



Team :  
TGI

산학연계  
프로젝트

# 메타버스 기반 웹채팅 플랫폼



최종 발표

2023.08.24

권좌  
영  
이승  
훈  
이태  
희  
이용  
호





# 목 차



- 01. 과제 개요
- 02. 과제 해결방안
- 03. 주요 개발내용
- 04. 개발성과 분석
- 05. 과제를 통해 얻게 된 교훈





01.



## 과제 소개

주제 / 추진 배경 / 기업 요구사항 / 과제 목표 / 팀 소개



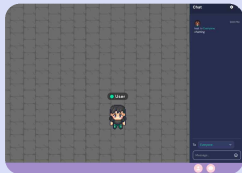


# 프로젝트 주제

주제 : “비대면 시대의 새로운 메타버스 플랫폼”

-> 메타버스 기술을 기반으로 자신의 채팅 룸 공간에서 많은 사람들과  
함께

대화를 나눌 수 있는 플랫폼 개발



출처 : 기업측 제공 받은 화면





# 추진 배경



코로나 19로 인해 비대면 시대가 도래

-> 온라인 환경에서의 소통과 협업이 중요해짐.

-> 다양한 분야에서 메타버스를 통한 새로운 서비스가 생겨나고 있음.

따라서 이러한 변화에 발맞춰 기존의 **채팅 플랫폼**을 메타버스 상에서 구축하는  
플랫폼을 개발하고자 함.





# 기업 요구사항



필요인원  
: 3~5명



프로젝트 배경기술



프로젝트 진행을  
위해 요구되는 기술

요구사항

- 필요 인원 : 3~5명
- 프로젝트 배경기술 / 프로젝트 진행을 위해 요구되는 기술
- 본 시스템을 개발할 수 있는 SW 활용

출처 : 기업 측 제공





# 과제 목표



메타버스 내 캐릭터 생성

아이디 색깔을 통해  
상대방과 구분 가능



구성원 전체와 대화할 수  
있는  
전체 채팅방 개설



웹 소켓 기반의  
채팅 시스템 구축



캐릭터 이동 가능  
특정 사물과  
상호작용





## 팀 소개



권좌영  
(팀장)



이승훈  
(팀원)



이태희  
(팀원)



이용호  
(팀원)



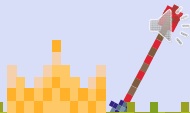




02.

## 과제 해결 방안

접근 방법 / 시스템 구조 / 개발 환경





# 접근 방법

- 처음엔 팀원들 모두 메타버스에 대해 생소
- 여러 가지 기존 메타버스 찾아봄  
ex) 게더타운 등
- 1차 피드백 후 제한된 시간 안에 기존에 구현되어 있는 메타버스 플랫폼 정도의 서비스 구현은 **현실적으로 불가능**하다고 판단
  - > **과제 목표** 소폭 수정
  - > 멘토님과 상의하며 여러가지 오픈소스 탐색
- 여러가지 검색해 본 결과 **Phaser3 + cloyseus**를 알게됨.





# 시스템 구조

- 이외에도 여러가지 개발 프레임워크가 있었지만  
개발의 편의성 때문에 그 중 phaser3 + colyseus 채택



출처 : COLYSEUS





# 시스템 구조 - Client

## [ PHASER ]

- JavaScript로 2D게임을 만들 수 있는 프레임워크, 즉 게임 엔진
- 예제 코드나 API문서들이 잘 정리되어 있어서 금방 따라할 수 있음

### 단점:

- 2D게임만 만들 수 있다.
- 다른 엔진들에 비해 많이 무거운 편



출처 : PHASER





# 시스템 구조 - Server

## [ COLYSEUS ]

- 멀티플레이어 게임 개발에 특화된 프레임워크
- 게임 서버와 게임 클라이언트 간의 통신을 **WebSocket**을 기반으로 하는 네이티브 프로토콜을 사용하고, 게임 로직을 서버에서 처리
- 게임 클라이언트는 게임 로직을 처리하는 추가적인 부담을 갖지 않아도 되고, 게임 서버에서 게임 상태를 효율적으로 처리하므로, 게임의 성능과 **안정성**이 향상

### 단점:

- **TypeScript**에 익숙하지 않은 개발자에게는 적합하지 x
- JavaScript와는 약간 다른 문법





# 개발 환경

OS - windows, Mac OS

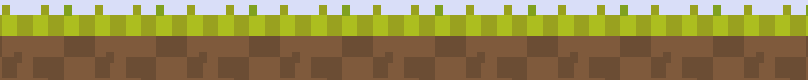


개발 언어 - TypeScript



프레임 워크

- 게임 : Phaser3
- 서버 : Colyseus





03.

## 주요 개발 내용

구성요소별 개발사항





# 구성요소별 개발사항

## 클라이언트와 서버연결

-> phaser3 + colyseus 를 이용해 구현

```
// DEPENDENCIES
import Phaser from "phaser";
// RUN MAIN
import { GameScene } from "../src/GameScene";
// RUN START
import { StartScene } from "../src/StartScene";
```



```
import { listen } from "@colyseus/tools";

// Import Colyseus config
import app from "../app.config";

// Create and listen on 2567 (or PORT environment variable.)
listen(app);
```







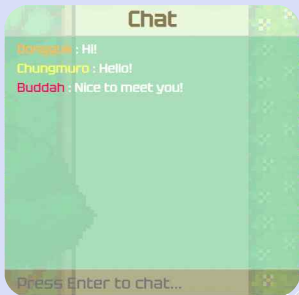
# 구성요소별 개발사항

## 2. 웹소켓 기반의 채팅 기능 구현

-> colyseus 에 포함된 기능

```
<body>
<!-- ..... -->
<!-- Include entrypoint TypeScript file as a module -->
<!-- ..... -->
<div id="game-container" style="position: absolute; z-index: 1"></div>
<!-- 채팅 영역 -->
<div id="chat-container">
  <div id="chat-head">Chat</div>
  <div id="chat-messages"></div>
  <input id="chat-input" type="text" placeholder="Press Enter to chat..." />
</div>

<script src="index.ts" type="module"></script>
</body>
```





# 구성요소별 개발사항

## 3. 플레이어의 애니메이션 상호작용 움직임 동기화



-> 캐릭터 PNG 이미지와 JSON 애니메이션 파일을 불러온다.

이때 각각의 캐릭터마다 고유한 스프라이트 시트와 애니메이션 데이터가 필요

```
export const createAnimations = (anims: Phaser.Animations.AnimationManager) => {
  const frameRate = 10;

  // 아래로 걷는 모션
  anims.create({
    key: 'idle_down',
    frames: anims.generateFrameNames('character', {
      prefix: 'Adam_idle_anim_',
      start: 19,
      end: 24,
      zeroPad: 0,
      suffix: '.png',
    }),
    frameRate,
    repeat: -1,
  });
};
```

```
// 왼쪽으로 걷는 모션
anims.create({
  key: 'idle_left',
  frames: anims.generateFrameNames('character', {
    prefix: 'Adam_idle_anim_',
    start: 19,
    end: 18,
    zeroPad: 0,
    suffix: '.png',
  }),
  frameRate,
  repeat: -1,
});
```

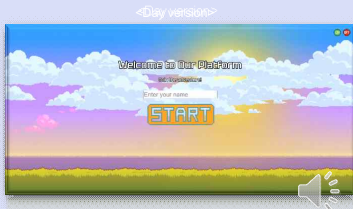
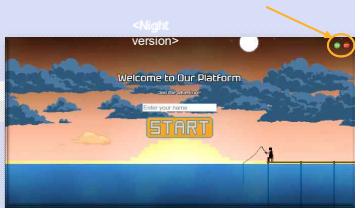




# 구성요소별 개발사항

## 4. 시작화면 추가

-> 바로 게임화면으로 시작하는 건 어색하다고 판단하여 **시작화면** 추가  
두가지 버전으로 제작  
토글 버튼(on/off) 추가하여 반전될 수 있게 함.





04.

## 개발성과 분석

목표 대비 개발성과 / 기대 효과





# 목표대비 개발성과

현재 수정된 목표 대비 개발을 모두 완료한 상태

예상보다 개발이 빠르게 진행이 되어서 추가한 부분 :

1) 랜덤으로 생성되는 session ID대신

시작화면에서 직접 입력한 내 아이디를 사용하게 함

2) 시작화면 Toggle 기능 추가

3) 채팅창에서 아이디별로 색깔을 다르게 함으로써

아이디 구분 용이하게 함






# 기대 효과

 사용자들은 가상 공간에서 캐릭터를 통해 다른 사용자들을 만나 새로운 관계 형성 가능

 사용자들은 다른 사람들과 채팅을 함으로써 소통과 협업 가능

 사물과의 상호작용으로 현실성 부여

 직관적이고 손쉬운 조작법으로 적응 쉬움





05.



## 과제를 통한 교훈

배운 내용 / 주요 문제점 & 극복사항 / 향후 계획





# 배운 내용

1) 처음 써보는 언어인 **Typescript**를 써 봄으로써 Typescript의 장단점 알게 됨.

장점 :

- Handbook 문서로 튜토리얼이 정리되어 있음
- Typescript는 Javascript의 상위 집합이므로 Javascript를 사용할 줄 안다면 Typescript도 금방 익힐 수 있음

단점 :

- 초반 세팅이 불편함
- **TypeError**로 인한 빨간 줄을 많이 보기도 함
- Javascript와 비교해 코드의 길이가 불가피하게 길어지게 됨  
-> 가독성 떨어짐







# 배운 내용

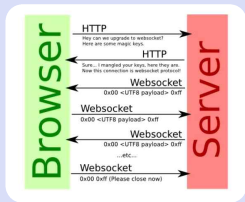
## 2) 웹소켓 기반 서버와 클라이언트의 통신 장단점 알게 됨

### 장점 :

- 양방향 통신
- 실시간 통신

### 단점 :

- Cross Browser 문제
- HTTP와 다르게 상태를 유지하기 때문에 서버와 클라이언트는 연결을 항상 유지해야함 -> 부하가 발생할 수 있다는 단점
- 프로그램 구현에 보다 많은 복잡성을 초래할 가능성이 있음.





# 주요 문제점 & 극복 사항

1)

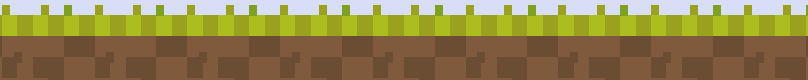
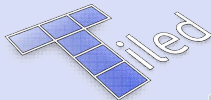
문제점 : 맵을 구성하는 것에 고민을 많이 하고 어려움을 겪음

극복방법 : 도트 찍는 걸 외주를 맡겨야 하나 고민 중에

<https://limezu.itch.io/modernexteriors> 에서

저렴한 가격으로 수정 가능한 타일들 제공하는 것 발견

테마에 맞는 적절한 타일 구매 후 "Tiled" 프로그램 이용하여 맵 제작





# 주요 문제점 & 극복 사항

2)

문제점 : 시작화면을 따로 구성하는데 애를 먹음

극복방법 : 기존 랜딩 페이지처럼 만들려고 시도 -> 실패

Phaser는 게임기반 프레임워크라서 **씬 전환**으로

비슷하게 구현 가능한걸 알게됨,

게임 씬, 시작화면 씬 분리하여 작성하였고

코드 안 " **scene : []** " 안에 같이 넣어줌.

```
// game config
const config: Phaser.Types.Core.GameConfig = {
  type: Phaser.AUTO,

  scale: {
    mode: Phaser.Scale.ScaleModes.RESIZE,
    width: window.innerWidth,
    height: window.innerHeight,
  },
  backgroundColor: "#343a40",
  parent: "phaser-example",
  physics: { default: "arcade" },
  pixelArt: true,
  dom: {
    createContainer: true
  },
  scene: [StartScene, GameScene],
};
```





# 주요 문제점 & 극복 사항

3)

문제점 : 플레이어 움직임 설정 문제

극복방법 : 초반엔 플레이어의 움직임을 좌표기반으로 구현하였는

그렇게 하면 phaser3에서 제공해주는 충돌기능을 사용

-> velocity 기반으로 수정



```
if (this.currentPlayer) {
    //내플레이어 조종 구현
    this.cameraScene.cameras.main.startFollow(this.currentPlayer, true);
    if (this.cursorKeys.left.isDown) {
        this.currentPlayer.setVelocityX(-200);
        this.containerBody.setVelocityX(-200);
    } else if (this.cursorKeys.right.isDown) {
        this.currentPlayer.setVelocityX(200);
        this.containerBody.setVelocityX(200);
    } else {
        this.currentPlayer.setVelocityX(0);
        this.containerBody.setVelocityX(0);
    }

    if (this.cursorKeys.up.isDown) {
        this.currentPlayer.setVelocityY(-200);
        this.containerBody.setVelocityY(-200);
    } else if (this.cursorKeys.down.isDown) {
        this.currentPlayer.setVelocityY(200);
        this.containerBody.setVelocityY(200);
    } else {
        this.currentPlayer.setVelocityY(0);
        this.containerBody.setVelocityY(0);
    }
}

// containerBody의 위치를 현재 플레이어의 위치로 업데이트함
this.containerBody.x = this.currentPlayer.x;
this.containerBody.y = this.currentPlayer.y - 60;
```



4)

극복방법 : **setTimeout** 함수 혹은 **phaser** 에서 제공하는  
지연함수 등을 이용하여 **지연속도**를 조절하여 해결하





# 향후 계획

사용자들의 수요를 높이려면 기능을 더 추가 해야 할 것이라고 생각

- 화상채팅 기능 추가할 계획



- 채팅방 링크 공유 기능 추가할 계획



- 캐릭터 추가로 캐릭터 선택 기능 추가할 계획



- 의자 앉기 외에 사물과 상호작용 추가할 계획





THANKS!



Do you have any questions?

[https://github.com/T-G-I-Web/Webtaverse\\_Server](https://github.com/T-G-I-Web/Webtaverse_Server)

[https://github.com/T-G-I-Web/Webtaverse\\_client](https://github.com/T-G-I-Web/Webtaverse_client)

