

목차

1. 기본 본을 리타겟하기 위한 표를 본다
2. 언리얼에 들어가서 믹사모에서 가져온 모델의 스켈레톤에 들어가서 리타겟을 위한 작업을 한다.
3. 리타겟 할 대상에 리타겟을 한다.(애니메이션, 애님 몽타주, 애니메이션 블루프린트 등이 가능)
4. PublicCharacter를 상속받은 EnemyCharacter를 생성한다.
5. 열거형 CharacterState를 생성한다.
6. EnemyCharacter 처리 1
7. ABAnimInstance 처리
8. PublicCharacter 처리
9. EnemyCharacter 처리 2



ANIMATION RE-TARGETING WITH UNREAL USING MIXAMO ASSETS

By Wojtek | November 10, 2018 | Development, Tutorial

 0

 4

 9 Comments

<http://jollymonsterstudio.com/2018/11/10/animation-re-targeting-with-unreal-using-mixamo-assets/>

0. 리타겟 출처

UNREAL - BASE	MIXAMO
Root	Hips
Pelvis	Hips
spine_01	Spine
spine_02	Spine1
spine_03	Spine2
clavicle_l	LeftShoulder
UpperArm_L	LeftArm
lowerarm_l	LeftForeArm
Hand_L	LeftHand
clavicle_r	RightShoulder
UpperArm_R	RightArm
lowerarm_r	RightForeArm
Hand_R	RightHand

1. 기본 본을 리타겟하기 위한 표1

Hand_R	RightHand
neck_01	Neck
Head	head
Thigh_L	LeftUpLeg
calf_l	LeftLeg
Foot_L	LeftFoot
Thigh_R	RightUpLeg
calf_r	RightLeg
Foot_R	RightFoot

1.1 기본 본을 리타겟하기 위한 표2

UNREAL - ADVANCED	MIXAMO
index_01_I	LeftHandIndex1
index_02_I	LeftHandIndex2
index_03_I	LeftHandIndex3
middle_01_I	LeftHandMiddle1
middle_02_I	LeftHandMiddle2
middle_03_I	LeftHandMiddle3
pinky_01_I	LeftHandPinky1
pinky_02_I	LeftHandPinky2
pinky_03_I	LeftHandPinky3
ring_01_I	LeftHandRing1
ring_02_I	LeftHandRing2

1.2 고급 본을 리타겟 하기 위한 표1

ring_03_l	LeftHandRing3
thumb_01_l	LeftHandThumb1
thumb_02_l	LeftHandThumb2
thumb_03_l	LeftHandThumb3
lowerarm_twist_01_l	
upperarm_twist_01_l	
index_01_r	RightHandIndex1
index_02_r	RightHandIndex2
index_03_r	RightHandIndex3
middle_01_r	RightHandMiddle1
middle_02_r	RightHandMiddle2
middle_03_r	RightHandMiddle3

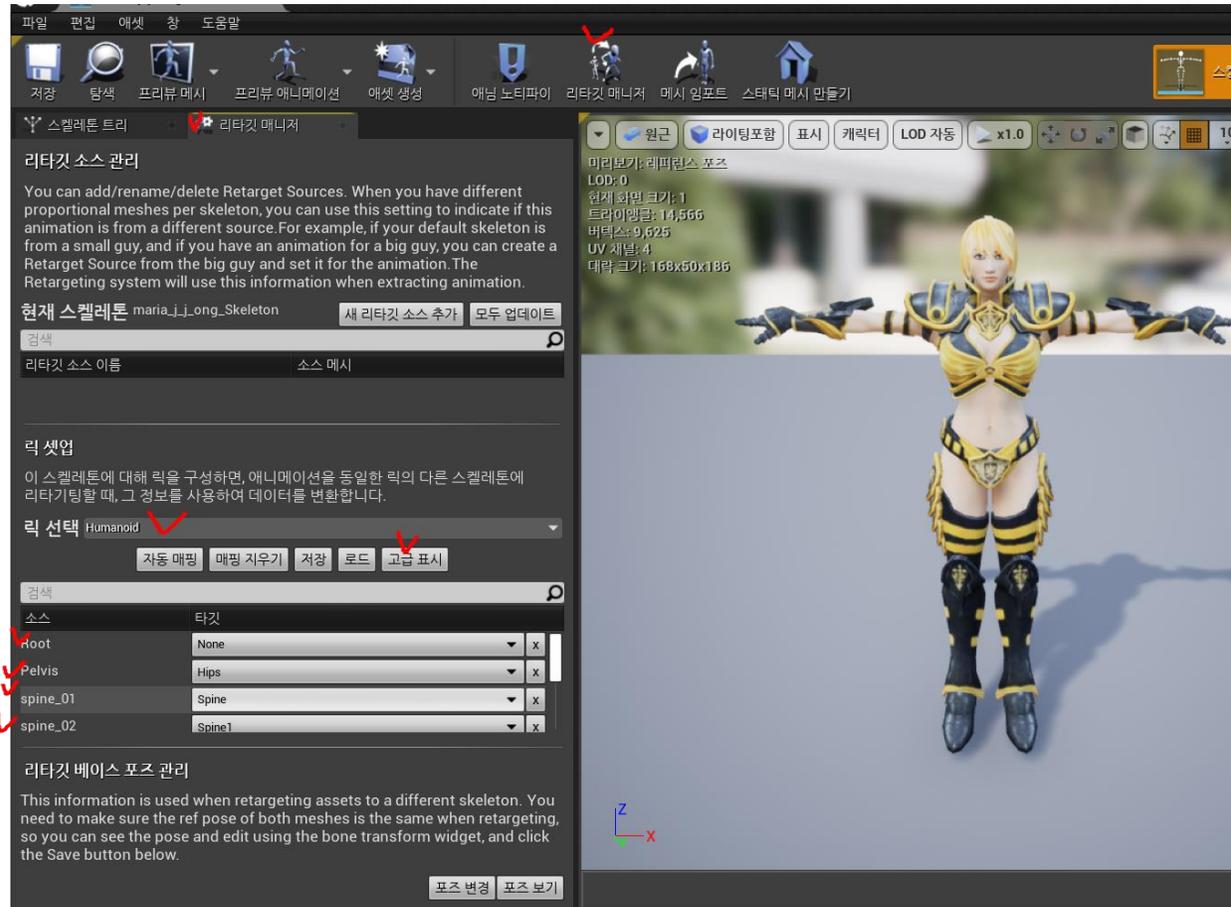
1.3 고급 본을 리타겟 하기 위한 표2

pinky_01_r	RightHandPinky1
pinky_02_r	RightHandPinky2
pinky_03_r	RightHandPinky3
ring_01_r	RightHandRing1
ring_02_r	RightHandRing2
ring_03_r	RightHandRing3
thumb_01_r	RightHandThumb1
thumb_02_r	RightHandThumb2
thumb_03_r	RightHandThumb3
lowerarm_twist_01_r	
upperarm_twist_01_r	
calf_twist_01_l	

1.4 고급 본을 리타겟 하기 위한 표3

thumb_02_r	RightHandThumb2
thumb_03_r	RightHandThumb3
lowerarm_twist_01_r	
upperarm_twist_01_r	
calf_twist_01_l	
ball_l	LeftToeBase
thigh_twist_01_l	
calf_twist_01_r	
ball_r	RightToeBase
thigh_twist_01_r	
ik_foot_root	
ik_foot_l	
ik_foot_r	

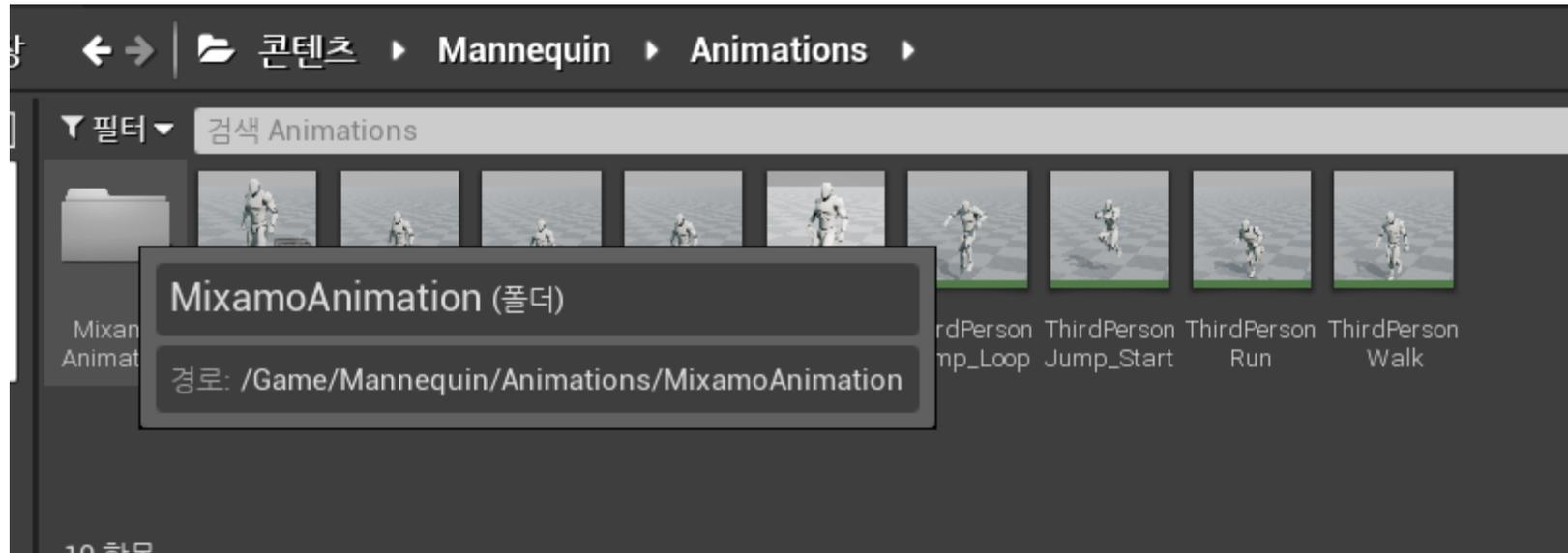
1.5 고급 본을 리타겟 하기 위한 표4



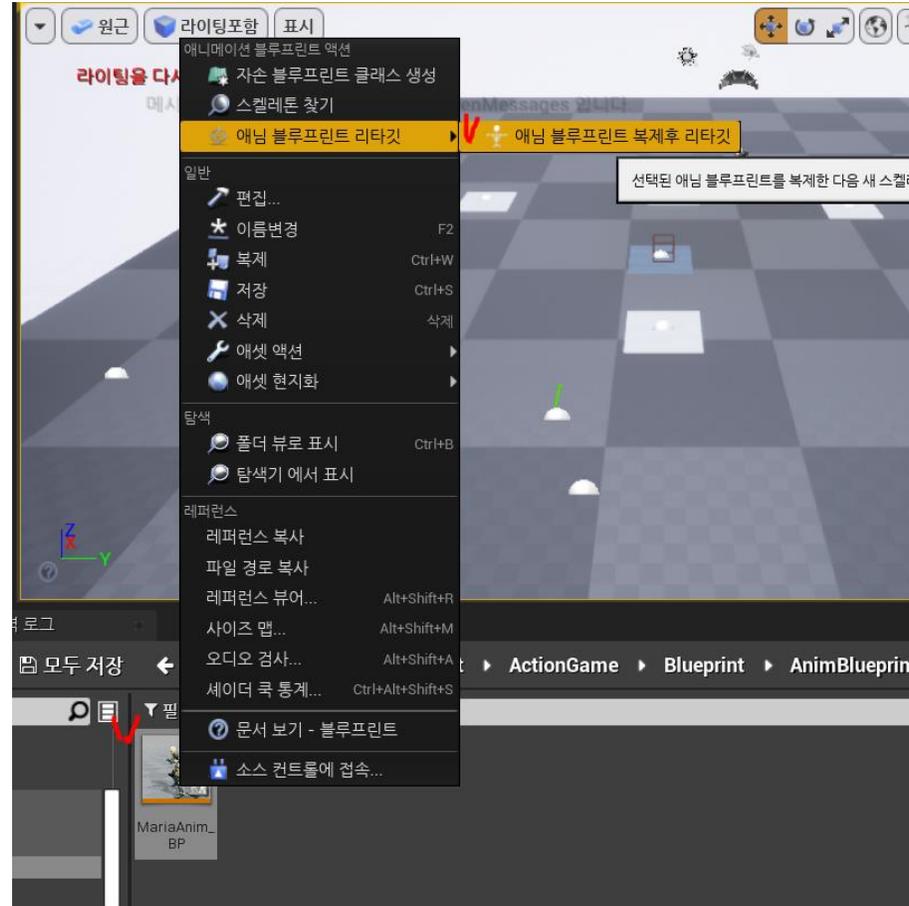
2. 언리얼에 들어가서 믹사모에서 가져온 모델의 스켈레톤에 들어가서 리타겟을 위한 작업을 한다.



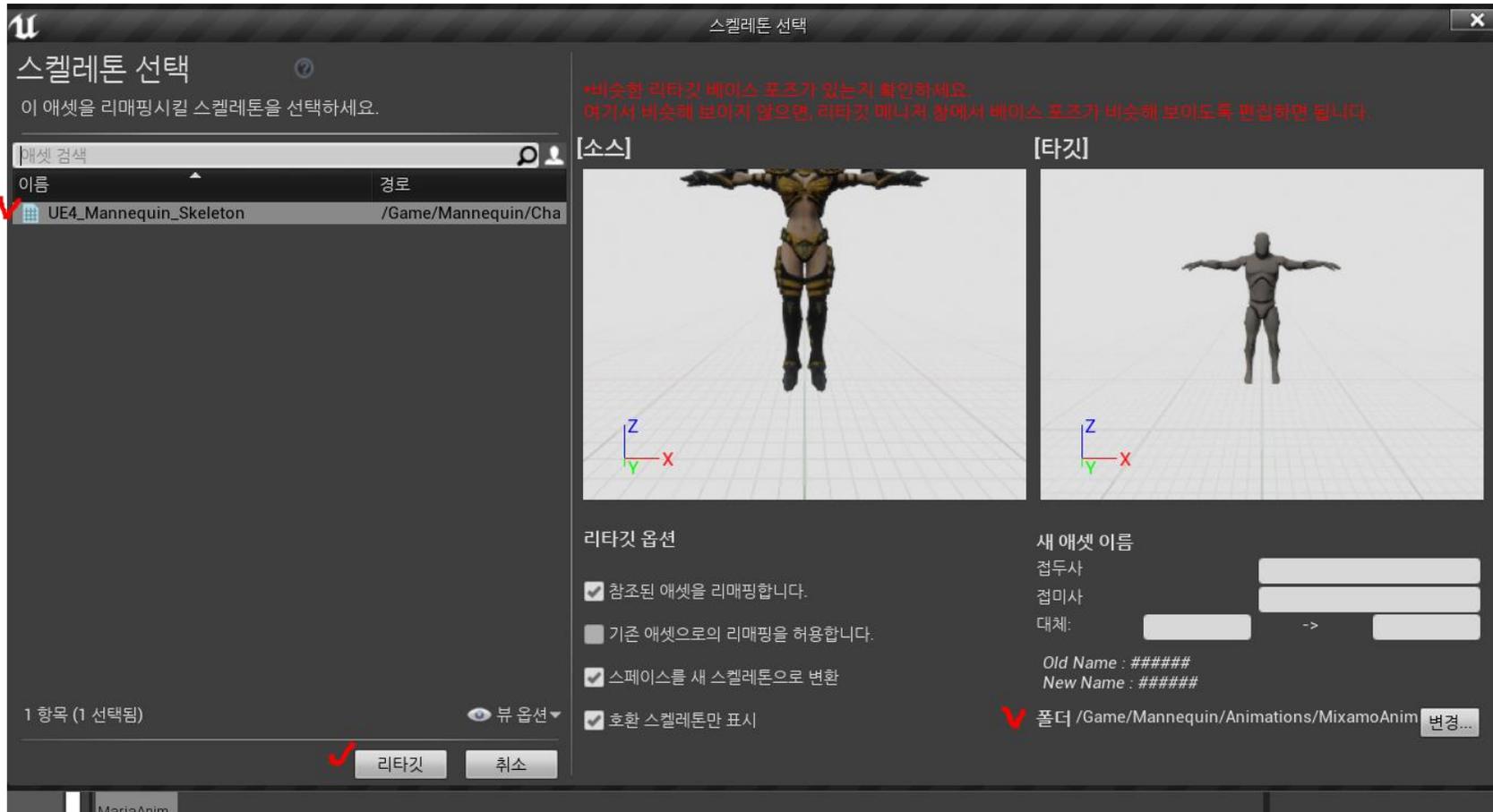
2.1 언리얼에서 제공되는 모델의 스켈레톤에 들어가서 믹사모 캐릭터와 똑같은 포즈를 하게 하고 저장한다.



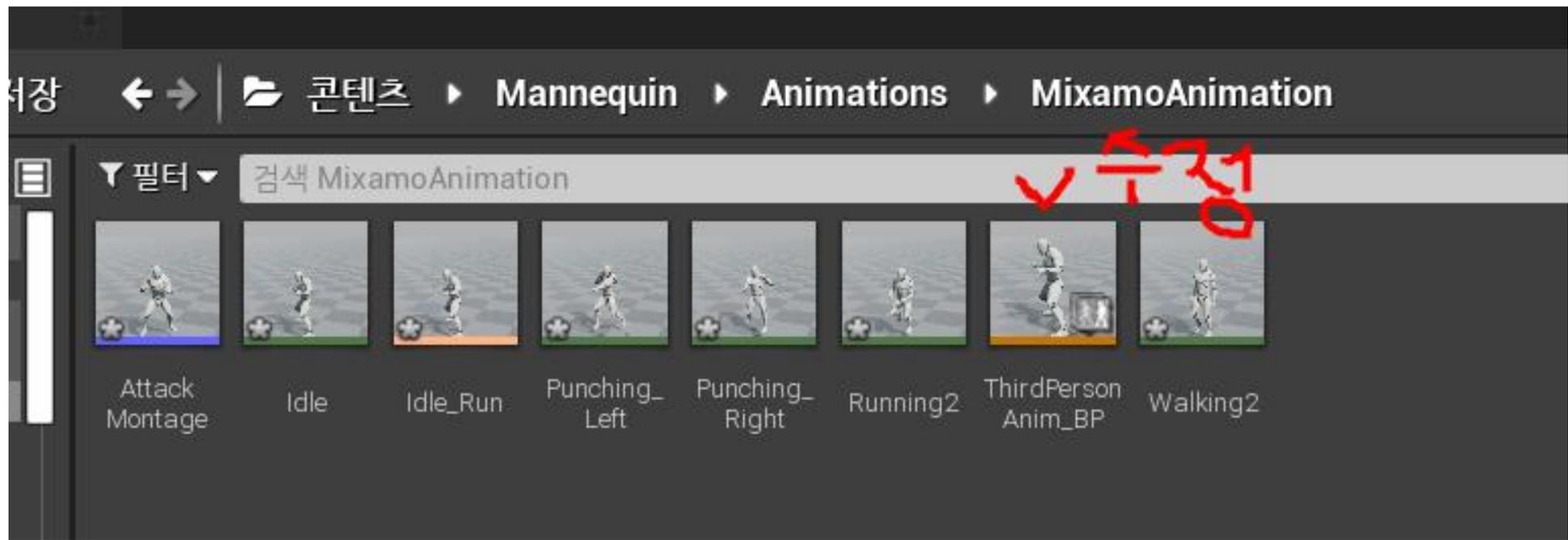
2.2 애니메이션을 저장할 폴더를 생성한다.



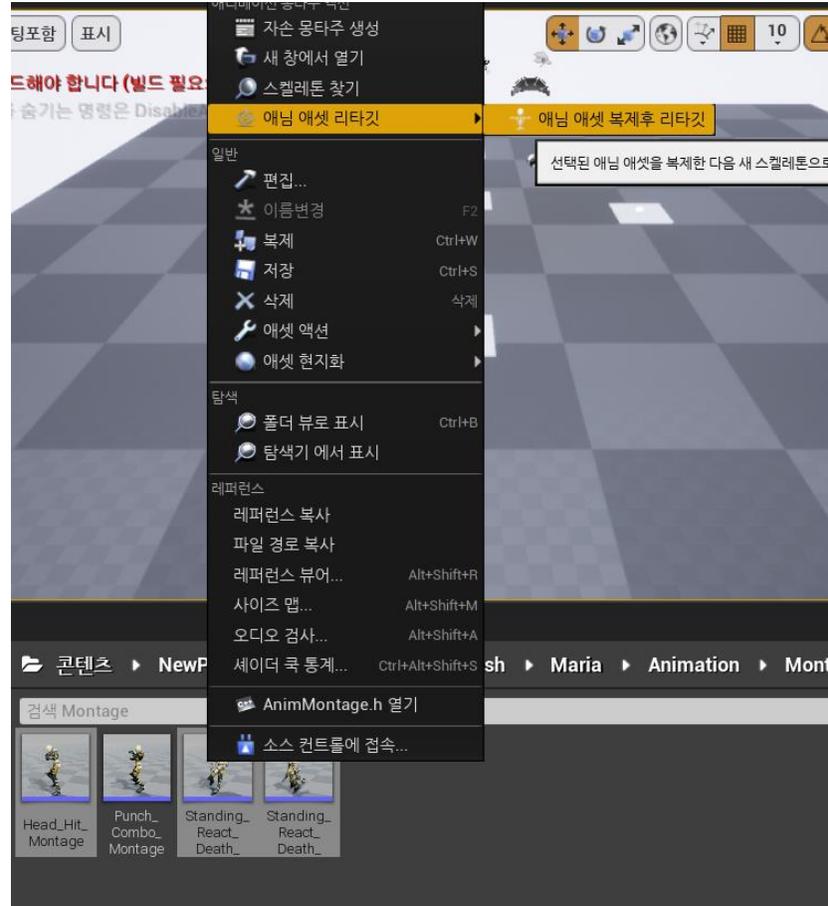
3. 리타겟 할 대상에 리타겟을 한다.(애니메이션, 애님 몽타주, 애니메이션 블루프린트 등이 가능)



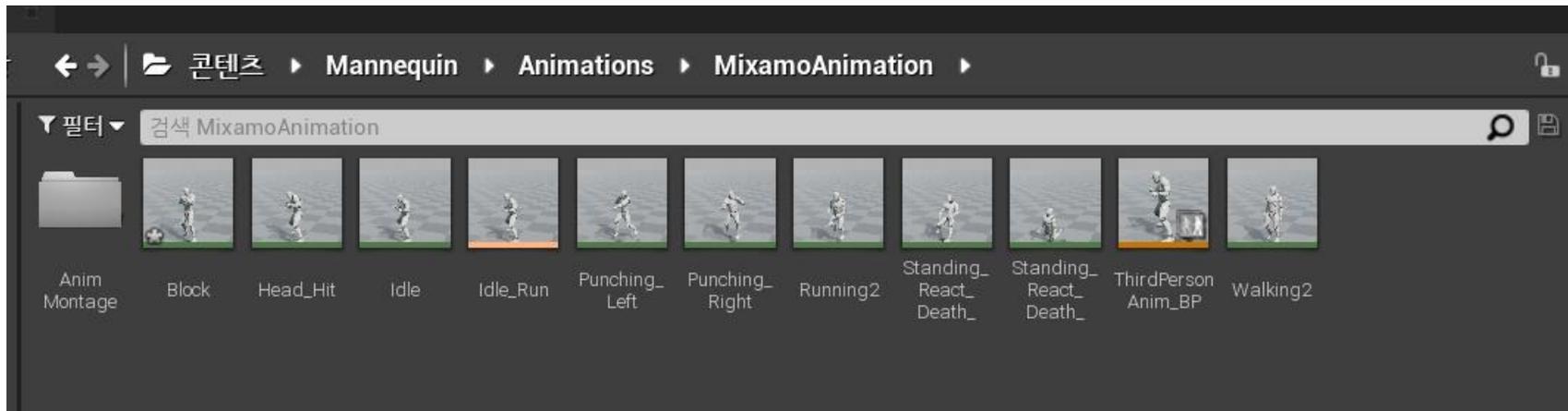
3.1 기타 설정을 한 후 리타겟 한다.



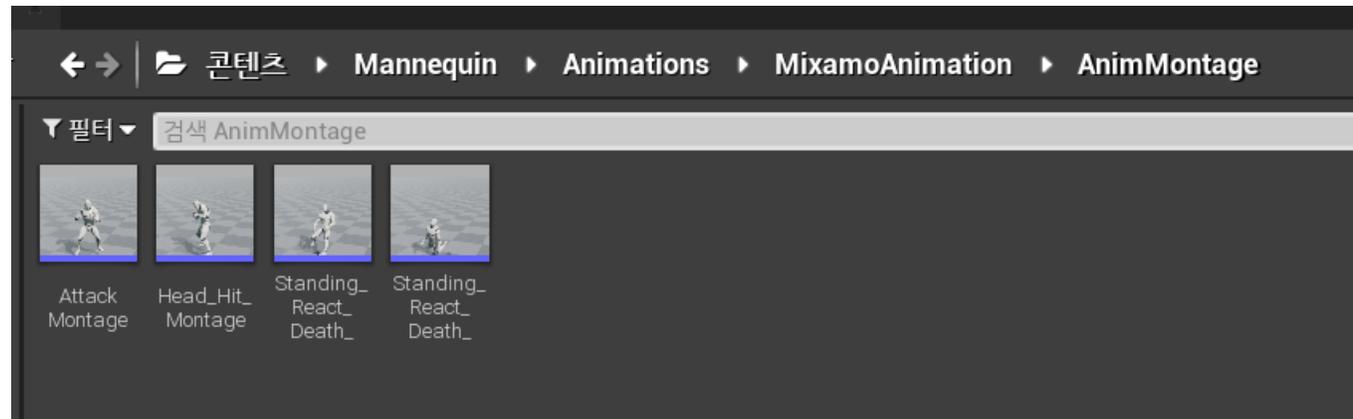
3.2 리타깅 된 모습



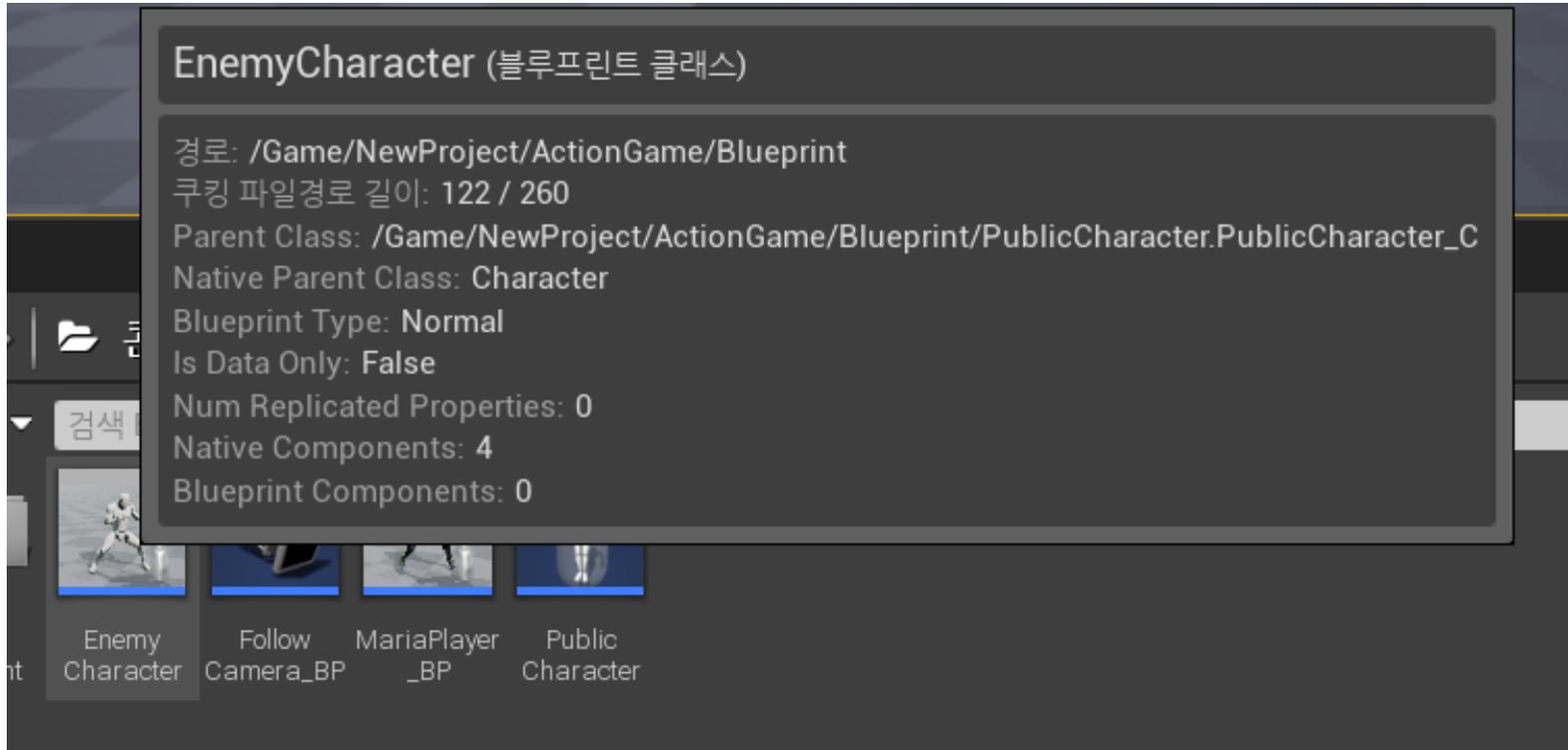
3.3 그 외에 필요한 애셋 리타깃



3.4 리타깃 된 모습1



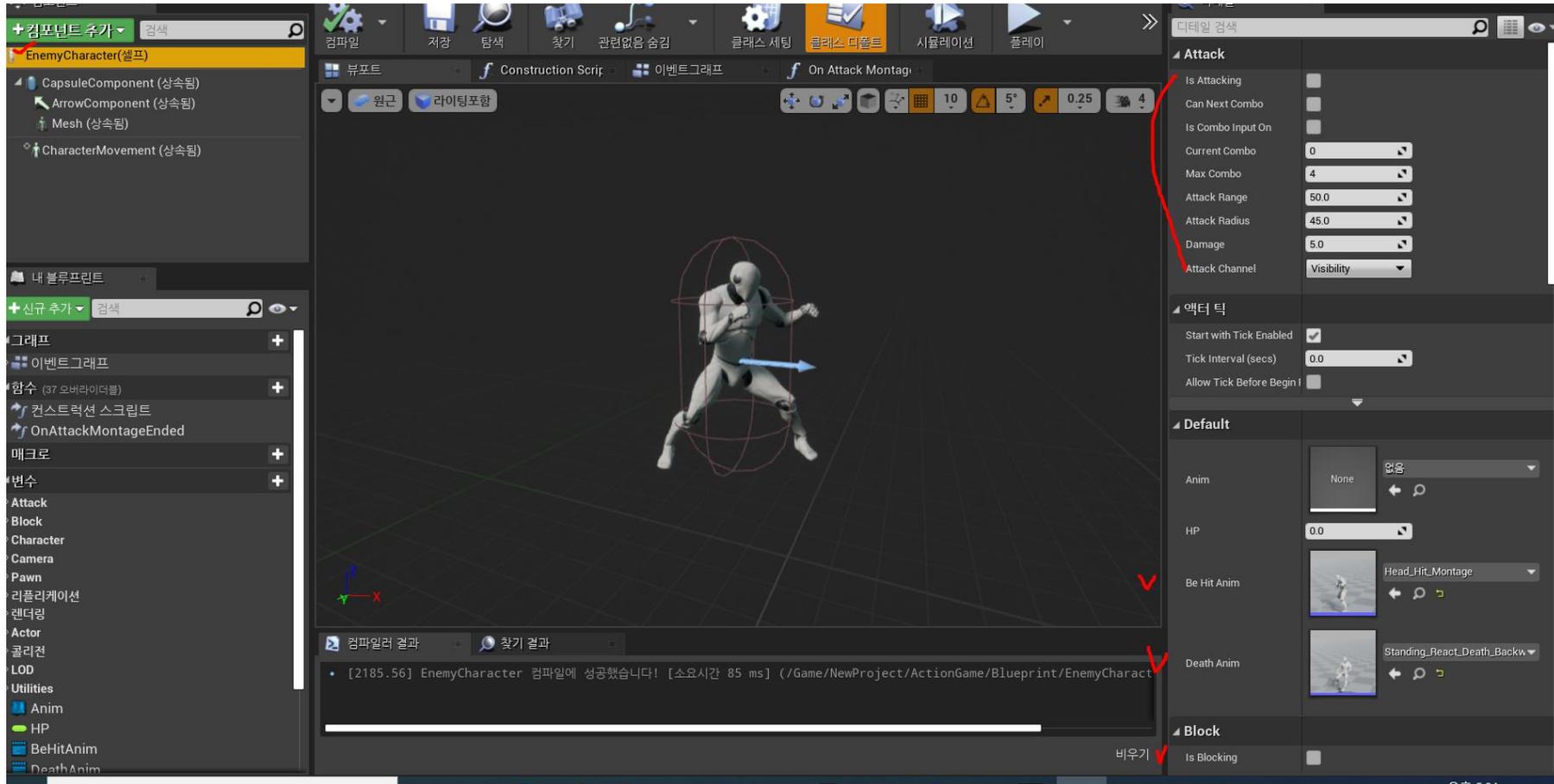
3.5 리타킷 된 모습2



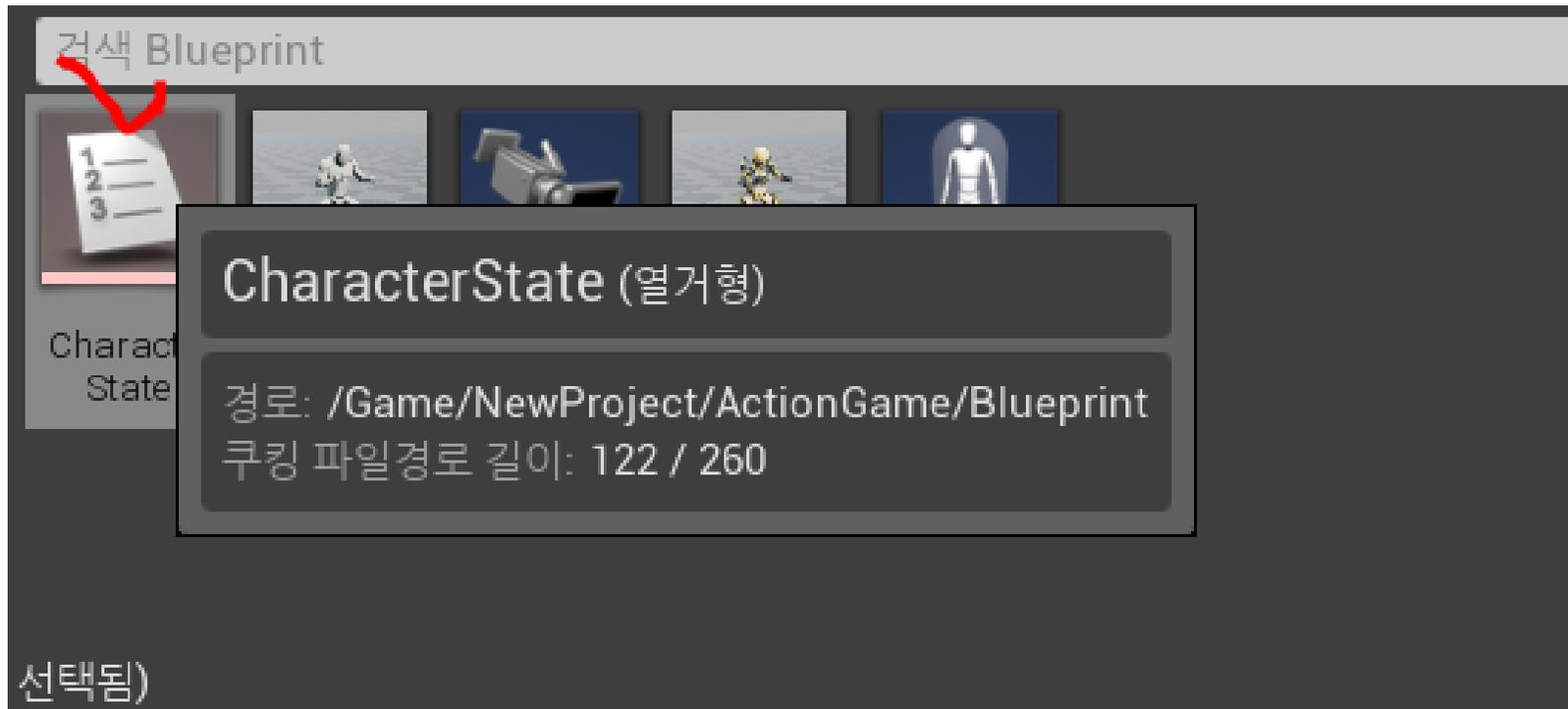
4. **PublicCharacter**를 상속받은 **EnemyCharacter**를 생성한다.



4.1 메시에 들어가서 스켈레톤고 애니메이션 블루프린트를 설정한다.



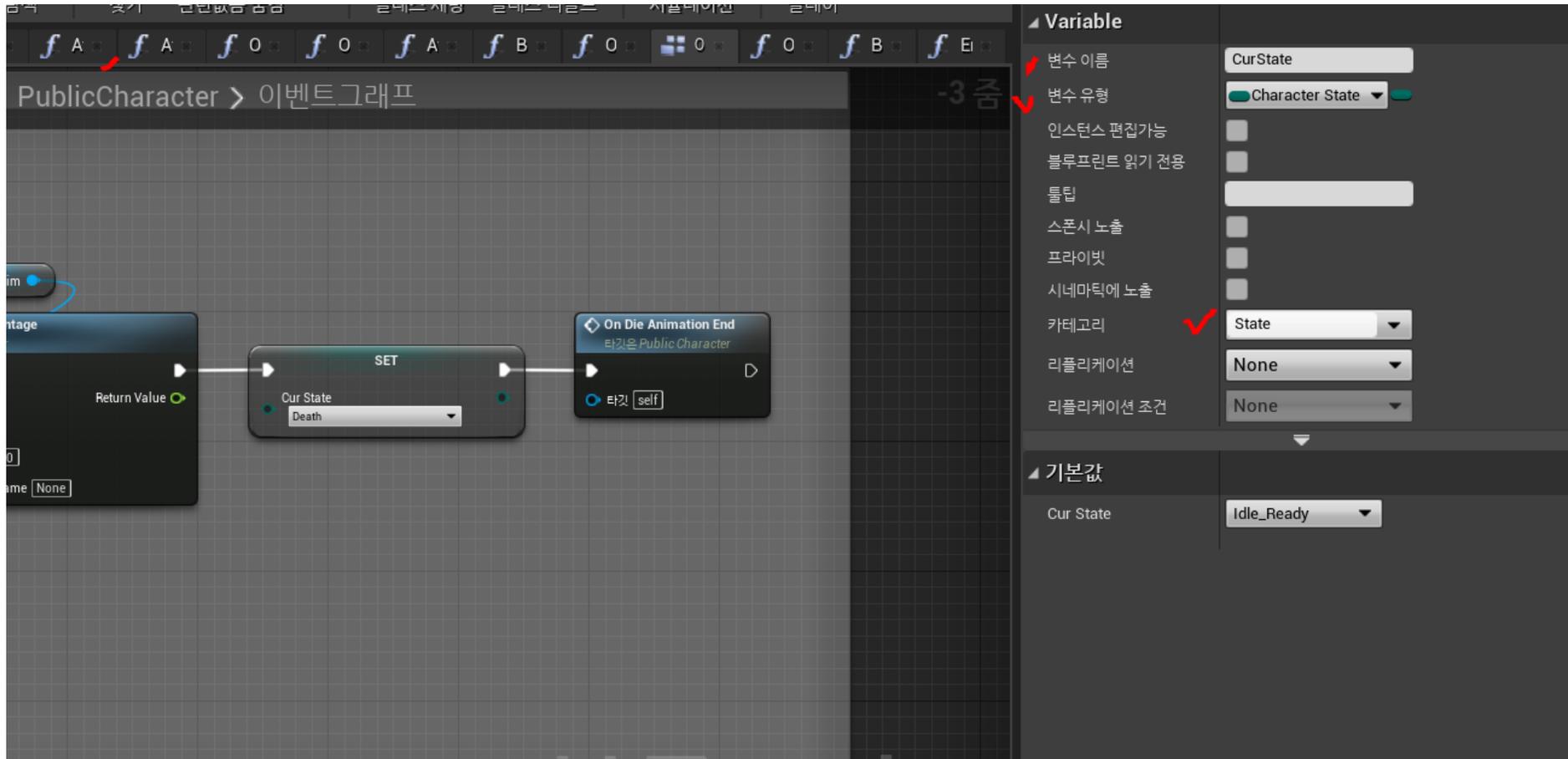
4.2 상속받은 값을 수정한다.



5. 열거형 CharacterState를 생성한다.



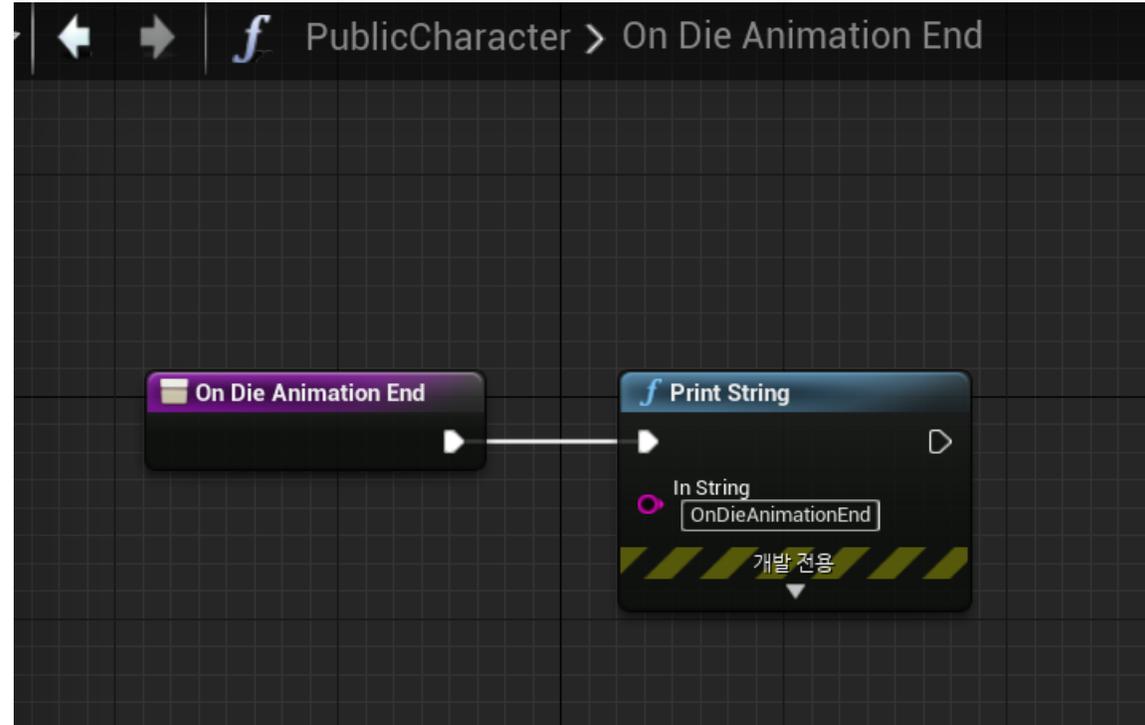
5.1 각 속명을 넣는다.



5.2 PublicCharacter에 들어가서 CharacterState CurState를 생성한다.



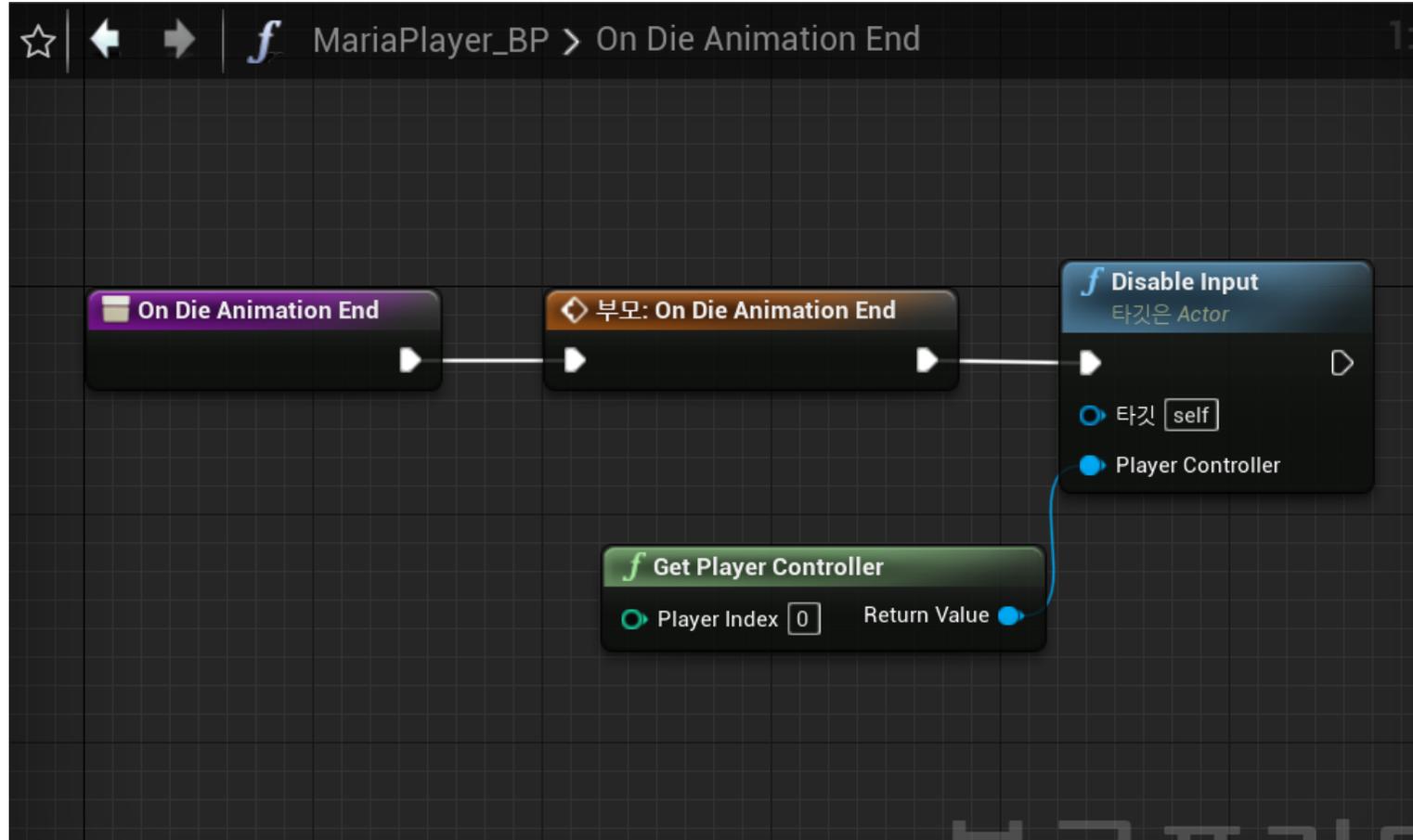
5.3 AnyDamage 이벤트 죽는 처리 부분에 CurState를 Death로 SET 한다.



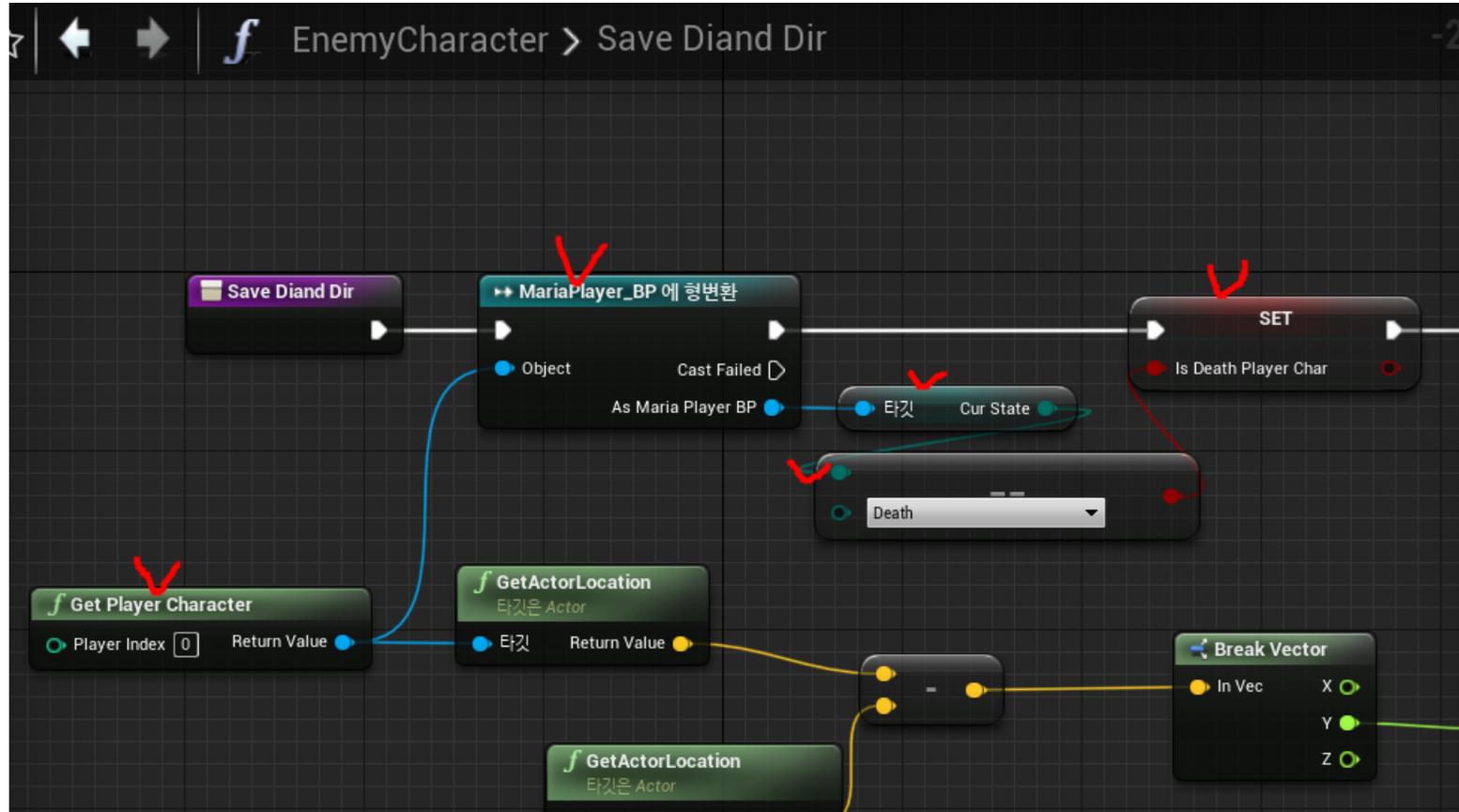
5.4 OnDieAnimationEnd의 내용을 제거한다. (상속받은 클래스에서 구현)



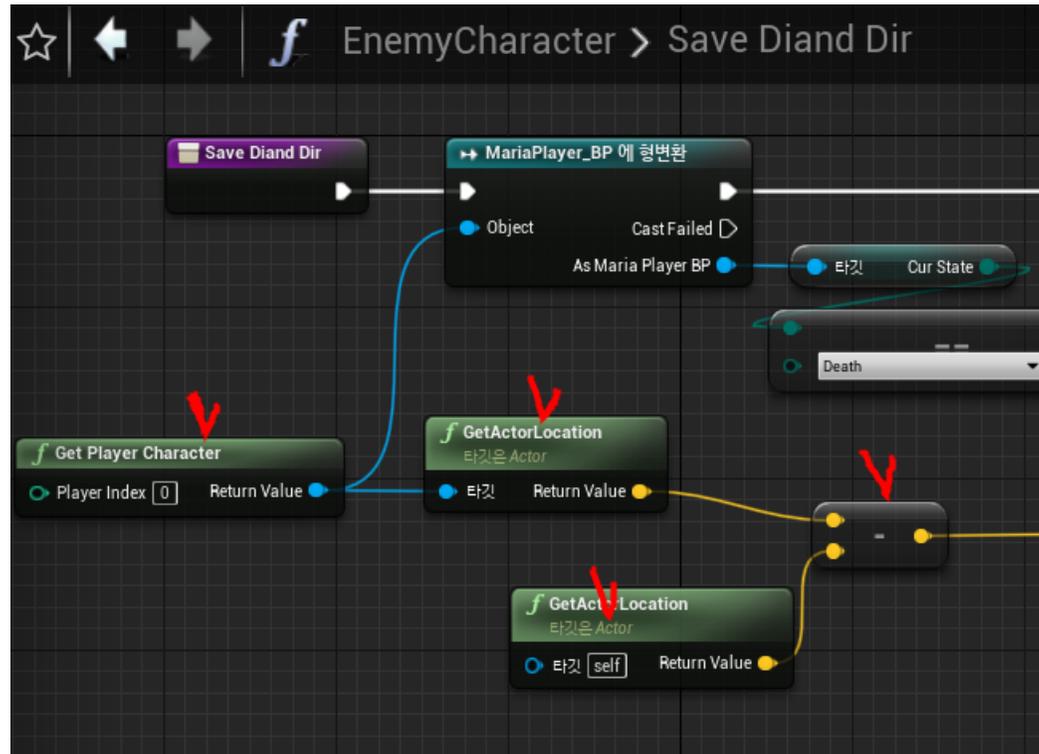
5.5 기존의 `SetTimerbyFuctionName`을 제거하고 `OnDieAnimationEnd` 호출



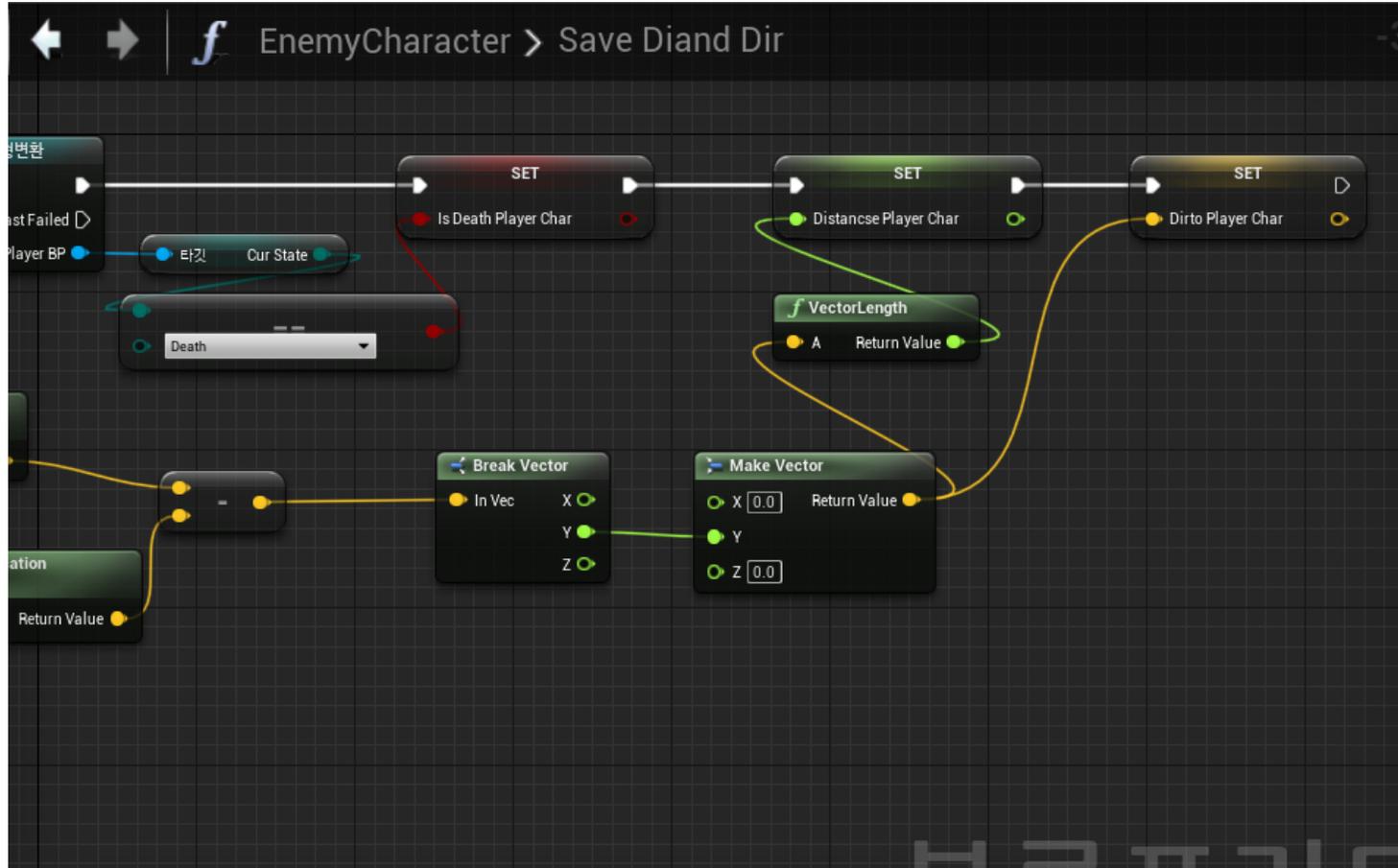
5.6 MariaPlyaeer_BP에 들어가서 OnDieAnimationEnd 함수를 오버라드 해서 내용을 넣는다.



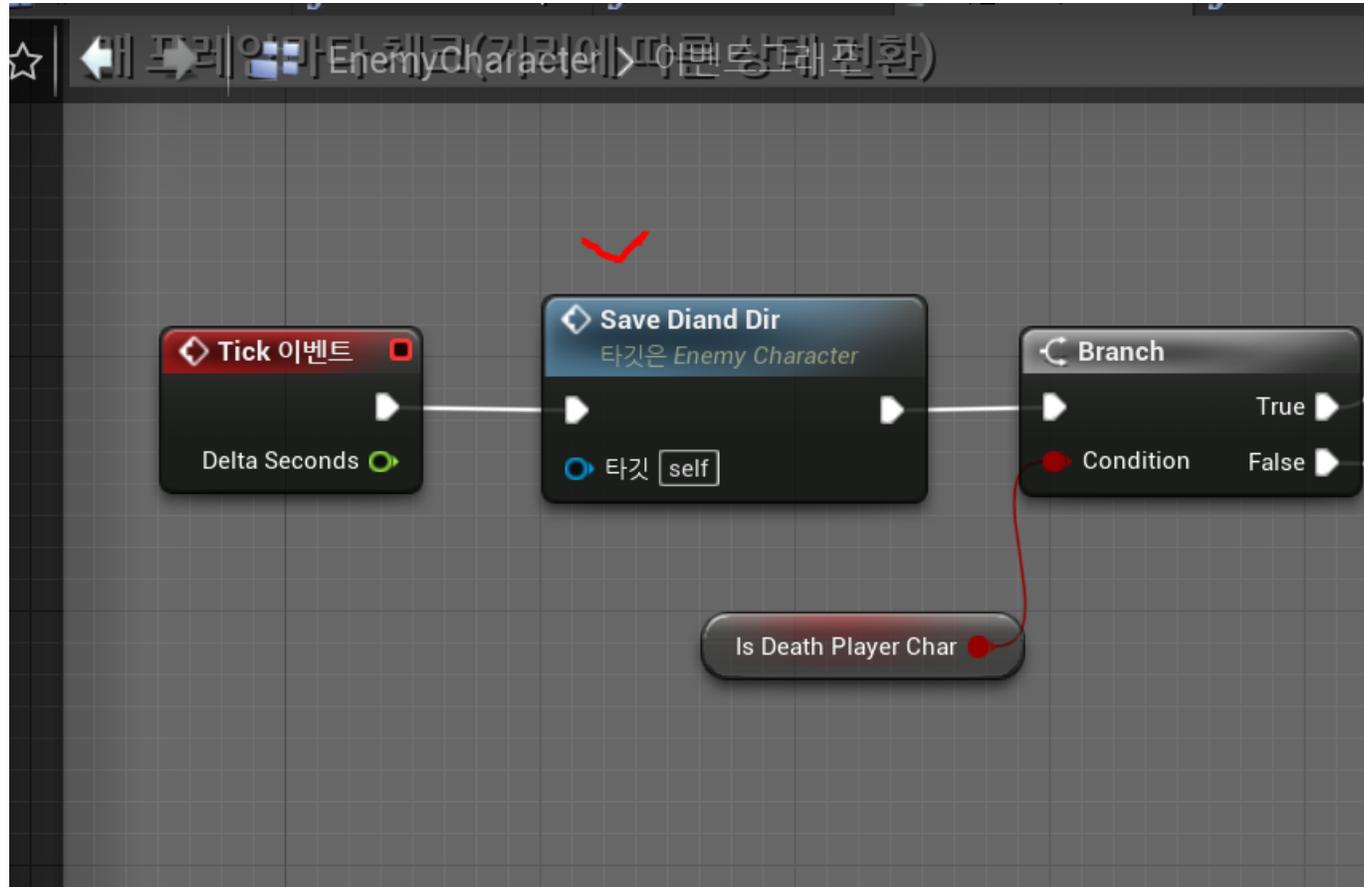
6. EnemyCharacter에 들어가서 SaveDiandDir함수를 생성하고, 내용을 추가한다.



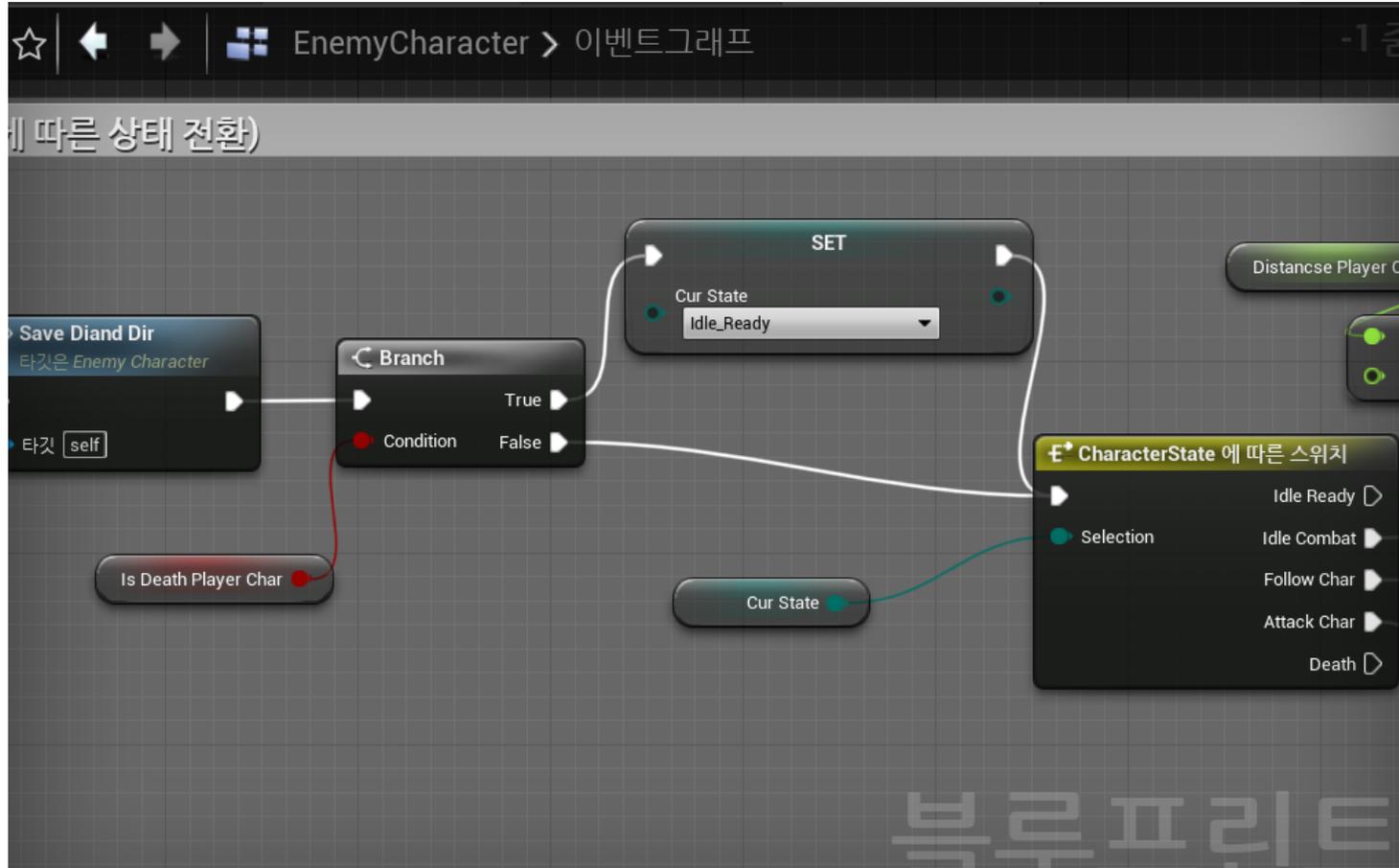
6.1 상대편과 자신의 위치를 뺀다.



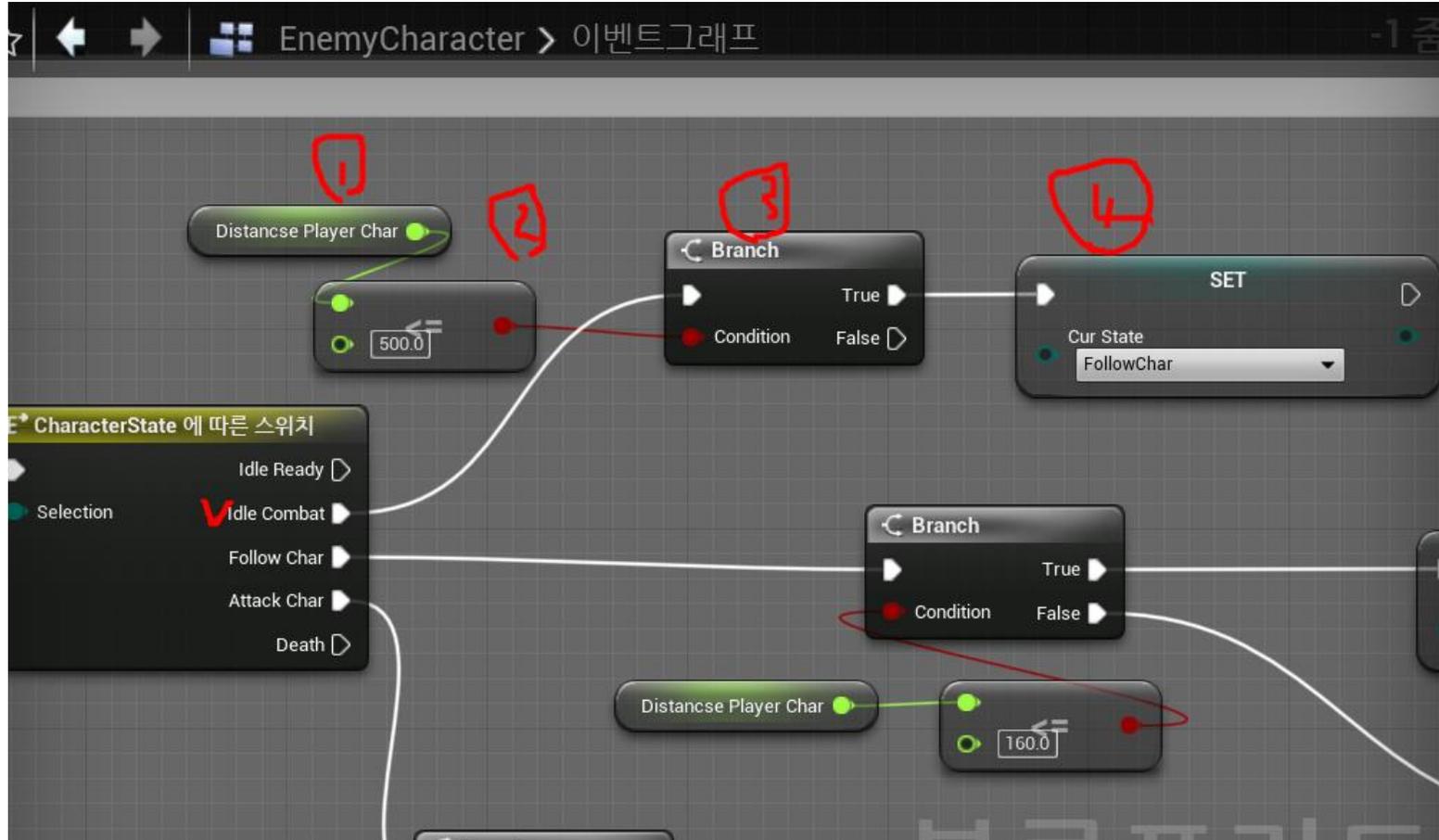
6.2 결과 값의 크기를 DistancePlayerChar, 값을 DirtoPlayerChar로 한다.



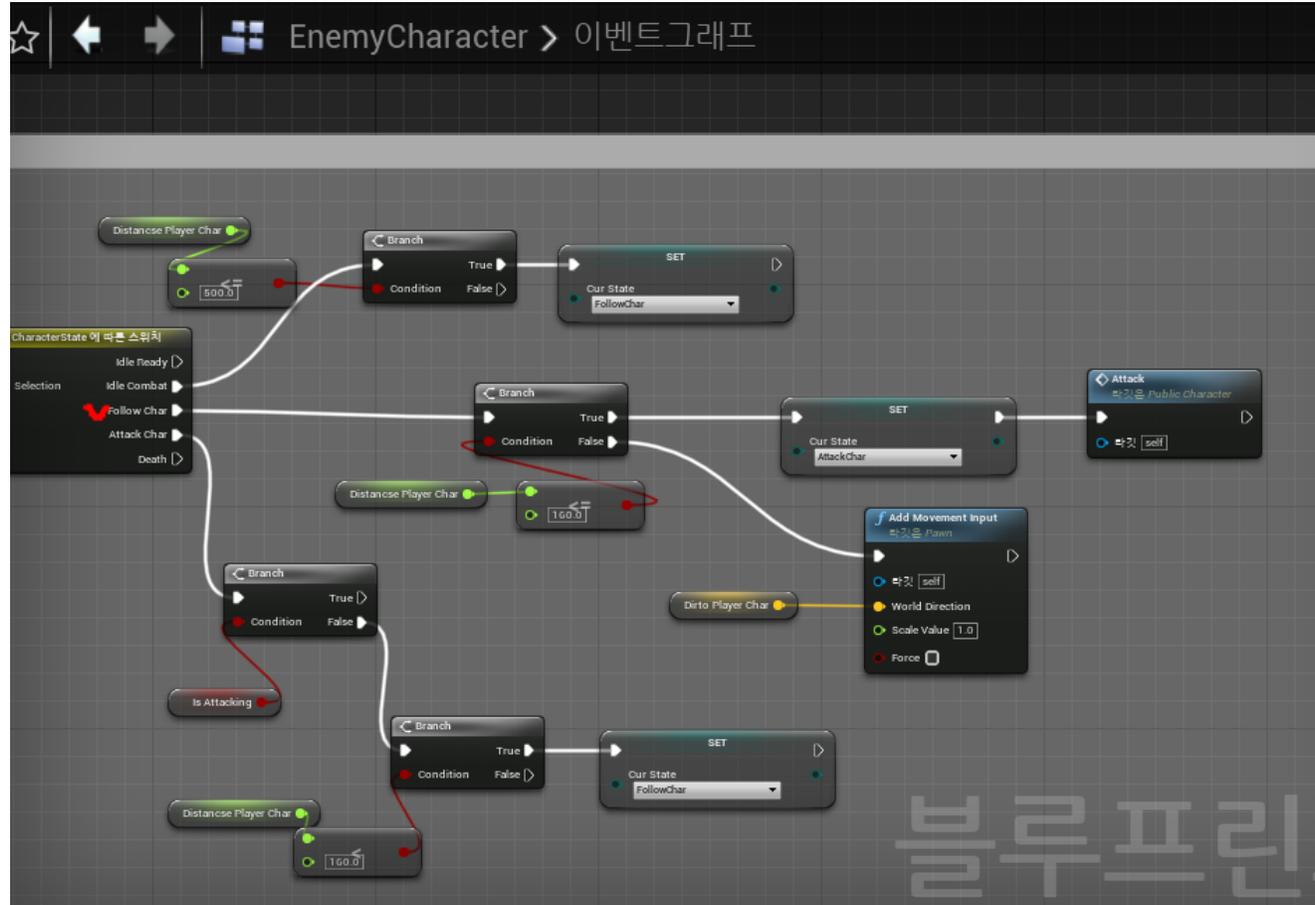
6.3 Tick 이벤트에 SaveDiandDir를 호출한다.



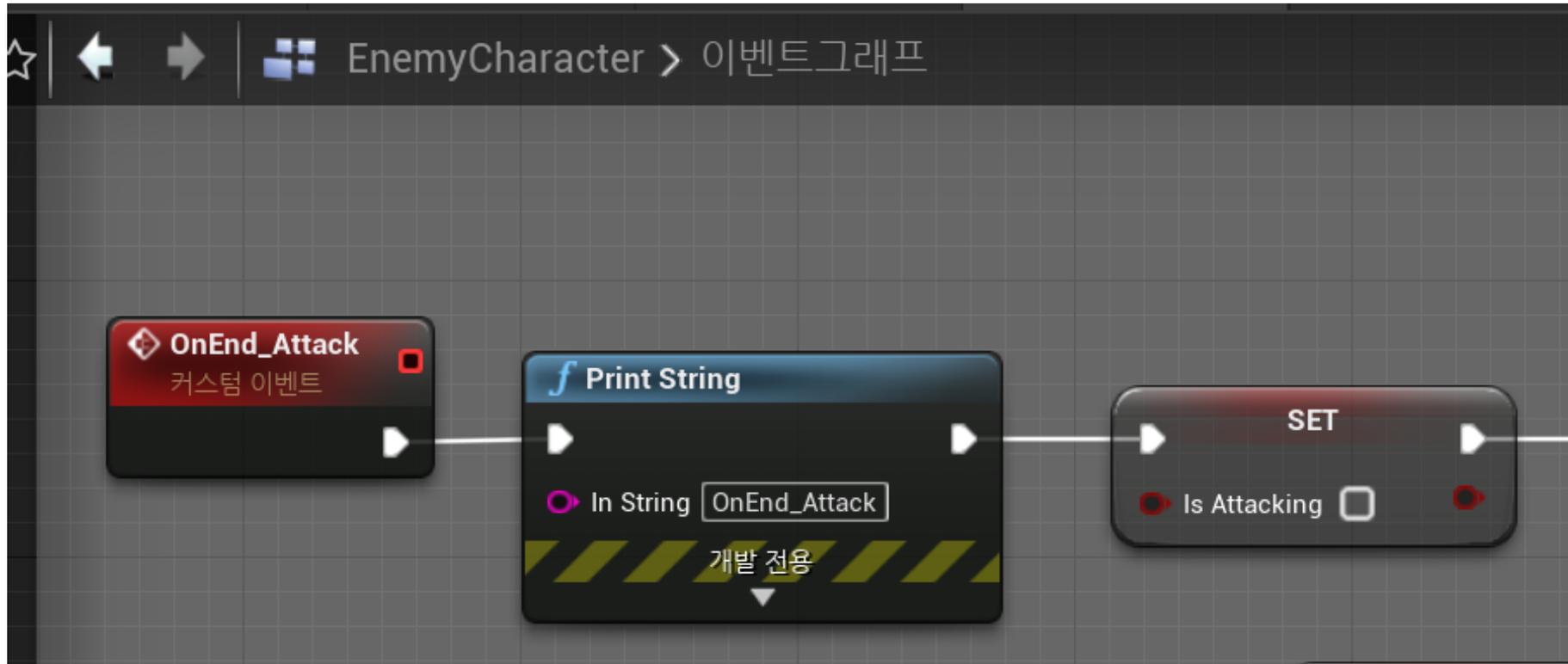
6.4 플레이어가 사망하면 Idle_Ready로 아니면 CurState로 결정한다.



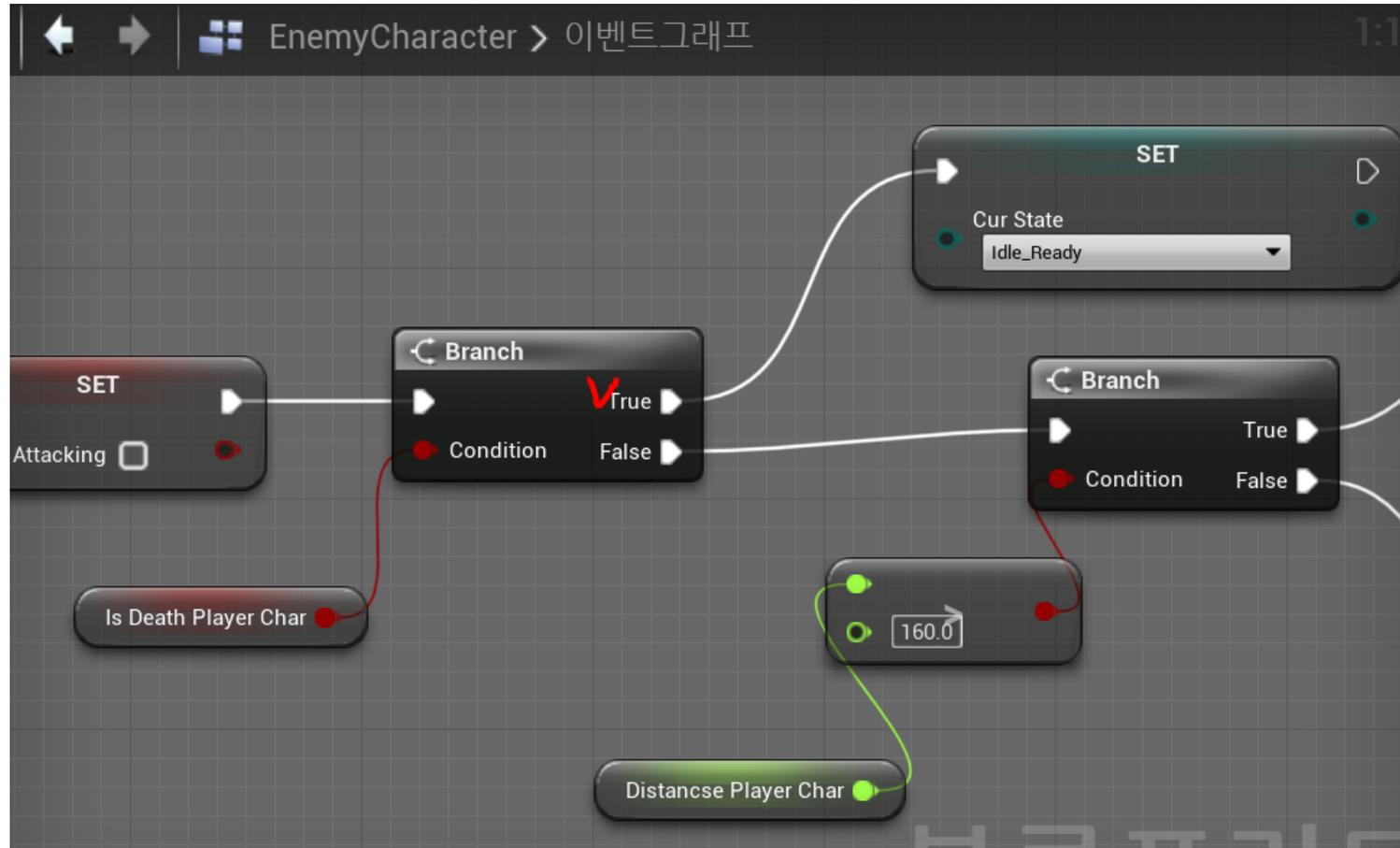
6.5 Idle_Combat 상태에 플레이어와의 거리가 500 안에 있으면 FollowChar 상태가 된다.



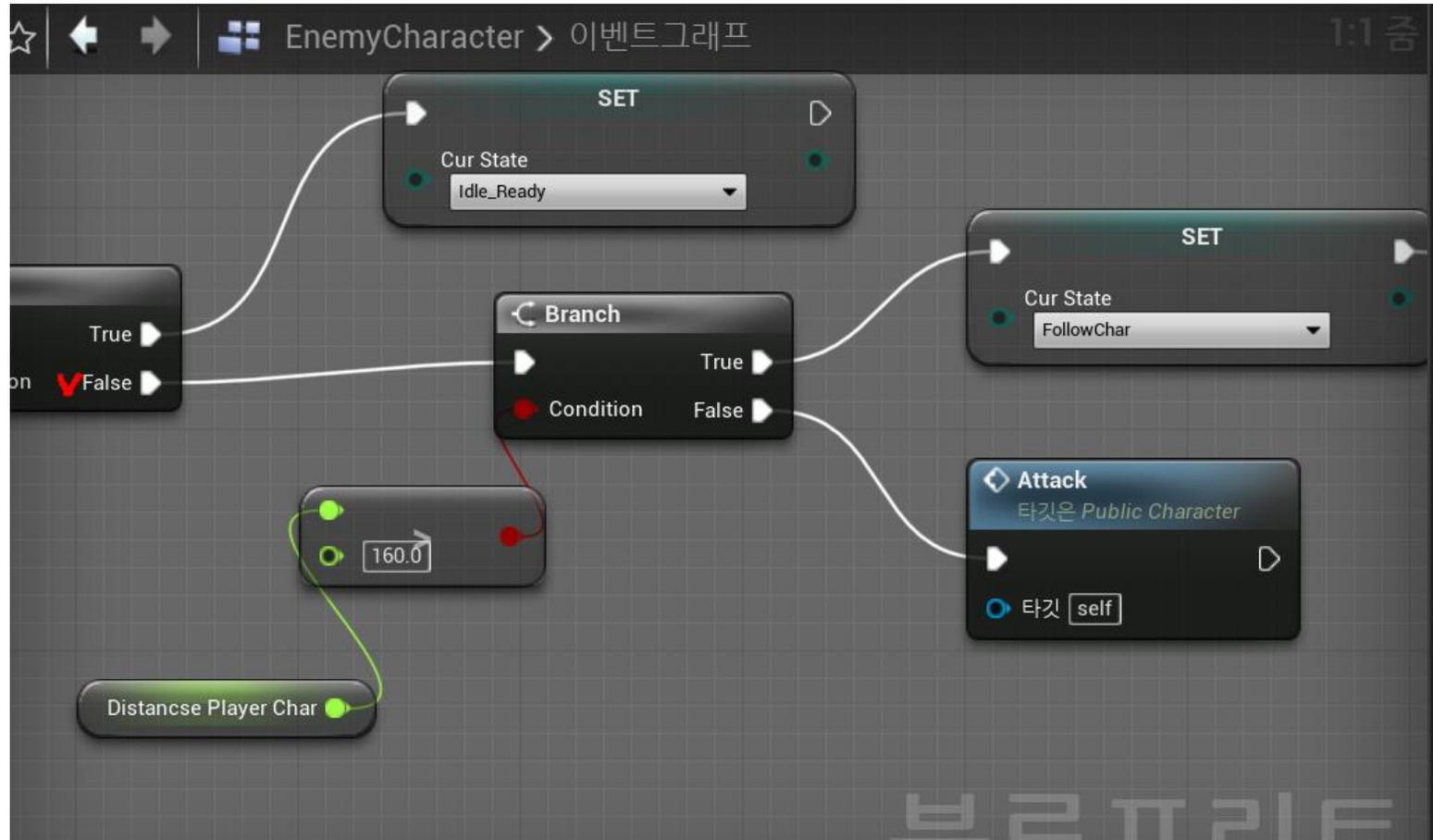
6.6 FollowChar 상태에 거리가 160안에 있으면 AttackChar 상태가되고 Attack을 호출하고, 아니면 이동한다.



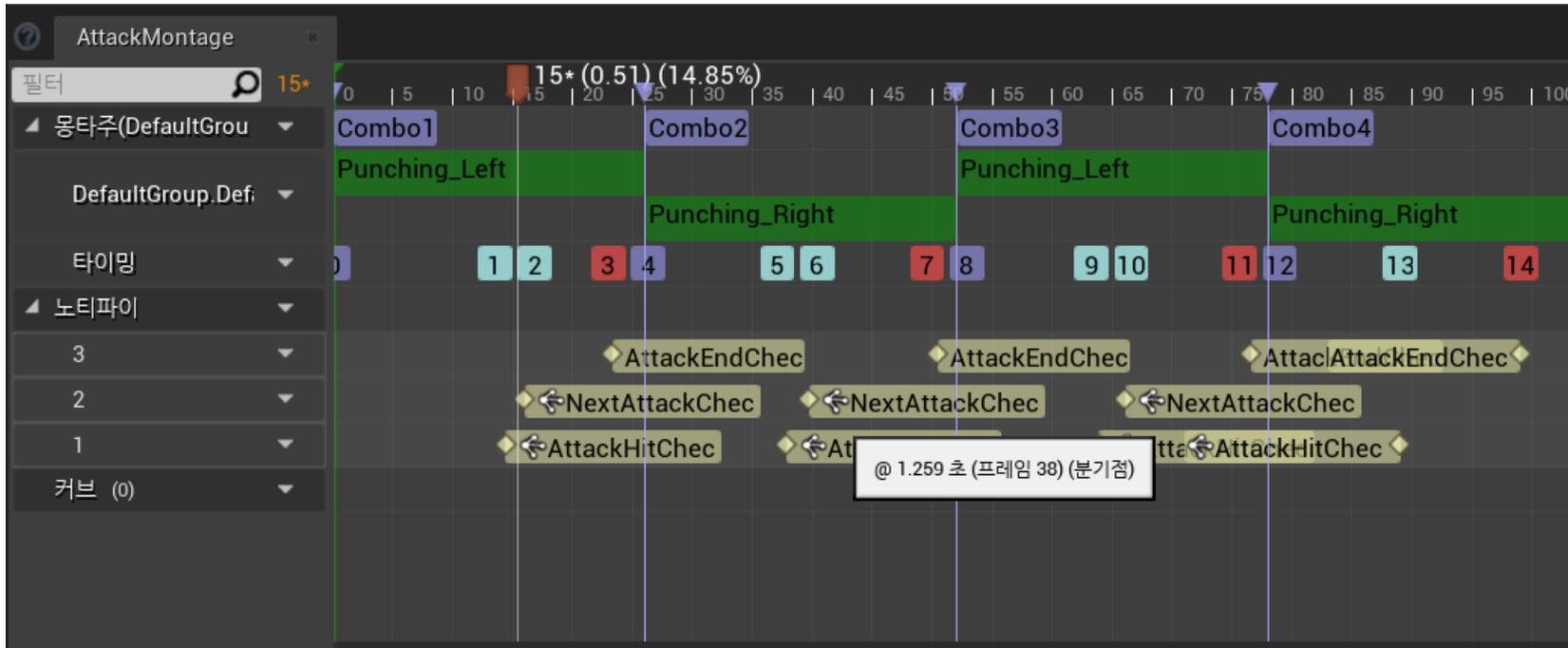
6.8 OnEnd_Attack 이벤트를 추가하고 내용을 넣는다.



6.9 플레이어가 죽은 상태이면 Idle_Ready 상태가 된다.



6.10 플레이어가 죽지 않았다면 거리에 따라 추격하거나 공격을 한다.



6.11 AttackMontage 생성된 두 몽티지에 들어가서 노티파이 AttackEndCheck를 추가하고 애니메이션 끝나는 지점에 둔다.

```
hello.cpp  ABAAnimInstance.h  ABAAnimInstance.cpp  BoardCharacterBase.h  BoardCharacterBase.cpp
(전역 범위)
7  #include "ABAAnimInstance.generated.h"
8
9  /**
10  *
11  */
12
13  DECLARE_DYNAMIC_MULTICAST_DELEGATE(FOnNextAttackCheckDelegate);
14  DECLARE_DYNAMIC_MULTICAST_DELEGATE(FOnAttackHitCheckDelegate);
15  DECLARE_DYNAMIC_MULTICAST_DELEGATE(FOnAttackEndCheckDelegate);
16
17  UCLASS()
18  class HELLO_API UABAAnimInstance : public UAnimInstance
19  {
20  public:
21  GENERATED_BODY()
22
23  // 생성자
24
```

7. ABAAnimInstance.h에 들어가서 FOnAttackEndCheckDelegate를 추가한다.

```
o.h  hello.cpp  ABAnimInstance.h  ABAnimInstance.cpp  BoardCharacterBase.h  BoardCharacterBase.cpp
hello  (전역 범위)
34
35  UPROPERTY(BlueprintAssignable)
36  FOnNextAttackCheckDelegate OnNextAttackCheck;
37
38  UPROPERTY(BlueprintAssignable)
39  FOnAttackHitCheckDelegate OnAttackHitCheck;
40
41  UPROPERTY(BlueprintAssignable)
42  V FOnAttackEndCheckDelegate OnAttackEndCheck;
43
44  private:
45  // 특정 타이밍에 공격을 체크
46  UFUNCTION()
47  void AnimNotify_AttackHitCheck();
48
49  // 다음 공격으로 넘어가는 공격을 체크
50  UFUNCTION()
51  void AnimNotify_NextAttackCheck();
```

7.1 FOnAttackEndCheckDelegate OnAttackEndCheck를 선언

```
lo.cpp  ABAnimInstance.h  ABAnimInstance.cpp  BoardCharacterBase.h  BoardCharacter
(전역 범위)

private:
    // 특정 타이밍에 공격을 체크
    UFUNCTION()
    void AnimNotify_AttackHitCheck();

    // 다음 공격으로 넘어가는 공격을 체크
    UFUNCTION()
    void AnimNotify_NextAttackCheck();

    UFUNCTION()
    void AnimNotify_AttackEndCheck();

    // 섹션의 이름을 출력하는 함수
    FName GetAttackMontageSectionName(int32 Section);

public:
```

7.2 AnimNotigy_AttackEndCheck 함수를 선언

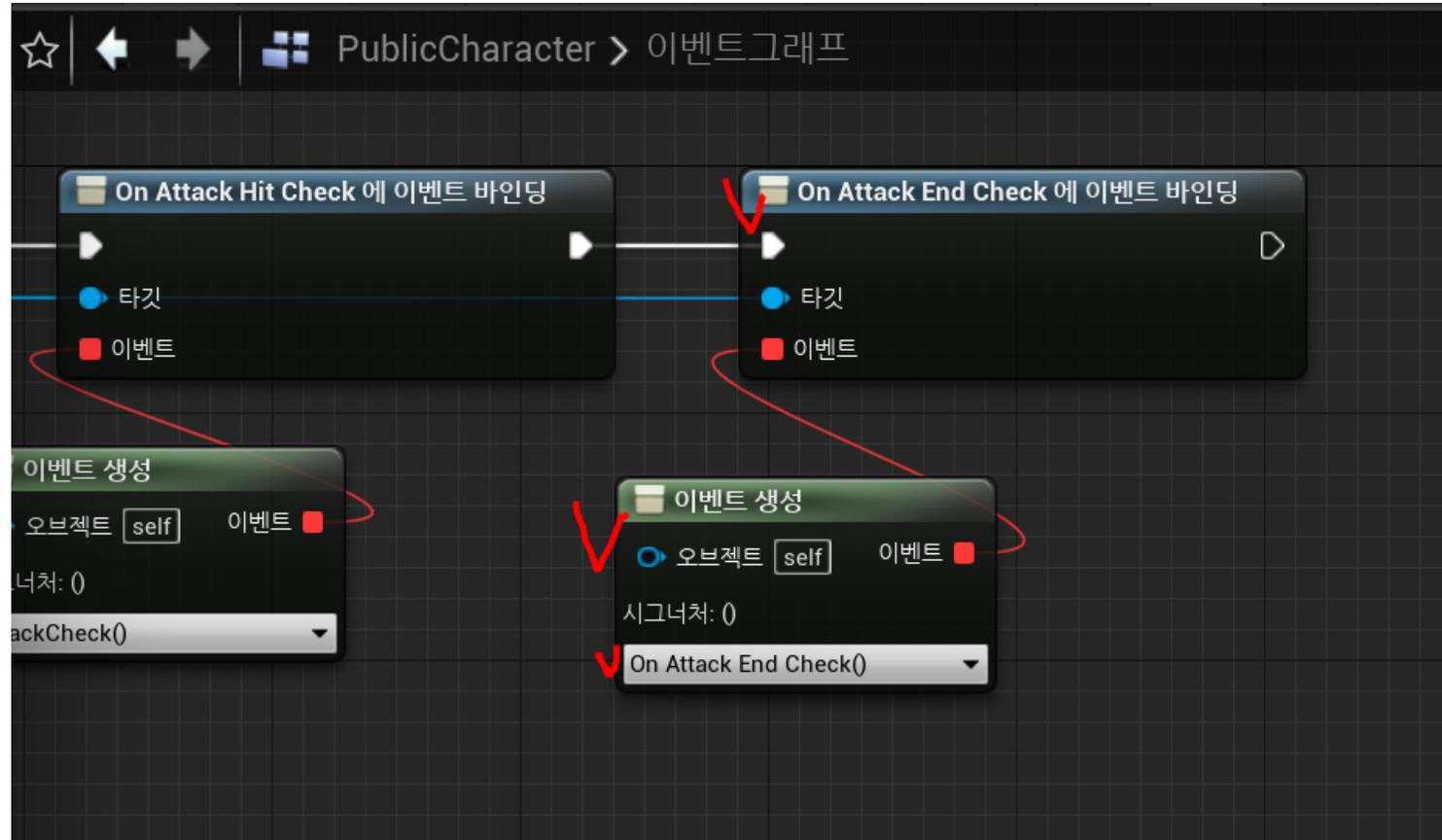
```
hello.cpp | ABAnimInstance.h | ABAnimInstance.cpp | BoardCharacterBase.h | BoardCharacterBase.cpp
(전역 범위)
```

```
void UABAnimInstance::AnimNotify_NextAttackCheck()  
{  
    ABLOG_S(Warning);  
    OnNextAttackCheck.Broadcast();  
}
```

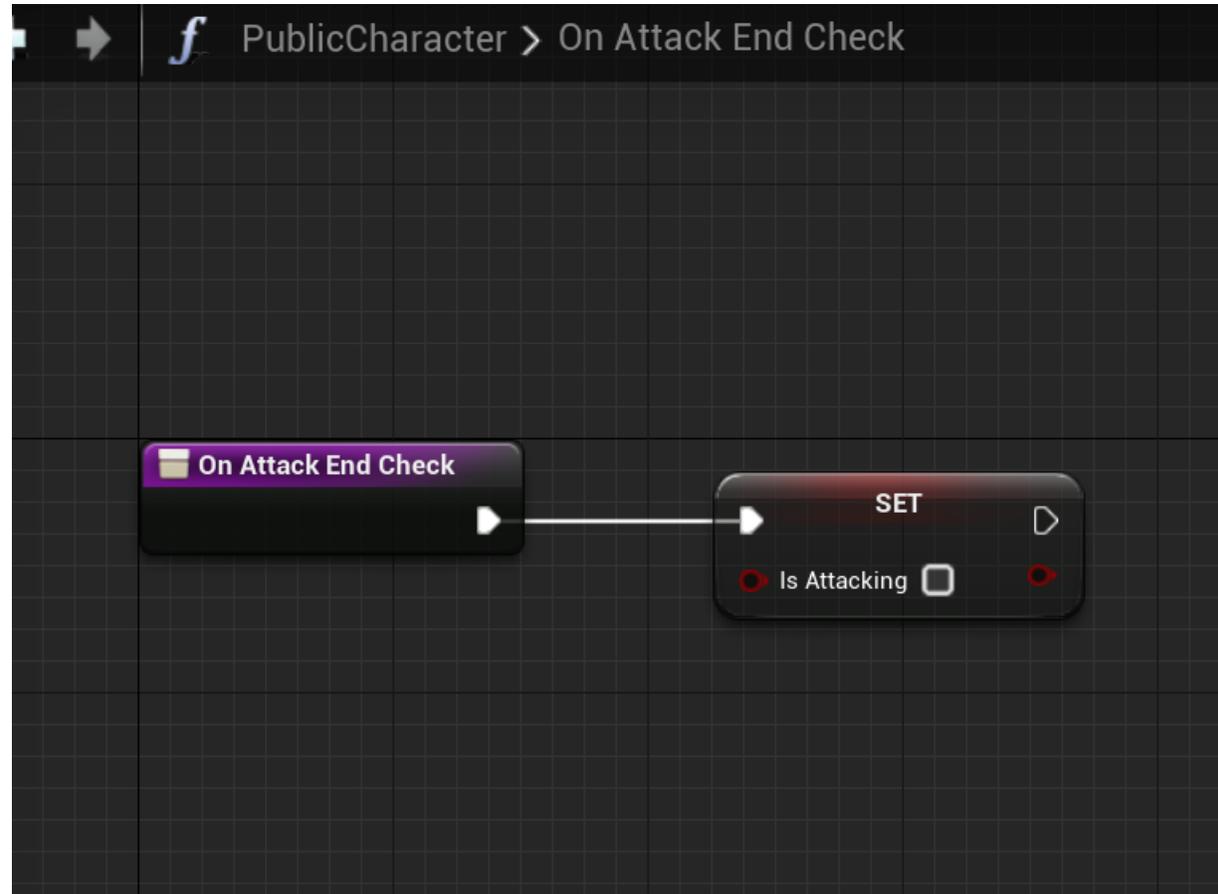
```
void UABAnimInstance::AnimNotify_AttackEndCheck()  
{  
    ABLOG_S(Warning);  
    OnAttackEndCheck.Broadcast();  
}
```

```
FName UABAnimInstance::GetAttackMontageSectionName(int32 Section)  
{  
    ABCHECK(FMath::IsWithinInclusive<int32>(Section, 1, 4), NAME_None);  
    return FName(*FString::Printf(TEXT("Combo%d"), Section));  
}
```

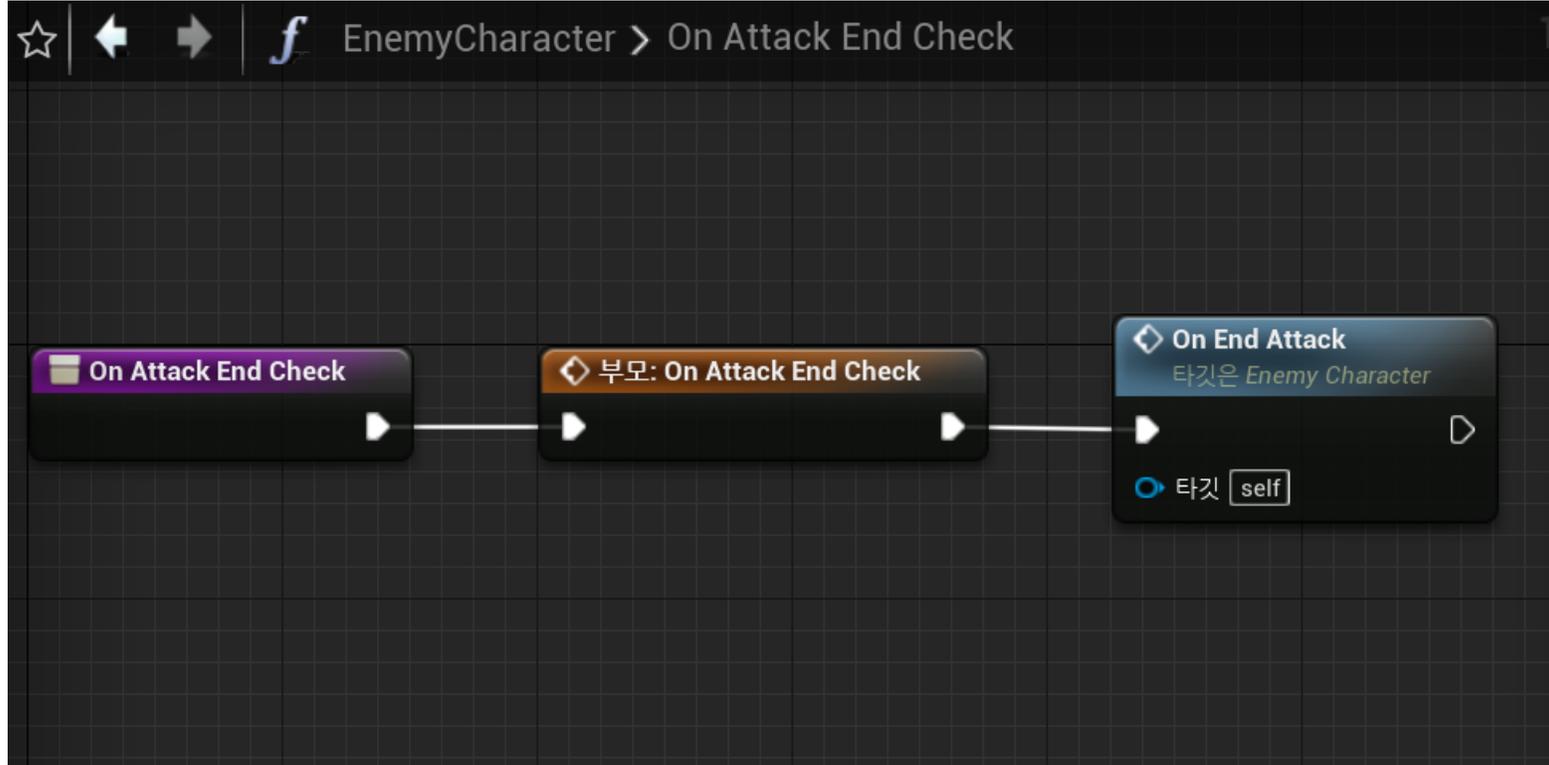
7.3 구현부로 넘어가서 AnimNotify_AttackEndCheck에 OnAttackEndCheck를 Broadcast한다.



8. PublicCharacter 이벤트 그래프에 BeginPlay에 Anim에서 OnAttacEndCheck를 바인딩한다.



8.1 OnAttackEndCheck 함수에 IsAttacking을 False로 SET 한다.



9. `EnemyCharacter`에 들어가서 `OnAttackEndCheck`를 오버라이드하고 `OnEndAttack`을 호출한다.