Meta

개발 인원	3명	본인 역할	팀장(서버)	개발 기간	3주
개발 언어	C#				
사용 툴	Visual Studio 2022, Unity, Oracle DB				

1. 개발 목적

메타버스에 알맞는 게임을 만들고 싶었습니다.

플레이어가 자신의 방을 만들어 꾸미고, 다양한 게임을 함께 즐길 수 있는 공간을 만들어 보고 싶었습니다. 4팀중 클라이언트와 서버의 연결과 DB를 맡은 서버팀에서 작업하였습니다.

2. 기능 구현

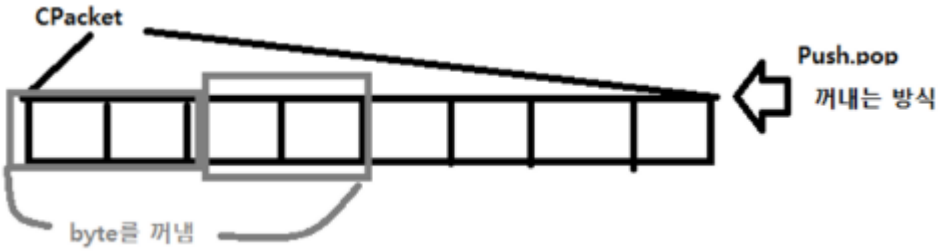
2.1. 서버/패킷

서버는 C#으로 구현하였습니다.

오픈소스인 FreeNet 서버를 수정하여 비동기 로직을 구현하였습니다.

패킷 처리 부분을 수정하여 Recv 처리 시 Task를 사용하여 Message를 처리하도록 구현하였습니다.

패킷은 Size와 명령어로 구성된 Header와 명령어에 필요한 변수들을 담은 Body로 구분하여 처리하였습니다.




```
C:₩WINDOWS₩system32₩cmr
[5] A client connected. handle 1136,  count 1
MServer.CGameUser Client Connected
Oracle Server 연결 성공
process send : Success, transferred 8, sent count 1
process send : Success, transferred 9, sent count 2
process send : Success, transferred 51, sent count 3
process send : Success, transferred 51, sent count 4
process send : Success, transferred 51, sent count 5
process send : Success, transferred 51, sent count 6
process send : Success, transferred 51, sent count 7
process send : Success, transferred 51, sent count 8
process send : Success, transferred 51, sent count 9
process send : Success, transferred 51, sent count 10
```

2.2. DB

클라이언트에서 저장해야하는 정보를 설계하여 DB Table을 구현하였습니다. 서버에서 사용하기 편하게 DAO를 만들어 제공하였습니다. Oracle 연결을 생성과 삭제 시 DB와 서버의 부담이 늘기 때문에 Connection Pool을 만들어 연결을 재사용 할 수 있도록 만들었습니다.

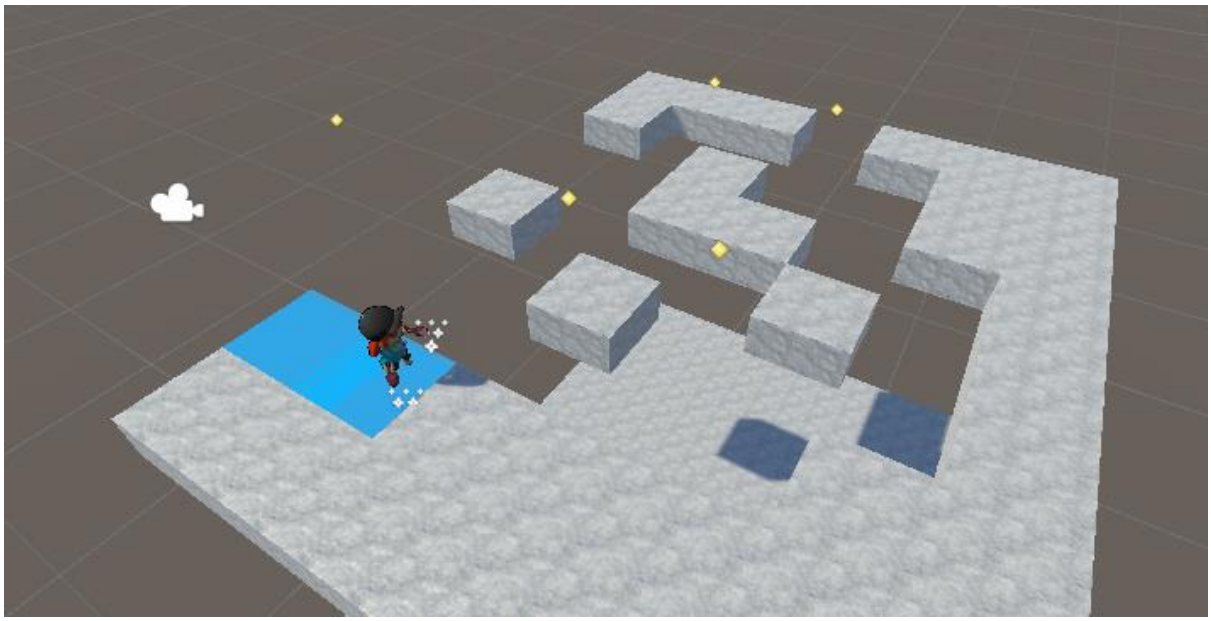
```
참조 0개
public CharacterVo GetCharacterData(string inuserid)
{
    CharacterVo characterVo = new CharacterVo();

    string sql = @"SELECT * FROM character WHERE user_id = :inuserid";

    // 커넥션풀에서 오라클연결 객체를 대여
    OracleConnection conn = connPool.GetConnection();

    try
    {
        OracleCommand cmd = new OracleCommand();
        cmd.Connection = conn;
        cmd.CommandText = sql;
        cmd.Parameters.Add("inuserid", inuserid);

        OracleDataReader reader = cmd.ExecuteReader();
        if (reader.Read())
        {
            characterVo.CHAR_NO = reader["char_no"].ToString();
            characterVo.COIN = reader["coin"].ToString();
            characterVo.USER_ID = reader["user_id"].ToString();
            characterVo.INVENT = reader["invent"].ToString();
        }

    }
```

2.3. 클라이언트 연동

2.3.1. 미니 게임 연동

2.3.1.1. ICE BREAK

발판이 깨질때 다른 플레이어들의 발판도 깨지도록 구현하였습니다. 위치값을 저장해두고 플레이어가 발판에 올라갈 시 서버에 위치값을 전달하여 다른 플레이어들의 발판에서 같은 위치에 있는 발판을 지워줄 수 있도록 구현하였습니다.



2.3.1.2. SHOOTING

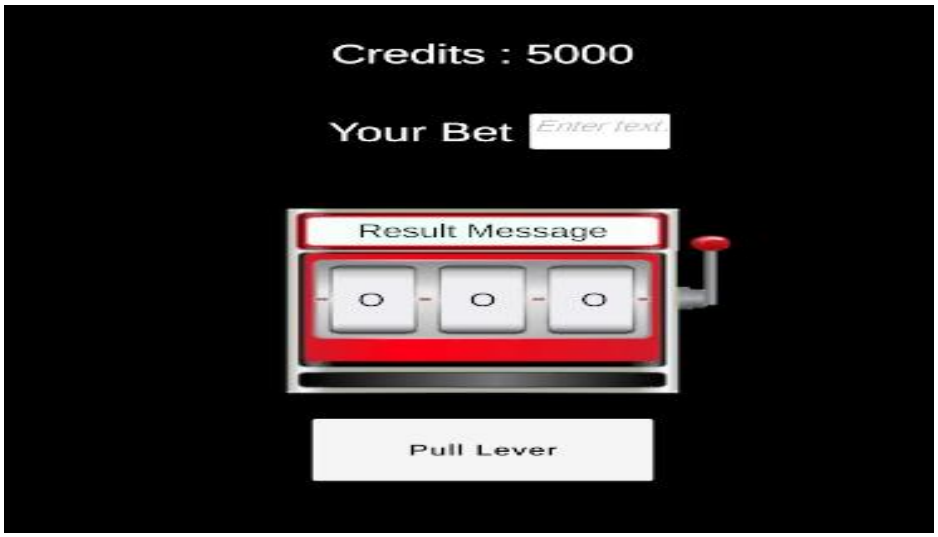
총알을 연동시켜 플레이어가 상대방을 쏘았을때 같은 위치에서 총알이 나갈 수 있도록 구현하였습니다. RayCast를 사용하여 총알을 구현하였는데, RayCast 시 원점 좌표와 방향 벡터를 서버에 보내 동기화 시켰습니다.

2.3.1.3. SKATING

플레이어의 위치를 동기화 시켜 다른 플레이어의 모습이 보여지게 구현하였습니다. 플레이어의 위치뿐만 아니라, 플레이어의 현재 에니메이션, 회전값도 같이 동기화를 시켜 더 실감나는 게임을 즐길 수 있도록 구현하였습니다.

2.3.2. 생활 컨텐츠 연동

플레이어가 가진 돈을 동기화 시켰습니다. 플레이어는 생활/카지노 등의 행동에서 돈을 벌거나 잃는 부분을 동기화 해 플레이어가 게임을 다시 접속했을 때에도 플레이어의 돈이 유지되도록 구현하였습니다.



2.3.3. 마이룸 연동

플레이어가 설치한 오브젝트의 정보를 리스트로 저장해두고 Json을 사용하여 리스트를 String으로 변환 후 데이터를 서버에 저장하였습니다. 다시 접속하여도 자신의 마이룸 정보는 저장이 되고, 다른 플레이어가 상대방의 마이룸에 접속 시 오브젝트를 불러올 수 있도록 구현하였습니다.