창직종합설계프로젝트1 최종보고서

Do Mine

B611125 유민호

B611064 민정인

B611154 이유림

목차

1. 프로젝트 개요

2. 프로그램 계획

3. 현재 진행상황

4. 계획과의 비교

5. 추후 계획

6. 참고 문헌

1. 프로젝트 개요

* 1. 개요
     1. 게임 제목 : Do Mine

1. 게임 장르 : 2D 온라인 소셜 디덕션 게임
2. 플레이 시간 : 15분 내외
3. 플레이 인원 : 3~10인
4. 플랫폼 : Unity, Photon
   1. 프로젝트 선택 이유
      1. 코로나 19 확산에 따른 비대면 활동 지향 추세
      2. 시간과 장소에 따른 제한이 적은 온라인 게임의 특성을 보드게임 ‘사보타지’에 적용
   2. 게임 시스템 컨셉
5. 소모임 규모 인원이 친목 목적으로 즐길 수 있는 마피아게임
6. 게임 플레이에 피로를 느끼지 않는 15분 내외의 플레이 타임 지향
7. 게임 컨트롤 보다는 전략적인 플레이에 중점을 둠
8. 광부는 시간 내로 보물 상자에 있는 금을 찾아 시작지점으로 가져와야 함
9. 사보타지는 광부로 위장한 채 광부들의 활동을 방해해야 함
10. 동종 장르의 대표적인 게임들과는 다르게 사망을 통한 탈락 시스템 없음
11. 광부들도 금을 입금한 광부만 승리하므로 광부 간의 경쟁 유도
12. 사보타지와 광부 모두 아이템을 사용하여 다른 플레이어 방해 가능
    1. 게임 배경 컨셉
       1. 깊은 지하의 용암이 차오르는 금광 배경
       2. 광부들은 서로 먼저 금을 차지하기 위해 경쟁함
       3. 광부들이 금을 찾지 못하도록 사보타지들이 정체를 숨겨가며 방해함
    2. 게임 진행 개요
13. 게임 시작 시 플레이어는 사보타지나 광부 중 한 팀으로 배정됨
14. 금괴는 시작 위치를 제외한 공간에 임의로 배치되며 금괴 획득을 위해서는 벽을 제거하여 경로를 개척해야 함
15. 플레이어들은 각종 방해 아이템을 사용하여 다른 플레이어들을 방해하고 금괴를 획득함
16. 설정 된 게임 시간 내에 일정량의 금괴를 입금하면 광부의 승리로 게임 종료
17. 설정 된 게임 시간 내에 일정량의 금괴를 입금하지 못하면 사보타지의 승리로 게임 종료

2. 프로그램 계획

사용하는 개발 툴

Unity (2019.2.0f1)

Visual Studio 2019

Photon

사용언어 : C#

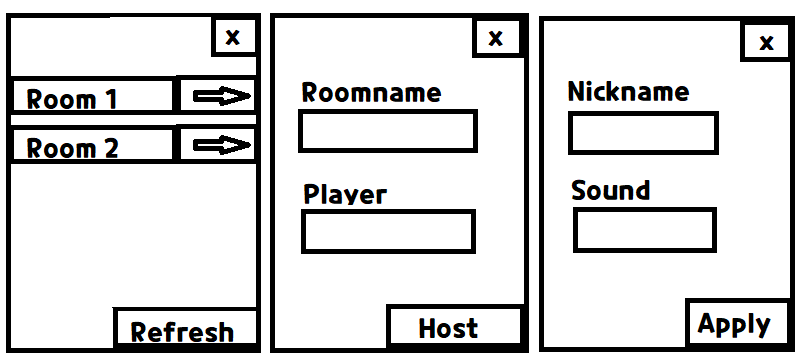
사용 라이브러리

유니티 기본 라이브러리와 Photon을 사용

프로그램 구조

유니티에선 Scene이란 단위로 구분하는데 이것을 기준으로 설명

Title Scene



(대략적인 구상도)

타이틀 씬에선 게임 실행 이후 게임 입장 이전을 다루는 씬이다

게임룸에 입장하면 Lobby씬으로 전환된다

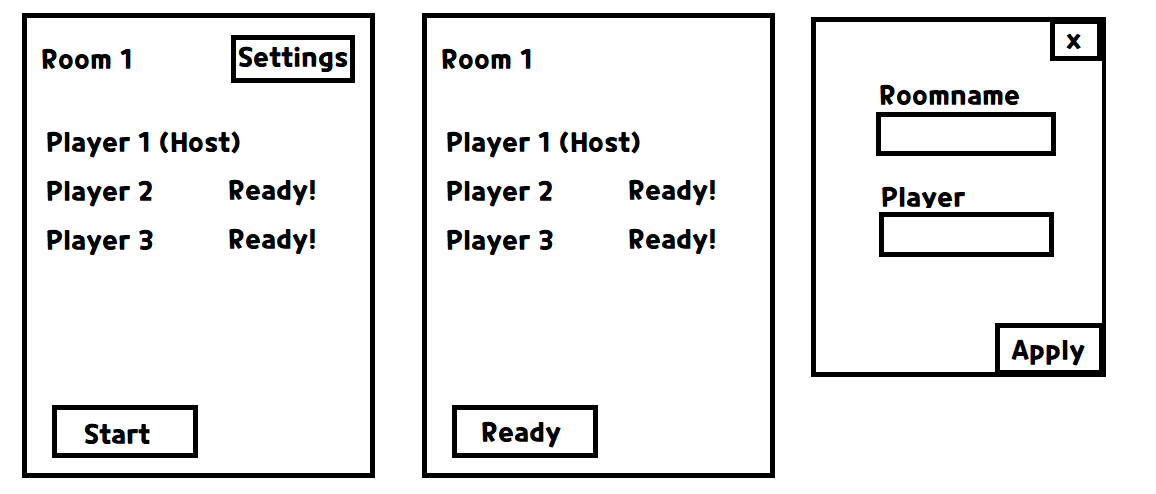
HostGame – 새로운 게임룸 생성

JoinGame – 이미 만들어진 게임 룸에 입장

Option – 닉네임이나 기타 게임 내 설정을 변경(닉네임은 타이틀에서만)

Lobby Scene

로비 씬은 게임 룸에 입장한 이후 게임 시작 전까지를 다루는 씬이다



(대략적인 구상도)

역할은 호스트와 클라이언트로 나뉘며

Exit키를 누르면 해당 게임 룸에서 연결이 해제되며 타이틀 씬으로 돌아갈 수 있다

호스트는 Setting을 이용해 방의 정보를 수정할 수 있으며

클라이언트들은 Ready키로 준비상태로 넘어갈 수 있다

호스트는 본인제외 모든 플레이어가 준비상태일 때 Start키를 눌러 게임을 시작할 수 있다

호스트가 퇴장 시 가장 순번이 높은 클라이언트가 호스트를 승계받는다

게임을 시작하면 Ingame 씬으로 넘어간다

Ingame Scene

인게임 씬은 전체적인 게임 흐름과 세세한 게임 내 요소 2가지로 나눠서 설명한다

게임진행

1. 게임시작 : 플레이어의 직업 결정, 맵 생성과 금괴배치

사보타지 : 33퍼센트내외 랜덤하게 배정하여 정확한 수를 예측하기 힘들게 한다.  
(ex 3~4 0~1인, 5~7 1~2인, 8~10 2~3인)

금괴수 : 인원대비 40~50프로 승리에는 여기서 -1개 필요  
(3~4인 2개, 5~7인 3개, 8~10인 4개)

금괴위치 : 중앙의 안전지대를 제외한 나머지 공간에 랜덤 배치

맵 크기 : 현재는 100\*100의 크기이며 외곽줄을 제외하면 99\*99의 활동가능범위

게임 시간 : 15분

목표 승자 : 사보타지든 광부든 총 게임 인원의 30~50프로가 승리 할 수 있도록 계획

2. 게임 플레이 초반 (초반 1/3) : 광부와 사보타지는 길을 개척하면서 금을 찾으면서 각자의 임무를 한다.

경로 개척 : 맵 한계 이내에선 자유롭게 벽을 제거 후 경로개척가능

비상탈출기능 : 어딘가 갇히거나 길을 잃었을 시 사용가능, 모든 소지품을 현재 위치에 내려놓고 시작위치로 귀환

보물상자 : 여러 개 생성되며 금이나 아이템을 보관하거나 빈 상자로 존재 벽과 같은 판정

3. 게임플레이 중반단계(중반 1/3) : 서로 간의 견제와 방해를 할 수 있고 금을 차지한 광부는 안전지대의 금고에 입금 후 광부승리가 되면 승리하게 됨.

사보타지들의 특수능력이 사용가능해짐

방해 아이템이 사용 가능해짐

금을 발견하면 금을 들고 이동하여 시작지점에 입금해야 함

4. 게임플레이 후반단계(후반 1/3) : 제한시간이 다가오면서 맵이 끝에서부터 좁아짐

승리에 필요한 금이 입금되야 승리하기 때문에 성공한 광부가 타 광부에 협력함

맵이 끝에서부터 파괴되는데 해당 위치의 플레이어는 자동 귀환 되고 해당 위치의 금은 파괴

5. 게임이 끝나는 조건

광부가 승리에 필요한 금을 모두 입금 (입금한 광부만 승리)

15분이 지나거나 금이 파괴되어 승리에 필요한 양보다 금이 부족한 경우 (사보타지 승리)

게임 내 요소

맵 생성

정사각형구조

해당 칸과 1대1대응이되도록 배열을 만들고 랜덤으로 값을 배정

각 장애물들은 곡괭이를 통해 제거 가능 및 바리케이드를 통해 생성 가능

보물상자의 경우, 일정범위에서 랜덤으로 생성되도록 함

기본 조작

이동

이동방식 : 가상 패드 형식

카메라이동 : 탑뷰, 캐릭터에 고정

경로 개척

캐릭터에서 가장 가까운 벽 중, 캐릭터가 바라보고 있는 방향의 벽에 표시 생성 및 키 활성화

키 입력 시 벽 파괴

벽 파괴 횟수 한정, 베이스캠프로 귀환 시 횟수 회복 가능

금괴 관련 상호작용

금괴 발견 및 범위 내에 도달할 시 버튼 활성화후 버튼 누를 시 획득

금괴 획득 시 캐릭터의 이동속도 감소

금괴 내려놓기 가능

베이스캠프 도착 시 자동으로 회수 및 점수로 환산

통신 및 게임화면 업데이트

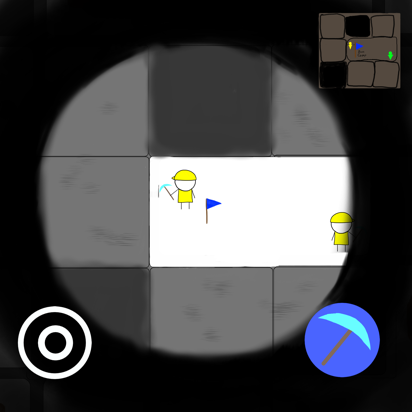
플레이어를 포함한 모든 게임 내 오브젝트들과 몇몇 정보들을 실시간으로 동기화

시야

캐릭터 주변 일정 범위 내 시야 확보

시야 범위 밖은 보이지 않고, 시야 범위 내라도 벽 존재 시 보이지 않음

UI 정보

 <- UI 예시

미니맵

현재 캐릭터의 위치, 보물상자의 위치를 표시(나침반형식)

가상패드

캐릭터를 이동시킬 때 사용

상호작용키

현재 플레이어에게 활성화된 활동을 하는 키(경로개척 금 줍기 등)

사보타지

목표

시간 내에 광부가 필요한 개수만큼 금을 얻지 못하게 막아야 한다.

특수 능력

입금과 금획득을 제외한 기본적으로 광부가 갖는 기능은 모두 사용가능

바리케이트 소환으로 광부의 이동을 막을 수 있음

광부보다 넓은 시야를 가짐

기타 기능 추가 고려(은신 등등)

3. 현재 진행상황

- 인게임

싱글 플레이 조작과 상호작용

모든 인게임 관련 스크립트는 GameController 오브젝트에 통합하여 운용

(추후 오브젝트를 통합해서 동기화할경우 GameController에 게임 기능적인 관리오브젝트를 맡기고 추가로 UserController함수를 유저마다 배정해서 사용 가능하도록 할 예정)

플레이어 이동

PlayerControl 스크립트에서 처리

Player에게 Rigidbody2D로 물리속성을 부여하고 이를 스크립트를 통해서 움직임

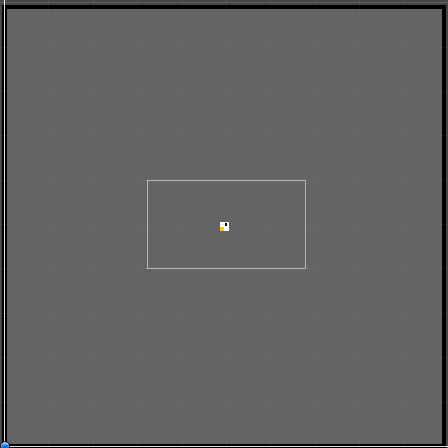
단순 이동이아닌 AddForce기능을 이용해서 public으로 선언된 Power만큼의 힘으로 해당 물체를 움직여줌

맵 관련 코드

MapController 스크립트에서 처리

맵 생성

최초에 맵의 정보를 2차원 배열로 저장하는 mapArray에 MakeMapArr()함수에서 맵 정보를 받음

이 함수는 현재는 외곽엔 파괴불가능한 벽 내부엔 파괴 가능한벽 맵 중앙 4칸엔 유저가 활동할 수 있도록 빈 공간을 만들어 줌

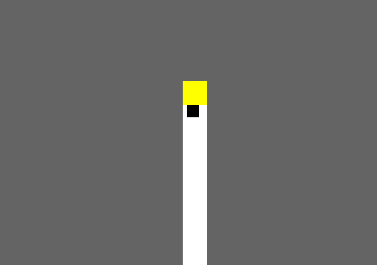
(이런 형태로 생성된다)

이를 이제 CreateMap함수에서 여기서 받은 맵 정보를 토대로 실제 생성되는 벽들이 저장되는 GameObject배열인 mapObject 배열에 Instantiate를 사용하여 저장된 프리팹을 기반으로 새로운 오브젝트를 생성하고 저장

여기서 오브젝트가 생성되는 좌표는 배열의 주소랑 그대로 대응됨

근처의 벽 추적

FindWall 함수를 프레임당 Update마다 호출하여 벽 오브젝트들의 배열인 mapObject배열과 player의 위치를 비교해서 가장 가까운 벽의 좌표와 가까운 벽의 오브젝트(nearestWall)를 갱신하고 여기서 해당좌표와 플레이어가 일정 거리 이내 일시 wallIndicator오브젝트를 해당 좌표로 활성화시켜 해당 벽으로 락온되도록 함



(현재 인근 벽에 노란색으로 indicator가 표시되는데 이때 a키로 벽을 부술 수 있다)

-> 추후 현재 플레이어의 좌표를 토대로 근처의 9개의 벽만 탐색하는 식으로 리소스 소모를 줄일 수 있을 듯함(현재 10000개의 벽들모두와 비교는 불필요한 연산소모가 많음)

벽 파괴

PlayerControl 스크립트에서 GameController에 있는 MapController의 DestroyWall함수를 호출해서 처리

해당 벽 오브젝트를 없애고 맵정보배열도 0으로 초기화시킴

또 여기에 쿨타임 기능으로 한번 벽을 파괴 할 시 일정 시간 동안 벽을 파괴하지 못하도록 설정

아이템관련 코드 ItemController 스크립트에서 처리

아이템 생성

CreateItem함수로 생성 아이템 코드와 좌표값을 받아서 새로운 아이템 클래스를 new로 생성한 뒤에 해당 위치에 알맞는 아이템 오브젝트를 생성하고 좌표값 코드 오브젝트등 정보들을 추가하여 아이템들의 리스트인 items에 추가함

아이템 삭제

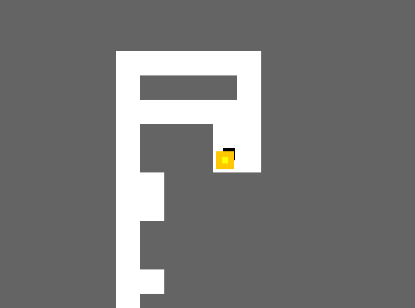
DeleteItem 함수로 아이템 클래스로 해당 오브젝트를 삭제하고 리스트에서 제외 (C#엔 Garbage Collector가 있어서 따로 new로 생성한 오브젝트를 메모리 할당 해제하는 방법이 없음)

인근아이템 추적

FindItem함수를 이용한다 기본적인 원리는 FindWall함수와 비슷하다

대신 좌표를 기준으로 for문을 돌리는것과 아이템 리스트에 있는 항목들을 foreach로 호출하는 차이가 있음

가장 가까운 아이템은 nearestItem항목에 연결되고 또 플레이어와 아이템이 일정 거리 이내라면 이 좌표에 Itemindicator가 표시됨



(근처의 아이템에 indicator가 표시됨 이때 s키로 아이템을 획득 할 수 있다)

아이템 획득

PlayerControl 스크립트에서 GameController에 있는 ItemController에서 GetItem함수를 호출해서 처리

Item클래스를 전달받아서 해당 플레이어에게 아이템을 획득한 정보를 주고 해당 아이템을 삭제시킴

전체적 게임 트리거

GameController

현재 자세한 구현이 되지않아 최초 플레이어의 위치랑 상호작용실험을 위한 금을 일정위치에 생성하는것만 구현되어있고 Time.deltaTime을 이용하여 900초의 제한시간이 작동하는 것을 구현해놓았음

카메라관련

CameraController스크립트에서 담당하고 유저의 좌표를 따라다니도록 설정

클래스로 선언된 자료구조

Item

아이템코드와 좌표가 int값으로 저장되고 실제 생성되는 게임오브젝트도 이곳에 소속됨

ItemController에서 사용할 생성자도 선언

Inventory

각 플레이어들에게 선언될 인벤토리 클래스 플레이어들의 아이템 보유목록을 나타내줄 것

현재는 금만 bool변수로 선언되어있음

Player

각 플레이어들에 대한 정보를 담은 클래스 GameController에 리스트로 생성될 예정

xy좌표값과 사보타지유무의 변수가 있고 따로 인벤토리 클래스가 선언됨

추후 게임오브젝트 연결도 할 예정

오브젝트 구조

MainCamera – 플레이어를 쫓아가는 카메라

WallIndicator – 플레이어에 근접한 벽을 표시해주는 오브젝트 근접한 벽이 없을시엔 비활성화

ItemIndicator – 플레이어에 근접한 아이템을 표시해주는 오브젝트 근접한 아이템이 없을시엔 비활성화

Players – 동적으로 생성될 Player 객체에 대한 부모 객체 빈 오브젝트로 선언

Player – 조작 가능한 플레이어, 충돌하는 물리를 가짐 현재는 프리팹화 되어있지 않으나 추후 프리팹화할것

Grid – 맵의 배경을 이루는 타일맵을 저장하는 오브젝트

Walls – 동적으로 생성될 Wall 객체에 대한 부모 객체 빈 오브젝트로 선언

Item – 동적으로 생성될 Item 객체에 대한 부모 객체 빈 오브젝트로 선언

GameController – 인게임상의 모든 스크립트를 실행시킬 객체 빈 오브젝트로 선언

EventSystem – Unity의 기본적인 EventSystem객체

Canvas - UI들을 띄울 창

Wall1, 2 – 프리팹화된 오브젝트로 동적으로 생성되며 충돌하는 물리는 가졌지만 좌표는 고정된다

Gold – 프리팹화된 오브젝트로 동적으로 생성되며 충돌판정은 없다

- 서버

플레이어 이름 설정

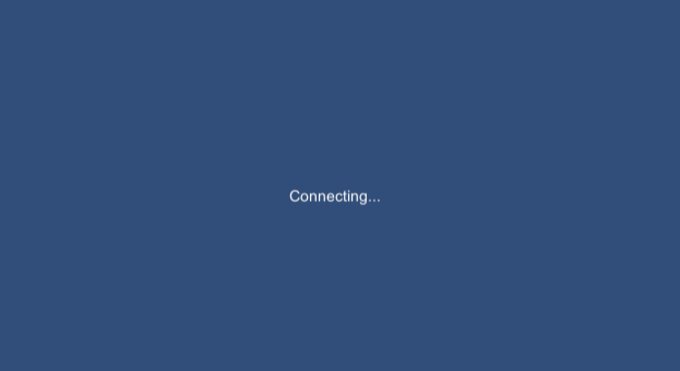
우선 싱글플레이가 아닌 멀티플레이 게임으로 인해 각 플레이어들이 자신의 캐릭터를 구분할 수 있어야 하므로 각자 플레이어 이름을 입력하도록 함. 또한 다시 플레이하게 될 때 플레이어의 이름 정보를 저장해두어 해당 텍스트를 다시 불러옴.



이로 인해 처음 로비에서 게임에 접속시 플레이어의 이름을 입력하도록 하였고, 공란으로 둘 경우 에러메시지를 출력하고 플레이가 멈추도록 함.

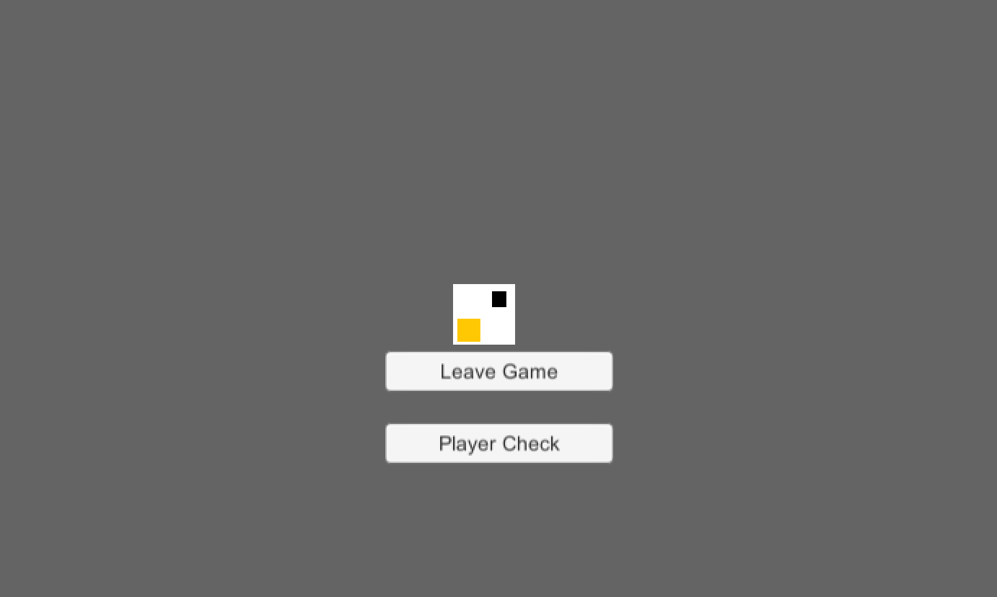
Play 버튼 클릭

Play 버튼을 클릭하게 되면, Lobby 씬에서 기존에 표시되던 텍스트 필드와 Play 버튼을 Active : True -> False 로 상태를 변화시켜 화면에 보이지 않게 하고, 실질적으로 플레이하는 씬인 Ingame씬이 완전히 준비되기 전까지 ‘Connecting…’메시지를 출력하는 Progress Label을 Active : False -> True 로 상태를 변화시켜 플레이어에게 게임으로 접속중이라는 것을 인지할 수 있도록 함.



또한 씬이 넘어가기 전 플레이어가 지정한 이름을 PhotonNetwork.Nickname 변수의 값으로 선언하고, 해당 값을 PhotonNetwork.PlayList에 입력하도록 하여 다른 플레이어가 접속할 시 뒤에 이어붙일 수 있도록 하였다. 그리고 방이 이미 있는 경우와 없는 경우를 나누어, 방이 없는 경우에는 Unity Console에 방이 없다는 메시지를 표시하고 새로운 방을 만들도록 하고, 이미 있는 경우에는 그 방으로 접속하도록 함.

Ingame 씬에서 플레이어 정보 확인



다음으로 Ingame 씬으로 넘어오게 되면 앞서 인게임 플레이에 대한 설명대로 플레이할 수 있게 되는데, Ingame 씬 기준 우하단에 위치한 Leave Game과 Player Check의 경우 Leave Game은 게임을 종료하고 Lobby 씬으로 복귀하는 버튼, Player Check 버튼의 경우 해당 게임에 있는 플레이어의 수와 해당 플레이어들의 이름을 출력하도록 함. 이 때, 앞서 저장해두었던 PhotonNetwork.PlayerList를 활용하여 각 플레이어들의 이름을 출력하는 방식을 사용함.

텍스트이(가) 표시된 사진

자동 생성된 설명

구현 중 발생한 문제점

현재까지 진행하는 부분에서 발생한 오류도 많이 존재하는데, 다른 플레이어 없이 혼자서 진행할 경우 의도한 대로 프로그램이 진행되지만, 멀티플레이에 중점을 둔 프로젝트인 만큼 멀티플레이가 중요시되는 상황에서 다른 플레이어가 Play 버튼을 클릭하고 방에 들어올 때 오류가 발생하게 됨. 또한 이러한 오류 상황 이후 Ingame씬에 들어온 경우 플레이어의 수와 해당 플레이어의 이름은 출력이 되고 있으나, Leave Game 버튼을 통해 Lobby 씬으로 돌아가는 부분에서 이미 플레이어의 상태가 게임을 떠나고 있는 상태라는 오류를 발생시키며 해당 버튼의 LeaveRoom 함수가 호출되지 못하는 상황이 발생하고 있음. 이후에는 현재 오류들을 개선하는 것에 중점을 두어야 할 것으로 생각됨.

- DB(폐기)

User : 플레이어 정보 저장. Room과 InGame field에서 참조

1. userInfo
   1. Bool isInGame : 플레이어 게임 참가 여부
   2. String nickname : 닉네임 정보
   3. String userId : 플레이어 고유 키

Room : 게임 플레이 시작 전 플레이어 모집을 위한 방 정보 저장

1. Bool isActive : 현재 방 활성화 상태
2. int capacity : 방 최대 인원 설정
3. int gameTime : 게임 종료 시간 설정
4. Bool isPrivate : 방 공개 여부
   1. String privPw : 비공개 방 입장을 위한 암호.
5. String gameKey : 방 고유 키
6. UserList []
   1. Bool isOnline : 플레이어 접속 상태 (장기 미 응답 시 강제 퇴장)
   2. Bool isHost : 호스트인지를 나타냄
   3. userInfo : User field 내용 참조

Ingame : 맵, 게임 플레이 데이터 저장

1. String gameKey : 방 고유 키
2. map : 맵의 구조를 저장
   1. int x : 맵 크기 x 좌표
   2. int y : 맵 크기 y 좌표
3. int timeLast : 남은 시간
4. int goldAmount : 게임 내 금의 개수
5. playerStatus[] : 플레이어의 상태를 저장
   1. userInfo : User field 내용 참조
   2. location
      1. int x : 플레이어 위치 x 좌표 값
      2. int y : 플레이어 위치 y 좌표 값
   3. Bool isSabotage : 플레이어 진영 확인. 사보타지일 경우 true
   4. int playerActing : 현재 플레이어가 하고 있는 행동 저장(중립, 걷기, 채굴, 사보타지 활동 등)
   5. bool isPlaying : 현재 플레이어가 플레이 하고 있는지 저장
   6. inventory : 보유한 아이템 일람
      1. bool gold : 금
      2. int barricade : 바리케이드, 획득하면 숫자가 증가하고 사용하면 감소

플레이어 데이터 저장과 맵 오브젝트 데이터 저장을 위해 사용할 예정이었으나 firebase를 사용하지 않고도 Photon만을 이용해 많은 기능 구현이 가능해 폐기하게 됨. 하지만 Photon의 경우, 게임 플레이 데이터의 저장은 지원하지 않기 때문에 랭킹 등의 추가적인 데이터 저장을 구현하게 될 경우firebase 사용을 염두에 두고 있음.

4. 계획과의 비교

1. 인게임의 경우

-게임 실행 : 서버와 연결 및 유저 고유코드 부여, 닉네임 설정 -> 닉네임과 서버 연결은 완성되었으나 현재 고유코드가 아닌, 생성한 순으로 번호가 부여됨

-메인 화면 : 현재 미구현(우선순위로 보았을 때 게임 자체가 제대로 돌아가는게 중요도에서 더 높다고 판단)

-Ingame 씬

-화면 : 현재 게임 자체의 수행에 대해 우선도를 잡아둠으로 인해 별도의 Game Start, Room Settings 등을 구현하지 않음. 다만 Leave Game 버튼을 통해 Lobby 씬으로 복귀하는 방식을 구현함.

-인게임 플레이

-플레이어 역할 설정 : 미구현

-플레이어 이동 및 상호작용 : 십자 방향키를 통해 플레이어가 이동할 수 있게 하고, 길이 없을 경우 A버튼을 눌러 노란색으로 표시된 부분으로 길을 팔 수 있음. 또한 금을 발견했을 때 S키를 이용해 금 채굴 가능.

-금 입금 : 미구현

-제한시간으로 인한 맵 크기 감소 : 제한시간의 경우 시간 설정만 구현. 맵 크기 감소는 미완성.

-게임 승리 조건 : 미구현

-맵 생성 : n\*m 형태(고정값) 구현

-게임 종료 : 미구현

\*게임의 백엔드적인 요소에 대한 기반을 다지는 데 초점을 둠. 추후 세부적인 요소들은 이를 기반으로 구현할 계획임.

2. 서버의 경우

-서버-DB간 결합(Photon - FireBase) : 현재 FireBase의 필요성 상실로 인해 계획에서 제외됨.

-매치메이킹

-빠른 참가 : 무작위 매치메이킹의 경우 현재 방의 유무에 따라 생성 or 참가는 설정하였으나, 2인 이상의 플레이어 존재시 오류 발생

-룸 옵션 : 미구현

-인게임 : 현재 플레이어 숫자, 플레이어 이름을 Unity Console에 출력하는 방식을 사용함. 캐릭터들의 상태 동기화 등은 미구현

\*현재 서버의 경우, 우선적으로 오류 수정이 가장 우선시됨. 또한 멀티플레이시 플레이어 간 동기화가 중요할 것이라 판단됨.

3. DB의 경우

현재 FireBase의 필요성 상실으로 인해 폐기

5. 추후 계획

1.게임플레이

->게임트리거 제작(시간대별 맵 상호작용(ex : 맵 크기 축소 등), 승리 조건 설정 등)

1.1 상호작용

->아이템 추가

->사보타지 역할 및 기능 추가

->비상탈출 시스템 추가

->시야 범위 추가

1.2 멀티플레이(서버)

->싱글플레이시 문제가 없었으나 멀티플레이시 발생하고있는 오류들 개선

->룸의 유무만을 판단하여 1개의 룸을 생성하는 것이 아닌, 다수의 룸 생성 및 리스트 출력, 입장 구현

->서버 – 플레이어 간 동기화

->플레이어 고유 아이템, 맵 공통 오브젝트 등

2.UI

->플레이에게 게임의 전반적인 정보를 제공할 계획

2.1 미니맵

->전체 맵의 축소판 및 캐릭터의 위치, 오브젝트의 위치 등을 표시할 수 있도록 구현

2.2 모바일이식

->조작버튼 및 가상패드 구현

3.그래픽 2학기

->Unity Asset Store 내의 프리소스 이용 예정

6. 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>