# Qt_3일차_보고서_20기_인턴_2025402040_컴퓨터정보공학부_노기문

## -hw_02 : 큐,스택기능 활용한 UI제작 -

<목차>

## <문제 분석>

# 과제 2

- Queue 또는 stack 기능을 활용한 UI 제작

Queue : FIFO(First In First Out), 제일 처음 넣은 데이터가 처음으로 빠져나오는 구조
Stack : LIFO(Last In First Out), 제일 마지막에 넣은 데이터가 처음으로 빠져나오는 구조

벡터를 활용하여 구현할 것

문제 요지

Queue(큐) 또는 Stack(스택)의 동작 원리를 직접 구현하고, 이를 활용한 간단한 UI 프로그램을 작성하는 과제이다.
Queue는 FIFO(First In First Out) 구조로, 가장 먼저 넣은 데이터가 가장 먼저 빠져나온다.
Stack은 LIFO(Last In First Out) 구조로, 가장 마지막에 넣은 데이터가 가장 먼저 빠져나온다.
즉, 두 자료구조의 기본 동작 차이를 이해하고, 이를 시각적으로 확인할 수 있는 UI를 제작하는 것이 핵심이다.

필수조건

벡터를 활용하여 직접 Queue 또는 Stack을 구현해야 한다.
단순한 콘솔 출력이 아니라 UI 형태로 동작이 보이도록 만들어야 한다.
Queue/Stack 중 하나만 구현해도 되지만, 가능하면 두 개 다 구현하면 더 확실하다.

핵심요약

Queue의 push, pop 동작 → 입력 순서대로 출력되는지 확인.
Stack의 push, pop 동작 → 마지막에 넣은 것이 먼저 빠져나오는지 확인.
벡터의 push_back(), erase(), back(), front() 등을 적절히 활용해야 함.
UI 상에서 데이터가 들어가고 빠져나오는 과정을 시각적으로 보여주는 것이 중요.
단순히 자료구조를 쓰는 것이 아니라, 직접 vector 기반으로 구현하는 것이 조건임.

# <알고리즘설명>

1. 자료구조와 기본 동작

각 병(BottleItem)은 QGraphicsRectItem을 상속받아 직사각형으로 그려진다.
병 안에는 최대 3개의 공(CAP=3)이 들어갈 수 있고, 공은 QColor로 표현된다.
push()는 병에 공을 하나 넣고 다시 그림을 그린다.
pop()은 병에서 마지막(맨 위) 공을 꺼내고 다시 그림을 그린다.
isUniformFull()은 병이 가득 찼을 때(3개) 모두 같은 색인지 확인한다.
즉, 각 병은 Stack (LIFO) 구조로 동작한다.

2. 화면 초기화

MainWindow에서 QGraphicsScene과 QGraphicsView를 준비해 그래픽을 표시한다.
setupScene()은 병 3개를 가로로 배치한다.
initLevel()은
첫 번째 병에 (빨-파-빨)
두 번째 병에 (파-파-빨)
세 번째 병은 비워둔다.
상태표시줄(statusBar)에는 "소스 병을 클릭하세요"라는 안내 메시지를 띄운다.

3. 클릭 이벤트 흐름

사용자가 병 하나를 클릭하면 onBottleClicked()가 호출된다.
첫 클릭은 소스 병 선택이다. → 병 테두리를 초록색으로 바꾸고 "타깃 병을 클릭하세요"라고 안내한다.
같은 병을 다시 클릭하면 선택이 해제된다.
두 번째 클릭은 타깃 병 선택이다.
소스 병이 비었으면 이동 불가.
타깃 병이 가득 차 있으면 이동 불가.
조건을 만족하면 소스 병에서 pop()으로 맨 위 공을 꺼내고 타깃 병에 push()로 넣는다.
이동이 끝나면 선택 상태를 초기화하고 "소스 병을 클릭하세요" 안내로 돌아간다.

4. 승리 조건 검사

checkWin()은 각 병이 isUniformFull()인지 검사한다.
세 병 중에서 두 개 이상이 같은 색으로 3개 꽉 찬 상태가 되면 승리.
승리 메시지를 띄우고 initLevel()로 다시 초기화한다.
핵심 알고리즘 요약
병을 Stack처럼 구현 (push, pop).
소스 병 선택 → 타깃 병 선택 → 공 하나 이동.
매 이동 후 두 병 이상이 동일 색으로 가득 차면 승리.

# <코드설명>

## 1. mainwindows.hpp

```
class BottleItempublic QGraphicsRectItem{
public:
    static constexpr intCAP =3;
    BottleItem(MainWindow*owner, int id, const QRectF&r);
    int id() constreturnid_; }
    bool canPush() constreturn balls_.size() <CAP; }
    bool canPop() constreturn !balls_.isEmpty(); }
    bool isUniformFull() const;
    bool push(const QColor&c);
    bool pop(QColor&out);
    void redraw();
protected:
    void mousePressEvent(QGraphicsSceneMouseEvent*ev) override;
private:
MainWindow*owner_;
    intid_;
QVector<QColor>balls_;// back == top
};
```

CAP = 3: 병 용량. balls_.size() < 3일 때만 push 가능.

QVector<QColor> balls_: 공 스택. “뒤가 꼭대기(top)”이므로 push_back이 최상단 공을 의미.

canPush()/canPop(): 삽입/추출 가능 여부 빠른 체크.

isUniformFull():

의도: 공이 3개 꽉 찼고 모두 같은 색인지 판단. 퍼즐의 “완성된 병” 조건에 사용.

예: balls_ == {red, red, red} → true.

push(const QColor& c):

용량 확인 후 색 c를 최상단에 추가. 성공 시 true.

예: 비어있는 병에 push(Qt::blue) → balls_={blue}.

pop(QColor& out):

비지 않았을 때 최상단 색을 out으로 꺼내고 제거. 성공 시 true.

예: balls_={blue, red} → pop → out=red, balls_={blue}.

redraw():

병 사각형과 내부 원(공)들을 다시 그림. 최상단이 위쪽에 보이도록 y좌표를 계산해 0~2칸에 배치.

mousePressEvent(QGraphicsSceneMouseEvent*):

클릭되면 owner_의 onBottleClicked(this) 호출로 메인 창에 알림.

```
class MainWindowpublic QMainWindow{
Q_OBJECT
public:
    explicit MainWindow(QWidget*parent=nullptr);
    void onBottleClicked(BottleItem*b);
private:
    void setupScene();
    void initLevel();
    void checkWin();
QGraphicsView*view_;
QGraphicsScene*scene_;
BottleItem*bottles_[3]{nullptr,nullptr,nullptr};
BottleItem*selected_{nullptr};
```

```
};
```

멤버:

view_, scene_: 2D 장면/보기 구성.

bottles_[3]: 3개의 병을 소유.

selected_: 선택된 출발 병 포인터. 없으면 nullptr.

MainWindow(QWidget*):

setupScene()에서 scene_ 생성, 병 3개를 적절한 좌표에 만들고 view_에 붙임.

initLevel()에서 시작 배치를 정함. 예: 병0에 {빨, 파, 빨}, 병1에 {파, 파, 빨}, 병2 비움.

onBottleClicked(BottleItem* b) 동작 예시:

1. 아직 선택 없음

selected_==nullptr이고 b->canPop()이면 출발 병으로 선택. 시각적으로 테두리 강조 후 b->redraw().

2. 이미 하나 선택됨

다시 같은 병을 누르면 선택 해제.

다른 병 b2를 누르면 이동 시도:

출발 병 selected_에서 pop(color) 시도.

목적 병 b2가 canPush()이고, 규칙이 있다면(예: 최상단 색 같거나 빈 병) 조건 확인.

가능하면 b2->push(color), 두 병 모두 redraw().

이동 불가면 꺼낸 색을 원래 병에 push하여 롤백.

선택 해제 후 checkWin().

checkWin():

모든 병이 isUniformFull()이거나 비어 있는 규칙을 만족하면 승리 처리.

예: 병0= {R,R,R}, 병1= {B,B,B}, 병2= {} → 승리.

## 2. mainwindows.cpp

```
bool BottleItem::push(const QColor&c) {
    if(!canPush()) return false;
    balls_.push_back(c);
    redraw();
    return true;
}
bool BottleItem::pop(QColor&out) {
    if(!canPop()) return false;
out =balls_.back();
    balls_.pop_back();
    redraw();
    return true;
}
```

공이 {빨, 파}(아래→위)일 때 push(빨) → {빨, 파, 빨}. pop(out) → out=빨, {빨, 파}로 복원.

```
bool BottleItem::isUniformFull() const{
    if(balls_.size() !=CAP) return false;
    return balls_[0] ==balls_[1] &&balls_[1] ==balls_[2];
}
```

3개 꽉 차고 모두 같은 색이면 승리 판정에 사용.

```
void BottleItem::redraw() {
// remove previous ball drawings
    const autokids =childItems();
    for(auto*ch : kids) deletech;
    constQRectF r =rect();
    constqreal margin =10;
    constqreal innerW =r.width() -2*margin;
    constqreal ballH  =(r.height() -2*margin) /CAP;
    for(int=0; i <balls_.size(); ++i) {
        intlevelFromBottom =i;// 0 bottom
qreal y =r.bottom() -margin -(levelFromBottom+1)*ballH;
qreal x =r.left() +margin;
        auto*=new QGraphicsEllipseItem(x, y, innerW, ballH-4, this);
        e->setBrush(QBrush(balls_[i]));
        e->setPen(QPen(Qt::black, 1));
        e->setZValue(1);
}
    update();
}
```

병 내부를 CAP=3층으로 등분. i=0은 바닥, i=2는 꼭대기. 각 공을 타원으로 그리고 부모를 병(this)로 설정해 함께 이동/삭제.

```
void BottleItem::mousePressEvent(QGraphicsSceneMouseEvent*ev) {
    QGraphicsRectItem::mousePressEvent(ev);
    if(owner_) owner_->onBottleClicked(this);
}
```

자기 클릭을 메인창에 알림. 병 인스턴스 포인터를 넘겨 어떤 병인지 식별.

```
MainWindow::MainWindow(QWidget*parent)
: QMainWindow(parent),
    view_(new QGraphicsView(this)),
    scene_(new QGraphicsScene(this)) {
    setWindowTitle("Bottle Sort - 3 Bottles");
    resize(540, 360);
    view_->setScene(scene_);
    view_->setRenderHint(QPainter::Antialiasing, true);
    view_->setAlignment(Qt::AlignCenter);
    auto*central =new QWidget(this);
    auto*lay =new QVBoxLayout(central);
    lay->setContentsMargins(0,0,0,0);
    lay->addWidget(view_);
    setCentralWidget(central);
    statusBar()->showMessage("소스 병을 클릭하세요");
    setupScene();
    initLevel();
}
```

생성자 그래픽 장면/뷰 구성. 중앙 위젯에 QGraphicsView 배치. 상태바 메시지 초기화 후 병 배치와 퍼즐 초기화 호출.

```
void MainWindow::setupScene() {
    scene_->setSceneRect(0,0,520,300);
```

```
    constqreal W =120;
    constqreal H =220;
    constqreal gap =40;
    constqreal startX =20;
    constqreal y =40;
    for(inti=0;i<3;++i) {
QRectF r(startX +i*(W+gap), y, W, H);
        bottles_[i] =new BottleItem(this, i, r);
        scene_->addItem(bottles_[i]);
// base decoration under bottle
QRectF base(r.left()-2, r.bottom()+2, r.width()+4, 8);
        scene_->addRect(base, QPen(Qt::NoPen), QBrush(Qt::lightGray));
}
}
```

=> 장면구성

```
void MainWindow::initLevel() {
// clear
    for(auto*b : bottles_) {
        if(!b) continue;
QColor dummy;
        while(b->pop(dummy)) {}
        b->setPen(QPen(Qt::black, 2));
}
// Bottle 0: (빨/파/빨) bottom->top
    bottles_[0]->push(Qt::red);
    bottles_[0]->push(Qt::blue);
    bottles_[0]->push(Qt::red);
// Bottle 1: (파/파/빨) bottom->top
    bottles_[1]->push(Qt::blue);
    bottles_[1]->push(Qt::blue);
    bottles_[1]->push(Qt::red);
// Bottle 2: empty
selected_ =nullptr;
    statusBar()->showMessage("소스 병을 클릭하세요");
}
```

레벨 초기화

```
void MainWindow::onBottleClicked(BottleItem*b) {
    if(!selected_) {
selected_ =b;
        statusBar()->showMessage(QString("타깃 병을 클릭하세요 (선택: %1)").arg(b->id()+1));
        b->setPen(QPen(Qt::darkGreen, 4));
        return;
}
    if(b ==selected_) {
        selected_->setPen(QPen(Qt::black, 2));
selected_ =nullptr;
        statusBar()->showMessage("소스 병을 클릭하세요");
        return;
}
QColor top;
```

```
    if(!selected_->canPop()) {
        selected_->setPen(QPen(Qt::black, 2));
selected_ =nullptr;
        statusBar()->showMessage("빈 병에서 이동 불가. 소스 병을 다시 선택.");
        return;
}
    if(!b->canPush()) {
        selected_->setPen(QPen(Qt::black, 2));
selected_ =nullptr;
        statusBar()->showMessage("타깃 병이 가득 참. 다른 타깃을 선택.");
        return;
}
    selected_->pop(top);
    b->push(top);
    selected_->setPen(QPen(Qt::black, 2));
selected_ =nullptr;
    statusBar()->showMessage("소스 병을 클릭하세요");
    checkWin();
}
```

첫 클릭은 소스 지정. 두 번째 클릭은 타깃 지정. 가득 참/비어 있음 검사 후 이동. 규칙 단순형(색 일치 제한 없음).

```
void MainWindow::checkWin() {
    intuniformFull =0;
    for(auto*b : bottles_) if(b->isUniformFull()) ++uniformFull;
    if(uniformFull >=2) {
        QMessageBox::information(this, "승리", "두 병이 같은 색으로 3개씩 채워졌습니다. 게임 종료.");
        initLevel();
}
}
```

세 병 중 최소 두 병이 “단색 3개”이면 승리로 간주하고 초기 배치로 리셋.

사용 흐름 예시

시작 시 병0=(빨/파/빨), 병1=(파/파/빨), 병2=빈.

병0 클릭 → 소스 선택. 테두리 초록 두껍게.

병2 클릭 → 병0의 top=빨을 pop → 병2에 push. 두 병 redraw()로 즉시 반영.

계속 이동하여 두 병이 {R,R,R}, {B,B,B} 상태가 되면 checkWin()이 메시지 박스를 띄우고 initLevel()로 재설정.

핵심 메커니즘: 클릭으로 소스·타깃를 정해 pop→push→redraw를 일으키고, 각 병의 내부 상태는 QVector<QColor>가 보관하며, 그릴 때마다 자식 타원 아이템을 재생성해 시각 상태를 정확히 동기화한다.

## <result screen>

영상 참조 바람.

-끝-