ROS2 day2 hw2 결과보고서

2025407012/로봇학부/송연우

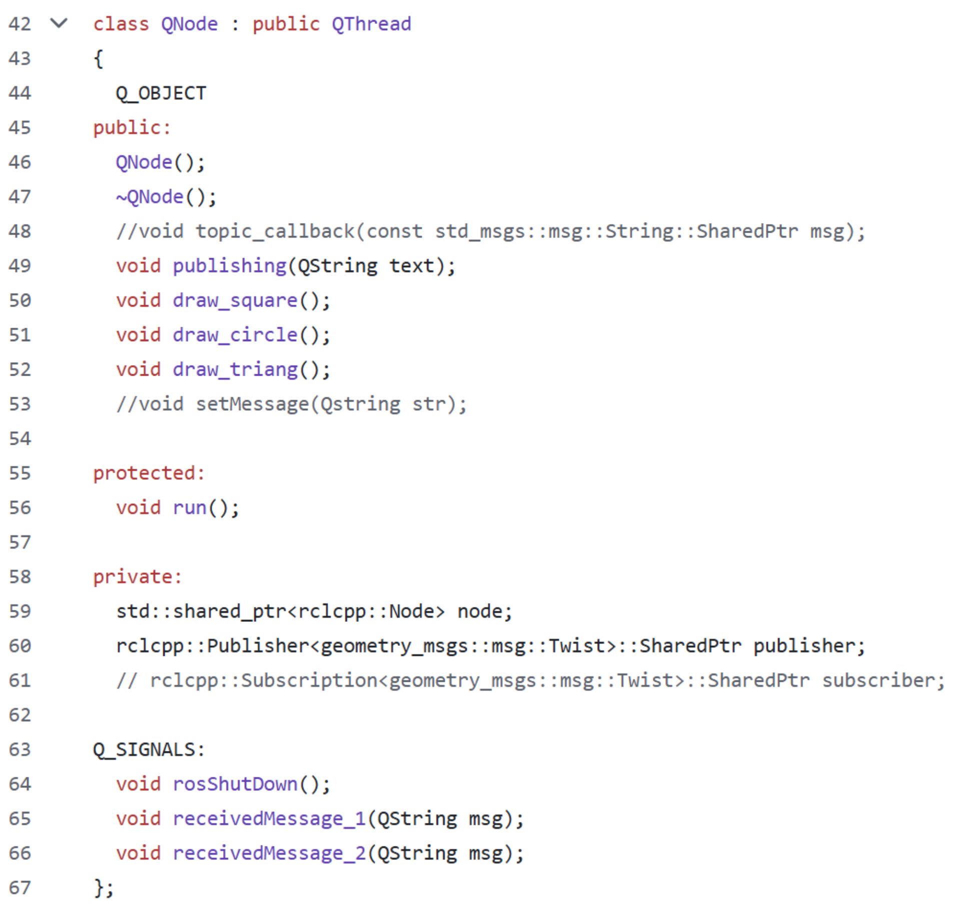
목차

1. Hpp

(1) Mainwindow: 헤더파일의 클래스 선언부입니다. Trutlesim의 펜 색, 굵기를 변경하기 위해 turtlesim/srv/set_pem.hpp를, 이동을 위해 Twist형을 사용했습니다.

```cpp
#include "geometry_msgs/msg/twist.hpp"
#include "turtlesim/srv/set_pen.hpp"

/*****************************************************************************
** Interface [MainWindow]
*****************************************************************************/
/**
 * @brief Qt central, all operations relating to the view part here.
 */
class MainWindow : public QMainWindow
{
  Q_OBJECT

public:
  Ui::MainWindowDesign* ui;
  MainWindow(QWidget* parent = nullptr);
  ~MainWindow();
  QNode* qnode;
  void label_2(QString *msg);

private slots:
  void on_pushButton_clicked();

  void on_pushButton_2_clicked();

private:
  void closeEvent(QCloseEvent* event);
};
```

(2) Qnode

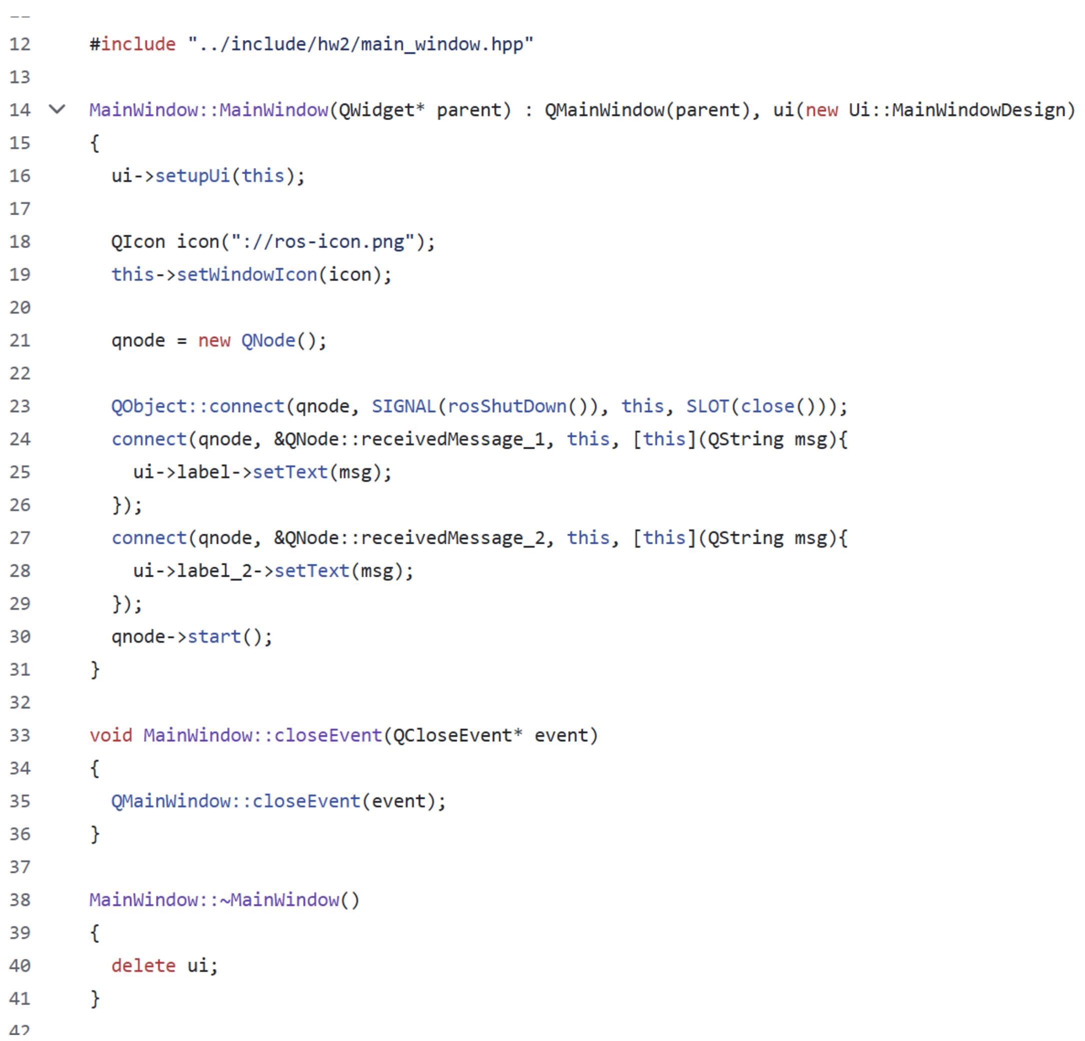


```cpp
class QNode : public QThread
{
  Q_OBJECT
public:
  QNode();
  ~QNode();
  //void topic_callback(const std_msgs::msg::String::SharedPtr msg);
  void publishing(QString text);
  void draw_square();
  void draw_circle();
  void draw_triang();
  //void setMessage(Qstring str);

protected:
  void run();

private:
  std::shared_ptr<rclcpp::Node> node;
  rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr publisher;
  // rclcpp::Subscription<geometry_msgs::msg::Twist>::SharedPtr subscriber;

Q_SIGNALS:
  void rosShutDown();
  void receivedMessage_1(QString msg);
  void receivedMessage_2(QString msg);
};
```

QNode클래스 선언 부분입니다. 거북이를 움직여 그림을 그리는 함수는 public에 선언하였고 Q_SIGNALS에 메시지를 수신했을 때 GUI에 변화를 주기 위한 함수를 2개 추가했습니다.

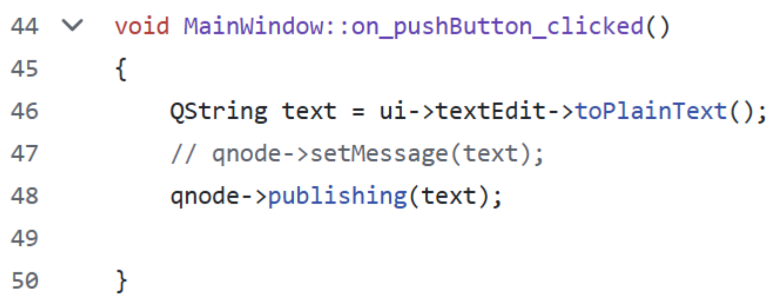
2. cpp파일

(1) mainwindow



```cpp
#include "../include/hw2/main_window.hpp"

MainWindow::MainWindow(QWidget* parent) : QMainWindow(parent), ui(new Ui::MainWindowDesign)
{
  ui->setupUi(this);

  QIcon icon("://ros-icon.png");
  this->setWindowIcon(icon);

  qnode = new QNode();

  QObject::connect(qnode, SIGNAL(rosShutDown()), this, SLOT(close()));
  connect(qnode, &QNode::receivedMessage_1, this, [this](QString msg){
    ui->label->setText(msg);
  });
  connect(qnode, &QNode::receivedMessage_2, this, [this](QString msg){
    ui->label_2->setText(msg);
  });
  qnode->start();
}

void MainWindow::closeEvent(QCloseEvent* event)
{
  QMainWindow::closeEvent(event);
}

MainWindow::~MainWindow()
{
  delete ui;
}
```

Mainwindow의 소스파일 중 생성자에서는 qnode 객체에 메모리를 할당한 뒤 connect부분에서는 cmd/vel 값의 일부를 GUI에 나타내기 위해 메시지를 수신했을 때의 시그널과 label값을 변경시키는 슬롯함수를 묶는 코드를 추가했습니다.

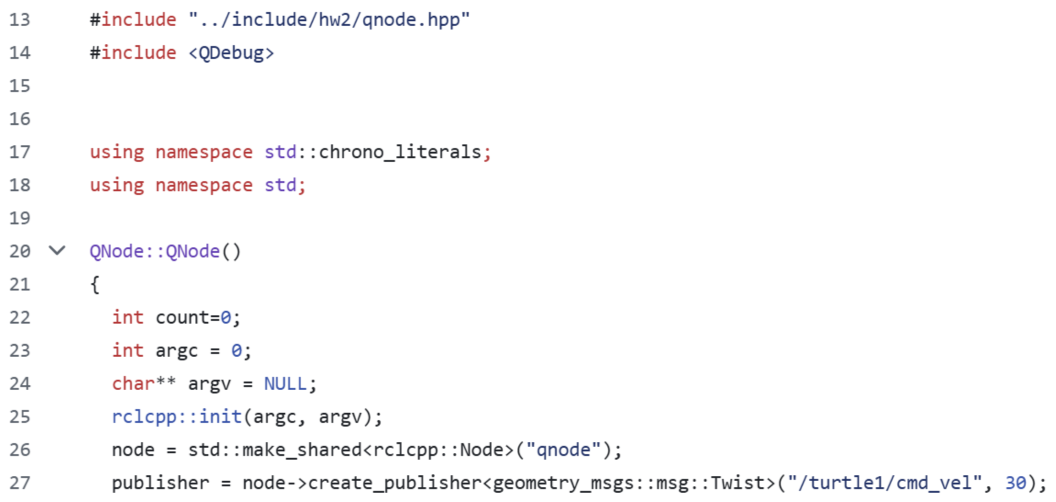


```cpp
void MainWindow::on_pushButton_clicked()
{
    QString text = ui->textEdit->toPlainText();
    // qnode->setMessage(text);
    qnode->publishing(text);

}
```

그리고 publish버튼을 눌렀을 때 메시지를 발행할 수 있도록 qnode 객체의 publishing()

함수를 호출하도록 했습니다.

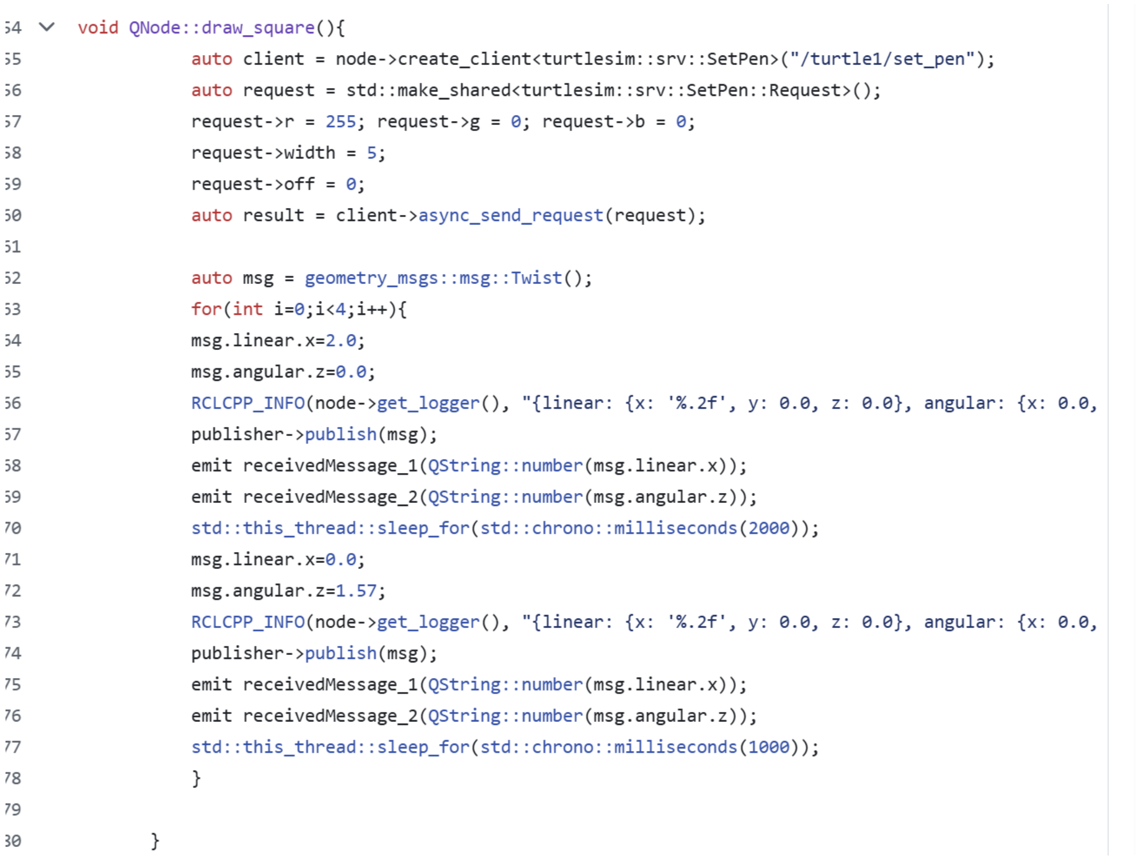
(2) qnode



```cpp
#include "../include/hw2/qnode.hpp"
#include <QDebug>


using namespace std::chrono_literals;
using namespace std;

QNode::QNode()
{
  int count=0;
  int argc = 0;
  char** argv = NULL;
  rclcpp::init(argc, argv);
  node = std::make_shared<rclcpp::Node>("qnode");
  publisher = node->create_publisher<geometry_msgs::msg::Twist>("/turtle1/cmd_vel", 30);
```

Qnode 소스파일에서 Twist클래스를 사용하고 turtlesim을 움직이게 하는 토픽을 사용하도록 발행자를 초기화했습니다.

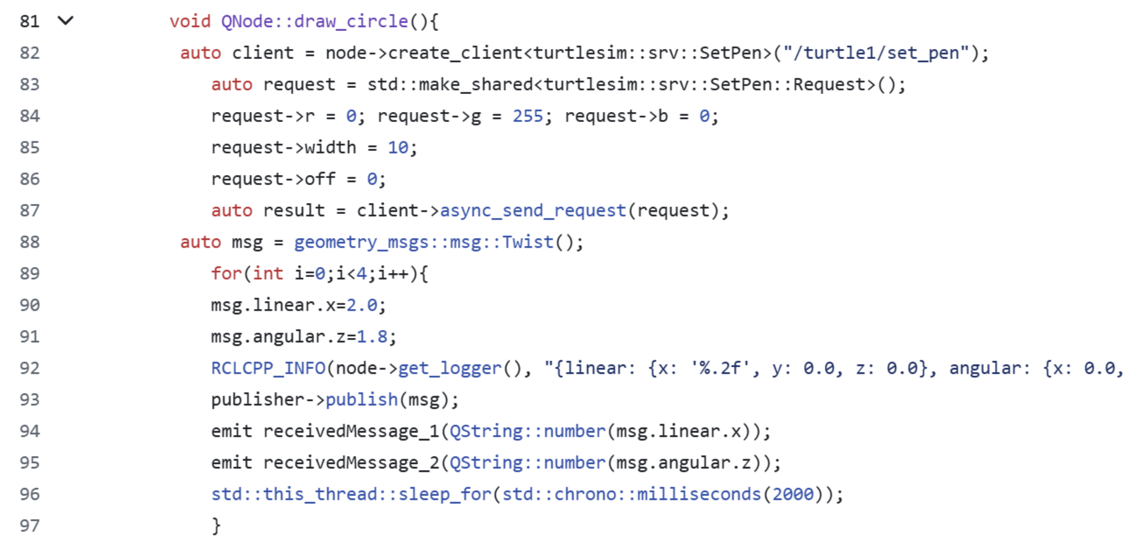
다음으로 기존 파일에서의 변경점 중 draw_square, draw_circle, draw_triang함수입니다.



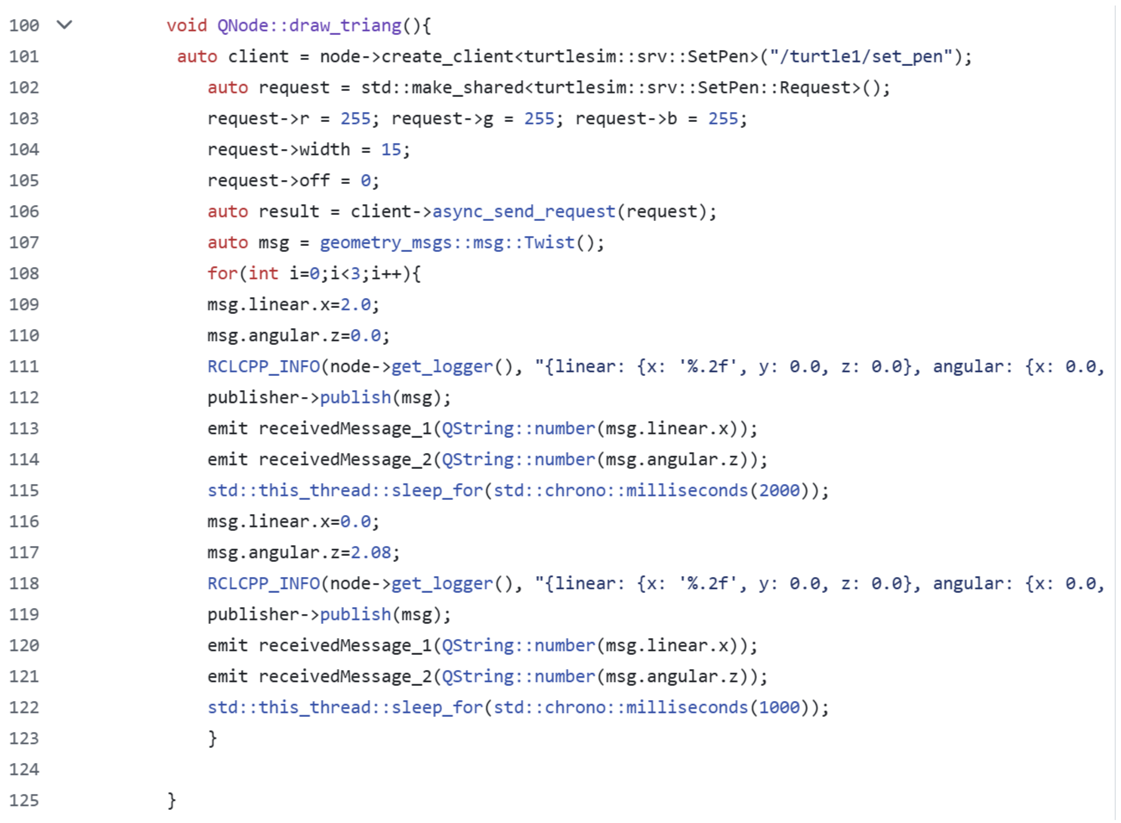
```cpp
void QNode::draw_square(){
        auto client = node->create_client<turtlesim::srv::SetPen>("/turtle1/set_pen");
        auto request = std::make_shared<turtlesim::srv::SetPen::Request>();
        request->r = 255; request->g = 0; request->b = 0;
        request->width = 5;
        request->off = 0;
        auto result = client->async_send_request(request);

        auto msg = geometry_msgs::msg::Twist();
        for(int i=0;i<4;i++){
        msg.linear.x=2.0;
        msg.angular.z=0.0;
        RCLCPP_INFO(node->get_logger(), "{linear: {x: '%.2f', y: 0.0, z: 0.0}, angular: {x: 0.0,
        publisher->publish(msg);
        emit receivedMessage_1(QString::number(msg.linear.x));
        emit receivedMessage_2(QString::number(msg.angular.z));
        std::this_thread::sleep_for(std::chrono::milliseconds(2000));
        msg.linear.x=0.0;
        msg.angular.z=1.57;
        RCLCPP_INFO(node->get_logger(), "{linear: {x: '%.2f', y: 0.0, z: 0.0}, angular: {x: 0.0,
        publisher->publish(msg);
        emit receivedMessage_1(QString::number(msg.linear.x));
        emit receivedMessage_2(QString::number(msg.angular.z));
        std::this_thread::sleep_for(std::chrono::milliseconds(1000));
        }

    }
```

위의 사진은 client와 request를 선언해 펜의 색과 굵기를 변경한 후 차례대로 메시지를 발행하는 부분입니다. turtlesim에서 거북이의 움직임을 조정할 수 있는 turtlesim/cmd_vel이 topic이었다면 펜의 색과 굵기 등을 조절할 수 있는 SetPen은 service이므로 그에 맞는 메시지 자료형을 사용해 발행했습니다.

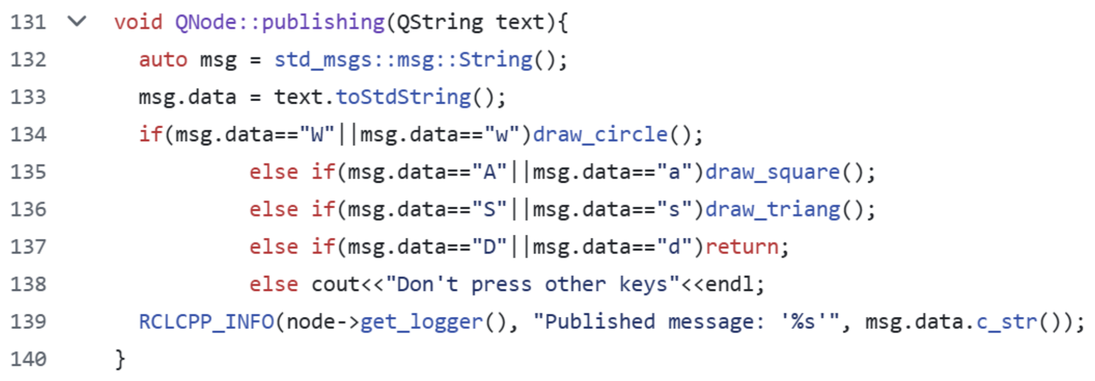
아래는 위와 같이 다른 도형을 그리는 함수를 정의한 부분입니다.



```
void QNode::draw_circle(){
 auto client = node->create_client<turtlesim::srv::SetPen>("/turtle1/set_pen");
    auto request = std::make_shared<turtlesim::srv::SetPen::Request>();
    request->r = 0; request->g = 255; request->b = 0;
    request->width = 10;
    request->off = 0;
    auto result = client->async_send_request(request);
 auto msg = geometry_msgs::msg::Twist();
    for(int i=0;i<4;i++){
    msg.linear.x=2.0;
    msg.angular.z=1.8;
    RCLCPP_INFO(node->get_logger(), "{linear: {x: '%.2f', y: 0.0, z: 0.0}, angular: {x: 0.0,
    publisher->publish(msg);
    emit receivedMessage_1(QString::number(msg.linear.x));
    emit receivedMessage_2(QString::number(msg.angular.z));
    std::this_thread::sleep_for(std::chrono::milliseconds(2000));
    }
```



```
void QNode::draw_triang(){
 auto client = node->create_client<turtlesim::srv::SetPen>("/turtle1/set_pen");
    auto request = std::make_shared<turtlesim::srv::SetPen::Request>();
    request->r = 255; request->g = 255; request->b = 255;
    request->width = 15;
    request->off = 0;
    auto result = client->async_send_request(request);
    auto msg = geometry_msgs::msg::Twist();
    for(int i=0;i<3;i++){
    msg.linear.x=2.0;
    msg.angular.z=0.0;
    RCLCPP_INFO(node->get_logger(), "{linear: {x: '%.2f', y: 0.0, z: 0.0}, angular: {x: 0.0,
    publisher->publish(msg);
    emit receivedMessage_1(QString::number(msg.linear.x));
    emit receivedMessage_2(QString::number(msg.angular.z));
    std::this_thread::sleep_for(std::chrono::milliseconds(2000));
    msg.linear.x=0.0;
    msg.angular.z=2.08;
    RCLCPP_INFO(node->get_logger(), "{linear: {x: '%.2f', y: 0.0, z: 0.0}, angular: {x: 0.0,
    publisher->publish(msg);
    emit receivedMessage_1(QString::number(msg.linear.x));
    emit receivedMessage_2(QString::number(msg.angular.z));
    std::this_thread::sleep_for(std::chrono::milliseconds(1000));
    }

}
```



```cpp
void QNode::publishing(QString text){
  auto msg = std_msgs::msg::String();
  msg.data = text.toStdString();
  if(msg.data=="W"||msg.data=="w")draw_circle();
           else if(msg.data=="A"||msg.data=="a")draw_square();
           else if(msg.data=="S"||msg.data=="s")draw_triang();
           else if(msg.data=="D"||msg.data=="d")return;
           else cout<<"Don't press other keys"<<endl;
  RCLCPP_INFO(node->get_logger(), "Published message: '%s'", msg.data.c_str());
}
```

Publishing 함수입니다. 메시지를 stdstring 형태로 바꾸어 그 값에 따라서 각기 다른 함수를 호출합니다.