ROS 2 Humble

I have to change some part of the code to accommodate the difference between ROS 1 and ROS 2,

Broadcaster code:

```
import rclpy
from geometry_msgs.msg import TransformStamped, Quaternion
from turtlesim.msg import Pose
import tf2_ros
import math

def handle_turtle_pose(msg, turtlename, tf_broadcaster):
    transform = TransformStamped()
    transform.header.stamp = rclpy.time.Time().to_msg()
    transform.header.frame_id = "world"
    transform.child_frame_id = turtlename
    transform.transform.translation.x = msg.x
    transform.transform.translation.y = msg.y
    transform.transform.translation.z = 0
    quat = tf2_ros.transformations.quaternion_from_euler(0, 0, msg.theta)
    transform.transform.rotation = Quaternion(x=quat[0], y=quat[1], z=quat[2], w=quat[3])

    tf_broadcaster.sendTransform(transform)

def main(args=None):
    rclpy.init(args=args)
    node = rclpy.create_node('turtle_tf_broadcaster')
    tf_broadcaster = tf2_ros.TransformBroadcaster(node)
    turtlename = node.declare_parameter('turtle', 'turtle').value

    subscription = node.create_subscription(Pose, '/' + turtlename + '/pose', lambda msg:
handle_turtle_pose(msg, turtlename, tf_broadcaster), 10)

    while rclpy.ok():
        rclpy.spin_once(node)

    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Listener Code:

```
import rclpy
from geometry_msgs.msg import Twist
import tf2_ros
import math

def main():
    rclpy.init(args=None)
    node = rclpy.create_node('turtle_tf_listener')

    tf_buffer = tf2_ros.Buffer()
    tf_listener = tf2_ros.TransformListener(tf_buffer, node)

    turtle_vel = node.create_publisher(Twist, 'turtle2/cmd_vel', 1)
    rate = node.create_rate(10)

    while rclpy.ok():
        try:
            transform = tf_buffer.lookup_transform('turtle2', 'turtle1', rclpy.time.Time())
        except (tf2_ros.LookupException, tf2_ros.ConnectivityException,
tf2_ros.ExtrapolationException):
            continue

        angular = 4 * math.atan2(transform.transform.translation.y, transform.transform.translation.x)
        linear = 0.5 * math.sqrt(transform.transform.translation.x ** 2 + transform.transform.translation.y
** 2)

        cmd = Twist()
        cmd.linear.x = linear
        cmd.angular.z = angular
        turtle_vel.publish(cmd)

        rate.sleep()

    node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

setup.py

```
from setuptools import setup

package_name = 'learning_tfp'

setup(
    name=package_name,
    version='0.0.1',
    packages=[],
    py_modules=[
        'learning_tfp.turtle_tf_broadcaster',
        'learning_tfp.turtle_tf_listener'
    ],
    install_requires=[
        'setuptools',
        'rclpy',
        'geometry_msgs',
        'tf2_ros',
        'turtlesim'
    ],
    zip_safe=True,
    maintainer='Your Name',
    maintainer_email='your_email@example.com',
    description='ROS 2 package for learning_tf',
    license='Apache License 2.0',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'turtle_tf_broadcaster = learning_tfp.turtle_tf_broadcaster:main',
            'turtle_tf_listener = learning_tfp.turtle_tf_listener:main',
        ],
    },
)
```

Package.xml

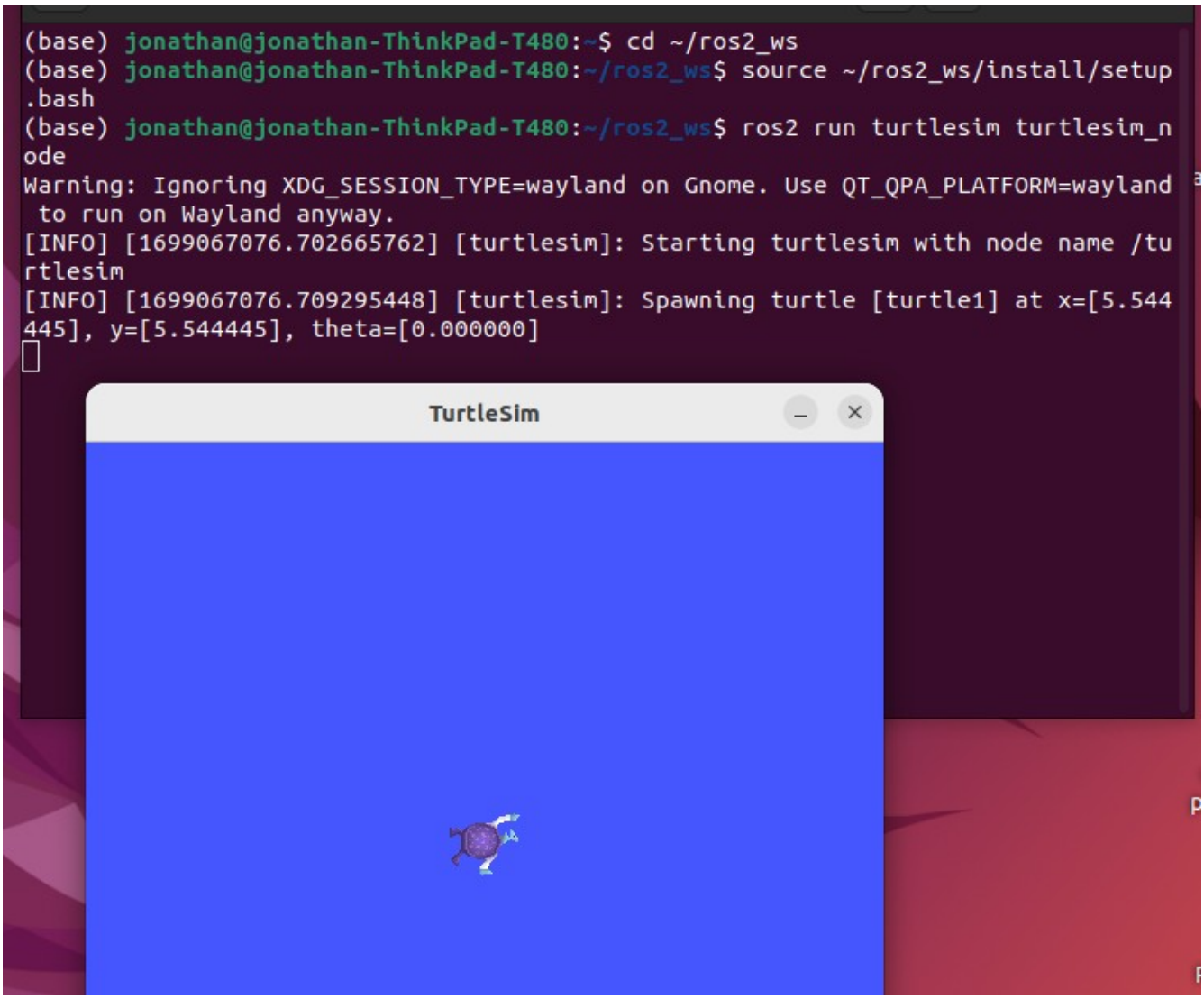
```
<?xml version="1.0"?>
<?xml-model href="http://download.ros.org/schema/package_format3.xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>learning_tfp</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="106852617+Grlee316@users.noreply.github.com">jonathan</maintainer>
  <license>TODO: License declaration</license>

  <test_depend>ament_copyright</test_depend>
  <test_depend>ament_flake8</test_depend>
  <test_depend>ament_pep257</test_depend>
  <test_depend>python3-pytest</test_depend>

  <exec_depend>rclpy</exec_depend>
  <exec_depend>geometry_msgs</exec_depend>
  <exec_depend>turtlesim</exec_depend>
  <exec_depend>tf2_ros</exec_depend>

  <export>
    <build_type>ament_python</build_type>
  </export>
</package>
```

Running Turtlesim:



When running turtle broadcaster:

```
(base) jonathan@jonathan-ThinkPad-T480:~/ros2_ws$ ros2 run learning_tfp turtle_tf_bro
adcaster __name:=turtle1_tf_broadcaster /turtle1
[WARN] [1699067156.223675837] [rcl]: Found remap rule '__name:=turtle1_tf_broadcaster
'. This syntax is deprecated. Use '--ros-args --remap __name:=turtle1_tf_broadcaster'
 instead.
```

```
(base) jonathan@jonathan-ThinkPad-T480:~/ros2_ws$ ros2 run learning_tfp turtle_tf_bro
adcaster __name:=turtle2_tf_broadcaster /turtle2
[WARN] [1699067251.350674570] [rcl]: Found remap rule '__name:=turtle2_tf_broadcaster
'. This syntax is deprecated. Use '--ros-args --remap __name:=turtle2_tf_broadcaster'
 instead.
```

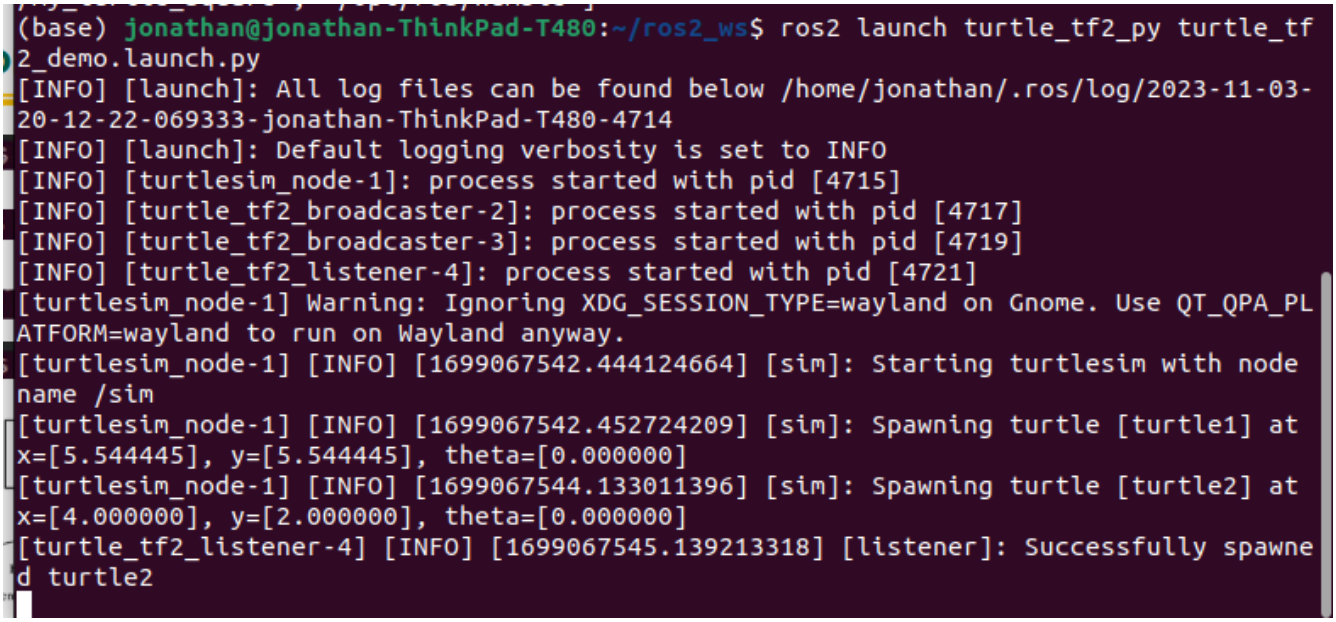
When running turtle listener:

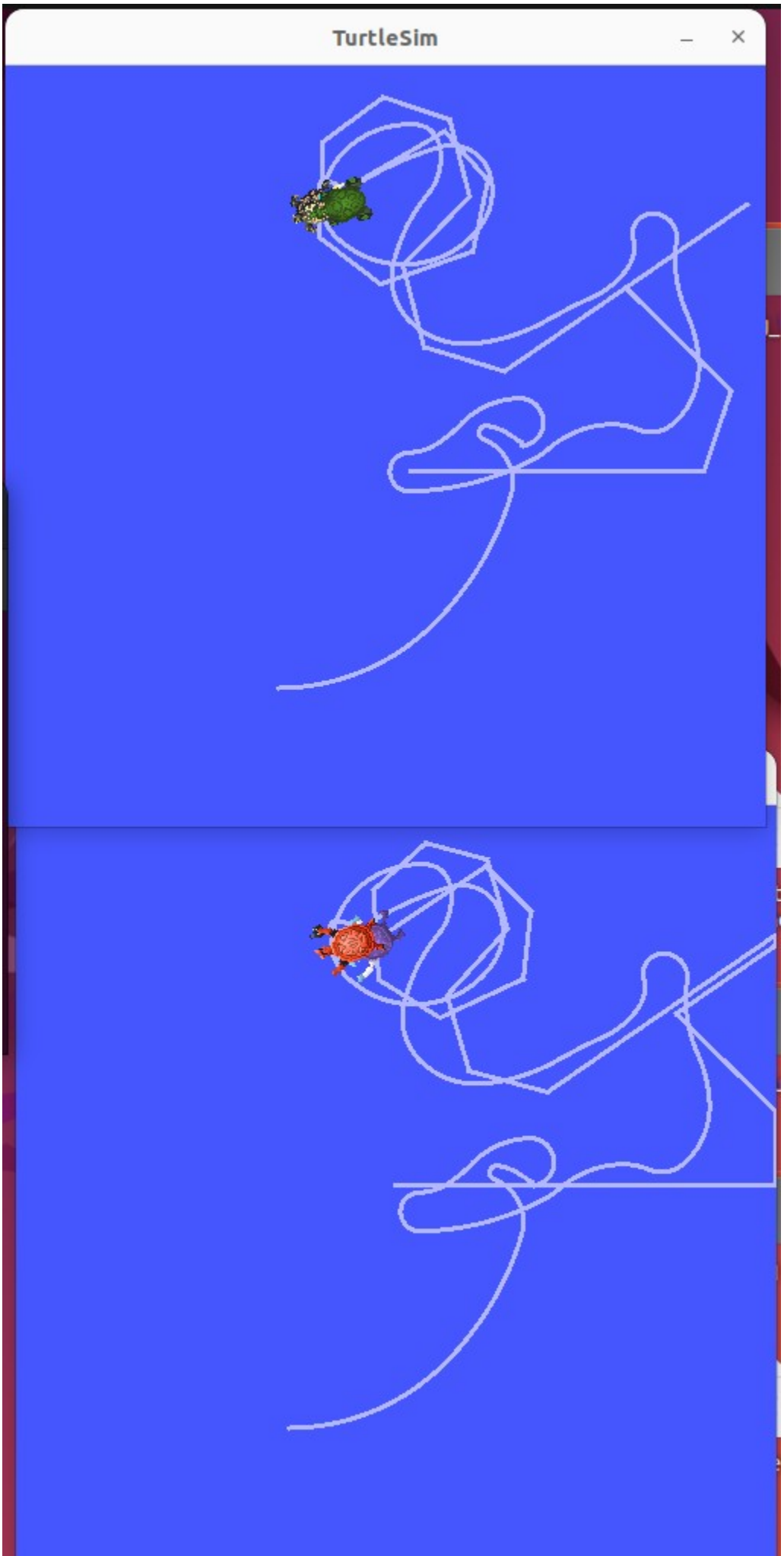
```
Package 'learning_tf' not found
(base) jonathan@jonathan-ThinkPad-T480:~/ros2_ws$ ros2 run learning_tfp turtle_tf_lis
tener
```

I think the listener does not work well with python and Ros2, or at least the code that I tried to run after converting the python code from ros to ros2 does not worked properly.

I tried some other iteration that uses tf2 that was written at the lecture before, and I think it worked

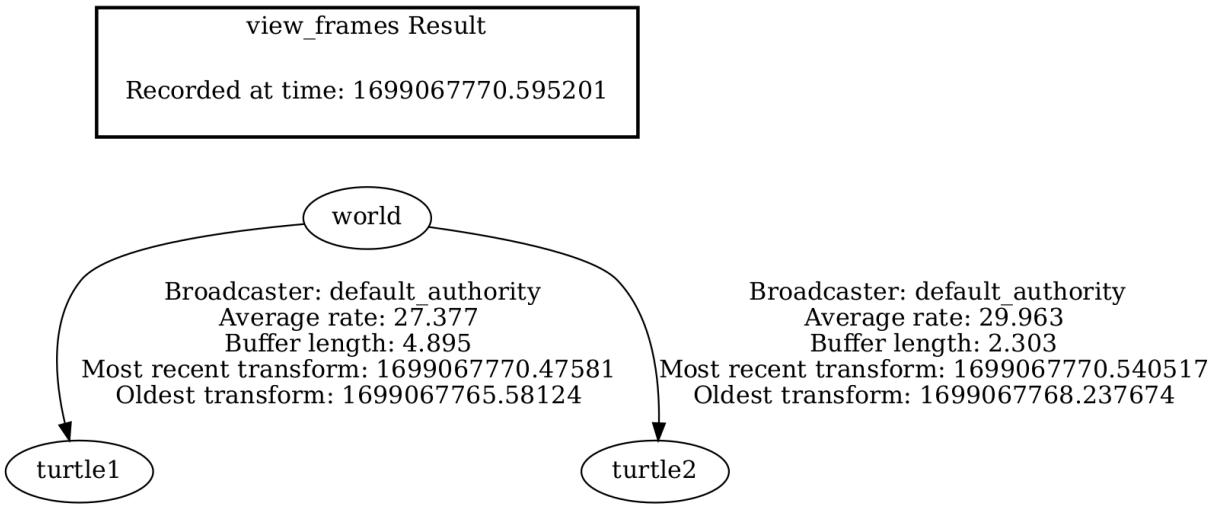
On lecture 7, there's an example for tf2 for turtle, so I tried:




TurtleSim

And apparently the demo did worked. I also tried to show the frames to show the broadcaster and listener from this turtlesim, and I got the frame results as a pdf file.

```
(base) jonathan@jonathan-ThinkPad-T480:~/ros2_ws$ ros2 run tf2_tools view_frames
[INFO] [1699067765.490377898] [view_frames]: Listening to tf data for 5.0 seconds...
[INFO] [1699067770.542985086] [view_frames]: Generating graph in frames.pdf file...
[INFO] [1699067770.590803097] [view_frames]: Result:tf2_msgs.srv.FrameGraph_Response(
frame_yaml="turtle1: \n  parent: 'world'\n  broadcaster: 'default_authority'\n  rate:
 27.377\n  most_recent_transform: 1699067770.475810\n  oldest_transform: 1699067765.5
81240\n  buffer_length: 4.895\nturtle2: \n  parent: 'world'\n  broadcaster: 'default_
authority'\n  rate: 29.963\n  most_recent_transform: 1699067770.540517\n  oldest_tran
sform: 1699067768.237674\n  buffer_length: 2.303\n")
(base) jonathan@jonathan-ThinkPad-T480:~/ros2_ws$
```



and it does print the frame, so I think it is possible to use the TF2 on ros2, we just need to find the right code (the python code does not worked properly with the broadcaster and listener for this exercise). So I will try again to use c++ version of this code, and see if it works, because I think something wrong with my c++ because usually it's hard for me to compile the c++ version of ros2 code (something with dependencies and such).