

Team Blue Bird

Poor Guys Project

도전! 폴가이즈 클론 프로젝트



목차

01

소개



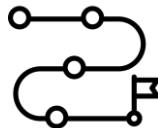
02

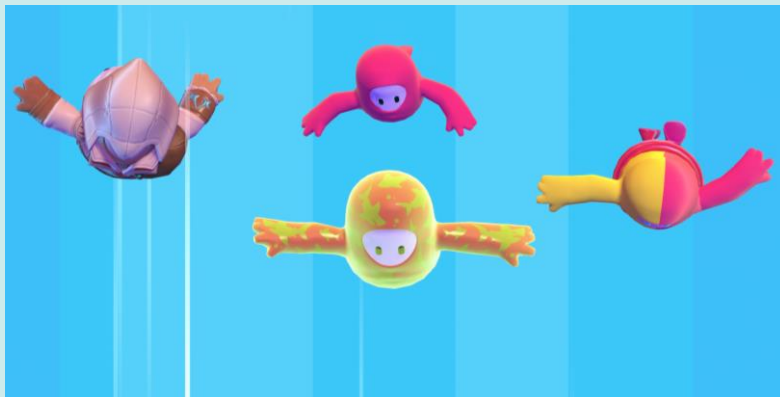
프로젝트



03

에필로그





TEAM BLUEBIRD

소개

팀 소개



Team **Blue Bird**

자랑스러운 팀원들,
나아가 캠프에 참여하는 청춘들,
우리 모두가 각자의 목표를 향해
자유롭게 날아갔으면 하는 마음

팀 소개

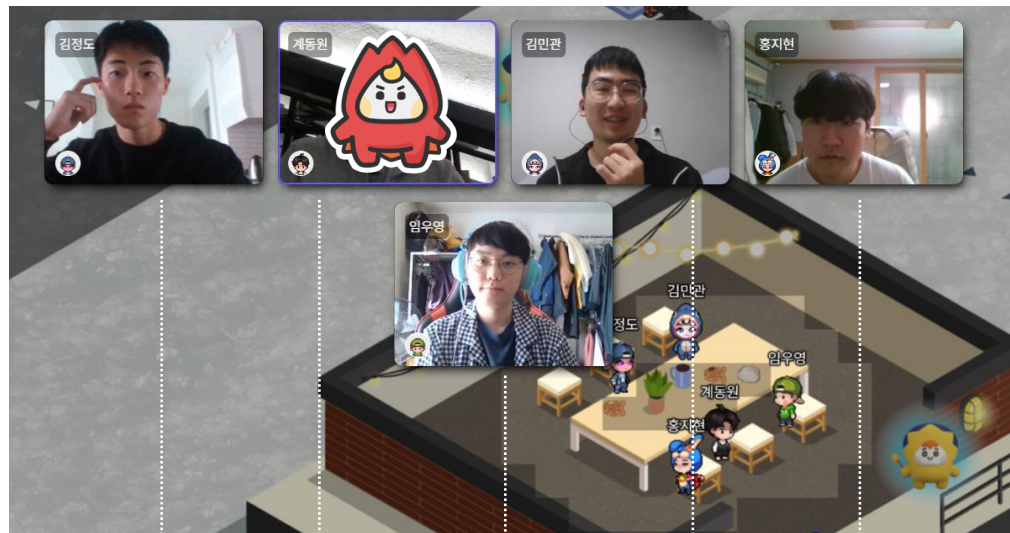
**POOR
GUYS**

PROJECT **Poor Guys**

Poor (가난한) + Guys (사람들)

학교 생활은 끝나가고
그렇다고 취업을 한 것도 아닌
가난한 우리들의 모습

팀원 소개



김정도



캠프장님



임우영



김민관



홍지현

프로젝트 선정



게임 개발이라는 공통된 관심사로 한 자리에 모인 팀원들
하지만 각자 선호하는 장르가 달라 의견을 모으기가 쉽지 않음

프로젝트 선정



재미있게 플레이한 경험이 있음

모두가 흥미롭게 개발할 수 있는 공통된 게임 분야

팀원 개개인의 도전 목표가 적절히 녹아있는 게임

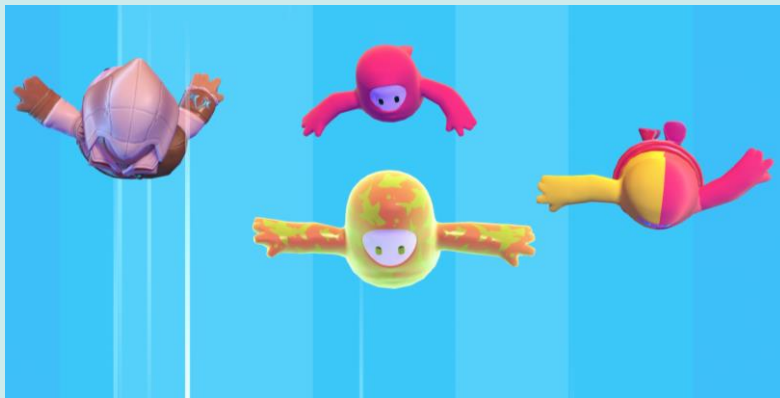
폴 가이즈 (Fall Guys)



- 남녀노소 모두가 즐길 수 있는
Unity 기반의 3D 플랫폼머 게임
- 최대 60명의 인원이 함께 플레이
하는 멀티 플레이 게임
- 토너먼트 방식의 경쟁 게임
- 자동 매칭 시스템을 통해,
쉽게 게임 참여 가능

시연 영상

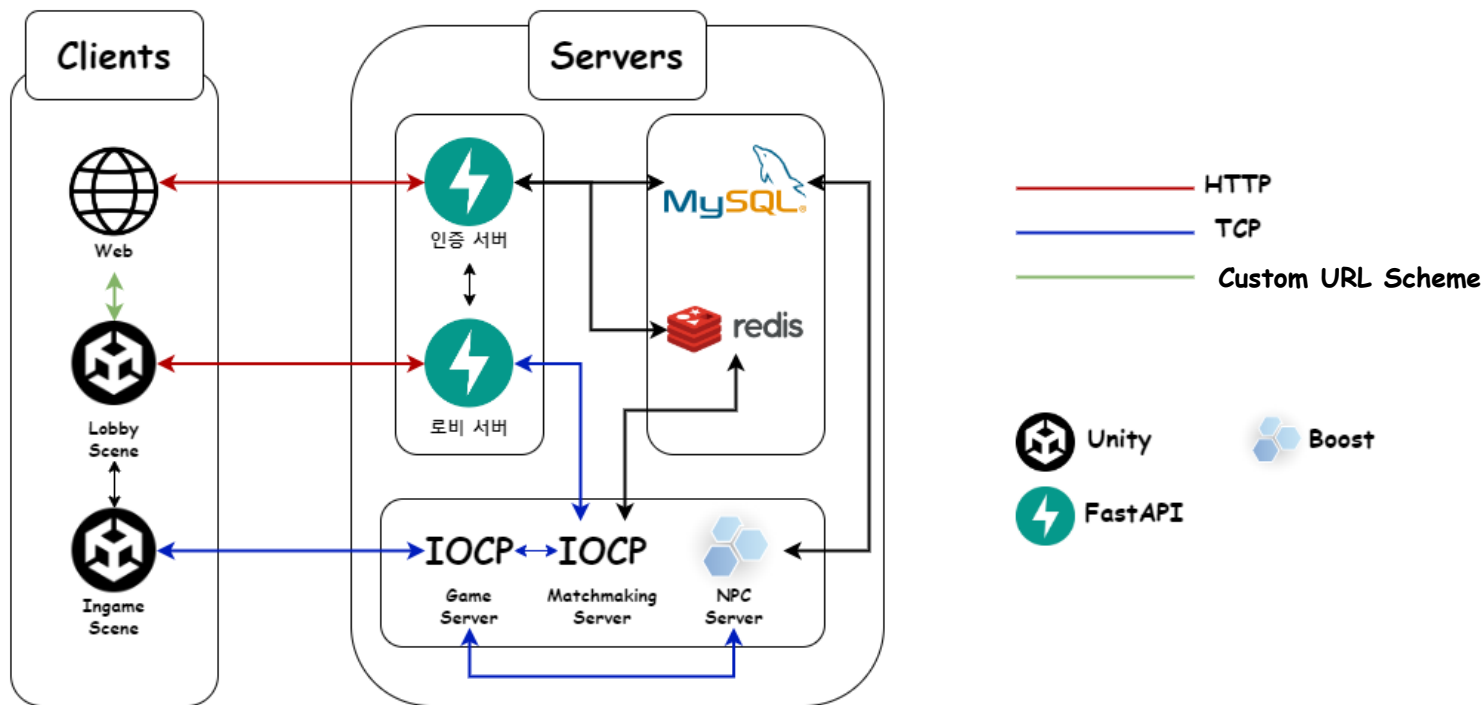




TEAM BLUEBIRD

프로젝트

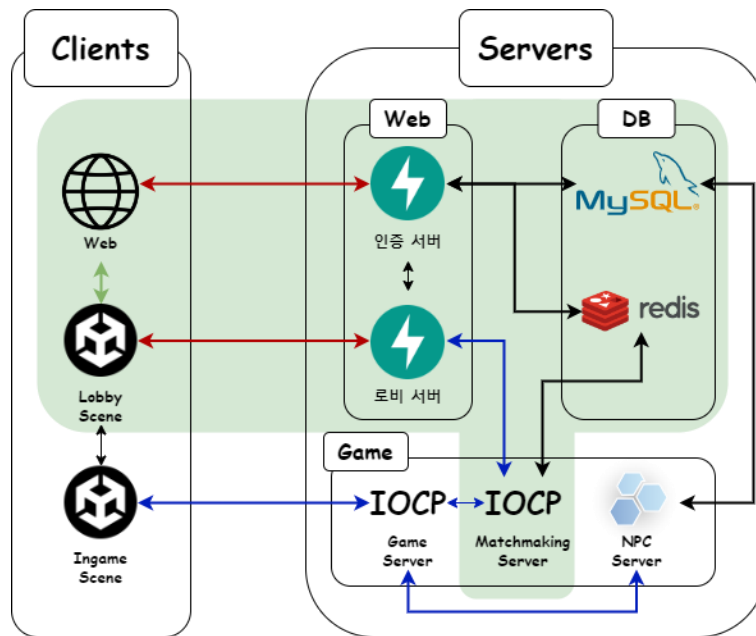
아키텍처 설계





WEB PLATFORM | LOBBY SERVER

김정도





WEB PLATFORM | LOBBY SERVER

김정도



01 Web Frontend

- 로그인 페이지와 회원가입 페이지
- 설치 파일 배포
- Custom URL Scheme로 게임 클라이언트 등록

02 Auth Server

- 회원가입 및 DB 운용 (MySQL)
- 로그인 및 JWT 인증
- Redis를 통한 접속 상태 관리

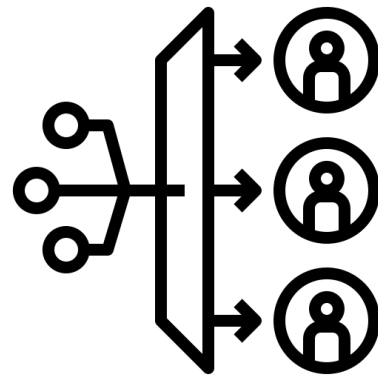
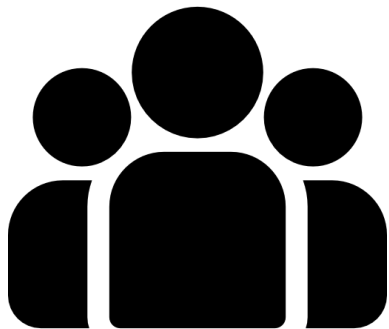
03 Lobby Server

- 게임 클라이언트의 게임 매칭 요청 처리
- 매치메이킹 서버의 매칭 결과 저장
- HTTP, TCP, Thread, ProtocolBuf



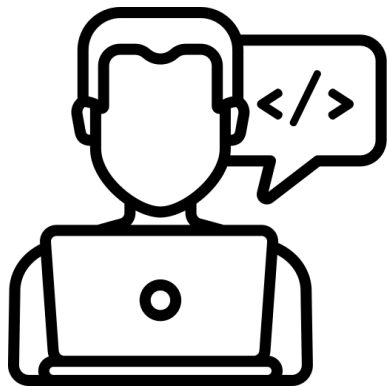
웹 게임 플랫폼

➤ 게임 플랫폼



게임을 플레이하는 유저들에 대한 정보를 관리하고
서비스를 제공하기 위해서는 게임 플랫폼이 필요함

웹 게임 플랫폼



웹 백엔드 개발자로서 진로와 게임 개발 프로젝트 사이에서 얻을 수 있는 것

웹 게임 플랫폼

➤ 웹 게임 플랫폼




메이플스토리가 웹에서 회원정보를 관리했던 것처럼

웹 클라이언트

➤ 이슈: Custom URL Scheme

R

What is 'Custom URL Scheme'?



A custom URL scheme is a protocol that allows applications to communicate with each other through URLs that are registered by developers to launch or trigger specific actions within their app.


For example, when you click on a link that starts with "mailto:", your email client. Similarly, when you click on a link that starts with "myapp://", your mobile device can launch your app, or trigger a specific action within the app.

Custom URL schemes are often used to create deep links, which point directly to a specific content or functionality within an app. They are commonly used in mobile apps, but can also be used in web applications and desktop applications.

Developers can register a custom URL scheme for their app by adding it to the app's configuration file, and then using that scheme in links on their website. This allows users to easily access specific content within the app with just a click of a link.

R

'커스텀 URL 스킴'이란?



커스텀 URL 스킴은 개발자가 등록한 URL을 통해 애플리케이션이 서로 통신하여 자체 응용 프로그램을 실행하거나 응용 프로그램 내에서 특정 작업을 트리거할 수 있도록 하는 프로토콜입니다.

예를 들어, "mailto:"로 시작하는 링크를 클릭하면 컴퓨터가 전자 메일 클라이언트를 엽니다. 마찬가지로 "myapp://"로 시작하는 링크를 클릭하면 모바일 장치가 앱을 실행하거나 앱 내에서 특정 작업을 트리거할 수 있습니다.

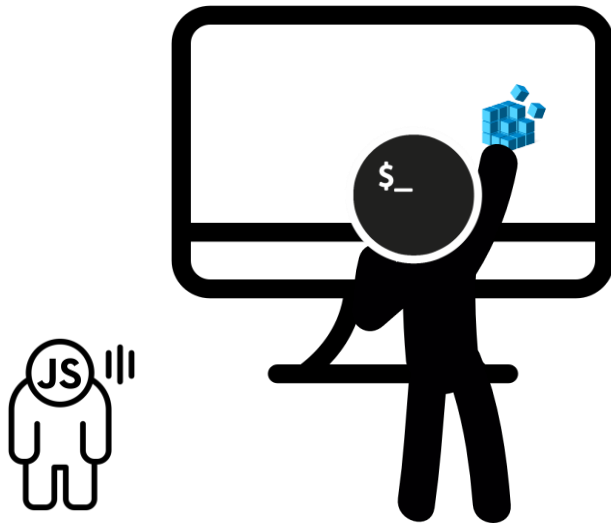
커스텀 URL 스킴은, 유저를 앱내의 특정의 콘텐츠나 기능에 직접 접속하는 링크인 딥 링크를 작성하는 경우에 자주 사용됩니다.일반적으로 모바일 앱에서 사용되지만 웹 애플리케이션 및 데스크톱 애플리케이션에서도 사용할 수 있습니다.

개발자는 앱의 구성 파일에서 스킴을 정의한 후 앱 내 또는 웹사이트 링크에서 스킴을 사용하여 앱에 대한 커스텀 URL 스킴을 등록할 수 있습니다.이를 통해 사용자는 링크 클릭 한 번으로 앱 내의 특정 콘텐츠나 기능에 쉽게 액세스할 수 있습니다.

웹 클라이언트가 데스크탑에 설치한 게임 클라이언트를 실행할 수 있는 방법이 필요

웹 클라이언트

➤ 이슈: Custom URL Scheme



자바스크립트로 클라이언트를 조작하기엔 제약 사항이 많음

설치 파일을 생성할 때 Custom URL Scheme를 레지스트리에 등록하는 코드를 추가하여 해결

웹 클라이언트

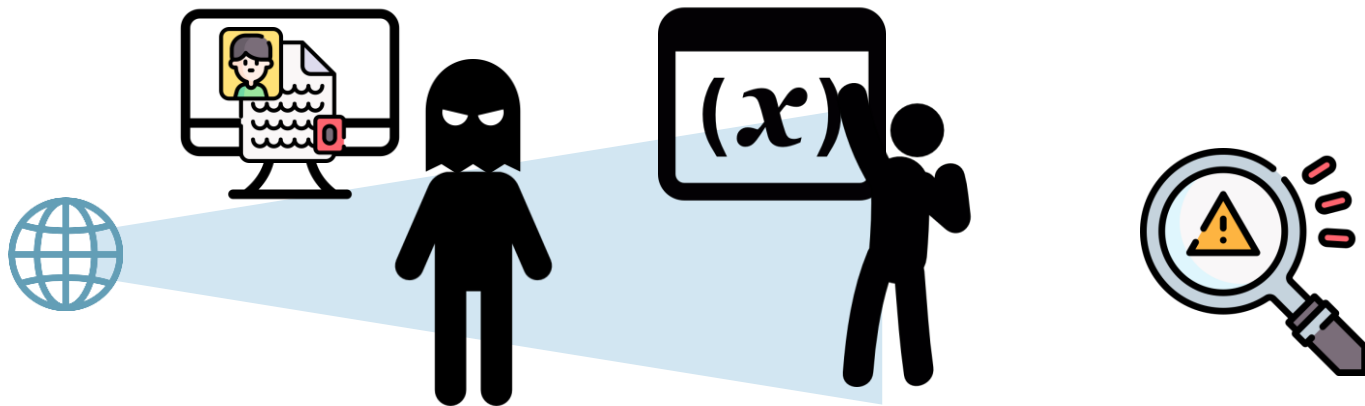
➤ 이슈: Custom URL Scheme



웹 클라이언트에서 로그인 한 유저가 게임 클라이언트로 연결될 때,
해당 유저 정보를 식별할 수 있는 정보를 전달할 방법이 필요함

웹 클라이언트

➤ 이슈: Custom URL Scheme



로그인할 때 유저 정보를 클라이언트 PC에 파일로 저장
(보안 문제)

Custom URL Scheme에 매개변수로 전달
(말끔히 해결)

느낀 점

부족한 점이 많지만 그래도 팀의 어엿한 일원으로서 프로그램 개발에 참여하고 기여할 수 있었다

그룹에도 불구하고 아직 가야할 길이 멀다

내가 가진 강점을 인지하고 또 보완할 수 있었던 귀중한 시간

팀원들과 앞으로도 좋은 동료가 되었으면

자신감

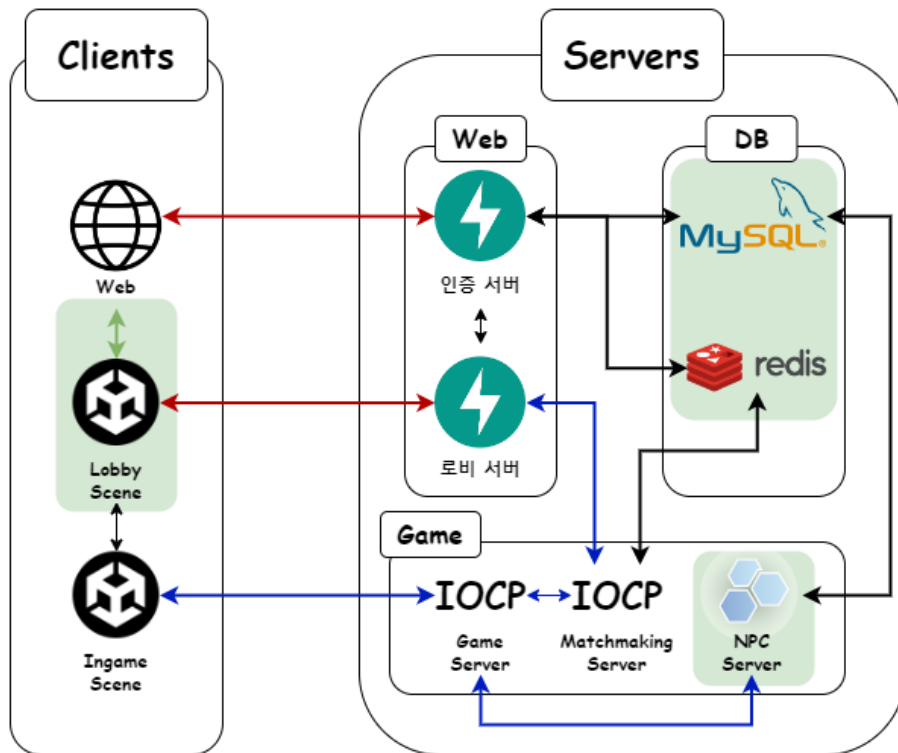
두려움

든든함





NPC 서버 | 로비 클라이언트 임우영





NPC 서버 | 로비 클라이언트

임우영



01 NPC 서버

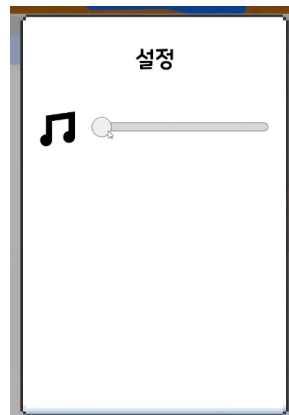
- MySQL로부터 장애물 정보 수신
- 게임 서버로 장애물 정보 송신
- 장애물 위치 및 상태 동기화 로직 구현

02 로비 클라이언트

- 로비 서버와 HTTP 통신
- 인게임 클라이언트에 현재 플레이어 정보 전달
- 매치메이킹 시작 및 취소 로직

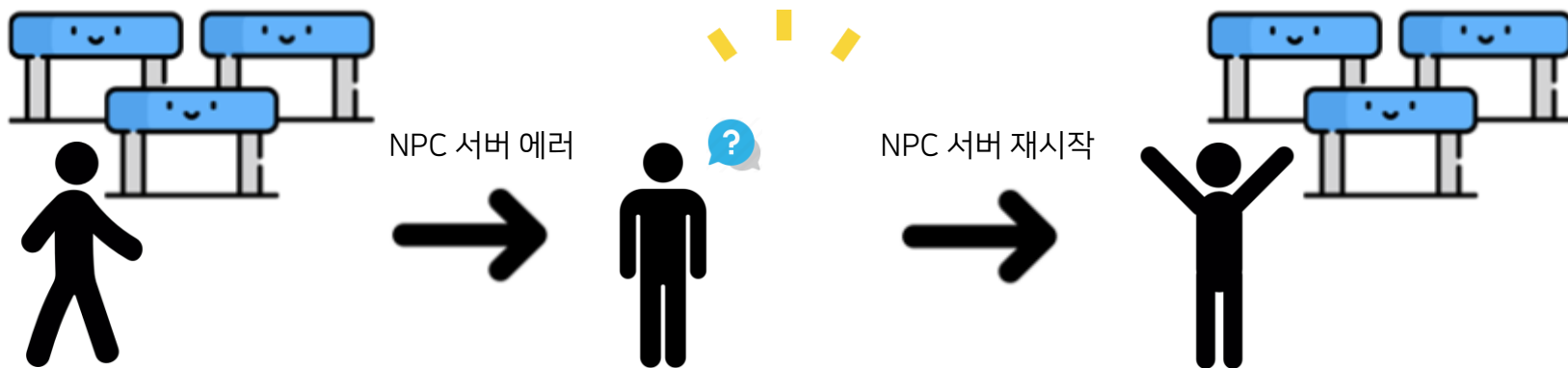
03 데이터베이스

- Database Server 환경 구축
- MySQL와 NPC 서버 연동
- Redis와 매치메이킹 서버 연동



NPC 서버

➤ NPC 서버?



NPC 서버에 문제가 생겨도, 전체 서버를 재시작할 필요가 없다!

NPC 서버

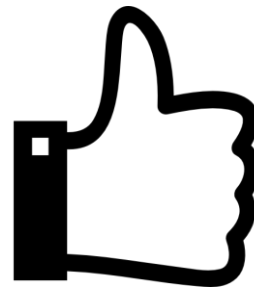
➤ WHY USE BOOST?



- IOCP 서버 구축을 위한 지식 부족
- 실시간 게임을 위한 비동기 기능이 필요



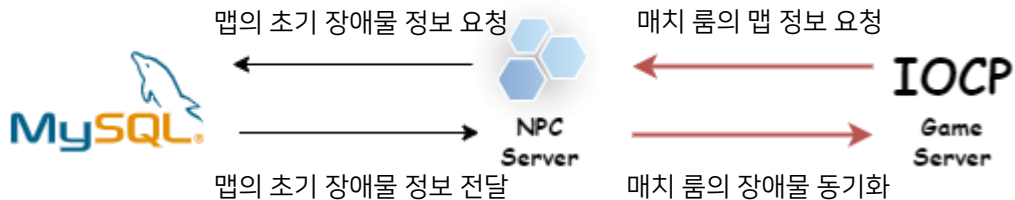
비동기 서버를 빠르고 쉽게 구축할 수 있는 **Boost Asio**를 선택



NPC 서버

➤ 흐름도

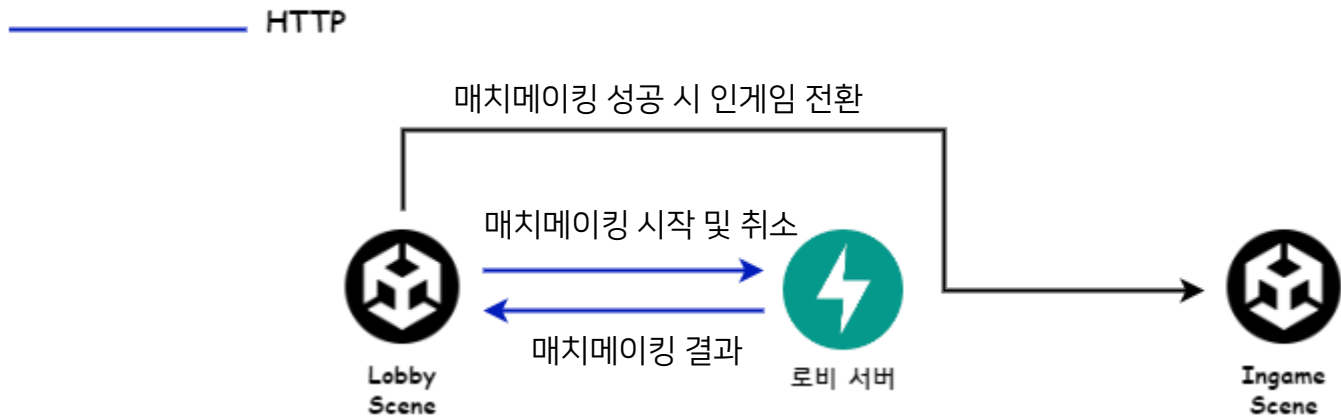
_____ Protobuf



매치 룸에서 진행되고 있는 맵의 장애물 위치를 동기화

로비 클라이언트

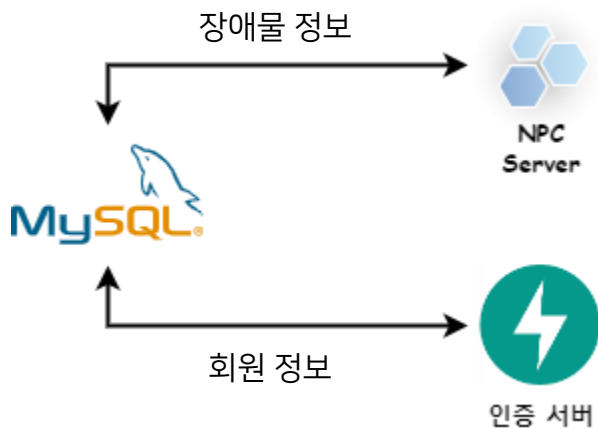
➤ 흐름도



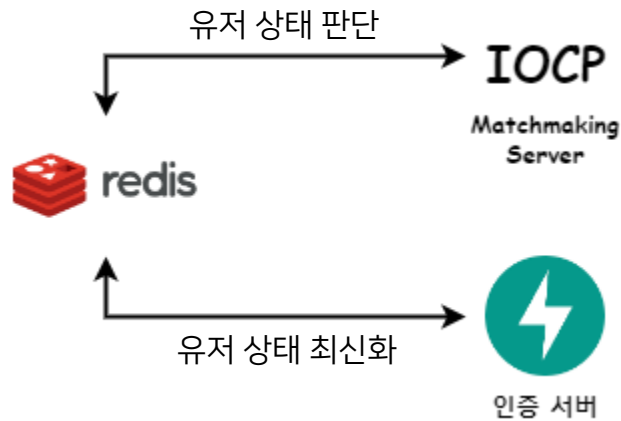
매치메이킹에 필요한 유저 정보를 로비 서버에 전달 + 인게임 전환

데이터베이스

➤ 흐름도



회원 + 맵의 장애물 정보 저장



매치메이킹을 시작한 유저의 무결성 검사

힘들었던 점

➤ **THREAD** 의 **CPU 점유율**

장애물을 개별적으로 동기화 → 장애물 하나 당 하나의 Thread 사용

+

모든 스레드가 쉬지 않고 장애물 위치를 계산



CPU 점유율 **100%** 를 달성하는 쾌거(..)를 이룸



힘들었던 점

➤ **THREAD** 의 **CPU** 점유율

장애물 위치를 실시간으로 계산 X → 최대 거리와 속도를 기반으로 **시간**을 계산

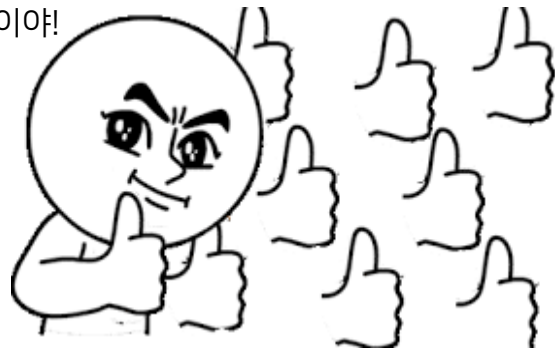


Boost::Thread 라이브러리의 Sleep 을 이용



CPU 점유율 문제 해결!

Boost 는 신이야!



아쉬웠던 점

➤ **REDIS** 활용 부족

유저가 게임에 정상적으로 접속했는지를 인증 서버에서만 최신화

게임 클라이언트 변조가 일어나도 매치메이킹 서버에서는 판단 불가

무엇보다 Redis 의 실시간성을 활용하지 못한 점이 아쉬움

➤ **NPC** 서버의 한계

단순한 스폰 위치 + 장애물 위치 동기화

NPC 서버에서 자주 사용되는 AI 를 제작하지 못함

NPC 서버의 특성을 살리지 못한 점이 아쉬움

느낀 점

C++ 로 서버를 밑단부터 제작하며, 실제 서비스에 사용되는 라이브러리를
폭넓게 사용해볼 수 있었음 (ex. Boost, STL, Protobuf)

초기 목표였던 프로그래밍 언어 활용 능력을 매우 높일 수 있었던 발판

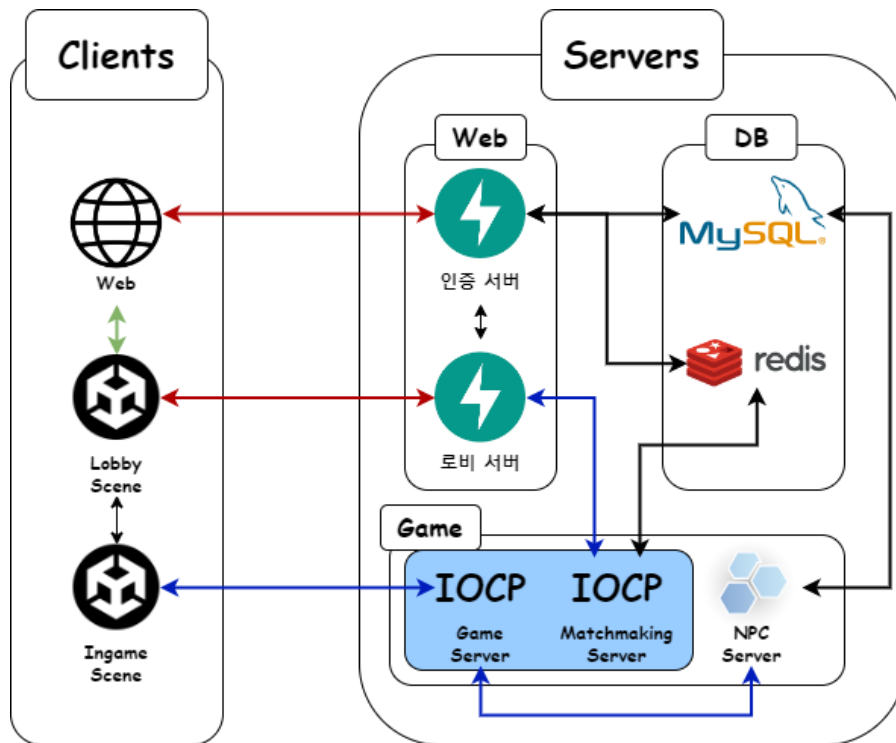
NPC 서버 이외에도 로비 클라이언트 제작 + 게임 클라이언트를 보조

**그럼에도 프론트 / 백엔드 양측 모두의
어떤 부분에서 개발이 즐거웠는지를 찾을 수 있었던 귀중한 경험**

풀스택 개발자의 길은 멀고도 (위)험하다



IOCP

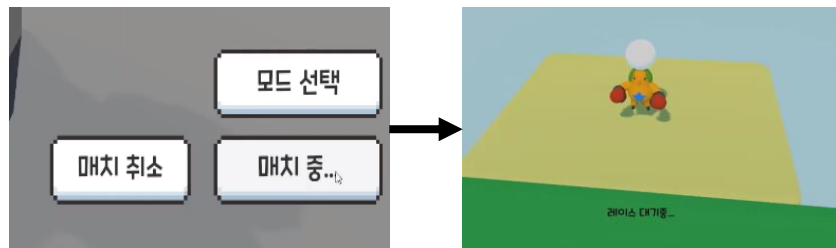
게임 서버
홍지현

IOCP

게임 서버
홍지현

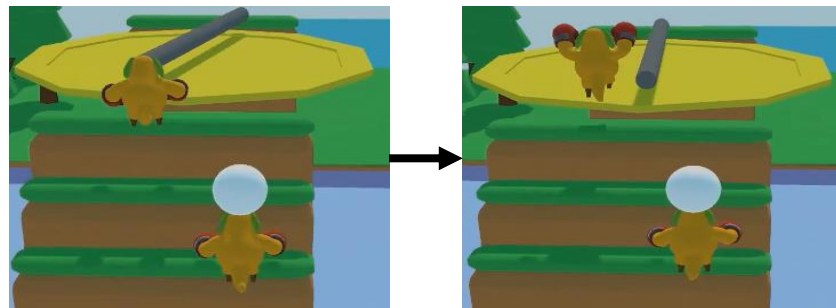
01 Matchmaking Server

- 게임 모드에 따른 매치메이킹 로직 구현
- (매치 성공, 실패, 취소 등등)
- Redis를 활용한 유저 ID 체크
- 로비 서버와 게임 서버와의 통신



02 Game Server

- 게임 클라이언트와 NPC 서버와 통신
- 게임 콘텐츠 로직 구현
- 게임 클라이언트 간의 동기화
- Protocol Buffer 버전 통일화 및 환경 구성



매치메이킹 서버 로직

```
enum
PlayerLevel
{
EMPTY
, SOLO
, DUO
, SOLO_DUO
, THREE
, SOLO_THREE
, DUO_THREE
, ALL
};
```



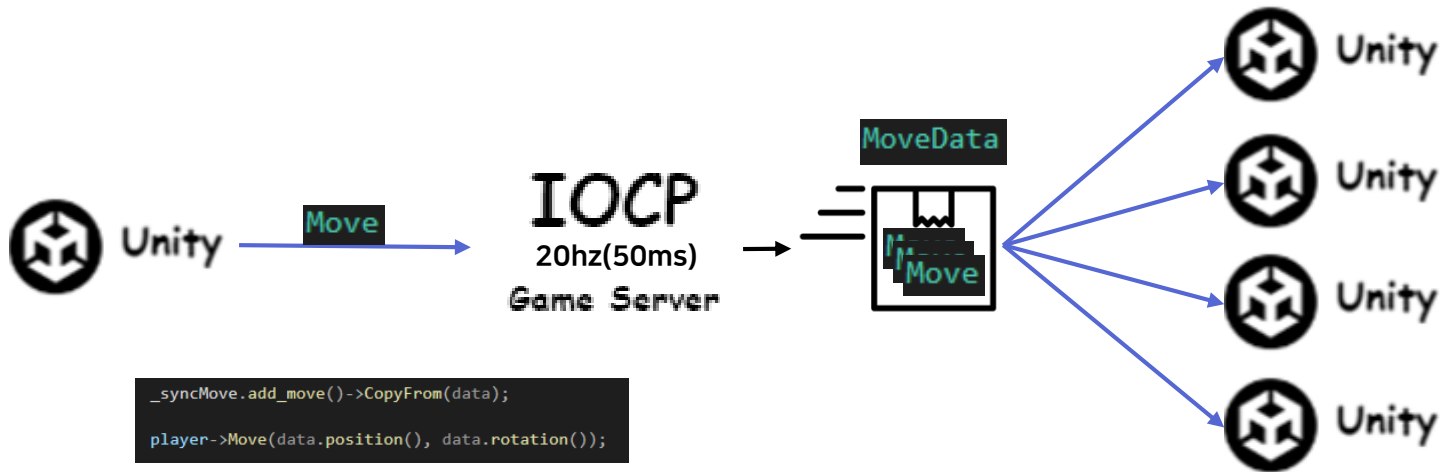
```
_playerWait[level].push_back(id);

if (level & SOLO)
    _playerSize[SOLO].fetch_add(1);
if (level & DUO)
    _playerSize[DUO].fetch_add(1);
if (level & THREE)
    _playerSize[THREE].fetch_add(1);
```

```
if (_playerSize[SOLO].load() >= SOLO_PLAYER_COUNT)
    DoAsync(&MatchManager::PlayerOutputMatch, { SOLO,
if (_playerSize[DUO].load() >= DUO_PLAYER_COUNT)
    DoAsync(&MatchManager::PlayerOutputMatch, { DUO, S
if (_playerSize[THREE].load() >= THREE_PLAYER_COUNT)
    DoAsync(&MatchManager::PlayerOutputMatch, { THREE,

DoTimer(50, &MatchManager::CheckingMatchPull);
```

게임 서버 로직



```
_syncMove.add_move()->CopyFrom(data);  
player->Move(data.position(), data.rotation());
```

```
void Room::GameSync()  
{  
    if (_start) {  
        //동기화 테스트  
        if (_syncMove.move_size() > 0) {  
            Broadcast(GameHandler::MakeSendBuffer(_syncMove, Protocol::PLAYER_SYNC));  
            _syncMove.Clear();  
        }  
        _syncMove.set_time(GetTickCount64());  
        DoTimer(50, &Room::GameSync);  
    }  
}
```

힘들었던 점



로컬에서는 잘 되는 테스트가



팀원들과 하는 테스트에서는 에러가 발생..
(특히 특정인원이 네트워크에서 튕김)



아쉬웠던 점

➤ 다양한 이동 동기화 방식을 시도 X

이동 동기화를 하려면 클라와 함께 시도를 해야 하지만 완성을 목표로 하다보니 계획이 밀리게 됨.

그래서 끝나기 3~4일 전에 동기화를 시도..

결국 많이 시도하지 못하고 미완성인 채로 프로젝트가 마무리..

➤ 매치 로직 구현하였으나 인게임에서 사용 X

솔로, 듀오, 트리오에 대한 로직을 구현하였으나 인게임 구현 X

그래서 매치 메이킹 서버의 제대로 된 테스트를 하지 못함.

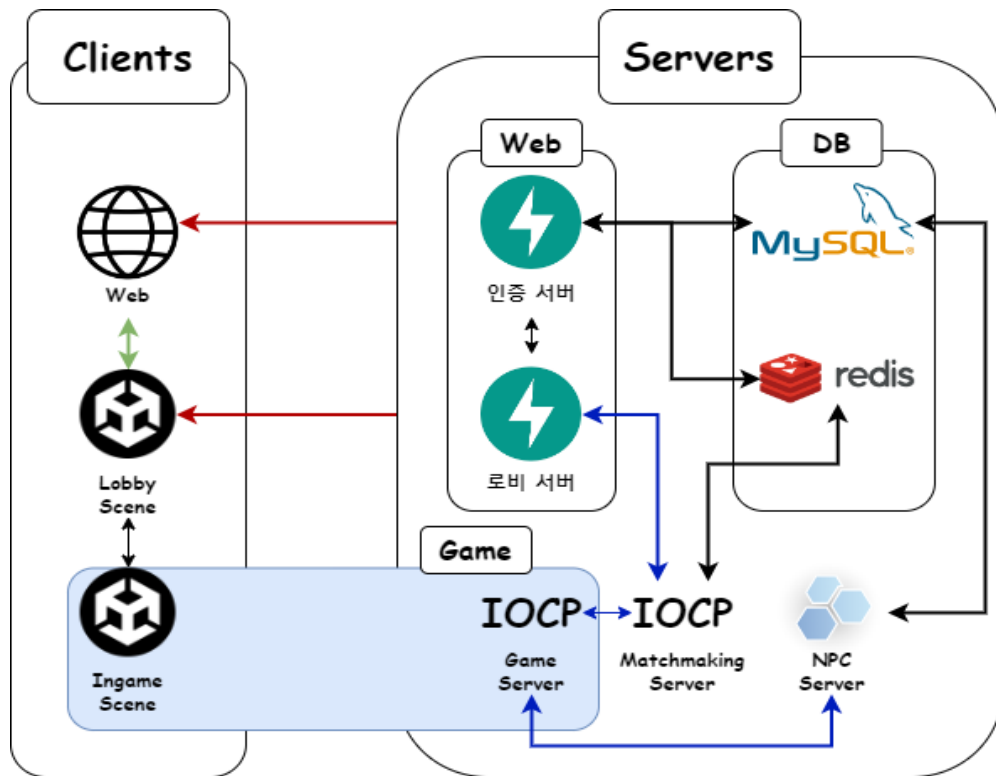
느낀 점

- ✓ 로컬에서는 발생하지 않았던 네트워크 오류들을 찾고 수정하면서, 협업 과정에서 게임 콘텐츠에만 경험해볼 수 있었다고 생각했던 자기 자신을 되돌아 볼 수 있었음
- ✓ 프로젝트를 진행하면서 처음 설계한 것과는 달리 매치 메이킹 로직, 패킷 형식 등을 많이 갈아 엮었고, 이를 통해 설계의 중요성을 알게 됨
- ✓ 코드 리뷰를 통해 지적해주신 부분들을 STL과 비트연산, 상속 등으로 코드를 리팩토링 하는 과정을 통해 개인 목표였던 C++ 초급자 탈출에 조금씩 가까워지고 있음



게임 클라이언트

김민관





게임 클라이언트 김민관

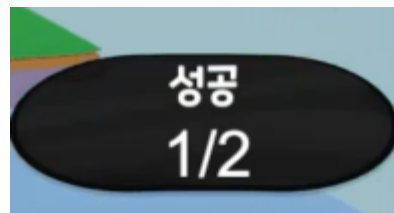
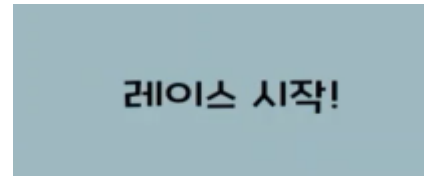
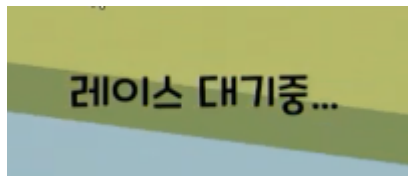
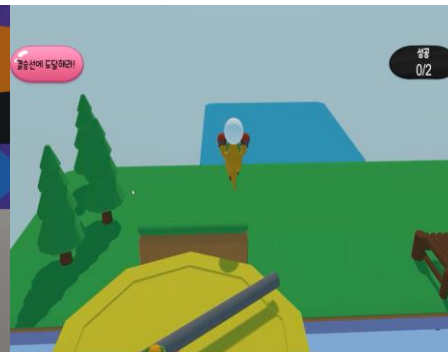


01 클라이언트 개발

- Player 행동 구현
- 관전 카메라
- InGame UI

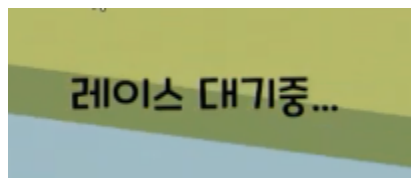
02 게임 서버 연동

- Protobuf를 이용한 데이터 통신
- 이동 동기화, 애니메이션 동기화



클라이언트 개발

➤ InGame UI

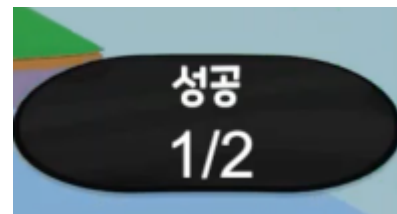


공정한 레이스



레이스 시작!

성공 인원 체크



Game Start 패킷이 오면 모든
참가자에게 조작 권한 부여

결승선에 Player가 충돌할
때마다 성공 인원 UI 수정

클라이언트 개발

➤ 관전카메라



MyPlayer

GOAL



생존 중인 Enemy로 카메라 전환



Enemy

GOAL



Destroy

게임 서버 연동

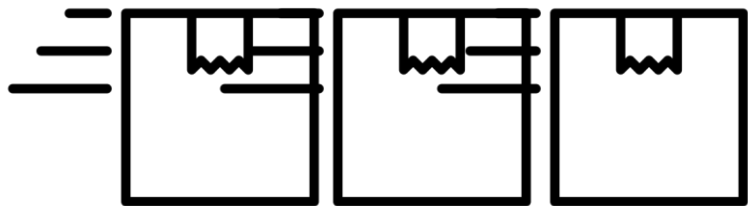
➤ 이동 동기화



송신하는 패킷의 양을 줄이기 위해, 이전의 위치와 수신 받은 위치가 다르면 자동으로 IDLE로 전환

힘들었던 점

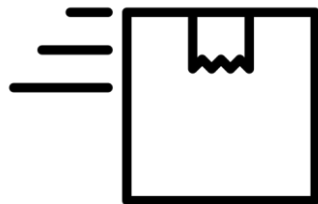
➤ 점프에서의 캐릭터 떨림 현상



연속된 위치 패킷 전달



점프 떨림



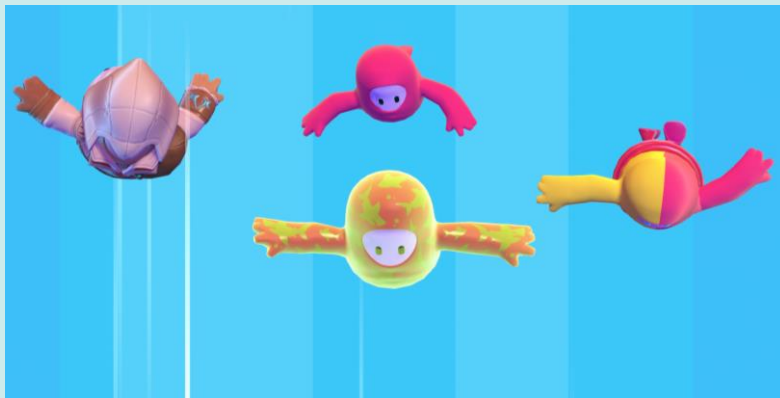
하나의 상태 패킷 전달



점프 성공

느낀 점

- 서버로부터 패킷을 수신 받는 것에 애를 먹어, 클라이언트에 힘을 주지 않아 아쉬웠다.
- 원활한 테스트를 위해서, 팀원들간의 공통된 개발환경 구축의 중요성을 알게 됐다.
- 객체지향 원칙을 고려하지 않고 개발하자, 하나의 기능을 수정하면 여러 스크립트를 수정해야 했다. 이를 통해, 객체지향 원칙의 중요성을 알게 됐다.
- 멀티플레이 게임을 만드는 것이 노력은 배로 들지만, 지식도 배로 쌓이는 것을 느꼈다.



TEAM BLUEBIRD

에필로그

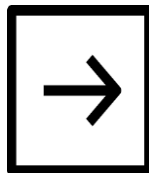
팀 목표



더불어 성장할 수 있는 기회로 삼기



즐거운 개발 문화 경험하기



프로젝트 경험을 RAM이 아닌 ROM에 저장시키기

팀원간 활발한 질문 + 의사소통!

구현에 대한 서로 다른 방향성
(각자의 가치관 + 경험의 차이)

개인적인 공부에 많은 시간 투자

속마음을 털어놓는 시간
(a.k.a 진실의 방)

개발에 너무 치중
로직 + 마일스톤의 잦은 변경 → 자료를 기록하는데
어려움을 겪음

Commit 메시지를 기반으로 주요
내용을 다시 기록해 볼 필요성

추가적으로 해볼 것

맵 추가

웹 플랫폼을 Java Spring 으로 리팩토링

NPC 서버에 AI 추가

채팅 서버

잡기, 벽 등반 등의 상호작용 추가

네트워크 로직 수정

이동 동기화

캐릭터 커스터마이징

성능 검증

프로젝트를 마치며..

2D 할 걸...



TEAM BLUEBIRD

Fin.