01 Navigation2

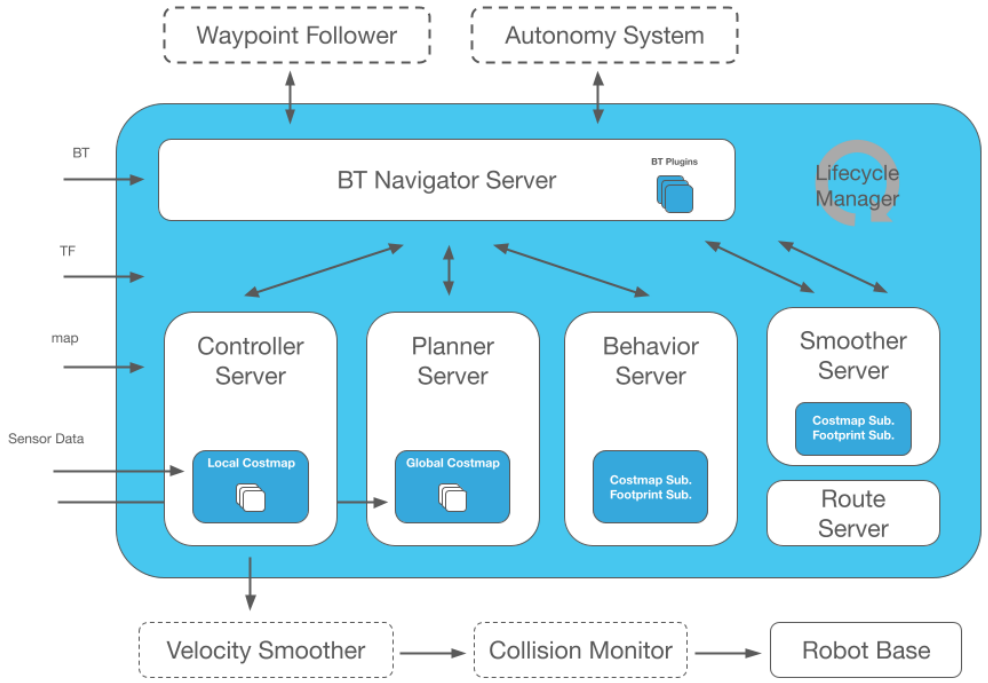
Nav2

Nav2란?

- 자율 주행 차량에 사용되는 기술을 기반으로 개발되워 모바일 및 지면 로봇용으로 최적화 및 재작업된 ROS 네비게이션 스택의 전문적으로 지원된다.

- 모듈화를 통해 Planner 및 Control 알고리즘을 쉽게 교체할 수 있도록 한다.

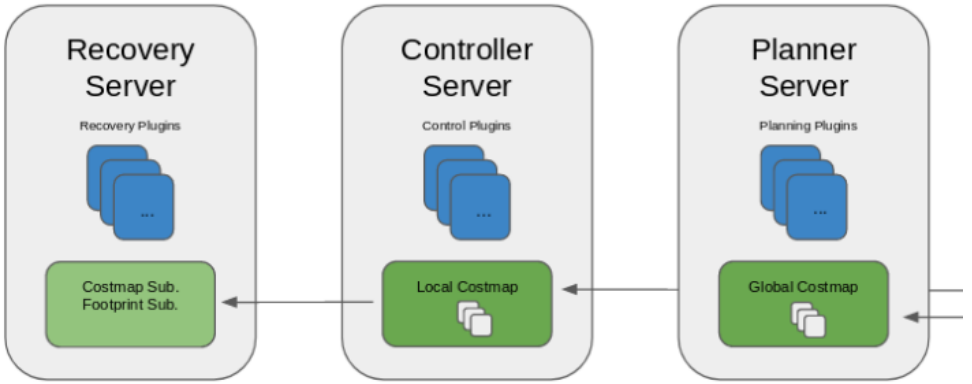
- Python 혹은 다른 프로그래밍 언어로 Planner / Controller 작성이 가능하다.



Nav2 – Modules

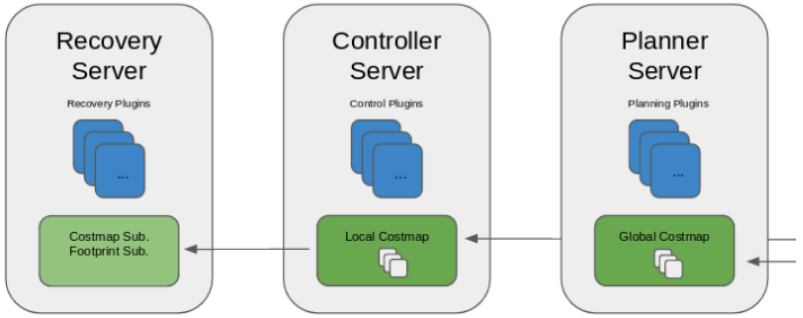
서버들의 구조

- Allow for:
  - Run-time selection of custom algorithms
  - Fully-replaceable servers for power-users
  - Leveraging multi-core processors



- Design pattern:
  - Action server interface to handle goals, provide feedback, and return results
  - Map of plugins to support N algorithms per server to use in unique contexts
  - Environmental representation based on algorithm needs for zero-latency access

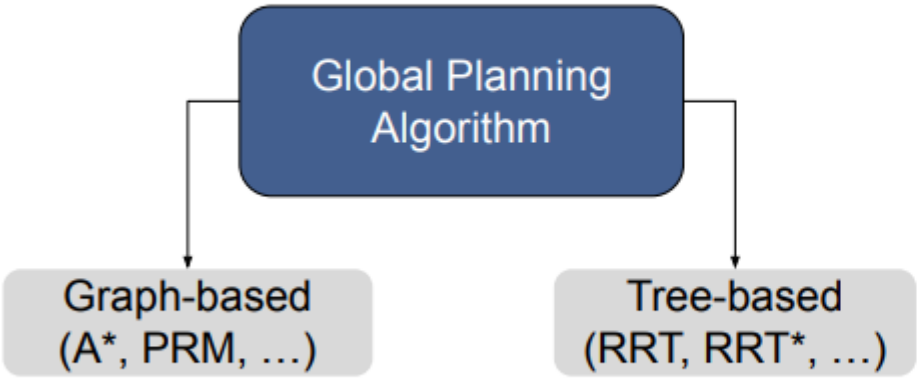
Nav2 – Modules

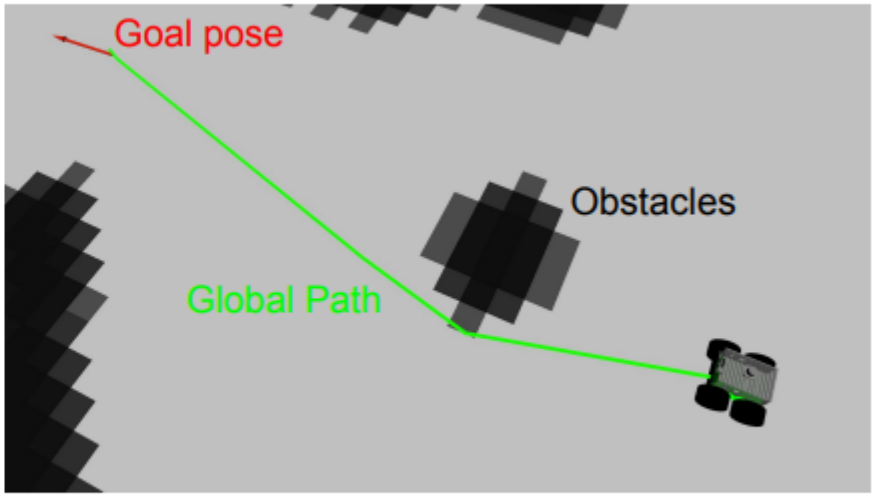


- 임의의 자율 주행을 Nav2 Pkg를 이용하여 구현하려면,
  - Planner Server : Grid기반 Nav2 Planner (A* or Dijkstra)
  - Controller Server : DWB Controller 등 선택
  - Behavior Server : failure case에 대한 recovery behavior 처리

(behavior plugin: spin, backup, drive on heading, wait, assisted teleop)

- Path from current state to global goal
- Optimistic → unknown space = free

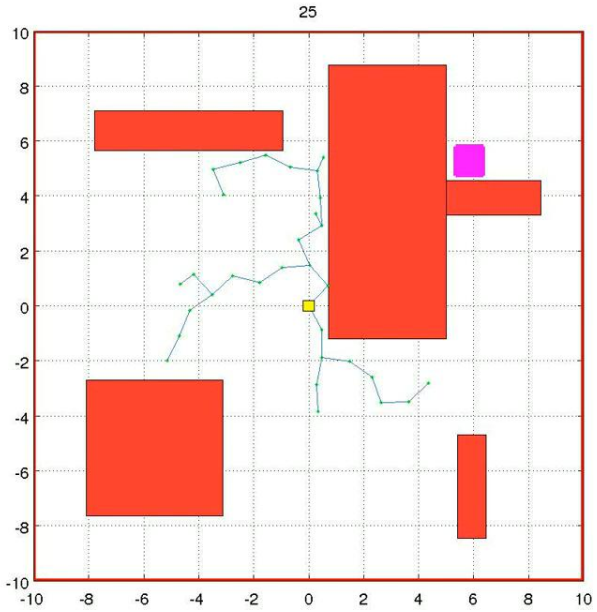




OMPL (Open Motion Planning Library) - https://ompl.kavrakilab.org

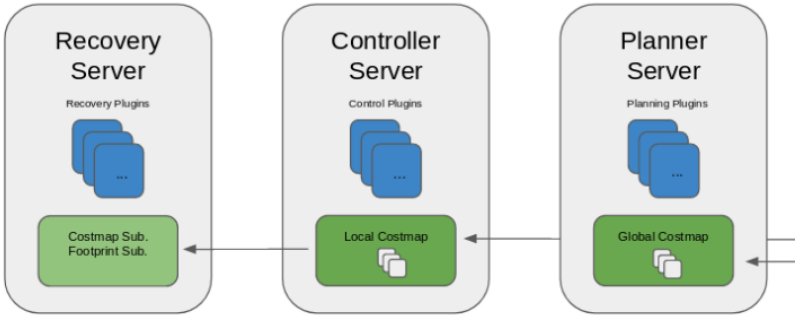
Nav2 – Modules

ROS2 Navigation2



Nav2 – Modules

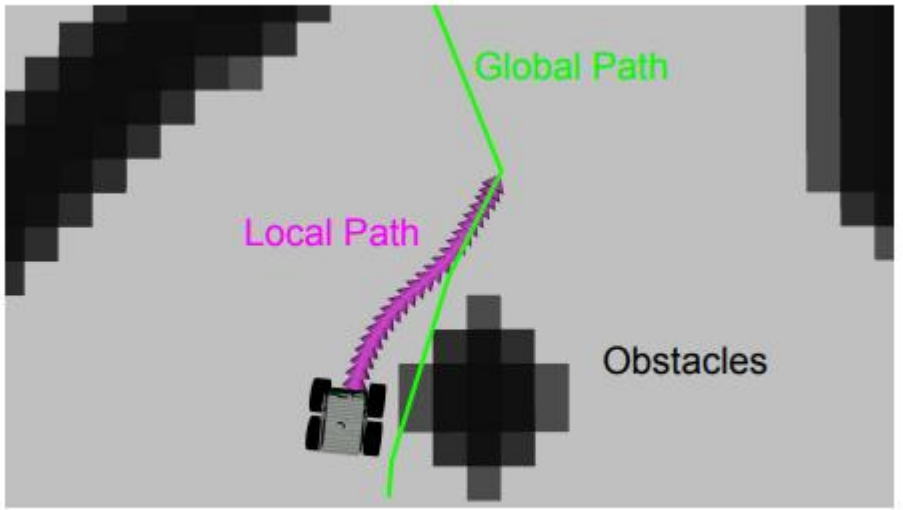
ROS2 Navigation2



- 임의의 자율 주행을 Nav2 Pkg를 이용하여 구현하려면,
  - Planner Server : Grid기반 Nav2 Planner (A* or Dijkstra)
  - Controller Server : DWB Controller 등 선택
  - Behavior Server : failure case에 대한 recovery behavior 처리

(behavior plugin: spin, backup, drive on heading, wait, assisted teleop)

- Compute locally optimal paths
  - Vehicle dynamics
  - Obstacle avoidance
- Pessimistic
  → unknown space = occupied



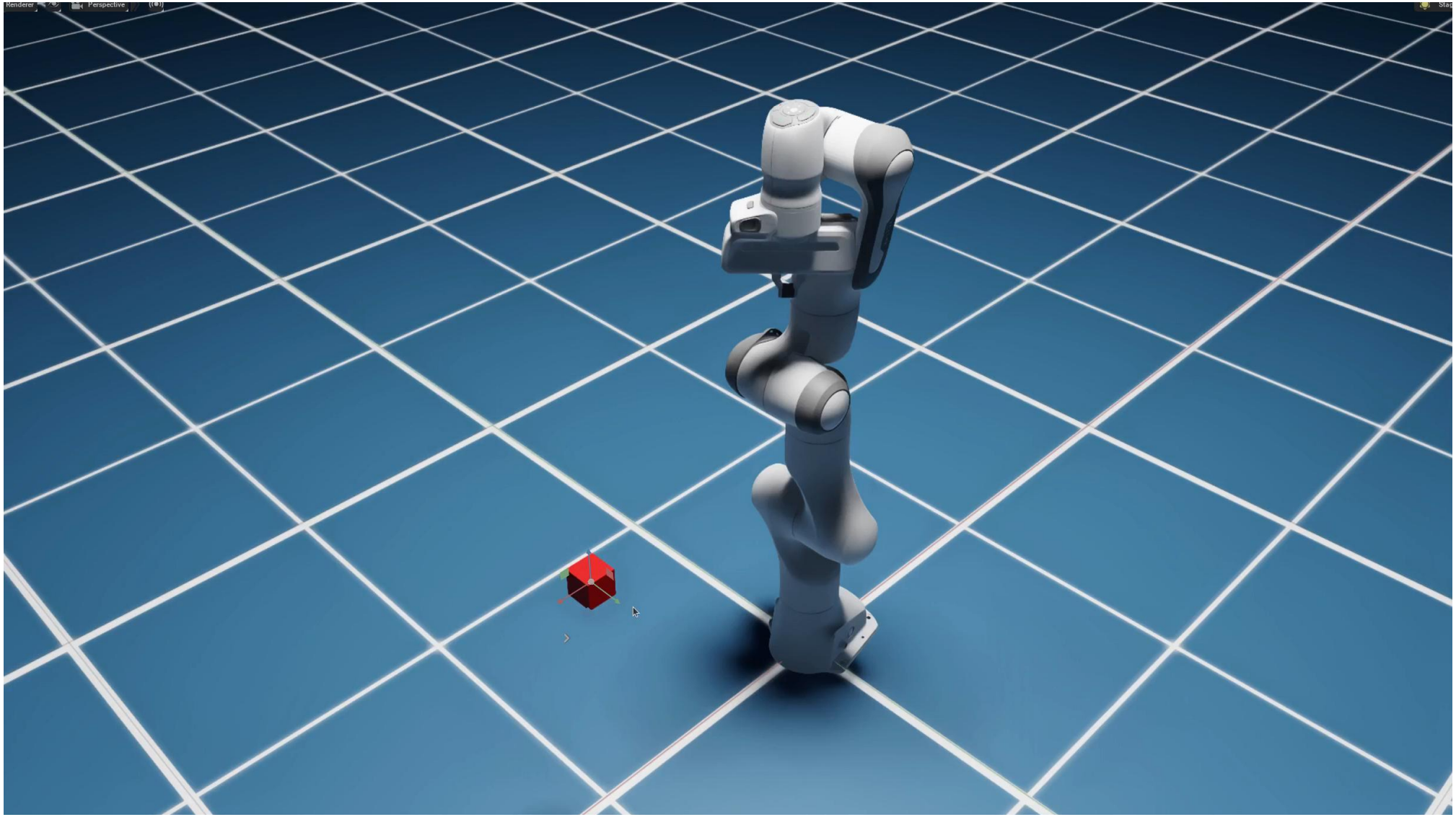
technische universität
dortmund

teb_local_planner

An optimal trajectory planner for mobile robots based on Timed-Elastic-Bands

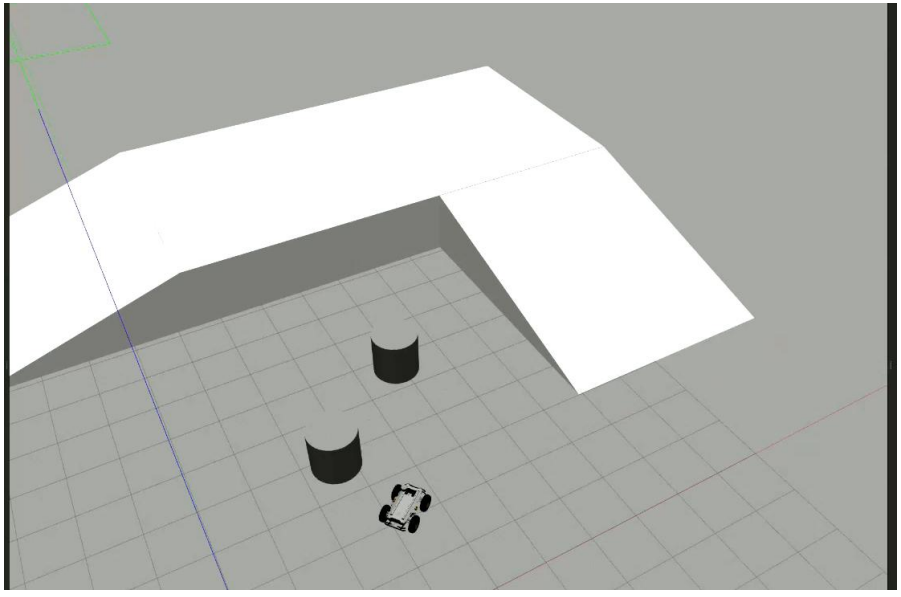
http://wiki.ros.org/teb_local_planner

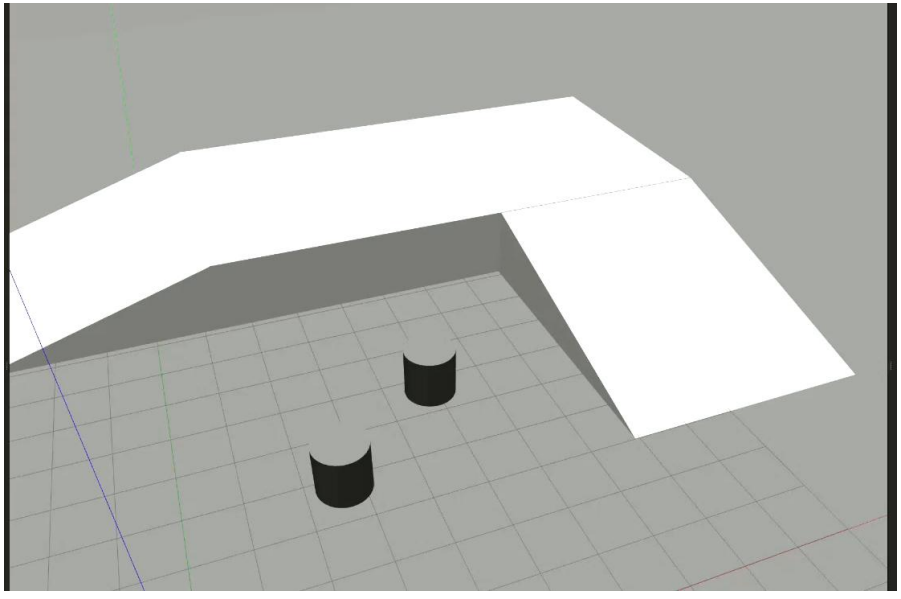
Nav2 – Modules

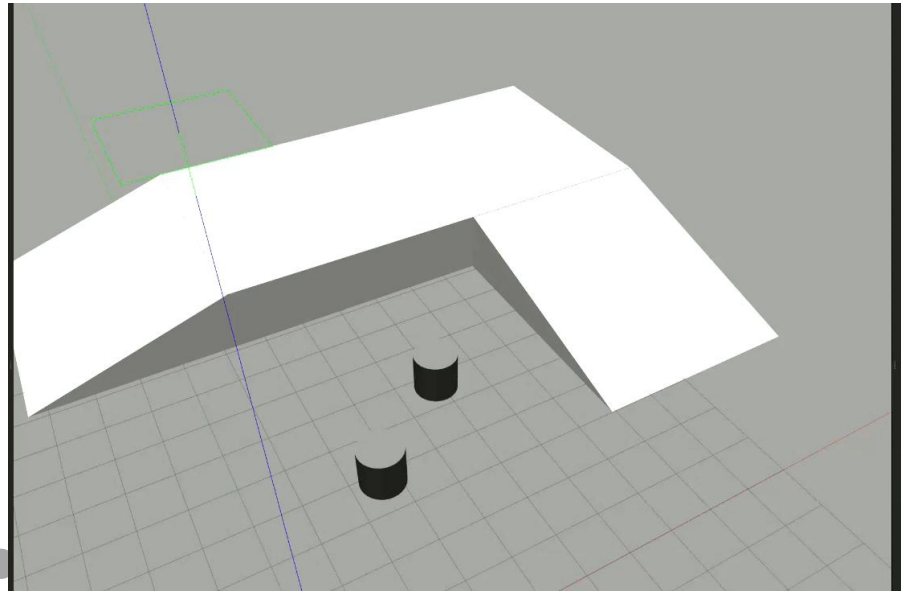


Nav2 – Modules

ROS2 Navigation2 (DWB, MPPI, Pure-Pursuit, TEB의 예)

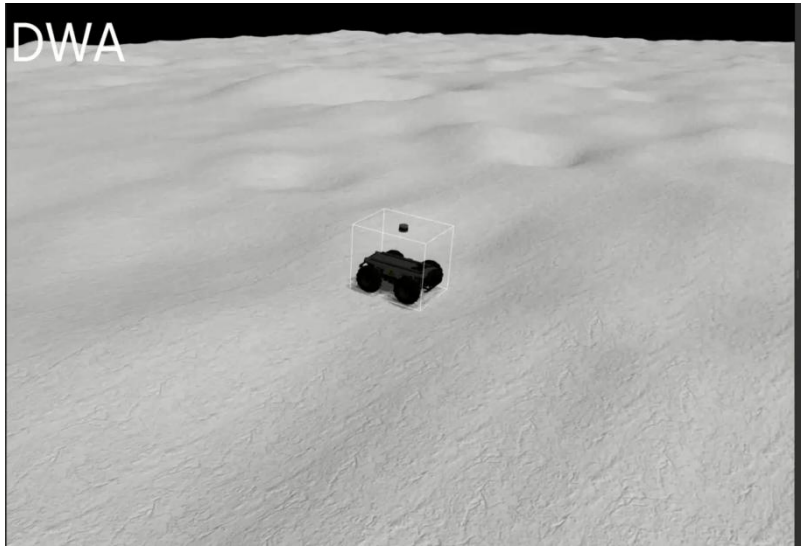






Nav2 – Modules

ROS2 Navigation2 (DWB, MPPI, Pure-Pursuit, TEB의 예)

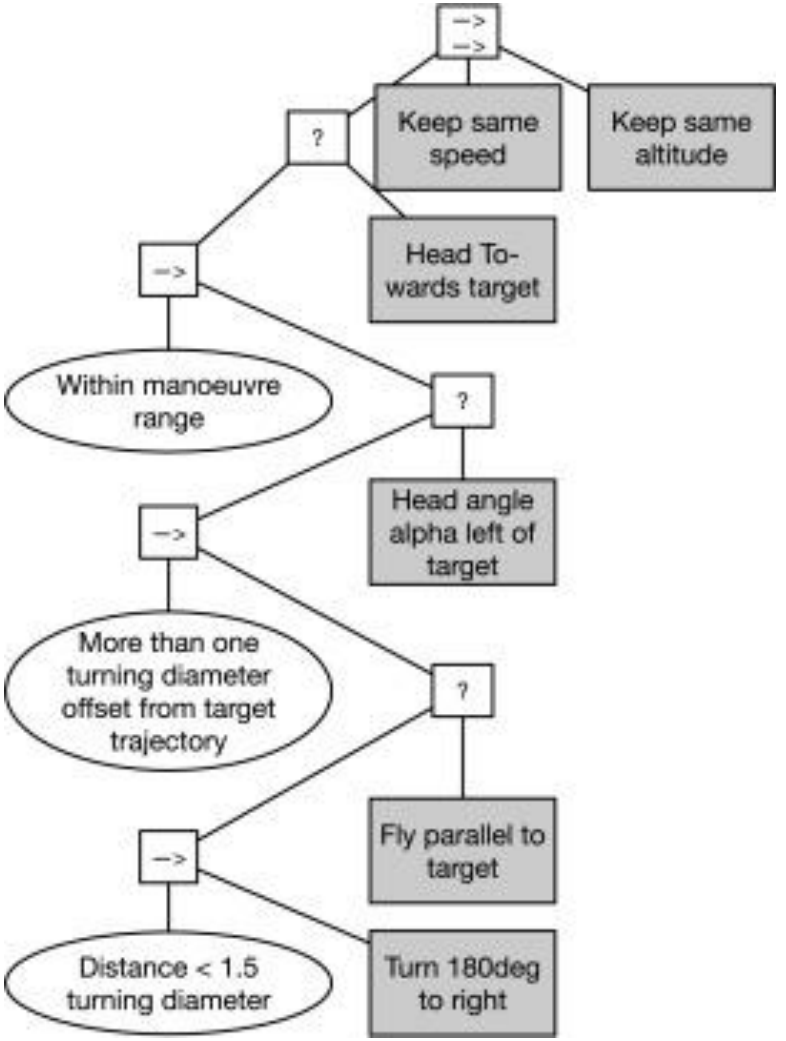


Nav2 – Behavior Tree

- Behavior Trees
  - Tree-based execution model and task planner
  - Model more complex tasks than practical with FSM

- Behavior Trees
  - Plugin nodes: control, action, decorator, condition
  - Use custom nodes to extend or change behavior
    - Elevator and automatic door API
    - Multiple path and trajectory planners in context
    - Dynamic object following

- BT Navigator
  - Runtime load custom behavior tree XML files
  - BT node may call specialized task server in other processes
    - Planner, Controller, Recovery Servers
  - BT node may compute some value itself



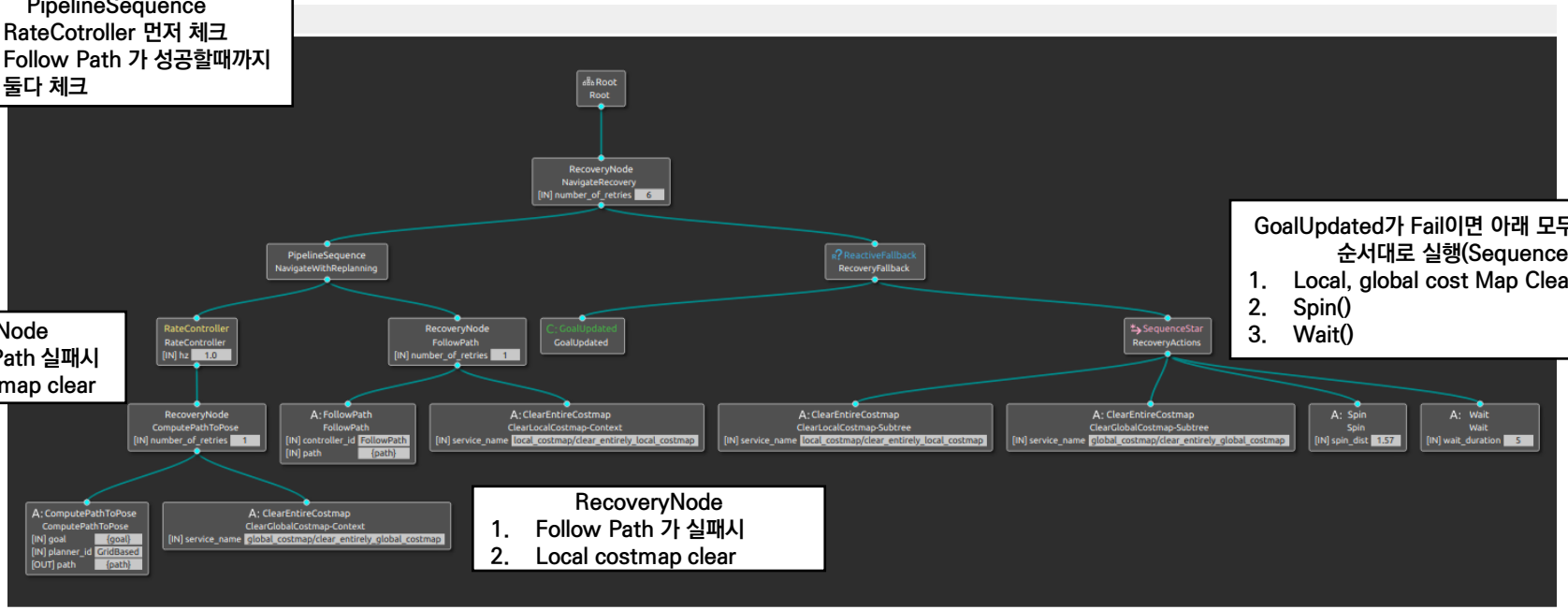
Nav2 – Behavior Tree

Waypoint Follower 시나리오 구현 예시

- Behavior Tree XML 작성 예시
  - Navigation to Pose With Replanning and Recovery\
  - Global path를 1초마다 replan하면서 recovery action을 포함
  - 다양한 BT XML [구성 방법]

```
<root main_tree_to_execute="MainTree">
  <BehaviorTree ID="MainTree">
    <RecoveryNode number_of_retries="6" name="NavigateRecovery">
      <PipelineSequence name="NavigateWithReplanning">
        <RateController hz="1.0">
          <RecoveryNode number_of_retries="1" name="ComputePathToPose">
            <ComputePathToPose goal="{goal}" path="{path}" planner_id="GridBased"/>
            <ClearEntireCostmap name="ClearGlobalCostmap-Context" service_name="global_costmap/clear_entirely_global_costmap"/>
          </RecoveryNode>
        </RateController>
        <RecoveryNode number_of_retries="1" name="FollowPath">
          <FollowPath path="{path}" controller_id="FollowPath"/>
          <ClearEntireCostmap name="ClearLocalCostmap-Context" service_name="local_costmap/clear_entirely_local_costmap"/>
        </RecoveryNode>
      </PipelineSequence>
      <ReactiveFallback name="RecoveryFallback">
        <GoalUpdated/>
        <SequenceStar name="RecoveryActions">
          <ClearEntireCostmap name="ClearLocalCostmap-Subtree" service_name="local_costmap/clear_entirely_local_costmap"/>
          <ClearEntireCostmap name="ClearGlobalCostmap-Subtree" service_name="global_costmap/clear_entirely_global_costmap"/>
          <Spin spin_dist="1.57"/>
          <Wait wait_duration="5"/>
        </SequenceStar>
      </ReactiveFallback>
    </RecoveryNode>
  </BehaviorTree>
</root>
```

PipelineSequence
1. RateCotroller 먼저 체크
2. Follow Path 가 성공할때까지 둘다 체크



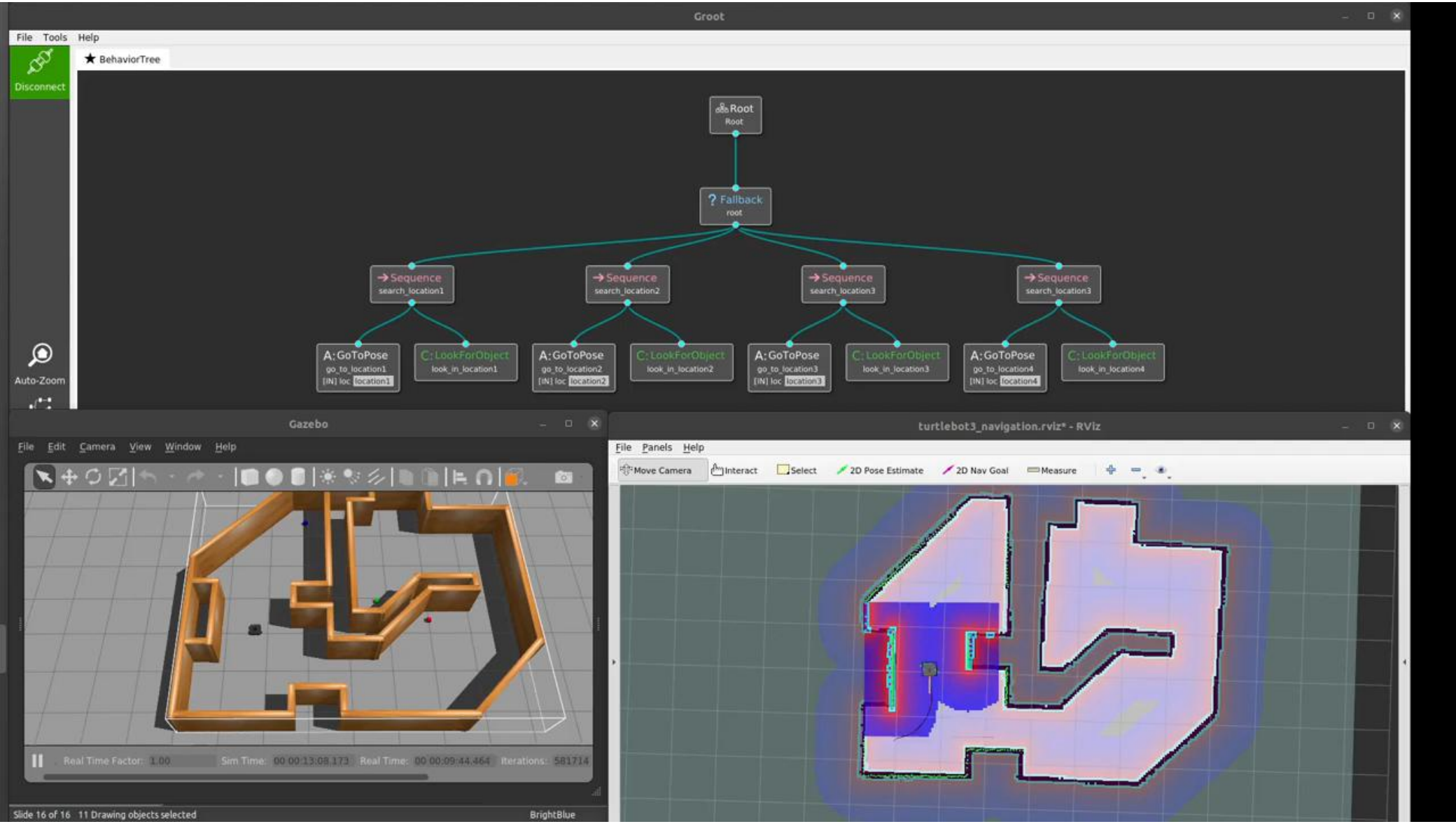
RecoveryNode
1. CompuatePath 실패시
2. Global costmap clear

GoalUpdated가 Fail이면 아래 모두 성공할때까지 순서대로 실행(SequenceStar)
1. Local, global cost Map Clear()
2. Spin()
3. Wait()

RecoveryNode
1. Follow Path 가 실패시
2. Local costmap clear

https://navigation.ros.org/behavior_trees/overview/nav2_specific_nodes.html (BT 노드 설명)

Navigation2 – Behavior-Tree Navigation



Navigation 2 개요

Waypoint Follower 시나리오 구현 예시

- Planner 서버 구현
  - Global Planner 서버 (nav2__planner/planner_server.hpp) : Action 서버로 구성

```
planner_server:
  ros__parameters:
    expected_planner_frequency: 20.0
    planner_plugins: ['GridBased']
    GridBased:
      plugin: 'nav2_navfn_planner/NavfnPlanner'
```

  - Waypoint Follower 클래스 (LifeCycleNode 상속), Action Server 구성
  - void computePlanThroughPoses(), computePlan()함수에서 기능 구현 (getPlan()함수 호출)
  - 실제 알고리즘 구현은 plugin에 의해서 구현 (NavfnPlanner, SmacPlanner, ThetaStarPlanner)

https://github.com/ros-planning/navigation2/blob/main/nav2_planner/include/nav2_planner/planner_server.hpp
https://github.com/ros-planning/navigation2/tree/main/nav2_navfn_planner
https://github.com/ros-planning/navigation2/blob/main/nav2_core/include/nav2_core/global_planner.hpp

02 Simulation

Simulation Setting

- Prerequisite (Lecture 11의 코드를 navigation2_ws/src에서 압축을 풀어 주세요.)

Dependencies

```
$ sudo apt install ros-humble-navigation2 ros-humble-nav2-bringup

$ sudo apt install ros-humble-turtlebot3*

$ sudo apt install ros-humble-slam-toolbox
```

Bashrc (gedit ~/.bashrc 에서)

```
source ${HOME}/navigation_ws/install/setup.bash
export GAZEBO_MODEL_PATH=:/opt/ros/humble/share/turtlebot3_gazebo/models:${your_root}/navigation_ws/src/neo_simulation2/models:$GAZEBO_MODEL_PATH

export MY_ROBOT=mpo_500
export MAP_NAME=neo_workshop
export TURTLEBOT3_MODEL=waffle
```

입력 후 터미널에 source ~/.bashrc 입력 (사용하는 모든 터미널에서 해야합니다)

your_root는 사용자의 Home directory의 위치!

03 SLAM

SLAM

SLAM 노드 실행

Terminal 1

```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

Terminal 2

```
$ ros2 launch nav2_bringup navigation_launch.py use_sim_time:=True
```

Terminal3

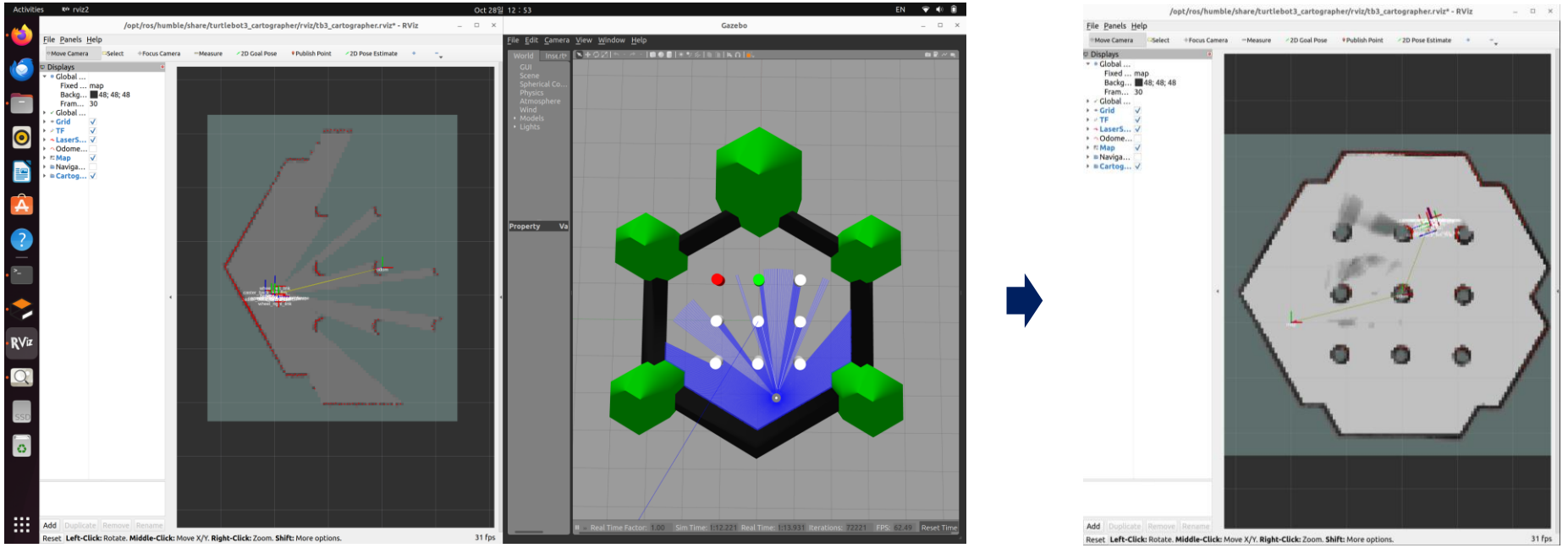
```
$ ros2 launch slam_toolbox online async_launch.py use_sim_time:=True
```

Terminal 4

```
$ ros2 run rviz2 rviz2 –d /opt/ros/humble/share/nav2_bringup/rviz/nav2_default_view.rviz
```

SLAM

SLAM 노드 실행



Terminal – 활성화 후 Turtlebot을 움직여서 맵을 그려보세요

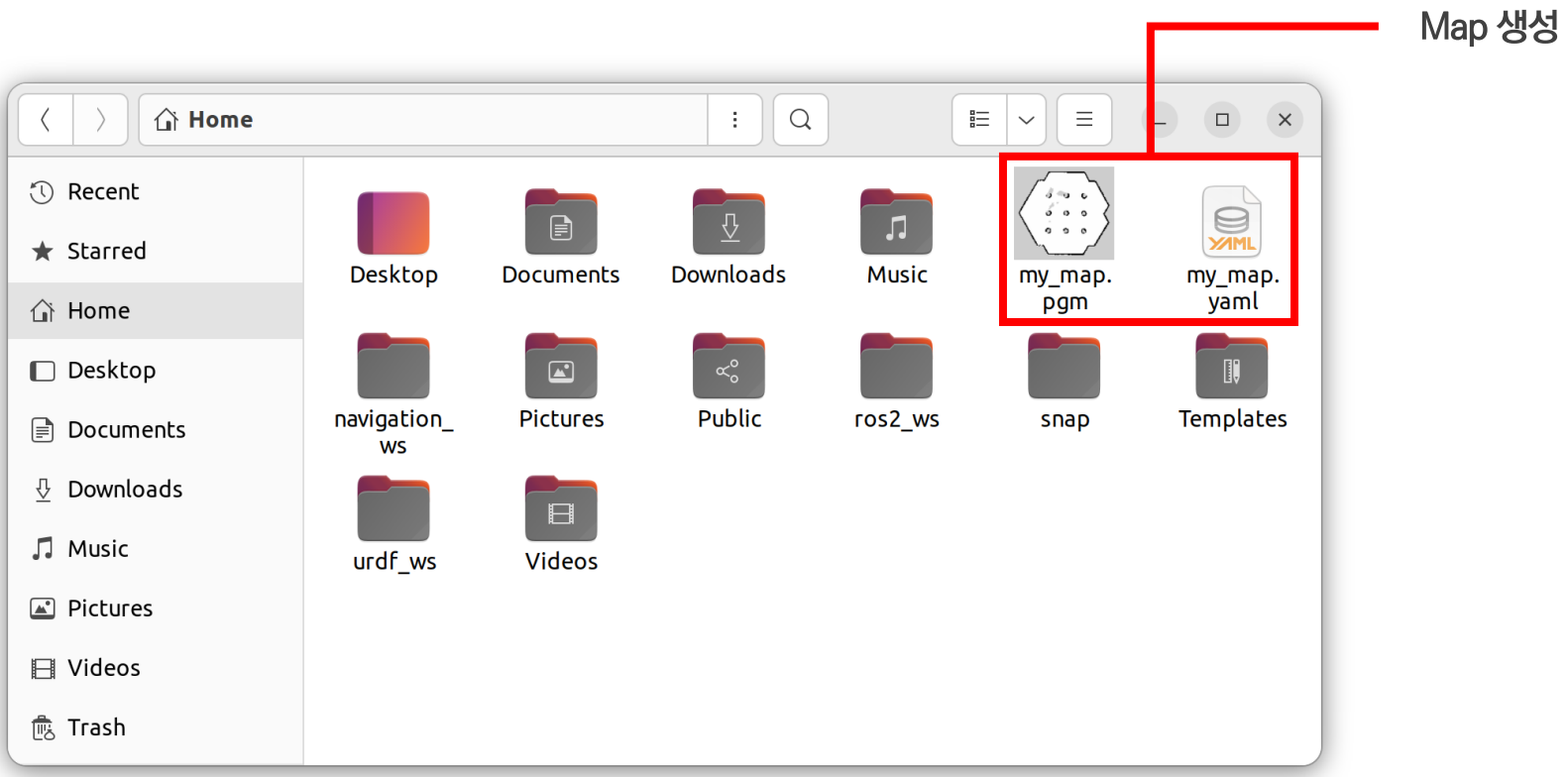
```
$ ros2 run turtlebot3_teleop teleop_keyboard
```

Terminal (현재 Terminal의 위치에 Map이 형성!)

```
$ ros2 run nav2_map_server map_saver_cli –f my_map
```

SLAM

SLAM 노드 실행



04 Navigation

Navigation

Navigation 노드 실행

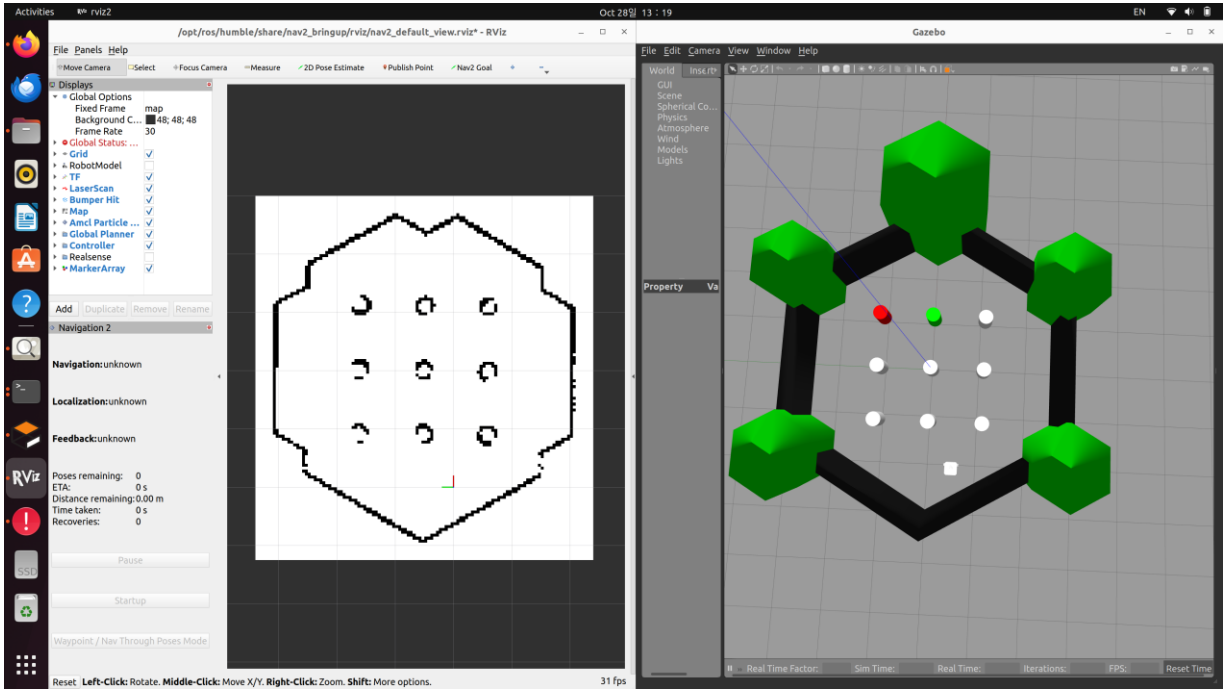
Terminal 1

```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

Terminal 2

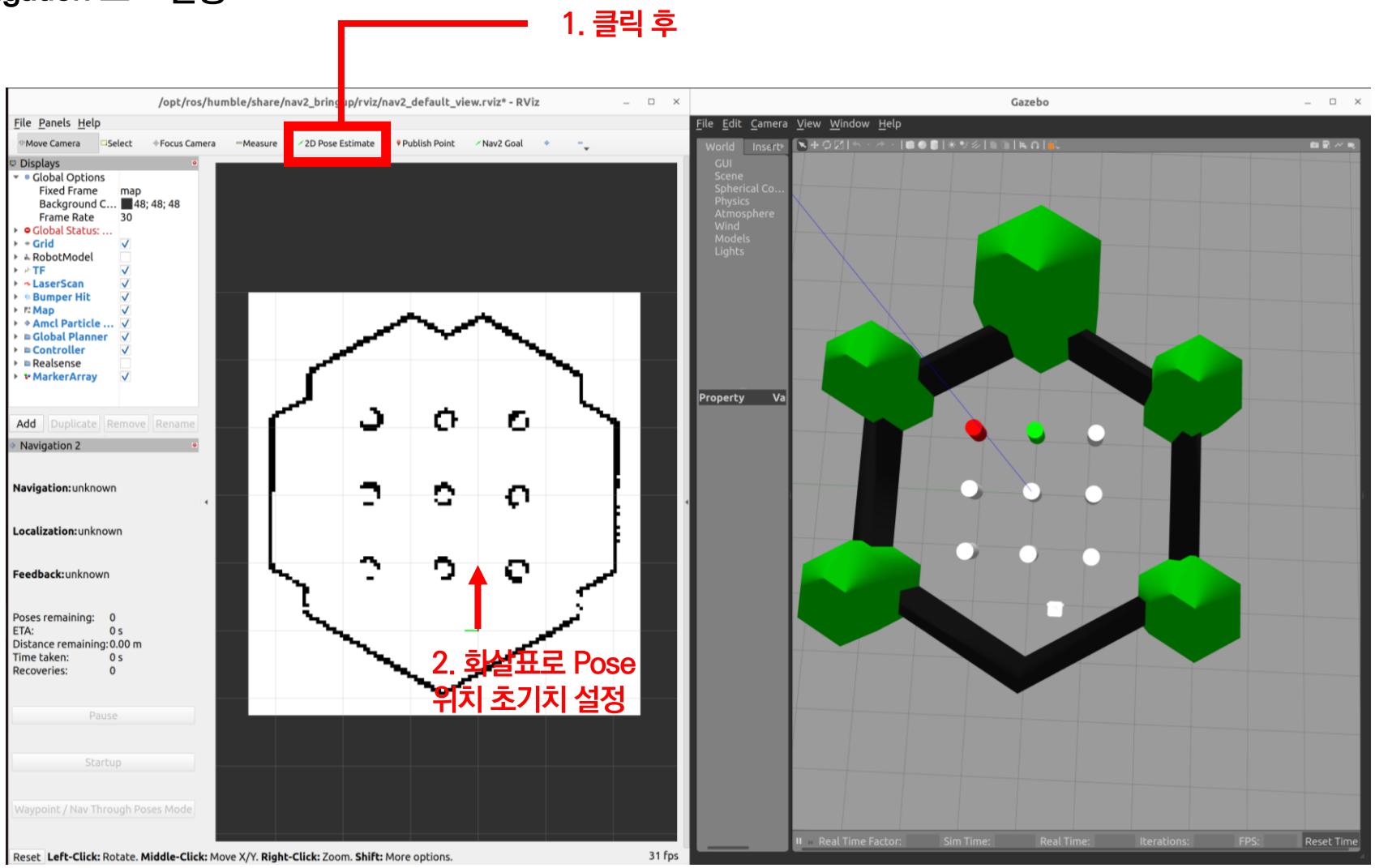
```
$ ros2 launch turtlebot3_navigation2 navigation2.launch.py use_sim_time:=True map:=path/to/my_map.yaml
```

Home에 위치한다면 ${HOME}/my_map.yaml



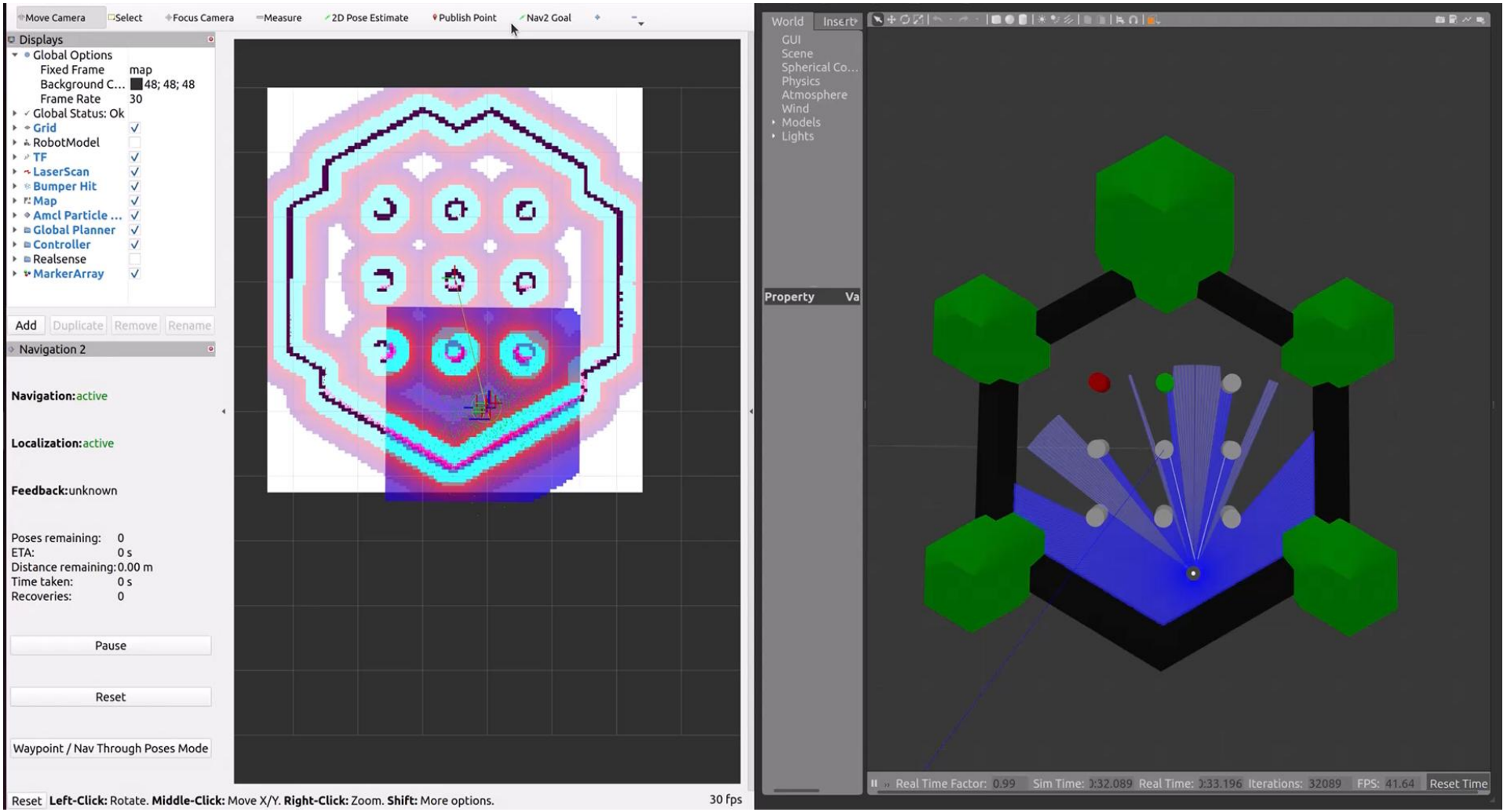
Navigation

Navigation 노드 실행



Navigation

Navigation 노드 실행



Neobotix Navigation

Navigation 노드 실행

Terminal 1

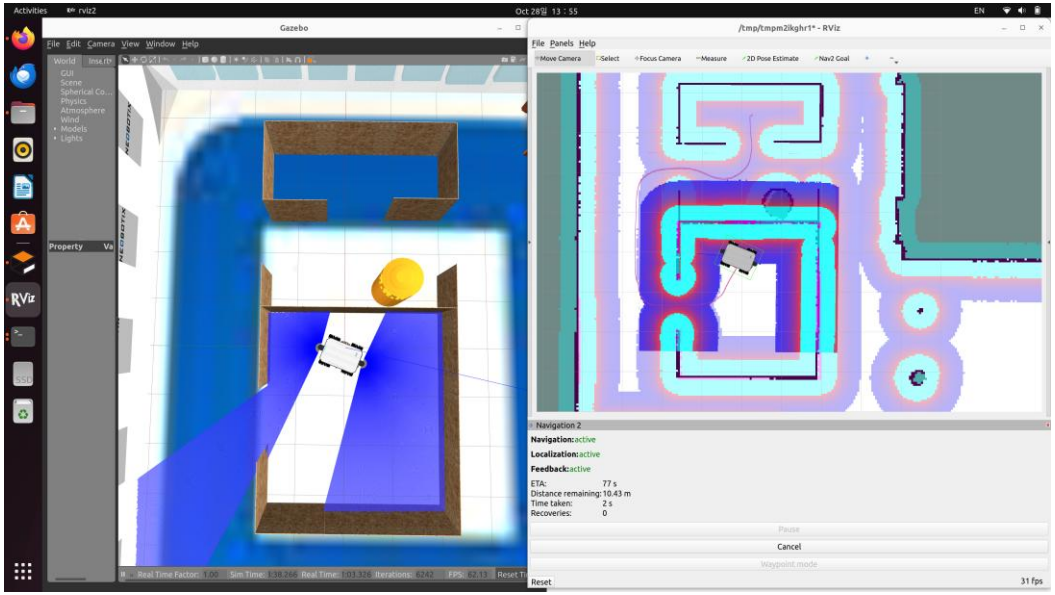
```
$ ros2 launch neo_simulation2 simulation.launch.py
```

Terminal 2

```
$ ros2 launch neo_simulation2 navigation.launch.py
```

Terminal 3

```
$ ros2 launch neo_nav2_bringup rviz_launch.py
```



감사합니다