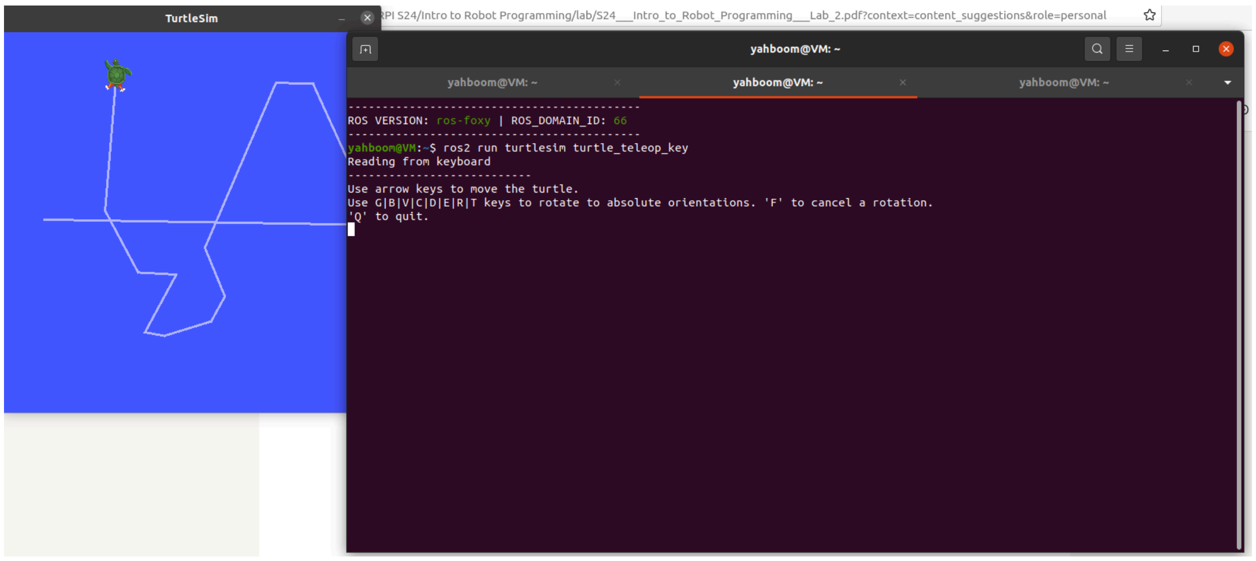
starting nodes


TurtleSim
RPI S24/Intro to Robot Programming/lab/S24___Intro_to_Robot_Programming___Lab_2.pdf?context=content_suggestions&role=personal
yahboom@VM: ~
yahboom@VM: ~
yahboom@VM: ~
yahboom@VM: ~

```
-------------------------------------------
ROS VERSION: ros-foxy | ROS_DOMAIN_ID: 66
-------------------------------------------
yahboom@VM:~$ 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 rotation.
'Q' to quit.
```

Fig1 Starting nodes

Topics:

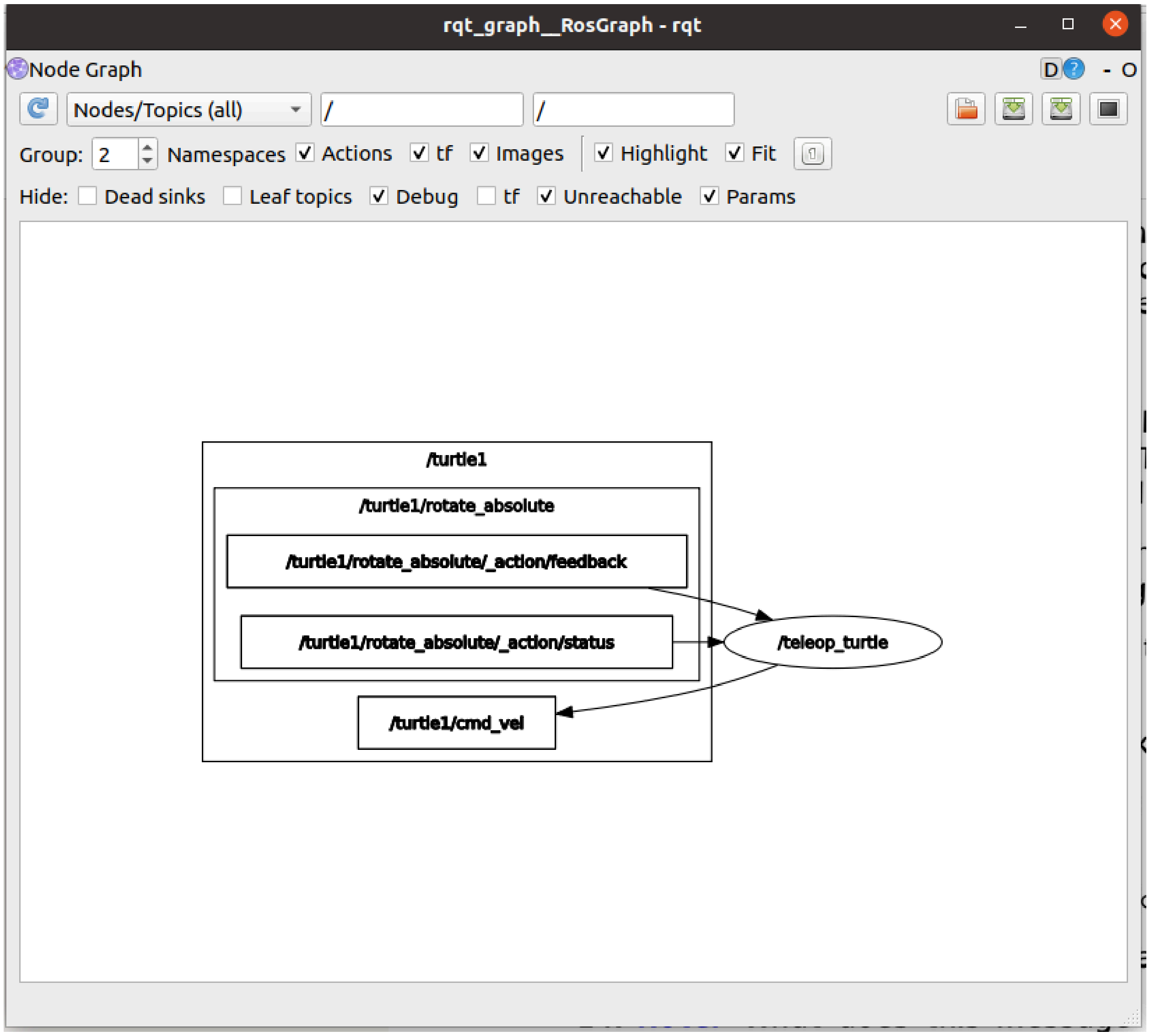
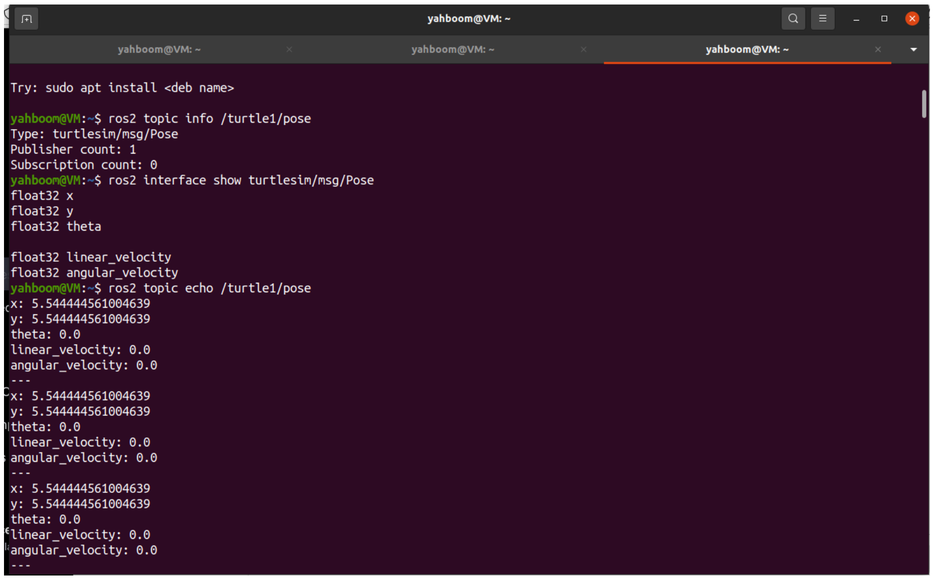


Fig2: multiple terminals

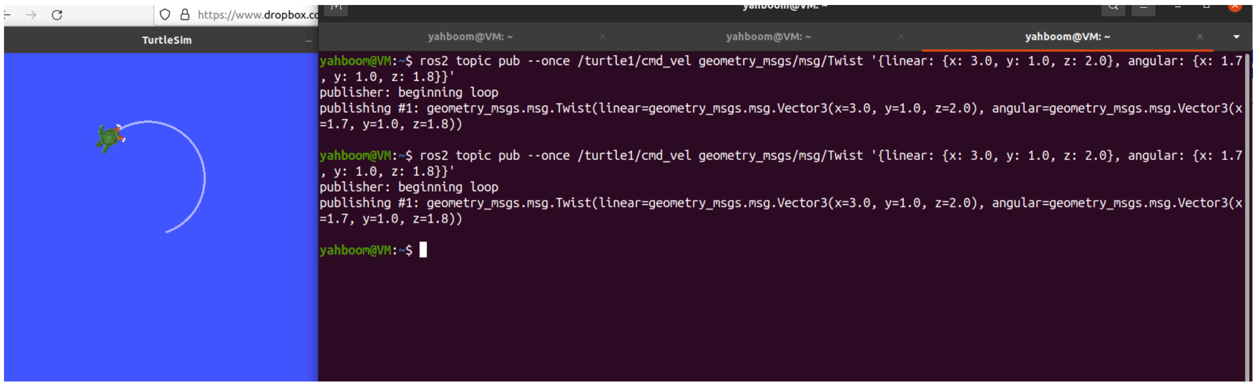
Q21



```
Try: sudo apt install <deb name>

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

float32 linear_velocity
float32 angular_velocity
yahboom@VM:~$ ros2 topic echo /turtle1/pose
x: 5.544444561004639
y: 5.544444561004639
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 5.544444561004639
y: 5.544444561004639
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 5.544444561004639
y: 5.544444561004639
theta: 0.0
linear_velocity: 0.0
angular_velocity: 0.0
---
```

Fig3: Screenshot of three commands for Q21

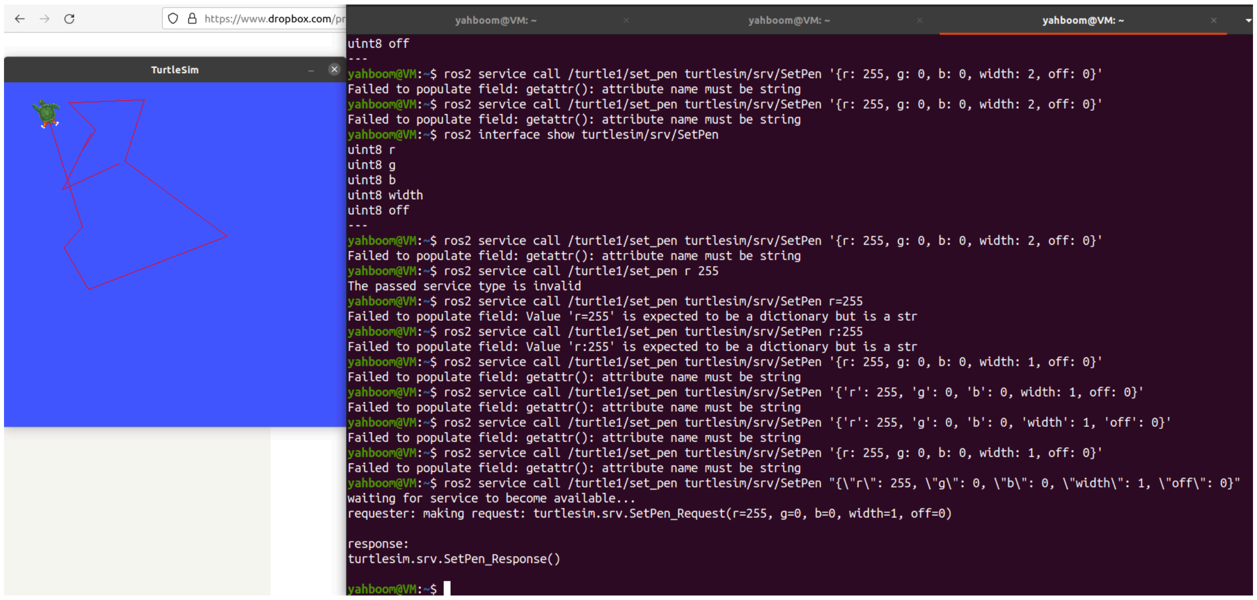
Q25



```
yahboom@VM:~$ ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist '{linear: {x: 3.0, y: 1.0, z: 2.0}, angular: {x: 1.7
, y: 1.0, z: 1.8}}'
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=3.0, y=1.0, z=2.0), angular=geometry_msgs.msg.Vector3(x
=1.7, y=1.0, z=1.8))

yahboom@VM:~$ ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist '{linear: {x: 3.0, y: 1.0, z: 2.0}, angular: {x: 1.7
, y: 1.0, z: 1.8}}'
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=3.0, y=1.0, z=2.0), angular=geometry_msgs.msg.Vector3(x
=1.7, y=1.0, z=1.8))

yahboom@VM:~$ 
```

Fig4: proform some actions by changing the number of the command

Challenge:

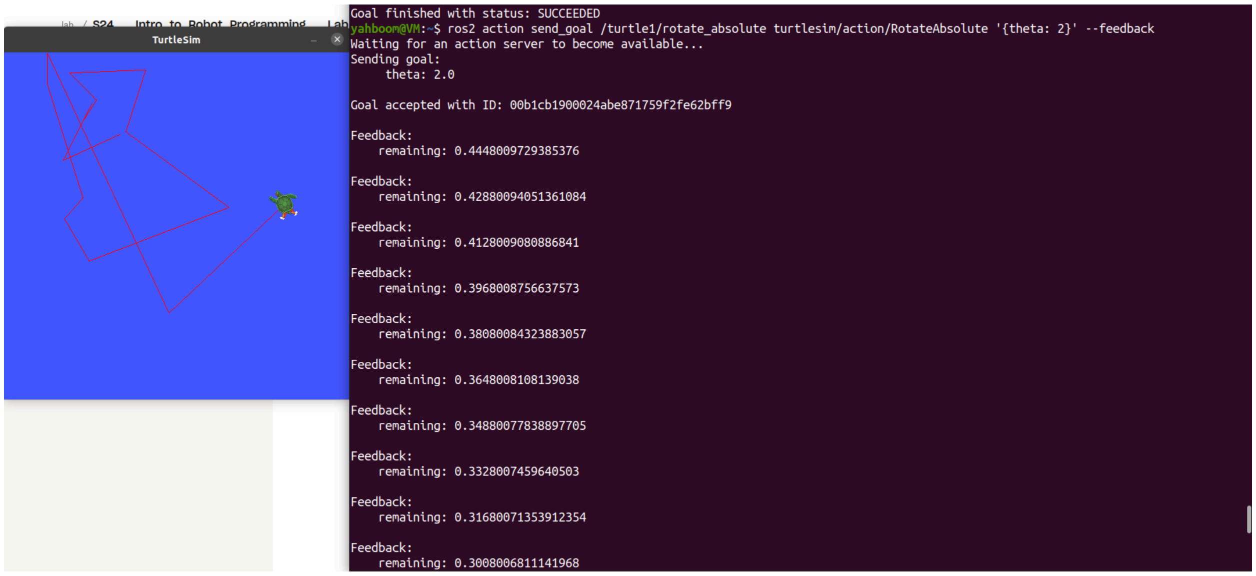


```
uint8 off
---
yahboom@VM:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen '{r: 255, g: 0, b: 0, width: 2, off: 0}'
Failed to populate field: getattr(): attribute name must be string
yahboom@VM:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen '{r: 255, g: 0, b: 0, width: 2, off: 0}'
Failed to populate field: getattr(): attribute name must be string
yahboom@VM:~$ ros2 interface show turtlesim/srv/SetPen
uint8 r
uint8 g
uint8 b
uint8 width
uint8 off
---
yahboom@VM:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen '{r: 255, g: 0, b: 0, width: 2, off: 0}'
Failed to populate field: getattr(): attribute name must be string
yahboom@VM:~$ ros2 service call /turtle1/set_pen r 255
The passed service type is invalid
yahboom@VM:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen r=255
Failed to populate field: Value 'r=255' is expected to be a dictionary but is a str
yahboom@VM:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen r:255
Failed to populate field: Value 'r:255' is expected to be a dictionary but is a str
yahboom@VM:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen '{r: 255, g: 0, b: 0, width: 1, off: 0}'
Failed to populate field: getattr(): attribute name must be string
yahboom@VM:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen '{'r': 255, 'g': 0, 'b': 0, width: 1, off: 0}'
Failed to populate field: getattr(): attribute name must be string
yahboom@VM:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen '{'r': 255, 'g': 0, 'b': 0, 'width': 1, 'off': 0}'
Failed to populate field: getattr(): attribute name must be string
yahboom@VM:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen '{r: 255, g: 0, b: 0, width: 1, off: 0}'
Failed to populate field: getattr(): attribute name must be string
yahboom@VM:~$ ros2 service call /turtle1/set_pen turtlesim/srv/SetPen "{\"r\": 255, \"g\": 0, \"b\": 0, \"width\": 1, \"off\": 0}"
waiting for service to become available...
requester: making request: turtlesim.srv.SetPen_Request(r=255, g=0, b=0, width=1, off=0)

response:
turtlesim.srv.SetPen_Response()

yahboom@VM:~$
```

Fig5: changing the turtle line from white to red

Action:

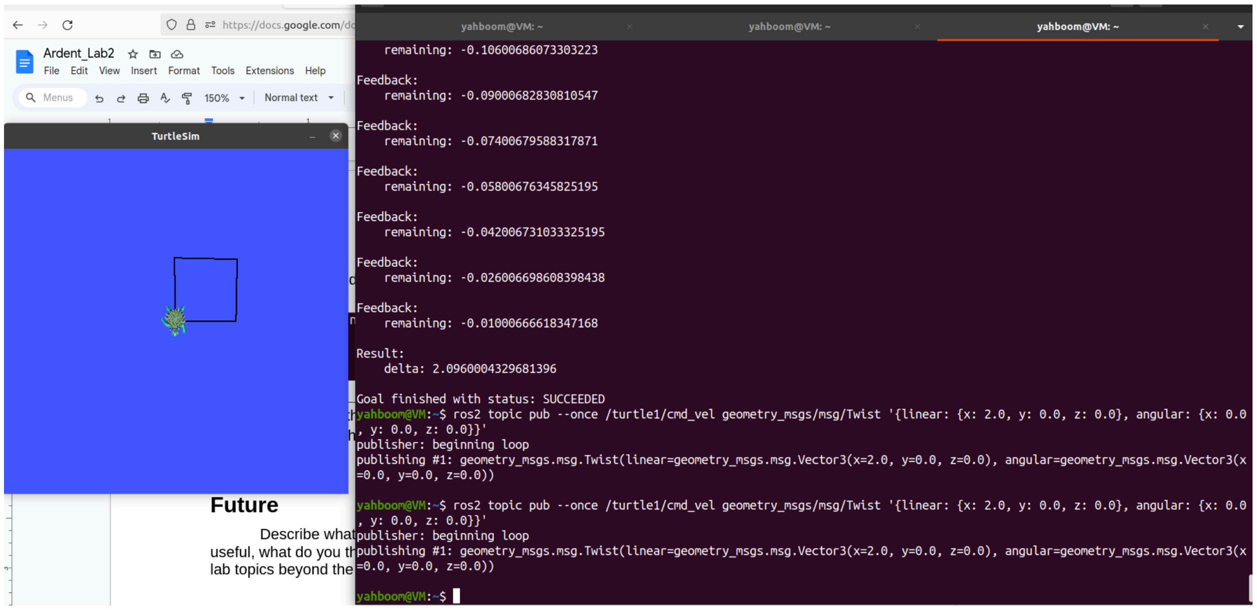
I set the value to 2. Here is the output



```
Goal finished with status: SUCCEEDED
yahboom@VM:~$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute '{theta: 2}' --feedback
Waiting for an action server to become available...
Sending goal:
     theta: 2.0

Goal accepted with ID: 00b1cb1900024abe871759f2fe62bff9

Feedback:
    remaining: 0.4448009729385376

Feedback:
    remaining: 0.42880094051361084

Feedback:
    remaining: 0.4128009080886841

Feedback:
    remaining: 0.3968008756637573

Feedback:
    remaining: 0.38080084323883057

Feedback:
    remaining: 0.3648008108139038

Feedback:
    remaining: 0.34880077838897705

Feedback:
    remaining: 0.3328007459640503

Feedback:
    remaining: 0.31680071353912354

Feedback:
    remaining: 0.3008006811141968
```

Fig6: using a new value to execute the command

Task3:



```
    remaining: -0.10600686073303223

Feedback:
    remaining: -0.09000682830810547

Feedback:
    remaining: -0.07400679588317871

Feedback:
    remaining: -0.05800676345825195

Feedback:
    remaining: -0.042006731033325195

Feedback:
    remaining: -0.026006698608398438

Feedback:
    remaining: -0.01000666618347168

Result:
    delta: 2.0960004329681396

Goal finished with status: SUCCEEDED
yahboom@VM:~$ ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist '{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0
, y: 0.0, z: 0.0}}'
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x
=0.0, y=0.0, z=0.0))

yahboom@VM:~$ ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist '{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0
, y: 0.0, z: 0.0}}'
publisher: beginning loop
publishing #1: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=2.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x
=0.0, y=0.0, z=0.0))

yahboom@VM:~$
```

I use three main commands to reach this goal:

1. This command will start the turtle:
   ros2 run turtlesim turtlesim_node
2. This command sets the RGB value for the turtle's line, the RGB code for black is r:0, g: 0, b: 0.
   ros2 service call /turtle1/set_pen turtlesim/srv/SetPen "{\"r\": 0, \"g\": 0, \"b\": 0, \"width\": 2, \"off\": 0}"
3. This command moves the turtle forward.
   yahboom@VM:~$ ros2 topic pub --once /turtle1/cmd_vel geometry_msgs/msg/Twist '{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}'
4. This command rotate the turtle to a certain direction, for example, 1.57 will rotate the turtle upward, 3.14 will rotate the turtle to the left; while 4.71, which is 1.57*3, will let the turtle face downward.
   yahboom@VM:~$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute '{theta: 3.04}' --feedback

Here is how I draw the square:

1. Restart the turtle program using command 1
2. Using the second command to set the turtle's line to black using r: 0, g: 0, b: 0.
3. By default, the turtle will face right, use the third command to move the turtle forward
4. Using the forth command, with the theta value of 1.57 to let the turtle face upward
5. Repeat the forth command to move the turtle upward
6. Use the forth command, with the theta value of 3.14 to rotate the turtle to face the left
7. Use the third command to move the turtle to the left
8. Use the forth command, with the theta value of 4.71 to rotate the turtle downward
9. Use the third command to move the turtle downward
10. Now we got the complete graph