ROS2_2 일차_최범수

19 기 예비단원 최범수

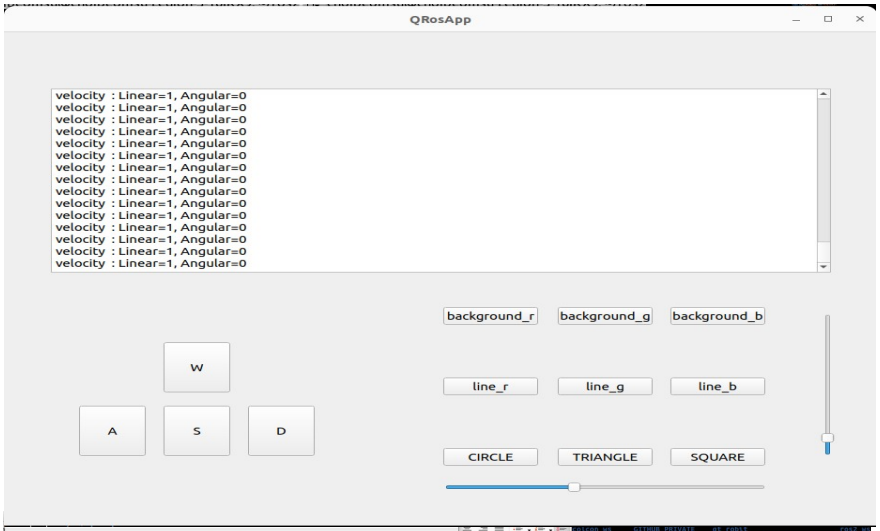
1. HW_001
이 과제는 1 일차의 거북이 관련 조종, 색, 두께 등을 qt 로 UI 를 구성해 버튼을 누르는 방식으로 turtlesim_node 를 조절하는 것이다. 관련 설정을 위해 우선 패키지를 작성하였고,

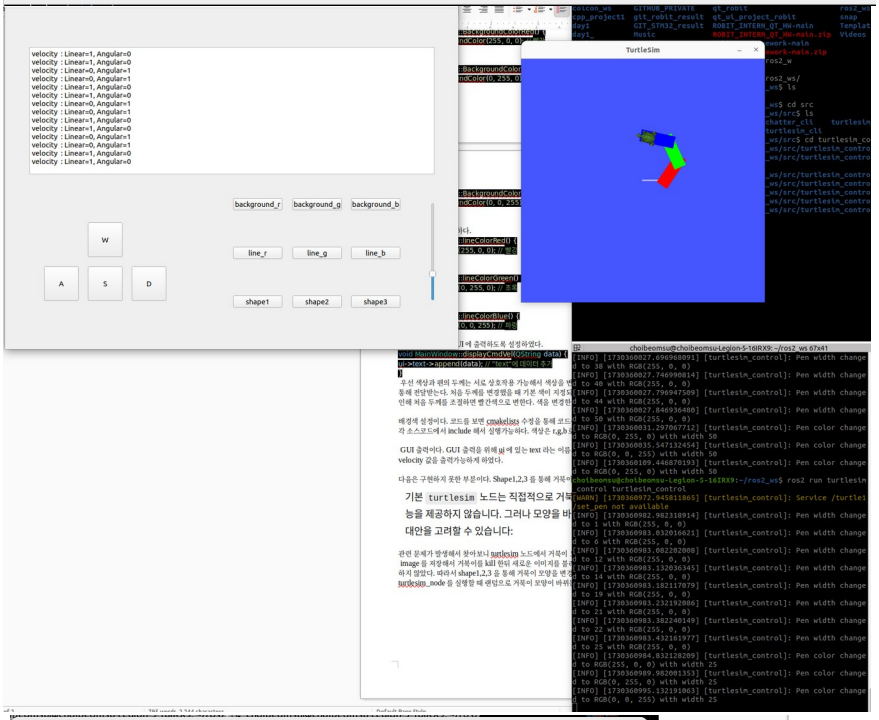
```
#include <rclcpp/rclcpp.hpp>
#include <QObject>
#include <QThread>
#include <geometry_msgs/msg/twist.hpp>
#include <std_srvs/srv/empty.hpp>
#include <rclcpp/parameter_client.hpp> // 파라미터 관련 서비스
#include <turtlesim/srv/set_pen.hpp>
```

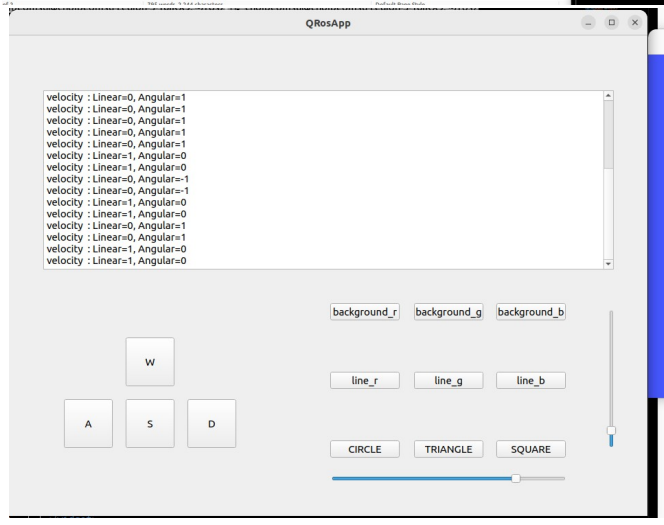
파라미터와 관련된 기능, 펜 색 두께 등을 조절하는 기능, GUI 와 관련된 기능과 이동에 쓰이는 기능 등 여러 함수가 include 되어 있다.

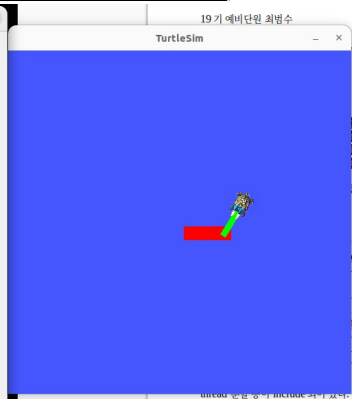
main_window.hpp 부터 설명하겠다. 이 헤더파일에서는 GUI 에서 쓰이는 여러 함수 및 기능 구현을 위한 함수가 있다. Go to slot 을 이용해 각 버튼이 눌렸을 때의 함수들이 선언되어 있다. Qnode.hpp 이다. publishVelocity 와 backgroundColor, linecolor, draw 기능과 같은 함수들이 선언되어 있고 각 함수들이 접근이 가능하도록 private 멤버 변수로서 구성되어있다. Q_SIGNALS 를 통해 cmd_vel 이 출력된다.

소스파일들에 대해서 설명하겠다. 우선 main_window.cpp 를 보면 처음부터 connect 함수를 사용해 ui 에서의 버튼 클릭이 각각 함수와 연결되어 있는 것을 볼 수 있다. 기능은 wasd, background_r,g,b 와 line_r,g,b 그리고 도형을 그리기 위한 shape1, shape2, shape3 등이 connect 로서 연결되어 있다. 출력을 위한 cmd_vel 은 displayCmdVel 함수와 connect 되어 있다. 밑의 함수를 보면 w,a,s,d 로 조종이 가능하고 background line 색 변경이 가능하다. cmd_vel 은 `ui->text->append(data); // "text"에 데이터 추가` 를 통해서 계속 데이터를 append 추가 하는 방식으로 되어있다.
다음은 qnode.cpp 를 설명하겠다. 맨 위에 qnode 에서의 생성자가 있고 그림을 그릴 때를 위한 set_pen, thread 분할 등이 include 되어 있다. 처음 시작할 때 현재의 색상을 0, 0, 0 으로 초기화하고 parameter 에 접근해서 배경색상을 변경하도록 하였다.
속도 제어를 위해서 1 일차 과제와 마찬가지로 twist 함수를 사용했다. wasd 버튼에 따른 linear, angular 값의 변화로 인해 조절이 가능하다. 추가로 Q_EMIT 을 사용해서 cmd_vel 즉 velocity 값을 출력하기 위해 신호를 보내도록 하였다. RCLCPP_WARN 이라는 기능을 통해서 디버깅을 하였고 신호가 보내졌는지 또는 클리어에 성공했는지를 확인하도록 하였다. linecolor 는 파라미터에 접근해 setpen 서비스를 이용하였다. 지금 현재의 값을 읽고 저장된 두께로 조종이나 그림을 그리는 과정을 진행하도록 하였다. 처음 색상은 빨간색으로 초기화하여 만약 두께만 늘리고 그림 그리는 작업을 시작한다면 빨간색으로 먼저 그려지도록 하였다. 이 부분 역시도 RCLCPP_WARN 이라는 문구를 이용해 터미널에 오류가 뜨는지 안 뜨는지 확인할 수 있었다. 원, 삼각형, 사각형을 그리는 과정 역시 1 일차와 비슷했다. 원을 끝까지 그리는 작업을 병렬적으로 진행하기 위해서 thread 를 분리하였고 둘레와 이동 시간을 계산해 작업을 실행하였다. 삼각형과 사각형 역시 입력받은 값을 기준으로 작도하였고 M_PI 라는 변수를 지정한뒤 180 도의 값에 특정한 수를 곱해서 각도만큼 회전하도록 하였다.
Main.cpp 에서는 처음 패키지를 빌드했을 때와 달라진 것이 없지만 GUI 를 출력하는 코드가 작성되어 있다.
UI 는 w,a,s,d 버튼과 background r,g,b 그리고 line r,g,b circle, triangle,square 로 구성되어있다.

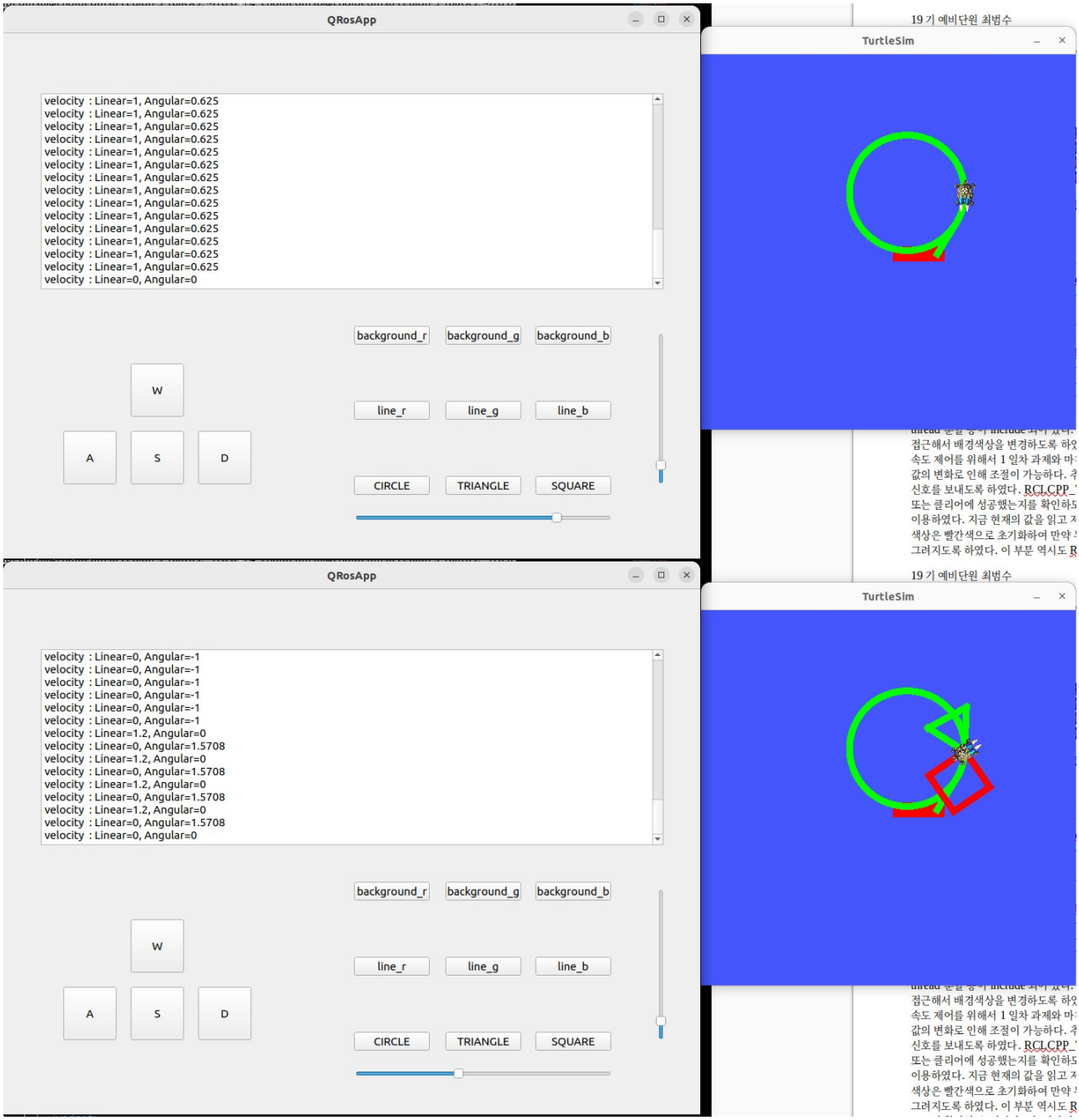

QRosApp
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
background_r
background_g
background_b
W
line_r
line_g
line_b
A
S
D
CIRCLE
TRIANGLE
SQUARE


TurtleSim
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=0, Angular=1
velocity : Linear=0, Angular=1
velocity : Linear=1, Angular=0
velocity : Linear=0, Angular=1
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=0, Angular=1
velocity : Linear=0, Angular=1
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
background_r
background_g
background_b
W
line_r
line_g
line_b
A
S
D
shape1
shape2
shape3


QRosApp
velocity : Linear=0, Angular=1
velocity : Linear=0, Angular=1
velocity : Linear=0, Angular=1
velocity : Linear=0, Angular=1
velocity : Linear=0, Angular=1
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=0, Angular=-1
velocity : Linear=0, Angular=-1
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
velocity : Linear=0, Angular=1
velocity : Linear=0, Angular=1
velocity : Linear=1, Angular=0
velocity : Linear=1, Angular=0
background_r
background_g
background_b
W
line_r
line_g
line_b
A
S
D
CIRCLE
TRIANGLE
SQUARE


TurtleSim

19 기 예비단원 최범수



unread 분들 중에 include 최적 있다.
접근해서 배경색상을 변경하도록 하
속도 제어를 위해서 1 일차 과제와 마
값의 변화로 인해 조절이 가능하다. 추
신호를 보내도록 하였다. RCLCPP_
또는 클리어에 성공했는지를 확인하도
이용하였다. 지금 현재의 값을 읽고 저
색상은 빨간색으로 초기화하여 만약
그려지도록 하였다. 이 부분 역시도 R

19 기 예비단원 최범수

unread 분들 중에 include 최적 있다.
접근해서 배경색상을 변경하도록 하
속도 제어를 위해서 1 일차 과제와 마
값의 변화로 인해 조절이 가능하다. 추
신호를 보내도록 하였다. RCLCPP_
또는 클리어에 성공했는지를 확인하도
이용하였다. 지금 현재의 값을 읽고 저
색상은 빨간색으로 초기화하여 만약
그려지도록 하였다. 이 부분 역시도 R

2. HW_002

이 과제는 1 일차의 chatter_cli 과제를 GUI 를 구성해서 GUI 상에 수신받은 메시지를 띄우는 과제이다. 헤더파일부터 설명하겠다. Listener.hpp 이다. 1 일차와 마찬가지로 rclcpp, std_msgs, 그리고 GUI 를 위한 부분들이 include 되어 있다.listener class 를 만들어 signal 에서는 메시지를 전달하고 callback 함수를 구현해 메시지를 수신 받고 subscribe 하도록 만들었다. Talker.hpp 에서는 talker class 를 만들어 퍼블리시 하는 함수와 퍼블리셔를 구현하였다. main_window.hpp 에서는 GUI 에서 사용된 여러 버튼, label 등을 include 했고 talker 노드 객체와 listener 노드 객체를 구성해서 메시지를 퍼블리시하거나 subscribe 하는 부분, 생성자 소멸자부분을 구현해 놓았다. 역시 go to slot 을 통해 버튼이 눌렀을 때에 대한 기능을 추가하였고 메시지 입력 부분과 출력 라벨 등이 private 멤버 변수에 들어가 있다. 추가로 구현한 기능은 closeEvent 로 쓰인 이벤트 핸들러인데 메인 윈도우가 닫힐 때 호출되고 창을 닫기 전에 talker 와 listener 노드를 먼저 종료할 수 있는 기능이다.

소스파일에 들어가보면 thread 를 나누어서 작업을 완료하였다. thread 를 나누어서 병렬적으로 작업을 실행시킬 수 있고, talker 와 listener 를 main.cpp 에서 thread 를 나누어서 별도로 작업하게 하였다. Listener 객체에서 subscribe 를 구현해 chatter 라는 이름의 토픽을 구현하고 있고 std::bind 를 통해서 수신할 때마다 callback 함수를 호출한다. 호출된 callback 함수는 포인터를 통해 메시지를 전달받고 emit 을 통해서 메시지를 수신받았다는 신호를 보낸다. RCLCPP_INF0 를 통해서 메시지가 수신은 받았는데 Qlabel 에 출력이 안되는건지 수신을 못받았는지를 확인해 디버깅하였다.
talker.cpp 에서는 퍼블리셔를 구현해서 메시지를 입력받고 보내도록 하였다. main_window.cpp 에서는 버튼과 퍼블리시, 서브스크라이브를 connect 하여서 버튼을 눌렀을 때 종속함수로 신호가 가도록 하였다. 또한 Qlabel 에 메시지 데이터들을 띄우는 기능과 closeEvent 를 통해 안전하게 작업을 종료할 수 있다. Main.cpp 이다. 여기서 thread 를 구분해 주었는데 singleThreadedExecutor 는 이벤트 처리 루프를 관리하는 역할인데 내가 쓴 역할에서는 talker_node 와 listener_node 를 별도로 스레드를 구분할 때 사용하였다. ros_executor 는 이제 실행할 때 talker_node 와 listener_node 를 추가해서 별도로 활성화하도록 하였다. Spin() 함수를 통해 별도의 스레드에서 실행되었고 이를 통해 작동되었다. 마지막에 ros_thread.join 을 통해서 종료시켰다.

```
auto ros_executor = std::make_shared<rclcpp::executors::SingleThreadedExecutor>();
ros_executor->add_node(mainWindow.talker_node);  // mainWindow 내의 talker_node를 ROS 스핀에 추
ros_executor->add_node(mainWindow.listener_node); // mainWindow 내의 listener_node를 ROS 스핀
std::thread ros_thread([&]() { ros_executor->spin(); });

int result = app.exec();

rclcpp::shutdown();
ros_thread.join();
```


QRosApp
hi
hi
Publish


QRosApp
hi
hi
Publish