

Fight Pong! & Defense Pong!

[파퐁! 디퐁!]

# Qt 프로젝트 발표

3조 : 20기 인턴 노기문, 최세영

# 00. 목차

TEAM RO:BIT

01

제작동기

02

알고리즘,게임  
시나리오

03

코드 설명

04

시연영상

05

시연

01

# 제작동기

## 제작동기



### 퐁[Pong]

The Mother of All Video Games

모든 비디오 게임의 어머니

The First, Video Games

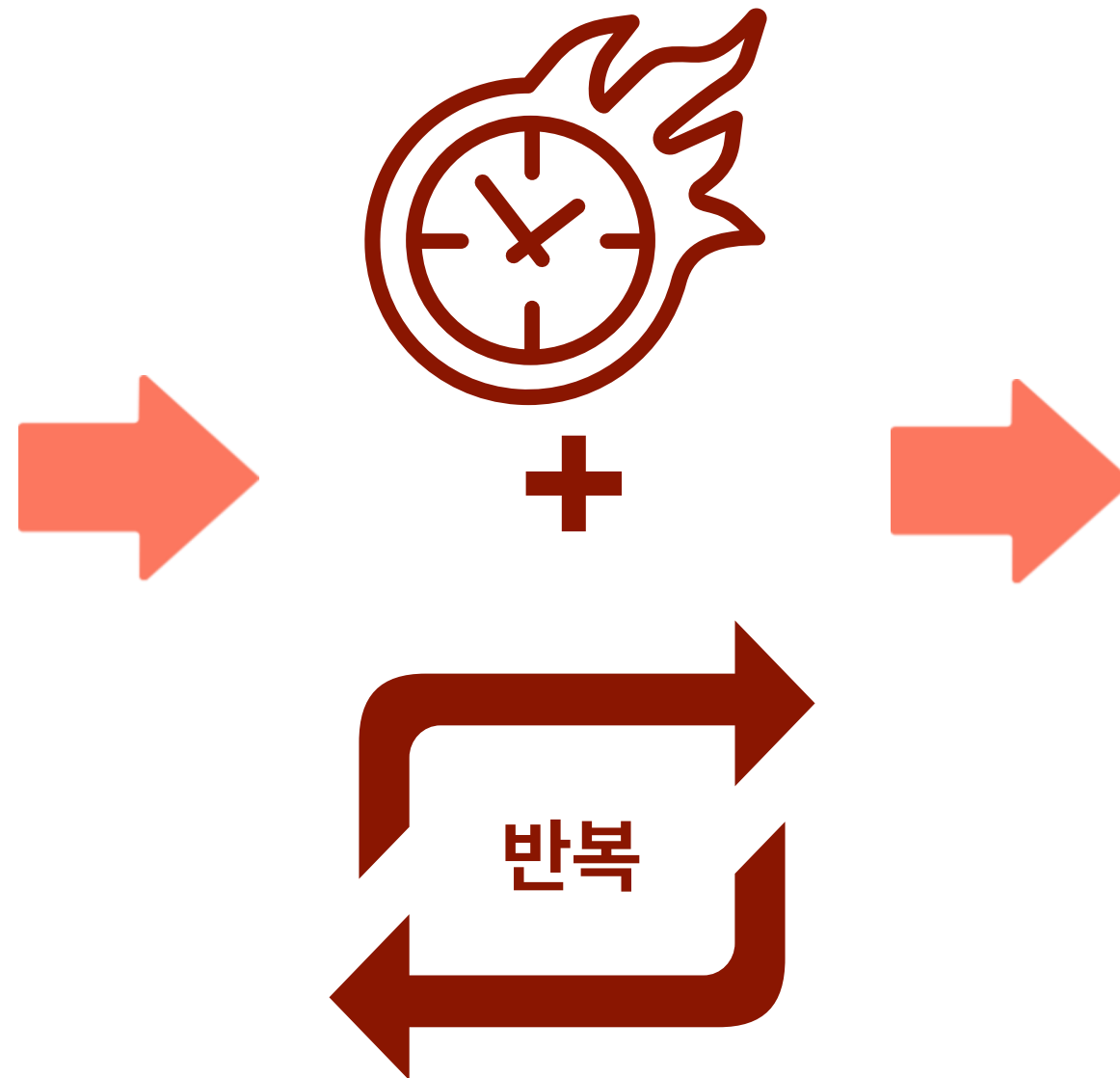
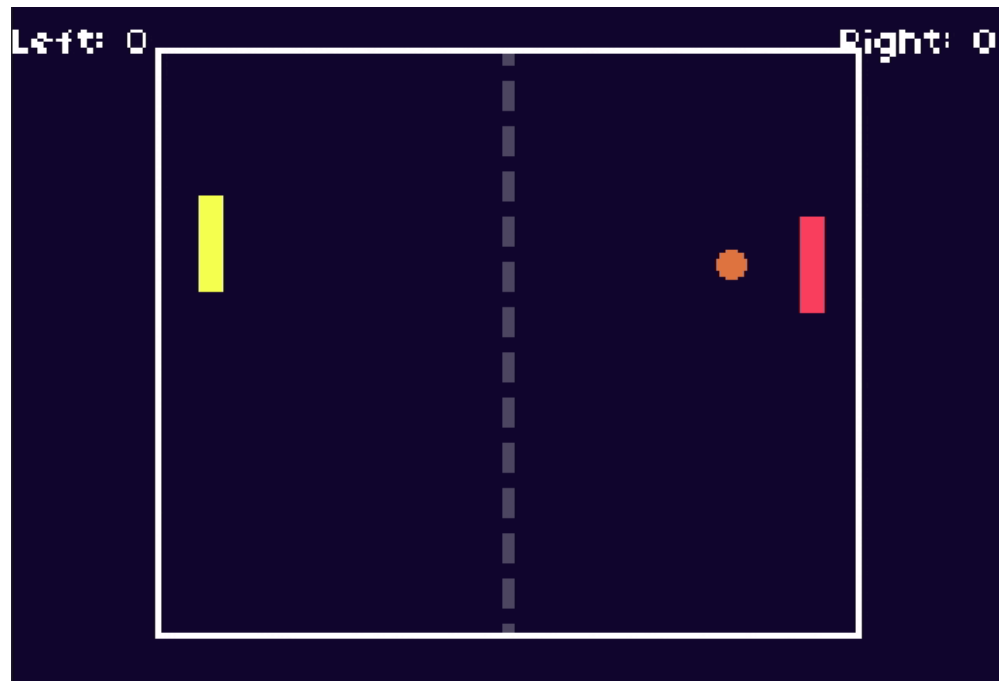
최초의 비디오게임

1970년대 초반 출시 후 2년 만에 약 8,000대의 아케이드 기기를 판매하며 폭발적인 인기를 얻었고, 이는 하루 약 35000~40000달러의 수익을 창출하는 등 비디오 게임 산업의 초석을 다지는 데 크게 기여

하지만...

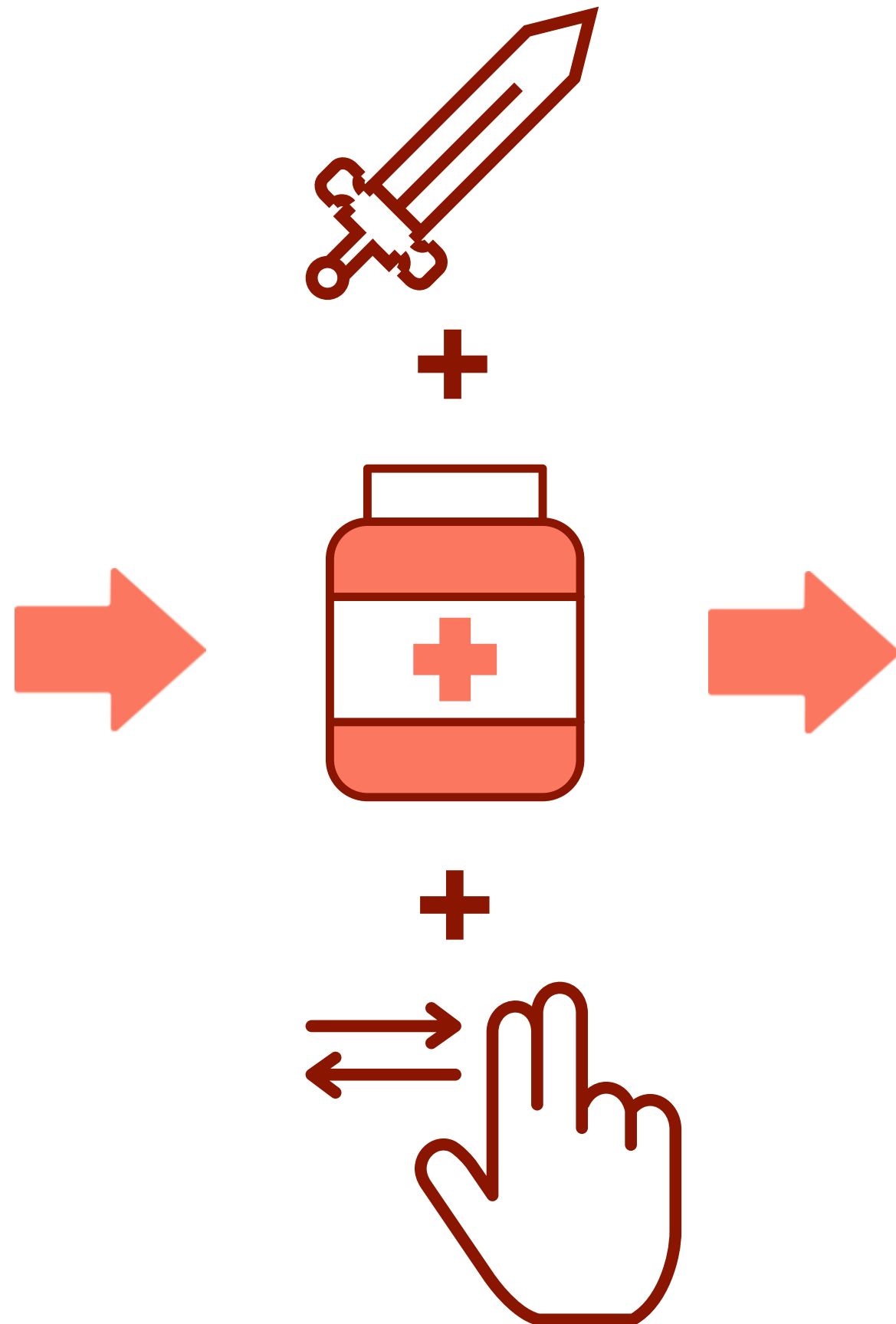
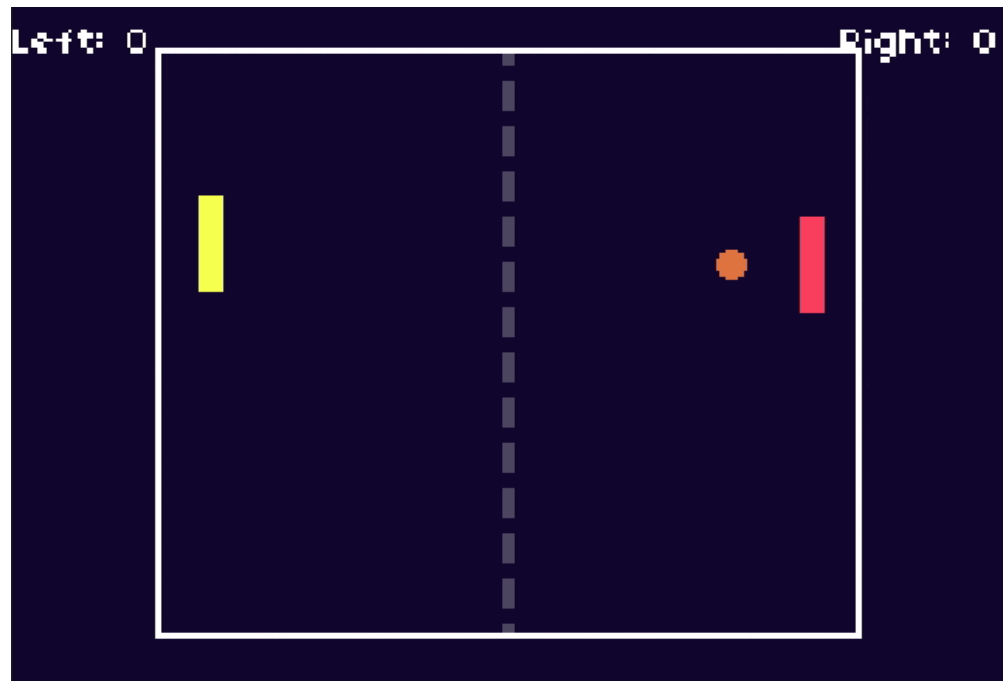
# 01. 제작동기

TEAM RO:BIT



# 01. 제작동기

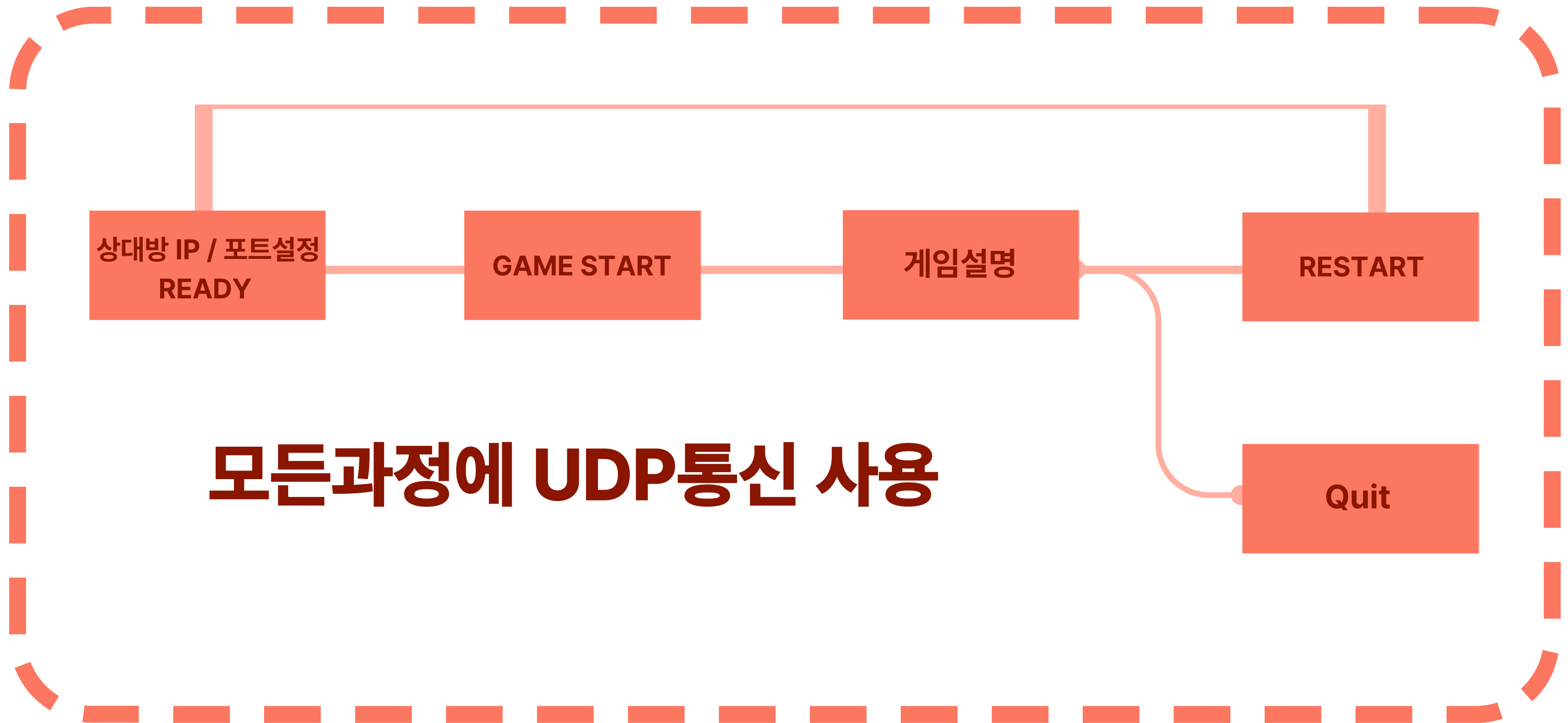
TEAM RO:BIT



02

# 알고리즘, 게임 시나리오

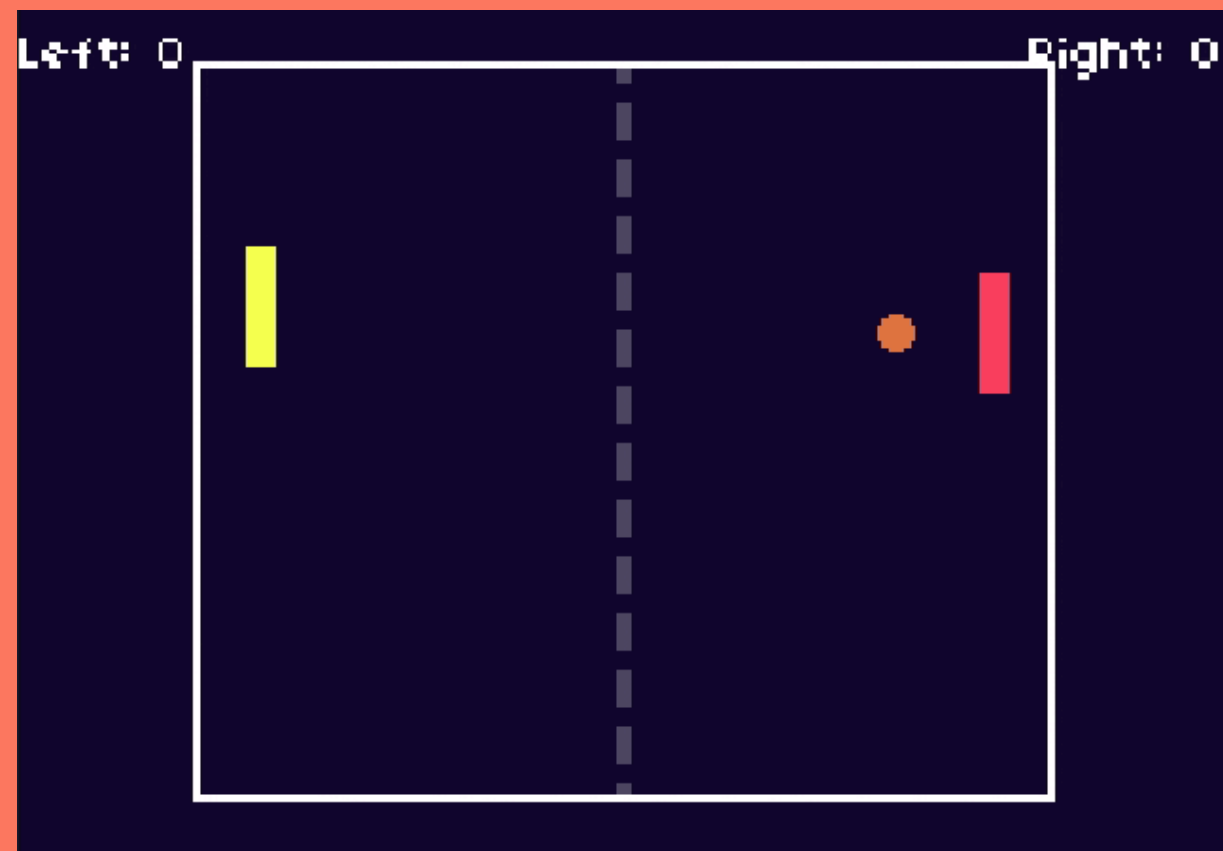
## 알고리즘설명



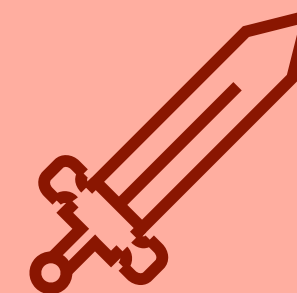


## 게임 시나리오

### 기존/기본



### 새로운점



노란색 아이템  
상대방을 맞추면  
HP : -1



핑크색 아이템  
획득하면  
HP : +1

03

# 코드 설명

## 주요코드설명

인터페이스  
클래스

homescreen.ui / gamewidget.ui외에 클래스 구성을 설명합니다.

통신설정

통신 설정에대한 코드설명입니다. IP,포트설정이 나옵니다.

게임기본

게임의 기본인 풍을 구현한 코드 설명입니다.

아이템구현

반사한 공을 통해 획득한 아이템에 대한 설명입니다

03.

TEAM RO:BIT

## 주요코드설명 UI구성

01. homescreeen.ui

### 인터페이스

Local Port:

Enter your local port, opponent IP and port, then click Connect. Opponent IP:  Opponent Port:

Font Combo Box

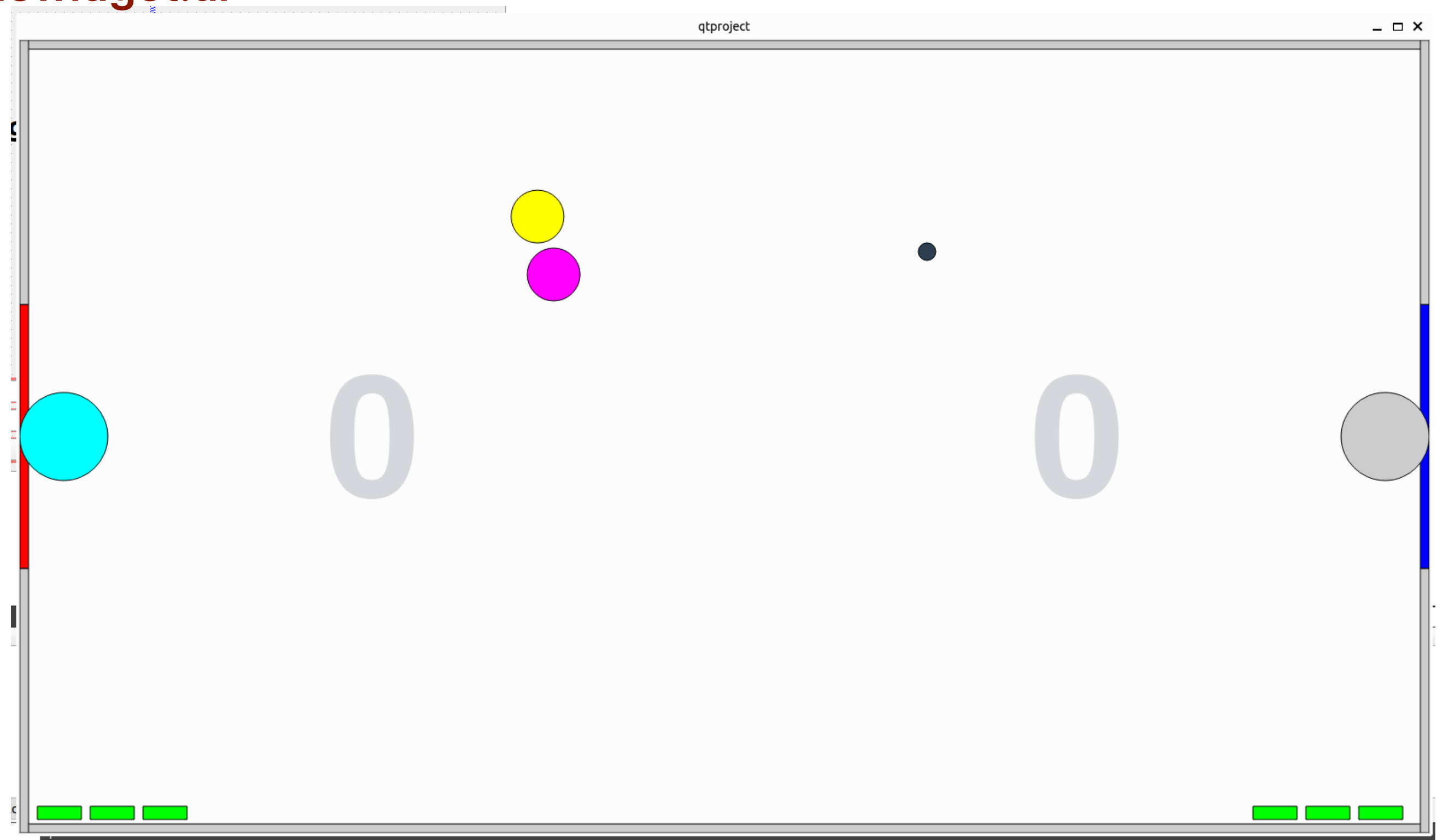
03.

## 주요코드설명 UI구성

01.

gamewidget.ui

## 인터페이스



03.

## 주요코드설명 homescreen.h

01.

### 클래스 역할

1. 로컬 포트 바인드, 상대 IP/포트 입력 검증.
2. Ready/상대 Ready 상태 동기화.
3. Host/Client 결정 및 안내.
4. 재경기 요청/수락/거절 처리.
5. 결과 화면, 카운트다운, 상태 메시지 표시.

## 03.

### 주요코드설명 gamewidget.h

#### 01.

#### 클래스 역할

1. 그래픽/씬 관리
2. 게임 루프 실행
3. 입력 처리
4. 네트워크 동기화
5. 아이템 처리

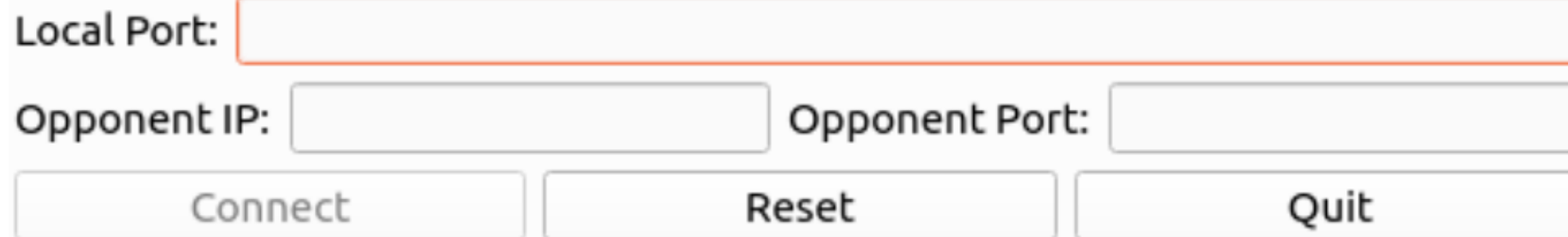
03.

**주요코드설명** 먼저 IP와 포트 설정하는 과정은 다음과 같이 이루어져 있습니다.

02.

1. 입력 → 2. 검증 → 3. 저장

## 통신설정



Local Port:

Opponent IP:  Opponent Port:

먼저 `homescreen.ui`에서  
다음과 같이 위젯이  
구성되어있습니다.



03.

## 주요코드설명

TEAM RO:BIT

02.

## 통신설정

Local Port:

Opponent IP:  Opponent Port:

```
1 위젯:  
2 local_port_input_line, ip_input_line, port_input_line.  
3 onConnectClicked()  
4  
5 QString localPortStr = ui->local_port_input_line->text();  
6 QString ip = ui->ip_input_line->text();  
7 QString peerPortStr = ui->port_input_line->text();  
8 quint16 localPort = localPortStr.toUShort(&okLocal);  
9 quint16 peerPort = peerPortStr.toUShort(&okPeer);
```

```
1 m_localPort = localPort;           // 내가 바인드할 로컬 수신 포트  
2 m_peerAddress = QHostAddress(ip);  // 상대 IP  
3 m_peerPort = peerPort;             // 상대 수신 포트
```

숫자 변환 실패시 0, 공란이면 오류  
표시 후 리턴합니다.

여기서 나와 상대의  
IP/포트를 설정하고 나면  
[입력]



이 코드를 통해 상대와 UDP  
통신을 검증하게 되며  
[검증]



성공 시 내부 상태에 고정 저  
장합니다.  
[저장]

**주요코드설명** 그런다음 상대의 움직임의 통신은 다음과정을 거칩니다.

02.

통신설정

상대의 키 입력 감지 → 상대가 자신의 위치 송신 → 호스트(내)가 입력 패킷 수신  
→ 호스트에서 INPUT 처리

GameWidget::keyPressEvent  
/ keyReleaseEvent 안에서  
m\_isMovingUp1,  
m\_isMovingDown1,  
m\_isMovingLeft1,  
m\_isMovingRight1 플래그가 바  
뀜.  
updateAnimation()에서 이 불리  
언 값을 보고 m\_paddle1(자신의 패  
들) 좌표를 moveBy()로 이동.

```
if (!m_isHost) {
    sendPaddlePosition();
}
```

```
void MainWindow::onReadyRead() {
    while (sock->hasPendingDatagrams())
    {
        QByteArray buf;
        buf.resize(int(sock
        >pendingDatagramSize()));
        QHostAddress from; quint16 port;
        sock->readDatagram(buf.data(),
        buf.size(), &from, &port); // 여기서 UDP 패
        킷 읽음
        gw_->onDatagramReceived(buf);
    }
}
```

```
else if (header == "INPUT" &&
m_isHost) {
    qreal remoteX, remoteY;
    in >> remoteX >> remoteY;
    m_paddle2-
    >setPos(remoteX, remoteY);
    qDebug() << "GameWidget:
    Host received INPUT: " <<
    remoteX << remoteY;
}
```

[입력감지]



[위치송신]



[패킷수신]



[INPUT 처리]

[클라이언트(상대)]

[호스트(나)]

## 주요코드설명 WASD키 입력 처리

## 03.

### 게임기본

```
void GameWidget::keyPressEvent(QKeyEvent *event) {
    if (event->isAutoRepeat()) return;
    switch (event->key()) {
        case Qt::Key_W: m_isMovingUp1 = true; break;
        case Qt::Key_S: m_isMovingDown1 = true; break;
        case Qt::Key_A: m_isMovingLeft1 = true; break;
        case Qt::Key_D: m_isMovingRight1 = true; break;
        default: QWidget::keyPressEvent(event);
    }
}

void GameWidget::keyReleaseEvent(QKeyEvent *event) {
    if (event->isAutoRepeat()) return;
    switch (event->key()) {
        case Qt::Key_W: m_isMovingUp1 = false; break;
        case Qt::Key_S: m_isMovingDown1 = false; break;
        case Qt::Key_A: m_isMovingLeft1 = false; break;
        case Qt::Key_D: m_isMovingRight1 = false; break;
        default: QWidget::keyReleaseEvent(event);
    }
}
```

## 주요코드설명 벽에 공이 튕기는 원리

## 03.

### 게임기본

```
m_ellipseItem->moveBy(m_ellipseSpeedX, m_ellipseSpeedY);
QRectF ellipseRect = m_ellipseItem->sceneBoundingRect();
if (ellipseRect.top() <= sceneRect.top()) {
    m_ellipseSpeedY *= -1;           // 위쪽 벽 반사
    m_ellipseItem->setY(sceneRect.top());
}
if (ellipseRect.bottom() >= sceneRect.bottom()) {
    m_ellipseSpeedY *= -1;           // 아래쪽 벽 반사
    m_ellipseItem->setY(sceneRect.bottom() - ellipseRect.height());
}
if (ellipseRect.left() <= sceneRect.left() && !m_ellipseItem->collidesWithItem(m_goal1)) {
    m_ellipseSpeedX *= -1;           // 왼쪽 벽 반사 (단, 골대는 예외)
    m_ellipseItem->setX(sceneRect.left());
}
if (ellipseRect.right() >= sceneRect.right() && !m_ellipseItem->collidesWithItem(m_goal2)) {
    m_ellipseSpeedX *= -1;           // 오른쪽 벽 반사 (단, 골대는 예외)
    m_ellipseItem->setX(sceneRect.right() - ellipseRect.width());
}
```

## 주요코드설명 공이 패들에 튕기는 원리

03.

## 게임기본

```
1  auto handlePaddleCollision = [&](QGraphicsEllipseItem* paddle, bool isPlayer1) {
2      if (m_ellipseItem->collidesWithItem(paddle)) {
3          m_lastPlayerToHitBall = isPlayer1 ? 1 : 2;
4
5          // 톱날볼일 때 체력 감소 처리
6          if (m_isSawBladeBall) {
7              if (isPlayer1) { m_player1Health--; normalizeBall(); }
8              else { m_player2Health--; normalizeBall(); }
9              updateHealthDisplay();
10             if (m_player1Health <= 0) { endGame("PLAYER 2 WINS!"); return; }
11             if (m_player2Health <= 0) { endGame("PLAYER 1 WINS!"); return; }
12         }
13
14         // 반사 각도 계산
15         qreal speed = qSqrt(m_ellipseSpeedX * m_ellipseSpeedX + m_ellipseSpeedY * m_ellipseSpeedY) + BALL_SPEED_INCREMENT;
16         qreal intersectY = (m_ellipseItem->pos().y() + 10) - (paddle->pos().y() + 50);
17         qreal normY = intersectY / 50.0;
18         qreal angle = normY * qDegreesToRadians(45.0);
19         m_ellipseSpeedX = speed * qCos(angle) * (isPlayer1 ? 1 : -1);
20         m_ellipseSpeedY = speed * qSin(angle);
21     }
22 };
23 handlePaddleCollision(m_paddle1, true);
24 handlePaddleCollision(m_paddle2, false);
```

## 주요코드설명 득점하는 원리

## 03.

## 게임기본



```
1
2  if (m_ellipseItem->collidesWithItem(m_goal1) || m_ellipseItem->collidesWithItem(m_goal2)) {
3      const bool scoredLeft = m_ellipseItem->collidesWithItem(m_goal1); // 왼쪽 골대 득점 여부
4      if (scoredLeft) m_player2Score++;
5      else m_player1Score++;
6      updateScoreDisplay();
7
8      // 승리 조건 확인
9      if (m_player1Score >= WINNING_SCORE) { endGame("PLAYER 1 WINS!"); return; }
10     if (m_player2Score >= WINNING_SCORE) { endGame("PLAYER 2 WINS!"); return; }
11
12     // 중앙으로 공 리셋 + 방향 설정
13     m_ellipseItem->setPos(0, 0);
14     m_ellipseSpeedX = scoredLeft ? +INITIAL_BALL_SPEED : -INITIAL_BALL_SPEED;
15     m_ellipseSpeedY = (QRandomGenerator::global()->bounded(2) == 0 ? 1 : -1) * INITIAL_BALL_SPEED;
16     normalizeBall();
17 }
18
```

03.

TEAM RO:BIT

주요코드설명 아이템 먹는 처리 -체력 아이템

04.

아이템구현



```
1  if (m_healthItem && m_ellipseItem->collidesWithItem(m_healthItem)) {
2      if (m_lastPlayerToHitBall == 1 && m_player1Health < MAX_HEALTH)
3          m_player1Health++;
4      else if (m_lastPlayerToHitBall == 2 && m_player2Health < MAX_HEALTH)
5          m_player2Health++;
6
7      updateHealthDisplay();
8
9      // 아이템 제거
10     m_scene->removeItem(m_healthItem);
11     delete m_healthItem;
12     m_healthItem = nullptr;
13
14     // 5초 뒤 새로 생성
15     QTimer::singleShot(5000, this, &GameWidget::spawnHealthItem);
16 }
```



03.

TEAM RO:BIT

주요코드설명 아이템 먹는 처리 -공격 아이템

04.

아이템구현

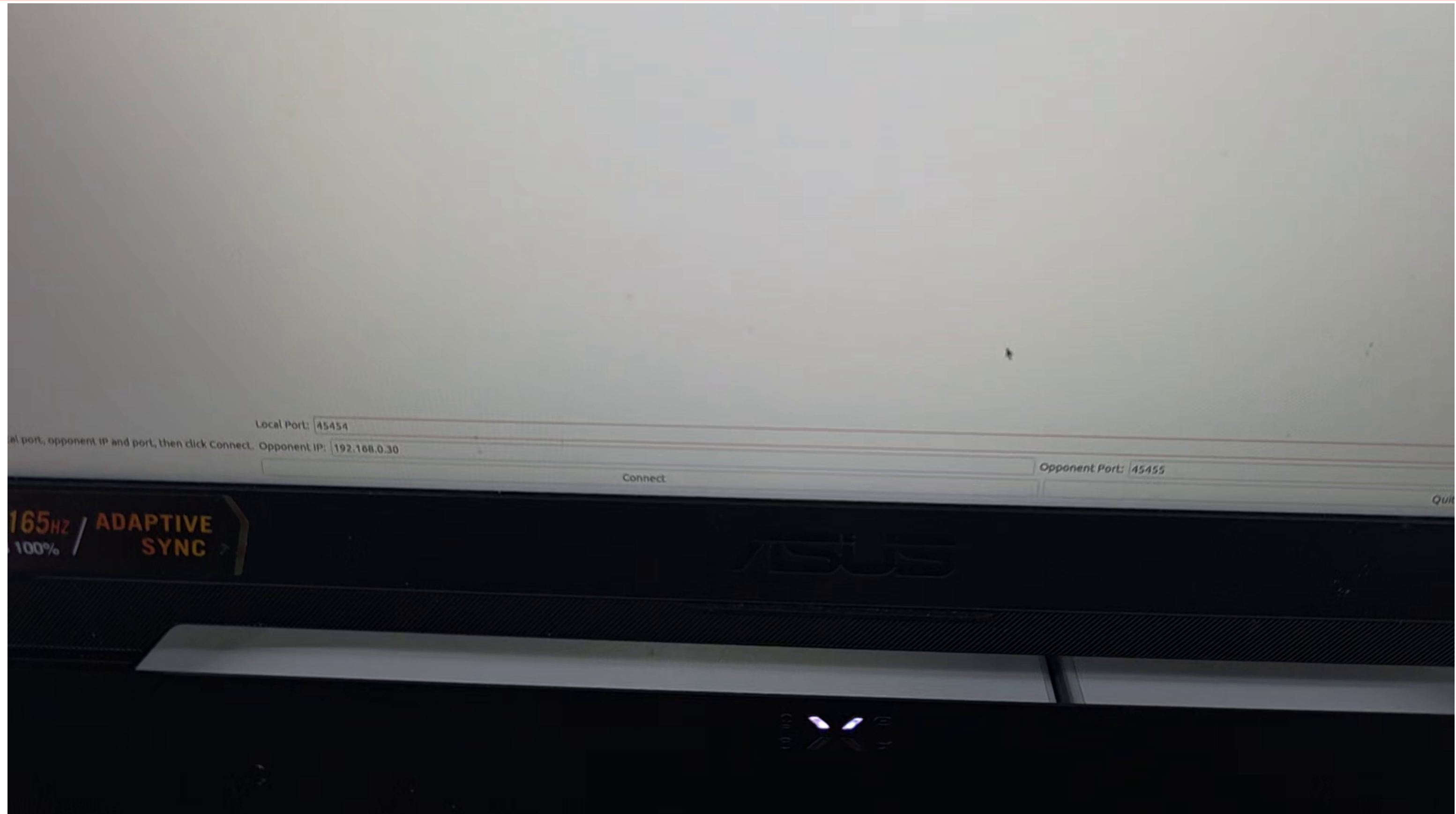


```
1  if (m_sawItem && m_ellipseItem->collidesWithItem(m_sawItem)) {  
2      m_isSawBladeBall = true;  
3      m_ellipseItem->setBrush(QColor("#FFFFE0")); // 공 색을 바꿈  
4  
5      QTimer::singleShot(5000, this, &GameWidget::normalizeBall); // 5초 뒤 원래 공으로 복귀  
6  
7      m_scene->removeItem(m_sawItem);  
8      delete m_sawItem;  
9      m_sawItem = nullptr;  
10  
11     // 5초 뒤 새로 스폰  
12     QTimer::singleShot(5000, this, &GameWidget::spawnSawItem);  
13 }
```



04

# 시연 영상



05

# 시연

Fight Pong! & Defense Pong!

**감사합니다**

20기 인턴 노기문, 최세영

## 주요코드설명 homescreen.h

### 01.

**클래스** setStatus(QString, bool) 상태 텍스트 표시. 오류 시 색상 등 구분.  
resetUI() 초기화.

showCountdown(int) 경기 시작 카운트다운 표시.

showResultPage(QString) 결과 페이지 전환.

showConnectionPage() 접속 페이지 전환.

setRematchStatus(QString) 재경기 상태 텍스트.

enableRematchButtons(bool) 재경기 버튼 활성화/비활성.

Getters: getPeerAddress(), getPeerPort(), getLocalPort(), isHost()

setMyAddress(QHostAddress) 로컬 IP 표시/저장.

## 주요코드설명 homescreen.h

### 01.

**클래스** 시그널 (외부로 알림)

`startGame(bool isHost)` 게임 위젯 시작 트리거.

`sendDatagram(QByteArray, QHostAddress, quint16)` 패킷 송신 요청. 실제 송신은 상위가 수행.

`rematchRequested()` 내가 재경기 눌렀음.

`opponentVotedForRematch()` 상대가 재경기 수락.

`opponentQuit()` 상대가 종료.

`Reset()` 홈으로 초기화 요청.

`bindSocket(quint16 localPort)` 상위에 소켓 바인드 지시.

## 주요코드설명 homescreen.h

### 01.

**클래스** 슬롯 (수신/버튼/타이머)

`onDatagramReceived(QByteArray, QHostAddress, quint16, bool isConnected)`

상위에서 받은 UDP 패킷을 UI 상태로 반영. 예: READY, CONNECT, REMATCH, QUIT 같은 헤더 분기 처리에 사용.

버튼:

`onConnectClicked()` 입력 검증 → `bindSocket(m_localPort)` → 상대방에게 READY/HELLO 송신 → 상태 갱신.

`onQuitClicked()` 종료 의사 송신 → `opponentQuit()` 처리 → 홈 복귀.

`onRematchClicked()` 재경기 의사 송신 → `rematchRequested()`.

`onExitToHomeClicked()` 결과 화면에서 홈으로.

`on_reset_botton_clicked()` UI 초기화.

03.

## 주요코드설명 homescreen.h

01.

**클래스** 입력 변화:

onInputChanged() IP/포트 유효성 검사 → checkReadyState().

타이머:

sendReadyStatus() 주기적으로 READY 패킷 브로드캐스트(상대 동기화용).



## 주요코드설명 homescreen.h

### 01.

#### 클래스 전형적 흐름

사용자가 로컬 포트·상대 IP/포트 입력 → onInputChanged()가 유효성 확인 → onConnectClicked()에서 bindSocket(localPort) 시그널 → 상위가 QUdpSocket::bind.

sendReadyStatus()가 주기적 READY 패킷을 sendDatagram(...)로 요청.

상대 READY 수신 시 onDatagramReceived(...)가 m\_opponentReady=true로 세팅하고 UI 업데이트.

m\_isReady && m\_opponentReady가 되면 역할 결정(m\_amlHost) 후 startGame(m\_amlHost) 시그널 방출.

게임 종료 후 결과 페이지 표시. 재경기 버튼 → onRematchClicked() → 상대 수락 패킷 수신 시 opponentVotedForRematch() → startGame(...) 재호출.

## 주요코드설명 gamewidget.h

### 01.

#### 클래스 인터페이스

`GameWidget(QWidget*) / ~GameWidget()` 초기화·리소스 관리.

`startGame(bool isHost)` 호스트/클라 모드 지정 후 게임 시작.

`onDatagramReceived(const QByteArray&) slot` ⇒ UDP 수신 바이트를 해석해 상태 반영.

`keyPressEvent / keyReleaseEvent` : WASD 입력을 내부 플래그에 반영.

`gameOver(const QString&) signal` : 게임 종료 알림.

`sendDatagram(const QByteArray&) signal` : 상위(MainWindow)로 송신 요청.

## 주요코드설명 gamewidget.h

### 01.

**클래스** 내부 슬롯(타이머 구동)

`updateAnimation()`

매 프레임 처리. 이동, 충돌, 득점, 상태 송신(호스트), 입력 송신(클라).

`spawnHealthItem()` / `spawnSawItem()`

아이템 생성. 호스트만 수행.

`relocateItems()`

아이템 주기적 재배치.

`normalizeBall()`

튍날 효과 해제.

## 03.

### 주요코드설명 gamewidget.h

#### 01.

#### 클래스 프라이빗 유틸

`resetGame()` 초기 배치와 수치 리셋.

`endGame(const QString&)` 종료 처리 및 통지.

`sendPaddlePosition()` 클라가 자신의 패들 좌표를 "INPUT"로 송신.

`sendGameState()` 호스트가 권위 상태 "STATE" 송신.

`updateScoreDisplay()` / `updateHealthDisplay()` UI 갱신

## 03.

### 주요코드설명 gamewidget.h

#### 01.

**클래스** 씰.오브젝트

m\_scene 전체 그리기 컨테이너.

m\_paddle1, m\_paddle2 원형 패들.

m\_ellipseltem 공.

m\_goal1, m\_goal2 좌·우 골대.

점수/게임오버 텍스트: m\_player1ScoreText, m\_player2ScoreText,  
m\_gameOverText.

03.

## 주요코드설명 gamewidget.h

01.

클래스 타이머

m\_animationTimer 16ms 틱. 게임 루프.

m\_itemRelocationTimer 아이템 위치 주기 변경.

## 주요코드설명 gamewidget.h

### 01.

#### 클래스 물리/상태

공 속도: m\_ellipseSpeedX, m\_ellipseSpeedY.

입력 플래그(로컬 제어 대상):

m\_isMovingUp1/Down1/Left1/Right1

m\_isMovingUp2/...(현 구조에선 미사용 또는 확장용).

점수/체력: m\_player1Score, m\_player2Score, m\_player1Health, m\_player2Health.

체력바 아이템 배열: m\_player1HealthBars, m\_player2HealthBars.

## 03.

### 주요코드설명 gamewidget.h

#### 01.

#### 클래스 아이템

m\_healthItem 체력 회복 구슬.

m\_sawItem 톱날 효과 구슬.

m\_isSawBladeBall 톱날 상태 플래그.

m\_lastPlayerToHitBall 마지막 타자(1/2).

스폰 영역: m\_SpawnArea(사용), m\_leftItemSpawnArea /  
m\_rightItemSpawnArea(확장 여지).



# 03.

## 주요코드설명 gamewidget.h

### 01.

#### 클래스 네트워크 모드

m\_isHost 호스트 여부.

호스트: 물리 계산 + sendGameState() 전송.

클라: 입력만 sendPaddlePosition() 전송, 상태는 수신해 반영.