ROS2_day1

Day_1-1

```
#include "geometry_msgs/msg/twist.hpp"
#include "rclcpp/parameter_client.hpp"
#include "turtlesim/srv/set_pen.hpp"
#include "std_srvs/srv/empty.hpp"
#include "turtlesim/srv/spawn.hpp"
#include "turtlesim/srv/kill.hpp"
```

- geometry_msgs/msg/twist.hpp
  : twist 메세지 타입 포함, 터틀의 선속도, 각속도 제어
    3개의 floated 변수 ⇒ 묶음 두개(linear묶음, angular 묶음)

```
rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr contr
control_publisher_ = this->create_publisher<geomtr...~>("/tur

void TurtlesimCliNode::control_mode() {
    std::cout << "w,a,s,d 를 눌러 조종화세요, q 입력시 종료\n";
    geometry_msgs::msg::Twist twist;
    char input;
    while ((std::cin >> input) && (input != 'q')) {
        if (input == 'w')
            twist.linear.x = 1.0;
        else if (input == 's')
            twist.linear.x = -1.0;
        else if (input == 'a')
            twist.angular.z = 1.0;
        else if (input == 'd')
            twist.angular.z = -1.0;

        control_publisher_->publish(twist);
        twist.linear.x = 0.0;
        twist.angular.z = 0.0;
```

```
    }
}
```

control_publisher_ : 퍼블리셔 선언, turtle1/cmd_vel 토픽 pub
Twist타입의 twist 객체 생성, twist에 명령 담아서 퍼블리시
조건문으로 각각 기능 구현 (선속도,각속도)
ex) 전진 (1.0, 0.0), 제자리 회전(0.0, 1.0), 자동차처럼 회전하면서 전진(1.0, 1.0)

&& (!'q') 해줌으로써 q 입력전까지는 제어하도록 구현

cli입력이므로 입력후 (0.0, 0.0)으로 다시 정리 초기화

- rclcpp/parameter_client.hpp
  : ros2에서 파라미터 관리를 위한 클라이언트 관련 툴 제공(터틀심 터틀 노드 ⇒ 배경색 설정)
  'AsyncParametersClient' 클래스를 제공 - 비동기 방식
  비동기 방식 : 요청을 보낸 후 결과를 기다리지 않고 다른 작업을 수행하게 해주는 방식

```
void TurtlesimCliNode::set_background_color() {
    std::string color_input;
    std::cout << "r,g,b 셋 중 하나를 입력하세요: ";
    std::cin >> color_input;

    int r = 0, g = 0, b = 0;
    if (color_input == "r")
        r = 255;
    else if (color_input == "g")
        g = 255;
    else if (color_input == "b")
        b = 255;

    auto parameter_client = std::make_shared<rclcpp::AsyncPar
    if (parameter_client->wait_for_service(std::chrono::secon
        parameter_client->set_parameters({
            rclcpp::Parameter("background_r", r),
            rclcpp::Parameter("background_g", g),
            rclcpp::Parameter("background_b", b)
        });
```

```
        // Empty 서비스 요청을 통해 배경 초기화
        auto request = std::make_shared<std_srvs::srv::Empty:
        auto result = clear_client_->async_send_request(reque

        if (rclcpp::spin_until_future_complete(this->shared_f
            std::cout << "배경 색 변경\n";
        }
    }
}
```

'AsyncParametersClient' 객체인 parameter_clinet 통해 'turtlesim' 의 배경색 파라미터(background r,g,b) 설정
parameter_client→set_parameters()함수를 사용해 배경색 값 설정 - 비동기방식
여기서 Empty 서비스 요청을 통해 배경을 초기화 해줌

서비스 : ros2 토픽과는 다른 통신 방식으로 요청,응답 구조

```
#include "std_srvs/srv/empty.hpp"

rclcpp::Client<std_srvs::srv::Empty>::SharedPtr clear_client_

auto request = std::make_shared<std_srvs::srv::Empty::Request
auto result = clear_client_->async_send_request(request);
```

std_srvs::srv::Empty::Request 객체를 생성, empty이므로 깡통요청 객체(트리거 역할 느낌)
clear_client_ 터틀심 노드에 화면초기화 서비스인 /clear 서비스 존재, 이걸 받아와서 화면 초기화
즉 변경된 파라미터 값(background_r,g,b)을 설정 해줌

- turtlesim/srv/spawn.hpp, turtlesim/srv/kill.hpp

```
auto spawn_request = std::make_shared<turtlesim::srv::Spawn::
spawn_request->x = 5.5;
spawn_request->y = 5.5;
spawn_request->theta = 0.0;
spawn_request->name = "turtle1";
```

```
if (spawn_client_->wait_for_service(std::chrono::seconds(1)))
    auto spawn_result = spawn_client_->async_send_request(spa
    if (rclcpp::spin_until_future_complete(this->shared_from_
        std::cout << "새로운 거북이가 생성되었습니다.\n";
}
```

spawn 서비스의 구조
request : 새로운 터틀의 초기 위치, 방향, 이름 지정 가능
response : 생성된 터틀의 이름을 반환

spawn_request 객체 생성 설정(위치,방향,이름)해주고, spawn_client_ 를 통해 비동기 방식으로 Spawn 서비스 요청 전송 ⇒ spin_until_future_complete로 요청이 완료될 때까지 대기
대기 해주는 이유는 비동기 방식임에도 대기가 없으면 요청의 결과가 오기도 전에 다음 코드가 실행됨.

사실상 비동기 방식의 이점은 못 살린다.

kill 서비스의 구조
request : 제거할 터틀의 이름을 지정
response : 별도로 없음

kill_request 객체 생성
kill_client_→async_send_request(kill_request);로 요청을 보내고,
spin_until_future_complete로 요청이 완료될 때까지 대기
요청이 성공하면 메시지를 출력

- turtlesim/srv/set_pen.hpp
  : 터틀심 거북이가 남기는 pen 색상과 두께 설정(서비스 타입)

```
auto set_pen_request = std::make_shared<turtlesim::srv::SetPe
set_pen_request->r = 255;       // 빨간색 설정
set_pen_request->g = 0;         // 녹색 없음
set_pen_request->b = 0;         // 파란색 없음
set_pen_request->width = 3;     // 펜 두께 설정
set_pen_request->off = 0;       // 펜 활성화

// 비동기 방식으로 서비스 요청 전송
if (set_pen_client_->wait_for_service(std::chrono::seconds(1)
```

```
    auto result = set_pen_client_->async_send_request(set_pen
    if (rclcpp::spin_until_future_complete(this->shared_from_
        std::cout << "펜 설정이 완료되었습니다.\n";
    }
}
```

SetPen 서비스는 request와 response로 구성됩니다. 이 서비스는 터틀의 펜 속성을 설정하기 위해 사용된다. 여기에서 요청(Request)만 정의되며, 응답(Response)은 필요하지 않다

Request

터틀의 펜 색상과 두께, 그리고 펜의 활성화 상태를 설정하는 필드를 포함합니다.

필드 정의:

uint8 r: 펜의 **빨간색** 성분 (0-255).

uint8 g: 펜의 **녹색** 성분 (0-255).

uint8 b: 펜의 **파란색** 성분 (0-255).

uint8 width: 펜의 **두께**를 나타냅니다.

uint8 off: 펜의 활성 상태를 나타내는 필드입니다.

'0'이면 펜이 **활성화**되어 이동할 때 선을 남깁니다.

'1'이면 펜이 **비활성화**되어 이동해도 선이 남지 않습니다.

이 요청 구조를 통해 사용자는 터틀이 이동할 때 남기는 선의 색상과 두께를 설정할 수 있습니다.

set_pen_client_→async_send_request(set_pen_request);를 사용하여 비동기 방식으로 요청을 전송

spin_until_future_complete로 요청의 완료를 확인하고, 성공 시 메시지를 출력

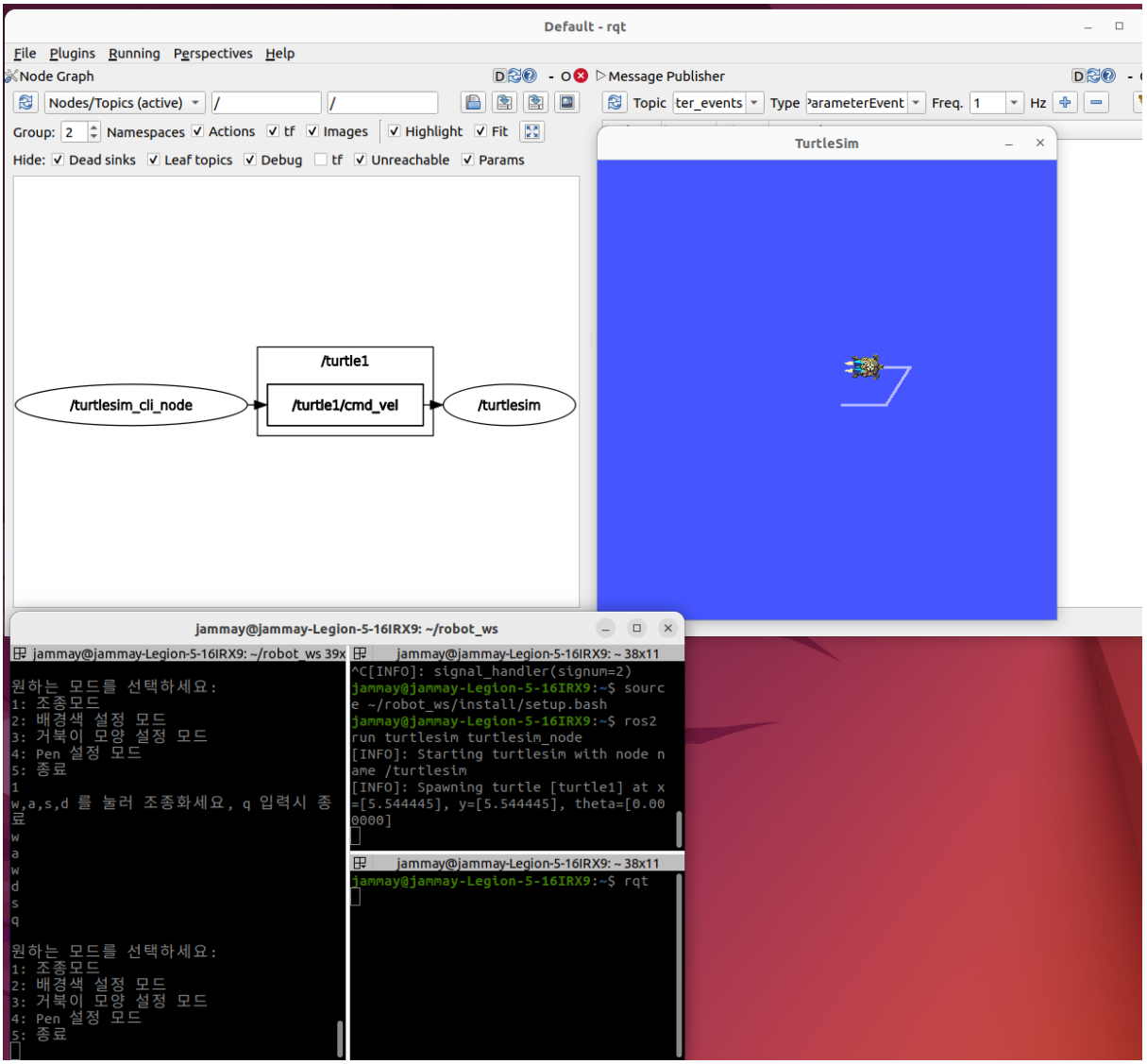
기타

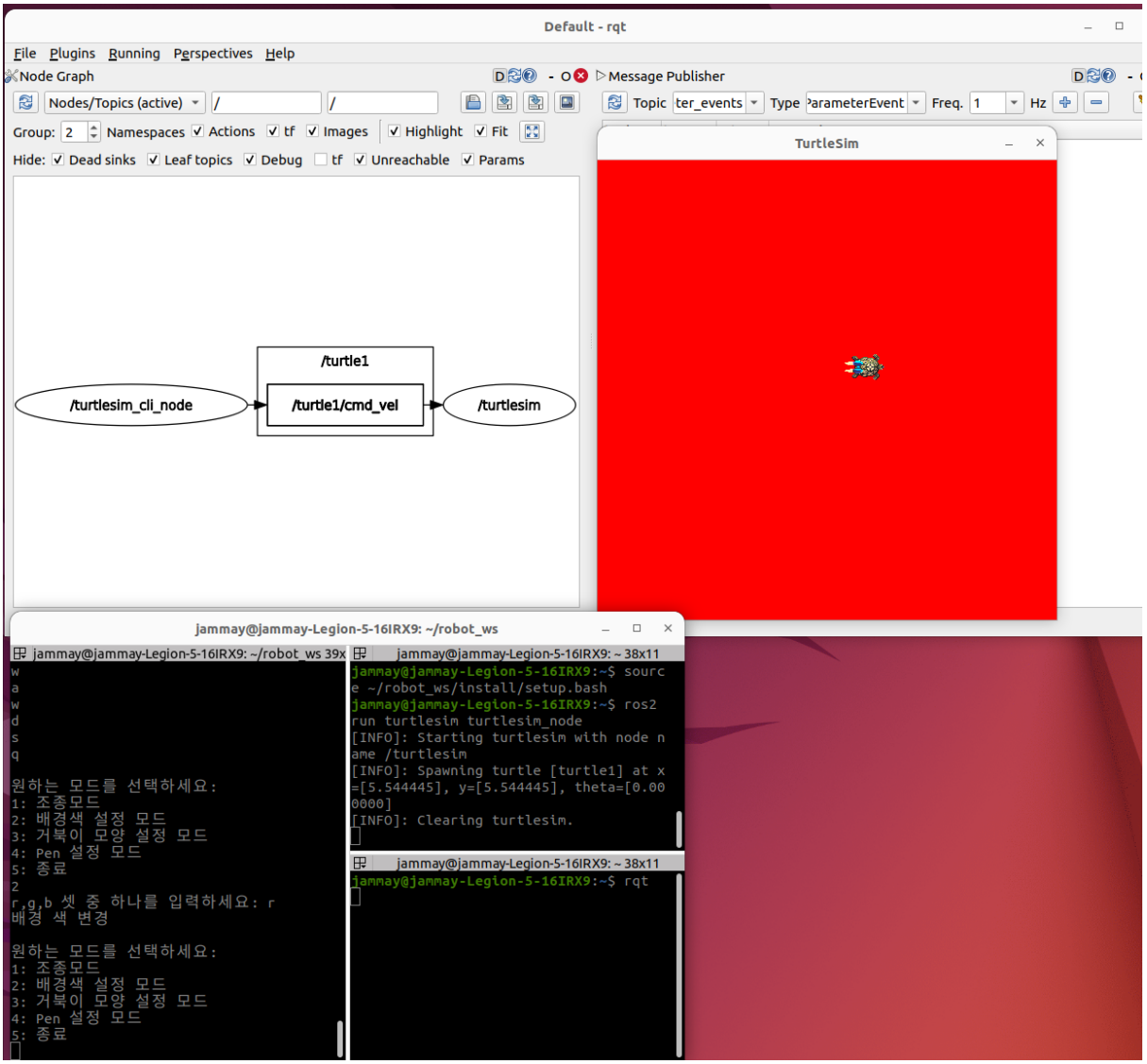
- 스마트 포인터
  std::shared_ptr
  역할 :
  여러 개의 소유자가 동일한 자원에 접근할 수 있도록 하는 공유 포인터
  동일한 자원을 참조하는 포인터들이 모두 소멸될 때까지 자원이 유지

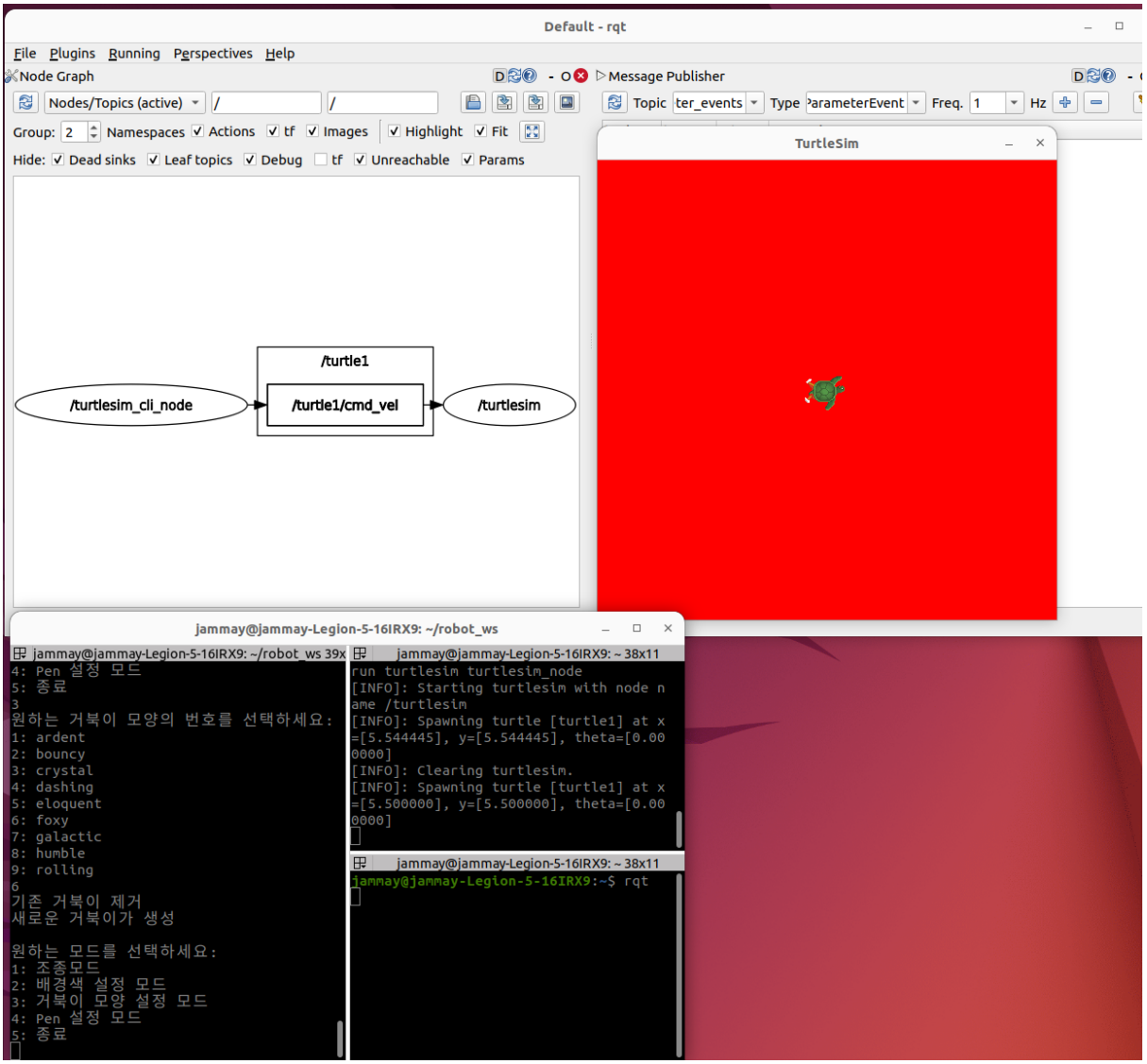
  ros2 사용 :
  노드를 생성하거나 클라이언트와 서비스 객체를 정의 관리, ROS2에서 퍼블리셔와 서브스 크라이버가 메시지를 주고받을 때 관리. 이는 메시지가 여러 곳에서 참조되더라도 참조가 모두 해제될 때까지 메모리가 안전하게 유지

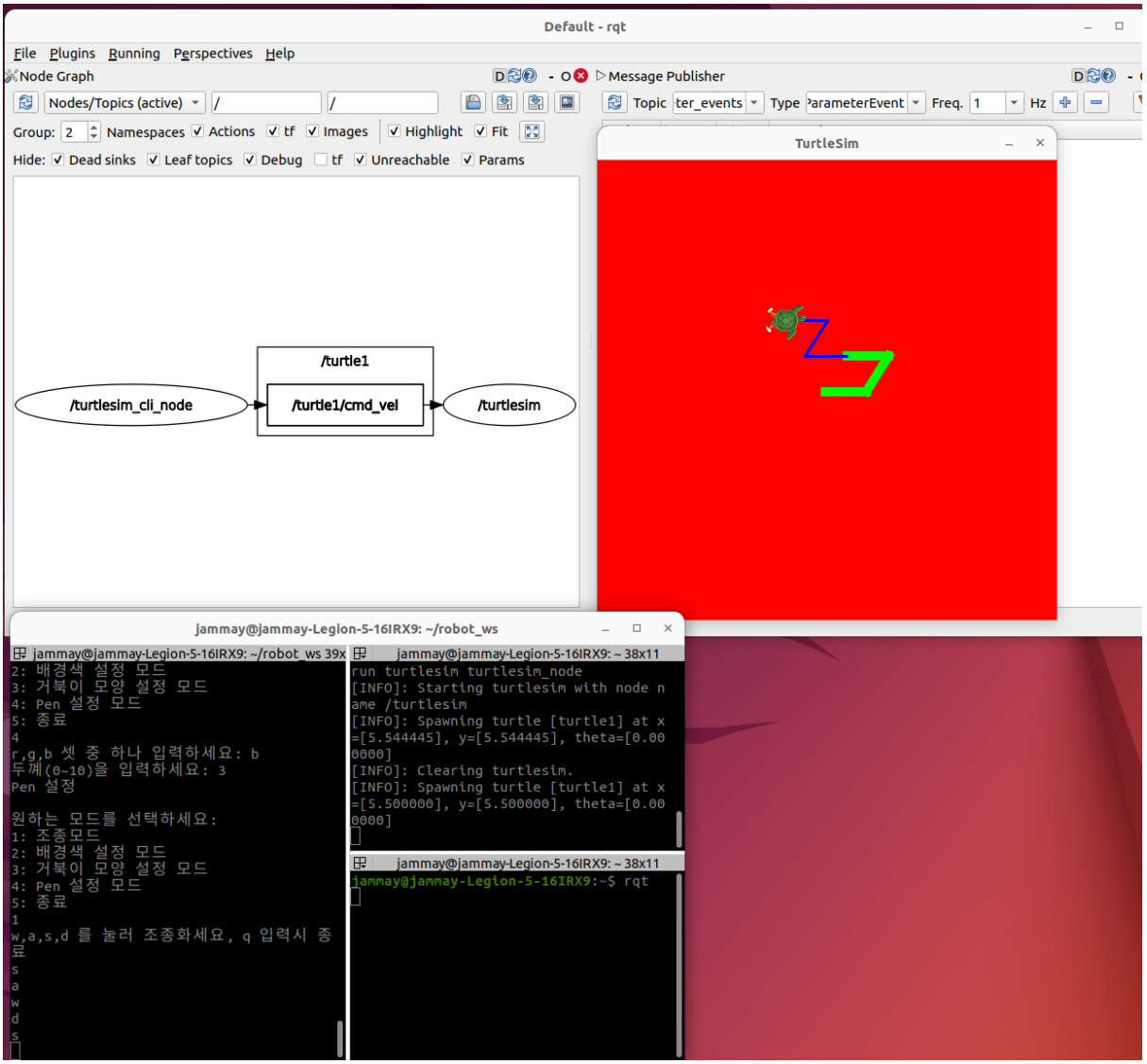
피드백

- 통신 방식, 스마트 포인터, 비동기 방식 등 자세하게 공부하지 않고 완성시킨 점(gpt 사용이 원인)
- 거북이 설정 모드에서 랜덤으로 거북이 킬,스폰만 구현함
- 아이디어로 기존 거북이에 기존 거북이 이름 지정⇒킬⇒랜덤(turtle1)스폰 순서의 모드 하나,
  설정된 거북이 이름(example_fix_name) 스폰 하는 모드 나누었으면 구현 가능 했을 거 같다.


Default - rqt
File Plugins Running Perspectives Help
Node Graph
Nodes/Topics (active)
Group: 2 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
/turtle1
/turtlesim_cli_node
/turtle1/cmd_vel
/turtlesim
Message Publisher
Topic ter_events Type arameterEvent Freq. 1 Hz
TurtleSim
jammay@jammay-Legion-5-16IRX9: ~/robot_ws
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39x
원하는 모드를 선택하세요:
1: 조종모드
2: 배경색 설정 모드
3: 거북이 모양 설정 모드
4: Pen 설정 모드
5: 종료
1
w,a,s,d 를 눌러 조종화세요, q 입력시 종료
w
a
w
d
s
q
원하는 모드를 선택하세요:
1: 조종모드
2: 배경색 설정 모드
3: 거북이 모양 설정 모드
4: Pen 설정 모드
5: 종료
jammay@jammay-Legion-5-16IRX9: ~ 38x11
^C[INFO]: signal_handler(signum=2)
jammay@jammay-Legion-5-16IRX9:~$ source ~/robot_ws/install/setup.bash
jammay@jammay-Legion-5-16IRX9:~$ ros2 run turtlesim turtlesim_node
[INFO]: Starting turtlesim with node name /turtlesim
[INFO]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
jammay@jammay-Legion-5-16IRX9: ~ 38x11
jammay@jammay-Legion-5-16IRX9:~$ rqt


Default - rqt
File Plugins Running Perspectives Help
Node Graph
Nodes/Topics (active)
Group: 2 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
/turtle1
/turtlesim_cli_node
/turtle1/cmd_vel
/turtlesim
Message Publisher
Topic ter_events Type arameterEvent Freq. 1 Hz
TurtleSim
jammay@jammay-Legion-5-16IRX9: ~/robot_ws
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39x
w
a
w
d
s
q
원하는 모드를 선택하세요:
1: 조종모드
2: 배경색 설정 모드
3: 거북이 모양 설정 모드
4: Pen 설정 모드
5: 종료
2
r,g,b 셋 중 하나를 입력하세요: r
배경 색 변경
원하는 모드를 선택하세요:
1: 조종모드
2: 배경색 설정 모드
3: 거북이 모양 설정 모드
4: Pen 설정 모드
5: 종료
jammay@jammay-Legion-5-16IRX9: ~ 38x11
jammay@jammay-Legion-5-16IRX9:~$ sourc
e ~/robot_ws/install/setup.bash
jammay@jammay-Legion-5-16IRX9:~$ ros2
run turtlesim turtlesim_node
[INFO]: Starting turtlesim with node n
ame /turtlesim
[INFO]: Spawning turtle [turtle1] at x
=[5.544445], y=[5.544445], theta=[0.00
0000]
[INFO]: Clearing turtlesim.
jammay@jammay-Legion-5-16IRX9: ~ 38x11
jammay@jammay-Legion-5-16IRX9:~$ rqt


Default - rqt
File Plugins Running Perspectives Help
Node Graph
Nodes/Topics (active)
Group: 2 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
/turtle1
/turtlesim_cli_node
/turtle1/cmd_vel
/turtlesim
Message Publisher
Topic ter_events Type arameterEvent Freq. 1 Hz
TurtleSim
jammay@jammay-Legion-5-16IRX9: ~/robot_ws
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39x
4: Pen 설정 모드
5: 종료
3
원하는 거북이 모양의 번호를 선택하세요:
1: ardent
2: bouncy
3: crystal
4: dashing
5: eloquent
6: foxy
7: galactic
8: humble
9: rolling
6
기존 거북이 제거
새로운 거북이가 생성
원하는 모드를 선택하세요:
1: 조종모드
2: 배경색 설정 모드
3: 거북이 모양 설정 모드
4: Pen 설정 모드
5: 종료
jammay@jammay-Legion-5-16IRX9: ~ 38x11
run turtlesim turtlesim_node
[INFO]: Starting turtlesim with node n
ame /turtlesim
[INFO]: Spawning turtle [turtle1] at x
=[5.544445], y=[5.544445], theta=[0.00
0000]
[INFO]: Clearing turtlesim.
[INFO]: Spawning turtle [turtle1] at x
=[5.500000], y=[5.500000], theta=[0.00
0000]
jammay@jammay-Legion-5-16IRX9: ~ 38x11
jammay@jammay-Legion-5-16IRX9:~$ rqt



day1-2

day1-1에서 다룬 기능들은 제외하겠습니다.

여기서는 main.cpp파일은 따로 만들지 않아서 while(rclcpp::ok())을 사용해 cli가 끊기지 않게 하였다.

rclcpp::ok()는 ros2 노드가 실행중인지 확인하기 위해 루프내에서 사용된다. 즉 ctrl+c 누르면 false로 설정

모드 도형 그리기 선택시

각 함수 실행

```
if (shape == 1)
        {
            draw_triangle(size);
        }
        else if (shape == 2)
        {
            draw_circle(size);
        }
        else if (shape == 3)
        {
            draw_square(size);
        }
```

```
void TurtlesimDraw::draw_triangle(int size)
{
    auto msg = geometry_msgs::msg::Twist();
    for (int i = 0; i < 3; i++)
    {
        msg.linear.x = size;
        msg.angular.z = 0.0;
        cmd_vel_pub_->publish(msg);
        rclcpp::sleep_for(std::chrono::seconds(1));
        msg.linear.x = 0.0;
        msg.angular.z = 2.094; // 120도
        cmd_vel_pub_->publish(msg);
        rclcpp::sleep_for(std::chrono::seconds(1));
    }
}
```

120도를 직접 잡아주어 3번 반복하여 삼각형
직진→회전(3번 반복)

rclcpp::sleep_for(std::chrono::seconds(1));
: 스레드를 멈추게 하는 부분인데 딜레이 기능이라고 생각해 직진후 충분한 회전시간을 주기 위해 넣었다.

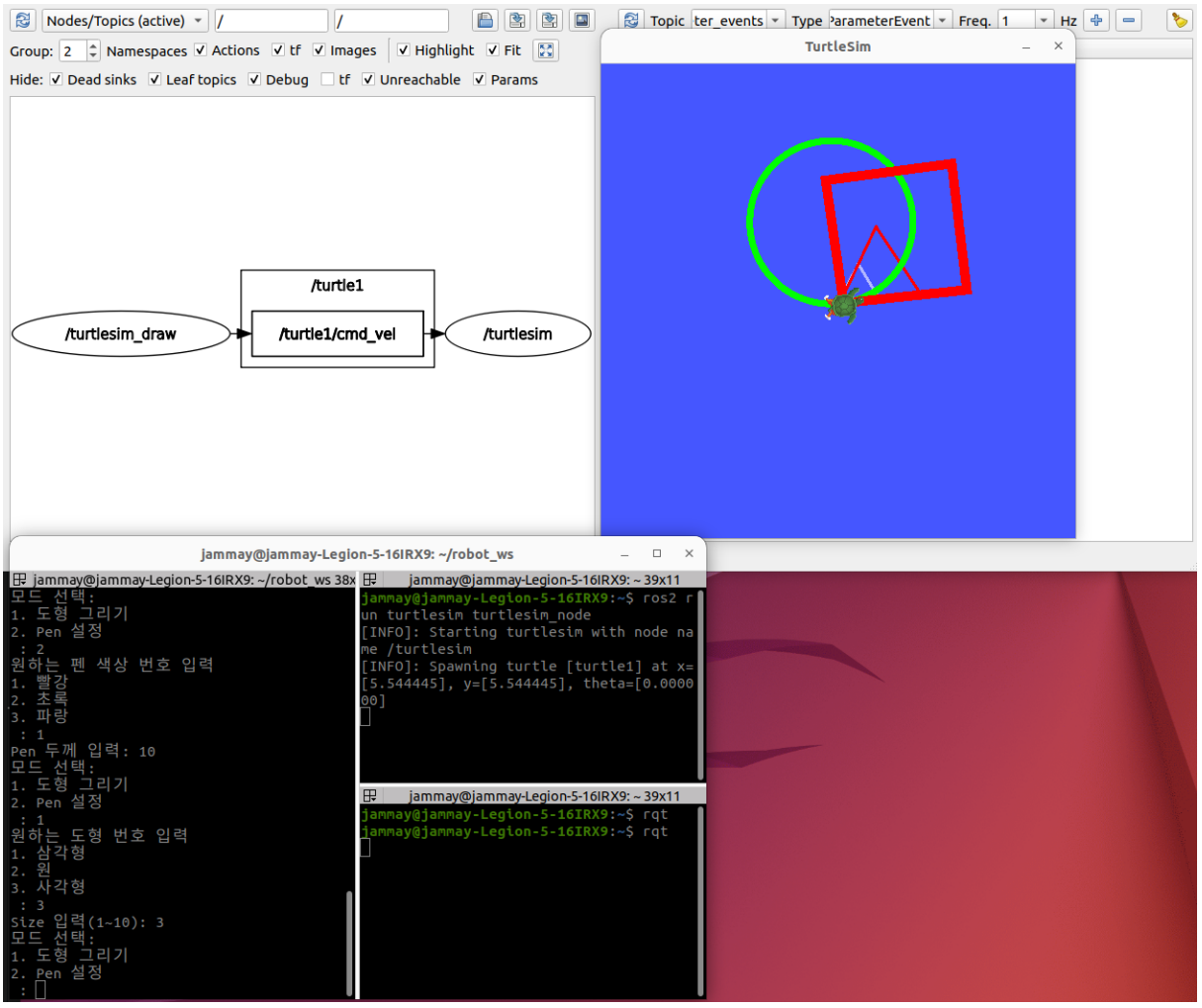
```
void TurtlesimDraw::draw_circle(int size)
{
    auto msg = geometry_msgs::msg::Twist();
    for (int i = 0; i < 12; i++)
    {
        msg.linear.x = size / 2;
        msg.angular.z = 0.5236; // 30도
        cmd_vel_pub_->publish(msg);
        rclcpp::sleep_for(std::chrono::seconds(1));
    }
}
```

마찬가지로 30도로 12번 돌려주었다.
대신 원으로 그려야 하기에 ex(1.0, 1.0)식으로 자동차처럼 앞으로 가며 회전하는게 특징

```
void TurtlesimDraw::draw_square(int size)
{
    auto msg = geometry_msgs::msg::Twist();
    for (int i = 0; i < 4; i++)
    {
        msg.linear.x = size;
        msg.angular.z = 0.0;
        cmd_vel_pub_->publish(msg);
        rclcpp::sleep_for(std::chrono::seconds(1));
        msg.linear.x = 0.0;
        msg.angular.z = 1.5708; // 90도
        cmd_vel_pub_->publish(msg);
        rclcpp::sleep_for(std::chrono::seconds(1));
    }
}
```

90도 4번 반복

로직은 삼각형과 동일하다



day1-3

hpp 파일들로 틀을 보자면

- talker.hpp

```
#ifndef TALKER_HPP
#define TALKER_HPP

#include <rclcpp/rclcpp.hpp>
#include <std_msgs/msg/string.hpp>
#include <std_msgs/msg/int64.hpp>
```

```
class Talker : public rclcpp::Node {
public:
    Talker();
private:
    void publish_message();
    rclcpp::Publisher<std_msgs::msg::String>::SharedPtr publi
    rclcpp::Publisher<std_msgs::msg::Int64>::SharedPtr count_
    rclcpp::TimerBase::SharedPtr timer_;
};

#endif // TALKER_HPP
```

talker 노드
Talker 클래스 : rclcpp::Node 상속받아 ros2노드로서의 기능 제공

publish_message() : 퍼블리셔 역할을 해줄 함수
timer_; : 별도의 스레드 관리를 안하기 위해(간단한 퍼블리시, 서브스크라이브 기능을 하는 패키지)

- listener.hpp

```
#ifndef LISTENER_HPP
#define LISTENER_HPP

#include <rclcpp/rclcpp.hpp>
#include <std_msgs/msg/string.hpp>
#include <std_msgs/msg/int64.hpp>

class Listener : public rclcpp::Node {
public:
    Listener();
private:
    void string_callback(const std_msgs::msg::String::SharedP
    void int_callback(const std_msgs::msg::Int64::SharedPtr m
    rclcpp::Subscription<std_msgs::msg::String>::SharedPtr st
    rclcpp::Subscription<std_msgs::msg::Int64>::SharedPtr int
};
```

```
#endif // LISTENER_HPP
```

listener 노드

퍼블리시 처럼 서브스크립션 변수 선언
콜백함수
: talker노드에서 퍼블리시된 해당 토픽이 수신될 때 트리거 됨

- talker.cpp

```
Talker::Talker() : Node("talker") {
    publisher_ = this->create_publisher<std_msgs::msg::String
    count_publisher_ = this->create_publisher<std_msgs::msg::
    timer_ = this->create_wall_timer(std::chrono::seconds(1),
        std::bind(&Talker::publish_message, this));
}
```

publisher_, count_publisher_ : 퍼블리셔로써 각각 String, Int64를 보낸다, 토픽 명은 각각
"/chatter_cli", "/chatter_count"이다.

timer_ = this→create_wall_timer(std::chrono::seconds(1),
std::bind(&Talker::publish_message, this));
: 주기적으로 트리거 되는 타이머를 생성하는 메서드 호출

std::chrono::seconds(1) : 콜백 간의 간격 지정

std::bind(&Talker::publish_message, this) : 멤버 함수 publish_message()를 현재 인스턴스(this)에 바인딩(특정 함수나 변수를 다른 요소에 연결하는 것을 의미, 멤버함수와 인스턴스를 바인딩하면 해당 함수 호출시 어느 인스턴스에서 실행되어야 하는지 명확이 지정 가능) 하는데 사용, 이를 통해 타이머가 트리거 될 때 publish_message함수가 올바른 객체에서 호출 되도록한다.

create_wall_timer는 호출 가능한 객체를 기대하기 때문에 std::bind 가 필요하며, publish_message()는 호출될 인스턴스가 필요한 멤버 함수이기 때문입니다.

create_wall_timer : 타이머를 생성

std::bind : 멤버 함수를 호출 가능한 객체로 만들어주기 위해 사용, 멤버 함수는 클래스 인스턴스와 함께 호출되어야 하기 때문에, this 포인터를 바인딩하여 publish_message함수가 Talker 클래스의 인스턴스에서 호출

```
std::cin.ignore(std::numeric_limits<std::streamsize>::max(),

    publisher_->publish(msg);
    count_publisher_->publish(count_msg);
```

카운트 값을 읽은 후, std::cin.ignore를 사용하여 입력 버퍼를 지워, 입력에 방해가 되지 않도록

- listener.cpp

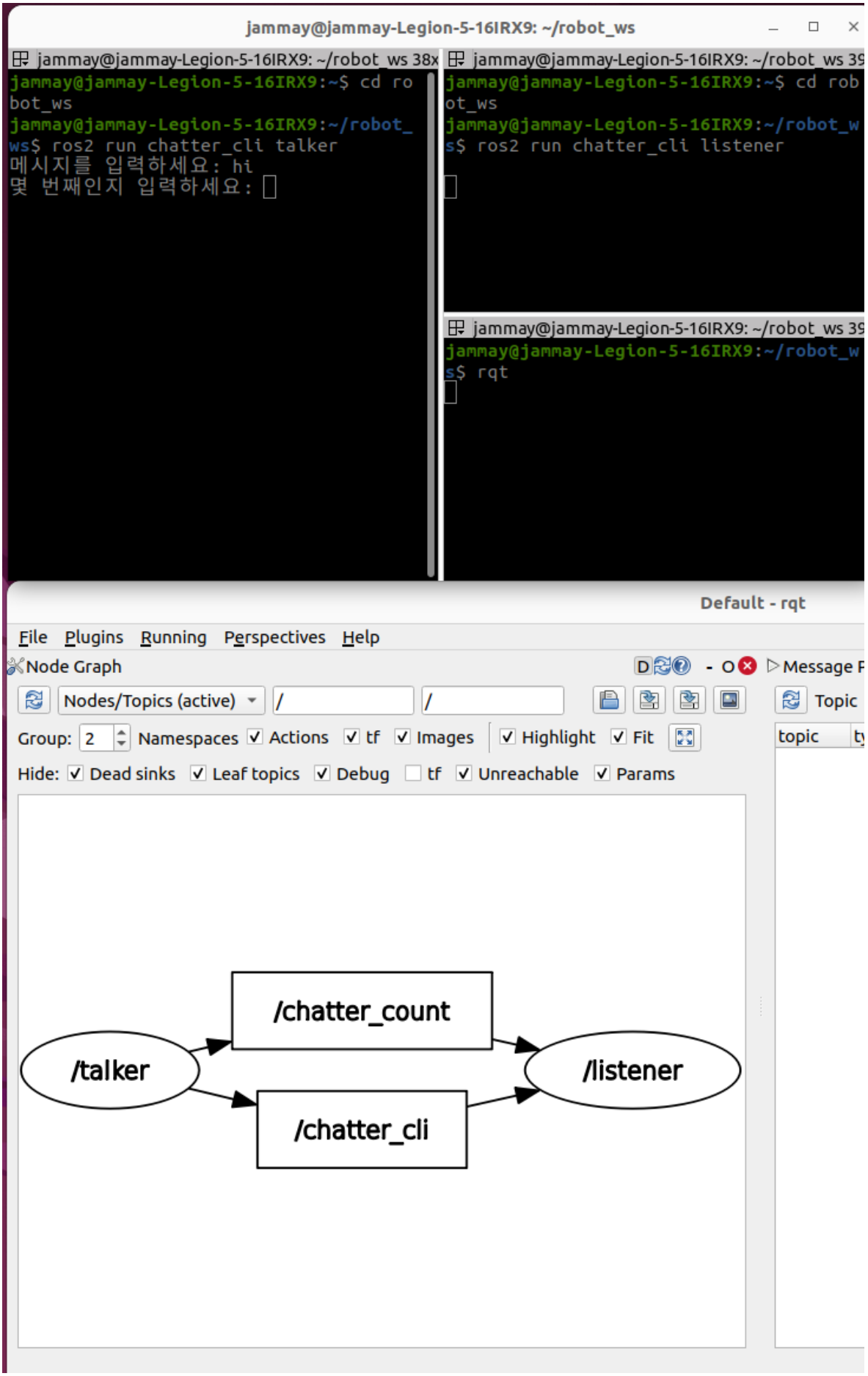
```
Listener::Listener() : Node("listener") {
    string_subscription_ = this->create_subscription<std_msgs
    ("/chatter_cli", 10, std::bind(&Listener::string_callback

    int_subscription_ = this->create_subscription<std_msgs::m
    ("/chatter_count", 10, std::bind(&Listener::int_callback,
}

void Listener::string_callback(const std_msgs::msg::String::S
    RCLCPP_INFO(this->get_logger(), "Subscribed: '%s'", msg->
}

void Listener::int_callback(const std_msgs::msg::Int64::Share
    RCLCPP_INFO(this->get_logger(), "Count: '%ld'", msg->data
}
```

string_callback 함수
: /chatter_cli 토픽에서 문자열 메세지를 수진했을 때 호출, 메세지 출력

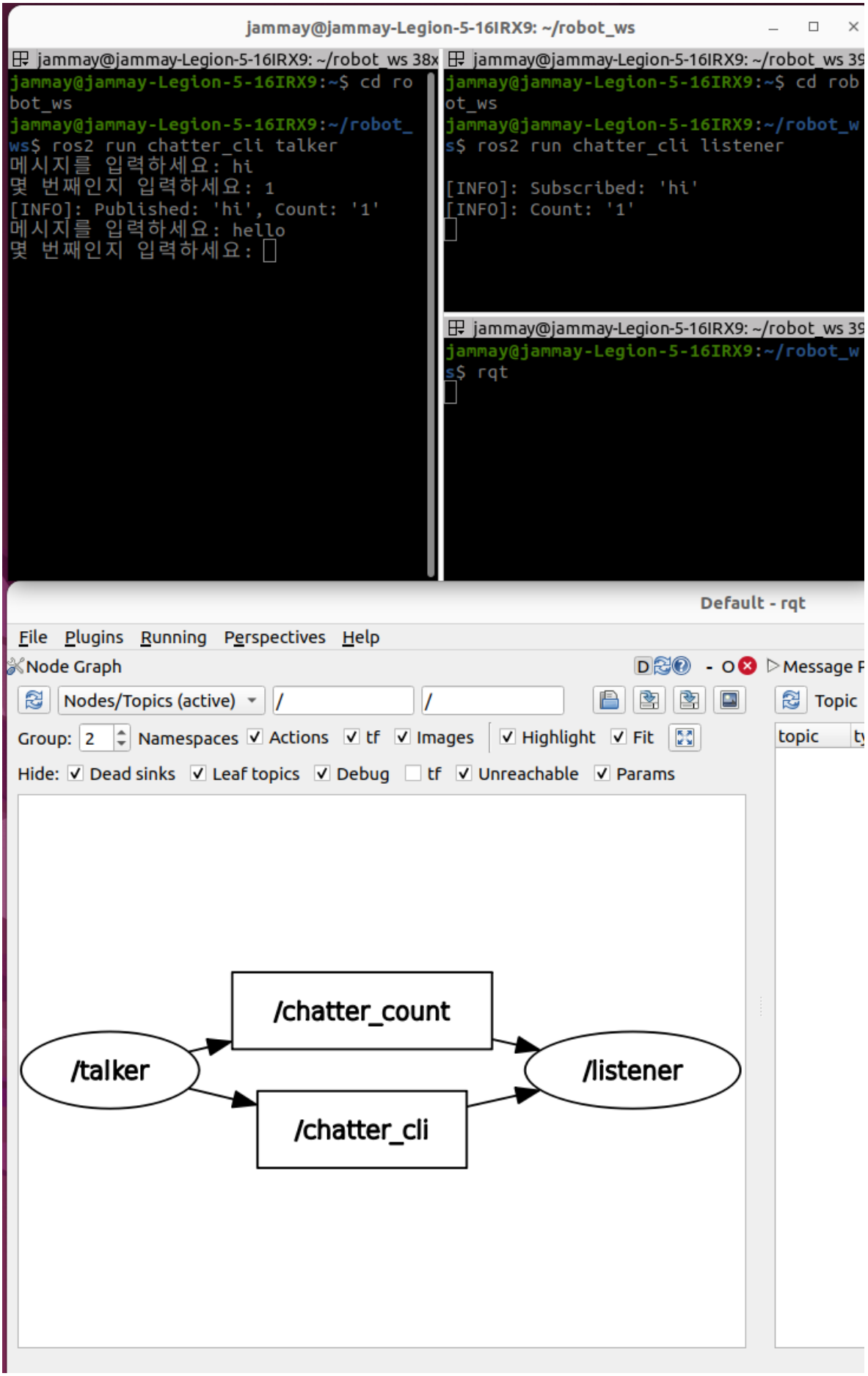
int_callback 함수

: /chatter_count 토픽에서 정수 메세지 수신 시 호출 ⇒ 메세지 출력

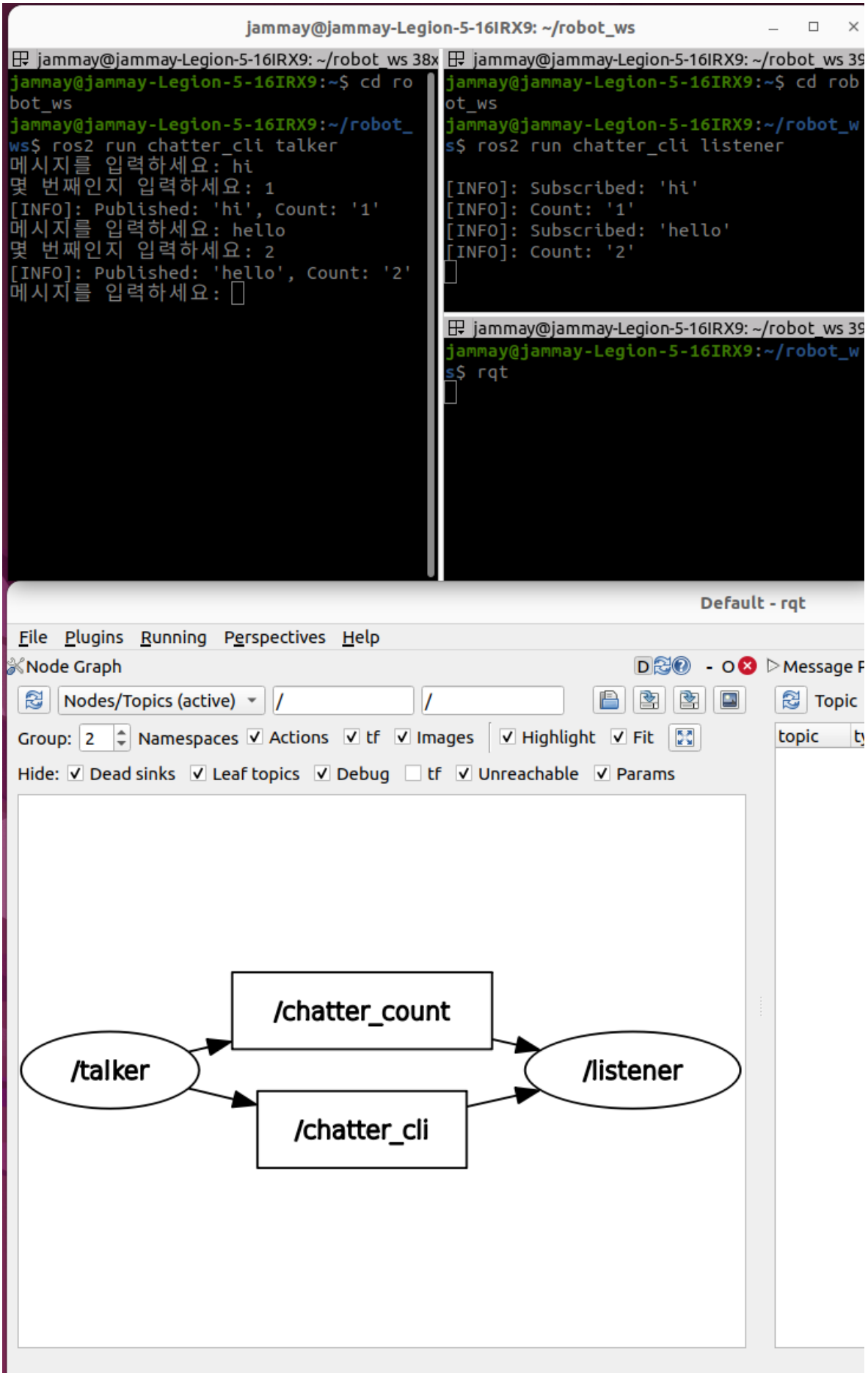
- 첫번째 string만 입력시


jammay@jammay-Legion-5-16IRX9: ~/robot_ws
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 38x
jammay@jammay-Legion-5-16IRX9:~$ cd ro
bot_ws
jammay@jammay-Legion-5-16IRX9:~/robot_
ws$ ros2 run chatter_cli talker
메시지를 입력하세요: hi
몇 번째인지 입력하세요:
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39
jammay@jammay-Legion-5-16IRX9:~$ cd rob
ot_ws
jammay@jammay-Legion-5-16IRX9:~/robot_w
s$ ros2 run chatter_cli listener
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39
jammay@jammay-Legion-5-16IRX9:~/robot_w
s$ rqt
Default - rqt
File Plugins Running Perspectives Help
Node Graph
Message P
Nodes/Topics (active)
Topic
Group: 2 Namespaces Actions tf Images Highlight Fit
topic
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
/chatter_count
/talker
/listener
/chatter_cli

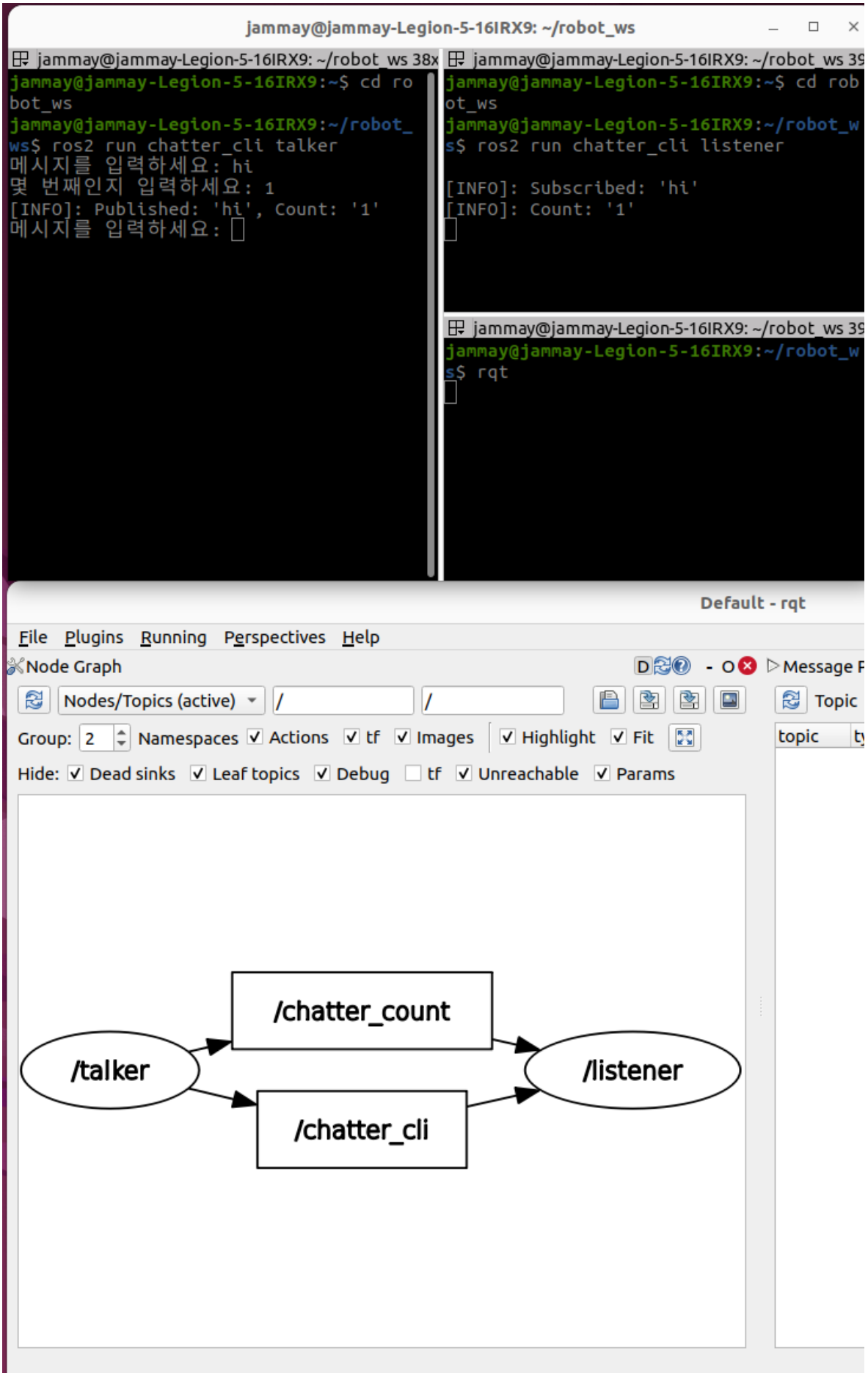
- 첫번째 int 입력시 ⇒ listen 출력


jammay@jammay-Legion-5-16IRX9: ~/robot_ws
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 38x
jammay@jammay-Legion-5-16IRX9:~$ cd robot_ws
jammay@jammay-Legion-5-16IRX9:~/robot_ws$ ros2 run chatter_cli talker
메시지를 입력하세요: hi
몇 번째인지 입력하세요: 1
[INFO]: Published: 'hi', Count: '1'
메시지를 입력하세요: hello
몇 번째인지 입력하세요:
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39
jammay@jammay-Legion-5-16IRX9:~$ cd robot_ws
jammay@jammay-Legion-5-16IRX9:~/robot_ws$ ros2 run chatter_cli listener
[INFO]: Subscribed: 'hi'
[INFO]: Count: '1'
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39
jammay@jammay-Legion-5-16IRX9:~/robot_ws$ rqt
Default - rqt
File Plugins Running Perspectives Help
Node Graph
Nodes/Topics (active)
Group: 2 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
Topic
topic
/chatter_count
/talker
/listener
/chatter_cli

- 두번째 string 입력


jammay@jammay-Legion-5-16IRX9: ~/robot_ws
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 38x
jammay@jammay-Legion-5-16IRX9:~$ cd robot_ws
jammay@jammay-Legion-5-16IRX9:~/robot_ws$ ros2 run chatter_cli talker
메시지를 입력하세요: hi
몇 번째인지 입력하세요: 1
[INFO]: Published: 'hi', Count: '1'
메시지를 입력하세요: hello
몇 번째인지 입력하세요: 2
[INFO]: Published: 'hello', Count: '2'
메시지를 입력하세요:
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39
jammay@jammay-Legion-5-16IRX9:~$ cd robot_ws
jammay@jammay-Legion-5-16IRX9:~/robot_ws$ ros2 run chatter_cli listener
[INFO]: Subscribed: 'hi'
[INFO]: Count: '1'
[INFO]: Subscribed: 'hello'
[INFO]: Count: '2'
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39
jammay@jammay-Legion-5-16IRX9:~/robot_ws$ rqt
Default - rqt
File Plugins Running Perspectives Help
Node Graph
Nodes/Topics (active)
Group: 2 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
Topic
topic
/chatter_count
/talker
/listener
/chatter_cli

- 두번째 int 입력 ⇒ listen 출력


jammay@jammay-Legion-5-16IRX9: ~/robot_ws
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 38x
jammay@jammay-Legion-5-16IRX9:~$ cd robot_ws
jammay@jammay-Legion-5-16IRX9:~/robot_ws$ ros2 run chatter_cli talker
메시지를 입력하세요: hi
몇 번째인지 입력하세요: 1
[INFO]: Published: 'hi', Count: '1'
메시지를 입력하세요:
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39
jammay@jammay-Legion-5-16IRX9:~$ cd robot_ws
jammay@jammay-Legion-5-16IRX9:~/robot_ws$ ros2 run chatter_cli listener
[INFO]: Subscribed: 'hi'
[INFO]: Count: '1'
jammay@jammay-Legion-5-16IRX9: ~/robot_ws 39
jammay@jammay-Legion-5-16IRX9:~/robot_ws$ rqt
Default - rqt
File Plugins Running Perspectives Help
Node Graph
Nodes/Topics (active)
Group: 2 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
Topic
topic
/chatter_count
/talker
/listener
/chatter_cli