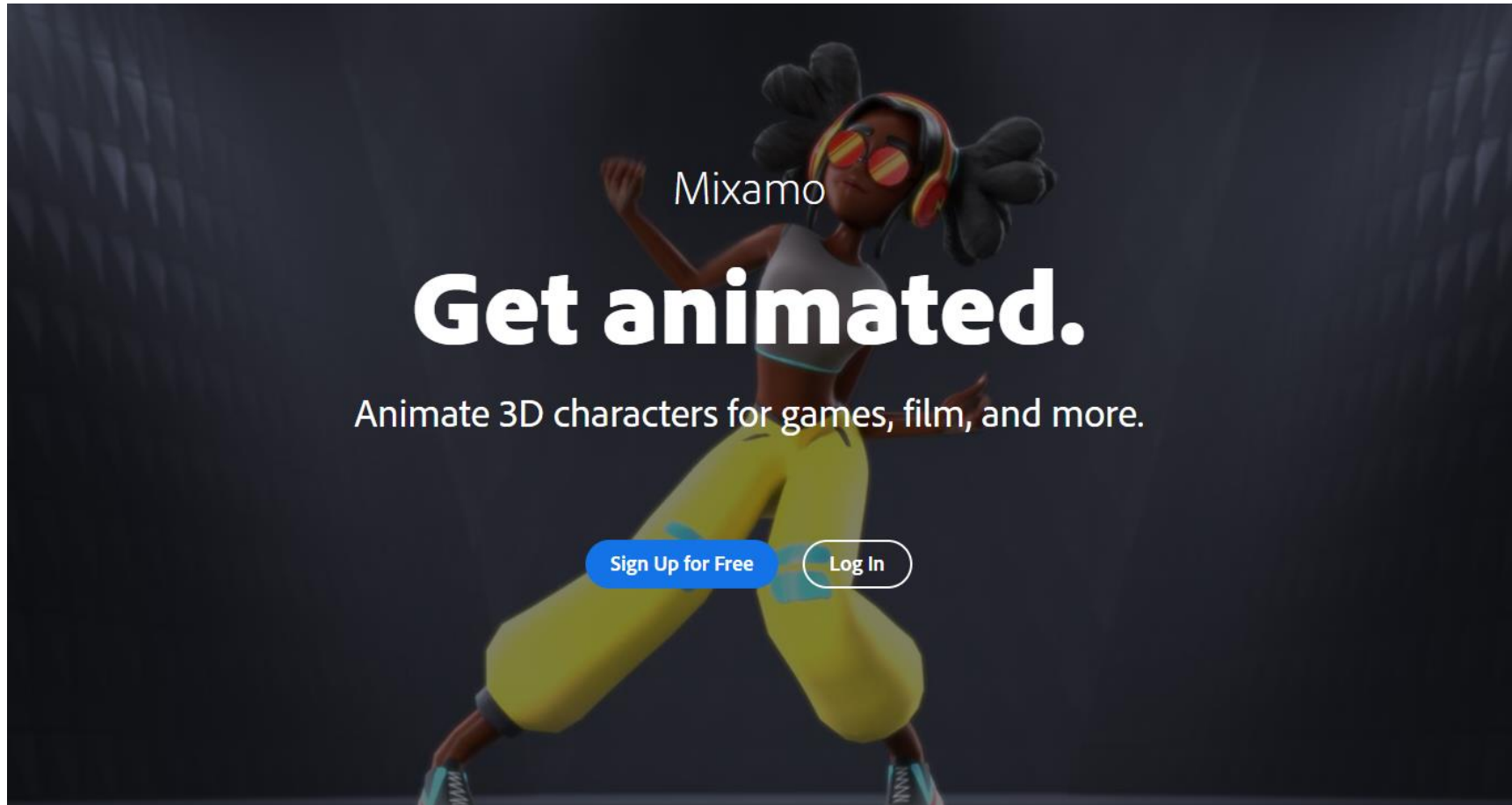
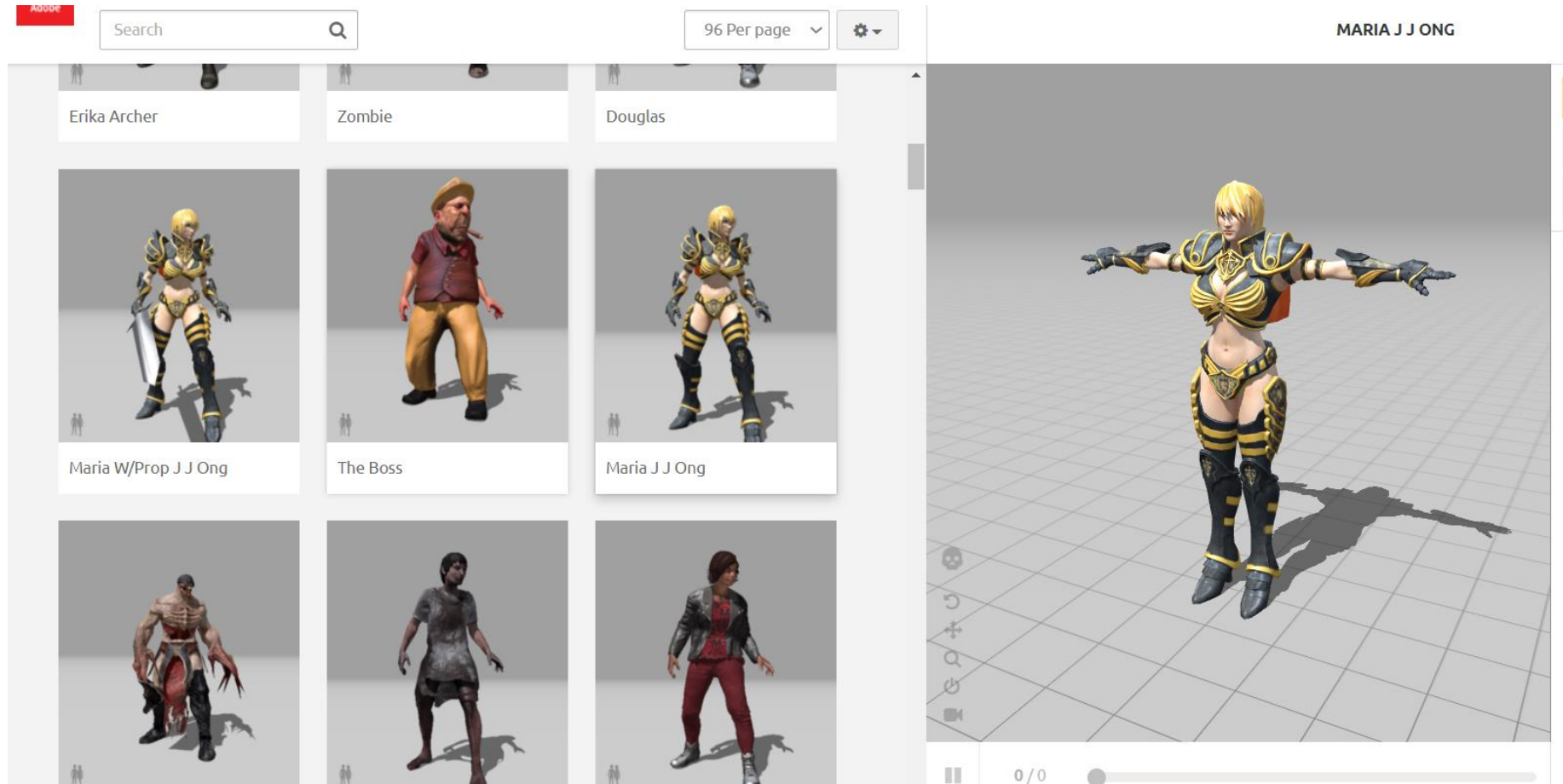


목차

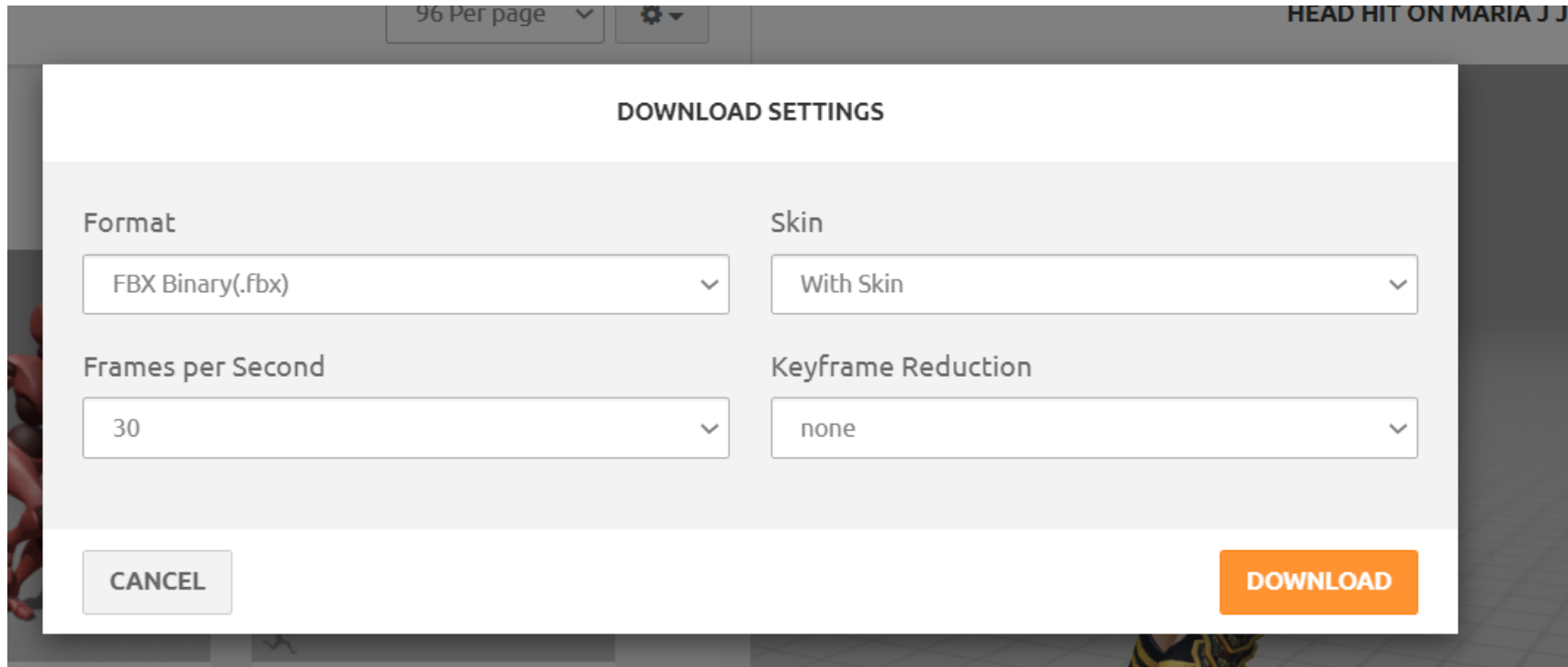
1. 캐릭터와 애니메이션을 얻기 위해서 믹사모에 들어가서 로그인 한다.
2. 애니메이션에 사용되는 블렌드 스페이스 1D를 생성한다.
3. Head_Hit 모션과 Death 모션의 애님 몽타주를 생성한다.
4. AnimInstance를 상속받은 ABAnimInstance을 생성한다.
5. ABAnimInstace를 부모로 하고 믹사모에서 가져온 모델을 스켈레톤으로 한 애니메이션 블루프린트를 생성한다.



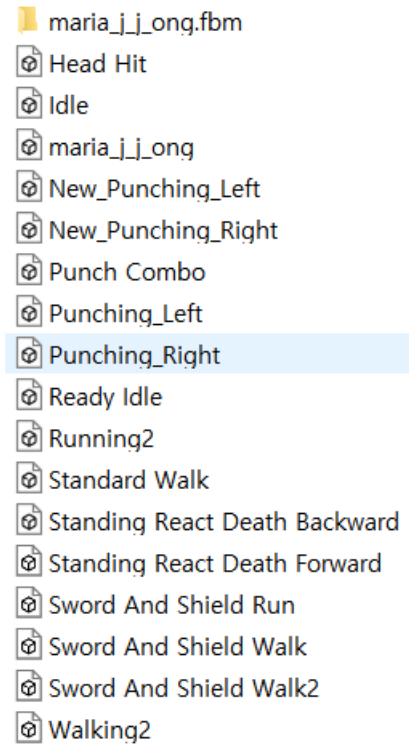
1. 캐릭터와 애니메이션을 얻기 위해서 믹사모에 들어가서 로그인 한다.



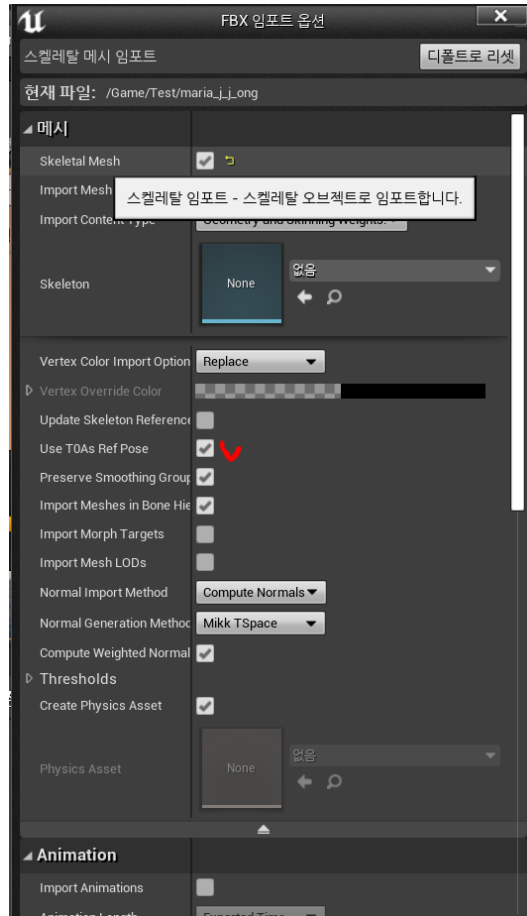
1.1 원하는 캐릭터를 선택한다.



1.2 원하는 애니메이션을 검색하고 나면 다운로드 버튼을 눌러서 다운로드 한다.



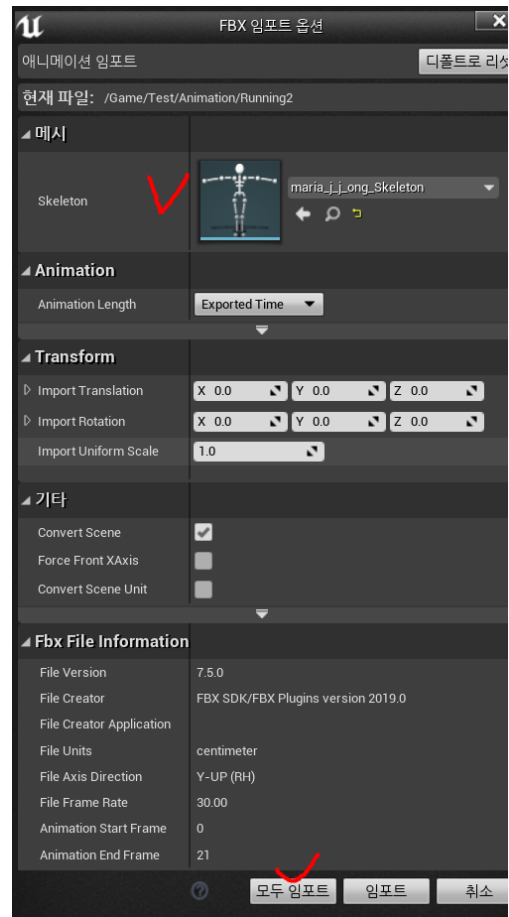
1.3 컴퓨터에 다운로드 된 애니메이션



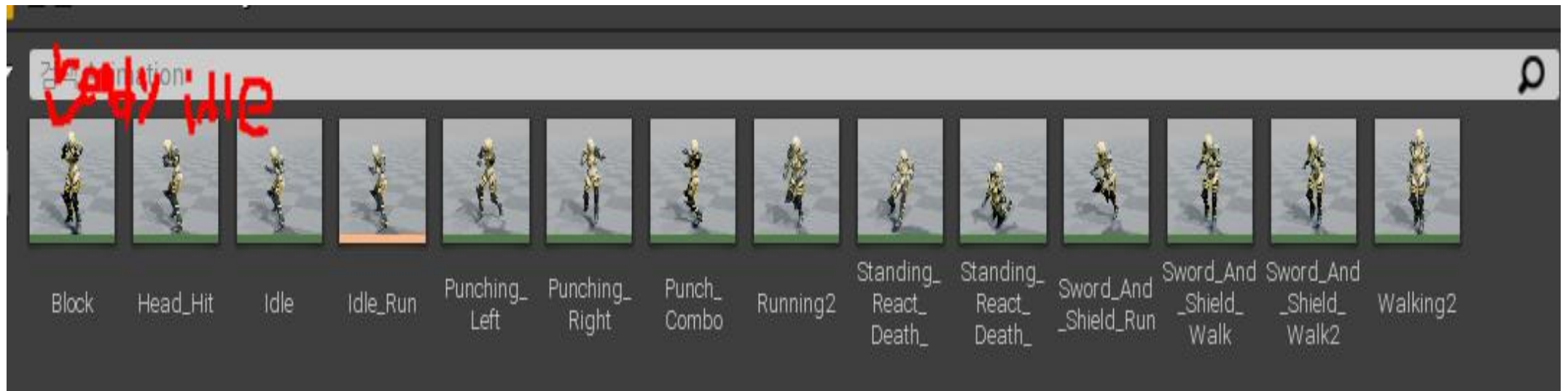
1.4 용량이 많은 텍스처와 같이 있는 모델링 파일을 언리얼에 임포트 한다.



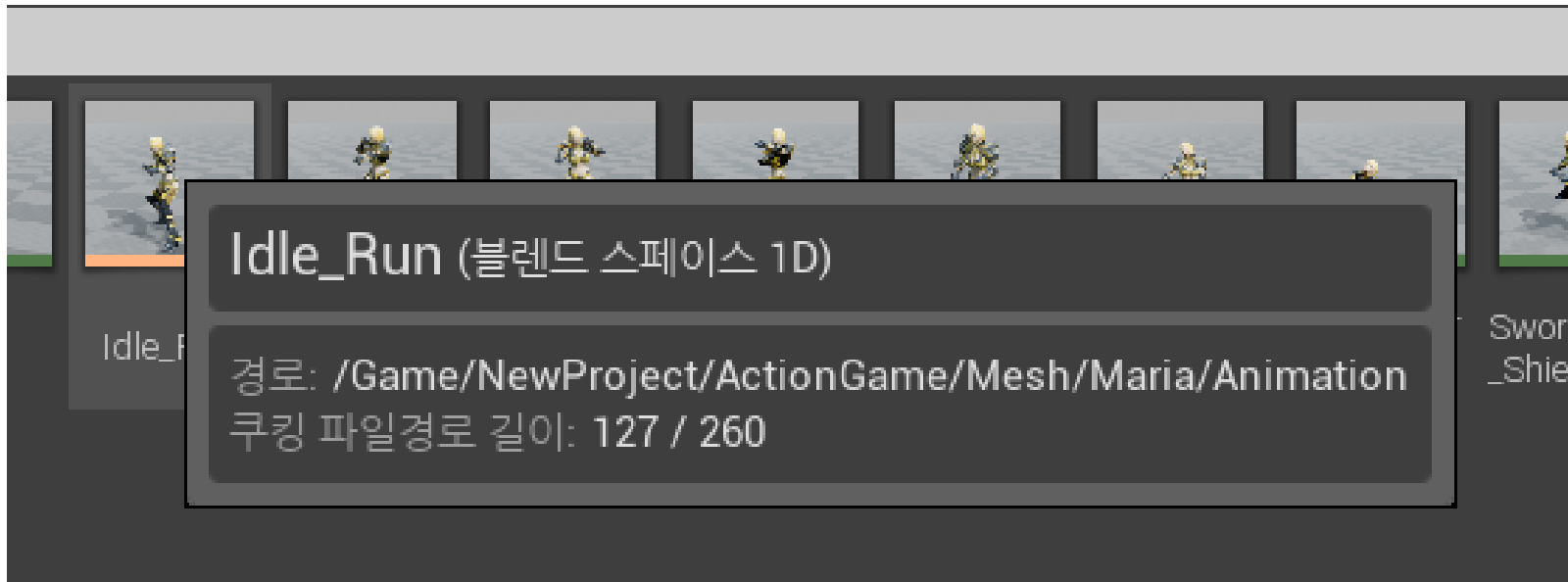
1.4.1 모델링이 импорт 된 모습



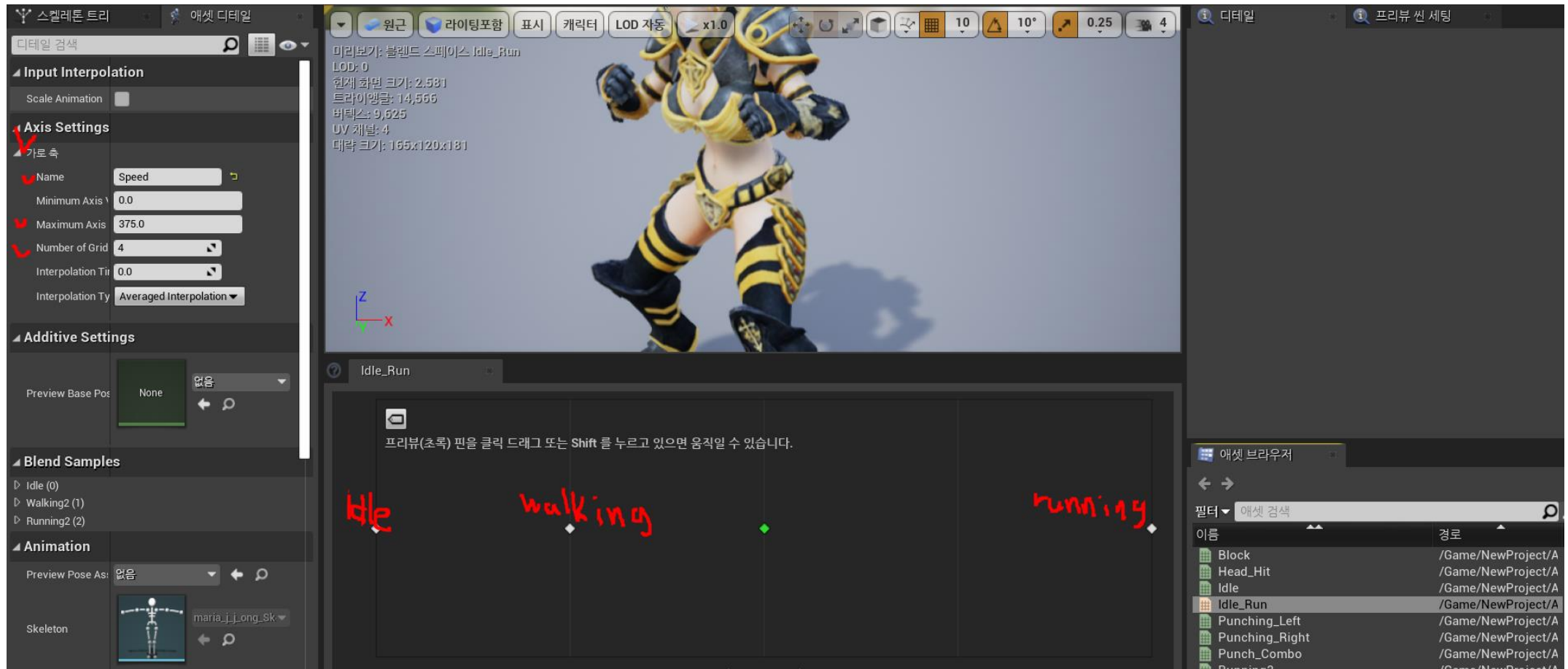
1.5 모델링 파일을 제외한 애니메이션 파일을 언리얼에 임포트한다.



1.51 애니메이션이 импорт 된 모습



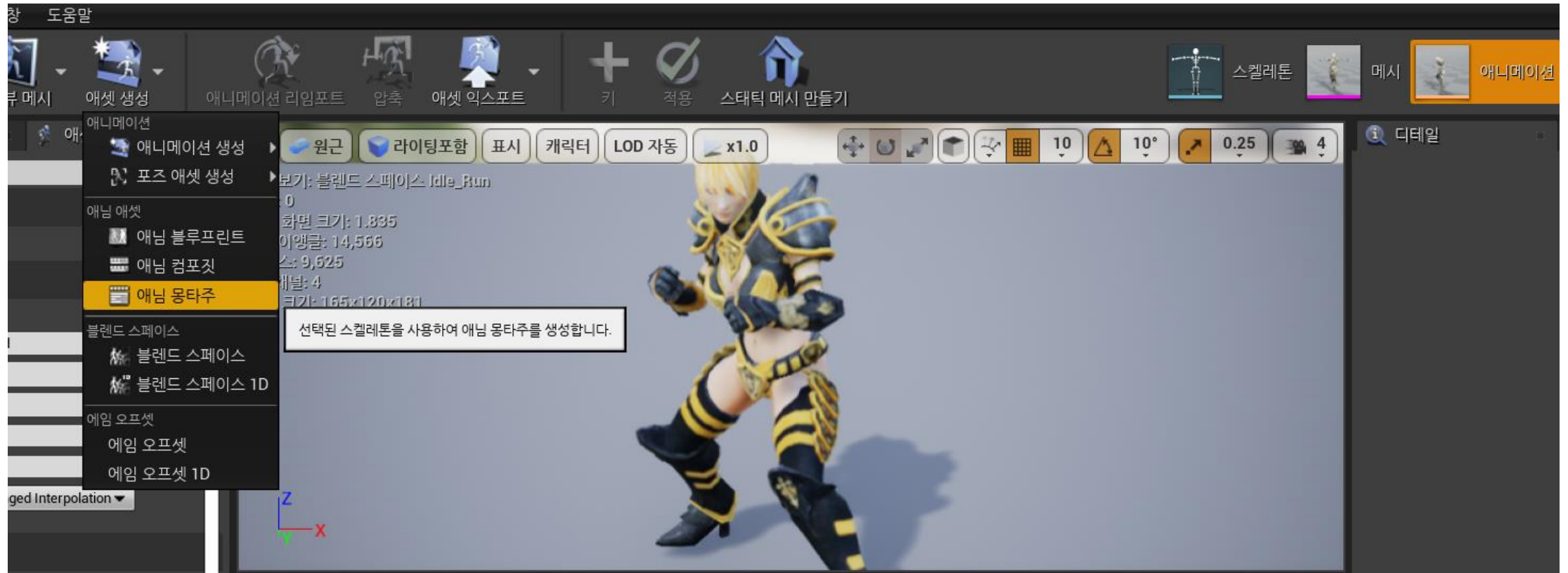
2. 애니메이션에 사용되는 블렌드 스페이스 1D를 생성한다.



2.1 블렌드 스페이스 1D Idle_Run에 들어가서 설정을 한다.



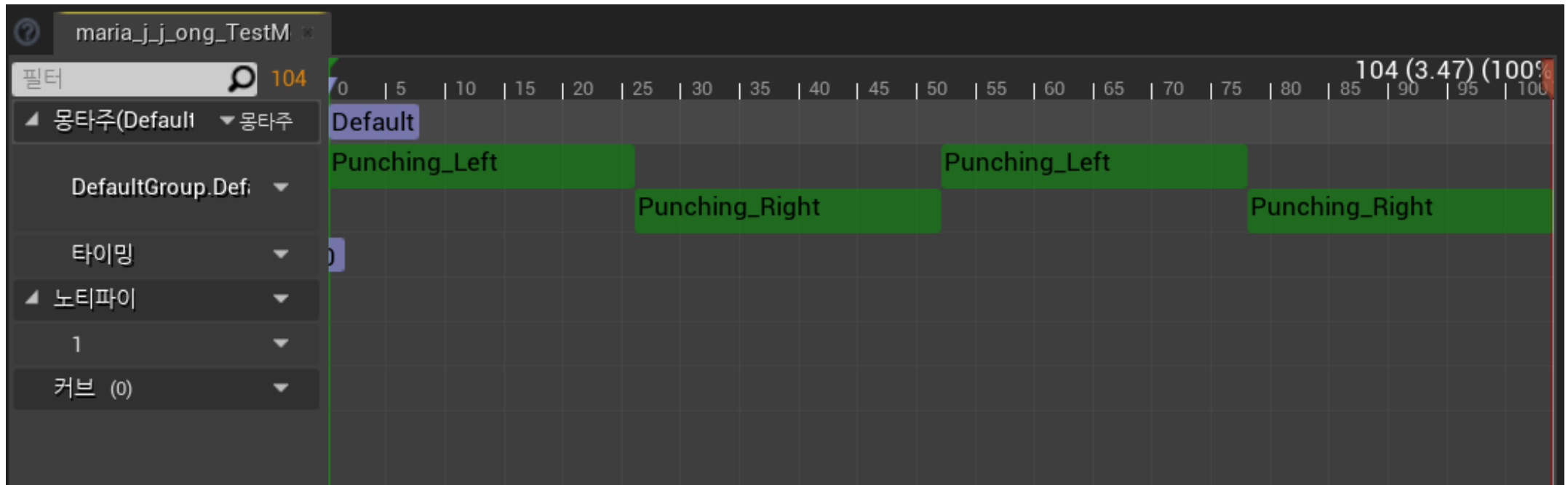
3. Head_Hit 모션과 Death 모션의 애니 몽타주를 생성한다.



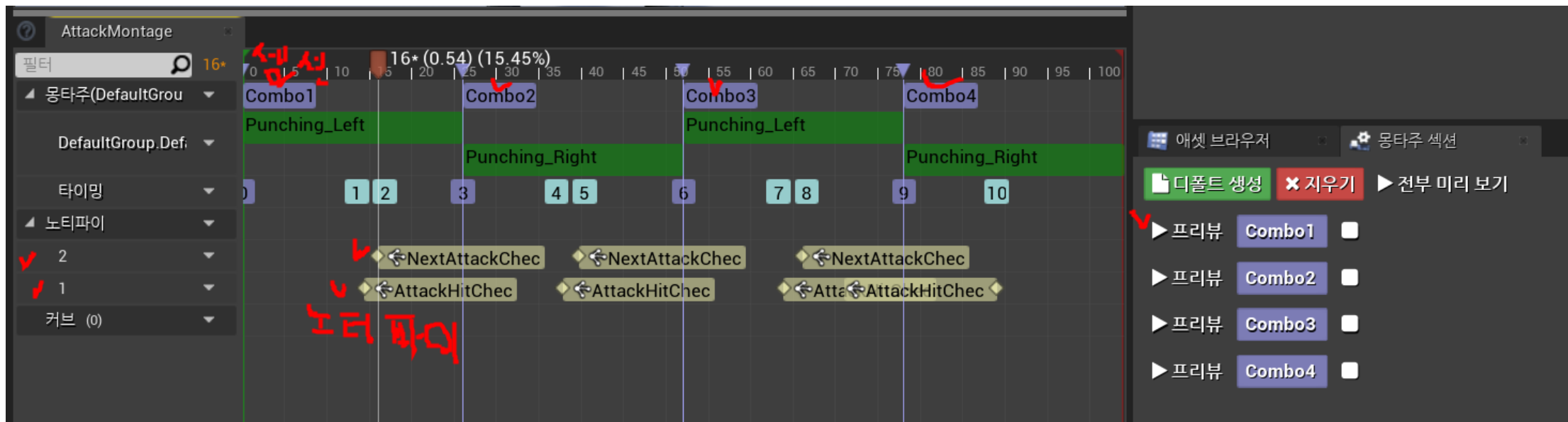
3.1 해당 캐릭터 애니메이션 창을 열어서 애니 몽타주 생성 메뉴를 누른다.



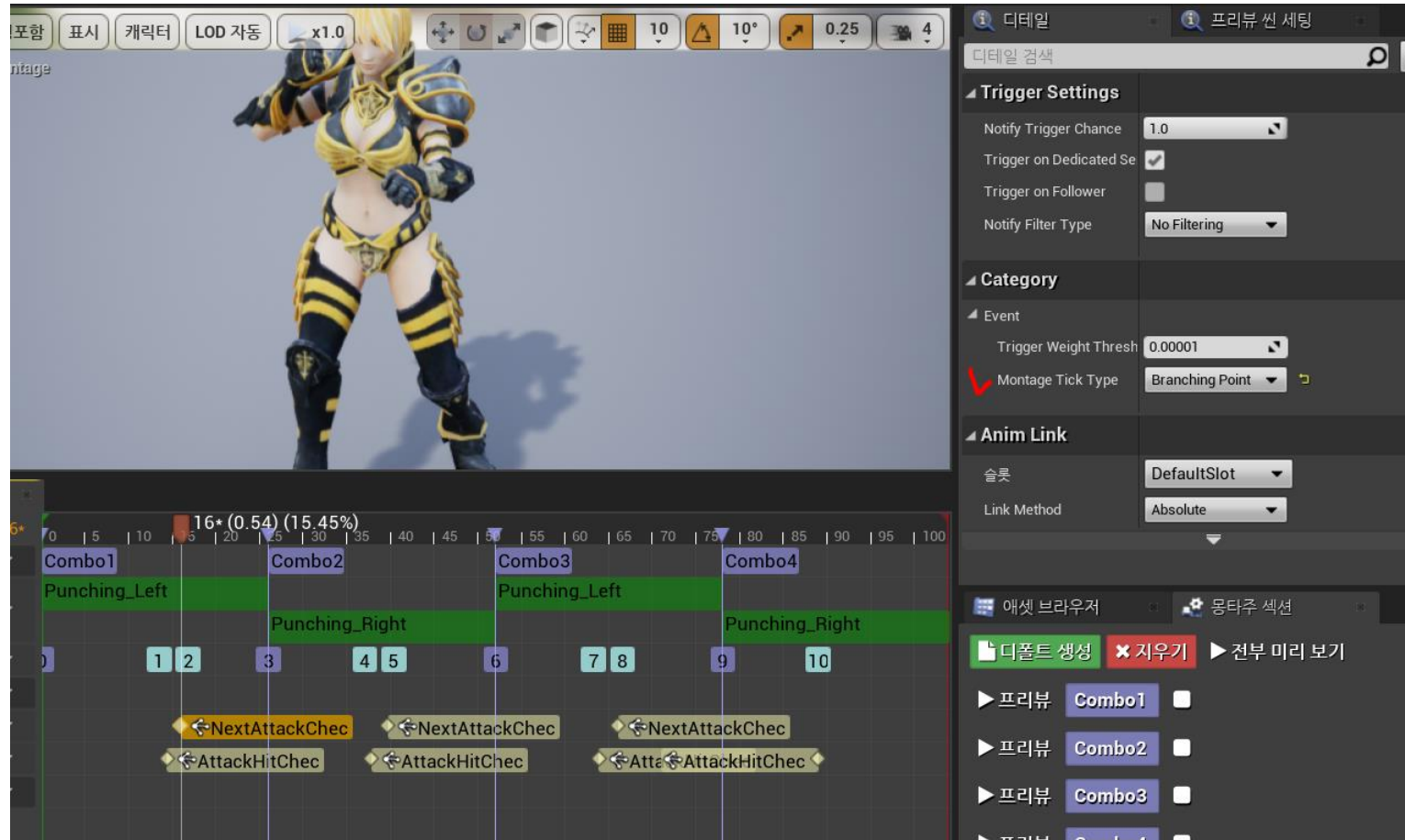
3.2 이름과 저장할 위치를 선택한다.



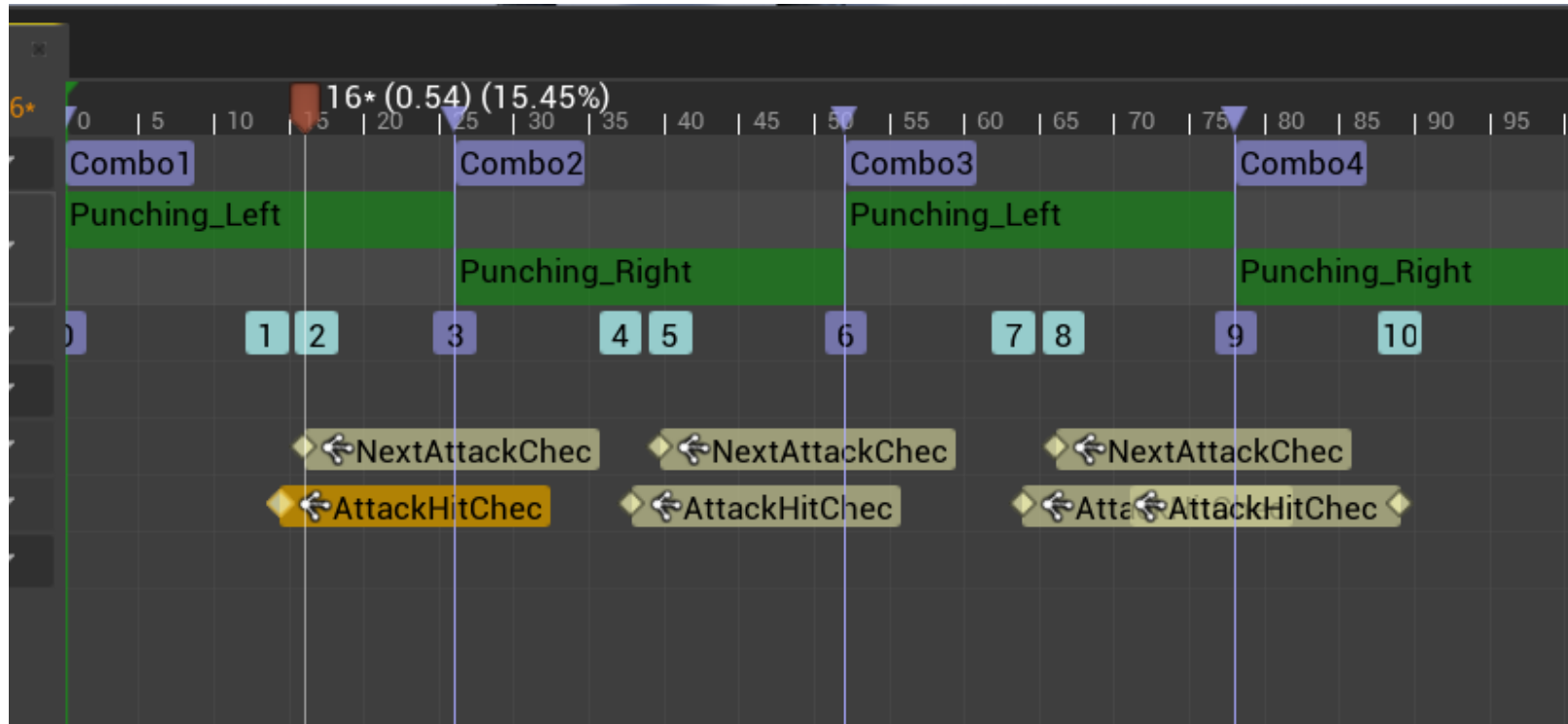
3.3 생성된 몽타주에 애니메이션을 넣는다. 그리고 이름을 AttackMontage로 한다.



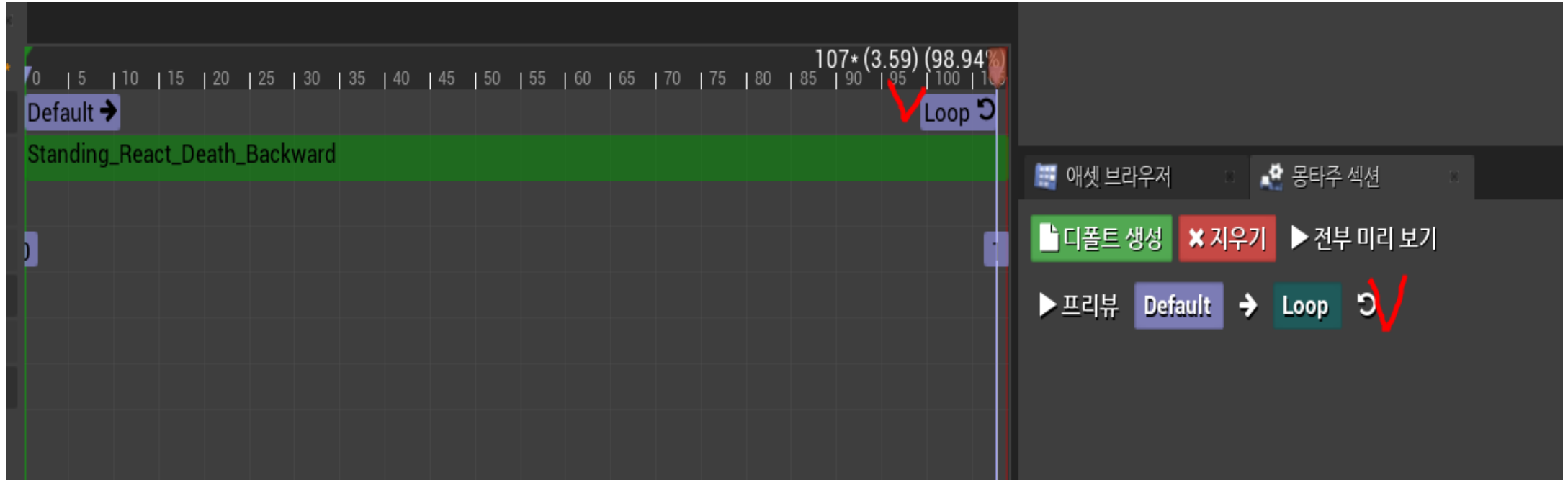
3.4 몽타주 섹션과 노티파이를 추가한다.



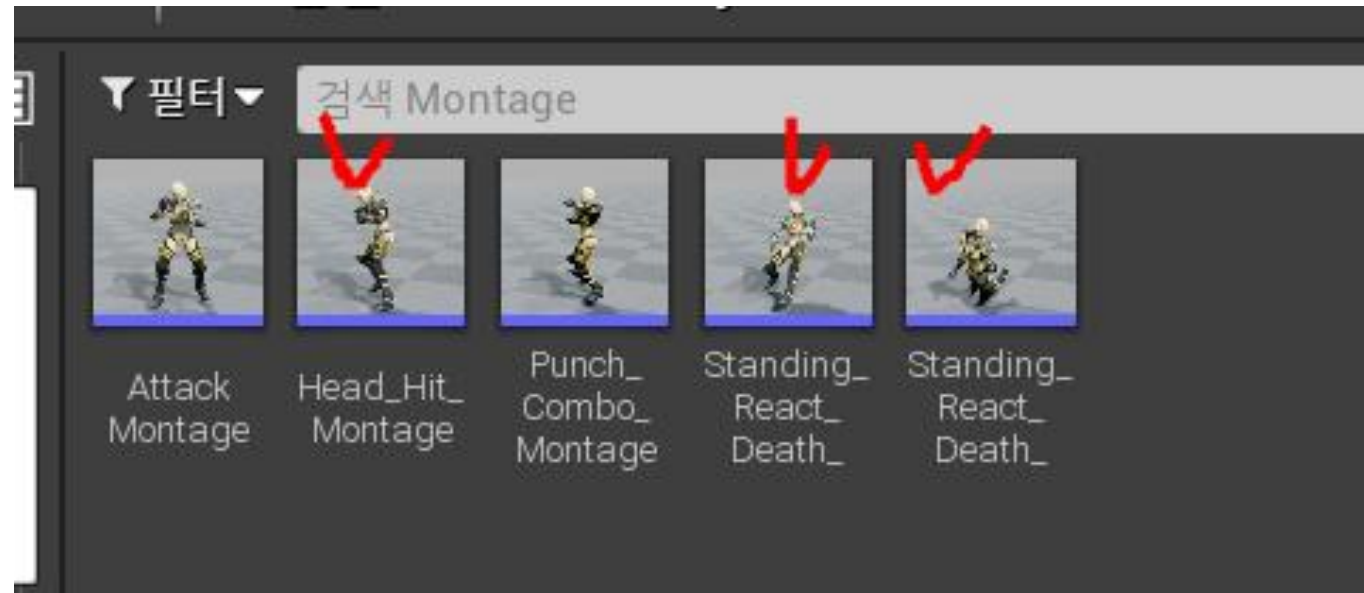
3.5 모든 노티파이에 값을 변경한다.



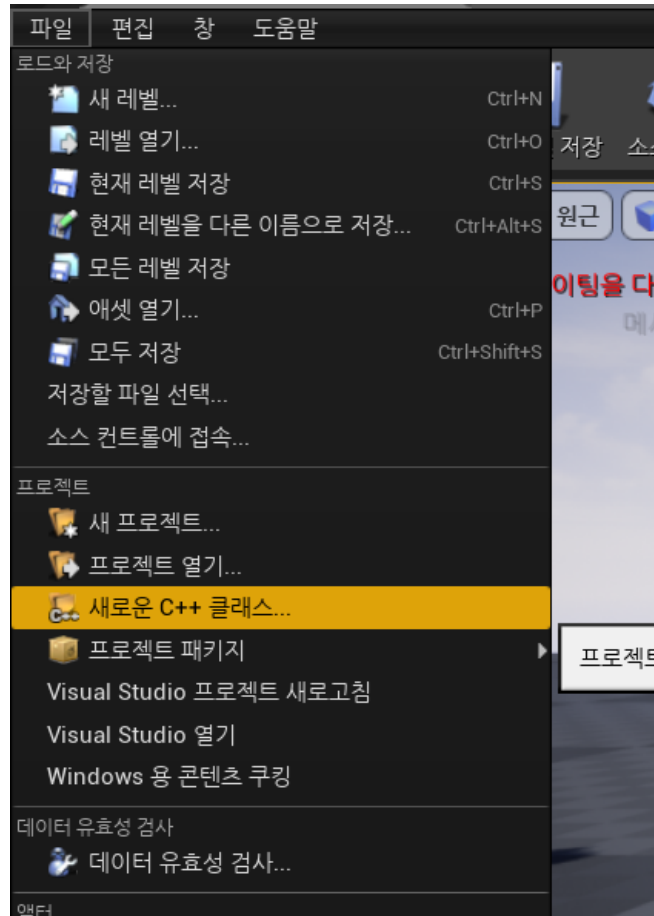
3.6 보충 설명으로 노티파이의 이름은 AttackHitCheck, NextAttackCheck이고, AttackHitCheck는 공격이 들어가는 지점, NextAttackCheck는 다음 콤보를 넣을 수 있는지 확인하는 지점이다.



3.7 Death 모션에 들어가서 Loop라는 몽타주 섹션을 추가하고 애니메이션이 끝나는 지점에 놓는다.



3.8 몽타주가 생성된 모습



4. 메뉴에서 새로운 C++ 클래스를 추가한다.



4.1 부모 클래스로 AnimInstance를 선택하고, 이름을 ABAnimInstance로 한다.

Unreal Engine 4.25.4 설치 옵션



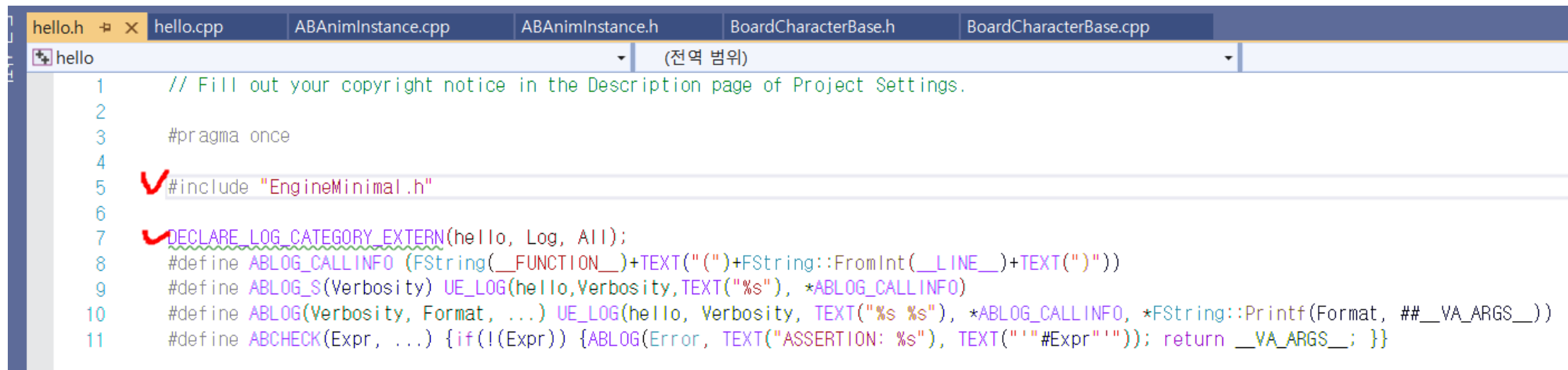
시작용 콘텐츠	867.29 MB	<input checked="" type="checkbox"/>
템플릿 & 피쳐 팩	2.53 GB	<input checked="" type="checkbox"/>
엔진 소스	178.14 MB	<input checked="" type="checkbox"/>
 디버깅을 위한 편집기 기호	27.34 GB	<input checked="" type="checkbox"/>
▼ 타겟 플랫폼		

다운로드 크기: 0.00 B

필요한 저장소 공간: 62.79 GB

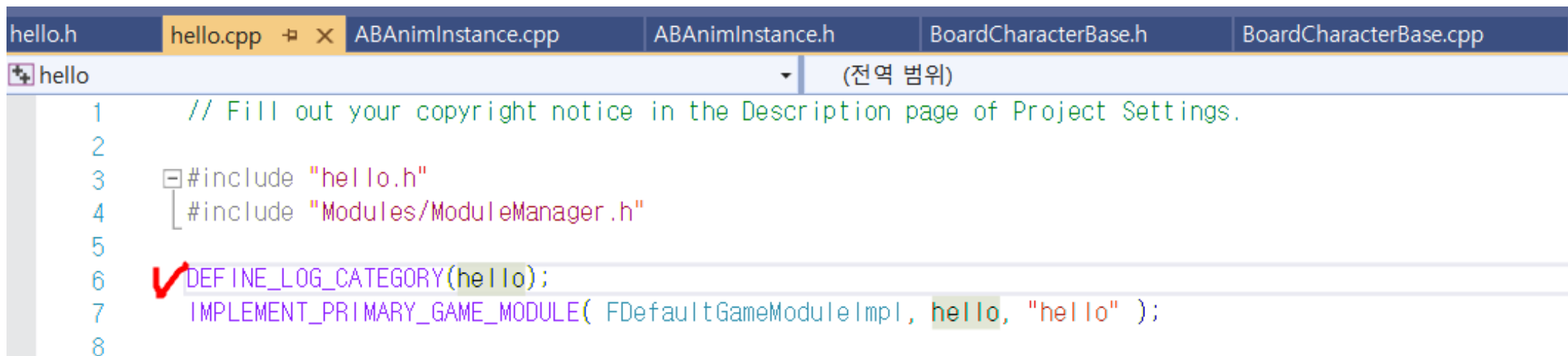
적용

4.2 자신이 사용중인 엔진의 옵션에 들어가서 해당 사항을 체크하고 적용한다.



```
hello.h x hello.cpp ABAnimInstance.cpp ABAnimInstance.h BoardCharacterBase.h BoardCharacterBase.cpp
hello (전역 범위)
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 ✓ #include "EngineMinimal.h"
6
7 ✓ DECLARE_LOG_CATEGORY_EXTERN(hello, Log, All);
8 #define ABLOG_CALLINFO (FString(__FUNCTION__)+TEXT("(")+FString::FromInt(__LINE__)+TEXT(")"))
9 #define ABLOG_S(Verbosity) UE_LOG(hello, Verbosity, TEXT("%s"), *ABLOG_CALLINFO)
10 #define ABLOG(Verbosity, Format, ...) UE_LOG(hello, Verbosity, TEXT("%s %s"), *ABLOG_CALLINFO, *FString::Printf(Format, ##__VA_ARGS__))
11 #define ABCHECK(Expr, ...) {if(!(Expr)) {ABLOG(Error, TEXT("ASSERTION: %s"), TEXT("'"#Expr"'))}; return __VA_ARGS__; }}
```

4.3 프로젝트 이름의 헤더 파일에 들어가서 내용을 추가한다.



```
hello.h | hello.cpp | ABAAnimInstance.cpp | ABAAnimInstance.h | BoardCharacterBase.h | BoardCharacterBase.cpp
hello (전역 범위)
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #include "hello.h"
4 #include "Modules/ModuleManager.h"
5
6 DEFINE_LOG_CATEGORY(hello);
7 IMPLEMENT_PRIMARY_GAME_MODULE( FDefaultGameModuleImpl, hello, "hello" );
8
```

4.4 프로젝트 이름의 CPP 파일에 들어가서 내용을 추가한다.

```
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "EngineMinimal.h"
6
7 DECLARE_LOG_CATEGORY_EXTERN(hello, Log, All);
8 #define ABLOG_CALLINFO (FString(__FUNCTION__)+TEXT("(")+FString::FromInt(__LINE__)+TEXT("))")
9 #define ABLOG_S(Verbosity) UE_LOG(hello, Verbosity, TEXT("%s"), *ABLOG_CALLINFO)
10 #define ABLOG(Verbosity, Format, ...) UE_LOG(hello, Verbosity, TEXT("%s %s"), *ABLOG_CALLINFO, *FString::Printf(Format, ##__VA_ARGS__))
11 #define ABCHECK(Expr, ...) {if(!(Expr)) {ABLOG(Error, TEXT("ASSERTION: %s"), TEXT("'"#Expr"'))}; return __VA_ARGS__; }}
```

4.5 다시 헤더 파일로 넘어가서 내용을 추가한다.

```
hello.h  hello.cpp  ABAnimInstance.h  ABAnimInstance.cpp  BoardCharacterBase.h  BoardCharacterBase.cpp
hello
1  // Fill out your copyright notice in the Description page of Project Settings.
2
3  #pragma once
4
5  ✓ #include "hello.h"
6  #include "Animation/AnimInstance.h"
7  #include "ABAnimInstance.generated.h"
8
9  /**
10   *
11   */
12
13  DECLARE_DYNAMIC_MULTICAST_DELEGATE(FOnNextAttackCheckDelegate);
14  DECLARE_DYNAMIC_MULTICAST_DELEGATE(FOnAttackHitCheckDelegate);
15
16
17  UCLASS()
18  class HELLO_API UABAnimInstance : public UAnimInstance
19  {
20  GENERATED_BODY()
```

4.6 ABAnimInstance.h에 들어가서 내용을 넣는다. 별표친 것은 매우 중요하다

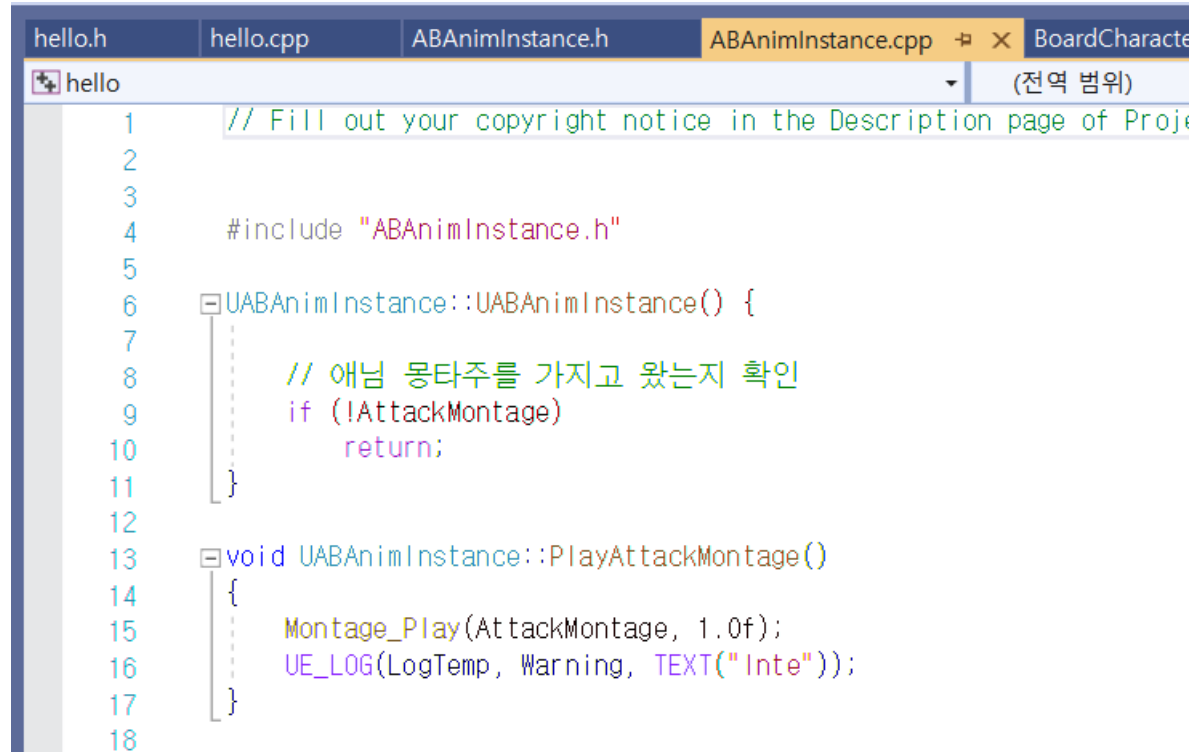
```
hello.h  hello.cpp  ABAnimInstance.h  ABAnimInstance.cpp  BoardCharacterBase.h  Bc
hello  ▾  UABAnimInstance
16
17  UCLASS()
18  class HELLO_API UABAnimInstance : public UAnimInstance
19  {
20      GENERATED_BODY()
21
22  public:
23
24      // 생성자
25      UABAnimInstance();
26
27      UFUNCTION(BlueprintCallable)
28      // 몽타주 애니메이션을 재생
29      void PlayAttackMontage();
30
31      UFUNCTION(BlueprintCallable)
32      // 다음 섹션으로 이동
33      void JumpToAttackMontageSection(int32 NewSection);
34
35      UPROPERTY(BlueprintAssignable)
36      FOnNextAttackCheckDelegate OnNextAttackCheck;
37
38      UPROPERTY(BlueprintAssignable)
39      FOnAttackHitCheckDelegate OnAttackHitCheck;
40
```

4.6.1 클래스에 들어가서 내용을 추가한다.

```
hello.h x hello.cpp ABAnimInstance.h x ABAnimInstance.cpp BoardCharacterBase.h
D:\unreal_workspace\hello\Source\hello\hello.h UABAnimInstance

34
35 UPROPERTY(BlueprintAssignable)
36 FOnNextAttackCheckDelegate OnNextAttackCheck;
37
38 UPROPERTY(BlueprintAssignable)
39 FOnAttackHitCheckDelegate OnAttackHitCheck;
40
41 private:
42     // 특정 타이밍에 공격을 체크
43     UFUNCTION()
44     void AnimNotify_AttackHitCheck();
45
46     // 다음 공격으로 넘어가는 공격을 체크
47     UFUNCTION()
48     void AnimNotify_NextAttackCheck();
49
50     // 섹션의 이름을 출력하는 함수
51     FName GetAttackMontageSectionName(int32 Section);
52
53 public:
54
55     // 공격 몽타주 예셋
56     UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category = Attack)
57     UAnimMontage* AttackMontage;
58
59 };
```

4.6.2 클래스에 내용을 추가한다.



The screenshot shows a code editor with several tabs at the top: 'hello.h', 'hello.cpp', 'ABAnimInstance.h', 'ABAnimInstance.cpp' (which is the active tab), and 'BoardCharacter'. The active tab contains the following C++ code:

```
1 // Fill out your copyright notice in the Description page of Project Settings
2
3
4 #include "ABAnimInstance.h"
5
6 UABAnimInstance::UABAnimInstance() {
7     // 애님 몽타주를 가지고 왔는지 확인
8     if (!AttackMontage)
9         return;
10 }
11
12
13 void UABAnimInstance::PlayAttackMontage()
14 {
15     Montage_Play(AttackMontage, 1.0f);
16     UE_LOG(LogTemp, Warning, TEXT("Inte"));
17 }
18
```

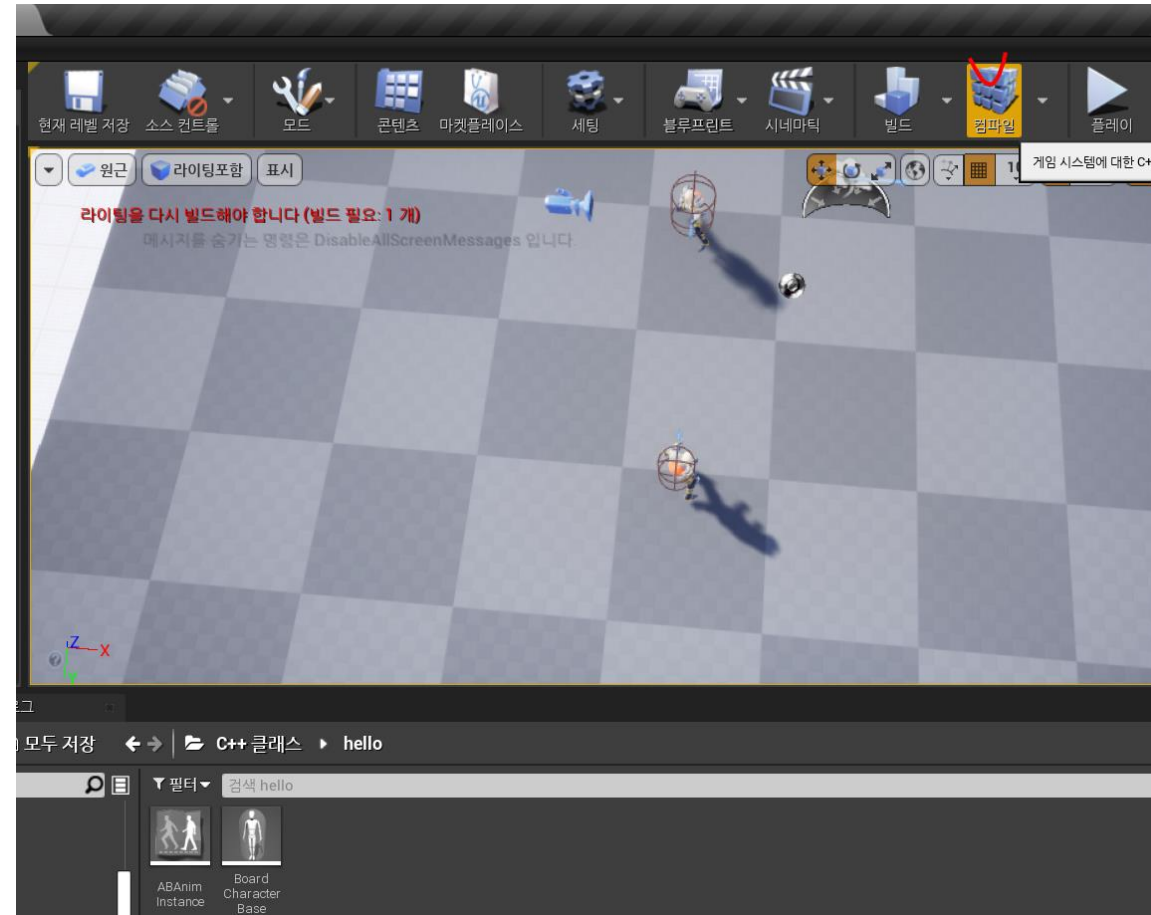
4.7 ABAnimInstance.cpp에 들어가서 내용을 추가한다.

```
hello.h | hello.cpp | ABAnimInstance.h | ABAnimInstance.cpp | BoardCharacterBase.h | BoardCharacterBase.cpp
hello (전역 범위)
16     UE_LOG(LogTemp, Warning, TEXT("Inte"));
17 }
18
19 void UABAnimInstance::JumpToAttackMontageSection(int32 NewSection)
20 {
21     ABHECK(Montage_IsPlaying(AttackMontage));
22     UE_LOG(LogTemp, Warning, TEXT("Combo: %d"), NewSection);
23     Montage_JumpToSection(GetAttackMontageSectionName(NewSection), AttackMontage);
24 }
25
26 void UABAnimInstance::AnimNotify_AttackHitCheck()
27 {
28     ABLOG_S(Warning);
29     OnAttackHitCheck.Broadcast();
30 }
31
32 void UABAnimInstance::AnimNotify_NextAttackCheck()
33 {
34     ABLOG_S(Warning);
35     OnNextAttackCheck.Broadcast();
36 }
37
38 FName UABAnimInstance::GetAttackMontageSectionName(int32 Section)
39 {
40     ABHECK(FMath::IsWithinInclusive<int32>(Section, 1, 4), NAME_None);
41     return FName(*FString::Printf(TEXT("Combo%d"), Section));
42 }
```

4.8 함수들에 내용을 추가한다.



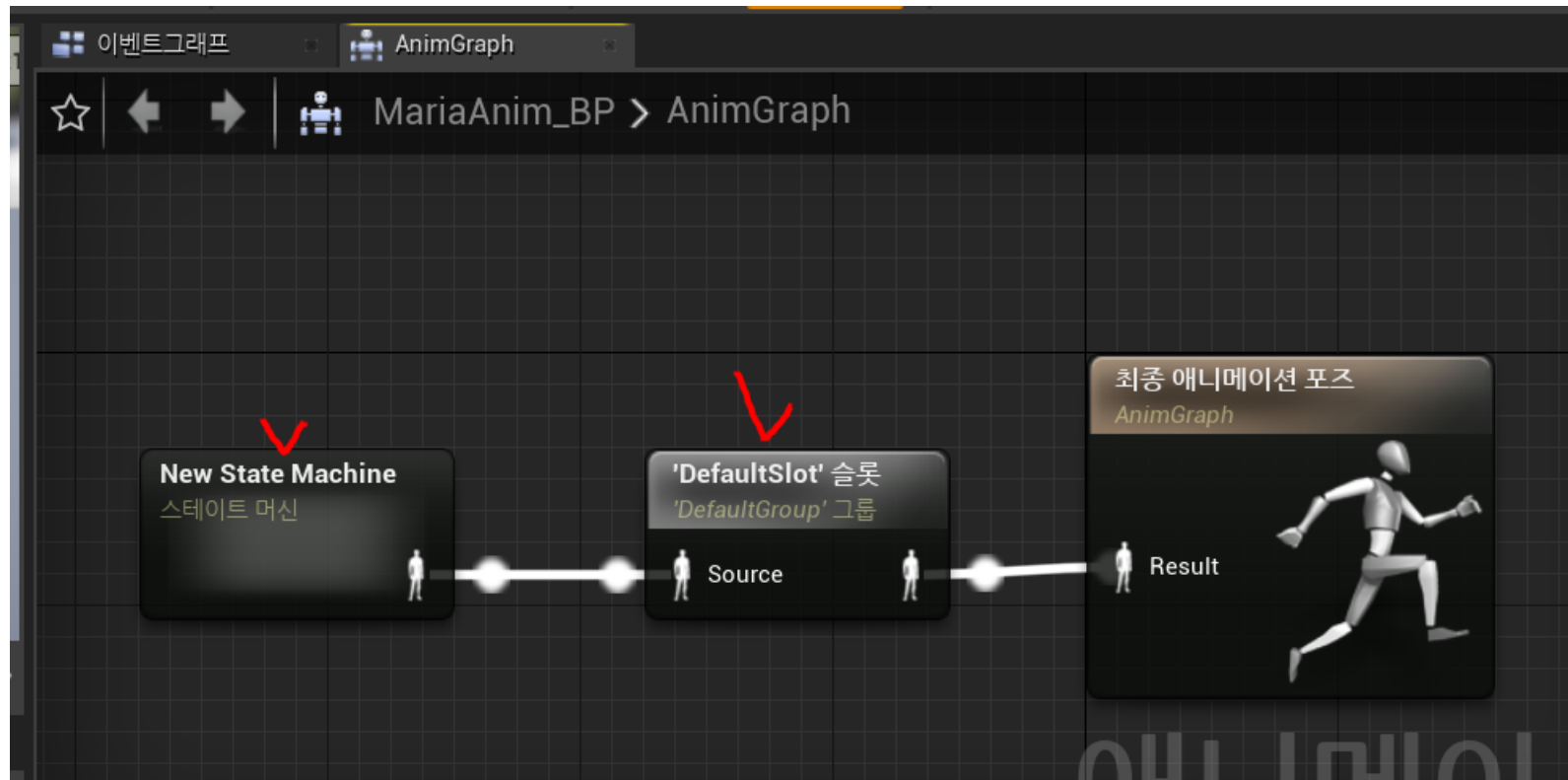
4.9 코드가 완성되면 저장하고 빌드한다.



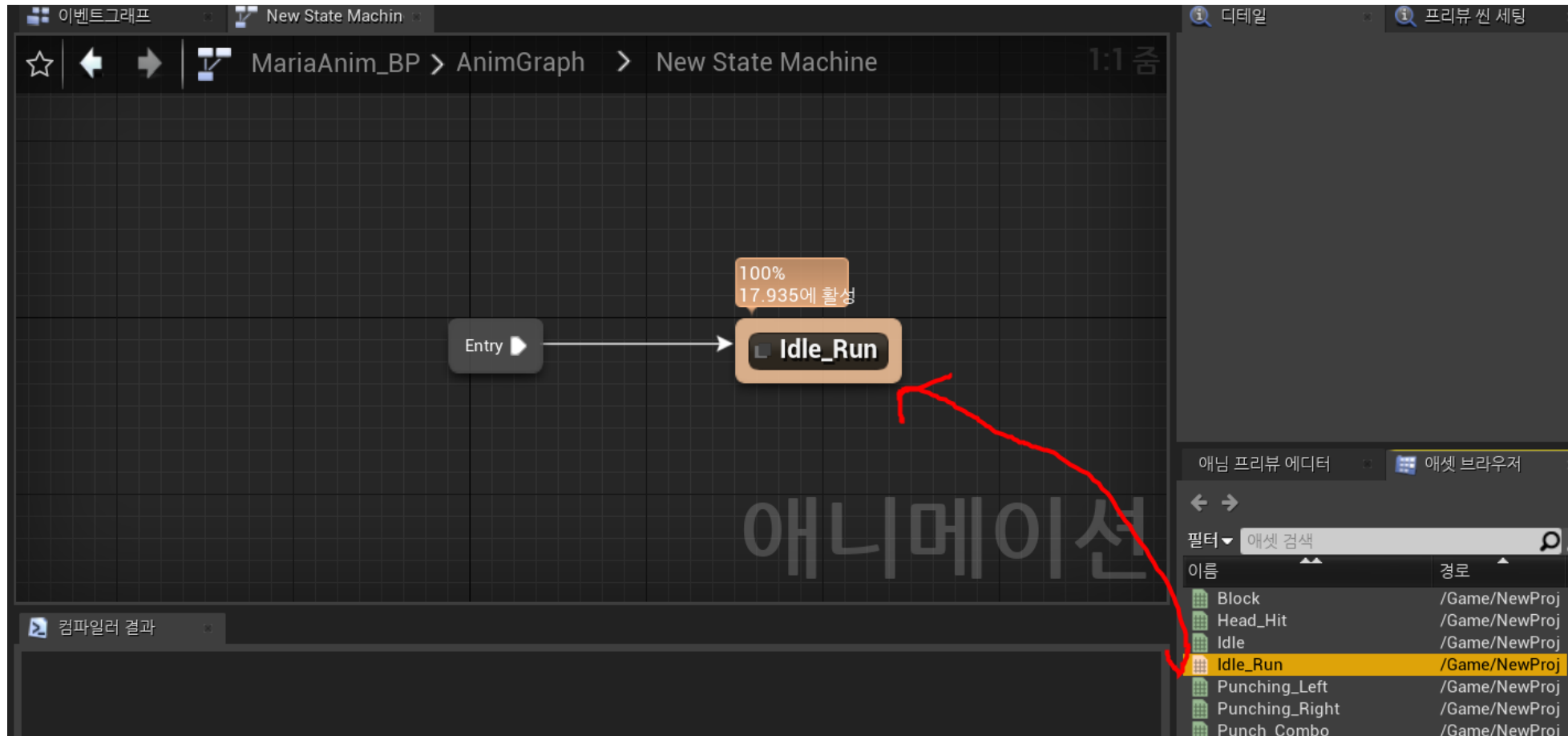
4.10 엔진으로 넘어가서 컴파일 버튼을 눌러서 컴파일한다



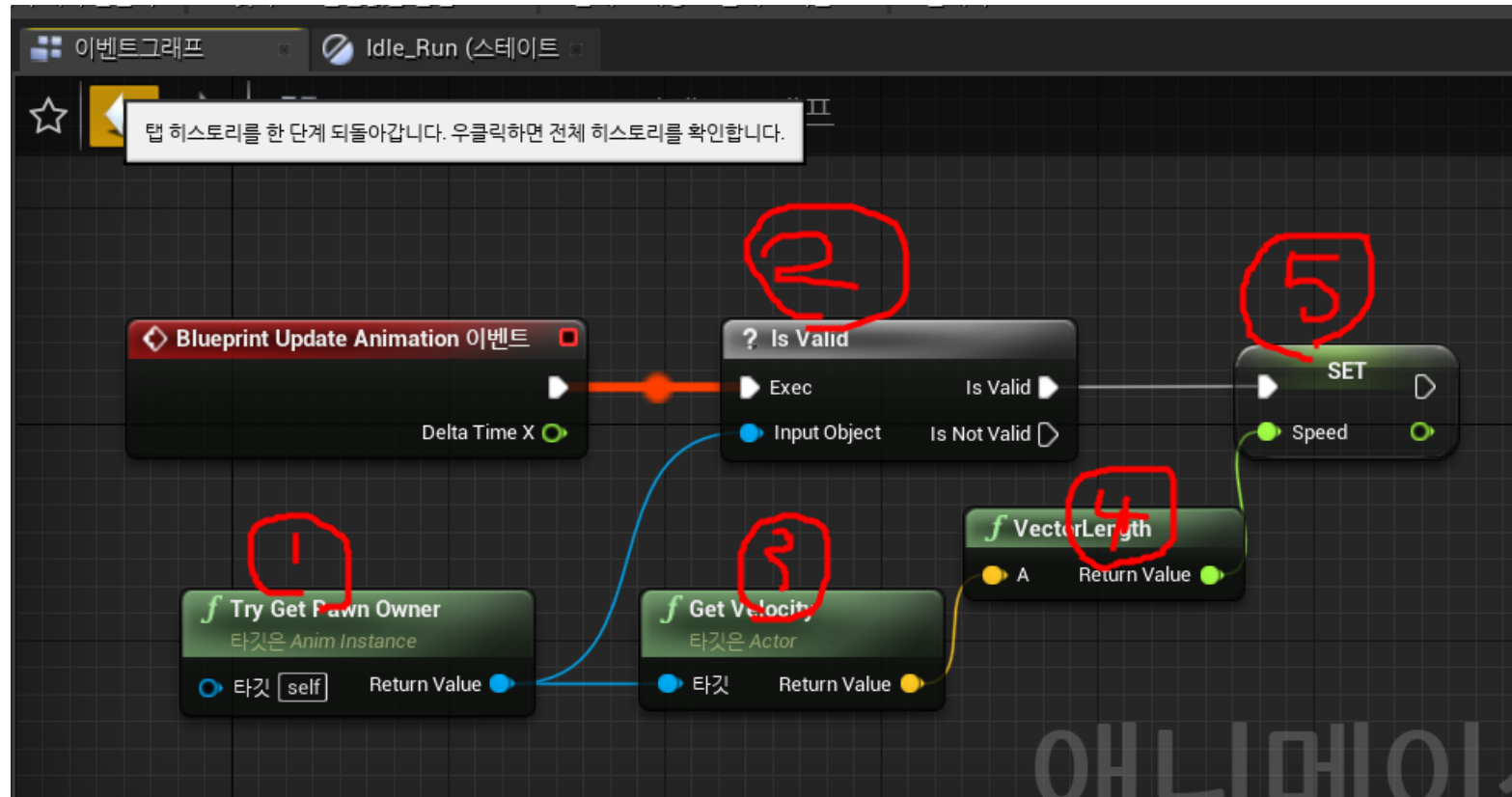
5. ABAnimInstace를 부모로 하고 믹사모에서 가져온 모델을 스켈레톤으로 한 애니메이션 블루프린트를 생성한다.



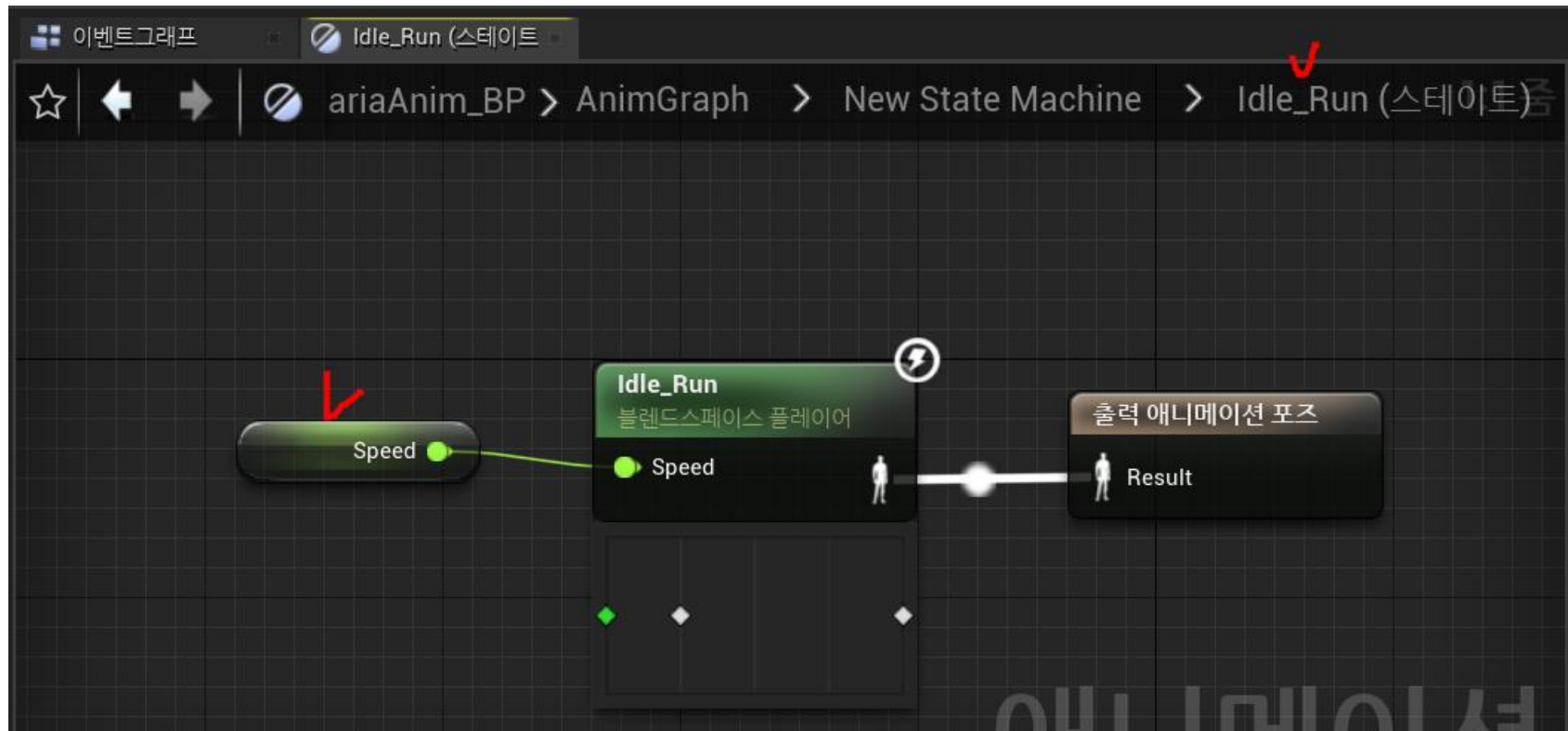
5.1 State Machine과 DefalutSlot를 생성해서 연결한다.



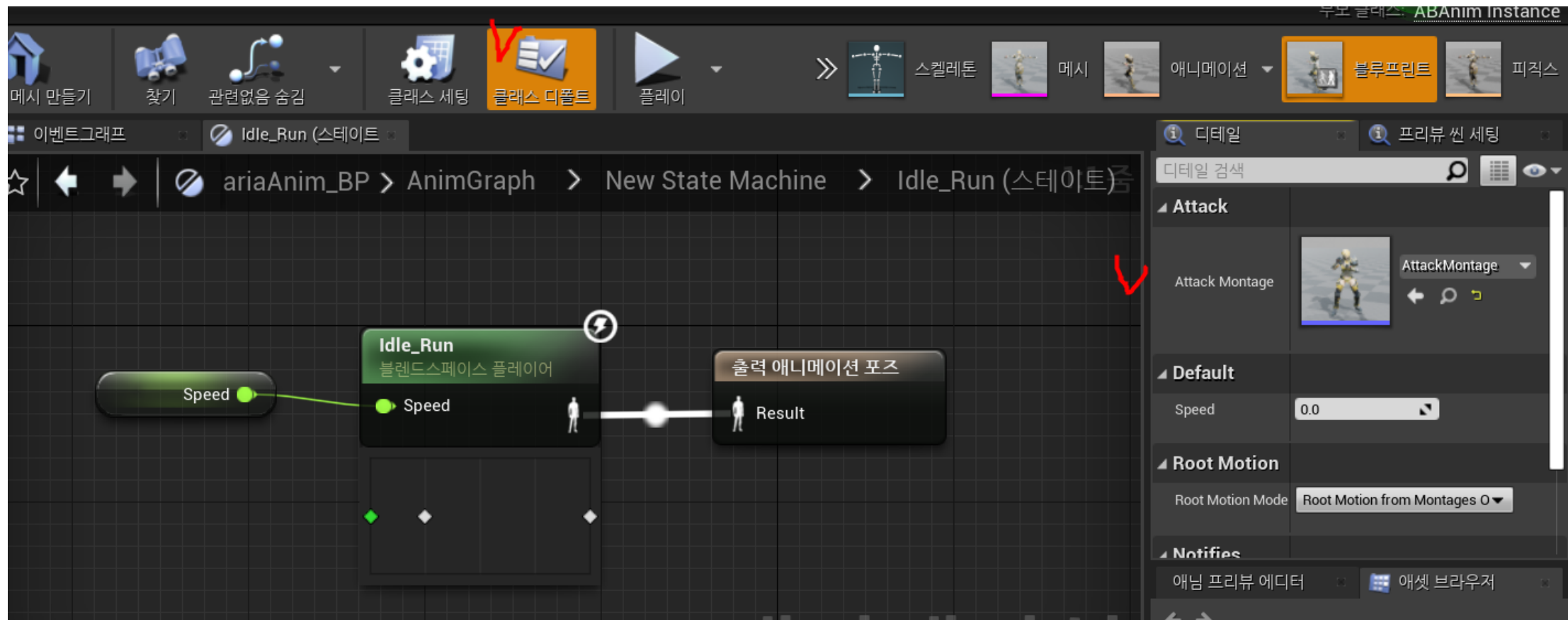
5.2 NewStateMachine에 들어가서 애니메이션을 가져오고 연결한다.



5.3 이벤트 그래프로 넘어가서 float 형 Speed 변수를 생성한다.



5.4 스테이트 머신의 Idle_Run에 들어가서 Speed 변수를 Idle_Run의 Speed에 연결한다.



5.5 ABAnimInstance에 추가한 변수 AttackMontage에 AttackMontage 애님 몽타주를 넣는다.