10_ 트리거 사용하기

<제목 차례>

10_ 트리거 사용하기 ····· 1
1. 개요 ····· 2
2. 레벨 블루프린트에서 트리거 사용하기 ····· 3
3. 블루프린트 클래스에서 콜리전 사용하기 ····· 8

인천대학교 컴퓨터공학부 박종승

1. 개요

이 장에서는 트리거에 대해서 학습한다. 트리거는 상호작용 구현의 기본이 되는 기법이다.

충돌과 트리거에 대해서 알아보자.
두 물체가 부딪히는 것을 **충돌**(collision; 콜리전)이라고 한다. 게임에서의 충돌에는 두 물체가 물리적으로 부딪히는 충돌뿐만 아니라 가상의 물체와 부딪히는 충돌도 있다. 가상의 물체를 사용하는 예를 들면, 결승점을 통과하는 자동차를 체크하기 위하여 결승점에 가상의 상자를 배치하고 자동차와 가상의 상자의 충돌을 검사하는 방법이 있다. 또다른 예로, 자동문 가까이 접근하면 이를 인식하도록 구현하기 위하여 자동문 인접 영역에 가상의 상자를 배치하고 사람이 가상의 상자와의 충돌을 검사하는 방법이 있다.
우리는 앞으로 충돌을 두 종류의 세부 부류인 블록과 겹침으로 구분하자. 실체가 있는 물체의 물리적인 충돌을 **블록**(block)이라고 하고, 물리적이지 않고 단지 겹치는 것을 체크하는 용도의 충돌을 **겹침**(overlap, 오버랩)이라고 구분하자. 블록에서는 서로 뚫고 지나가는 것이 불가능하지만, 겹침에서는 물체가 없는 것처럼 관통하여 지나간다.
겹침 활용을 목적으로 하여 만든 볼륨을 **트리거**(trigger)라고 한다. 트리거는 보이는 실체가 없고 물리적인 움직임과도 무관하다. 단지 겹침을 검출하기 위한 볼륨으로만 사용된다. 위의 예에서 결승점에 배치한 가상의 상자나 자동문 영역에 배치한 가상의 상자가 트리거에 해당한다.

2. 레벨 블루프린트에서 트리거 사용하기

이 절에서는 트리거에 대해서 학습한다. 먼저, 레벨 블루프린트에서 트리거를 사용해보자.
트리거는 겹침 활용을 목적으로 만든 레벨에 배치할 수 있는 액터이다. 기본으로 제공하는 트리거의 종류에는 트리거 박스, 트리거 스피어, 트리거 캡슐이 있다.
게임플레이 시에, 겹침이 시작될 때에는 **OnActorBeginOverlap** 이벤트가 발생되고, 진행되는 겹침이 종료될 때에는 **OnActorEndOverlap** 이벤트가 발생된다.
상호작용 구현을 위해서는 먼저 레벨이 트리거를 배치한다. 그리고, 배치된 트리거 인스턴스에 **OnActorBeginOverlap** 이벤트나 **OnActorEndOverlap** 이벤트를 추가하면 된다. 그다음, 레벨 블루프린트에서 해당 이벤트 그래프를 작성하면 된다.
이제부터 트리거에 대해서 학습해보자.

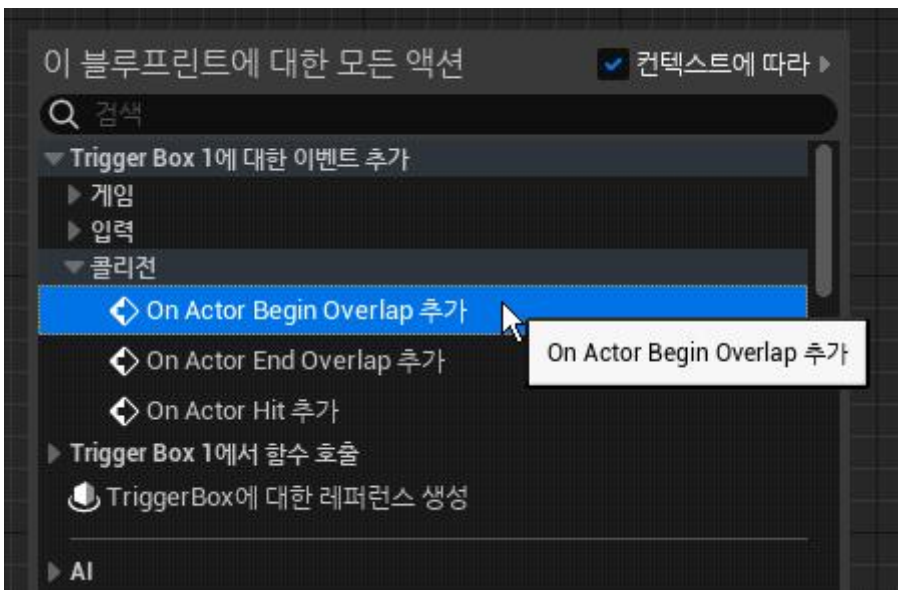
---

1. 새 프로젝트 **Pboxtrigger**를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Pboxtrigger로 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. **액터 배치** 탭을 열자.
기본 탭에 있는 **플레이어 스타트** 액터를 드래그하여 배치하자. 위치를 (-100,0,92)로 지정하자.

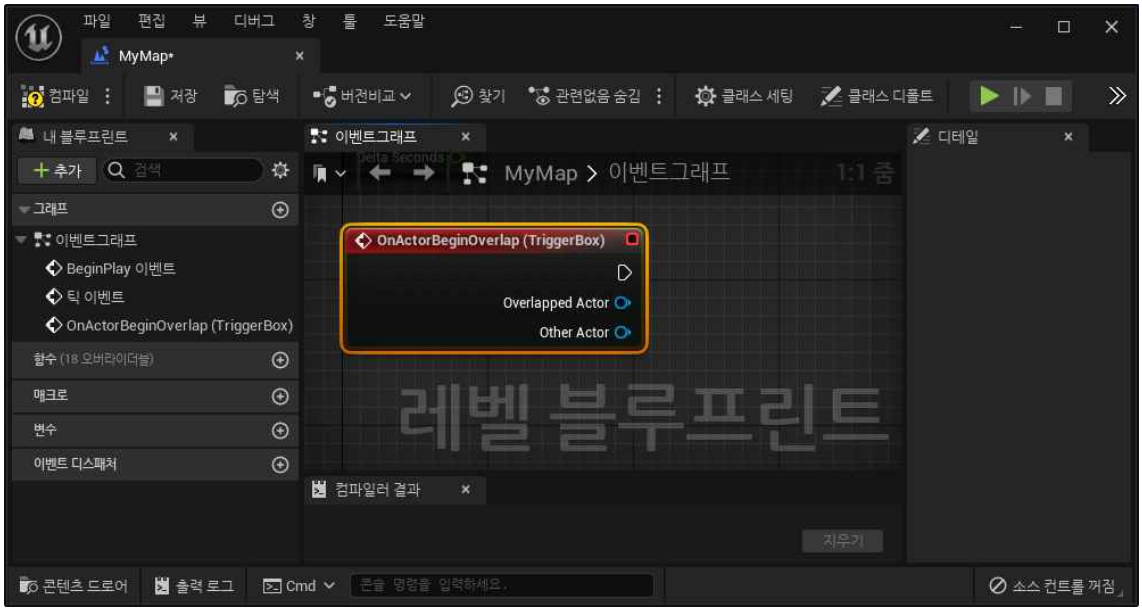
3. **기본** 탭에 있는 **트리거 박스**(TriggerBox)를 레벨에 배치하자.
트리거 박스는 겹침 이벤트를 생성할 목적으로 배치되는 80x80x80 크기의 박스 모양의 볼륨으로, 액터를 상속한 클래스이다. 크기와 위치를 적절히 조절해도 되지만 우리는 크기는 그대로 두고 위치는 (0,0,40)에 배치하자. 디폴트로 **TriggerBox**라는 이름으로 배치된다.

4. 뷰포트나 **아웃라이너**에서 배치된 **트리거 박스**인 **TriggerBox**를 선택하자.
그리고 툴바의 **블루프린트** 아이콘을 클릭하고 **레벨 블루프린트 열기**를 클릭하여 레벨 블루프린트 에디터를 열자.
이벤트 그래프 탭의 격자판의 빈 곳에서 우클릭하자. 액션선택 창의 위쪽에서 **TriggerBox1에 대한 이벤트 추가** 아래에 **콜리전** 영역에서 **OnActorBeginOverlap 추가**를 선택하자. 검색하면 더 빠르게 찾을 수 있다.



OnActorBeginOverlap 노드는 트리거 액터가 다른 액터와 오버랩되기 시작할 때에 호출되는 이벤트 노드이다.

5. **레벨 블루프린트**에 **OnActorBeginOverlap (TriggerBox)** 이벤트 노드가 추가되었을 것이다.

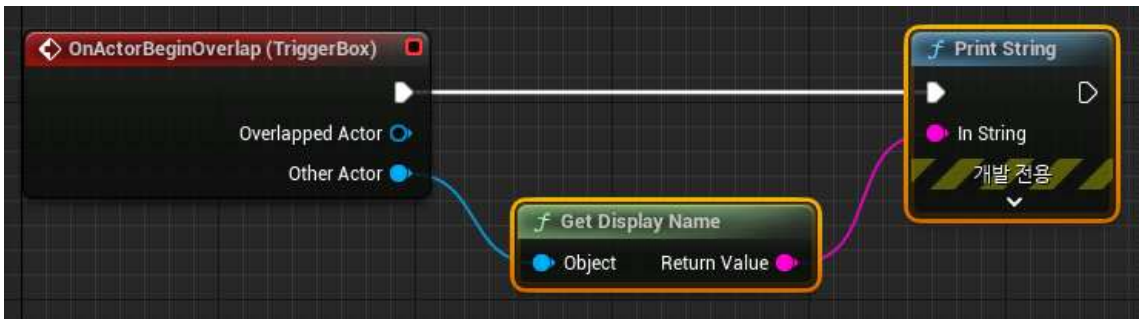


트리거는 수동적으로 배치되어 있는 볼륨이다. 플레이어나 다른 움직이는 물체가 다가와서 트리거와 콜리전을 일으킬 것이다.

OnActorBeginOverlap (TriggerBox) 이벤트 노드의 출력핀인 **OverlappedActor**는 콜리전된 액터인 **TriggerBox** 자신이 된다. **OtherActor**는 콜리전을 일으킨 액터인 플레이어 또는 와서 부딪힌 물체가 될 것이다.

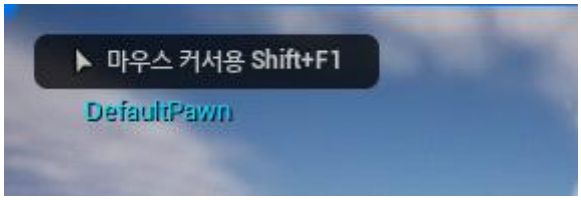
6. 콜리전을 일으킨 액터인 **OtherActor**의 이름을 출력해보자.

먼저, 격자판의 빈 곳에서 **PrintString** 노드를 배치하자. 그다음, **OnActorBeginOverlap (TriggerBox)** 이벤트 노드와 실행핀을 연결하자. 그다음, **OnActorBeginOverlap (TriggerBox)** 노드의 **OtherActor** 출력핀을 **PrintString** 노드의 **InString** 입력핀에 연결하자. 이 때에, **OtherActor** 출력핀의 타입은 액터 레퍼런스 타입인데 반해 **InString** 입력핀은 **String** 타입이어서 타입 불일치가 일어난다. 이런 경우에 에디터에서는 적절히 타입 변환이 되도록 노드를 삽입해준다. 다만 항상 가능한 것은 아니므로 주의해야 한다.

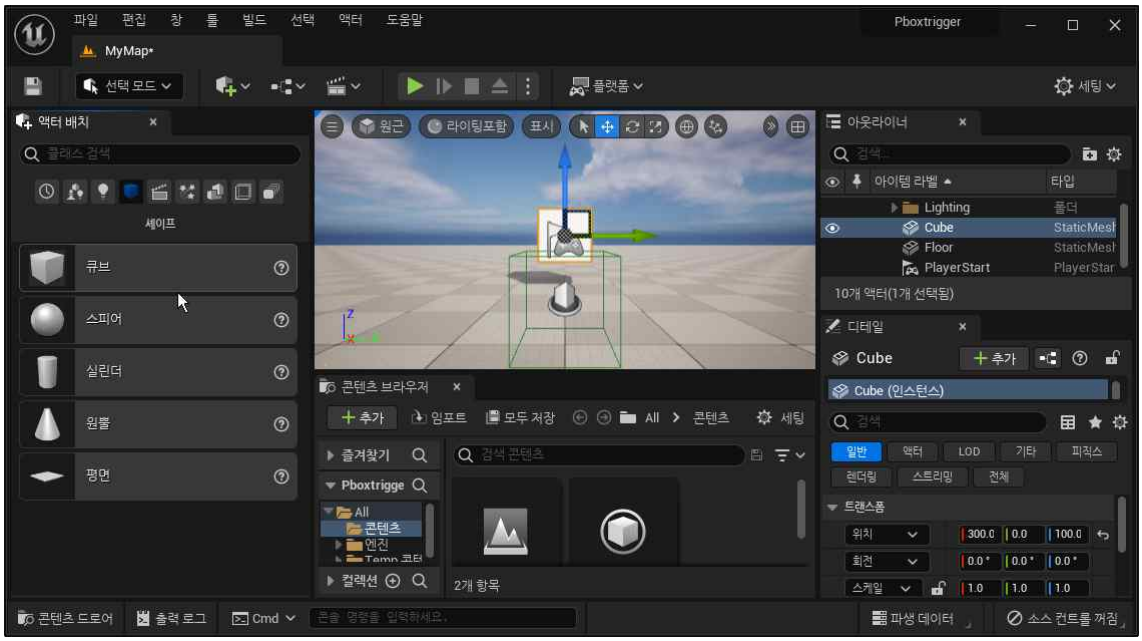


컴파일하고 저장하자.

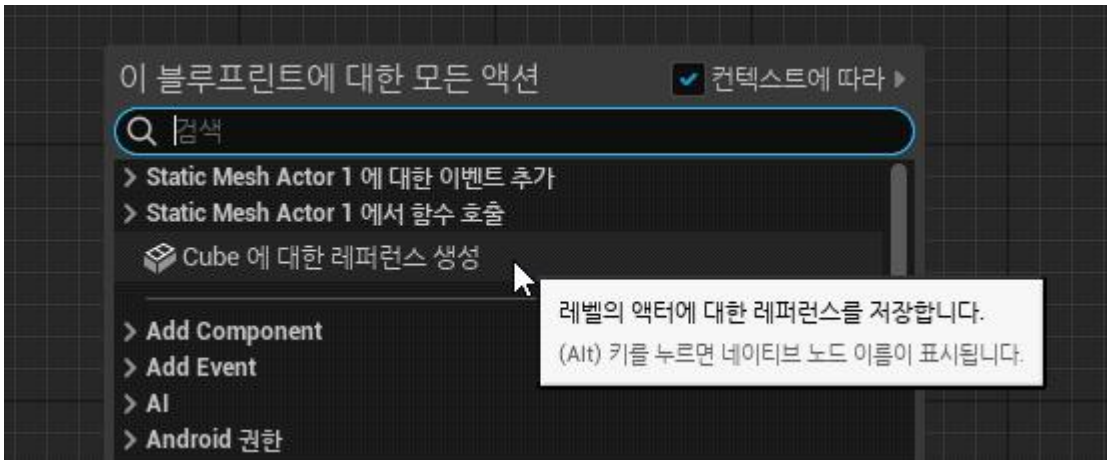
7. 먼저 플레이해보자. 뷰포트에서 이동 키를 사용하여 앞으로 조금만 전진해보자. 트리거 박스와 겹치게 되면 문자열이 출력될 것이다. 콜리전을 일으킨 액터인 **DefaultPawn**이 출력된다.



8. 이제, 시각적으로 변화가 생기도록 만들어보자.
먼저, **액터 배치** 탭에서 **세이프** 탭을 클릭하고 **큐브**를 드래그해서 레벨에 배치하자. 배치된 큐브 액터의 이름을 **Cube**라고 하자. **Cube**의 **트랜스폼** 속성에서 **위치**는 (300,0,100)으로 지정하자.



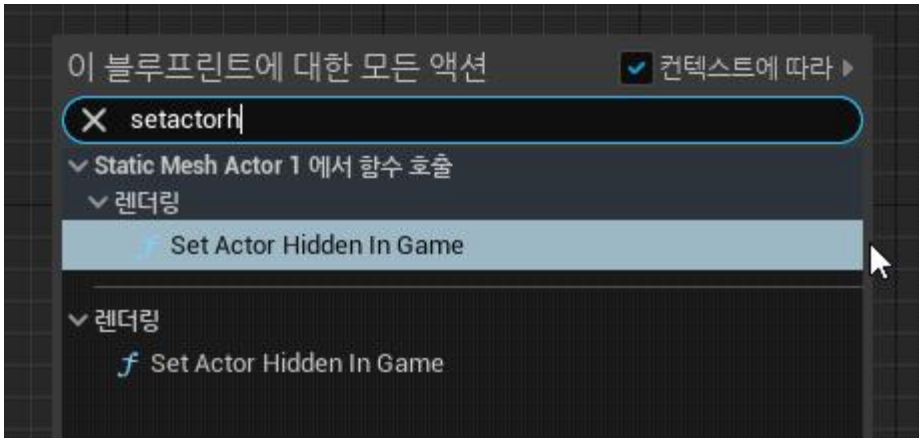
9. 배치된 **Cube** 액터가 선택된 상태에서 레벨 블루프린트 에디터로 가자. 격자판의 빈 곳에서 우클릭하면 액션선택 창이 뜬다. 여기서 **Cube에 대한 레퍼런스 생성**을 선택하자. 레벨에 배치된 **Cube** 액터의 레퍼런스 노드가 배치될 것이다.



또다른 방법으로, **아웃라이너**에서 배치된 **Cube** 액터를 드래그해서 격자판에 드롭해도 **Cube** 액터의 레퍼런스 노드가 배치된다.

10. 이제부터, 겹침이 발생되면 배치된 **Cube** 액터가 사라지게 만들어보자.
배치된 **Cube** 레퍼런스 노드의 출력핀을 드래그하고 액션선택 창에서 검색하여 **SetActorHiddenInGame** 함수 노드를 배치하자.
한편, 이보다 더 편리한 배치 방법이 있다. 이에 대해서 알아보자.
우리는 먼저 **Cube** 액터의 레퍼런스 노드를 배치한 후에 **SetActorHiddenInGame** 함수 노드를 배치하였

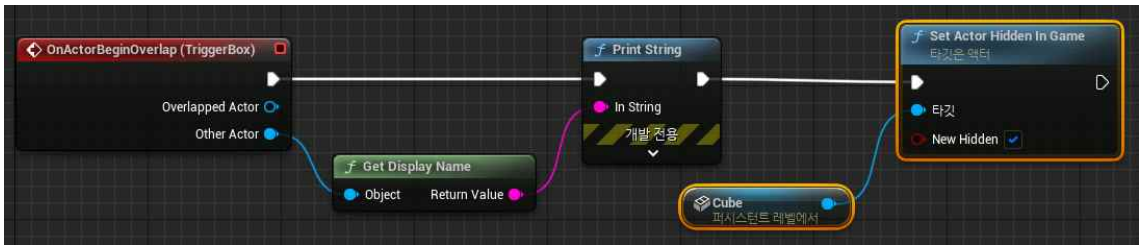
다. 이 두 과정을 한 번에 할 수가 있다. **Cube** 액터의 레퍼런스 노드를 배치하기 전에 격자판의 빈 곳에서 우클릭하여 액션선택 창이 뜨면 바로 **SetActorHiddenInGame** 함수를 검색해보자.



검색 결과를 보면 아래에는 **렌더링** 카테고리 아래의 **SetActorHiddenInGame** 함수가 나열되어 있다. 이것을 선택하면 함수 노드가 하나 배치된다. 한편, 목록의 위에는 **Static Mesh Actor 1에서 함수 호출** 카테고리 아래에 같은 함수가 나열된다. 여기서 **Static Mesh Actor 1**은 현재 레벨에서 선택된 액터인 **Cube** 액터의 인스턴스 ID이다. 따라서 이것은 **Cube** 액터 인스턴스에 대한 함수를 호출하겠다는 의미이다. 이 함수를 선택하면 **Cube** 레퍼런스 노드와 **SetActorHiddenInGame** 함수 노드가 함께 연결된 채로 생성된다.

<참고> 우리의 예제에서의 **Cube** 액터의 인스턴스 ID는 **Static Mesh Actor 1**이다. 그러나 어떤 경우에는 **Static Mesh Actor 2**가 될 수도 있고 **Static Mesh Actor 3**가 될 수도 있다. 이는 엔진 내부에서의 처리와 관련된 것이므로 순차적이지 않더라도 무시하자.

11. 이제 **SetActorHiddenInGame** 함수 노드의 **NewHidden** 입력핀을 체크하자. 노드가 실행되면 액터가 렌더링되지 않도록 숨길 것이다.
그다음, 이전의 **PrintString** 노드의 출력 실행핀을 **SetActorHiddenInGame** 함수 노드의 입력 실행핀에 연결하자. 전체적으로 다음과 같은 모습의 그래프가 될 것이다.

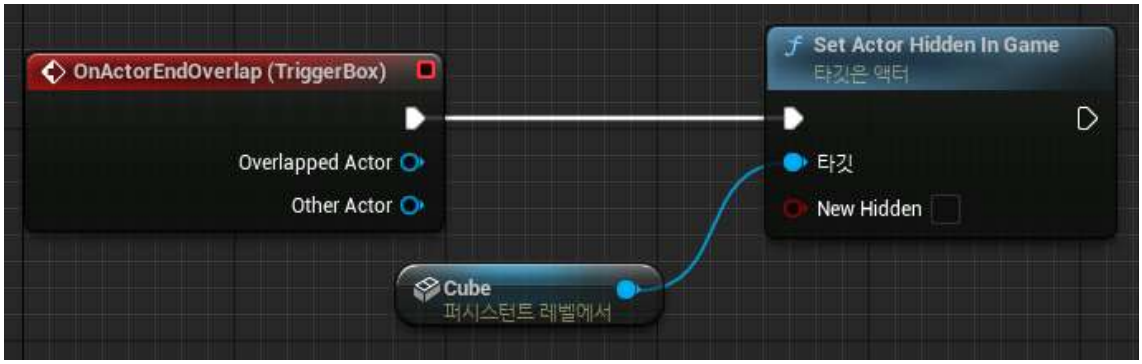


컴파일하고 저장하자.

12. 이제, 레벨을 플레이해보자. 앞으로 다가가보자. 겹침이 발생하면 **Cube** 액터가 사라질 것이다.

13. 이제, 겹침끝 이벤트에 대해서도 추가해보자.
아웃라이너 탭에서 **TriggerBox**를 선택하자. 그 다음, 레벨 블루프린트 에디터의 이벤트 그래프 탭의 격자판에서 빈 곳에서 우클릭하자. 액션선택 창의 위쪽에서 **TriggerBox1에 대한 이벤트 추가** 아래에 **콜리전** 영역에서 **OnActorEndOverlap 추가**를 선택하자. **OnActorEndOverlap (TriggerBox)** 이벤트 노드가 추가될 것이다.

14. 그다음, 다시 **SetActorHiddenInGame** 함수 노드를 추가하자. 이전에서의 방법을 반복해도 되지만 이미 배치되어 있는 **Cube 레퍼런스 노드**와 **SetActorHiddenInGame** 함수 노드를 모두 선택하고 **Ctrl+C** 키와 **Ctrl+V** 키를 눌러 복사하자. 이를 **OnActorEndOverlap (TriggerBox)** 이벤트 노드의 실행핀에 연결시키자. **NewHidden** 입력핀에는 체크 해제하여 다시 보이도록 하자. 전체적으로 다음과 같은 모습의 그래프가 될 것이다.



컴파일하고 저장하자.

15. 이제, 레벨을 플레이해보자. 앞으로 다가가서 겹침이 시작되면 **Cube** 액터가 사라질 것이다. 다시 뒤로 물러나서 겹침이 끝나면 **Cube** 액터가 다시 보일 것이다.

지금까지, 트리거 박스를 배치하고 트리거 박스에서 겹침시작 이벤트가 발생할 때에 문자열을 출력하는 스크립트를 레벨 블루프린트에서 작성하는 방법을 학습하였다. 또한 겹침시작 이벤트와 겹침끝 이벤트에 대해서 함수가 실행되도록 해 보았다.

3. 블루프린트 클래스에서 콜리전 사용하기

이 절에서는 블루프린트 클래스에서 콜리전 컴포넌트를 사용하는 방법에 대해서 학습한다.

트리거는 겹침 활용을 목적으로 하는 액터를 상속한 클래스로 레벨에 배치할 수 있다. 기본으로 제공되는 트리거는 **트리거 박스**, **트리거 스피어**, **트리거 캡슐**이다. 이들의 정확한 명칭은 각각 **TriggerBox**, **TriggerSphere**, **TriggerCapsule**이다.
트리거에서의 겹침 활용을 위한 기능 구현은 내부에 특정 컴포넌트로 구현되어 있다. 이러한 겹침 활용을 목적으로 하는 컴포넌트를 콜리전 컴포넌트라고 한다. 각 트리거 액터는 충돌 기능을 위한 콜리전 컴포넌트를 가지고 있다. **트리거 박스**, **트리거 스피어**, **트리거 캡슐**은 내부에 각각 **BoxCollision** 컴포턴트, **SphereCollision** 컴포넌트, **CapsuleCollision** 컴포넌트를 가지고 있다. 이 컴포넌트가 겹침 이벤트를 발생시키는 컴포넌트이다. 이들은 콜리전 볼륨의 모양만 다르고 모두 동일한 방식으로 동작한다.

<참고> 우리는 앞으로 트리거라는 용어는 컴포넌트가 아닌 액터일 경우에만 한정하여 사용하기로 하자. 액터의 컴포넌트를 지칭하고자 하는 경우에는 콜리전 컴포넌트로 지칭하자. 한편, 사실상 많은 액터가 내부에 콜리전 컴포넌트를 포함하고 있다. 우리는 겹침 활용을 주 목적으로 하는 액터만 트리거라고 하자.

콜리전 컴포넌트를 사용하면 커스텀 액터를 트리거처럼 활용할 수 있도록 만들 수 있다. 액터에 콜리전 컴포넌트를 추가하여 겹침 활용 기능을 탑재해두면 배치된 레벨 블루프린트에서 겹침 이벤트 그래프를 매번 구현할 필요가 없어진다.
게임플레이 시에, 콜리전 컴포넌트에서 겹침이 시작될 때에는 **OnComponentBeginOverlap** 이벤트가 발생되고, 진행되는 겹침이 종료될 때에는 **OnComponentEndOverlap** 이벤트가 발생된다.
상호작용 가능한 액터를 구현하기 위해서는 먼저 액터에 콜리전 컴포넌트를 추가한다. 그리고, 추가된 콜리전 컴포넌트에 **OnComponentBeginOverlap** 이벤트나 **OnComponentEndOverlap** 이벤트를 추가하면 된다. 그다음, 블루프린트 클래스에서 해당 이벤트 그래프를 작성하면 된다. 액터가 완성된 후에는 액터를 레벨에 배치하기만 하면 겹침 기능이 자체적으로 수행된다.

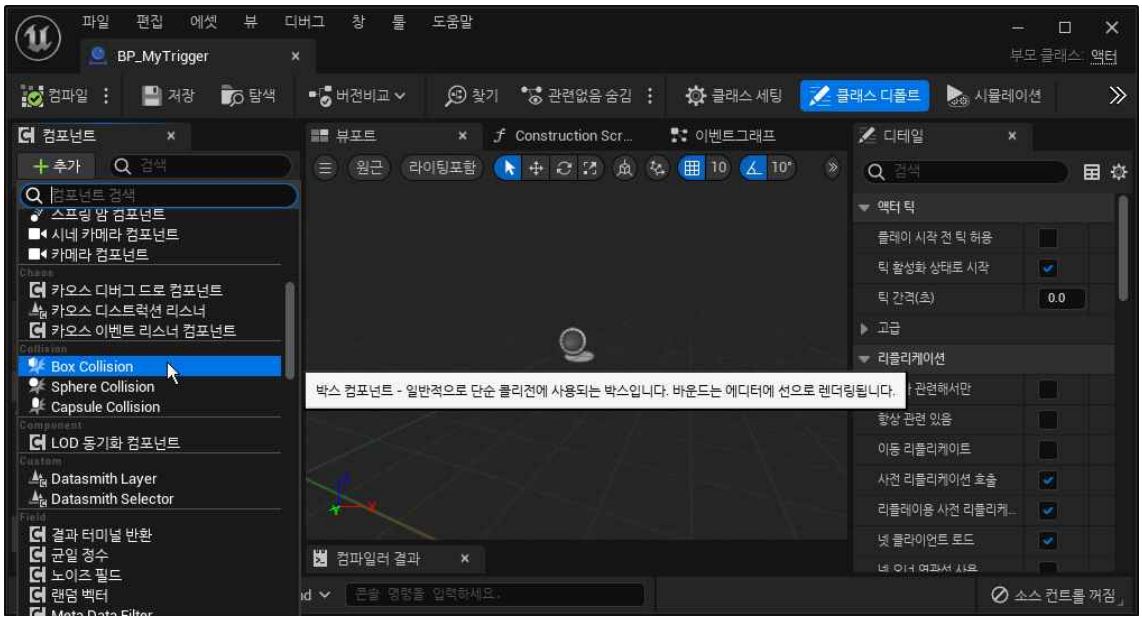
이번 예제에서는, 내부에 **BoxCollision** 컴포턴트를 가지는 커스텀 액터를 직접 만들어보자. 그리고 이 커스텀 액터가 트리거 박스처럼 동작하도록 해보자.
즉 트리거 기능을 갖춘 액터를 블루프린트 클래스로 작성해본다.

---

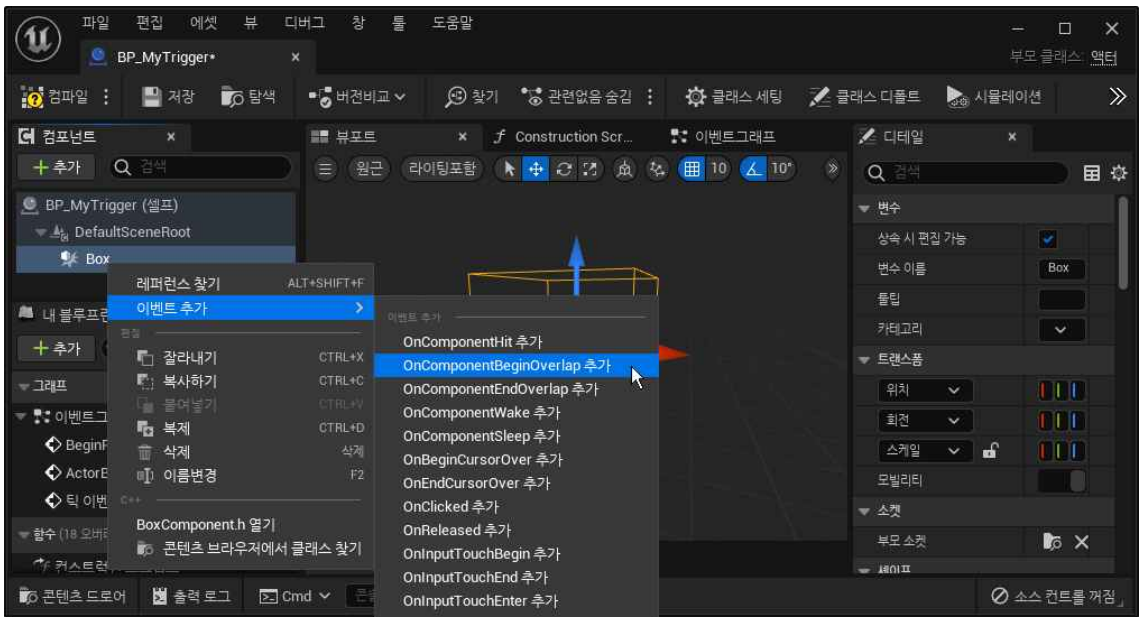
1. 새 프로젝트 **Pboxcollision**를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Pboxcollision로 입력하고 **생성** 버튼을 클릭하자. 프로젝트가 생성되고 언리얼 에디터 창이 뜰 것이다.
창이 뜨면, 메뉴바에서 **파일** » **새 레벨**을 선택하고 **Basic**을 선택하여 기본 템플릿 레벨을 생성하자. 그리고, 레벨 에디터 툴바의 저장 버튼을 클릭하여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

2. **액터 배치** 탭을 열자. **기본** 탭에 있는 **플레이어 스타트** 액터를 드래그하여 배치하자. 위치를 (-100,0,92)로 지정하자.

3. **콘텐츠 브라우저** 탭에서 **+추가**를 클릭하자. 그다음, 드롭다운 메뉴에서 **블루프린트 클래스**를 선택하자. 그다음, **부모 클래스 선택** 창에서 **Actor**를 선택하자.
콘텐츠 폴더 아래에 블루프린트 클래스가 생성될 것이다. 이름을 **BP_MyTrigger**라고 하자.
그다음, **BP_MyTrigger**를 더블클릭하여 블루프린트 에디터를 열자.
그다음, 블루프린트 에디터의 **컴포넌트** 탭에서 **+추가**를 클릭하고 **BoxCollision**을 선택하자.



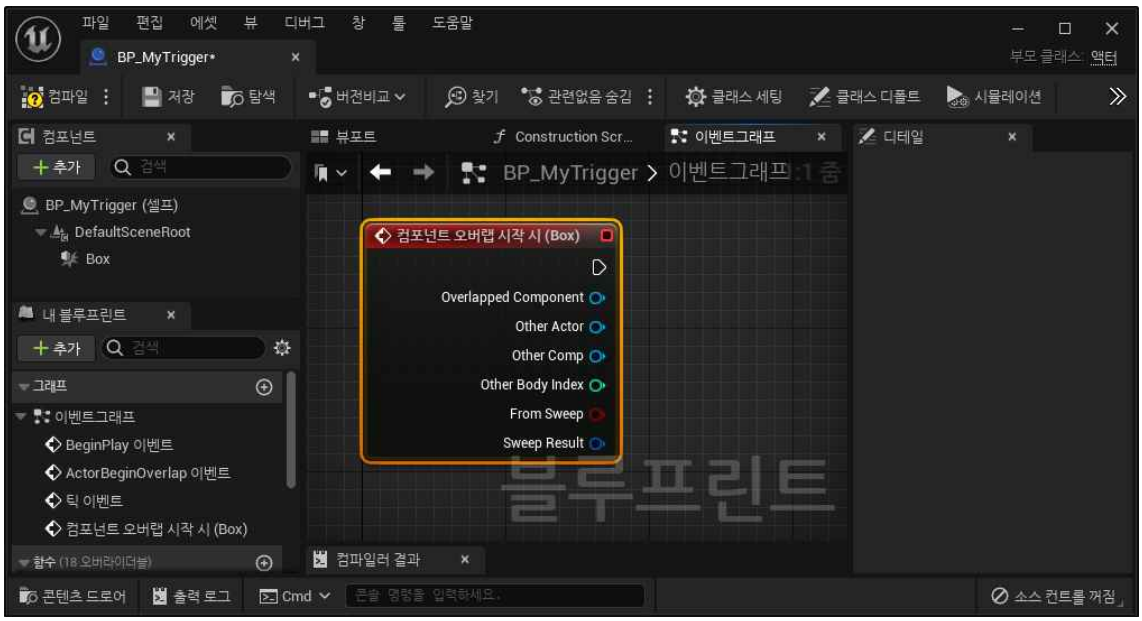
4. **BoxCollision** 컴포넌트가 추가되었다. 기존의 **DefaultSceneRoot** 씬 컴포넌트 아래에 **Box**라는 이름으로 추가되었을 것이다. 이 **Box** 컴포넌트 위에서 우클릭하고 팝업메뉴에서 **이벤트 추가** » **OnComponent BeginOverlap 추가**를 선택하자.



<참고> **OnActorBeginOverlap**과 **OnComponentBeginOverlap**의 차이를 알아보자. 한 액터 내에는 여러 컴포넌트가 있다. 겹침 이벤트가 발생 가능한 컴포넌트에는 하나 이상의 콜리전 볼륨이 포함되어 있다. 한 컴포넌트

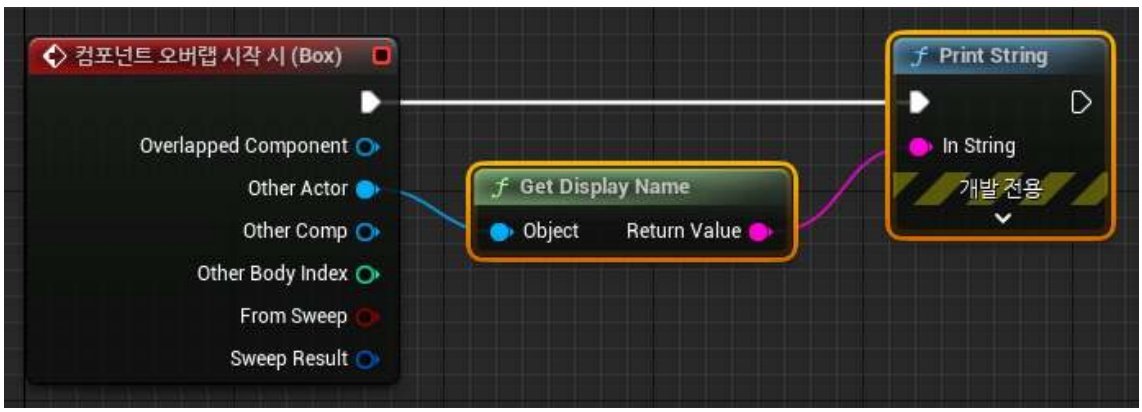
내에 있는 콜리전 볼륨 중의 하나에서 발생한 겹침 이벤트에 대해서는 **OnComponentBeginOverlap** 함수가 호출된다. 한편 액터의 모든 컴포넌트의 모든 콜리전 보륨에서 컵침 이벤트가 발생하면 **OnActorBeginOverlap** 함수가 호출된다.

5. 이벤트 그래프 탭에 **OnComponentBeginOverlap** 노드가 다음과 같이 배치될 것이다.



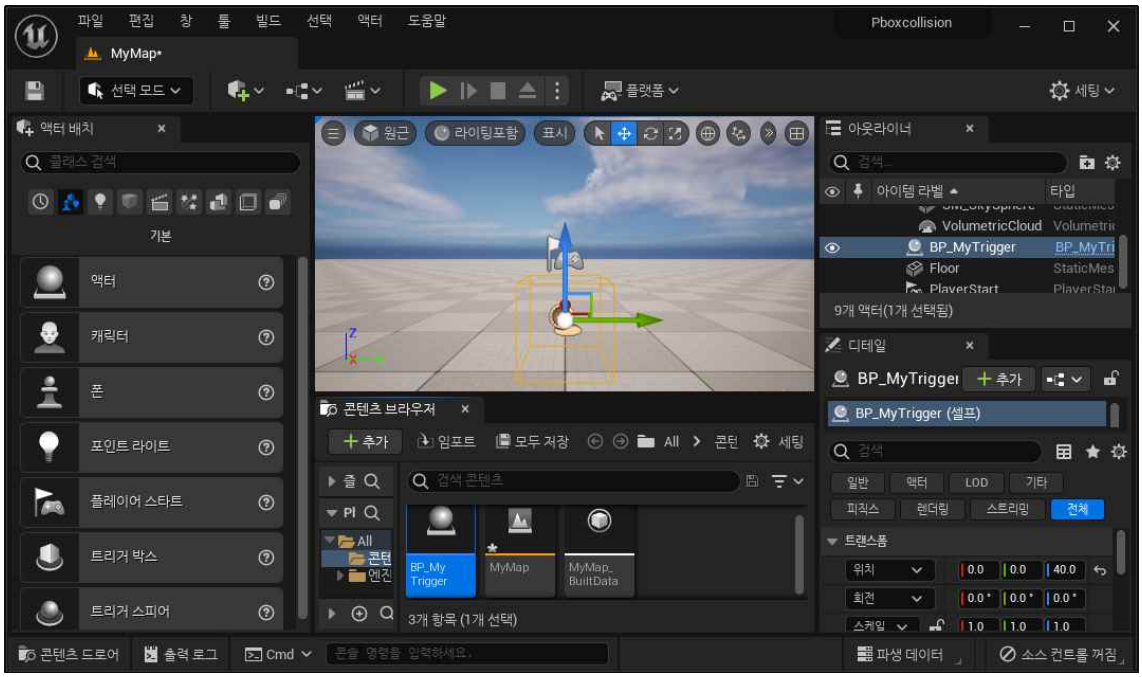
이전 예제에서는 이벤트의 이름이 **OnActorBeginOverlap (TriggerBox)** 이벤트였다. 레벨 블루프린트에서는 컴포넌트 단위의 겹침이 아니라 액터 단위의 겹침을 처리하였기 때문이었다. 또한 호출 대상 액터를 구분하기 위해서 액터 이름인 **TriggerBox**도 괄호로 이벤트명 뒤에 붙여서 표시하였다.
이번 예제에서는 한 액터 내에 있는 컴포넌트에 이벤트를 추가하는 것이다. 따라서 이벤트명에서 **Actor** 단어 대신 **Component**로 표시하였다. 또한 호출 대상이 컴포넌트 자신이므로 컴포넌트 이름인 **Box**가 괄호로 붙여서 **OnComponentBeginOverlap (Box)**로 표시된다.
배치된 **OnComponentBeginOverlap** 노드는 **OnActorBeginOverlap** 노드와는 다르게 많은 출력 노드들이 있다. 이중에서, 이벤트 노드의 출력핀인 **OverlappedComponent**는 콜리전된 컴포넌트인 **Box** 자신이 된다. **OtherActor**는 콜리전을 일으킨 액터인 플레이어 또는 와서 부딪힌 물체가 될 것이다.

6. 콜리전을 일으킨 액터인 **OtherActor**의 이름을 출력해보자. 이전 예제에서와 동일한 방식으로 그래프를 다음과 같이 만들자.

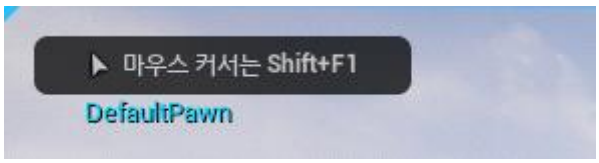


컴파일하고 저장하자.

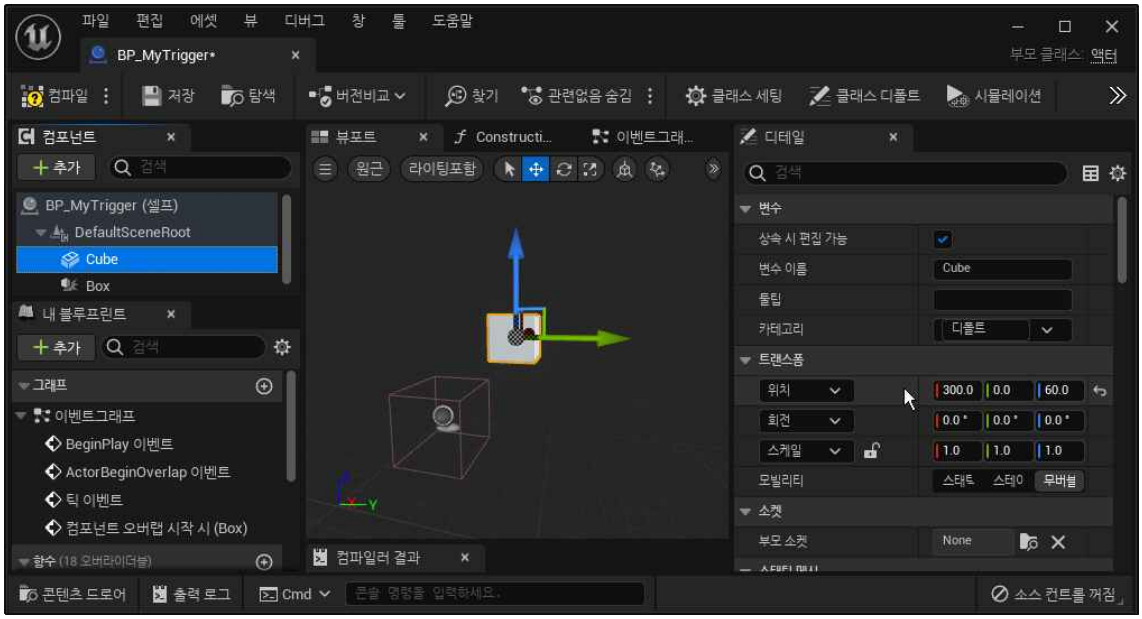
7. 레벨 에디터에서 **BP_MyTrigger**를 레벨에 배치하자.
크기와 위치를 적절히 조절해도 되지만 우리는 크기는 그대로 두고 위치는 (0,0,40)에 배치하자. 디폴트로 **BP_MyTrigger1**의 이름으로 배치되었다.



8. 플레이해보자. 앞으로 다가가 보자. **BP_MyTrigger1**와 겹치게되면 문자열이 출력될 것이다. 이전 예제에서와 동일하게 콜리전을 일으킨 액터인 **DefaultPawn**이 출력된다.



9. 이제, 시각적으로 변화가 생기도록 만들어보자.
BP_MyTrigger 블루프린트 에디터로 가자. **컴포넌트** 탭에서 **+추가**를 클릭하고, **큐브**를 선택하자. 큐브 스태틱 메시 컴포넌트가 추가될 것이다. 컴포넌트 이름이 디폴트로 **Cube**로 하여 추가되었다.
Cube의 **트랜스폼** 속성에서 **위치**는 (300,0,60)으로 지정하자. 이벤트 그래프 탭 대신 뷰포트 탭을 클릭하면 모습을 확인할 수 있다.

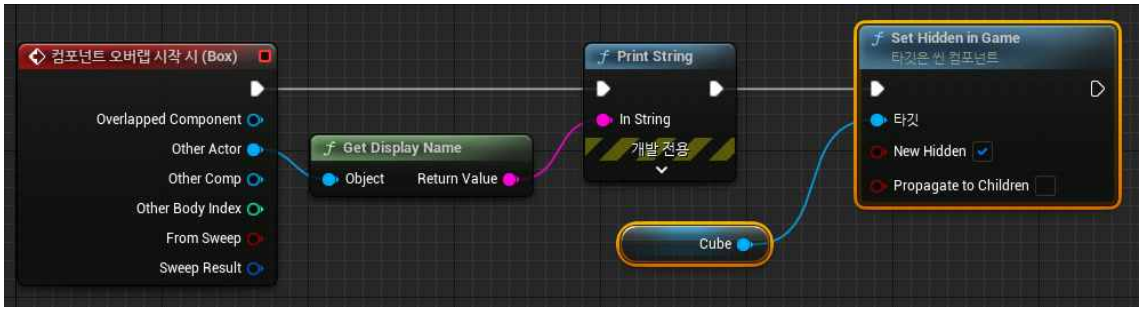


이전 예제에서는 트리거 박스가 (0,0,40)에 있었고 큐브가 (300,0,100)에 있었다. 이번 예제에서도 이와 동일하게 하기 위해서 큐브를 트리거 박스에 상대적인 위치인 (300,0,60)으로 한 것이다.

10. 겹침이 발생되면 배치된 **Cube** 액터가 사라지게 만들어보자.
다시 이벤트 그래프 탭으로 가자.
먼저, 컴포넌트 탭에서 **Cube**를 드래그하여 **이벤트 그래프** 격자판에 드롭하자. **Cube** 컴포넌트의 레퍼런스 노드가 배치된다.
배치된 **Cube** 레퍼런스 노드의 출력핀을 드래그하고 액션선택 창에서 검색하여 **SetHiddenInGame** 함수 노드를 배치하자.
한편, 이보다 더 편리한 배치 방법이 있다. 이에 대해서 알아보자. 격자판의 빈 곳에서 우클릭하여 액션선택 창이 뜨면 **SetHiddenInGame** 함수를 검색해보자.

여러 버전이 나열될 것이다. 그 중에서 Cube 컴포넌트 이름이 뒤에 붙어있는 함수를 선택하자. 이 함수를 선택하면 **Cube** 레퍼런스 노드와 **SetHiddenInGame** 함수 노드가 함께 연결된 채로 생성된다.

11. 이제 **SetHiddenInGame** 함수 노드의 **NewHidden** 입력핀을 체크하자. 노드가 실행되면 컴포넌트가 렌더링되지 않도록 숨길 것이다.
그다음, 이전의 **PrintString** 노드의 출력 실행핀을 연결하자. 전체적으로 다음과 같은 모습의 그래프가 될 것이다.



컴파일하고 저장하자.

12. 이제, 레벨을 플레이해보자. 앞으로 다가가보자. 겹침이 발생하면 **Cube** 액터가 사라질 것이다.

13. 이제, 겹침끝 이벤트에 대해서도 추가해보자.

컴포넌트 탭에서 **Box** 컴포넌트 위에서 우클릭하고 팝업메뉴에서 **이벤트 추가** » **OnComponentEndOverlap 추가**를 선택하자. 이벤트 그래프 탭에 **OnComponentEndOverlap** 노드가 배치될 것이다.

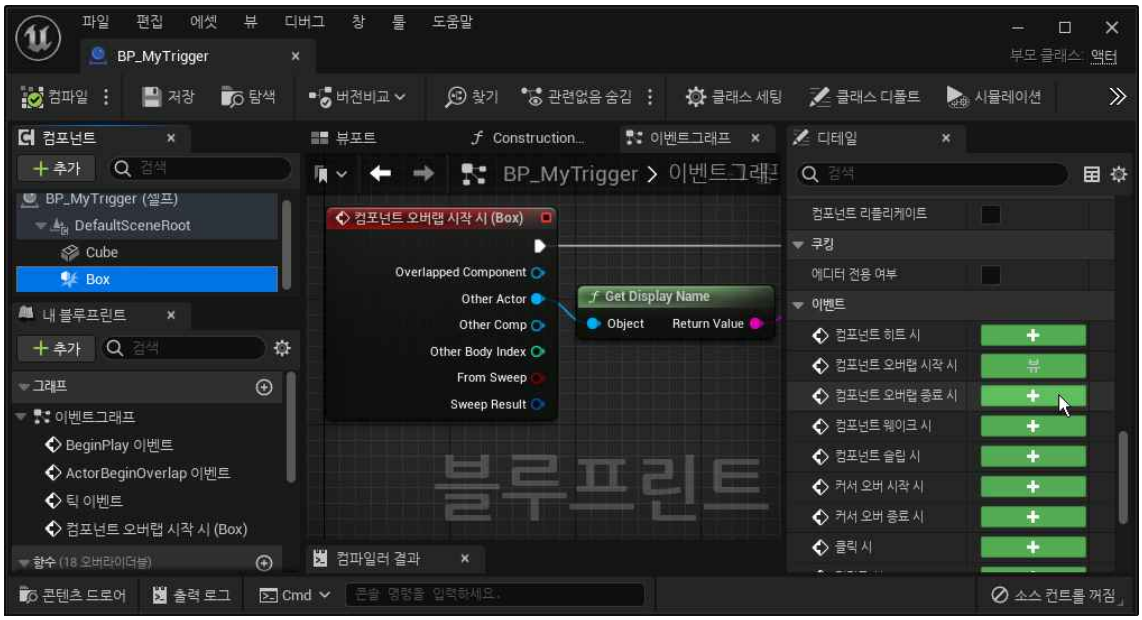
한편, 이벤트 노드를 배치하는 또다른 방법이 있다.

컴포넌트 탭에서 **Box** 컴포넌트를 선택하고 **디테일** 탭을 살펴보자.

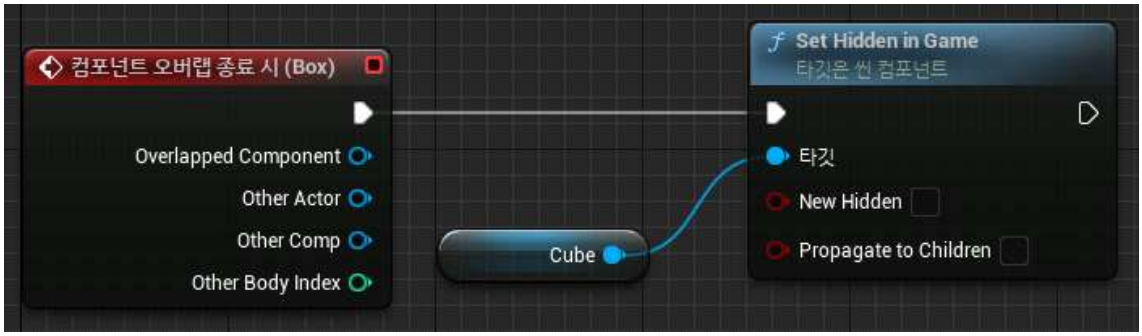
이벤트 영역이 있을 것이다. 이 영역에는 **Box** 컴포넌트에 추가 가능한 모든 이벤트가 나열되어 있다. 이미 추가된 이벤트는 녹색 버튼으로 **뷰**라고 표시된다. 이 버튼을 클릭하면 해당 이벤트 그래프로 이동한다.

아직 추가되어 있지 않은 이벤트는 +라고 표시된다. 이 버튼을 클릭하면 해당 이벤트 노드가 추가되고 그 위치로 이동한다.

우리는 **컴포넌트 오버랩 시작 시(OnComponentEndOverlap)** 이벤트의 + 버튼을 클릭하자.



14. 그다음, 이전 예제에서와 같이 다음과 같은 모습의 그래프를 만들자.



컴파일하고 저장하자.

15. 이제, 레벨을 플레이해보자. 이전 예제에서와 같이 앞으로 다가가서 겹침이 시작되면 **Cube** 컴포넌트가 사라질 것이다. 다시 뒤로 물러나서 겹침이 끝나면 **Cube** 컴포넌트가 다시 보일 것이다.

지금까지, 커스텀 액터를 생성하고 박스 콜리전 컴포넌트를 추가하여 커스텀 액터가 트리거와 같이 동작하도록 만들었다.
이전 예제에서는 레벨 블루프린트에서 이벤트 그래프를 작성하였으나 이번 예제에서는 액터의 블루프린트 클래스 내부에서 이벤트 그래프를 작성하였다.

□