



배움에 두려움이 없는 개발자 유명훈입니다

About Me

안녕하세요! 신입 개발자 유명훈입니다.

2년간 웹 기획자로 일하다가 언리얼 엔진의 가능성을 보고
개발을 시작하게 되었는데요. 디자인부터 기획까지 다양한
도전을 해왔기에, 개발 역시 두려움 없이 도전하게 되었고
학원을 다닌 6개월 동안 5개의 게임 프로젝트를 완료했을 정도로
빠른 습득력과 과제 수행 능력을 가지고 있습니다.

Phone \ 010_5220_9785

Homepage \ <https://mhoo999.github.io>

E-mail \ mhoo999@naver.com

Github \ <https://github.com/mhoo999>

01

Network / Unreal Engine / C++

이스케이프 프롬 새싹

작업 인원 2명 / 기간 37일(24년 2월)

멀티플레이 익스트렉션 루트 슈팅 장르 게임 프로젝트로
다중 인원이 특정 맵에 진입하여 타겟 제거, 아이템 획득 등의
목적을 수행하고 무사히 탈출하는 것이 목적인 게임입니다.

멀티플레이와 사용자 조작 및 예정, AI의 역할 수행을
개발 중점으로 목표하고 진행한 프로젝트입니다.
AI와 쿼스트, 전반적인 기획을 담당했습니다.

깃 링크 : <https://github.com/mhoo999/SesacProject5>

소개 영상 : <https://www.youtube.com/watch?v=5dy0EVVp6bU>

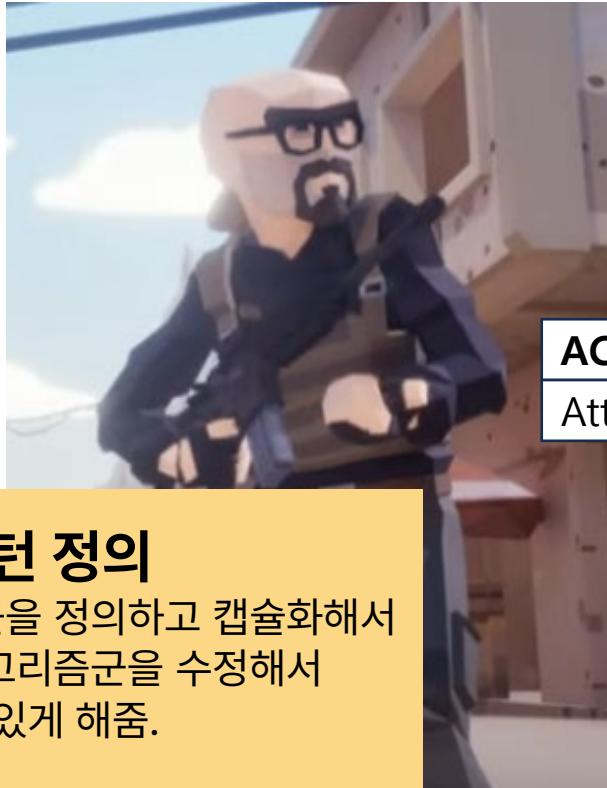


1-1. AI FSM 설계

요구사항 : 수정·확장에 용이한 FSM을 구현

구현내용 : 개발 관련 특강에서 전략 패턴을 사용하여 FSM을 구성할 수 있다는 힌트를 듣고, 전략 패턴을 사용하여 구현하려 했으나, 개발 목적에 더 부합하는 상태패턴으로 구현 방향을 변경(다음 페이지 계속)

*공격 시 무기에 따라 다른 행동을 할 수 있도록 전략 패턴은 추후 전투 기능에 적용하도록 기록



AChaseComp
Attack()

근접 무기

원거리 무기



전략 패턴 정의

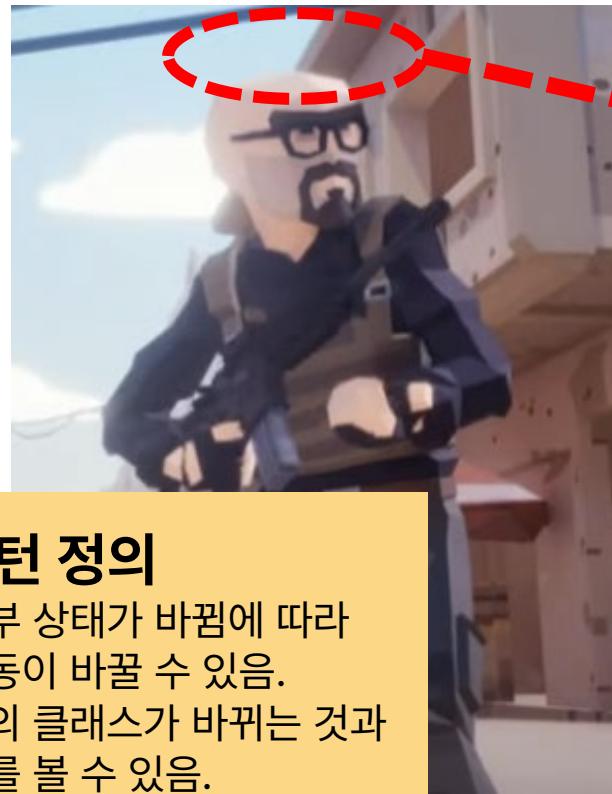
알고리즘군을 정의하고 캡슐화해서 각각의 알고리즘군을 수정해서 사용할 수 있게 해줌.



1-1. AI FSM 설계

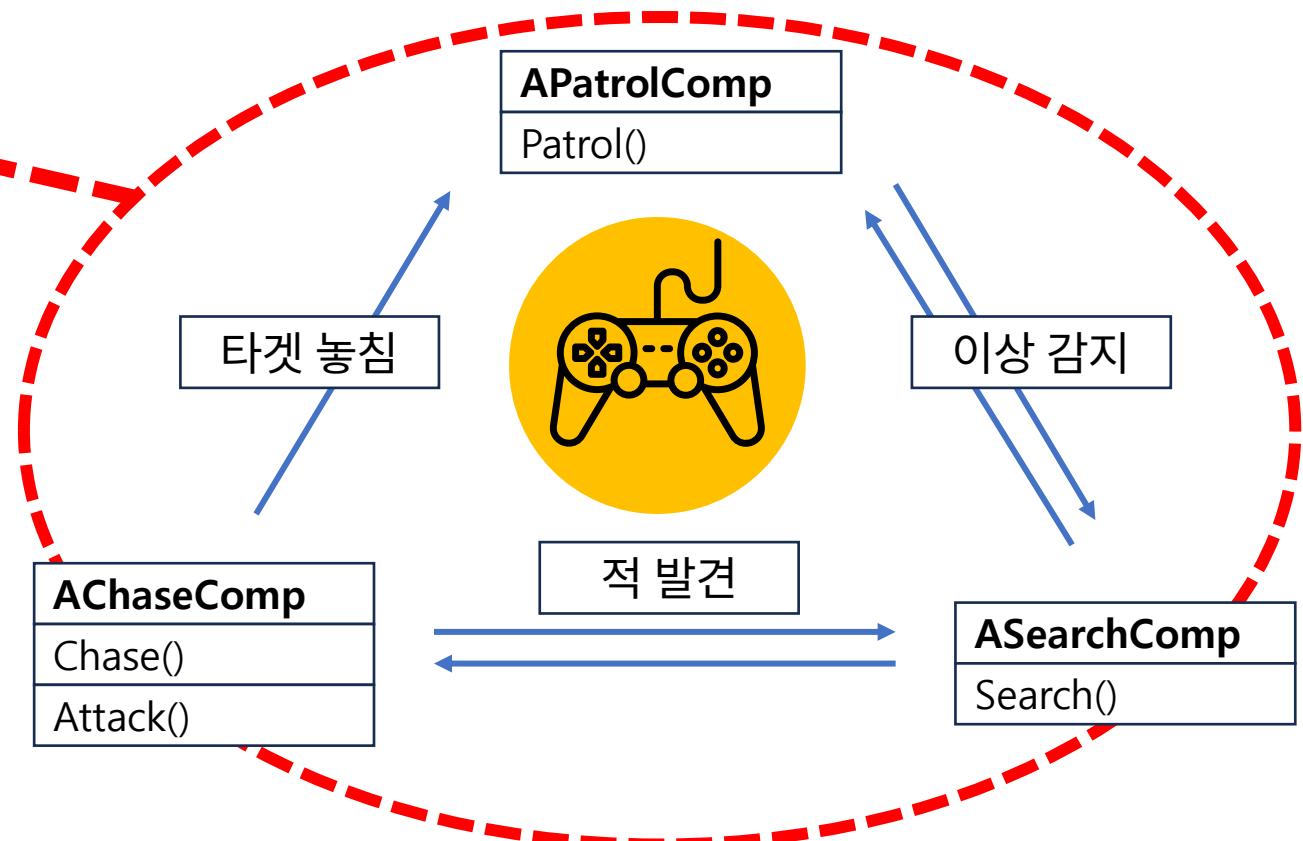
요구사항 : 수정·확장에 용이한 FSM을 구현

구현내용 : 디자인 패턴 중 하나인 상태 패턴을 이용하여 구현. 클래스가 비대해지는 것을 막을 수 있었고, 클래스 별 종속성이 약해져, 각 행동 별 알고리즘에 집중하여 비교적 깔끔한 코드를 구현할 수 있었음



상태 패턴 정의

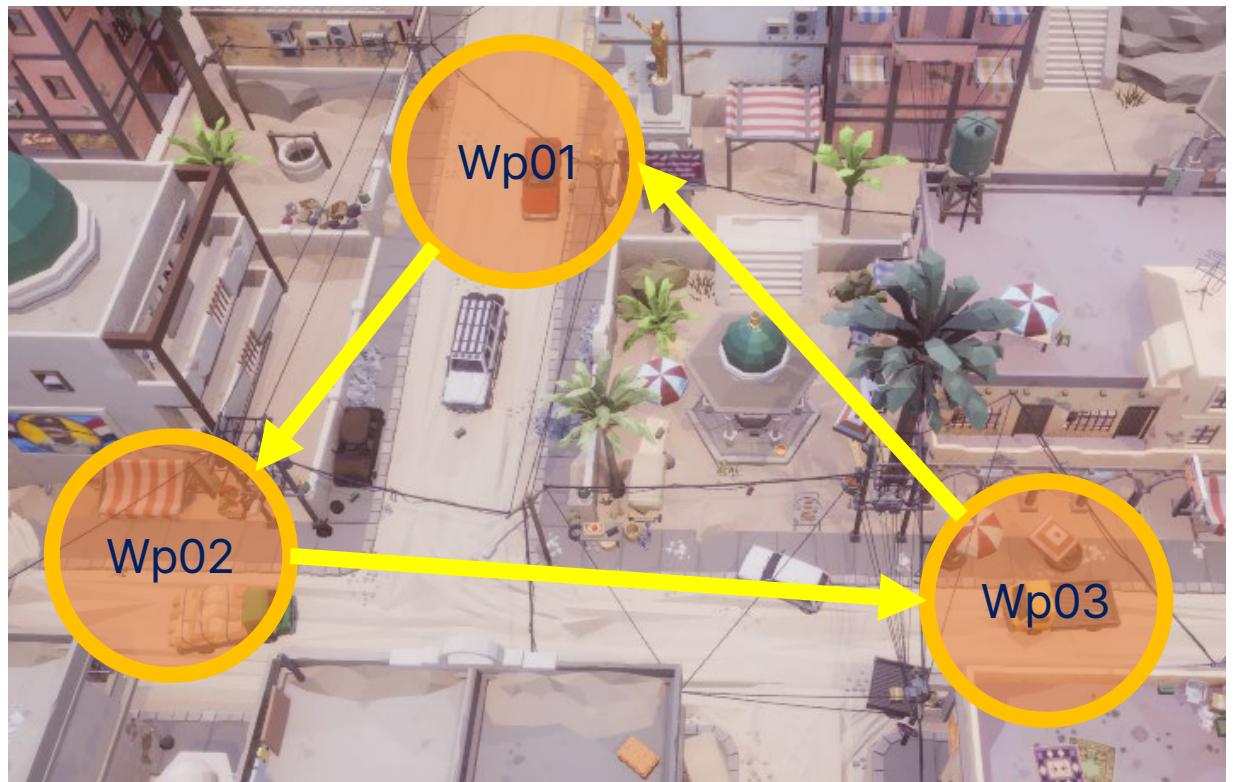
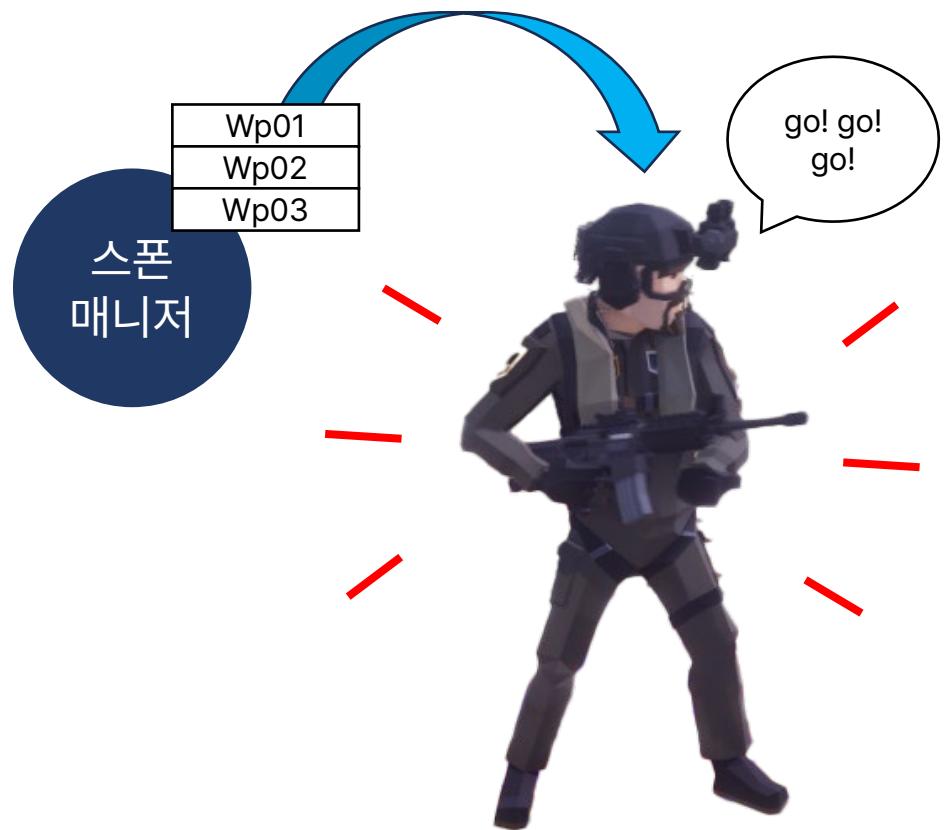
객체의 내부 상태가 바뀜에 따라
객체의 행동이 바꿀 수 있음.
마치 객체의 클래스가 바뀌는 것과
같은 결과를 볼 수 있음.



1-2. AI 패트롤 기능 설계

요구사항 : '특정 구역을 순찰하는 ai'를 생성하는 기능을 구현

구현내용 : 스폰 매니저가 특정 구역에 대한 정보를 가지고 있다가 생성 시점에 순찰 구역으로 ai에게 전달하도록 구현

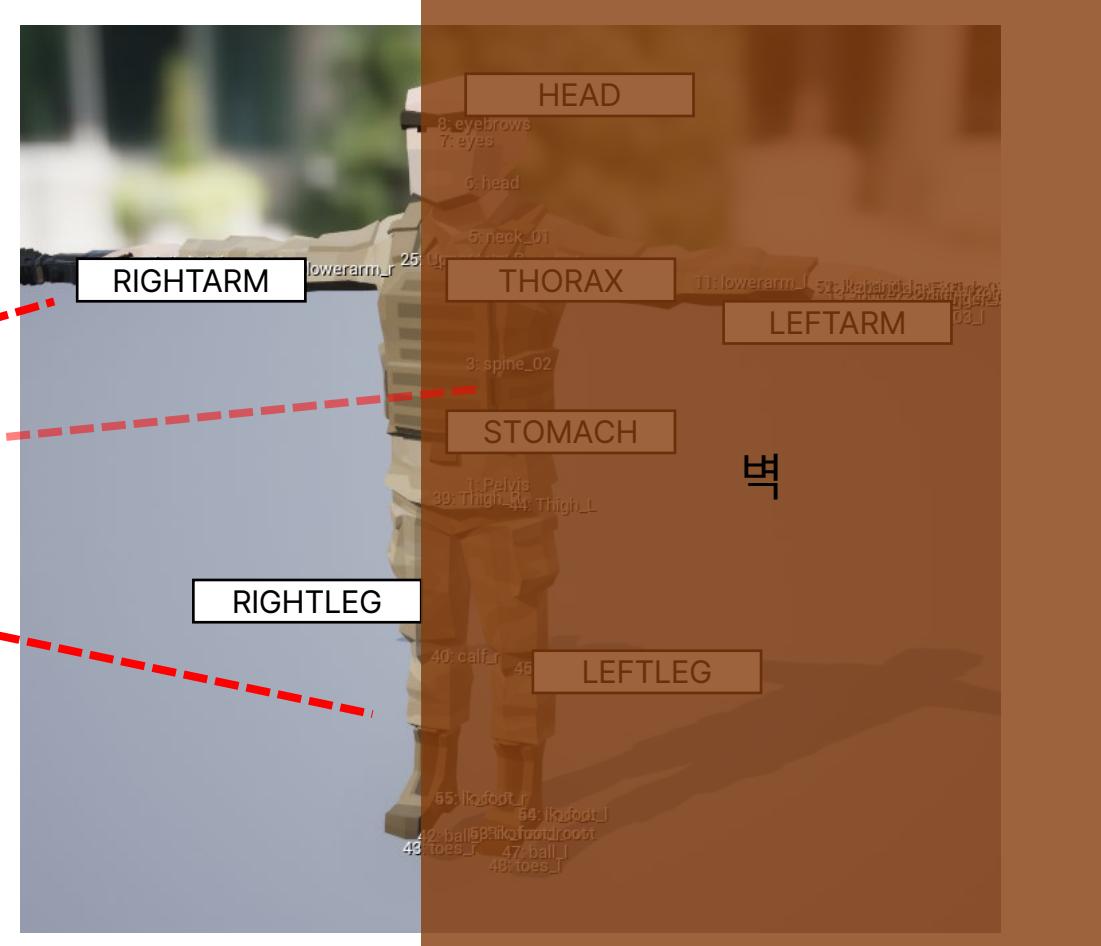


1-3. AI 부위 별 타격 기능 설계

요구사항 : 캐릭터는 각 신체 부위 별로 타격이 가능해야 함

구현내용 : 탐지한 대상의 스켈레탈이 가진 소켓을 검색하여 현재 조준 가능하고, 우선순위가 높은 부위를 우선 사격

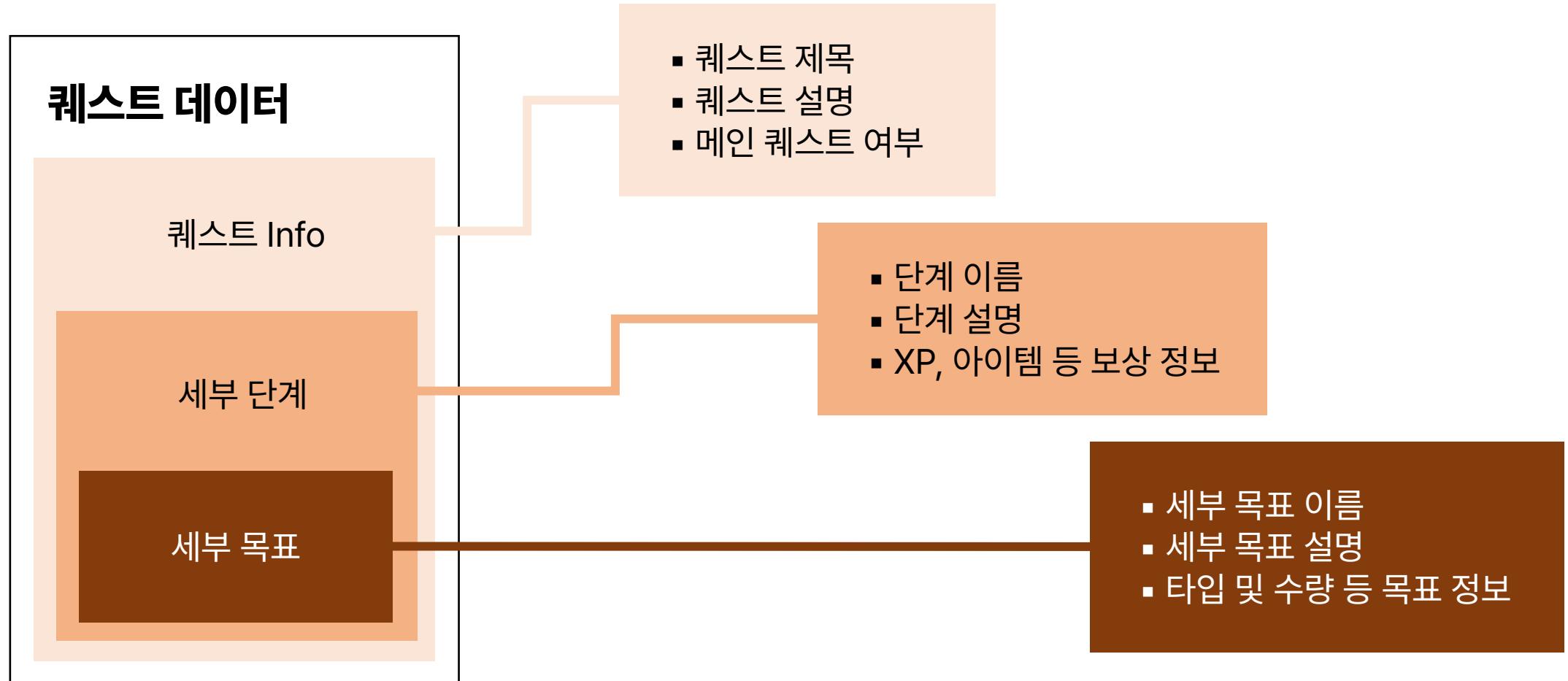
```
if (SocketName.Equals(Other: "spine_03", ESearchCase::IgnoreCase) && checkResult.GetActor() == targetActor)
{
    TargetPart THORAX { .name: & "THORAX", .priority: 1, .isVisible: true, .partLoc: SocketLocation};
    if (THORAX.priority < currentTarget.priority)
    {
        currentTarget = THORAX;
    }
}
```



1-4. 퀘스트 구조 설계

요구사항 : 확장에 용이한 퀘스트 구조 설계

구현내용 : 다양한 퀘스트 진행이 가능하도록 단계별 설계



02

Network / Unreal Engine / C++

좀비 머스트 다이

작업 인원 3명 / 기간 23일(24년 1월)

스팀에서 퍼블리싱하는 액션 TPS 디펜스 게임
오크 머스트 다이를 레퍼런스로 진행한 프로젝트

좀비를 제압하여 얻은 전리품으로
트랩 설치와 무기 업그레이드를 통해 거점을
안전하게 수비하는 것이 게임의 목표입니다.
플레이어 기능 및 조작, 네트워크 작업을 담당했습니다.



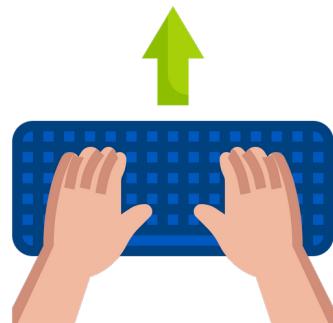
깃 링크 : <https://github.com/mhoo999/SesacProject4>

영상 링크 : https://youtu.be/uZjpwt6pkco?si=bKS7JNoPLJ_pR4ib

2-1. 옵저버 패턴을 활용한 입력 시스템 설계

요구사항 : 공동 컴포넌트 작업이 가능하도록 입력 시스템을 구현

구현내용 : 종속성을 끊고, 각 객체를 독립적으로 활용할 수 있도록 옵저버 패턴을 적용하여 구현



APlayerBase
SetupInput()

AMoveComp
SetupInput()
IA_Move

AFireComp
SetupInput()
IA_Fire



받는 입력을 갖는
Delegate 선언

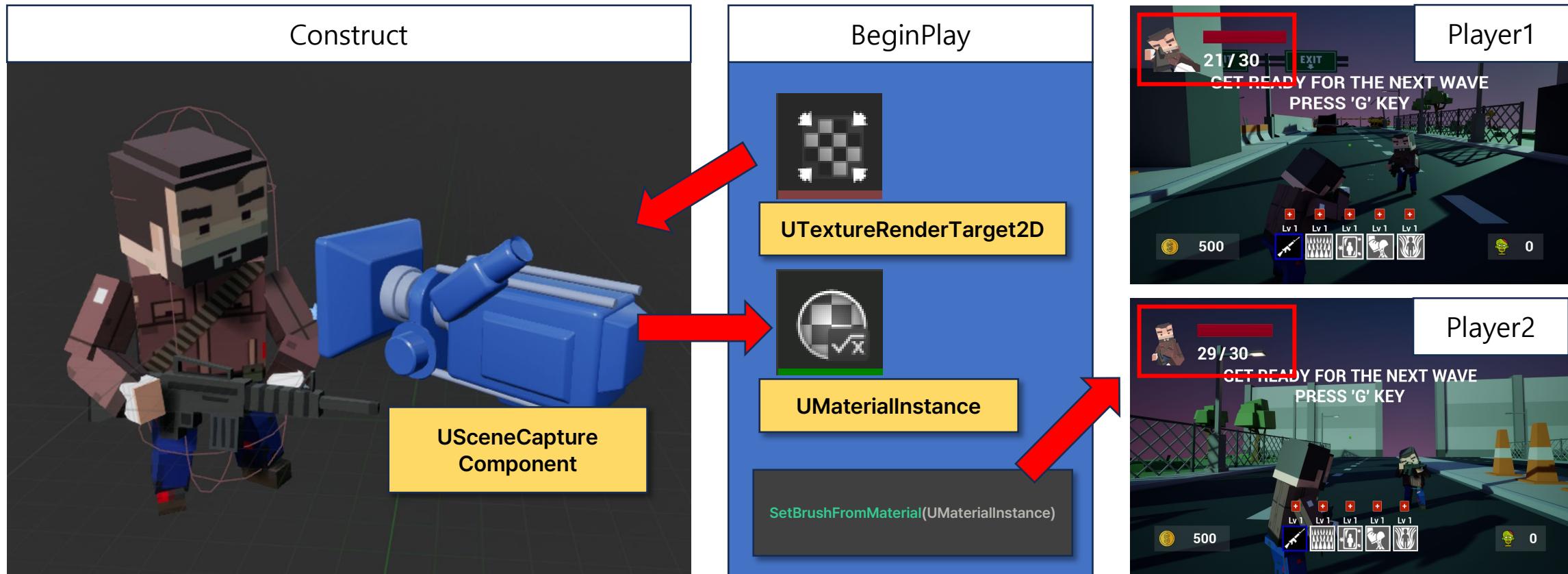
입력이 발생할 때마다
Broadcast

각 객체는 Bind 된 함수를 호출

2-2. 컴포넌트 동적 생성 구현

요구사항 : 멀티 환경에서 플레이어마다 본인의 캐릭터 초상화 출력

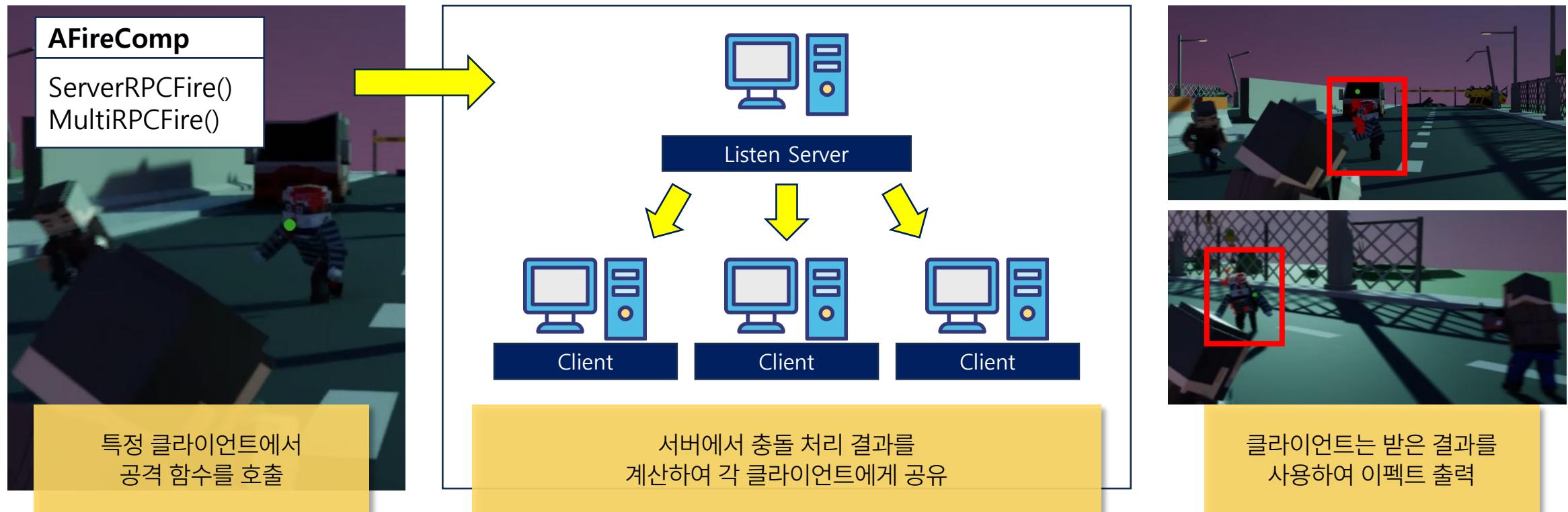
구현내용 : 플레이어 캐릭터 생성 시, 캐릭터 초상화 캡처를 런타임에 생성하여 화면에 출력



2-3. 네트워크 작업

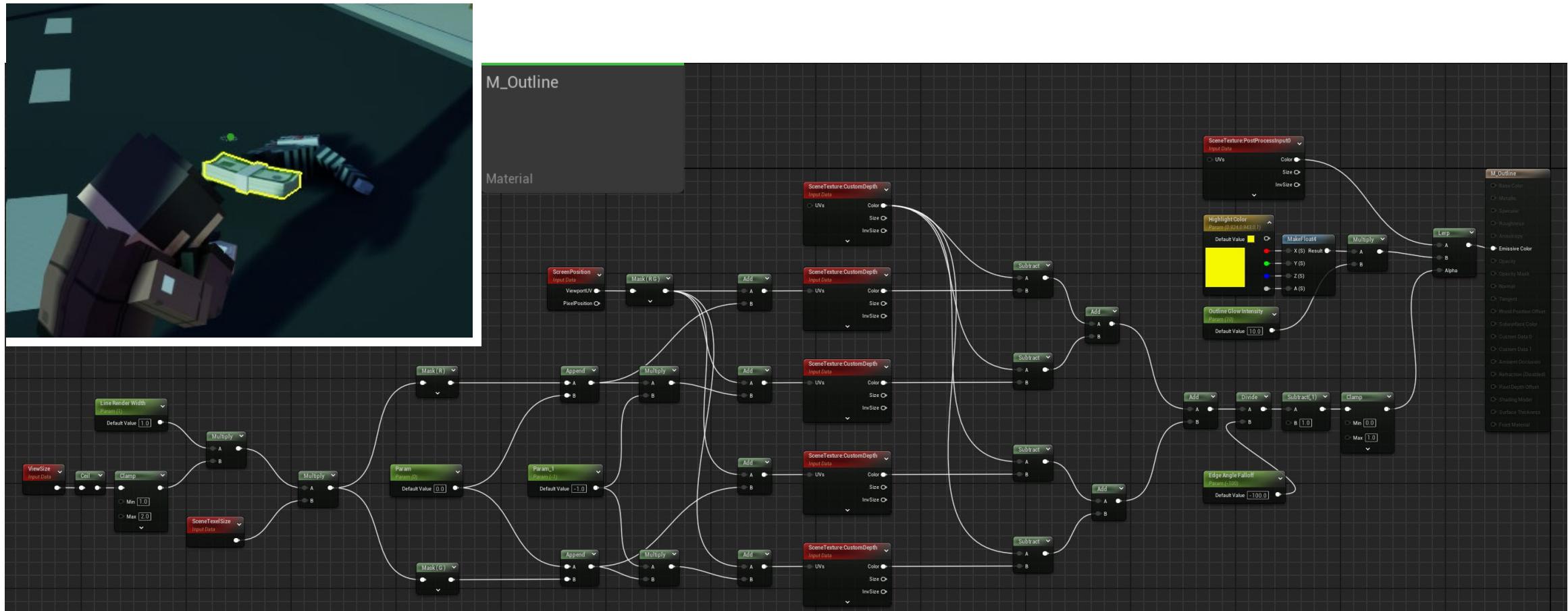
요구사항 : 멀티플레이에서 동기화된 작업을 수행

구현내용 : 서버에서 로직을 수행한 후, 각 클라이언트에게 전달하여 처리



2-4. 기획적인 요소

아이템 강조 : 생성된 아이템이 눈에 띄도록 포스트프로세스 머티리얼로 아웃라인을 만들고, 회전 로직을 추가하여 위치를 강조함



2-4. 기획적인 요소

요구사항 : 게임 내 어색한 부분 처리



자연스러운 애니메이션 처리

발포 후, 바로 Idle 상태로 돌아가는 것이 어색하여 일정 시간 동안 조준 상태를 유지한 뒤, Idle 상태로 돌아가는 디테일을 구현



탄착 지점에 따른 효과 구분

공격 시 감지되는 엑터가 일반 엑터일 경우 탄착 모양의 데칼을 생성하고, 적일 경우 피가 튀는 이펙트를 생성하도록 구현
*데칼은 일러스트레이터를 사용하여 직접 제작함

03

VR / Unreal Engine / C++

XR 칼리버

작업 인원 3명 / 기간 17일(23년 12월)

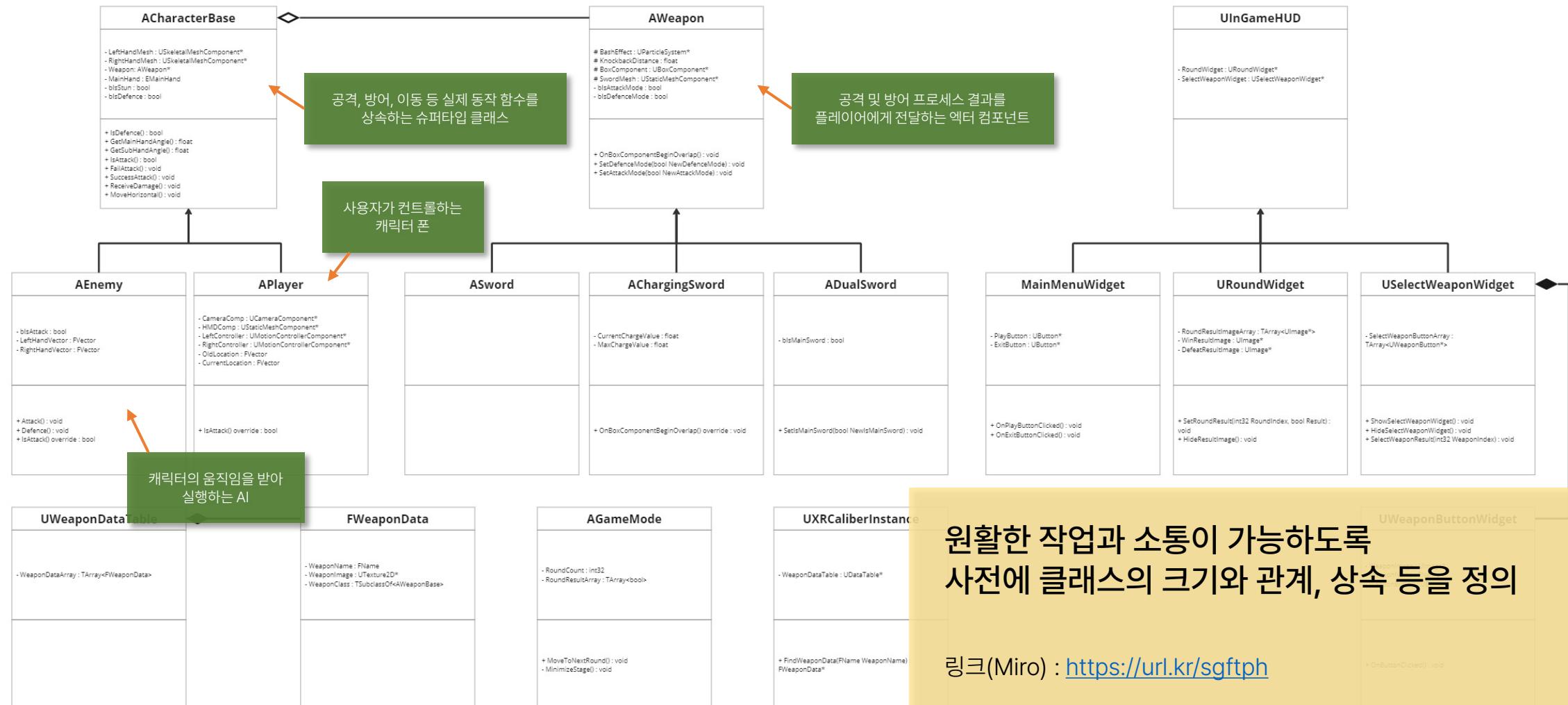
닌텐도 스위치의 스포츠 검술 게임을
레퍼런스 삼아 진행한 VR게임 프로젝트

컨트롤러를 휘둘러 공격을 수행하고
상대방을 원 밖으로 밀어내면 승리하는 게임입니다.
플레이어 캐릭터의 움직임과 모션 컨트롤러를
사용한 공격, 방어 등의 기능을 담당했습니다.

깃 링크 : <https://github.com/mhoo999/SesacProject3>
영상 링크 : <https://youtu.be/qxyVVmiFSUU>



UML 작성



3-1. 컨트롤러 모션에 따른 공격 판단 구현

요구사항 : 컨트롤러를 일정 크기로 휘둘렀을 때 시스템 내에서 해당 움직임을 공격으로 판단해야 함

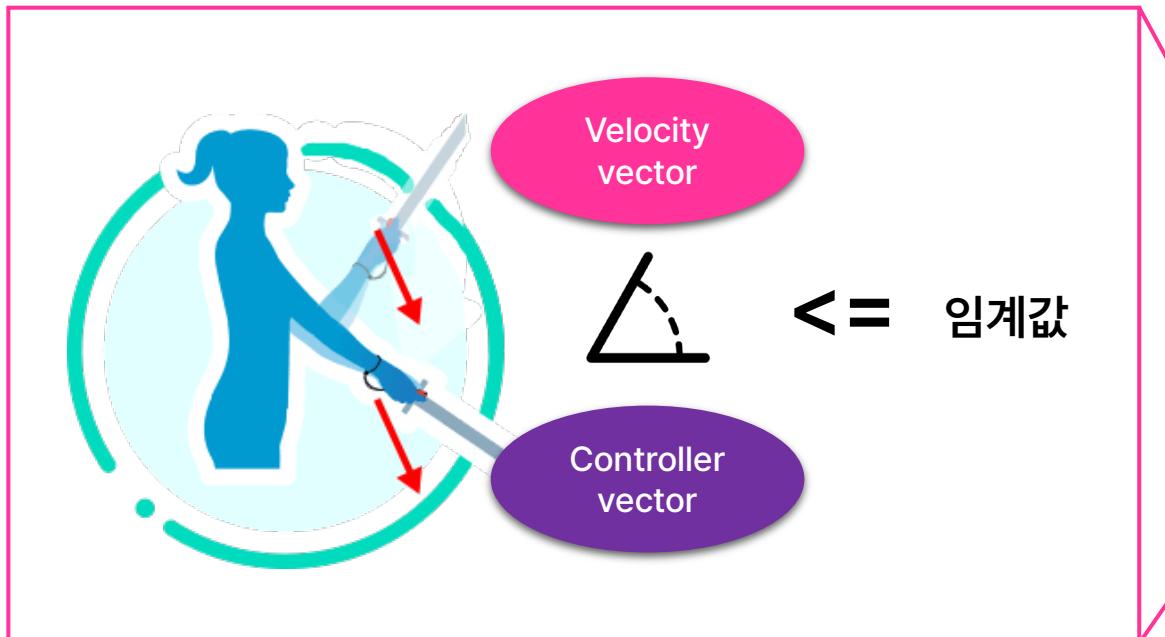
구현내용 : 컨트롤러의 과거 위치와 현재 위치를 계산한 값이 특정 임곗값을 넘어갈 때 공격으로 판단

... 되도록 구현했지만 원치 않는 상황에서도 공격이 발생



3-1. 컨트롤러 모션에 따른 공격 판단 구현

문제해결 : 컨트롤러의 정면 벡터와 휘두르는 방향 벡터의 내각을 구해, 해당 각도가 미리 설정한 값의 오차범위 내인 경우 공격으로 판단하도록 코드를 수정하여 해결



```
if (CurrentTime >= AttackTimer)
{
    CurrentTime = 0;

    OldLocation = CurrentLocation;
    CurrentLocation = RightController->GetComponentLocation();

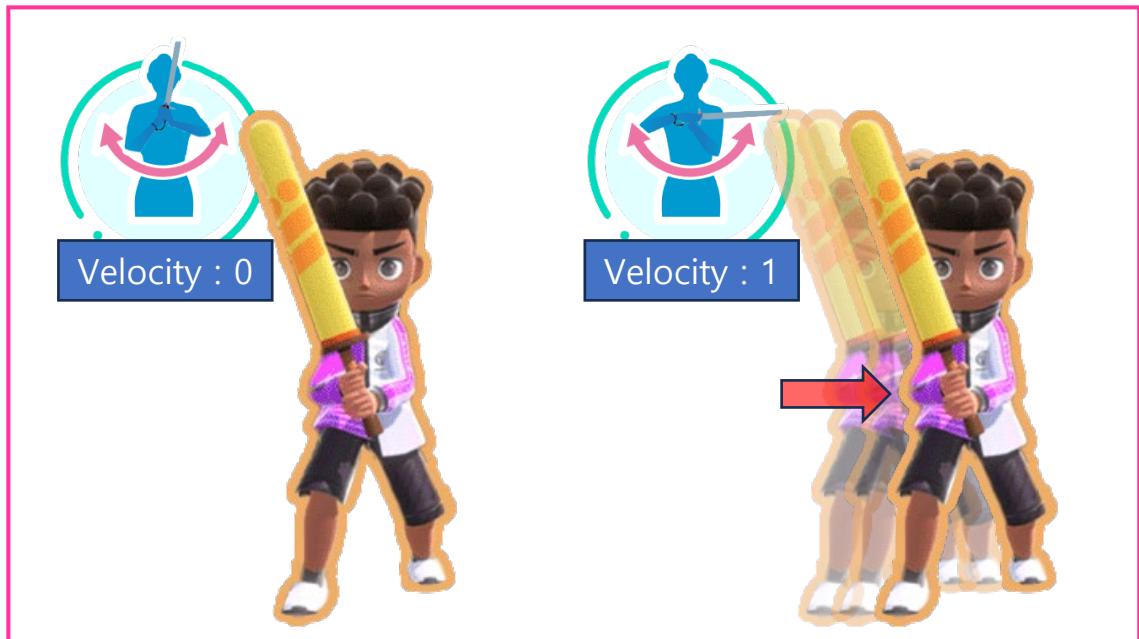
    // 속력 = 이동 거리 / 걸린 시간
    FVector Velocity = (OldLocation - CurrentLocation) / DeltaTime;
    float MaxValue = RightHandMesh->GetRightVector().Dot(Velocity);

    // MaxValue 값이 100 % 이상, bIsAttack이 false일 경우, StartAttack()을 실행
    if (MaxValue <= AttackSuccessValue && bIsAttack == false && bIsDefence == false)
    {
        StartAttack();
    }
    else if (MaxValue > AttackSuccessValue && bIsAttack == true)
    {
        StopAttack();
    }
}
```

3-2. 검의 방향, 각도에 따른 이동 기능 구현

요구사항 : 방어 상태일 경우, 검의 방향에 따라 횡 이동하고, 검의 각도가 수평에 가까울수록 속력이 커져야 함

구현내용 : 컨트롤러의 Roll 값을 0과 1 사이의 값으로 정규화하고, 기본 속도에 곱해 크기에 따른 이동 속도 차이를 만들었음



```
void ACharacterBase::MoveHorizontal(float SwordAngle)
{
    if (Target == nullptr) return;

    float MaxValue = 180.0f;
    float MinValue = 0.0f;

    // 검의 벡터 값을 받아서 0-1값으로 정규화
    float NewAngle = (SwordAngle - MinValue) / (MaxValue - MinValue);
    // 정규화된 값으로 새로운 Distance 도출
    float NewDistance = MoveHorizontalDistance * NewAngle;

    // 타겟 방향을 단위 벡터로 정규화
    FVector ForwardVector = (Target->GetActorLocation() - GetActorLocation()).GetSafeNormal();
    AddActorWorldOffset(DeltaLocation: FRotator(0, 0, 0).RotateVector(ForwardVector) * NewDistance);
}
```

A screenshot from the game Elden Ring. It shows a character in dark armor standing on a rocky, coastal area. In the background, there's a massive, ancient-looking tree with many roots hanging down, and a range of snow-capped mountains under a dark sky.

04

Unreal Engine / Blueprint

Elden Ring

작업 인원 3명 / 기간 42일(23년 11월)

프롬소프트웨어의 액션RPG 엘든링을
레퍼런스 삼아 진행한 게임 프로젝트

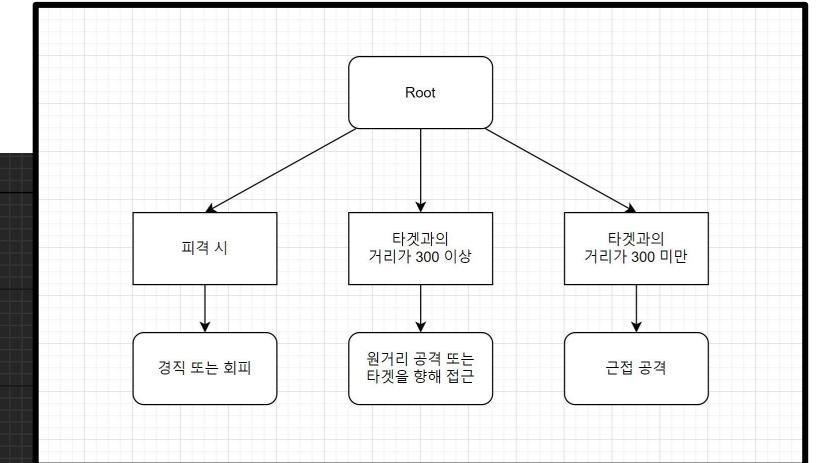
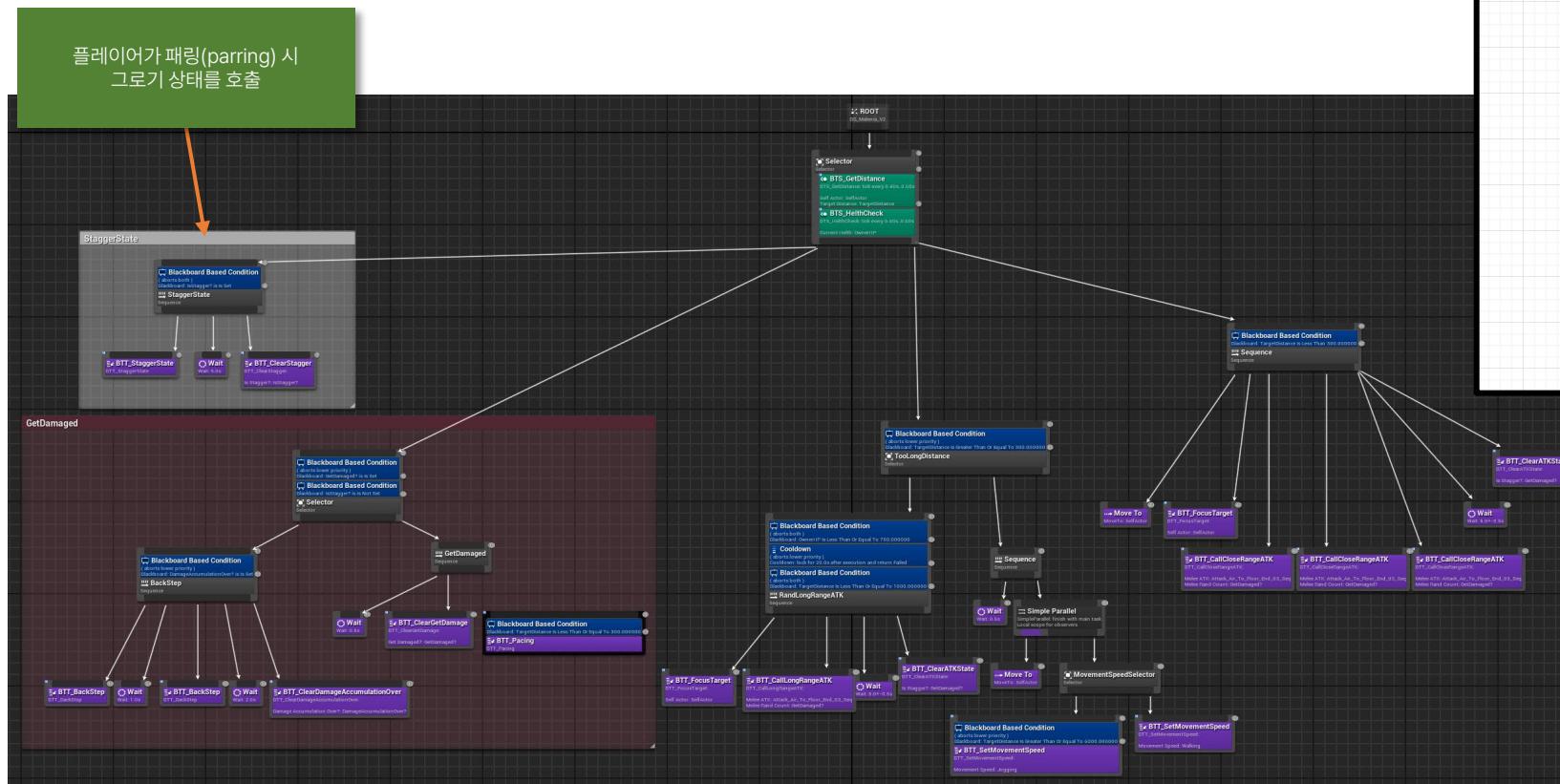
사소한 움직임에도 큰 피해를 입을 수 있어
신중한 조작을 요하는 전투가 특징인 게임입니다
플레이어가 상대해야 하는 두 개의 보스 중
최종 보스의 AI를 담당했습니다

영상 링크 : <https://youtu.be/vwTZQs57loo>

4-1. Enemy AI 구현

요구사항 : 엘든링의 보스 중 하나인 말레니아의 ai를 제작

구현내용 : 대상의 패턴을 분석하여, 언리얼엔진에서 제공하는 Behavior Tree를 사용하여 구현





05

Unreal Engine / Blueprint

OUT LOST

작업 인원 2명 / 기간 34일(23년 10월)

1인칭 시점의 생존 호러 게임 아웃라스트를
레퍼런스 삼아 진행한 첫 번째 게임 프로젝트

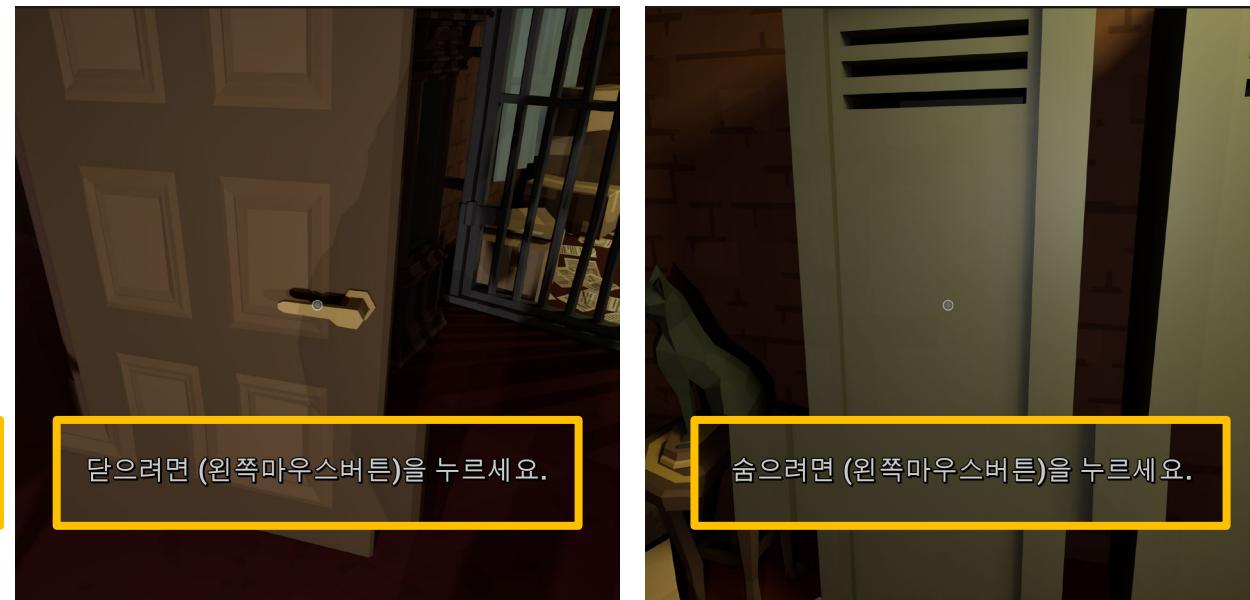
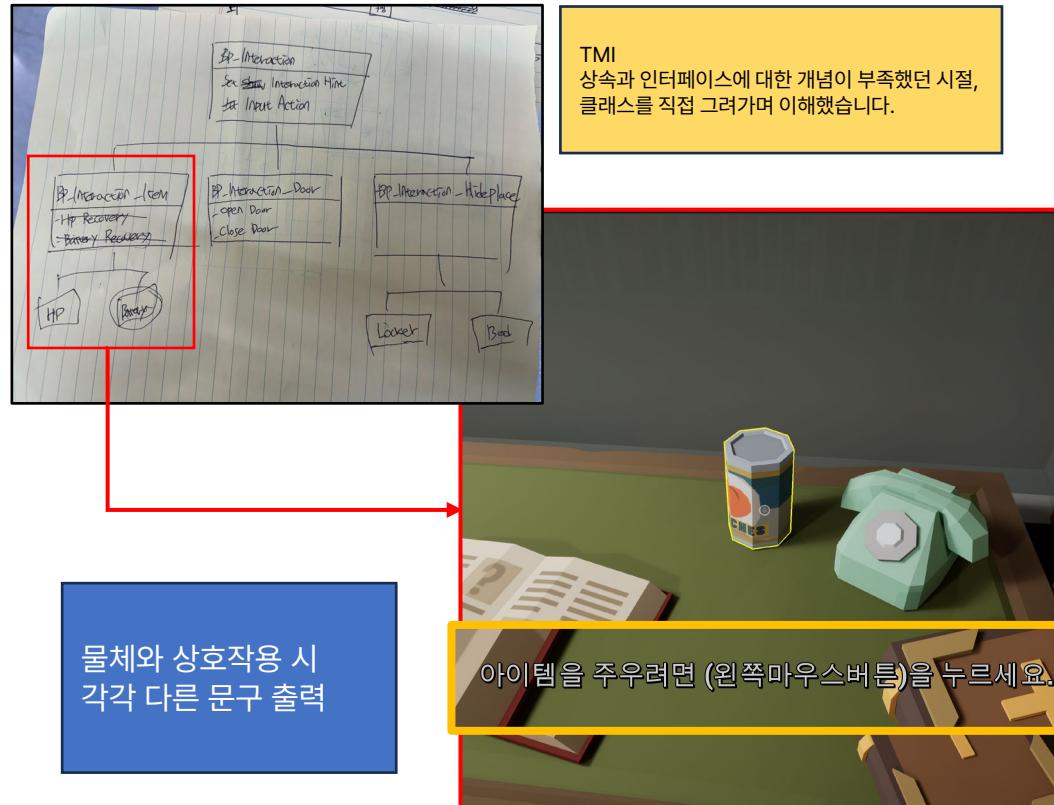
플레이어는 밀폐된 공간 안에서 자신을 공격하는
Ai를 피해 도망치며 숨겨진 퍼즐을 풀고 탈출하는
내용의 게임입니다. 게임 내 일정 부분을 모작 진행하였고
플레이어 조작 기능과 맵 제작을 담당했습니다

영상 링크 : https://youtu.be/5A8AQsEz_eE

5-1. 상호작용

요구사항 : 플레이어가 상호작용 엑터를 바라볼 때, 각각의 엑터별 서로 다른 UI 출력과 효과 호출

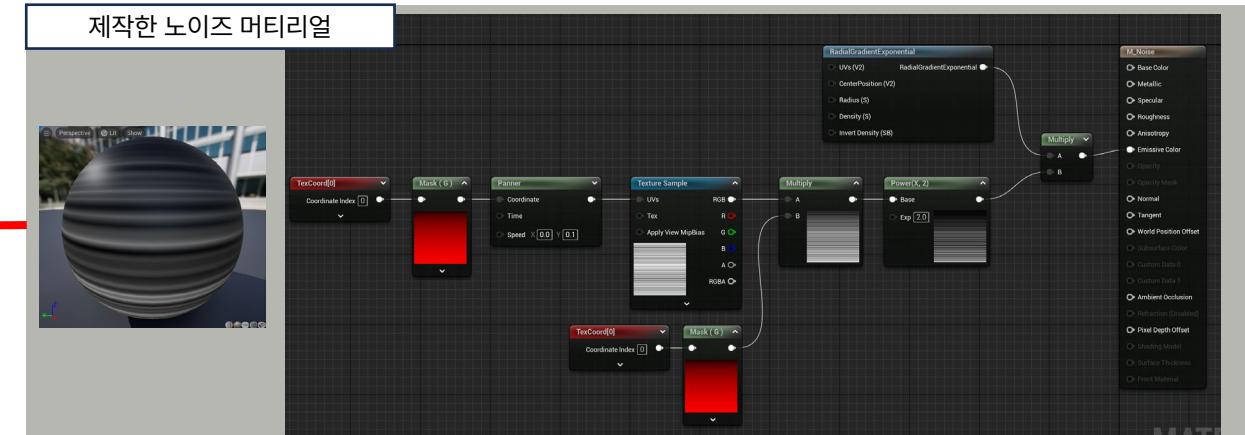
구현내용 : 상속 관계의 함수를 만들어서 플레이어가 함수를 호출할 때, 서브 타입 클래스가 상속받은 함수를 호출



5-2. 캠코더 머티리얼

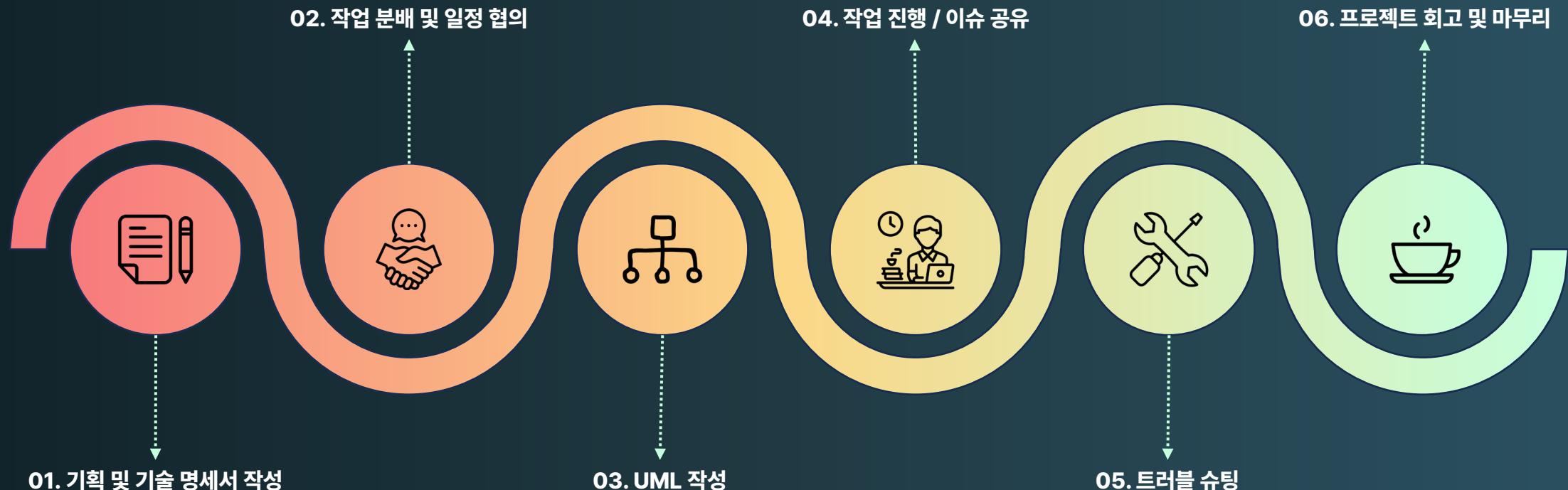
요구사항 : 레퍼런스 게임 내 캠코더 뷰를 제작

구현내용 : 나이트 비전 머티리얼과 노이즈 머티리얼을 만든 후, 플레이어 카메라의 포스트 프로세스 기능으로 사용



PROJECT PROCESS

프로젝트는 아래와 같은 순서로 진행했습니다.



읽어 주셔서 감사합니다!

Steadily Growing!

오른쪽 사진은 저의 공부를 기록하는 노션 페이지입니다.

기획자 시절부터 최근 후에 공부하는 습관을 유지하고 있습니다.

아직 부족하지만, 어제보다 더 나은 제가 될 수 있도록

꾸준히 노력하겠습니다. 감사합니다.

노션 링크 : <https://www.notion.so/Study>

블로그 링크 : <https://mhoo999.github.io/blog>

The screenshot shows a Notion page titled "Study". At the top left is a stack of three books icon. Below the title is a lightbulb icon followed by the text "점진적 과부하의 원칙". There are two sections under "Study": "Vocabulary" and "기술 용어집".

The main content area is organized into four columns:

- UE**:
 - 1 언리얼엔진 교양
 - 언리얼엔진을 활용한 인터...
 - Inventory System in Unre...
 - Behavior Trees
 - CDO / UCLASS / UObject
 - Gameplay Ability System ...
 - 언리얼엔진 알아가기
- 그래픽스**:
 - [이호성]블리자드 라이팅 ...
 - [이민형]픽사 라이팅 아티...
 - 콜로소 레벨 아트 입문을 ...
 - [William Faucher]Lighting...
 - 한 번에 끝내는 게임 배경...
 - 스킬트리랩 - 시네마틱
 - 이현우)나이아가라 시작...
- 개발**:
 - C++ 기초
 - Codeit
 - [페컴]게임 개발 패키지
 - 컴퓨터 개론
 - Do it! 알고리즘 코딩테스트
 - 이것이 자료구조 + 알고리...
 - 혼자 공부하는 컴퓨터 구...
 - GoF의 디자인 패턴
- 도서**:
 - ABOUTFACE - 앤런 쿠퍼
 - 사용자를 끌어들이는 UX/...
 - 도널드노먼의 디자인과 인...
 - 마이크로카피 - 칸너렛 이...
 - 비전공자를 위한 이해할 ...
 - 메이크타임 - 제이크 냅, ...
 - UX Design Process Best P...

Below the columns are two sections:

- Archive**: A gallery view with a grid icon and the text "갤러리 보기".
- ETC**: A list of various topics:
 - UX KPI
 - project scoping
 - 커뮤니케이션 so what
 - MECE방법론
 - 생각 구조화하기
 - LF FOOD WMS 스터디
 - 조코딩 게임 강의
 - 파인만 테크닉
 - English
 - House

At the bottom right, there is a timeline of entries with labels like WORK, LIFE, and WORK:

- WORK 2023년 7월 15일 오후 5:46
- WORK 2023년 7월 11일 오후 8:39
- WORK 2023년 7월 11일 오후 8:25
- WORK 2023년 7월 11일 오후 8:18
- WORK 2023년 7월 11일 오후 8:01
- WORK 2023년 4월 5일 오후 2:44
- LIFE 2023년 4월 1일 오후 10:08
- WORK 2021년 12월 2일 오후 10:22
- WORK 2021년 11월 27일 오후 10:45
- LIFE 2021년 10월 25일 오후 9:37

THANKYOU

Phone \ 010_5220_9785

Homepage \ <https://mhoo999.github.io>

E-mail \ mhoo999@naver.com

Github \ <https://github.com/mhoo999>