2일차 과제 보고서

- 로빛 19기 예비단원 김근형

2일차 과제

qnode.cpp와 qnode.hpp를 생성한 이유

- ROS 노드를 Qt에 통합하기 위함이다.
- `QNode` 클래스는 Qt의 이벤트 루프와 ROS의 `spin` 함수를 관리하여 두 시스템이 원활하게 작동하도록한다.

Qt signal-Slot

- `QNode` 클래스는 ROS 콜백에서 발생하는 이벤트를 Qt의 `signal`로 전달하고, 이를 GUI에서 `slot`으로 처리할 수 있도록 한다.

1. **QT GUI ros 패키지로 turtlesim 조종**

- 기능
  - 1일차 1번, 2번 기능 전부 포함
  - QPushButton 사용
  - cmd_vel 값 gui출력

⇒ turtle 모양 바꾸는 것은 구현하지 못하였다.

- qnode

```
class QNode : public QThread {
    Q_OBJECT // Qt의 시그널과 슬롯을 사용하기 위해 필요
public:
    QNode(); // 생성자
    ~QNode(); // 소멸자

protected:
    void run(); // 스레드에서 실행할 메서드
};
```

`QThread`를 상속받음으로써, ROS의 이벤트 루프와 메시지 처리 작업을 효과적으로 분리한다.

```
Q_SIGNALS: // Qt 시그널 정의
    void rosShutDown(); // ROS 종료 시그널
    void cmdVelUpdated(double linear_x, double angular_z); // 속도 업데이트 시그
```

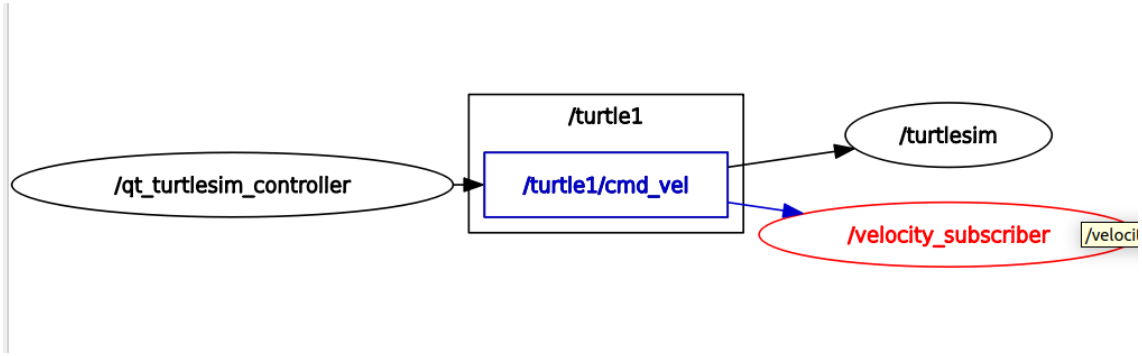
`QNode` 클래스는 Qt의 시그널/슬롯 메커니즘을 활용하여 이벤트 기반으로 한다.

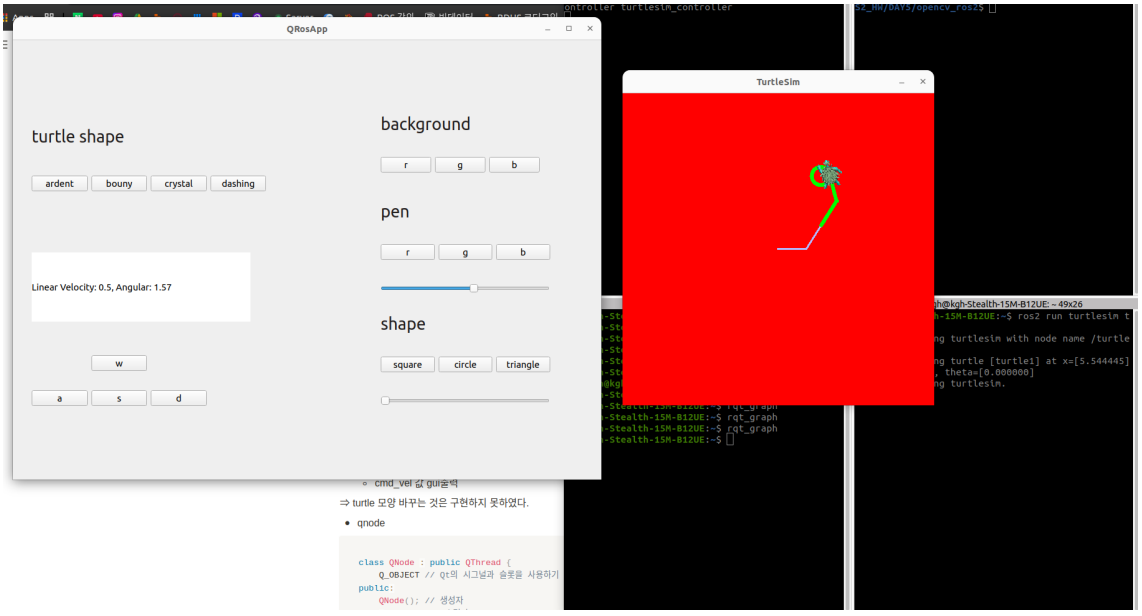
`cmdVelUpdated` 는 `geometry_msgs::msg::Twist` 메시지를 통해 거북이의 속도가 업데이트될 때 호출된다.

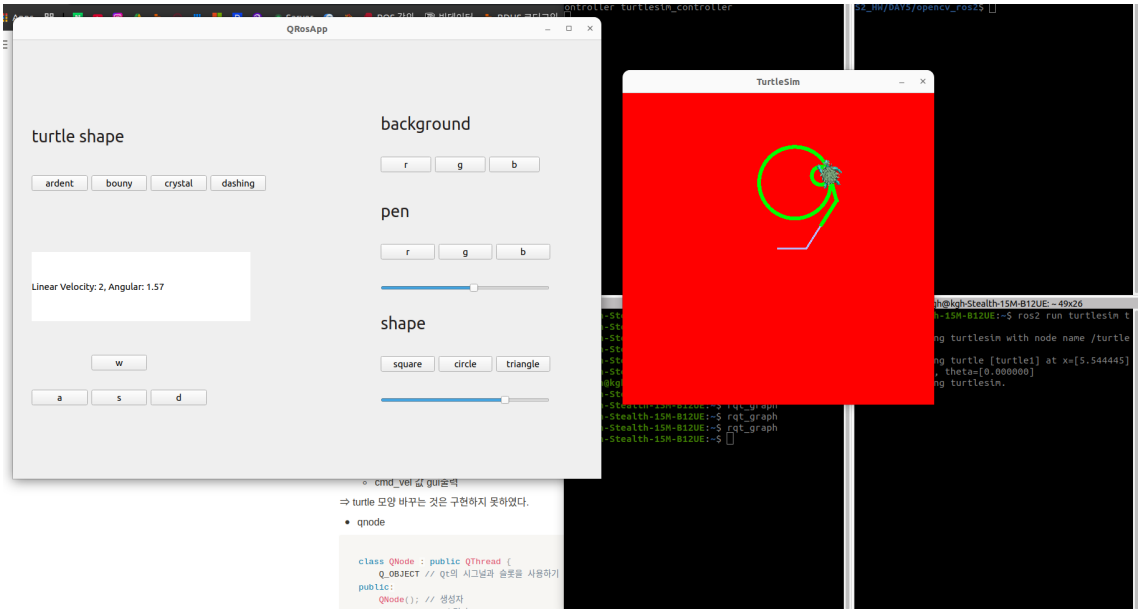
```
// cmd_vel 구독 관련 설정
cmd_vel_subscription = node->create_subscription<geometry_msgs::msg::Twist>(
    "/turtle1/cmd_vel",      // 토픽 이름
    10,                      // 큐 사이즈
    [this](geometry_msgs::msg::Twist::SharedPtr msg) { // 콜백 함수
        emit cmdVelUpdated(msg->linear.x, msg->angular.z);
    }
);
```

`emit`은 C++의 Qt 프레임워크에서 시그널을 발생시키는 데 사용

`emit cmdVelUpdated(linear, angular);`는 `cmdVelUpdated` 시그널을 발생시키며, 이 시그널은 `linear`와 `angular` 값을 전달







2. **QT로 string입력 받아 버튼 눌러 talker/listener 노드 제작**

- 기능
  - 버튼 눌러서 퍼블리시
  - 서브스크라이브 하면 라벨값 변경

```
auto ros_executor = std::make_shared<rclcpp::executors::SingleThreadedExecuto
ros_executor->add_node(mainWindow.talker_node);  // mainWindow 내의 talker_nod
ros_executor->add_node(mainWindow.listener_node); // mainWindow 내의 listener_
std::thread ros_thread([&]() { ros_executor->spin(); });
```

`std::make_shared<rclcpp::executors::SingleThreadedExecutor>()`

- `SingleThreadedExecutor`는 ROS 2의 executor로, 여러 노드를 단일 스레드에서 실행하도록 설정
- `make_shared`는 객체의 메모리를 관리하는 스마트 포인터를 생성합니다. 이 방법은 성능과 안전성을 개선하는 데 유리하다.

`add_node(mainWindow.talker_node)`

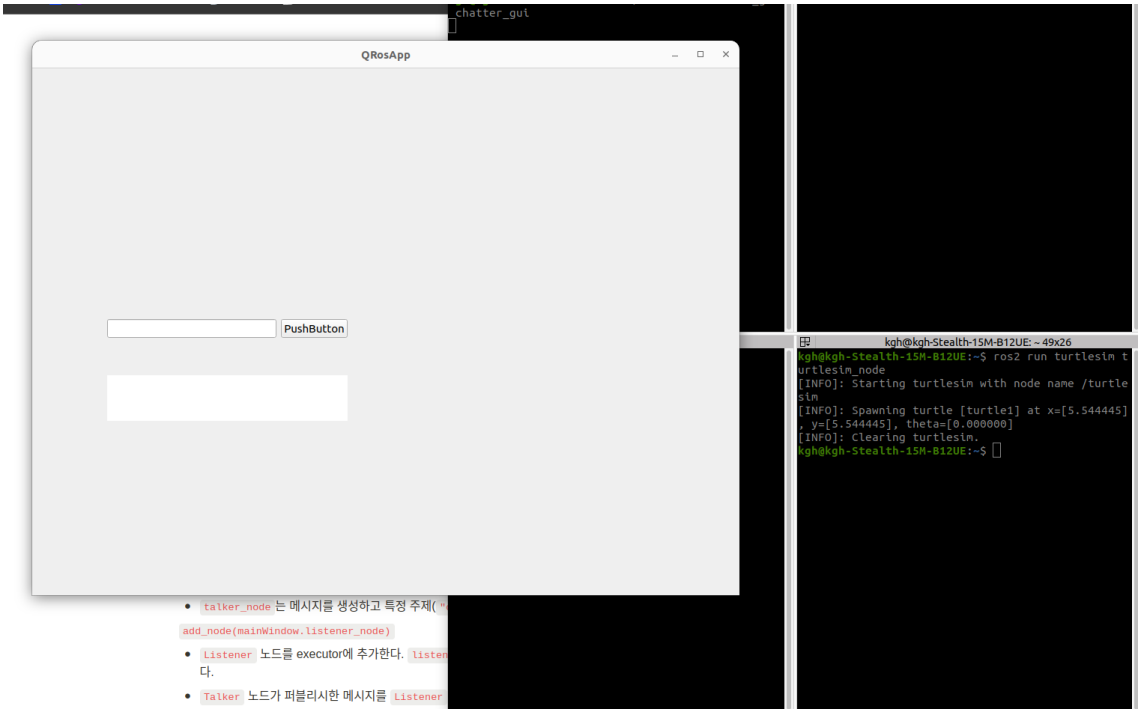
- `mainWindow.talker_node`를 ROS executor에 추가한다. 이렇게 함으로써 `talker` 노드가 ROS의 메시지 발행 시스템에 등록되고, 발행된 메시지를 수신하는 것이 가능해진다.
- `talker_node`는 메시지를 생성하고 특정 주제(`"chatter"`)에 퍼블리시하는 기능을 담당

`add_node(mainWindow.listener_node)`

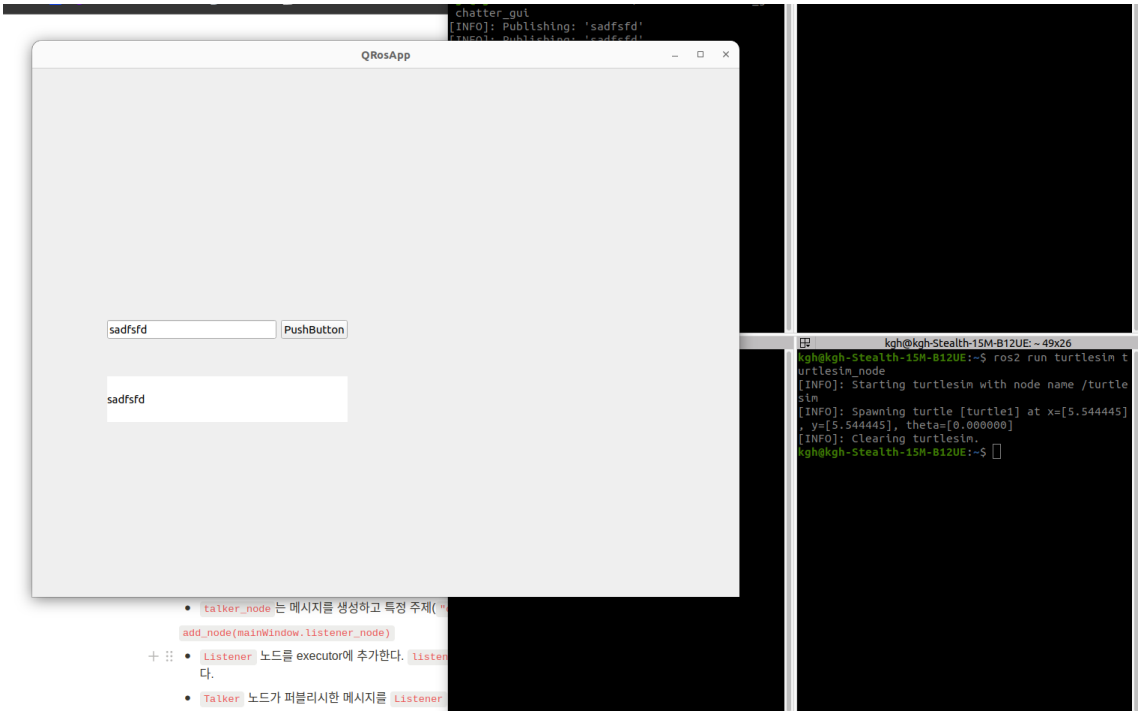
- `Listener` 노드를 executor에 추가한다. `listener_node`는 `"chatter"` 주제에서 메시지를 수신한다.
- `Talker` 노드가 퍼블리시한 메시지를 `Listener` 노드가 수신할 수 있다.

스레드 생성

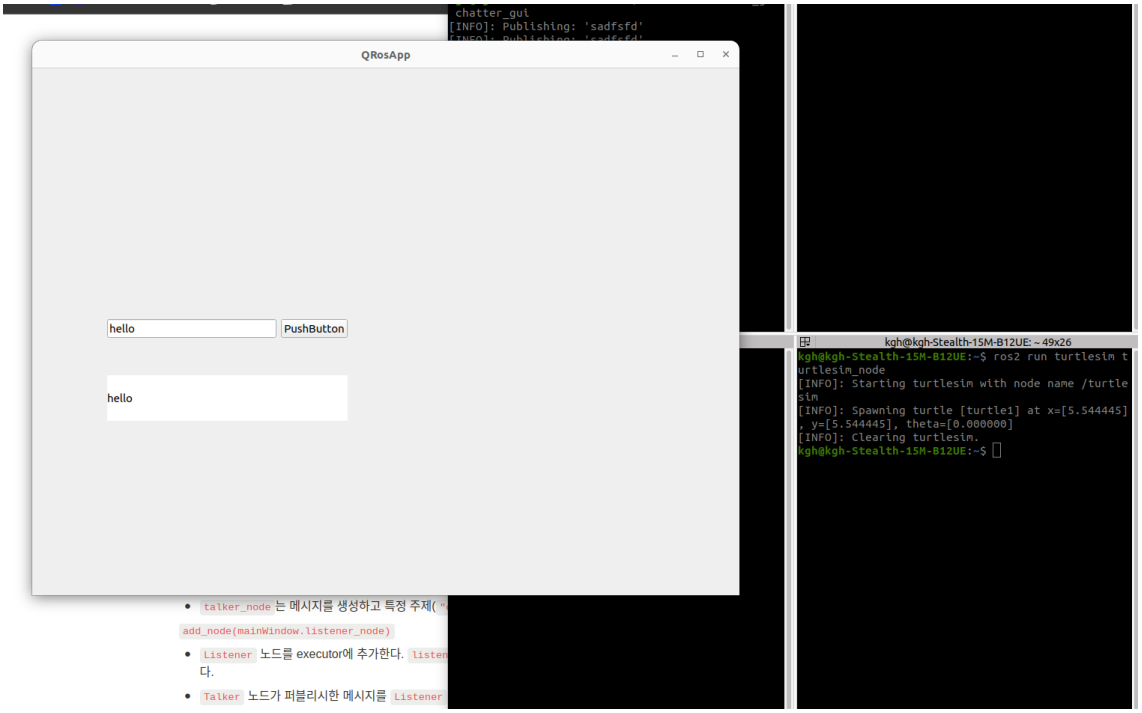
- `std::thread`를 사용하여 새로운 스레드를 생성한다. 이 스레드는 `ros_executor->spin()` 함수를 실행한다.



- `talker_node`는 메시지를 생성하고 특정 주제("
`add_node(mainWindow.listener_node)`
- `Listener` 노드를 executor에 추가한다. `listen`
다.
- `Talker` 노드가 퍼블리시한 메시지를 `Listener`



- `talker_node`는 메시지를 생성하고 특정 주제("
`add_node(mainWindow.listener_node)`
- `Listener` 노드를 executor에 추가한다. `listen`
다.
- `Talker` 노드가 퍼블리시한 메시지를 `Listener`



- `talker_node`는 메시지를 생성하고 특정 주제("
`add_node(mainWindow.listener_node)`
- `Listener` 노드를 executor에 추가한다. `listen`
다.
- `Talker` 노드가 퍼블리시한 메시지를 `Listener`