



국민대학교  
소프트웨어융합대학  
소프트웨어학부


# 캡스톤 디자인 I

## 종합설계 프로젝트

|        |               |
|--------|---------------|
| 프로젝트 명 | <i>Raider</i> |
| 팀 명    | 17            |
| 문서 제목  | 결과보고서         |

|         |             |
|---------|-------------|
| Version | 2.0         |
| Date    | 2021-MAY-26 |

|    |           |
|----|-----------|
| 팀원 | 안 희운 (조장) |
|    | 권 순민      |
|    | 김 승중      |
|    | 김 주연      |
|    | 이 가희      |

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |


#### CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 캡스톤 디자인 I 수강 학생 중 프로젝트 "Raider"를 수행하는 팀 "17"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "17"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

## 문서 정보 / 수정 내역


|                 |                          |
|-----------------|--------------------------|
| <b>Filename</b> | 17 조 수행결과보고서-Raider.docx |
| <b>원안작성자</b>    | 안희운, 권순민, 김승중, 김주연, 이가희  |
| <b>수정작성자</b>    | 안희운, 권순민, 김승중, 김주연, 이가희  |

| 수정날짜       | 대표수정자 | Revision | 추가/수정 항목 | 내 용               |
|------------|-------|----------|----------|-------------------|
| 2021-05-24 | 권순민   | 1.0      | 최초 작성    | 개요 작성, 전체적인 내용 작성 |
| 2021-05-25 | 김승중   | 1.1      | 내용 수정    | 연구 개발 내용 추가       |
| 2021-05-25 | 안희운   | 1.2      | 내용 수정    | 참고문헌, 부록 추가       |
| 2021-05-25 | 안희운   | 1.3      | 내용 수정    | 시스템 구조 및 설계도 추가   |
| 2021-05-26 | 이가희   | 1.4      | 내용 수정    | 연구 개발 내용 수정       |
| 2021-05-26 | 김주연   | 1.5      | 내용 수정    | 연구 개발 내용 수정       |
| 2021-05-26 | 팀 전체  | 2.0      | 최종 점검    |                   |

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

## 목 차

|       |                     |    |
|-------|---------------------|----|
| 1     | 개요                  | 4  |
| 1.1   | 프로젝트 개요             | 4  |
| 1.2   | 추진 배경 및 필요성         | 4  |
| 2     | 개발 내용 및 결과물         | 5  |
| 2.1   | 목표                  | 5  |
| 2.2   | 연구/개발 내용 및 결과물      | 7  |
| 2.2.1 | 연구/개발 내용            | 7  |
| 2.2.2 | 시스템 기능 요구사항         | 25 |
| 2.2.3 | 시스템 비기능(품질) 요구사항    | 26 |
| 2.2.4 | 시스템 구조 및 설계도        | 27 |
| 2.2.5 | 활용/개발된 기술           | 33 |
| 2.2.6 | 현실적 제한 요소 및 그 해결 방안 | 33 |
| 2.2.7 | 결과물 목록              | 33 |
| 2.3   | 기대효과 및 활용방안         | 35 |
|       | 있다.                 | 35 |
| 3     | 자기평가                | 35 |
| 4     | 참고 문헌               | 36 |
| 5     | 부록                  | 37 |
| 5.1   | 사용자 매뉴얼             | 37 |
| 5.2   | 서버 운용 매뉴얼           | 37 |
| 5.3   | 테스트 케이스             | 40 |

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

# 1 개요

## 1.1 프로젝트 개요

본 프로젝트는 컴퓨터의 윈도우 운영체제를 기반으로 3D 멀티플레이 공포 게임을 개발하는 것을 목표로 두고 있다. 게임 소프트웨어 과목을 통해 배운 유니티 엔진을 사용하여 게임을 개발하며 개발 파트는 크게 게임에 사용될 서버(마스터 서버, 룸 서버), 플레이어(조작, 아이템 사용 등), 적(플레이어 감지 센서, 추적, 공격 등), 맵과 미니게임, 사용자 인터페이스(UI)로 역할을 나눠 진행했다. 먼저 개개인이 파트별 개발을 진행하고 이후 한 부분 씩 네트워크에 연동시켜가며 최종적으로 하나의 프로그램이 되도록 하였다.

본 프로젝트는 최대 4 인의 사용자가 함께 즐길 수 있는 게임이다. 게임 내 사용자를 감지하면 쫓아와 방해하는 순찰로봇을 피해가며 미니게임 형태의 미션을 모두 해결하는 방식의 게임으로, 최종적으로는 정해진 공간으로부터의 탈출을 목표로 하고 있다. 게임의 조작법이 어렵지 않아 남녀노소 누구나 빠르게 게임에 적응할 수 있으며, 멀티 플레이를 지원함으로써 타인과의 협동심을 기를 수도 있다. 또한 1 회 플레이 시간이 길지 않기 때문에, 일반적인 PC 게임보다 가벼운 마음으로 즐기기 좋다.

## 1.2 추진 배경 및 필요성

게임 시장에 출시된 많은 공포게임들은 대개 잔인하고 폭력적인 요소들이 많이 배치되어 있다. 멀티플레이 공포게임의 대표작으로 볼 수 있는 데드 바이 데이라이트를 예시로 들면 플레이어를 쫓아다니며 '살인마'가 실제로 플레이어를 잔인하게 죽이는 장면이 자세하게 연출된다. 이렇게 '살인마'라는 존재는 공포 게임의 메인 요소인 긴장감과 공포감을 유발하는데 큰 역할을 하지만, 동시에 일부 사용자에게는 거부감을 줄 만큼의 과한 폭력성과 잔인성을 띠다고 생각했다. 또한 게임에 대한 정보가 부족한 사람에겐 첫인상 만으로 게임 전체에 대한 부정적 인식이 못박힐 수도 있다. 아동, 청소년들이 이러한 게임을 접했을 경우에는 부정적인 영향을 받을 수도 있다는 의견은 과거부터 많이 거론되는 주제다.

여기서 폭력적인 요소와 잔인한 요소가 없어도 공포게임 특유의 공포감과 긴장감을

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

조성할 수 있는 방법에 대한 아이디어가 거론되었다. 피가 튀거나 누군가가 죽지 않아도 충분한 스틸감과 재미를 느낄 수 있다면, 평소 잔인한 소재에 거부감을 느껴 공포게임을 즐기지 못했던 사용자도 비교적 쉽게 게임을 접할 수 있을 것이고, 아동, 청소년들에게 가해질 수도 있는 부정적인 영향을 최소화할 수 있을 것이라 판단했다. 나아가 이와 같은 게임들이 계속 개발되고 게임 시장을 차지하는 비율이 늘어난다면, 사회 전반에 걸친 게임에 대한 부정적인 인식을 개선하는 데에도 큰 도움이 될 수 있을 것이라 생각했다. 이러한 흐름의 첫 걸음이 본 프로젝트가 되기를 바라는 마음에서 게임의 컨셉을 계획하고 진행하게 되었다.

## 2 개발 내용 및 결과물


### 2.1 목표

아동과 청소년들에게 게임의 폭력성과 잔인함으로 인해 발생하는 부정적인 영향을 최소화하고, 다양한 사용자 계층이 편하게 즐길 수 있는 3D 공포게임을 개발한다.

장르의 특성상 불가피하게 존재하는 폭력적인 요소를 최소화하기 위해 현실성이 떨어지는 플레이어와 적을 기획, 사람 형태의 캐릭터가 아닌 로봇을 사용자와 적으로 설정한다. 기존 공포게임의 대표적인 연출인 피, 절단 같은 연출을 모두 제거하며 피격에는 애니메이션과 오디오 소스만을 활용해 잔인하게 느껴질 수 있는 요소를 배제하도록 한다.

게임 서버의 구조는 보편적인 PC 게임과 유사한 구조를 가지도록 한다. 플레이어는 멀티플레이를 위한 방을 직접 만들 수도, 이미 타인에 의해 만들어진 방에 참여할 수도 있도록 하여 자유도를 높인다. 하나의 방에는 최대 4 인이 입장할 수 있도록 서버를 구성한다.

게임 내에서 사용하는 ‘플레이어 캐릭터’는 게임을 즐기는 사용자가 직접 조작하도록 한다. 보편적인 게임처럼 키보드와 마우스를 이용해 조작하도록 하여 게임의 접근 난이도를 낮추고, 시야조절은 마우스, 플레이어 캐릭터의 실질적인 움직임은 키보드로 진행해 최대한 사용자들이 원하는 방향으로 편하게 캐릭터를 조종할 수 있도록 고안한다. 협동 게임의 특색을 살리기 위해, 플레이어들은 다른 플레이어가 벽 너머에 있어도 위치를 확인할 수 있으며, 자연스러운 시점 구현으로 몰입도를 높이기 위해 카메라가 만들어진 벽을 관통해 텅 빈 게임 외적 공간을 보게 되는 등의 오류를 최소화한다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

게임의 주된 재미요소는 간단하고 다양한 미션, 게임 진행을 방해하는 적의 출현으로 한다. 다양한 미니게임을 구현하고 이를 통해 기존 공포게임과 차별점을 주는 것을 목표로 한다. 단, 한정된 인원으로 개발과 기획을 같이 진행해야 되기 때문에 시간적, 기술적으로 많은 미션을 기획하는 것은 무리가 있다고 판단, 여러 조건을 고려해 세개의 미션만을 기획하고 개발한다.

적은 게임 AI 를 구현해 스스로 움직이게 하는 것을 목표로 한다. 적은 주어진 맵을 순찰하며 플레이어 캐릭터들을 탐지하고, 발견할 경우 플레이어를 추적하도록 한다. 플레이어 캐릭터를 탐지하기 위해 시각, 청각 센서를 구현한다. 플레이어와 일정 거리 이하로 가까워질 경우 플레이어 캐릭터를 공격하여, 게임의 클리어를 방해하는 장애물이 되도록 한다. 이를 통해 공포 게임다운 긴장감을 유도한다.

사용자 인터페이스(UI) 요소로는 게임을 실행하자마자 나오는 스타트 씬, 게임의 콘텐츠가 로딩되는 동안 표시되는 로딩 씬, 게임을 클리어 한 후 최종 결과를 표시하고 영상적 연출을 하는 엔딩 씬 개발을 목표로 한다. 본격적으로 게임이 실행되고 난 후의 인게임 내에서는 게임 진행도, 함께 멀티 플레이를 즐기는 타 사용자들의 상태 표시창, 게임의 사운드를 조절하거나 게임 중도 종료 등의 행동이 가능한 메뉴창을 구현한다.

공포감 조성을 위해 어두운 조명과 분위기에 맞는 음악 등 시각, 청각적 요소를 이용한다. 플레이어의 시야를 제한하고 어두운 조명을 이용해 보이지 않는 곳에서 오는 공포감을 조성하며 이를 돕기 위해 적절한 배경음악과 효과음을 사용하도록 한다. 새로운 음악을 만드는 일은 현실적으로 불가능하기 때문에 시각적인 요소에 더 집중할 수 있도록 한다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

## 2.2 연구/개발 내용 및 결과물

### 2.2.1 연구/개발 내용

#### 1. 방 생성, 방 접속

마스터 서버는 Master server Toolkit API 를 이용하여 개발을 진행하였다. 게임을 시작하면 Start Scene 에서 마스터서버에 접속한다.

플레이어가 Create Room Button 을 클릭할경우 Matchmaker 가 방을 만들겠다는 요청을 실행하면 서버의 Spawner 모듈이 그에 반응해 룸서버 인스턴스를 새로 실행하도록 한다. 룸서버가 성공적으로 작동될경우 그 룸서버를 마스터 서버가 RoomModule 에 등록시킨다. 플레이어는 그 정보를 바탕으로 접속한다.

플레이어가 Find Room Button 을 클릭할경우 플레이어가 방 이름을 바탕으로 접속하는 과정이 이루어진다. 플레이어 클라이언트 상에 있는 Matchmaker 가 RoomModule 로부터 룸서버 정보들을 가져올 것을 요청한다. MatchMaker 가 받으면 플레이어는 그 룸서버에 접속하게 된다.

#### 2. 대기실

방으로 접속한 뒤 방 안에서 이루어지는 동기화 작업은 Mirror Network API 를 통해 진행되었다.

플레이어가 대기실 씬으로 전환되면 플레이어의 접속 상태와 이름 등의 정보가 담긴 RoomPlayer 오브젝트가 접속한 인원수에 맞게 생성되어 방 안의 모든 플레이어들에게 보여진다. 플레이어가 준비 버튼을 누를 경우 다른 플레이어들도 그 플레이어가 준비되었는지를 확인할 수 있다. 서버는 플레이어들의 준비된 상태를 확인하며 모든 플레이어가 준비 되어 있을 시 방장에게만 Start Button 을 활성화 시킨다. 방에서 방장이 나갈경우 서버에서 방인원 중에 새로운 방장을 선정할 수 있도록 한다. 방장이 Start Button 을 누를 경우 서버는 인게임 씬으로 전환된다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

### 3.적

#### 3.1 Enemy.cs

게임 AI 를 구현하는데 사용하는 대표적인 방법 중 하나인 FSM 을 이용해 개발하였다. FSM(Finite State Machine)을 구현하기 위해 5 가지 상태, 기본(Idle), 순찰(Patrol), 추적(Chase), 공격(Attack), 기절(Dizzy)을 기획하였고 유지보수, 확장성에 용이하도록 상태 패턴(state pattern)을 적용해 개발하였다.

abstract class State 를 만들고 State 에 virtual 함수를 만들어 상속받은 각 상태에서 오버라이딩해 사용할 수 있게 하고 각 상태에서는 MonoBehaviour 대신 State 를 상속 받도록 하였다. State 클래스의 virtual 함수로는 Enter, LogicUpdate, Exit 등이 있으며 Enter 와 Exit 은 상태로 들어갈때와 나갈때 실행되어야 하는 코드들을 정의하고 LogicUpdate 는 게임이 진행되는 동안 계속해서 실행되어야 하는 코드들을 정의한다. 생성자에는 Enemy 클래스를 매개변수로 받게 만들어 Enemy 에 접근하기 쉽게 하였다.

또 FSM 의 트랜지션과 처음 FSM 이 구동될 때 초기화를 진행해주는 StateMachine 클래스를 만들었다. ChangeState 를 만들어 트랜지션 기능을 구현하였다. ChangeState 에서는 현재 상태의 Exit 을 호출하고, 현재 상태를 저장하고, 현재 상태를 다음 상태로 바꾸고, 다음 상태의 Enter 를 호출하게 하였다.

```
private void Awake()
{
    enemyStateMachine = new StateMachine();
    idle = new IdleState(this);
    patrol = new PatrolState(this);
    attack = new AttackState(this);
    dizzy = new DizzyState(this);
    chase = new ChaseState(this);

    enemyStateMachine.Initialize(idle);
}

private void FixedUpdate()
{
    if (enemyNet != null && !NetworkServer.active)
    {
        return;
    }
    enemyStateMachine.currentState.LogicUpdate();
}
```

<Enemy.cs 일부>



|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

실제 적에 적용되는 클래스인 Enemy 클래스를 만들어 Awake()와 FixedUpdate()로 FSM 을 돌리게 하고 클래스에 적과 관련된 모든 함수들을 구현하였다. StateMachine 오브젝트와, State 오브젝트들을 생성하고 관리하며 각 State 오브젝트의 생성자 매개변수로 Enemy 클래스를 받도록 했다. 단 모든 변수를 private 으로 선언해 캡슐화를 진행했고 함수를 통해서만 접근, 수정이 가능하게 하였다. Enemy 클래스는 Awake 함수에서 필요한 오브젝트들 (StateMachine, IdleState, PatrolState 등)을 생성하고 StateMachine 의 FSM 초기화 함수를 실행해 FSM 을 실행시킨다.

멀티플레이 게임이기 때문에 하드웨어 사양에 따라 다른 게임 화면이 실행되어서는 안 된다. 따라서 FixedUpdate()에서 각 상태들의 LogicUpdate()를 실행시켜 하드웨어로 발생할 수 있는 차이를 줄였다.

### 3.2 시각 센서

유니티에서 제공하는 Physics.OverlapSphereNonAlloc 함수를 이용했다. 이 함수는 Physics.OverlapSphere 와 유사하지만 메모리를 할당하지 않기 때문에 게임의 렉을 줄일 수 있어 선택하였다. 이 함수는 한 위치를 중심으로 하는 구체 만큼의 범위에 들어오는 게임 오브젝트들을 탐지하는데 특정 레이어를 지닌 오브젝트만을 탐지할 수 있다. 탐지된 게임 오브젝트의 수를 리턴 하면서 매개변수로 입력한 Collider 배열에 탐지된 오브젝트를 넣어준다.

Collider 배열에 할당된 collider 들은 각 플레이어의 collider 기 때문에 collider 의 transform 값은 플레이어 오브젝트의 transform 값이 될 수 있다. 이 transform 정보를 사용하기 위해 for 문을 돌며 각 collider 의 transform 정보를 따로 Transform 변수에 할당하면 이 변수는 플레이어의 transform 정보를 가지게 된다. 이 Transform 변수의 position 값과 적의 position 값을 빼고 normalize 해서 적과 플레이어 사이의 방향벡터를 만든다. 그 다음 적이 바라보고 있는 방향을 구하고 위에서 구한 방향벡터와 적이 바라보고 있는 방향사이의 각을 구해 정해 놓은 시야 각 보다 작다면 시야에 들어온 것으로 판단했다.

실제 시각처럼 구현하기 위해 벽이나 장애물 뒤에 있는 플레이어는 타겟으로 설정되어서는 안 된다. 때문에 Physics.Raycast 함수를 이용해 플레이어와 적 사이에 직선을 긋고 직선에 플레이어의 레이어를 제외한 다른 레이어가 있으면 적이 쫓아야 하는 타겟이 되지 않게 하였다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

### 3.3 청각 센서

플레이어의 애니메이션에 이벤트를 넣어 기능을 개발했다. 유니티에서는 애니메이션의 특정 프레임에 함수가 실행되게 할 수 있는데 이를 이용해 구현했다. 센서 범위 내에 있는지 확인하는 boolean 변수를 만들고 이벤트가 발생했을 때 적과 플레이어의 거리가 범위 내에 있으면 boolean 변수를 true 로 바꿔 타겟이 되도록 했다. 이때 Enemy 클래스에 접근해 청각 센서를 끄고 걸 수 있는 메소드에 접근해 센서를 켜서 Enemy 클래스에서 시각으로 탐지했을 때와 청각으로 탐지했을 때를 구분해 진행할 수 있게 했다.

### 3.4 타겟

타겟들은 List 로 관리했으며 공격과 기절 상태에 들어서면 타겟들을 초기화해 무한정 늘어나는 현상을 방지했다. 하나의 타겟만을 추적해야 하기 때문에 List 를 순회하면서 플레이어와의 거리를 측정했고 가장 가까운 플레이어의 인덱스를 기억해 그 플레이어만 최종적으로 타겟이 되도록 했다.

### 3.5 기본 상태(IdleState.cs)

적의 기본 상태를 구현했다. 5 초간의 딜레이를 줘서 로봇이 부팅되는 듯한 효과를 냈다. 타이머를 이용해 딜레이를 부여했는데 하드웨어에 따라 다른 값을 가지면 안 되기 때문에 Time.deltaTime 을 이용해 시간을 측정했다. 2 초 뒤에는 이전 상태가 공격상태였는지 확인하고 맞다면 적의 collider 를 통과가 안 되도록 만들었다. 5 초가 지나면 이전 상태를 체크해 공격이나 기절 상태였으면 NavMeshAgent 를 원래상태로 돌리고 순찰상태로 들어가게 했다.

### 3.6 순찰 상태(PatrolState.cs)

적이 맵을 순찰하면서 플레이어들을 탐지하는 상태이다. 순찰 상태에 걷는 애니메이션을 실행시키고 순찰 지점을 설정했다. 순찰 지점은 미리 배열로 정해 놔다. 그 뒤에 유니티에서 제공하는 UnityEngine.Random 을 이용해 임의의 index 값을 정하고 배열에 접근해 순찰지점을 선택하게 했다. 적을 순찰지점으로 이동시키기 위해 적을 NavMeshAgent 로 설정했고 NavMeshAgent 에 있는 SetDestination 을 이용했다. 적은 이동하고 이동하는 동안에는 센서가 작동되도록 했으며 순찰 지점에 다르면 다시 순찰 상태로 들어서 다른 순찰 지점으로 이동하게 했다. 만일 순찰 중에 플레이어를 탐지하면 추적 상태로 들어가게 했다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

### 3.7 추적 상태(ChaseState.cs)

순찰 중에 탐지된 타겟을 추적하는 상태이다. 플레이어에게 탐지되었다는 것을 알리기 위해 사이렌을 울리고 순찰 상태와 마찬가지로 SetDestination 을 이용하지만 순찰 지점이 아닌 탐지된 타겟을 대상으로 접근하게 했다. 추적 중에는 지속적으로 속도가 증가하며 애니메이션에 Blend Tree 를 적용해 점차 뛰는 애니메이션으로 바뀌게 했다. 또 플레이어와의 거리가 일정 수준 가까워지면 공격 상태로 들어가게 했다.

### 3.8 공격 상태(AttackState.cs)


공격 상태에 들어서면 추적 상태 때 울리던 사이렌을 멈추고 추적 상태와 순찰 상태에서 사용하던 NavMeshAgent 를 멈춰, 제자리에서 공격 애니메이션이 실행되도록 했다. 적의 공격은 팔로 공격하는 애니메이션일 이용했기 때문에 팔에 부딪힐 수 있도록 애니메이션에 이벤트를 부여했다. 공격 중에만 팔에 부여된 collider 를 켜서 충돌 판정이 일어날 수 있게 했고 공격이 끝나면 팔의 collider 를 끄게 했다. 공격 후에 플레이어가 도망가기 편한 환경을 만들기 위해 적의 몸에 부여된 collider 를 trigger 로 설정해 통과가 가능하게 만들었다. 공격 애니메이션이 끝나면 기본 상태로 들어가게 했으며 공격 상태 때 trigger 로 설정된 collider 는 기본 상태에 들어가면 trigger 옵션을 해제해 다시 통과가 되지 않는 상태로 만들었다.

### 3.9 기절 상태(DizzyState.cs)

기절 상태는 적이 플레이어에게 공격을 당했을 때 들어오게 되는 상태이다. 기절 상태에 들어오면 사이렌이 켜져 있는지 확인해 사이렌을 끄고 기절 상태에 들어서기 전에 가지고 있던 모든 타겟에 대한 정보, 센서 상태, 애니메이션 파라미터들을 초기화한다. 그 후에 기절한 것과 같은 애니메이션을 실행시켰으며 애니메이션이 끝나면 기본 상태로 돌아가게 했다.

### 3.9 애니메이션, 효과음

적의 상태에 맞게 애니메이션을 적용시켰다. 효과음이 필요한 애니메이션을 특정했고 그 애니메이션에는 이벤트를 부여해 자연스러운 효과음이 흘러나오도록 했다. 걷는 애니메이션과 뛰는 애니메이션은 Blend Tree 를 이용해 구현했으며 blend tree 는 두 애니메이션을 자연스럽게 연결해주는 기능이다. blend tree 를 이용하게 되면 임계점을 정해 임계점에 가까워지면 점차 다른 애니메이션으로 변하게 할 수 있다. 이를 이용해 걷는 애니메이션에서 뛰는 애니메이션으로 자연스러운 트랜지션을 구현했다.

|   |                         |             |             |
|---|-------------------------|-------------|-------------|
| <br><b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|   | <b>프로젝트 명</b>           | Raider      |             |
|   | <b>팀 명</b>              | Team 17     |             |
|   | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

## 4. 플레이어

### 4.1 플레이어 움직임

플레이어는 키보드와 마우스를 이용하여서 움직이게 만들었다. 물리엔진 시뮬레이션 계산 주기로 기본값은 0.02 초로 일정한 FixedUpdate 함수로 키보드와 마우스 입력을 받았다. 마우스의 움직임에 따라 플레이어 회전을 결정하는 함수 LookAround 를 만들었다. 마우스 움직이는 수치를 Input.GetAxis 함수에 Mouse X(마우스 좌우 수치) , Mouse Y(마우스 위아래 수치)를 매개 변수로 넣어서 가져올 수 있었다. 마우스 방향과 바라보는 방향을 일치시키기 위해서 기존 카메라 회전을 오일러 각으로 변환 시킨 값에 입력 받은 마우스 값을 빼준다. 그 뒤 각도를 제한 시켜준다. 그러면 마우스 회전에 따라서 플레이어가 회전한다. 키보드 입력에서는 c 를 누르면 앉고 스페이스 누르면 점프하게 만들었다. 왼쪽 쉬프트 값을 누르면 0.7 를 곱하여 기존 속도보다 느리게 만들었다. 해당 키보드 값을 받아 움직이게 만들었다.

### 4.2 플레이어 카메라

카메라는 3 인칭으로 플레이어가 보이게 만들었고 카메라가 벽에 닿으면 플레이어와 가까워지게 만들어 카메라 벽 통과 방지를 하였다. magnitude 는 벡터 길이를 나타내는 단일 차원 값인 Float 를 반환하고 normalized 는 약간의 반대 연산으로 벡터 방향을 반환하여 결과 벡터의 크기가 1 을 보장하는 것이다. 이 둘을 받아서 카메라의 충돌이 생기면 Lerp 로 부드럽게 카메라가 이동하게 구현했다. 또한, 카메라가 플레이어를 따라다니게 하려고 카메라와 위치를 조절해줄 Offset x,y,z 를 만들어 줘서 플레이어가 해당 Offset 만들 떨어지게 만들었다. 따라서 거리를 줘 3 인칭 시점으로 플레이어를 볼 수 있게 구현했다.

### 4.3 플레이어 체력

체력의 값은 최대 2 이다. 다칠 때, 죽을 때, 체력 상승할 때, 총 이 세가지 상태로 체력을 제어하게 된다. 말 그대로 다른 스크립트가 적용하기 쉽게 체력을 관리하는

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

스크립트를 따로 빼냈다. 플레이어가 OnTriggerEnter 로 태그를 받아와 적의 공격이면 공격받는 애니메이션을 취하며 Hit 을 불러온다. Hit(다칠 때)는 현재 플레이어 체력을 1 감소시키고 감소시킨 뒤 체력이 0 이면 Die 를 불러온다. Die(죽을 때)는 플레이어 현재 상태를 죽었다고 표현해준다. Heal 은 플레이어 체력이 1 일때만 발생한다. 체력을 1 더해주고 체력은 다시 원상태로 바뀌게 만들어준다.

#### 4.4 플레이어 인벤토리


플레이어가 주변에 아이템이 있다고 감지를 하고 E 키를 누르면 아이템을 습득할 수 있다. 습득하면 인벤토리로 넘어가게 되고 아이템을 사용할 때 플레이어와 상호작용하게 만들었다. 전기충격기 아이템을 들면 해당 관련 애니메이션과 root 위치에 설정해 놓은 아이템을 보이게 만들어 전기충격기를 잡고 있는 것처럼 보인다. Avatar 를 통해서 상체만 적용해 밑에는 걷고 뛰는 애니메이션을 그대로 실행하게 했다. 또한, 체력 물약도 들면 손에 잡고 있는 것처럼 보이게 해냈다.

#### 4.5 플레이어 디자인 & 셰이더

유니티 에셋 스토어에서 캐릭터 디자인을 골라서 구매하여 사용하였다.

플레이어들이 서로의 위치를 확인하기 위하여 셰이더를 수정하여 사용하였다. 기존 스탠다드 셰이더를 코드로 받아와 Pass 블록 코드를 만들었다. 해당 블록 코드에서는 ZTest 를 사용하며 ZTest 를 Always 로 설정해 해당 오브젝트가 우선시되어 그려지게 했다. 그 후에 해당 블록코드에 색을 설정해주면 벽 뒤에 있는 플레이어들을 설정한 색으로 볼 수 있게 된다. 이 상태에서는 플레이어가 아이템을 들고 있으면 아이템 너머로 플레이어가 보이게 된다. 그래서 셰이더를 아이템에도 적용 시켜서 Render Queue 의 값을 바꿔 해결했다. 순서를 플레이어 앞으로 옮긴다고 생각하면 된다.

하지만 최종적으로 프로젝트의 특성상 아이템 너머로도 플레이어가 보이는 게 좋다고 판단해 아이템에는 이 셰이더를 적용 시키지 않았다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

## 5. 맵 및 아이템과 미션

### 5.1 맵

맵은 유니티 에셋스토어에서 제공하는 에셋들 중에서 가장 기획의도와 적합하다고 판단되는 유료 에셋을 이용하여 맵을 구성하였다.

### 5.2 아이템박스

아이템 박스는 랜덤 함수를 이용해 간단한 사칙연산 문제를 생성하며 플레이어가 Raycast 를 이용해 아이템 박스와 상호 작용하여 문제를 해결하면 박스를 열고 아이템을 습득 가능하도록 만들었다.

### 5.3 아이템

#### 5.3.1 체력회복제

플레이어의 체력이 적에 의해 줄어들었을 때 사용하면 플레이어의 체력 값을 1 올리는 방식으로 회복하는 아이템으로 구현하였다.

#### 5.3.2 전기충격기

플레이어가 사용하면 오브젝트를 일직선으로 발사해 이 오브젝트가 적과 충돌하면 적의 상태를 기절로 바꾸며 발사한 오브젝트는 사라지도록 구현하였다.

### 5.4 미션

미션은 총 3 가지로 구성되어 있으며 모든 미션은 전부 랜덤 함수와 셔플 함수를 이용하여 생성된 미션코드들을 이용해 제작되었다.

첫번째 미션의 경우 9 개의 버튼이 주어지며 각 버튼은 1~9 의 값을 가지고 있다. 미션에 제공되는 코드는 3~5 자리로 총 3 개의 스테이지로 구현했다. 1~9 의 범위에서 랜덤하게 3~5 자리의 중복이 없는 값을 뽑아 각 스테이지의 코드로 지정했다. 미션이 시작되었을 때 코루틴을 이용해 각 스테이지의 코드에 해당하는 버튼들의 색을 순서대로 변환시켜 플레이어에게 색이 변한 순서대로 버튼을 누르면 미션을 해결할 수 있도록 설계하였다.

두번째 미션은 정답을 보여주는 화면과 4 개의 작은 화면 그리고 작은 화면들의 위아래에 존재하는 8 개의 버튼으로 구성했다. 미션 코드는 0~9 의 숫자를 이용한 4 자리의 수 2 개로 구성하여 각각 정답 화면과 4 개의 작은 화면에 띄워준다. 플레이어는 작은 화면의 위아래에 있는 버튼을 이용하여 화면의 숫자를 1 만큼 더 높은 수로

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

바꾸거나 작은 수로 바꿀 수 있으며 작은 화면에 주어진 4 개의 숫자가 정답화면의 숫자와 일치하면 해결되도록 설계하였다.

세번째 미션은 2 개의 화면과 4 개의 버튼으로 구성했다. 각 화면에는 정답으로 설정한 미션코드와 플레이어에게 주어지는 미션코드가 시간 형식으로 표시한다. 4 개의 버튼은 카세트 플레이어와 비슷하게 뒤로 빨리감기, 일시정지, 재생, 빨리감기로 구성되어 일시정지 버튼으로 정답과 일치하는지 판별하고 나머지 3 개의 버튼은 코루틴을 이용해 시간으로 표시된 코드 값이 자연스럽게 변하는 것처럼 보이도록 설계하였다.

## 6. UI


유니티 자체 제공 기능을 사용하여 각 씬과 버튼을 구현, 주로 UI 툴을 사용하였다. 버튼과 이미지 등의 디자인은 Unity 공식 홈페이지를 통해 거래가 이루어지는 유료 asset 을 사용하였으나, 세부 배치는 직접 기획하였다.

### 6.1. Start scene

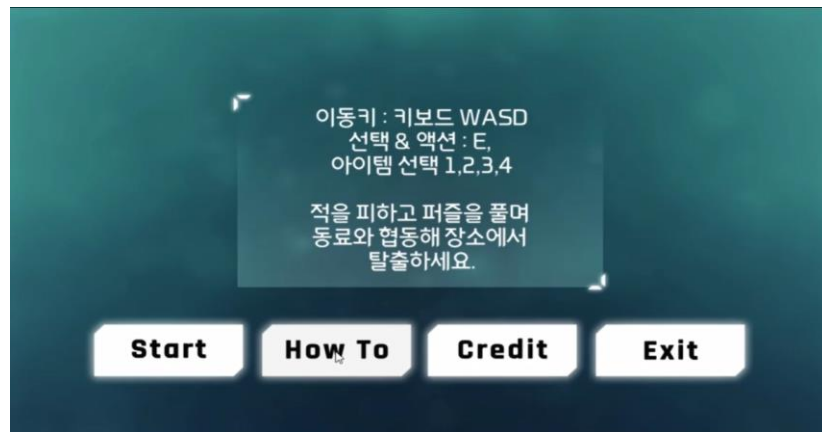


<Start Scene>

게임이 실행되자마자 나타나는 화면. 게임의 대표 로고와 함께 Start, How To, Credit, Exit 버튼이 존재한다. 버튼의 구현에는 Unity 자체 UI Button 기능을 사용하였다. 어떤 버튼을 언제 누르냐에 따라 화면에 보이는 객체의 active 상태가 반전된다. 각 버튼을 누르면 버튼에 해당하는 내용의 창이 떠오르고, 해당하지 않는 창이 active 상태였을 경우 inactive 상태로 되돌린다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

#### 6.1.1 How to



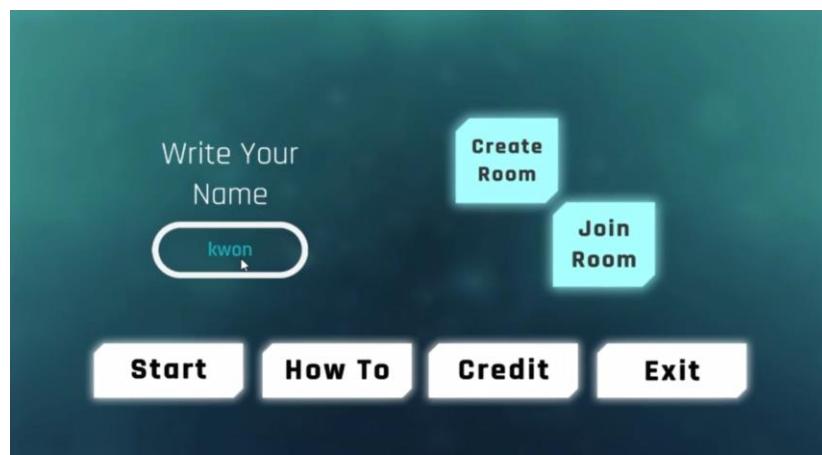
<How To 버튼 누른 후>

버튼을 누르면 게임의 개요와 조작법이 화면에 표시된다. 화살표 키를 눌러 다음 페이지로 넘길 수 있고, 버튼을 다시 한번 누를 시 설명창이 꺼진다.

#### 6.1.2 Exit

버튼을 누르면 게임이 완전히 종료되고 창이 꺼진다.

#### 6.1.3 Start



<Start 버튼 누른 후>

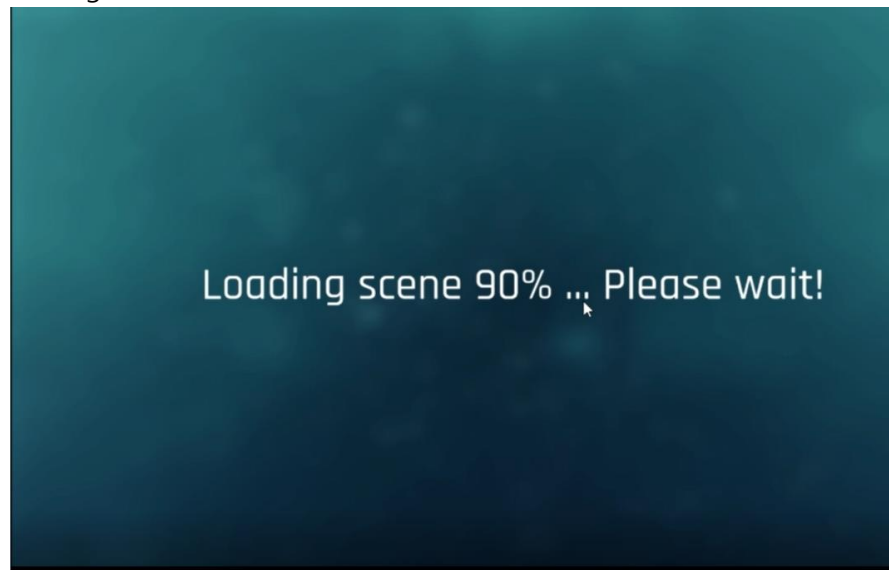
Start 를 누르면 게임을 시작하기 전 게임 내에서 사용할 닉네임과 직접 방을 생성할지, 아니면 이미 만들어진 방에 참여할지를 고를 수 있다. 닉네임을 입력하고, 호스트로 게임에 참여할 땐 Create Room 버튼을 누르는 것으로 대기실을 생성할 수 있고, 이미 만들어진 방에 들어갈 경우엔 영문 대문자 4 자로 이루어진 방의 코드를 입력하고 난 후에 Join 버튼을 눌러 대기실에 입장할 수 있다.



|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

Join 버튼을 누른 후 장면 전환에는 유니티 SceneManager 의 LoadScene 과 LoadSceneAsync 를 사용하였다. 이 기능들은 게임과 서버를 연결하고 콘텐츠를 다운 받는 동안 사용자들의 화면이 멈춰 어색함을 느끼지 않도록 그 시간 동안 정해진 씬을 서버 비동기화 상태로 호출해준다. 본 프로젝트에서는 Loading Scene 을 호출하도록 하였다. 이를 통해 사용자들은 게임에 문제가 생긴 것이 아닌, 게임의 콘텐츠를 다운받느라 딜레이 시간이 걸린다는 것을 시각적으로 알 수 있다.

## 6.2. Loading scene

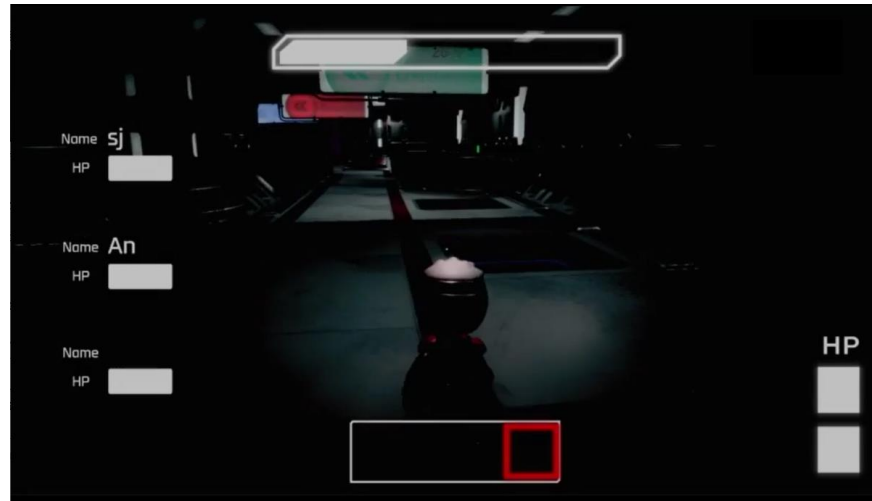


<Loading Scene>

마스터 서버와 플레이어가 소통이 되는 과정에서 Delegate void 문을 이용하여 로딩 메시지를 호출함. 또한 인게임 시작 중 플레이어가 다른 플레이어를 기다리는 경우 직접 Loading 오브젝트를 생성하여 로딩바와 관련된 애니메이션을 적용했다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

### 6.3. Ingame scene



<InGame Scene>

모든 사용자들이 접속하고, 게임이 실행되고 난 이후에 화면에 띄워지는 인터페이스에는 메뉴창, 미션 진행도 표시 바, 팀원들의 상태표시창, 본인의 체력 표시창, 인벤토리 등이 있다.

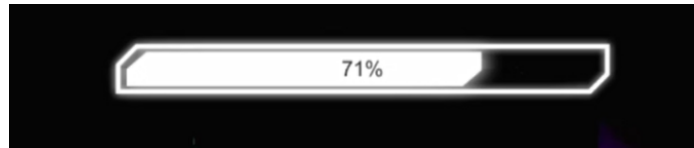
#### 6.3.1 메뉴창

게임 진행 중 ESC 키를 누르면 메뉴창 팝업이 형성된다. 메뉴창이 띄워져 있는 중에도 게임은 멈추지 않고 진행된다. 메뉴 창에서 할 수 있는 일은 How To 를 통해 게임 조작법 확인하기, 인게임 내 사운드 조절, 게임 종료 등이 있다. 메뉴창에서 각각의 메뉴 이동은 버튼에 적혀 있는 해당 키보드를 누르는 것으로 선택할 수 있고, 다시 ESC 키를 누르면 메뉴창이 꺼진다.

How to 창에서는 오른쪽 왼쪽 화살표를 누르는 것으로 페이지를 넘길 수 있다. 사운드 바는 배경음악, 적과 플레이어 효과음, 그 외 효과음의 총 세 가지 영역의 음량을 조절할 수 있고, 마우스와 키보드 방향키 모두 사용 가능하다. Game Exit 키를 누르면 start scene 로 돌아가거나 아예 게임을 종료할 수 있다. 게임이 클리어 되지 않았을 때에는 대기실로 돌아갈 수는 없다. 이 UI 는 서버와 연동 사항이 없는 오로지 로컬 인터페이스다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

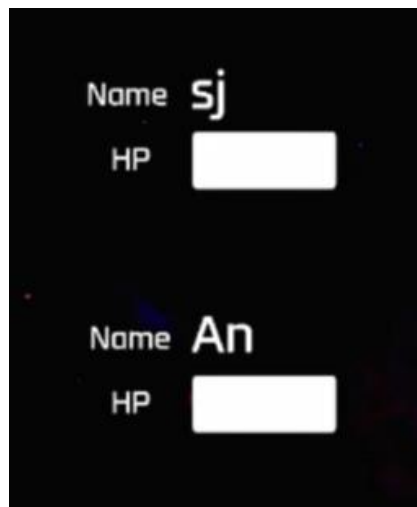
### 6.3.2 미션 진행도 표시 Bar



<미션 바>


유니티의 UI - Slider 기능을 이용해 구현하였다. 서버를 통해 실시간으로 미션 완수 정보를 받아와, 사용자 중 한명이 게임 내에 구현된 미니 게임 형태의 미션을 완수했을 경우 완료한 갯수에 비례해 미션 진행도의 바가 하얗게 차오른다. 동시에 미션을 완료한 갯수 / 총 미션 갯수 의 비율로 미션을 완성한 정도가 %로 나타난다. 예외로, 미션 달성도가 0%일 경우엔 슬라이더의 이미지가 막대 밖으로 튀어나오기 때문에 이를 방지하기 위해 0 이상의 초기값을 준다. 이때 텍스트로 표현되는 달성도는 0%를 유지한다.

### 6.3.3. 팀원들의 상태 표시창



<팀원 상태 표시창>

함께 게임을 진행하는 타 사용자들의 정보를 표시하는 인터페이스다. 서버로부터 함께 게임을 진행하는 사용자들의 정보를 받아와 그들의 닉네임, 남은 체력, 접속 상태 등을 표기한다. 접속 상태의 경우 평소에는 IDLE, 서버의 연결이 끊기거나 중간에 게임을 나가면 DISCONNECT, 몬스터에게 공격을 받아 체력이 0 이 되면 DEAD, 미션을 모두 완수하고 탈출에 성공하면 ESCAPE 등으로 표시된다. 상태창의 정보는 게임이 진행되는 실시간으로 갱신된다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

#### 6.3.4. 체력 표시 Bar



<체력 표시 바>


사용자 본인이 운영하는 캐릭터의 체력을 표시하는 상태창이다. 시작 시 총 2 칸이 채워져 있고, 맵을 순찰중인 적 AI에게 피격당할 때 체력이 1 칸 줄어든다. 여기서 회복 아이템을 사용하면 체력이 다시 1 칸 늘어난다. 현재 체력 수치 역시 서버를 통해 받아와 UI에 적용시킨다. 초기에는 UI - bar를 이용해 구현하였으나, 디자인적 요소를 위해 체력이 차 있을 때와 비어 있을 때, 총 2개의 이미지를 리스트화시켜 체력 상태에 따라 변경되도록 하였다. 체력이 줄어들 때는 위 칸부터 줄어들게 되며, 2칸을 초과하거나 0칸 미만으로 체력이 바뀌진 않는다. 체력이 0이 되어 체력 상태 창이 텅 비면 Game Over UI 창이 표시된다.

#### 6.3.5. 인벤토리



<인벤토리>

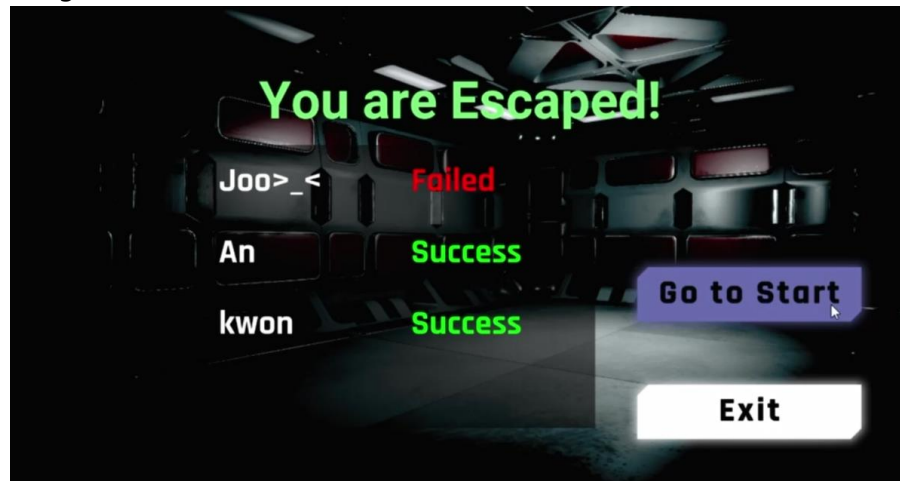
캐릭터가 게임을 진행하며 얻게 되는 아이템들의 개수와 종류를 표시하는 창이다. 총 4칸으로 이루어져 있다. 아이템을 획득하지 않았을 때는 오브젝트의 투명도를 0%로 하였다가, 아이템을 획득하면 투명도를 100%로 높이며 획득한 아이템 종류에 맞는

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

아이콘을 하단 화면에 표시한다. 게임을 진행하다 회복 아이템을 얻으면 하트 아이콘이, 적 방해 아이템을 얻으면 총 모양 아이콘이 왼쪽에서부터 차례로 생성된다.


아이템의 획득과 배치는 순서대로 이루어지나, 사용은 획득 순서에 상관없이 원하는 아이템을 사용할 수 있다. 왼쪽 슬롯에서부터 1, 2, 3, 4 순으로 키보드를 누르면 누른 칸으로 빨간 네모 모양의 커서가 움직인다. 그 상태에서 커서가 위치한 칸의 번호키를 한 번 더 누르면 해당 아이템을 사용할 수 있다. 커서의 움직임에는 Unity의 transform 함수를 사용하였고, 아이템의 선택과 사용은 키보드 입력에 bool 값으로 lock을 걸어, 키보드 번호키를 한 번 눌렀을 시 해당 칸의 bool 값이 true가 되고, 그 상태로 한 번 더 같은 키보드를 눌러야만 아이템 사용 함수가 호출된다. 인벤토리 또한 타 UI와 마찬가지로 서버와 실시간 연동이 이루어진다.

#### 6.4 Ending Scene



<Ending Scene>

플레이어가 엔딩 씬에 도달하였을 경우 카메라 FadeIn과 FadeOut을 통해 연출효과를 주었으며 파티클 시스템을 이용하여 텔레포터를 제작함.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

## 7. 조명 및 배경 음악

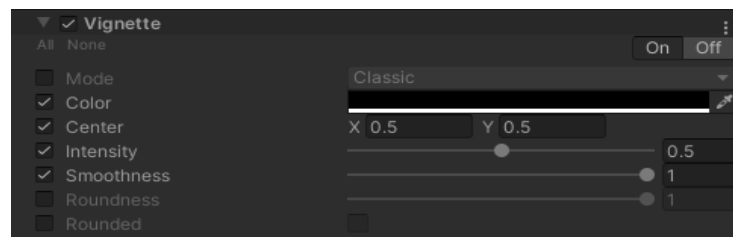
### 7.1 조명

공포감을 주기위해 어두운 분위기와 좁은 시야가 필요했다. 어두운 분위기를 위해 기본적으로 directional light 와 spot light 를 적극적으로 사용했으며 부족함을 느껴 Post Processing 을 이용했다. post processing 은 전체 화면에 필터를 씌우는 것과 같은 효과를 불러오고 시각적으로 극적인 효과가 나타나기 때문에 부족한 부분을 완벽히 채워주었다. 많은 기능 중에 극적인 효과를 준 기능은 Color Grading 과 Vignette 였다.



<Color Grading, Trackballs>

Color Grading 에서는 Lift, Gamma, Gain 값을 조정할 수 있다. 그 중 Lift 가 어두운 분위기를 내는데 효과적이었으며 과하게 어두워지지 않도록 Gamma 와 Gain 값을 설정했다.



<Vignette>

Vignette 에서는 시야를 좁힐 수 있다. 위의 사진에서 볼 수 있듯이 color, center, intensity, smoothness 를 설정할 수 있다. intensity 값을 정해 시야를 좁히고 smoothness 를 통해 경계면을 부드럽게 만들어 자연스러운 조명을 연출했다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |



<Post Processing 전(좌), 후(우)>

## 7.2 배경 음악

게임 컨셉에 맞게 긴장감을 유발할 수 있는 음악을 사용했다. 유니티 공식 에셋 스토어를 최대한 활용하였다.

## 8. 인게임 네트워크

서버가 인게임 씬으로 전환되면 다른 플레이어들이 Scene 을 로드하고 준비될 때까지 서버는 기다리게 된다. 모든 플레이어가 준비되면 플레이어 오브젝트, 적 오브젝트, 나머지 스폰 되어야 할 네트워크 오브젝트를 스폰 시킨다. 인 게임 중의 네트워크에는 플레이어, 적, 아이템박스, 미션, 아이템, UI, 엔딩 메시지의 네트워크 동기화가 이루어졌다.

플레이어 네트워크 동기화는 플레이어 움직임, 애니메이션, 물리엔진 등의 동기화가 이루어졌으며 또한 플레이어의 체력감소, 사망, 탈출 등의 상태변화도 동기화가 진행돼 모든 플레이어가 그 플레이어의 상태변화들을 확인할 수 있도록 하였다.

적의 네트워크 동기화는 움직임, 애니메이션, 물리엔진 등을 동기화를 진행하였다. 그리고 적을 서버에 스폰 시켜 서버가 적을 제어하도록 하였다. 아이템 박스와 미션의 네트워크 동기화작업은 플레이어가 아이템박스와 상호작용하고 있을 경우 다른 플레이어가 상호작용하지 못하도록 하였다. 미션을 클리어 했을 때 전체 미션 카운트가 올라갈 수 있도록 동기화를 진행하였다. 아이템의 경우 맵에 아이템과 플레이어가 직접가지고 있는 아이템을 네트워크 동기화 작업을 분리하여 진행하였다. 맵에 있는 아이템은 플레이어가 습득할 경우 그 맵 아이템이 다른 플레이어에게도 없어지도록 하였고, 플레이어가 손에 있는 아이템을 활성화시키는 것도 동기화를 진행하였다. 힐팩 아이템의 경우 플레이어 체력 상태변화를 통해 동기화가 진행되었고, 총 아이템의 경우 서버에 정해진 곳에 총알을 스폰 시켜 총알에 여러 Network Component 를 통해 총알의 궤적의 동기화 작업이 이루어지도록 하였다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

UI 네트워크 동기화는 다른 플레이어의 상태 변화를 파악할 수 있도록 동기화 작업이 이루어졌다. 체력이나 엔딩 상태 등은 플레이어 스크립트에서 동기화가 이루어졌기에 전체 플레이어의 정보를 담는 배열을 만들어 주기적으로 UI에서 그 정보를 파악할 수 있다.

엔딩 메시지에서 다른 플레이어들이 죽거나 탈출했을 때 정보를 가지고 있다가 플레이어가 엔딩에 다다르면 그 정보를 바탕으로 UI를 표시할 수 있도록 하였다.




|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

## 2.2.2 시스템 기능 요구사항

### 1. 게임 클라이언트

| 항목       | 내용   | 완료 여부 |
|----------|--|-------|
| 플레이어 움직임 | 플레이어 캐릭터의 움직임이 사용자가 원하는 대로 이루어져야 한다.                       | 완료    |
| 적 AI 구현  | 적은 추가적인 입력 없이 스스로 움직일 수 있어야 한다.                            | 완료    |
| 적 기능 구현  | 적은 순찰, 추적 기능이 있어야 하며 플레이어를 공격할 수 있어야 한다.                   | 완료    |
| 감각 센서 구현 | 적은 플레이어를 탐지할 수 있어야 하며, 시각과 청각이 있는 것처럼 보여야 한다.              | 완료    |
| 타겟 설정    | 적은 탐지된 플레이어와의 거리를 측정해서 가장 가까운 플레이어를 타겟으로 삼아야 한다.           | 완료    |
| 미션 공유    | 미션의 진행과 완료 상태가 모든 플레이어들에게 동등하게 공유되어야 한다.                   | 완료    |
| 아이템 기능   | 아이템 박스에 아이템이 배치되어 있어야 하고 플레이어와 상호작용을 할 수 있어야 한다.           | 완료    |
| 탈출 조건 설정 | 미션 진행도가 100%가 되어야만 엔딩을 진행할 수 있어야 한다.                       | 완료    |
| 인게임 편의기능 | 사용자 인터페이스를 통해 본인의 체력, 미션 진행도, 팀원의 상태 등이 원활하게 표시되어야 한다.     | 완료    |
| 공포감 조성   | 플레이어의 시야를 줄이고 어두운 분위기를 만들어야 한다.                            | 완료    |
| 미션 구현    | 직관적이고 다양한 미션을 만들고 디자인으로 어떤 미션인지 특정할 수 없도록 같은 디자인을 이용해야 한다. | 완료    |
| 유저 편의기능  | 유저들에게 버튼과 슬라이더 등을 통해 UI 를 제공하며 기능을 내포하는 텍스트도 같이 제공해야 한다.   | 완료    |

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

## 2. 게임 서버

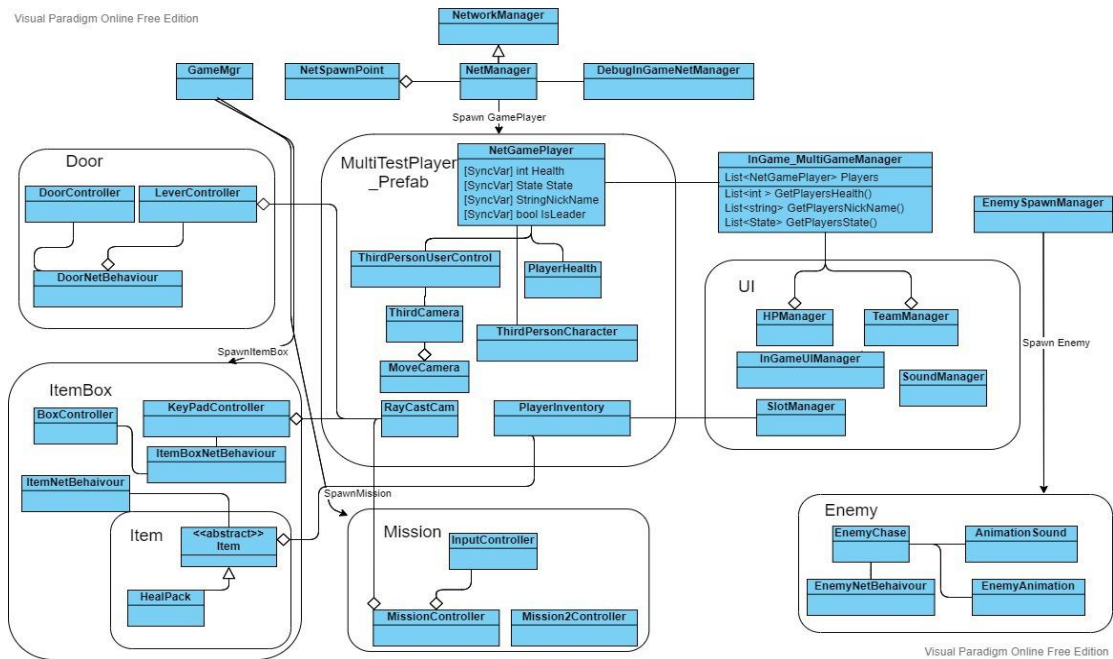
| 항목       | 내용   | 완료 여부 |
|----------|--|-------|
| 방 생성     | 유자가 원할 때 방을 생성할 수 있어야한다.                                       | 완료    |
| 방 접속     | 유자가 원하는 방을 찾아 접속할 수 있어야한다.<br>->유자는 미리 생성된 방의 코드를 입력해 방에 접속한다. | 변경    |
| 오브젝트 스폰  | 서버에서는 게임에 사용되는 플레이어와 적, 미션 오브젝트들을 정상적으로 스폰 시켜야 한다.             | 완료    |
| 오브젝트 동기화 | 모든 네트워크와 연관된 오브젝트 들은 각 플레이어들에게도 동등하게 공유되어야 한다.                 | 완료    |

### 2.2.3 시스템 비기능(품질) 요구사항

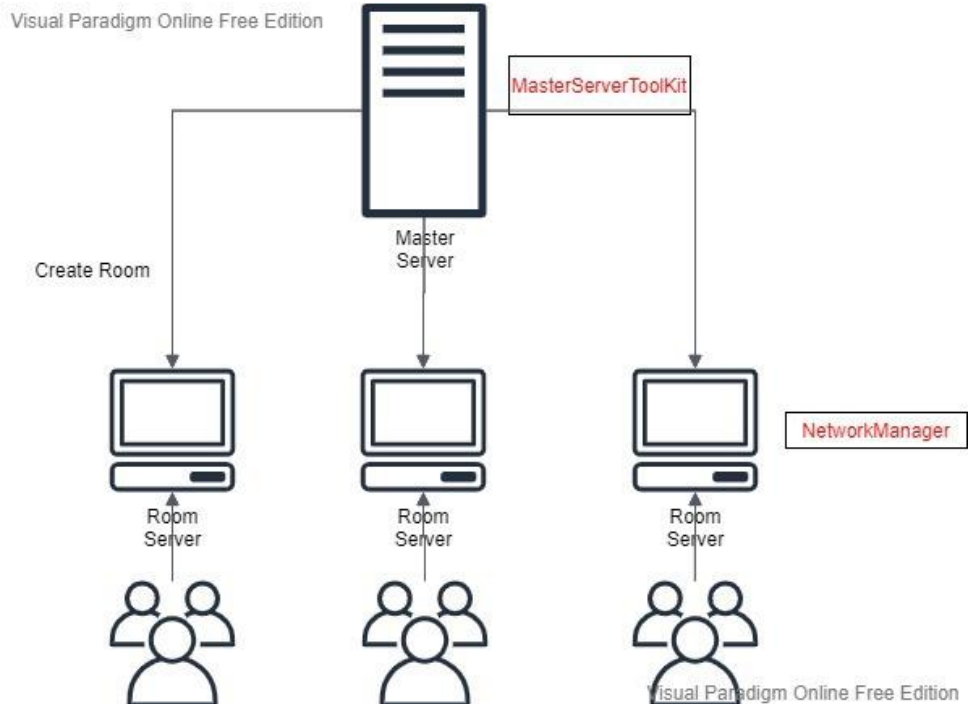
| 항목     | 설명  | 달성 여부 |
|--------|---|-------|
| 사용성    | 사용자가 쉽게 사용할 수 있어야 한다.                     | 달성    |
| 효율성    | 게임이 10 초 이내에 실행되고 메모리를 적게 차지해야 한다.        | 달성    |
| 신뢰성    | 게임을 클리어할 때까지 오류가 발생할 확률이 낮아야 한다.          | 달성    |
| 윤리적 책임 | 어린이, 청소년이 해도 관심을 정도로 부적절한 요소가 없어야 한다.     | 달성    |
| 접근성    | Windows PC 라는 환경에서는 누구나 설치하여 구동할 수 있어야한다. | 달성    |
| 안정성    | 한 플레이어의 접속 오류가 다른 플레이어에게 영향이 최소화돼야 한다.    | 달성    |



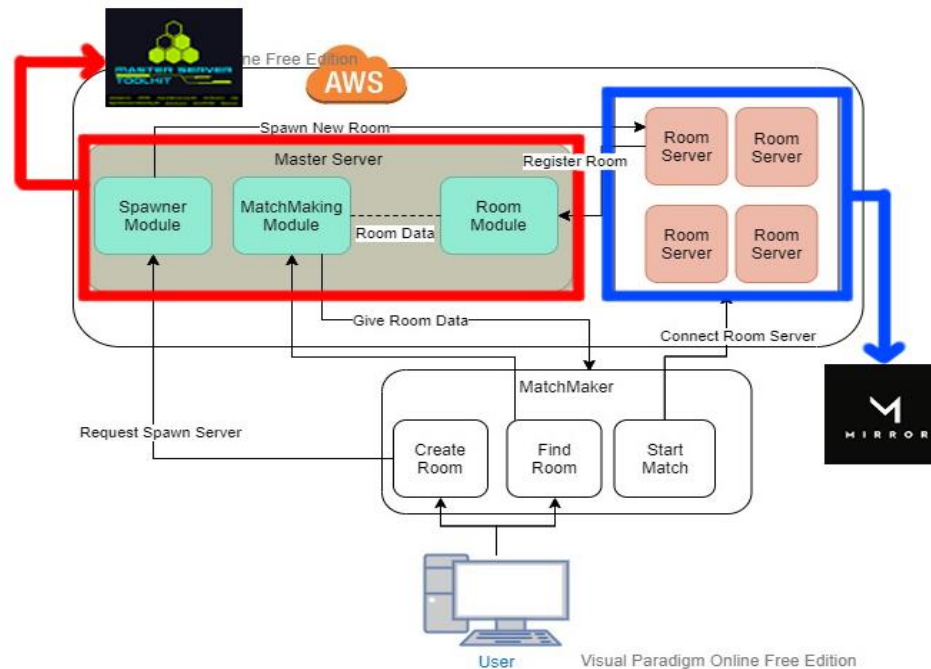
## 2.2.4 시스템 구조 및 설계도



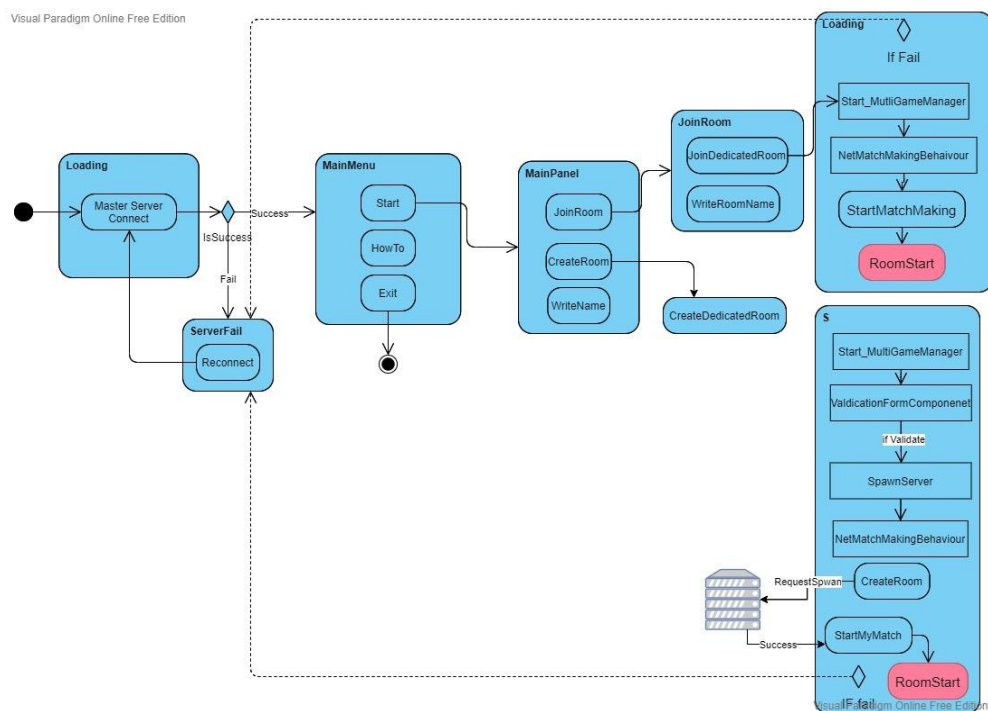
<전체 클래스 다이어그램>



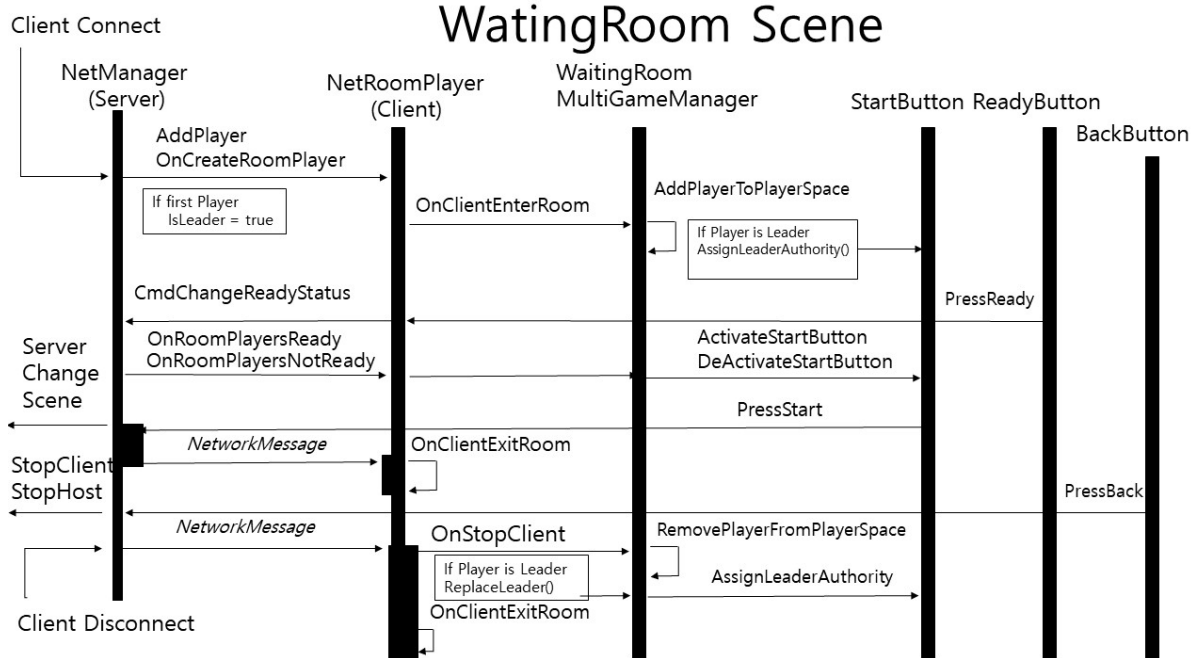
<마스터 서버- 룸 서버 계층 구조 다이어그램>



<마스터 서버- 방 생성, 방 조인 액티비티 다이어그램>

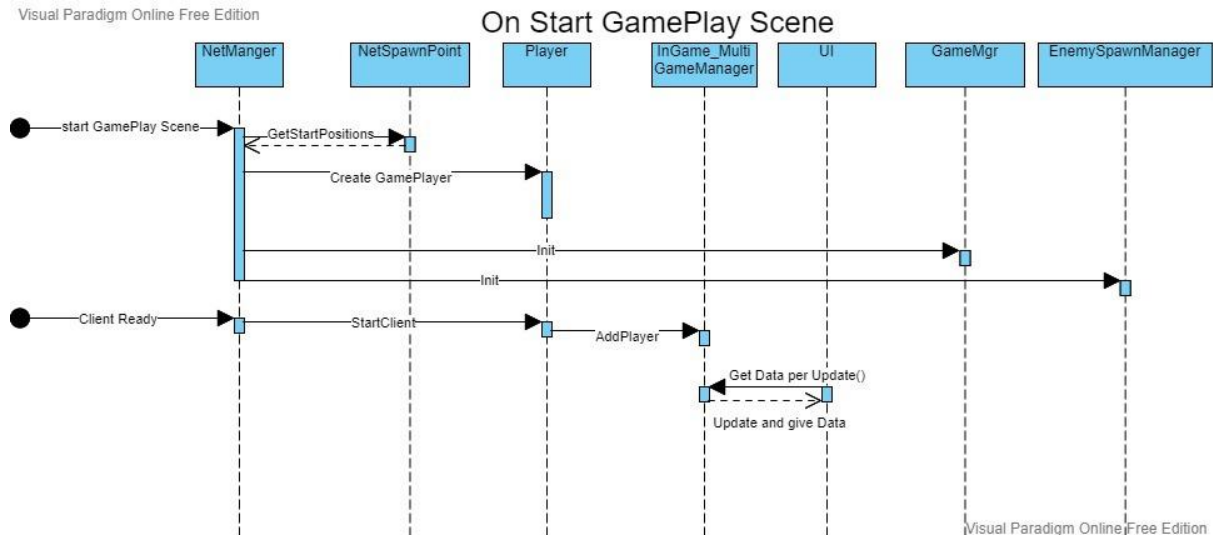


### <시작 씬 액티비티 다이어그램>



<대기실 시퀀스 다이어그램>

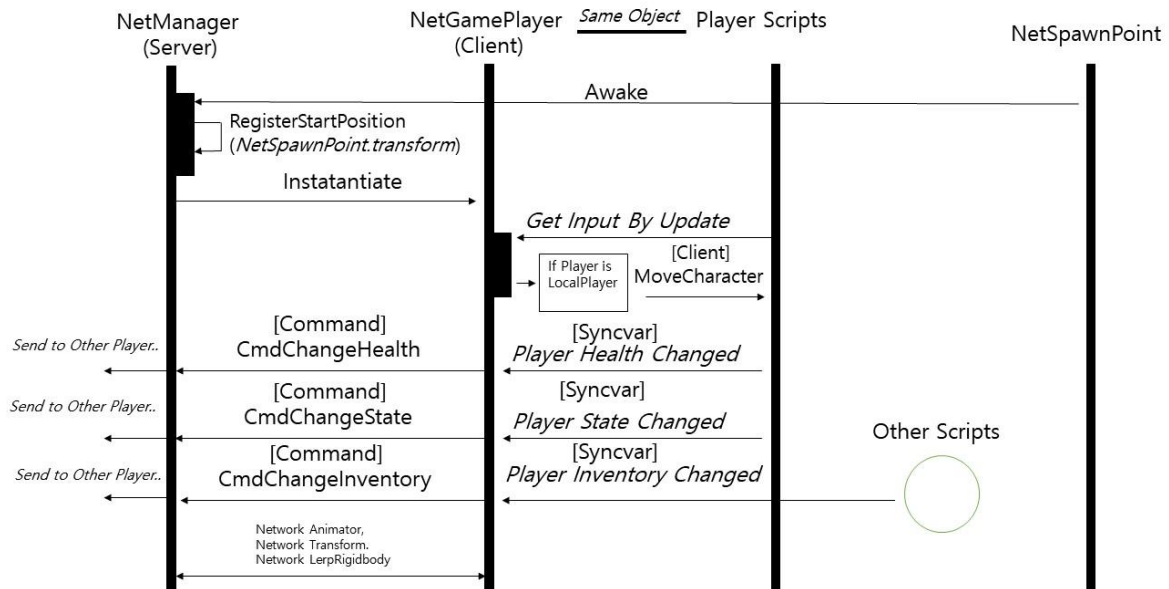
Visual Paradigm Online Free Edition



<인게임 플레이 시작 시퀀스 다이어그램>



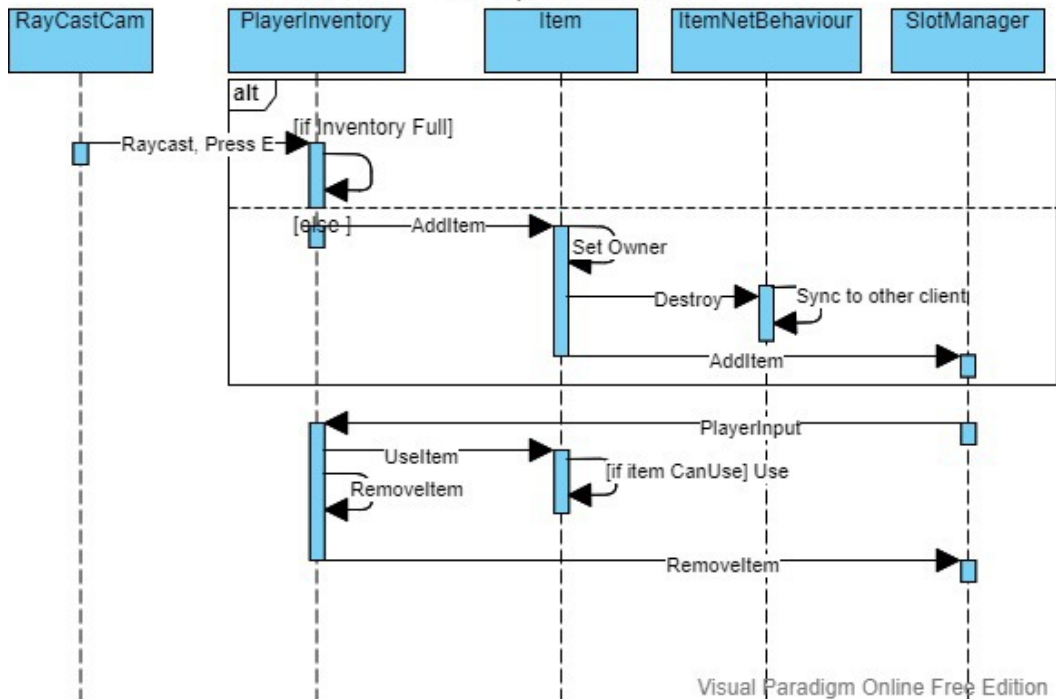
### GamePlay Scene (Network Player)



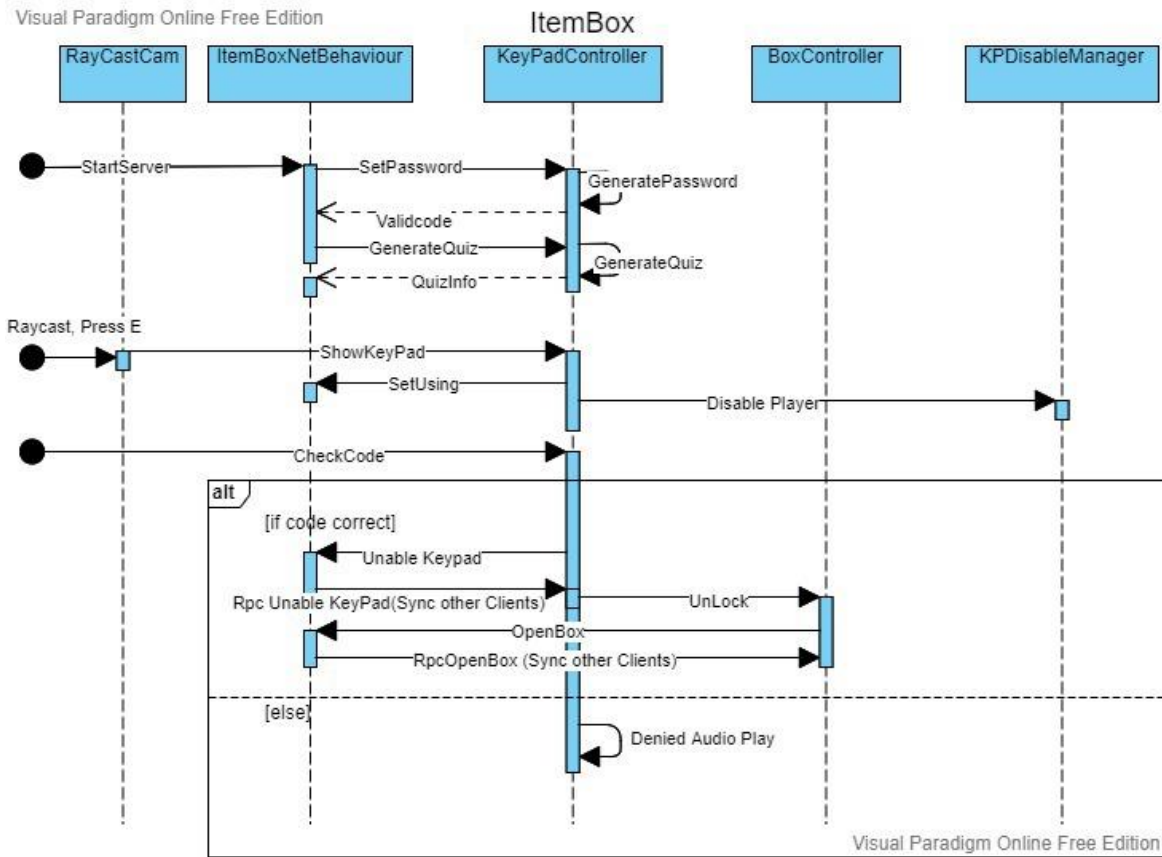
<플레이어 네트워크 시퀀스 다이어그램>

Visual Paradigm Online Free Edition

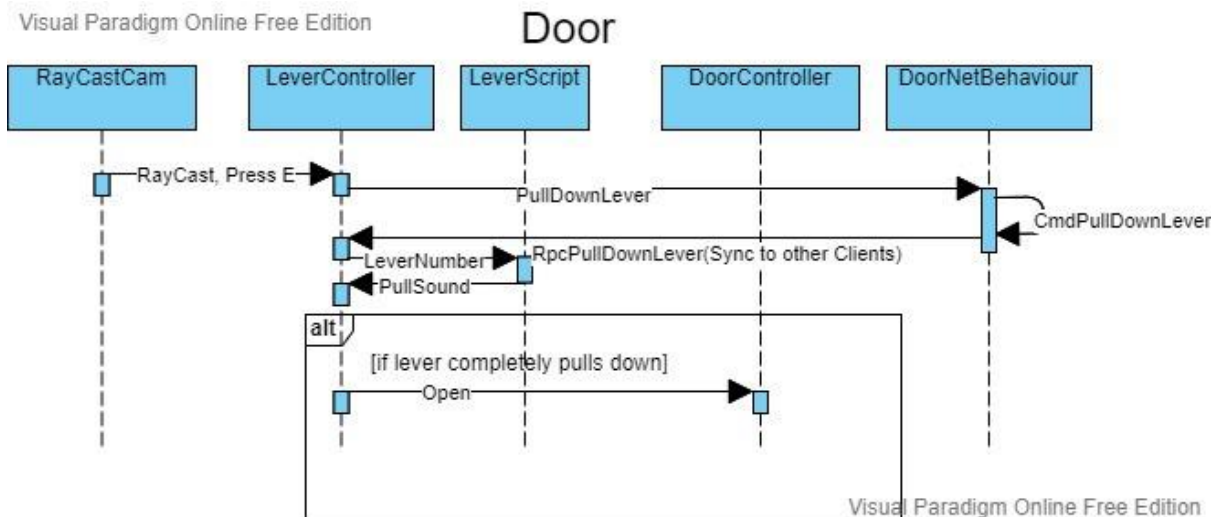
### Pick UP Item, Use Item



<플레이어 인벤토리 상호작용 다이어그램>



<아이템박스 시퀀스 다이어그램>

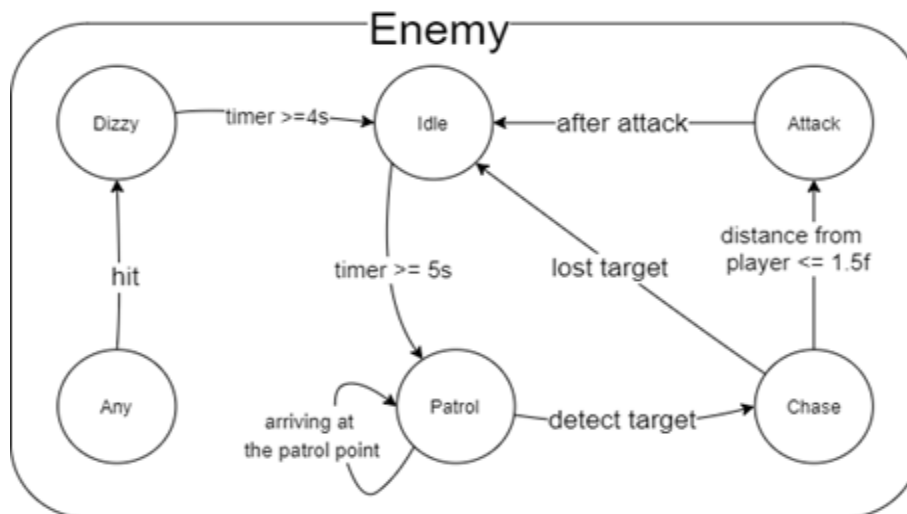
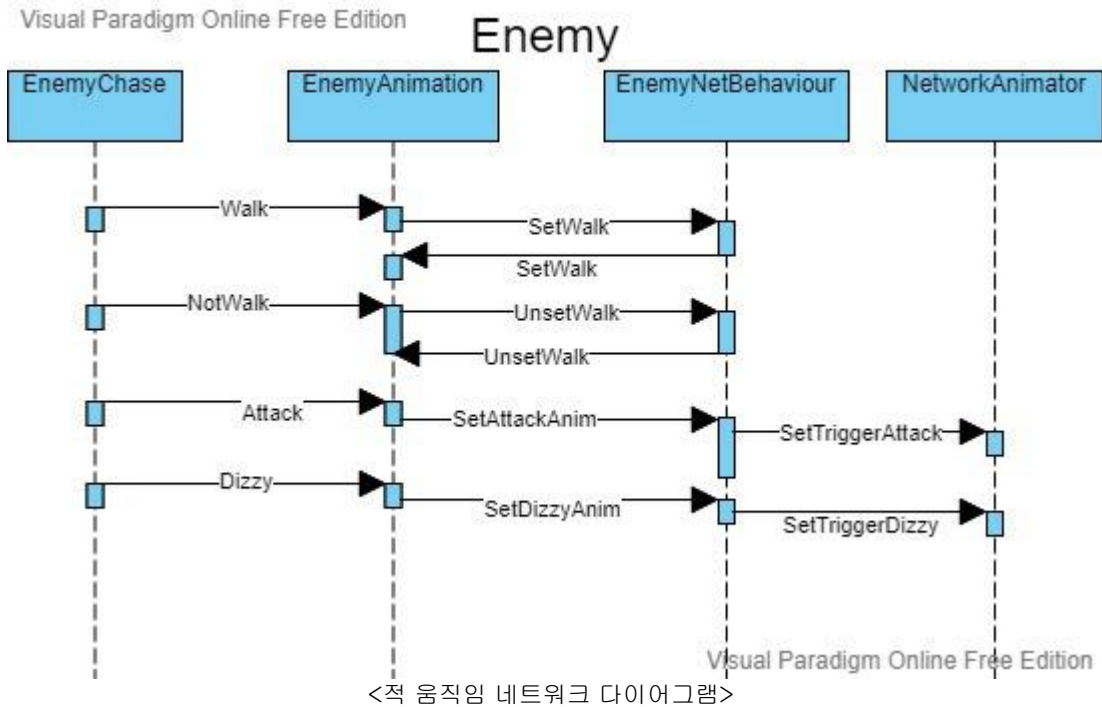


<탈출구 시퀀스 다이어그램>





| 결과보고서                   |             |             |
|-------------------------|-------------|-------------|
| 프로젝트 명                  | Raider      |             |
| 팀 명                     | Team 17     |             |
| Confidential Restricted | Version 2.0 | 2021-MAY-26 |





|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

### 2.2.5 활용/개발된 기술

**Unity Engine** : 게임 엔진으로 그래픽, 오디오, 물리엔진, 애니메이션, UI 시스템 등 게임 개발에 있어 필수적인 요소를 포함하고 있는 개발 소프트웨어

**Master Server Toolkit** : 유니티에 적용되는 오픈소스로 유니티 개발에 있어 Matchmaking 등 인 게임 플레이어 매칭에 도움이 되는 기술들을 포함하고 있음


**Mirror Network** : 유니티에 적용되는 오픈소스로 인게임 중 네트워크 상호작용에 관한 기술을 가지고 있음.

### 2.2.6 현실적 제한 요소 및 그 해결 방안

- ✓ 소프트웨어 상으로 다양한 플랫폼으로 지원하는 것은 무리가 있어 PC 버전으로만 선택하여 개발을 진행함.
- ✓ 프로젝트가 멀티플레이어 게임이기에 디버깅이 혼자 하기 쉽지 않았음. 때문에 마무리기간 동안 매일 QA 모임을 진행하여 버그를 수정함.
- ✓ 유니티 씬 파일은 github 로 auto merge 가 안된다는 단점이 있었음. 때문에 최대한 메시지를 통해 충돌이 안나도록 소통하며 진행함.

### 2.2.7 결과물 목록

| 대분류      | 소분류       | 기능                              | 진행 상태 | 수정 사항 |
|----------|-----------|---------------------------------|-------|-------|
| 네트워크     | 마스터 서버    | 방 생성, 방 접속                      | 완료    |       |
|          | 인게임 네트워크  | 게임 중 오브젝트와의 네트워크 동기화            | 완료    |       |
| 플레이어 캐릭터 | 움직임       | 키보드, 마우스 입력으로 인하여 플레이어 움직임을 제어  | 완료    |       |
|          | 카메라       | 3 인칭 카메라 구현, 벽 통과 방지            | 완료    |       |
|          | 아이템과 상호작용 | 아이템 사용 시 root 에 있는 아이템 오브젝트 활성화 | 완료    |       |
|          | 애니메이션     | 여러 상태에 맞는 애니메이션 적용              | 완료    |       |

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

|        |        |   |                                |    |
|--------|--------|---|--------------------------------|----|
|        | 쉐이더    | 벽 뒤에 있을 때 플레이어 위치를 색으로 확인할 수 있음   | 완료                             |    |
|        | 체력     | 적에게 맞을 때 체력 감소<br>체력 회복제를 마시면 체력 상승   | 완료                             |    |
|        | 적      | FSM   | 상태들에 맞게 동작하며 트랜지션이 자연스럽게 일어난다. | 완료 |
| 맵 & 미션 | 애니메이션  | 적의 자연스러운 애니메이션 구현   | 완료                             |    |
|        | 감각 센서  | 시각과 청각을 가진 것처럼 행동한다.  | 완료                             |    |
|        | 맵      | 기획의도와 맞는 게임의 맵 구성   | 완료                             |    |
|        | 미션     | 게임의 요소인 미션 3 개 제작   | 완료                             |    |
| UI     | 아이템 박스 | 아이템 획득이 가능한 박스 제작   | 완료                             |    |
|        | 아이템    | 체력회복제, 전기충격기 구현   | 완료                             |    |
|        | 스타트 씬  | 대기실 입장, 게임 안내문, 게임 종료 키, 게임 크레딧 생성  | 완료                             |    |
|        | 인게임 씬  | 미션 진행도, 팀원 상태 표시창, 인게임 메뉴 UI, 플레이어 체력 표시창, 인벤토리 구현  | 완료                             |    |
| 조명& 음악 | 로딩 씬   | 서버 비동기화 상태에서 로드 가능한 로딩용 씬을 제작, 씬과 씬이 체인지 될 때마다 사이에 삽입되도록 한다.  | 완료                             |    |
|        | 엔딩 씬   | 게임을 졌지 않고 클리어 한 플레이어와 그렇지 못한 플레이어를 구분 지어 표기한다. 엔딩 창에서 다시 start scene, 혹은 waiting room 으로 돌아갈 수 있는 버튼을 구현한다. | 완료                             |    |
| 조명& 음악 | 시야     | Post Processing 의 Vignette 를 이용해 시야를 줄였다.   | 완료                             |    |

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |


|  |        |   |    |  |
|--|--------|---|----|--|
|  | 어두운 조명 | Post Processing 의 Color Grading 옵션 중 Lift, Gamma, Gain 값을 수정해 어두운 톤을 극대화. | 완료 |  |
|  | 배경음    | 게임 기획에 맞게 긴장감이 느껴지는 배경음악을 선택.   | 완료 |  |
|  | 효과음    | 자연스러운 연출을 위해 애니메이션에 이벤트 부여  | 완료 |  |

## 2.3 기대효과 및 활용방안

1. 자극적이지 않은 디자인과 연출을 사용하여 공포게임을 선호하지 않았던 유저들도 재미있게 플레이 할 수 있다.
2. 여러 사용자가 함께 공동의 목표를 이루며 게임을 진행하여 협동심을 기를 수 있다.
3. 직관적이고 쉬운 게임 난이도로 게임의 진입 장벽을 낮춰 다양한 유저층이 플레이할 수 있다.
4. 게임의 폭력성과 잔인한 요소가 없어 아동, 청소년들에게 부정적인 영향을 미치지 않는다.


## 3 자기평가

1. 팀원 전원이 게임 개발 경험, 다인 개발 경험, 지식이 부족했음에도 불구하고 성공적으로 초기 기획 의도에 맞는 Dedicated Server 방식의 멀티플레이어 게임을 만들었다.
2. 각기 다른 역할의 팀원들이 같이 하나의 게임을 만들어보면서 게임 개발 경험을 쌓을 수 있었다.
3. 기획부터 개발, QA 까지 실제 게임업계에서 진행되는 절차대로 프로젝트를 진행해 게임개발의 전체적인 흐름을 알 수 있었다.
4. Matchmaking, UI 애니메이션 효과, 디자인 다양화 등 퀄리티 업과 관련된 요소를 추가하지 못한 것에 아쉬움이 남으나, 시간과 개인의 능력을 고려해 계획을 세워야 한다는 멘토링 피드백의 결과를 반영하여 구현 요소를 추가하는 것보다 게임의 안정화에 시간을 들였다.
- 5.조명과 오디오로 공포감을 이끌어냈지만 음악적인 요소가 조명에 비해 약하다는 아쉬운 점이 있다.

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

## 4 참고 문헌

| 번호 | 종류   | 제목  | 출처  | 발행년도 | 저자                             | 기타 |
|----|------|---|---|------|--------------------------------|----|
| 1  | 웹페이지 | Unity User Manuel   | <a href="https://docs.unity3d.com/Manual/index.html">https://docs.unity3d.com/Manual/index.html</a>   | 2020 | Unity                          |    |
| 2  | 웹페이지 | Master Server Toolkit Tutorial                                | <a href="https://master-toolkit.com/#page-start">https://master-toolkit.com/#page-start</a>   | 2021 | Aevien                         |    |
| 3  | 웹페이지 | Mirror Network Document                                       | <a href="https://mirror-networking.gitbook.io/docs/">https://mirror-networking.gitbook.io/docs/</a>   | 2021 | vis2k,<br>james<br>-<br>frowen |    |
| 4  | 웹페이지 | State · Design Patterns Revisited · Game Programming Patterns | <a href="http://gameprogrammingpatterns.com/state.html">http://gameprogrammingpatterns.com/state.html</a>   |      |                                |    |
| 5  | 웹페이지 | Post Processing   | <a href="https://docs.unity3d.com/Packages/com.unity.postprocessing@2.3/manual/index.html">https://docs.unity3d.com/Packages/com.unity.postprocessing@2.3/manual/index.html</a> | 2020 | Unity                          |    |
| 6  | 웹페이지 | ShaderLab:SubShader   | <a href="https://docs.unity3d.com/kr/530/Manual/SL-Pass.html">https://docs.unity3d.com/kr/530/Manual/SL-Pass.html</a>   | 2018 | Unity                          |    |

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

## 5 부록

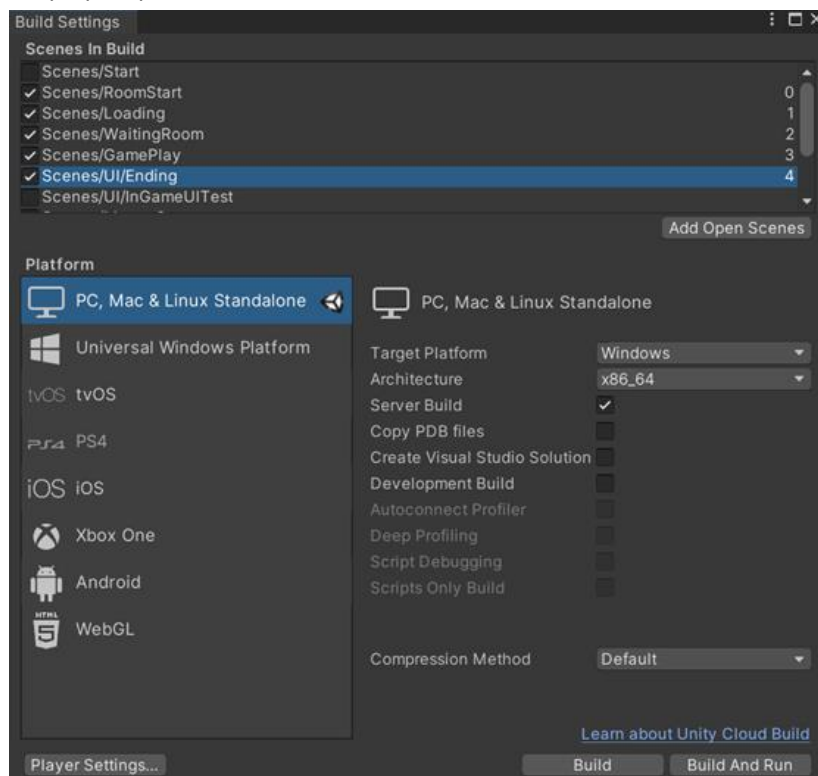
### 5.1 사용자 매뉴얼

깃 허브 프로젝트 Released 탭에 있는 파일을 다운로드 후 Raider.exe 파일을 실행


| 이름                     | 수정한 날짜              | 유형         | 크기       |
|------------------------|---------------------|------------|----------|
| Horror3DMultiGame_Data | 2021-05-24 오전 12:59 | 파일 폴더      |          |
| MonoBleedingEdge       | 2021-05-24 오전 12:59 | 파일 폴더      |          |
| Raider_Data            | 2021-05-24 오전 4:25  | 파일 폴더      |          |
| application            | 2021-05-24 오전 12:59 | CFG 파일     | 0KB      |
| Log                    | 2021-05-24 오후 5:54  | 텍스트 문서     | 2KB      |
| Raider                 | 2021-05-24 오전 4:25  | 응용 프로그램    | 639KB    |
| UnityCrashHandler64    | 2020-12-22 오전 3:43  | 응용 프로그램    | 1,221KB  |
| UnityPlayer.dll        | 2020-12-22 오전 3:43  | 응용 프로그램 확장 | 27,523KB |

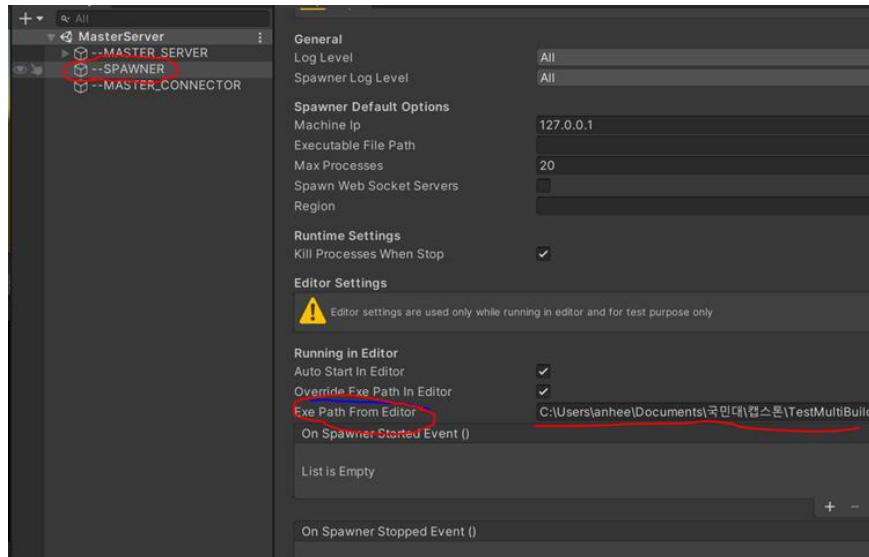
### 5.2 서버 운용 매뉴얼

Unity 2020.2 이상의 버전이 필요함.

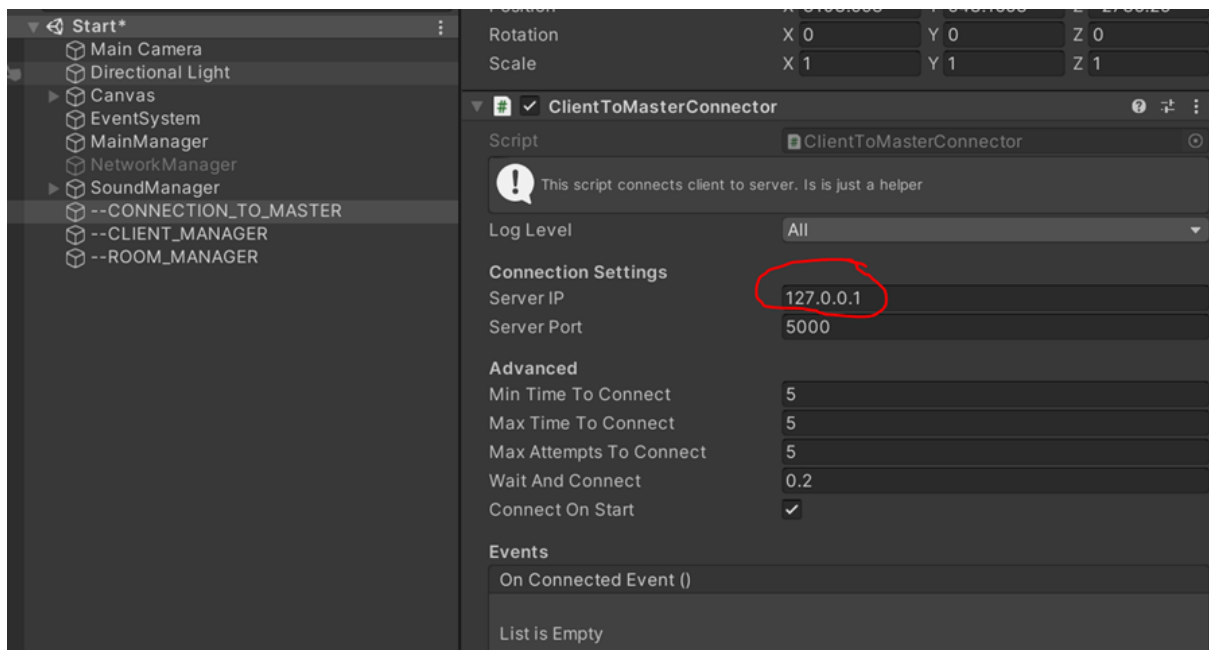


1. 소스코드를 다운 후 다음과 같은 설정으로 Room Server Build 를 진행


|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <div> 국민대학교<br/> 소프트웨어학부<br/> 캡스톤 디자인 I </div> | 결과보고서                   |             |             |
|  | 프로젝트 명                  | Raider      |             |
|  | 팀 명                     | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

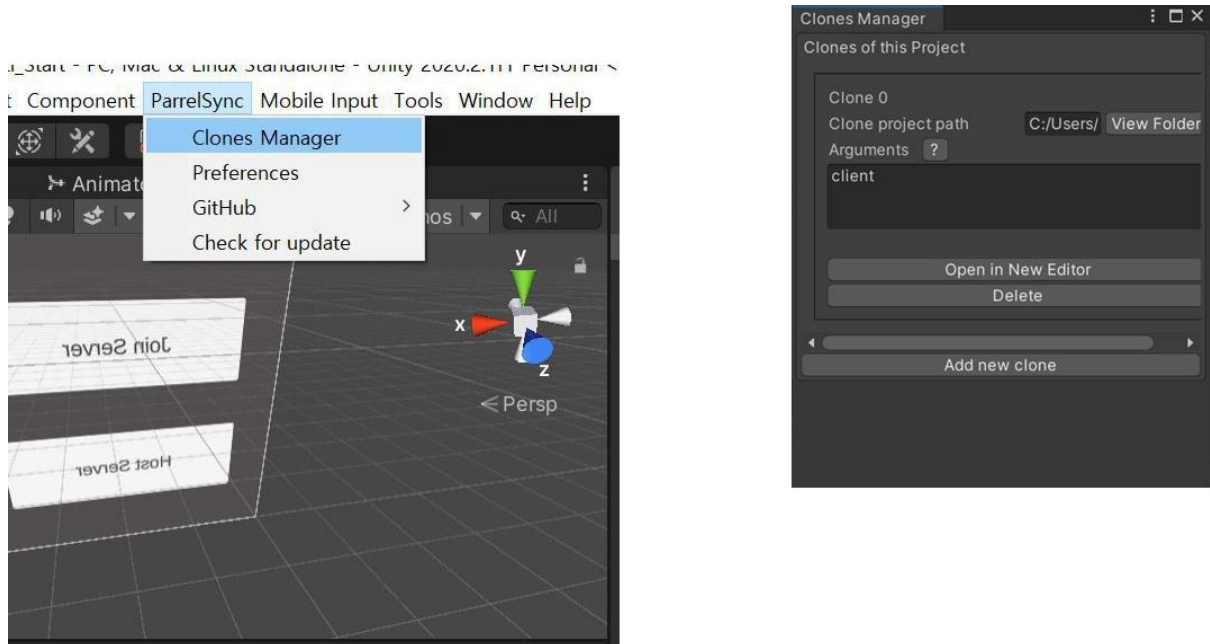


2. 이후 MasterServer 씬에서 Spawner Object 의 Exe Path From Editor 의 경로 설정을 룸 서버 빌드 파일 설정으로 완료함.

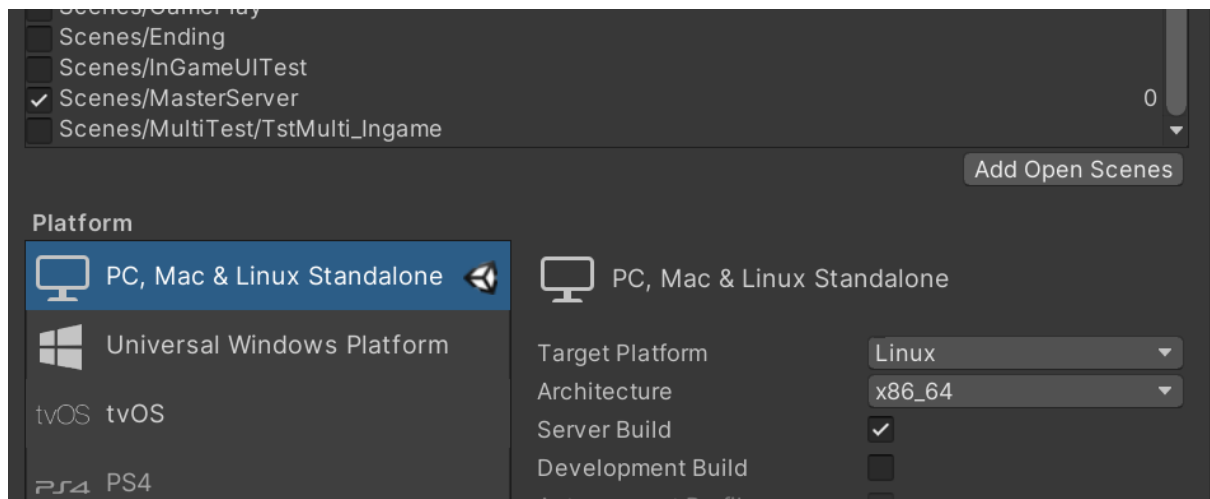


3. Start Scene 에서 --Connection\_TO\_MASTER 의 ServerIP 를 설정


|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |




4. 이후 ParrelSync 를 통해 복수의 에디터를 실행한후 한 에디터는 MasterServer Scene 실행, 한 에디터는 Start Scene 을 실행하면서 결과를 확인할 수 있음.



5. 추가적으로 서버 빌드를 위해서는 MasterServer 씬만 따로 빌드한 뒤

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

 \*application - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)


```
-mstStartMaster=True
-mstStartSpawner=True
-mstStartClientConnection=True
-mstRoomExe=[Room Server Build 폴더 경로지정]
```

6. application.cfg 파일을 다음과 같이 수정할 시 에디터를 사용하지 않고 마스터 서버를 실행할 수 있음

### 5.3 테스트 케이스

| 대분류  | 소분류      | 기능            | 테스트 방법   | 기대 결과                   | 테스트 결과 |
|------|----------|---------------|--|-------------------------|--------|
| 네트워크 | 마스터 서버   | 방 생성,<br>방 접속 | AWS 에서 구동되는 서버를 바탕으로 접속해본다. 각기 다른 플레이어가 방이름을 바탕으로 한 방에 접속될 수 있는지 확인한다. | 대기실 내에 다른 플레이어가 확인된다.   | 성공     |
|      | 인게임 네트워크 | 인게임 씬 전환      | 대기실 내에서 인게임으로 전환됐을 시 모든 플레이어, 적, 다른 네트워크 오브젝트 들이 빠짐없이 스폰 되는지 확인한다.     | 다른 오브젝트가 스폰 되는 것이 확인된다. | 성공     |




|   |  |                         |             |             |
|---|--|-------------------------|-------------|-------------|
|  | <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|   |  | <b>프로젝트 명</b>           | Raider      |             |
|   |  | <b>팀 명</b>              | Team 17     |             |
|   |  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |


|             |             |                                    |   |  |    |
|-------------|-------------|------------------------------------|---|--|----|
| 네트워크        | 인게임<br>네트워크 | 상호작용                               | 맵 오브젝트와<br>상호작용할 때 다른<br>플레이어가 접근할 수<br>없는지 확인한다.   | 다른 유저가<br>오브젝트와<br>상호작용한다는<br>디버그 메시지가<br>띄워진다.  | 성공 |
| 플레이어<br>캐릭터 | 애니메이션       | 걷기, 앉기,<br>아이템사용,<br>사망 모션<br>등 실행 | 서버내에서 접속됐을시<br>모든 플레이어에게 같은<br>애니메이션이 보이는 지<br>확인한다.  | 잘못된<br>애니메이션 없이<br>여러<br>애니메이션이<br>공유되는 것이<br>확인된다.  | 성공 |
|             | 인벤토리        | 아이템 손에<br>들기,<br>아이템 사용            | 플레이어가 손에<br>아이템을 들었을시 다른<br>플레이어도 그<br>플레이어가 손에<br>아이템을 든것이<br>확인한다. 총 아이템의<br>경우 총알이 정상적으로<br>작동되는지 확인한다.            | 플레이어가<br>아이템을 손에<br>든 것이<br>공유되며 총알이<br>정상계도로<br>발사된다.   | 성공 |
|             | 체력          | 체력감소,<br>체력증가,<br>사망               | 플레이어 체력이 다른<br>플레이어에게도 같은<br>수치로 유지되는지<br>확인한다.   | 체력 UI 를 통해<br>플레이어의 체력<br>상태변화가<br>실시간으로<br>확인된다.  | 성공 |
| 적           | FSM         | 기본                                 | 적에게 카메라를 부여해<br>정상적으로 멈추는지<br>확인한다.<br><br>이전 상태에 따라 다른<br>행동을 해야 하기 때문에<br>Debug.Log 를 이용해<br>이전 상태에 따른 결과를<br>출력한다. | 부여한 딜레이<br>만큼 기본<br>애니메이션을<br>재생한 뒤<br>순찰상태로<br>들어가 순찰을<br>진행한다.<br><br>이전상태에 따라<br>정상적으로<br>핸들링 된다. | 성공 |
|             |             | 순찰                                 | 순찰 지점을<br>Debug.Log 로<br>출력하면서 그 지점에<br>제대로 도달하는지   | 새로운 Log 가<br>발생할 때마다   | 성공 |

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

|   |       |            |   |  |    |
|---|-------|------------|---|--|----|
| 적 | FSM   | 순찰         | 시점이 자유로운 카메라를 이용해 확인.   | 다른 위치로 이동한다.   |    |
|   |       | 추적         | 플레이어를 조작해 적 주변을 돌아다니며 탐지되도록 한다.                                   | 탐지가 되면 사이렌을 울리며 쫓아오고 점차 뛰는 애니메이션으로 바뀌고 빨라진다.           | 성공 |
|   |       | 공격         | 플레이어를 적의 정면에 놔둬 제대로 공격하는지 확인하고 정상적으로 피격 판정이 발생하는지 확인한다.           | 플레이어가 공격에 맞고 애니메이션을 실행하며 적의 collider 가 정상적으로 작동한다.     | 성공 |
|   |       | 기절         | 적을 공격해 정상적으로 기절 애니메이션이 나오는지 확인한다.                                 | 정상적으로 기절 애니메이션을 실행하고 기본 상태로 이동한다.                      | 성공 |
|   | 감각 센서 | 시각         | 시야에 들어온 플레이어에게 Raycast 로 선을 긋고, 타겟으로 설정되면 Debug.Log 를 이용해 확인한다.   | 시야에 들어온 모든 플레이어에 선이 그어지고 그 중 가장 가까운 적의 이름만 Log 로 출력된다. | 성공 |
|   |       | 청각         | 청각 범위 내에서 플레이어를 움직여 정상적으로 추적상태로 들어가는지 확인. 이번에도 Debug.Log 로 타겟 확인. | 적의 뒤에서 움직여도 탐지해 추적한다.                                  | 성공 |
|   | 맵     | 맵 오브젝트 안정성 | 플레이어가 맵에서 떨어지지 않도록 맵이 잘 구성되었는지 확인한다.                              | 맵 밖으로 플레이어가 나가지 않는다.                                   | 성공 |
|   |       | 맵 & 미션     |   |  |    |

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

|    |             |                               |  |   |    |
|----|-------------|-------------------------------|--|---|----|
|    | 미션          | 미션 상호작용                       | 플레이어가 미션에 상호작용해서 미션을 해결하는데 문제가 발생하는지 확인한다.                 | 문제없이 미션을 해결할 수 있다.  | 성공 |
|    | 아이템박스       | 아이템박스 상호작용                    | 플레이어가 아이템 박스에 상호작용해 박스를 열었을 때 아이템이 있는지 확인한다.               | 박스가 열리며 아이템 획득이 가능하다.   | 성공 |
| UI | Start Scene | Start scene > Waiting room 접속 | Start 씬에서 닉네임, 방장 여부 등을 정한 뒤 Waiting 룸에 무사히 참여가 가능한지 확인한다. | Start scene 에서 정한 닉네임과 방장 여부가 waiting 룸에 적용되어 플레이어의 정보가 표시되고, 서버의 끊김이 존재하지 않는다. | 성공 |
|    | 팀 UI        | 이름, 체력, 엔딩 상태 표시              | 서버 접속 뒤 플레이어 상태변화에 따라 즉각적으로 변화가 되는지 확인한다.                  | 다른 플레이어 체력이 줄어든다.0 일시 죽은 상태가 표시된다   | 성공 |
|    | 인게임 메뉴 UI   | 설정, How To, 게임종료가 포함          | 모든 메뉴 UI 의 마우스버튼이 작동되며 버튼에 따른 기능이 동작되는지 확인한다.              | 각기 다른 UI 버튼을 눌렀을 때 정상 작동된다.   | 성공 |
|    | 플레이어 상태 UI  | 미션 진행도, 현재 체력 상태 표시창          | 게임 사용자의 정보가 실시간으로 서버와 연동되고 맞는 상태로 UI 가 표시되는지 확인한다.         | 미션의 진행도에 따라 흰색으로 미션 진행 바가 차오르고, 플레이어의 체력이 적의 피격 여부나 아이템의 사용 여부에 따라 즉각 정보가 갱신된다. | 성공 |

|  |                         |             |             |
|--|-------------------------|-------------|-------------|
|  <b>국민대학교</b><br><b>소프트웨어학부</b><br><b>캡스톤 디자인 I</b> | <b>결과보고서</b>            |             |             |
|  | <b>프로젝트 명</b>           | Raider      |             |
|  | <b>팀 명</b>              | Team 17     |             |
|  | Confidential Restricted | Version 2.0 | 2021-MAY-26 |

|  |    |                   |  |                               |    |
|--|----|-------------------|--|-------------------------------|----|
|  | 엔딩 | 텔레포터,<br>엔딩 상태 UI | 플레이어가 텔레포터에<br>들어간 후에 엔딩 UI 가<br>뜨는지 확인한다.<br>정상적으로 플레이어의<br>상태가 나오는지 처음<br>들어간 플레이어,<br>마지막으로 들어간<br>플레이어 둘 다 정상인지<br>확인한다. | 정상적으로<br>플레이어의 엔딩<br>상태가 보인다. | 성공 |
|--|----|-------------------|--|-------------------------------|----|