Qt_3일차_보고서_20기_인턴_2025402040_컴퓨터정보공학부_노기문

-hw_01 : 로봇 팔 만들기 -

<목차>

<문제 분석>

과제 1

- 로봇 팔 만들기

원하는 각도에 맞게 관절을 수동 조절 가능하고,
자동으로 시계방향 또는 반시계방향으로 각 관절을 돌릴 수 있는 로봇 팔 UI 제작
필수 기능 및 조건)
각 관절 각도 수동 조정 기능
각관절 자동 반시계/시계방향 회전 기능
현재 로봇 팔의 상태(각 관절의 각도, 회전방향 등)를 txt 파일로 저장하는 기능
저장한 로봇 팔의 상태를 불러오는 기능
Class를 활용할 것

위 다섯 조건 외에 다른 건 모두 자유.
(기능의 구현 방식 등//원하면 다른 기능을 추가해도 상관x)
단, 사용자 편의성을 고려하여 만들 것

문제요지

과제 1은 로봇 팔 시뮬레이션 UI 프로그램을 만드는 것이다. 사용자가 원하는 각도에 맞춰 관절을 조정할 수 있고, 자동 회전 기능과 상태 저장·불러오기 기능을 포함해야 한다. 반드시 클래스를 활용한 객체지향적 구현을 요구한다.

필수 조건

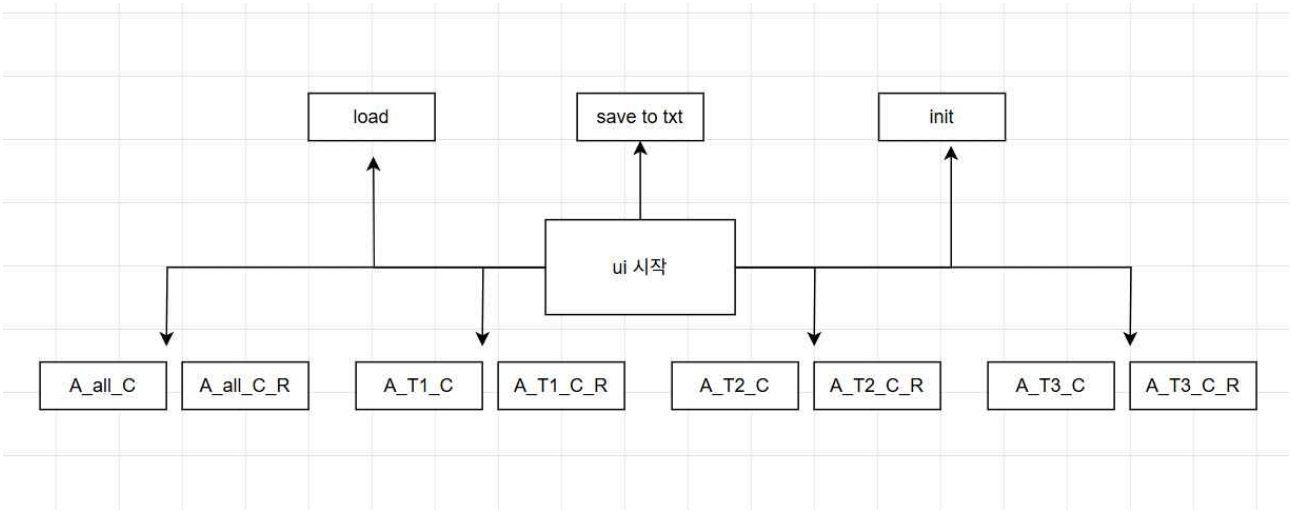
1. 각 관절 수동 조정 기능 : 사용자가 조절한 각도로 특정 관절을 직접 움직일 수 있어야 함.
2. 자동 회전 기능 : 관절이 시계/반시계 방향으로 자동 회전하는 모드 제공 / 일정 속도로 각도를 계속 변화시키는 기능.
3. 현재 상태 저장 : 모든 관절의 각도, 회전 방향 등 상태를 txt 파일에 저장.
4. 상태 불러오기 : 저장된 txt 파일을 읽어서 동일한 상태로 로봇 팔 UI 갱신.
5. 클래스 사용 : 로봇 팔과 관절을 객체로 정의 확장성 있게 구조 설계.

핵심 요약

로봇 팔 UI 제작 (관절 조작 가능)
각도 수동/자동 조정
상태 저장/불러오기(txt)
Class 기반 구현

즉, 이 과제는 객체지향 설계 능력 + 파일 입출력 + UI 프로그래밍 + 로봇 시뮬레이션 논리를 종합적으로 묻는 문제다.

<알고리즘설명>



요지: 3자유도 로봇팔을 QGraphicsScene 위에 계층(parent-child)으로 만들고, 슬라이더·타이머·버튼으로 각 관절 회전(수동/자동), 상태 저장·불러오기를 수행한다.

핵심 흐름

1. 초기 설정
   scene 생성 후 graphicsView에 연결. 안티앨리어싱 켬, 장면 사각형 설정.
   buildArm() 호출로 링크와 조인트 도형 생성.
   각 슬라이더 범위를 0~360으로 설정.
   setAngles(90,90,90)로 초기 자세.
   timer 간격 30ms, timeout → tick() 연결.

2. 로봇팔 구성 (buildArm)
   링크1: QGraphicsRectItem(0,-5,L1,10)을 원점(0,0)에 두고 setTransformOriginPoint(0,0)로 회전 중심을 링크의 왼쪽 끝으로 지정.
   링크2: 부모를 link1로 설정하고 setPos(L1,0)로 링크1 끝에 부착. 회전 중심은 링크2의 좌측 끝.
   링크3: 부모를 link2로 설정하고 setPos(L2,0)로 링크2 끝에 부착.
   부모-자식 변환 체인 때문에 상위 링크가 회전하면 하위 링크도 함께 따라간다.
   조인트는 작은 원(QGraphicsEllipseItem)으로 각 링크의 회전점을 시각화.

3. 각도 적용 (setAngles)
   setRotation()으로 각 링크의 로컬 회전각을 지정.
   clamp360은 qBound(0,360)로 범위를 제한.
   뷰를 원점으로 센터링.

4. 수동 제어(슬라이더 슬롯)
   각 슬롯에서 자신이 아닌 나머지 링크의 현재 회전값을 읽고, 바뀐 값만 교체하여 setAngles

호출.
전체 슬라이더(on_all_toque_slide_valueChanged)는 세 슬라이더 값을 동시에 갱신하므로 blockSignals(true)로 재귀 신호를 잠깐 막고 값만 반영한 뒤 해제.

5. 자동 회전 상태 관리

dir[3] 배열: 각 관절의 회전 방향. +1 시계, -1 반시계, 0 정지.
toggleDir(idx,want): 같은 버튼을 다시 누르면 토글로 0(정지), 아니면 want로 설정.
updateTimer(): 세 항목이 모두 0이면 타이머 정지, 하나라도 움직이면 시작.

6.타이머 틱

현재 각도 읽기 → dir[i]*step만큼 증감.
내부 wrap360: 0..359로 모듈로 래핑(음수 보정 포함).
setAngles로 그래픽 갱신.
슬라이더와 값 동기화. 이때도 blockSignals로 재귀 방지.
마지막에 updateTimer()로 필요 시 자동 정지.

7. 전체/개별 자동 회전 버튼

개별: on_A_Tx_C/_C_R에서 toggleDir(x, +1/-1) 호출.
전체: 모두가 같은 방향이면 다시 누를 때 정지, 아니면 일괄 +1 또는 -1로 설정.

8. 초기화/저장/불러오기

on_init_clicked: 세 슬라이더와 링크 각도를 90°로 맞추고 dir을 0으로 초기화.
on_save_to_txt_clicked: 현재 세 관절 각도를 arm_state.txt에 줄 단위로 기록.
on_load_prev_clicked: 파일에서 세 값을 읽고 clamp360 후 슬라이더·링크에 반영.

주의 포인트

장면 좌표계는 링크 로컬 좌표 + 부모 변환이 합성됨. 회전 중심이 정확히 왼쪽 끝으로 지정되어야 자연스러운 관절이 됨.
슬라이더 신호 차단이 없으면 값 세팅 → 슬롯 재진입 → 무한 루프 가능.
clamp360은 360도 값을 허용하지만 wrap360은 0..359로 정규화. 일관성을 원하면 둘 중 하나로 통일 가능.
파일 포맷은 매우 단순(각도 3줄). 확장 시 속도나 방향도 함께 저장하도록 스키마를 늘리면 된다.

<코드설명>

1. mainwindow.h

```
#ifndefMAINWINDOW_H
#defineMAINWINDOW_H
#include <QMainWindow>
#include <QTimer>
#include <QGraphicsScene>
#include <QGraphicsRectItem>
#include <QGraphicsEllipseItem>
QT_BEGIN_NAMESPACE
namespace Uiclass MainWindow; }
QT_END_NAMESPACE
```

include guard: 중복 include 방지.

Qt include: QMainWindow는 GUI 창, QTimer는 주기적 실행, QGraphicsScene과 item 클래스는 로봇 팔 그리기에 사용.

Ui 네임스페이스: Qt Designer .ui 파일과 연결되는 자동 생성 클래스.

```
class MainWindowpublic QMainWindow{
Q_OBJECT
public:
    explicit MainWindow(QWidget *parent=nullptr);
    ~MainWindow();
```

Q_OBJECT: Qt의 시그널/슬롯 매커니즘 활성화 매크로.

생성자/소멸자: UI 초기화와 자원 해제를 담당.

```
void on_toque1_slide_valueChanged(int);
    void on_toque2_slide_valueChanged(int);
    void on_toque3_slide_valueChanged(int);
    void on_all_toque_slide_valueChanged(int);
// 버튼(자동 회전 토글)
    void on_A_T1_C_clicked();    void on_A_T1_C_R_clicked();
    void on_A_T2_C_clicked();    void on_A_T2_C_R_clicked();
    void on_A_T3_C_clicked();    void on_A_T3_C_R_clicked();
    void on_A_all_C_clicked();  void on_A_all_C_R_clicked();
// 초기화/저장/불러오기
    void on_init_clicked();
    void on_save_to_txt_clicked();
    void on_load_prev_clicked();
```

슬라이더: 각 관절 토크(=각도) 조절.

자동 회전 버튼: 특정 관절 또는 전체 회전을 CW/CCW로 토글.

기능 버튼: 초기화, 현재 상태 저장, 이전 상태 불러오기.

```
Ui::MainWindow *ui;
// 그래픽
QGraphicsScene *scene =nullptr;
    const doubleL1=120, L2=90, L3=70;
QGraphicsRectItem *link1=nullptr,*link2=nullptr,*link3=nullptr;
QGraphicsEllipseItem *j1=nullptr,*j2=nullptr,*j3=nullptr;
```

UI 객체 포인터. Qt Designer에서 만든 위젯 접근용.

scene: 로봇팔 전체를 그릴 그래픽 공간.

L1,L2,L3: 링크 길이.

link1~3: 사각형으로 표현된 팔의 각 링크.

j1~3: 원형으로 표현된 각 관절.

```
// 자동 회전
QTimer timer;
    int dir[3]={0,0,0};// -1 CCW, 0 stop, +1 CW
    intstep=1;// 틱당 변화량(도)
```

timer: 일정 주기로 tick() 실행.

dir: 각 관절 회전 방향.

step: 회전 속도(도 단위).

```
void buildArm();
    void setAngles(int a1,int a2,int a3);// 0~360
    void tick();
    void toggleDir(int idx,int want);
    void updateTimer();
```

buildArm(): 링크와 관절을 그래픽 객체로 생성 및 배치.

setAngles(): 세 관절 각도 적용.

tick(): QTimer가 호출하는 주기적 업데이트. 각도 변경 후 다시 그림.

toggleDir(): 특정 관절의 회전 방향 전환.

updateTimer(): 활성화된 회전이 있으면 타이머 동작, 없으면 정지.

2. mainwindow.cpp

```
MainWindow::MainWindow(QWidget *parent)
: QMainWindow(parent), ui(new Ui::MainWindow) {
    ui->setupUi(this);
    setWindowTitle("3-DOF Robot Arm");
// .ui: QGraphicsView의 objectName은 반드시 "graphicsView"
scene =new QGraphicsScene(this);
    ui->graphicsView->setScene(scene);
    ui->graphicsView->setRenderHint(QPainter::Antialiasing, true);
    scene->setSceneRect(-300,-300,600,600);
    ui->graphicsView->centerOn(0,0);
    buildArm();
// 슬라이더 범위(0~360)
    if(ui->toque1_slide) ui->toque1_slide->setRange(0,360);
    if(ui->toque2_slide) ui->toque2_slide->setRange(0,360);
    if(ui->toque3_slide) ui->toque3_slide->setRange(0,360);
    if(ui->all_toque_slide) ui->all_toque_slide->setRange(0,360);
// 초기값
    setAngles(90,90,90);
    if(ui->toque1_slide) ui->toque1_slide->setValue(90);
    if(ui->toque2_slide) ui->toque2_slide->setValue(90);
    if(ui->toque3_slide) ui->toque3_slide->setValue(90);
    if(ui->all_toque_slide) ui->all_toque_slide->setValue(90);
// 타이머
    timer.setInterval(30);
    connect(&timer,&QTimer::timeout,this,&MainWindow::tick);
```

```
}
```
실행하면 graphicsView 안에 로봇팔이 보이고, 세 슬라이더 값이 전부 90도로 맞춰져 시작한다.

```
void MainWindow::buildArm() {
QPen pen(Qt::black,2); QBrush brush(Qt::lightGray);
// 링크1: 원점 기준, 회전 중심(0,0)
link1 =new QGraphicsRectItem(0,-5,L1,10);
    link1->setPen(pen); link1->setBrush(brush);
    link1->setTransformOriginPoint(0,0);
    link1->setPos(0,0);
    scene->addItem(link1);
// 링크2: 링크1의 자식, 링크1 끝에 부착
link2 =new QGraphicsRectItem(0,-5,L2,10, link1);
    link2->setPen(pen); link2->setBrush(brush);
    link2->setTransformOriginPoint(0,0);
    link2->setPos(L1,0);
// 링크3: 링크2의 자식, 링크2 끝에 부착
link3 =new QGraphicsRectItem(0,-5,L3,10, link2);
    link3->setPen(pen); link3->setBrush(brush);
    link3->setTransformOriginPoint(0,0);
    link3->setPos(L2,0);
// 조인트 원
    const doubler=6;
=new      QGraphicsEllipseItem(-r,-r,2*r,2*r,link1);      j1->setBrush(Qt::darkGray);
j1->setPos(0,0);
=new      QGraphicsEllipseItem(-r,-r,2*r,2*r,link2);      j2->setBrush(Qt::darkGray);
j2->setPos(0,0);
=new      QGraphicsEllipseItem(-r,-r,2*r,2*r,link3);      j3->setBrush(Qt::darkGray);
j3->setPos(0,0);
}
```
이렇게 부모-자식 구조를 만들면, link1을 회전시키면 그 밑의 link2, link3도 함께 회전한다. 실제 로봇팔처럼 동작.

```
void MainWindow::on_toque1_slide_valueChanged(int v) {
    inta2 =link2 ?int(link2->rotation()) :0;
    inta3 =link3 ?int(link3->rotation()) :0;
    setAngles(v, a2, a3);
}
```
첫 번째 슬라이더를 움직이면 link1의 각도만 바뀌고, 나머지 링크 각도는 그대로 유지된다.
예: 슬라이더1을 45도로 옮기면 link1=45°, link2=90°, link3=90°로 됨. 나머지 toque2,3도 같은 로직으로 동작

```
void MainWindow::on_A_T1_C_clicked()   { toggleDir(0, +1); }
```

“관절1 시계방향” 버튼을 누르면 dir[0]=+1.
tick()이 30ms마다 호출되어 각도가 1도씩 계속 증가.
예: 누르면 link1=91→92→93→… 식으로 자동 회전. 다시 누르면 정지. 나머지 관절 2,3,전체 도 동일 로직.

```
void MainWindow::tick() {
    autowrap360 =[](int x){
```

```
// 0..360로 래핑
x %=360; if(x <0) x +=360;
        returnx;
    inta1 =link1 ?int(link1->rotation()) :0;
    inta2 =link2 ?int(link2->rotation()) :0;
    inta3 =link3 ?int(link3->rotation()) :0;
=wrap360(a1 +dir[0]*step);
=wrap360(a2 +dir[1]*step);
=wrap360(a3 +dir[2]*step);
// 그래픽 갱신
    setAngles(a1, a2, a3);
// 슬라이더 동기화(신호 중첩 방지)
    boolb1 =ui->toque1_slide->blockSignals(true);
    boolb2 =ui->toque2_slide->blockSignals(true);
    boolb3 =ui->toque3_slide->blockSignals(true);
    if(ui->toque1_slide) ui->toque1_slide->setValue(a1);
    if(ui->toque2_slide) ui->toque2_slide->setValue(a2);
    if(ui->toque3_slide) ui->toque3_slide->setValue(a3);
    ui->toque1_slide->blockSignals(b1);
    ui->toque2_slide->blockSignals(b2);
    ui->toque3_slide->blockSignals(b3);
    updateTimer();// 모두 멈췄으면 타이머 자동 정지
}
```

dir[0]=+1일 때 30ms마다 a1에 +1. 결과적으로 초당 33도 정도 회전.

슬라이더 값도 같이 업데이트되어 화면과 슬라이더가 동기화됨.

```
void MainWindow::on_save_to_txt_clicked() {
QFile f("arm_state.txt");
    if(!f.open(QIODevice::WriteOnly |QIODevice::Text)) return;
QTextStream out(&f);
    const inta1 =link1 ?int(link1->rotation()) :0;
    const inta2 =link2 ?int(link2->rotation()) :0;
    const inta3 =link3 ?int(link3->rotation()) :0;
out <<a1 <<'\n'<<a2 <<'\n'<<a3 <<'\n';
}
```

저장 버튼을 누르면 현재 각도가 arm_state.txt에 기록된다.

```
void MainWindow::on_load_prev_clicked() {
QFile f("arm_state.txt");
    if(!f.open(QIODevice::ReadOnly |QIODevice::Text)) return;
QTextStream in(&f);
    inta1=90, a2=90, a3=90;
>>a1 >>a2 >>a3;
// 슬라이더 동기화
    boolb1 =ui->toque1_slide->blockSignals(true);
    boolb2 =ui->toque2_slide->blockSignals(true);
    boolb3 =ui->toque3_slide->blockSignals(true);
    if(ui->toque1_slide) ui->toque1_slide->setValue(clamp360(a1));
    if(ui->toque2_slide) ui->toque2_slide->setValue(clamp360(a2));
    if(ui->toque3_slide) ui->toque3_slide->setValue(clamp360(a3));
    ui->toque1_slide->blockSignals(b1);
    ui->toque2_slide->blockSignals(b2);
    ui->toque3_slide->blockSignals(b3);
    setAngles(clamp360(a1), clamp360(a2), clamp360(a3));
```

```
}
```

불러오기 버튼을 누르면 파일 값을 읽어와 팔과 슬라이더를 동일하게 맞춘다.

```
void MainWindow::on_init_clicked() {
    const intinitAng =90;
// 슬라이더 먼저 동기화
    boolb1 =ui->toque1_slide->blockSignals(true);
    boolb2 =ui->toque2_slide->blockSignals(true);
    boolb3 =ui->toque3_slide->blockSignals(true);
    if(ui->toque1_slide) ui->toque1_slide->setValue(initAng);
    if(ui->toque2_slide) ui->toque2_slide->setValue(initAng);
    if(ui->toque3_slide) ui->toque3_slide->setValue(initAng);
    if(ui->all_toque_slide) ui->all_toque_slide->setValue(initAng);
    ui->toque1_slide->blockSignals(b1);
    ui->toque2_slide->blockSignals(b2);
    ui->toque3_slide->blockSignals(b3);
    dir[0]=dir[1]=dir[2]=0;
    setAngles(initAng, initAng, initAng);
    updateTimer();
}
```

모든 관절이 다시 90도로 돌아가고 자동 회전도 멈춘다.

<result screen>

영상 참조 바람.

-끝-