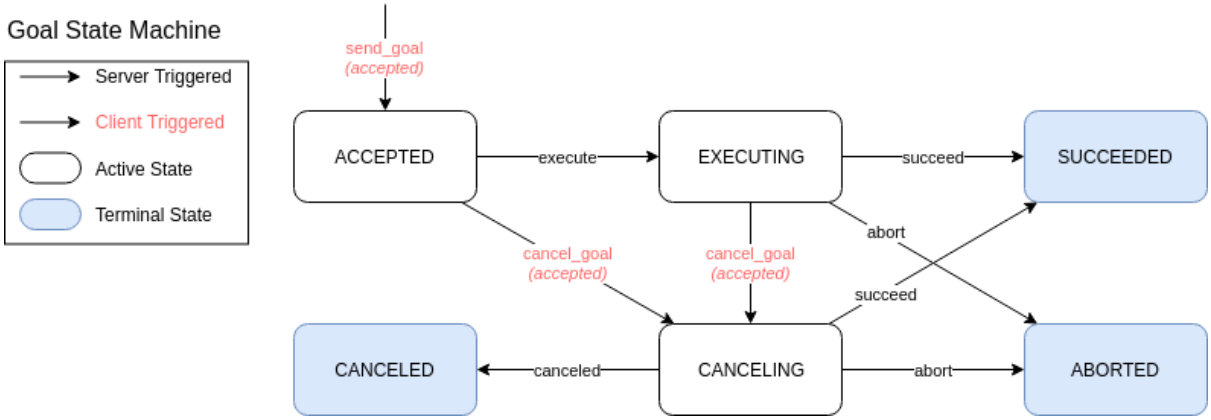
Week4

태그	
내용	

13장 ROS2 액션

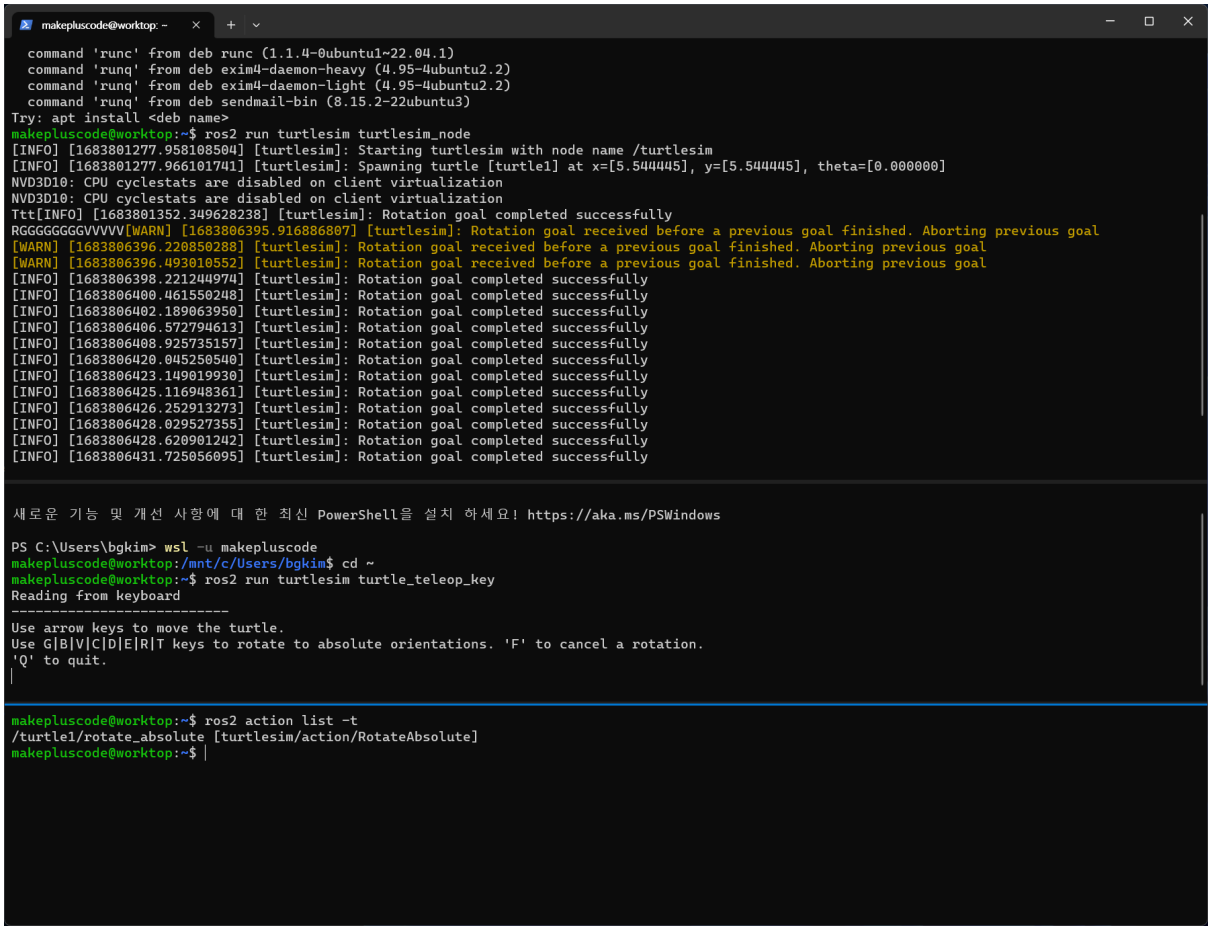
액션은 동기식 양방향 메시지 송수신 방식으로 수행됩니다.

1. 액션을 받아서 특정 태스크를 수행합니다.
2. 중간 피드백을 제공합니다.
3. 액션 결과를 제공합니다.

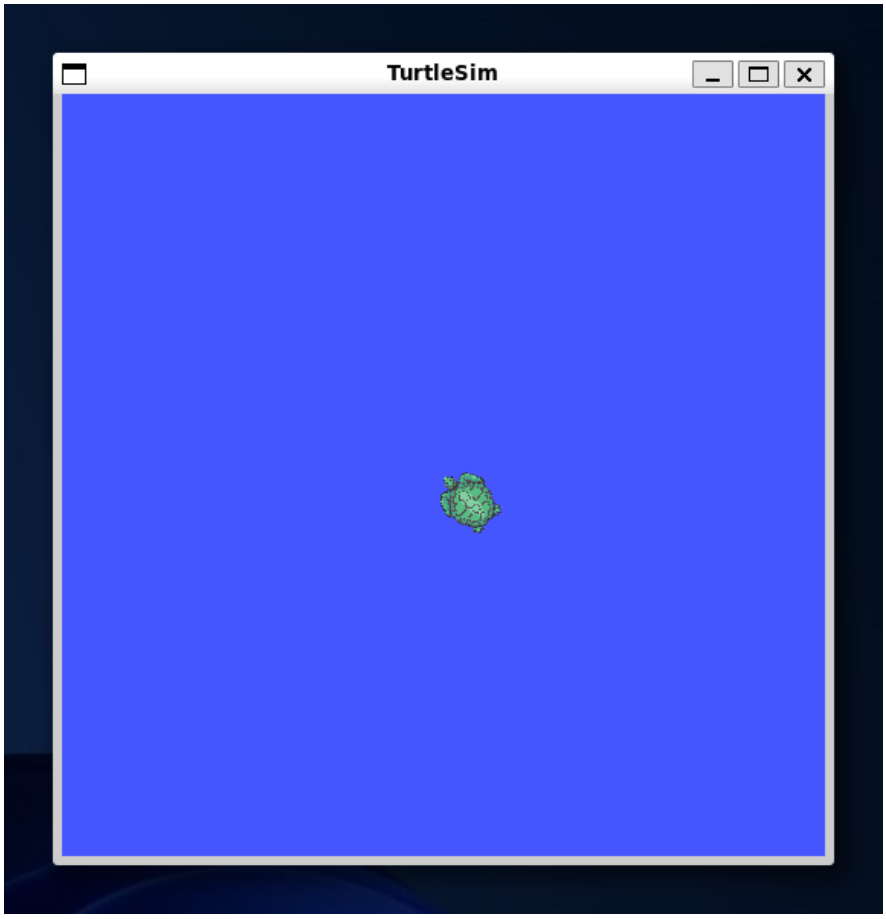


액션 서버와 클라이언트

`turtlesim_node`와 `turtle_teleop_key` 패키지를 실행하여 액션 서버와 클라이언트를 실행해봅시다.


makepluscode@worktop: ~
command 'runc' from deb runc (1.1.4-0ubuntu1~22.04.1)
command 'runq' from deb exim4-daemon-heavy (4.95-4ubuntu2.2)
command 'runq' from deb exim4-daemon-light (4.95-4ubuntu2.2)
command 'runq' from deb sendmail-bin (8.15.2-22ubuntu3)
Try: apt install <deb name>
makepluscode@worktop:~$ ros2 run turtlesim turtlesim_node
[INFO] [1683801277.958108504] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1683801277.966101741] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
NVD3D10: CPU cyclestats are disabled on client virtualization
NVD3D10: CPU cyclestats are disabled on client virtualization
Ttt[INFO] [1683801352.349628238] [turtlesim]: Rotation goal completed successfully
RGGGGGGGGVVVVV[WARN] [1683806395.916886807] [turtlesim]: Rotation goal received before a previous goal finished. Aborting previous goal
[WARN] [1683806396.220850288] [turtlesim]: Rotation goal received before a previous goal finished. Aborting previous goal
[WARN] [1683806396.493010552] [turtlesim]: Rotation goal received before a previous goal finished. Aborting previous goal
[INFO] [1683806398.221244974] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806400.461550248] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806402.189063950] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806406.572794613] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806408.925735157] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806420.045250540] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806423.149019930] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806425.116948361] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806426.252913273] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806428.029527355] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806428.620901242] [turtlesim]: Rotation goal completed successfully
[INFO] [1683806431.725056095] [turtlesim]: Rotation goal completed successfully
새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치하세요! https://aka.ms/PSWindows
PS C:\Users\bgkim> wsl -u makepluscode
makepluscode@worktop:/mnt/c/Users/bgkim$ cd ~
makepluscode@worktop:~$ 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.
makepluscode@worktop:~$ ros2 action list -t
/turtle1/rotate_absolute [turtlesim/action/RotateAbsolute]
makepluscode@worktop:~$



액션 목록

현재 환경에서 실행 중인 액션의 목록은 action list 명령어를 통해 확인할 수 있습니다.

```
~$ ros2 action list -t
/turtle1/rotate_absolute [turtlesim/action/RotateAbsolute]
```

액션 정보

action info를 사용하면 액션 이름과 서버, 클라이언트 노드의 이름과 개수를 확인할 수 있습니다.

```
$ ros2 action info /turtle1/rotate_absolute
Action: /turtle1/rotate_absolute
Action clients: 1
    /teleop_turtle
Action servers: 1
    /turtlesim
```

액션 목표 전달

ros2 action send_goal을 사용하여 목표를 전달할 수 있습니다.

```
$ ros2 action send_goal /turtle1/rotate_absolute turtlesim/action/RotateAbsolute "{theta: -1.5708}" --feedbac
k
Waiting for an action server to become available...
Sending goal:
     theta: -1.5708

Goal accepted with ID: 564b5ffe2fb94806a1511ddd225e1c3f
```

```
Feedback:
    remaining: 2.342829465866089
```

14장 ROS2 인터페이스

ROS 노드 간의 통신을 위해서는 토픽, 서비스, 액션 등의 인터페이스를 사용합니다. 인터페이스란 사용되는 데이터의 형태를 의미합니다.

ROS2 통신 인터페이스

ROS 애플리케이션은 메시지, 서비스 및 액션의 세 가지 형태의 노드 간 통신을 지원합니다. 인터페이스는 IDL (interface definition language) 형식으로 작성할 수 있습니다.

기존 ROS Classic과 같이 msg, srv 등의 포맷을 지원합니다.

ROS2 메시지 명세하는 방법

다음과 같이 ROS 메시지의 필드를 설명하는 간단한 텍스트 파일을 작성합니다.

```
fieldtype1 fieldname1
fieldtype2 fieldname2
fieldtype3 fieldname3
```

예를 들어, GPS 측위 데이터는 다음과 같이 작성할 수 있습니다.

```
float64 latitude
float64 longitude
```

ros2 github의 Path.msg 예제를 보면, 기존에 정의된 Message 형태를 재사용하여 다음과 같이 정의할 수 있습니다.

```
# An array of poses that represents a Path for a robot to follow.

# Indicates the frame_id of the path.
std_msgs/Header header

# Array of poses to follow.
geometry_msgs/PoseStamped[] poses
```

메시지는 경계가 있는 배열의 형태로 정의할 수 있습니다.

```
int32[] unbounded_integer_array
int32[5] five_integers_array
int32[<=5] up_to_five_integers_array

string string_of_unbounded_size
string<=10 up_to_ten_characters_string

string[<=5] up_to_five_unbounded_strings
string<=10[] unbounded_array_of_string_up_to_ten_characters each
string<=10[<=5] up_to_five_strings_up_to_ten_characters_each
```

ROS2 메시지 정의 규칙은 다음과 같습니다.

- 영문 소문자와 숫자를 사용할 수 있으며, 알파벳 문자로 시작해야 합니다.
- 밑줄로 끝나면 안 되며, 밑줄이 연속으로 나오면 안 됩니다.
- 초기값을 지정할 수 있습니다. 예: float64 latitude 37.532600

- 문자열은 ' 또는 " 로 정의해야 합니다.
- 상수는 대문자로 표기하며, 등호 기호 (=)를 사용하여 정의합니다.

ROS2 서비스 인터페이스

ROS 서비스에서 요청과 응답을 분리하여 작성합니다. 예를 들어, 다음은 문자열을 받아서 문자열을 반환하는 서비스의 매우 간단한 예입니다.

```
string str
---
string str
```

ROS2 서비스 정의 규칙은 다음과 같습니다.

- -- 를 사용하여 요청과 응답을 분리한다.
- 서비스 내부에 다른 서비스를 포함할 수 없다.

ROS2 액션 인터페이스

아래 액션의 데이터 타입은 "msg"와 "srv"의 변형으로, 액션 인터페이스라고 합니다. 다음 명령어를 사용하여 인터페이스를 확인할 수 있습니다.

```
$ ros2 interface show turtlesim/action/RotateAbsolute
# The desired heading in radians
float32 theta
---
# The angular displacement in radians to the starting position
float32 delta
---
# The remaining rotation in radians
float32 remaining
```

15장 ROS2 토픽, 서비스, 액션 정리 및 비교

토픽, 서비스, 액션은 ROS의 중요한 컨셉입니다. 이 세 가지 요소는 ROS 프로그래밍에서 매우 중요합니다. 여기서는 이들 간의 비교를 통해 토픽, 서비스, 액션의 서로 다른 특징을 살펴봅니다. 연속성, 방향성, 동기성, 다자간 연결, 노드 역할, 동작 트리거, 인터페이스를 각각 토픽, 서비스, 액션의 특징으로 볼 수 있습니다.

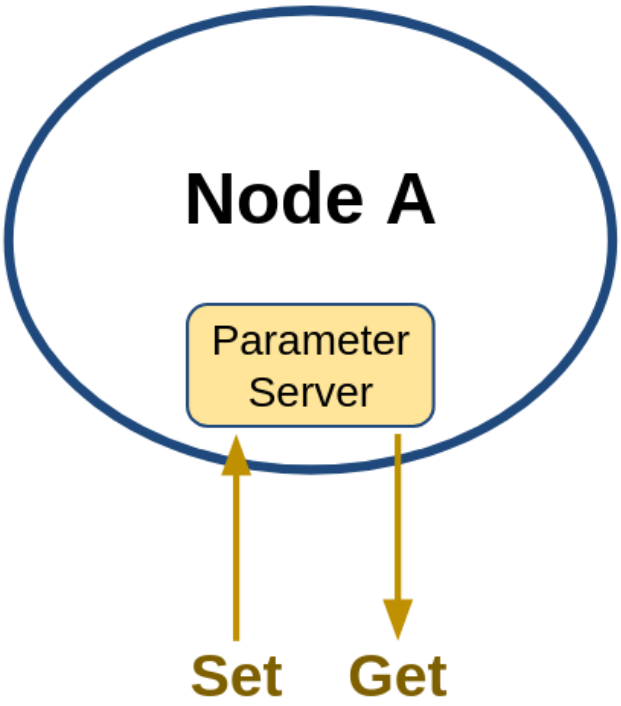
	토픽 (topic)	**서비스 (service)**	**액션 (action)**
연속성	연속성	일회성	복합 (토픽+서비스)
방향성	단방향	양방향	양방향
동기성	비동기	동기	동기 + 비동기
다자간 연결	1:1, 1:N, N:1, N:N (publisher:subscriber)	1:1 (server:client)	1:1 (server:client)
노드 역할	발행자 (publisher) 구독자 (subscriber)	서버 (server) 클라언트 (client)	서버 (server) 클라언트 (client)
동작 트리거	발행자	클라언트	클라언트
인터페이스	msg 인터페이스	srv 인터페이스	action 인터페이스
CLI 명령어	ros2 topic ros2 interface	ros2 service ros2 interface	ros2 action ros2 interface
사용 예	센서 데이터, 로봇 상태, 로봇 좌표, 로봇 속도 명령 등	LED 제어, 모터 토크 On/Off, IK/FK 계산, 이동 경로 계산 등	목적지로 이동, 물건 파지, 복합 태스크 등

아래의 표 2는 토픽, 서비스, 액션에서 사용되는 msg, srv, action 인터페이스를 비교한 내용입니다. 모든 인터페이스는 msg 인터페이스의 확장형이라고 생각하면 됩니다. - 구분자는 0, 1, 2개로 데이터를 구분하는 방법에 차이가 있습니다.

	msg 인터페이스	srv 인터페이스	action 인터페이스
확장자	*.msg	*.srv	*.action
데이터	토픽 데이터 (data)	서비스 요청 (request) --- 서비스 응답 (response)	액션 목표 (goal) --- 액션 결과 (result) --- 액션 피드백 (feedback)
형식	fieldtype1 fieldname1 fieldtype2 fieldname2 fieldtype3 fieldname3	fieldtype1 fieldname1 fieldtype2 fieldname2 --- fieldtype3 fieldname3 fieldtype4 fieldname4	fieldtype1 fieldname1 fieldtype2 fieldname2 --- fieldtype3 fieldname3 fieldtype4 fieldname4 --- fieldtype5 fieldname5 fieldtype6 fieldname6
사용 예	**[geometry_msgs/msg/Twist]**	**[turtlesim/srv/Spawn.srv]**	**[turtlesim/action/RotateAbsolute.action]**
	Vector3 linear Vector3 angular	float32 x float32 y float32 theta string name --- string name	float32 theta --- float32 delta --- float32 remaining

16장 ROS2 파라미터

ROS 2에서 파라미터(parameter)는 각 노드에서 파라미터 관련 Parameter server를 실행하여 그림 1과 같이 작동합니다. 외부 Parameter client와 통신하여 파라미터를 변경할 수 있으며, ROS 2 서비스(service)의 응용으로 생각할 수 있습니다. 서비스가 RPC(remote procedure call)를 위한 서비스 요청과 응답을 목적으로 한다면, 파라미터는 노드 내 매개변수를 서비스 데이터 통신 방식을 사용하여 노드 내부 또는 외부에서 쉽게 지정하거나 변경할 수 있으며, 가져와서 사용할 수 있게 하는 역할을 합니다. 이점에서 파라미터의 사용 목적이 서비스와는 다르다고 볼 수 있습니다.



파라미터 목록 확인 방법

현재 환경의 노드와 파라미터를 확인하려면, ros2 param list 명령을 사용하면 됩니다.

```
$ ros2 param list
/teleop_turtle:
  qos_overrides./parameter_events.publisher.depth
  qos_overrides./parameter_events.publisher.durability
  qos_overrides./parameter_events.publisher.history
  qos_overrides./parameter_events.publisher.reliability
  scale_angular
  scale_linear
  use_sim_time
/turtlesim:
  background_b
  background_g
  background_r
  qos_overrides./parameter_events.publisher.depth
  qos_overrides./parameter_events.publisher.durability
  qos_overrides./parameter_events.publisher.history
  qos_overrides./parameter_events.publisher.reliability
  use_sim_time
```

파라미터 내용 확인

ros2 param describe 명령어를 사용하면 파라미터의 내용을 확인할 수 있습니다.

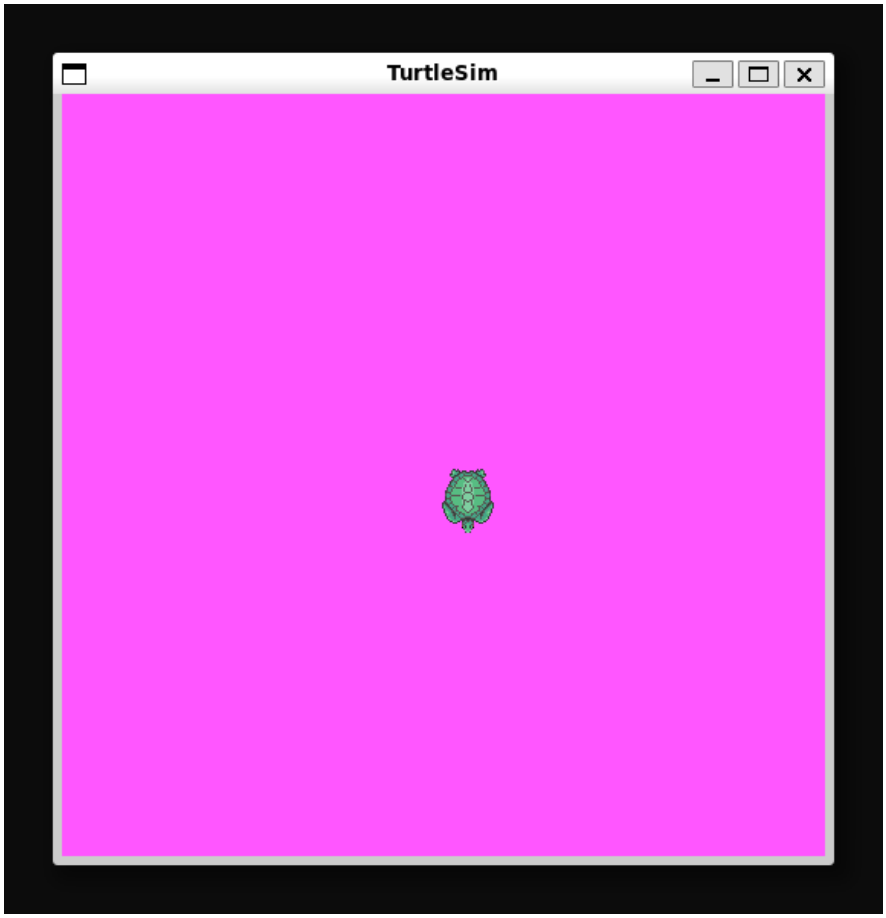
```
$ ros2 param describe /turtlesim background_b
Parameter name: background_b
  Type: integer
  Description: Blue channel of the background color
  Constraints:
    Min value: 0
    Max value: 255
    Step: 1
```

파라미터 내용 읽기

```
$ ros2 param get /turtlesim background_r
Integer value is: 69
```

파라미터 내용 쓰기

```
$ ros2 param set /turtlesim background_r 255
Set parameter successful
```



파라미터 덤프

```
$ ros2 param dump /turtlesim
/turtlesim:
  ros__parameters:
    background_b: 255
    background_g: 86
    background_r: 255
    qos_overrides:
      /parameter_events:
        publisher:
          depth: 1000
          durability: volatile
          history: keep_last
          reliability: reliable
    use_sim_time: false
```