07_ 게임모드와 폰

<제목 차례>

인천대학교 컴퓨터공학부 박종승

1. 개요

이 장에서는 언리얼 엔진의 기본적인 프레임워크에 대해서 학습한다.
객체와 클래스의 개념에 대해서 학습한다.
또한, 게임 모드와 폰에 대해서 학습한다.
그리고, 블루프린트에 대한 개념이 처음으로 소개된다.

먼저, 앞으로 빈번하게 사용될 **클래스**라는 용어에 대해서 알아보자. 특히 C++에 대한 지식이 없는 개발자는 **클래스**의 개념을 이해하고 다음으로 넘어가도록 하자. 이 교재에서는 언리얼 C++에 대해서는 다루지 않으므로 최소한의 개념만 이해하자.
C++ 언어는 클래스라는 방식으로 객체를 표현하는 객체 지향적 언어이다. 언리얼은 C++에 기반한 게임 엔진이다. 따라서 언리얼에서 사용하는 객체(object)는 클래스(class)로 구현되어 있다. 우리가 지금까지 다룬 액터(actor)도 객체의 한 세부 부류이고 클래스로 구현되어 있다. 우리는 앞으로 객체의 구현물을 언급할 때에는 **클래스**라는 용어로 언급할 것이다.
언리얼에서는 C++과 더불어 시각적 스크립트 언어인 **블루프린트**(blueprint)를 지원한다. C++과 블루프린트는 그대로 대응되도록 설계되어 있다. 실제로, 작성된 블루프린트는 내부적으로 C++ 클래스로 처리된다. 따라서 C++ 클래스와 블루프린트 클래스는 대부분의 특성을 공유한다. 앞으로 우리는 클래스의 구현 방식이 C++이든 블루프린트이든 크게 중요하지 않는 경우에는 단순히 **클래스**로만 언급할 것이다. 만약 언어의 구분이 필요한 경우에는 **C++ 클래스** 또는 **블루프린트 클래스**로 언급할 것이다. 지금까지 우리는 언리얼에서의 클래스 용어의 의미에 대해서 알아보았다.

이제, 언리얼 클래스의 세부 종류에 대해서 알아보자.
언리얼 엔진은 많은 수의 객체를 클래스 형태로 제공한다. 이들 클래스는 계층 구조를 이루고 있다. 즉, 기반이 되는 부모 클래스가 있고 이를 상속하여 구현한 자식 클래스가 있다.
가장 상위의 기반 클래스를 **객체**(object) 클래스라고 한다. 따라서 모든 클래스가 객체 클래스에 해당하므로 우리는 단순히 클래스라고만 해도 객체 클래스와 동일한 의미가 된다.
객체 클래스를 상속한 자식 클래스 중에서 레벨에 배치 가능한 클래스를 **액터**(actor) 클래스라고 한다. 우리가 지금까지 언급한 액터는 바로 이 액터 클래스에 해당한다.
액터 클래스에는 중요한 클래스들이 많이 있다. 일단 몇 개만 나열해보자.
액터 클래스를 상속한 자식 클래스 중에서 플레이어나 인공지능의 물리적인 모습을 나타내는 클래스를 **폰**(pawn) 클래스이라고 한다.
그리고, 액터 클래스를 상속한 자식 클래스 중에서 플레이어 폰을 제어하는 클래스를 **플레이어 컨트롤러**(player controller) 클래스라고 한다.
그리고, 폰 클래스를 상속한 자식 클래스 중에서 기본적인 걸어다니는 이족보행 운동 기능을 가지고 있는 클래스를 **캐릭터**(character) 클래스라고 한다.
모든 객체가 클래스이므로 클래스라는 용어를 생략하는 경우가 흔하다. 즉, 액터 클래스, 폰 클래스, 캐릭터 클래스, 플레이어 컨트롤러 클래스를 객체, 액터, 폰, 캐릭터, 플레이어 컨트롤러로 언급한다.
영어 명칭은 **Actor**, **Pawn**, **Character**, **Player Controller**와 같이 각 단어는 대문자로 시작하고 여러 단어일 경우에는 공백없이 붙여서 표기하기도 한다. 실제로 이들 클래스의 C++로 구현된 클래스 명칭이 하나의 접두사 문자를 추가한 `UObject`, `APawn`, `ACharacter`, `APlayerController`인데, 우리는 접두사 문자를 제외하고 클래스 명칭으로 언급하는 것이다.

이제, 언리얼의 게임 모드에 대해서 알아보자.
게임이 플레이되려면 기본적인 규칙이 정의되어야 한다. 언리얼에서는 게임 플레이에 필요한 게임 규칙을 **게임 모드**(game mode)라는 객체로 표현한다. 게임 모드도 객체이므로 클래스로 구현된다. 게임 모드에는 게임플레이 프레임워크와 관련된 다소 복잡한 내용이 포함되어 있다. 이 장에서는 우선 쉬운 수준으로만 이해하자.
게임 모드도 객체이고 액터의 파생 클래스이다. 게임 모드 클래스는 내부적으로 게임플레이의 기본적인 규칙과 관련된 여러 클래스를 정의한다. 대표적으로, 플레이어를 제어하는 **플레이어 컨트롤러** 클래스를 정의한다. 그리고, 플레이어의 물리적인 모양을 표현하는 **폰** 클래스를 정의한다.

<참고> 언리얼의 게임플레이 프레임워크에 대해서는 다음의 문서를 참조하자.
https://docs.unrealengine.com/gameplay-framework-in-unreal-engine/

2. 클래스와 게임모드 이해하기

언리얼 엔진은 C++ 클래스와 동일한 개념으로 많은 객체를 클래스 형태로 제공한다. 또한 이들 클래스는 기반이 되는 부모 클래스와 이를 상속하여 구현한 자식 클래스의 계층 구조를 이루고 있다. 가장 상위의 기반 클래스는 **객체** 클래스이고, 그 아래에 **액터** 클래스가 있고, 그 아래에 **폰** 클래스가 있고, 그 아래에 **캐릭터** 클래스가 있다.
언리얼에서의 매우 중요한 클래스 중 하나로 **게임 모드**가 있다. 이 클래스는 게임플레이 프레임워크에서 게임 플레이에 필요한 게임 규칙을 표현하는 객체이다.
이제부터 클래스와 게임 모드의 개념을 학습하기 위해 실습해보자.

---

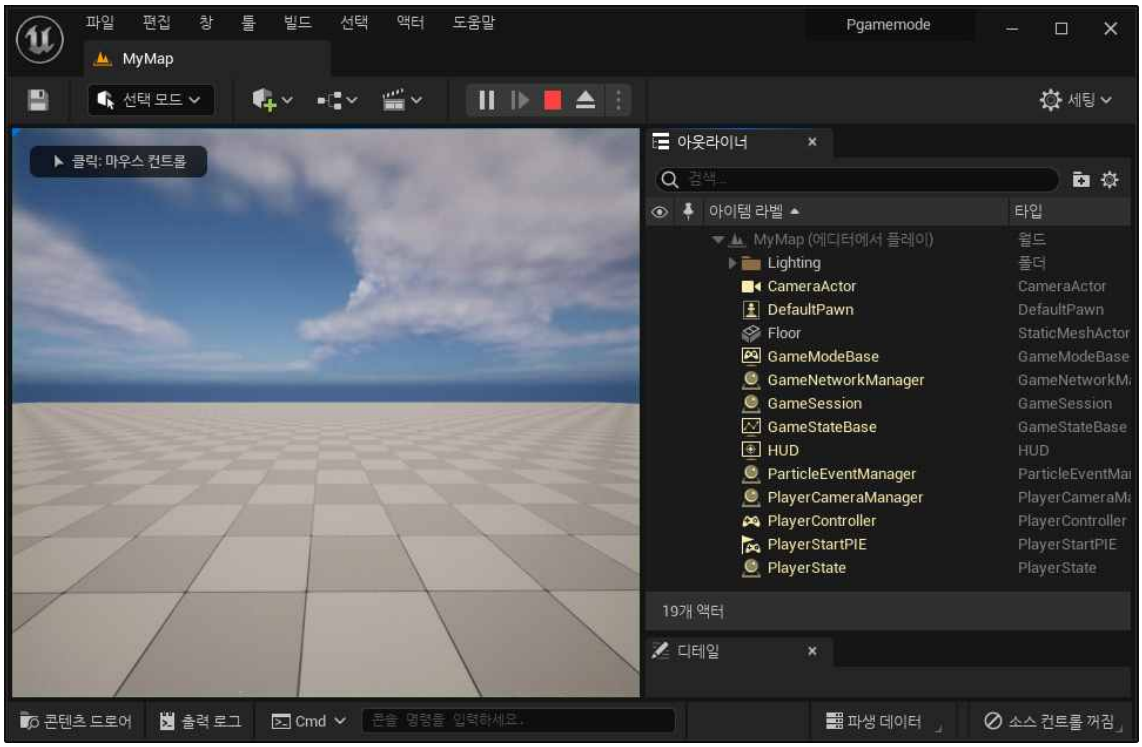
1. 새 프로젝트 Pgamemode를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Pgamemode로 입력하고 **생성** 버튼을 클릭하자. 프로젝트가 생성되고 언리얼 에디터 창이 뜰 것이다. 창이 뜨면, 메뉴바에서 **파일** » **새 레벨**을 선택하고 **Basic**을 선택하여 기본 템플릿 레벨을 생성하자. 그리고, 레벨 에디터 툴바의 저장 버튼을 클릭하여 현재의 레벨을 MyMap으로 저장하자.
그리고, **프로젝트 세팅** 창에서 **맵&모드** 탭에서의 **에디터 시작 맵** 속성값을 MyMap으로 수정하자.

2. **아웃라이너** 탭을 살펴보자.

Floor는 디폴트로 배치되는 바닥 스태틱 메시 액터이다.
그리고 **Lighting** 폴더 아래에 6개의 액터가 있다. 이들은 조명과 관련된 액터들이다. 이들 액터에 대해서 간단히 알아보자.
DirectionalLight는 태양에 해당하는 광원을 위한 방향 광원 엑터이다.
ExponentialHeightFog는 안개 효과를 표현해서 장면을 자연스럽게 만들어주는 액터이다.
SkyAtmosphere는 지구 대기권을 통과하는 빛의 산란 효과를 추정해서 야외에서의 대기 표현에 사실감을 더해주는 액터이다.
SkyLight는 표면의 반사 특성을 고려하여 하늘이나 간접광의 효과를 자연스럽게 표현하는 액터이다.
SM_SkySphere는 하늘의 시각적인 모습을 표현하기 위한 스태틱 메시 액터이다.

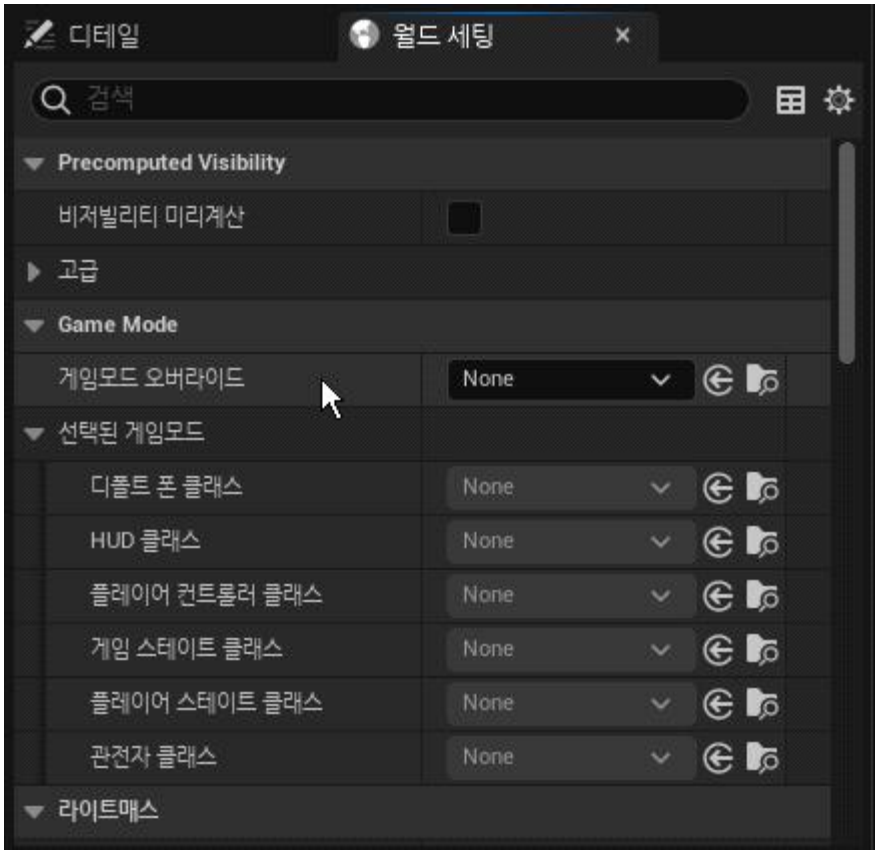
VolumetricCloud는 구름을 자연스럽게 표현하는 액터이다.
이들 액터에 대해서는 조명을 학습할 때에 자세히 다룰 것이다.

3. 이제, 플레이해보자. 그리고 **아웃라이너**를 살펴보자.



새로운 액터들이 생기는 것을 알 수 있다. **CameraActor**, **DefaultPawn**, **GameModeBase**, **GameNetworkManager**, **GameSession**, **GameStateBase**, **HUD**, **ParticleEventManager**, **PlayerCameraManager**, **PlayerController**, **PlayerStartPIE**, **PlayerState**의 액터가 새로 생성된다.
새로 생성된 액터에 대해서 이해하려면 게임 엔진의 자세한 내용을 학습해야 하므로 여기에서 모두 설명할 수는 없다. 단계적으로 하나씩 배워보기로 하자.
새로 생성된 액터 중에는 게임 모드 액터가 포함되어 있다. **GameModeBase**가 바로 게임 모드 액터이다. 다른 액터들도 거의 모두 게임 모드 액터와 관련이 있다. 즉, 게임 모드에서 어떤 액터들을 생성할 지를 지정해두었다는 의미이다.

4. 각 레벨마다 자신이 사용할 게임 모드를 지정하게 되어 있다. 그 레벨이 플레이될 때에 지정된 게임 모드가 적용되는 것이다.
메뉴바에서 **창** » **월드 세팅**을 클릭하자. **디테일** 탭 옆에 **월드 세팅** 탭이 추가될 것이다.

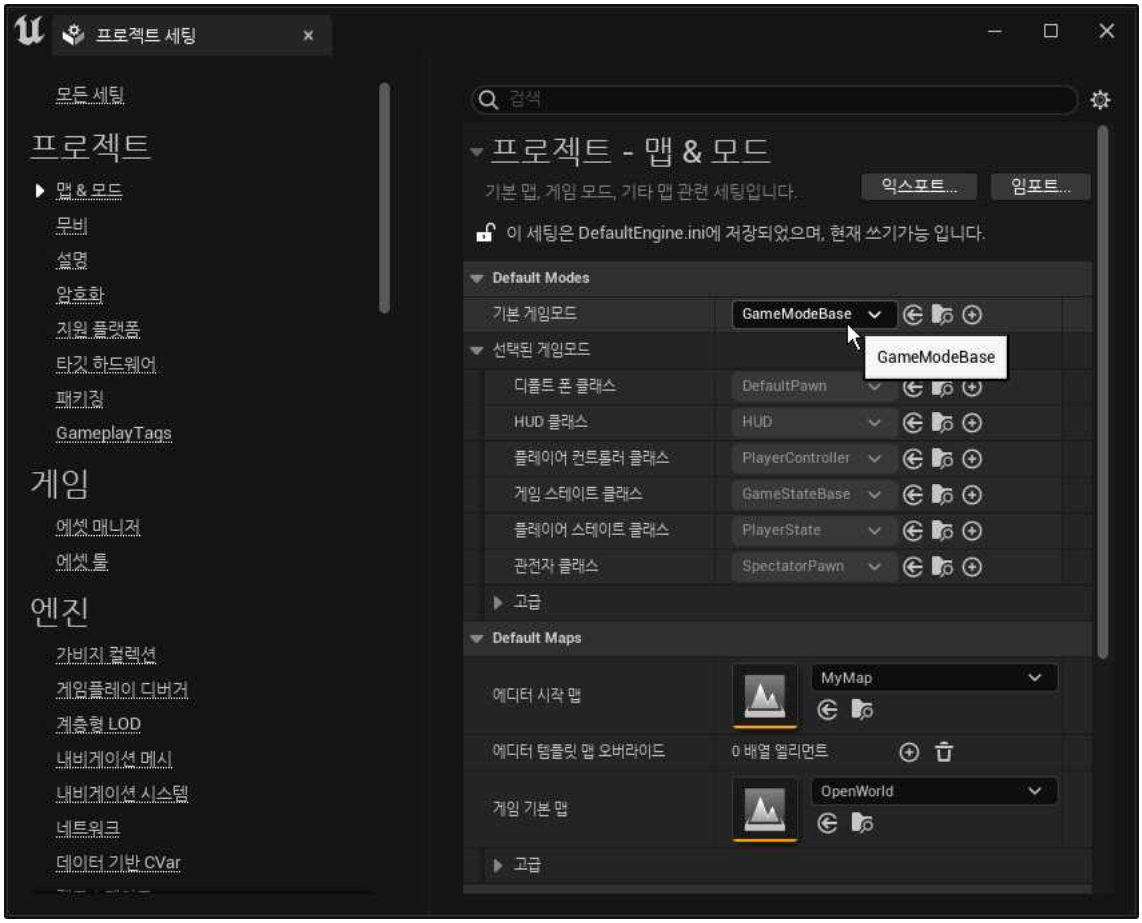


디테일 탭이 레벨에 배치된 각 액터의 상세 속성을 설정하는 창인 반면에, **월드 세팅** 창은 현재 레벨의 전역적 속성을 설정하는 창이다.

월드 세팅 창을 살펴보자. **GameMode** 영역이 있다. **게임모드 오버라이드** 속성이 **None**으로 되어 있다. 이것의 의미는 현재의 레벨에서는 게임 모드를 재정의하지 않고 프로젝트에서 정한 디폴트 게임 모드를 사용하겠다는 의미이다. 그 아래의 **선택된 게임모드**를 펼쳐보자. **GameMode**의 6개의 세부 속성값이 모두 **None**으로 되어 있고 수정할 수 없도록 되어 있다. 현재의 레벨에서는 게임 모드를 재정의하지 않기 때문에 세부 속성값도 수정할 수 없고 프로젝트의 디폴트 값이 그대로 사용된다.

<참고> 월드 세팅 창에 대한 자세한 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/world-settings-in-unreal-engine/

5. 메뉴바에서 **편집** » **프로젝트 세팅...**을 클릭 **프로젝트 세팅** 창을 열어보자. 왼쪽의 **프로젝트** » **맵 & 모드** 탭을 클릭하고 오른쪽에서 **Default Modes** 영역을 살펴보자.



기본 게임모드가 **GameModeBase**로 되어 있을 것이다. 따라서 프로젝트의 레벨에서 디폴트로 사용되는 게임 모드가 **GameModeBase** 클래스임을 알 수 있다. 우리는 게임을 플레이한 후에 **아웃라이너**에서 **GameModeBase** 액터가 생성된 것을 확인하였는데 이 레벨의 게임 모드로 프로젝트의 디폴트 값인 **GameModeBase** 클래스가 사용되었기 때문이다.

GameModeBase 클래스는 액터의 파생 클래스이고 모든 게임 모드 클래스의 부모 클래스이다. 게임 모드는 내부에서 게임플레이에 사용될 6개의 클래스를 명시하는 속성들이 있다. 이들 6개의 클래스는 바로 **디폴트 폰 클래스**(Default Pawn Class), **HUD 클래스**(HUD Class), **플레이어 컨트롤러 클래스**(Player Controller Class), **게임 스테이트 클래스**(Game State Class), **플레이어 스테이트 클래스**(Player State Class), **관전자 클래스**(Spectator Class)이다. 이들 속성에 명시된 값은 모두 클래스 이름들이다. 디폴트로 명시된 속성값은 각각 **DefaultPawn**, **HUD**, **PlayerController**, **GameStateBase**, **PlayerState**, **SpectatorPawn** 클래스이다. **GameModeBase** 클래스는 엔진 클래스이므로 속성값에 명시된 클래스명은 수정할 수 없도록 되어 있다. 앞으로 각 속성에 대해서는 하나씩 점차적으로 알아갈 것이다.

6. 게임 모드의 각 속성 중에서 먼저 **디폴트 폰 클래스**에 대해서 알아보자. 이 속성에는 현재 레벨에서 플레이어가 디폴트로 사용할 폰을 명시한다. 폰은 캐릭터의 물리적인 모습을 나타내는 액터이다. 현재 **DefaultPawn**이라는 클래스로 지정되어 있다. 우리는 게임을 플레이한 후에 **아웃라이너**에서 **DefaultPawn** 액터가 생성된 것을 확인하였는데, 게임 모드에서 플레이어 폰 액터로 사용될 **DefaultPawn**을 생성하였기 때문이다.

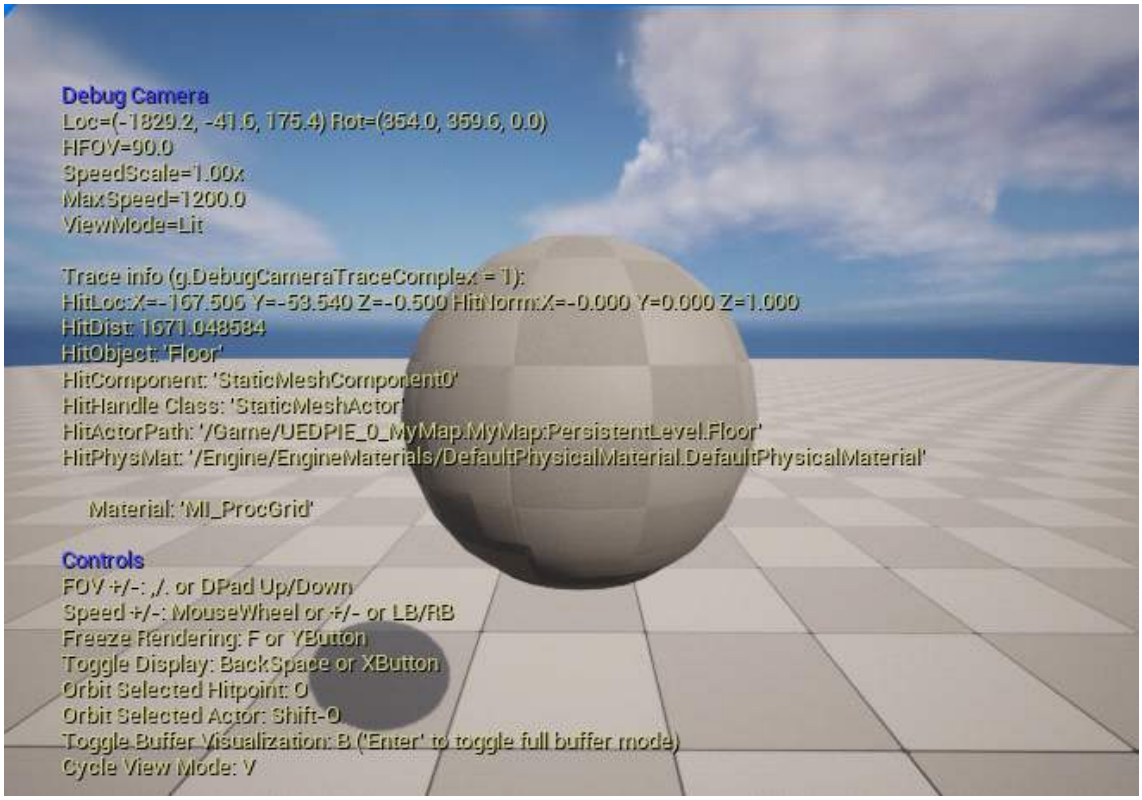
7. 게임 모드의 **디폴트 폰 클래스** 속성에 명시된 **DefaultPawn** 클래스는 레벨에 미리 배치되어 있는

것이 아니라 플레이 시에 생성된다.
또한 **DefaultPawn** 클래스는 게임 플레이어의 물리적인 실체에 해당하므로 플레이 시에 자신의 모습을 볼 수도 없다.
우리는 콘솔 명령어를 통해서 **DefaultPawn** 클래스의 모습을 확인해보자.
레벨 에디터에서는 개발자를 위하여 **디버그 카메라 모드**를 지원한다. 이 모드에서는 게임 플레이 중에 플레이어와 무관하게 카메라 시점을 마음대로 조작할 수 있도록 허용한다.
먼저 플레이 버튼을 눌러 게임을 플레이하자. 그다음, 뷰포트 내부 영역을 한번 클릭하여 마우스 제어가 에디터가 아닌 게임플레이의 의해 점유되도록 하자. 그다음, **Tab** 키 위에 있는 '`' 특수문자 키를 누르자. 뷰포트의 하단에 콘솔 명령 입력상자가 나타날 것이다. 입력 상자에서 'ToggleDebugCamera'를 입력하고 엔터키를 누르자. **디버그 카메라 모드**로 진입할 것이다.

<참고> 콘솔 명령 입력상자에서 일부만 입력해도 일치된 명령어들이 나열된다. 해당하는 명령어를 목록에서 키보드 화살표로 선택하면 된다. 문자열이 일치되어 하나의 명령어만 나열되는 때에는 **Tab** 키를 누르면 나머지 문자가 자동으로 입력된다. 콘솔 명령 입력상자가 보이는 상태에서 '`' 키를 한번 더 누르면 콘솔 입력 영역이 넓게 표시된다. '`' 키를 또한번 더 누르면 콘솔 입력 영역이 사라진다.

<참고> 뷰포트에서 '`' 키를 누르는 대신에, 뷰포트 하단에 항상 표시되는 콘솔 명령 입력상자에서 'ToggleDebugCamera'를 입력해도 된다. 다만 문자열 일치 기능이 없으므로 전체를 입력해야 한다.

8. **디버그 카메라 모드**에서는 카메라와 관련된 여러 문자 정보가 표시된다. 또한 카메라 조작으로 플레이어와 무관하게 시점을 움직여볼 수 있다.
카메라 조작으로 원래 위치에 있던 자신의 모습을 살펴보자. 약간 뒤로 물러서면 공중에 떠 있는 구체가 있을 것이다. 이것이 플레이어 폰에 해당하는 **DefaultPawn** 액터의 모습이다.



지금까지, 플레이 도중에 플레이어의 모습을 관찰할 수 있는 **디버그 카메라 모드**에 대해서 알아보았다. 자주 사용하는 기능은 아니지만 **DefaultPawn**의 실체를 확인하기 위해서 사용해보았다.

9. 현재 레벨의 게임 모드가 플레이어 폰으로 사용하는 **DefaultPawn**은 매우 단순한 형태인 구체로 구현되어 있다. 비록 폰의 모양은 단순한 구체 모양이지만 플레이어 입력 제어 기능을 모두 갖추고 있어서 플레이 중에 플레이어 카메라를 회전해보거나 월드의 다른 위치로 이동해볼 수 있다. 즉, **DefaultPawn** 클래스에서 플레이어 입력 제어 기능을 구현하여 제공하기 때문에 플레이어가 레벨을 내비게이션할 수 있는 것이다.

지금까지 클래스와 게임 모드의 개념을 학습하였다.

3. 커스텀 플레이어 폰 만들기

이 절에서는 디폴트로 사용되는 플레이어 폰 대신에 우리가 직접 플레어어 폰을 만들어보자. 우리는 블루프린트로 플레이어 폰을 만들어볼 것이다.

여기에서 블루프린트가 처음으로 등장하였다. 블루프린트에 대해서 알아보자.
블루프린트(blueprint)는 언리얼 엔진에서 사용되는 시각 스크립트 시스템(visual scripting system)이다. 또한, 블루프린트를 사용하여 만든 객체도 블루프린트라고 한다. 즉 블루프린트는 시각 스크립트 시스템을 의미하기도 하고 동시에 그 시스템으로 제작된 객체를 의미하기도 하므로 상황에 따라서 해석하면 된다.
언리얼에서 객체의 표현물인 클래스는 C++로 만들 수도 있고 블루프린트로 만들 수도 있다. 어느 방법으로 만들어도 거의 대응이 되며 유사한 기능을 수행할 수 있다. 그만큼 블루프린트는 클래스를 만드는 강력한 수단이다.
앞으로 우리는 블루프린트에 대한 여러 지식을 하나씩 배워나갈 것이다.

레벨에 배치할 수 있는 블루프린트 클래스를 액터라고 한다. 하나의 액터는 여러 컴포넌트를 담는 컴포넌트의 컨테이너로 볼 수 있다. 이제부터, 컴포넌트에 대해서 알아보자.
컴포넌트(component)는 액터에 추가되어 동작하는 클래스이다. 즉, 컴포넌트도 하나의 블루프린트 클래스이지만 독립적으로 레벨에 배치되는 액터 클래스가 아니라 다른 액터 클래스에 포함되어 기능이 추가되도록 설계된 액터가 아닌 클래스이다.
각 컴포넌트는 특정 기능을 수행하는 부속품으로 생각할 수 있다. 어떤 컴포넌트들을 액터에 담는지에 따라서 액터의 기능이 결정된다. 컴포넌트의 개념을 사용하면 복잡한 기능의 액터도 쉽게 구현할 수 있다.

이제, 컴포넌트의 유형에 대해서 알아보자.
컴포넌트의 유형에는 액터 컴포넌트, 씬 컴포넌트, 프리미티브 컴포넌트가 있다. 먼저, 액터 컴포넌트(actor component)는 가장 기반이 되는 컴포넌트이고 트랜스폼 정보를 가지지 않는다. 그다음, 씬 컴포넌트(scene component)는 액터 컴포넌트의 자식 클래스이며 트랜스폼을 추가한 컴포넌트이다. 그다음, 프리미티프 컴포넌트(primitive component)는 씬 컴포넌트의 자식 클래스이며 그래픽 표현을 추가하였다. 이들에 대해서는 추후에 자세히 알아보기로 하자.
우선, 우리는 프리미티프 컴포넌트는 씬 컴포넌트의 특별한 유형으로 생각하고 특별히 별도로 고려하지 않기로 하자. 따라서 컴포넌트의 유형에는 액터 컴포넌트와 씬 컴포넌트의 두 유형이 있다고 생각하자. 즉 트랜스폼 정보를 가지지 않는 액터 컴포넌트와 트랜스폼 정보를 가지는 씬 컴포넌트의 두 유형이 있다고 생각하자.

블루프린트 클래스에 포함되어 있는 컴포넌트는 액터 컴포넌트와 씬 컴포넌트의 두 유형으로 나눌 수 있다. 액터 컴포넌트는 트랜스폼 속성이 없는 컴포넌트이고 씬 컴포넌트는 트랜스폼 속성이 있는 컴포넌트이다. 트랜스폼 속성이 있는 씬 컴포넌트는 블루프린트 클래스 내에서 하나의 단일 계층 구조를 이루어야 한다. 계층 구조의 가장 상위의 컴포넌트를 루트 컴포넌트라고 하며 이 컴포넌트의 트랜스폼이 전체 블루프린트 클래스의 트랜스폼을 대표한다.
계층 구조에서 루트 컴포넌트는 절대적인 월드 좌표계로 표시되며 루트 컴포넌트가 아닌 컴포넌트

는 트랜스폼이 모두 부모 컴포넌트에 상대적으로 정의된다.
씬 컴포넌트가 아닌 액터 컴포넌트는 계층 구조에 참여할 수 없고 별도로 나열된다.

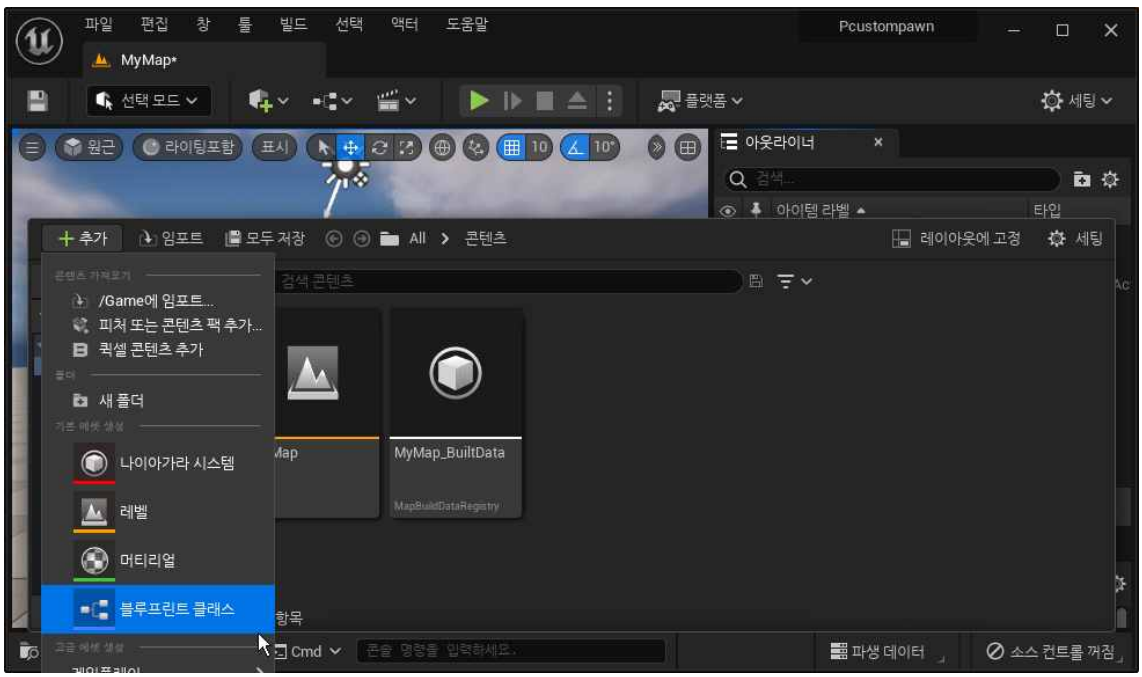
<참고> 컴퍼넌트와 관련된 자세한 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/components-in-unreal-engine/

지금부터 커스텀 플레이어폰을 만드는 방법을 학습해보자.

---

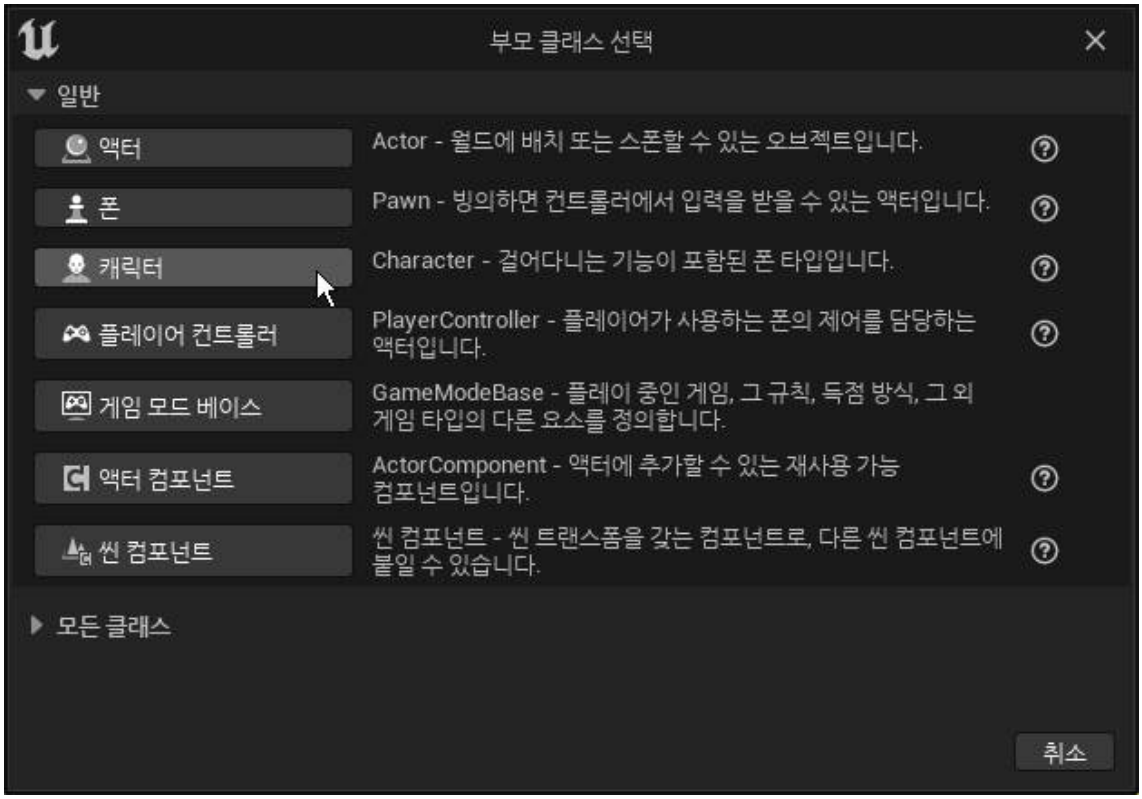
1. 새 프로젝트 **Pcustompawn**을 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Pcustompawn으로 입력하고 **생성** 버튼을 클릭하자. 프로젝트가 생성되고 언리얼 에디터 창이 뜰 것이다.
창이 뜨면, 메뉴바에서 **파일** » **새 레벨**을 선택하고 **Basic**을 선택하여 기본 템플릿 레벨을 생성하자. 그리고, 레벨 에디터 툴바의 저장 버튼을 클릭하여 현재의 레벨을 MyMap으로 저장하자.
그리고, **프로젝트 세팅** 창에서 **맵&모드** 탭에서의 **에디터 시작 맵** 속성값을 MyMap으로 수정하자.

2. 이제부터, 커스텀 플레이어 폰을 생성해보자.
레벨 에디터의 하단바에 있는 **콘텐츠 드로어** 버튼을 클릭하여 **콘텐츠 드로어**가 나타내게 하자.
그다음, 왼쪽 폴더 계층구조에서 생성될 애셋이 저장될 폴더를 선택하자. 우리는 생성할 애셋이 많지 않으므로 하위 폴더를 따로 생성하지 말고 간편하게 최상위 폴더인 **콘텐츠** 폴더에 저장하도록 하자. **콘텐츠** 폴더는 최초에 디폴트로 선택되는 폴더이므로 바로 **+추가** 버튼을 클릭하면 된다.
그다음, **콘텐츠 드로어**의 상단 툴바에서 **+추가**를 클릭하자.



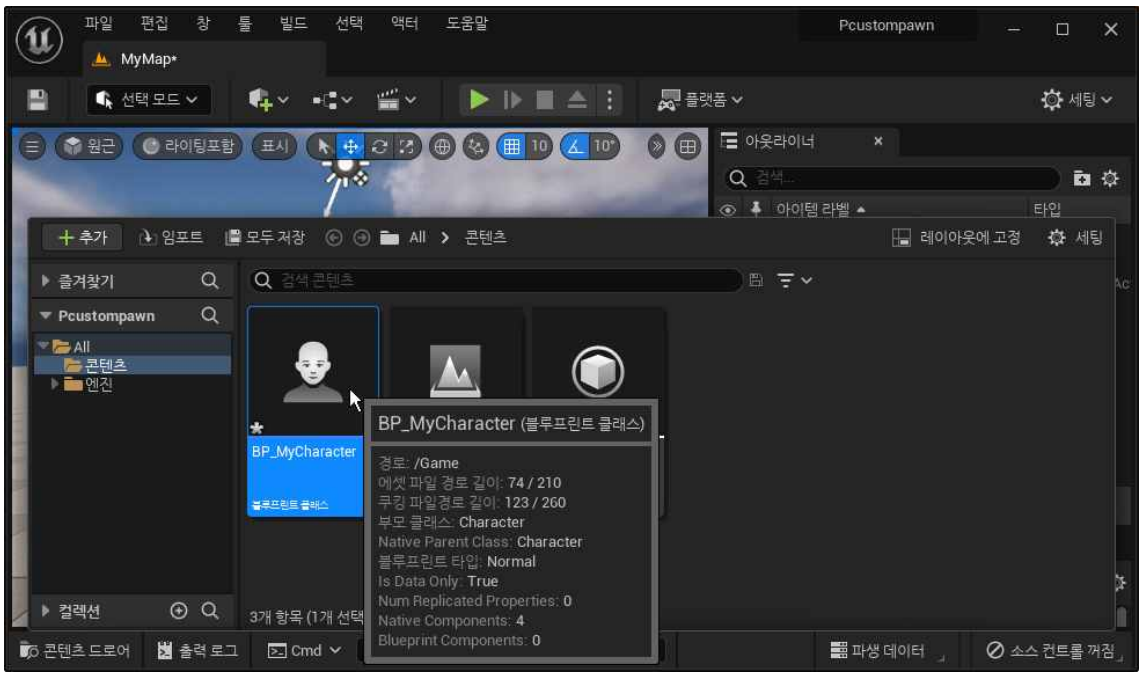
드롭다운 메뉴가 뜨면 생성할 애셋 타입을 선택하면 된다. 많은 다양한 종류의 애셋을 생성할 수 있다. 우리는 **블루프린트 클래스**를 선택하자.

3. 그다음, **부모 클래스 선택** 창이 뜬다. 이 창에서는 생성할 블루프린트 클래스의 부모 클래스를 선택한다.

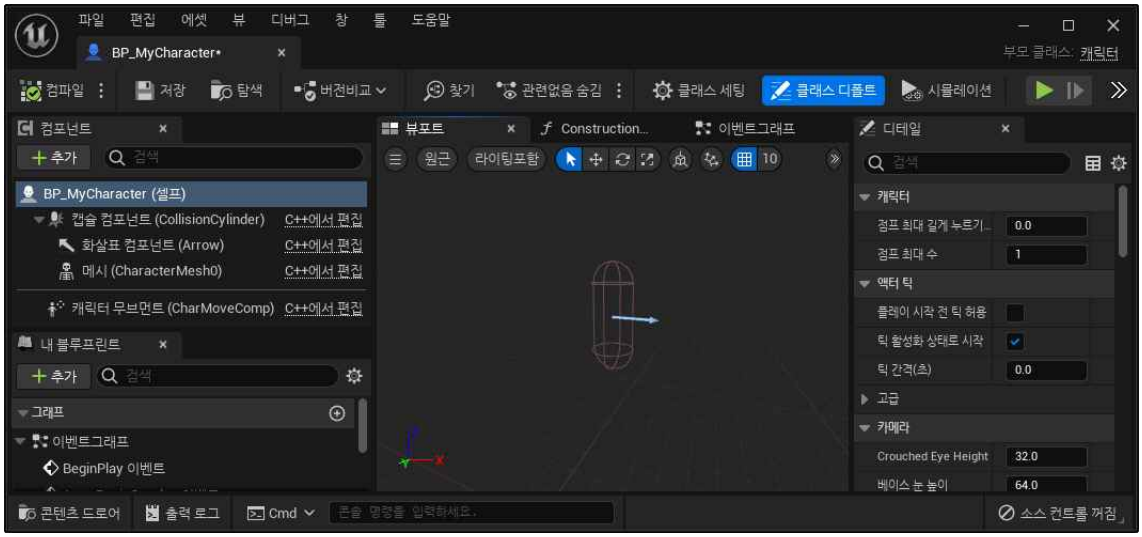


창에서, **일반** 영역에는 많이 사용되는 부모 클래스가 나열되어 있고 **모든 클래스** 영역에는 선택할 수 있는 모든 클래스가 나열되어 있다.

4. 우리는 부모 클래스로 **캐릭터(Character)**를 선택하자. **캐릭터** 클래스는 폰을 상속한 폰의 자식 클래스이며 폰의 기능에 이족 보행 기능이 추가되어 있는 클래스이다.
생성된 블루프린트 클래스의 이름을 **BP_MyCharacter**로 지정하자.

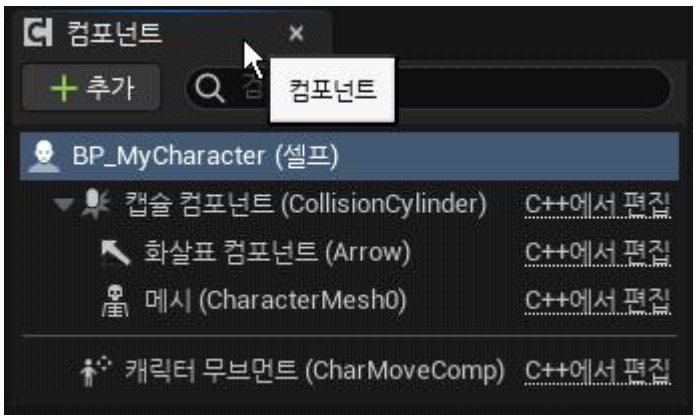


5. 생성된 **BP_MyCharacter** 애셋 아이콘을 더블클릭하자. 새 창이 뜰 것이다. 이 창을 **블루프린트 에디터**라고 한다.



블루프린트 에디터는 블루프린트를 편집하는 매우 중요한 창이다.
블루프린트 에디터의 레이아웃을 살펴보자. 레벨 에디터처럼 메뉴바가 있고 툴바가 있다. 중간에 뷰포트가 있고 오른쪽에 디테일 탭이 있다. 뷰포트에서의 조작은 레벨 에디터 뷰포트나 스태택 메시 에디터 뷰포트에서와 같이 모두 동일하다.

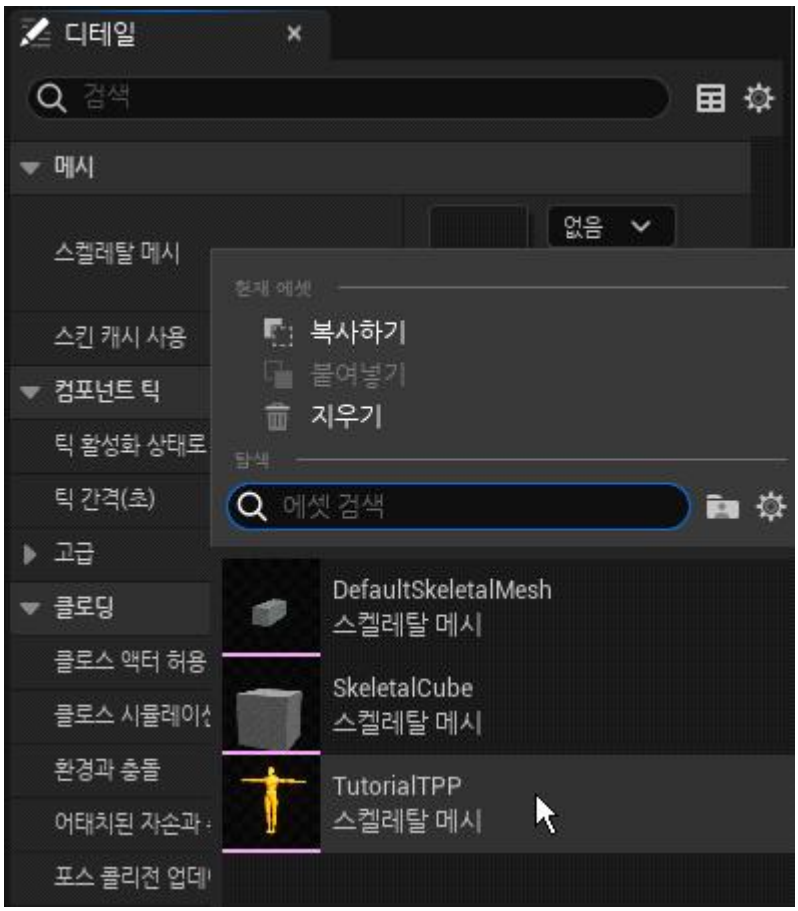
6. **블루프린트 에디터**에서 왼쪽에는 컴포넌트 탭이 있다. 이 탭에서는 현재의 블루프린트를 구성하는 컴포넌트의 목록을 보여준다.



캐릭터(**Character**) 클래스는 디폴트로 4개의 컴포넌트를 포함하고 있다. **캡슐 컴포넌트**(**Capsule Component**)는 단순 콜리전 볼륨으로 사용될 컴포넌트이다. **화살표 컴포넌트**(**Arrow Component**)는 객체가 바라보는 방향을 표시하는 컴포넌트이다. **메시**(**Mesh**)는 플레이어의 모습을 표현하는 스켈레탈 메시 컴포넌트이다. **캐릭터 무브먼트**(**Character Movement**)는 걷기와 같은 플레이어의 움직임을 처리하는 컴포넌트이다. 이 컴포넌트에는 비탈길을 오르거나, 바닥이 없는 곳에서 떨어진다거나 하는 것들이 구현되어 있어서, 액터를 쉽게 움직일 수 있다. 즉, 액터의 위치를 계산하고 조금씩 이동시키는 것이 아니라, 어떤 방향으로 어느 속도로 이동하라는 식의 정보만 설정하면 이 컴포넌트가 관련된 처리를 모두 수행한다.
컴포넌트 탭에서 가장 상단에는 블루프린트 클래스의 이름인 **BP_MyCharacter**가 표시된다. 그 아래에는 블루프린트에 포함되어 있는 각 컴포넌트가 나열된다. 블루프린트 클래스의 이름인 **BP_MyCharact**

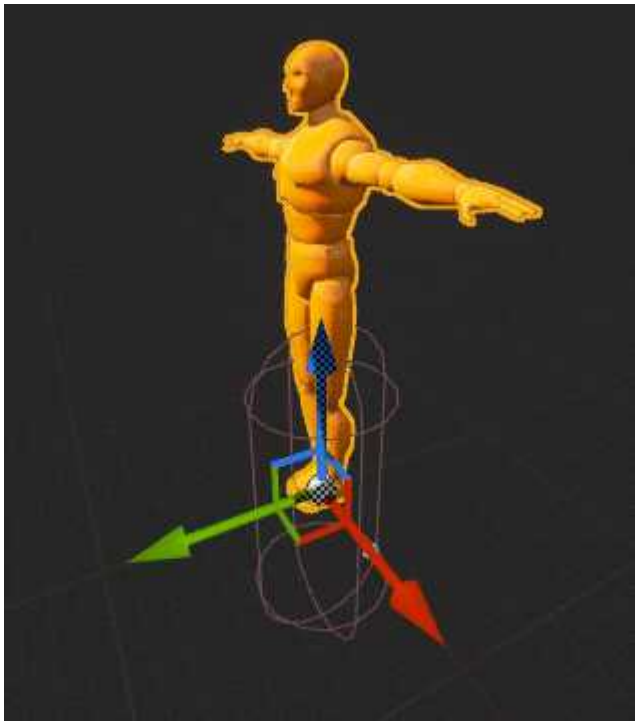
er를 클릭하면 디테일 탭에는 블루프린트 클래스의 각 속성에 대한 디폴트 값이 표시된다. 툴바의 **클래스 디폴트** 버튼을 클릭해도 동일한 정보가 표시된다.
한편, 각 컴포넌트를 클릭하면 해당 컴포넌트의 속성에 대한 디폴트 값이 표시된다. 즉, 가장 위의 블루프린트 클래스를 클릭하면 블루프린트 클래스의 속성 멤버 변수의 디폴트 값이 표시되고 각 컴포넌트를 클릭하면 해당 컴포넌트 클래스의 속성 멤버 변수의 디폴트 값이 표시된다.
블루프린트에 포함되어 있는 일부 컴포넌트가 계층 구조를 이루고 있다. 씬 컴포넌트는 모두 단일 계층 구조에 부착되어 있어야 한다. 위의 컴포넌트 탭을 보면 **캡슐 컴포넌트**가 루트 컴포넌트이며 그 아래의 **화살표 컴포넌트**와 **메시**의 두 컴포넌트가 자식 컴포넌트로 붙어있다. 계층 구조에서 루트 컴포넌트를 제외한 다른 컴포넌트는 트랜스폼이 부모 컴포넌트에 상대적으로 정의된다.
한편, 씬 컴포넌트가 아닌 액터 컴포넌트인 **캐릭터 무브먼트**는 계층 구조와 별개로 독립적으로 나열되어 있다.

7. **컴포넌트** 패널에서 **Mesh** 컴포넌트를 선택하자.
그다음, 오른쪽의 **디테일** 패널에서 메시 영역 아래의 **스켈레탈 메시**(SkeletalMesh)속성을 찾자. 디폴트로 **없음**으로 되어 있을 것이다. 플레이어 폰의 메시가 없다는 의미이므로 **디버그 카메라 모드**로 살펴보더라도 보이는 것이 없을 것이다. **없음**을 클릭하고 리스트에서 **TutorialTPP**를 선택하자.

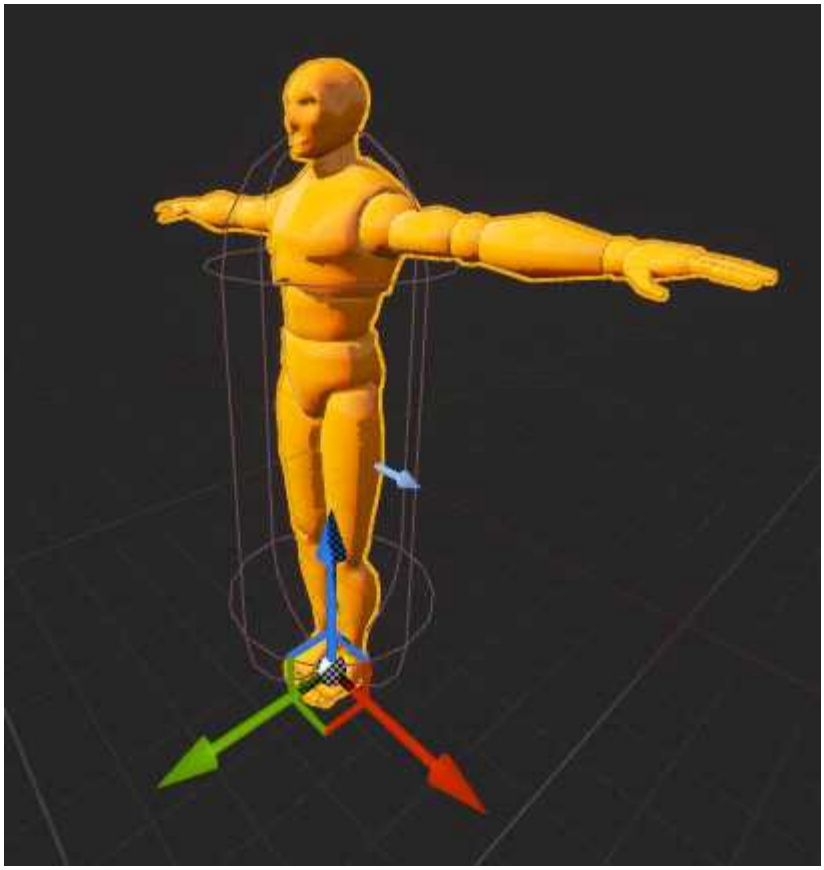


만약 **TutorialTPP**가 리스트에 나타나지 않는다면 검색상자의 오른쪽에 있는 설정 아이콘을 클릭하고 **엔진 콘텐츠 표시**에 체크하면 보일 것이다.

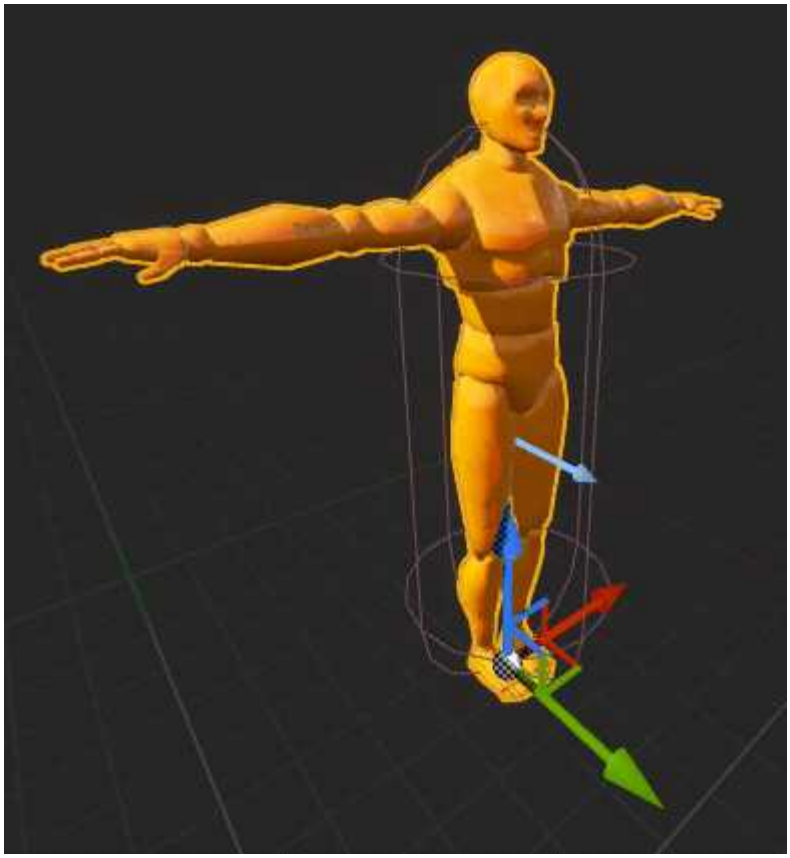
8. 뷰포트에 캐릭터 메시가 나타날 것이다.



9. 캐릭터 메시가 **Capsule Component**에 비해서 너무 위로 올라와 있다. 계층 구조에서 **Mesh**는 **Capsule Component**가 아래에 부착되어 있다. **Mesh**의 트랜스폼은 부모인 **Capsule Component**에 상대적으로 정의되는데, 디폴트로 메시의 피벗인 발바닥 위치가 부모의 피벗 위치인 캡슐의 중간에 놓여 있다. 우리는 **Mesh**를 약간 아래가 되도록 트랜스폼을 수정해보자. 왼쪽 **컴포넌트** 탭에서 **Mesh** 컴포넌트를 선택하고 디테일 탭에서 트랜스폼의 위치값을 (0,0,0)에서 (0,0,-90)으로 수정하자.



10. 화면의 중간에 X축으로 향하고 있는 하늘색 화살표가 있다. 이것은 **Arrow Component**이고 플레이어가 향하는 방향을 알려주는 특별한 컴포넌트이다. 즉 현재 X축이 전방 방향이라는 의미이다. 따라서 메시를 회전시키자. **Mesh** 컴포넌트를 선택하고 디테일 탭에서 **트랜스폼**의 **회전**값을 (0,0,0)에서 (0,0,-90)으로 수정하자. 다음과 같은 모습이 될 것이다.



참고로, 위의 메시의 이동 기즈모의 좌표축 방향이 바뀌었는데 이는 현재의 트랜스폼 기즈모의 좌표계 기준이 월드 좌표계가 아니라 로컬 좌표계로 되어있기 때문이다.

11. 블루프린트가 수정된 후에는 반드시 컴파일을 해야 한다.
블루프린트 에디터는 블루프린트 클래스의 소스 코드에 해당하는 부분을 편집하는 도구이다. 따라서 블루프린트 에디터에서수정 작업을 한 후에는 반드시 다시 컴파일을 해야 수정된 결과가 엔진으로 반영된다. 우리는 앞으로 블루프린트가 수정될 때마다 컴파일하는 습관을 가지도록 하자.
이제, 툴바의 **컴파일** 버튼을 눌러 컴파일하고, **저장** 버튼을 눌러 저장하자.

툴바에서 위의 컴파일 아이콘처럼 노란색 물음표 아이콘으로 표시된다면 컴파일되지 않아서 컴파일이 필요한 상황이라는 의미이다. 따라서 컴파일 버튼을 클릭하여 컴파일해주어야 한다. 컴파일된 후에는 컴파일 아이콘이 녹색 체크 아이콘으로 표시된다.
만약 수정된 내용에 오류가 있다면 컴파일 과정에서 오류를 알려준다.

12. 블루프린트 에디터의 간략 형태에 대해서 알아보자.
콘텐츠 브라우저에서 블루프린트 클래스를 더블클릭하면 블루프린트 에디터가 뜬다. 이 때에 종종 아래 그림과 같이 기본값만 표시되는 간략 형태의 블루프린트 에디터가 뜰 때가 있다.

일부 속성값만 수정된 간단한 블루프린트의 경우에 간략 형태의 에디터로 보여준다. 만약 정식 블루프린트 에디터의 레이아웃으로 열기를 원하는 경우에는 툴바 바로 아래의 안내문의 오른쪽에 있는 **풀 블루프린트 에디터 열기**를 클릭하면 된다.

지금까지 커스텀 플레이어 폰을 블루프린트 클래스로 만들어보았다.

4. 커스텀 게임 모드 만들기

이전에서 우리는 커스텀 플레이어 폰을 만들었다. 이제, 게임이 플레이될 때에 우리가 만든 커스텀 플레이어 폰을 사용해보자.

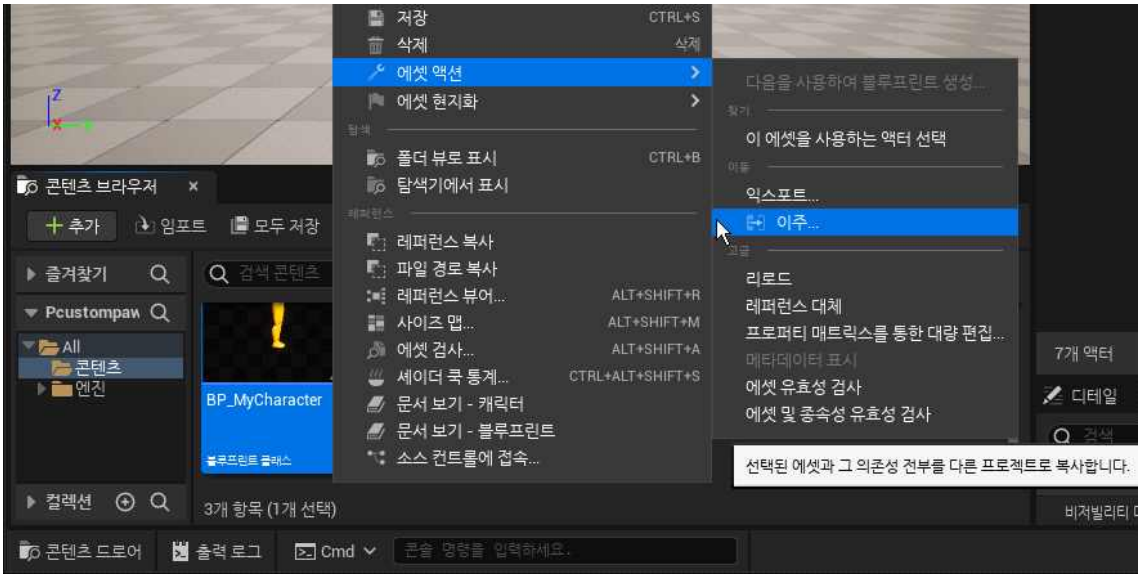
플레이 시에 사용될 플레이어 폰은 레벨의 게임 모드에서 명시하고 있다. 게임 모드의 **Default Pawn Class** 속성에 명시된 클래스가 게임 플레이 시에 사용될 플레이어 폰 클래스이다.
한편, 새로 생성된 프로젝트에서는 레벨에서 사용할 게임 모드는 프로젝트의 디폴트 게임 모드를 사용하도록 되어 있다. 프로젝트의 디폴트 게임 모드는 엔진 클래스인 **GameModeBase**로 되어 있다. **GameModeBase** 클래스에서는 **Default Pawn Class** 속성에 **DefaultPawn** 클래스가 지정되어 있다. **GameModeBase** 클래스나 **DefaultPawn** 클래스는 모두 엔진 클래스로 수정이 불가능하다. 따라서 우리는 우리의 커스텀 게임 모드 클래스를 만들고 이를 레벨의 게임 모드로 지정해주어야 한다.

지금부터 커스텀 게임 모드를 만들고 이를 레벨의 게임 모드로 지정하는 방법을 학습해보자.

---

1. 새 프로젝트 `Pcustomgamemode`를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 `Pcustomgamemode`로 입력하고 **생성** 버튼을 클릭하자. 프로젝트가 생성되고 언리얼 에디터 창이 뜰 것이다.
창이 뜨면, 메뉴바에서 **파일** » **새 레벨**을 선택하고 **Basic**을 선택하여 기본 템플릿 레벨을 생성하자. 그리고, 레벨 에디터 툴바의 저장 버튼을 클릭하여 현재의 레벨을 `MyMap`으로 저장하자.
그리고, **프로젝트 세팅** 창에서 **맵&모드** 탭에서의 **에디터 시작 맵** 속성값을 `MyMap`으로 수정하자.

2. 먼저, 이전 프로젝트해서 생성해둔 플레이어 폰 클래스인 **BP_MyCharacter** 블루프린트 클래스를 가져오자. 한 프로젝트에서의 애셋을 다른 프로젝트로 복사하는 것을 이주(migrate) 라고 한다. 이제부터 이주 방법에 대해서 알아보자.
먼저, 이전 프로젝트인 `Pcustompawn` 프로젝트를 열어야 한다. UE 프로그램을 다시 실행하여 이 프로젝트를 열자. 그다음, **콘텐츠 브라우저**에서 **BP_MyCharacter** 애셋 위에서 우클릭하자. 그다음, 팝업 메뉴에서 **에셋 액션** » **이주**를 선택하자.

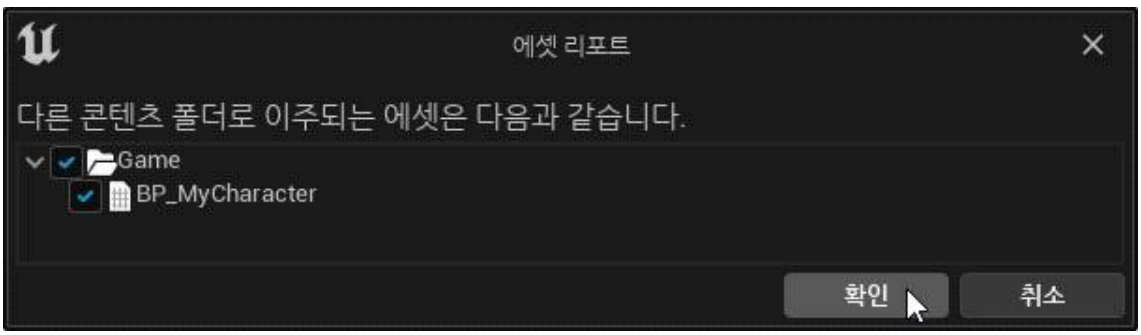


<참고> 만약 이주할 애셋이 많다면 여러 애셋을 선택하고 그 위에서 우클릭한 다음, 팝업 메뉴에서 **에셋 액션 » 이주**를 선택하면 여러 애셋을 선택할 수 있다.
만약 **콘텐트 브라우저**에서 폴더 전체를 이주하고 싶은 경우에는 폴더 위에서 우클릭하고 팝업 메뉴에서 **이주**를 선택하면 된다.

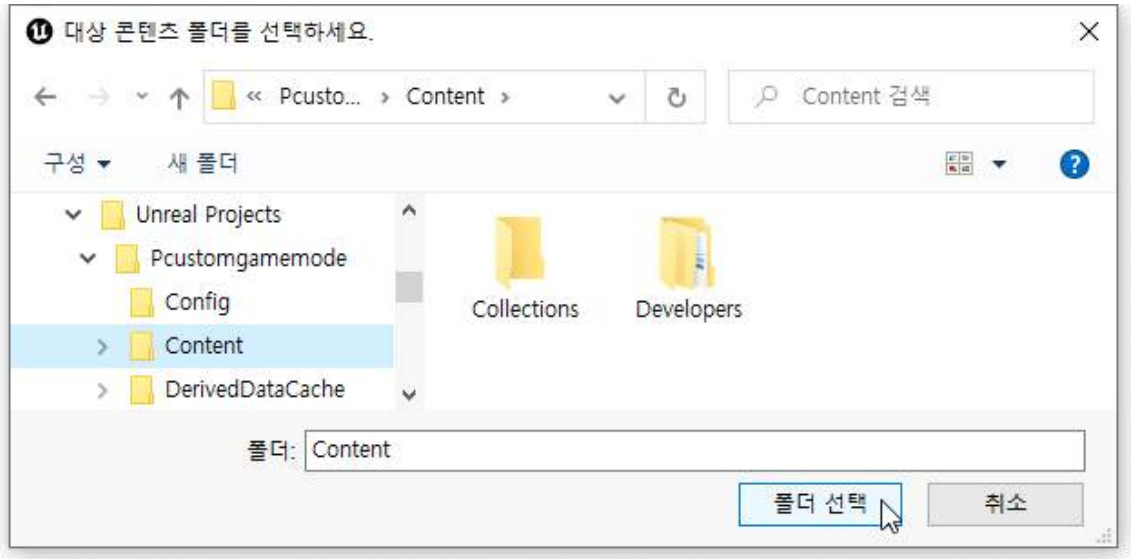
<참고> 이전 프로젝트인 Pcustompawn 프로젝트의 콘텐츠 브라우저에서 새 프로젝트인 Pcustomgamemode 프로젝트의 콘텐츠 브라우저로 애셋을 마우스 드래그로 복사하는 방법은 동작하지 않는다.

<참고> 우리가 만든 **BP_MyCharacter** 블루프린트 클래스 애셋은 이전 프로젝트인 Pcustompawn 프로젝트 폴더에서 Content 폴더 아래에 BP_MyCharacter.uasset 파일 이름으로 존재한다. 이 애셋 파일을 우리의 새 프로젝트인 Pcustomgamemode 프로젝트 폴더의 Content 폴더 아래로 복사하기 위해서 윈도우 파일 탐색기에서 복사(**Ctrl+C**)와 붙여넣기(**Ctrl+V**)를 해서 복사해도 된다. 그러나 이 방법은 프로젝트의 버전이 다르거나 또는 다른 이유로 정상적으로 동작하지 않을 수도 있으므로 사용을 권장하지 않는다.

3. 다음으로, **에셋 리포트** 창이 뜬다. 이 창에서는 이주할 애셋 목록을 확인한다. 여러 애셋을 이주하는 경우에는 체크 박스를 사용해서 일부를 이주 목록에서 제외할 수도 있다. **확인** 버튼을 클릭하자.



4. 다음으로, 이주할 목적지 폴더를 선택하기 위한 창이 뜬다.



이 창에서 우리는 새 프로젝트 폴더인 `Pcustomgamemode` 폴더 아래의 `Content` 폴더를 선택하자. 그리고 **폴더 선택** 버튼을 클릭하자. **BP_MyCharacter** 애셋이 새 프로젝트 `Pcustomgamemode`의 `Content` 폴더 아래에 복사된다.

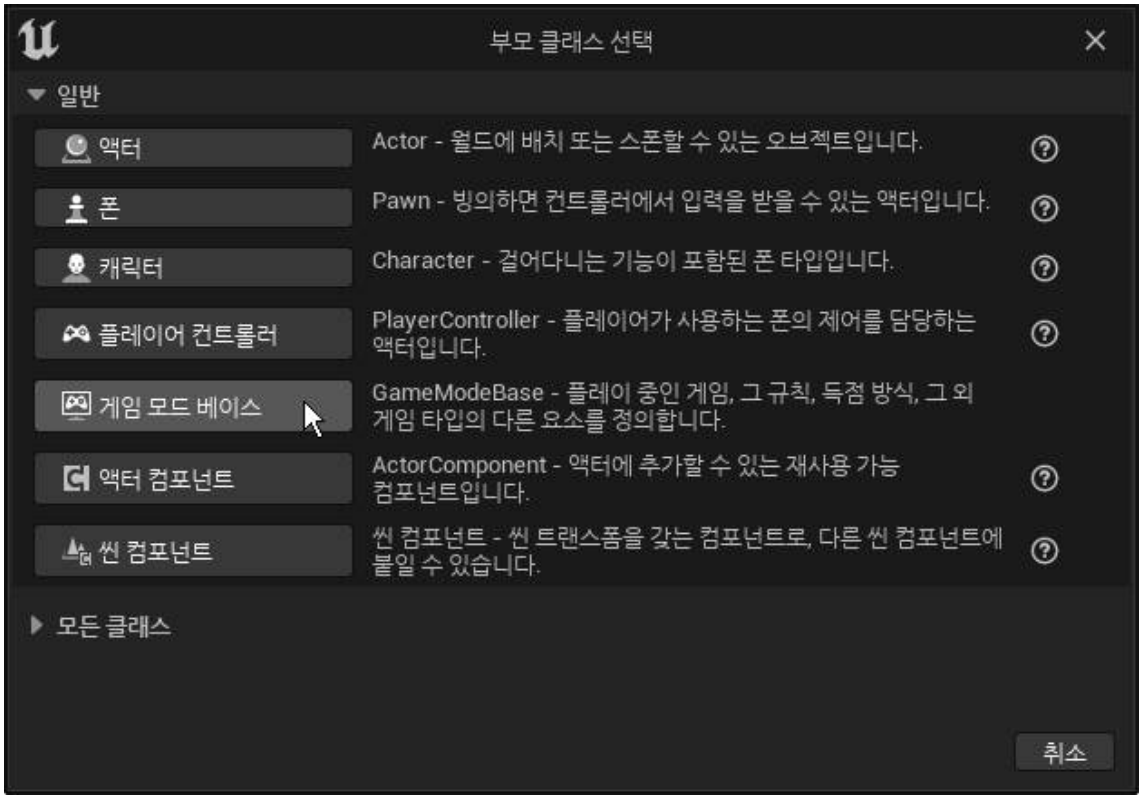
이제, 이전 프로젝트인 `Pcustompawn` 프로젝트 창은 종료하자.

지금까지 한 프로젝트에 있는 애셋을 다른 프로젝트로 복사하는 이주에 대해서 알아보았다. 매우 유용한 기능이므로 잘 익혀두자.

<참고> 애셋 이주에 대한 자세한 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/Basics/AssetsAndPackages/Migrate/

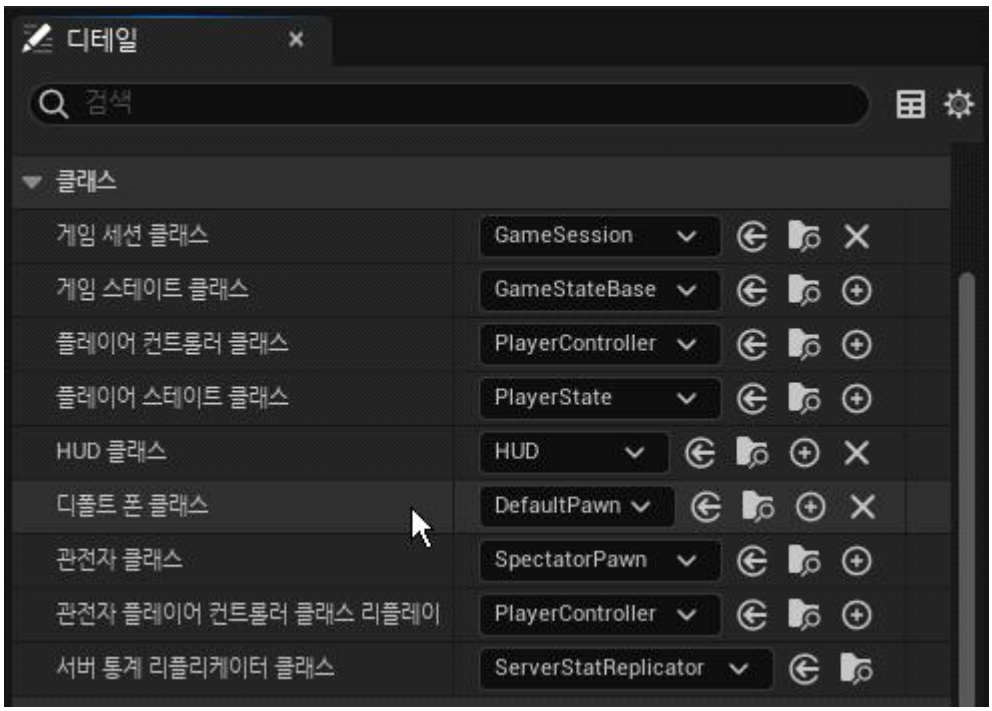
5. 이제부터, 커스텀 게임 모드를 생성해보자.

콘텐츠 브라우저에서 저장할 위치인 **콘텐츠**를 선택하고, 툴바의 **+추가** 버튼을 클릭하자. 그다음, 드롭다운 메뉴에서 **블루프린트 클래스**를 선택하자. 그다음, **부모 클래스 선택** 창에서 **게임 모드 베이스** (**GameModeBase**)를 클릭하자.



생성된 블루프린트 클래스의 이름을 **BP_MyGameMode**로 지정하자.

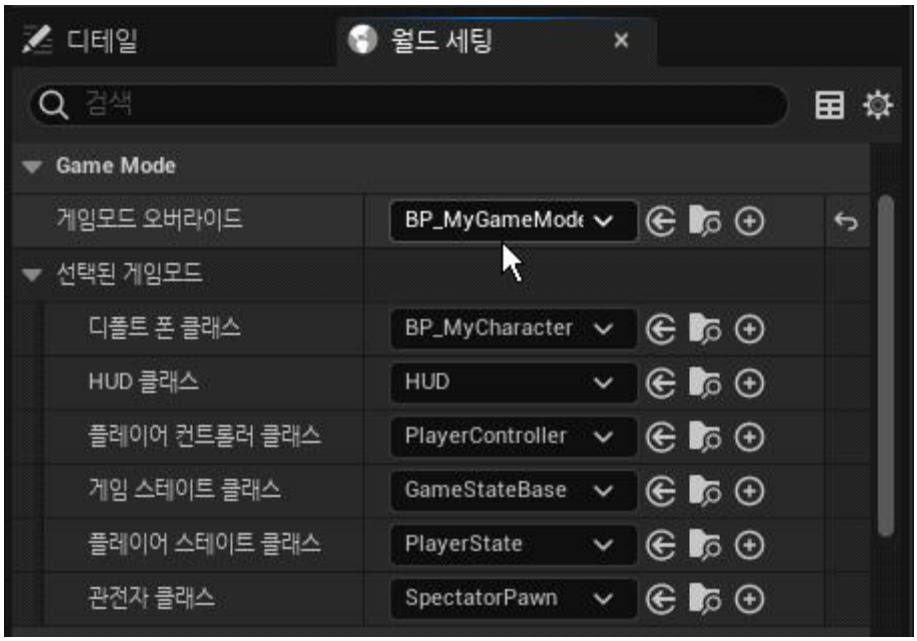
6. 이제, **BP_MyGameMode**를 더블클릭하여 블루프린트 에디터를 열자. 오른쪽 **디테일** 탭에서 **클래스** 영역을 살펴보자. 게임 플레이 시에 사용할 각종 클래스들을 명시하고 있다.



속성 중에서 **디폴트 폰 클래스**(**Default Pawn Class**)가 있다. 이 속성의 디폴트 값으로 **DefaultPawn**으로 되어 있다. 클릭하면 선택 가능한 여러 폰 클래스가 나열될 것이다. 그 중에서 우리의 커스텀 폰

클래스인 **BP_MyCharacter**를 선택하자.
컴파일하고 저장하자.

7. 이제, 레벨 에디터에서 **디테일** 탭 옆에 있을 **월드 세팅** 탭으로 가자. 만약 탭이 보이지 않는다면 메뉴바에서 **창** » **월드 세팅**을 선택하면 된다. **월드 세팅** 창은 현재 레벨의 전역적 속성을 설정하는 창이다. **Game Mode** 영역 아래에 **게임모드 오버라이드** 속성이 있을 것이다. 속성값이 **None**으로 되어 있을 것이다. 이것은 게임 모드를 재정의하지 말고 프로젝트 디폴트 값으로 명시된 게임 모드를 사용하라는 의미이다.
우리는 **None**을 클릭하고 드롭다운 메뉴에서 **BP_MyGameMode**를 선택하자. 이제 레벨의 게임 모드로 우리의 커스텀 게임 모드인 **BP_MyGameMode**가 사용된다.
바로 아래의 **선택된 게임모드** 속성을 펼쳐보자. **디폴트 폰 클래스**가 **BP_MyCharacter**로 되어 있음을 알 수 있다.

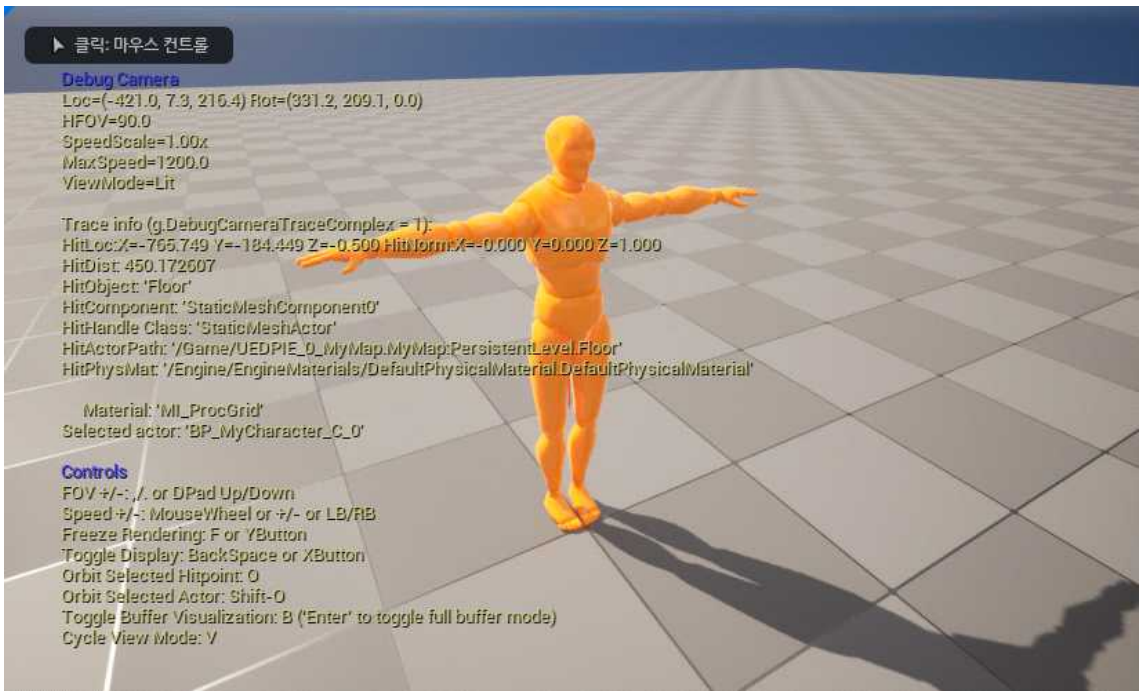


우리는 지금까지, 게임 모드 블루프린트 클래스를 만들고 이를 레벨의 게임 모드로 명시하는 과정을 진행하였다. 이제 커스텀 게임 모드를 레벨에 지정하는 과정이 모두 완료되었다.
한편, 월드 세팅 탭에서 **선택된 게임모드** 속성을 펼치면 그 아래에 여러 속성이 나열되는데 모두 읽기 전용이 아니라 수정 가능하도록 허용되어 있다. 만약 **선택된 GameMode** 아래의 나열된 속성값을 여기에서 수정한다면 **BP_MyGameMode**의 블루프린트 에디터를 열고 그 에디터에서 속성값을 수정하는 것과 동일하게 처리해준다. 즉, 여기에서 수정해도 **BP_MyGameMode**가 수정된다는 의미이다.

<참고> 우리는 우리의 커스텀 게임 모드를 **월드 세팅**에 있는 **게임모드 오버라이드** 속성에 지정하였다. 이는 현재의 레벨이 실행될 때 사용될 게임 모드를 지정한 것이다. 만약 우리의 커스텀 게임 모드를 현재 프로젝트에서의 모든 레벨에서 디폴트로 사용하도록 지정하려면 **프로젝트 세팅** 창에서 **프로젝트** » **맵 & 모드** 탭에 있는 **기본 게임모드** 속성에 지정해두면 된다.

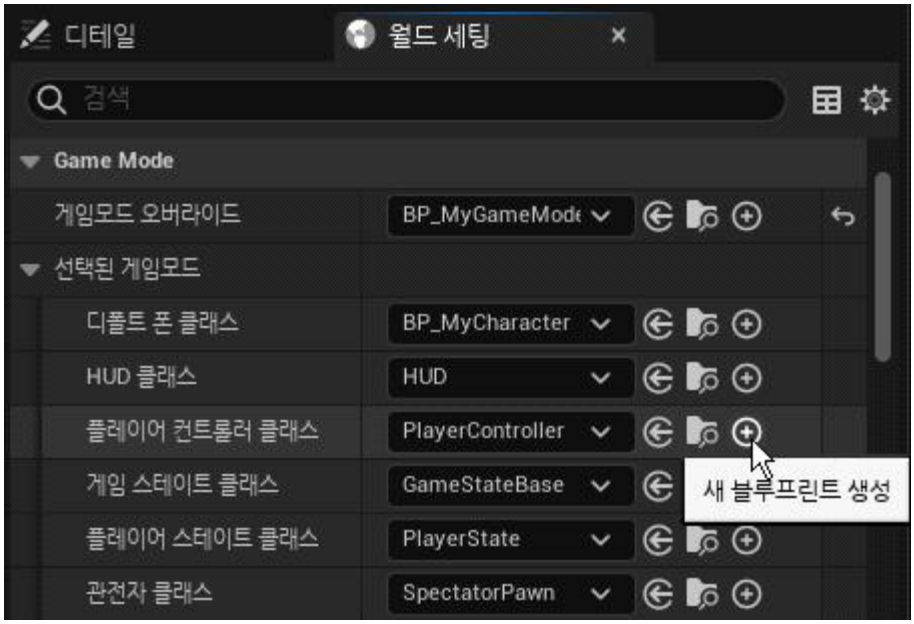
8. 이제, 플레이해보자. 디폴트 씬 외에는 아무것도 보이지 않을 것이다. 또한 입력 제어도 전혀 작동하지 않을 것이다. 그 이유는 우리가 만든 커스텀 플레이어 폰 클래스인 **BP_MyCharacter**에서는 아직 아무것도 구현하지 않았기 때문이다. 단지 메시만 바꾸었을 뿐이다.

9. 우리가 바꾼 메시가 실제로 적용되었는지 확인해보자. 플레이한 후에 디버그 카메라 모드로 진입해서 디버그 카메라를 움직여보자.



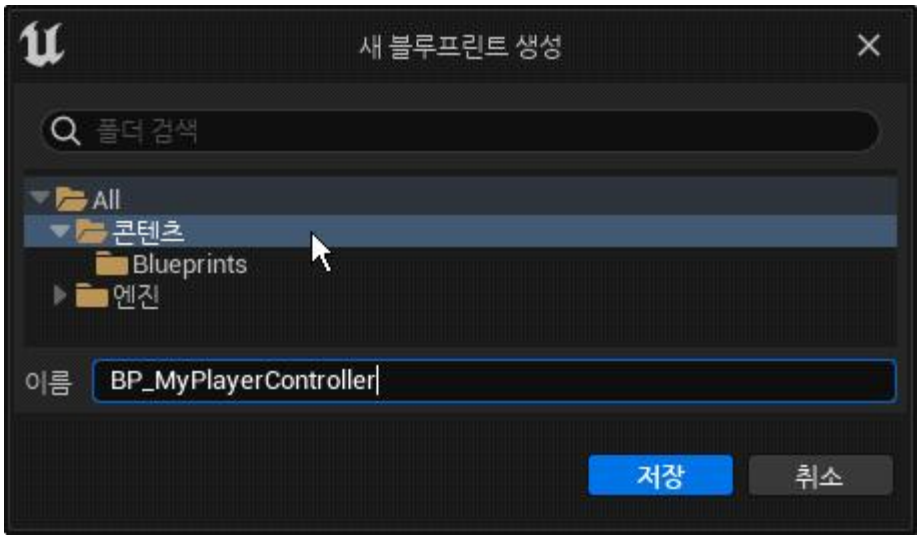
우리가 지정해둔 메시가 있는 것을 확인할 수 있다.

10. 다음으로, **플레이어 컨트롤러**도 커스텀 클래스로 생성해보자.
이전의 **BP_MyGameMode**의 경우와 같이, **콘텐츠 브라우저**에서 툴바의 **+추가** 버튼을 클릭하고, 생성할 애셋 유형으로 **블루프린트 클래스**를 선택한 후에, **부모 클래스 선택** 창에서 **PlayerController**를 클릭하여 생성해도 된다.
그러나 이번에는 다른 방법으로 생성해보자.
블루프린트 클래스를 만들고 이를 속성값으로 지정하는 과정을 한꺼번에 할 수 있는 방법이 있다.
월드 세팅 탭에서 **선택된 게임모드** 아래의 나열된 속성값을 보자.

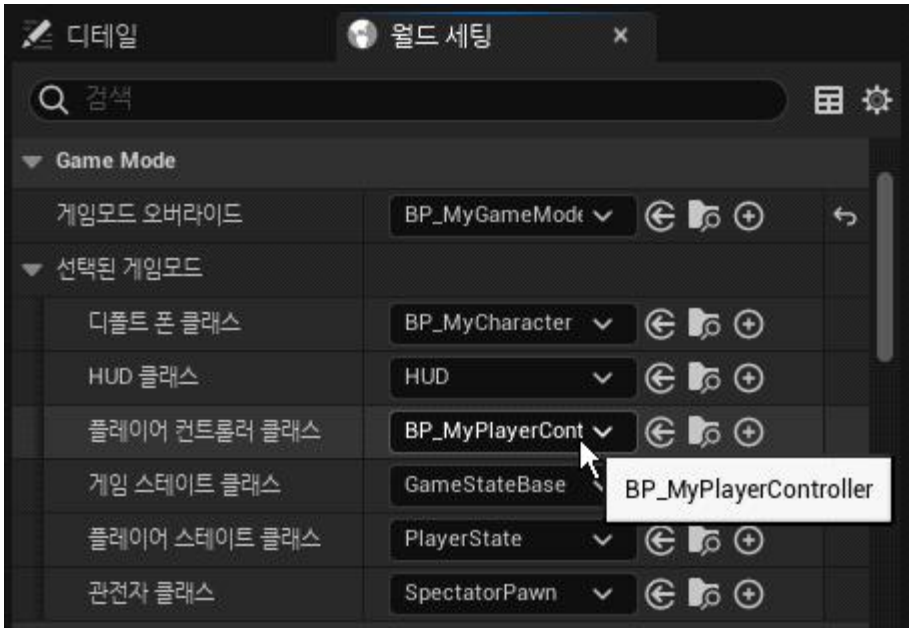


플레이어 컨트롤러 클래스 속성의 디폴트 속성값이 **PlayerController**로 되어 있다. 가장 오른쪽에 더하기 아이콘이 있다. 이를 클릭하면 **PlayerController** 클래스를 상속하여 새 플레이어 컨트롤러 클래스를 생성하고 생성된 클래스를 **플레이어 컨트롤러 클래스**의 속성값으로 지정한다.
이러한 더하기 아이콘은 **월드 세팅** 탭에서뿐만 아니라 다른 여러 창에서의 속성값 지정 부분에서 제공된다. 새 블루프린트 클래스를 쉽게 만들 수 있도록 하는 편리한 기능이므로 알아두도록 하자.

11. 더하기 아이콘을 클릭하면 **새 블루프린트 생성** 창이 뜬다. 디폴트로 `Blueprints` 하위 폴더 아래에 저장되도록 저장 위치를 보여준다. 이것이 일반적인 방법이지만 우리는 연습 과정이므로 상위의 **콘텐츠** 폴더에 저장하자. 생성될 블루프린트 클래스 이름은 **BP_MyPlayerController**로 지정하자.



12. **저장** 버튼을 클릭하면 **PlayerController** 클래스를 상속한 **BP_MyPlayerController** 블루프린트 클래스가 **콘텐츠** 폴더 아래에 생성될 것이다. 또한 **플레이어 컨트롤러 클래스** 속성을 확인해보면 **BP_MyPlayerController**가 지정되었을 것이다. 이는 **BP_MyGameMode**도 함께 수정되었음을 의미한다.



우리는 아직 **BP_MyPlayerController**를 전혀 수정하지 않았다. 그러나 **BP_MyPlayerController**는 기존의 디폴트값으로 되어있었던 **PlayerController** 클래스를 상속한 클래스이므로 동작에서의 변동은 없을 것이다.

지금까지 커스텀 게임 모드를 만들고 레벨에 지정하는 방법을 학습하였다.

□