Raider

17팀

조장 안희운 (20171646)
조원 권순민 (20161290)
김승중 (20163096)
이가희 (20163133)
김주연 (20181595)

목차

팀원 소개

권순민

#추적 AI
#라이트(맵 분위기)
#오디오

김승중

#맵 디자인, 오프젝트
#아이템 디자인
#미션게임

안희운

#갓조장
#서버
#멀티 플레이어

김주연

#플레이어
#카메라 움직임
#쉐이더

이가희

게임 UI
디자인
엔딩씬

게임 소개

장르 | 공포 / 퍼즐 / 탈출 / SF / 멀티 / 3인칭 / 3D

플랫폼 | PC

사용 엔진 | Unity

게임 규칙 | 적을 피해가며 주어진 미션을 완수해 탈출한다.

Raider ?

: 침입자

한정판 게임 구매에 성공한 우리, 어느 순간 정신을 차려보니 그 게임 안에 갇혀 있었다.
이곳에서 탈출할 수 있는 유일한 방법은 게임의 시스템(우주선)에 침입해
서버를 다운 시키는 것이라고 하는데…

과연 우리는 이 가상 세계의 우주선에서 탈출할 수 있을까?

기획 의도

멀티 플레이

협동 플레이 가능

최대 4인

술래잡기

AI로 술래 구현

공포스런 분위기

사운드, 라이트 사용

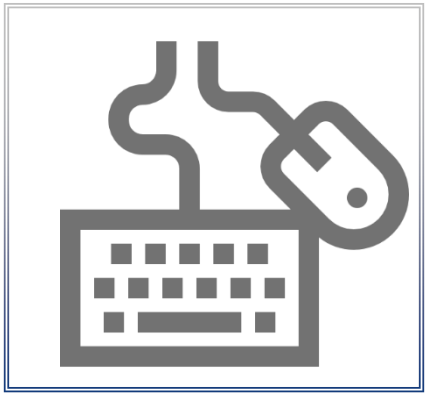
여러가지 미니게임

직관적 플레이 가능

타겟 층

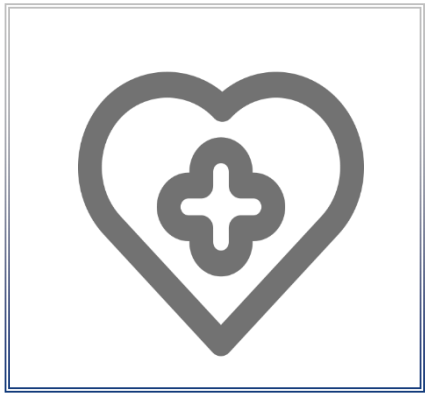
10 ~ 20대

게임 규칙



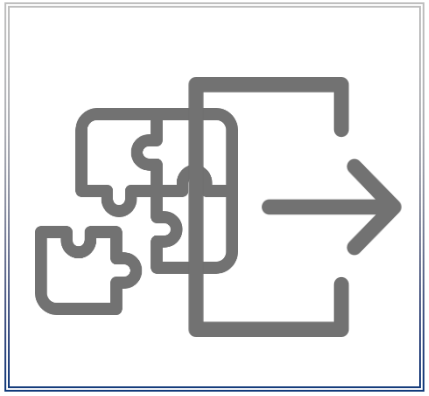
조작

키보드, 마우스 이용



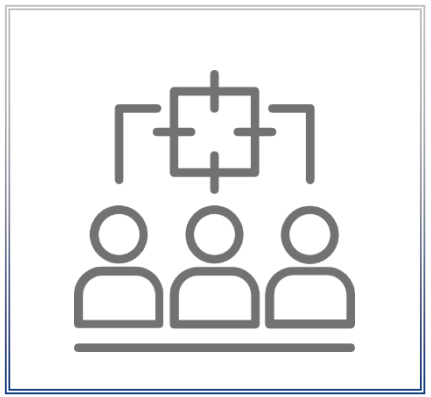
체력

체력은 총 2칸
회복 아이템 사용시 +1
적의 공격을 받을 시 -1
체력이 0이 되면 사망



탈출

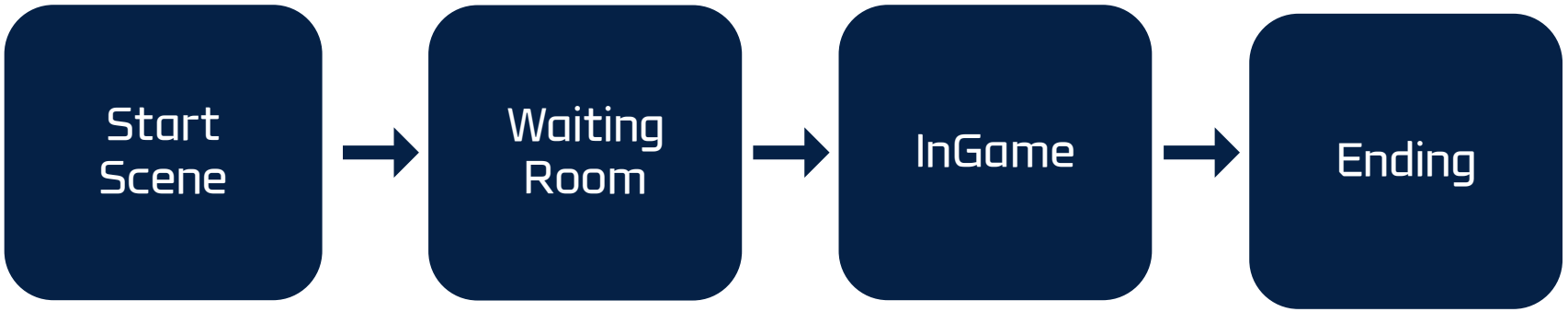
미니게임 형식
모든 미션 성공 시
탈출구 열림
-> 나가면 게임 성공



협동

1~4인용 (3~4인 권장)

씬 구성



Start Scene

- Start : Create Room 또는 Join Room 중 선택 가능
- How To : 게임 방법

Waiting Room

- 대기실 역할
- 팀원이 모두 준비되면 방장이 게임 시작 가능

Waiting Room - 서버 운영 방식

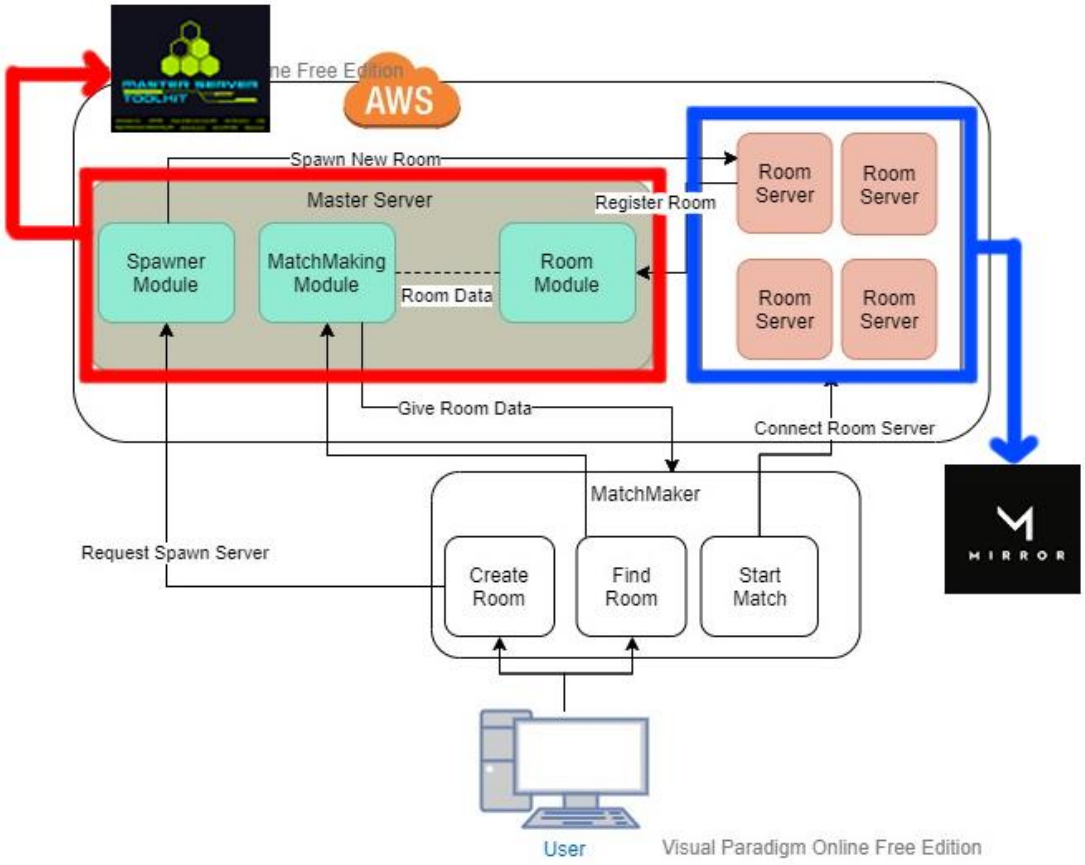
마스터 서버

- 플레이어가 처음으로 접속하는 서버
- 룸서버로 플레이어를 연결해줌

룸/게임 서버

- 대기실 과 인게임 의 네트워크를 담당함
- 실제 게임이 진행되는 서버

Waiting Room - 서버 운영 방식

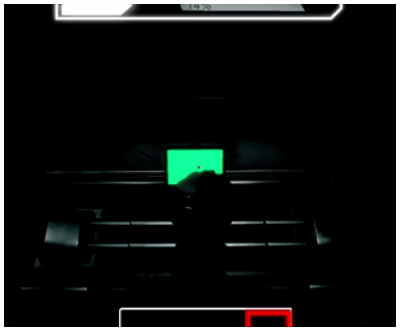


Waiting Room – 타 게임 예시

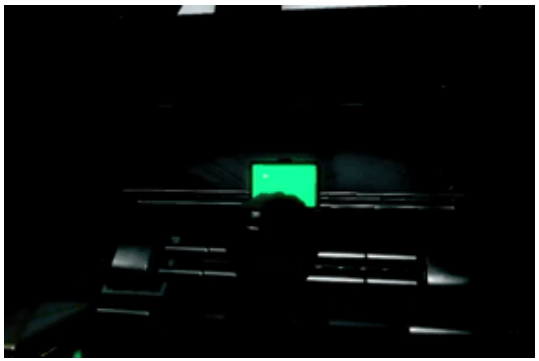
- 실제 다른 온라인게임에서도 비슷하게 사용되는 방법
- 게임 하나당 서버 인스턴스가 하나로 작동됨

InGame – 게임 내 요소

- 미니게임 : 3종류 구현. 마우스를 이용한 직관적 게임. 모두 완료 시 탈출구 활성화
- 방해 요소 : 침입자인 플레이어를 쫓아오는 AI 적 구현
- 아이템 사용 : 체력 회복, 적 움직임 방해 등의 기능 구현



미니게임



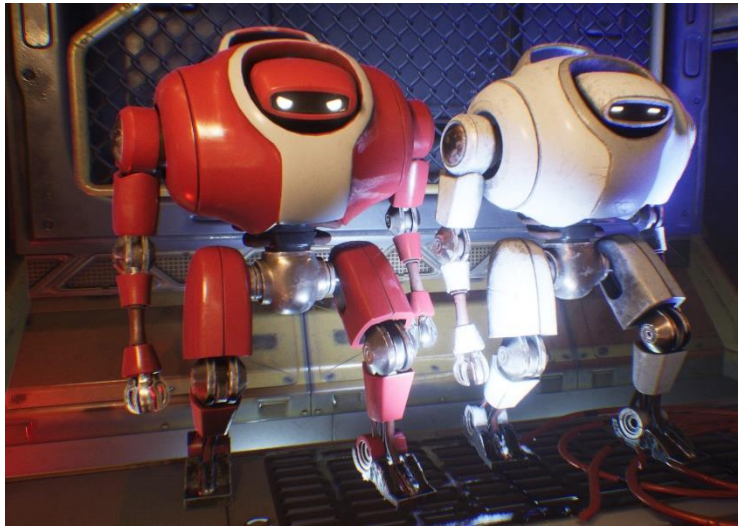
아이템 박스

탈출구

InGame - AI 적

- 시각, 청각 감지 센서
- 플레이어 감지하면서 자동 공격 , 플레이어 감지 안될 때는 순찰
- FSM

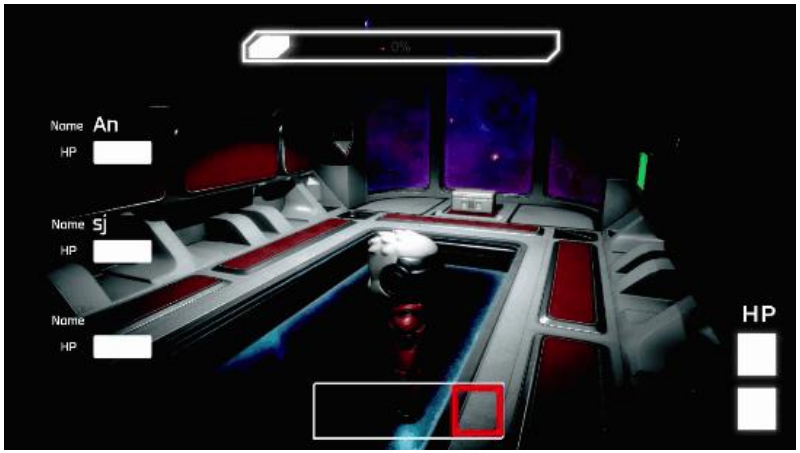
디자인



InGame – 편의 기능

- UI : 미션 진행바, 인벤토리, 메뉴창
- 팀원 간 상호작용 : UI 통한 팀원 상태 확인, 위치 표시
- 카메라 : 카메라 벽 통과 방지 기능 등

Shader 이용한 위치 표시

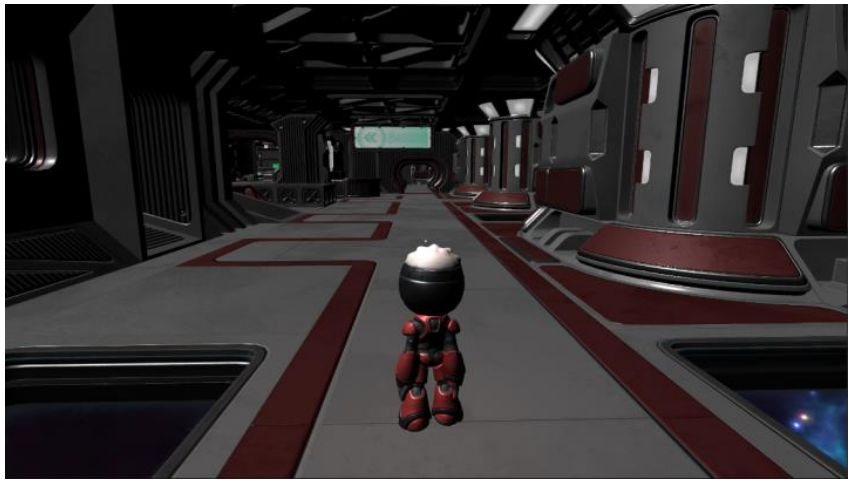


카메라 벽 통과 방지

InGame - 분위기 형성

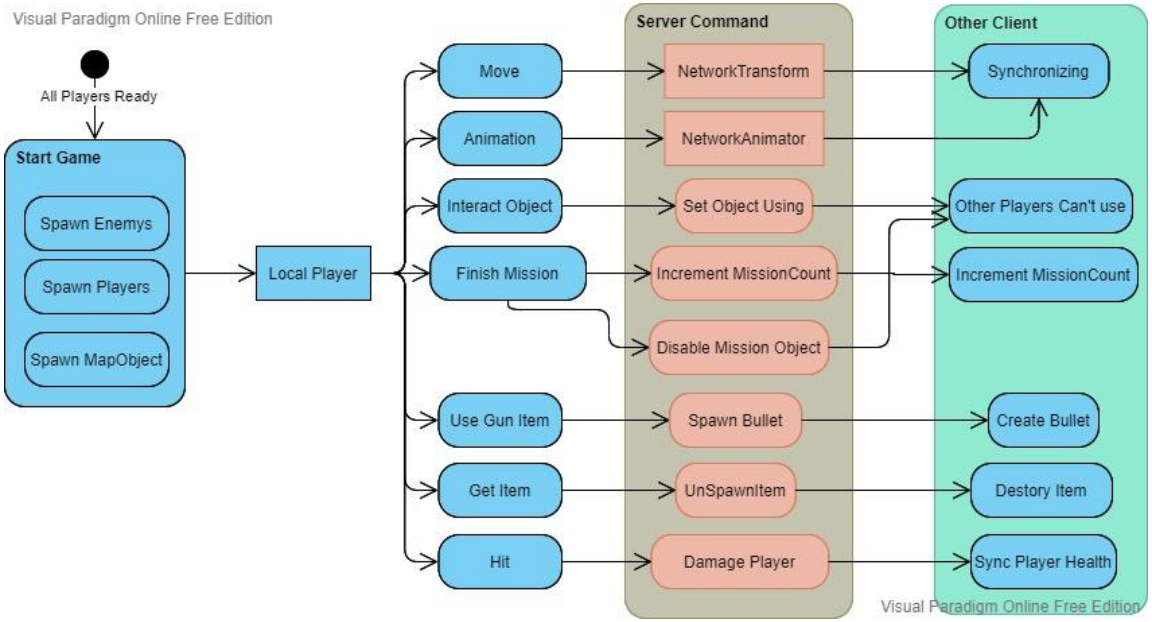
- 라이트, 사운드로 공포 분위기 형성

Before



After

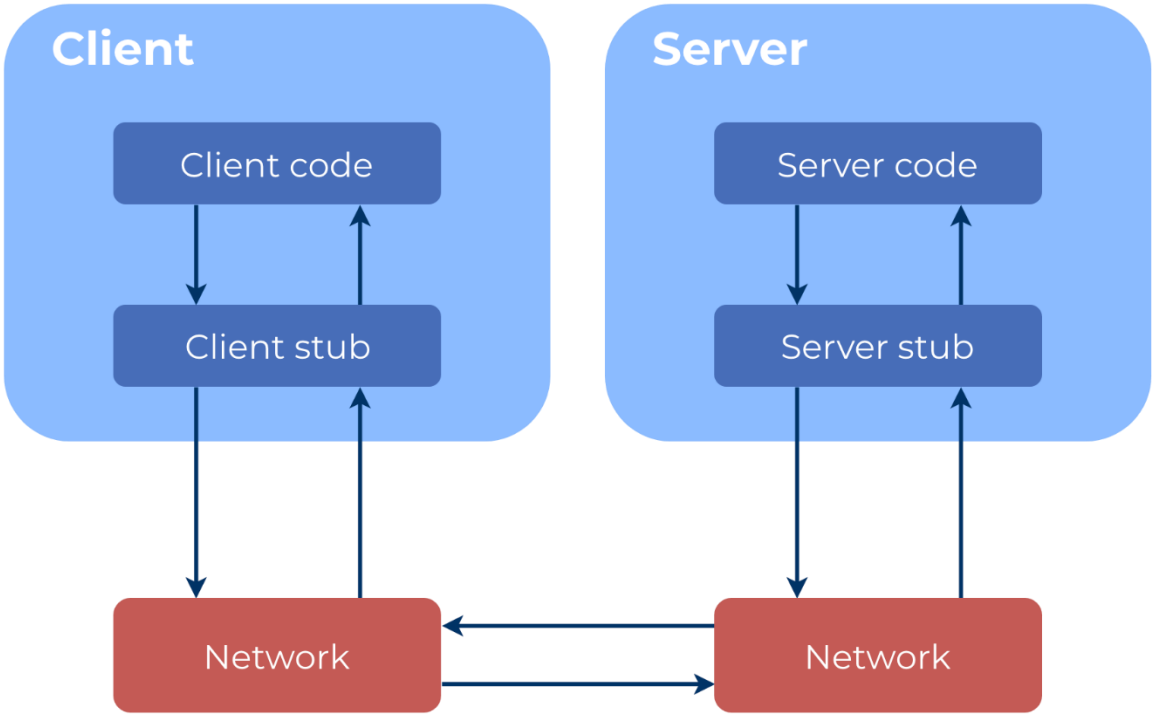
InGame - 네트워크 연동 방식



RPC - Remote Procedure Call(원격 프로시저 호출)

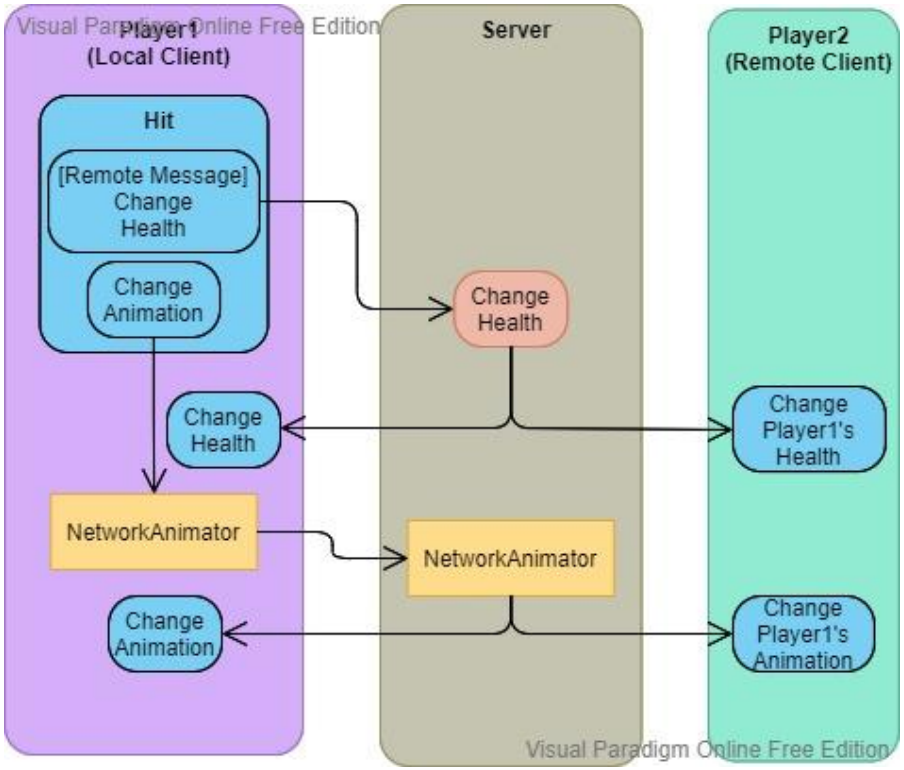
- 플레이어들은 룸 서버를 통해서 게임 내 네트워크 동기화를 진행함
  - 위치, 애니메이션, 물리엔진 - 네트워크 컴포넌트로 동기화
- 플레이어의 이름, 체력, 상태, 오브젝트의 상태 - *RPC를 통한 동기화

InGame – 원격 프로시저 호출



- 네트워크 상 다른 플레이어들이 동일한 코드를 이용할 수 있음.

InGame - 네트워크 연동 방식 예시

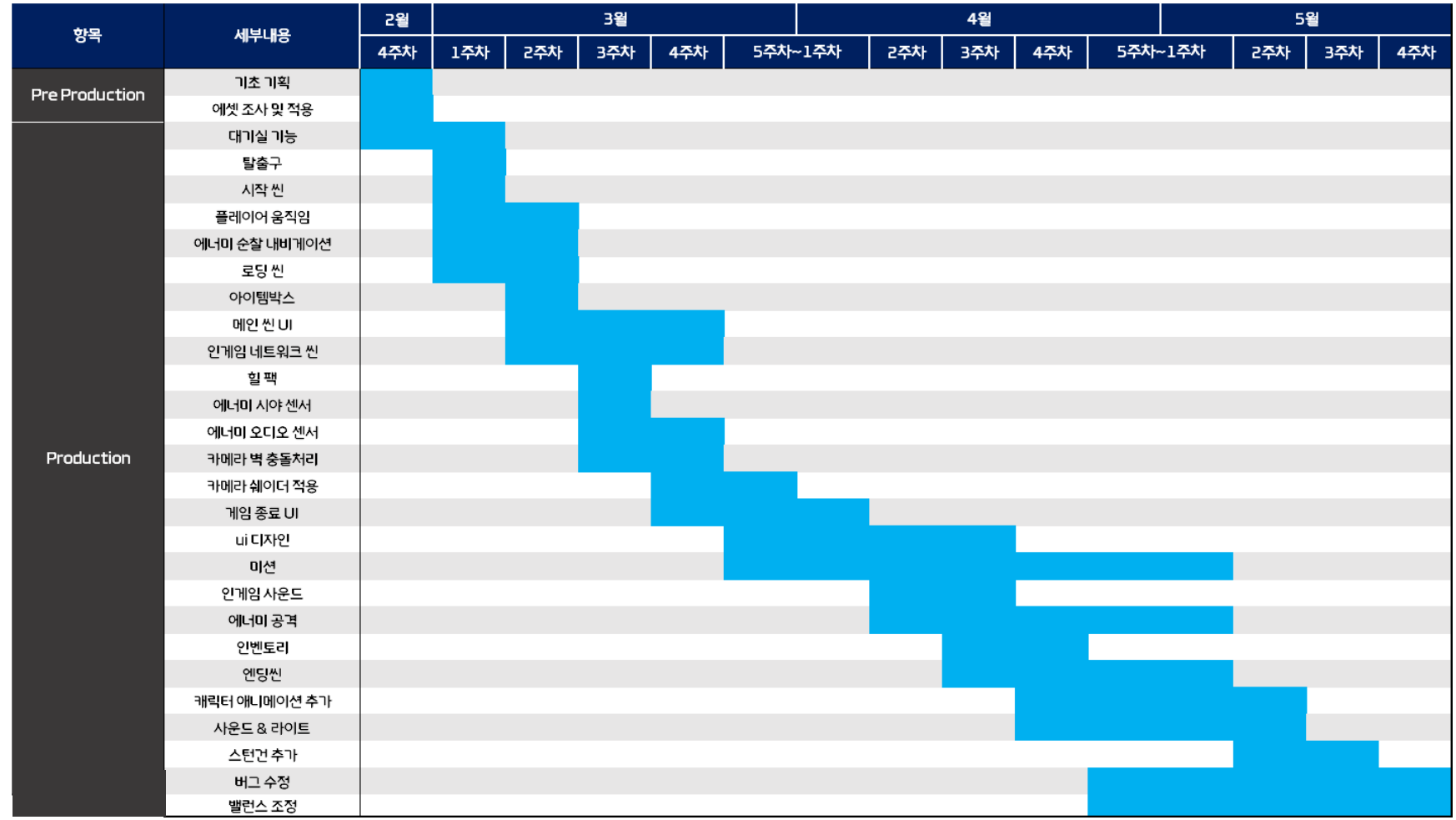


- 플레이어가 공격받을 시 체력이 감소하고 피격 애니메이션이 나옴
  - 체력 감소 - RPC를 통한 동기화
- 피격 애니메이션 - Network Animator라는 컴포넌트로 동기화

Ending Scene

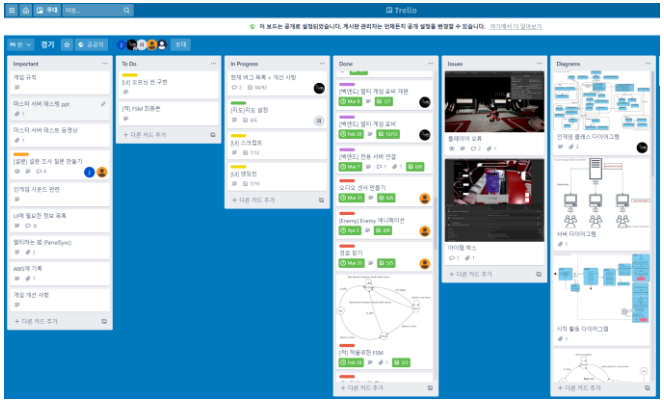
- 게임 클리어 시 나오는 화면
- InGame 내에서 죽었는지 살았는지 여부 표시
- 게임 스타트 화면으로 돌아가거나 게임 종료 가능

개발과정


항목
세부내용
2월
3월
4월
5월
4주차
1주차
2주차
3주차
4주차
5주차~1주차
2주차
3주차
4주차
5주차~1주차
2주차
3주차
4주차
Pre Production
기초 기획
에셋 조사 및 적용
Production
대기실 기능
탈출구
시작 씬
플레이어 움직임
에너미 순찰 내비게이션
로딩 씬
아이템박스
메인 씬 UI
인게임 네트워크 씬
힐 팩
에너미 시야 센서
에너미 오디오 센서
카메라 벽 충돌처리
카메라 쉐이더 적용
게임 종료 UI
ui 디자인
미션
인게임 사운드
에너미 공격
인벤토리
엔딩씬
캐릭터 애니메이션 추가
사운드 & 라이트
스턴건 추가
버그 수정
밸런스 조정

멘토링

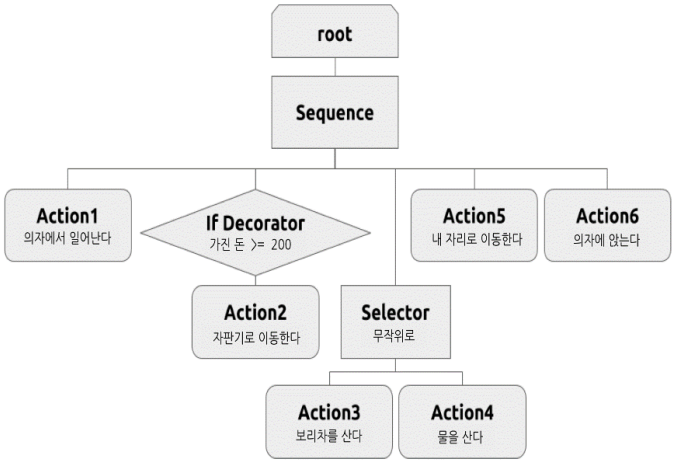
- 버그 해결과 프로젝트 개발 방향 조언에 많은 도움을 주심.
- 멘토님 조언 + 수차례 회의와 QA를 걸쳐 버그는 모두 해결



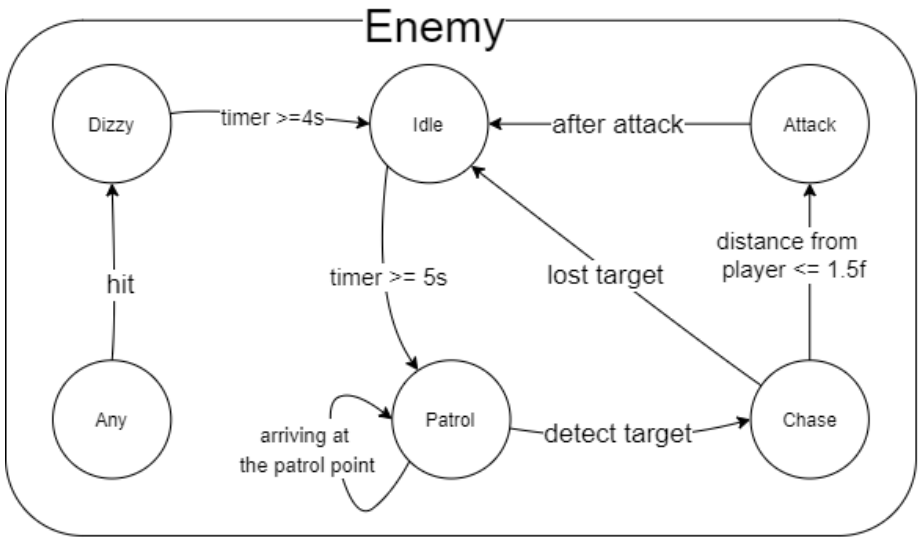
Trello

멘토링 - ex : AI 구현

- Behavior Tree? FSM? → FSM!



Behavior Tree



FSM

- 스테이트 패턴을 이용한 FSM
  - FSM : 유한 상태 기계 (Finite State Machine)

멘토링 – ex : AI 구현

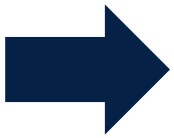
- Switch / Case vs State Pattern

```
public enum State
{
    Idle,
    Patrol,
    Move,
    Attack,
    Dizzy
}

public State state;
```

```
void Update()
{
    if(enemyNet != null && !NetworkServer.active)
    {
        return;
    }

    switch (state)
    {
        case State.Idle:
            IdleState();
            break;
        case State.Patrol:
            PatrolState();
            break;
        case State.Move:
            MoveState();
            break;
        case State.Attack:
            AttackState();
            break;
        case State.Dizzy:
            DizzyState();
            break;
    }
    FindTargets();
}
```



```
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();
}
```

프로젝트 결과 리뷰 - 아쉬운 점

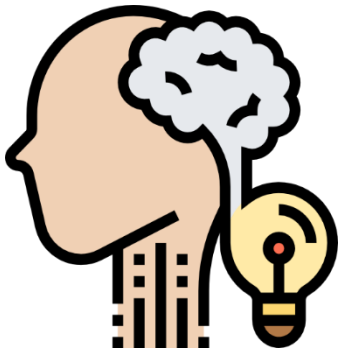
Raider

- ✓ 넣고 싶었으나 하지 못한 요소

  - Matchmaking , 사용자 인증 시스템, UI 애니메이션 효과, 디자인 다양화 등

- ✓ 경험 부족에 의한 시간 분배, 파트별 인원 분배 미흡.
- ✓ 서버 개발이 늦어져 전체적인 진행도 느려짐
- ✓ 게임은 완성은 되었지만 QA 시간조정실패로 인해 배포하진 못하였음.

프로젝트 결과 리뷰 - 만족스러운 점



게임 개발 경험 쌓기

처음 기획 목표 모두 달성

게임 개발 과정 체험

감사합니다