창직종합설계프로젝트 중간보고서

B611125 유민호, B611064 민정인, B611154 이유림

목차

1. 프로젝트 개요
2. 진행상황
   1. 인게임
   2. 서버
   3. DB
3. 현재 발생한 문제점
4. 앞으로의 계획
   1. 인게임
   2. 서버
   3. DB

1. 프로젝트 개요

제안서 내용에서 변동 없음

게임 제목 : Do Mine

게임 장르 : 2D 온라인 소셜 디덕션 게임

플레이 시간 : 8~15분

플레이 인원 : 3~10인

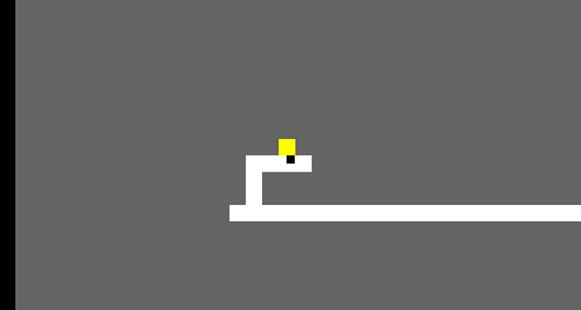
게임 컨셉 : 사보타지와 광부 두 팀이 경쟁하며 펼치는 마피아게임

2. 진행상황

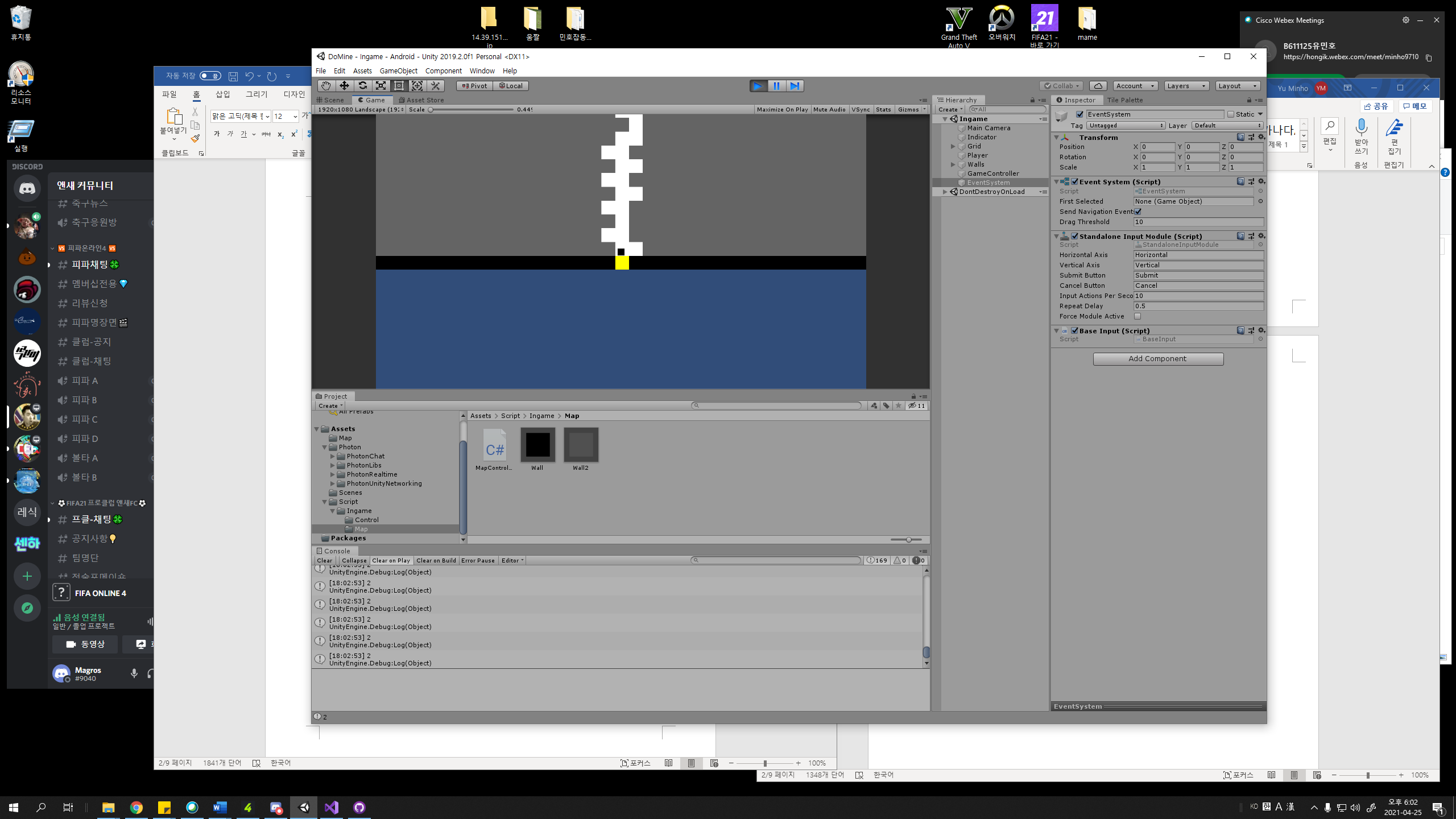
A. 인게임 구현

현재 인게임 개발은 1인클라이언트 위주로 개발하고 있으며, 기본적인 조작과 최초 게임판 생성, 게임진행에 관련되는 상호작용구현을 목표로 하고 있다. 이후 Photon과의 연결을 통해 멀티플레이를 구현할 예정이다.

로비-타이틀씬, 인게임씬 중에서 현재 인게임씬 위주로 제작중이다.



<인게임 스크린샷1 : 굴을 파는 모습>



<인게임 스크린샷2 : 맵 끝에 막힌 모습>

오브젝트 구조

* MainCamera – 플레이어를 쫓아가는 카메라
* Indicator – 플레이어에 근접한 벽을 표시해주는 오브젝트 근접한 벽이 없을시엔 비활성화
* Player – 조작 가능한 플레이어, 충돌하는 물리를 가짐
* Grid – 맵의 배경을 이루는 타일맵을 저장하는 오브젝트
* Walls – 동적으로 생성될 Wall 객체에 대한 부모 객체 빈 오브젝트로 선언
* GameController – 인게임상의 모든 스크립트를 실행시킬 객체 빈 오브젝트로 선언
* EventSystem – Unity의 기본적인 EventSystem객체
* Wall1, 2 – 프리팹화된 오브젝트로 동적으로 생성되며 충돌하는 물리는 가졌지만 좌표는 고정된다

스크립트 구조

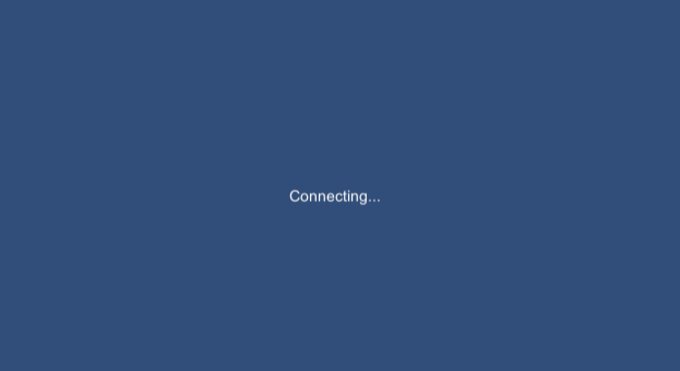
* Ingame
  + CamereController – 플레이어를 따라가는 카메라관련 스크립트
  + GameController – 게임 전체적 진행관련 트리거가 들어갈 스크립트
  + Control
    - PlayerControl – 플레이어가 조작하는 모든 조작은 이 스크립트를 통해 이루어짐, 다른 객체와의 상호작용은 이 스크립트의 클래스에 관련 클래스를 추가하여 함수를 호출하는 식으로 이루어짐
  + Map
    - MapController – 벽 생성과 근접한 벽 찾기 맵의 정보 저장 등 맵과 관련된 처리는 이 스크립트에서 이루어짐

함수 설명

* MapController
  + 맵은 동일한 크기의 이차원 배열에 int형으로 저장되며 각 배열의 인덱스는 실제 좌표와 1대1 대응한다. (0 : 비어있음, 1 : 파괴가능벽, 2 : 파괴불가벽 으로 저장된다.)
  + MakeMapArr 함수에서 일정 크기의 배열을 생성하는데 중앙의 4칸을 비우고 외각선은 파괴불가벽으로 내부는 파괴가능한 벽이 되도록 설정한다. (\*추후에 이는 랜덤요소배치나 미로등의 요소를 추가하여 다른 모양으로 생각해볼예정)
  + CreateMap 함수는 전달받은 배열을 토대로 해당 좌표에 프리팹화된 오브젝트를 동적으로 생성하는 함수
  + FindWall 함수는 프레임Update당 한번씩 호출되는 함수이며 현재 플레이어와 거리를 비교해서 가장 가까운 벽을 찾아서 좌표를 업데이트하고 Indicator를 해당위치로 옮겨주는 함수
  + DestroyWall 함수는 매개변수로 좌표를 받은뒤에 해당되는 좌표의 오브젝트를 삭제하고 배열도 빔처리 하는 함수 (현재는 플레이어가 요청하는 사안만 처리하지만 추후에는 다른 상황에서도 호출하려 한다.)
* PlayerControl
  + 프레임당 입력 명령을 받아서 처리 방향키 입력시엔 해당 방향으로 AddForce함수를 이용해 가속시킴 A키 입력시엔 현재 가장 가까운 벽을 일정거리 이내에 들어왔을시 파괴시키는 함수 호출
* CameraController 단순한 카메라를 Player에게 부착해서 움직일때마다 쫓아가게 하는 함수
* GameController 아직 구현되지 않음. 추후 게임 진행시간이나 플레이어 구별등등의 처리를 할 스크립트

B. 서버 구현

서버 개발의 경우 Photon 엔진을 Unity에 활용한 PUN(Photon Unity Network)를 사용 중이다. 기본적으로 로비 및 로비 UI 구현과 함께, 해당 유저 이름을 저장하여 서버에 내보내고 원하는 룸에 참여할 수 있도록 하는 것이 목표이다.. 서버 구현은 인게임 구현과 별개로 진행 후 구현이 완료되었을 때 결합할 예정이다.

<Lobby 씬 형태>

현재 PUN에 대한 이해와 서버 구축에 대한 학습이 필요하여 Photon 사이트에 있는 기술 문서를 참고하고 있다. 현재 플레이 시 닉네임을 작성 한 뒤 Play 버튼을 누르면 이후 중지하고 다시 플레이할 경우 닉네임을 저장되도록 만들어져 있고, Play 버튼을 눌렀을 때 Connecting 텍스트를 출력하도록 하고 있다.

Launcher.cs : 서버를 구동 시켜주는 스크립트로, 실행 시 자동적으로 시작하도록 구현.

PlayerNameInputField.cs : 이름을 입력하고, 해당 이름을 저장하는 스크립트로, 실행 시 Name InputField에 텍스트를 입력하고 Play 버튼을 누르게 되면 Name InputField의 Text component에 저장되어 이후 실행 시 해당 텍스트가 저장되어 출력됨.

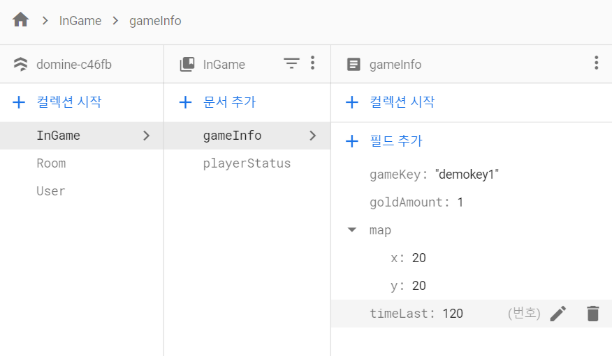
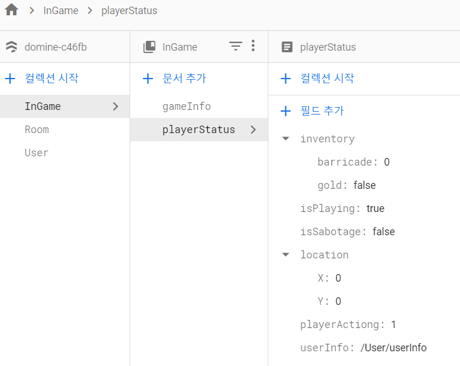
Control Panel : Play Button과 Name InputField를 제어하는 부분으로, 기존에는 true 상태를 유지하다가 Play 버튼 클릭 시 Launcher의 Start를 호출하여 false 상태로 바뀌어 존재하던 Name InputField와 Play 버튼이 화면에서 사라짐.

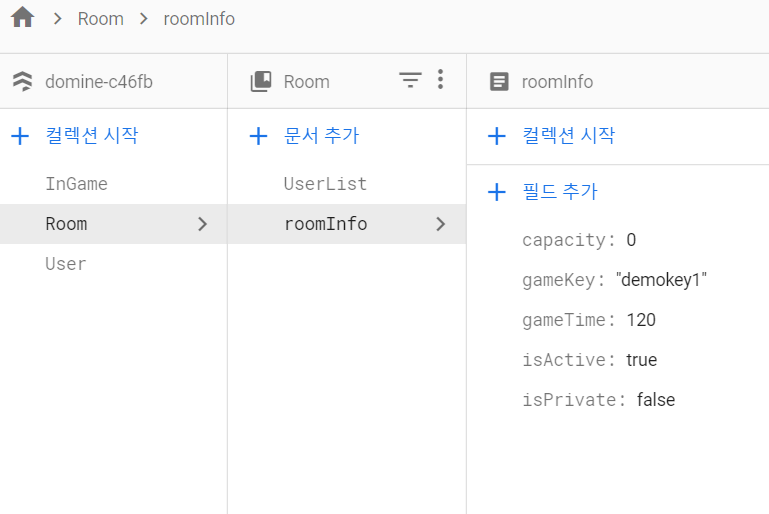
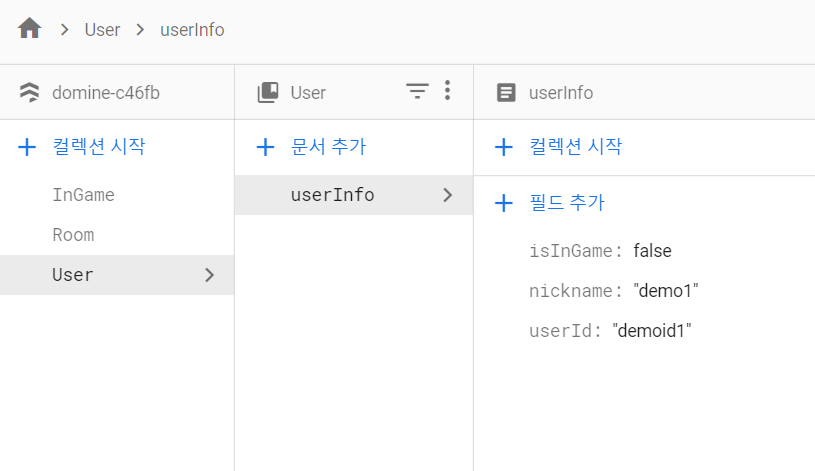
Progress Label : 대기를 위해 구현된 것으로, 기존에는 false 상태를 유지하다가 Play 버튼 클릭 시 Launcher의 Start가 호출되면 true 상태로 바뀌어 저장해둔 텍스트가 출력됨.

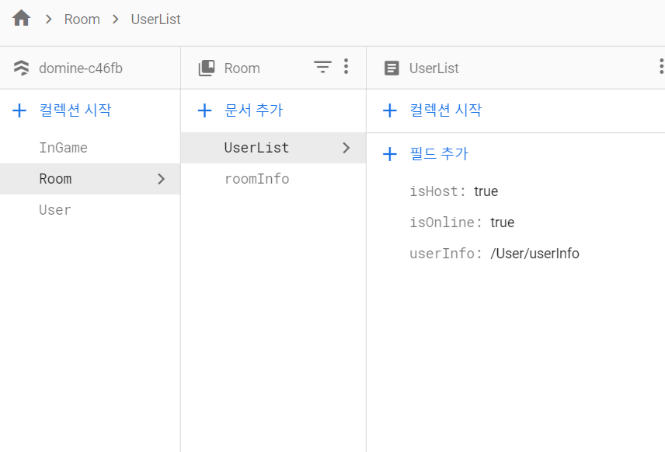
C. DB 구현

DB 개발의 경우, FireBase를 이용하여 플레이 할 때 맵에 존재하는 오브젝트들의 데이터를 저장하고, 해당 데이터의 갱신이 일어날 경우 DB도 함께 갱신되도록 하는 것이 목표이다.

예를 들어 게임 시작 시 사보타지가 지정되었을 때, 플레이어 중 사보타지로 지정된 사람의 정보가 DB에서 Bool isSabotage의 값을 true로 갱신되도록 한다.

 <DB 구조에 대한 스크린샷>

변수별 설명

User :플레이어 정보 저장. Room과 InGame field에서 참조

* userInfo
  + Bool isInGame : 유저 현재 게임 참가 여부
  + String nickname : 닉네임 정보
  + String userId : 유저 고유 키.

Room

* Bool isActive : 현재 방 활성화 상태
* int capacity : 방 최대 인원
* int gameTime : 게임 시간
* String gameKey : 방 고유 키
* UserList []
  + Bool isOnline : 유저 현재 접속상태 (장기 미응답시 강퇴)
  + Bool isHost : 호스트인지를 나타냄
  + userInfo : User field 내용 참조
  + Bool isPrivate : 공개/비공개 방 여부

Ingame

* String gameKey : 방 고유 키
* map : 맵의 구조를 저장
  + int x : 맵 크기 x 좌표
  + int y : 맵 크기 y 좌표
* int timeLast : 남은 시간
* int goldAmount : 게임 내 금의 개수
* playerStatus[] : 플레이어의 상태를 저장
  + userInfo : User field 내용 참조
  + location
    - int x : 유저 x 좌표값
    - int y : 유저 y 좌표값
  + Bool isSabotage : 현재 유저가 사보타지인지 확인
  + int playerActing : 현재 유저가 하고있는 행동 저장(중립,걷기,채굴,사보타지활동등)
  + bool isPlaying : 현재 유저가 활동 하고 있는지 저장
  + inventory : 보유한 아이템 일람
    - gold : 금
    - int barricade : 바리케이드

3. 현재 발생한 문제점

현재 게임 구현에는 큰 문제점 없이 진행되고 있으나, Photon이라는 플랫폼이 생각보다 많은 기능을 내포하고 있어 기존에 작업중이던 FireBase의 필요성에 대해 재고하게 됨.

우선적으로 Photon이 클라우드로 멀티플레이 서비스 구현을 지원하는데, 우리가 당장 필요한 인게임상의 Db관리에 있어서 FireBase를 사용할 필요성이 없어짐.

4. 앞으로의 계획

1. 인게임
   1. 우선순위 높음
      * + 추가적인 상호작용으로 아이템 획득 및 사용을 추가 할 예정이고, 기초적인 조작이 완료되었으므로 게임의 전반적인 시스템을 구현할 계획이다.
        + 이후 게임의 UI로 넘어가서 미니맵, 시야, 캐릭터의 상태 관련 변수 시각화를 구현할 계획이다.
        + 이로써 게임에 대한 전체적인 구현이 완료되면, 서버 및 DB의 구현 완료와 함께 프로젝트의 최종 목표였던 멀티플레이를 구현할 계획이다.
   2. 부가적인 계획
      * + 게임 전체적인 그래픽과 사운드 작업
        + 모바일로의 이식 작업(가상 조이스틱 등 모바일 전용 조작 구현)
        + 추가적인 상호작용(아이템 추가)
2. 서버, DB
   1. 우선순위 높음
      * + Photon
          - 매치메이킹 시스템 구현(Host, Join, 빠른 입장 등)
          - 멀티플레이시 유저간 동기화
   2. 부가적인 계획
      * + FireBase 활용 및 부가적인 서버를 구축하여 인게임 플레이 이외의 요소들(유저 스코어 기록, 친구 현황 등)을 구현

5. 참고문헌

* Unity Manual : <https://docs.unity3d.com/kr/current/Manual/UnityManual.html>
* PUN Tutorial : <https://doc.photonengine.com/ko-kr/pun/v2/demos-and-tutorials/pun-basics-tutorial/intro>