27_ 파티클 시스템

<제목 차례>

인천대학교 컴퓨터공학부 박종승

1. 개요

이 장에서는 파티클 시스템에 대해서 학습한다.

파티클 시스템은 많은 파티클을 만들어서 공간에 방출하는 시각 효과(visual effects, VFX)를 구현하는 시스템이다. 파티클 시스템을 사용하면 먼지나 불꽃이나 번개 등의 다양한 자연 현상이나 폭발이나 마법 사용 등의 멋진 시각 효과를 표현할 수 있다.

파티클 시스템은 대량의 파티클을 공간에 방출한다. 일반적인 콜리전 처리나 물리 엔진을 파티클에 대해서도 동일하게 적용하게 되면 엄청난 계산량의 부하가 생긴다. 이를 해결하기 위하여 파티클의 움직임에 대해서는 별도의 독립된 방식으로 움직임을 제어한다.

파티클 시스템은 엔진마다 구현 형태가 조금씩 다를 수 있지만 전체적으로 주된 방식은 유사하다. 파티클은 머티리얼이 있는 작은 평면 사각형인 스프라이트로 제작된다. 스프라이트는 항상 카메라 방향으로 향하도록 배치된다. 각각의 파티클은 각자가 크기, 속도, 수명 등의 속성을 가진다.

언리얼의 파티클 시스템을 **나이아가라**(Niagara)라고 한다. 나이아가라는 베타 버전으로 제공되다가 UE5에서는 메인 파티클 시스템 툴로 탑재되어 배포되기 시작하였다.

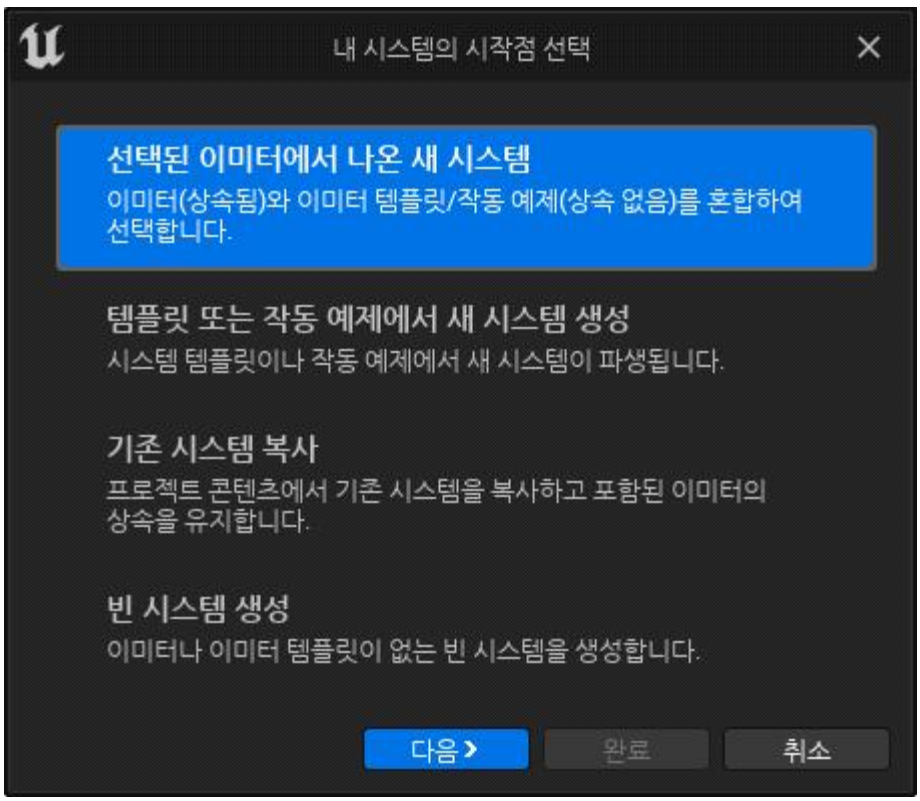
<참고> 나이아가라에 대해서는 다음의 문서를 참조하자.
https://docs.unrealengine.com/creating-visual-effects-in-niagara-for-unreal-engine/

<참고> **나이아가라** 이전에는 **캐스케이드**(Cascade)라는 툴이 파티클 시스템으로 사용되었다. 캐스케이드는 오랬동안 사용되어왔지만 UE5부터는 레거시로 되었다. 따라서 새로 배우는 학습자라면 **캐스케이드**에 관심을 가질 필요가 없다.
캐스케이드에 대한 자세한 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/4.27/RenderingAndGraphics/ParticleSystems/

이제부터, 언리얼의 시각 효과(VFX) 시스템인 나이아가라(Niagara)에 대해서 알아보자.
먼저 나이아가라와 관련된 용어인 **시스템**, **이미터**, **모듈**, **파라메터** 용어에 대해서 알아보자.
먼저, **시스템**(System)은 여러 이미터를 담고 있는 컨테이너이다. 여러 이미터가 결합되어서 하나의 시각 효과를 구현한다. 예를 들어 불꽃놀이의 경우에 불꽃이 동시에 여러 곳에서 터질 것이다. 각각의 터지는 불꽃이 각 이미터로 구현되고 이들의 모음으로 불꽃놀이라는 하나의 시스템을 구현하는 것이다.
시스템 에디터에는 타임라인 탭이 있다. 이 탭을 보면 어떤 이미터들이 시스템에 포함되어 있는지 확인할 수 있다. 시스템 에디터와 이미터 에디터는 UI가 거의 동일하다.
다음으로, **이미터**(Emitter)는 방출기를 구현하는 여러 모듈들을 담고 있는 컨테이너이다. 이미터에 포함된 각 모듈은 동일한 방출기에 대해서 각자의 방식으로 시뮬레이션할 수 있다. 예를 들어 불꽃놀이의 경우에서, 하나의 불꽃 입자에 대하여 스파크 렌더러와 리본 렌더러의 두 개의 모듈을 추가할 수 있다. 스파크 렌더러는 불꽃 하나를 스프라이트로 렌더링하여 불꽃처럼 보이게 하는 모듈이고 리본 렌더러는 불꽃이 지나가는 궤적에 빛의 자국을 표시하는 모듈이다.
다음으로, **모듈**(Module)은 나이아가라에서 가장 하위 레벨에 해당한다. 각 모듈은 스크립트로 작성

된 코드 블록에 해당한다. 이미터 내에 포함된 모듈들은 그룹으로 묶여서 관리된다. 기능별로 정해진 몇 개의 그룹이 있다. 각 그룹을 스택이라고 한다. 스택은 우선 순위가 있어서 위의 모듈부터 아래의 모듈의 순서로 실행된다.
각 모듈은 노드 네트워크의 그래프 형식으로 만든다. 이 그래프는 HLSL로 변환된다. 그래프에서 CustomHLSL 노드를 사용하면 실제로 직접 HLSL 코드로 코딩할 수도 있다.
다음으로, **파라메터**(Parameter)는 나이아가라에서 데이터 정보에 해당한다. 파라메터의 타입에는 수치값을 표현하는 Primitive, 열거형 타입인 Enum, 구조체 타입인 Struct, 외부 데이터 소스로부터 데이터를 제공하는 함수를 나타내는 Data Interface가 있다.
커스텀 파라메터를 추가하기 위해서는 먼저 파라메터를 지정하는 모듈인 **Set Parameter** 모듈을 모듈 스택에 추가한다. 그리고 그 모듈 내에서 기존 파라메터를 사용하거나 새 파라메터를 추가할 수 있다.

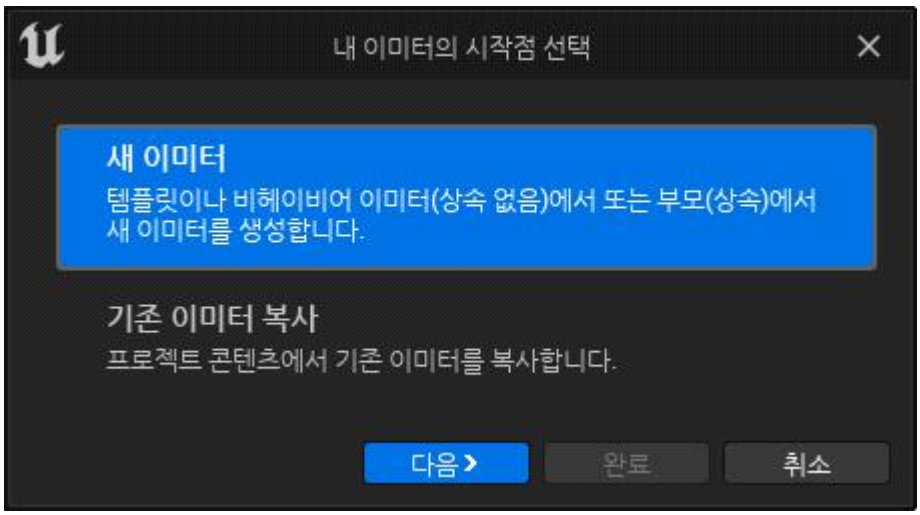
나이아가라 시스템이나 이미터를 생성할 때에 편리함을 위해서 몇가지 템플릿 중에서 하나를 선택하는 기능을 제공한다. 템플릿에는 기본적인 것들이 이미 추가되어 있으므로 자신의 목적에 맞게 약간 수정한 후에 바로 구현을 시작할 수 있다.
먼저 시스템 생성 도움창에 대해서 알아보자.



첫 번째, **선택된 이미터에서 나온 새 시스템** 옵션은 기존의 여러 이미터들을 선택하고 이들이 포함되도록 새 시스템을 생성한다. 이 옵션을 선택하면 이미터 목록을 보여주는 화면이 나타난다. 이 화면에서는 현재 프로젝트의 기존 이미터나 템플릿에서 제공하는 이미터가 모두 나열된다. 새 시스템에 포함시키고자 하는 이미터를 다수 개 선택하고 새 시스템을 생성하면 된다. 선택된 이미터가 기존 프로젝트의 이미터인 경우에는 이를 상속하여 새 이미터를 만들어 포함시킨다. 이미터 템플릿의 경우에는 상속한 이미터를 만드는 대신에 독자적인 이미터를 만든다.

두 번째, **템플릿 또는 작동 예제에서 새 새 시스템 생성** 옵션은 이미터 템플릿이 아닌 시스템 템플릿을 나열해준다. 시스템 템플릿을 선택하여 해당 템플릿에서 파생된 새 시스템을 만들면 된다.
세 번째, **기존 시스템 복사** 옵션은 기존 시스템을 복사하여 새 시스템을 만든다.
네 번째, **빈 시스템 생성** 옵션은 아무 이미터도 없는 빈 시스템을 만든다.
대부분의 경우 첫 번째 옵션을 사용하면 된다.

이제 이미터 생성 도움창에 대해서 알아보자.



첫 번째, **새 이미터** 옵션은, 기존의 가용한 이미터 템플릿들을 보여주고 하나를 선택하도록 한다. 선택된 이미터 템플릿으로부터 새 이미터를 생성한다. 이러한 템플릿을 선택하여 이미터를 만드는 방식은 상속하여 만드는 것이 아니라 새로운 독립적인 이미터를 만들어준다.
또한, 템플릿 선택 방식과 다른 방법으로도 이미터를 만들 수 있다. 기존에 만들어둔 이미터를 선택하고 선택된 이미터를 상속하여 새 이미터를 만드는 방법이다. 이는 약간의 속성값만 다른 여러 이미터를 만들고자 할 때에 부모 이미터를 만들고 이를 상속하여 여러 이미터를 만들때에 사용된다. 자식 이미터를 만들 후에 부모 이미터를 수정하는 경우에도 모든 자식 이미터가 그 수정내용을 반영해서 동작한다.
두 번째, **기존 이미터 복사** 옵션은, 이미 만들어진 이미터를 복사하여 새 이미터를 생성한다. 기존 이미터와 유사한 기능의 이미터를 만들 때에 사용하면 된다.
대체로 첫 번째 옵션을 사용하면 된다.

한 이미터의 내부 구조에 대해서 살펴보자.
이미터 내에는 모듈들을 배치하는 스택이 각 그룹별로 있다.
스택의 가장 위에는 **이미터 세팅** 그룹이 있다. 이 그룹에는 이미터 속성을 설정하는 **이미터 프로퍼티** 모듈이 있다. 그 아래에는 **이미터 스폰** 그룹, **이미터 업데이트** 그룹, **파티클 스폰** 그룹, **파티클 업데이터** 그룹, **이벤트 핸들러** 그룹, **렌더러** 그룹이 있다. 각 그룹 아래에 모듈들이 추가된다.
먼저, **이미터 스폰** 그룹에는 이미터가 처음으로 스폰될 때에 해야할 일을 수행하는 모듈을 배치하면 된다.
그다음, **이미터 업데이트** 그룹에는 시간의 경과에 따라서 이미터에 영향을 주는 모듈을 배치한다.
그다음, **파티클 스폰** 그룹에는 이미터로부터 파티클이 스폰될 때에 해야할 일을 수행하는 모듈을 배치한다.

그다음, **파티클 업데이트** 그룹에는 시간의 경과에 따라서 파티클에 영향을 주는 모듈을 배치한다. 그다음, **이벤트 핸들러** 그룹에는 이벤트의 생성 기능과 리스닝 기능을 추가할 수 있다. 이벤트 생성 기능은 이미터에서 특정 이벤트를 생성할 수 있도록 하고, 이벤트 리스닝 기능은 다른 이미터에서 생성된 이벤트에 대하여 반응을 하도록 한다. 이 영역에서 충돌 처리에 대한 기능도 설정한다. 그다음, **렌더러** 그룹에는 스폰된 각 파티클을 어떻게 렌더링할 지를 명시하는 렌더러 모듈들이 있다.

<참고> 이미터 모듈에 대한 레퍼런스는 다음의 문서를 참조하자. 각 그룹 별로 지원되는 모듈들이 자세하게 나열되어 있다.
https://docs.unrealengine.com/system-and-emitter-module-reference-for-niagara-effects-in-unreal-engine/

나이아가라는 잘 만들어진 수많은 모듈들을 제공하고 있다. 자신의 커스텀 모듈을 만들어야 할 일은 거의 없다. 제공되는 모듈들의 설정값을 적절하게 바꾸어 사용하면 된다. 즉 자신의 파티클 시스템을 만드는 일은 모듈들을 블록 형태로 적절하게 조합하여 구성하는 일과 동일하다. 많은 모듈들을 한번에 배우기는 어려운 일이다. 하나씩 선택해서 알아나가면 된다.

나이아가라에 대한 예제는 아직 많지 않은 편이다. 나이아가라 예제에 대해서는 엔진에서 학습용으로 제공되는 프로젝트인 **Content Examples**에 포함되어 있는 예제를 살펴보면 도움이 된다.

<참고> **Content Examples**에 포함되어 있는 나이아가라 예제에 대해서는 다음의 문서를 참조하자.
https://docs.unrealengine.com/4.27/Resources/ContentExamples/NiagaraEffects/
위의 링크의 나이아가라 예제를 설명한 다음의 문서도 도움이 된다.
https://www.cyanhall.com/tutorial/

이제부터 파티클 시스템에 대해서 학습해보자.

2. 스프라이트 파티클 이펙트 만들기

이 절에서 파티클 시스템 기초에 대해서 학습한다.
흔히 사용되는 시각 효과 방법은 스프라이트(sprite)를 이용하는 것이다. 스프라이트는 항상 카메라 방향을 바라보는 2D 평면이다. 평면에는 머티리얼을 적용한다.
이 절에서는 스프라이트를 사용하여 파티클 시스템을 만드는 방법에 대해서 학습한다.

이 절에서는 파티클 시스템을 처음으로 만들어본다. 예제로 스프라이트로 스모크 파티클 이펙트를 구현해본다. 스프라이트를 위한 머티리얼은 애니메이션 연속 이미지가 8x8 격자로 배치된 시트를 사용한다. 격자의 각 이미지를 스프라이트에 순차적으로 연속해서 보여주면 애니메이션처럼 보이게 된다. 이를 **SubUV 애니메이션**이라고 한다.

이 절의 예제에서, 나이아가라 시스템을 만들고 그 안에서 스프라이트를 다룰 수 있는 이미터를 설정한다. 그리고 시스템을 레벨에 배치하여 시스템에서 구현하는 시각 효과가 레벨에서 보이도록 할 것이다.

<참고> 이 절에서의 예제과 관련된 자세한 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/how-to-create-a-smoke-effect-using-sprite-particles-in-niagara-for-unreal-engine/

이제부터 예제를 통해서 학습해보자

---

1. 새 프로젝트 Ppsmoke를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Ppsmoke로 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.
그리고, 툴바의 **액터 배치** 아이콘을 클릭하고 **기본** 아래의 **플레이어 스타트** 액터를 드래그하여 레벨에 배치하자. 위치를 (0,0,92)로 지정하자.

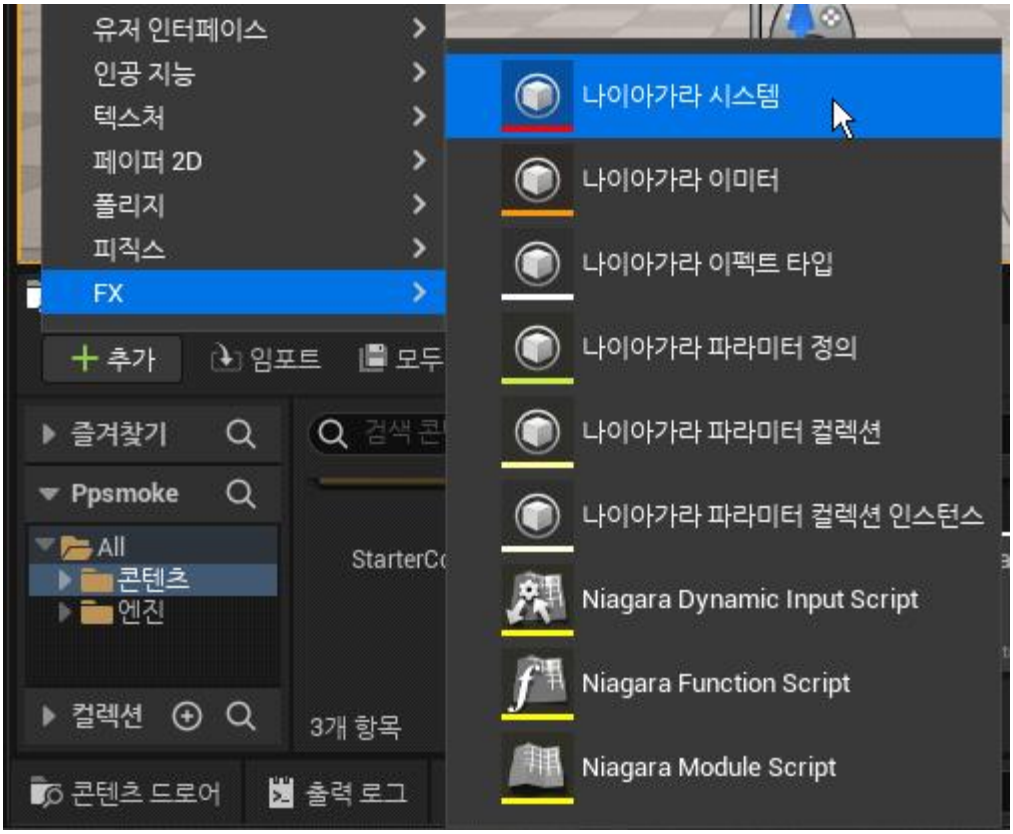
2. 이번 프로젝트에서 필요한 외부 애셋을 준비하자.
필요한 애셋은 **시작용 콘텐츠**에 포함되어 있다. **시작용 콘텐츠**가 포함되어 있는 다른 프로젝트를 찾아보거나 또는 **시작용 콘텐츠**가 포함되도록 새 임시 프로젝트를 만들자. 임시 프로젝트에서 **Content** 폴더 아래에서 다음의 파일들을 찾아보자. 다음의 파일들을 우리의 프로젝트로 복사하여 가져오자. 폴더 구조가 동일하게 되도록 하자. 가져온 후에 임시 프로젝트는 종료하고 삭제하면 된다.

```
StarterContent/Particles/Materials/M_smoke_subUV.uasset
StarterContent/Textures/T_Smoke_SubUV.uasset
StarterContent/Textures/T_Smoke_Tiled_D.uasset
```

<참고> 다른 프로젝트에 있는 머티리얼 애셋을 가져올 때에는 그 머티리얼 애셋이 내부적으로 사용하는 텍스처 애셋도 함께 가져와야 한다. 만약 내부의 의존 관계를 모르는 특정 머티리얼을 우리의 프로젝트로 가져오고 싶다면 단순히 복사하는 방법이 아니라 정식으로 가져오는 방법을 사용해야 한다.
우리는 이번 단계에서 이미 복사해온 머티리얼인 M_smoke_subUV 머티리얼 애셋을 복사가 아닌 정식 방법으로

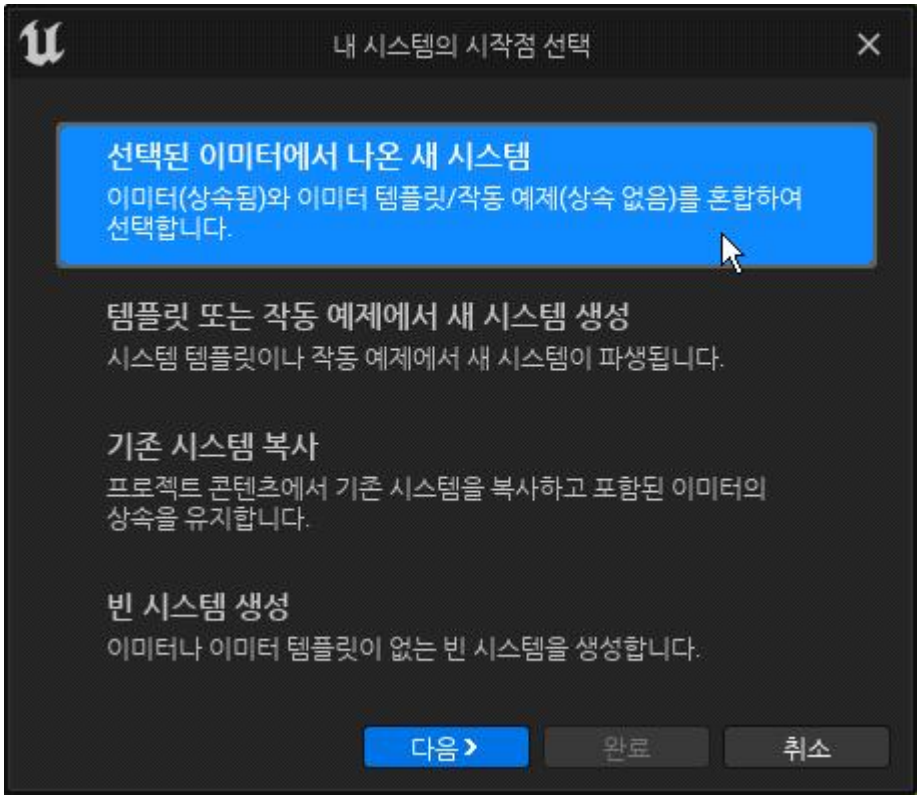
가져오는 경우를 생각해보자.
먼저, **시작용 콘텐츠**가 이미 포함되어 있는 프로젝트를 찾자. 포함하는 프로젝트가 있다면 그 프로젝트의 **콘텐츠 브라우저**에서 M_smoke_subUV 애셋을 검색하여 찾자. 그다음, 애셋 위에서 우클릭하고 **애셋 액션** » **이주**를 선택하자. 다음 창에서 선택된 애셋과 더불어 그 애셋이 내부적으로 사용하는 다른 애셋도 모두 나열해 준다. **확인**을 클릭하면 폴더 선택 대화상자가 뜬다. 여기서 우리의 새 프로젝트의 **Content** 폴더를 선택하자. 새 프로젝트의 **Content** 폴더 아래에 동일한 폴더 구조로 세 파일이 모두 복사될 것이다.

3. 먼저, **시스템**(System)을 생성하자.

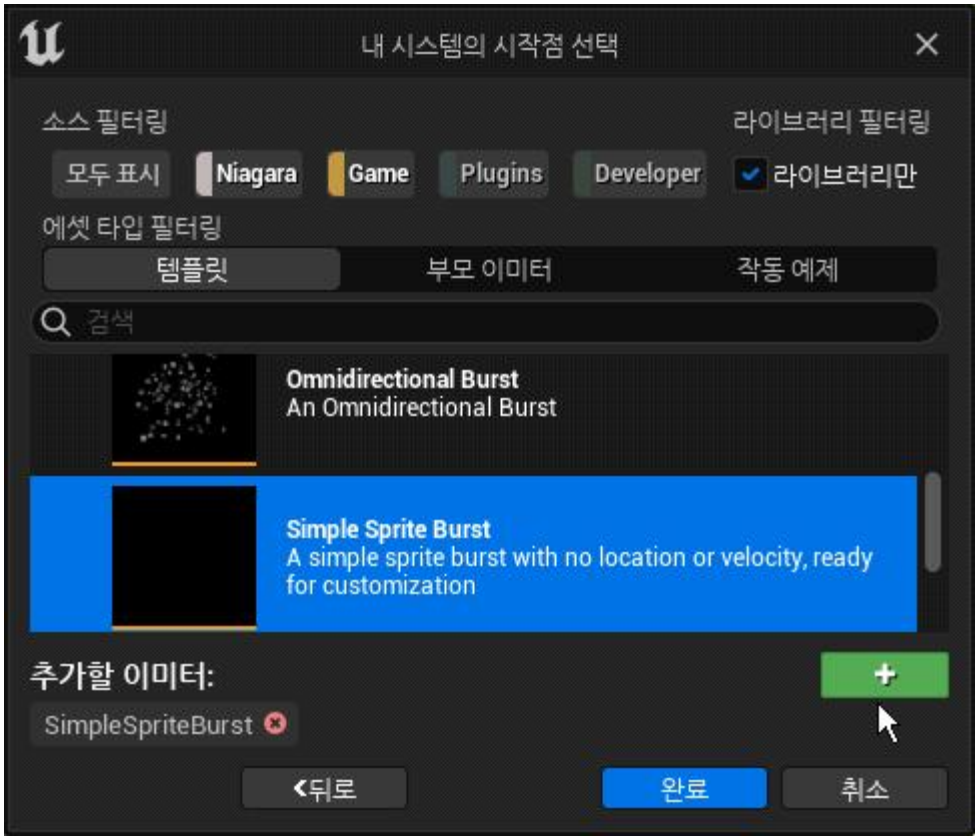


콘텐츠 브라우저에서, **+추가**를 클릭하고 **FX** » **나이아가라 시스템**을 선택하자.

4. **내 시스템의 시작점 선택** 창에서 **선택된 이미터에서 나온 새 시스템**을 선택하자.

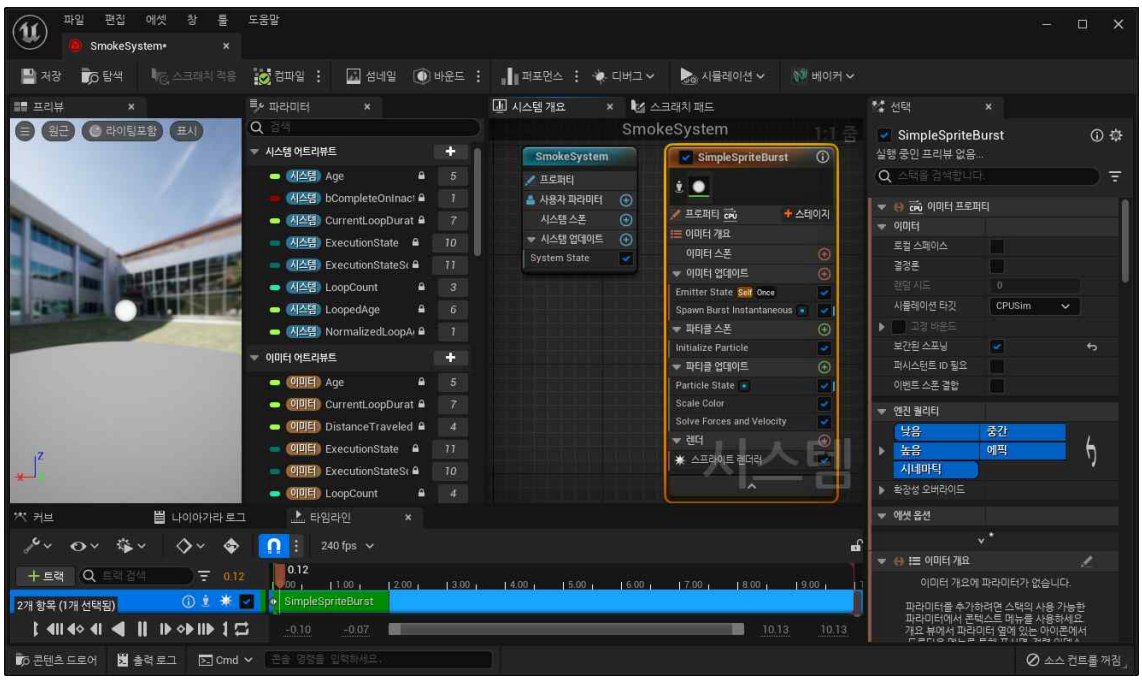


5. 다음 대화상자에서는 템플릿을 통해서 이미터를 쉽게 추가할 수 있도록 한다.
템플릿 영역에는 엔진이 제공하는 템플릿을 나열해서 보여준다.
우리는 템플릿 영역에서 **Simple Sprite Burst**를 클릭하여 선택하자. 그다음, + 버튼을 클릭하여 이미터를 추가하자. 그리고 **완료** 버튼을 클릭하자.



생성된 **시스템**의 이름을 **SmokeSystem**으로 수정하자.

6. **SmokeSystem**을 더블클릭하자. 나이아가라 에디터가 뜬다. 에디터는 아래와 같은 모습으로 보일 것이다.

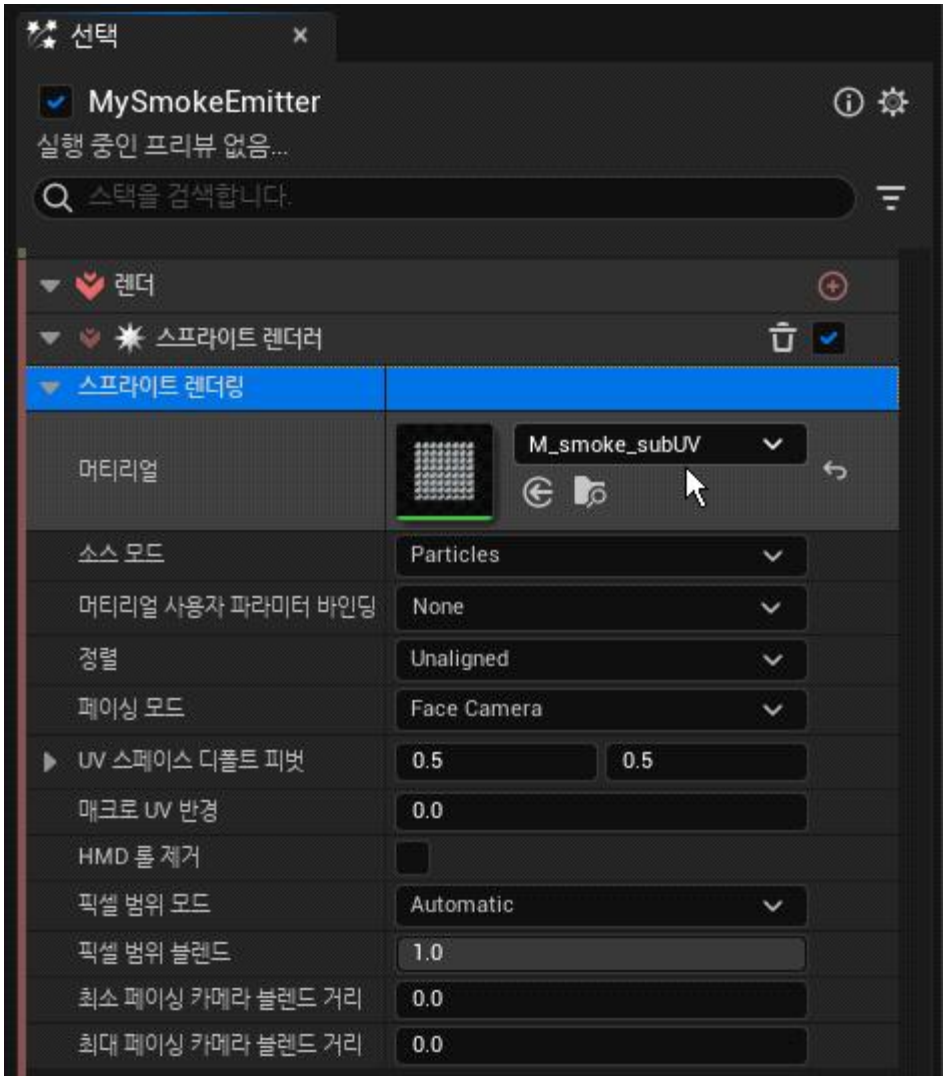


에디터의 왼쪽에는 **프리뷰** 탭이 있고 그 다음에 **파라미터** 탭이 있다.
에디터의 중간에는 격자판 모양의 **시스템 개요** 탭이 있고 **스크래치 패드** 탭이 있다. 디폴트로 **시스템**

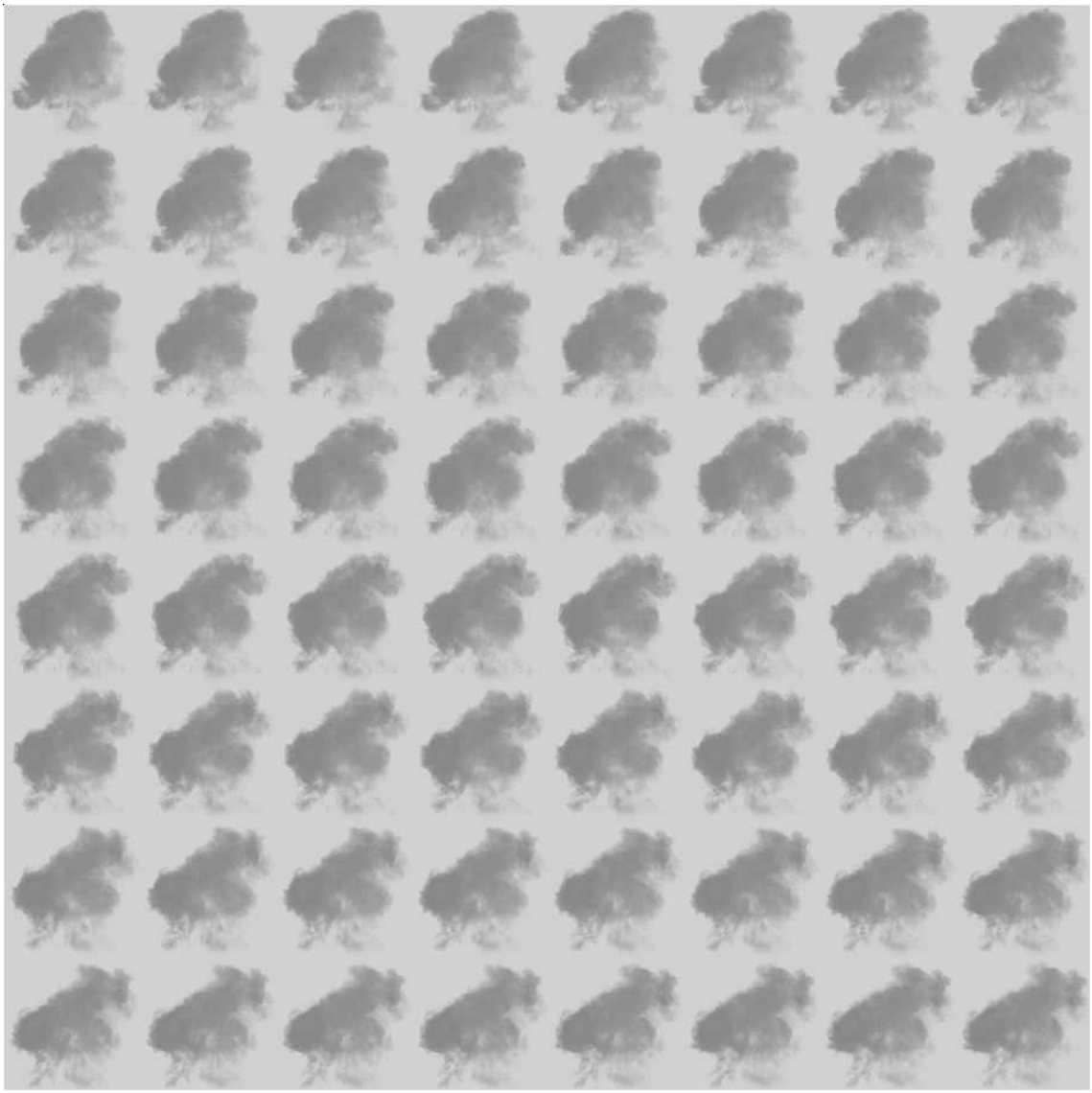
개요 탭이 표시된다.
시스템 개요 탭의 격자판에는 두 개의 노드가 있을 것이다. 왼쪽의 파란색 노드는 **시스템 오버뷰** 노드이다. 오른쪽의 갈색 노드는 **이미터 오버뷰** 노드이다. 노드를 클릭하거나 노드 내의 각 항목을 클릭하면 오른쪽의 **선택** 탭에 상세 정보가 표시된다.

7. **시스템 개요** 탭의 격자판의 오른쪽에 있는 **이미터 오버뷰** 노드는 이름이 **SimpleSpriteBurst**으로 되어 있다. 이것은 우리가 시스템 생성 시에 템플릿 영역 아래에 있는 이미터인 **Simple Sprite Burst**를 추가하였기 때문에 만들어진 이미터 인스턴스이다. 이 노드의 이름을 **MySmokeEmitter**로 수정하자. 노드의 상단 이름 부분에서 클릭하거나 F2 키를 누르면 수정할 수 있다.

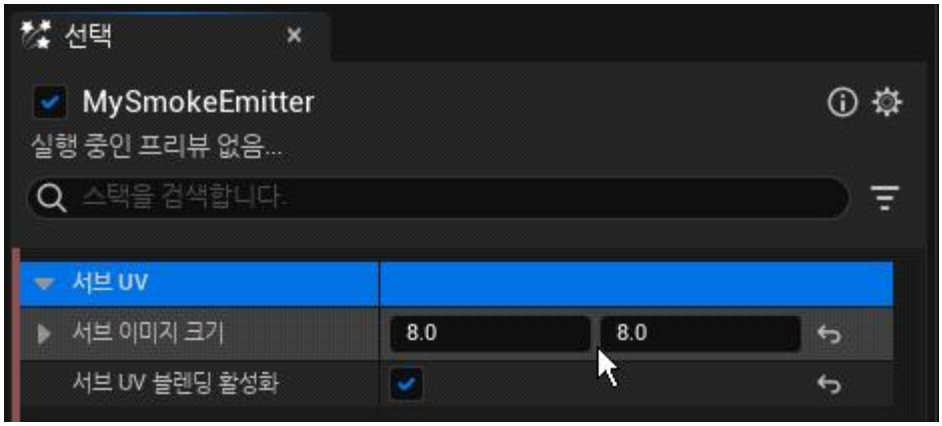
8. 먼저 화면에 무언가 보이도록 해보자.
시스템 개요 탭에서 **이미터 오버뷰** 노드의 가장 아래에 **스프라이트 렌더러**(**Sprite Renderer**)가 있다. 이 항목을 클릭하자. 오른쪽의 **선택** 탭에 상세 정보가 표시될 것이다. 여기에서 스프라이트에 입혀질 머티리얼을 지정하면 된다. **Material** 속성값에 디폴트로 **DefaultSpriteMaterial**로 되어있을 것이다. 이것을 우리가 사용할 머티리얼인 **M_smoke_subUV**를 지정하자.



9. 우리가 사용할 머티리얼인 **M_smoke_subUV**은 **SubUV** 텍스처를 다루는 머티리얼이다. 머티리얼에 지정되어 있는 텍스처에서 이미지 격자에 몇 개의 이미지가 있는지를 렌더러에게 알려주어야 한다. 우리의 머티리얼 **M_smoke_subUV**에서 사용하는 **SubUV** 텍스처는 **T_Smoke_SubUV**이다. 이 텍스처는 8x8 격자로 되어있다. 원본 텍스처의 크기는 2048x2048이고 알파 채널이 포함되어 있다.



10. 계속해서 **MySmokeEmitter**의 선택 탭에서 작업하자.
서브 UV 영역으로 가자.
서브 UV 영역에서 **서브 이미지 크기(SubImageSize)**를 디폴트인 1,1을 8,8로 수정하자.
또한, 그 아래의 **서브 이미지 블렌딩 활성화(SubUVBlendingEnabled)**는 디폴트로 체크되어 있지 않지만 우리는 체크하도록 하자.
저장하고 컴파일하자.



<참고> **SubUV** 텍스처에 대해서 알아보자. 작은 텍스처들을 격자 모양으로 인접하게 붙여서 만든 텍스처를 **SubUV** 텍스처라고 한다. SubUV 제작 방법에 대해서는 다음의 링크를 참조하자.

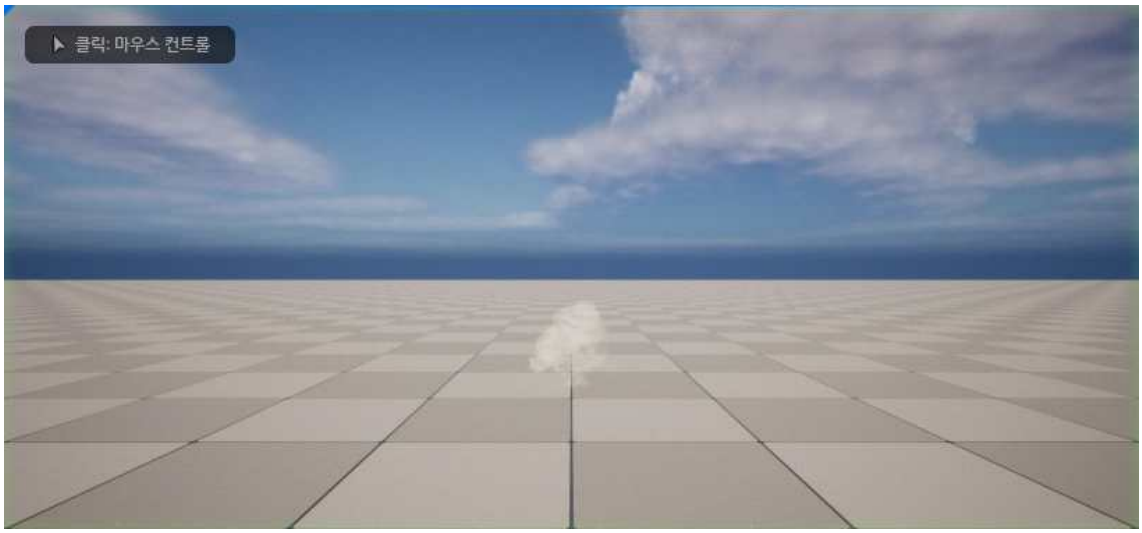
https://blog.naver.com/othiai4239/130144720423

11. 레벨 에디터로 가자.

콘텐츠 브라우저에서 **SmokeSystem**을 드래그하여 레벨에 배치하자.

위치는 (300,0,50)에 배치하자.

플레이해보면 잠깐 나타났다 사라질 것이다.



12. 이제부터 스모크 효과가 되도록 수정해가면서 만들어보자.

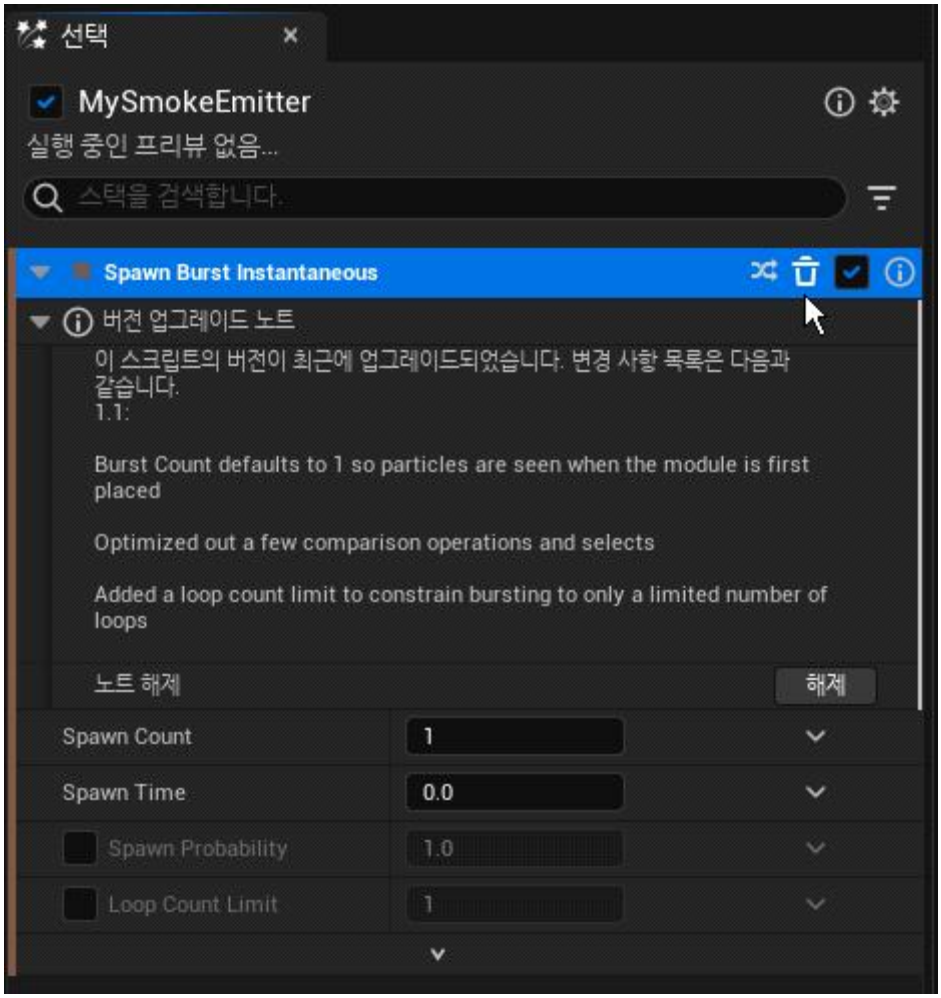
이미터 업데이트 그룹, **파티클 스폰** 그룹, **파티클 업데이트** 그룹의 순서로 각 그룹 내의 모듈에 대한 세팅을 수정해나갈 것이다.

가장 먼저, **이미터 오버뷰** 노드에 있는 **이미터 업데이트** 그룹에서의 세팅을 시작해보자.

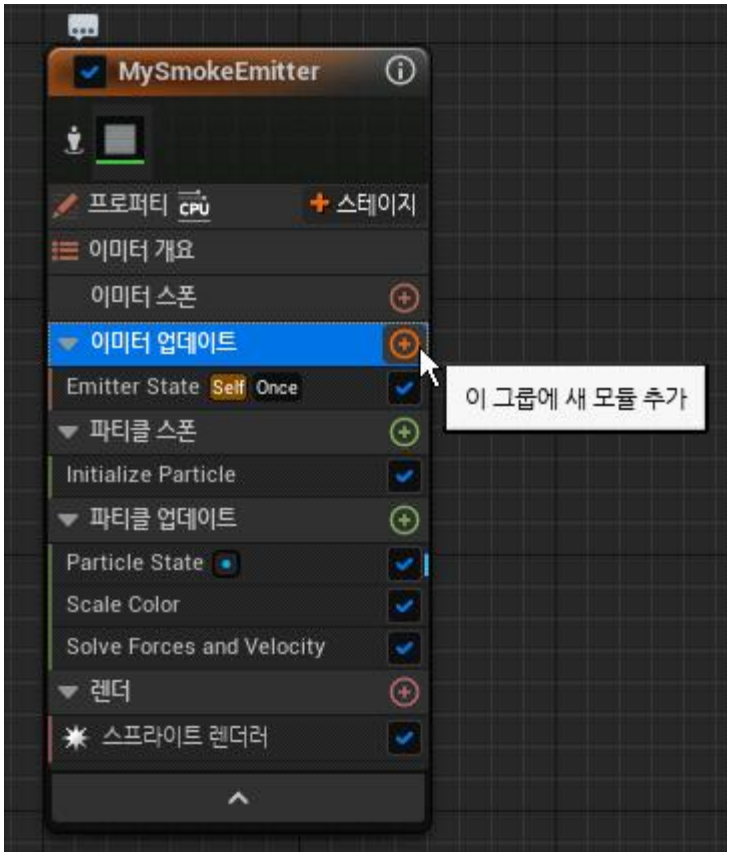
이미터 업데이트 그룹에 있는 모듈들은 이미터가 업데이트되도록 매 프레임마다 실행하는 모듈들이다.

이 모듈 중에서 **Spawn Burst Instantaneous** 모듈을 선택하고 오른쪽 **선택** 탭으로 가자.

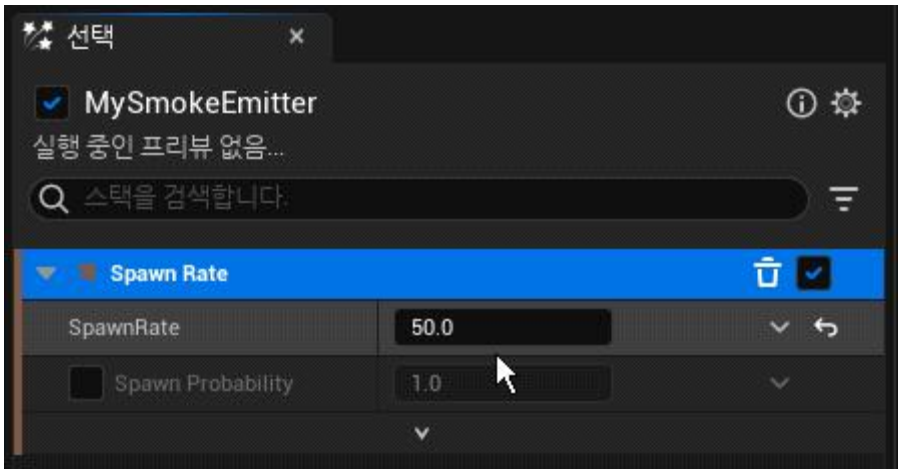
이 **Spawn Burst Instantaneous** 모듈은 일정 개수의 파티클을 한꺼번에 생성하여 방출하는 기능을 수행한다. 그런데 우리는 스모크 파티클이 일정하게 기둥 모양으로 올라오도록 할 것이다. 따라서 이 모듈은 우리에게 필요없는 것이다. 오른쪽 위의 쓰레기통 아이콘을 클릭하여 이 모듈을 삭제하자.



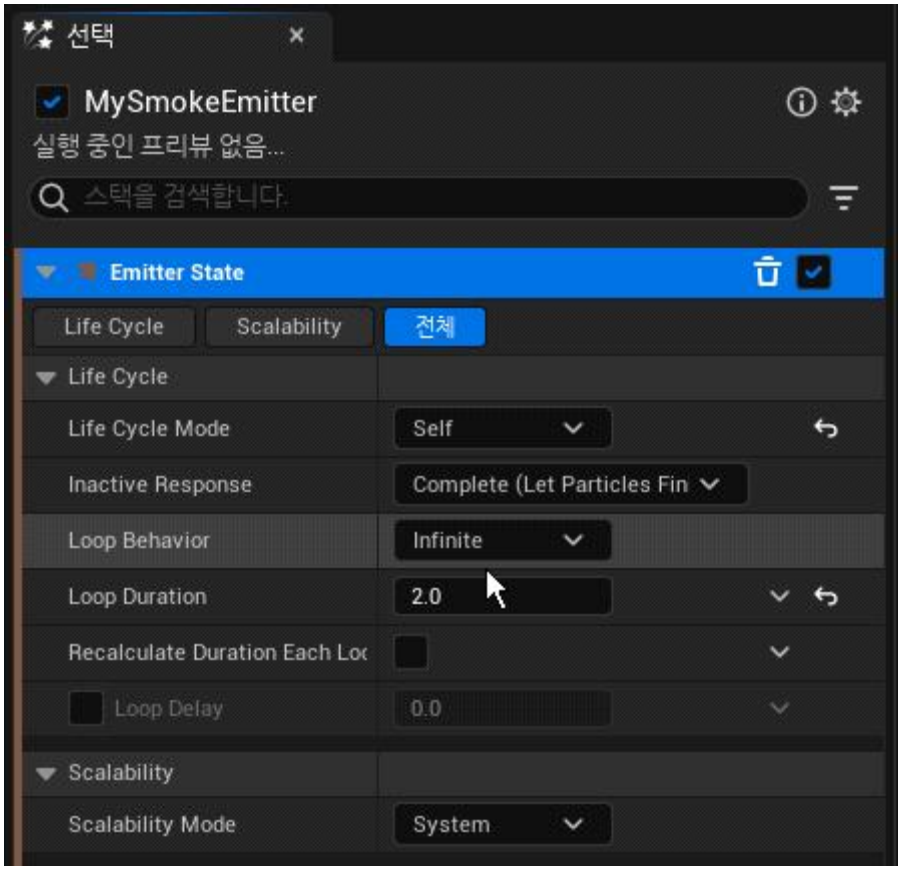
13. **이미터 오버뷰** 노드에서 **이미터 업데이트** 그룹의 오른쪽 + 아이콘을 클릭하여 새 모듈을 추가하자. **Spawn Rate** 모듈을 검색하여 추가하자. 이 모듈은 파티클을 정해진 비율로 계속해서 스폰한다.



14. 노드에 **Spawn Rate** 모듈을 선택하고 선택 탭에서 **SpawnRate** 속성값을 0에서 50으로 수정하자.
이 정도의 값은 뭉게뭉게 피어오르는 연기의 모습에 적당하다.

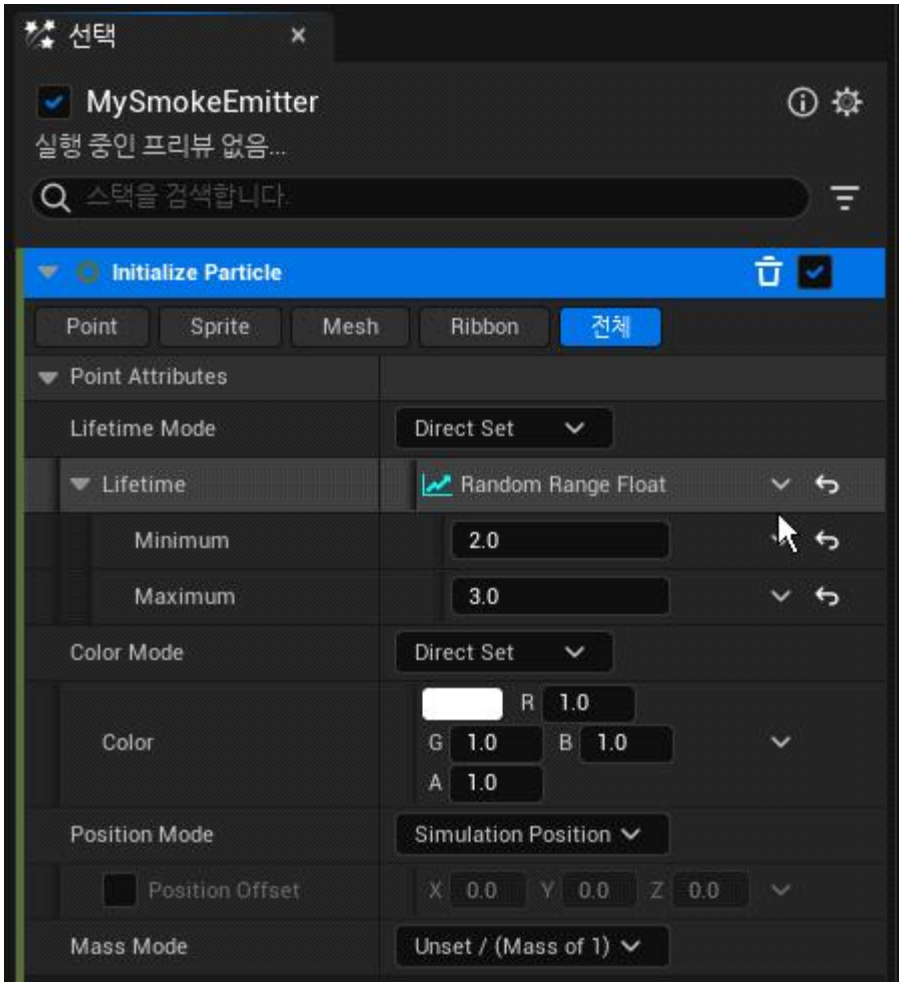


15. 플레이해보자. 연기가 한번 나타났다가 사라질 것이다.
사라지지 않고 계속 나타나도록 무한루프로 반복하도록 해보자.
이미터 오버뷰 노드의 **이미터 업데이트** 그룹에 있는 **Emitter State** 모듈을 선택하자.
그리고, **선택** 탭에서 **Loop Behavior**를 **Once**에서 **Infinite**로 수정하자.

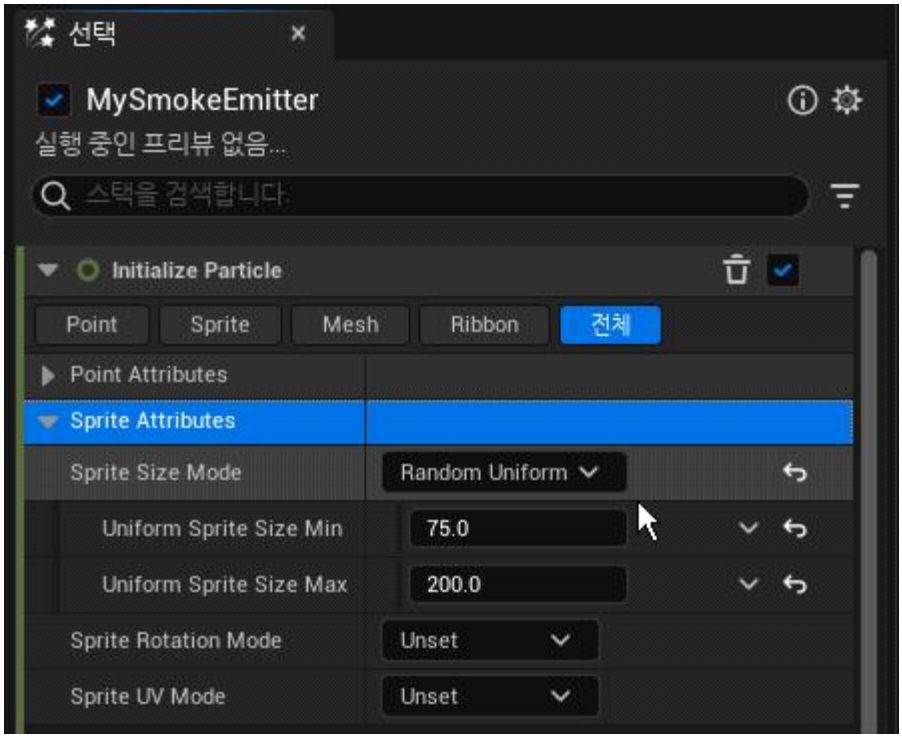


이제 연기가 사라지지 않고 계속 나타날 것이다.
지금까지, **이미터 업데이트** 그룹에 대한 세팅을 모두 완료하였다.

16. 이제부터는, **파티클 스폰** 그룹에 대한 세팅을 시작하자.
이 그룹의 모듈들은 파티클이 처음으로 스폰될 때에 실행되는 모듈들이다.
파티클 스폰 그룹에 있는 **Initialize Particle** 모듈을 선택하자. 이 모듈은 파티클의 초기화와 관련된 다양한 속성값을 가지고 있다.
먼저, **Point Attributes** 영역 아래의 **Lifetime** 속성값을 찾아보자. 이 속성값은 파티클의 수명을 지정한다. 지정된 수명 후에는 파티클이 소멸된다. 디폴트로 2로 되어있다. 단순히 고정된 시간에 사라지는 것보다 랜덤하게 수명을 지정하면 더욱 자연스럽게 된다.
Lifetime 속성값 입력상자의 오른쪽의 꺾쇠 모양을 클릭하면 **소스 필터링** 드롭다운 목록 창이 나타난다. 여기서 **Random Range Float**를 검색하여 선택하자.
Random Range Float를 지정한 후에는 바로 아래에 **Minimum**과 **Maximum** 입력칸이 추가로 나타난다. 디폴트로 되어있는 0과 1을 2와 3으로 수정하자.

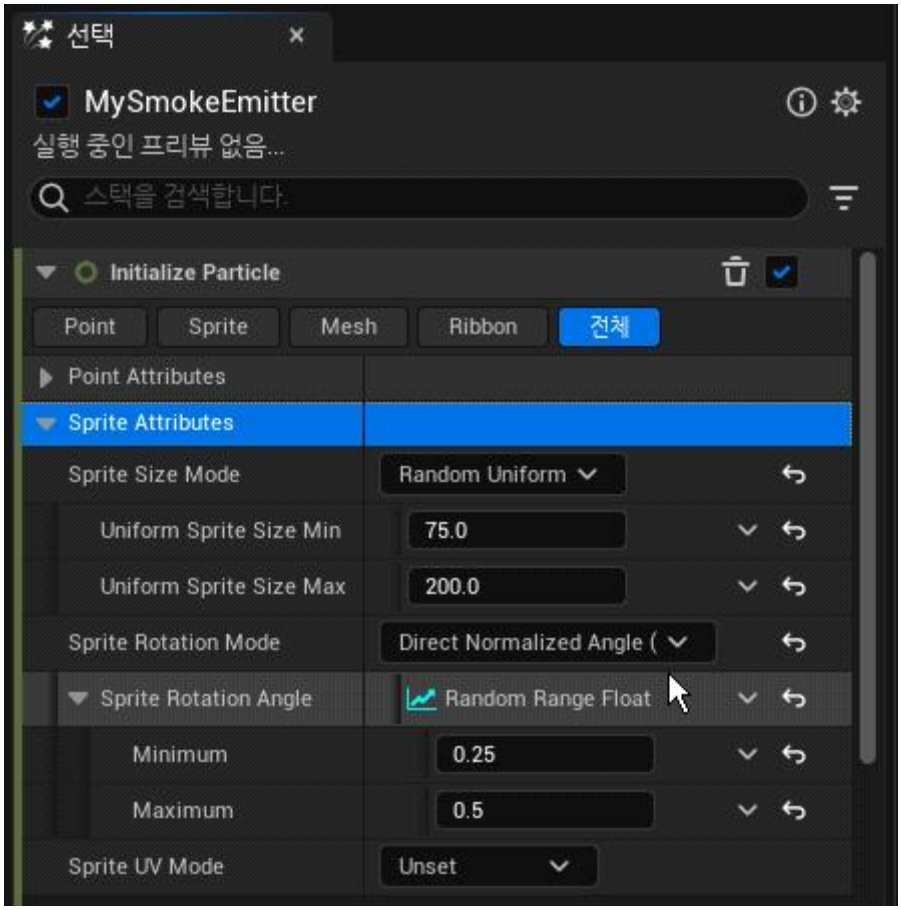


17. 계속해서 **Initialize Particle** 모듈의 **선택** 탭에서 작업하자.
이제 스프라이트의 크기를 조절해보자.
현재의 파티클의 크기는 작은 편이어서 더 크게 바꾸어보자.
선택 탭에서 **Sprite Attributes** 영역에서 **Sprite Size Mode**를 **Uniform**에서 **Random Uniform**으로 수정하자.
그리고 그 아래에 나타나는 **Uniform Sprite Size Min**과 **Uniform Sprite Size Max**에를 5와 10에서 75와 200으로 수정하자.



플레이해보자. 스프라이트의 크기가 커졌을 것이다.

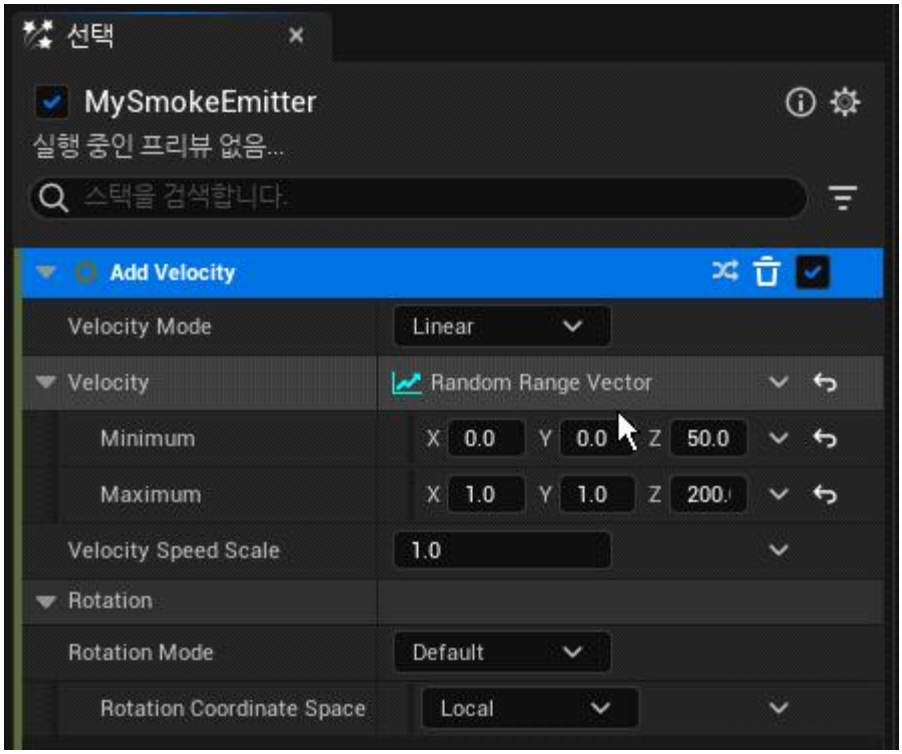
18. 계속해서 **Initialize Particle** 모듈의 **선택** 탭에서 작업하자.
이제 좀 더 다양성을 주기 위해서 스프라이트에 회전도 지정해보자.
바로 아래의 **Sprite Rotation Mode**를 찾아보자. 디폴트 값이 **Unset**으로 되어있을 것이다. 이것을 클릭하고 **Direct Normalized Angle (0-1)**로 바꾸자. 여기서 회전 각도는 도 단위 대신 [0,1] 범위의 값을 사용한다.
그다음, 그 아래에 나타나는 **Sprite Rotation Angle**의 오른쪽 꺾쇠 아이콘을 클릭하고 **Random Range Float**를 선택하여 지정하자. 그리고 그 아래에 생기는 **Minimum**과 **Maximum** 입력에 0과 1을 0.25와 0.5로 수정하자. 이제 0.25에서 0.5 범위에서 회전 각도가 랜덤하게 지정될 것이다.



19. 크기와 회전에 대해서는 적절하게 세팅하였다. 그러나, 연기가 움직이지 않고 제자리에서만 스핀한다. 연기는 스폰되면 바로 움직이도록 하는 것이 자연스럽다. 이를 위해서 초기 속도를 더해주자.

이미터 오버뷰 노드에서 **파티클 스폰** 그룹의 오른쪽 +를 클릭하고 **Add Velocity** 모듈을 검색하여 추가하자. 그다음, **Add Velocity** 모듈을 선택하고 **선택** 탭에서 **Velocity** 속성을 찾아보자. 값이 (0,0,50)으로 되어있을 것이다. 우리는 값의 오른쪽에 있는 꺾쇠 아이콘을 클릭하고 **Random Range Vector**를 선택하자.

그다음, 아래에 생기는 **Minimum**과 **Maximum**에 디폴트 값인 (0,0,0)과 (1,1,1)을 (0,0,50)과 (1,1,200)로 수정하자.

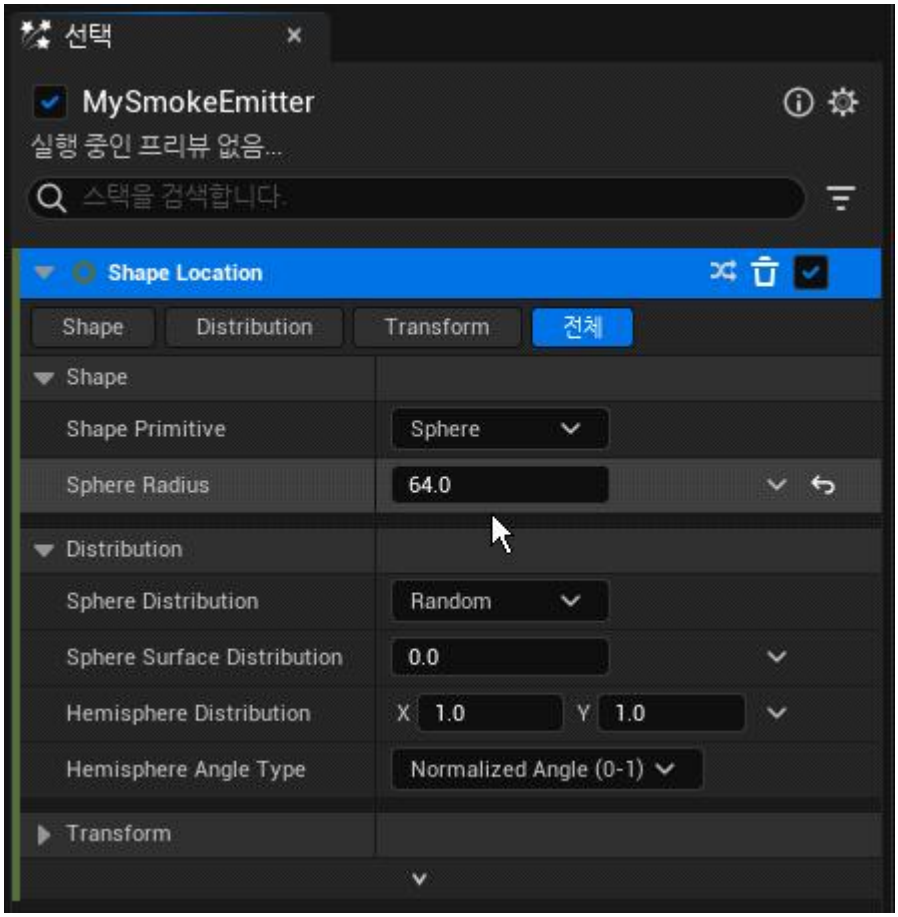


플레이해보자. 스프라이트가 위로 올라갈 것이다.

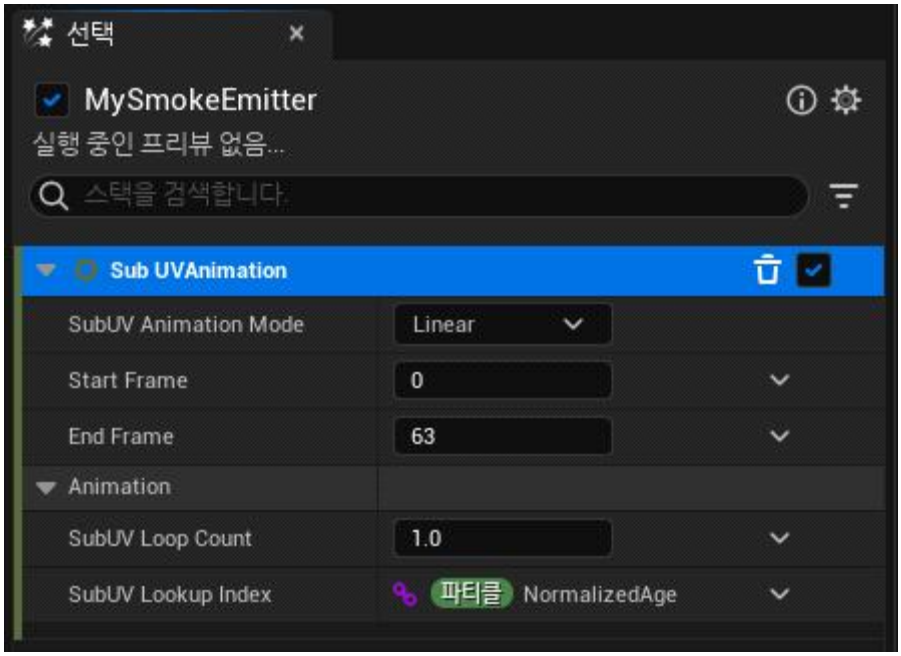
20. 스폰되는 위치에도 다양성을 주자. **Sphere Location** 모듈을 사용하면 스프라이트가 구체 내에서 스폰되도록 한다.

이미터 오버뷰 노드에서 **파티클 스폰** 그룹의 오른쪽 +를 클릭하고 **Sphere Location** 모듈을 검색하여 추가하자. 그다음, **Sphere Location** 모듈을 선택하고 **선택** 탭에서 **Sphere Radius** 속성값을 디폴트인 100을 64로 수정하자.

그 아래의 **Sphere Distribution**이 **Random**으로 지정되어있을 것이다. 따라서 구체 내에서 랜덤하게 스폰될 위치가 결정될 것이다.

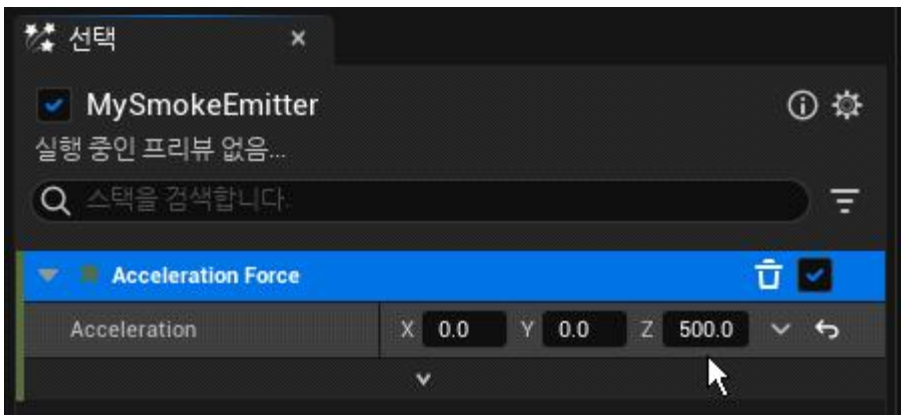


21. 우리가 파티클의 모습을 표현하기 위하여 사용하는 스프라이트 머티리얼은 **SubUV** 텍스처를 사용한다. **SubUV** 텍스처에는 애니메이션 되도록 보이게 하는 다수의 이미지가 포함되어 있다. 그런데, **SubUV**에 대한 설정을 해주지 않으면 렌더러는 첫 스프라이트 이미지만 사용하여 렌더링하게 된다. 따라서 **SubUV**의 전체 이미지가 사용되도록 명시적으로 설정해주어야 한다.
파티클 스폰 그룹에서 + 버튼을 클릭하고 **SubUV Animation** 모듈을 추가하자. 오른쪽 선택 탭에서 **SubUV Animation Mode**가 **Linear**로 되어있을 것이다. 또한 **Start Frame**과 **End Frame**이 0과 63으로 되어있을 것이다. 현재의 머티리얼 정보를 참조하여 에디터에서 이미 적절히 지정해주었다.



지금까지, **파티클 스폰** 그룹에 대한 세팅을 모두 완료하였다.

22. 이제부터는, **파티클 업데이트** 그룹에 대한 세팅을 시작하자.
이 그룹의 모듈들은 파티클이 매 프레임 갱신될 때마다 실행되는 모듈들이다.
이전의 파티클 스폰에서 **Add Velocity**를 추가하여 약간 위쪽으로 움직이는 속도를 파티클의 스폰 시에 주도록 하였다. 그러나 이것만으로 부족할 때가 있다. 시간이 흐름에 따라서 속도의 변화를 주고자 할 때가 있다. 이 경우에는 가속도나 힘을 가해주어야 한다. 우리는 연기가 위로 높이 올라가도록 매 업데이트 시마다 가속도를 가해주도록 해보자.
파티클 업데이트 그룹에서 + 버튼을 클릭하고 **Acceleration Force** 모듈을 추가하자.
그다음, **Acceleration Force** 모듈을 선택하고 **선택** 탭에서 **Acceleration** 속성값을 (0,0,0)에서 (0,0,500)으로 수정하자. 연기가 위쪽으로 빠르게 올라갈 것이다.



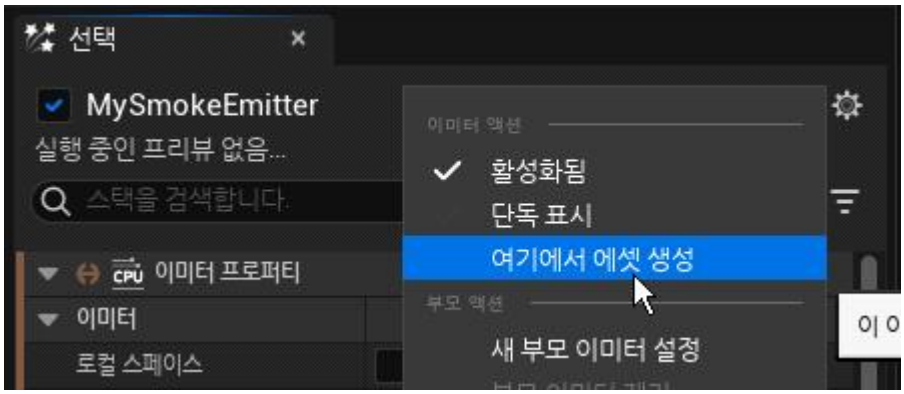
<참고> 이미터의 한 그룹에 배치된 모듈들은 하나의 스택 구조를 이루고 있다. 그리고 실행 시에는 스택의 위에서부터 아래로의 순서로 실행된다. 새로운 모듈이 스택에 추가될 때에는 엔진은 스택의 가장 마지막에 배치해준다.
한편 **Solve Forces And Velocity**라는 모듈은 특별한 모듈로 항상 마지막에 실행되어야 한다. 따라서 스택의 가장 마지막에 위치해야 한다. **Solve Forces And Velocity** 모듈이 스택에 배치되어 있는 경우에는 새로운 모듈이

추가될 때에 엔진은 가장 마지막이 아닌 이 모듈의 바로 위에 배치해준다.

23. 이제 이미터를 모두 완성하였다. **이미터 오버뷰** 노드는 다음과 같이 되었을 것이다.

24. 이제 이미터를 모두 완성하였다.
플레이해보자.

25. 완성한 이미터를 이 시스템에서만 사용할 것이라면 이미터를 따로 저장하지 않아도 된다. 그러나 작성한 이미터를 나중에 다른 시스템에서 재사용할 것이라면 애셋 파일로 저장해두면 된다. **이미터 오버뷰** 노드를 선택하고 **선택**에서 가장 위의 오른쪽 톱니 아이콘을 클릭하자.
드롭다운 메뉴에서 **여기에서 에셋 생성**을 선택하자.



저장할 이름을 묻는 대화상자가 뜰 것이다. 이름을 디폴트인 **MySmokeEmitter**로 그대로 두고 저장 버튼을 클릭하자. 콘텐츠 브라우저에 가서 확인해보면 **MySmokeEmitter**로 애셋이 생성되었을 것이다.

이 절에서는 스프라이트를 사용하여 스모크 파티클 시스템을 만드는 방법을 학습하였다.

3. 파티클 라이트 이펙트 만들기

이 절에서 파티클 라이트 이펙트에 대해서 학습한다.

조명 효과가 있는 파티클을 월드에 배치하면 레벨의 시각적 효과가 더 멋지게 된다. 이 절에서는 파티클과 라이트를 동시에 스폰하는 이미터를 만들어본다.

<참고> 이 절에서의 예제과 관련된 자세한 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/how-to-create-particle-effects-that-emit-light-in-niagara-for-unreal-engine/

이제부터 예제를 통해서 학습해보자

---

1. 새 프로젝트 **Pplight**를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Pplight로 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.
그리고, 툴바의 **액터 배치** 아이콘을 클릭하고 **기본** 아래의 **플레이어 스타트** 액터를 드래그하여 레벨에 배치하자. 위치를 (0,0,92)로 지정하자.

2. 이번 프로젝트에서 필요한 외부 애셋을 준비하자.
필요한 애셋은 **시작용 콘텐츠**에 포함되어 있다. **시작용 콘텐츠**가 포함되어 있는 다른 프로젝트를 찾아보거나 또는 **시작용 콘텐츠**가 포함되도록 새 임시 프로젝트를 만들자. 임시 프로젝트에서 **Content** 폴더 아래에서 다음의 파일들을 찾아보자. 다음의 파일들을 우리의 프로젝트로 복사하여 가져오자. 폴더 구조가 동일하게 되도록 하자. 가져온 후에 임시 프로젝트는 종료하고 삭제하면 된다.

```
StarterContent/Particles/Materials/M_Radial_Gradient.uasset
```

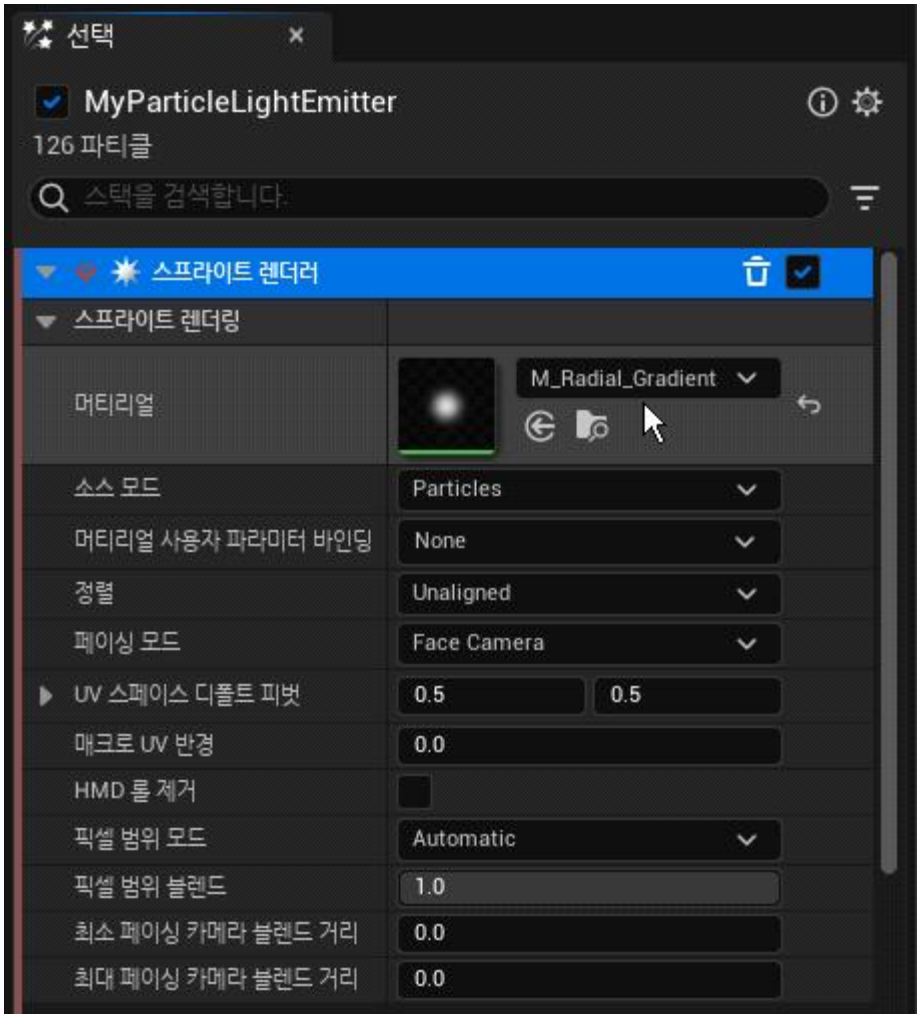
3. 먼저, **시스템**(System)을 생성하자.
콘텐츠 브라우저에서, **+추가**를 클릭하고 **FX** » **나이아가라 시스템**을 선택하자.
그다음, **내 시스템의 시작점 선택** 창에서 **선택된 이미터에서 나온 새 시스템**을 선택하자.
그다음, 선택 가능한 이미터들을 나열해서 보여준다. 이 중에서 템플릿 영역 아래에 있는 이미터인 **Fountain**을 클릭하여 선택하자. 그다음, **+** 버튼을 클릭하여 이미터를 추가하자. 그리고 **완료** 버튼을 클릭하자.
생성된 애셋의 이름을 **ParticleLightSystem**으로 수정하자.

4. **ParticleLightSystem**을 더블클릭하자. 나이아가라 에디터가 뜬다.
시스템 개요 탭의 격자판의 오른쪽에 **Fountain** 이름의 노드가 있다. 이것은 우리가 시스템 생성 시에 템플릿 영역 아래에 있는 이미터인 **Fountain**을 추가하였기 때문에 만들어진 이미터 인스턴스이다. 이 노드의 이름을 **MyParticleLightEmitter**로 수정하자.

5. 먼저 스프라이트 머티리얼을 적용하자.

시스템 개요 탭에서 **MyParticleLightEmitter 이미터 오버뷰** 노드의 가장 아래에 **스프라이트 렌더러**(**Sprite Renderer**)가 있다. 이 항목을 클릭하자.

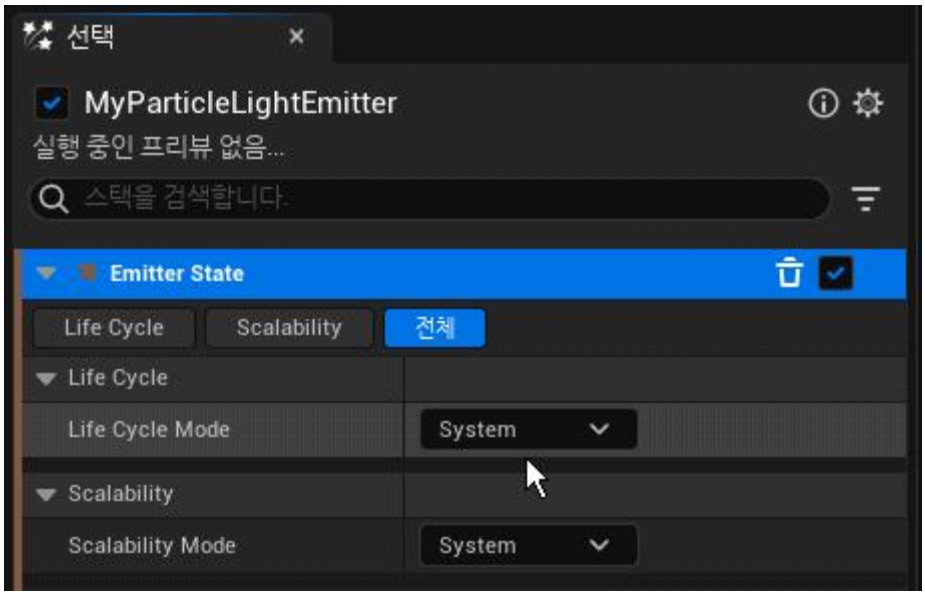
오른쪽의 **선택** 탭에 상세 정보가 표시될 것이다. 여기에서 스프라이트에 입혀질 머티리얼을 지정하면 된다. **Material** 속성값에 디폴트로 **DefaultSpriteMaterial**로 되어있을 것이다. 이것을 우리가 사용할 머티리얼인 **M_Radial_Gradient**를 지정하자.



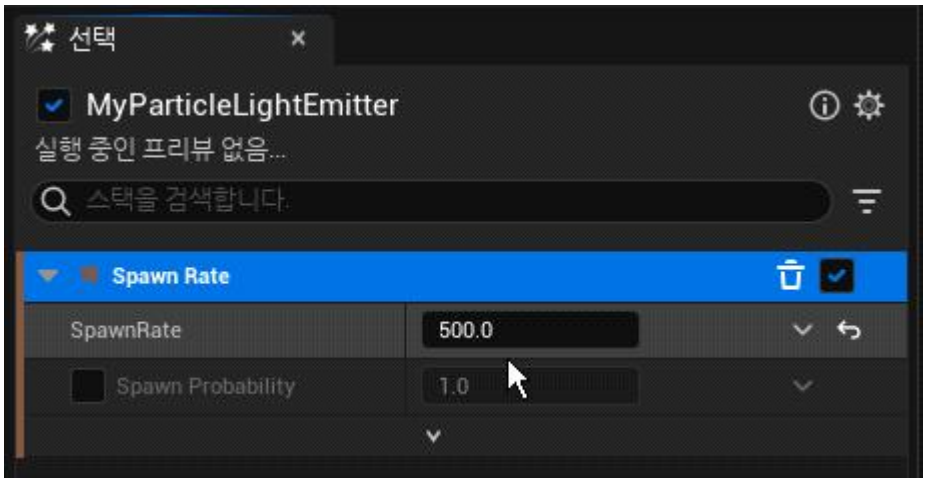
6. 레벨 에디터로 가자.
콘텐츠 브라우저에서 **ParticleLightSystem**을 드래그하여 레벨에 배치하자.
위치는 (300,0,50)에 배치하자.
플레이해보자.

7. 이제부터 파티클 라이트 효과가 되도록 수정해가면서 만들어보자.

이미터 업데이트 그룹, **파티클 스폰** 그룹, **파티클 업데이트** 그룹의 순서로 각 그룹 내의 모듈에 대한 세팅을 수정해나갈 것이다.
가장 먼저, **이미터 오버뷰** 노드에 있는 **이미터 업데이트** 그룹에서의 세팅을 시작해보자.
이 모듈 중에서 **Emitter State** 모듈을 선택하고 오른쪽 **선택** 탭으로 가자.
Life Cycle Mode 속성값이 **Self**로 되어있다. 이 속성값을 **System**으로 수정하자. **System**으로 수정하게 되면 시스템에서 라이프사이클 세팅을 계산하게 되어서 성능의 최적화에 더 유리하게 된다.



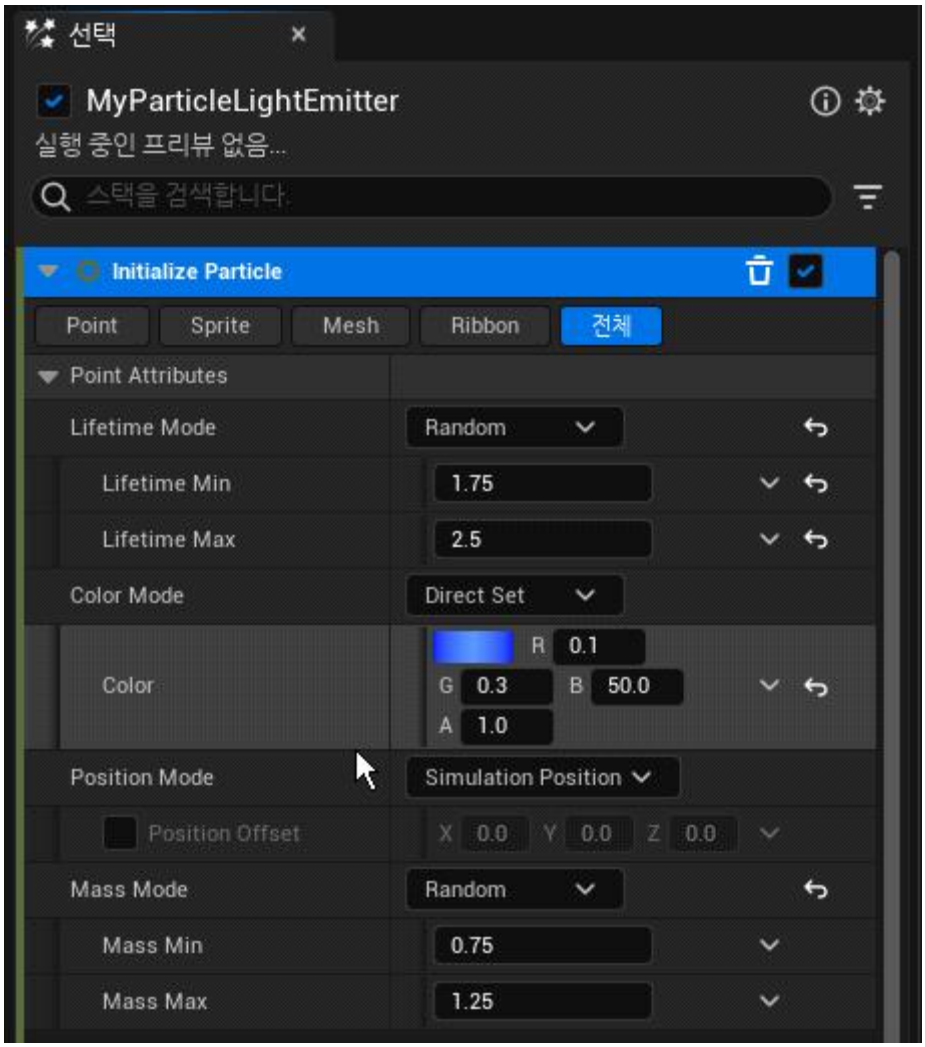
8. 다음으로, **이미터 오버뷰** 노드에서 **Spawn Rate** 모듈을 선택하자.
Spawn Rate 모듈은 이미터가 활성인 동안에 계속해서 파티클 스트림을 생성한다.
오른쪽 **선택** 탭에서 **SpawnRate** 속성값을 90에서 500으로 수정하자.



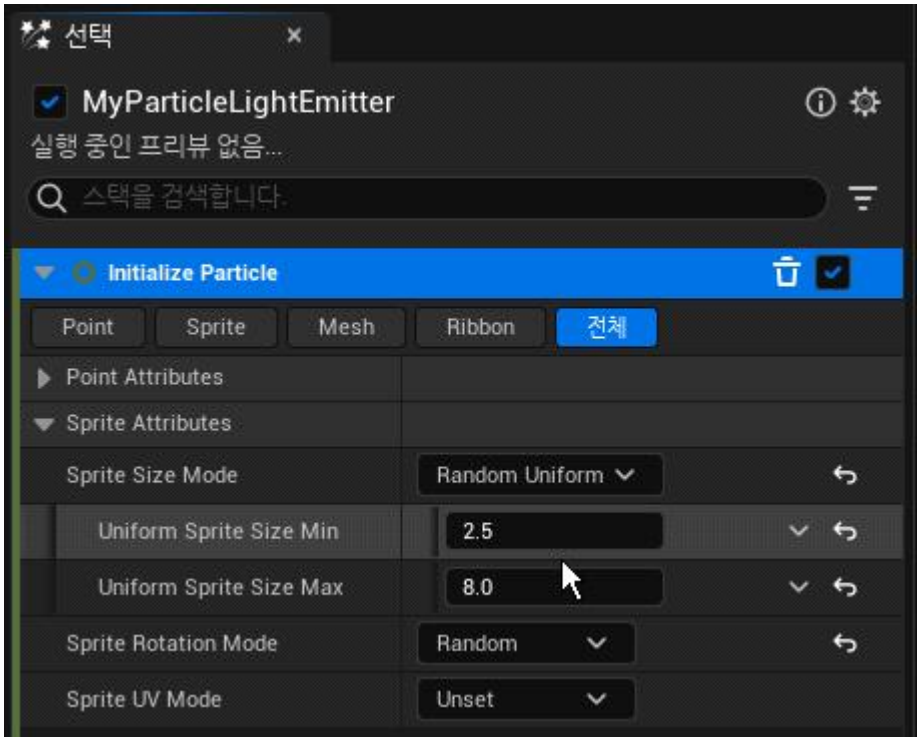
지금까지, **이미터 업데이트** 그룹에 대한 세팅을 모두 완료하였다.

9. 이제부터는, **파티클 스폰** 그룹에 대한 세팅을 시작하자.
파티클 스폰 그룹에 있는 **Initialize Particle** 모듈을 선택하자. 이 모듈은 파티클의 초기화와 관련된 다양한 속성값을 가지고 있다.
먼저, **Point Attributes** 영역 아래의 **LifetimeMode** 속성값이 **Random**으로 되어 있다. 파티클의 수명이

랜덤하게 지정되도록 되어 있다. 바로 아래에 **Lifetime Min**과 **Lifetime Max**를 1.4와 1.75에서 1.75, 2.5로 수정하자. 수명이 더 늘어날 것이다.
그다음, 바로 아래의 **Color** 속성값을 보자. 디폴트로 RGBA가 (1,1,1,1)로 되어 있을 것이다. 이것을 (0.1, 0.3, 50, 1)로 수정하자. 이 컬러는 파티클이 스폰될 때의 초기 컬러를 지정한다. RGB 값은 기본적으로 [0,1]의 범위이지만 1을 초과하는 경우에는 이미시브 컬러(emissive color)로 작동되어 스스로 빛을 내는 효과를 나타낸다.

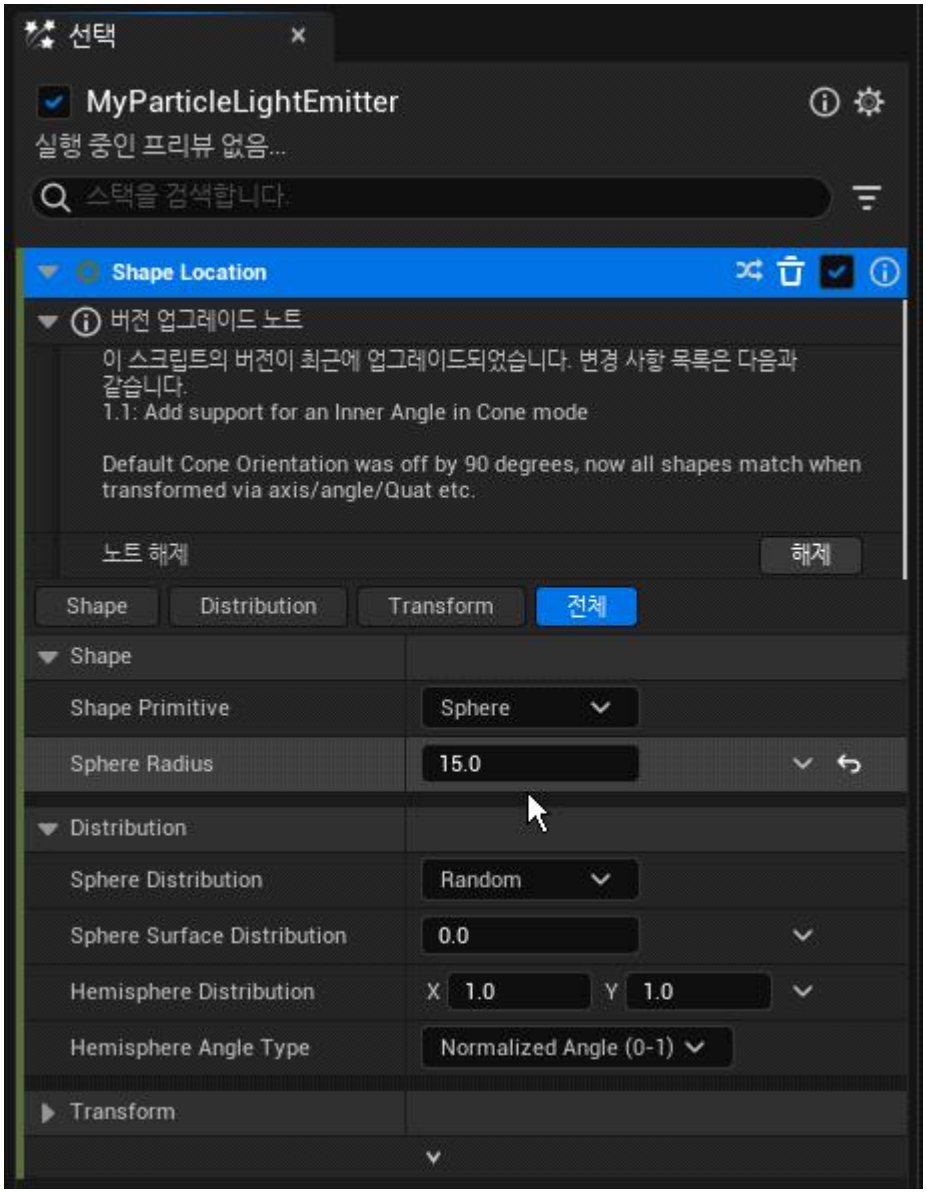


10. 이제 스프라이트의 크기를 조절해보자. 현재의 파티클의 크기는 너무 커서 작게 바꾸어보자. 바로 아래의 **Sprite Attributes** 영역에서 **Sprite Size Mode**를 **Random Uniform**으로 되어 있을 것이다. 그 아래에 나타나는 **Uniform Sprite Size Min**과 **Uniform Sprite Size Max**을 6과 12에서 2.5와 8로 수정하자.



플레이해보자. 스프라이트의 작아졌을 것이다.

11. 그다음, **Shape Location (Sphere)** 모듈을 선택하고 **선택** 탭에서 **Sphere Radius** 속성값을 8에서 15로 수정하자. 파티클이 스폰되는 위치의 범위가 커졌을 것이다.



12. 그다음, **Add Velocity (In Cone)** 모듈을 선택하자. 이 모듈은 파티클이 스폰될 때의 속도를 추가한다. 콘 모양에서의 꼭지점이 파티클의 스폰 위치이다. 콘이 펼쳐지면서 향하는 방향은 X,Y,Z값으로 지정하면 된다. 속도의 세기인 **Velocity Strength**는 랜덤하게 결정되도록 지정되어 있다.
우리는 **Velocity Speed**의 **Minimum**, **Maximum** 속성값을 500,850에서 300,600으로 수정하자.

지금까지, **파티클 스폰** 그룹에 대한 세팅을 모두 완료하였다.

13. 이제부터는, **파티클 업데이트** 그룹에 대한 세팅을 시작하자. 이 그룹의 모듈들은 파티클이 매 프레임 갱신될 때마다 실행되는 모듈들이다.
Gravity Force 모듈이 있다. 이 모듈은 중력이 물체에 영향을 주는 것을 구현한다.
Drag 모듈이 있다. 이 모듈은 파티클을 끌어당겨서 느려지게 만든다.
이 두 모듈은 그대로 두자.

14. 한편, 현재는 콜리전이 설정되어 있지 않아서 파티클이 바닥 아래로 뚫고 떨어진다. 따라서 콜리전을 추가해서 충돌이 일어나도록 해보자.
파티클 업데이트 그룹의 오른쪽의 + 아이콘을 클릭하고 **Collision** 모듈을 추가하자.
지금까지, **파티클 업데이트** 그룹에 대한 세팅을 모두 완료하였다.

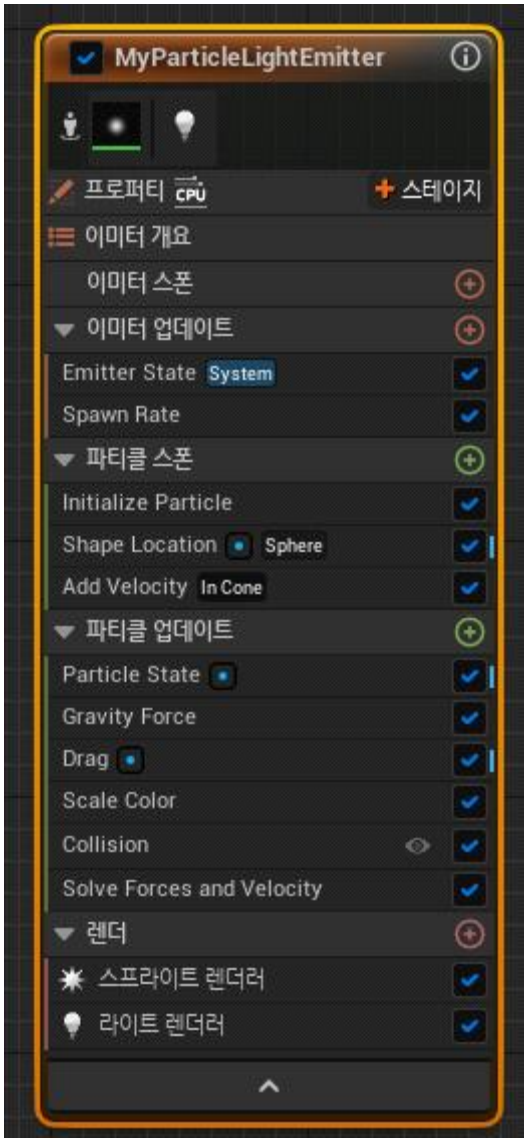
15. 이제부터는, **렌더** 그룹에 대한 세팅을 시작하자.
렌더 그룹에서 오른쪽의 + 아이콘을 클릭하고 목록에서 **라이트 렌더러**(**Light Renderer**) 모듈을 선택하

여 추가하자.

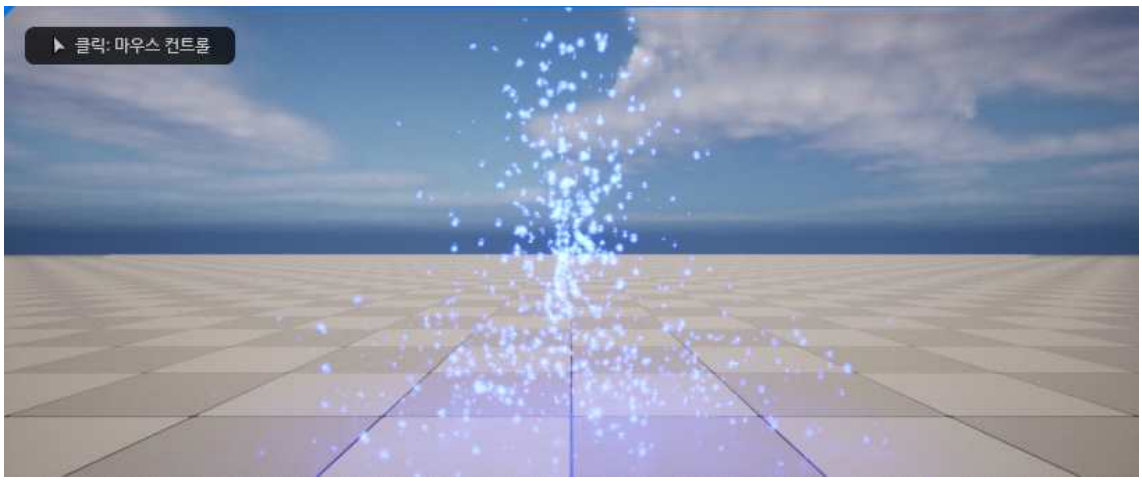
추가된 **라이트 렌더러** 모듈을 선택하고, **선택** 탭에서 **반경 스케일(Radius Scale)**을 찾자. 이 값은 파티클 스폰 위치에서부터 빛이 얼마나 멀리 퍼지는지에 대한 정보를 결정한다. 디폴트인 1을 5로 수정하자. 그다음, 그 아래의 **컬러 추가(Color Add)**를 찾자. 이 값은 이미터로부터 방출되는 조명의 컬러를 바꿀 수 있도록 한다. X,Y,Z로 되어 있지만 R,G,B에 해당하는 값이다. 우리는 값을 0,0,0에서 0,0,15로 수정하여 파티클의 컬러와 어울리도록 파란색 조명이 되도록 하자.

이제 렌더 세팅을 완료하였다.

16. 이제 이미터를 모두 완성하였다. **이미터 오버뷰** 노드는 다음과 같이 되었을 것이다.



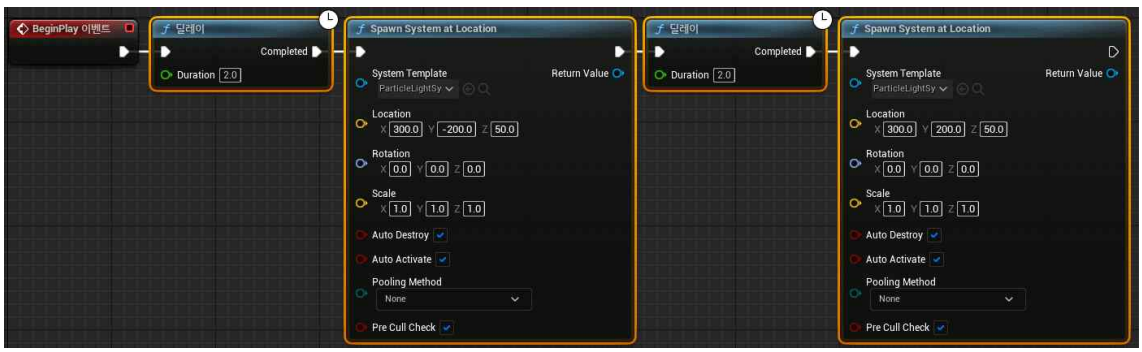
17. 레벨 에디터로 가서 플레이해보자.



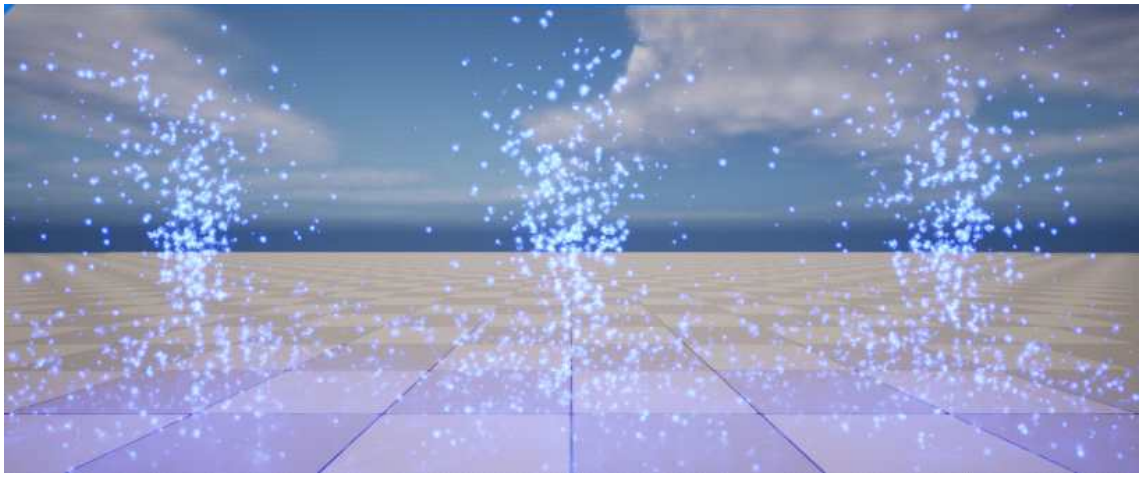
18. 광원을 끄고 파티클 라이트 이펙트를 확인해보자. 아웃라이너에서 **DirectionalLight**의 왼쪽에

있는 눈 모양의 **Visibility** 아이콘을 클릭하여 일시적으로 숨겨보자. 아래와 같이 표시될 것이다. 테스트한 후에는 다시 원래대로 해두자.

19. 이번에는 파티클 시스템을 플레이 중에 동적으로 생성하는 것에 대해서 알아보자.
동적으로 생성하는 방법은 **Spawn System at Location** 함수를 사용하여 배치하기 원하는 위치에 시스템을 스폰하면 된다.
먼저, 레벨 에디터의 툴바에서 **블루프린트** » **레벨 블루프린트 열기**를 선택해서 레벨 블루프린트 창을 열자.
그다음, **BeginPlay 이벤트** 노드를 드래그하고, **Delay** 노드를 검색하여 배치하자. 노드의 **Duration** 입력핀에 2를 지정하자.
그다음, **Delay** 노드를 드래그하고 **Spawn System at Location** 노드를 검색하여 배치하자. 그리고, **System Template** 입력핀을 클릭하고 **Particle Light System**을 선택하자. 그리고, **Location** 입력핀에 (300,-200,50)을 입력하자.
그다음, 이전의 **Delay** 노드와 **Spawn System at Location** 노드를 **Ctrl+C**를 누르고 **Ctrl+V**를 눌러 복사하여 마지막 노드 뒤에 연결하자. 그리고, 두 번째 **Spawn System at Location** 노드의 **Location** 입력핀에 (300,200,50)으로 수정하자.
저장하고 컴파일하자.



20. 플레이해보자. 처음에 중간의 파티클 효과가 나타나고 2초 후에, 왼쪽 뒤에 추가로 파티클 효과가 나타날 것이다. 다시 2초 후에 오른쪽에도 추가로 파티클 효과가 나타날 것이다.



이 절에서는 라이트를 방출하는 파티클 라이트 이펙트에 대해서 학습하였다.

□