03. 감염된 도시 탈출

게임명 : Escaping Infected City
제작 인원 : 1인
플랫폼 : Unreal
제작 도구 : UnrealEngine 5.1
제작 기한 : 2주
게임 장르 : FPS 서바이벌 호러

게임 선정 이유

언리얼 엔진이 추구하는 그래픽의 색감이 공포 게임과 잘 어울린다 생각해 단순 공포 게임을 제작하려 했으나 기능적으로 전투를 추가로 구현 하기 위해 좀비와 전투하는 창작 게임인 감염된 도시 탈출을 제작했습니다.

Used Skill

동영상 클릭

Name 김유민
Moblie 010 7480 0851
E-mail rladbals0129@gmail.com

언리얼의 기능과 고급 그래픽 렌더링을 활용해 몰입할 수 있도록 설계

플레이어와 적

블랜드 스페이스와 본 별로 그룹을 나눠 레이어 블랜드로 애니메이션을 구현했으며 블랙보드, 행동트리, AI컨트롤러를 이용해 AI를 구현 했습니다.

UI

UI는 플레이어의 상태에 따라 연동되고 게임 진행 상태에 따라 타이머가 표기됩니다.

시퀀서 연출과 카오스 디스드럭션

특정 조건을 만족하고 트리거 박스에 도달하면 그에 맞는 시퀀서 연출이 재생됩니다. 카오스 디스트럭션을 사용하여 화려한 연출을 구현했습니다.

01 플레이어

자연스러운 움직임을 위해 엔진에서 지원하는 기능을 적극 활용했으며 타격감을 살리기 위해 카메라의 기능 또한 적극 활용했습니다. 일부 기능은 C++코드를 활용 해 구현 해 봤습니다.

애니메이션

블랜드 스페이스로 애니메이션을 구성하고 애니메이션 블루프린트에서 거리와 속도를 구하는 로직을 작성했습니다. 그 외 레이어별 블랜드, 몽타주 등 다양한 기능을 활용했습니다.

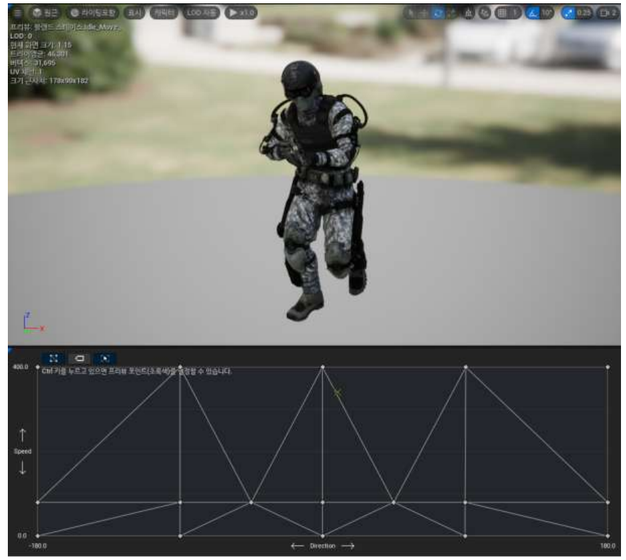
피직스를 이용한 피격

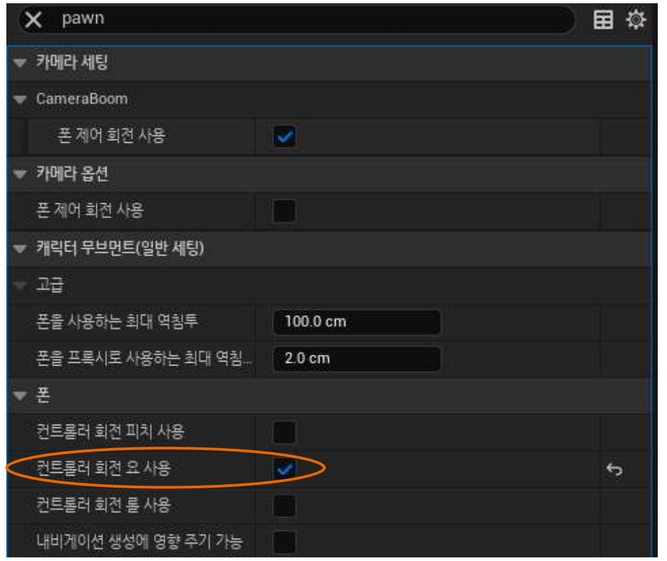
플레이어 피격시 피직스 에셋을 이용해 피격모션을 만들었습니다.

사격과 충돌처리

라인 트레이스를 이용해 충돌한 개체를 확인했으며 맞은 대상 별로 다른 이펙트를 생성 했습니다.

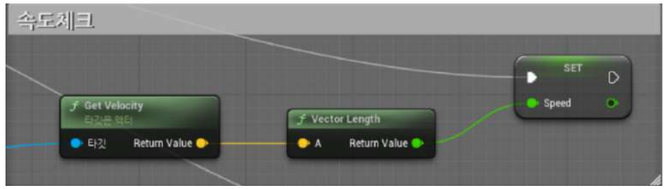
블랜드 스페이스 설계

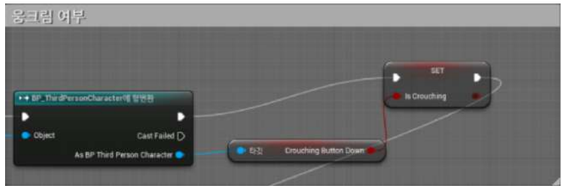


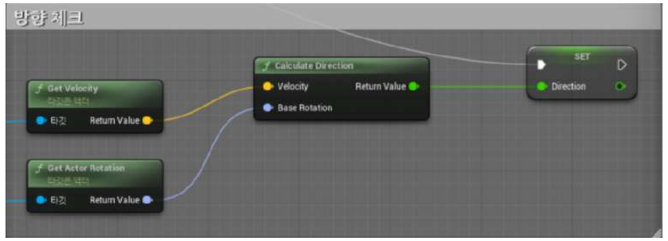


- 플레이어의 이동속도에 맞게 Speed값 구성
- Direction 값을 -180 ~ 180으로 구성
- 컨트롤러 회전 요 사용을 True로 만들어 폰과 컨트롤러가 매칭되게 설정

AnimBP에서 필요한 값 구하기

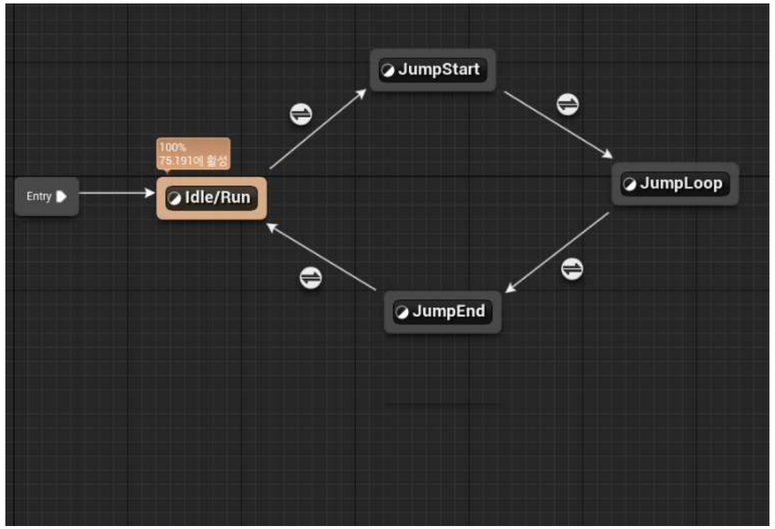


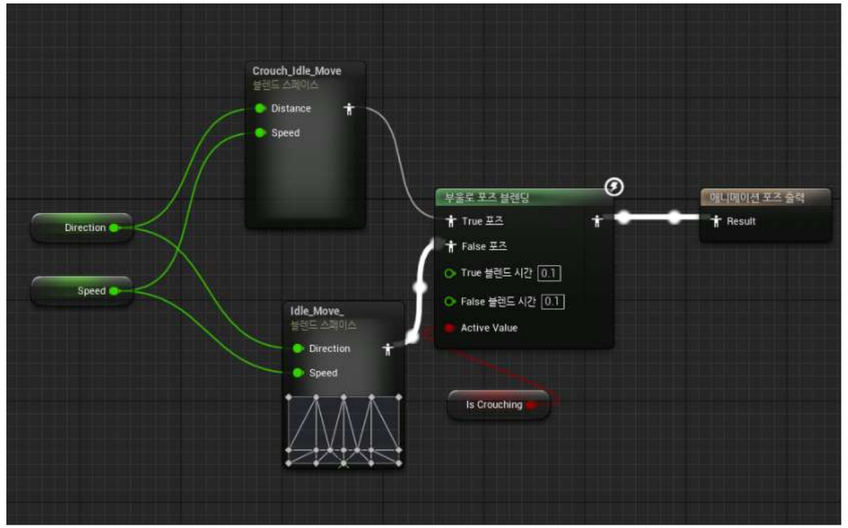




- AnimBP의 이벤트 그래프에서 변수를 만들어 필요한 값을 구함
- Speed와 Direction에 값이 할당되고 웅크리기 여부를 확인

스테이트 머신





- Idle / Run 의 내부
- 스테이트 머신의 이름을 MoveState로 지정
- 웅크림 여부에 따라 포즈 블랜딩

결과 화면

- 속도와 방향, 상태에 맞게 올바른 애니메이션 재생

구성한 이유

애니메이션 상태가 많아지면 스테이트머신이 굉장히 복잡해질 여지가 있기 때문에 묶을 수 있는 상태는 최대한 묶고 간소화 시킬 수 있는 상태는 최대한 간소화 시키기 위해 사용하였습니다.

사격 몽타주

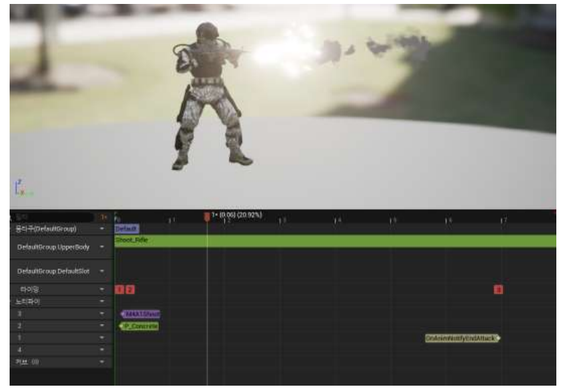
```
UPROPERTY(EditAnyWhere)
UAnimMontage* AttackMontage;

UFUNCTION(BlueprintCallable)
void DisableAttacking();


blic:
//UPROPERTY(EditAnyWhere);
UPROPERTY(BlueprintReadWrite);
int CurrentBulletNum;

UPROPERTY(BlueprintReadWrite);
bool IsReload;
```

```
void APortfoilioProjectCharacter::BeginPlay()
{
    // Call the base class
    Super::BeginPlay();

    //Add Input Mapping Context
    if (APlayerController* PlayerController = Cast<APlayerController>(Controller))
    {
        if (UEnhancedInputLocalPlayerSubsystem* Subsystem = ULocalPlayer::GetSubsystem<UEnhancedInputLocalPlayer
        {
            Subsystem->AddMappingContext(DefaultMappingContext, 0);
        }
    }

    InputComponent->BindAction("KeyPlayerAttack", IE_Pressed, this, &APortfoilioProjectCharacter::PlayerAttack);
}
```

```
void APortfoilioProjectCharacter::PlayerAttack()
{
    if (bAttacking == false && CurrentBulletNum > 0 && IsReload == false)
    {
        bAttacking = true;
        CurrentBulletNum--;
        PlayAnimMontage(AttackMontage);
    }
}


void APortfoilioProjectCharacter::DisableAttacking()
{
    bAttacking = false;
}
```

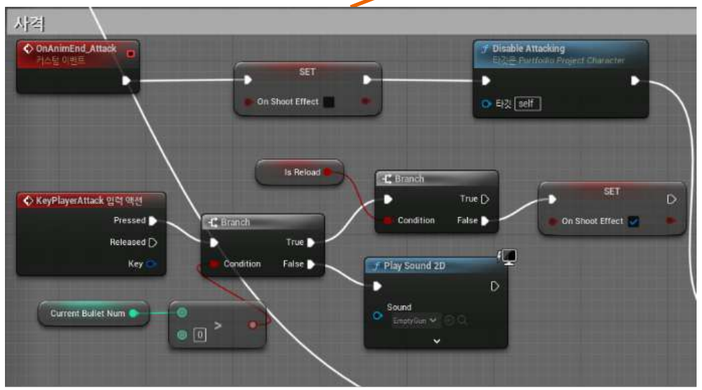
1. UPROPERTY(EditAnyWhere)를 이용해 몽타주 애니메이션을 에디터에서 설정할 수 있게 구성
2. UFUNCTION(BlueprintCallable)를 통해 함수를 블루프린트에서도 호출할 수 있게 설정
3. UPROPERTY(BlueprintReadWrite)를 통해 변수를 블루프린트에서도 접근할 수 있게 설정
4. PlayerAttack 함수를 구성하고 몽타주를 실행
5. 액션을 BeginPlay()에서 바인드

사격 몽타주 노티파이



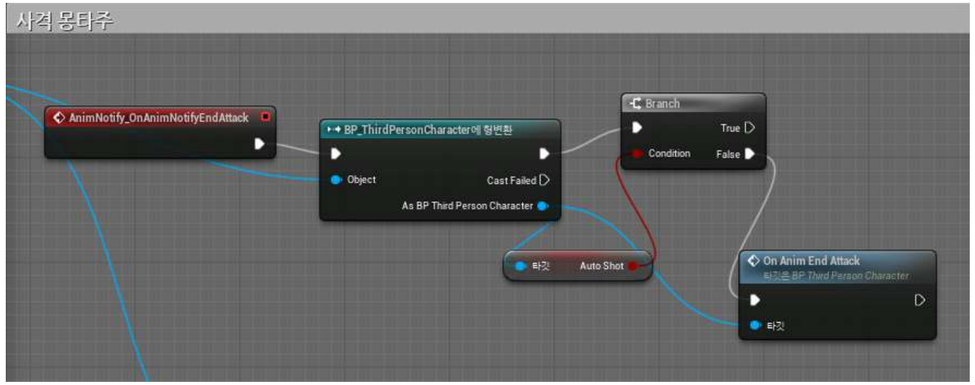
- 애니메이션이 시작되는 부분에 사운드 재생과 총구 이펙트 생성
- 프레임이 끝날때 쯤 노티파이를 만들어 사격이 끝났음을 알릴 준비

PlayerBP

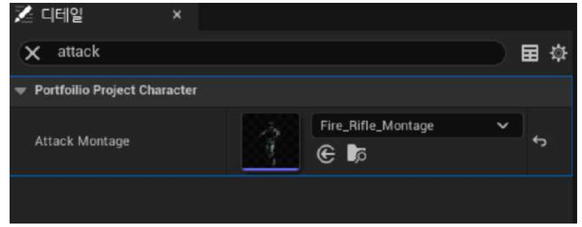


- 커스텀 이벤트를 만들어 함수 호출

AnimBP



- 커스텀 이벤트를 만들어 함수 호출

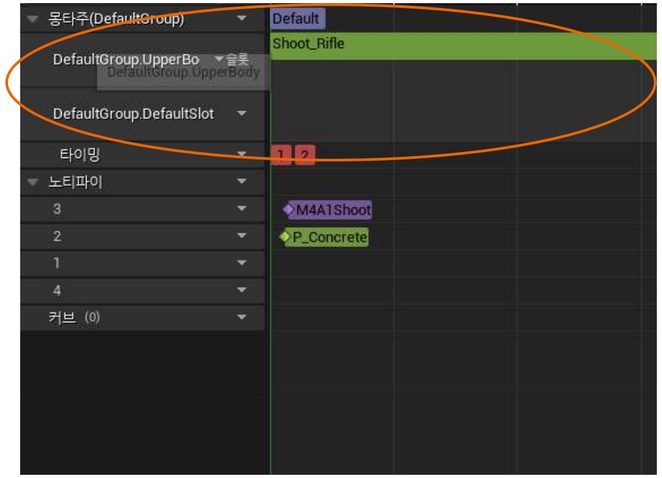


- 몽타주 애니메이션 지정

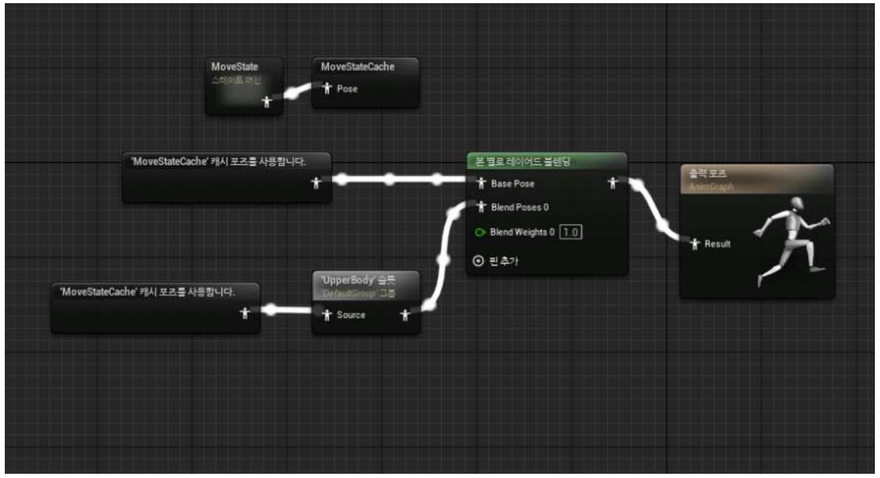
발생했던 문제점과 해결방법

사격 몽타주 애니메이션이 문제 없이 정상적으로 작동하고 사격 역시 잘 작동하는 상태지만 이동하면서 사격, 장전하면서 이동 등 몽타주를 실행하면서 다른 애니메이션이 재생될때 문제가 발생했습니다.
이를 해결하기 위해 애니메이션의 레이어를 Upper슬롯과 Default슬롯으로 분리하여 상반신과 하반신의 애니메이션 처리를 따로 실행해주었습니다.
상반신에서 처리할 수 있는 모든 애니메이션은 모두 이 방법으로 처리했습니다.

애니메이션 레이어 분리



1. 애니메이션 레이어 분리
2. UpperBody슬롯에 사격 애니메이션 지정



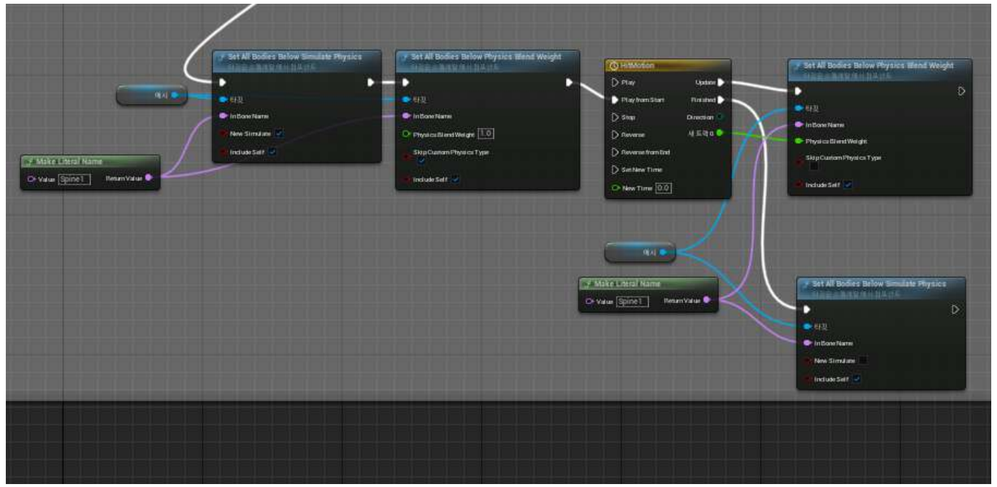
1. MoveState를 캐시 포즈로 저장
2. UpperBody와 기존 포즈를 본 별로 레이어드 블렌딩

결과 화면

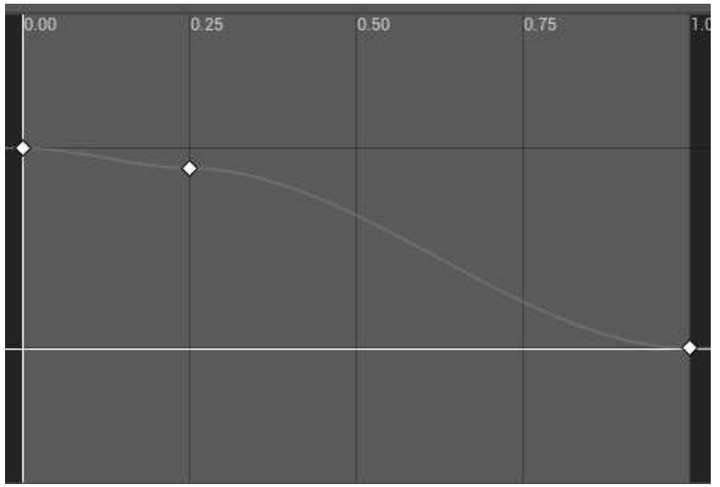
결과와 느낀점

상반신은 몽타주를 하반신은 걷기,뛰기 등 애니메이션을 재생해 다채로운 애니메이션이 구현되었습니다.
언리얼 엔진엔 내가 생각하는 웬만한 기능들이 이미 엔진에서 지원하고 있는 것 같습니다. 점차 엔진 사용이 익숙해 질 수록 모르는 기능들도 추론을 통해 더 쉽게 찾아가며 발전하고 있다는게 느껴졌습니다.

피직스 시뮬레이션

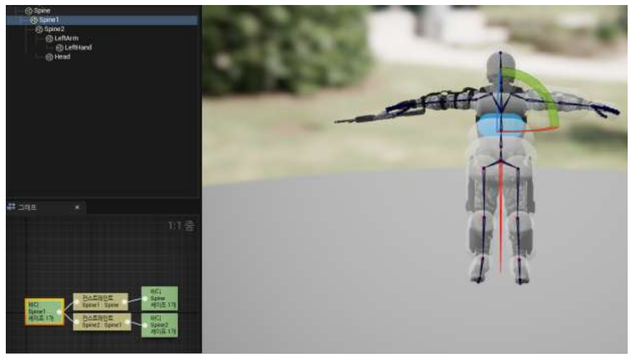


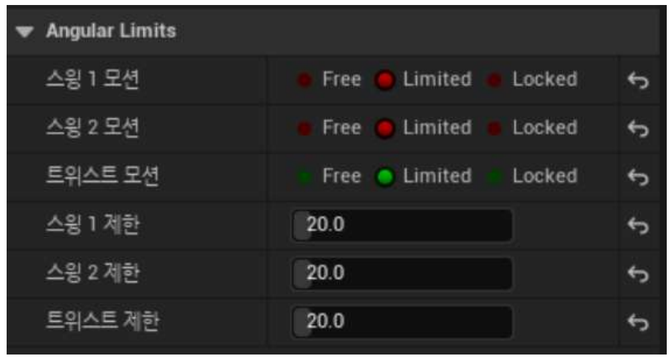
- 하위 본을 시뮬레이션 하는 노드를 연결
- Spine1을 기준으로하여 상반신에 1초간 피직스를 시뮬레이션



- 부드럽게 모션이 재생되었다가 원래 상태로 돌아가기 위해 타임라인 사용

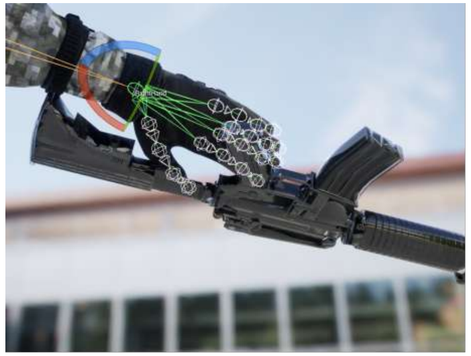
피직스 에셋





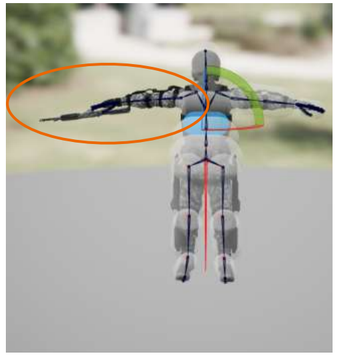
1. Spine1 보다 하위에 있는 시뮬레이션 되는 본의 Angular Limits의 모션을 Limited로 설정
2. 제한을 20으로 줄여 너무 과하게 시뮬레이션 되지 않게 제한

발생한 문제점



총기는 오른손에 연결되어 있는데 Spine1의 하위 본에 위치해 있습니다. 프리뷰 전용이라 직접 피직스가 시뮬레이션 되지는 않지만 오른쪽 팔, 손 등 상위에있는 본들이 움직이면서 괴상한 동작이 시뮬레이션 되는 문제가 있었습니다.

해결 방안

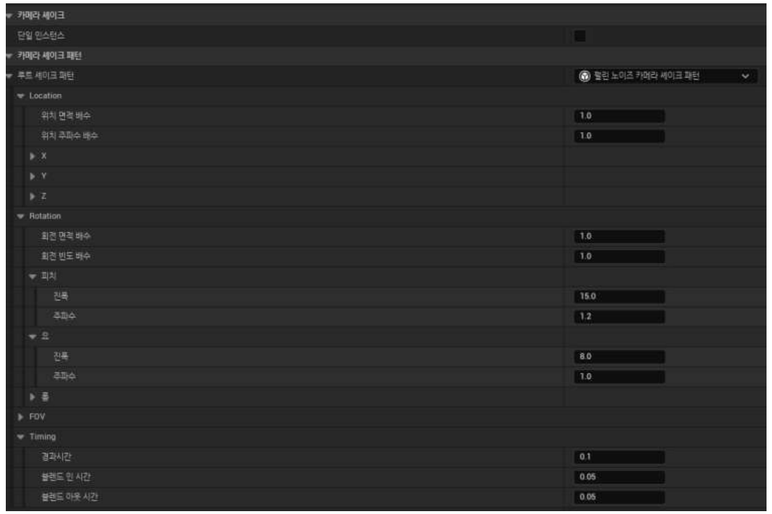


총기를 들고있는 오른쪽 팔의 본에 대한 피직스 연결을 끊어줘 시뮬레이션 하지 않는 상태로 만들어 피격 모션을 구현 했습니다.

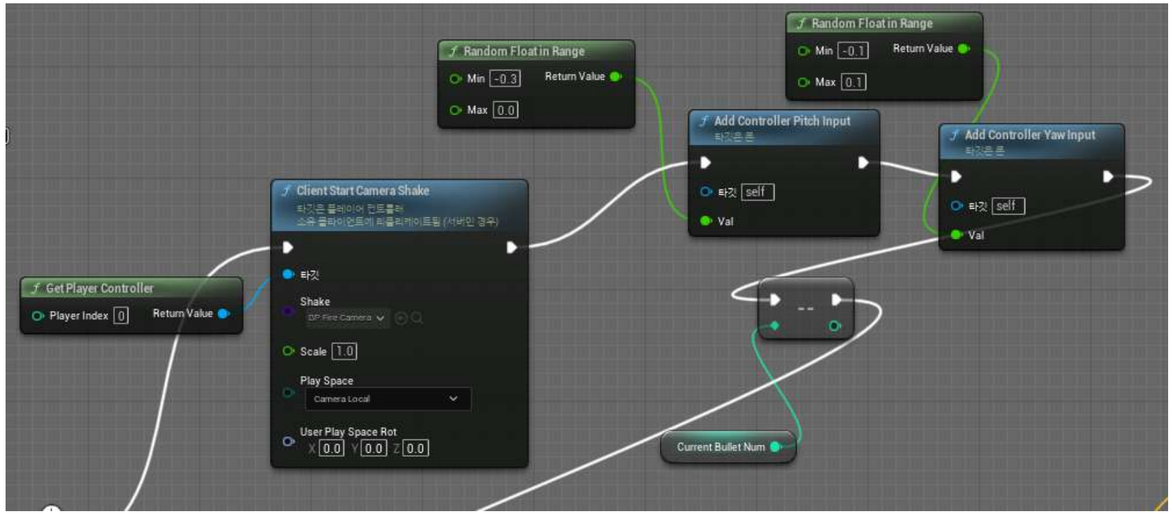
결과 화면

- 적에게 피격할 시 피직스를 시뮬레이션 하고 타임라인을 통해 원래 위치로 돌아옵니다.

총기 반동



1. 블루프린트 카메라 셰이크 패턴 생성
2. 펄린 노이즈 카메라 셰이크 패턴 사용
3. 총기 반동의 화면 흔들림 효과를 원하는 것이기에 Rotation값 조절
4. 피치값의 진폭을 15로 설정 해 상하 흔들림 값을 지정
5. 요값의 진폭을 8로 설정 해 좌우 흔들림 값을 지정
6. 빠르게 원래 자리로 회복되어야 하기 때문에 경과시간, 블랜드 시간 조절

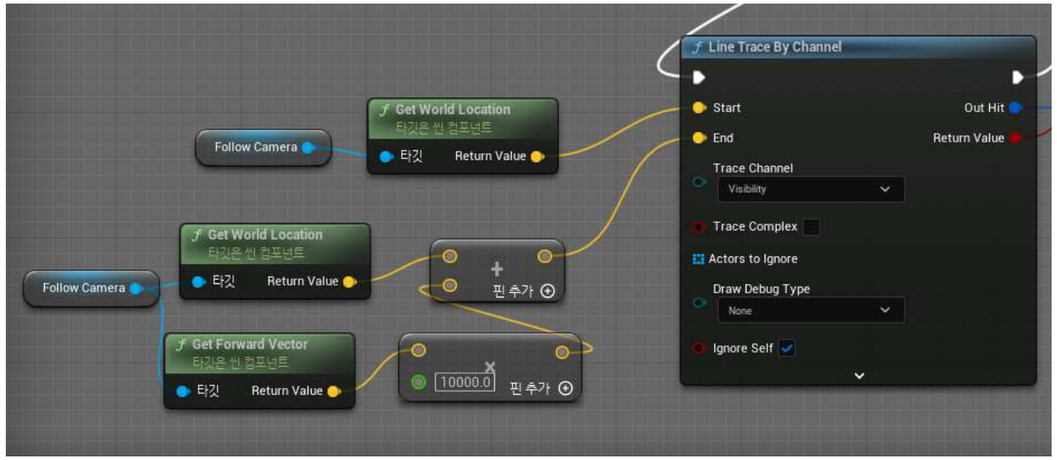


1. 설정한 카메라 셰이크 패턴 사용
2. Pitch값을 위쪽으로 올려줘 상하반동 구현
3. Yaw값의 최대값과 최소값을 지정한 랜덤값을 더해줘 좌우반동 랜덤 구현

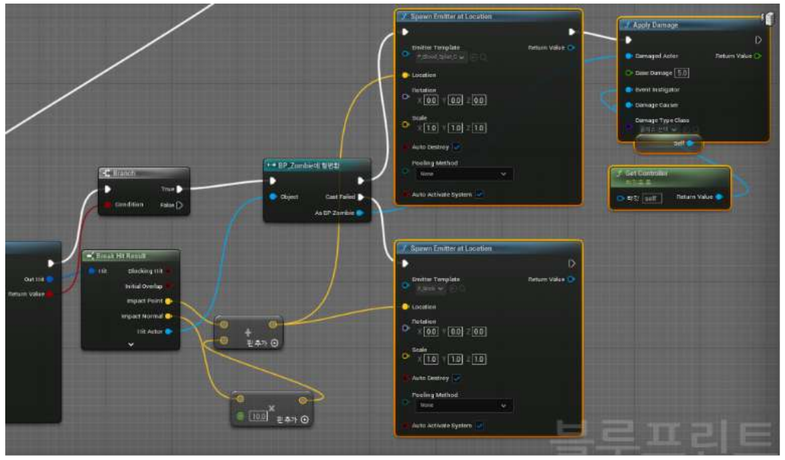
결과 화면

화면이 흔들리며 크로스헤어가 상, 좌우로 이동

충돌처리



1. 라인의 시작 위치를 카메라 좌표로 설정
2. 최대 사정거리를 설정

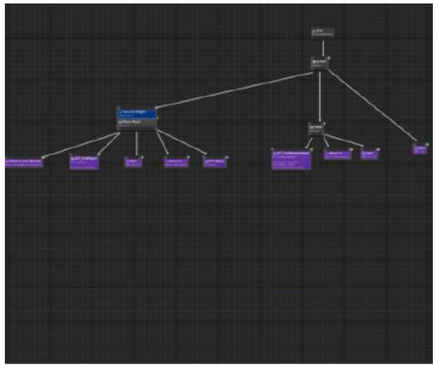


1. 이펙트가 생성되는 부분을 살짝 앞으로 당김
2. 맞은 대상이 좀비라면 블러드 이펙트
3. 좀비가 아니라면 탄피 이펙트를 생성

결과 화면

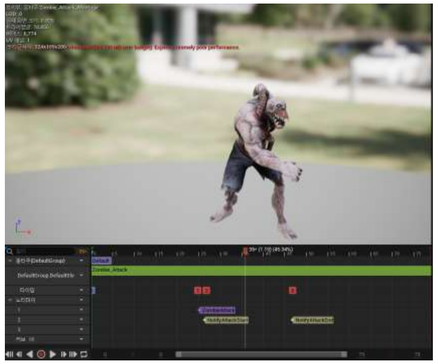
02 적

적의 종류는 총 3가지로 플레이어를 인식하기 전까지는 이동할 수 있는 랜덤한 위치로 패트롤하고 플레이어를 발견할 시 스크림 애니메이션 재생 후에 플레이어를 추적하며 공격 사정거리 안에 들었을 때 공격을 시행합니다. 마지막 타이머가 켜졌을 때는 플레이어가 어디에 있든 항상 추적합니다.



AI

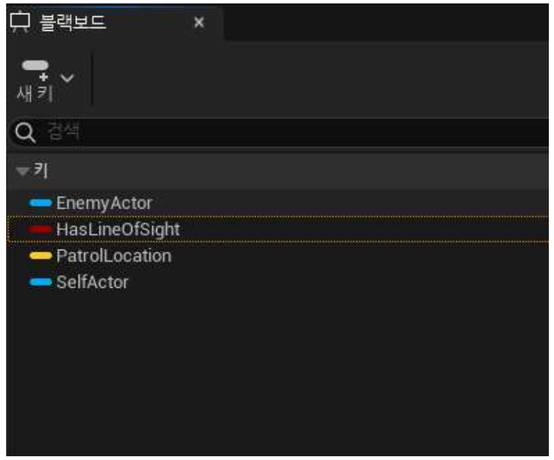
행동트리, 블랙보드, AI컨트롤러를 사용해 AI를 구현했습니다.



애니메이션

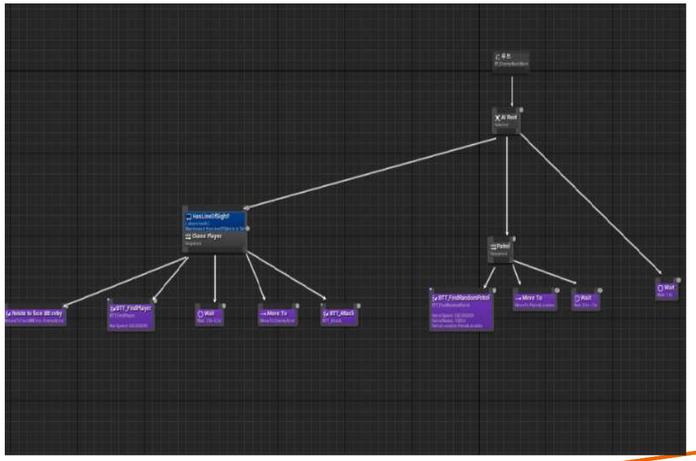
블랜드 스페이스1D를 이용해 속도에 대한 애니메이션을 구성하고 애니메이션 블루프린트에서 속도를 구하는 로직을 작성했습니다. 공격 몽타주에서 Attack Collision을 On / Off 해주었습니다.

블랙보드 키값 설정



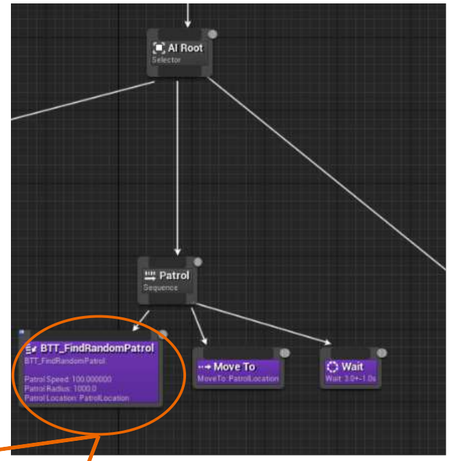
- AI구현에 필요한 블랙보드 키값 설정

행동트리 설계



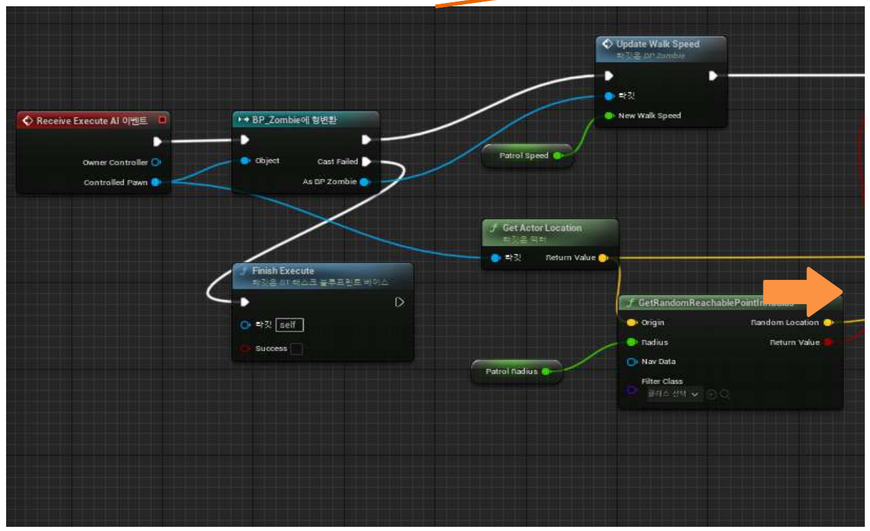
확대해서 설명하겠습니다.

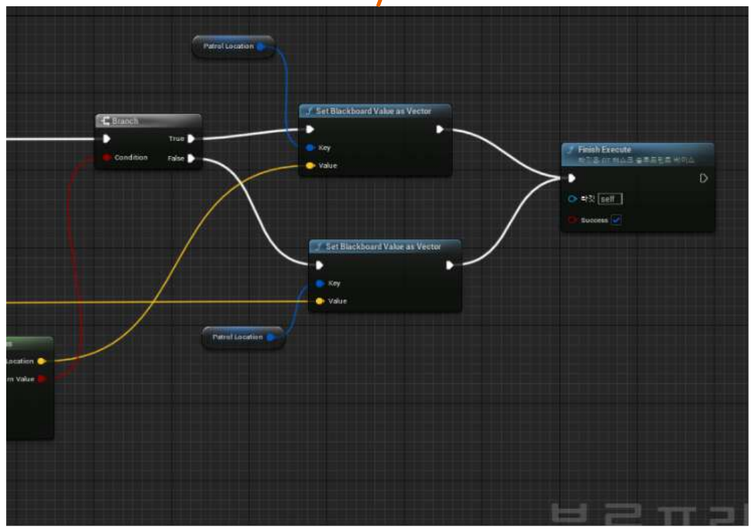
발견하지 못한 경우



- 랜덤한 위치로 정찰하는 시퀀스의 테스크노드 -

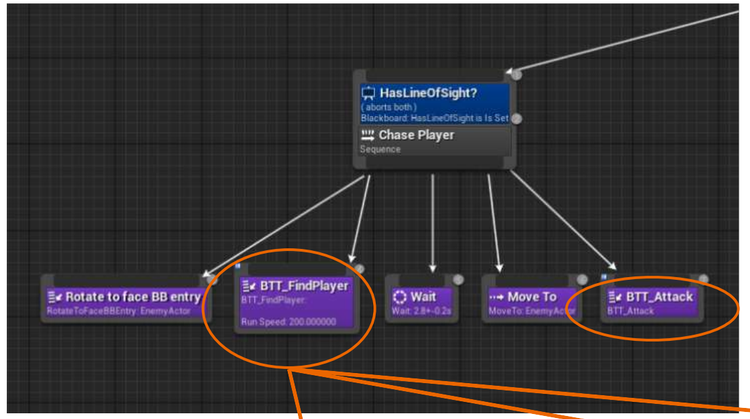
1. 랜덤한 정찰 위치 추적
2. 정찰 위치로 이동
3. +- 1초의 오차 범위로 대기





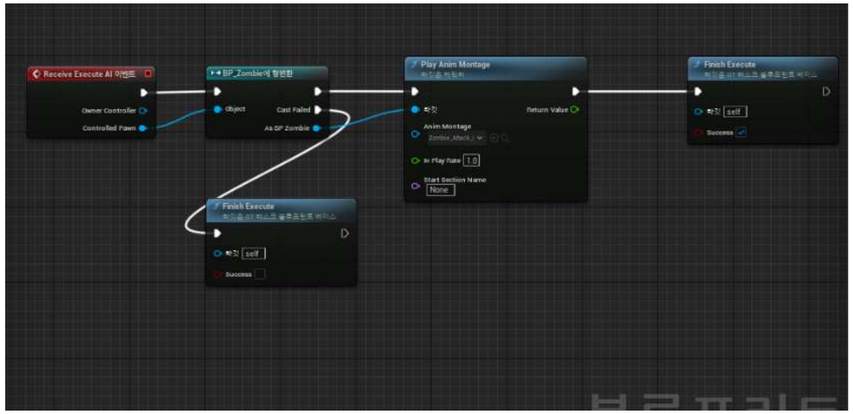
1. 정찰 속도를 100으로 지정
2. 정찰 반지름을 1000으로 지정
3. 현재 위치에서 반지름만큼 랜덤한 위치를 추적
4. 장소를 찾은 경우 해당 위치를 정찰 포인트로 지정하고 이동
5. 이동하지 못하는 곳이라 장소를 찾지 못한 경우 다시 장소를 추적

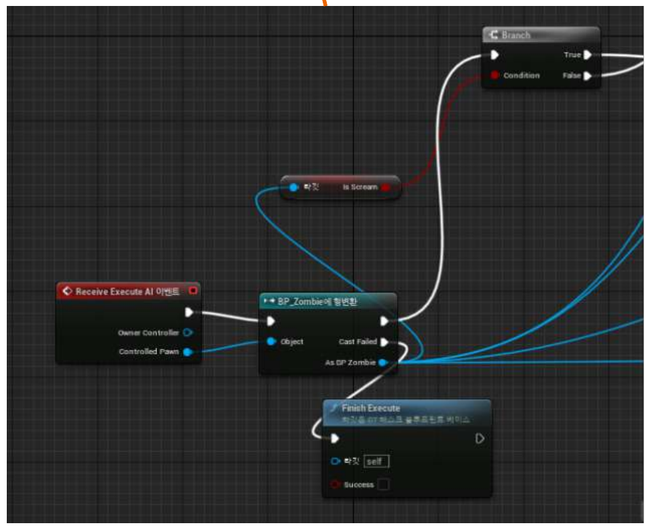
AI가 플레이어를 발견한 경우



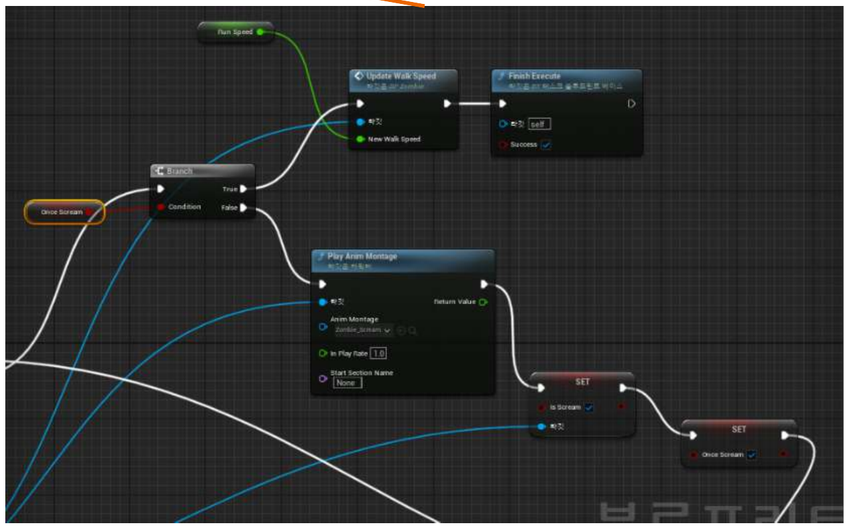
1. 플레이어를 발견 할 시 데코레이터 노드인 HasLineOfSight 값으로 인해 이 노드로 진입
2. 왼쪽부터 순차적으로 진행, 고개를 플레이어 방향으로 돌린다
3. FindPlayer 진입
4. 대기시간 2.8(오차범위 0.2)
5. 이동 (플레이어에게)
6. 가까워질시 공격

BTT_ATTACK

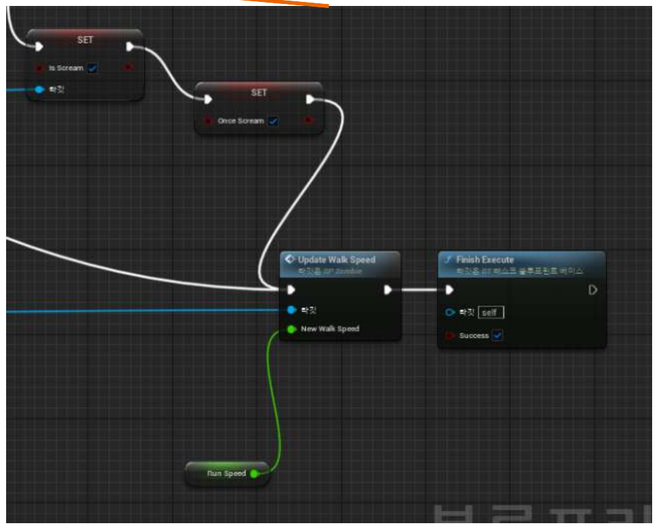




조우 후 스크림을 재생 했는지 체크



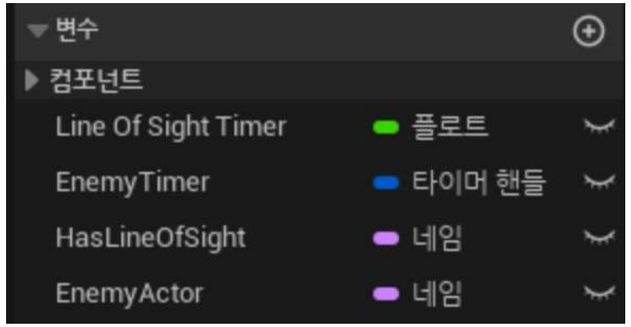
스크림을 재생하지 않았다면 스크림 진행 후 추적 스피드 업데이트 (200)



이미 한번 스크림 한 상태였다면 이동속도 바로 업데이트

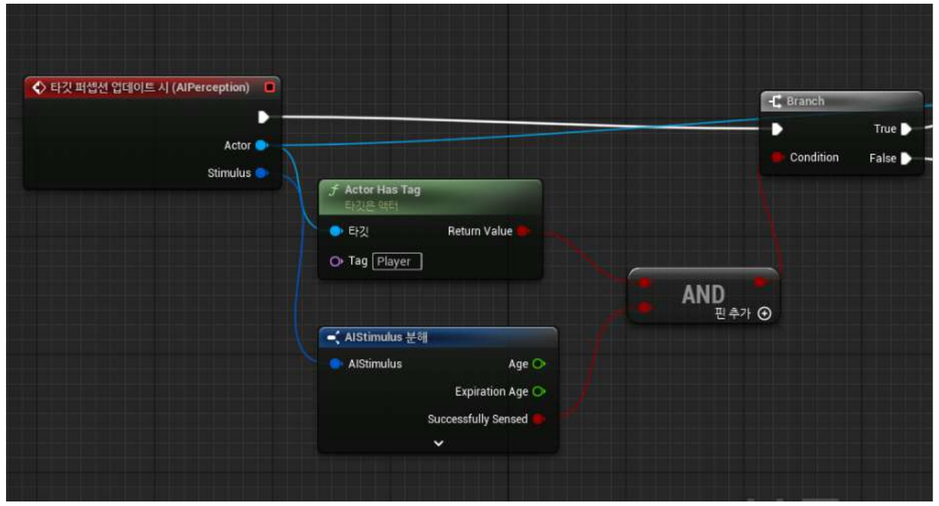
AI 컨트롤러

AI컨트롤러와 행동트리를 연결



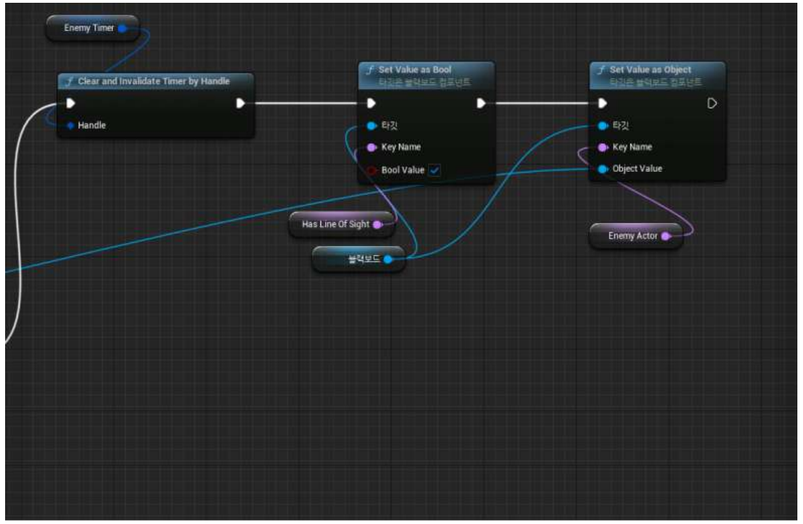
블랙보드에 있는 필요한 변수 초기화 진행
1. 감지가 초기화 될 시간 (3초)
2. 블랙보드로 넘겨줄 타이머 변수
3. 블랙보드로 넘겨줄 감지 변수
4. 블랙보드로 넘겨줄 감지 액터 변수

AI 퍼셉션으로 감지 시야범위 구성



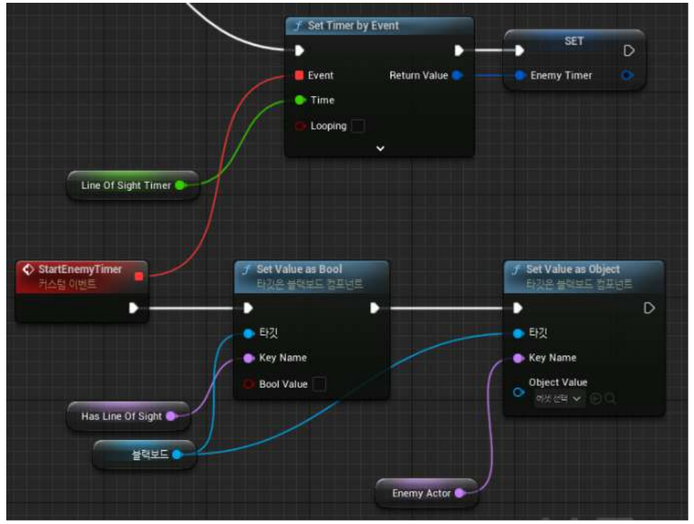
감지된 Actor의 태그가 Player이고 감각 업데이트에 성공여부

감지 성공 시



초기화 시간을 초기값으로 돌리고 플레이어를 추적하는 테스크 노드로 진입

감지 실패 시



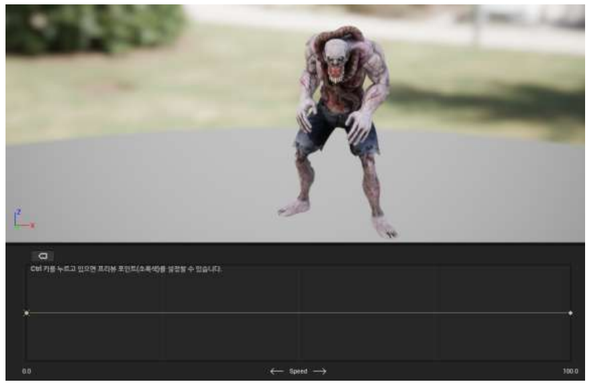
1. Line of Sight Timer의 값이 지나면 아래 이벤트를 실행
2. 행동트리가 감지 초기화 후 정찰 상태로 진입

결과 화면

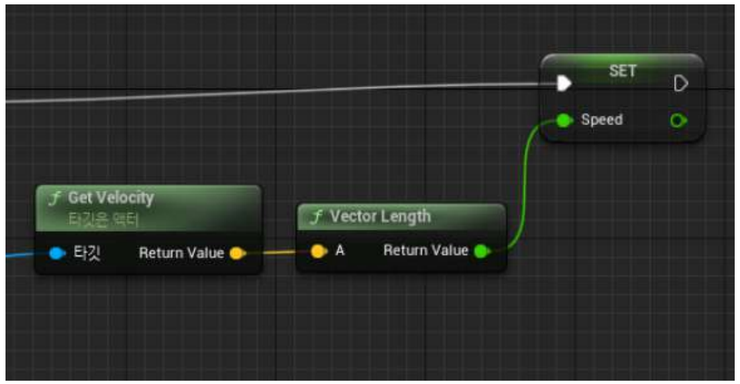
어려웠던 점

블랙보드, 행동트리, 컨트롤러 3개로 나뉘어서 AI를 구현함에 있어서 생소함에서 오는 어려움이 컸습니다. 구현을 진행하다 보니 점차 익숙해졌고 각각의 하는 역할을 이해하다 보니 사용하기 편해졌습니다.

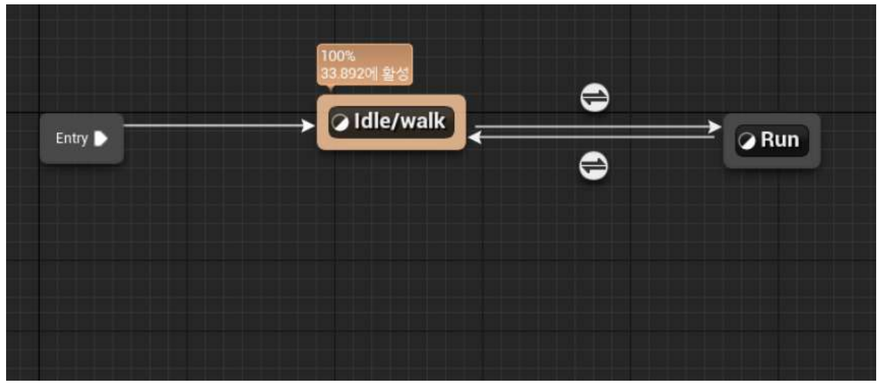
이동관련 애니메이션



블렌드 스페이스 1D를 이용해 Speed값

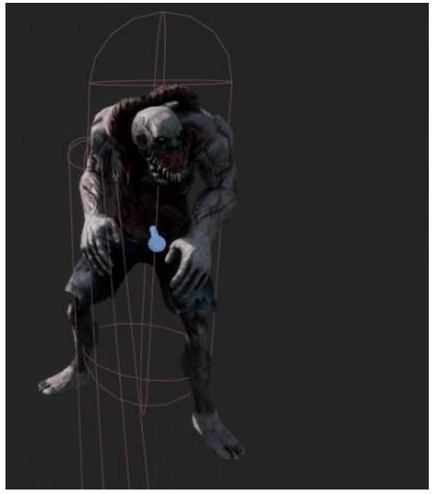


AnimBP에서 Speed값 할당

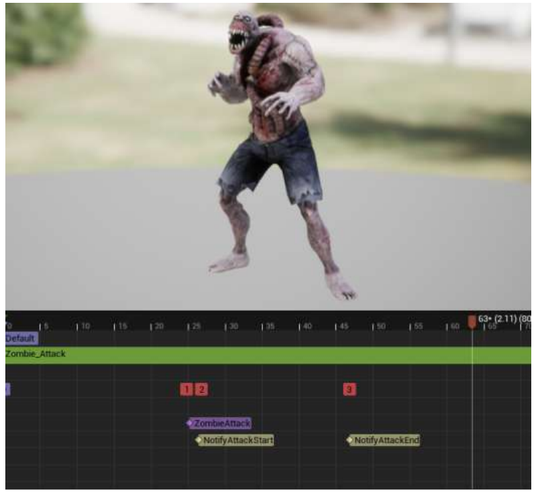


스테이트 머신에 Speed값으로 컨트롤되는 노드 구성

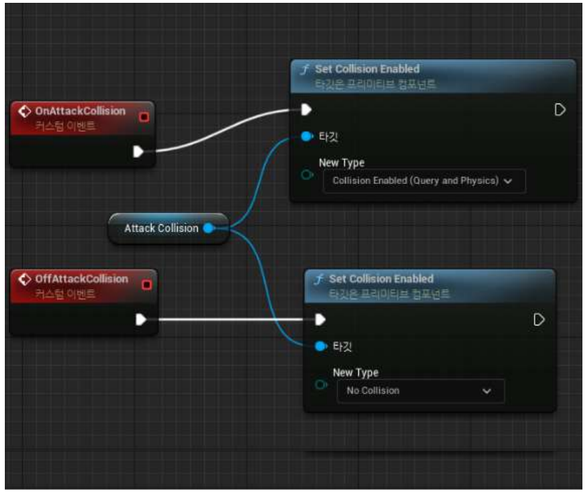
공격 관련 애니메이션



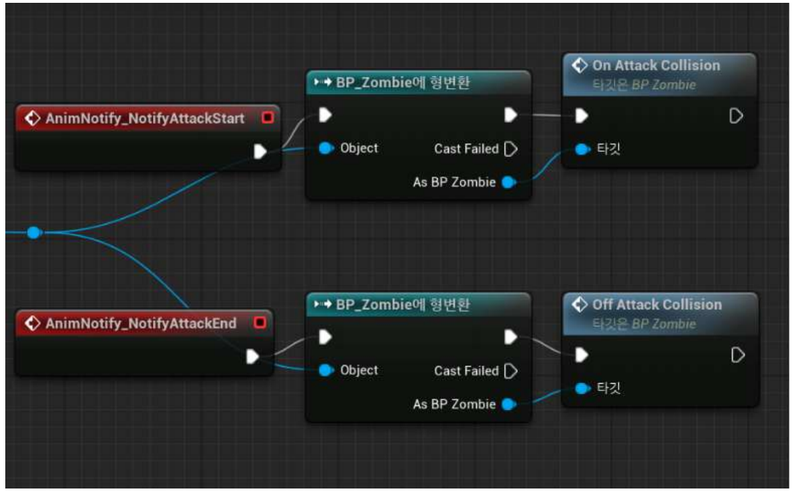
Collsion 구성



노티파이로 공격의 시작과 끝 설정



공격 충돌체 On / Off 이벤트 구성



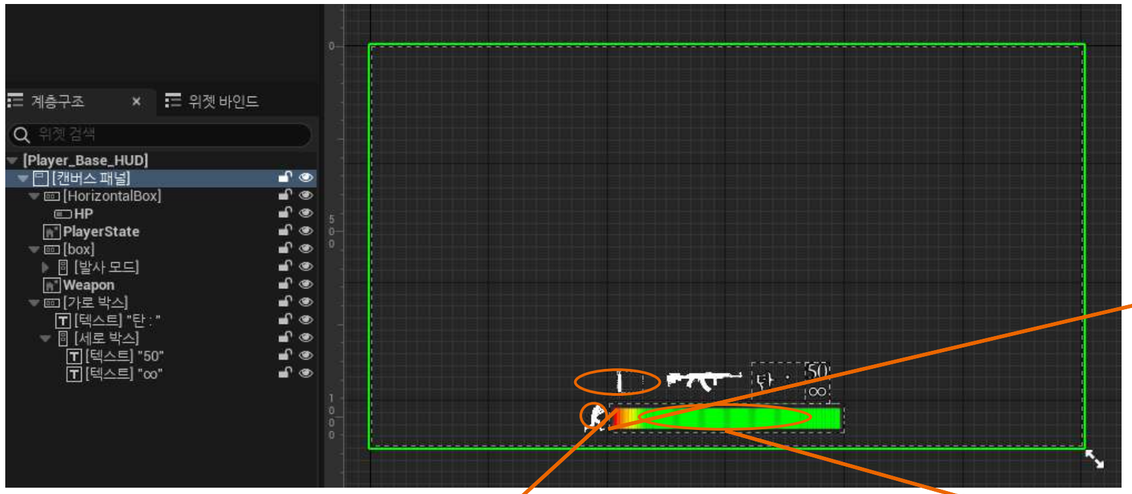
AnimBP에서 노티파이 부분에서 이벤트 호출

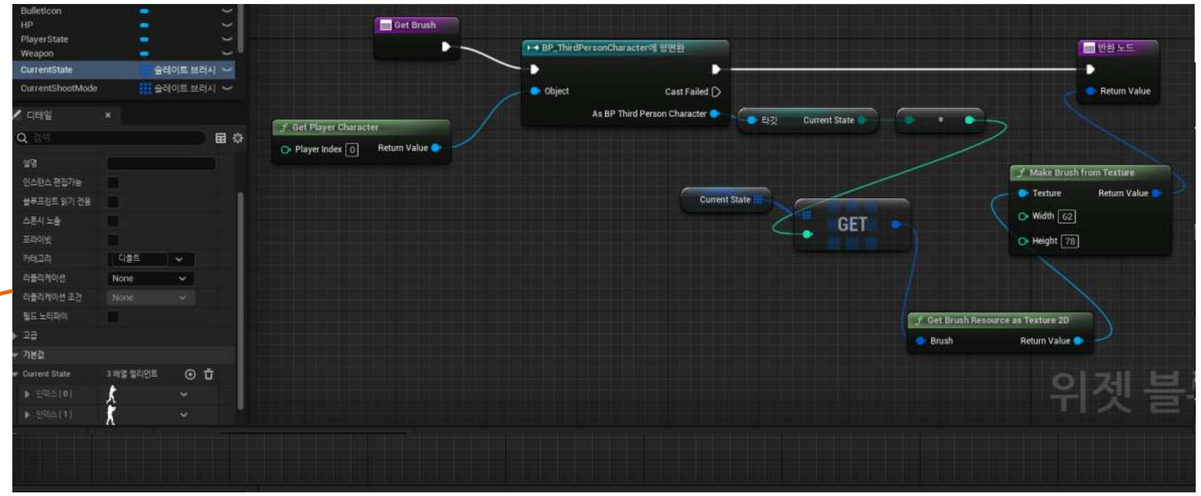
03 UI

플레이어와 게임 상태에 따라 상호작용하는 UI

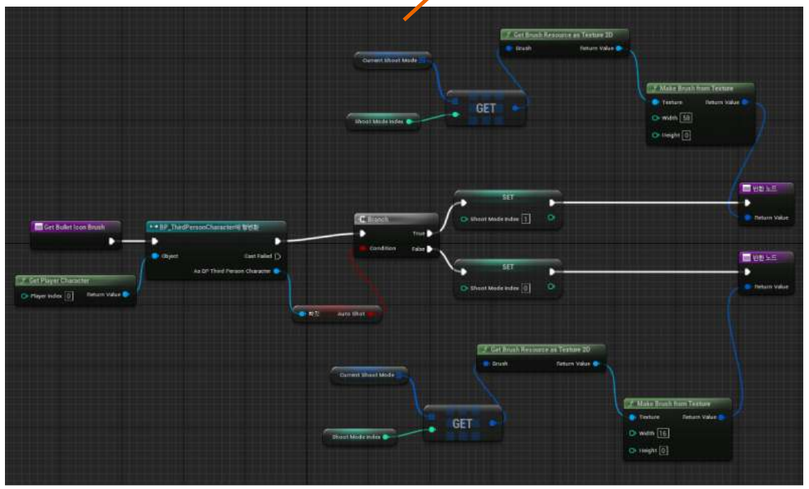
플레이어의 현재 상태를 바인드 하여 상호 작용하는 UI와
게임의 상태에 따라 생성되는 버티기 타이머 입니다.

캔버스 계층구조

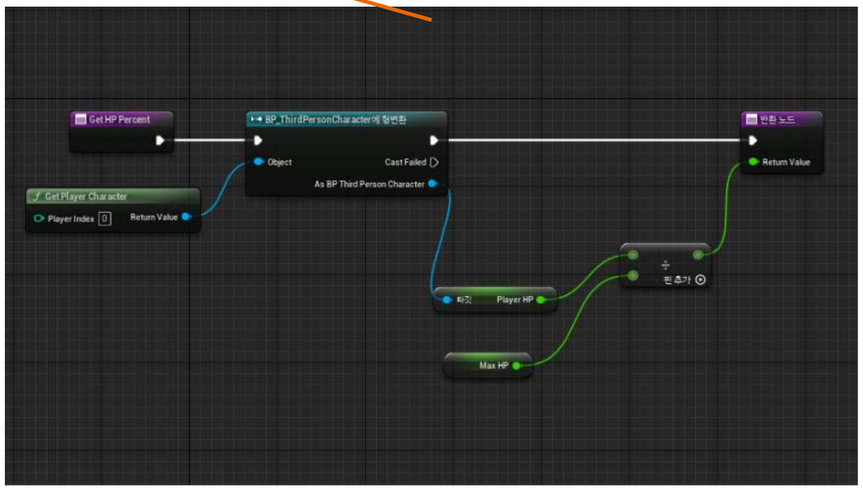




플레이어의 상태에 따라 걷기, 조준, 앉기 이미지 변경

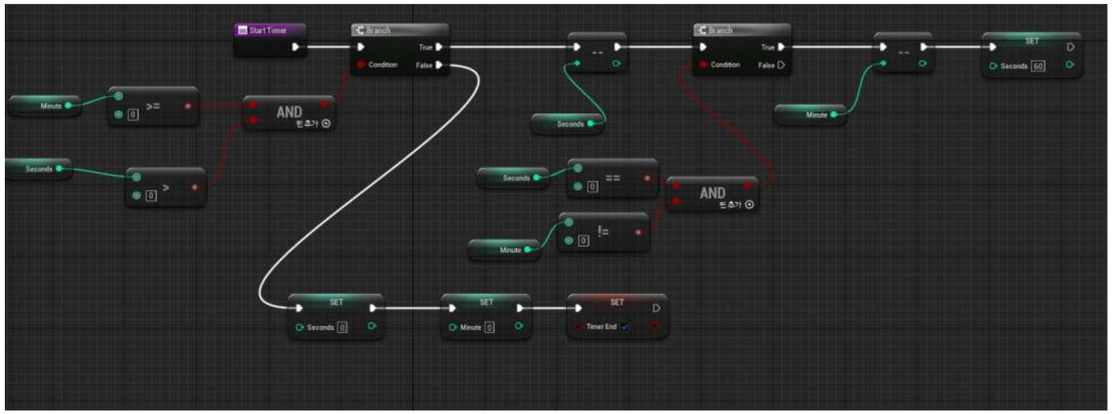


사격 모드에 따라 단발,연발 이미지 변경

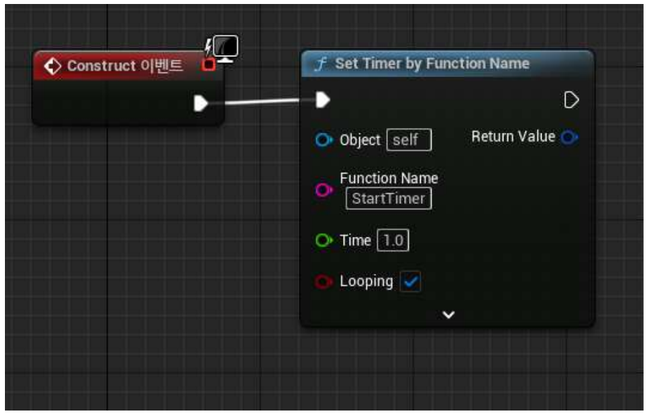


플레이어 체력을 비율로 받아 화면에 프로그래스바 연동

타이머

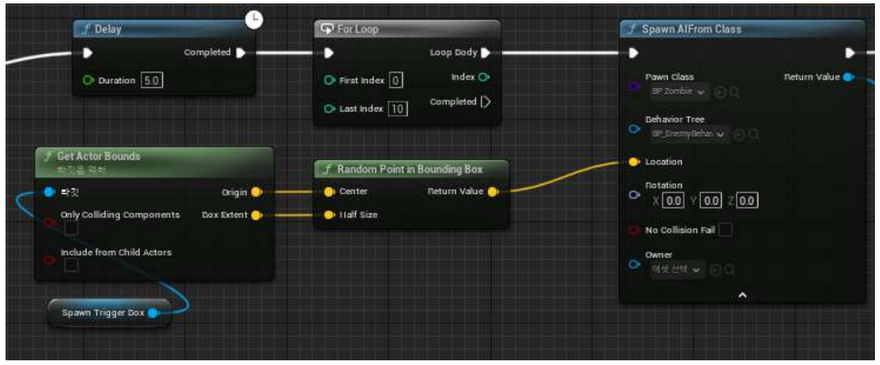


타이머 설정 로직



타이머가 활성화 되었을때부터 호출

좀비 스폰

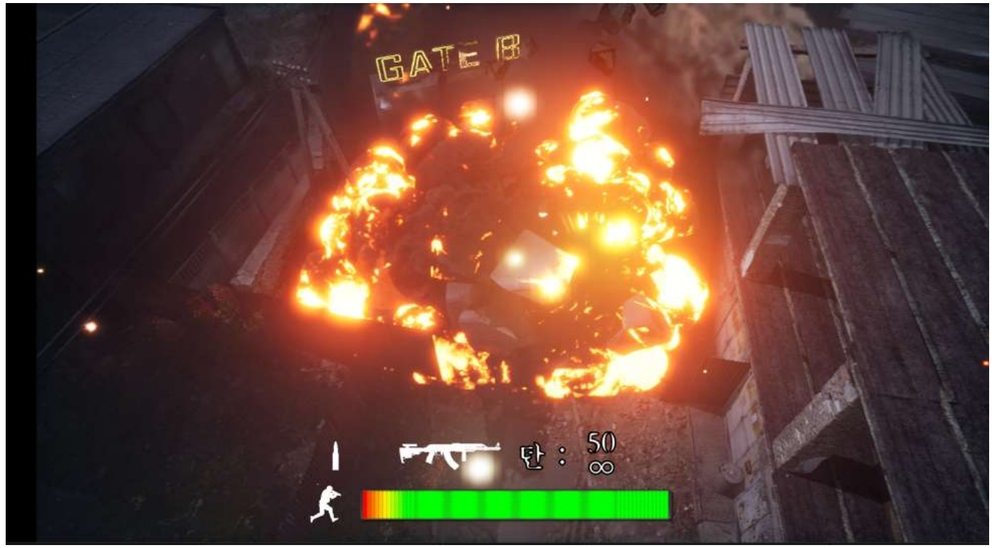


타이머가 활성화 되었을 시 좀비 스폰

결과 화면

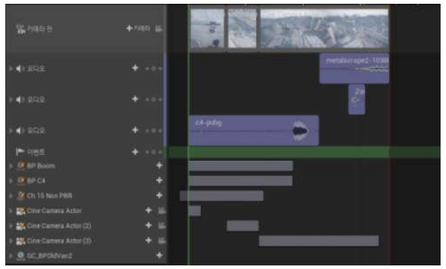
04 시퀀서와 카오스 디스트럭션

적의 종류는 총 3가지로 플레이어를 인식하기 전까지는 이동할 수 있는 랜덤한 위치로 패트롤하고 플레이어를 발견할 시 스크림 애니메이션 재생 후에 플레이어를 추적하며 공격 사정거리 안에 들었을 때 공격을 시행합니다. 마지막 타이머가 켜졌을 때는 플레이어가 어디에 있든 항상 추적합니다.



카오스 디스트럭션

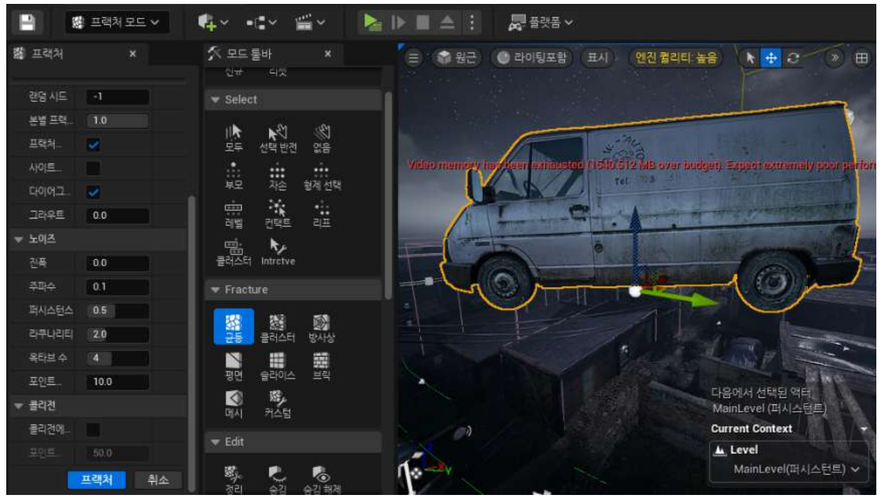
프렉쳐 모드에서 균등하게 조각낸 카오스 디스트럭션 오브젝트입니다. 시퀀스 카메라에서 연출해 주었는데 기본적으로 시퀀서에선 Static으로 설정되어 있어서 블루프린트를 이용해 산산조각 내주었습니다.



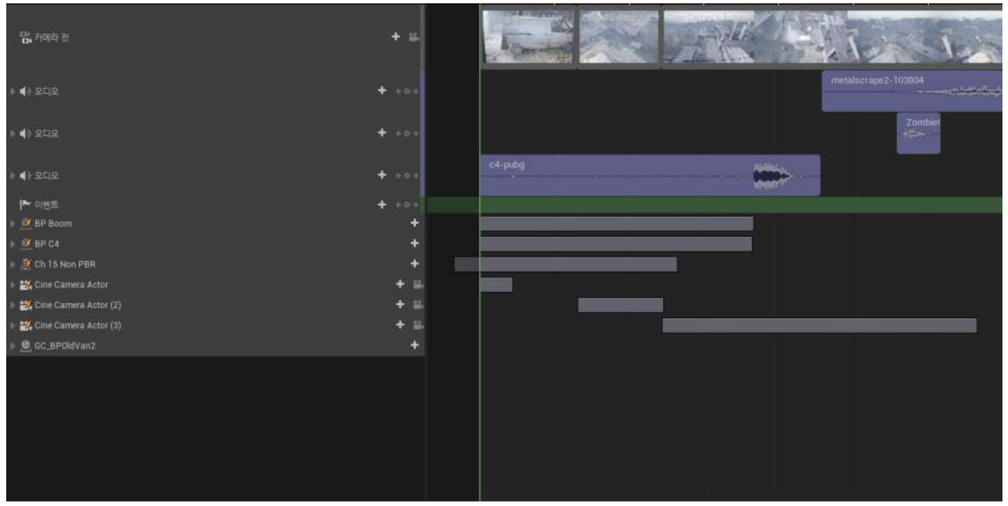
시퀀서

특정 트리거박스 도달 시 재생되는 레벨시퀀스 입니다. 총 5개가 있으며 뒷페이지에서 자세히 설명드리겠습니다.

GC 생성

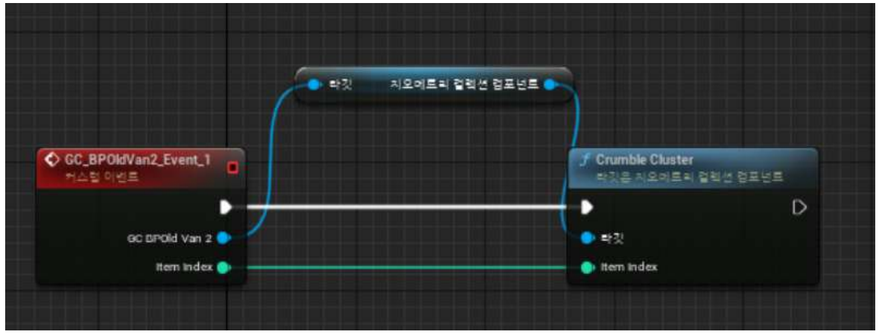


C4 시퀀스



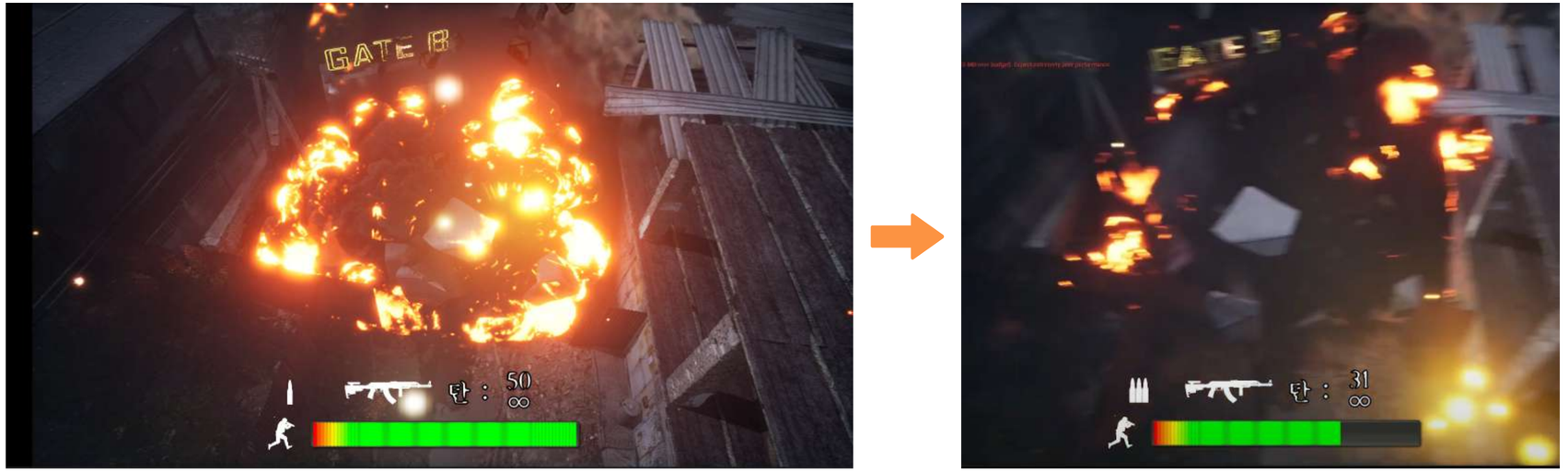
총 3개의 카메라 컷으로 연출

지오메트리 컬렉션인 OldVan에 이벤트 트리거를 설정



이벤트 트리거에서 지오메트리 컬렉션을 산산조각

결과 화면



어려웠던 점

시퀀스 카메라 내에서 GC를 파괴하는 방법을 찾는게 처음에 어려웠습니다. 이벤트 트리거를 생각하기 보단 Collision을 이용해 마스터필드랑 연동해주는 것을 먼저 생각했는데 공부를 진행하다 보니 시퀀서 내에선 연출을 위해 오브젝트가 별다른 처리를 하지 않았을 때 Static 상태 라는 것을 알게 되었고 이벤트 트리거로 산산조각 내는 노드를 알게 되어 효과를 구현했습니다.

결과/회고

언리얼 프로젝트를 마치며

게임 엔진의 유용함과 3D 환경에서의 프로그래밍의 깊이를 체감할 수 있는 시간이였습니다.

1. 언리얼 엔진을 사용하면서, 유용성을 한번 더 체감할 수 있었습니다. 엔진을 사용하니 복잡한 코딩, 개발 작업, 수학과 물리 등 시간이 걸리는 작업들을 엔진 레벨에서 처리 해주니 간단하게 처리할 수 있었고, 이를 통해 개발시간 단축과 프로젝트 효율성이 크게 증가 한다는 것을 느꼈고 직접 프로젝트를 진행해 보니 언리얼에 대한 숙련도가 높아졌습니다.

2. 3D 환경에서 게임을 제작 해 보니 게임 프로그래머 라는 직업이 더 매력적이게 느껴졌습니다. 정말 하나의 세계를 창조하는 느낌이 들었고 앞으로 현업에서 일한다면 10년이고 20년이고 유저들에게 사랑받는 게임을 만들고 싶다는 생각이 깊어졌습니다.

3. 엔진 레벨에서 지원해주는 수학과 물리 등 논리적 사고가 필요한 작업들을 심도있게 공부해 보고 싶다 라는 마음이 깊어졌습니다. 가져다 사용하는 사람이 아닌 직접 만들 수 있는 개발자가 되기 위해 열심히 달리겠다고 마음을 가질 수 있던 프로젝트였습니다.

영상을 보고싶으시면 여기를 클릭해주세요

Name 김유민
Moblie 010 7480 0851
E-mail rladbals0129@gmail.com