

Lab 2 Proof of Completion
Tong Wang
662115857

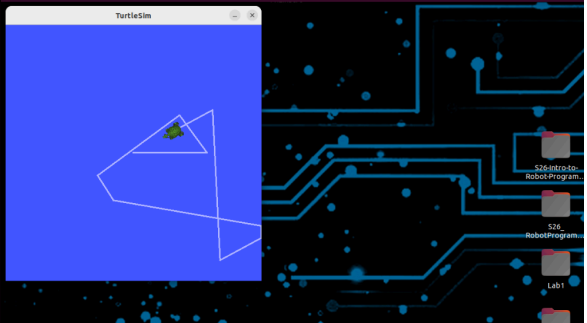
Task#2 Starting nodes

```
yahboom@yahboom-virtual-machine: ~$ ros2 pkg executables turtlesim
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
yahboom@yahboom-virtual-machine: ~$ ros2 run turtlesim turtlesim_node
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland
to run on Wayland anyway.
[INFO] [1770361232.589051341] [turtlesim]: Starting turtlesim with node name /tu
rlesim
[INFO] [1770361232.607644942] [turtlesim]: Spawning turtle [turtle1] at x=[5.544
445], y=[5.544445], theta=[0.000000]
[WARN] [1770361386.615834946] [turtlesim]: Oh no! I hit the wall! (Clamping from
[x=11.118094, y=8.726069])
[WARN] [1770361386.632391162] [turtlesim]: Oh no! I hit the wall! (Clamping from
[x=11.120421, y=8.731520])
[WARN] [1770361386.647491445] [turtlesim]: Oh no! I hit the wall! (Clamping from
[x=11.120421, y=8.736972])
[WARN] [1770361386.663885454] [turtlesim]: Oh no! I hit the wall! (Clamping from
yahboom@yahboom-virtual-machine: ~$ ros2 run turtlesim turtle_teleop_key
Reading from keyboard
-----
Use arrow keys to move the turtle.
Use G|B|V|C|D|E|R|T keys to rotate to absolute orientations. 'F' to cancel a
ation.
'Q' to quit.
[Q]

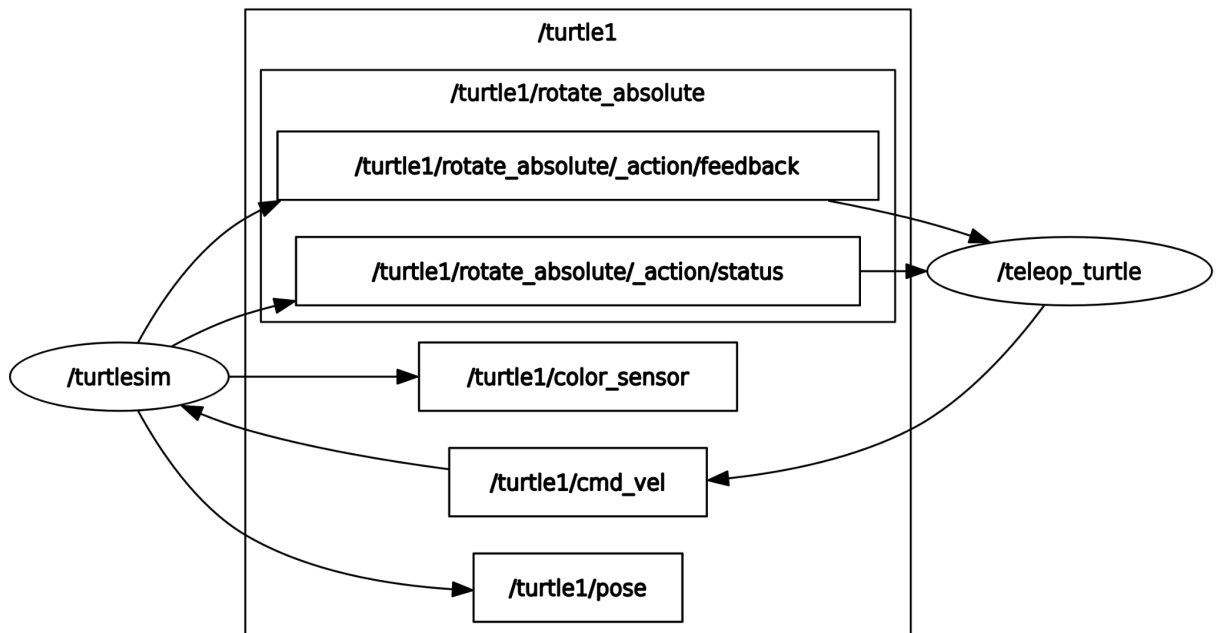
/teleop_turtle/list_parameters: rcl_interfaces/srv/ListParameters
/teleop_turtle/set_parameters: rcl_interfaces/srv/SetParameters
/teleop_turtle/set_parameters_atomically: rcl_interfaces/srv/SetParametersAt
omically
Service Clients:

Action Servers:

Action Clients:
/turtle1/rotate_absolute: turtlesim/action/RotateAbsolute
yahboom@yahboom-virtual-machine: ~$ rqt_graph
yahboom@yahboom-virtual-machine: ~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
yahboom@yahboom-virtual-machine: ~$ ros2 topic list -t
/parameter_events [rcl_interfaces/msg/ParameterEvent]
/rosout [rcl_interfaces/msg/Log]
/turtle1/cmd_vel [geometry_msgs/msg/Twist]
/turtle1/color_sensor [turtlesim/msg/Color]
/turtle1/pose [turtlesim/msg/Pose]
yahboom@yahboom-virtual-machine: ~$
```



Task#2_step#7



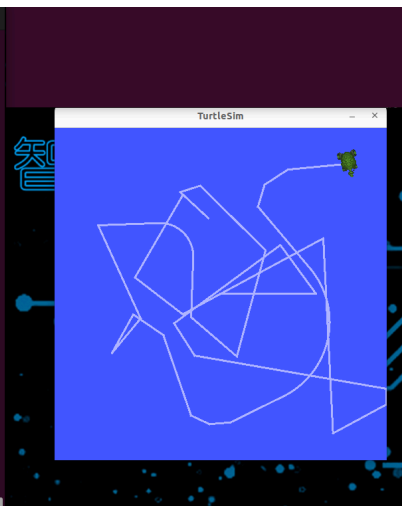
Task#2_step#21

```
yahboom@yahboom-virtual-machine:~$ ros2 topic info /turtle1/pose
Type: turtlesim/msg/Pose
Publisher count: 1
Subscription count: 0
yahboom@yahboom-virtual-machine:~$ ros2 interface show turtlesim/msg/Pose
float32 x
float32 y
float32 theta

float32 linear_velocity
float32 angular_velocity
yahboom@yahboom-virtual-machine:~$ ros2 topic echo /turtle1/pose
x: 7.0591936111450195
y: 6.9968414306640625
theta: -0.7791852951049805
linear_velocity: 0.0
angular_velocity: 0.0
---
```

Task#2_step#25

```
yahboom@yahboom-virtual-machine:~$ ros2 topic echo /turtle1/pose
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=5.0))
publishing #23: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=5.0))
publishing #24: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=5.0))
publishing #25: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=5.0))
publishing #26: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=5.0))
publishing #27: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=5.0))
publishing #28: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=5.0))
publishing #29: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0,
y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=5.0))
```



Task#2_services challenge

```
yahboom@yahboom-virtual-machine:~$ ros2 service call /turtle1/set_pen turtlesim/
srv/SetPen "{r: 255, g: 0, b: 0, width: 3, 'off': 0}"
waiting for service to become available...
requester: making request: turtlesim.srv.SetPen_Request(r=255, g=0, b=0, width=3
, off=0)

response:
turtlesim.srv.SetPen_Response()
```



Task#2_action_step#8

```
yahboom@yahboom-virtual-machine:~$ ros2 action send_goal /turtle1/rotate_absolute
turtlesim/action/RotateAbsolute "{theta: 0.0}" --feedback
Waiting for an action server to become available...
Sending goal:
  theta: 0.0

Goal accepted with ID: b3e265e17e9b4e0b9d71ffb952da6cef

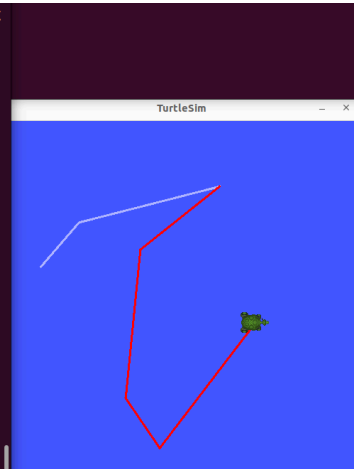
Feedback:
  remaining: -0.917549729347229

Feedback:
  remaining: -0.9015496969223022

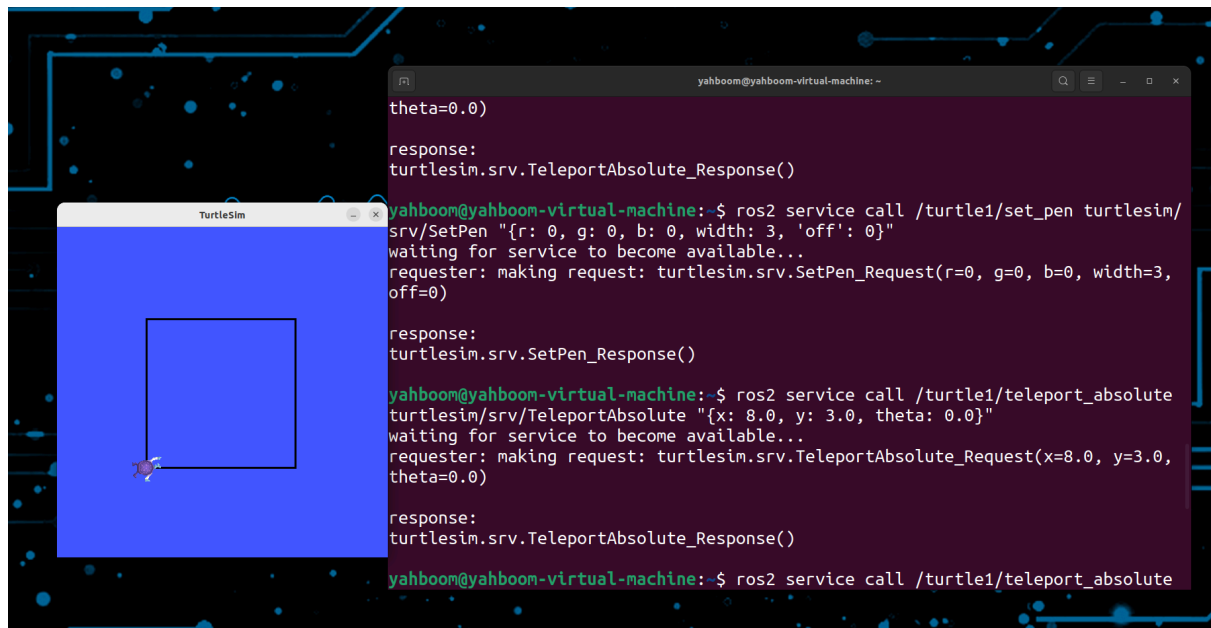
Feedback:
  remaining: -0.8855497241020203

Feedback:
  remaining: -0.8695496916770935

Feedback:
```



Task#3_square_drawing



To prevent unwanted lines connecting the center origin to the shape, I implemented a "pen-up/pen-down" logic. I first disabled the pen trace by calling `/turtle1/set_pen` with the parameter `off: 1` and moved the turtle to the starting vertex (3, 3) using `/turtle1/teleport_absolute`. After re-enabling the pen (`off: 0`), I sequentially teleported the turtle to coordinates (8, 3), (8, 8), (3, 8), and finally returned to (3, 3), which allows me to draw a clean and closed square.