



경성대학교

# 게임프로그래밍

언리얼 엔진

2019775045 이상배

소프트웨어학과

01

**언리얼 엔진 소개**

---

02

**언리얼 엔진 설치 및 실행 (프로젝트 생성)**

---

03

**언리얼 엔진 내 기능 소개**

---

04

**폰을 움직여보자! (블루프린트)**

---

05

**폰을 움직여보자! (C++)**

---

# 언리얼 엔진 소개

Chapter 1

## 언리얼 엔진이란?

>>> Real-time 3D Creation Tool



# UNREAL ENGINE

### 개요

- 그래픽 솔루션 창작 툴
- 게임 뿐만 아닌 영화 및 애니메이션도 창작 할 수 있는 툴
- 대규모 개발을 위한 툴

### 활용 사례

- 게임 : 배틀그라운드, 로스트아크, etc.
- 드라마 : 유미네 세포들, etc.

### 사용 언어

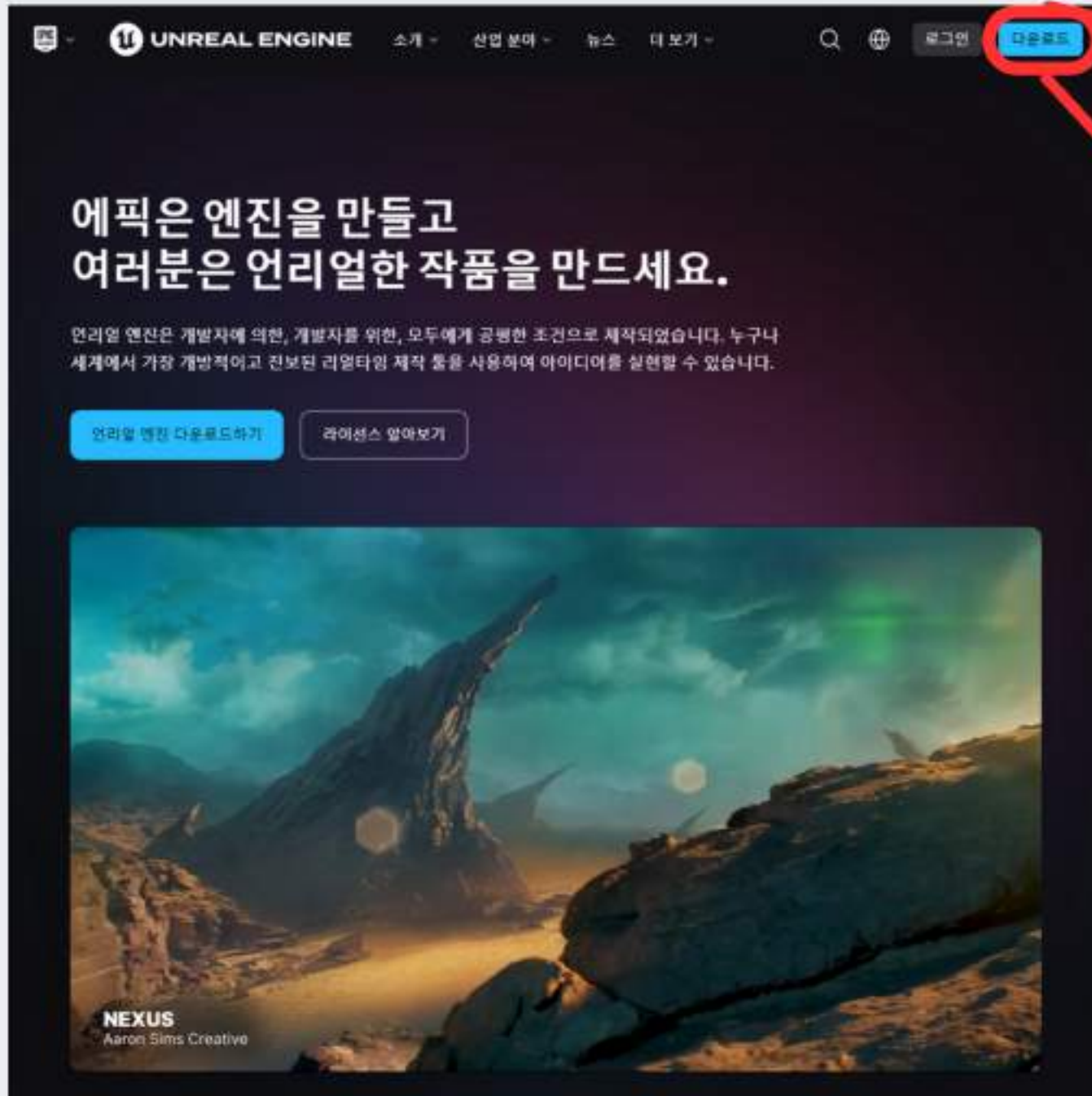
- C++, 비주얼 스크립팅 (BluePrint)

# 언리얼 엔진 설치 및 실행 (프로젝트 생성)

Chapter 2

# 언리얼 엔진 설치

>>> 언리얼 엔진 설치를 해보자



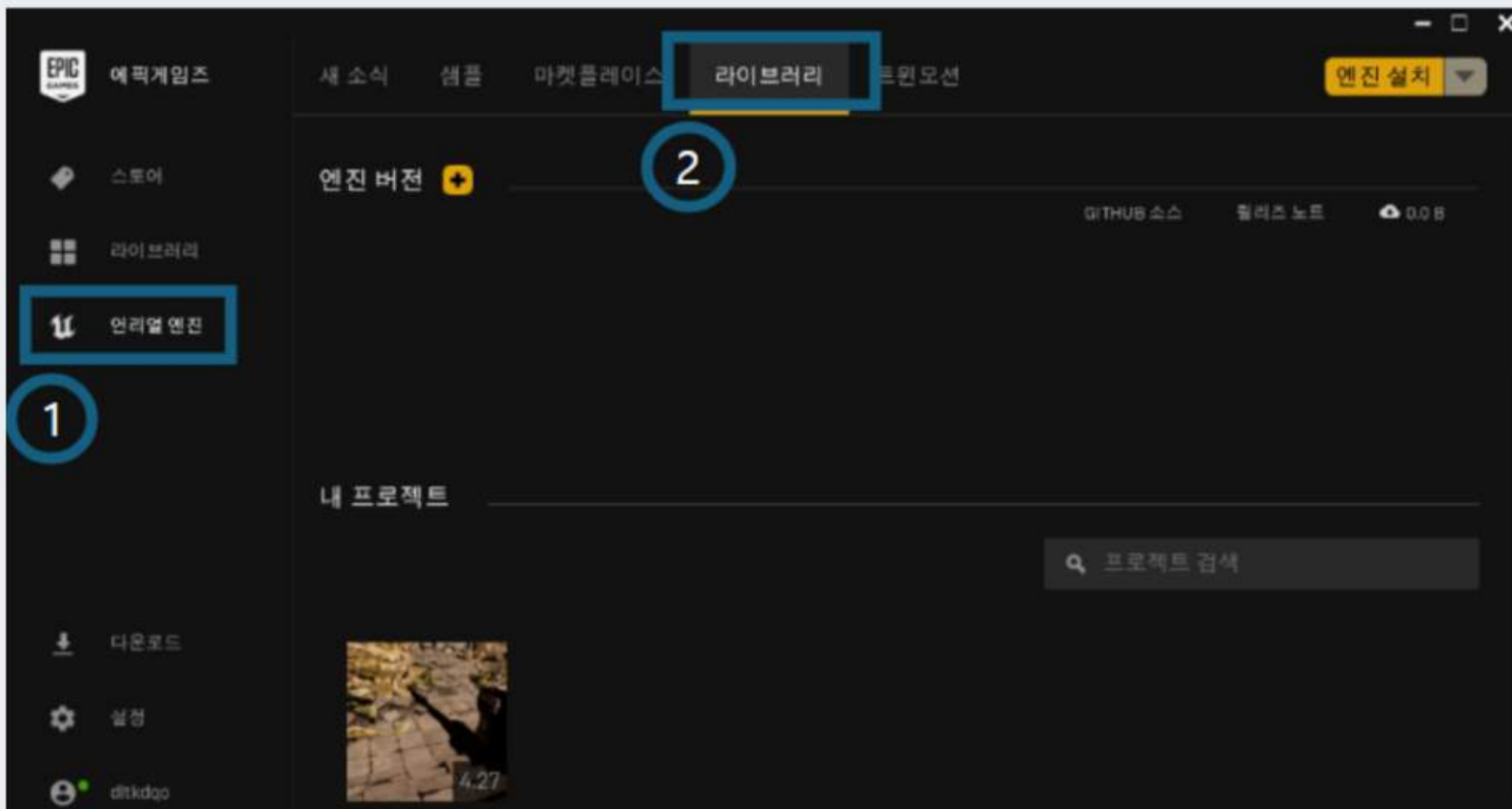
<https://www.unrealengine.com/ko>

에 들어가서 에픽 게임즈 런처를 다운받는다



# 언리얼 엔진 설치


>>> 언리얼 엔진 설치를 해보자

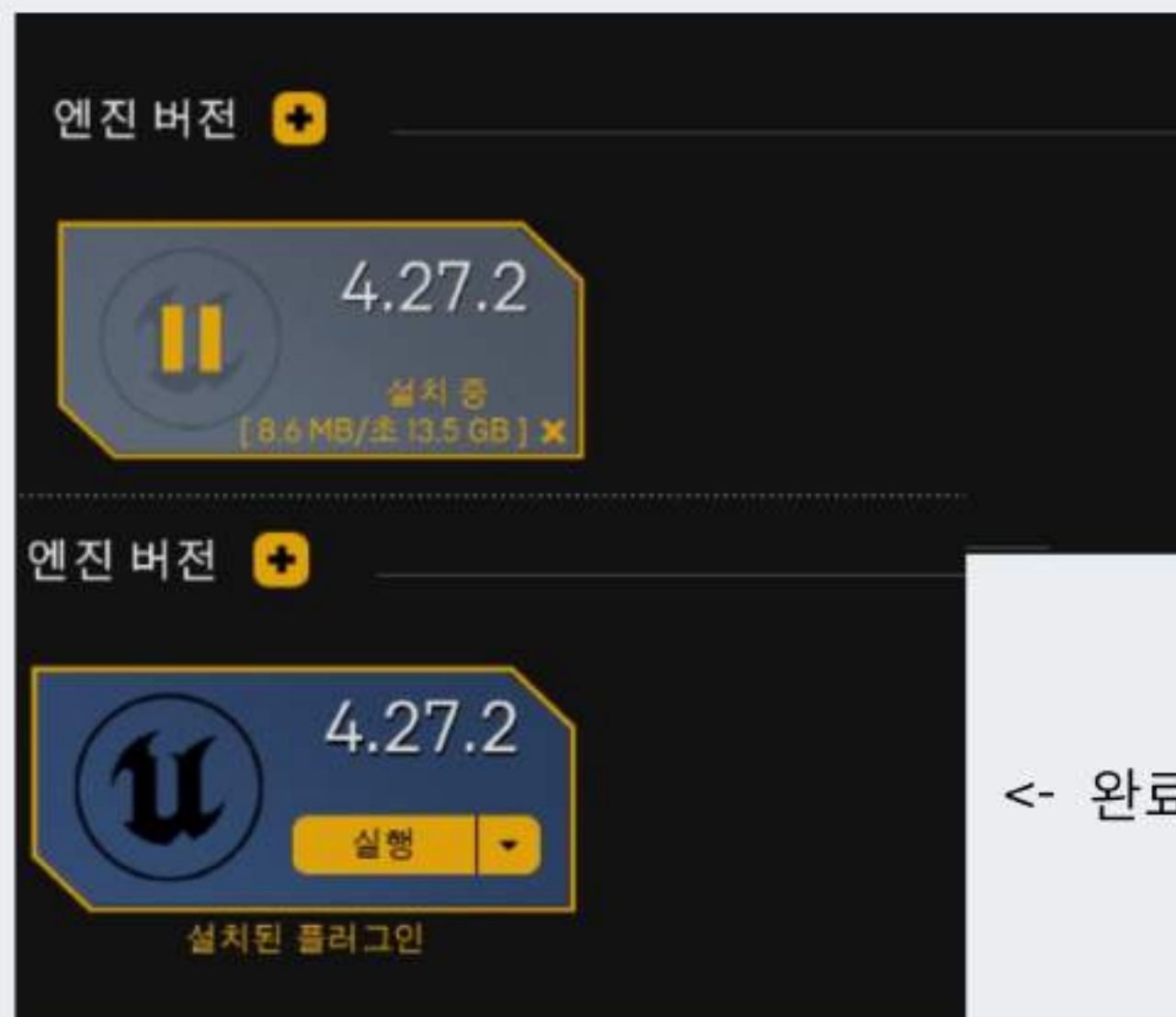


다운이 완료되면 언리얼 엔진 실행 후  
언리얼엔진 -> 라이브러리 항목으로 들어간다.

## 언리얼 엔진 설치

>>> 언리얼 엔진 설치를 해보자

 버튼을 누른 후 원하는 버전을 선택해 다운 받는다.



<- 완료 화면

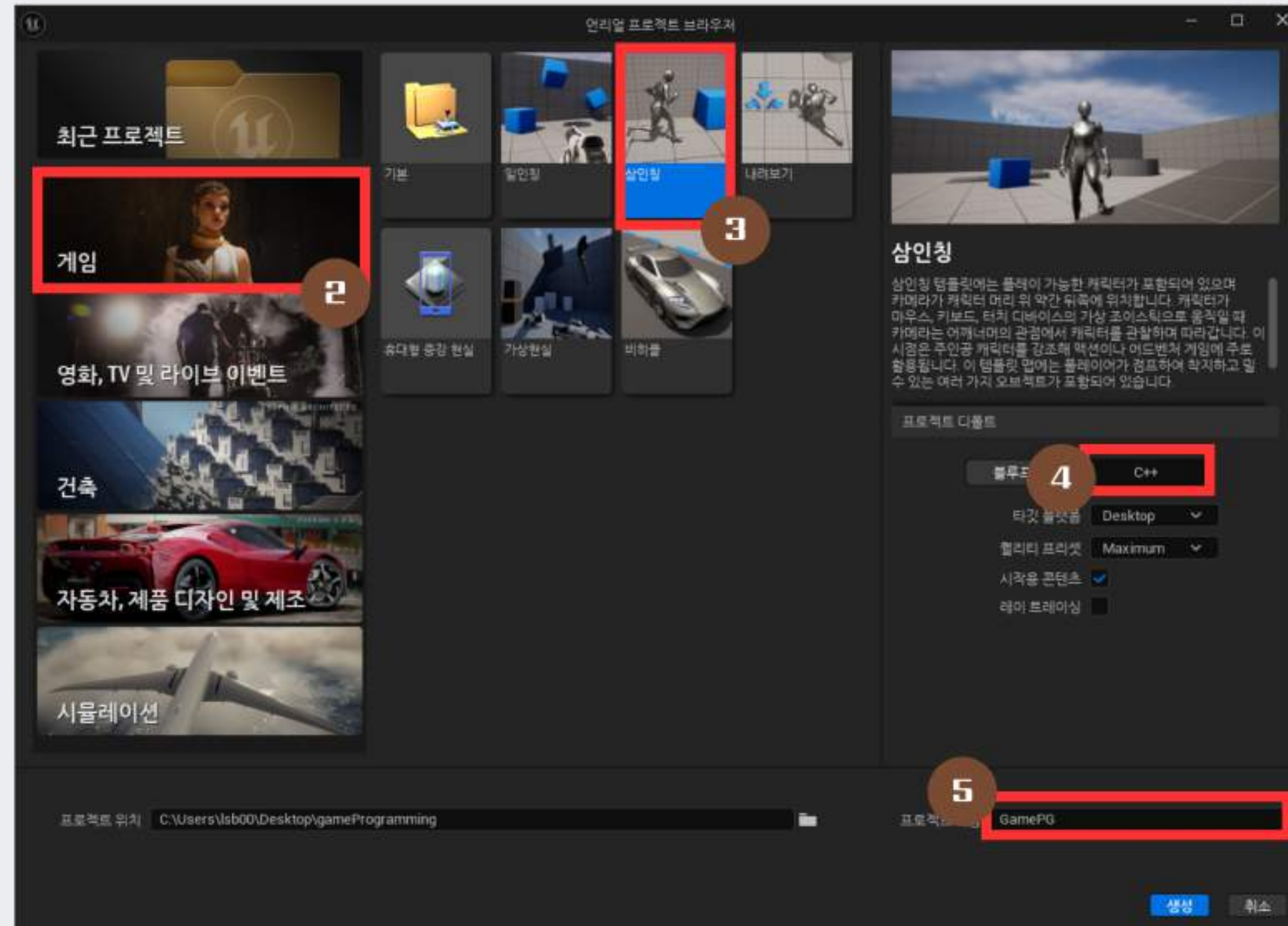


## 프로젝트 생성

>>> 프로젝트 생성을 해보자

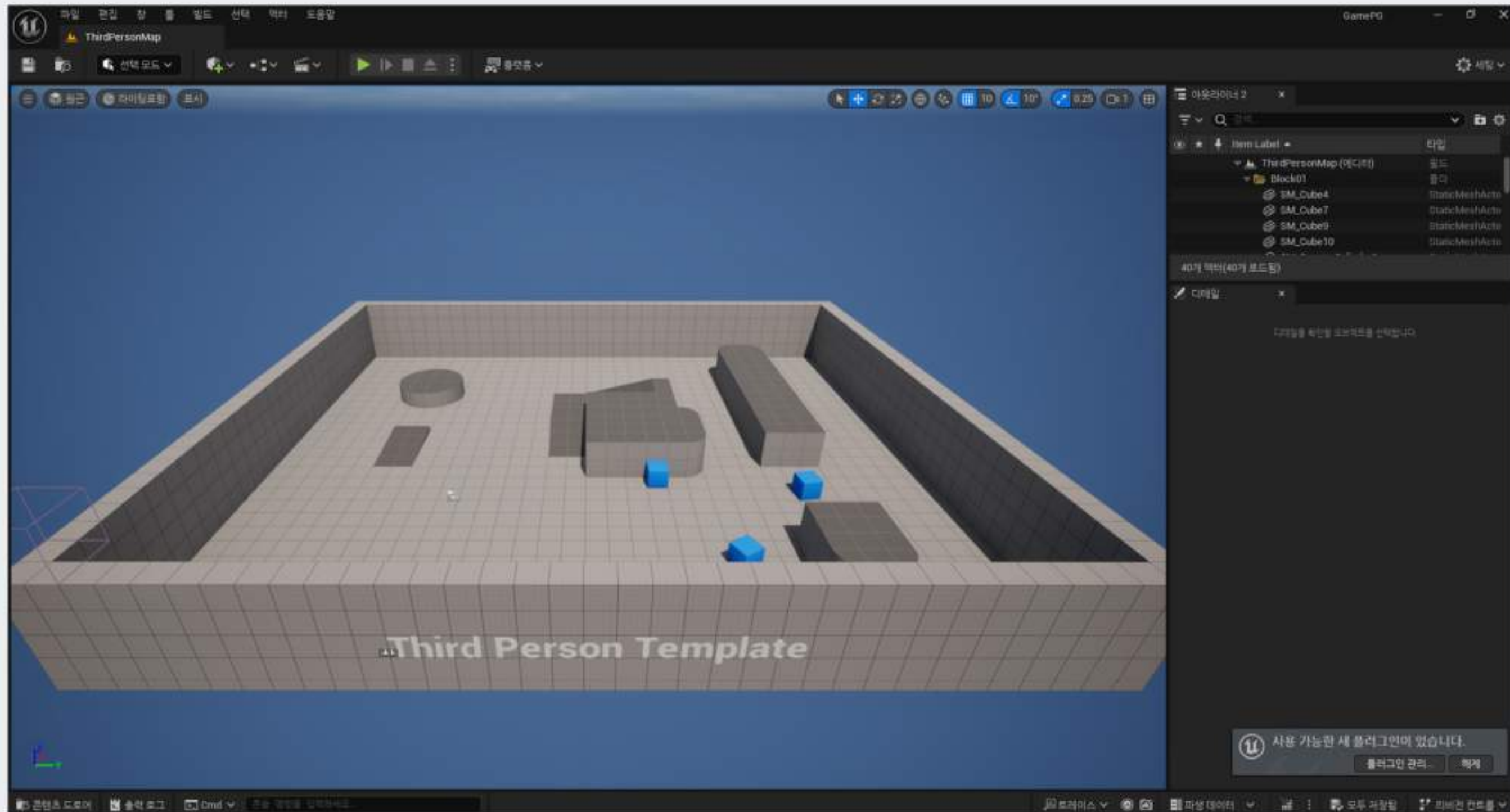


순서대로 클릭한  
후 5에서 프로젝  
트 이름을 입력후  
생성한다



# 프로젝트 생성

>>> 완료된 화면

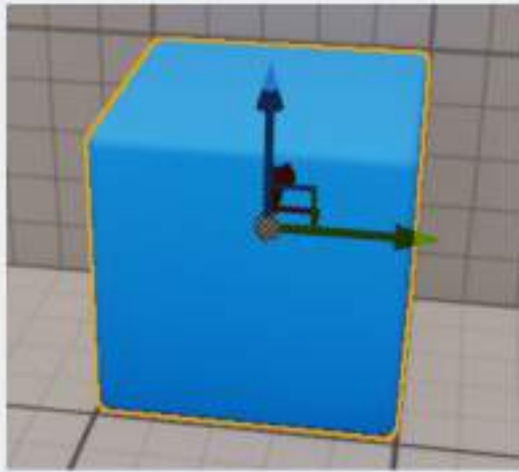


# 언리얼 엔진 내 기능 소개

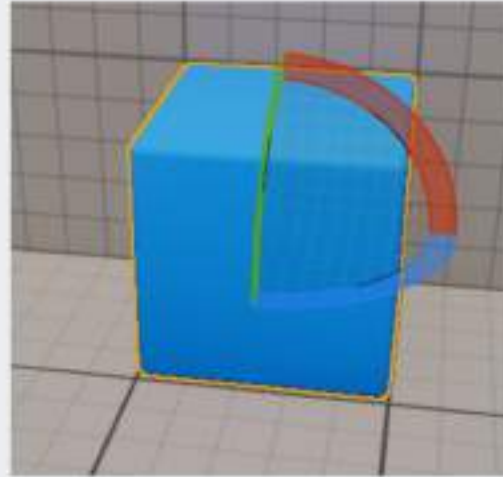
Chapter 3

## 언리얼 내 기능 소개

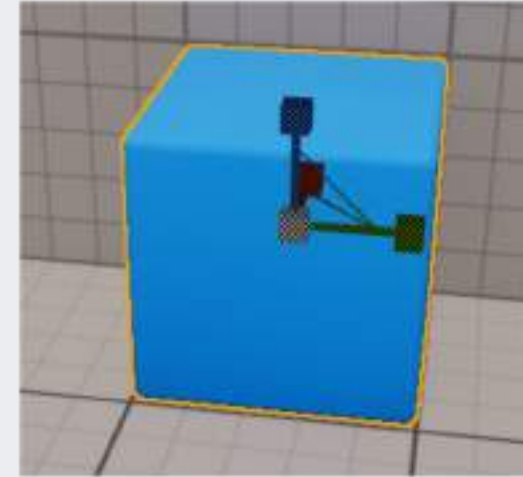
>>> 오브젝트 이동, 회전, 스케일



W - 이동



E - 회전



R - 스케일



## 언리얼 내 기능 소개

>>> 예제를 들어가기 전 알아야 할 것



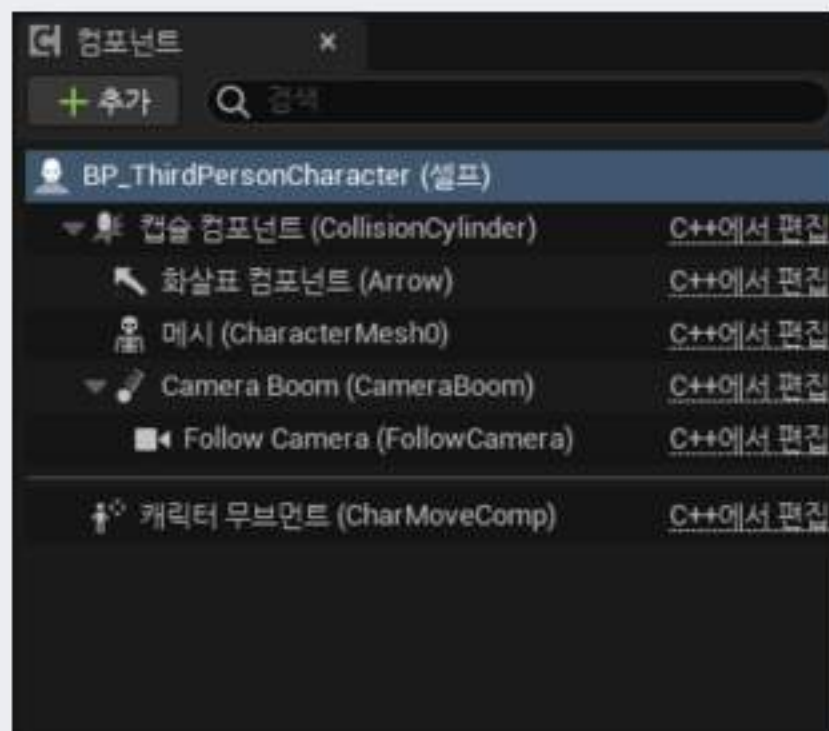
폰 (Pawn)

폰은 플레이어나 AI가 제어할 수 있는 모든 액터의 베이스 게임을 하게되면 내가 조작할 수 있는 오브젝트들이 있다. 대표적으로 내가 키보드를 통해 움직임을 명령하는 캐릭터같은 플레이어(혹은 AI)가 명령을 내리고 그것에 따라서 움직이는 액터들을 Pawn이라고 하는 것! 따라서 Actor보다 더 구체적으로 정의된 오브젝트 클래스라는 것이다.

컴포넌트는 액터(Actor)의 구성 요소로, 액터의 속성이나 동작을 정의하거나 확장하는 데 사용됩니다. 쉽게 말해, 컴포넌트는 액터의 기능적 블록이라고 할 수 있다.

컴포넌트의 특징

1. 재사용 가능성: 한 번 만든 컴포넌트를 여러 액터에 재사용할 수 있다.
2. 구조화: 액터의 특정 기능(예: 충돌, 렌더링, 움직임 등)을 독립적으로 관리할 수 있다.
3. 종속적 관계: 컴포넌트는 액터에 종속적이며, 액터가 제거되면 관련된 모든 컴포넌트도 제거된다.



컴포넌트 (Components)

## 언리얼 내 기능 소개

>>> 예제를 들어가기 전 알아야 할 것



매핑 (Mapping)

언리얼 엔진에서 매핑(Mapping)은 입력(Input)을 설정하고 관리하는 개념으로, 키보드, 마우스, 컨트롤러, 터치 등의 입력 장치를 통해 액터 또는 캐릭터가 특정 동작을 수행하도록 연결하는 작업이다.

그 중 두가지로 나뉘는데 축 매핑과 액션 매핑으로 나뉜다.

### 1. 축 매핑(Axis Mapping)

- 설명: 입력 값의 연속적인 변화를 처리하는 매핑. 주로 아날로그 입력 또는 누르고 있는 상태를 처리한다.

### 2. 액션 매핑(Action Mapping)

- 설명: 입력의 이벤트성 동작을 처리하는 매핑. 주로 키를 눌렀을 때 발생하는 동작에 사용된다.
- 특징:
  - 단일 이벤트로 동작 수행(누름/해제 구분).
  - "한 번 누름" 같은 순간적인 입력 처리.

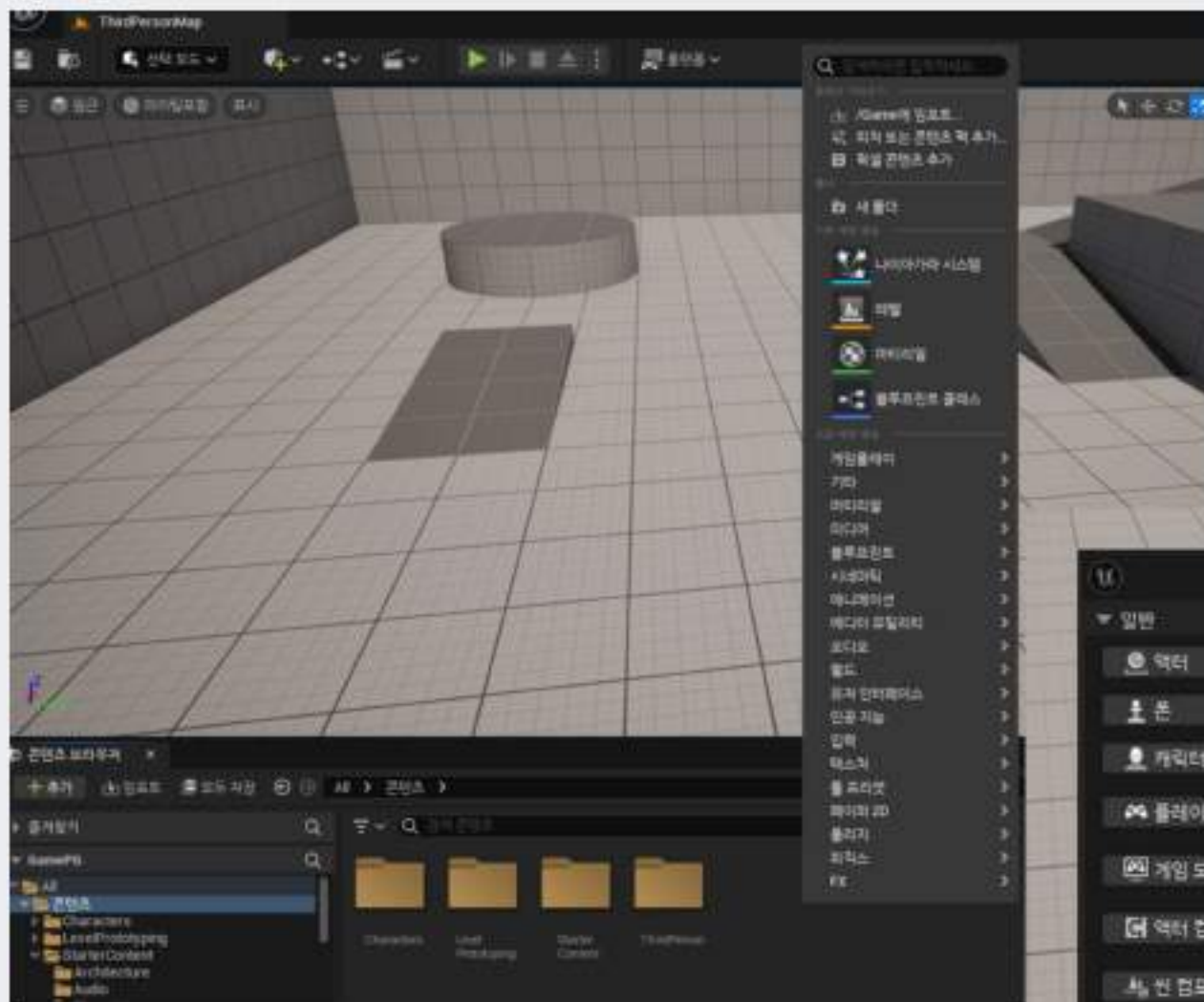


# 폰을 움직여보자!(BP)

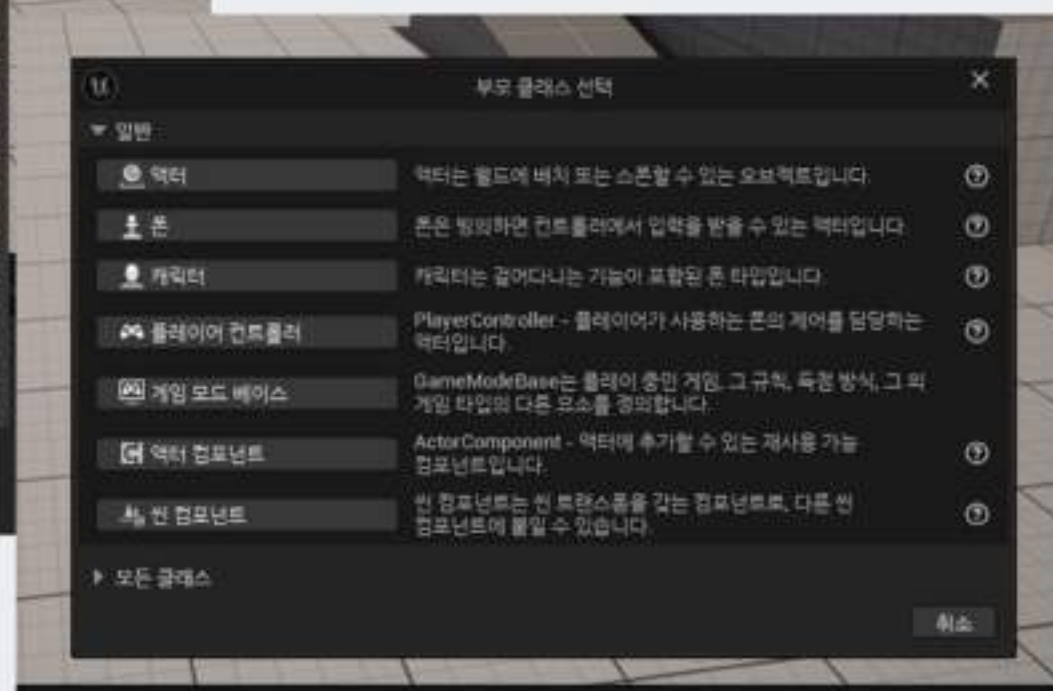
Chapter 4

# 폰을 움직여보자 (BLUEPRINT)

## >>> 폰 생성

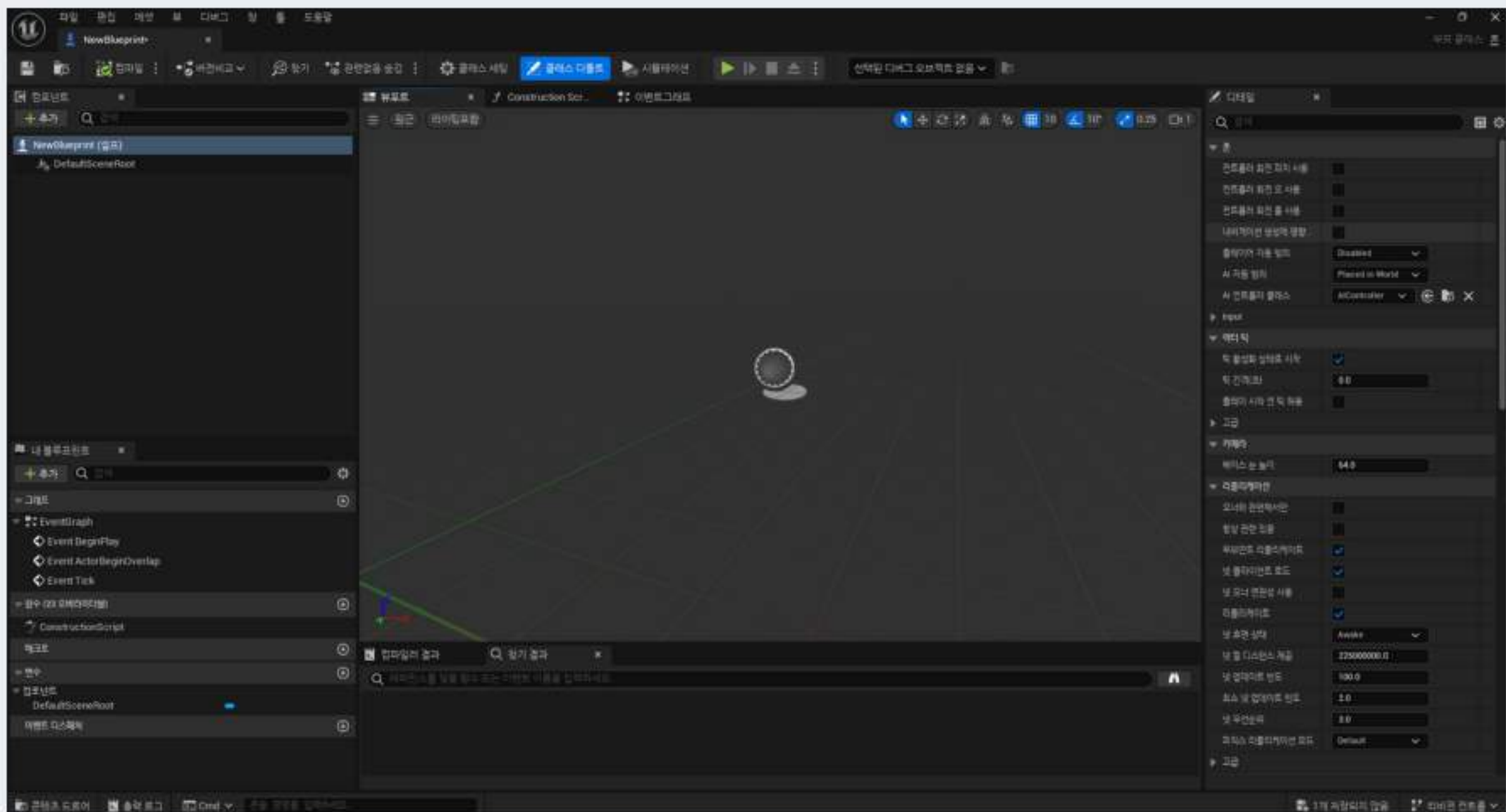


컨텐츠 브라우저에서 빈 공간에서 우클릭을 한  
뒤 블루프린트 클래스를 클릭 후 폰을 클릭



## 폰을 움직여보자 (BLUEPRINT)

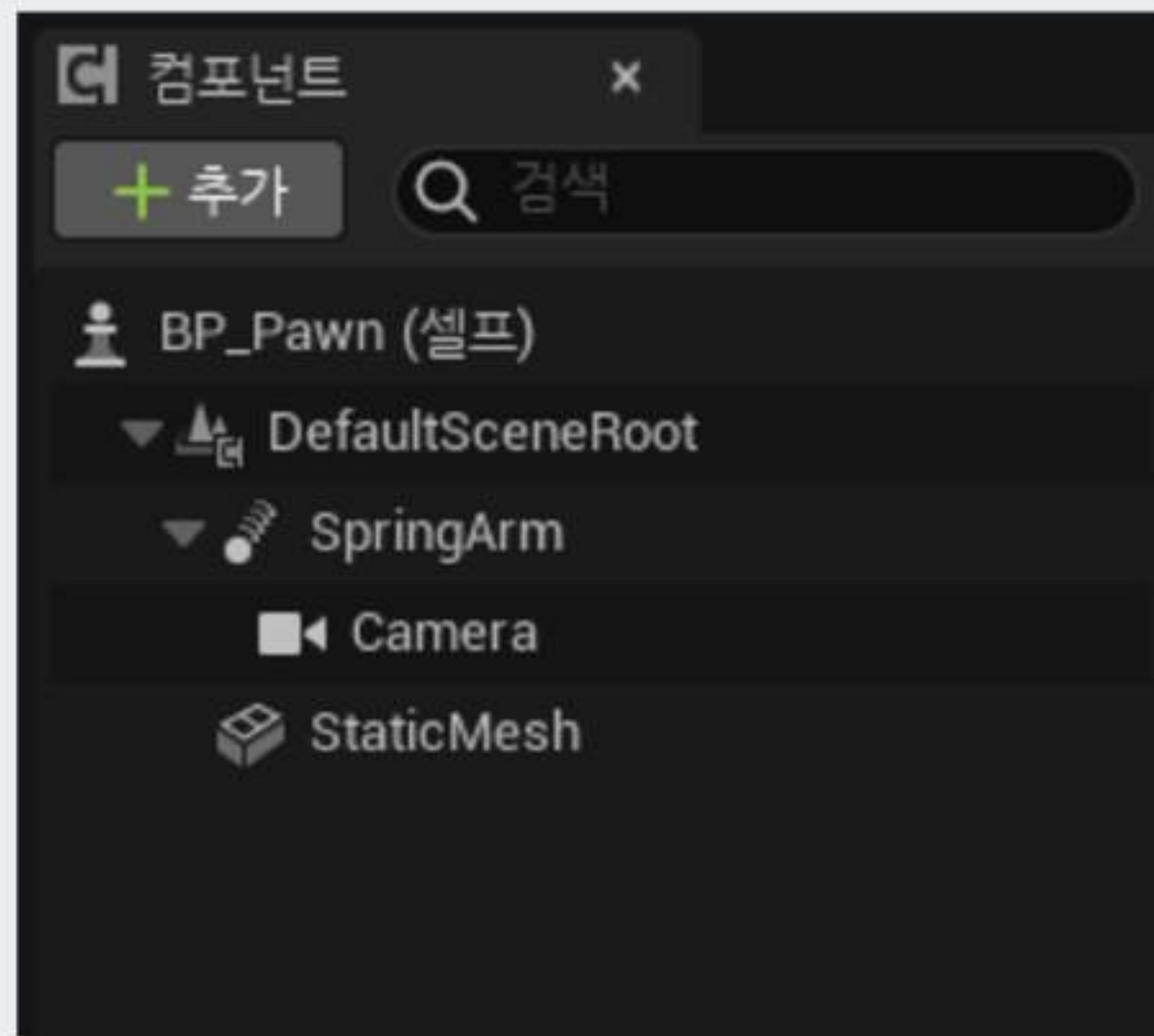
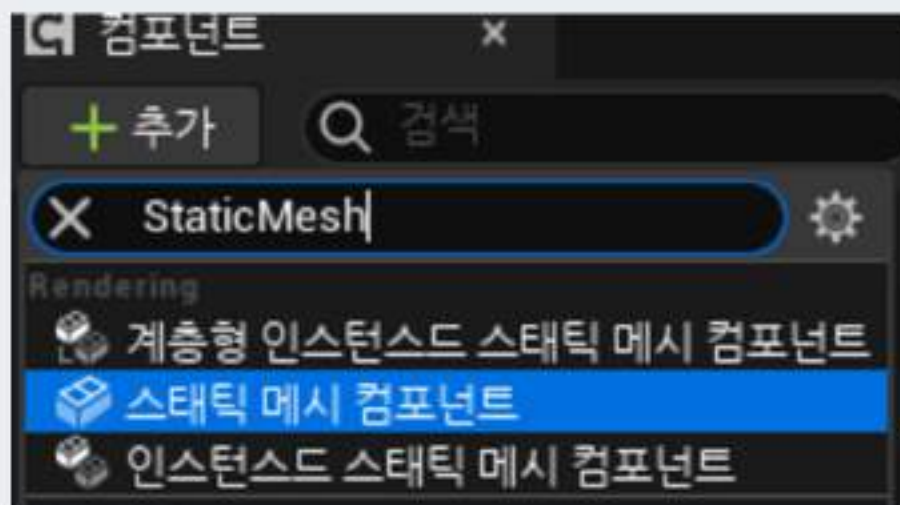
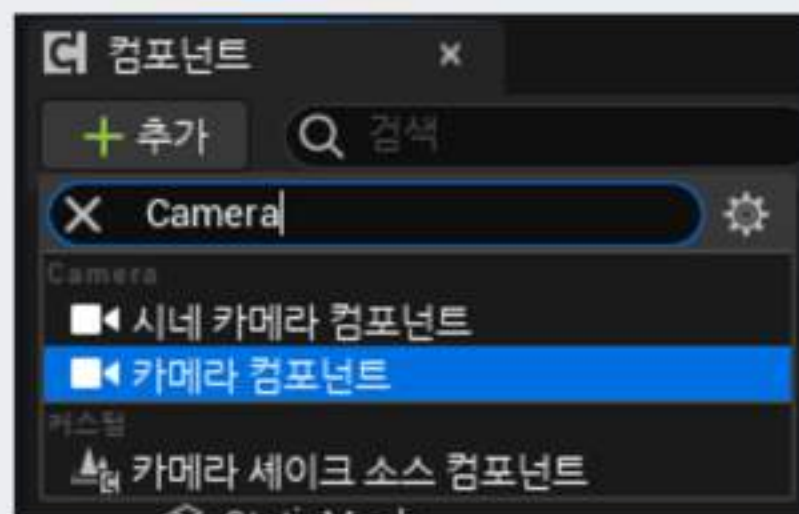
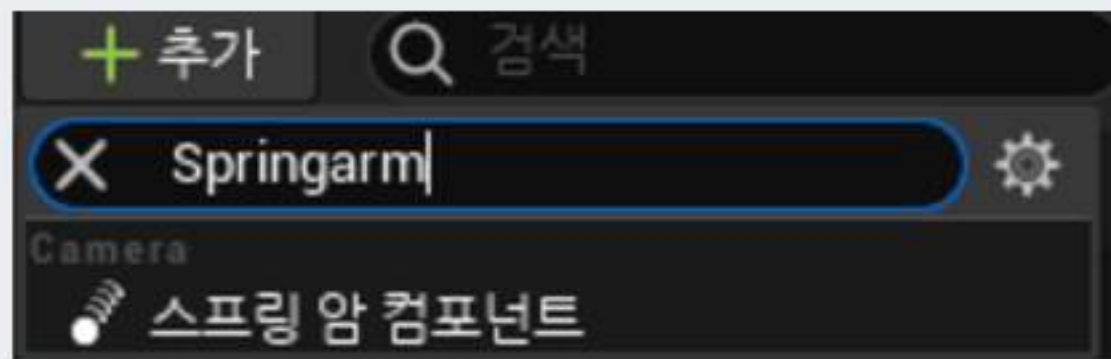
▶▶▶ 폰 블루프린트



블루프린트 화면이 열린 모습

## 폰을 움직여보자 (BLUEPRINT)

>>> 폰 블루프린트

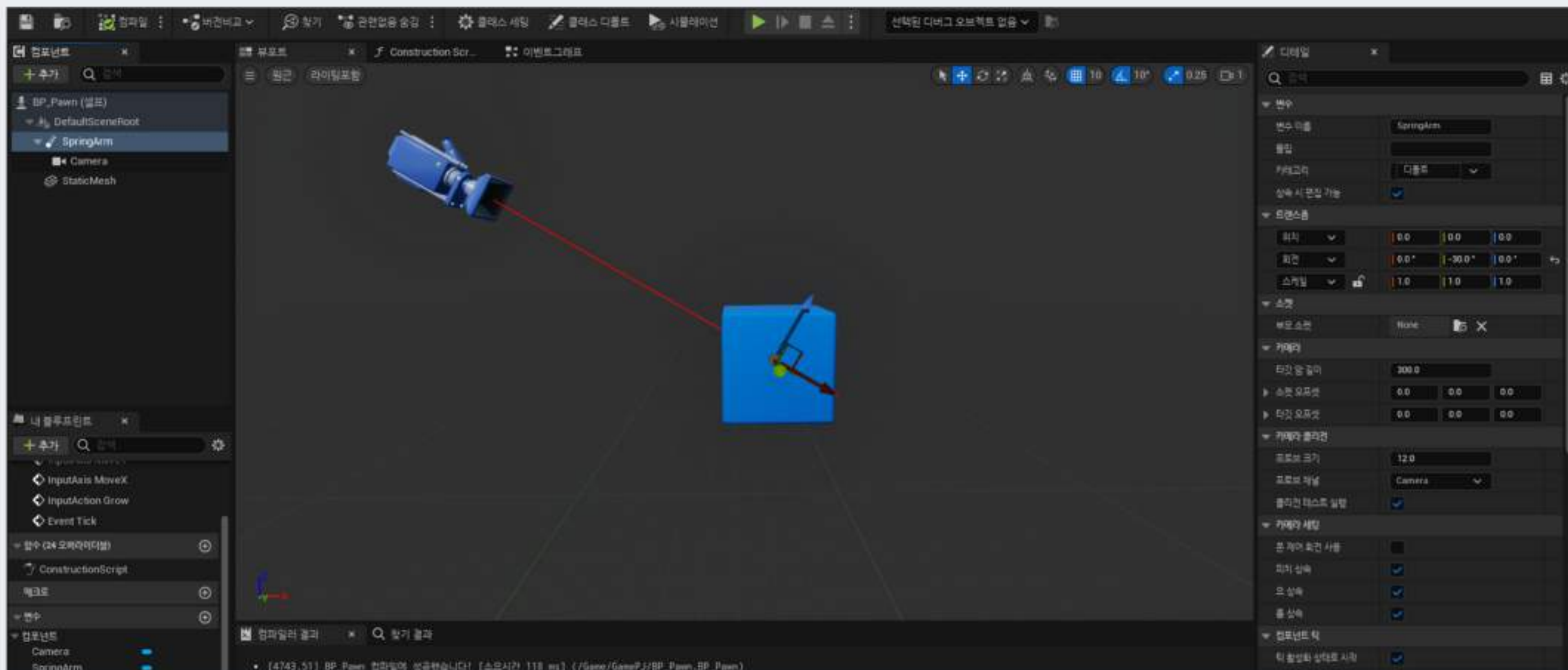


카메라 및 오브젝트 생성



# 폰을 움직여보자 (BLUEPRINT)

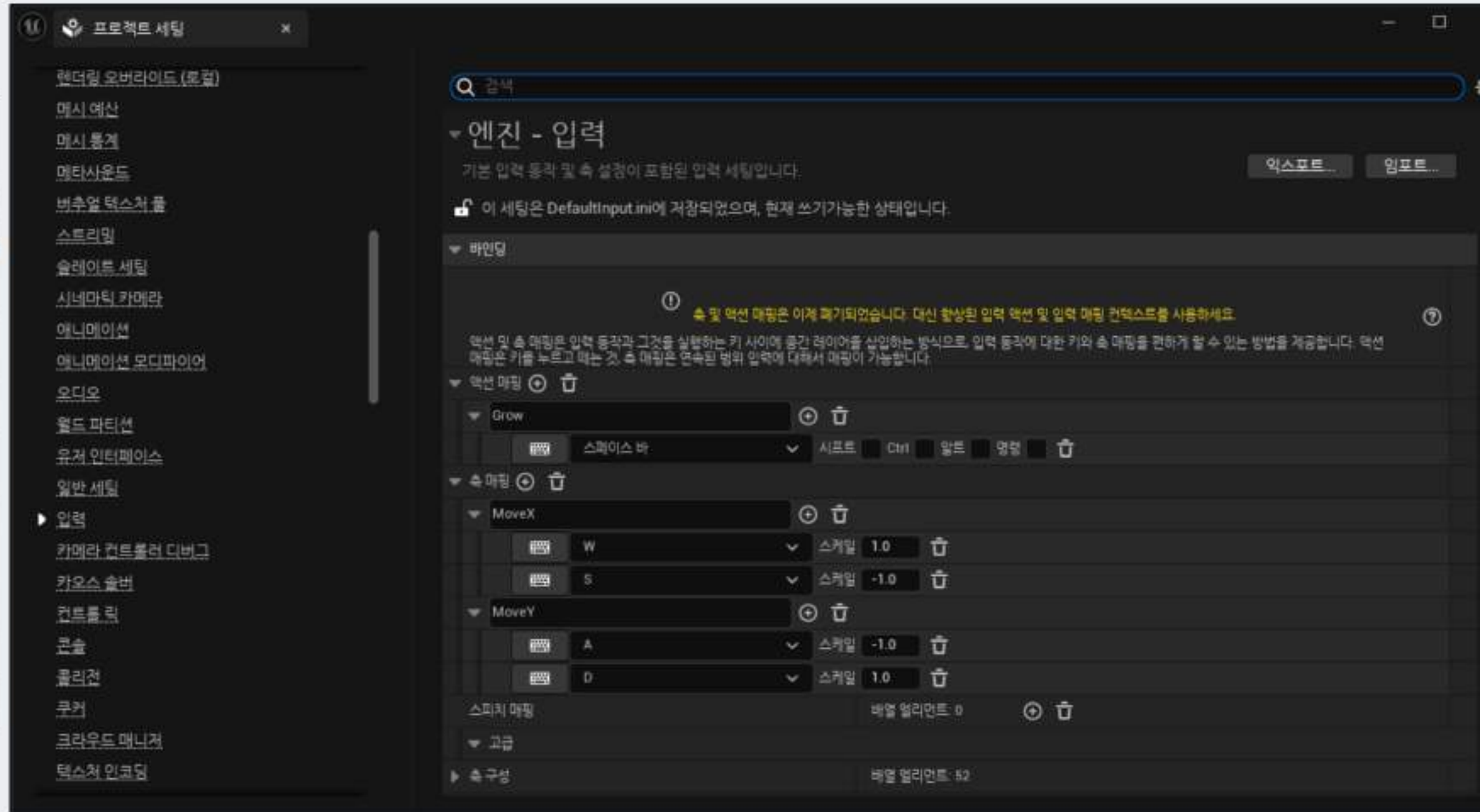
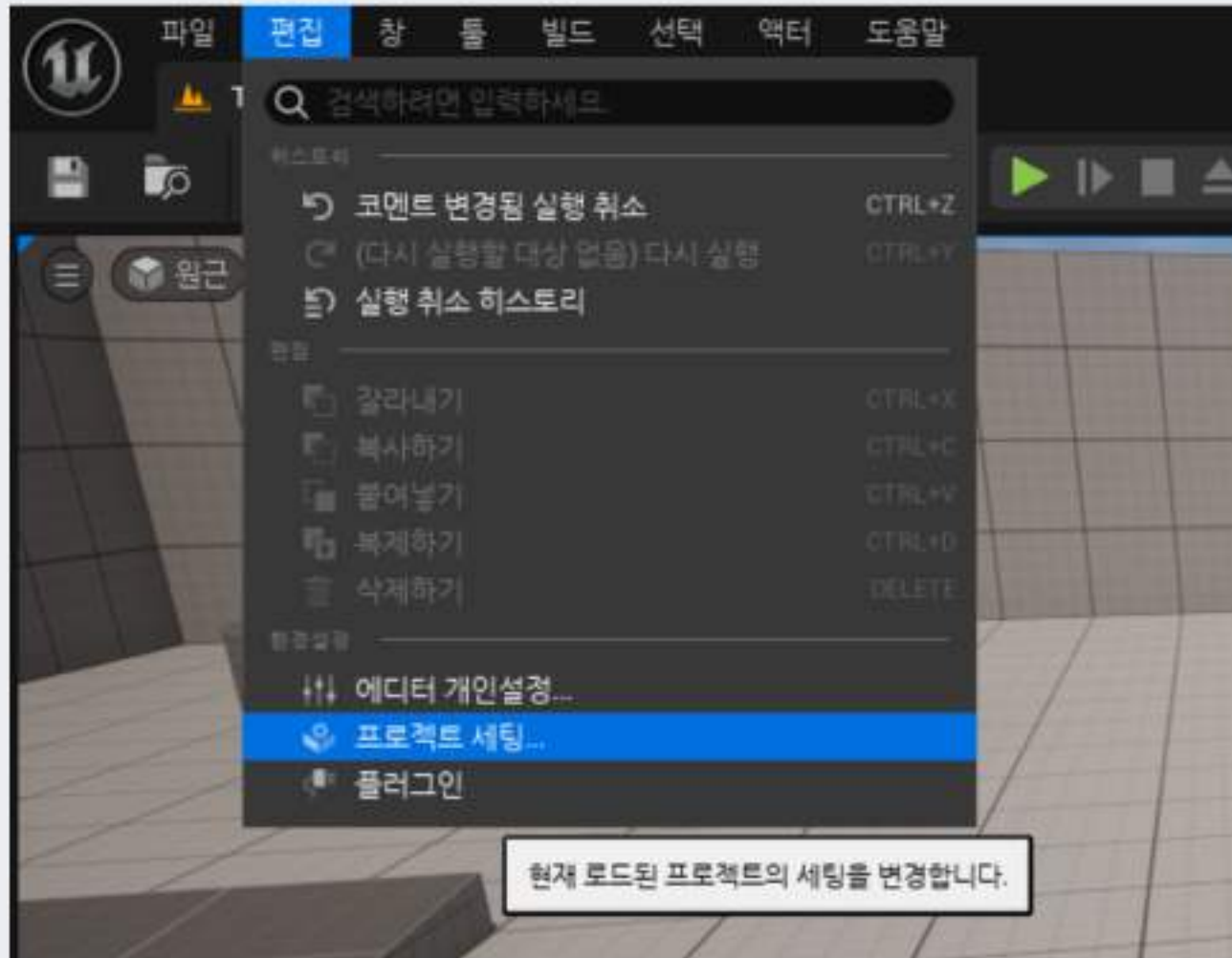
>>> 폰 블루프린트



스프링 암을 선택 후 회전 Y -30.0으로 설정 후  
상단 이벤트 그래프 클릭

# 폰을 움직여보자 (BLUEPRINT)

## >>> 폰 블루프린트

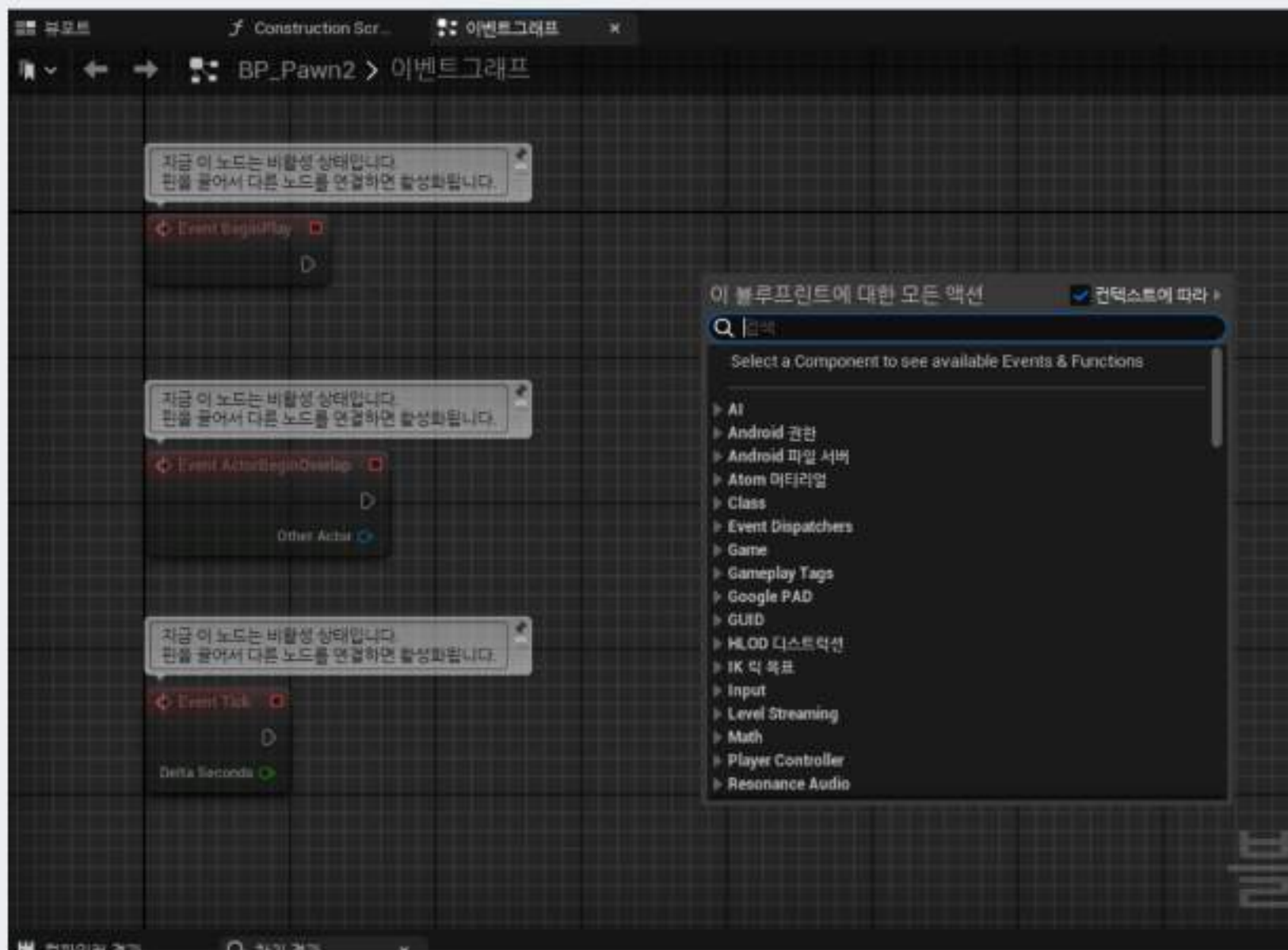


입력 선택후 액션 매핑 및 축 매핑을 다음과 같이 설정

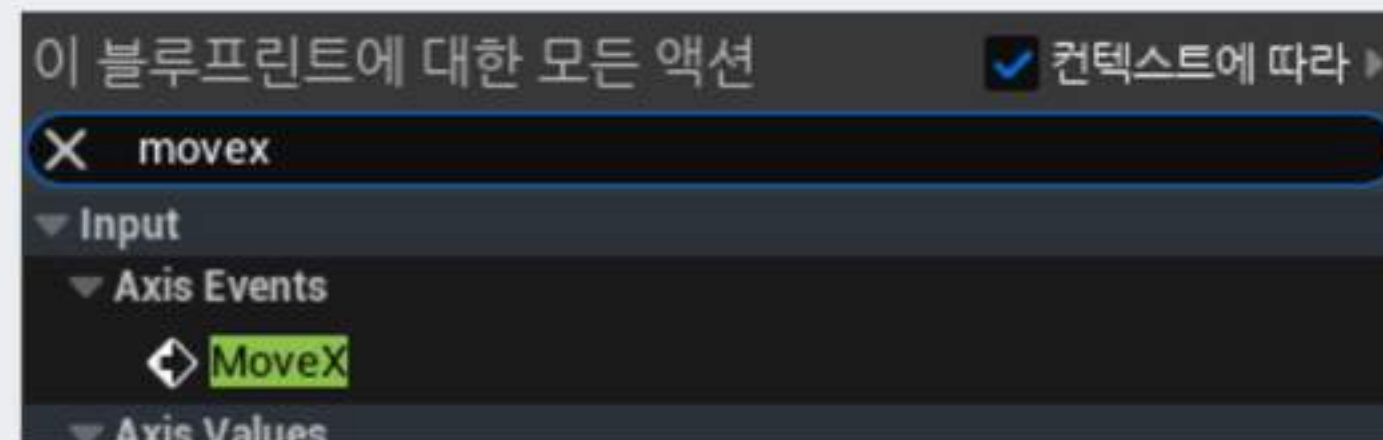


# 폰을 움직여보자 (BLUEPRINT)

>>> 폰 블루프린트



다시 블루프린트로 돌아와서 빈 공간에 우클릭 후 MoveX 검색



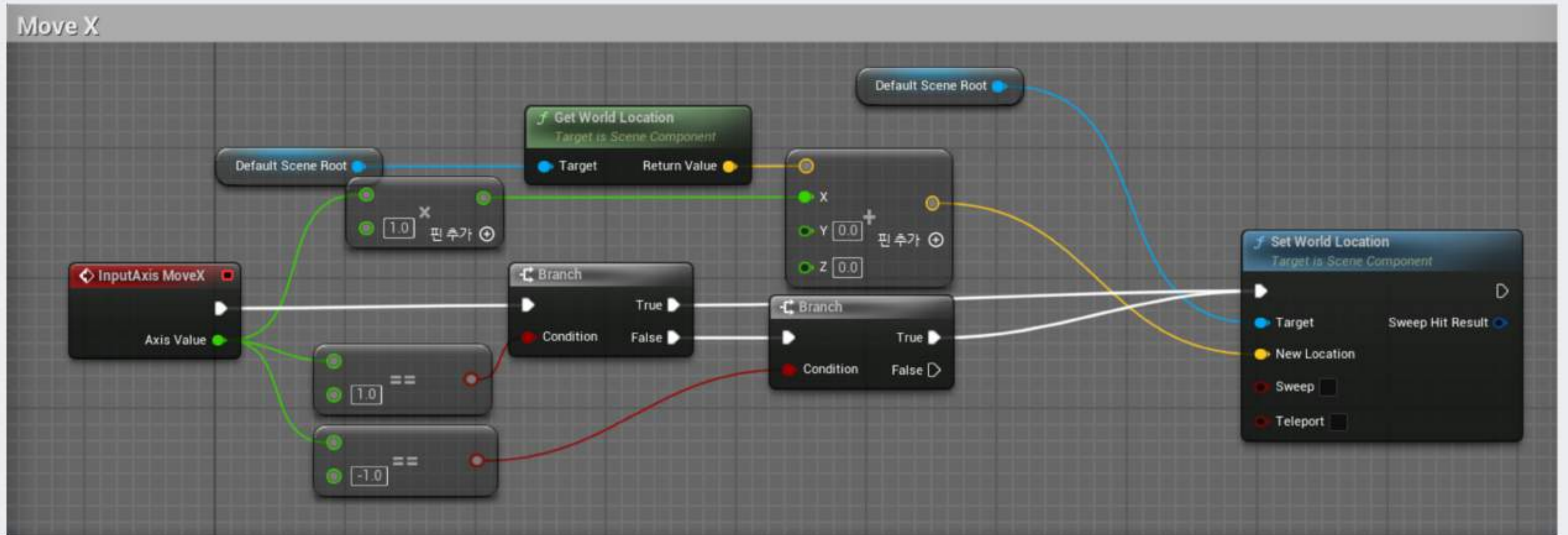
Axis Events에 있는 MoveX 클릭



이벤트가 생성된 모습

# 폰을 움직여보자 (BLUEPRINT)

>>> 폰 블루프린트

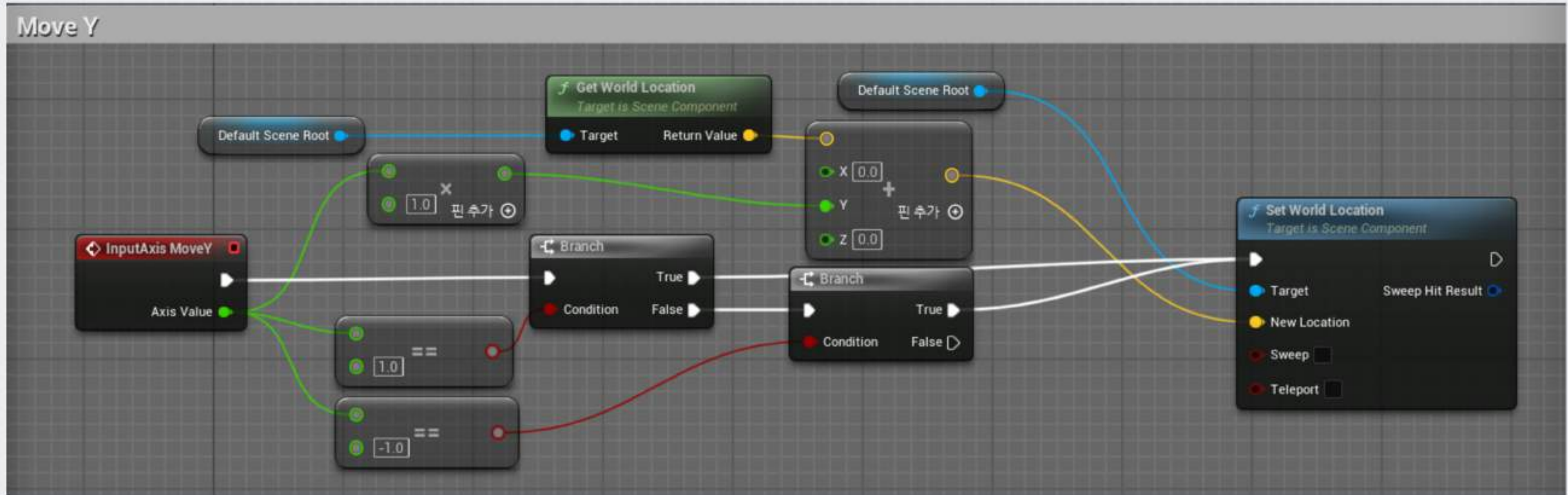


MoveX X축으로 움직이는 블루프린트



## 폰을 움직여보자 (BLUEPRINT)

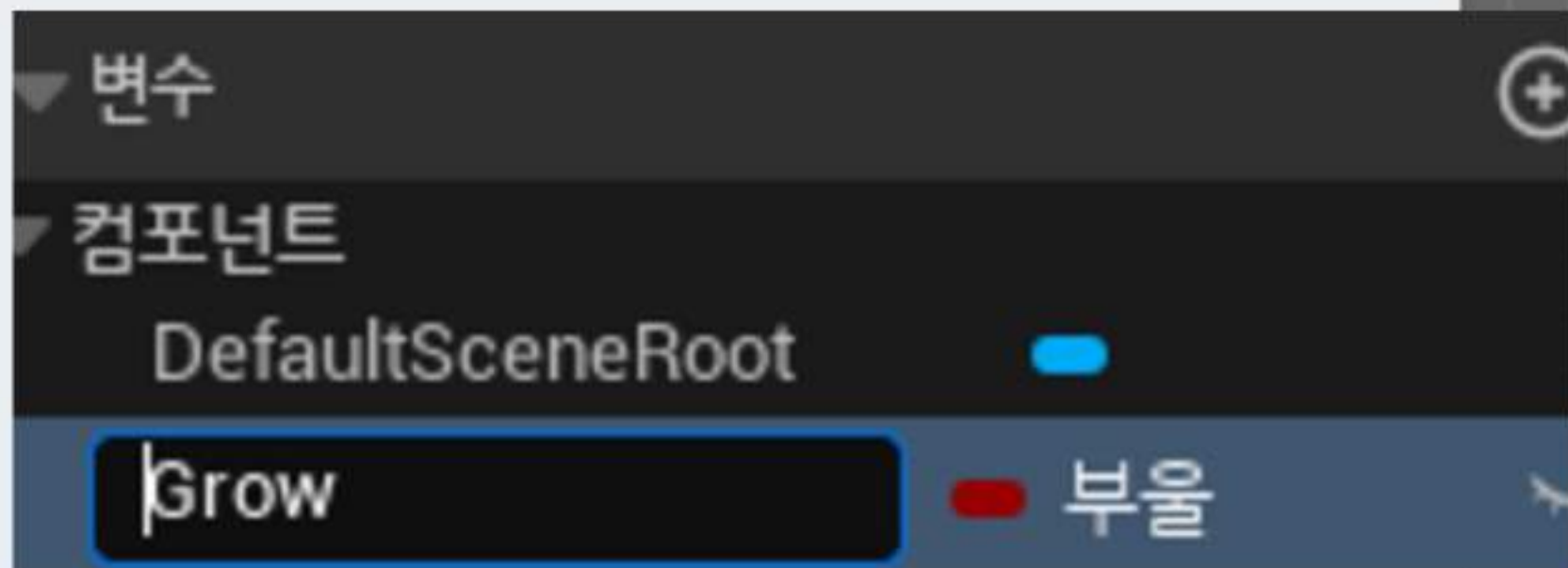
>>> 폰 블루프린트



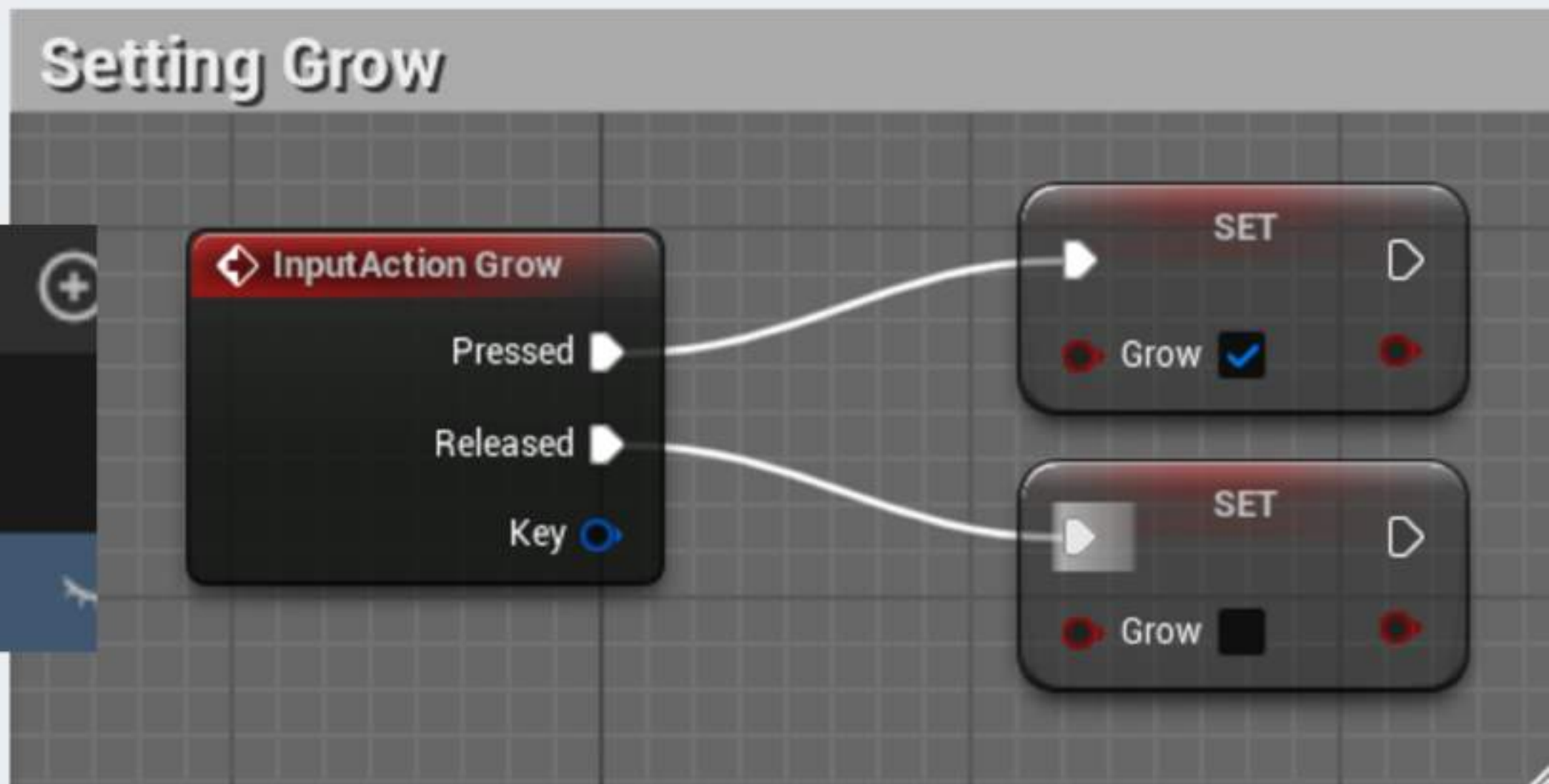
MoveX Y축으로 움직이는 블루프린트

## 폰을 움직여보자 (BLUEPRINT)

>>> 폰 블루프린트



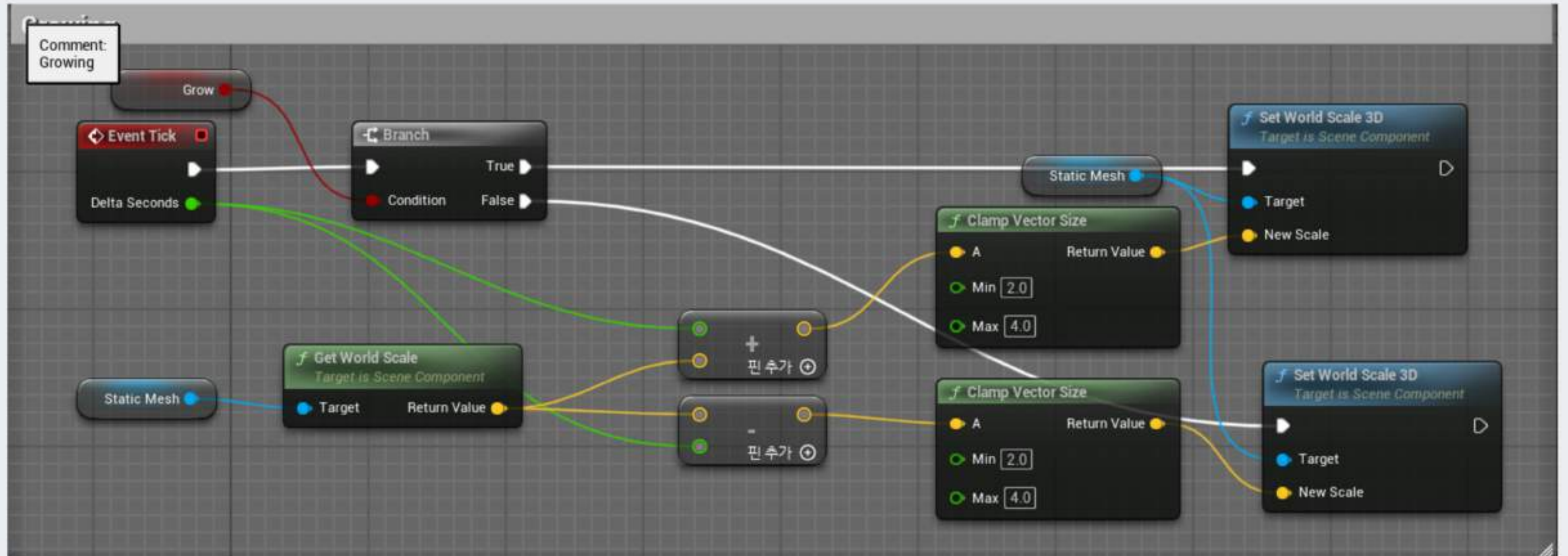
왼쪽 변수에서 +를 누른 후 Grow (bool) 생성



Grow 이벤트 생성

# 폰을 움직여보자 (BLUEPRINT)

>>> 폰 블루프린트

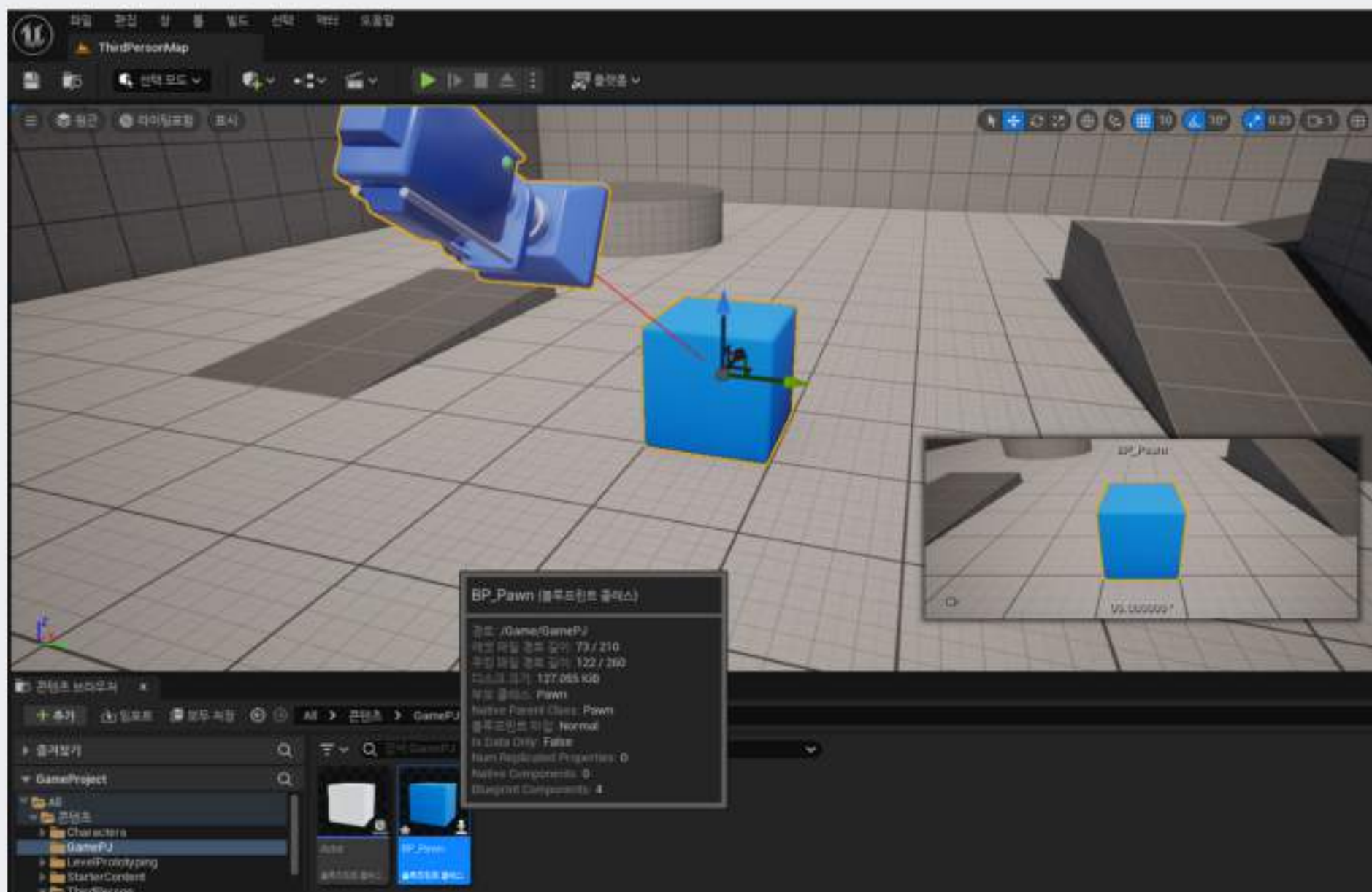


스페이스바를 누르면 어느정도 까지 커지고 다시 떼면 줄어드는 블루프린트

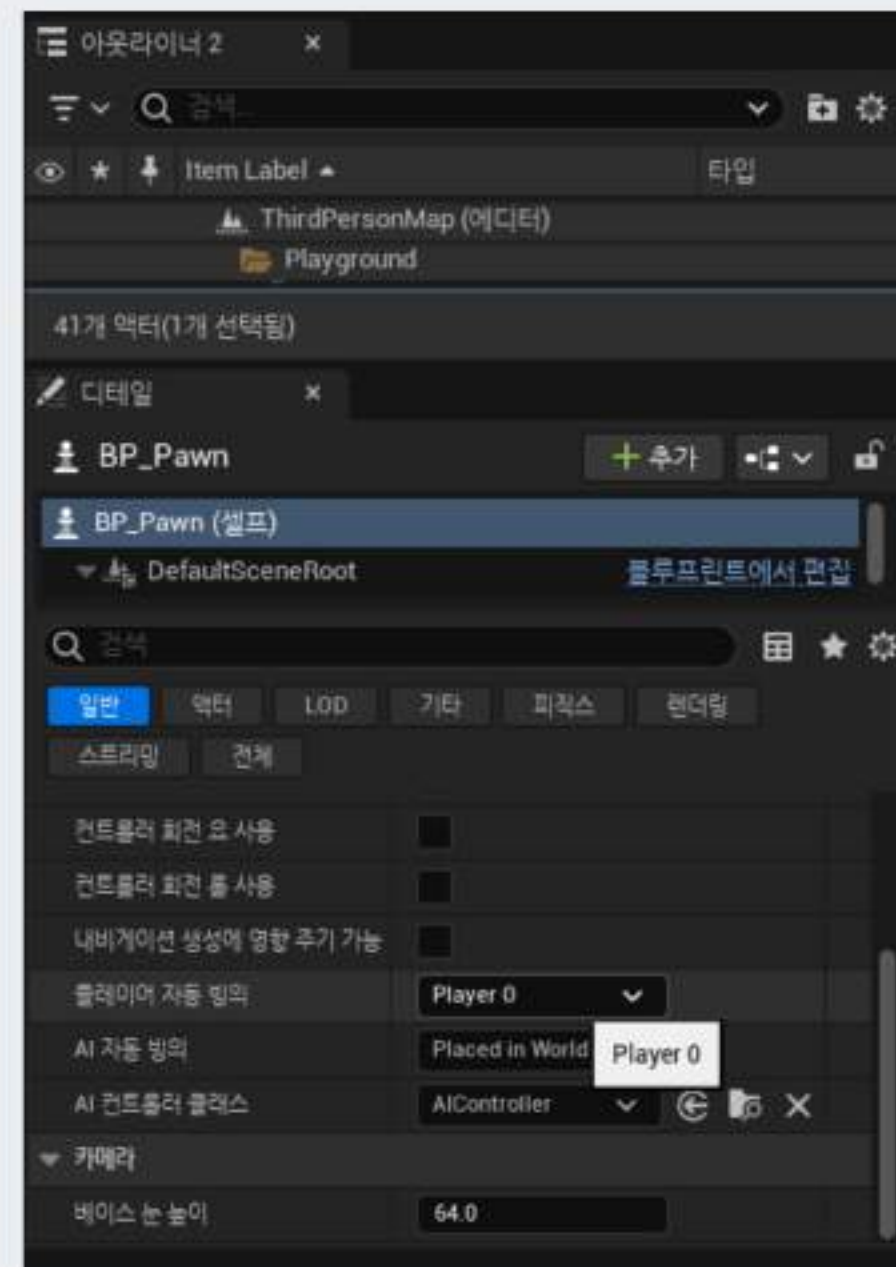


# 폰을 움직여보자 (BLUEPRINT)

## >>> 폰 블루프린트



완료된 블루프린트 클래스를 위 화면에 드래그 앤 드롭



우측 화면에서 플레이어 자동빙의를 Player 0으로 설정

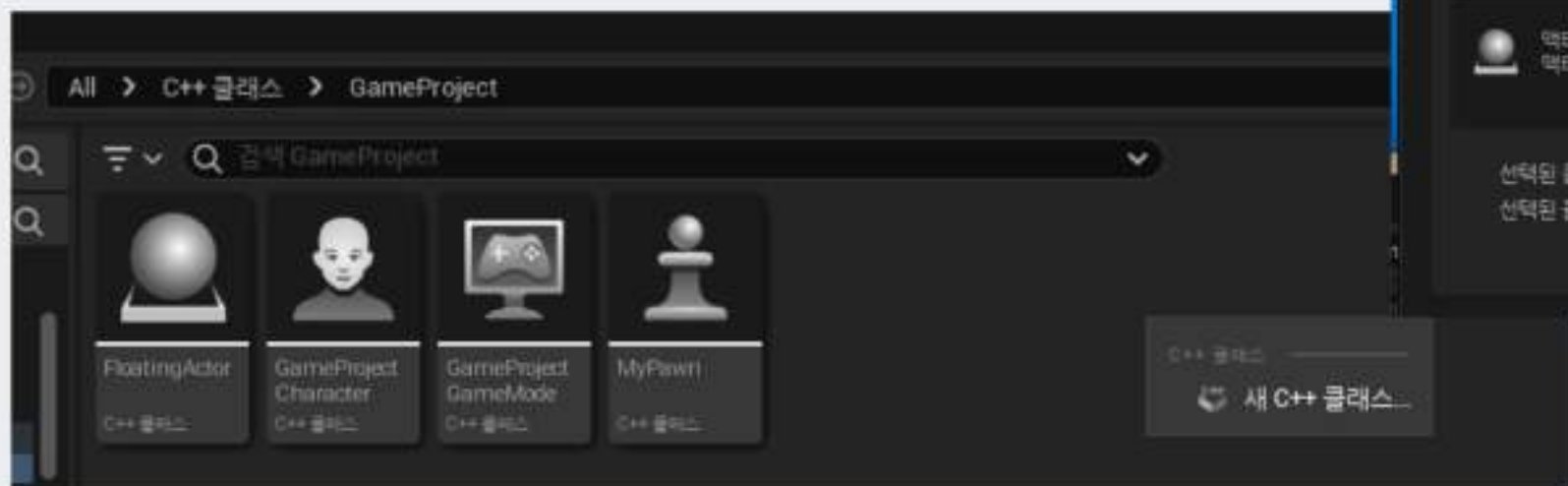


# 폰을 움직여보자!(C++)

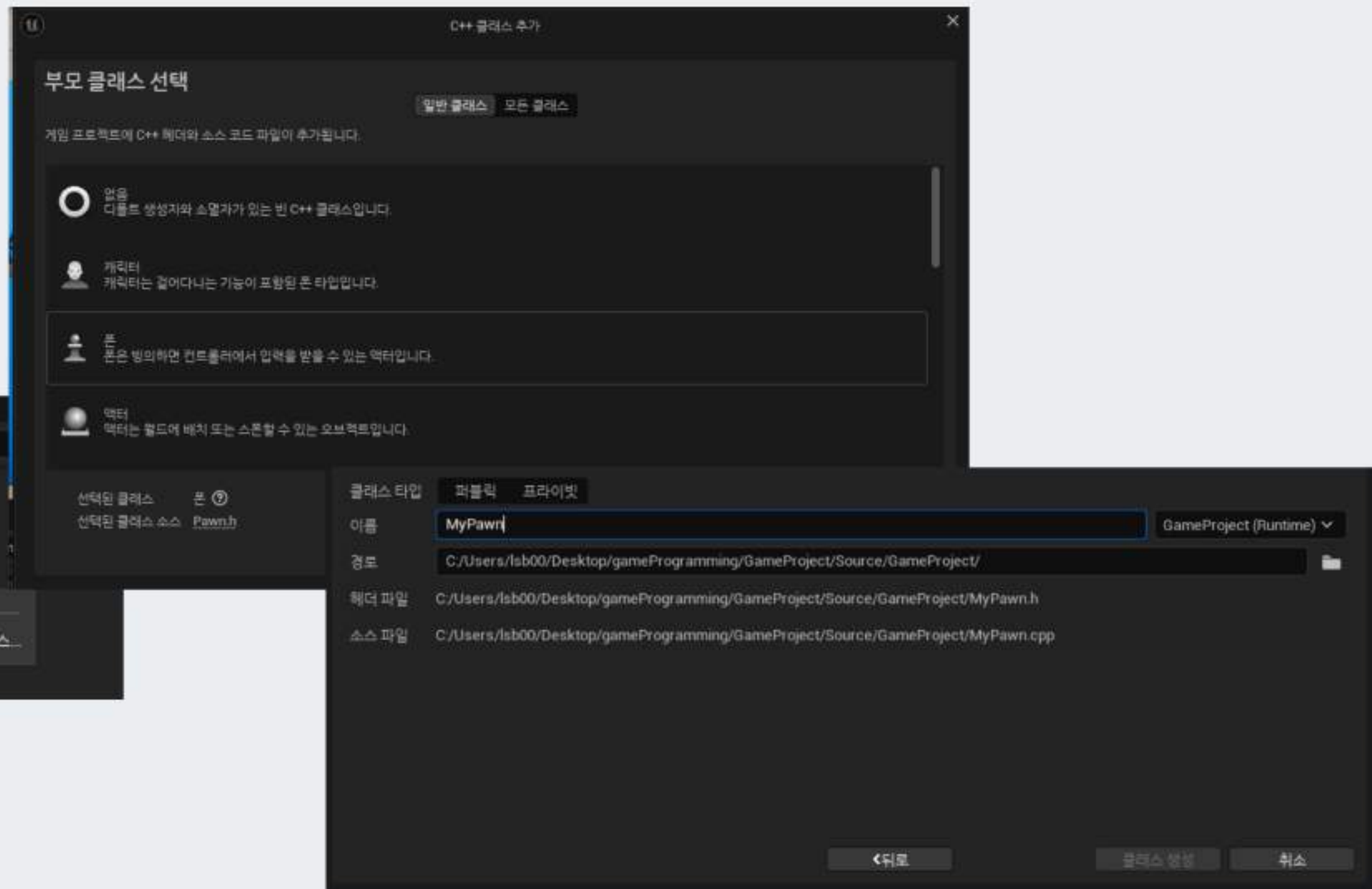
Chapter 5

# 폰을 움직여보자 (C++)

>>> 폰 생성



C++ 클래스 빈공간에서 새 c++클래스 클릭 후  
폰 클릭



폰누르고 MyPawn으로 폰 클래스 생성

# 폰을 움직여보자 (C++)

## >>> 폰 클래스

MyPawn.h 파일

```
4
5  #include "CoreMinimal.h"
6  #include "GameFramework/Pawn.h"
7  #include "MyPawn.generated.h"
8
9  UCLASS()
10 class GAMEPROJECT_API AMyPawn : public APawn
11 {
12     GENERATED_BODY()
13
14     UPROPERTY(EditAnywhere)
15     USceneComponent* OurVisibleComponent;
16
17     void Move_XAxis(float AxisValue);
18     void Move_YAxis(float AxisValue);
19     void StartGrowing();
20     void StopGrowing();
21
22     FVector CurrentVelocity;
23     bool bGrowing;
24
25
26 public:
27     // Sets default values for this pawn's properties
28     AMyPawn();
29
30 protected:
31     // Called when the game starts or when spawned
32     virtual void BeginPlay() override;
33
34 public:
35     // Called every frame
36     virtual void Tick(float DeltaTime) override;
37
38     // Called to bind functionality to input
39     virtual void SetupPlayerInputComponent(class UInputComponent* PlayerInputComponent) override;
40
41 };
```

```
UCLASS()
class GAMEPROJECT_API AMyPawn :
public APawn
{
    GENERATED_BODY()
```

```
    UPROPERTY(EditAnywhere)
    USceneComponent*
    OurVisibleComponent;
```

```
    void Move_XAxis(float AxisValue);
    void Move_YAxis(float AxisValue);
    void StartGrowing();
    void StopGrowing();
```

```
    FVector CurrentVelocity;
    bool bGrowing;
```



# 폰을 움직여보자 (C++)

## >>> 폰 클래스

### MyPawn.cpp 파일

```
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3
4 #include "MyPawn.h"
5 #include "GameProject.h"
6 #include "Camera/CameraComponent.h"
7
8 // Sets default values
9
10 AMyPawn::AMyPawn()
11 {
12     // Set this pawn to call Tick() every frame. You can turn this off to improve performance if you d
13     PrimaryActorTick.bCanEverTick = true;
14     AutoPossessPlayer = EAutoReceiveInput::Player0;
15
16     RootComponent = CreateDefaultSubobject<USceneComponent>(TEXT("RootComponent"));
17     UCameraComponent* OurCamera = CreateDefaultSubobject<UCameraComponent>(TEXT("OurCamera"));
18     OurVisibleComponent = CreateDefaultSubobject<UStaticMeshComponent>(TEXT("OurVisibleComponent"));
19
20     OurCamera->SetupAttachment(RootComponent);
21     OurCamera->SetRelativeLocation(FVector(-250.0f, 0.0f, 250.0f));
22     OurCamera->SetRelativeRotation(FRotator(-45.0f, 0.0f, 0.0f));
23     OurVisibleComponent->SetupAttachment(RootComponent);
24 }
25
26 // Called when the game starts or when spawned
27 void AMyPawn::BeginPlay()
28 {
29     Super::BeginPlay();
30 }
31
```

#include "Camera/CameraComponent.h" 꼭 넣기

AMyPawn::AMyPawn()

{

PrimaryActorTick.bCanEverTick = true;

AutoPossessPlayer = EAutoReceiveInput::Player0; // 플레이어 설정

RootComponent = CreateDefaultSubobject<USceneComponent>  
(TEXT("RootComponent"));

UCameraComponent\* OurCamera =

CreateDefaultSubobject<UCameraComponent>(TEXT("OurCamera"));

OurVisibleComponent = CreateDefaultSubobject<UStaticMeshComponent>  
(TEXT("OurVisibleComponent")); // 카메라 및 카메라 암을 컴포넌트에 넣는 코드

OurCamera->SetupAttachment(RootComponent);

OurCamera->SetRelativeLocation(FVector(-250.0f, 0.0f, 250.0f));

OurCamera->SetRelativeRotation(FRotator(-45.0f, 0.0f, 0.0f));

OurVisibleComponent->SetupAttachment(RootComponent);

// // 카메라 위치 및 회전 세팅

}

## 폰을 움직여보자 (C++)

>>> 폰 클래스

MyPawn.cpp 파일

```
void AMyPawn::Move_XAxis(float AxisValue)
{
    CurrentVelocity.X = FMath::Clamp(AxisValue, -1.0f, 1.0f) * 100.0f;
}

void AMyPawn::Move_YAxis(float AxisValue)
{
    CurrentVelocity.Y = FMath::Clamp(AxisValue, -1.0f, 1.0f) * 100.0f;
}

void AMyPawn::StartGrowing()
{
    bGrowing = true;
}

void AMyPawn::StopGrowing()
{
    bGrowing = false;
}
```

CurrentVelocity.X =  
FMath::Clamp(AxisValue, -1.0f, 1.0f) \* 100.0f;  
// x 좌표로 움직이기

CurrentVelocity.Y =  
FMath::Clamp(AxisValue, -1.0f, 1.0f) \* 100.0f;  
// y 좌표로 움직이기

bGrowing = true;  
// 크기가 커질 수 있다.

bGrowing = false;  
// 크기가 작아진다.



## 폰을 움직여보자 (C++)

>>> 폰 클래스

MyPawn.cpp 파일

```
void AMyPawn::SetupPlayerInputComponent(UInputComponent* PlayerInputComponent)
{
    Super::SetupPlayerInputComponent(PlayerInputComponent);
    InputComponent->BindAction("Grow", IE_Pressed, this, &AMyPawn::StartGrowing);
    InputComponent->BindAction("Grow", IE_Released, this, &AMyPawn::StopGrowing);

    InputComponent->BindAxis("MoveX", this, &AMyPawn::Move_XAxis);
    InputComponent->BindAxis("MoveY", this, &AMyPawn::Move_YAxis);
}
```

```
Super::SetupPlayerInputComponent(PlayerInputComponent);
InputComponent->BindAction("Grow", IE_Pressed, this, &AMyPawn::StartGrowing);
InputComponent->BindAction("Grow", IE_Released, this, &AMyPawn::StopGrowing);
```

```
InputComponent->BindAxis("MoveX", this, &AMyPawn::Move_XAxis);
InputComponent->BindAxis("MoveY", this, &AMyPawn::Move_YAxis); // "MoveX" 와 "MoveY" 두 이동 축의 값에 매 프레임 반응
```



# 폰을 움직여보자 (C++)

## >>> 폰 클래스

```
// Called every frame
void APawn::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
    float CurrentScale = OurVisibleComponent->GetComponentScale().X;

    if (bGrowing)
    {
        CurrentScale += DeltaTime;
    }
    else
    {
        CurrentScale -= (DeltaTime * 0.5f);
    }

    CurrentScale = FMath::Clamp(CurrentScale, 1.0f, 2.0f);
    OurVisibleComponent->SetWorldScale3D(FVector(CurrentScale));

    if (!CurrentVelocity.IsZero())
    {
        FVector NewLocation = GetActorLocation() + (CurrentVelocity * DeltaTime);
        SetActorLocation(NewLocation);
    }
}
```

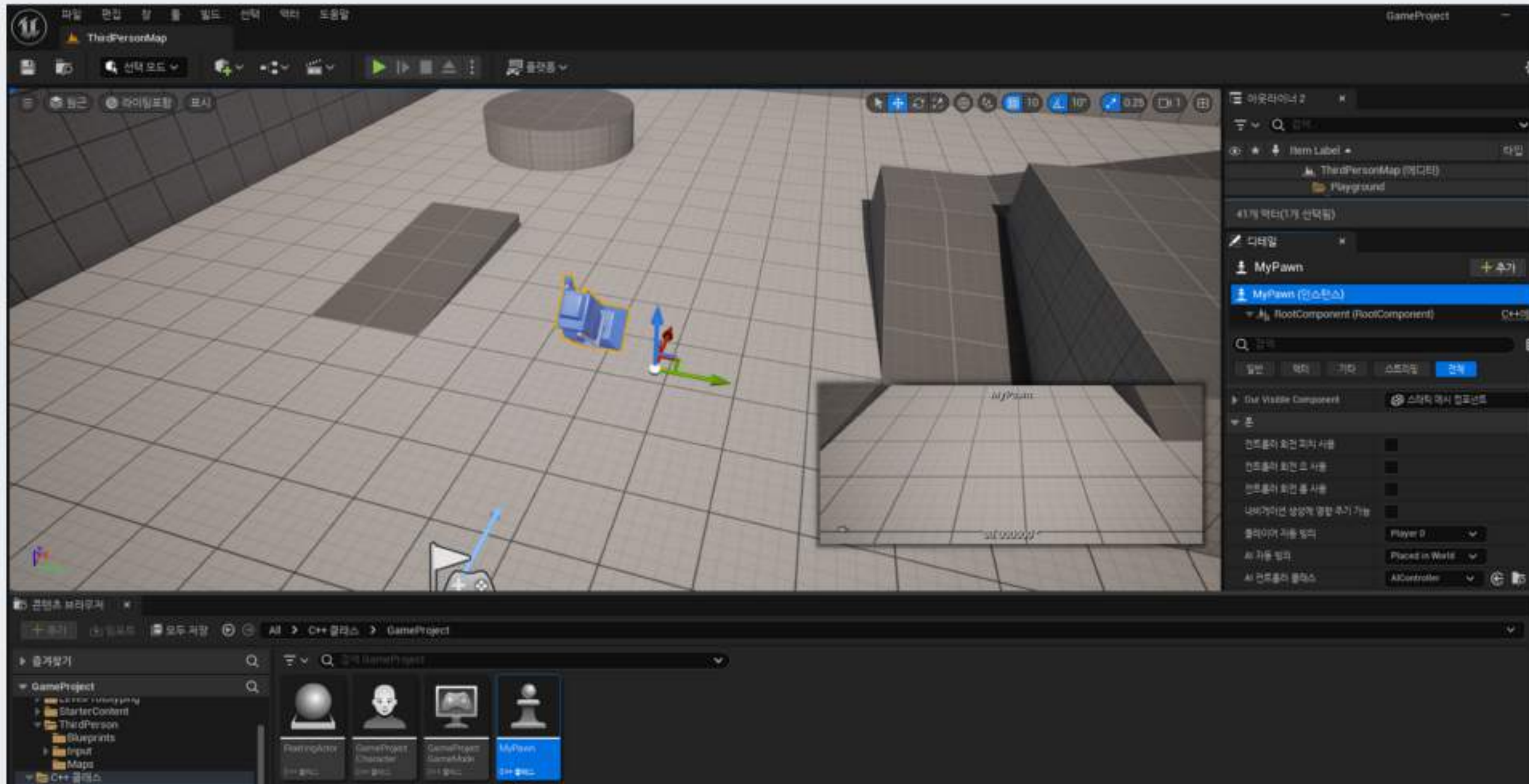
```
Super::Tick(DeltaTime);
float CurrentScale = OurVisibleComponent->GetComponentScale().X;
```

```
if (bGrowing)
{
    // 1 초에 걸쳐 두 배 크기로 키우기
    CurrentScale += DeltaTime;
}
else
{
    // 키운 속도대로 절반으로 줄이기
    CurrentScale -= (DeltaTime * 0.5f);
}
// 시작 크기 아래로 줄이거나 두 배 이상으로 키우지 않도록 함
CurrentScale = FMath::Clamp(CurrentScale, 1.0f, 2.0f);
OurVisibleComponent->SetWorldScale3D(FVector(CurrentScale));
```

```
if (!CurrentVelocity.IsZero())
{
    FVector NewLocation = GetActorLocation() + (CurrentVelocity * DeltaTime);
    SetActorLocation(NewLocation);
}
```

## 폰을 움직여보자 (C++)

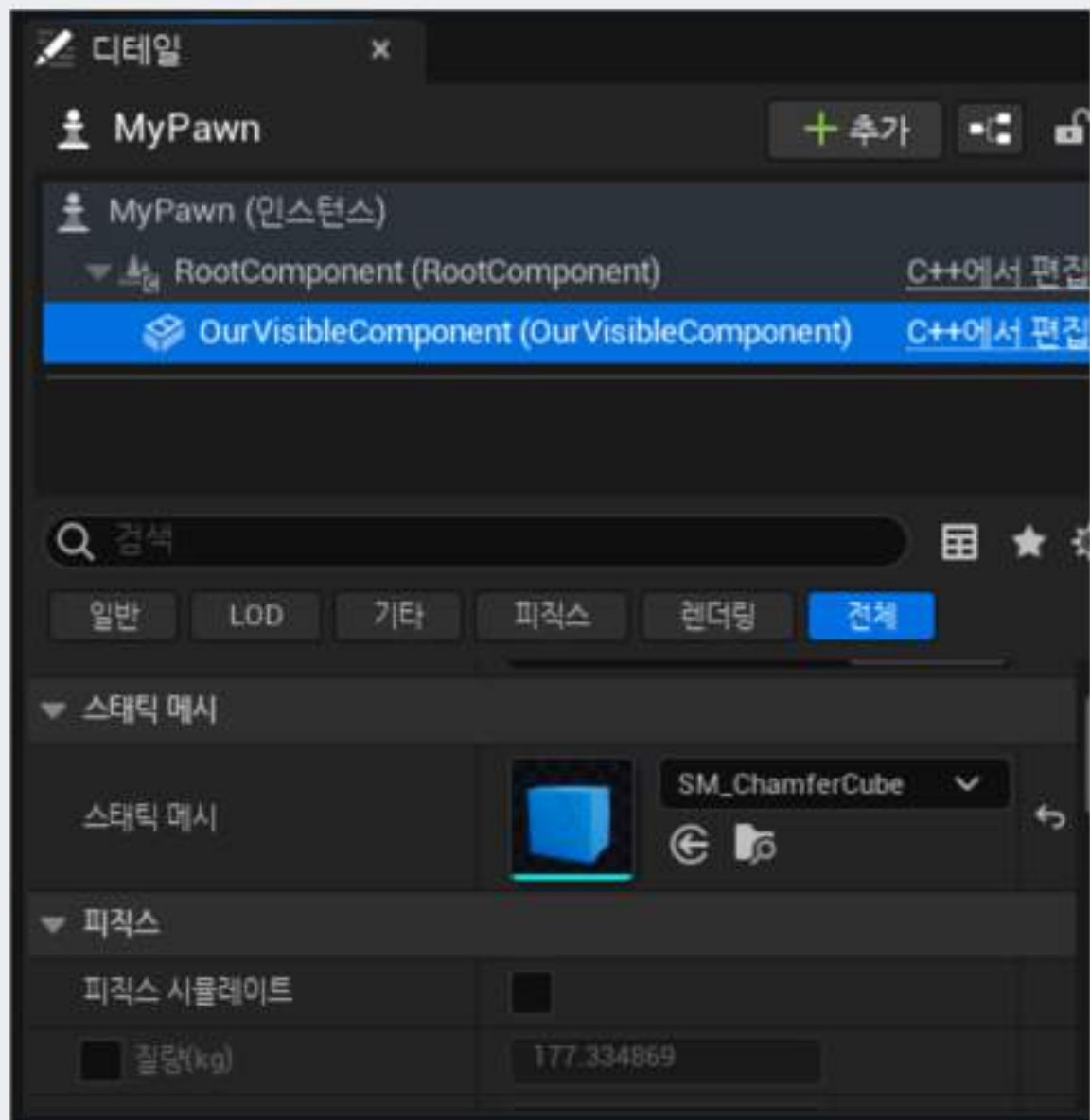
>>> 폰 클래스



완료된 블루프린트 클래스를 위 화면에 드래그 앤 드롭

## 폰을 움직여보자 (C++)

>>> 폰 클래스



우측 디테일 패널에서 OurVisibleComponent 선택후  
스태틱 메시 cube로 설정



## 참고 레퍼런스

예제 c++

[https://dev.epicgames.com/documentation/ko-kr/unreal-engine/player-input-and-pawns?application\\_version=4.27](https://dev.epicgames.com/documentation/ko-kr/unreal-engine/player-input-and-pawns?application_version=4.27)

<https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-5-documentation>