ROS2_1 일차_최범수

19th_예비단원_최범수

1. HW_001
크게 네가지 모드를 구성하였다. 조종모드, 배경색 변경 모드, 터틀의 모양 변경 모드, 펜의 색, 두께를 설정하는 모드이다. 첫번째, 조종을 위해 사용한 헤더파일 및 의존성으로는 기본적으로 rclcpp 와 geometry_msgs 에서의 twist 를 사용하였다. twist 는 선형속도와 각속도를 이용해 거북이의 움직임을 제어할 수 있다. 각각 x 축과 z 축을 사용해 제어했다. Parameter 를 이용해 배경색을 설정하는 과정을 진행했고 이는 turtlesim_node 와의 통신이 가능토록 하였다. 추가적으로 배경색을 바꿀 때는 clear 서비스를 이용해 한번 지금까지의 펜이 그린 경로및 배경을 지운뒤 다시 불러오는 방법으로 설정하였다. 거북이의 모양은 오로카 사이트 강좌에서 거북이를 kill 하고 다시 spawn 하는 서비스를 사용해 변경이 가능했다. 펜의 색, 두께를 설정하기 위해 set_pen 서비스를 호출해 변경하였다. 이번 과제를 진행하면서 변경할 수 있는 거북이를 나열하였고 하나를 선택하면 거북이 모양이 변하도록 구현하였습니다.

control_mode
기본적으로 w,a,s,d 를 통해서 twist.linear, twist.angular 를 이용한 선형속도, 각속도를 조정해 움직였다. 터미널 창에서 입력을 받고 그 값에 따라 설정된 속도를 publish 해 준다. 그 후 초기화하고 다시 publish 해준다.

set_background_color
cin 으로 입력받고 r,g,b 에 따라 변수에 255 값을 대입해준다. 그 후 parameter 클라이언트에 접근해 각 값을 대입해준다. 예를 들어 빨강이면 255,0,0 이 된다. 각 작업을 실행하기 전에 clear 서비스를 통해 전부 지우고 배경색을 변경하도록 하였다.

set_turtle_shape
각 터틀의 이름을 나열해놓았고 선택을 입력받는다. 그 후 변경을 한다면 kill 서비스를 통해 거북이를 제거하고 그 후 parameter 에 접근해 거북이 모양을 우선 설정한다. 그 다음에 spawn 서비스를 통해 각 거북이를 생성을 요청하고 생성과 관련된 응답을 받으면 디버그를 위해 생성되었다는 메시지를 출력한다. 시작 위치는 5.5, 5.5 로 고정한다.

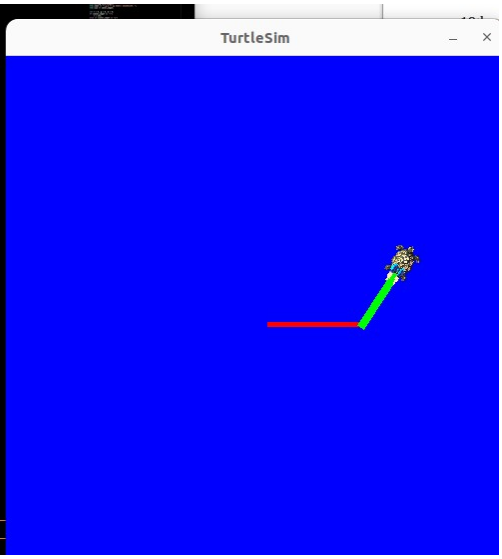
set_pen
r,g,b 값을 입력받고 펜의 두께를 입력받는다. static_cast 를 통해 정수형으로 형변환을 하고 0 으로 설정해 펜을 활성화시킨다. 그 후 turtlesim 의 setpen 서비스를 호출 요청하고 그 후 설정한 값으로 펜의 색상, 두께 등이 변한다.

Main.cpp 는단순히 실행 부분만 작성하였다.

Cmakelists.txt 에서 기본 설정에서 기본 패키지에서 geometry_msgs, turtlesim, std_srvs 등의 패키지를 추가적으로 설정했고 각 패키지는 속도, turtlesim_node 등에서 사용되었다. 뒤의 의존성 설정도 마찬가지로 추가해 빌드에 성공하였다.

```
: 배경색 설정 모드
: 거북이 모양 설정 모드
: Pen 설정 모드
: 종료
택: 4
,g,b 셋 중 하나 입력하세요: g
-10 넓이 값을 입력하세요: 8
 설정 완료

하는 모드를 선택하세요:
: 조종모드
: 배경색 설정 모드
: 거북이 모양 설정 모드
: Pen 설정 모드
: 종료
택: 1
,a,s,d 를 눌러 조종하세요, q 입력시
료
```

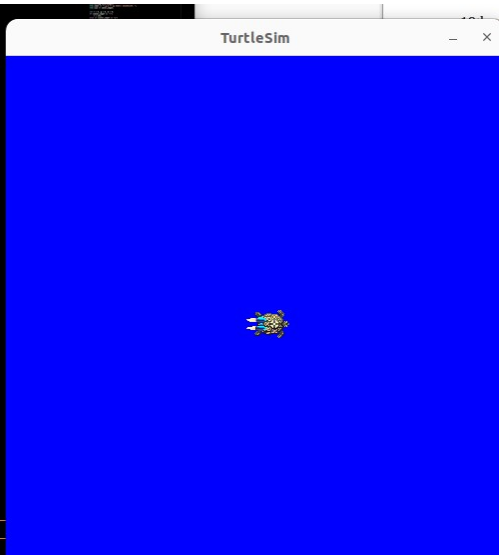
```
choibeomsu@choibeomsu-Legion-5-16IRX9:~
/ros2_ws$ ros2 run turtlesim turtlesim_
node
Warning: Ignoring XDG_SESSION_TYPE=wayl
and on Gnome. Use QT_QPA_PLATFORM=wayla
nd to run on Wayland anyway.
[INFO] [1730569132.833303168] [turtlesi
m]: Starting turtlesim with node name /
turtlesim
[INFO] [1730569132.834622958] [turtlesi
m]: Spawning turtle [turtle1] at x=[5.5
44445], y=[5.544445], theta=[0.000000]
[INFO] [1730569317.454021467] [turtlesi
m]: Clearing turtlesim.
[INFO] [1730569325.085897161] [turtlesi
m]: Spawning turtle [turtle1] at x=[5.5
00000], y=[5.500000], theta=[0.000000]
[INFO] [1730569365.918127279] [turtlesi
m]: Clearing turtlesim.
[INFO] [1730569377.934286197] [turtlesi
m]: Spawning turtle [turtle1] at x=[5.5
00000], y=[5.500000], theta=[0.000000]
```




```
78          twist.linear.x = 0.0;
79          twist.angular.z = 0.0;
80
81          // 잠시 대기 후 초기화된 메시지 발행
82          rclcpp::sleep_for(std::chrono::milliseconds(500));  // 50ms 대기
83          control_publisher ->publish(twist);
```

```
택: 3
하는 거북이 모양의 번호를 선택하세요

: ardent
: bouncy
: crystal
: dashing
: eloquent
: foxy
: galactic
: humble
: rolling

존 거북이가 제거되었습니다.
북이 모양이 설정되었습니다.
로운 거북이가 생성되었습니다.

하는 모드를 선택하세요:
: 조종모드
: 배경색 설정 모드
: 거북이 모양 설정 모드
: Pen 설정 모드
: 종료
택:
```

```
choibeomsu@choibeomsu-Legion-5-16IRX9:~
/ros2_ws$ ros2 run turtlesim turtlesim_
node
Warning: Ignoring XDG_SESSION_TYPE=wayl
and on Gnome. Use QT_QPA_PLATFORM=wayla
nd to run on Wayland anyway.
[INFO] [1730569132.833303168] [turtlesi
m]: Starting turtlesim with node name /
turtlesim
[INFO] [1730569132.834622958] [turtlesi
m]: Spawning turtle [turtle1] at x=[5.5
44445], y=[5.544445], theta=[0.000000]
[INFO] [1730569317.454021467] [turtlesi
m]: Clearing turtlesim.
[INFO] [1730569325.085897161] [turtlesi
m]: Spawning turtle [turtle1] at x=[5.5
00000], y=[5.500000], theta=[0.000000]
[INFO] [1730569365.918127279] [turtlesi
m]: Clearing turtlesim.
[INFO] [1730569377.934286197] [turtlesi
m]: Spawning turtle [turtle1] at x=[5.5
00000], y=[5.500000], theta=[0.000000]
```

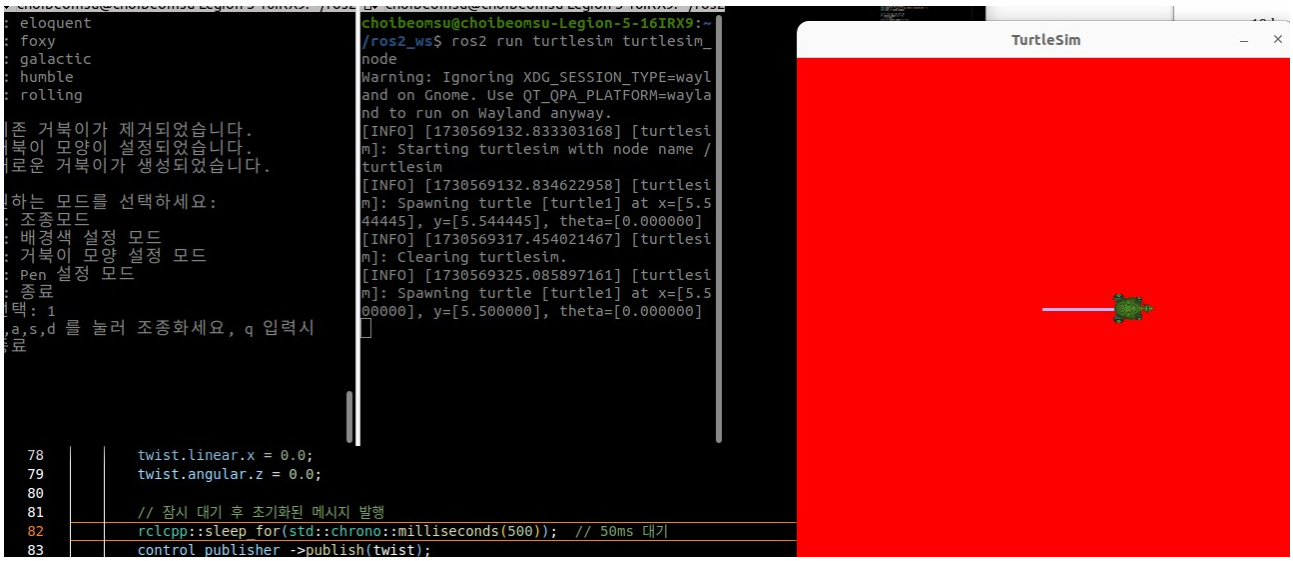



```
78          twist.linear.x = 0.0;
79          twist.angular.z = 0.0;
80
81          // 잠시 대기 후 초기화된 메시지 발행
82          rclcpp::sleep_for(std::chrono::milliseconds(500));  // 50ms 대기
83          control_publisher ->publish(twist);
```

```
료



하는 모드를 선택하세요:
: 조종모드
: 배경색 설정 모드
: 거북이 모양 설정 모드
: Pen 설정 모드
: 종료
택: 2
,g,b 셋 중 하나를 입력하세요: b
경 색 변경

하는 모드를 선택하세요:
: 조종모드
: 배경색 설정 모드
: 거북이 모양 설정 모드
: Pen 설정 모드
: 종료
택:
```

```
choibeomsu@choibeomsu-Legion-5-16IRX9:~
/ros2_ws$ ros2 run turtlesim turtlesim_
node
Warning: Ignoring XDG_SESSION_TYPE=wayl
and on Gnome. Use QT_QPA_PLATFORM=wayla
nd to run on Wayland anyway.
[INFO] [1730569132.833303168] [turtlesi
m]: Starting turtlesim with node name /
turtlesim
[INFO] [1730569132.834622958] [turtlesi
m]: Spawning turtle [turtle1] at x=[5.5
44445], y=[5.544445], theta=[0.000000]
[INFO] [1730569317.454021467] [turtlesi
m]: Clearing turtlesim.
[INFO] [1730569325.085897161] [turtlesi
m]: Spawning turtle [turtle1] at x=[5.5
00000], y=[5.500000], theta=[0.000000]
[INFO] [1730569365.918127279] [turtlesi
m]: Clearing turtlesim.
```


```
78          twist.linear.x = 0.0;
79          twist.angular.z = 0.0;
80
81          // 잠시 대기 후 초기화된 메시지 발행
82          rclcpp::sleep_for(std::chrono::milliseconds(500));  // 50ms 대기
83          control_publisher ->publish(twist);
```


: eloquent
: foxy
: galactic
: humble
: rolling
존 거북이가 제거되었습니다.
북이 모양이 설정되었습니다.
로운 거북이가 생성되었습니다.
하는 모드를 선택하세요:
: 조종모드
: 배경색 설정 모드
: 거북이 모양 설정 모드
: Pen 설정 모드
: 종료
택: 1
,a,s,d 를 눌러 조종하세요, q 입력시
료
choibeomsu@choibeomsu-Legion-5-16IRX9:~
/ros2_ws$ ros2 run turtlesim turtlesim_
node
Warning: Ignoring XDG_SESSION_TYPE=wayl
and on Gnome. Use QT_QPA_PLATFORM=wayla
nd to run on Wayland anyway.
[INFO] [1730569132.833303168] [turtlesi
m]: Starting turtlesim with node name /
turtlesim
[INFO] [1730569132.834622958] [turtlesi
m]: Spawning turtle [turtle1] at x=[5.5
44445], y=[5.544445], theta=[0.000000]
[INFO] [1730569317.454021467] [turtlesi
m]: Clearing turtlesim.
[INFO] [1730569325.085897161] [turtlesi
m]: Spawning turtle [turtle1] at x=[5.5
00000], y=[5.500000], theta=[0.000000]
TurtleSim
78 twist.linear.x = 0.0;
79 twist.angular.z = 0.0;
80
81 // 잠시 대기 후 초기화된 메시지 발행
82 rclcpp::sleep_for(std::chrono::milliseconds(500)); // 50ms 대기
83 control_publisher ->publish(twist);

2. HW_002

1 번과제에서 했던 turtlesim 패키지를 우선 추가하였고 역시 터미널 창의 입력을 통해 옵션을 설정하였다. 여기서는 1 번 과제에서 썼던 twist 기능이 아닌 teleport_absolute 라는 위치 변환의 방법을 통해 도형을 그리는 것을 우선 시도하였다. 하지만 원을 그리는 과정에서 거북이가 twist 와 같은 속도 변화를 통한 이동이 아니다 보니 거북이가 보는 방향이 절대적이였고 원을 그리는 과정에서도 현재 위치에서 원을 그리게 하려면 원의 중심을 실제 위치에서 입력받은 길이의 반지름만큼을 위로 지정해서 그려야 반지름을 먼저 긋지 않고 원을 그릴 수 있었다. 2 일차 과제와의 통일성을 주기 위해서 teleport_absolute 기능은 주석처리를 해놓았다. 따라서 실제로는 twist 기능을 통해 거북이가 이동하면서 도형을 그리고 있다. 먼저 main 함수에서 그릴 도형을 선택하고 각 string 값에 따라 다른 함수가 실행된다. 각 함수 값에는 원은 반지름과 속도를 전송하고 속도를 1 을 선택하였다. 나머지 도형은 각 변 값만을 전송한다. 도형을 다 그리고 나면 더 그릴 것인지를 묻는다.

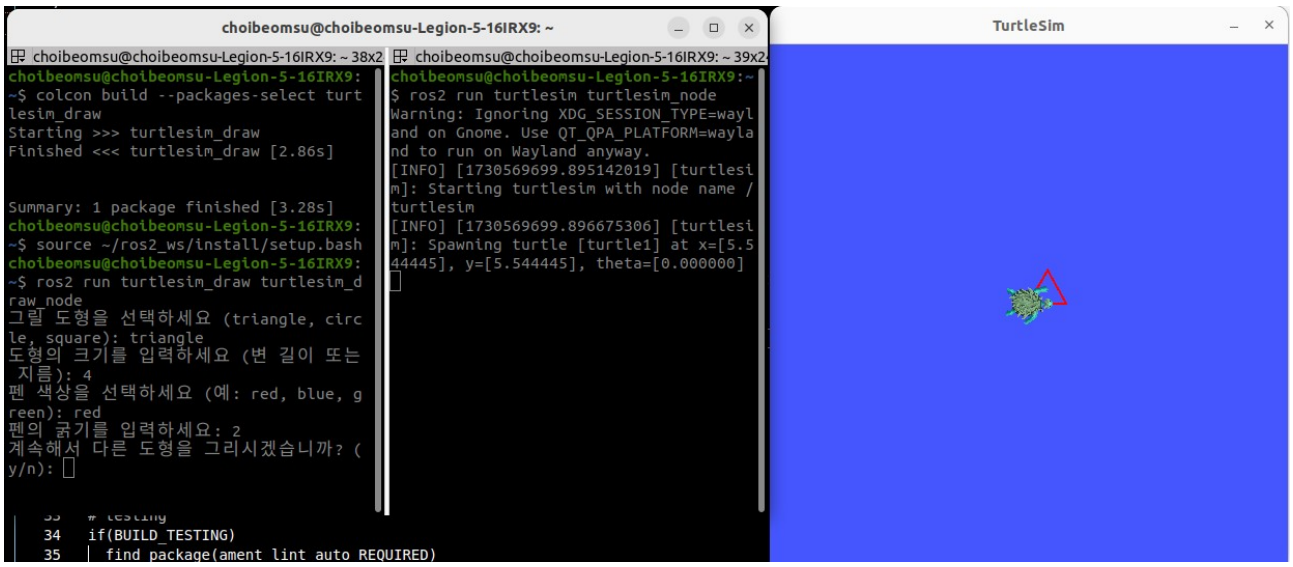
헤더파일 설명이다. 마찬가지로 rclcpp, twist, set_pen 기능이 포함되어있다. 각 클래스에서 쓸 함수 또한 정의되어있다. 소스파일에선 작도를 위한 math, chrono 와 같은 시간 지연 라이브러리, thread 를 포함하였다. 마찬가지로 대기 가능하는 기능을 포함한다. 이 기능은 원을 그리기 위해 필요한 함수들이다. 삼각형, 사각형과는 달리 원은 한 점에서의 같은 거리에 있는 점의 집합이라 반지름을 계산해 그 값과 속도 값와 각도를 조절해 원을 그린다.

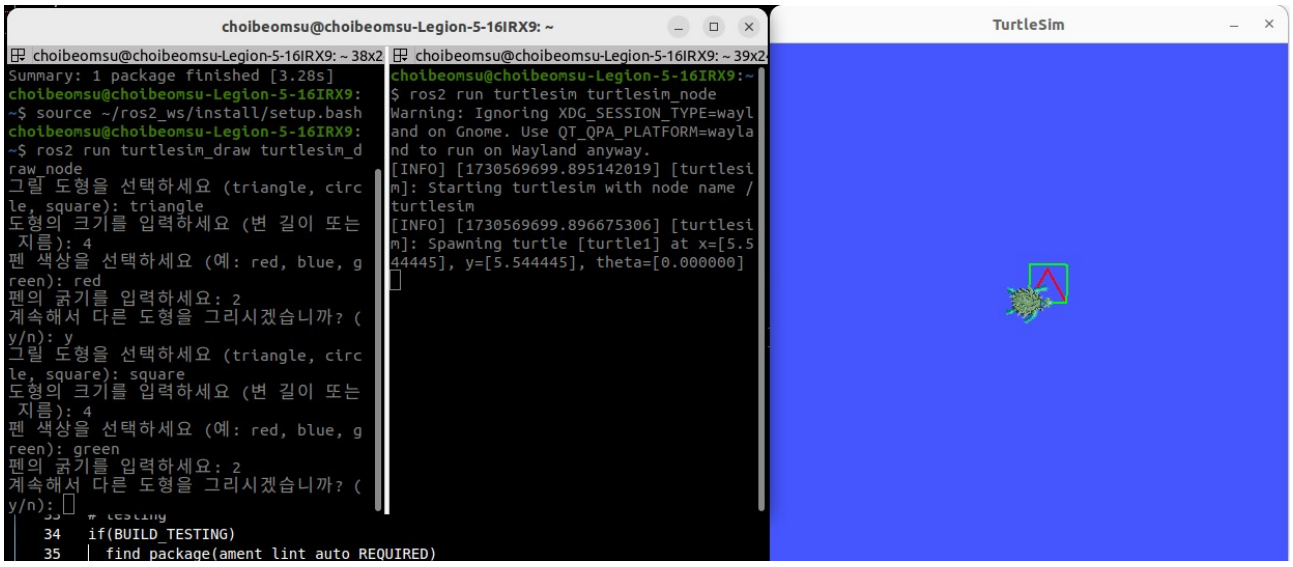
SetPen 함수이다. 색상설정과 두께 설정과 같은 기능과 setpen 서비스를 요청한다. PublishVelocity 는 linear, angular 와 같은 소고도 제어를 twist 에 전송하거나 입력받는 기능이다.

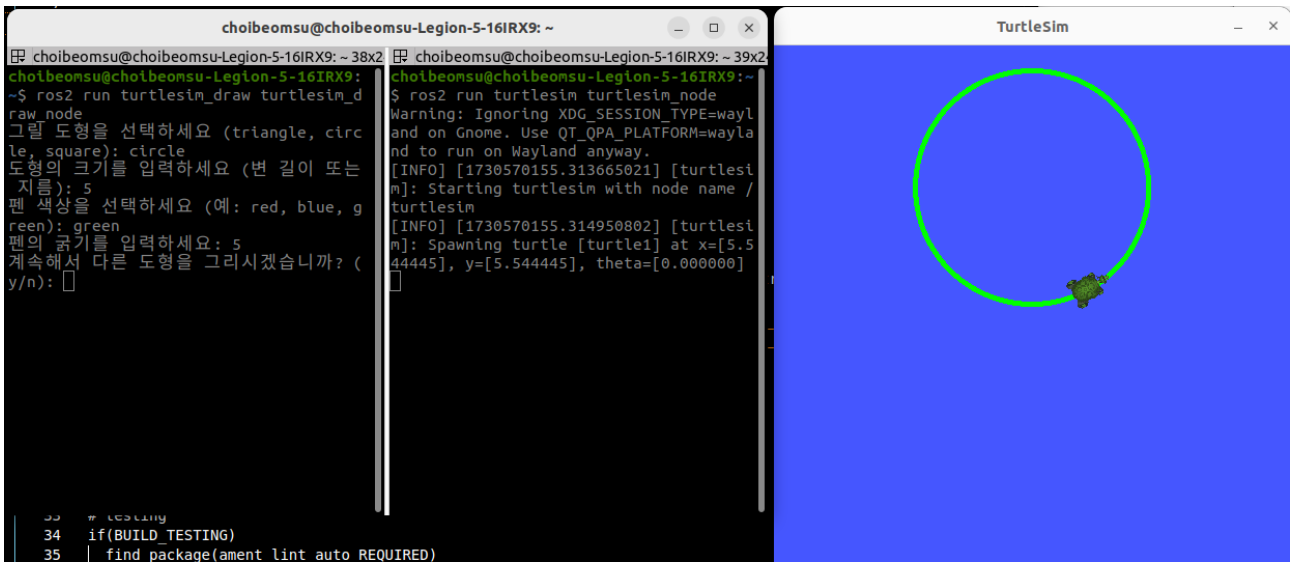
drawCircle 이다. thread 를 통해 drawCircle 함수를 별도의 스레드에서 실행한다. 이 작업을 한 이유는 thread 를 분리하지 않으면 메인 스레드가 다른 작업(계산) 등을 하는 과정에서 분리하지 않았더니 원을 그리다가 멈추는 오류가 발생하였기 때문에 분리하였다. Thread 분리를 통해 여러 작업이 동시에 진행된다. 그 후 둘레를 계산해 저장하고 둘레를 속도로 나눠서 시간을 저장받았다. 이때 시간을 24 로 나눠서 부드러운 원을 그리도록 설정할 수 있다. 24 가 아닌 더 큰 값을 통해 더 부드럽게 작도할 수 있다. 그 후 직접 속도를 publish 하는데 26 번 실행해서 나는 원이 완벽히 채워지도록 조금 더 움직여서 그리게 하였다. 횟수마다 대기 시간이 설정되어있다.

drawTriangle 과 drawSquare 함수이다. 이 함수들은 복잡한 계산이 필요없기에 thread 분리를 하지 않았다. 일정 길이 만큼 움직이고 정삼각형을 위해 120 도 및 90 도 회전을 통해 각을 그렸다. 대기시간을 설정하는 것이 혹시 모를 오류를 대비할 수 있기에 마찬가지로 지연시간을 설정하였다.

Cmakelists.txt 에서 1 번 과제에서의 setpen 과 twist 를 위한 geometry_msgs 와 같은 패키지가 추가되어있고 의존성 역시 관리되어 있다.


choibeomsu@choibeomsu-Legion-5-16IRX9: ~
choibeomsu@choibeomsu-Legion-5-16IRX9: ~ 38x2
choibeomsu@choibeomsu-Legion-5-16IRX9: ~$ colcon build --packages-select turtlesim_draw
Starting >>> turtlesim_draw
Finished <<< turtlesim_draw [2.86s]
Summary: 1 package finished [3.28s]
choibeomsu@choibeomsu-Legion-5-16IRX9: ~$ source ~/ros2_ws/install/setup.bash
choibeomsu@choibeomsu-Legion-5-16IRX9: ~$ ros2 run turtlesim_draw turtlesim_draw_node
그릴 도형을 선택하세요 (triangle, circle, square): triangle
도형의 크기를 입력하세요 (변 길이 또는 지름): 4
펜 색상을 선택하세요 (예: red, blue, green): red
펜의 굵기를 입력하세요: 2
계속해서 다른 도형을 그리시겠습니까? (y/n):
choibeomsu@choibeomsu-Legion-5-16IRX9: ~ 39x2
choibeomsu@choibeomsu-Legion-5-16IRX9:~$ ros2 run turtlesim turtlesim_node
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
[INFO] [1730569699.895142019] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1730569699.896675306] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
34 if(BUILD_TESTING)
35 find_package(ament_lint_auto REQUIRED)
TurtleSim


choibeomsu@choibeomsu-Legion-5-16IRX9: ~
choibeomsu@choibeomsu-Legion-5-16IRX9: ~ 38x2
Summary: 1 package finished [3.28s]
choibeomsu@choibeomsu-Legion-5-16IRX9: ~$ source ~/ros2_ws/install/setup.bash
choibeomsu@choibeomsu-Legion-5-16IRX9: ~$ ros2 run turtlesim_draw turtlesim_draw_node
그릴 도형을 선택하세요 (triangle, circle, square): triangle
도형의 크기를 입력하세요 (변 길이 또는 지름): 4
펜 색상을 선택하세요 (예: red, blue, green): red
펜의 굵기를 입력하세요: 2
계속해서 다른 도형을 그리시겠습니까? (y/n): y
그릴 도형을 선택하세요 (triangle, circle, square): square
도형의 크기를 입력하세요 (변 길이 또는 지름): 4
펜 색상을 선택하세요 (예: red, blue, green): green
펜의 굵기를 입력하세요: 2
계속해서 다른 도형을 그리시겠습니까? (y/n):
choibeomsu@choibeomsu-Legion-5-16IRX9: ~ 39x2
choibeomsu@choibeomsu-Legion-5-16IRX9:~$ ros2 run turtlesim turtlesim_node
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
[INFO] [1730569699.895142019] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1730569699.896675306] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
34 if(BUILD_TESTING)
35 find_package(ament_lint_auto REQUIRED)
TurtleSim


choibeomsu@choibeomsu-Legion-5-16IRX9: ~
choibeomsu@choibeomsu-Legion-5-16IRX9: ~ 38x2
choibeomsu@choibeomsu-Legion-5-16IRX9: ~$ ros2 run turtlesim_draw turtlesim_draw_node
그릴 도형을 선택하세요 (triangle, circle, square): circle
도형의 크기를 입력하세요 (변 길이 또는 지름): 5
펜 색상을 선택하세요 (예: red, blue, green): green
펜의 굵기를 입력하세요: 5
계속해서 다른 도형을 그리시겠습니까? (y/n):
choibeomsu@choibeomsu-Legion-5-16IRX9: ~ 39x2
choibeomsu@choibeomsu-Legion-5-16IRX9:~$ ros2 run turtlesim turtlesim_node
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland anyway.
[INFO] [1730570155.313665021] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1730570155.314950802] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
34 if(BUILD_TESTING)
35 find_package(ament_lint_auto REQUIRED)
TurtleSim

3. HW_003

헤더파일을 먼저 설명하자면 listener 에서는 rclcpp, std_msgs int64, string 들을 사용하기 위해 포함하였다. ListenerNode 는 Callback 함수를 통해서 수신된 메시지를 변수에 저장받는다. count_cb 함수는 정수형 메시지를 수신할 때 사용하였고 count 하는 역할이다. 그 후 출력 함수를 만들어 데이터를 출력시켰다. Cpp 구현에 필요한 멤버 변수 또한 사용하였다. subscription 과 같은 구독자와 데이터 저장 공간들을 포함하고 있다.

Talker 헤더파일에서는 Listener 와 마찬가지로 정수형, string 등을 포함시켰고 메시지를 퍼블리시하는 기능이 있다. 클래스 private 변수 역시 정수형과 문자열형 등의 메시지를 퍼블리시 한다. 횟수 역시 카운트 되고 있다.

소스파일을 설명하겠다. 기본적으로 2 가지 방법이 있다. 스레드를 분리해 talker 와 listener 의 작업을 동시에 작동 가능하게 하는 방법과 분리하지 않고 모든 작업을 순서대로 처리한다. 단순히 메시지를 구독하는 기능은 스레드를 분리하지 않아도 된다는 것을 보고 나는 스레드를 분리하지 않고 모든 작업을 순차적으로 진행되게 하였다.

count_cb 함수를 통해 데이터 수신 여부를 확인하고 그 다음에 메시지를 구독하고 메시지를 구독한 경우 count_cb 를 호출하고 chatter_cli, chatter_count 등에 메시지를 구독한다.Callback 함수에 대해서 더 자세히 설명하자면 이 함수는 메시지가 수신될 때 호출이되고 메시지의 데이터를 변수에 저장해 display 함수를 호출해서 출력한다. count_cb 함수는 메시지를 count 해서 횟수를 출력한다.

Talker.cpp 에 대해서 설명하겠다. Count 변수를 통해 역시 메시지 횟수를 기록한다. publisher 를 통해 메시지를 보낸다. 메시지 publish 함수에서 콘솔에 형식에 맞는 메시지가 출력되고 입력받는다. 문자열은 auto 를 이용해 데이터를 저장한 뒤 메시지를 publish 합니다. 역시 과정에서 카운트 값이 증가하면서 횟수를 증가시킨다. RCLCPP_INF0 와 rqt 를 통해 디버깅이 가능했다.