|  |
| --- |
| **종합 설계 프로젝트 수행 보고서** |

**한국공학대학교 AR 모바일 게임**

**지도교수 : 한경숙 (서명)**

**팀 번호:**

**학번 : 2017152042 이름 : 황찬욱**

**학번 : 2017156019 이름 : 염민규**

**학번 : 2018158022 이름 : 이수민**

**학번 : 2018156046 이름 : 박영훈**

**종합설계 수행 계획서 2023년 2월 25일**

목차

[Ⅰ. 팀 정보 3](#_Toc128049949)

[Ⅱ. 서론 4](#_Toc128049950)

[Ⅱ – 1 개발 주제 및 목표 4](#_Toc128049951)

[Ⅱ – 2 기존 시장 분석 6](#_Toc128049952)

[Ⅱ – 3 개발 기대 효과 8](#_Toc128049953)

[Ⅱ - 4 업무 분담 9](#_Toc128049954)

[Ⅱ - 5 개발 일정 10](#_Toc128049955)

[Ⅱ - 6 협업 Github 주소 11](#_Toc128049956)

[Ⅲ. 게임 시나리오 12](#_Toc128049957)

[Ⅲ– 1 개발 환경 및 시스템 구성도 12](#_Toc128049958)

[Ⅲ - 2. 게임 흐름도 13](#_Toc128049959)

[Ⅲ - 3. 화면 스토리 보드 15](#_Toc128049960)

[Ⅲ– 4 상세 설계 37](#_Toc128049961)

[Ⅲ– 5 구현 간 문제점 및 해결방안 56](#_Toc128049962)

[Ⅲ– 6 Prototype 구현 및 시험 결과 57](#_Toc128049963)

# Ⅰ. 팀 정보

|  |  |  |  |
| --- | --- | --- | --- |
| 이수민 | 염민규 | 황찬욱 | 박영훈 |
|  |  |  |  |
| 2018158022 | 2017156019 | 2017152042 | 2018156046 |

# Ⅱ. 서론

## Ⅱ – 1 개발 주제 및 목표

* + 주제

1. 현실의 랜드마크를 이용하여, 각 랜드마크에 배정된 AR 컨텐츠를 즐길 수 있는 AR 모바일 게임을 개발한다.
2. 이는 기존 수집형 AR 게임 형태에서 벗어나서, 랜드마크를 기반으로 사용자가 주도적으로 환경을 탐사하고 컨텐츠를 즐기는 체험형 메타버스 AR 게임 개발을 시도하는 것에 의의를 둔다.
3. 본 종합설계에서는 개발 범위 및 일정, 자본을 고려하여, ‘한국공학대학교’ 구역 내의 랜드마크를 기반으로 컨텐츠를 구현하는 것으로 개발 범위를 설정하여 주제의 효용성을 시험하고자 한다..

* 개발 목표
  1. 먼저 게임 구현의 베이스가 될 메인 씬 및 DB를 구축한다.
     1. 게임 구현을 위한 라이브러리 및 서비스, 기술 조사.
  2. 각종 미니게임을 구현한다.
     1. 현실의 오브젝트와 상호작용 가능한 AR 미니게임 설계 및 구현.
     2. 메인 씬 및 DB와 미니게임 간의 데이터 연동
  3. 여러가지 부가요소를 구현한다.
     1. 상점 및 챌린지 시스템 구현.
     2. 플레이어 데이터 저장 및 연동 기능 구현
  4. 구현한 서비스에 대한 테스팅을 수행한다.

1. 각 모듈 실험. 미니게임 등 게임 세부사항 장애요소 유무/ 실시간 게임 서버 정상적인 작동 유무/ DB와 서버의 정상적인 연결 유무 – 단위 테스트
2. 전체적인 게임 흐름 속 장애요소 유무 – 통합 테스트
3. 사용자 입장이 되어 모바일 환경에 실행파일을 설치해보고, 호환 및 정상적인 게임 수행에 지장이 없는지 테스트 – 인수 테스트
4. 여러 번 게임을 테스트하며 자잘한 버그 및 게임 밸런스 판단 – 시스템 테스트
5. 완성 이후, 게임 내부 특정 요소 업데이트 시 장애요소가 생기는지 유지보수 측면 테스트 – 리그레션 테스트

## Ⅱ – 2 기존 시장 분석

* 1. 현재 AR게임 시장 분석

1. 현재 AR게임은 휴대폰 화면으로 증강현실을 구현해 가상의 오브젝트를 수집하는, Pokemon Go와 같은 형식의 게임이 주류를 이루고 있다.
2. 현재까지 등장한 대표적인 국산 AR 게임 2가지를 분석하고자 한다.
   1. 캐치몬 – 엠게임
      1. ‘캐지몬’은 ‘포켓몬 GO’와 비슷한 수집 방식에, 보드게임 요소를 추가한 경쟁형 AR 게임이다.
      2. 수집 이후, 이를 통해 새로운 컨텐츠를 창조하였지만, 친숙하지 않은 자체 IP를 사용하여 수집 욕구를 이루어내지 못했고, 게임을 제대로 즐기기 위해서는 많은 시간을 수집에 투자해야 한다는 점 때문에 인기를 얻지 못했다.

\*(별첨) 캐치몬 – 엠게임

텍스트, 도로, 건물, 실외이(가) 표시된 사진

자동 생성된 설명

텍스트, 실내이(가) 표시된 사진

자동 생성된 설명

* 1. 소울캐쳐 – 한빛소프트
     1. ‘캐지몬’은 ‘포켓몬 GO’와 비슷한 수집 방식에, 보드게임 요소를 추가한 경쟁형 AR 게임이다.
     2. 수집 이후, 이를 통해 새로운 컨텐츠를 창조하였지만, 친숙하지 않은 자체 IP를 사용하여 수집 욕구를 이루어내지 못했고, 게임을 제대로 즐기기 위해서는 많은 시간을 수집에 투자해야 한다는 점 때문에 인기를 얻지 못했다.

지도이(가) 표시된 사진

자동 생성된 설명\*(별첨) 소울캐쳐 – 한빛소프트

1. 현재 시장에 분포한 AR 게임 형태가 ‘포켓몬 GO’와 비슷한 수집형 AR 게임 형태에 머물러 있기에, AR 게임의 새로운 활용 방안을 실험해보기 위하여, 메타버스와 결합한 체험형 AR 게임 컨텐츠를 개발하고자 함..

## Ⅱ – 3 개발 기대 효과

* + 기대 효과
    - 1. 증강 현실 (Augmented Reality) :
         * AR 기술을 활용하여, 현실 세계에서의 랜드마크를 활용하여, 가상 세계에서 새로운 컨텐츠를 창출해 낼 수 있음.
      2. 피트니스(Exercise) :
         * 사용자가 직접적으로 움직여 컨텐츠를 탐사하기에, 피트니스적인 성질을 지니고 있음..
      3. 모험(Adventure) :
         * 사용자는 주도적으로 주변 환경을 탐험하여, 각 랜드마크 별로 설정된 컨텐츠를 즐길 수 있음.
      4. 커뮤니티(Social) :
         * 본 게임은 그 자체만으로 커뮤니티적인 성질을 띄며, 게임의 결과 및 위치, 공략등은 그 자체만으로 커뮤니티의 화제가 될 수 있는 포텐셜을 가지고 있음.

## Ⅱ - 4 업무 분담

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 염민규 | 이수민 | 박영훈 | 황찬욱 |
| 자료수집 | * 뒤끝 API * Google Map API | * Unity AR Foundation Framework * Unity 기본 지원 기능 * Unity UI Toolkit | | * Unity에서의 Database 사용 |
| 설 계 | * 게임 서버 설계 | * 미니 게임 설계 * 게임 UI 구성도 | * 미니 게임 설계 * 게임 기반 설계 | * MySQL Query 설계 |
| 구 현 | * 게임 서버 구현 및 연동 | * 미니 게임 구현 * UI 구현 | * 미니 게임 구현 * 게임 기반 구현 | * DB 구축 및 서버연동 |
| 테스트 | * 각 모듈 실험. 미니게임 등 게임 세부사항 장애요소 유무/ 실시간 게임 서버 정상적인 작동 유무/ DB와 서버의 정상적인 연결 유무 – **단위 테스트** * 전체적인 게임 흐름 속 장애요소 유무 – **통합 테스트** * 사용자 입장이 되어 모바일 환경에 실행파일을 설치해보고, 호환 및 정상적인 게임 수행에 지장이 없는지 테스트 – **인수 테스트** * 여러 번 게임을 테스트하며 자잘한 버그 및 게임 밸런스 판단 – **시스템 테스트** * 완성 이후, 게임 내부 특정 요소 업데이트 시 장애요소가 생기는지 유지보수 측면 테스트 – **리그레션 테스트** | | | |

## Ⅱ - 5 개발 일정

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 항목 | 추진 사항 | 12 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 |
| 요구사항 정의 및 분석 | * 요구사항 정의 및 분석 * 요구사항 명세 |  |  |  |  |  |  |  |  |  |  |  |
| 시스템 설계 및 상세 설계 | * 시스템 설계 * 상세 설계 |  |  |  |  |  |  |  |  |  |  |  |
| 마일스톤 1 | * 게임의 베이스가 되는 기본 틀 구현 |  |  |  |  |  |  |  |  |  |  |  |
| 마일스톤 2 | * 게임 서버 구현 * 데이터베이스 구축 및 서버와 연동 |  |  |  |  |  |  |  |  |  |  |  |
| 마일스톤 3 | * 미니게임 구현 * 팀 대항 게임 구현 |  |  |  |  |  |  |  |  |  |  |  |
| 마일스톤 4 | * 게임 내 기타 요소 구현(챌린지, 상점 등) |  |  |  |  |  |  |  |  |  |  |  |
| 마일스톤 5 | * 마일스톤 1~4 결과물 연동 |  |  |  |  |  |  |  |  |  |  |  |
| 시험 및 데모 | * 테스트 진행 * 진행졸업작품 완전성 보강 |  |  |  |  |  |  |  |  |  |  |  |
| 문서화 및 발표 | * 졸업작품 중간 보고서 작성 * 발표 |  |  |  |  |  |  |  |  |  |  |  |
| 산업기술대전 | * 산업 기술대전 참가 |  |  |  |  |  |  |  |  |  |  |  |
| 졸업작품 최종 보고서 작성 및 패키징 | * 졸업작품 최종 보고서 작성 * 작품 패키징(문서, 사용법, 프로그램, 개발환경, 데모 동영상 등) |  |  |  |  |  |  |  |  |  |  |  |

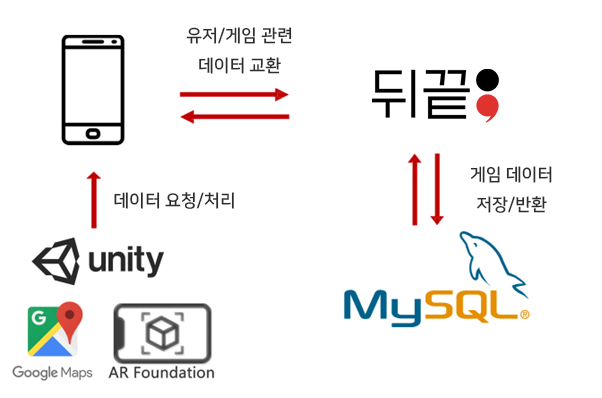
## Ⅱ - 6 협업 Github 주소

* https://github.com/ymg5218/CapstoneARtp.git

# Ⅲ. 게임 시나리오

## Ⅲ– 1 개발 환경 및 시스템 구성도

|  |  |
| --- | --- |
| 개발 요소 | 사용 기술 |
| 게임 기반 구현 | Unity |
| AR 미니게임 구현 | Unity AR Foundation |
| UI 요소 구현 | Unity UI Toolkit |
| DB | MySQL |
| 탑 다운 뷰 메인 지도 화면 | Google Map API |
| 각 랜드마크에 미니게임들 배치 | Haversine formula |
| 게임 네트워크 서버 | 뒤끝 API |
| 메신저 | 뒤끝 API |



## Ⅲ - 2. 게임 흐름도

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |  |  |
| --- | --- | --- |
| 설명 | 흐름도 | 사용 기술 |
| 게임 시작 전 |  | - Unity에서 게임 기반 구현  - Unity UI Toolkit을 이용해 UI 구현  - 게임 호스트 및 게임 참가는 Photon Cloud로 구축한 게임 서버를 이용  - 유저와 게임 정보 및 데이터 등은 게임 서버를 통해 MySQL에 저장 및 불러옴. |
| 게임 시작 준비 |  |
| 인 게임 |  | - Unity에서 게임 기반 및 팀 대항 게임 구현  - Photon Cloud로 구축한 게임 서버를 통해 실시간 게임 정보 갱신 및 유저간의 상호작용 가능  - 게임 서버와 MySQL을 연동하여 실시간 게임 정보 및 유저 정보 갱신 및 불러옴.  - Google Map API를 기반으로 탑 다운 뷰의 인 게임 메인화면 구현  - Haversine 공식을 이용하여 각 랜드마크 위치에 각종 미션들 배치  - Unity AR Foundation을 이용하여 미니게임들 구현 |
| 게임 마무리 |  | - 게임 서버를 통해 MySQL에 게임 내용 저장 및 유저 데이터 갱신  - Unity UI Toolkit을 이용해 UI 구현 |

## Ⅲ - 3. 화면 스토리 보드

|  |  |
| --- | --- |
| A1000 – 타이틀 화면 | 앱 실행시 가장 먼저 나오는 화면 |
|  | 1.’한국공학대 AR 운동회’가 적혀져 있는 희망찬 분위기의 로고  2. 아이디 입력 공간  3. 비밀번호 입력 공간, 비밀번호는 \*로 표시  4. DB와 연동하여 아이디와 비밀번호를 매칭한 후, 해당되는 사용자가 있다면 진입  5. ([A1100 – 회원가입](#A1100)) 화면으로 이동  6. 뒤로가기 누를 시 종료 팝업  7,8. ID,PW 미입력 오류 팝업  9. ID/PW 매칭 실패 팝업 |

1. A1100 – 회원가입 화면
   1. 회원 가입 데이터를 받아 처리하는 화면
   2. 화면 경로 : (A1000 – 타이틀 화면) – (A1100 – 회원가입 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 아이디 입력 공간  2. 비밀번호 입력 공간, 비밀번호는 \*로 표시  3. 닉네임 입력공간  4. 입력한 데이터로 회원가입을 시도한다.  5. 뒤로가기 누를 시 종료 팝업  6. 회원가입 성공 팝업  7. 회원가입 실패 팝업. ID/PW 미입력 또는 DB와 연결 실패 등의 이유를 오류코드로서 표기한다. |

1. B1000 - 메인 화면
   1. 게임 매칭, 게임 참가, 내 정보 확인 및 꾸미기, 규칙 설명 화면에 접근할 수 있는 중간 지점 역할의 화면
   2. 화면 경로 : (B1000 - 메인 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 사용자의 인게임 아바타, ID, 칭호를 표시한다.  2. ([B1100 – 챌린지 화면](#B1100))으로 이동  3. ([B1200 – 게임 호스트 화면](#B1200))으로 이동  4. ([B1300A – 게임 참가 화면 1](#B1300A))으로 이동  5. ([B1400 – 상점 화면](#B1400))으로 이동  6. ([B1500 – 게임 규칙 설명 화면](#B1500) 으로 이동  7. \*화면 요소 아님\* 뒤로가기 입력 시 ([A1000 – 타이틀 화면](#A1000))으로 이동  8. 1번 요소 터치 시 호출. 사용자 ID 및 획득한 칭호의 변경이 가능하다. |

1. B1100 - 챌린지 화면
   1. 도전과제, 주간 미션 등을 제공하는 화면
   2. 화면 경로 : (B1000 - 메인화면) -> (B1100 - 챌린지 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 도전과제, 일간 챌린지, 주간 챌린지 탭 구분 라벨  2. 각 챌린지의 내용과 보상 표시. 성공한 챌린지는 컬러화하여 구분이 가능하다.  3. 화면 스크롤 기능  4. \*화면 요소 아님\* 뒤로가기 입력 시, ([B1000 – 메인 화면](#B1000))으로 이동 |

1. B1200 - 게임 호스트 화면
   1. Photon Cloud로 부터 접속코드를 배정받아, 게임에 참여할 인원을 모으는 화면
   2. 화면 경로 : (B1000 - 메인 화면) -> (B1200 - 게임 호스트 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. Photon Cloud로부터 배정받은 접속 코드를 표시  2. 현재 설정된 게임 모드에 대한 정보를 보여줌  3. 현재 연결된 인원에 대한 정보를 보여줌. 목표 인원 충족 시 ([C1000A – 사전 준비 화면](#C1000A)) 으로 이동  4. 2번 요소를 클릭하면, 호스트 방의 설정 정보들을 수정할 수 있음.  5. 뒤로가기 입력 시, 메인 화면 이동 팝업 |

1. B1300A - 게임 참가 화면 1
   1. 호스트로부터 전달받은 접속코드를 입력하여 게임에 참여하는 화면
   2. 화면 경로 : (B1000 - 메인 화면) -> (B1300A - 게임 참가 화면 1)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 호스트로부터 접속 코드를 알아내어 입력하는 부분  2. 접속 코드 입력 후, 해당 코드를 통해 호스트와 연결을 시도함  3. 접속 코드 미입력시 발생하는 팝업  4. 호스트와의 연결에 실패 시 발생하는 팝업  5. 뒤로가기 입력 시, 메인 화면 이동 팝업 |

1. B1300B - 게임 참가 화면 2
   1. 호스트와의 연결에 성공하여 게임 시작까지 대기중인 화면
   2. 화면 경로 : (B1000 - 메인 화면) -> (B1300A - 게임 참가 화면 1) -> (B1300B - 게 임 참가 화면 2)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 현재 연결된 접속 코드  2. 호스트가 설정한 게임 모드에 대한 간략한 정보를 보여줌  3. 현재 연결된 인원에 대한 정보를 보여줌. 목표 인원 충족 시, ([C1000A – 사전 준비 화면](#C1000A))으로 이동  4 뒤로가기 입력 시, 메인 화면 이동 팝업 |

1. B1400 - 상점 화면
   1. 게임, 챌린지 등으로 얻은 재화를 사용하여 내 계정을 꾸밀 수 있는 화면
   2. 화면 경로 : (B1000 - 메인 화면) -> (B1400 - 상점 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 자신의 게임 재화  2. 내 아바타 표시. 해당 아바타가 인게임에서의 나의 모습이 된다.  3. 구매할 요소를 결정하는 탭  4.아이템 목록. 터치 시 구매할 수 있음  5. 화면 스크롤 기능  6. \*화면 요소 아님\* 뒤로가기 입력 시, ([B1000 – 메인 화면](#B1000))으로 이동  7. 구매 확인 팝업  8. 구매 성공 팝업  9. 구매 실패 팝업 |

1. B1500 - 게임 규칙 설명 화면
   1. 전반적인 게임 규칙을 한 화면으로 설명하는 화면
   2. 화면 경로 : (B1000 - 메인 화면) -> (B1500 - 게임 규칙 설명 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 전체적인 설명 이미지. 전체 화면을 미리 맞춤 제작한 이미지로 표시한다.  2. \*화면 요소 아님\* 뒤로가기 입력 시, ([B1000 – 메인 화면](#B1000))으로 이동 |

1. C1000A - 사전 준비 화면
   1. 본 게임에 진입하기 전 에셋 로딩, 팀 분배와 같이 게임에 필요한 요소와 환경을 자 동으로 처리하는 화면. 로딩을 위한 아이캐치 화면으로도 사용한다.
   2. 화면 경로 : (C1000A - 사전 준비 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 청팀 구성 인원. 인원은 랜덤으로 배치한다.  2. 홍팀 구성 인원. 인원은 랜덤으로 배치한다.  3. 큰 ‘VS’ 이미지  4. (C1000A – 사전 준비 화면)과 (C1000B – 게임 규칙 설명 화면)이 표시될 동안, 본 게임의 로딩을 진행한다. |

1. C1000B - 게임 규칙 설명 화면
   1. 본 게임 시작 전, 전반적인 게임 규칙을 한 화면으로 설명하는 화면. 로딩을 위한 아 이캐치 화면으로도 사용한다.
   2. 화면 경로 : (C1000A - 사전 준비 화면) -> (C1000B - 게임 규칙 설명 화면

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 전체적인 설명 이미지. 전체 화면을 미리 맞춤 제작한 이미지로 표시한다. (B1500 – 게임 규칙 설명 화면)에서 사용한 이미지를 재활용한다.  2. 전원 준비 시, ([C1000C – 카운트 화면](#C1000C))으로 이동  3. (C1000A – 사전 준비 화면)과 (C1000B – 게임 규칙 설명 화면)이 표시될 동안, 본 게임의 로딩을 진행한다. |

1. C1000C - 카운트 화면
   1. 전원 로딩이 끝났을 때, 본 게임 시작까지의 카운트 화면
   2. 화면 경로 : (C1000A - 사전 준비 화면) -> (C1000B - 게임 규칙 설명 화면) -> (C1000C - 카운트 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 카운트 요소. 카운트가 만료되면 ([D1000 – 메인 게임 화면](#D1000))으로 이동 |

1. D1000 -메인 게임 화면
   1. 인 게임에서 다양한 상호작용을 시작하는 기반 화면
   2. 화면 경로 : (D1000 - 메인 게임 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 청팀/홍팀의 현 승리 점수와 라운드 제한시간 표시  2. 현재 라운드의 각 팀이 접근 가능한 미션 수 및 보너스 상황, 포인트 상황 표시  3. 현실의 한국공학대를 맵박스로 구현하여 게임 전용 필드 생성  4. 미션 포인트. 한국공학대의 특징적인 장소(암벽 앞, 벙커 앞 등)에 미리 설치됨. 사용자는 해당 장소로 접근하여 해당 위치에 설치된 미니게임 미션을 수행할 수 있음. 각 미션은 라운드 별로 팀당 1번만 수행할 수 있고, 각각 난이도와 보상이 다르므로, 전략적으로 수행할 필요가 있음.  5. 자신의 아바타  6. 자신의 닉네임 및 칭호 표시  7. 팀 메신저. 안 읽은 메시지가 있을 시, 알림이 들어온다. 터치 시, ([D1300 - 팀 메신저 화면](#D1300))으로 이동  8. 4번 요소 터치 시 호출됨. 미션 진입 전 확인용 팝업. ‘네!’ 터치 시 ([D1100A - 미니게임 설명 화면](#D1100A))으로 이동  9. 1번 요소의 제한 시간이 다 되거나 기타 요소로 호출되는 인터럽트 팝업 화면. ‘네!’ 터치 시 (D1200A - 팀 대항 게임 사전 준비 화면)으로 이동 |
|  |

1. D1100A - 미니게임 설명 화면
   1. 미니게임 게임 시작 전, 미니게임의 규칙을 한 화면으로 설명하는 화면. 로딩을 위한 아이캐치 화면으로 사용한다.
   2. 화면 경로 : (D1000 - 메인 게임 화면) -> (D1100A - 미니게임 설명 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 전체적인 설명 이미지. 전체 화면을 미리 맞춤 제작한 이미지로 표시한다.  2. 로딩이 완료될 시, 터치하여 준비할 수 있다. 터치 시, ([D1000B – 미니게임 수행 화면](#D1100B))으로 이동 |

1. D1100B - 미니게임 수행 화면
   1. 실제로 미니게임을 수행하는 화면
   2. 화면 경로 : (D1000 - 메인 게임 화면) -> (D1100A - 미니게임 설명 화면) -> (D1100B - 미니게임 수행 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 현재 점수를 보여주는 부분  2. 바닥을 인식하여 생성되는 목표 과녁판이다. 정중앙에 맞을 수록 높은 점수를 받을 수 있다.  3. 제한 시간을 나타내는 UI이다.  4. 플레이어가 움직일 수 있는 활 오브젝트이다. 활 오브젝트를 프레스하면 활을 당기는 동작을 한다. 활을 당기는 시간이 길수록 화살 오브젝트가 더 멀리 나간다 |

1. D1100C – 미니게임 결과 발표 화면
   1. 미니게임의 결과를 확인하고, 해당하는 포인트를 얻을 수 있는 화면
   2. 화면 경로 : (D1000 - 메인 게임 화면) -> (D1100A - 미니게임 설명 화면) -> (D1100B - 미니게임 수행 화면) -> (D1100C - 미니게임 결과 발표 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 미니게임에서 얻은 포인트를 표시하는 요소  2. 점수에 따른 포인트 매칭 표  3. 전에 해당 미션을 수행한 인원이 있을 경우, 저장된 해당 유저의 기록을 그래프에 표시 |

1. D1200A - 팀 대항 게임 사전 준비 화면
   1. 팀 대항 게임 시작 전, 팀 대항 게임의 힌트와 상대와의 포인트 차이를 기반으로, 해 당 미션에 얼마나 포인트를 투자할 지 토의하는 화면.
   2. 화면 경로 : (D1000 - 메인 게임 화면) -> (D1200A - 팀 대항 게임 사전 준비 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 팀 대항 미션 시작까지 남은 시간 표시. 이 시간 동안 팀원과 토의하여 팀 대항 미션에 얼마의 포인트를 투자할 지 결정한다. 시간 소진 시 ([D1200B - 팀 대항 게임 설명 화면](#D1200B))으로 이동  2. 전 라운드에서 각 팀이 모은 포인트의 총 합 표시  3. 팀 대항 게임에 투자할 포인트 결정. 이는 팀 대항 미션에서 사용할 수 있는 여러가지 자원으로 변환된다. 수치 조작은 팀원 전원이 가능하다. 버튼을 통해 10단위, 50단위로 조절할 수 있다.  4. 토의용 메신저 기본적인 형식은 (D1300 - 팀 메신저 화면)과 동일 |

1. D1200B - 팀 대항 게임 설명 화면
   1. 팀 대항 게임 시작 전, 팀 대항 게임의 규칙을 한 화면으로 설명하는 화면. 로딩을 위한 아이캐치 화면으로 사용한다.
   2. 화면 경로 : (D1000 - 메인 게임 화면) -> (D1200A - 팀 대항 게임 사전 준비 화면)

-> (D1200B - 팀 대항 게임 설명 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 전체적인 설명 이미지. 전체 화면을 미리 맞춤 제작한 이미지로 표시한다.  2. 로딩이 완료될 시, ([D1200C – 팀 대항 게임 수행 화면](#D1200C))으로 이동 |

1. D1200C - 팀 대항 게임 수행 화면
   1. 실제로 팀 대항 게임을 수행하는 화면
   2. 화면 경로 : (D1000 - 메인 게임 화면) -> (D1200A - 팀 대항 게임 사전 준비 화면)

-> (D1200B - 팀 대항 게임 설명 화면) -> (D1200C - 팀 대항 게임 수행 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 현재 대결 중인 유저들을 나타낸다.  2. 현재 득점 현황을 나타낸다. 더 높은 포인트를 가진 쪽이 득점하고 3판 2선승제이다. |

1. D1200D - 팀 대항 게임 결과 발표 화면
   1. 팀 대항 게임의 결과를 확인하는 화면
   2. 화면 경로 : (D1000 - 메인 게임 화면) -> (D1200A - 팀 대항 게임 사전 준비 화면)

-> (D1200B - 팀 대항 게임 설명 화면) -> (D1200C - 팀 대항 게임 수행 화면)

-> (D1200D - 팀 대항 게임 결과 발표 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 현재 승리하여 1점을 득점한 유저를 나타낸다. 해당 유저가 가진 포인트는 뒷 배경으로 나타난다.  2. 현재 득점 현황이다. 블루 팀이 이겼다는 시나리오이므로 블루 팀의 득점을 나타낸다. |

1. D1300 - 팀 메신저 화면
   1. 팀 단위로 소통할 수 있는 메신저 화면
   2. 화면 경로 : (D1000 - 메인 게임 화면) -> (D1300 - 팀 메신저 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 청팀/홍팀의 현 승리 점수와 라운드 제한시간 표시  2. 팀 메신저  3. \*화면 요소 아님\* 뒤로가기 입력 시, ([D1000 – 메인 게임 화면](#D1000))으로 이동 |

1. E1000 – 마무리 화면
   1. 팀 단위로 소통할 수 있는 메신저 화면
   2. 화면 경로 : (E1000 – 마무리 화면)

|  |  |
| --- | --- |
| 화면 UI | 각 요소 설명 |
|  | 1. 승리/패배 글씨. 자신의 팀의 승리 여부에 따라 변동.  2 팀 죄송 점수 및 누적 포인트  3 각 팀의 인원별 세부 스탯. 수행 미션 수/ 획득 포인트/ 미션당 포인트로 표시한다. 최고 성적 플레이어 순으로 표시하며, MVP를 받은 플레이어 에게는 별모양 표시가 붙는다.  4 게임 마무리 후 메인 로비로 돌아가기. 클릭 시 ([B1000 – 메인화면](#B1000))으로 이동. |

## Ⅲ– 4 상세 설계

* 메인 씬 클래스 다이어그램

텍스트, 실내이(가) 표시된 사진

자동 생성된 설명

* 메인 씬 클래스 상세 설명

|  |  |
| --- | --- |
| Singleton 클래스 | |
| 개요 | GameManager, SceneTransManager등 게임 내에서 하나의 인스턴스만 가져야 하는 클래스를 위한 베이스 클래스 |
| 필드 | instance : 싱글턴 객체를 저장하는 정적 인스턴스. Instance Property를 이용하여 싱글턴 객체를 생성한다. |
| 메소드 | Awake() : 씬 전환에서 싱글턴 객체가 소멸되지 않도록 DontDestroyOnLoad를 호출하는 메소드. |
| 예시 코드 | 텍스트이(가) 표시된 사진  자동 생성된 설명 |

|  |  |
| --- | --- |
| GameManager 클래스 | |
| 개요 | 플레이어를 위한 객체를 생성하는 메소드 |
| 필드 | currentPlayer : 현재 플레이어의 정보를 담는 인스턴스, 프로퍼티 CurrentPlayer를 이용하여 비어있는 currentPlayer에 데이터를 집어넣는다. |
| 메소드 | (-) |
| 예시 코드 | 텍스트이(가) 표시된 사진  자동 생성된 설명 |

|  |  |
| --- | --- |
| SceneTransManager 클래스 | |
| 개요 | 유니티 씬 전환을 위한 클래스 |
| 필드 | (-) |
| 메소드 | LoadScene : 다음 씬을 비동기적으로 로드시킨 뒤, 씬이 완전히 준비되면 이벤트를 발생 시키고 다음 씬으로 넘어간다. objectsToMove 매개변수를 이용하여, 오브젝트를 이동시킬 수 있다.  GoToScene : LoadScene를 코루틴(Unity식 Update())으로 돌린다. |
| 예시 코드 | 텍스트이(가) 표시된 사진  자동 생성된 설명 |

|  |  |
| --- | --- |
| m\_Factory 클래스 | |
| 개요 | 미션 포인트를 생성하는 클래스.  현재는 프로토타입 테스트용으로 임시로 플레이어 주변에 미션이 설치되지만, 이후에는 GPSManager 및 LandmarkManager와 연동하여 원하는 위치에 미션을 설치하는 방식으로 변환시킬 필요 있음. |
| 필드 | availableMission : 게임 내에서 생성 가능한 미션 리스트  waitTime : 새 미션 생성 전 대기시간  minRange : 미션 생성시 생성되는 최소거리  maxRange : 미션 생성시 생성되는 최대 거리  startingMissions : 클래스 시작시 생성되는 미션 수  liveMission : 현재 활성화 된 미션 리스트  Player : 사용자 정보 처리용 변수, 후술할 Player 클래스를 사용함. |
| 메소드 | Start : 클래스 시작 메소드  GenerateMission : waitTime에 지정한 시간마다 InstantiateMission을 돌리는 메소드.  InstantiateMission : 미션을 배치하고, liveMission에다 집어넣기  GenerateRange : mix/maxRange 이용해서 생성 위치 랜덤 돌리기. |
| 예시 코드 | 텍스트이(가) 표시된 사진  자동 생성된 설명  텍스트이(가) 표시된 사진  자동 생성된 설명 |

|  |  |
| --- | --- |
| GPSManager 클래스 | |
| 개요 | 사용자의 스마트폰에서 현재 위치 정보를 수신받는 클래스. |
| 필드 | Latitude : 위도 정보 저장  Longtitude : 경도 정보 저장 |
| 메소드 | Start : 클래스 생성자. 본 클래스가 호출되는 즉시, 위치 파악 서비스로부터 위치 정보를 받아 Latitude/Longtitude에 데이터를 집어넣는다.  OnDestroy : 클래스 소멸자. |
| 예시 코드 | 텍스트이(가) 표시된 사진  자동 생성된 설명 |

|  |  |
| --- | --- |
| LandmarkManager 클래스 | |
| 개요 | GPSManager에서 받아온 위치 정보를 이용해, 미리 설정해 둔 랜드마크 위치와의 거리를 계산하는 클래스  (현재 프로토타입에서의 기능은 X, 나중에 이거로 미션 배치) |
| 필드 | (-) |
| 메소드 | Start : 클래스 생성자. 본 클래스가 호출되는 즉시, 받아온 데이터를 이용하여 아래의 Haver 메소드를 동작시킨다.  Haver : 하버사인 공식 메소드  텍스트이(가) 표시된 사진  자동 생성된 설명 |
| 예시 코드 |  |
| Mission 클래스 | |
| 개요 | 미션 포인트 터치시 AR 미니게임 씬으로 전환되도록 구현한 클래스.. |
| 필드 | (-) |
| 메소드 | OnMouseDown : 미션 포인트 터치시, m\_SceneMAnager 클래스를 통해, AR 미니게임 미션으로 씬 전환을 수행한다. |
| 예시 코드 |  |

|  |  |
| --- | --- |
| m\_SceneManager클래스 | |
| 개요 | Mission 클래스에서 발생한 이벤트를 받아 AR 미션으로 씬 전환을 수행하는 클래스.  임시 클래스. |
| 필드 |  |
| 메소드 | (-) |
| 예시 코드 | (-) |

|  |  |
| --- | --- |
| Player 클래스 | |
| 개요 | 플레이어와 관련된 데이터를 저장하는 클래스 |
| 필드 | Point : 플레이어가 미션을 수행함으로써 얻는 포인트  M\_lst : 미션 수행 리스트 |
| 메소드 | Start : 클래스 생성자. 플레이어 데이터가 저장된 파일에서 플레이어 정보를 불러온다.  addPoint : 플레이어가 가진 포인트 추가  Add\_Mission : 미션 리스트에 미션 추가  Save : 본 플레이어 데이터를 파일화 시킨 뒤, 저장  LoadData : 저장된 파일을 불러오는 메소드. Start와 엮어서 사용. |
| 예시 코드 |  |

|  |  |
| --- | --- |
| LoadingScreenManager 클래스 | |
| 개요 | 로딩 창을 보여주는 씬 매니저  SceneTransManager와 나중에 살짝 통합 필요 |
| 필드 | nextScene : 로딩 완료시 넘어갈 씬  progressBar : 로딩 바 이미지 |
| 메소드 | Start : 클래스 생성자. LoadScene() 호출.  LoadScene : 씬 전환 하이재킹.  LoadScene : 씬 로드작업 진행률에 따라 씬 활성화 여부 설정. 완전 로드시 코루틴 종료. |
| 예시 코드 |  |

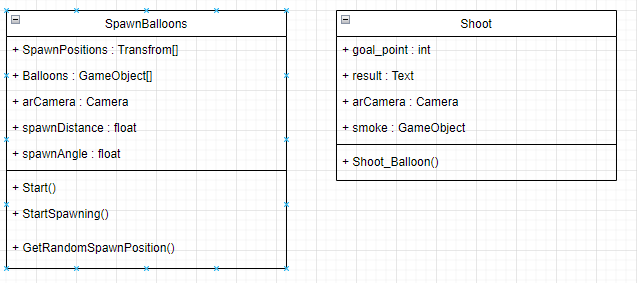
* 농구 미니게임 클래스 다이어그램
* 테이블이(가) 표시된 사진

  자동 생성된 설명
* 농구 미니게임 클래스 상세 설명

|  |  |
| --- | --- |
| BallController 클래스 | |
| 개요 | 농구 미니게임과 관련된 대부분의 요소를 제어하는 클래스 |
| 필드 | resetTime : 샷 미스났을 때의 리셋 타이머  result : 점수 표시기  goal\_point : 골 성공 횟수 카운팅용 변수  rb : rigidbody 인스턴스  isReady : 현재 공이 던질 준비가 된 상황인지 확인하는 변수  startPos : 터치 시작점 |
| 메소드 | Start : 클래스 생성자.  Update : 주기적으로 공 상태를 확인한 뒤, 공이 준비된 상황에서 공의 샷을 구현하는 메소드. 터치 및 드래그한 거리에 따라 공을 발사할 힘을 계산하여 발사한다.  SetBallPosition : 공의 위치를 사용자 카메라의 전방 0.5m, 하단 0.2m 지점에 위치시키는 함수. Update와 엮어서 사용한다.  ResetBall : 공 리셋용 함수  OnCollisionEnter : 공 충돌 감지 함수. 골대 오드벡트의 Collider 영역과 충돌했을 시, 점수를 얻고, 공을 리셋시키는 역할을 맡는다. |
| 예시 코드 |  |

|  |  |
| --- | --- |
| c\_manager 클래스 | |
| 개요 | 농구 골대를 세팅하기 위한 스크립트 |
| 필드 | Indicator : 크로스헤어. 사용자에게 설치 위치를 알려주는 역할을 한다  myCar : 농구 골대 오브젝트.  relocateDistance : 재배치 확인용 하드코딩 거리.  arManager : Raycast 매니저  placedObject : 갖다 박아둔 농구 골대 오브젝트 |
| 메소드 | Start : 클래스 생성자.  Update : 농구 골대가 설치되지 않았을 때, DetectGround 메소드를 주기적으로 호출하여 해당 좌표를 크로스헤어로 표시하고, 사용자가 해당 크로스헤어를 터치하였을 때, 농구 골대 오브젝트를 생성하는 함수  DetectGround : 땅을 탐지한 뒤, 해당 위치 정보를 유니티 월드 좌표로 저장하는 메소드. Raycast 기술을 사용한다.  Raycast란, 화면 상에서 보이지 않는 직선 광선을 쏘아, 오브젝트와 충돌하는지 체크한다. 만약 Plane 타입의 오브젝트, 즉 실제 바닥면을 인식하게 되면 hitInfos 변수에 충돌 정보를 유니티 월드 좌표 형태로 저장한다. |
| 예시 코드 |  |

* 풍선 맞추기 미니게임 클래스 다이어그램



* 풍선 맞추기 미니게임 클래스 상세 설명

|  |  |
| --- | --- |
| SpawnBalloons 클래스 | |
| 개요 | 목표 풍선을 생성하는 오브젝트 |
| 필드 | Balloons : 풍선 오브젝트  arCamera : 시야 오브젝트  spawnDistance : AR 카메라 기준 생성 최대거리  spawnAngle : AR 카메라 기준 생성 최대 각도 |
| 메소드 | Start : 클래스 생성자.  StartSpawning : 풍선을 생성하는 메소드. 3개를 기준으로 무작위 위치에 풍선이 계속해서 생성된다.  GetRandomSpawnPosition : AR 카메라의 위치 및 각도 데이터를 얻은 뒤, 미리 하드코딩된 정해진 수치의 거리 및 각도 사이에서 임의의 거리를 도출해내는 함수. StartSpawning에서 풍선을 생성할 때, 풍선이 등장하는 위치가 된다.  그림입니다. 원본 그림의 이름: sdvsdvsdv.PNG 원본 그림의 크기: 가로 519pixel, 세로 531pixel |
| 예시 코드 |  |

|  |  |
| --- | --- |
| Shoot 클래스 | |
| 개요 | 생성된 풍선을 사격하는 스크립트 |
| 필드 | goal\_point : 점수 계산용 내부 함수  result : 결과창 표시용 함수  arCamera : 시야 오브젝트  smoke : 이펙트 오브젝트 |
| 메소드 | Shoot\_Balloons : 앞서 언급한 Raycast를 이용하여, 만약 광선이 Balloon 태그를 자긴 오브젝트에 닿았을 경우, 해당 풍선 오브젝트를 파괴한 뒤, 해당 위치에 이펙트를 발생시키고 점수를 올리는 메소드 |
| 예시 코드 | 텍스트이(가) 표시된 사진  자동 생성된 설명 |

## Ⅲ– 5 구현 간 문제점 및 해결방안

|  |  |
| --- | --- |
| 문제점 1 : ~~~~ | |
| 문제 내용 | AAAA |
| 발생 원인 | BBBB |
| 해결 방안 | CCCC. |
| 변경점  &  결과 |  |

## Ⅲ– 6 Prototype 구현 및 시험 결과