ROS2_day1_HW3

20기 인턴 강은구

목차

1 기본 설명

1.1 전체 흐름

```cpp
int main(int argc, char **argv)
{
    rclcpp::init(argc, argv);
    auto node  = std::make_shared<publisherNode>();
    auto shape = std::make_shared<make_shape>(node);

    node->shape = shape;
    rclcpp::spin(node);
    rclcpp::shutdown();
    return 0;
}
```

- 해당 프로그램의 흐름을 살펴보자면, 처음 시작은 main함수에서 실행된다. main함수에서 모든 노드를 초기화하고, 발행자 역할을 하는 publisherNode와 도형 형성을 담당하는 make_shpae의 객체를 생성한다. 이후 publisehrNode의 shape멤버에 make_shape객체를 연결해서 두 클래스간에 변수와 함수를 공유할 수있도록 하였다. 실행이 시작되면 주기적으로 키보드 입력 상태를 확인해서 특정 도형을 형성한다. 이때, 도형 그리기 함수에서는 get_xz함수를 통해 publisher의 변수를 변경해주고, publisher내에서는 주기적으로 x, z에 저장된 값을 Twist메시지로 발행하여 turtle1/cmd_vel게 전달한다

2 코드설명

include > eungoo > mytopic.hpp > publisherNode

```cpp
#include "rclcpp/rclcpp.hpp"
#include "std_msgs/msg/string.hpp"
#include "geometry_msgs/msg/twist.hpp"

class publisherNode;

class make_shape : public rclcpp::Node{
private:
    rclcpp::TimerBase::SharedPtr timer_;
    std::shared_ptr<publisherNode> pubNode;
    int count=0;
public:
    make_shape(std::shared_ptr<publisherNode> pub);
    void make_triangle();
    void make_square();
    void make_star();
    void make_circle();
    void count_up();
    make_shape();
};
class publisherNode : public rclcpp::Node
{
private:
    rclcpp::TimerBase::SharedPtr timer1_,timer2_;
    void timer_callback();
    void get_keyboard();
    rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr mycpp_publisher_;
public:
    char inputkey='\0';
    char same='\0';
    float x_val=0;
    float z_val=0;
    publisherNode();
    void get_xz(int get_x, int get_z);
    std::shared_ptr<class make_shape> shape;
};


class subscriberNode : public rclcpp::Node
{

private:
    void topic_callback(const std_msgs::msg::String::SharedPtr msg);
    rclcpp::Subscription<std_msgs::msg::String>::SharedPtr mycpp_subscriber_;
public:
    subscriberNode();
};
```

2.1 hpp파일: 발행자, 도형 형성함수 클래스 및 내부의 함수들을 선언하였다.

2.2 헤더파일 설명

- #include “rclcpp/rclcpp.hpp”: ros2 c++을 사용할 때 호출하는 기본 헤더인다.
- #getmetry_msgs/msg/twist.hpp: 거북이의 선형 석도와 회전 속도를 전달하는 과정에서 지정된 메시지 타입을 사용하기 위한 헤더이다.
- Std_msgs/msg/string.hpp: 문자열 메시지를 위한 타입이다.

2.3 주요 함수 설명

```
void publisherNode::get_keyboard(){
    scanf("%c",&inputkey);
    if(inputkey == 'w') {
        shape->make_triangle();    // shape 객체를 접근할 수 있어야 함
    }
```

- publisherNode::get_keyboad():
  키보드 입력을 통핸 키에 대응하는 도형을 생성하는 함수를 호출한다. 예를 들어, w를 입력하면 삼각형으 ㄹ그리는 make_triangle()함수가 실해오딘다. Scanf를 통해 문자를 입력 받아 변수에 저장한다. 타이머를 통해 주기적으로 키보드 값을 입력 받는다.

```
void publisherNode::get_xz(int get_x, int get_z){
    x_val=get_x;
    z_val=get_z;
}
```

- publisherNode::get_xa(int get_x, int get z):
  make_shape클래스에서 전달된 선형, 회전 속도 값을 내부에 위치한 변수 x_val, z_val에 저장한다. 메시지 타입에서 지정된 여섯개의 값 중 직접 사용하는 linear.x와 angular.z만 수정할 수 있도록 변수는 두개만 선언하였다.

```
void publisherNode::timer_callback(){
    auto msg=geometry_msgs::msg::Twist();
    msg.linear.x = x_val;
    msg.angular.z = z_val;
    mycpp_publisher_->publish(msg);
}
```

- publisherNode::timer_callback():
  일정 시간마다 실행되는 콜백 함수로, Twist메시지를 생성하여 현재 x_val과 z_val을 반영한 메시지를 발행한다. 메시지는 퍼블리셔를 통해 /turtle1/cmd_vel토픽으로 발행되어 거북이를 움직인다.

```
69  void make_shape::count_up(){
70      count++;
71  }
```

- make_shape::count_up():
  주기적으로 실행되는 타이머증가 함수로, count변수를 1씩 증가시킨다. 이때 카운트변수는 도형그리기 함수 내에서 동잒이 실행되는 시간으로 사용된다. 이를 통해거북이는 정해진 시간동안 이동 및 회전을 한다.

src > make_shape.cpp > make_shape(std::shared_ptr<publisherNode>)

```cpp
#include "rclcpp/rclcpp.hpp"
#include "eungoo/mytopic.hpp"
#include <chrono>
#include "geometry_msgs/msg/twist.hpp"

using namespace std::chrono_literals;


make_shape::make_shape(std::shared_ptr<publisherNode> pub): Node("shapenode"), pubNode(pub) {
    timer_=this->create_wall_timer(100ms,std::bind(&make_shape::count_up,this));
    count = 0;
}

void make_shape::make_triangle(){
    if(count<6)pubNode->get_xz(0,1);
    if(count>6&&count<21)pubNode->get_xz(1,0);
    if(count>21&&count<36)pubNode->get_xz(0,1);
    if(count>36&&count<51)pubNode->get_xz(1,0);
    if(count>51&&count<57)pubNode->get_xz(0,1);
    if(count>57&&count<72)pubNode->get_xz(1,0);
    pubNode->get_xz(0,0);
    count=0;
}
void make_shape::make_square(){
    if(count < 10) pubNode->get_xz(1, 0);        // 전진 (1초)
    else if(count < 25) pubNode->get_xz(0, 1);   // 90도 회전 (1.5초)
    else if(count < 35) pubNode->get_xz(1, 0);   // 전진
    else if(count < 50) pubNode->get_xz(0, 1);   // 90도 회전
    else if(count < 60) pubNode->get_xz(1, 0);   // 전진
    else if(count < 75) pubNode->get_xz(0, 1);   // 90도 회전
    else if(count < 85) pubNode->get_xz(1, 0);   // 전진
    else if(count < 100) pubNode->get_xz(0, 1);  // 90도 회전
    else {
        pubNode->get_xz(0, 0);
        count = 0;
    }
}
```

```cpp
void make_shape::make_square(){
    if(count < 10) pubNode->get_xz(1, 0);        // 전진 (1초)
    else if(count < 25) pubNode->get_xz(0, 1);   // 90도 회전 (1.5초)
    else if(count < 35) pubNode->get_xz(1, 0);   // 전진
    else if(count < 50) pubNode->get_xz(0, 1);   // 90도 회전
    else if(count < 60) pubNode->get_xz(1, 0);   // 전진
    else if(count < 75) pubNode->get_xz(0, 1);   // 90도 회전
    else if(count < 85) pubNode->get_xz(1, 0);   // 전진
    else if(count < 100) pubNode->get_xz(0, 1);  // 90도 회전
    else {
        pubNode->get_xz(0, 0);
        count = 0;
    }
}


void make_shape::make_star(){
    if(count < 10) pubNode->get_xz(1, 0);        // 전진
    else if(count < 35) pubNode->get_xz(0, 1);   // 144도 회전 (~2.5초)
    else if(count < 45) pubNode->get_xz(1, 0);   // 전진
    else if(count < 70) pubNode->get_xz(0, 1);   // 144도 회전
    else if(count < 80) pubNode->get_xz(1, 0);   // 전진
    else if(count < 105) pubNode->get_xz(0, 1);  // 144도 회전
    else if(count < 115) pubNode->get_xz(1, 0);  // 전진
    else if(count < 140) pubNode->get_xz(0, 1);  // 144도 회전
    else if(count < 150) pubNode->get_xz(1, 0);  // 전진
    else if(count < 175) pubNode->get_xz(0, 1);  // 144도 회전
    else {
        pubNode->get_xz(0, 0);
        count = 0;
    }
}
void make_shape::make_circle(){
    if(count < 100) {   // 원하는 반복 횟수 (100번 정도 돌리면 원에 가까움)
        if(count % 2 == 1) {
            pubNode->get_xz(1, 0);   // 홀수: 전진
        } else {
            pubNode->get_xz(0, 1);   // 짝수: 회전
        }
    } else {
        pubNode->get_xz(0, 0);   // 멈춤
        count = 0;               // 초기화
    }
}
```

- make_shape::triangle, square, star, circle():
  각각 삼각형, 사각형, 별, 원을 그리는 함수이다. 회전 속도를 고려하여 각 지점에서 회전 시간을 지정해주었고, 이동 시간 도한 속도를 고려하여 시간을 지정해주었다. 원의 경우 완전한 원을 구현하지 못해 0.1초라는 기본 단위의 시간동안 회전 및 이동을 반복하도록 하였다.

2.4 turtlesim관련 함수

- create_publisher<>():
  rclcpp::Node 클래스의 멤버 함수로, 특정 메시지 타입과 토픽 이름ㅇ르 지정하여 퍼블리셔를 생성한다.
- 템플릿 인자<geometry_msgs::msg::Twist>:
  발행할 메시지의 타입을 지정한다. Twist는 선형 속도와 각속도를 저장하는 메시지 구조체
- 첫 번째 인자"/turtle1/cmd_vel":

turtlesim이 구독하는 토픽 이름으로, 여기에 메시지를 발행하면 거북이가 이를 따라 움직인다.

- 두 번째 인자 10:
  Qos(큐 사이즈)를 의미한다. 메시지가 빠르게 발행될 때 버퍼에 10개까지 저장 가능하다.