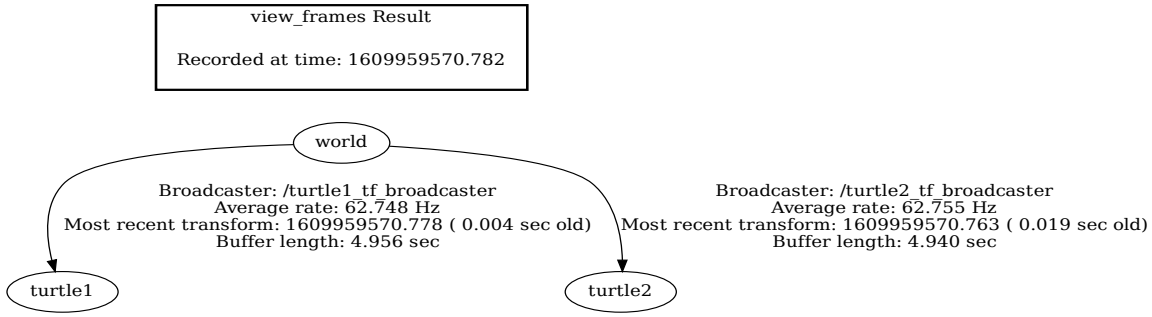
```
decoded = vstr.decode('utf-8')
```

```
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()
```