

MFC Project : Rummikub

JOKER

김선재 김환희 이준혁 임정빈 차준혁



목차



팀 소개

프로젝트에 참여한 팀원들과 각자 맡은 역할에 대해 간단히 소개합니다.

Rummikub

주제 소개

선택한 프로젝트 주제와 선정 이유를 소개합니다.

Rummikub

주요 코드 소개

프로그램에 사용된 주요 코드를 설명합니다.

Rummikub

예외 처리

프로그램 개발 중에 발생한 예상치 못한 예외 처리 과정을 소개합니다.

Rummikub

프로그램 시연

완성된 프로그램을 발표 현장에서 직접 시연합니다.

Rummikub

팀 소개

JOKER



김선재

소켓 통신 기능
턴 관련 모든 기능
게임 시작 기능



김환희

타일 이동 기능
보고서 정리



이준혁

타일 이동 기능
공용판 검사 기능
Pass 기능



임정빈

Setback 기능
Receive 기능
PPT 정리



차준혁

타일 시각화
UI 랜더링
보고서 정리

주제 소개

: MFC로 보드게임 루미큐브 구현



주제 소개

: MFC로 보드게임 루미큐브 구현



루미큐브란?

루미큐브는 4가지 색, 13가지 숫자의 조합과 조커로 이루어진 타일 덱에서 타일을 나눠 가지고, 플레이어들은 차례가 되면 규칙에 맞게 자신의 타일들을 공용판에 등록하는 턴 기반 보드게임이다. 처음으로 자신의 모든 타일을 먼저 등록한 플레이어가 승리한다.

루미큐브를 주제로 선택한 이유

1. 구현 난이도가 높다. (게임 규칙 복잡, 실시간 통신 기능 필요)
2. 지금까지 우리가 배웠던 MFC 기능을 연계하기 적합하다.
3. 팀원들이 평소에 즐겨 하던 게임이다.

루미큐브 구현을 위해 어떤 기능이 필요한가?

실시간 통신 기능, 턴 관리 기능, 타일 배분 기능, 타일 이동 기능, 공용판 조건 검사 기능, 공용판 원상복귀 기능

주제 소개

: MFC로 보드게임 루미큐브 구현



루미큐브 규칙

0. 용어 정의

- a. **공용판**은 플레이어가 타일을 등록하는 공간으로, 이 곳의 타일은 모두가 볼 수 있다.
- b. **개인판**은 플레이어 자신의 타일을 보관하는 공간으로, 이 곳의 타일은 자신만 볼 수 있다.
- c. 개인판의 타일을 공용판에 제출하는 것을 **등록**이라고 한다.

1. 게임 준비

게임 시작 전에 플레이어들은 덱에서 각자 14개의 타일을 가져간다.

2. 게임 중

게임이 시작되고 자신의 턴이 돌아오면 플레이어는 다음 중 하나를 수행한다.

- a. **타일 등록 원칙**에 맞춰 타일을 등록하고 턴을 넘긴다. 이때 첫 등록이면 타일 숫자 총합이 30 이상이어야 한다.
- b. 해당 턴에 타일을 등록하지 못했다면 타일 하나를 개인판으로 가져온 뒤 턴을 넘긴다.

주제 소개

: MFC로 보드게임 루미큐브 구현



루미큐브 규칙

3. 타일 등록 원칙

- a. 3장 이상의 타일을 묶어 등록해야 한다.
- b. 타일 묶음은 같은 색의 연속된 숫자 또는 다른 색의 같은 숫자인 타일들로 구성되어야 한다.
- c. 조커는 모든 색, 모든 숫자로 사용이 가능하다.

4. 게임 종료

가장 먼저 개인판의 타일을 모두 등록한 플레이어가 승리한다.

주요 코드 소개

타일 구조체 정의 및 시각화

소켓 통신

게임 시작 및 턴 넘기기

공용판 검사

Setback

Pass

Receive

게임 종료

타일 옮기기

주요 코드 소개 : 타일 구조체 정의 및 시각화

타일 구조체 정의

타일은 다음과 같은 구조체로 정의했다.

Color : RED/GREEN/BLUE/BLACK

num : 타일의 숫자 (1~13), 빈 타일은 0

isJoker : 조커 여부 (2장)

tileId : 빈 타일 구분, 소켓 통신을 위해 별도 변수로 타일 index 관리

```
// 색상을 정의하는 열거형(enum)
enum Color {
    RED,
    GREEN,
    BLUE,
    BLACK
};

// 타일 구조체
struct Tile
{
    Color color;
    int num;
    bool isJoker;
    int tileId = -1; // 나머지 1~106, 비어있는 판으로
                      // 표시하기 위해 -1로 초기화
};
```

주요 코드 소개 : 타일 구조체 정의 및 시각화

타일 저장 배열 정의

서버는 다음 멤버 변수로 타일을 저장하고 관리한다.

1. `std::array<Tile, 106> m_tile_list;`
→ 전체 106개의 타일들을 순서대로 정렬해서 저장하는 원본 덱 배열
2. `std::array<Tile, 106> m_rand_tile_list;`
→ 실제 게임에서 사용하기 위해 원본 덱을 랜덤으로 셔플해서 저장하는 덱 배열

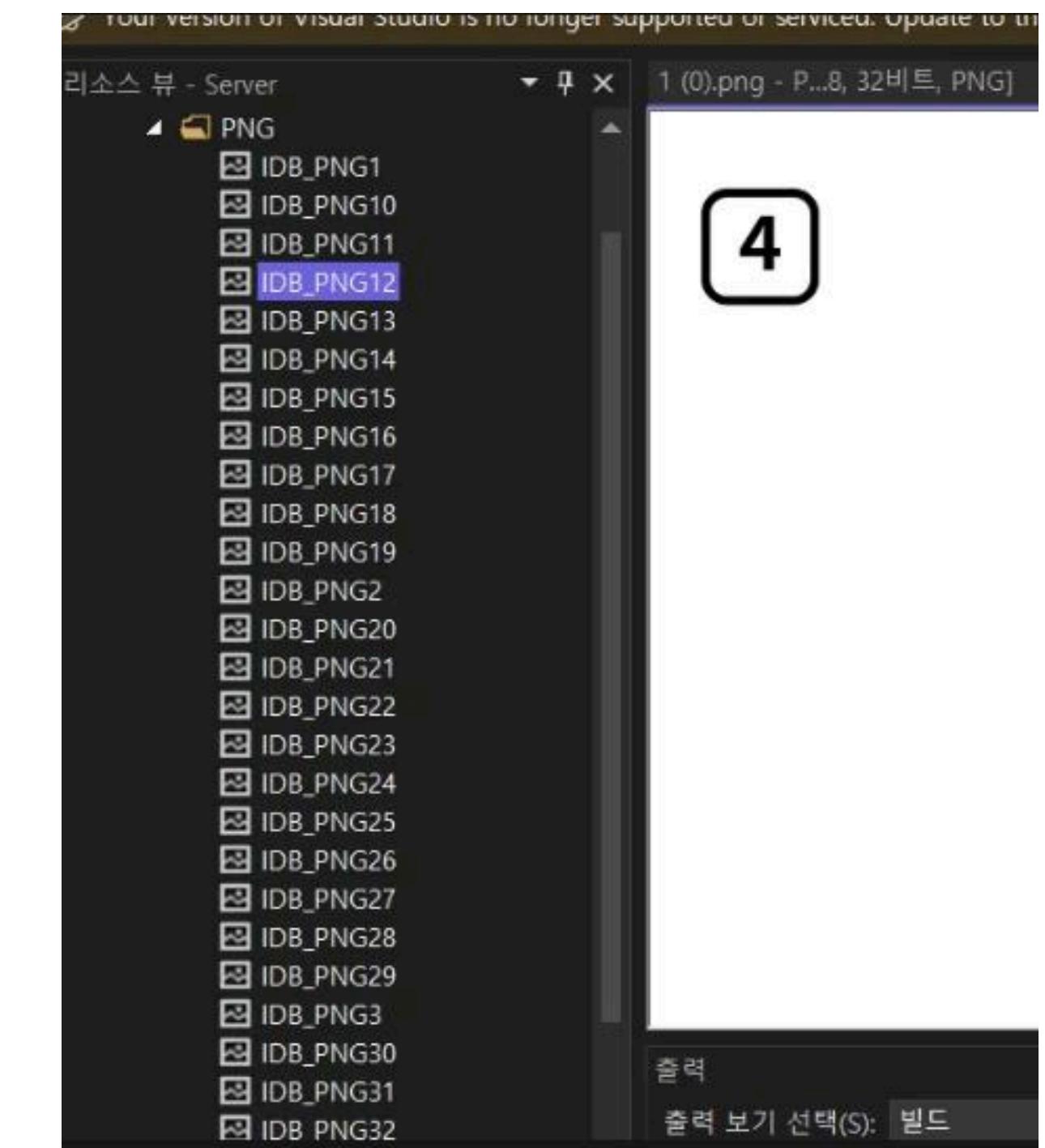
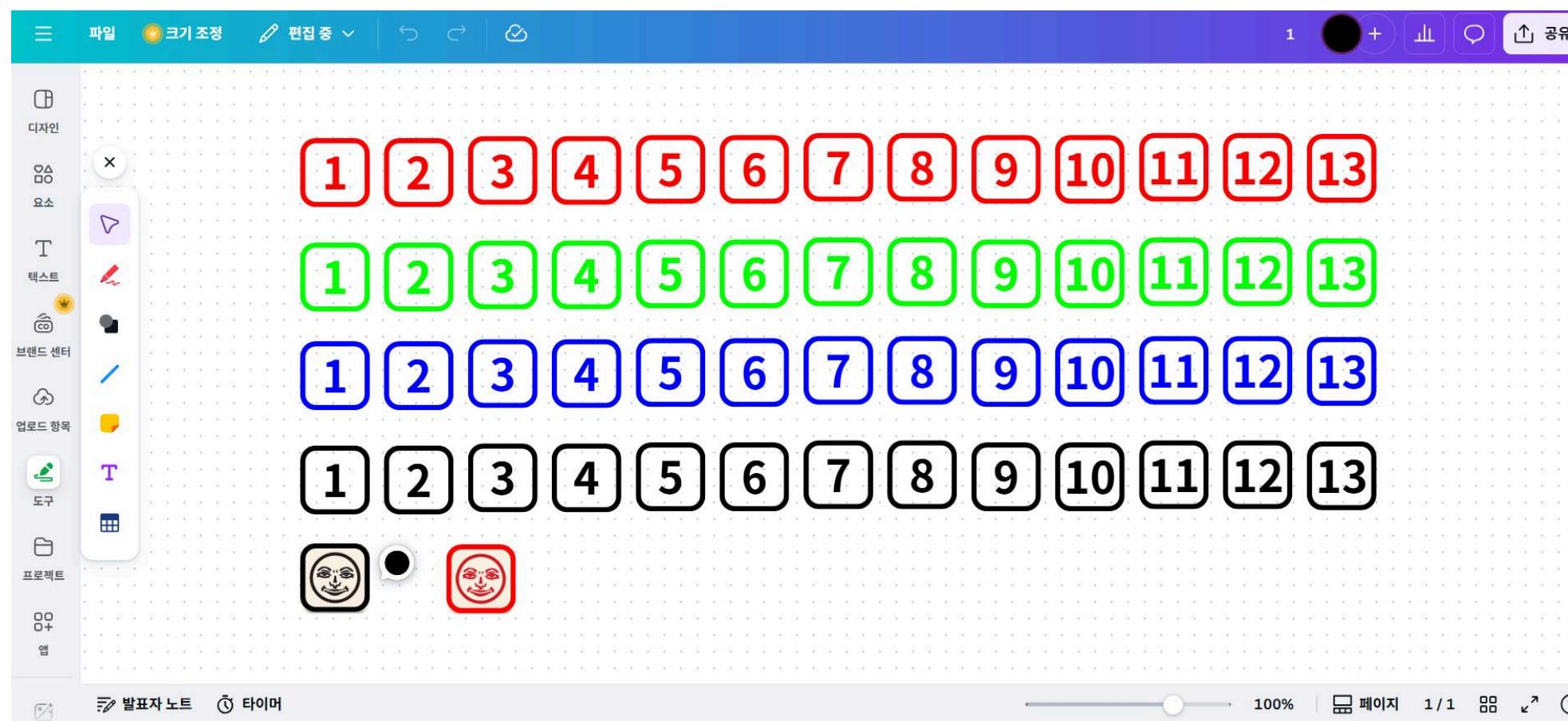
서버, 클라이언트를 포함하는 모든 플레이어는 다음 멤버 변수로 공용판/개인판을 관리한다.

1. `Tile m_public_tile[14][28];`
→ 공용판(13×27)의 각 칸에 위치하는 타일 정보를 저장하는 배열
2. `Tile m_private_tile[4][18];`
→ 개인판(3×17)의 각 칸에 위치하는 타일 정보를 저장하는 배열

주요 코드 소개 : 타일 구조체 정의 및 시각화

타일 이미지 제작

Canva로 타일을 디자인하고 다운받은 뒤, MFC 리소스에 등록해 사용했다.



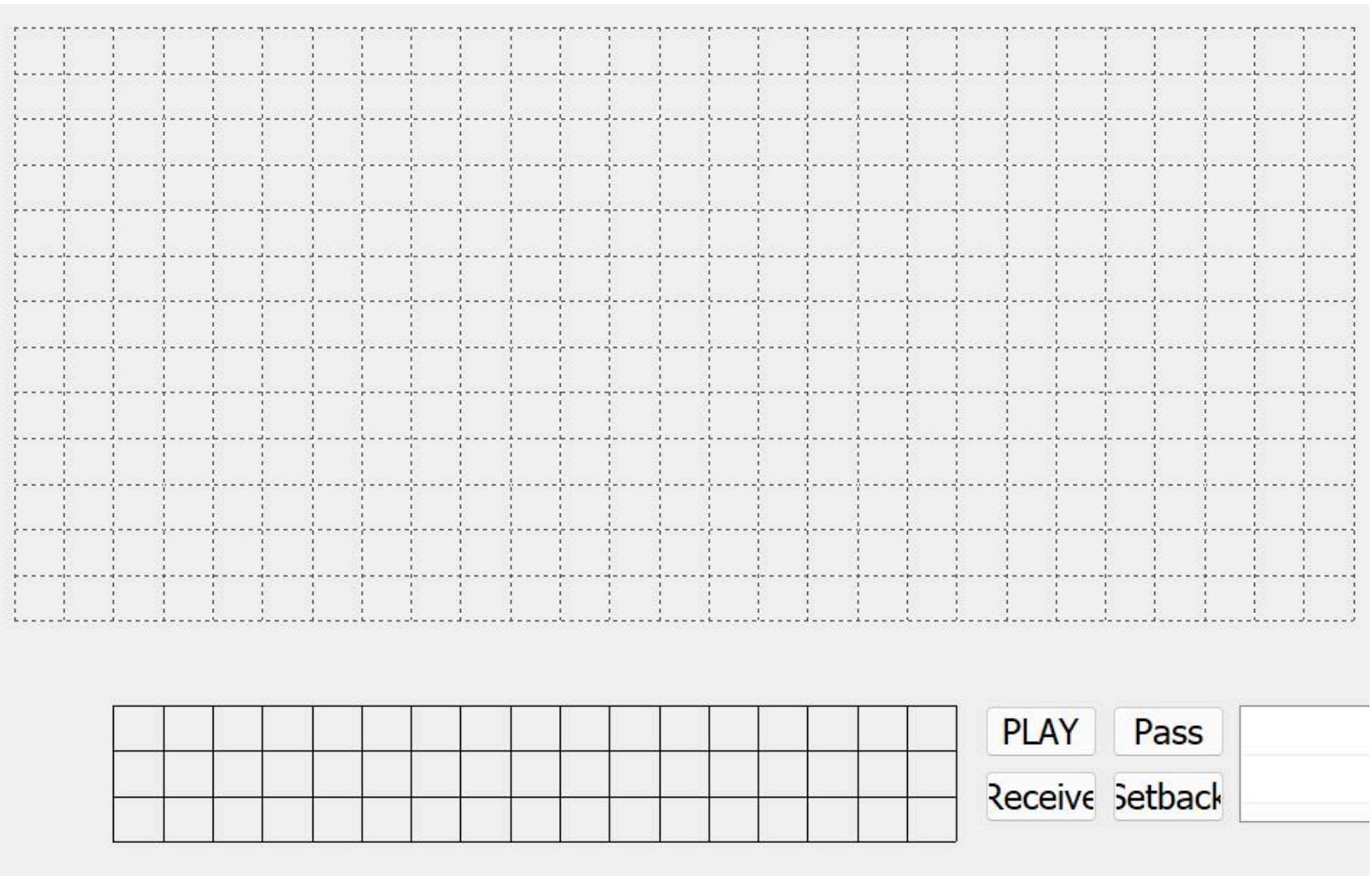
이미지 로드

CImage m_tile_image_list[106] 배열에 LoadPngFromResource()로 전체 타일 이미지를 로드해 두고 활용한다.

주요 코드 소개 : 타일 구조체 + 타일 시각화

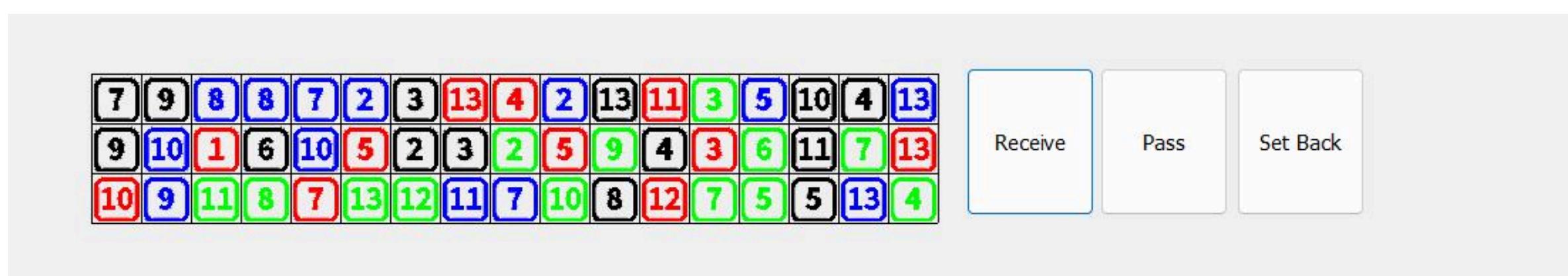
공용판/개인판 그리기

OnPaint()에서 CPen을 사용해 공용판/개인판의 각 칸을 그리도록 했다.



공용판/개인판에 이미지 적용 : DrawMyTiles()

- DrawMyTiles()는 공용판/개인판을 순회하며 각 칸에 위치한 타일에 대응되는 이미지를 출력한다.
- DrawMyTiles()는 OnPaint()에서 호출하도록 했다.



주요 코드 소개 : 소켓 통신

구현 방법: AsyncSocket 클래스 (Server)

AsyncSocket을 기본 구현으로 가지는 ListenSocket, ServiceSocket을 통해 클라이언트와 서버를 연결하게 한다.

ListenSocket : 클라이언트의 연결을 대기하는 소켓으로 클라이언트에서 연결을 시도하면 연결 로직이 진행된다.

ServiceSocket : 클라이언트가 원하는 서비스를 제공하는 소켓으로 실질적인 통신은 여기서 이루어진다.

구현 방법: AsyncSocket 클래스 (Client)

AsyncSocket을 기본 구현으로 가지는 ClientSocket 클래스 구현을 통해 서버와 연결을 시도한다.

ClientSocket: 서버에 연결을 시도하고, 서버와 통신을 진행한다. 서버에 메시지를 전달하고, 반대로 처리할 수 있다.

통신의 구조

서버에서 연결된 ListenSocket으로 클라이언트의 연결을 기다린다.

이후 ListenSocket으로 연결이 되면 클라이언트에 대응하는 ServiceSocket이 생성된다.

ServiceSocket을 관리하는 List를 추가하고, 메시지가 전달되면 그에 맞는 대응을 한다.

주요 코드 소개 : 소켓 통신

메시지 전달방법

소켓끼리 통신을 진행할때 메시지를 통해 내용을 전달한다.

이때 데이터의 크기를 줄여서 통신을 진행하기 위해 문자열단위로 기능단위 개발을 진행하였다.

이때 개발의 일관성을 유지하기 위해 다음의 형식을 적용하였다.

메시지의 구조

type : CHAT, PLACE, PASS, RECEIVE, PLAY ... type의 구분은 if문으로 해결

sender : 소켓의 주인의 이름 (소켓을 보낸 사람)

content : 전달할 내용. type에 따라 다른 정보가 입력될 예정

type:CHAT|sender:김선재|content:로그를뛰운다 : 로 key, value 구분, | 로 내용 구분

메시지의 헤더

추가적으로 동시에 문자열이 전달되는 버그로 인해 header(문자열의 길이)를 먼저 보낸다.

이후 서버가 header와 실제 메시지를 읽어서 여러 메시지가 혼합된 형태도 자연스럽게 진행된다.

주요 코드 소개 : 소켓 통신

메시지 전달 주요 함수

void ResponseMessage(const CString& strMsg, CServiceSocket* pSender);

클라이언트인 pSender에게 메시지를 전달한다. (단일 대상)

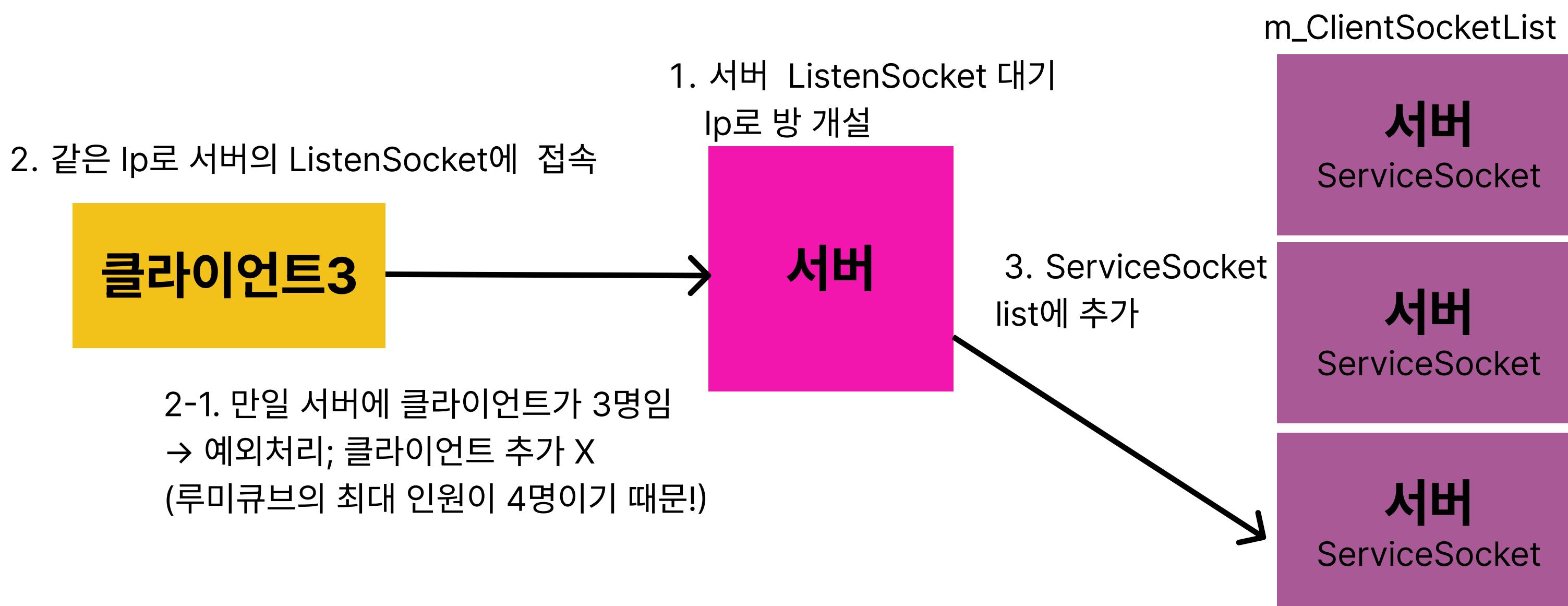
void BroadcastMessage(const CString& strMsg, CServiceSocket* pSender);

pSender를 제외한 클라이언트 모두에게 메시지를 전달한다. (다중 대상)

void RequestMessage(CString& strMsg);

클라이언트가 서버에게 메시지를 전달한다.

예제



주요 코드 소개 : 소켓 통신

메시지 전달 주요 함수

void ResponseMessage(const CString& strMsg, CServiceSocket* pSender);

클라이언트인 pSender에게 메시지를 전달한다. (단일 대상)

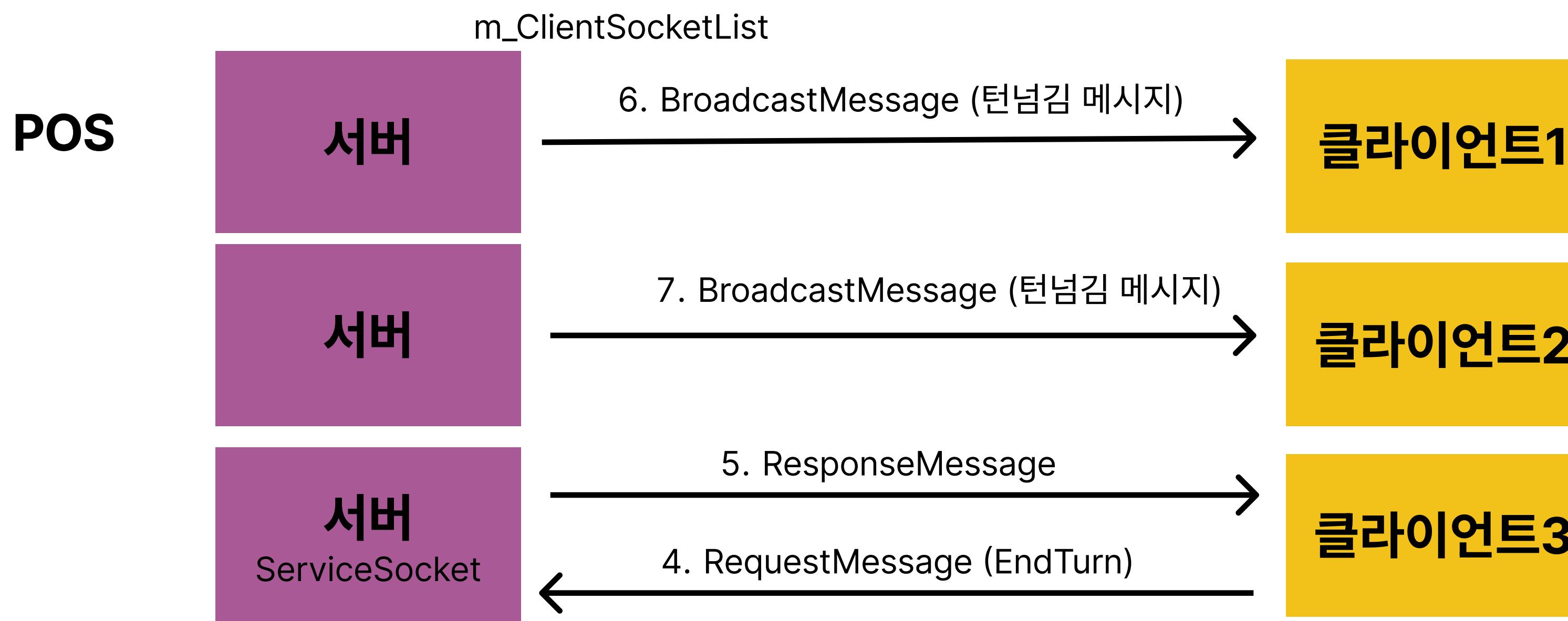
void BroadcastMessage(const CString& strMsg, CServiceSocket* pSender);

pSender를 제외한 클라이언트 모두에게 메시지를 전달한다. (다중 대상)

void RequestMessage(CString& strMsg);

클라이언트가 서버에게 메시지를 전달한다.

예제



주요 코드 소개 : 게임 시작 및 턴 넘기기

게임 시작 (타일 분배)

1. 타일 섞기 : 순서대로 정렬된 기본덱을 섞어 게임 덱 만들기

2. 턴 초기화 : 서버부터 게임 시작

3. 모든 플레이어들에게 각각 14개의 타일 분배

서버 : 덱을 관리하기에 타일을 바로 받는다.

클라이언트 : 덱을 관리하지 않기에 서버로부터 타일 ID가 담긴 메시지를 받는다.

클라이언트는 타일 ID를 보고 그에 대응하는 타일 생성해서 개인판에 넣는다. (**ParseldtoTile(tileid)**)



주요 코드 소개 : 게임 시작 및 턴 넘기기

구현

게임 시작 : 서버의 턴부터 게임이 시작 // 턴 넘기기 : NextTurn() 함수로 구현

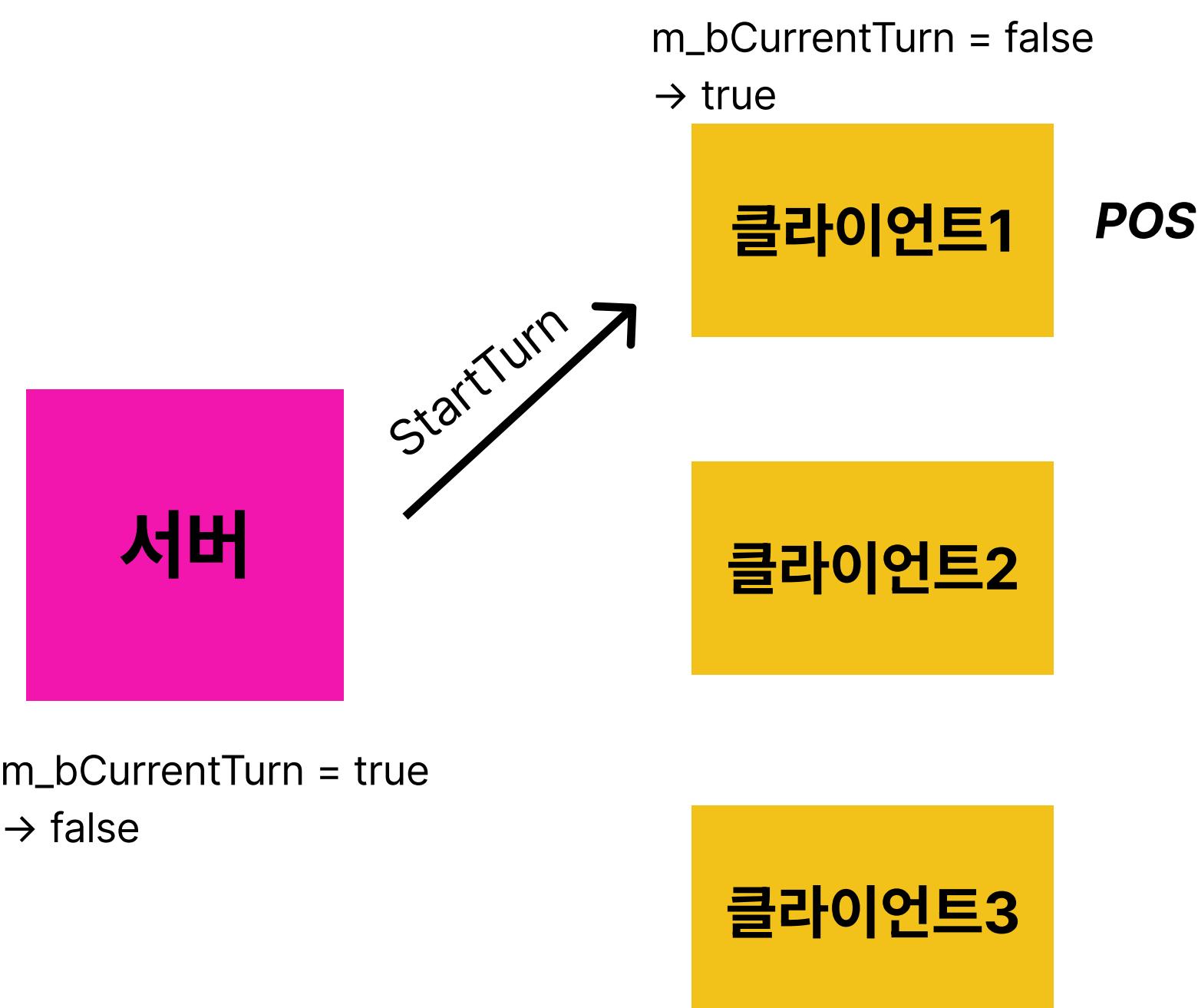
전체 턴 관리는 서버가, 클라이언트는 서버에게 **EndTurn** 메시지를 보내 턴 종료를 알림

m_clientSocketList의 POS : 현재 턴이 누구인지 가리킨다. (POS == null 이면 서버의 턴)

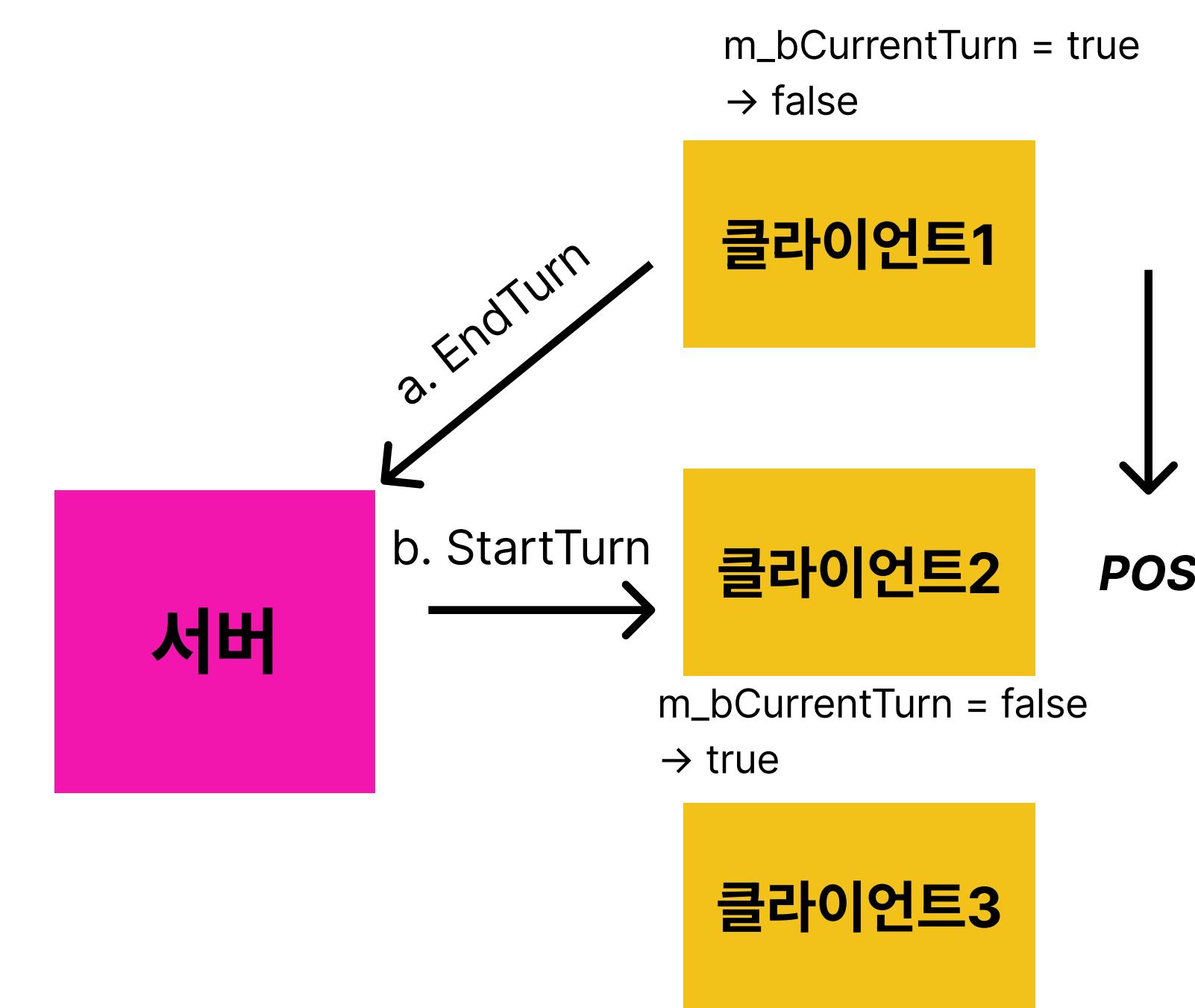
m_bCurrentTurn : 내 차례인지 여부를 알려주는 bool형 변수. 플레이어가 각각 가지고 있다.

서버는 **m_clientSocketList**에 포함되어있지 않으므로 세가지로 나누어 구현이 진행된다.

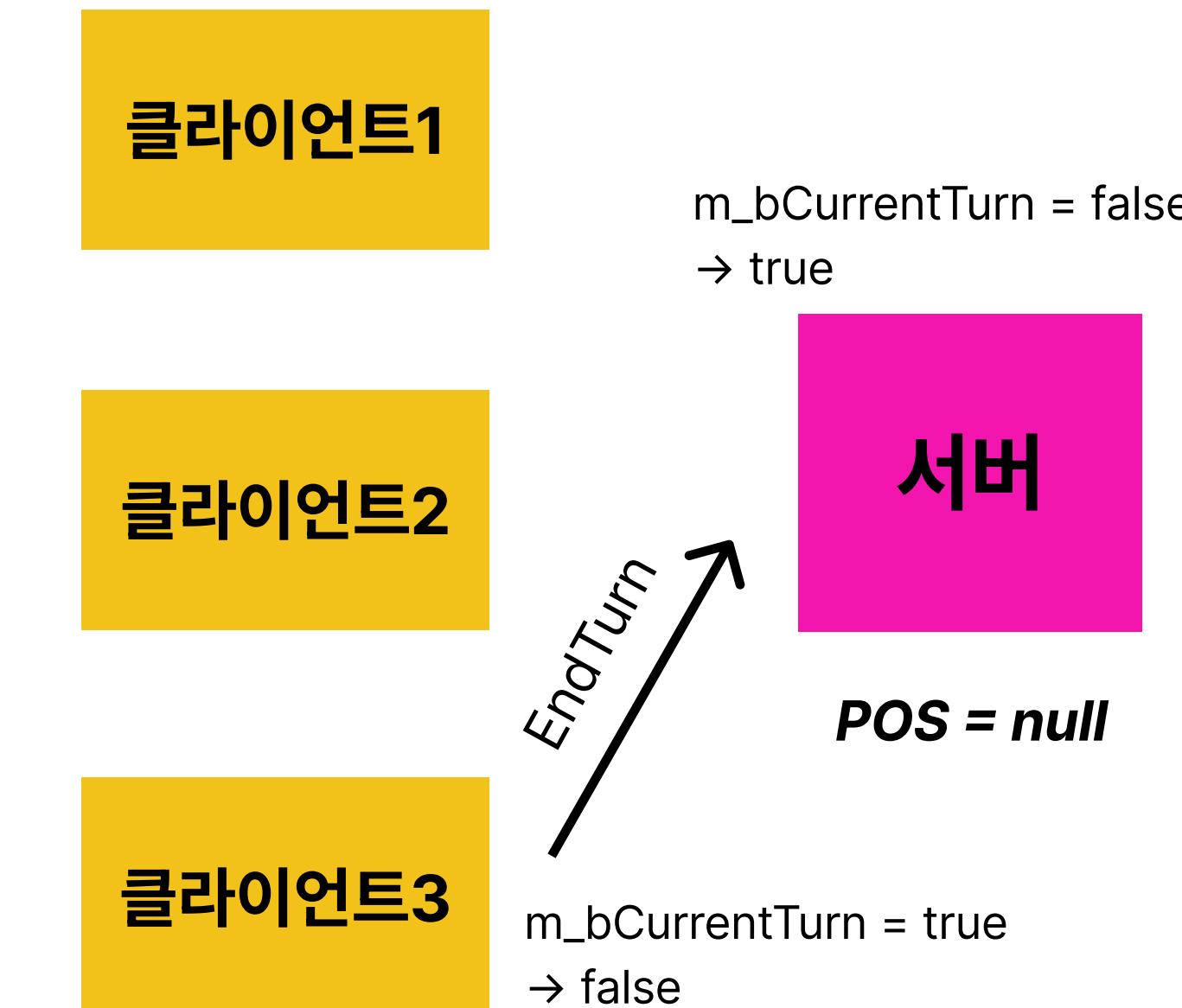
1. 서버 → 클라이언트



2. 클라이언트 → 클라이언트



3. 클라이언트 → 서버



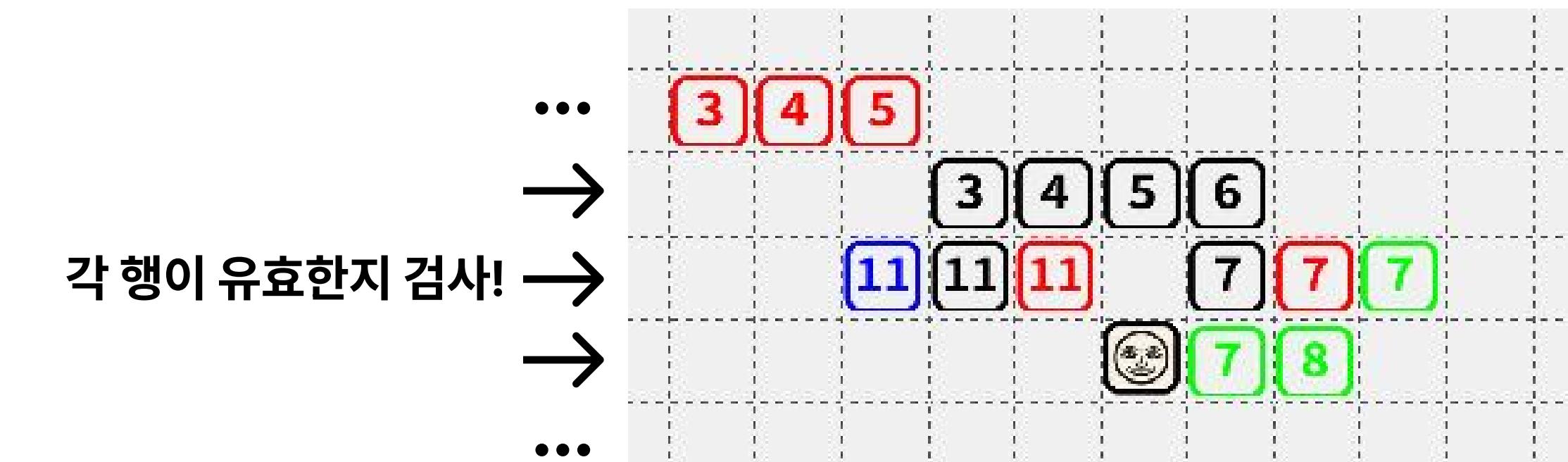
주요 코드 소개 : 공용판 검사 (타일 등록 원칙 검사)

1. 공용판 순회

공용판의 각 행(1~13행)에 대해 반복문을 돌며 IsRowValid를 호출한다.

IsRowValid 는 각 행이 유효한지 검사한다.

또한 첫 제출(등록)인 경우 새로 낸 타일들의 점수 총합이 30 이상인지도 추가로 검사한다.

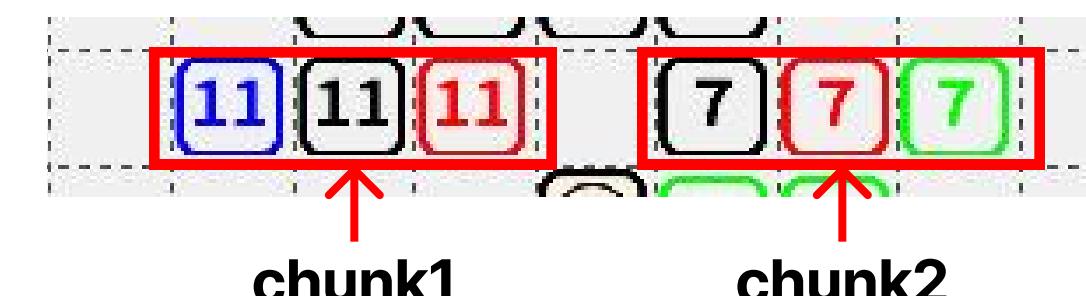


2. 행에서 청크 추출 (IsRowValid)

IsRowValid에서는 하나의 행에 놓인 타일들을 인접한 타일들의 묶음인 청크로 파싱하여 검사한다.

즉, 해당 행의 모든 칸을 순회하며 타일을 tileChunk 리스트에 담는다.

이후 IsRunValid, IsGroupValid를 각각 호출하여 tileChunk 리스트의 각 청크가 유효한지 검사한다.



주요 코드 소개 : 공용판 검사 (타일 등록 원칙 검사)

3. Run 검사 (IsRunValid)

Run은 최소 3개 이상, 같은 색상, 연속된 숫자의 타일들로 구성된 청크이다.

IsRunValid에서는 다음과 같은 알고리즘으로 해당 청크가 유효한 run인지 검사한다.

1. 조커가 아닌 첫 번째 타일의 색상을 기준 색으로 잡는다.
2. 각 타일에 대해 일반 타일인지 조커인지에 따라 조건 검사를 수행한다.
 - 일반 타일: 기준 색과 같은지 확인하고, 값이 이전 숫자 + 1인지 확인한다.
 - 조커: 어떤 숫자든 될 수 있으므로 패스한다.



4. Group 검사 (IsGroupValid)

Group은 최소 3개~최대 4개, 같은 숫자, 서로 다른 색상으로 구성된 청크이다.

IsGroupValid에서는 다음과 같은 알고리즘으로 해당 청크가 유효한 group인지 검사한다.



1. 조커가 아닌 첫 번째 타일의 숫자를 기준 숫자로 잡는다.
2. 중복/일치 검사:
 - 모든 타일이 기준 숫자와 같은지 확인한다.
 - 이미 나온 색상이 또 나오는지(중복 색상) 확인한다.

주요 코드 소개 : Setback

자기 차례에 타일을 등록하기 위해 공용판의 타일을 움직이면서 여러가지 시도를 해볼 수 있다.
그런데 그 시도가 실패해서 턴을 그냥 넘겨야 하는 상황이 오면? ⇒ 턴 시작 때의 정상적인 상태로 되돌려야 한다!

가장 중요한 사전 작업 : 턴 시작 때의 게임판의 상태를 저장해둬야 한다!

주요 코드 소개 : Setback

개인판, 공용판에 대한 백업 배열을 하나 더 정의해둔다

Tile m_old_public_tile[14][28] : set back을 위해 턴이 시작될 때의 공용판을 백업

Tile m_old_private_tile[4][18] : set back을 위해 턴이 시작될 때의 개인판을 백업

턴이 시작될 때마다 백업을 진행한다 ⇒ 서버의 턴을 관리하는 NextTurn() 함수에서 진행

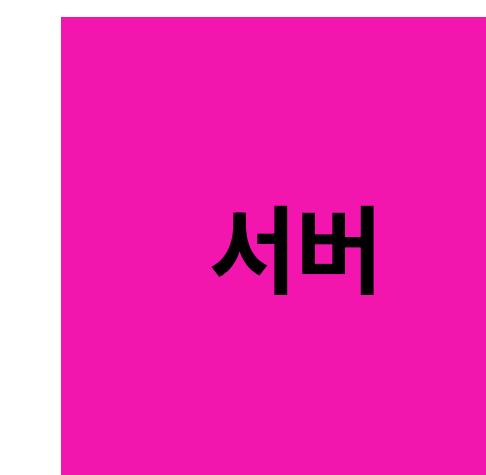
현재 차례인 사람 : 개인판과 공용판 모두 백업

현재 차례가 아닌 사람 : 공용판만 백업

메시지 흐름 (2인 게임으로 가정)

- 턴이 시작되면

1. 서버 플레이어 백업



3. Backup 메시지 수신 후 클라이언트 플레이어 백업 진행



2. Backup 메시지



주요 코드 소개 : Setback

현재 차례인 사람이 Setback 버튼을 누르면? ⇒ 백업 배열의 내용을 현재 게임판 배열에 복사한다!

현재 차례인 사람 : 개인판과 공용판 모두 Setback

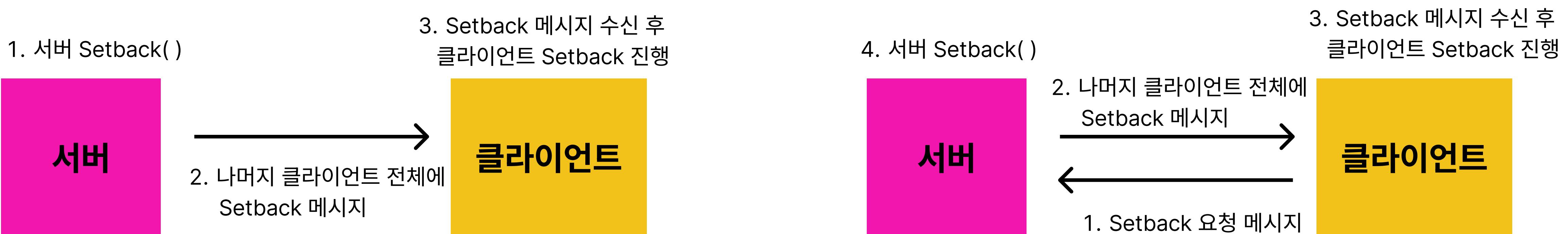
현재 차례가 아닌 사람 : 공용판만 Setback

메시지 흐름 (2인 게임으로 가정)

- 서버의 차례일 때



- 클라이언트의 차례일 때



주요 코드 소개 : Pass

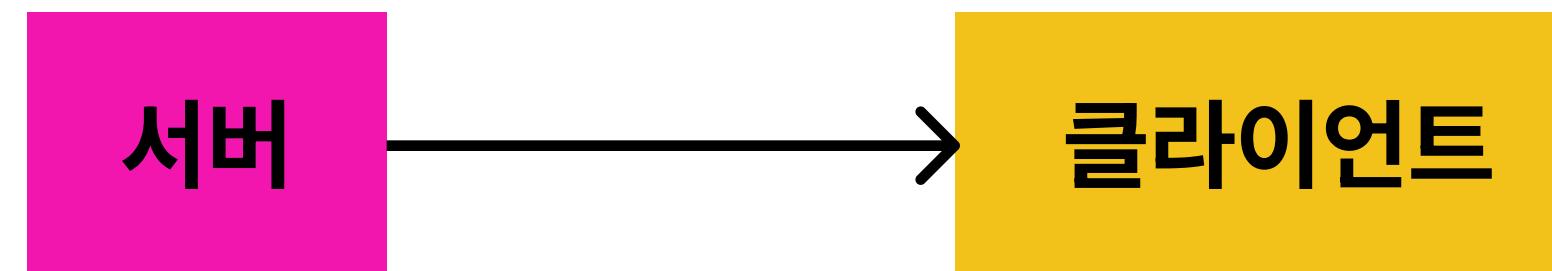
다음 플레이어에게 턴을 넘기는 Pass 버튼

Pass 버튼을 누르면 공용판 조건 검사 이후 턴을 넘긴다.

이때 타일 개수 최신화를 위해, 다음과 같이 서버와 클라이언트의 동작이 다르다.

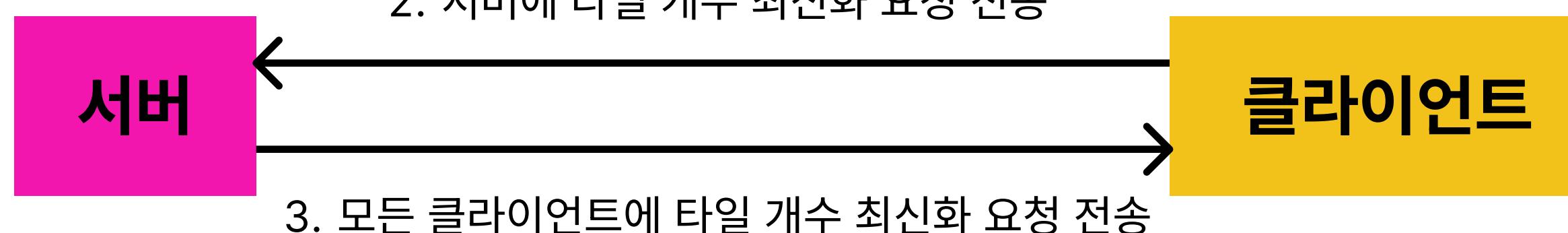
- 서버에서 pass를 한 경우

1. 자신의 공용판 조건 검사
2. 모든 클라이언트에 타일 개수 최신화 요청 전송
3. 타일 개수 최신화
4. 다음 플레이어에게 턴 넘기기



- 클라이언트에서 pass를 한 경우

1. 자신의 공용판 조건 검사
2. 타일 개수 최신화
3. 다음 플레이어에게 턴 넘기기



주요 코드 소개 : Receive

자기 차례에 1장의 타일도 내지 못하게 된다면, 덱에서 타일을 하나 가져온 후, 차례를 넘겨야 한다.

Receive에서 타일을 1장 가져오는 기능만 구현하게 된다면 여러 번거로움이 생긴다.

1. 차례 넘기기 : Receive 버튼을 눌렀다는 것은 차례를 넘기겠다는 의사와 같은데 Pass 버튼을 또 눌러야한다.
2. Setback : 타일을 이리저리 움직여보다 실패해서 차례를 그냥 넘겨야한다. Receive 버튼을 눌러 패를 받고, Setback 버튼을 따로 누르고, Pass 버튼을 눌러서 차례를 넘겨줘야한다.

따라서 Receive 버튼을 눌렀을 때 다음 행동을 한번에 진행한다!

Setback ⇒ 타일 한 장 받기 ⇒ Pass

자기 차례에 타일 등록에 실패한 경우, Receive 버튼 한번에 처리할 수 있다.

주요 코드 소개 : Receive

타일을 개인판에 한 장 가져오는 로직

1. 자신의 턴인지 & 개인판이 꽉 찼는지 검사 (개인판의 현재 타일 개수 추적하는 변수를 가지고 검사)

```
if (m_bCurrentTurn == true && m_intPrivateTileNum!=51) (개인판 용량은 51칸)
```

⇒ 둘 중 하나라도 거짓이면 Receive 불가능 상태로 판정 ⇒ 함수 진행을 막는다.

2. 개인판을 대상으로 반복문을 순회하며 비어있는 공간을 찾기

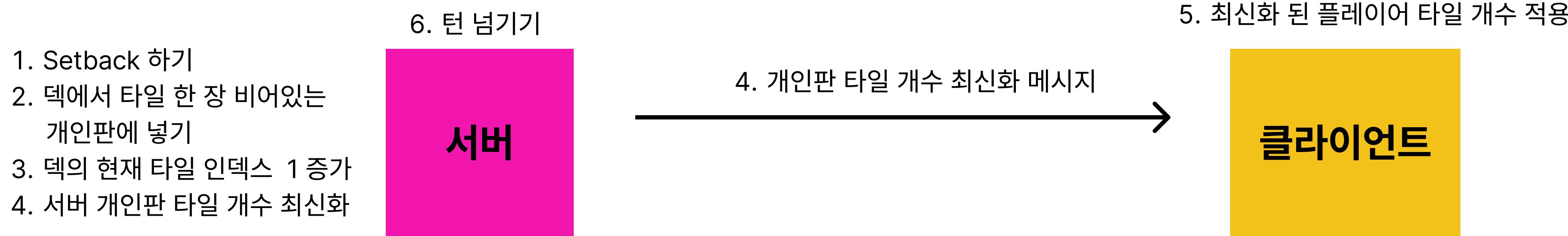
(Tile 구조체는 tileid 변수를 -1로 초기화 해뒀기에 tileid가 -1 이면 비어있는 공간으로 인식)

3. 비어있는 공간에 타일 한 장 넣기 + 현재 데크의 인덱스를 1 올린다.

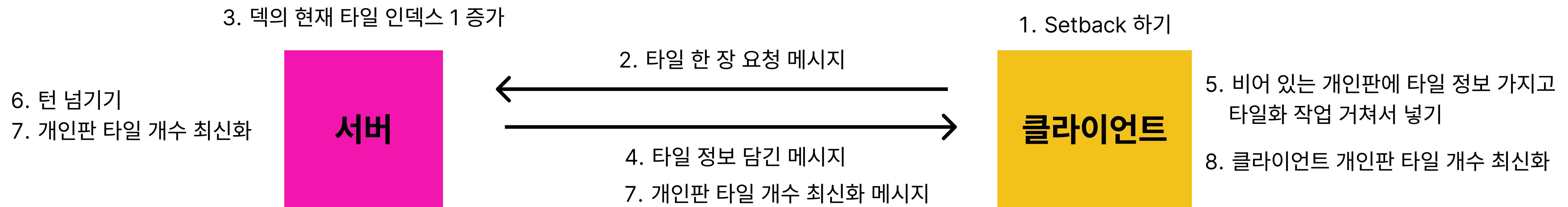
주요 코드 소개 : Receive

메시지 흐름 (2인 게임으로 가정, Setback의 메시지 흐름은 이미 앞에서 설명했기에 생략)

- 서버의 차례일 때



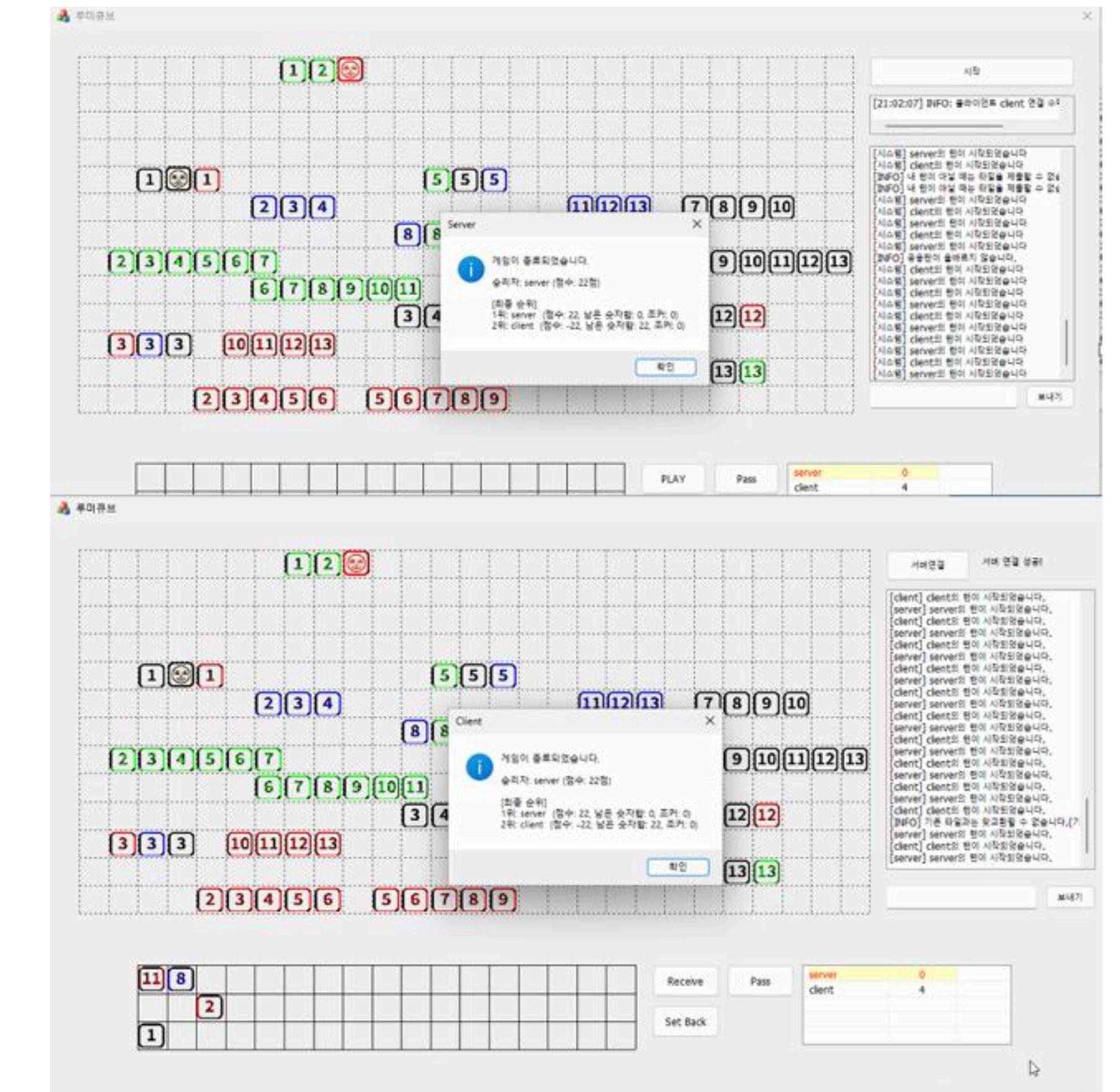
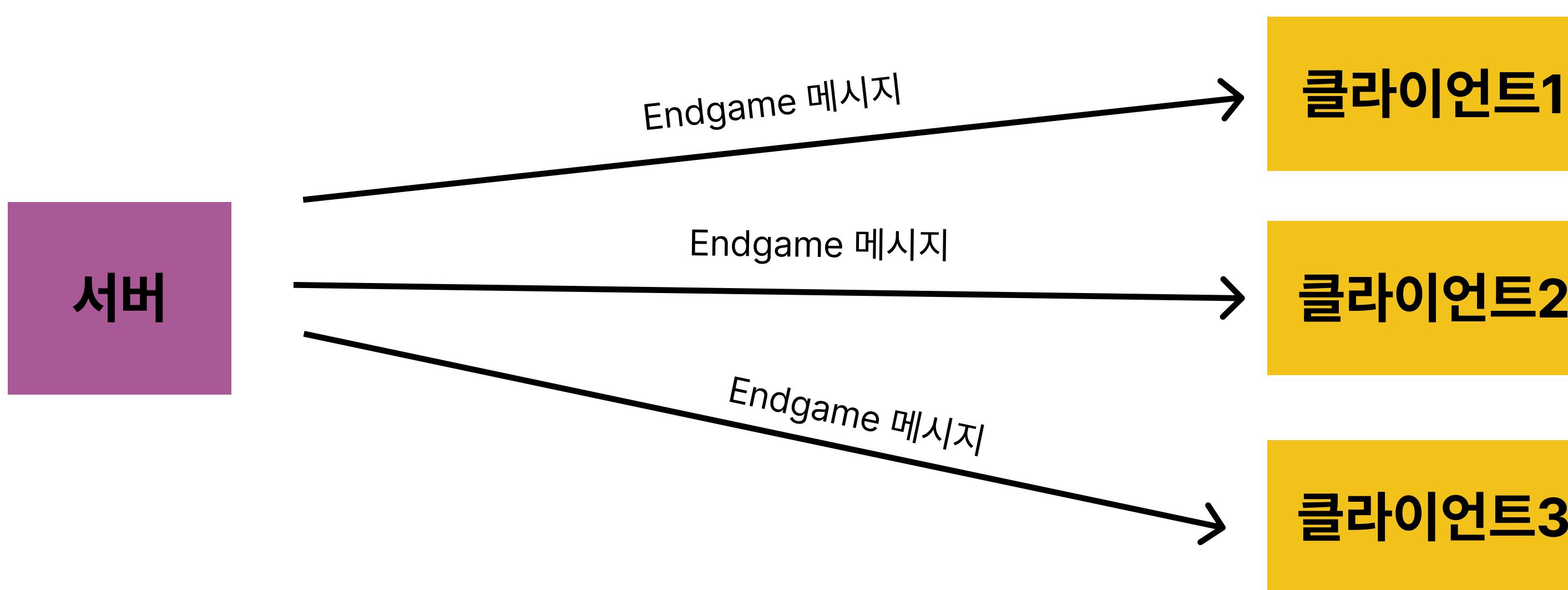
- 클라이언트의 차례일 때



주요 코드 소개 : 게임 종료

정상적인 종료 - 누군가 모든 타일을 소진하면 게임이 종료된다!

누군가 타일을 모두 낸 후 Pass 버튼을 누르면 서버에서 모든 플레이어들에게 결과 대화상을 띄우는 메시지를 보낸다. 이후 게임은 자동으로 종료되고 모든 플레이어들의 게임창도 꺼진다.



주요 코드 소개 : 게임 종료

비정상적인 종료 - 게임 중 특정 플레이어가 나가는 경우

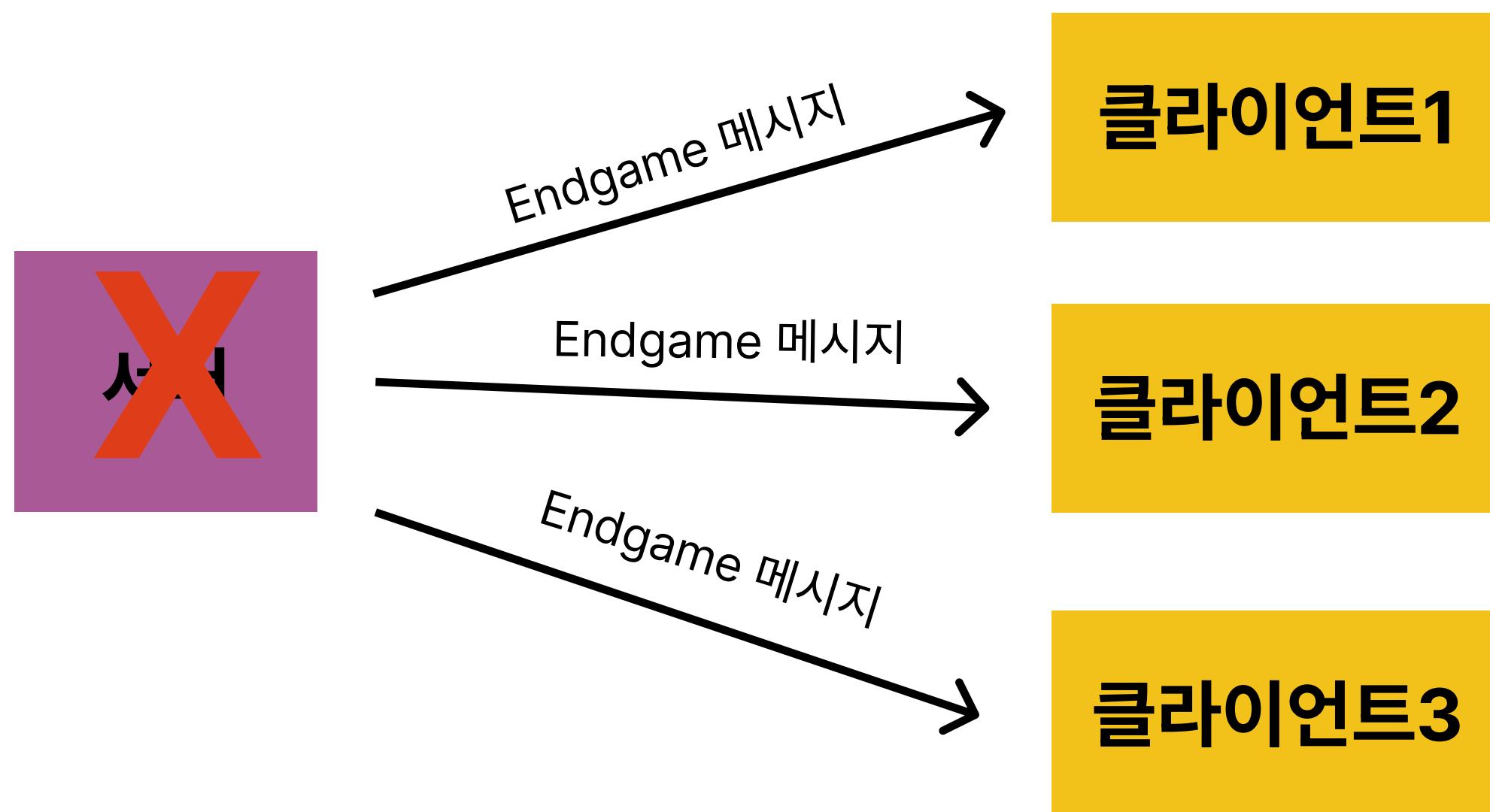
게임 시작 전 대기방에서 나가는 것은 문제가 되지 않는다.

그러나 **게임 중간에 플레이어가 나가면** 더이상 게임을 진행할 수 없다.

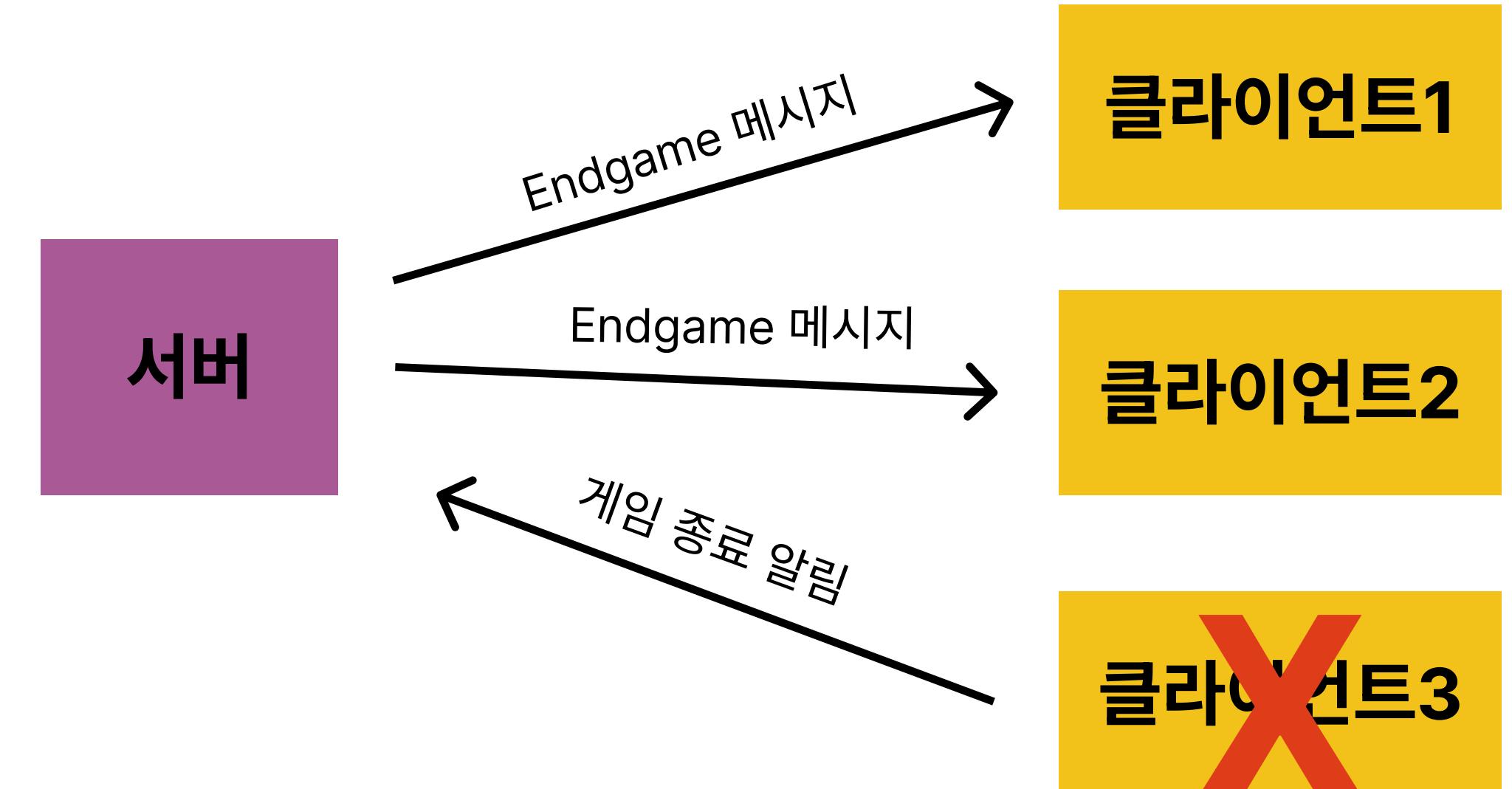
이 경우 전체 플레이어의 게임을 강제 종료한다.

대화상을 닫으면 실행되는 **Onclose()** 함수 ⇒ **RemoveClient()** 호출, 게임 종료 로직 진행

- 서버가 나간 경우 :



- 클라이언트가 나간 경우 :



주요 코드 소개 : 타일 옮기기

타일 선택 및 이동 : OnLButtonDown()

OnLButtonDown()에서 사용자의 마우스 왼쪽 클릭 이벤트가 발생했을 때 타일을 이동하는 기능을 구현했다.

마우스 클릭으로 타일을 선택한 뒤 다른 타일을 선택하면 타일을 교환하도록, 빈 칸을 선택하면 해당 칸으로 이동하도록 구현했다.

타일 이동 예외 처리

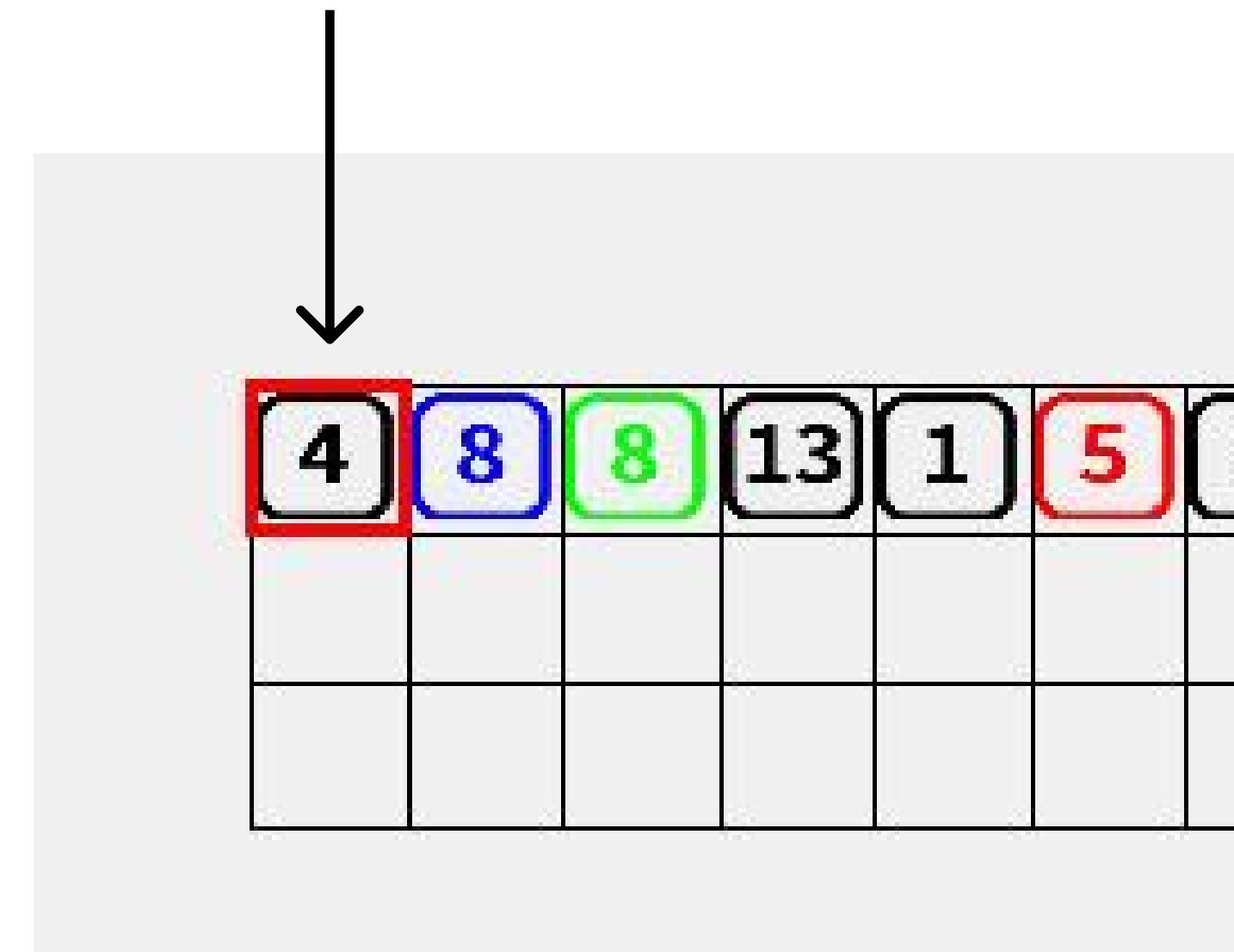
타일 이동과 관련하여 다음과 같은 상황에 대한 예외 처리를 적용했다.

1. 자신이 내지 않은 타일은 공용판에서 개인판으로 가져올 수 없다.

→ setback에서 사용했던 기존 타일 저장 배열을 활용해 기존에 공용판에 올라와 있던 타일인지 판단했다.

2. 자신의 턴이 아니면 공용판 타일을 이동할 수 없다.

타일을 선택하면 빨간색 테두리가 쳐진다.
옮기고 싶은 곳을 선택하면 타일이 옮겨진다.
다른 타일을 선택하면 그 타일과 교환된다.
빈 칸은 먼저 선택 불가능하게 설정!

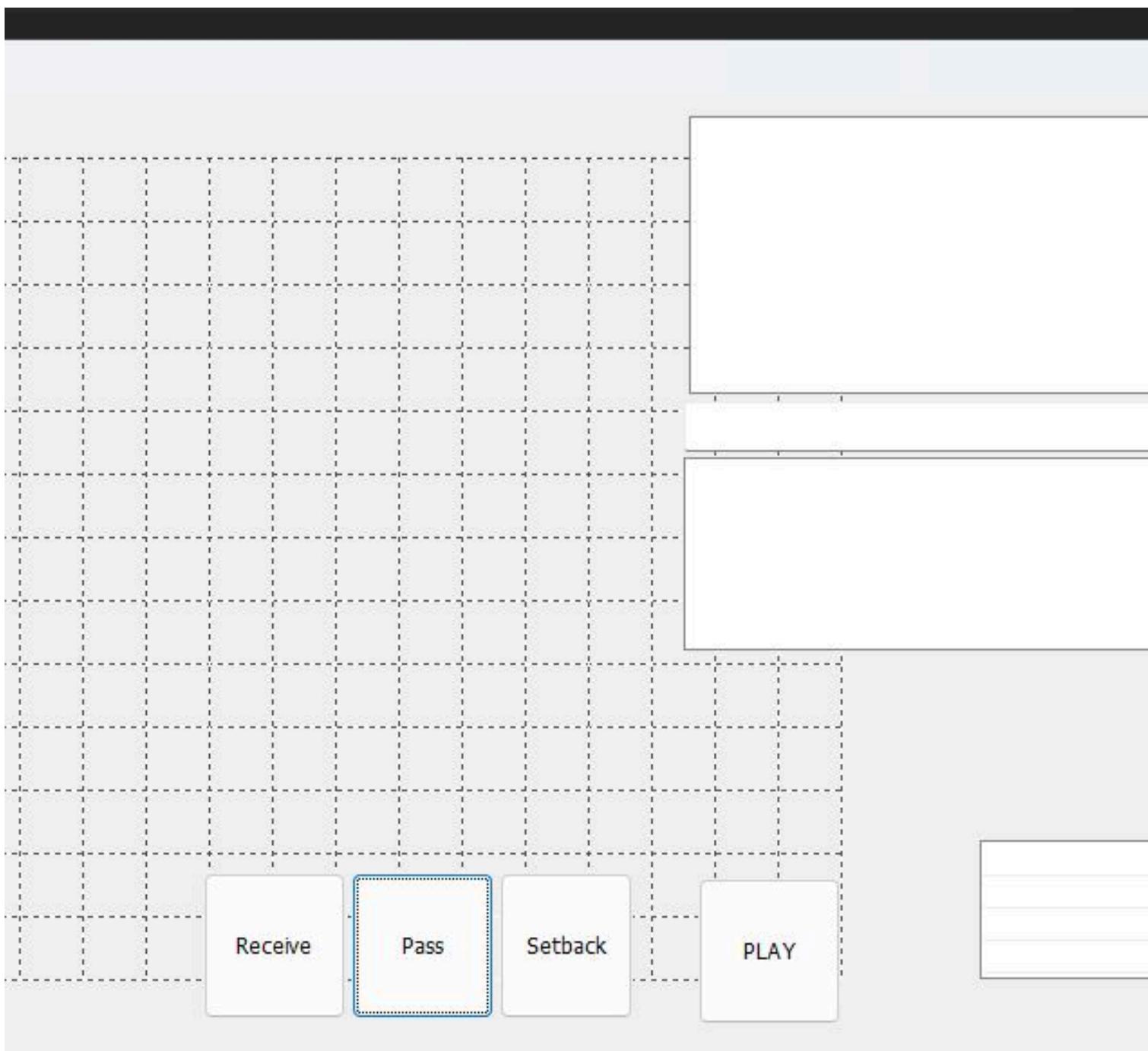


예외 처리 사항

노트북, PC환경 (display)에 따른 창크기 조절

처음에는 공용판/개인판을 고정된 픽셀 값으로 그렸었는데, 다른 컨트롤들은 디스플레이 크기 및 배율에 따라 동적으로 위치가 정해져서 공용판과 겹치는 문제가 있었다.

→ 다른 컨트롤들과 같이 공용판/개인판 위치를 동적으로 정하도록 수정했다.



예외 처리 사항

게임 진행 중 탈주자 발생

게임을 진행하다가 탈주자가 생기는 경우에 대한 처리는 게임 시작 여부에 따라 구현했다.

- 게임을 이미 시작했는데 탈주자가 발생한 경우, 탈주 메시지를 출력하고 모든 플레이어의 다이얼로그를 종료하는 것으로 구현했다.
→ 탈주한 다이얼로그의 OnClose()에서 탈주 메시지를 전송하도록 했다.
- 게임을 아직 시작하지 않은 경우, 소켓 리스트 및 리스트 컨트롤에서 제거하는 것으로 구현했다.

모달 대화상자와 비동기 소켓 충돌 예외

처음엔 게임 시작, 턴 시작, 잘못된 동작 알림 등을 메시지 박스로 띄웠었는데, 메시지 박스가 떠 있을 때 메인 메시지 루프가 정지되면서 서버/클라이언트 간에 전송한 메시지가 처리되지 않거나 액세스 오류가 발생하는 문제가 있었다.
→ 게임 집중에 방해되기도 하고, 이런 문제도 있어서 메시지 박스 대신 로그에 뜨도록 수정했다.

예외 처리 사항

그 외의 처리 사항들

추가로 다음과 같은 예외 사항들을 해결했다.

- 로그창/플레이어 리스트 가로 길이 문제
- 개인판이 꽉 찼을 때 Receive 동작 예외
- 덱 소진 후에도 Receive가 계속 시도되는 문제
- Receive 버튼 연타 시 타일 여러 장 들어오는 버그
- SetBack 버튼이 동작하지 않던 문제
- Client2에서 Receive하면 타일이 Client1으로 들어가던 문제
- 게임 시작 전 / 내 턴이 아닐 때 버튼 눌렀을 때 무반응하도록 수정
- 아무것도 안 냈는데 Pass가 되는 문제
- 채팅 입력 시 Enter/ESC 때문에 게임이 종료되는 문제
- 모달 대화상자 띄운 상태에서 턴이 넘어가면 터지던 문제
- 첫 제출 30점·남의 타일 섞기 금지 규칙 미반영
- 조커를 1 왼쪽 또는 13 오른쪽에 둘 수 있던 문제
- 이전 턴 공용판 타일을 교환 방식으로 개인판으로 가져올 수 있는 버그
- 타일을 다 써도 게임이 끝나지 않던 문제 + 점수 합산 후 순위 측정
- 게임 중 서버가 꺼지거나 클라이언트가 게임 종료할 때 방이 애매하게 남는 문제
- 게임 시작 전 클라이언트 퇴장 시 리스트/소켓 상태가 어긋나는 문제
- 최대 플레이어 수(4명) 초과 접속 문제
- 플레이어별 타일 수를 UI에서 볼 수 없던 문제
- 타일 클릭 → 빙칸 클릭과 빙칸 클릭 → 타일 클릭에서 제출 개수 계산이 달랐던 문제
- ...



루미큐브 게임 플레이 시연