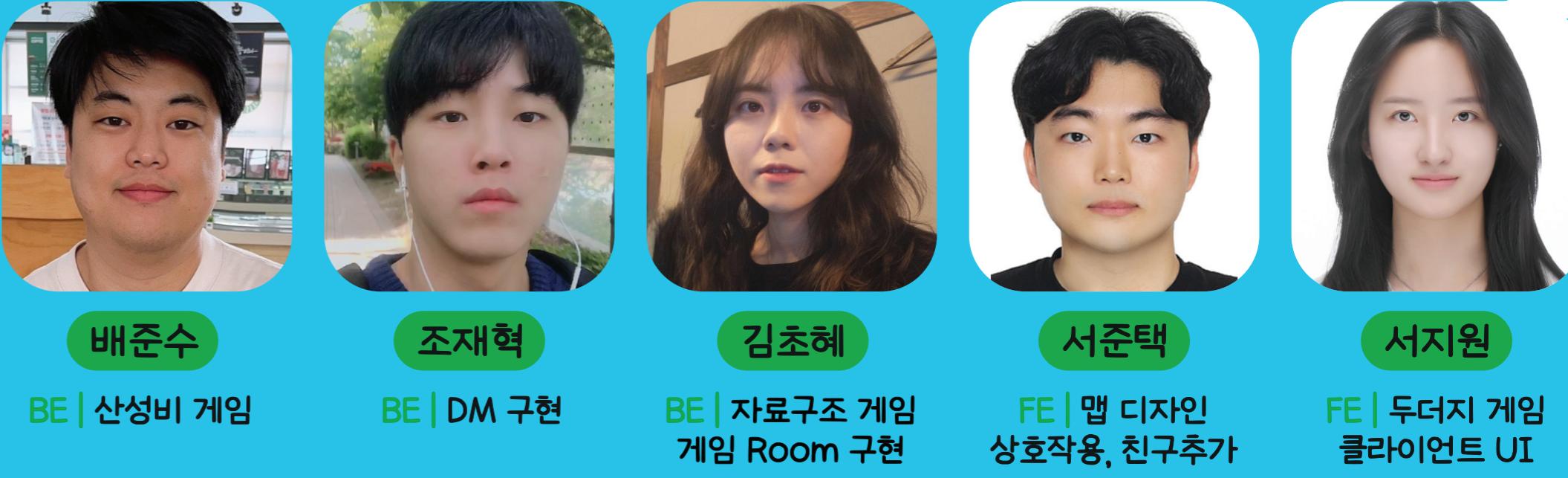


# Code Eat

키즈들의 코딩공간, 코드잇



## 서비스 소개

### 코딩 게임 for 키즈, 키즈들의 코딩 공간!

저희 코드잇은 키즈들이 코딩에 대해 좋은 첫인상을 가질 수 있도록 돋는 미니 게임 플랫폼입니다. 산성비 게임, 두더지 게임과 같이 익숙한 게임들에 코딩 관련 지식을 녹여냈습니다. 코드잇은 아이들이 게임하는 동안 자연스럽게 코딩을 배울 수 있는 환경을 제공합니다.

코딩 학습의 첫걸음을 코드잇에서 시작하세요.

## 기능 소개



### 친구, 채팅 & 유저 관리

#### 친구 추가

친구 관계와 친구 요청 Collection을 별도로 관리하였으며, 각각의 친구는 User Collection의 Object를 참조하도록 설계하여, 친구의 정보를 바로 쿼리할 수 있도록 설계하였습니다.

#### 채팅

소켓 이벤트를 기반으로 실시간 채팅을 구현하였습니다. 모든 유저 간에 1:1 채팅이 가능하도록 하였으며, 마지막 대화와 Room ID, 그리고 대화 내역을 기록하는 Collection을 별도로 구성하였습니다.

#### 유저 관리

Phaser, Colyseus, DB 간의 철저한 정보 관리를 통해 유저가 일관된 정보를 유지하고 관리할 수 있도록 구축하였습니다.

### 맵, 캐릭터 & 상호작용

Tiled를 이용해 Foreground, Object, Building, Ground 등의 이미지 계층을 형성하고, Phaser를 통해 사물과의 충돌 로직 및 캐릭터 이동 애니메이션을 추가하였습니다.



Phaser Scene 안에 Object를 형성하여 캐릭터가 다양한 아이템과 상호작용을 통해 게임을 시작할 수 있도록 구현하였습니다.



### 다양한 코딩게임 & 레벨



#### 코딩 산성비 게임

- 화면에서 떨어지는 단어를 정확하게 빠르게 입력하는 게임입니다.  
- 각 단어별 Property에 좌표, 속도, 아이템 등을 설정하고 짧은 주기로 단어의 좌표를 갱신하였습니다. 하한선을 감지하여 득점/실점을 판별하였으며, Debouncing으로 중복 오류를 방지하였습니다.



#### 두더지 게임

- 출제되는 문제의 정답을 들고있는 두더지를 잡는 게임입니다.  
- 두더지 번호를 무작위로 선별하였으며, 주어진 시간초 이내에만 두더지를 클릭할 수 있도록 구현하였습니다.



#### 자료구조 게임

- 주어진 문제에 맞게 동물의 조합을 완성하는 게임입니다.  
- 플레이어가 상황에 알맞는 자료구조를 선택하고, 그 자료구조에 해당하는 연산만을 사용해 정답을 맞춥니다.  
- 출제되는 문제의 ID를 기반으로 정답을 처리하고, 정해진 추가점수 테이블을 참조해 상황에 따라 다르게 배점되도록 구현하였습니다.

### 레벨, 경험치, 랭킹

- 게임의 승패에 따라 플레이어에게 경험치를 부여하며, 레벨업을 할 수 있습니다.  
- 레벨과 경험치를 기준으로 랭킹을 관리합니다.

## 문제 해결 과정

### 서비스 구분을 통한 통신 최적화

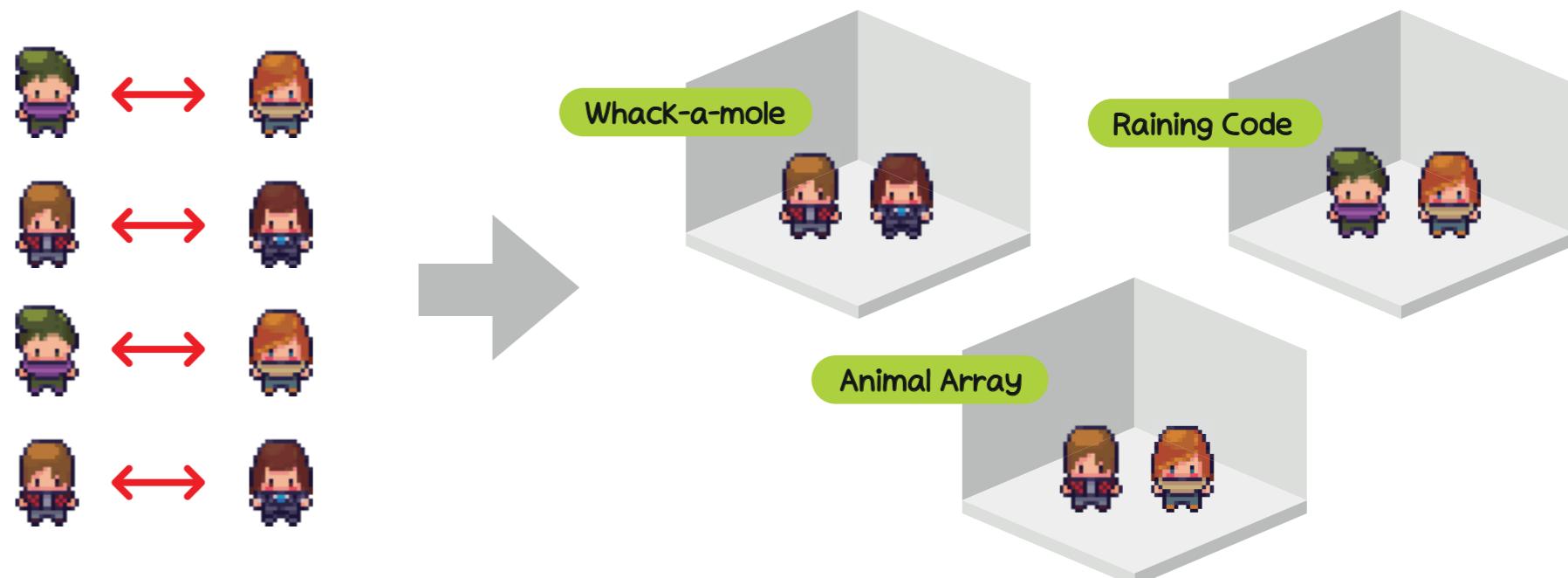
서비스에 접속하는 많은 유저들을 관리하고 캐릭터의 움직임을 실시간으로 동기화하는 메인 맵, 실시간으로 이루어지는 멀티 플레이 게임, 그리고 실시간 채팅 서비스까지 모두 한 포인트에서 관리된다면 지연 및 간섭이 발생할 수 있으며, 메인 서비스와 게임 모두에 성능 저하를 일으킬 수 있습니다.

#### 문제점

1. Phaser를 기반으로 캐릭터의 움직임을 렌더링하고, 다수의 인원이 들어가고 나가는 메인 맵과 소수의 유저에게 지연 없이 동일한 정보를 제공해야하는 게임룸은 성질이 다르다
2. 많은 양의 데이터를 실시간으로 처리해주어야 하는 두 가지 서비스를 한 포인트에서 관리한다면 통신 지연이나 간섭의 발생 가능성이 커진다
3. 유저의 고유 ID를 이용해 P2P 통신을 맺는 기존 방식은 게임 로직을 관리하기에 부적합

#### 해결방안 및 결과

- 메인 맵, 게임, 채팅 각각의 기능에 맞게 네트워크 인스턴스를 분리하고 별도의 정의를 내려 관리하였습니다.
- Colyseus 프레임워크를 기반으로 메인 맵과 각각의 게임의 로직에 맞게 Room Interface를 정의하여, 해당하는 품(메인 또는 게임)에 참여한 클라이언트는 해당하는 로직에만 접근할 수 있도록 분리하였으며, 이를 통해 메인룸과 게임룸 간의 간섭을 완전히 없앨 수 있었습니다.
- P2P가 아닌 WebSocket 기반의 실시간 통신으로 게임 서버에서 관리하는 내용을 클라이언트에서 빠르게 업데이트 받음으로서 원활한 멀티 플레이를 구현할 수 있었습니다.



### 실시간 멀티플레이 게임 동기화

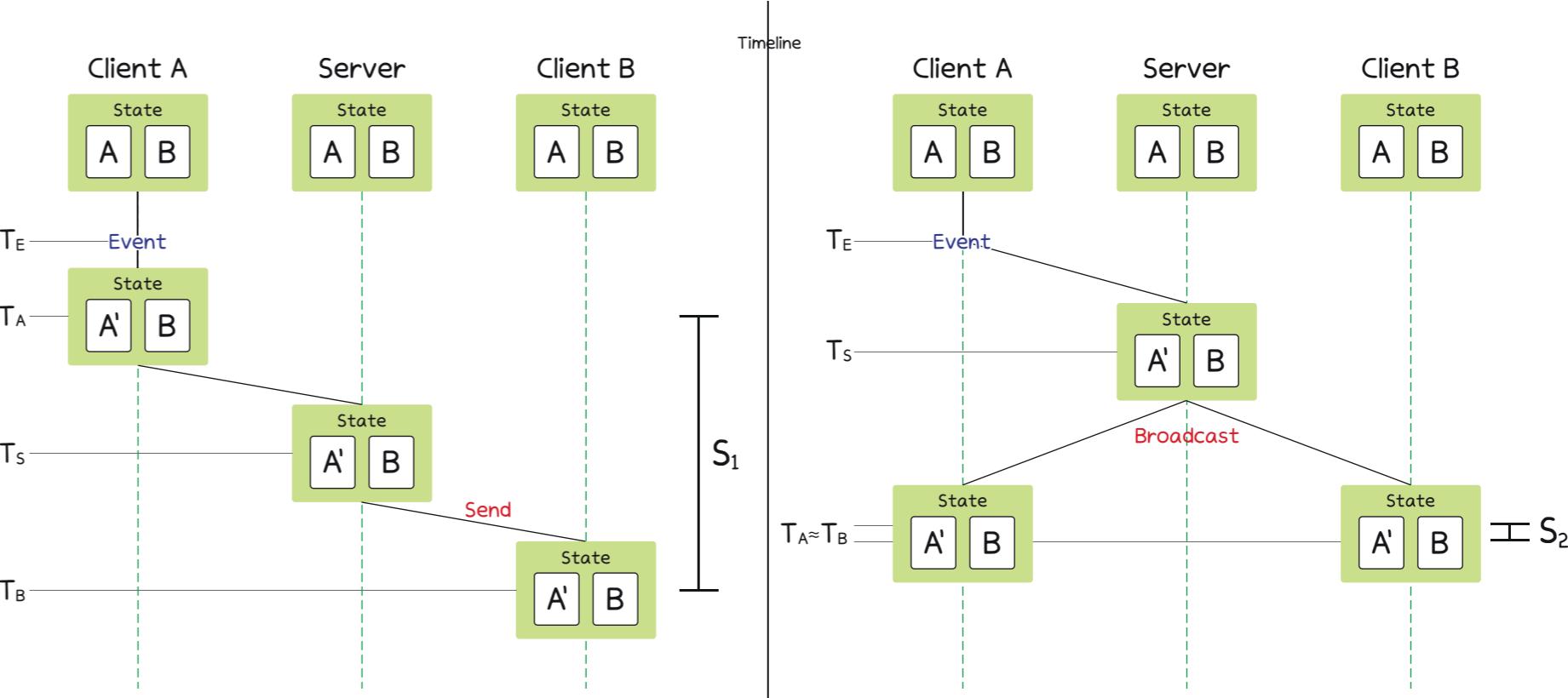
다양한 실시간 멀티플레이 게임을 제공하는 서비스 특성상, Client-Client & Client-Server 상태 동기화를 바탕으로 원활한 플레이 환경을 제공하는 것이 중요했습니다.

#### 문제점

1. 한 유저의 화면에서 이루어지는 액션이 다른 플레이어의 화면에 동기화되기까지 걸리는 시간 차이
2. 동기화가 이루어지는 동안 새로운 이벤트 발생으로 인한 간섭 발생

#### 해결방안 및 결과

- 클라이언트에서 유저 이벤트가 발생하였을 때 클라이언트 화면에 바로 반영하는 것이 아닌, 서버에서 중앙집중적으로 처리하고 연결된 모든 클라이언트에게 동시에 broadcast하도록 구현하였습니다.
- 이를 통해 두 플레이어 간의 상태차이를 최소화할 수 있었으며, 새로운 이벤트에 대해서도 서버의 통제 하에 간섭 및 오류를 없앨 수 있었습니다.
- 또한, 서버에서 로직을 처리함으로서 클라이언트에서의 부하를 현저히 줄일 수 있었습니다.



## 아키텍처

