W1+2: Familiarization with ROS & Python

Material

Familiarize (and write the code) yourself for the following tutorials:

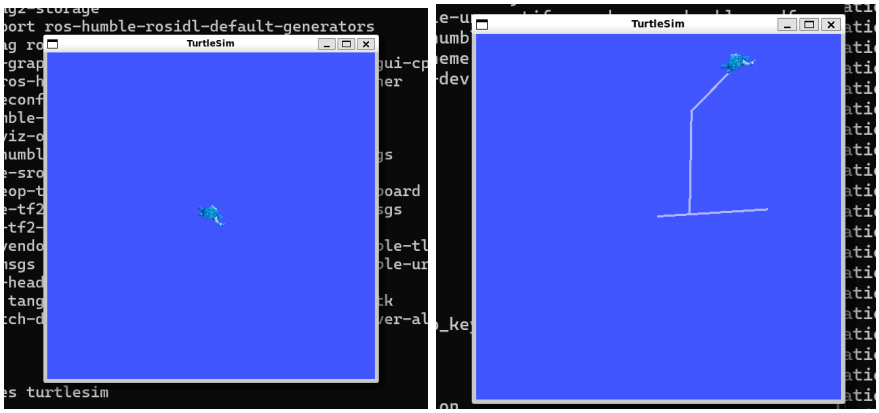
- All tutorials here: https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools.html
- The following tutorials: Colcon, Create workspace, Create package, pubsub python, service python, custom messages and services, using params
- The following tutorial: https://ros2-industrial-workshop.readthedocs.io/en/latest/_source/basics/ROS2-Turtlesim.html#

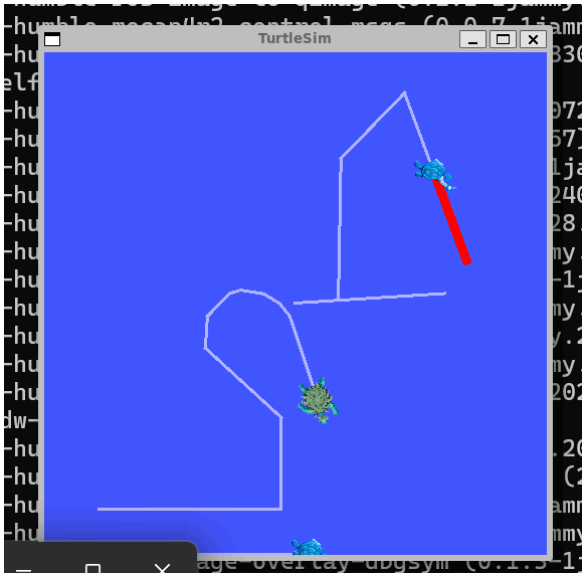
https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Configuring-ROS2-Environment.html

Goated tutorial ^
export ROS_DOMAIN_ID= 70

9/29: TurtleSim Lol(1)
https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Introducing-Turtlesim/Introducing-Turtlesim.html
Instead of directly running Ubuntu 22.04, run WSL on Command prompt, which should open ubuntu 22.04.

Struggled for 2 hours on getting xeyes and display to work. After searching online





Finished 3 turtles:

Understanding Nodes (2) :
Useful Commands:

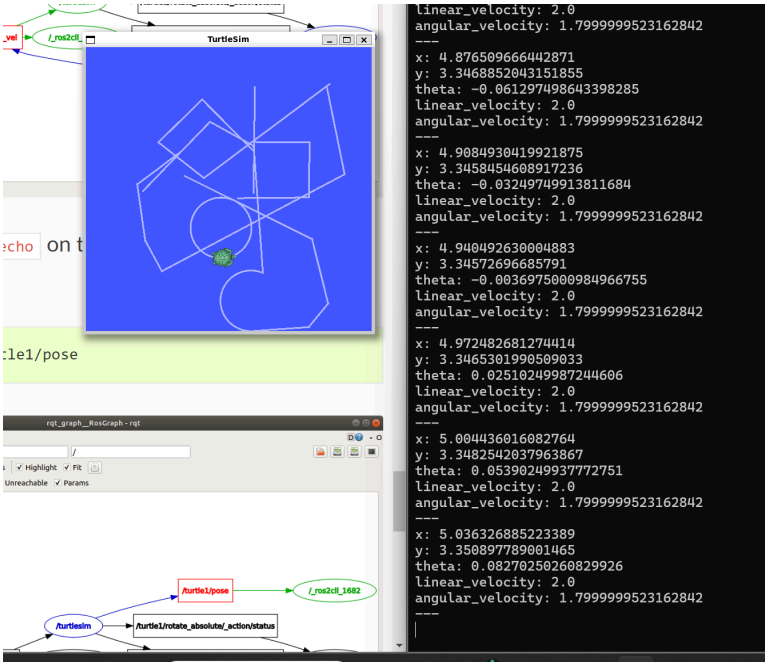
sudo apt update first

- ros2 run
- ros2 pkg executables <node>
- ros2 node list </node>
- ros2 node info </node>

Understanding Topics (3):
topics are the “bus” which nodes publish across

Commands:

- ros2 topic pub <topic_name> <msg_type> '<args>'
- ros2 topic list / ros2 topic list -t
- **ros2 topic pub**
- ex:
  - **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: 1.8}}"**
  - 
  - **ros2 topic pub --rate 1 /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: 1.8}}"**
  - 
  - **ros2 topic echo /turtle1/pose**



- **ros2 topic pub /pose geometry_msgs/msg/PoseStamped '{header: "auto", pose: {position: {x: 1.0, y: 2.0, z: 3.0}}}'**

- **ros2 topic hz <>**
  - ex: ros2 topic hz /turtle1/pose
    - publishes the rate at which the “pose” topic is being updated

<u>Understanding Services (4):</u>

- Topics are publisher-subscriber based
- Services are call and receiver based
  - ex: node one sends a request for data from node 2, and node 2 send a message with data

Commands:

- **ros2 service list**: lists the services available to be used in the system
- **ros2 service type <service_name>**: provides type of service
- NOTE: adding **“-t”** will add on the type
- **ros2 service find <type_name>**: returns type
  - EX: **ros2 service find std_srvs/srv/Empty** will return /clear and /reset
- ros2 interface show <type_name>:
  - ros2 interface show turtlesim/srv/Spawn
    - float32 x float32 y float32 theta
    - string name # Optional. A unique name will be created and returned if this is empty
    - ---
    - string name

★ **ros2 service call <service_name> <service_type> <arguments>**

- ros2 service call /clear std_srvs/srv/Empty // SERVICE WHICH CLEARS TURTLE PATH
- ros2 service call /spawn turtlesim/srv/Spawn "{x: 2, y: 2, theta: 0.2, name: ''}"
- ^// SERVICE WHICH SPAWNS A NEW TURTLE
  requester: making request: turtlesim.srv.Spawn_Request(x=2.0, y=2.0, theta=0.2, name='')
  response:
  turtlesim.srv.Spawn_Response(name='turtle2')

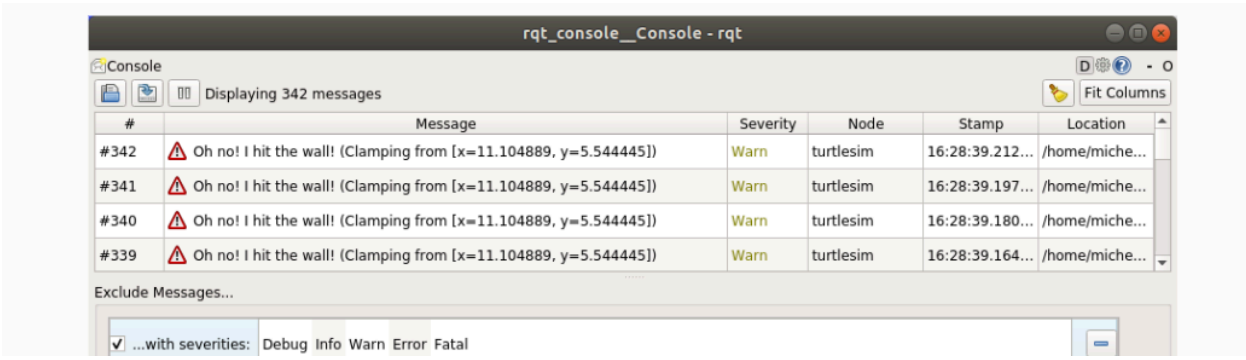
Understanding Parameter (5):

- **ros2 param list**
  - lists node namespaces, and the parameters each node has
- **ros2 param set**
  - ros2 param set /turtlesim background_r 0
    - Sets the parameter “background_r” under /turtlesim node to 0
- **ros2 param dump /turtlesim**
  - lists out the parameter values of a node
- ros2 param dump /turtlesim > turtlesim.yaml
- ros2 param load /turtlesim turtlesim.yaml
  - ^loads the parameters of a file (turtlesim.yaml) into the current node
- **ros2 run <package_name> <executable_name> --ros-args --params-file <file_name>**
  - ros2 run turtlesim turtlesim_node --ros-args --params-file turtlesim.yaml
  - **Will run turtlesim_node but with the preset parameters of turtlesim.yaml**
- ★ YAML FILES are used to define parameters for ROS nodes
- ★ TLDR; you can Get / set parameters of ros2 nodes

Understanding Actions (6):

- Actions are services with feedback, rather than single responses
- running “ros2 node info /turtlesim” will show the **action servers** and **action clients** of a node
- **ros2 action list -t**
  - shows the actions available
- **ros2 interface show turtlesim/action/RotateAbsolute**
- **ros2 action send_goal**
  - EX: ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: 1.57}"
- –feedback
  - EX: ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: -1.57}" **--feedback’**
    - **shows angle remaining**
- ★ **ACTIONS are CANCELLABLE, provide FEEDBACK(--feedback), used for Navigation**

rqt console (7):

- startup: **ros2 run rqt_console rqt_console**
- **EXAMPLE OF RQT CONSOLE**
    - **ros2 topic pub -r 1 /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}}"**
    - 

- **ROS2 Logger Levels**
    - Fatal
    - Error
    - Warn
    - Info
    - Debug
- ★ **Use RQT Logger to debug and find errors**

Launching Nodes (8):

(1) ros2 launch turtlesim multisim.launch.py

```
# turtlesim/launch/multisim.launch.py

from launch import LaunchDescription
import launch_ros.actions

def generate_launch_description():
    return LaunchDescription([
        launch_ros.actions.Node(
            namespace= "turtlesim1", package='turtlesim', executable='turtlesim_node',
output='screen'),
        launch_ros.actions.Node(
            namespace= "turtlesim2", package='turtlesim', executable='turtlesim_node',
output='screen'),
    ])
```

(2) Run commands for each terminal

ros2 topic pub /turtlesim1/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: 1.8}}"

ros2 topic pub /turtlesim2/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: -1.8}}"

Recording and playing back data (FINAL):

- ros2 bag record <topic_name>
  - ros2 bag record /turtle1/cmd_vel
    - records data published by /turtle1/cmd_vel

```
matthewchen132@DESKTOP-9SIE7E0:/mnt/c/Users/awolf/bag_files$ ros2 bag record -o subset
/turtle1/cmd_vel /turtle1/pose
[INFO] [1727731120.727866832] [rosbag2_recorder]: Press SPACE for pausing/resuming
[INFO] [1727731120.770012404] [rosbag2_storage]: Opened database 'subset/subset_0.db3'
for READ_WRITE.
[INFO] [1727731120.771204520] [rosbag2_recorder]: Listening for topics...
[INFO] [1727731120.771226349] [rosbag2_recorder]: Event publisher thread: Starting
[INFO] [1727731120.776159176] [rosbag2_recorder]: Subscribed to topic '/turtle1/pose'
[INFO] [1727731120.780798781] [rosbag2_recorder]: Subscribed to topic '/turtle1/cmd_vel
'
[INFO] [1727731120.780943625] [rosbag2_recorder]: Recording...
[INFO] [1727731120.781161728] [rosbag2_recorder]: All requested topics are subscribed.
Stopping discovery...
[INFO] [1727731263.974125770] [rosbag2_cpp]: Writing remaining messages from cache to t
he bag. It may take a while
[INFO] [1727731264.000205146] [rosbag2_recorder]: Event publisher thread: Exiting
[INFO] [1727731264.000483014] [rosbag2_recorder]: Recording stopped
matthewchen132@DESKTOP-9SIE7E0:/mnt/c/Users/awolf/bag_files$ ros2 bag info subset

Files:             subset_0.db3
Bag size:          569.4 KiB
Storage id:        sqlite3
Duration:          143.184181346s
Start:             Sep 30 2024 14:18:40.786654091 (1727731120.786654091)
End:               Sep 30 2024 14:21:03.970835437 (1727731263.970835437)
Messages:          9104
Topic information: Topic: /turtle1/cmd_vel | Type: geometry_msgs/msg/Twist | Count: 154
 | Serialization Format: cdr
                   Topic: /turtle1/pose | Type: turtlesim/msg/Pose | Count: 8950 | Seri
alization Format: cdr
```

  - records using ros2 bag record
  - “-o” is used to create an original name. “subset” is the new name
  - **ros2 bag info subset** → information about the file “subset”
  - **ros2 bag play subset** → follows the path recorded
  -