



국민대학교 2019 캡스톤 프로젝트

# REVISION

Surreal (16조) – 김성훈, 김종균, 송영륜, 장윤지, 갈경달



국민대학교 2019 캡스톤 프로젝트

# REVISION

Surreal (16조) – 김성훈, 김종균, 송영륜, 장윤지, 갈경달



# INDEX

1. REVISION 이란?
2. 게임 기획 의도
3. 프로젝트 진행 방법 및 Art Concept
4. 수행 내용 및 기술 설명 (주요 기술)
5. 전체 배경 기술
6. 팀원 소개
7. 마치며

# REVISION 이란?

## SYNOPSIS

부상당한 주인공은 기억상실의 상태로 깨어난다.

옆 마을은 그리폰(보스 몬스터)으로 인해 폐허가 된 상태이다.

옆 마을이 파괴된 사건을 조사하면서, 사람들은 그리폰이 불로 된 동굴 안에 산다는 것을 알게 되고

그 서식처에는 매우 희귀한 광물인 크리스탈도 있다는 것을 알게 된다.

마을 사람들은 크리스탈에 대한 욕심으로 토벌대를 조직하여 그곳으로 모험을 떠나는데,

이에 주인공도 참여하며 게임이 진행된다.

# REVISION 이란?

## REVISION의 의미

RE(다시) + VISION(보다)

- > 우리 삶의 가치들을 되돌아 본다(게임의 의의와 연관)
- > 우리가 믿고 있는 사실들을 다시 본다(스토리과 연관)



# REVISION 이란?

-Unreal 엔진을 통해 만든 어드벤처 RPG 프로젝트이다. 기본 플랫폼은 PC이며, 키보드와 마우스로 게임 조작이 가능하다. 게임 진행 중에 퀘스트를 통해 사용자에게 자율성을 부여하여 선택할 수 있는 분기점을 만든다. 최종 보스를 물리치면 게임이 끝나게 되며 게임 중 선택해 온 선택 결과들에 의해 엔딩이 달라진다.

# REVISION 이란?

장르

어드벤처 RPG

플랫폼

PC

그래픽

High poly 3D

3인칭 자유 시점

사용 엔진

Unreal Engine 4.21

# 게임 기획 의도

## 목표

- 직관적이고 자유로운 조작 방식과 타격감 넘치는 전투 시스템, 그리고 흥미로운 시나리오를 통해 사용자에게 **재미를 준다.**
- 게임을 진행하며 플레이어가 생각하는 가치에 근거하여 선택을 해 나가야 하는 상황들을 통해, **개인이 살아가며 추구해야 할 가치를 되돌아 보는 시간**을 갖게 된다. 이는 내면 정비에 도움을 주어 게임의 재미 요소와 함께 **플레이어의 스트레스 경감에** 도움을 준다.



# 프로젝트 진행

The screenshot displays a Trello board for 'Unreal 캡스톤 디자인 - Surreal의 REVISION'. The board is organized into several columns representing different stages of the project:

- 공지사항 & 전달사항(Notice):** Contains cards about YouTube video uploads, role distribution, and team GitHub links.
- 해결해야할 Bug:** Lists bugs like monster movement and character selection issues.
- 해야하는 작업들(To do list):** Includes Level Design and Quest System tasks.
- 개발 하는중(Doing):** Shows progress on BP, Quest System, Inventory, and Character features.
- 개발 완료(Be done):** Lists completed tasks like Bug Fix, Interface, Dodge, and NPC dialogues.
- 개발 아이디어 & 의견 & 각Tip:** A column for ideas and tips, featuring a volcano image.
- 회의록(Meeting Minutes):** Contains meeting minutes for REVISION, including dates and topics.
- 참고 자료(Reference):** Lists references like UNC support methods and Unreal Engine 4.
- 전체 진행과정(누적):** A cumulative progress section with cards for weekly milestones (e.g., 5월 3주차 일정, 5월 2주차 일정, etc.).
- 임시 참고용:** A temporary reference section with cards for locomotion, movement, and pitch deck.

<Trello 사용>

# 프로젝트 진행

kookmin-sw / 2019-cap1-2019\_16

Watch 3 Star 3 Fork 0

Code Issues 35 Pull requests 0 Projects 1 Wiki Security Insights Settings

2019-cap1-2019\_16 / Unreal Engine / 3D Adventure RPG <https://kookmin-sw.github.io/2019-cap...> Edit

Manage topics

132 commits 22 branches 4 releases 1 environment 5 contributors


Branch: master New pull request Create new file Upload files Find File Clone or download

610ksh [프로젝트 예러 최종 수정] 컴파일 안되는 문제 최종적으로 해결함. -Saved, Intermediate 폴더 삭제후 Gen... Latest commit d3646a7 on 20 Apr

Docs	업로드용 pdf	a month ago
LostDark	[프로젝트 예러 최종 수정] 컴파일 안되는 문제 최종적으로 해결함. -Saved, Intermediate 폴더 삭제후 Gen...	a month ago
.gitignore	Add Unreal git ignore (gitignore 추가)	2 months ago
README.md	Trello link, 소개페이지 link 폰트수정	a month ago
_config.yml	Set theme jekyll-theme-cayman	2 months ago
index.md	링크 폰트 수정	a month ago

README.md

## 2019 소프트웨어 융합대학 캡스톤 디자인 I 16조 Surreal



**SURREAL**

<GITHUB 사용>

# 프로젝트 진행

The screenshot displays the 'OPEN DESIGN' website interface. At the top, there's a navigation bar with '디자인' (Design), '그룹' (Group), '디자이너' (Designer), and a red '디자인 등록' (Design Registration) button. A search bar and links for '로그인' (Login) and '회원가입' (Sign Up) are also present.

The main content area features a large header for a project titled '(16조) Unreal 엔진을 통한 액션 어드벤처 ...' (16th Group: Action Adventure through Unreal Engine...). The project is by '작성자 김성훈' (Author: Kim Seung-hoon) in the '카테고리 응용SW' (Category: Application SW) category. It has 745 views, 1 heart, 0 comments, and 2 tags. The creation date is '2019-03-08' and it was last updated 6 hours ago.

Below the header, there are five columns of project milestones:

- 기획** (Planning): Includes '프로젝트명' (Project Name) by 김성훈 (Kim Seung-hoon) 7 days ago, '프로젝트 목적' (Project Purpose) by 김성훈 (Kim Seung-hoon) 14 days ago, '프로젝트 목표' (Project Goal) by 김성훈 (Kim Seung-hoon) 14 days ago, and '프로젝트 필요성' (Project Necessity).
- 요구사항 분석** (Requirement Analysis): Includes '브레인스토밍 1' (Brainstorming 1) by 송영훈 (Song Young-hoon) 2 days ago and '브레인스토밍 2' (Brainstorming 2) by 송영훈 (Song Young-hoon) 2 days ago.
- 소프트웨어 설계** (Software Design): Includes '키보드' (Keyboard) by 장윤지 (Jang Yun-ji) 1 day ago, '조작 방법' (Control Method) by 장윤지 (Jang Yun-ji) 1 day ago, '배턴 플레이 화면' (Button Play Screen) by 장윤지 (Jang Yun-ji) 1 day ago, and 'Wireframe ppt' by 장윤지 (Jang Yun-ji) 1 day ago.
- 캐릭터 중간발표** (Character Mid-report): Includes '중간발표 PPT' (Mid-report PPT) by 김성훈 (Kim Seung-hoon) 28 days ago, '수행계획서 PDF' (Action Plan PDF) by 김성훈 (Kim Seung-hoon) 28 days ago, and '중간보고서 PDF' (Mid-report PDF) by 김성훈 (Kim Seung-hoon) 28 days ago.
- 시스템** (System): Includes '구체' (Concrete) by 김성훈 (Kim Seung-hoon) and '작업' (Work) by 김성훈 (Kim Seung-hoon).

The footer contains 'Copyright @ 2019 Open Design Inc.' and links for '사이트 소개' (Site Introduction), '이용약관' (Terms of Use), and '개인정보보호정책' (Privacy Policy).

<오픈 소스 디자인 홈페이지 활용>



# Art Concept





# Art Concept





# Art Concept



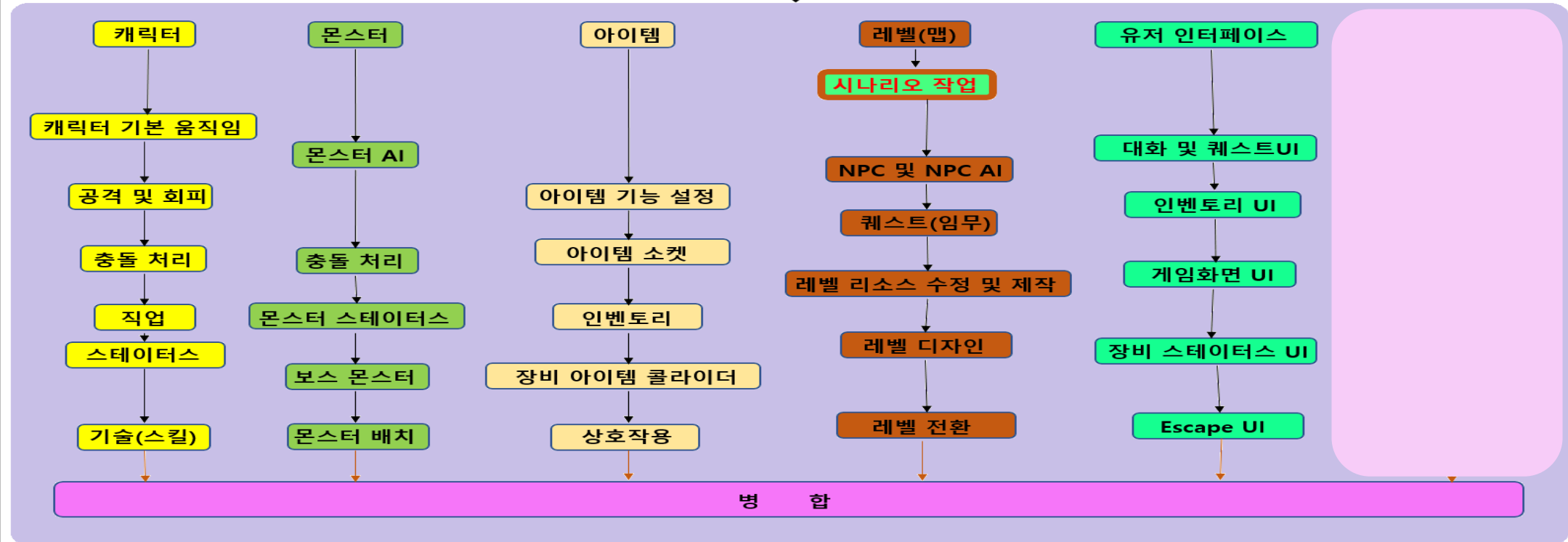


# Art Concept



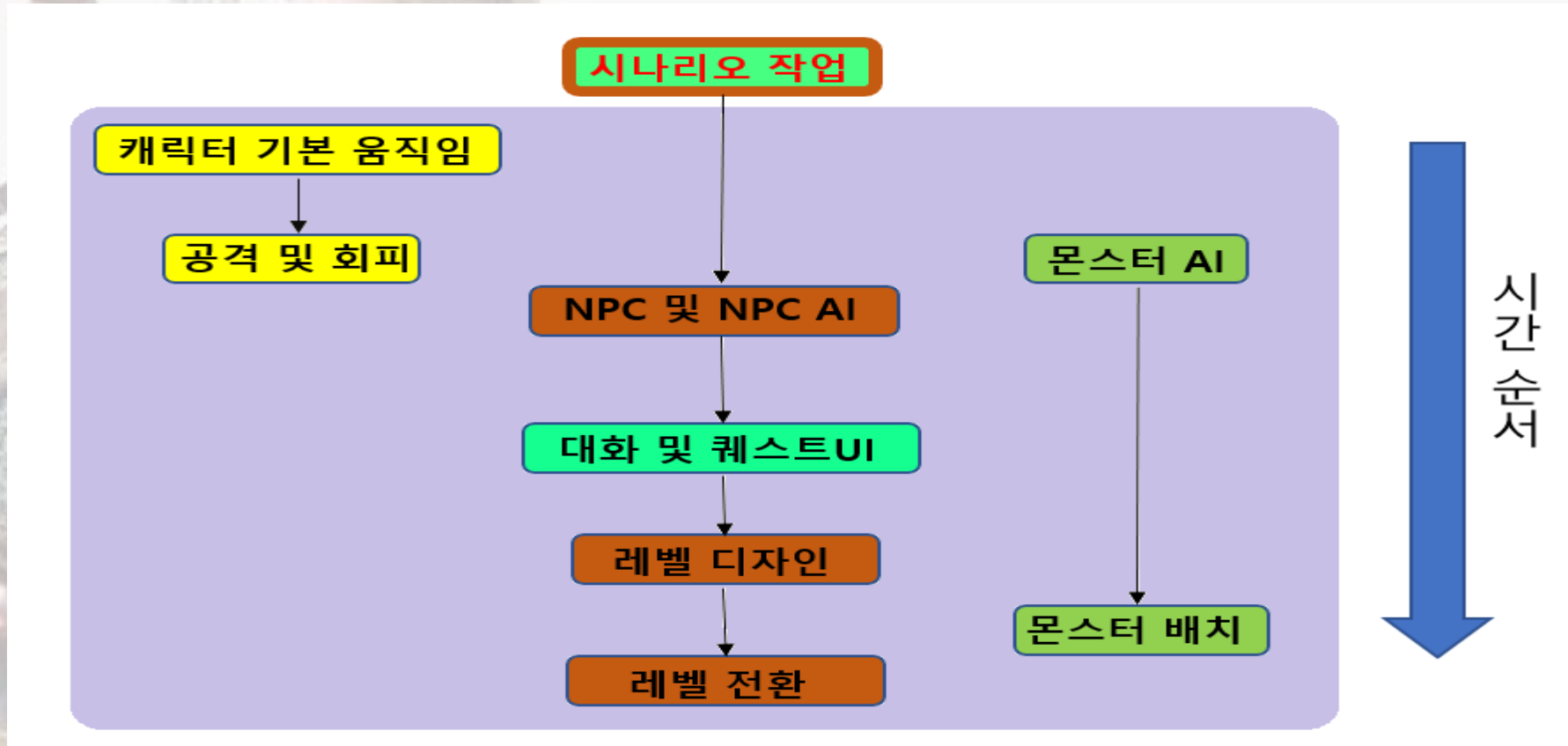
# 절차도

개발 시작



개발 완료

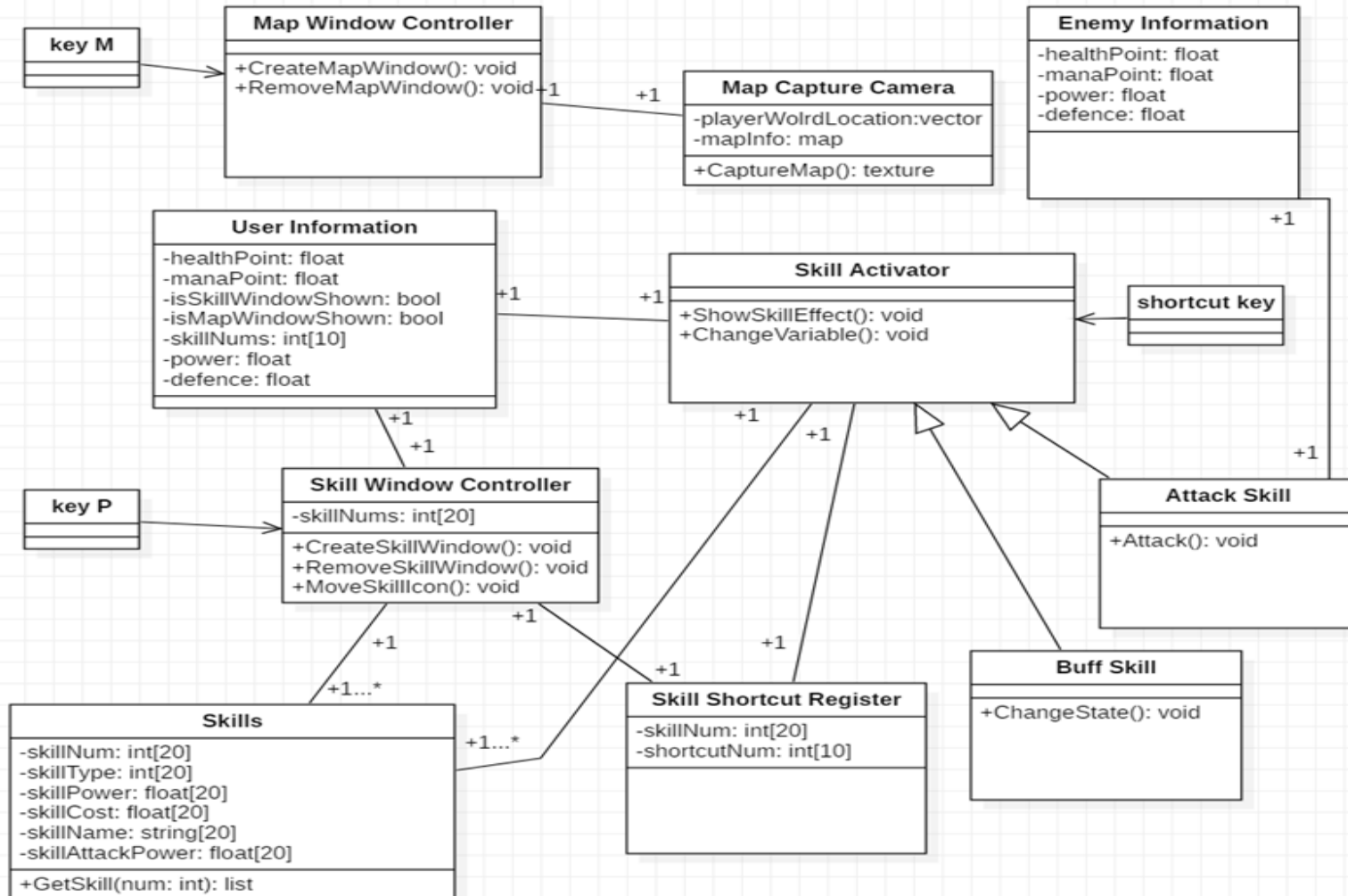
# 수행 내용(일부)



시나리오 작업이 먼저 선행되어야 게임의 방향성이 잡히고, 그 이후 플레이어가 조종할 캐릭터를 구현한다. 거의 동시에 플레이어와 대척점을 지닌 몬스터들을 만들고, 캐릭터에게 정보를 주거나 임무를 주는 NPC (Non-player Character)를 만든 다음, 만든 맵(레벨)에 몬스터와 NPC들을 적재적소에 배치한다.



# Class Diagram(UI)



User Interface를 도식화한 다이어그램이다.

가령 기술(Skills)에는  
기술 넘버(이름에 해당),  
기술의 타입(마법, 물리 등)  
기술이 소모하는 값(Cost)  
기술이 어떤 수치를 올려주는지

등이 변수로 들어간다.

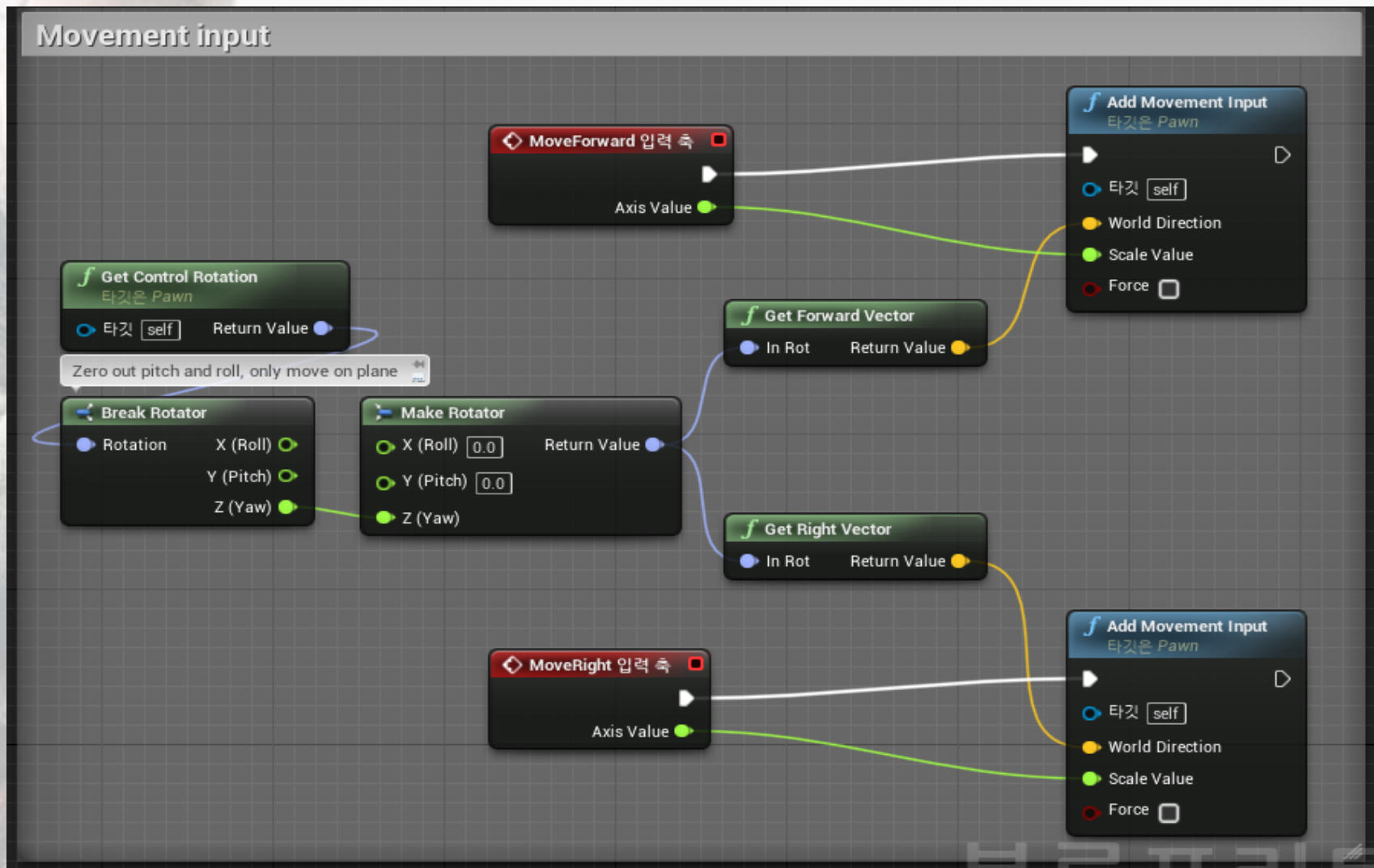
# 캐릭터 이동

## 캐릭터 기본 움직임

우측은 캐릭터 움직임에 대한 Blueprint 코드이다.

Blueprint는 Unreal Engine에서 제공하는 기능으로, C++코드에 쓰이는 명령어들을 노드형식으로 바꿔 코드의 진행방향을 알기 쉽게 만들어 놓은 장치이다.

C++에 비해 섬세한 기능을 구현하기는 힘들지만, 명령어를 검색하여 빨리 찾을 수 있고, 디버깅할 때 노드간의 연결을 통해 오류를 빨리 잡을 수 있다.



# 캐릭터 이동

```
ABPawn.h  ABPlayerController.h  ABCharacter.h  ABAnimInstance.h  ABGameMode.h  ABPawn.cpp  ABPlayerController.cpp  ABCharacter.cpp  ABAnimInstance.cpp  ABGameMode.cpp
ArenaBattle.h  AABCharacter
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "ArenaBattle.h"
6 #include "GameFramework/Character.h"
7 #include "ABCharacter.generated.h"
8
9 UCLASS()
10 class ARENABATTLE_API AABCharacter : public ACharacter
11 {
12     GENERATED_BODY()
13
14 public:
15     // Sets default values for this character's properties
16     AABCharacter();
17
18 protected:
19     // Called when the game starts or when spawned
20     virtual void BeginPlay() override;
21
22     // 새로 추가
23     enum class EControlMode
24     {
25         GTA,
26         DIABLO
27     };
28     // 컨트롤 모드 설정 함수
29     void SetControlMode(EControlMode NewControlMode);
30     // 열거형 변수 선언 및 초기화 (기본 GTA)
31     EControlMode CurrentControlMode = EControlMode::GTA;
32     // 방향 앞방향 시선값. UPROPERTY를 사용하지 않는 FVector는 항상 초기값을 미리 지정해야 안전하다.
33     // 만약 축 입력 이벤트가 발생하면, 해당 변수를 업데이트하고 Tick로직에서 이것을 이용해 처리한다.
34     FVector DirectionToMove = FVector::ZeroVector;
35
36     // 시점 변경할때 사용할 변수들
37     float ArmLengthTo = 0.0f;
38     FRotator ArmRotationTo = FRotator::ZeroRotator;
39     float ArmLengthSpeed = 0.0f;
40     float ArmRotationSpeed = 0.0f;
41
42 public:
43     // Called every frame
44     virtual void Tick(float DeltaTime) override;
45
46     // PostInitializeComponents 컴포넌트 초기화 이후, 액티비티 컴포넌트 초기화 완료 후 호출됩니다.
47
48 ArenaBattle.cpp
49 AABCharacter
50 1 // Fill out your copyright notice in the Description page of Project Settings.
51
52 #include "ABCharacter.h"
53 // 공격 애니메이션을 재생하라는 명령을 넣기 위해서.
54 #include "ABAnimInstance.h"
55 // 디버깅드로잉
56 #include "DrawDebugHelpers.h"
57
58 // Sets default values
59 AABCharacter::AABCharacter()
60 {
61     // Set this character to call Tick() every frame. You can turn this off to improve performance if
62     PrimaryActorTick.bCanEverTick = true;
63
64     // 컴포넌트 추가
65     SpringArm = CreateDefaultSubobject<USpringArmComponent>(TEXT("SPRINGARM"));
66     Camera = CreateDefaultSubobject<UCameraComponent>(TEXT("CAMERA"));
67
68     // 컴포넌트를 상속시킴.
69     SpringArm->SetupAttachment(GetCapsuleComponent());
70     Camera->SetupAttachment(SpringArm);
71
72     GetMesh()->SetRelativeLocationAndRotation(FVector(0.0f, 0.0f, -88.0f), FRotator(0.0f, -90.0f, 0.0f);
73     SpringArm->TargetArmLength = 400.0f;
74     SpringArm->SetRelativeRotation(FRotator(-15.0f, 0.0f, 0.0f));
75
76     static ConstructorHelpers::FObjectFinder<USkeletalMesh> SK_CARDBOARD(TEXT("/Game/InfinityBladeWarr:
77
78     if (SK_CARDBOARD.Succeeded())
79     {
80         GetMesh()->SetSkeletalMesh(SK_CARDBOARD.Object);
81     }
82
83     GetMesh()->SetAnimationMode(EAnimationMode::AnimationBlueprint);
84
85     static ConstructorHelpers::FClassFinder<UAnimInstance> WARRIOR_ANIM(TEXT("/Game/Book/Animations/War
86
87     if (WARRIOR_ANIM.Succeeded())
88     {
89         GetMesh()->SetAnimInstanceClass(WARRIOR_ANIM.Class);
90     }
91
92     // 최초 기본 모드는 DIABLO형식이다.
93     SetControlMode(EControlMode::DIABLO);
94
95     ArmLengthSpeed = 3.0f;
```



# 캐릭터 이동



이전 페이지의 코드를 토대로 동작을 생성. 꺾데기만 씌우면 코드대로 동작하도록 되어있음.

# 캐릭터 이동

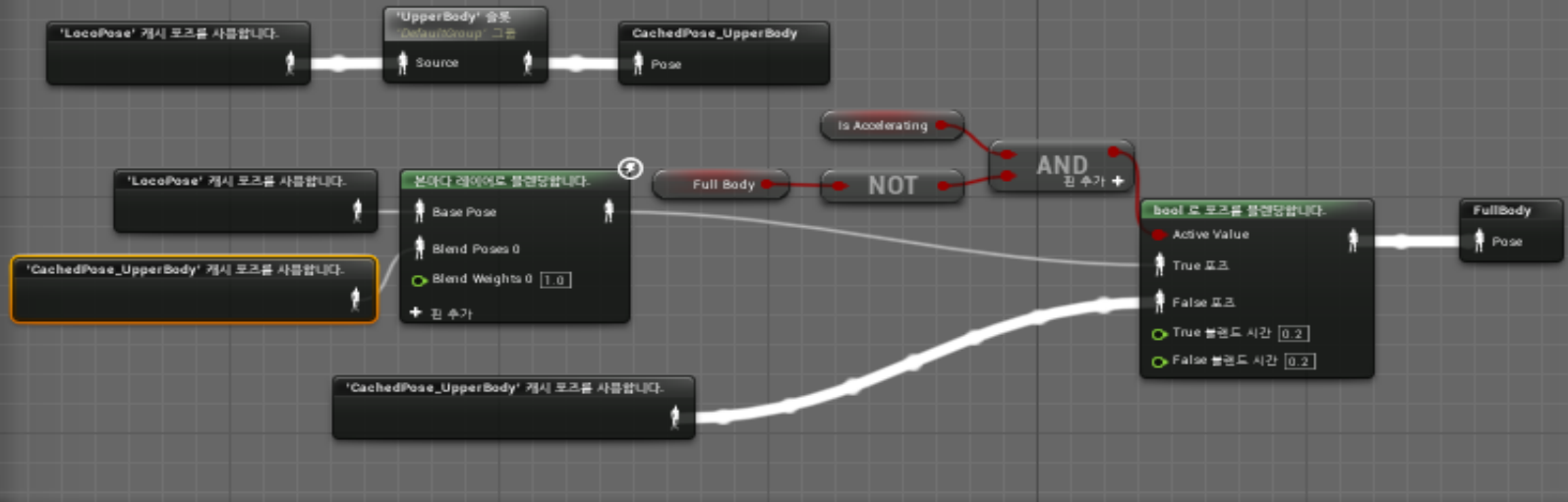


앞의 코드에 에셋(겍데기)을 멋진 전사로 바꿔 줌. 추가로 시점 조절기능까지 구현.

# 애니메이션

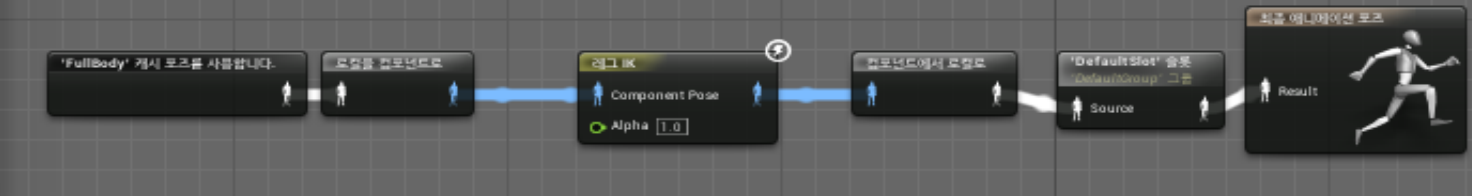
## Upper Body Layer

### Upper Body Layer



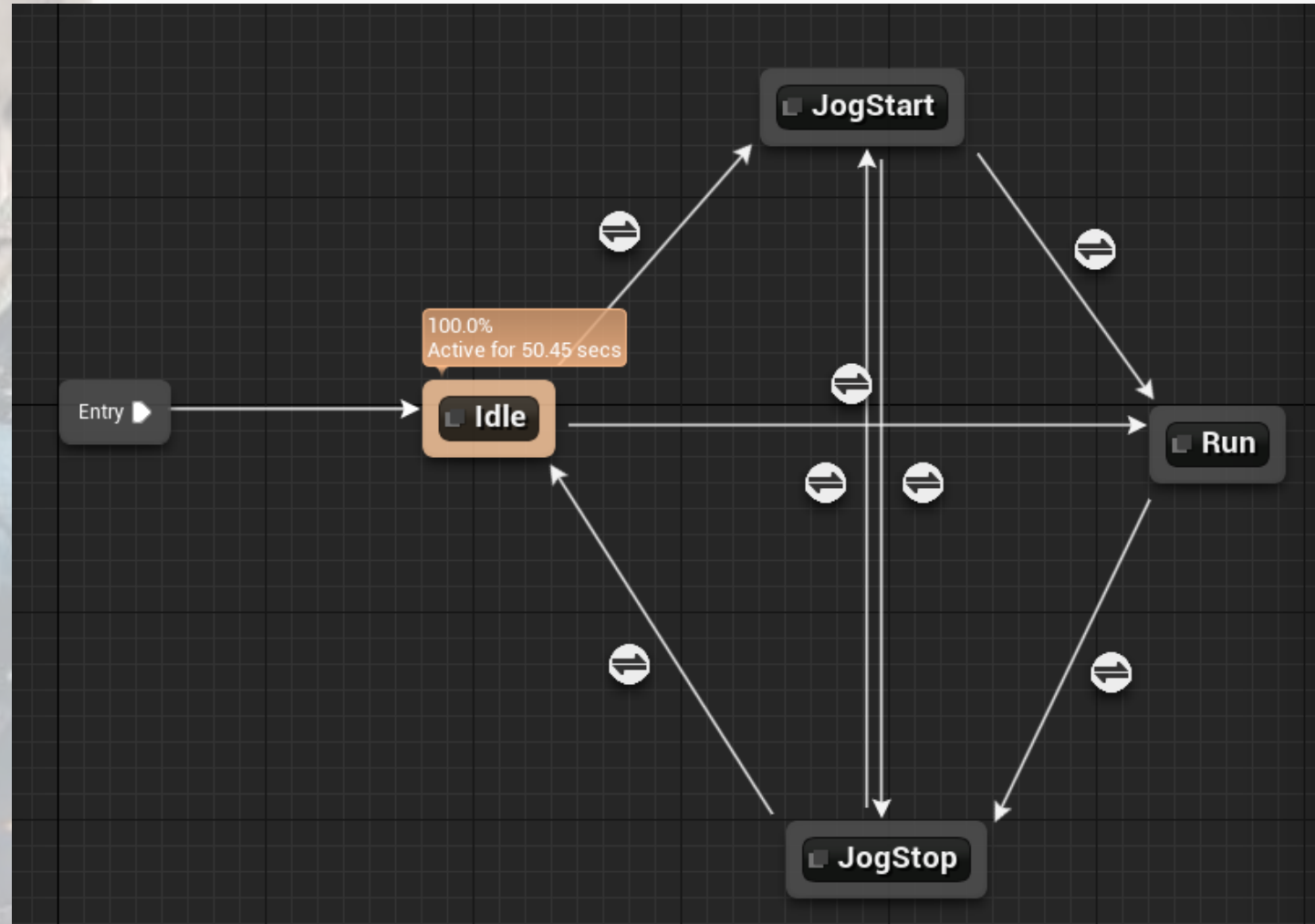
## Foot IK controls

### Foot IK controls





# 애니메이션



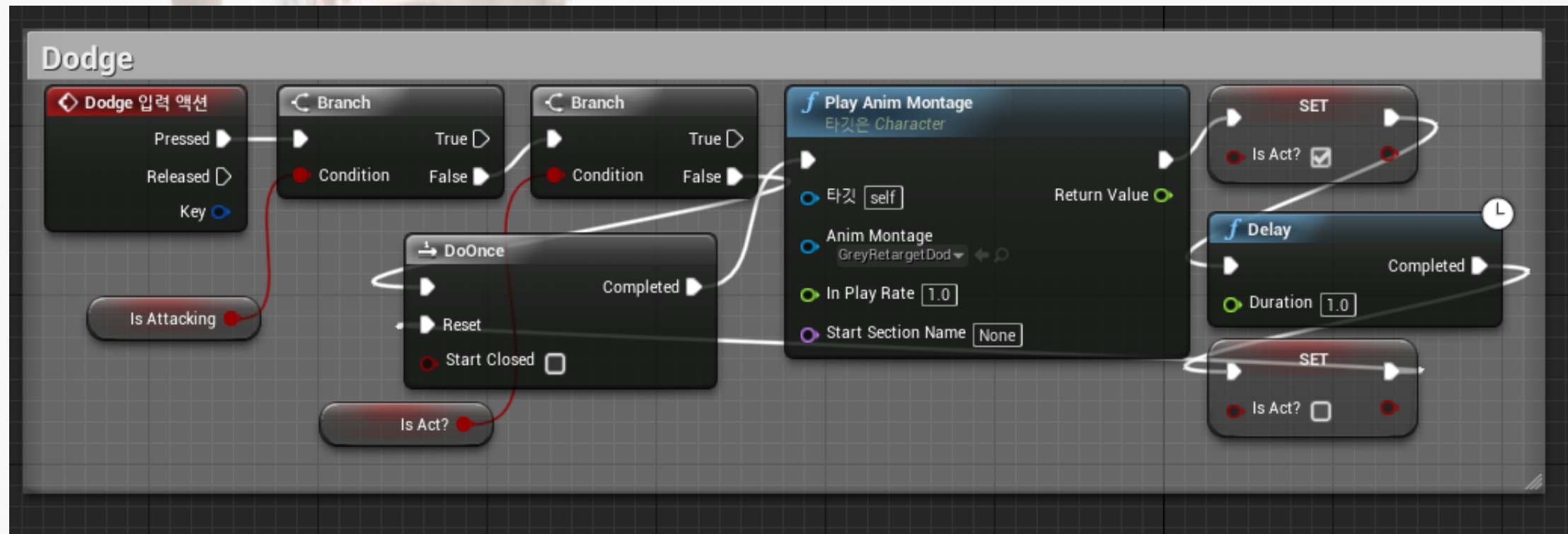
# 애니메이션

```
ABPawn.h  ABPlayerController.h  ABCharacter.h  ABAnimInstance.h  ABGameMode.h
ArenaBattle.h
ArenaBattle (전역 범위) DECLARE_MULTICAST_DELEGATE(FOnNextAttackCheckDelegate);
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "ArenaBattle.h"
6 #include "Animation/AnimInstance.h"
7 #include "ABAnimInstance.generated.h"
8
9 // 멀티캐스트 델리게이트 선언
10 DECLARE_MULTICAST_DELEGATE(FOnNextAttackCheckDelegate);
11 DECLARE_MULTICAST_DELEGATE(FOnAttackHitCheckDelegate);
12
13 /**
14  *
15  */
16 UCLASS()
17 class ARENABATTLE_API UABAnimInstance : public UAnimInstance
18 {
19     GENERATED_BODY()
20
21 public:
22     // 생성자 만들어주고
23     UABAnimInstance();
24     // tick 마다 호출하는 가상함수 선언.
25     virtual void NativeUpdateAnimation(float DeltaSeconds) override;
26     // 몽타주 플레이하는 함수
27     void PlayAttackMontage();
28
29     // 다음 몽타주 섹션으로 점프하는 함수
30     void JumpToAttackMontageSection(int32 NewSection);
31
32 public:
33     // 델리게이트
34     FOnNextAttackCheckDelegate OnNextAttackCheck;
35     FOnAttackHitCheckDelegate OnAttackHitCheck;
36
37     // 죽는 애니메이션 설정
38     void SetDeadAnim() { IsDead = true; }
39
40 private:
41     // 노티파이를 위한 함수.
42     UFUNCTION()
43     void AnimNotify_AttackHitCheck();
44
45     // 다음 공격 체크 노티파이
46     UFUNCTION()
```

```
ABPawn.cpp  ABPlayerController.cpp  ABCharacter.cpp  ABAnimInstance.cpp  ABGameMode.cpp
ArenaBattle.cpp
ArenaBattle UABAnimInstance NativeUpdateAnimation(float DeltaSeconds)
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #include "ABAnimInstance.h"
4
5 // 생성자
6 UABAnimInstance::UABAnimInstance()
7 {
8     // 초기 속도는 0이다
9     CurrentPawnSpeed = 0.0f;
10    // 초기 공중 상태는 false다.
11    IsInAir = false;
12    // 초기 죽었는지 변수는 false;
13    IsDead = false;
14
15    // 애니메이션 몽타주 등록
16    static ConstructorHelpers::FObjectFinder<UAnimMontage> ATTACK_MONTAGE(TEXT("/Game/Book/Animations/"));
17    // 경로에 애니메이션 몽타주가 유효하다면,
18    if (ATTACK_MONTAGE.Succeeded())
19    {
20        // 애니메이션 몽타주 변수에 해당 몽타주 애니메이션 정보를 등록한다.
21        AttackMontage = ATTACK_MONTAGE.Object;
22    }
23 }
24
25 // tick마다 호출되는 함수. 즉, Tick과 거의 동일하다고 봐도 무방할듯.
26 void UABAnimInstance::NativeUpdateAnimation(float DeltaSeconds)
27 {
28     Super::NativeUpdateAnimation(DeltaSeconds);
29
30     // Pawn 객체 생성, 폰에 접근해 속력값을 가져올때 사용함.
31     auto Pawn = TryGetPawnOwner();
32
33     // 폰에 접근하지 못했던면 반환
34     if (!IsValid(Pawn)) return;
35
36     // 죽지 않았다면
37     if (!IsDead)
38     {
39         // 애니메이션스턴스의 프로퍼티(멤버변수)값에 폰의 현재속력을 업데이트 시킨다.
40         CurrentPawnSpeed = Pawn->GetVelocity().Size();
41
42         // 임시 캐릭터 변수를 만들어 우리가 만든 캐릭터 클래스 정보를 넘겨주고,
43         auto Character = Cast<ACharacter>(Pawn);
44         // 캐릭터가 유효하다면,
45         if (Character)
46         {
```



# 회피 기능



# 회피(구르기) 기능



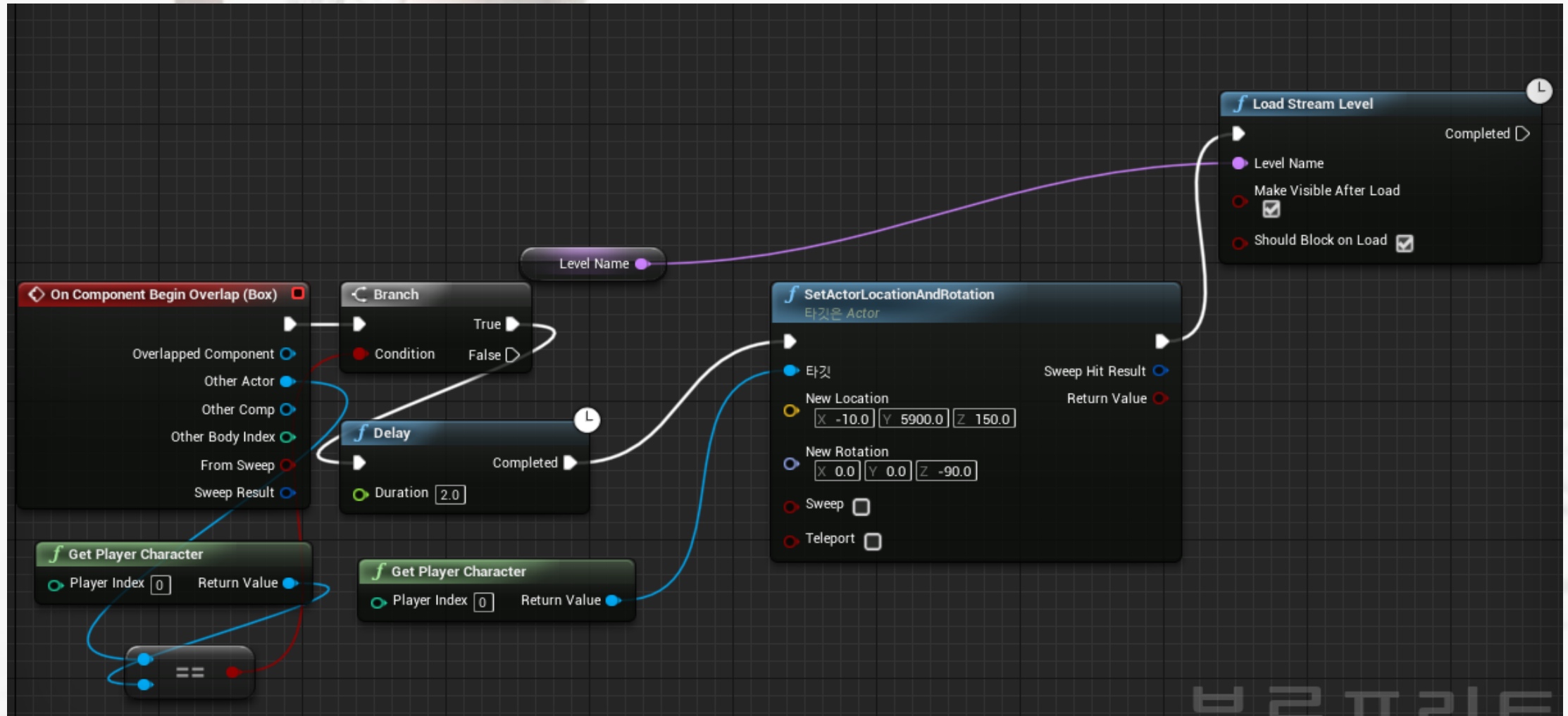


# 회피(구르기) 기능



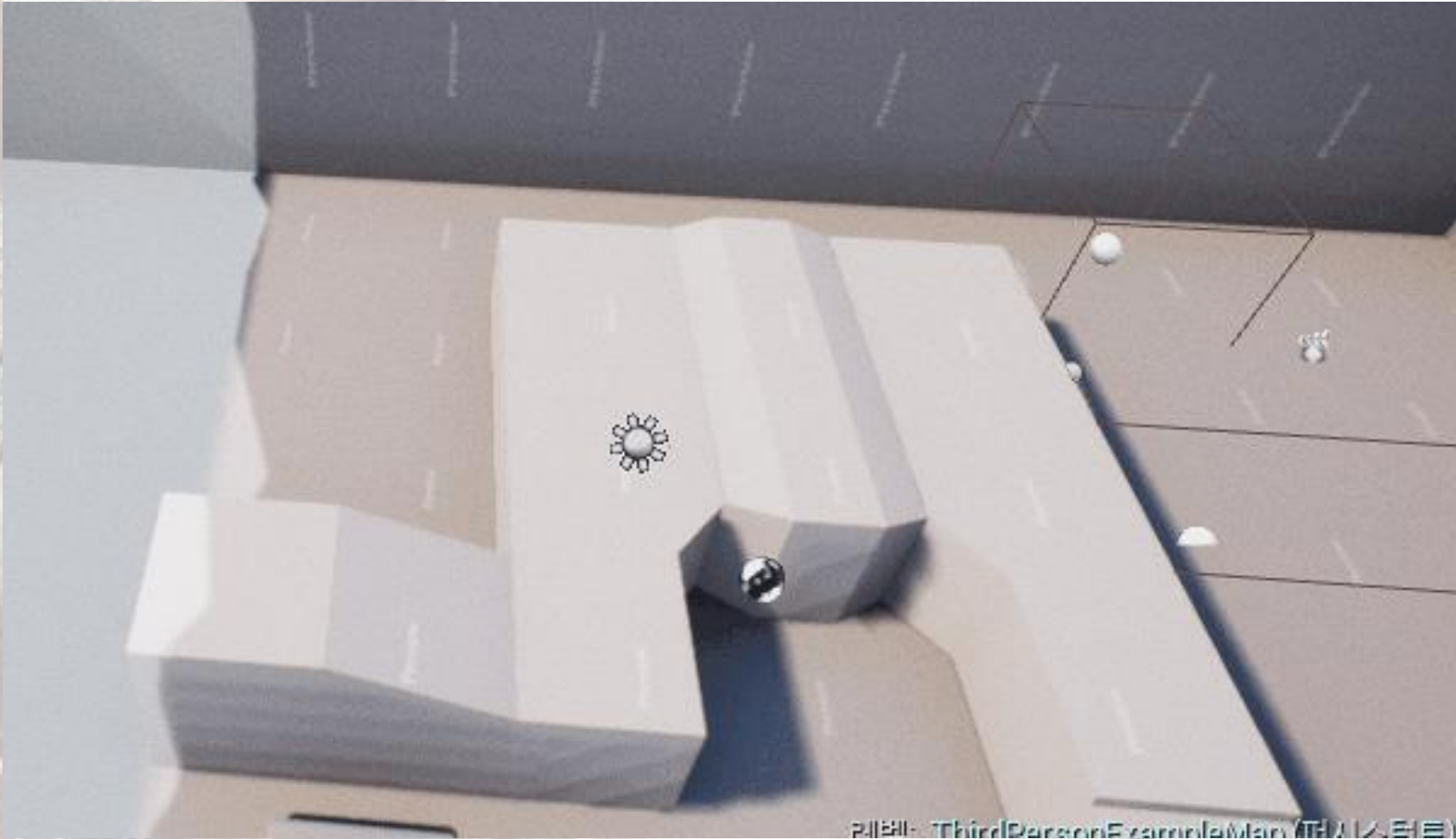
앞의 구르기에 멋진 기사 에셋(껍데기)을 씌워준 다음' 캐릭터가 직접 앞으로 이동' 기능을 추가했다.

# 레벨 전환 (맵 이동)





# Fade in, Fade out



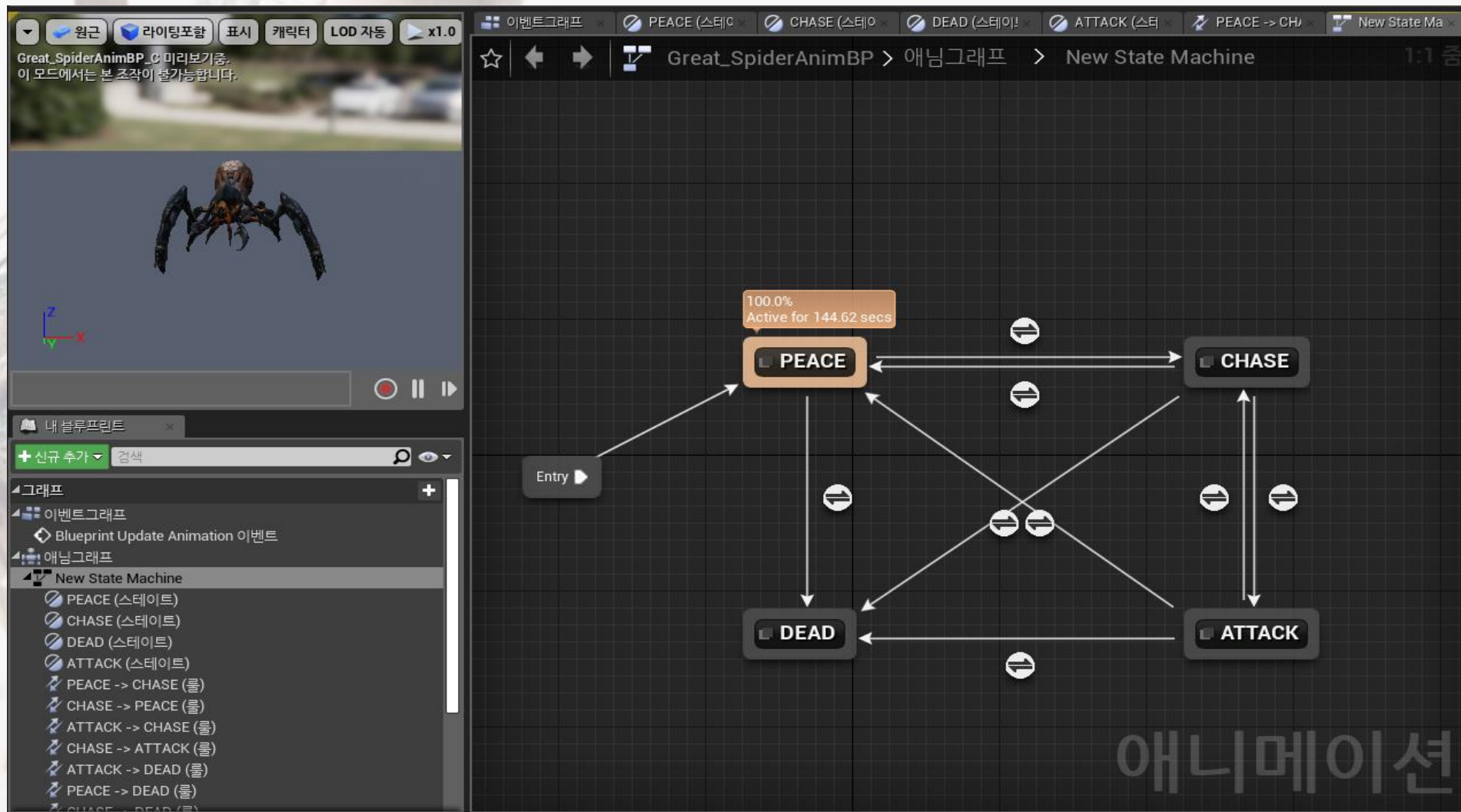
PIER: ThirdPersonExampleMain (THU 12:51E)

# 연출



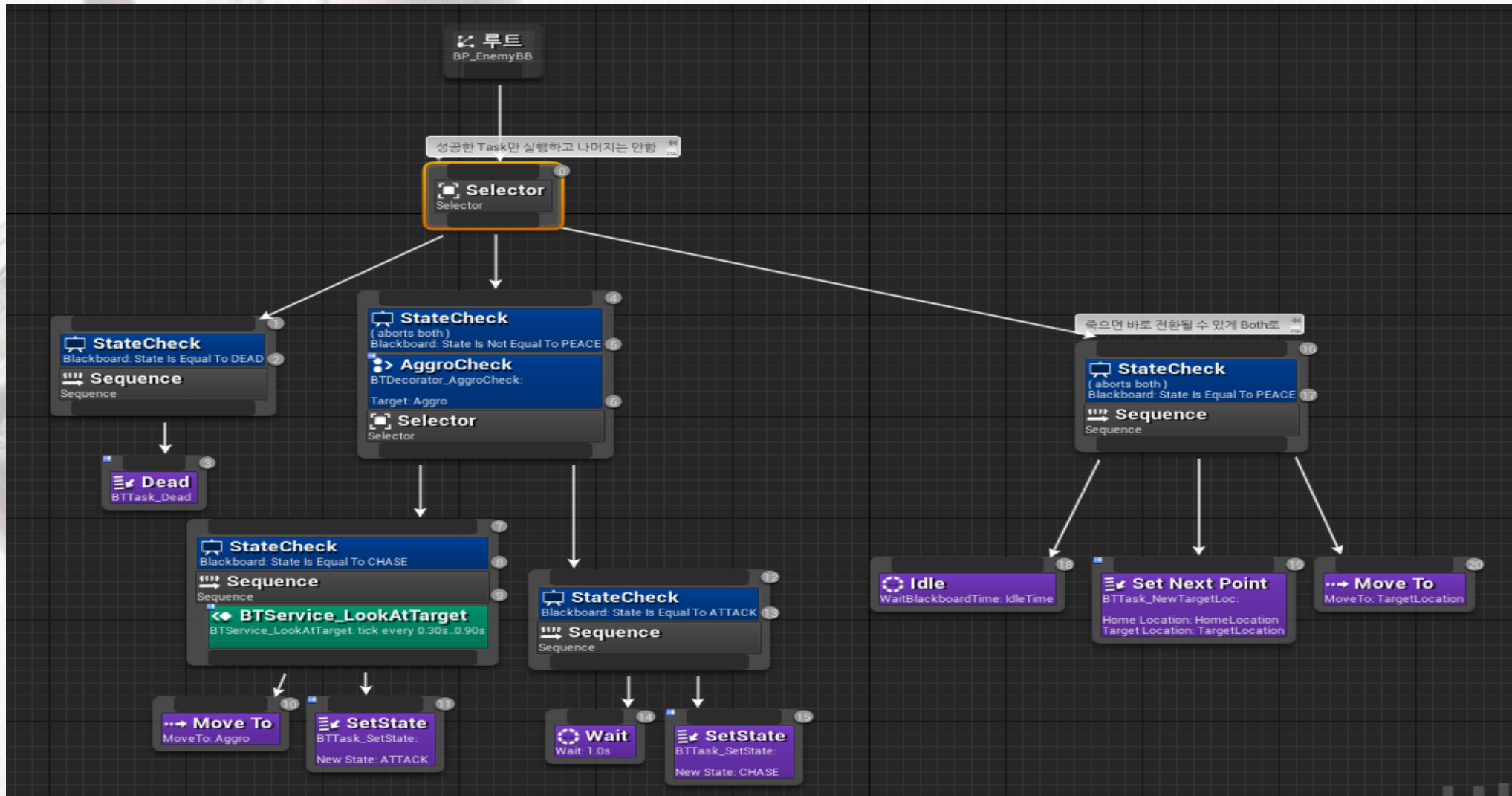
유저가 게임에 몰입하도록 하기 위해서는 중간중간 영화와 같은 연출이 필요하다.  
카메라 시점의 전환, 화면의 fade in fade out과 같은 기술들을 통해 몰입감을 극대화한다.

# Monster 구현

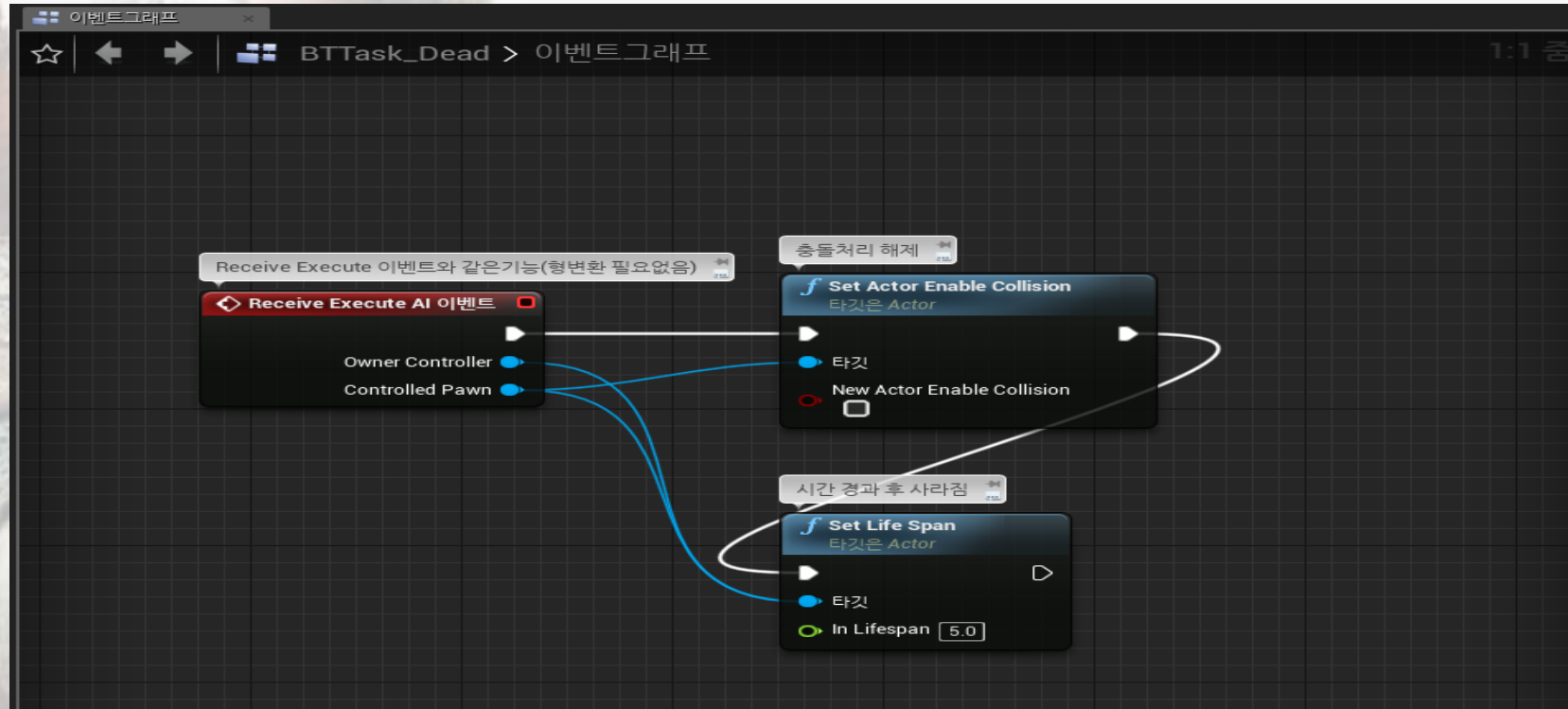




# Behavior Tree

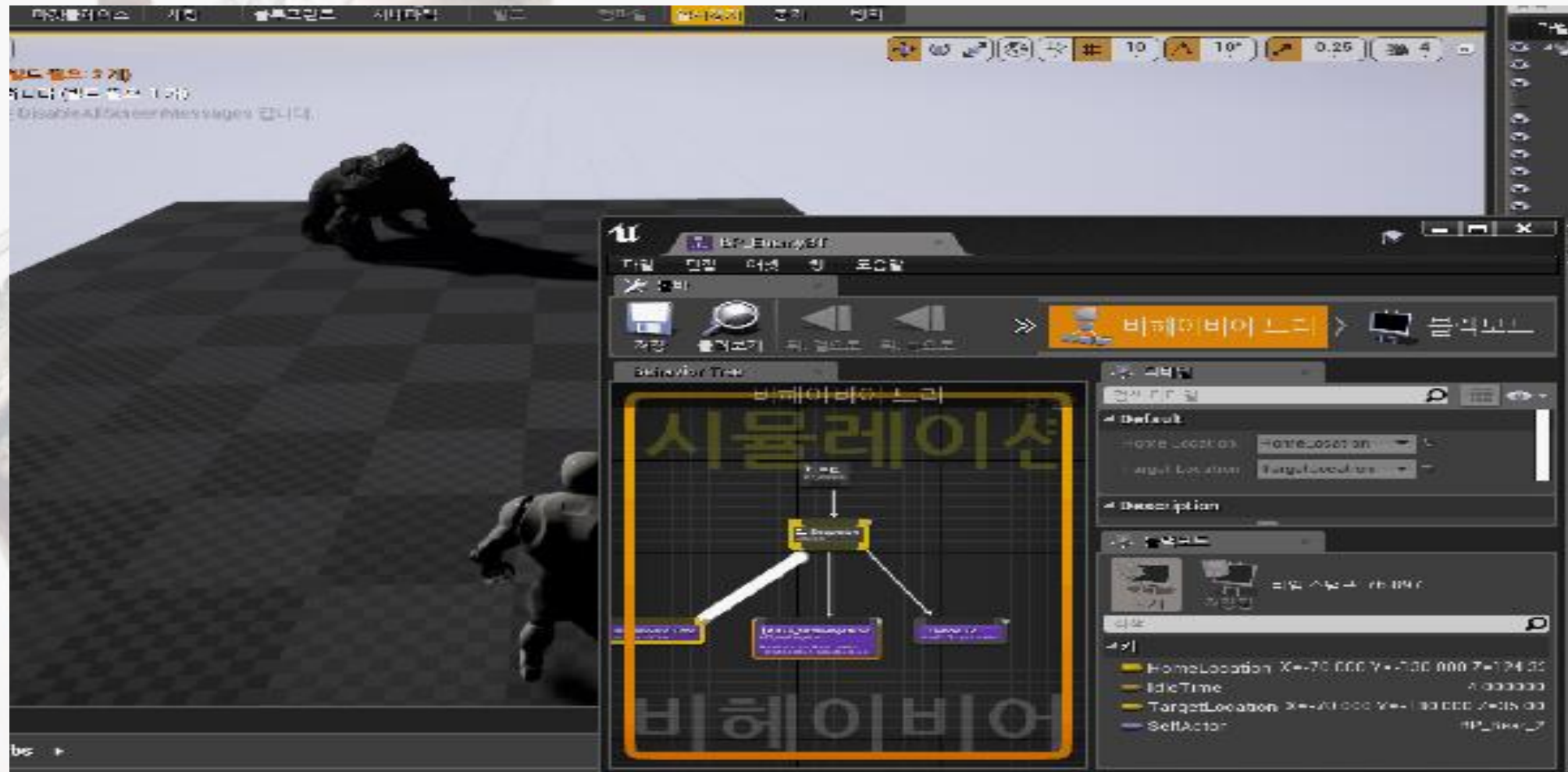


# Behavior Tree



Behavior Tree란 플레이어가 조종하지 않는 캐릭터의 인공지능을 구현하기 위한 기술이다. 예를들면, 몬스터의 경우 몬스터의 시야에 플레이어가 없으면 랜덤으로 움직이다가, 몬스터가 플레이어를 발견하면 공격 태세에 들어가고 플레이어를 따라간다. 플레이어가 몬스터 시야에서 몇 초간 사라지면 몬스터는 다시 기본 상태(Idle)로 돌아간다. 이러한 것들을 구현하기 위한 기술이 바로 Behavior Tree이다.

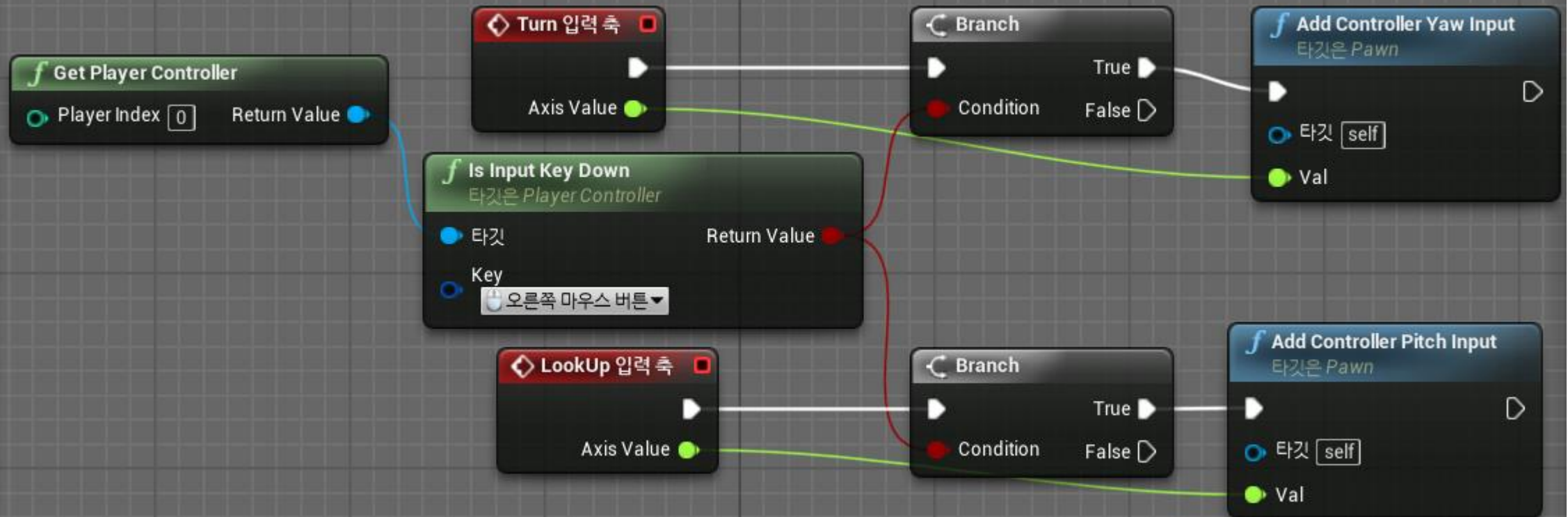
# Behavior Tree





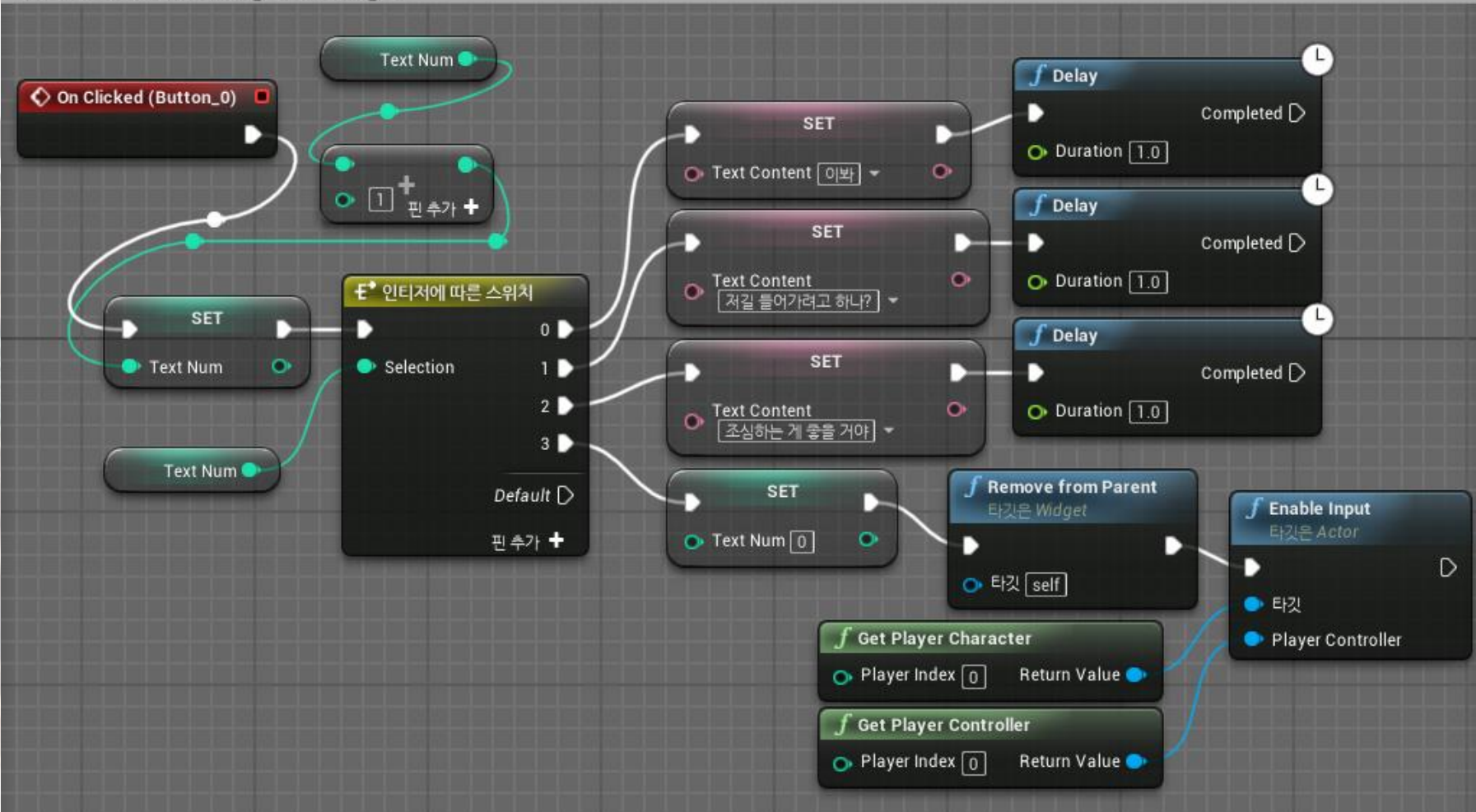
# Mouse 시점 변환

## Mouse rotation input

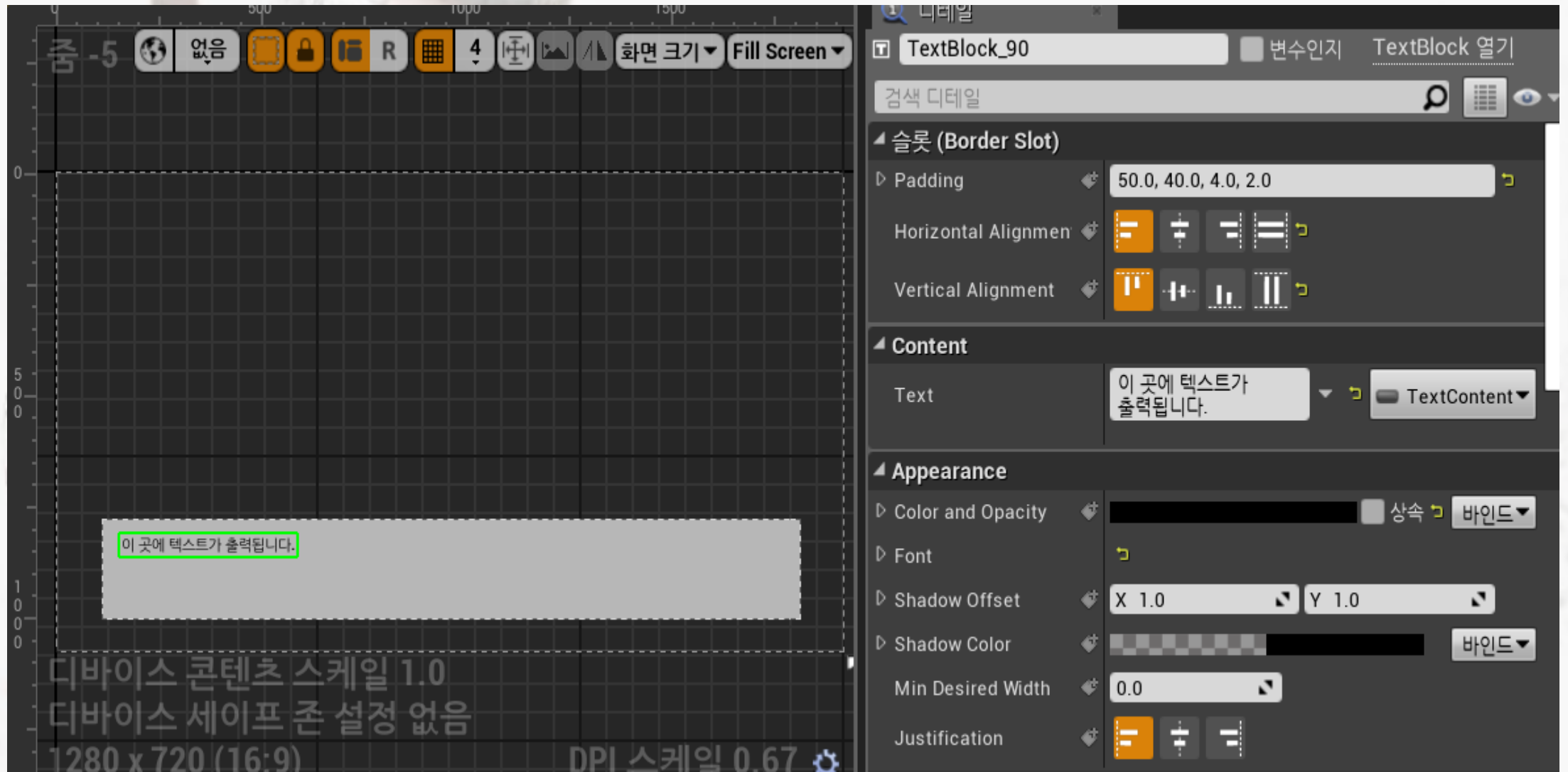


# 대화창 기능

Event for message change



# 대화창 기능

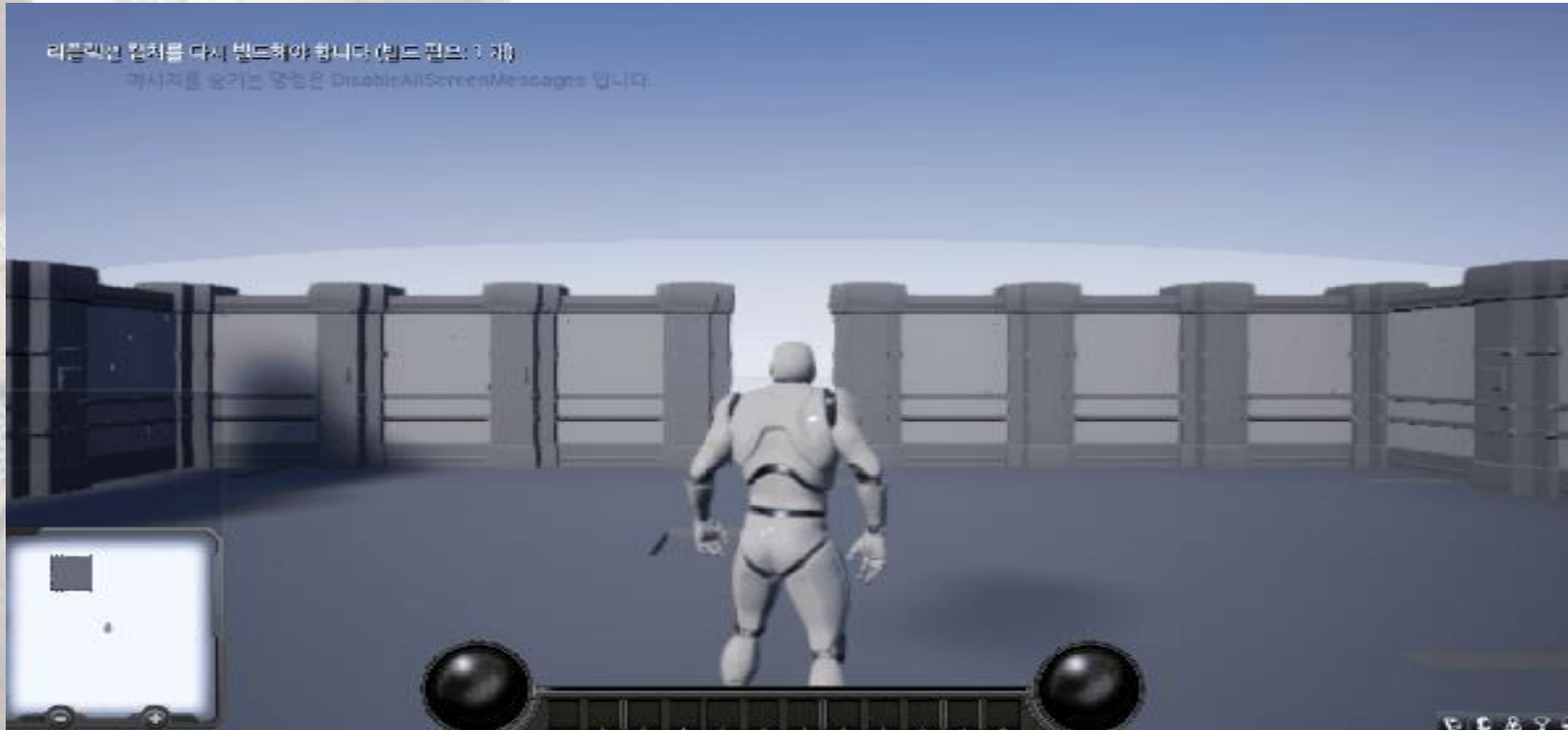




# 대화창 기능



# 캐릭터 UI



캐릭터의 정보를 숫자로 표시하는 것은 정확한 수치이지만, 몰입감을 저해한다. 캐릭터의 중요한 정보인 Health Point(체력)등의 정보는 Interface화 하여 그림으로 표현하는게 좋다. 좌측 하단에는 작은 지도인 미니맵이 표시되어 있고, 스킬(기술)창을 통해 내가 사용할 기술들을 퀵 슬롯에 넣어 사용하도록 한다.

# 상호 작용



앞서 만든 몬스터와 캐릭터 사이의 상호작용에 대한 영상이다. 앞에서 만든 코드들만으로는 부족하고 몬스터와 캐릭터가 닿았을 경우, 누가 누구에게 먼저 의도를 가지고 공격을 했는지, 서로의 공격력과 Health Point(체력)은 얼마나 되는지, 그냥 공격이 아니라 기술을 사용하는지, 전부 분기를 만들어 놓는다.



# 시연 영상



# 배경 기술

## 1. C++과 자료구조, 알고리즘에 대한 이해 ( 프로그래머 )

- 기본적으로 Unreal Engine은 C++을 기반으로 프로그래밍 하게 되어있다. 이때 캐릭터를 움직이고 몬스터를 캐릭터의 행동에 따라 반응하게 하는 과정들은 자료구조와 알고리즘에 대한 이해가 없으면 쉽게 만들기가 어렵다.

## 2. Unreal Engine과 BluePrint에 대한 이해 ( 게임 기획자 및 프로그래머 )

- 코드를 사용하긴 하지만 게임엔진 위에 코드를 빌드해서 사용하는 것이므로, 엔진 자체에 대한 이해가 필요하다. 그 중에서도 BluePrint는 C++코드를 블럭화(노드화)해서 게임에서 쓰이는 명령어들을 찾아보기 쉽게 만들어 놓은 것이므로 반드시 각 노드의 기능과 노드 연결 과정에 대해 알고 있어야 한다.

## 3. Animation과 Anim BluePrint, Anim State Machine에 대한 이해 ( Unreal 애니메이터 )

- 애니메이션은 디자이너가 만들어 주지만, 만들어준 애니메이션을 어느 상황에서 어떤 때 출력이 되는지는 프로그래머가 정한다. 그 애니메이션 과정들을 Anim BluePrint와 Anim State Machine을 통해 정한다. 예를들어 Anim State Machine을 통해 가만히(idle), 걷기(walk), 뛰기(run) 상태를 만들고 해당 동작들에 애니메이션을 연결한 다음, 걷기 뛰기 가만히 상태에서 공격이나 방어를 하게되면 Anim BluePrint에 분기점을 만들어 두어, 해당 애니메이션들을 연결한다. 그렇게 하면 걸으면서 공격, 뛰면서 공격 등의 애니메이션이 겹치지 않고 실행된다.

## 4. User Interface와 Widget BluePrint에 대한 이해 ( UI 디자이너 및 개발자 )

- 게임은 외적인 요소가 매우 중요하다. 변수로 Health Point 1000을 만들어봤자, 사람들이 직관적으로 느끼기에는 충분하지 않다. 그래서 UI 디자이너가 게임 분위기에 맞게 인터페이스를 디자인하고, 개발자가 디자인 된 인터페이스를 프로그래밍한 변수들에 연결해주어 변수의 값들이 그림으로 표현이 되게끔 프로그래밍을 한다.



# 배경 기술

## 5. Texture와 Material, Light 그리고 Level Design에 대한 이해 ( 3D 그래픽 & 레벨 디자이너 )

- 게임은 글로 된 시나리오를 화면으로 출력해서 보여주는 결과물이다. 시나리오에 따라 게임의 분위기가 정해지고 그에 따라 레벨(소위 Map이라고 부름)을 설계하여 플레이어가 활동할 세계를 만들어준다. 내가 구축한 세계관이 무엇 이냐에 따라 게임의 분위기가 달라지고, Light의 색이나 밝기가 달라지며, 집이냐 밖이냐 해변이냐 에 따라 화면의 질감을 달리 해야 한다. Ultraviolet의 강도, 특정 질감과 특정 질감을 적절히 배합하거나, 광택에 따라 Metallic 값을 추가하기도 하고, 그 외에 normal, roughness, opacity, refraction 등의 값을 주어 새로운 질감을 만들어내 적용한다.

## 6. Cinematic Sequencer, Matinee에 대한 이해 ( 연출 디자이너 )

- 게임은 유저가 직접 플레이하여 진행하는 프로그램이다. 유저가 몰입하여 게임할 수 있게끔 만들어주는 여러 작업 들이 있는데, 그 중에서도 유저의 마음을 사로잡는 기술은 유저가 빠져들 수밖에 없게 만드는 영화같은 '연출' 작업이다. 위 기능들은 Unreal에서 제공하는 기능들인데, 이미 만든 캐릭터와 주변 인물들, 그리고 몬스터와 레벨(Map)을 이용해 마치 영화의 한 장면처럼 카메라를 이용해 촬영하고, 캐릭터를 움직이며 특수한 효과( 슬로우 모션, Fade in out 등)를 준다. 게임의 몰입도는 매우 중요하기 때문에 Cinematic 연출 디자이너가 따로 작업을 한다.

## 7. Machine Learning과 Artificial Intelligence 그리고 Behavior Tree ( AI 개발자 )

- 게임에는 플레이어외에 게임 세상에서만 존재하는 액터들이 있다. 예를들어 몬스터나 NPC와 같은 액터들은 플레이어가 조종하는게 아니라 게임상에서 특정 알고리즘들을 통해 행동을 하게 되어있다. 단순히 걷거나 뛰는 액터들이 있는 반면, 캐릭터의 행동에 따라 유기적으로 반응하는 액터들도 있다. 그때 인공지능 알고리즘들을 이용하여 AI를 구축한다. Behavior Tree는 AI를 노드화해서 좀 더 쉽게 구현 가능하도록 만들어 주는 기능이다.



# 팀원 사진



[좌측부터 김종균, 송영륜, 갈경달, 장윤지, 김성훈]

# 팀원 사진

이름	개발 내용
김성훈 (팀장)	-Character Move, Attack, Dodge C++ 개발 각 코드에 Animation 적용 -Character Skill 구현 및 게임 시스템(시점, 캐릭터 회전 등) 개발
김종균	-Level 및 Landscape Design, Inventory UI 개발 및 기능 구현 -Scene 전환 및 Cinematic 연출, Sound Que 구현
송영륜	-Enemy AI 개발 및 Animation 적용, Enemy 종류별 구현 -Boss Monster AI & Skill 구현, Enemy Attack 구현
장윤지	-Camera Moving 및 시점 변환, Text UI, NPC 상호작용 -Quest 및 캐릭터 UI(캐릭터 정보, 미니맵, 스킬) 코드 연동 및 기능 구현
갈경달	-시나리오 중국어 번역 및 게임 Tester
<공통>	-게임 시나리오 작성 및 컨셉 회의 -전체적인 작업 디버깅 및 병합





감사합니다