Qt 프로젝트

Bingo with Canon

로빛 20기 인턴 | 김민조, 강은구

목차

프로그램 개요

물리법칙 적용
- 대포알 포물선 궤적따라 이동

조작 방식
- 키보드로 각도 조절
- 마우스로 파워 조절

빙고 시스템
- 빈 칸에 대포알 맞출 시 색칠

맵
- 3X3 격자판

클래스

그래픽, 물리법칙, 빙고판 등
통합하여 관리

```
private:
    Ui::MainWindow *ui;
    QGraphicsScene *scene;

    QGraphicsEllipseItem *ball;       // 빨간 공 (내 공)
    QGraphicsEllipseItem *enemyBall;  // 초록 공 (상대 공)
    QTimer *timer;        // 내 공 타이머
    QTimer *enemyTimer;   // 상대 공 타이머

    QUdpSocket *udpSocket; // UDP 소켓
    QTimer *powerTimer;    // 파워 충전용 타이머
    bool spacePressed;

    // 내 공 변수
    double angle1, angle2, power;
    double vx, vy, vz;
    double x, y, z;
    double t;

    // 상대 공 변수
    double enemyAngle1, enemyAngle2, enemyPower;
    double evx, evy, evz;
    double ex, ey, ez;
    double et;

    // 환경 변수
    double g;
    double wallZ;

    // 시작 위치
    double startX, startY;     // 빨간 공 시작 위치
    double enemyX, enemyY;     // 초록 공 시작 위치

    // 벽 셀
    QGraphicsRectItem* wallCells[3][3];

    int board[3][3];        // 0=미점유, 1=내(빨강), 2=상대(초록)
    bool gameOver;          // 게임 종료 플래그

    void markCell(int row, int col, int owner);
    bool checkBingo(int owner) const;
    void endGame(int winner); // 1=빨강 승, 2=초록 승
```

물리법칙

중력 가속도 및 벡터 반영

=> 좌표에 따라 입체적으로 표현

```
void MainWindow::updateBall()
{
    x = vx * t;
    y = vy * t - 0.5 * g * t * t;
    z = vz * t;

    if (z >= wallZ) {
        timer->stop();
        if (gameOver) return;

        double scale = wallZ/(z);
        double hitX = startX + x * scale;
        double hitY = startY - y * scale;

        int cellW = 100;
        int cellH = 50;
        int col = (int)((hitX + 150) / cellW);
        int row = (int)((hitY + 150) / cellH);
        if(hitY<-150||hitX<-150)return;
        qDebug() << "[내 공 충돌]"
             << "hitX:" << hitX
             << "hitY:" << hitY
             << "row:" << row
             << "col:" << col;


        if (row >= 0 && row < 3 && col >= 0 && col < 3) {
            markCell(row, col, 1);
            if (board[row][col] == 1) wallCells[row][col]->setBrush(QBrush(Qt::red));
            else if (board[row][col] == 2) wallCells[row][col]->setBrush(QBrush(Qt::green));
        }
        return;
    }

    double c = 7.14, K = 571.4;
    double size = K / (z + c);
    if (size > 80) size = 80;
    if (size < 10) size = 10;

    t += 0.03;

    ball->setRect(-size/2, -size/2, size, size);
    ball->setPos(startX + x * wallZ / (z + 1.0), startY - y * wallZ / (z + 1.0));
}
```

빙고시스템

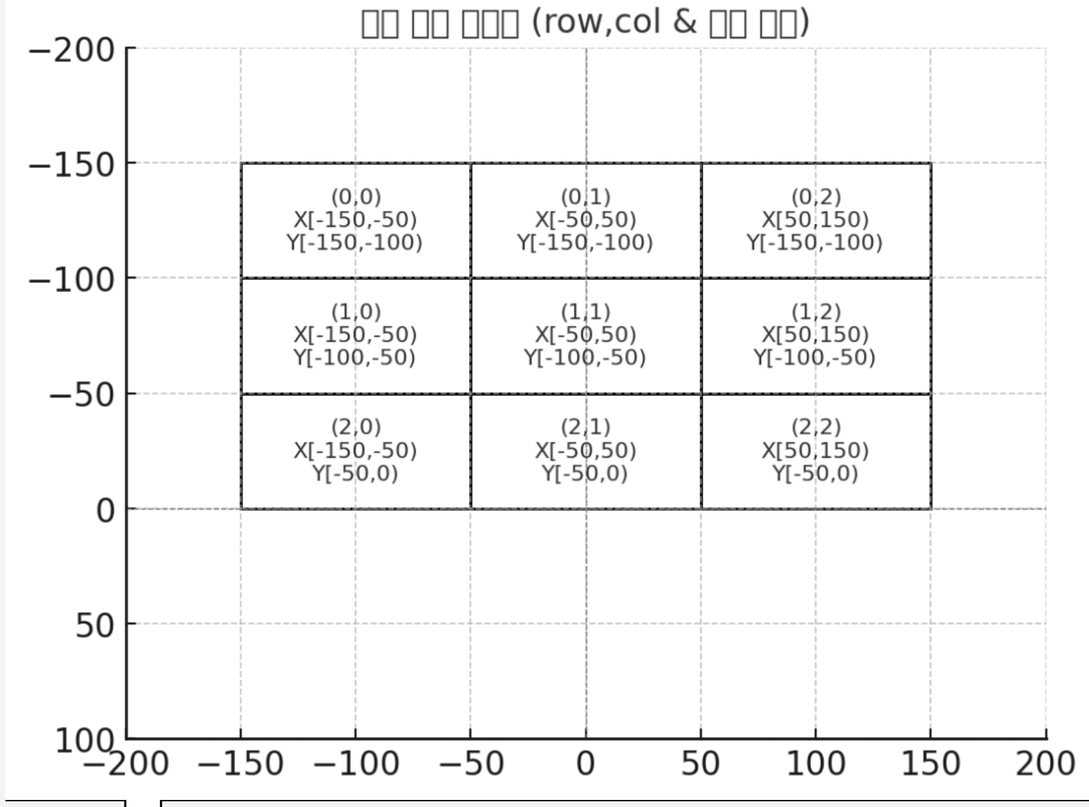
상대가 이미 점유한 칸은 X
한 줄로 이어지는 부분 있는지 체크

```cpp
bool MainWindow::checkBingo(int owner) const
{
    for (int i = 0; i < 3; ++i) {
        if (board[i][0] == owner && board[i][1] == owner && board[i][2] == owner)
            return true;
        if (board[0][i] == owner && board[1][i] == owner && board[2][i] == owner)
            return true;
    }
    if (board[0][0] == owner && board[1][1] == owner && board[2][2] == owner)
        return true;
    if (board[0][2] == owner && board[1][1] == owner && board[2][0] == owner)
        return true;

    return false;
}
```

```cpp
void MainWindow::markCell(int row, int col, int owner)
{
    if (row < 0 || row >= 3 || col < 0 || col >= 3) return;
    if (gameOver) return;

    if (board[row][col] != 0 && board[row][col] != owner) {
        qDebug() << "이미 상대가 점유한 칸:" << row << col;
        return;
    }

    board[row][col] = owner;

    if (checkBingo(owner)) {
        endGame(owner);
    }
}
```

맵

셀 1개: 100x50 사이즈




```
// 벽 3x3
int cellW = 100;
int cellH = 50;
for (int row = 0; row < 3; row++) {
    for (int col = 0; col < 3; col++) {
        int x0 = -150 + col * cellW;
        int y0 = -150 + row * cellH;
        wallCells[row][col] = scene->addRect(
            x0, y0, cellW, cellH,
            QPen(Qt::black), QBrush(Qt::white));
        board[row][col] = 0; // 빙고 보드 초기화
    }
}
```

통신

readPendingDatagrams() 호출

```
udpSocket = new QUdpSocket(this);

if (!udpSocket->bind(12345, QUdpSocket::ShareAddress)) {
    qDebug() << "UDP 바인딩 실패!";
} else {
    qDebug() << "UDP 수신 대기중 (포트 12345)";
}

connect(udpSocket, &QUdpSocket::readyRead,
        this, &MainWindow::readPendingDatagrams);
```

조작

키보드로 좌우상하 각도 조절
마우스로 파워 조절

```
connect(ui->btnFire, &QPushButton::pressed, this, [=]() {
    if (gameOver) return;
    spacePressed = true;
    powerTimer->start();
});

connect(ui->btnFire, &QPushButton::released, this, [=]() {
    if (gameOver) return;
    spacePressed = false;
    powerTimer->stop();
    fireBall();
    power = 10;
    ui->sliderPower->setValue(power);
});
```

```
void MainWindow::keyPressEvent(QKeyEvent *event)
{
    switch (event->key()) {
    case Qt::Key_A:
        angle2 -= 1;
        if (angle2 < -90) angle2 = -90;
        ui->sliderAngle2->setValue(angle2);
        break;
    case Qt::Key_D:
        angle2 += 1;
        if (angle2 > 90) angle2 = 90;
        ui->sliderAngle2->setValue(angle2);
        break;
    case Qt::Key_W:
        angle1 += 1;
        if (angle1 > 90) angle1 = 90;
        ui->sliderAngle1->setValue(angle1);
        break;
    case Qt::Key_S:
        angle1 -= 1;
        if (angle1 < 0) angle1 = 0;
        ui->sliderAngle1->setValue(angle1);
        break;
    default:
        QMainWindow::keyPressEvent(event);
    }
}
```

Qt 프로젝트

감사합니다

로빛 20기 인턴 | 김민조, 강은구