Speed Run

Unity 3D 개인 프로젝트

메타플밍 6기 안종화

개발기간 2023.06.17 ~ 2023.07.10

GitHub : https://github.com/jong212/Project_RunRunRun.git

시연 영상 : https://youtu.be/Pb0yJ-BoPto

목차

시연영상

시연연상 : https://youtu.be/PbOyJ-BoPto

쾌속 질주 !!

폴 가이즈

장애물 ★★★★★

속도감 ★☆☆☆☆

테일즈런너

장애물 ★★★☆☆

속도감 ★★★★★

Speed Run

장애물 ★★★★★

속도감 ★★★★★

나의 컨트롤을 보여 주지 !

5개 이상의 캐주얼
캐릭터 제공 !

캐주얼

인게임

Speed Run

아웃게임

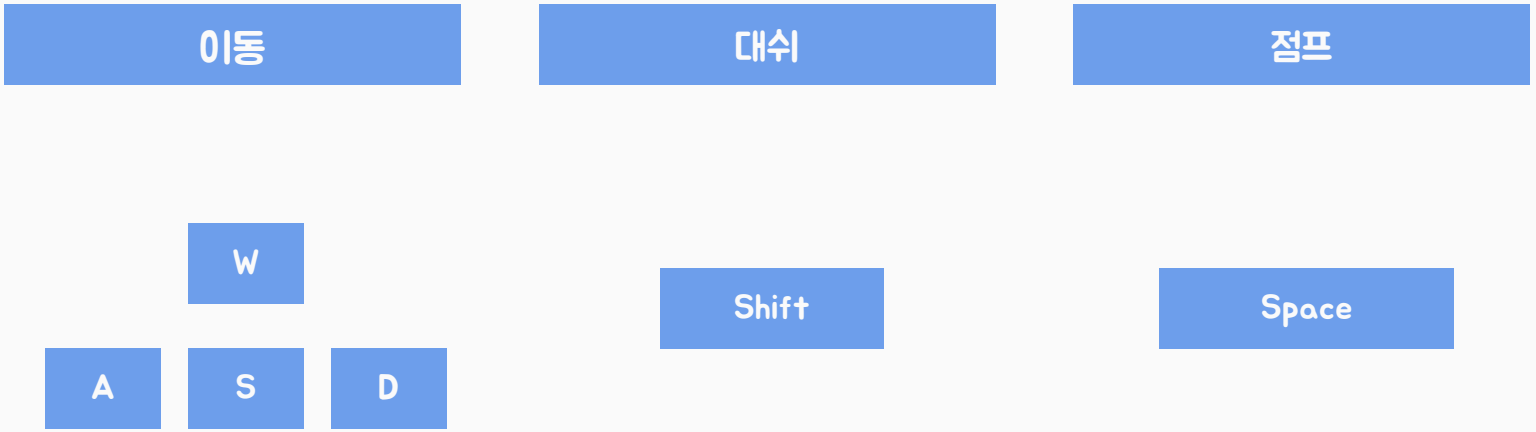
상점, 개인 로비, 캐릭터 변경 가능 !

친구들과 함께 하자 !

멀티플레이

하이라이트

최고의
플레이어는 누구?


이동
W
A
S
D
대쉬
Shift
점프
Space

로그인화면
- 회원가입

로비화면
- 방목록 / 상점 캐릭터 변경

게임방입장
- 채팅 / 준비 카운트다운

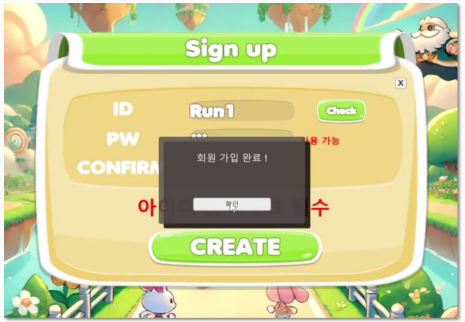
게임시작
- 실시간 순위 / 장애물 / 하이라이트 영상

로그인화면

로그인 UI

로그인 성공 시 로비로 이동합니다

회원가입



회원가입 기능을 제공합니다.

회원정보 체크

DB에 저장된 정보와 일치하는지 체크하는 로직이 있습니다.

로그인화면
└ 회원가입

로비화면
└ 방목록 / 상점 캐릭터 변경

게임방입장
└ 채팅 / 준비 카운트다운

게임시작
└ 실시간 순위 / 장애물 / 하이라이트 영상

로비 UI

골드

게임 시작 전 연습맵 제공

로그인 화면으로 이동

방 목록, 상점, 캐릭터 변경 버튼

방목록

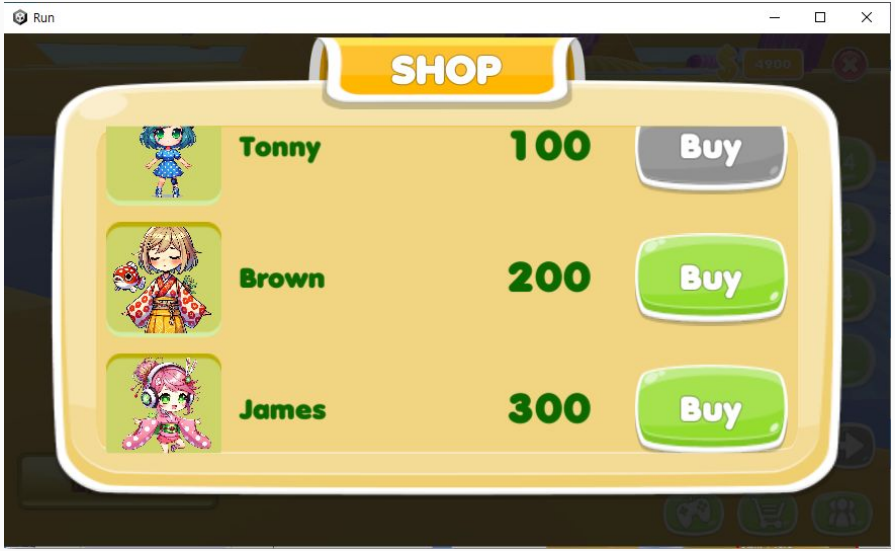
클라이언트 닉네임

동시 접속자 수

방 리스트 및 참여 인원

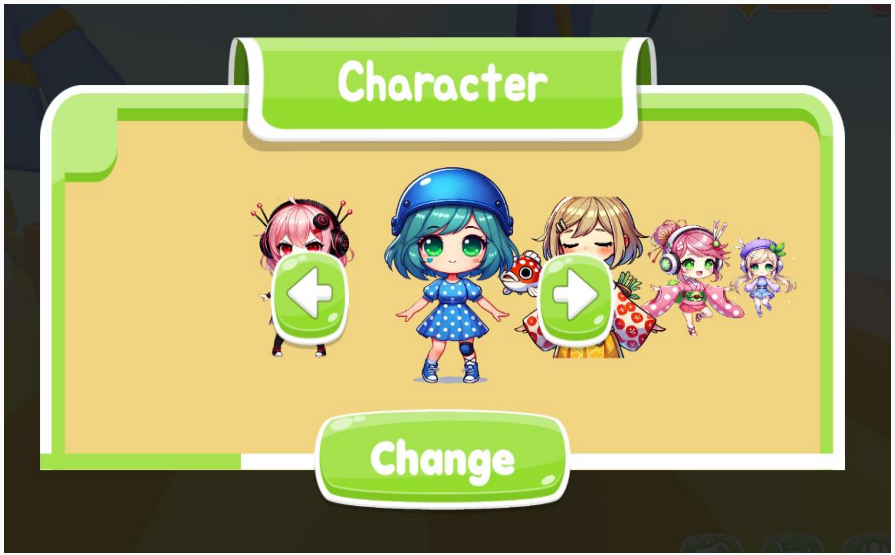
방 만들기

상점



- 보유한 골드로 캐릭터를 구매 할 수 있습니다. (DB 처리)
- 보유한 캐릭터의 구매 버튼은 비활성화 됩니다. (중복구매방지)
- 위 아래로 스크롤 할 수 있습니다.

캐릭터 변경



- 좌, 우 버튼을 통한 슬라이드 기능이 제공됩니다.
- Change 버튼을 누르면 가운데 위치한 캐릭터로 변경 됩니다.
- 재접속 시 마지막에 착용한 캐릭터로 로드 됩니다.

Room (대기방)

로비로 이동

게임 준비 버튼

채팅시스템

카운트다운 5..4..3..2..1.. Start

모든 유저의 준비 상태가 확인 되면

5초 카운트다운 후 게임이 시작 됩니다.

로그인화면
- 회원가입

로비화면
- 방목록 / 상점 캐릭터 변경

게임방입장
- 채팅 / 준비 카운트다운

게임시작
- **실시간 순위** / 장애물 / 하이라이트 영상

꼴찌는 나의 것 !!

- 실시간 순위
- 로컬 플레이어의 패널 배경 색상은 초록색 으로 표시

로그인화면
- 회원가입

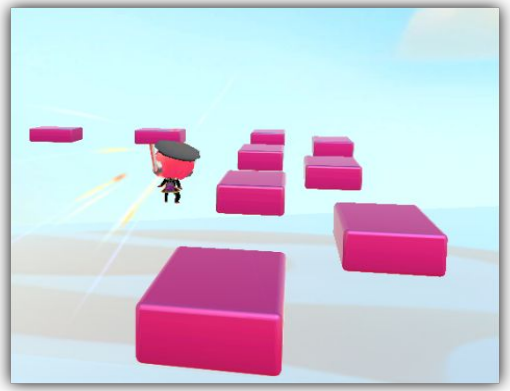
로비화면
- 방목록 / 상점 캐릭터 변경

게임방입장
- 채팅 / 준비 카운트다운

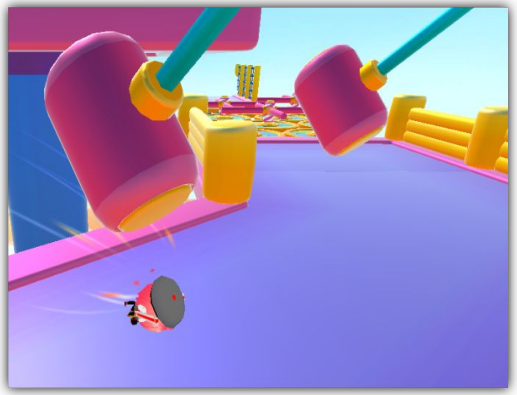
게임시작
- 실시간 순위 / **장애물** / 하이라이트 영상

다양한 장애물

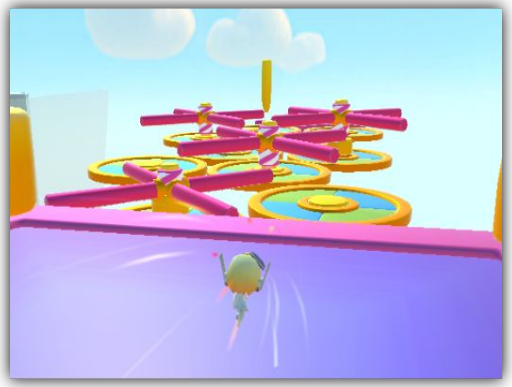
움직이는 장애물 아슬한 플레이!



해머에 닿으면 3초간 스턴



세밀한 컨트롤 유도



로그인화면
└ 회원가입

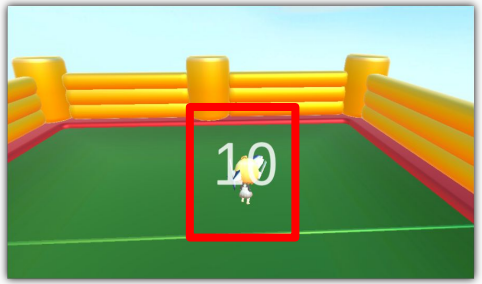
로비화면
└ 방목록 / 상점 캐릭터 변경

게임방입장
└ 채팅 / 준비 카운트다운

게임시작
└ 실시간 순위 / 장애물 / 하이라이트 영상

하이라이트 (게임 종료)

1 최초 도착
카운트 다운 10초



2 결산
1등이 활약한 하이라이트 영상을
모든 클라이언트에서 실행

3 로비로 이동

런타임 초기화
└ 데이터 드리븐(Xml)

DataBase
└ 로그인 프로세스
└ 상점 프로세스

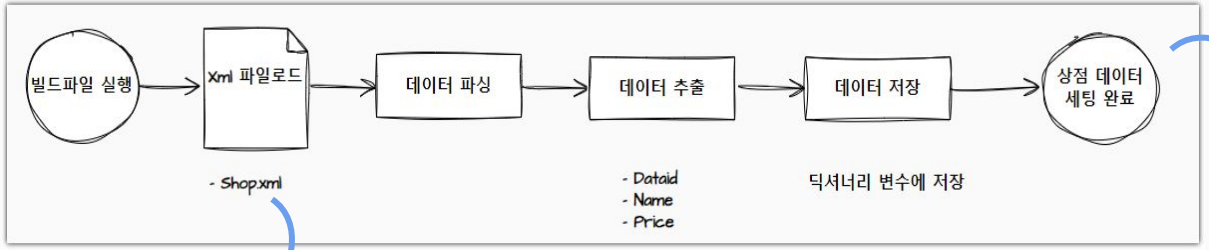
UI
└ UIManager

포톤 Network (동기화)
└ 준비 / 카운트 다운 / 하이라이트 / Room / 장애물 회전

데이터 드리븐

- 상점 데이터를 Xml파일에 별도 관리하기 위해 데이터 드리븐 방식을 사용했습니다.

데이터 드리븐 과정




```
▼<Shop>
  ▼<dataCategory>
     <data DataId="1" Name="Player" Description="기본 캐릭터" Price="0"/>
     <data DataId="2" Name="Tonny" Description="느린 러너" Price="100"/>
     <data DataId="3" Name="Brown" Description="보통 러너" Price="200"/>
     <data DataId="4" Name="James" Description="빠른 러너" Price="300"/>
     <data DataId="5" Name="David" Description="매우 빠른 러너" Price="400"/>
   </dataCategory>
 </Shop>
```

세팅 완료

런타임 초기화
└ 데이터 드리븐(Xml)

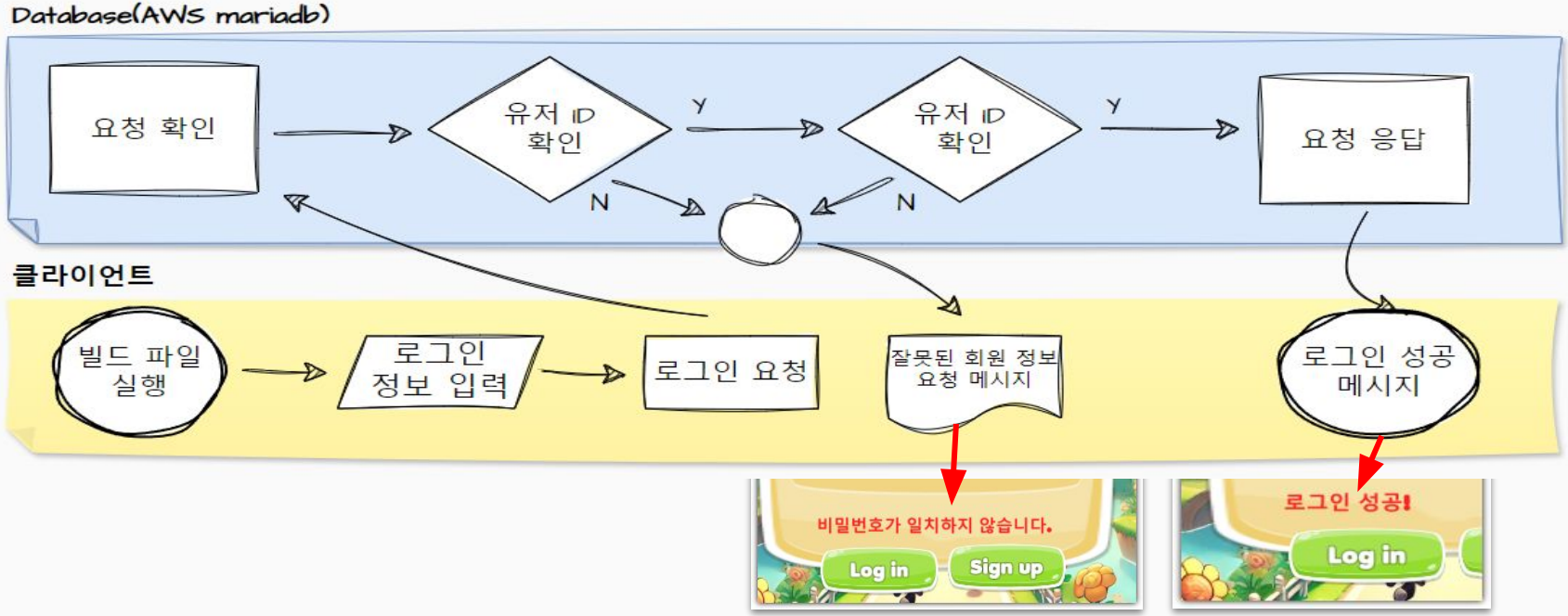
DataBase
└ 로그인 프로세스
└ 상점 프로세스

UI
└ UIManager

포톤 Network (동기화)
└ 준비 / 카운트 다운 / 하이라이트 / Room / 장애물 회전

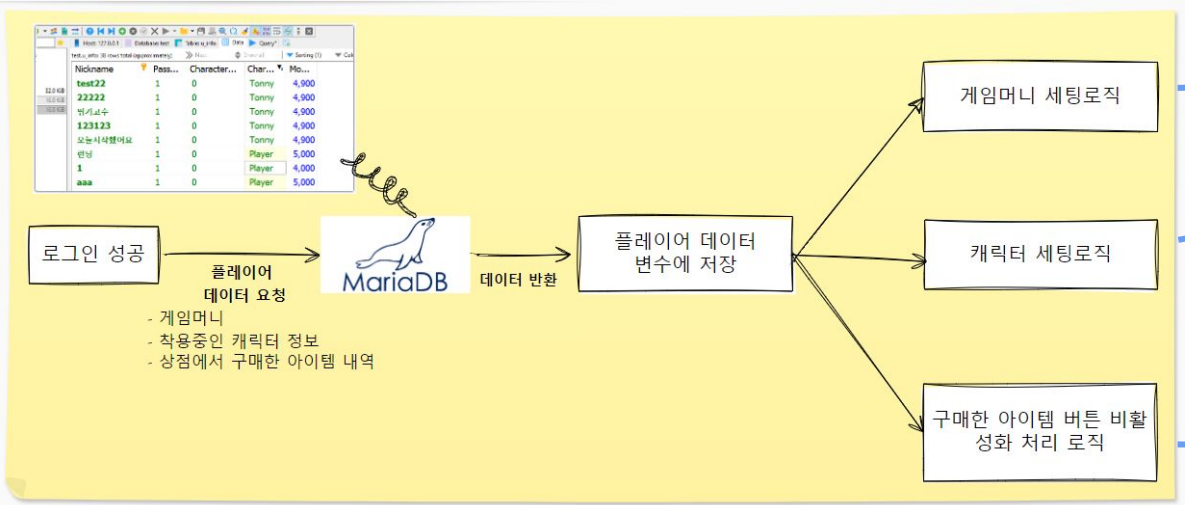
로그인 - Login Flow (1/2)

● 로그인 계정정보는 AWS 서버의 Maria DB에 저장되도록 설계하였습니다.



로그인 - Setting Flow (2/2)

- 로그인 성공 시 해당 유저의 게임머니, 착용중인 캐릭터 정보 등 필요한 데이터를 DB에 요청하여 **반환** 된 데이터로 **초기화 작업**을 진행하였습니다.



런타임 초기화
└ 데이터 드리븐(Xml)

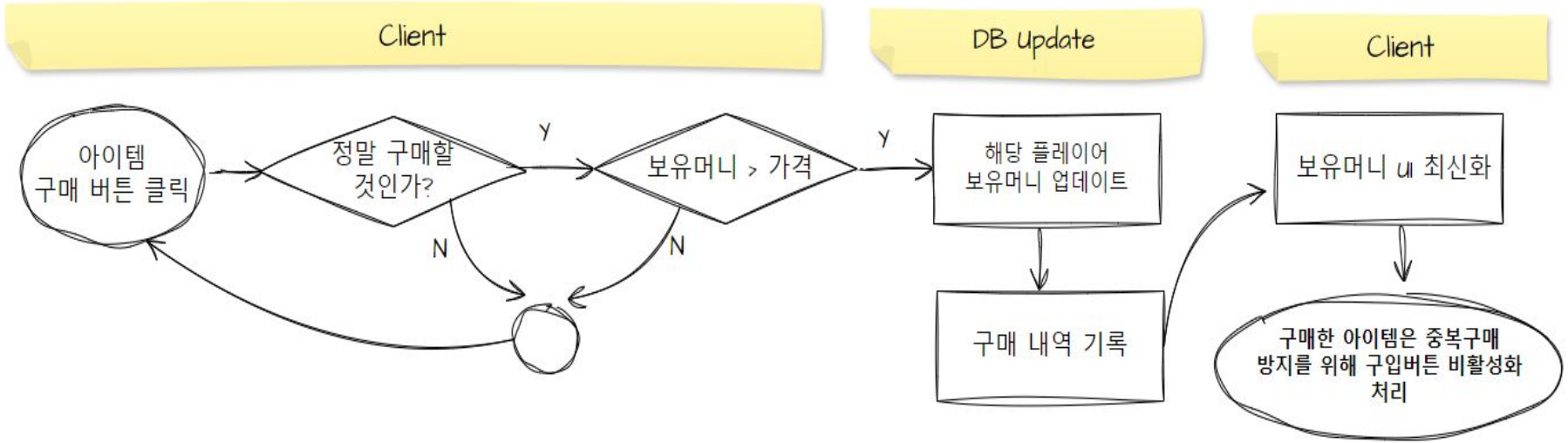
DataBase
└ 로그인 프로세스
└ **상점 프로세스**

UI
└ UIManager

포톤 Network (동기화)
└ 준비 / 카운트 다운 / 하이라이트 / Room / 장애물 회전

상점 - Buy Flow

- 상점 내 아이템 구매 시 보유금액에서 아이템 금액을 뺀 값으로 DB 처리 후 클라이언트 UI도 동기화 되도록 로직을 구성하였습니다.



런타임 초기화
└ 데이터 드리븐(Xml)

DataBase
└ 로그인 프로세스
└ 상점 프로세스

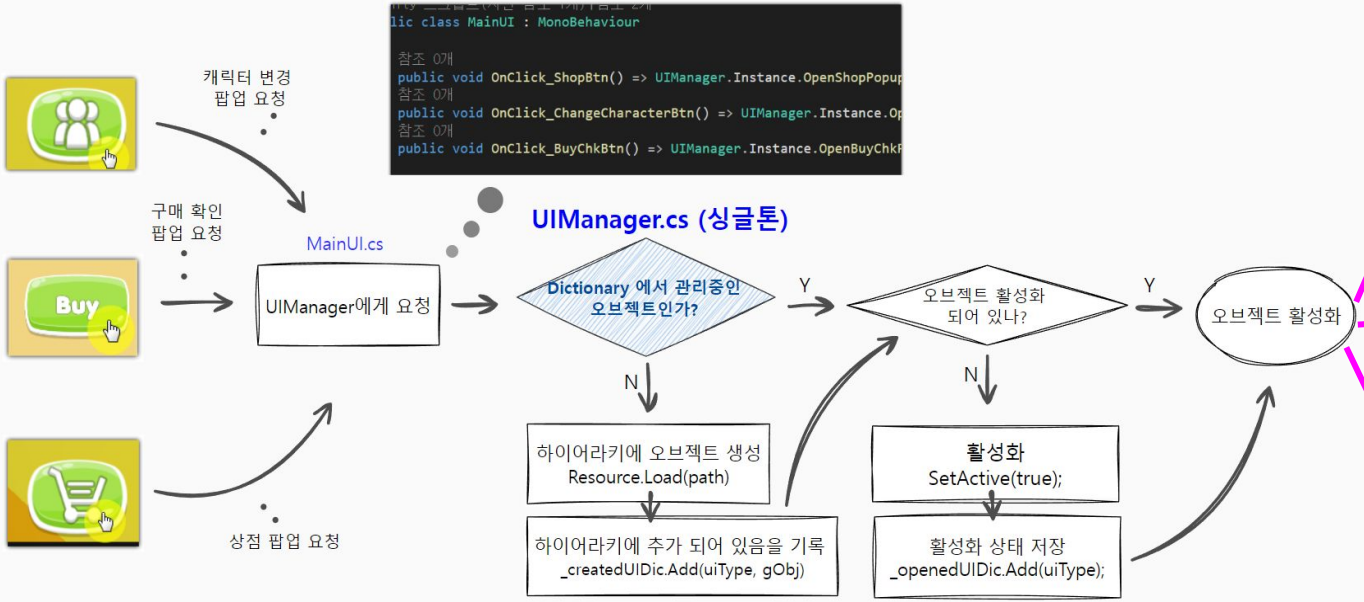
UI
└ UIManager

포톤 Network (동기화)
└ 준비 / 카운트 다운 / 하이라이트 / Room / 장애물 회전

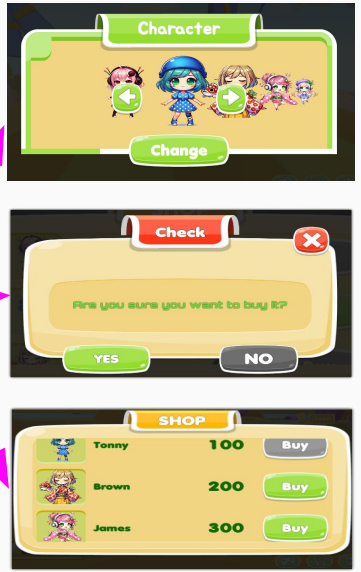
UIManager - Flow

● UIManager에서 오브젝트의 생성, 활성화, 비활성화 여부를 관리하도록 하여 **생성 타이밍**을 제어할 수 있으며, 코드 확장성을 높였습니다.

UIManager 순서도



결과



런타임 초기화
└ 데이터 드리븐(Xml)

DataBase
└ 로그인 프로세스
└ 상점 프로세스

UI
└ UIManager

포톤 Network (동기화)
└ 준비 / 카운트 다운 / 하이라이트 / Room / 장애물 회전

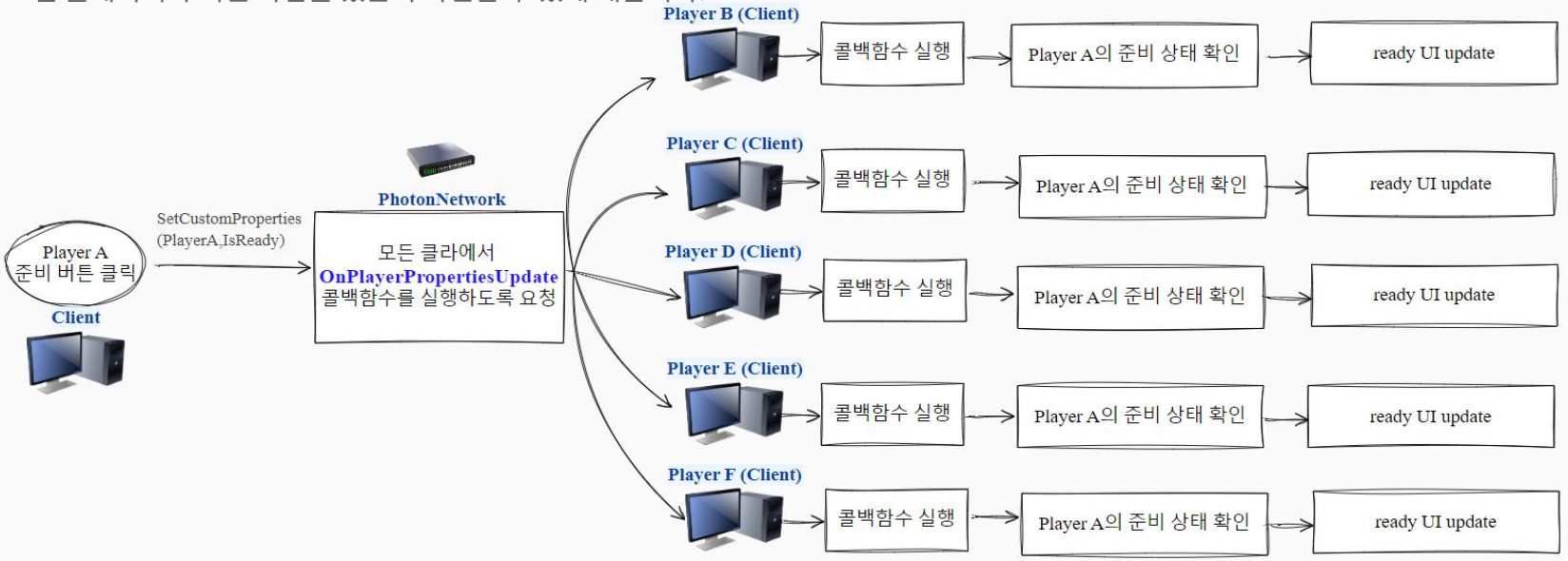
동기화 방법 3가지

- Photon Network에서 제공하는 동기화 함수 중 **갱신 빈도**에 알맞는 함수를 사용하여 서버 부하를 최소화했습니다.

갱신이 아주 드문 경우	갱신이 드문 경우	갱신이 빈번한 경우
Custom Properties	RPC	PhotonView
• 준비현황	• 카운트 다운 • 하이라이트 • Room	• 장애물 회전 • 플레이어 움직임 • 플레이어 애니메이션

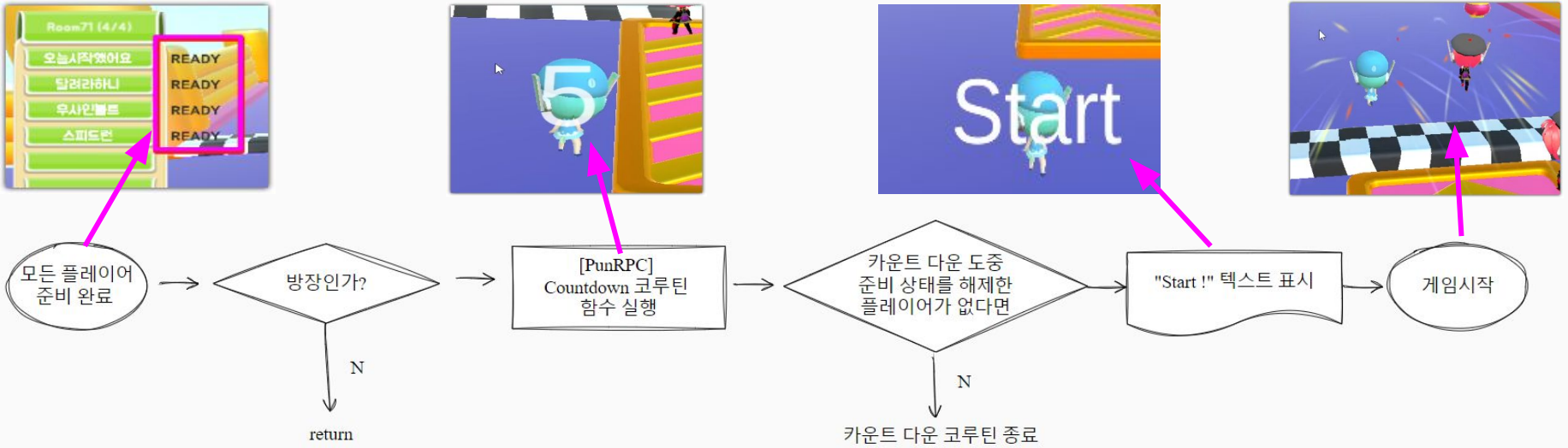
준비 - SetCustomProperties 함수 사용

- 로컬 플레이어가 준비 버튼을 눌렀을 때 모든 클라이언트가 이를 알 수 있도록 SetCustomProperties 함수를 사용했습니다.
- 포톤에서 제공하는 SetCustomProperties 함수는 **모든 클라이언트**에서 OnPlayerPropertiesUpdate **콜백 함수**를 실행하며, 특정 로컬 플레이어가 어떤 작업을 했는지 확인할 수 있게 해줍니다.



카운트 다운 - IsMasterClient 속성 사용

- 모든 클라이언트가 동일한 시간에 정확히 카운트다운을 시작하여 **동시에 출발**할 수 있도록 포톤 네트워크의 **IsMasterClient** 속성을 사용하여 방장이 카운트다운 함수를 실행하도록 했습니다.



런타임 초기화
└ 데이터 드리븐(Xml)

DataBase
└ 로그인 프로세스
└ 상점 프로세스

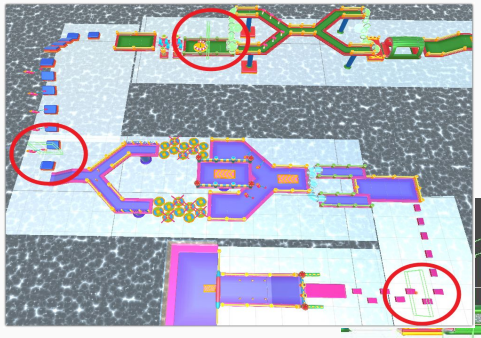
UI
└ UIManager

포톤 Network (동기화)
└ 준비현황 / 카운트 다운 / **하이라이트** / Room / 장애물 회전

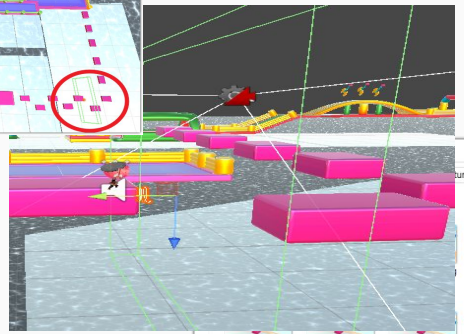
하이라이트 - Flow

1. **구간별 이미지 캡처**
2. 플레이어 도착 (카운트다운 > 게임종료 > 순위 집계)
3. 1위 플레이어의 클라이언트에서 PNG > 동영상 변환
4. 구글 드라이브에 동영상 업로드
5. 업로드 주소 반환
6. 모든 클라이언트에게 RPC로 반환 주소 동기화
7. 컷씬 실행
8. 모든 클라이언트에서 하이라이트 영상 재생

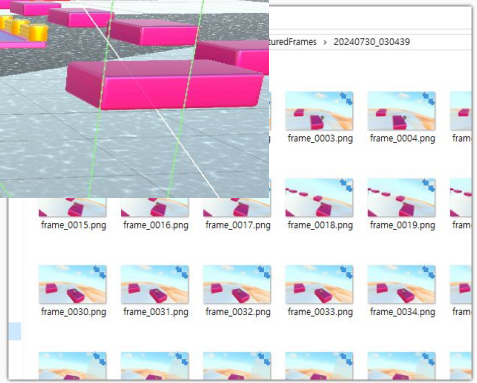
ScreenshopPoint



BoxCollider



PNG 파일 저장



런타임 초기화
└ 데이터 드리븐(Xml)

DataBase
└ 로그인 프로세스
└ 상점 프로세스

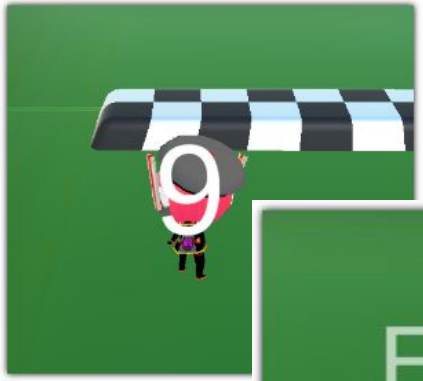
UI
└ UIManager

포톤 Network (동기화)
└ 준비현황 / 카운트 다운 / **하이라이트** / Room / 장애물 회전

하이라이트 - Flow

1. 구간별 이미지 캡처
2. **플레이어 도착** (카운트다운 > 게임종료 > 순위 집계)
3. 1위 플레이어의 클라이언트에서 PNG > 동영상 변환
4. 구글 드라이브에 동영상 업로드
5. 업로드 주소 반환
6. 모든 클라이언트에게 RPC로 반환 주소 동기화
7. 컷씬 실행
8. 모든 클라이언트에서 하이라이트 영상 재생

게임 종료 카운트 다운 10초



게임 종료

순위 집계

런타임 초기화
└ 데이터 드리븐(Xml)

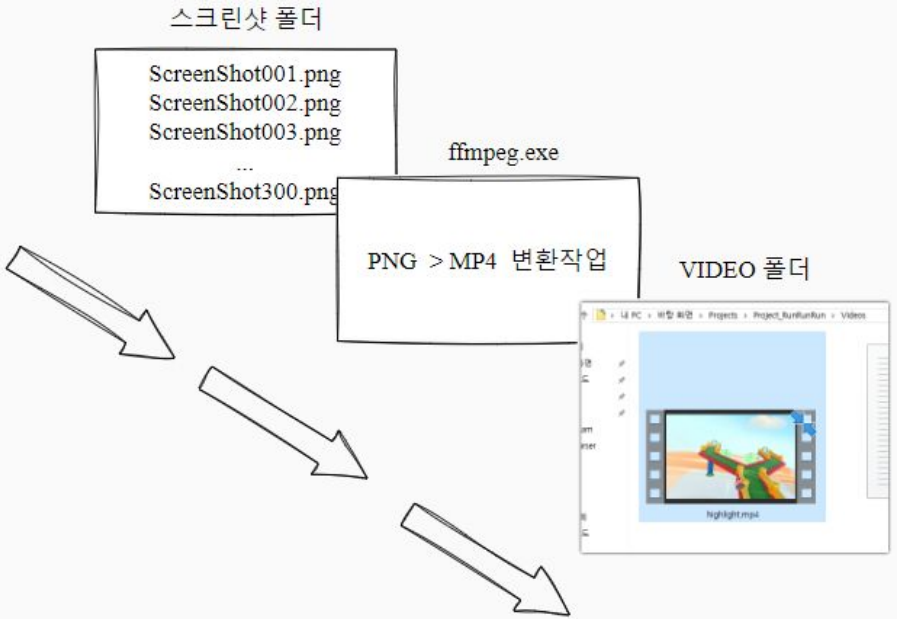
DataBase
└ 로그인 프로세스
└ 상점 프로세스

UI
└ UIManager

포톤 Network (동기화)
└ 준비현황 / 카운트 다운 / **하이라이트** / Room / 장애물 회전

하이라이트 - Flow

1. 구간별 이미지 캡처
2. 플레이어 도착 (카운트다운 > 게임종료 > 순위 집계)
3. **1위 플레이어의 클라이언트에서 PNG > 동영상 변환**
4. 구글 드라이브에 동영상 업로드
5. 업로드 주소 반환
6. 모든 클라이언트에게 RPC로 반환 주소 동기화
7. 컷씬 실행
8. 모든 클라이언트에서 하이라이트 영상 재생



런타임 초기화
└ 데이터 드리븐(Xml)

DataBase
└ 로그인 프로세스
└ 상점 프로세스

UI
└ UIManager

포톤 Network (동기화)
└ 준비현황 / 카운트 다운 / **하이라이트** / Room / 장애물 회전

하이라이트 - Flow

1. 구간별 이미지 캡처
2. 플레이어 도착 (카운트다운 > 게임종료 > 순위 집계)
3. 1위 플레이어의 클라이언트에서 PNG > 동영상 변환
4. **구글 드라이브에 동영상 업로드**
5. 업로드 주소 반환
6. 모든 클라이언트에게 RPC로 반환 주소 동기화
7. 컷씬 실행
8. 모든 클라이언트에서 하이라이트 영상 재생

런타임 초기화
└ 데이터 드리븐(Xml)

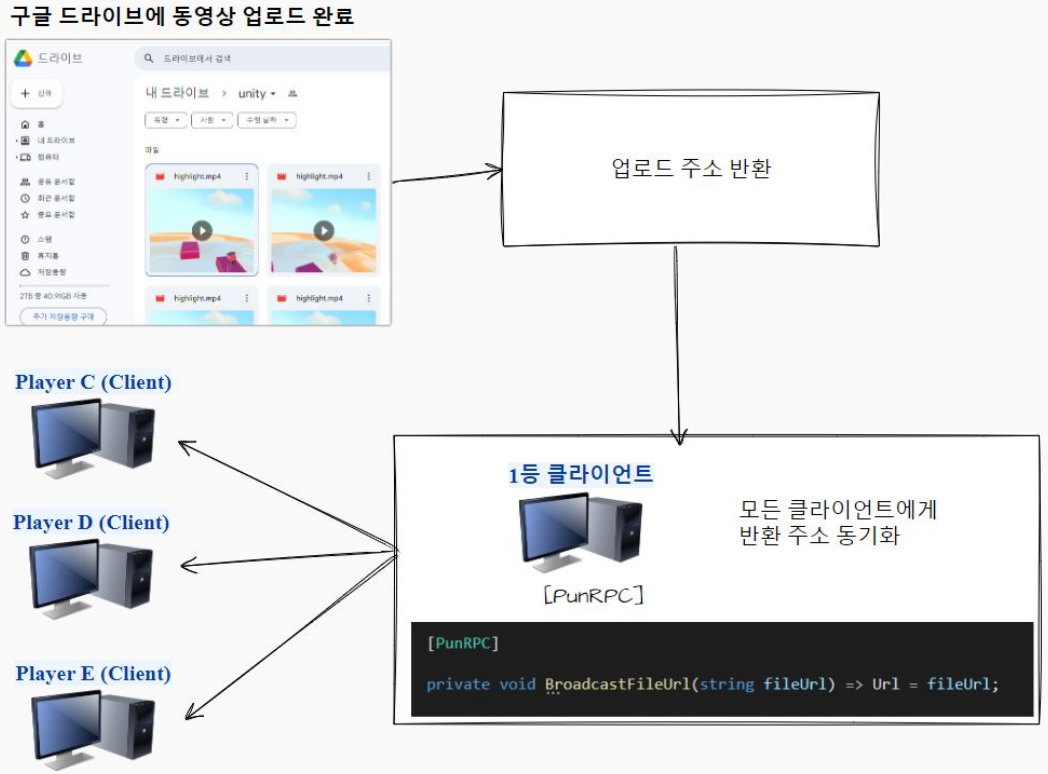
DataBase
└ 로그인 프로세스
└ 상점 프로세스

UI
└ UIManager

포톤 Network (동기화)
└ 준비현황 / 카운트 다운 / **하이라이트** / Room / 장애물 회전

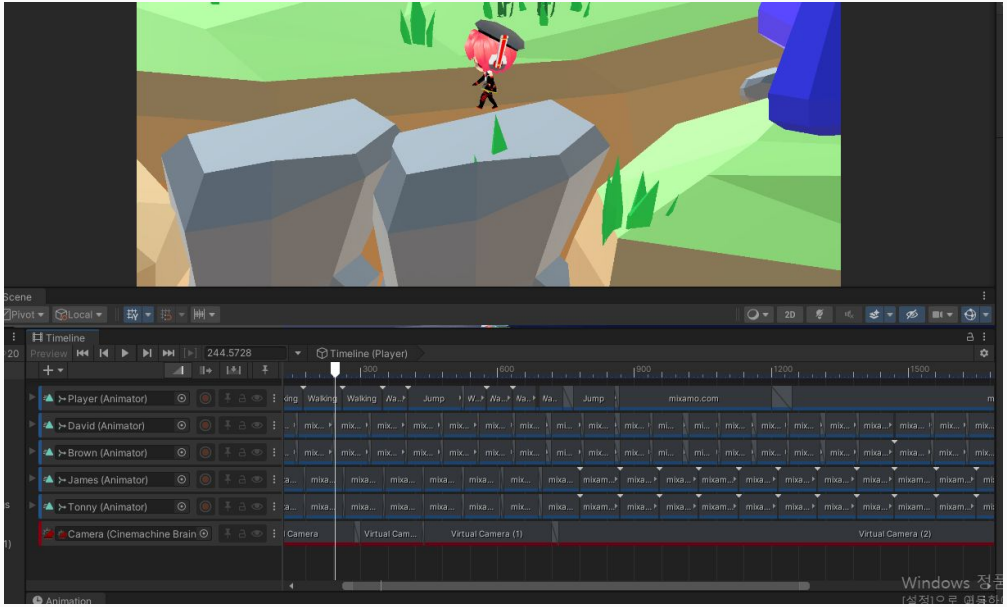
하이라이트 - Flow

1. 구간별 이미지 캡처
2. 플레이어 도착 (카운트다운 > 게임종료 > 순위 집계)
3. 1위 플레이어의 클라이언트에서 PNG > 동영상 변환
4. 구글 드라이브에 동영상 업로드
5. **업로드 주소 반환**
6. **모든 클라이언트에게 RPC로 반환 주소 동기화**
7. 컷씬 실행
8. 모든 클라이언트에서 하이라이트 영상 재생



하이라이트 - Flow

1. 구간별 이미지 캡처
2. 플레이어 도착 (카운트다운 > 게임종료 > 순위 집계)
3. 1위 플레이어의 클라이언트에서 PNG > 동영상 변환
4. 구글 드라이브에 동영상 업로드
5. 업로드 주소 반환
6. 모든 클라이언트에게 RPC로 반환 주소 동기화
7. **컷씬 실행**
8. 모든 클라이언트에서 하이라이트 영상 재생



하이라이트 - Flow

1. 구간별 이미지 캡처
2. 플레이어 도착 (카운트다운 > 게임종료 > 순위 집계)
3. 1위 플레이어의 클라이언트에서 PNG > 동영상 변환
4. 구글 드라이브에 동영상 업로드
5. 업로드 주소 반환
6. 모든 클라이언트에게 RPC로 반환 주소 동기화
7. 컷씬 실행
8. **모든 클라이언트에서 하이라이트 영상 재생**

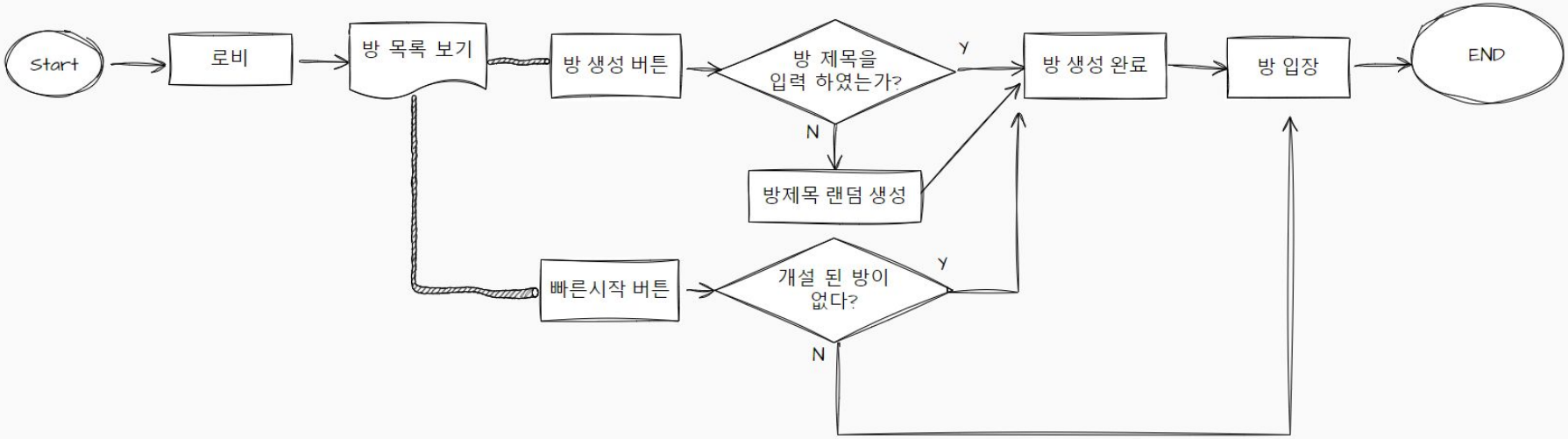
런타임 초기화
└ 데이터 드리븐(Xml)

DataBase
└ 로그인 프로세스
└ 상점 프로세스

UI
└ UIManager

포톤 Network (동기화)
└ 준비현황 / 카운트 다운 / 하이라이트 / **Room** / 장애물 회전

Room - 방 생성 Flow



런타임 초기화
└ 데이터 드리븐(Xml)

DataBase
└ 로그인 프로세스
└ 상점 프로세스

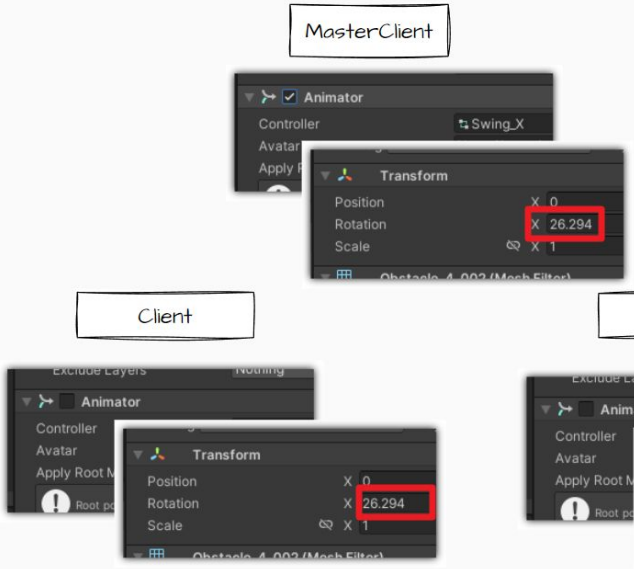
UI
└ UIManager

포톤 Network (동기화)
└ 준비현황 / 카운트 다운 / 하이라이트 / Room / **장애물 회전**

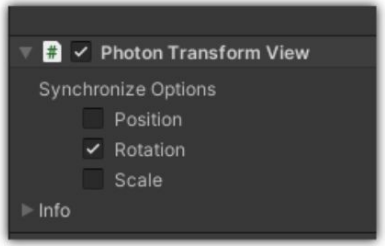
장애물 회전 - MasterClient

- 모든 클라이언트의 장애물이 **동일한 회전값**을 갖도록 하기 위해 **MasterClient**인 경우에만 애니메이터를 활성화 하였습니다.

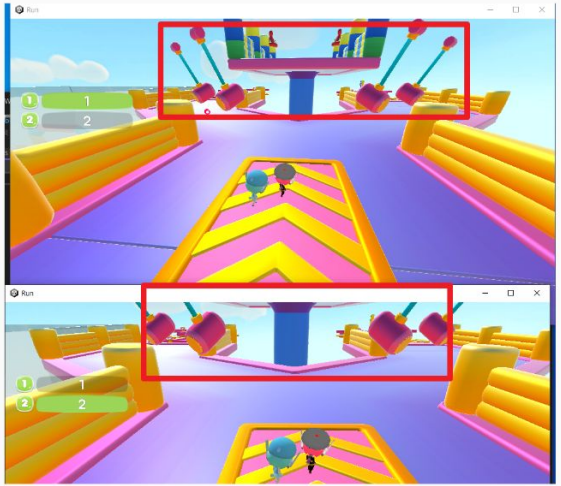
1. MasterClient의 오브젝트만 애니메이터 활성화



2. 오브젝트의 회전 값 모든 클라이언트에게 동기화



3. 동기화 된 모습



이슈해결

1. 이해하지 못 한 코드 사용 2. 프레임 드랍 현상

문제현상

Unity 에디터 환경에서는 캐릭터 변경 팝업창이 정상적으로 열리지만, 빌드된 파일에서는 팝업이 열리지 않는 문제가 발생했습니다.

해결방법

작성한 코드에서 #if UNITY_EDITOR 줄을 제거했습니다.

결과

빌드된 파일에서도 캐릭터 변경 팝업이 정상적으로 열리는 것을 확인했습니다.

느낀점

#if UNITY_EDITOR 코드의 의미를 충분히 이해하고 작성했다면, 문제의 원인을 더 빨리 찾을 수 있었을 것 같습니다.

예상하지 못했던 부분에서 문제가 발생해 어려움을 겪어보면서 앞으로는 이해하지 못 한 코드를 작성하지 않아야 겠다고 생각했습니다.

이슈해결

1. 이해하지 못 한 코드 사용 **2. 프레임 드랍 현상**

문제현상

게임 시작 후 특정 구간을 지날 때 하이라이트 영상용 동영상 파일을 제작하기 위해 동영상 녹화가 되도록 하였는데 실시간으로 영상을 녹화 하다 보니 프레임 드랍 현상이 심해 게임을 플레이 하는데 많은 지장을 주었습니다.

해결방법

동영상을 녹화하는 방법 대신 스크린샷을 여러장 캡쳐 하도록 하였습니다.

결과

프레임 드랍 현상이 해결되었습니다.

느낀점

핵심 기능인 하이라이트 기능을 반드시 구현해야 하는 상황에서, 프레임 드랍 현상이 발생하였습니다. 이 문제를 해결하기 위해 다른 동영상 녹화 에셋을 찾아보았지만, 적합한 해결책을 찾지 못했습니다. 그래서 하이라이트 동영상을 제작하려는 원래 의도에 맞춰 기능을 차분하게 다시 설계하였고, 결국 문제를 해결할 수 있었습니다. 이번 경험을 통해 문제가 발생했을 때 차분하게 생각하는 힘을 기를 수 있었습니다.

감사합니다 !