26_ 라이팅

<제목 차례>

26_ 라이팅 ····· 1
1. 개요 ····· 2
2. 디폴트 레벨에서의 라이팅 학습 ····· 3
3. 다양한 라이트를 사용한 건물 만들어보기 ····· 11

인천대학교 컴퓨터공학부 박종승

1. 개요

이 장에서는 라이팅에 대해서 다룬다.

장면에는 라이트가 필요하다. 라이트가 없다면 깜깜한 어둠처럼 물체가 있어도 아무것도 보이지 않는다. 기본적인 작업을 위해서는 디렉셔널 라이트를 한두개 추가해주면 대부분 해결된다. 그러나 장면의 완성도를 높이려면 본격적인 라이팅 작업이 필요하다.

라이트(light)와 관련된 여러 작업이나 기법을 라이팅(lighting)이라고 한다. 라이트를 레벨에 배치하여 환하게 비춘다거나, 광택이 나도록 한다거나, 그림자가 지도록 하는 등의 포괄적인 것들을 포함한다.
라이팅은 레벨의 보이는 모습의 결정에 있어서 매우 중요하다. 라이팅 작업은 물체를 더욱 현실감있게 보이도록 해준다. 또한 그림자로 물체의 존재감을 더해줄 수 있고, 조명을 비추어 장면에 특정한 분위기를 만들 수도 있다.

라이트에는 대표적으로 디렉셔널 라이트, 포인트 라이트, 스포트 라이트가 있다. 디렉셔널 라이트는 밝은 장면을 넓게 비추는 광역 조명에 해당한다. 포인트 라이트와 스포트 라이트는 어두운 장면을 좁게 비추는 국부 조명에 해당한다.

<참고> 라이팅의 전체적인 내용에 대해서는 다음의 문서를 참조하자.
https://docs.unrealengine.com/lighting-the-environment-in-unreal-engine/

2. 디폴트 레벨에서의 라이팅 학습

이 절에서 디폴트 레벨에 배치되어 있는 라이팅 기능에 대해서 학습한다.

우리는 지금까지 프로젝트를 생성할 때에 템플릿 목록에서 **기본** 템플릿을 선택하고 프로젝트를 생성하였다. 프로젝트가 생성되면 **디폴트 레벨**이 만들어진다. **디폴트 레벨**에는 라이팅과 관계된 많은 액터들이 배치된다. 이 절에서는 디폴트 레벨의 라이팅 관련 액터에 대해서 알아본다.

가장 기본적인 라이트인 **디렉셔널 라이트**에 대해서 알아보자. 여러 라이트 종류 중에서 **디렉셔널 라이트**는 흔히 실외에서 태양의 조명 용도로 사용된다. 디렉셔널 라이트(directional light)는 광원의 방향만 중요하고 위치는 의미를 가지지 않는 라이트이다. 하나만 배치해도 맵 전체의 시야를 확보할 수 있으며 감쇠하지 않고 맵의 모든 곳에 빛을 보낸다.
디렉셔널 라이트의 디테일 탭에서 라이트 영역의 **강도(Intensity)** 속성값은 빛이 발산되는 총 에너지를 정의한다.

<참고> **디렉셔널 라이트**의 각 속성값에 대해서는 다음의 문서를 참조하자.
https://docs.unrealengine.com/directional-lights-in-unreal-engine/

이제부터는, 대기를 표현하는 방법에 대해서 알아보자.
태양에 해당하는 디렉셔널 라이트를 추가한 후에도 하늘이 여전히 깜깜하게 보일 것이다. 그 원인은 태양 빛이 하늘에서 산란하는 대기 효과를 표현하지 못하기 때문이다. 엔진은 대기에서의 빛의 산란 효과를 표현하는 여러 액터를 제공한다. 이 액터들은 지구 대기권을 통과하는 빛의 산란 효과를 추정해서 야외에서의 대기 표현에 사실감을 더해준다.
대표적으로 **SkyAtmosphere** 액터가 있다. 이 액터는 태양에 해당하는 방향 광원에 의한 대기의 산란(scattering) 효과를 표현한다. **SkyAtmosphere** 액터는 태양의 현재 상태에 연동하여 대기를 표현한다. 태양의 상태에 따라 아침의 여명과 한낮의 밝은 햇빛과 저녁의 노을과 자정의 어두운 밤하늘로 변한다. 태양의 상태는 레벨에 배치된 디렉셔널 라이트로부터 받아온다. 태양의 방향만 사용되고 위치는 무시된다. 이때 디렉셔널 라이트의 **Atmosphere Sun Light**가 체크되어 있어야 방향 정보를 받아올 수 있다.

<참고> 대기에서의 포그나 구름이나 하늘의 표현에 대한 자세한 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/environmental-light-with-fog-clouds-sky-and-atmosphere-in-unreal-engine/
SkyAtmosphere 액터의 자세한 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/sky-atmosphere-component-in-unreal-engine/

이제부터는, 간접광을 포함한 전체적으로 라이팅 효과를 표현하는 방법을 알아보자.
라이팅에서는 광원으로부터 나오는 광선이 물체의 표면으로 곧바로 들어가서 표면이 보이도록 하는 직접광만 있는 것이 아니다. 표면에 들어간 광선은 다시 반사되어 나와서 다른 표면을 비추게 되는 등의 수많은 반사를 통해서 전체적으로 라이팅 효과를 만들어낸다. 이러한 반사 특성을 실제와 유사하게 재현하는 액터를 추가해주면 더욱 사실적인 모습을 만들 수 있다.
간접광이 없다면 그림자도 직접광으로부터만 만들어질 것이므로 그림자가 매우 짙게 나타날 것이다. 그러나 현실 세계에서는 다양한 간접광으로 인하여 그림자가 어둡지 않고 부드럽게 나타난다. 엔진에서는 이러한 복잡한 반사 특성을 큐브맵을 사용하여 재현하는 액터를 제공한다. 바로 **스카이라이트(SkyLight)** 액터가 이 기능을 제공한다. 이 액터는 하늘이나 간접광의 효과를 자연스럽게 표현

한다.

<참고> **스카이 라이트**(Sky Light)에 대해서는 다음의 문서를 참조하자.
https://docs.unrealengine.com/sky-lights-in-unreal-engine/

이제부터는, **리플렉션 캡처**에 대해서 알아보자.
이전에서 하늘이나 넓은 영역에서의 전체적인 간접광의 표현을 위해서 **스카이 라이트**(**SkyLight**) 액터를 사용한다고 하였다.
한편, 아주 먼 거리나 영역에서의 간접광뿐만 아니라 국부적인 지점에 대해서도 간접광을 표현할 수 있다. 이 기능을 **리플랙션 캡처**라고 한다. **리플랙션 캡처** 액터는 표면의 반사 특성을 표현하는 액터이다. 하늘이나 간접광의 효과를 자연스럽게 표현한다. 모든 표면은 반사 특성을 가지고 있다. 언리얼에서는 큐브맵을 사용하여 이러한 반사 특성을 유사하게 재현한다.
이전에서 설명한 **SkyLight** 액터도 이러한 **리플랙션 캡처** 액터와 동일한 원리로 작동한다.
SkyLight 액터는 원거리 반사 특성을 표현한다. 즉, 레벨의 먼 부분을 캡처하여 그 부분을 씬에 라이트로 적용한다. 특히 하늘 표현을 캡처하여 스카이 큐브맵을 만든다.
한편, **ReflectionCapture**는 근거리 반사 특성을 표현한다. 즉, 특정 위치에 배치하면 그 위치에서의 국부적인 큐브맵을 캡처한다. 따라서 특정 위치에 배치해두면 그 위치에서의 반사의 정확성이 향상된다. **ReflectionCapture**에는 **SphereReflectionCapture** 엑터와 **BoxReflectionCapture** 액터의 두 종류가 있다. 씬에서의 각 재질은 자신에서 일정 거리 이내에 있는 캡처 액터 중에서 가장 가까이 있는 캡처 액터의 큐브맵을 참조한다. 만약 일정 거리 이내에 있는 캡처가 없다면 **SkyLight** 큐브맵을 참조한다. **SkyLight**가 없으면 아무거나 가장 가까이 있는 캡처 액터를 참조한다.

<참고> 리플랙션 캡처와 관련된 더 구체적인 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/reflections-captures-in-unreal-engine/
환경 매핑과 스카이 라이트에 대해서는 다음의 문서를 참조하자.
https://lifeisforu.tistory.com/375
https://lifeisforu.tistory.com/377

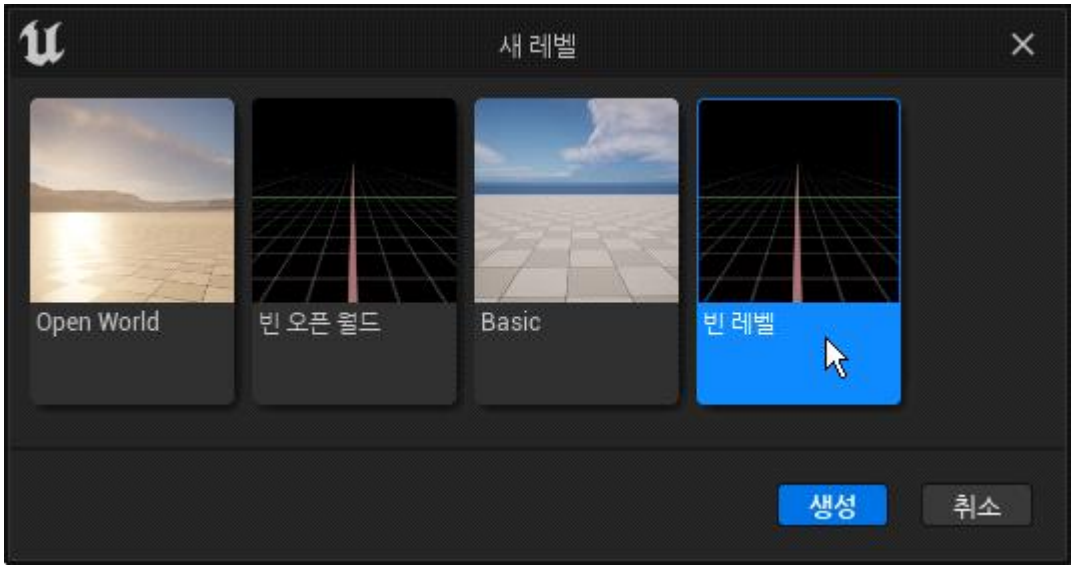
이제부터 예제를 통해서 학습해보자

---

1. 새 프로젝트 **Pdirlight**를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Pdirlight로 입력하고 **생성** 버튼을 클릭하자. 프로젝트가 생성되고 언리얼 에디터 창이 뜰 것이다. 레벨 에디터가 뜨면 **디폴트 레벨**이 준비되어 있을 것이다.
디폴트로 생성되는 레벨은 **오픈 월드**(**Open World**) 템플릿의 레벨이다.
레벨의 템플릿은 **오픈 월드**(**Open World**), **빈 오픈 월드**, **베이직**(**Basic**), **빈 레벨**의 네 가지가 있다. **베이직**과 **빈 레벨**은 고전적인 형태이고 **오픈 월드**와 **빈 오픈 월드**는 **베이직**과 **빈 레벨**의 오픈 월드 버전이다. 오픈 월드는 자동으로 월드 공간을 격자로 나누고 필요한 셀을 스트리밍하는 UE의 최신 기술이다.
메뉴바에서 **파일** » **새 레벨**을 선택하여 각각의 템플릿을 살펴보자.

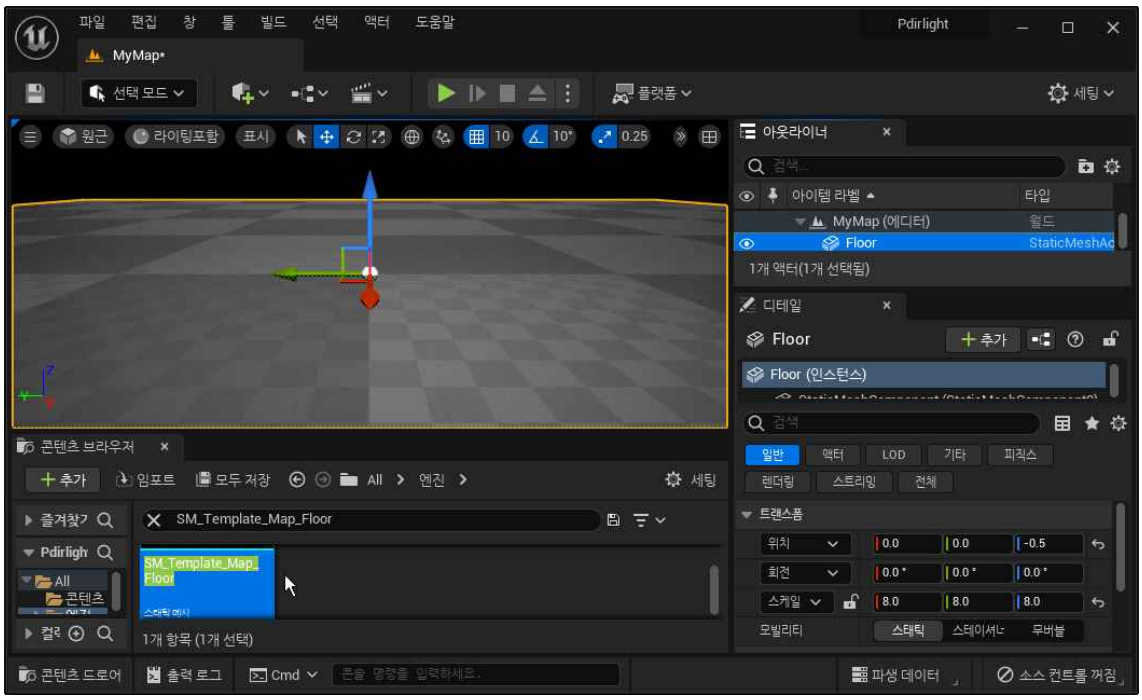
2. 레벨 에디터가 뜨면 디폴트 레벨인 **오픈 월드** 레벨이 준비되어 있을 것이다. 우리는 이 디폴트 레벨을 사용하지 말고 **빈 레벨**에서 라이팅과 관련된 액터를 하나씩 추가하면서 배워보자.
빈 레벨에서 시작해서 이 예제가 모두 수행된 후에는 레벨이 **Basic** 레벨과 유사하게 바뀌게 될 것이다.

메뉴바에서 **파일** » **새 레벨**을 선택하자. 아래와 같은 템플릿 선택 창이 뜬다. **빈 레벨**을 선택하자. 이제 아무 액터가 없는 깜깜한 레벨이 보일 것이다.



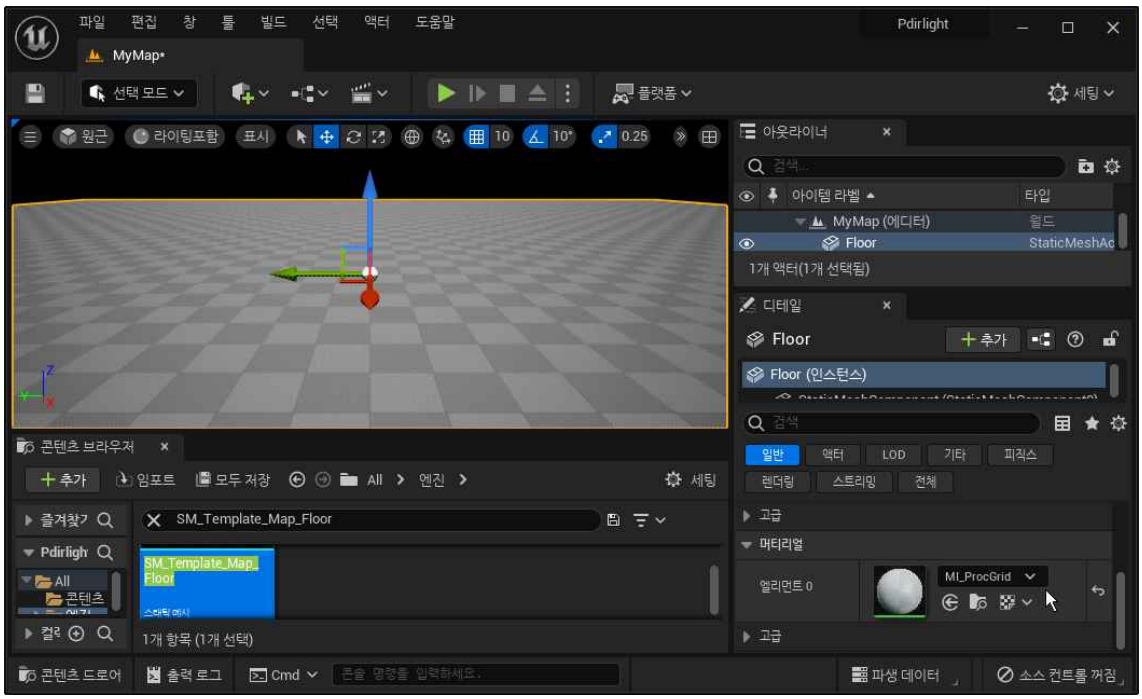
이제, 레벨 에디터 툴바의 **저장** 버튼을 클릭하여 현재의 레벨을 MyMap으로 저장하자.
그리고, **프로젝트 세팅** 창에서 **맵&모드** 탭에서의 **에디터 시작 맵** 속성값을 MyMap으로 수정하자.
뷰포트에 깜깜한 화면만 보일 것이다.

3. 먼저, **Basic**에 레벨에 있었던 라이팅과 무관한 **Floor** 액터를 배치하자.
먼저, **콘텐츠 브라우저**에서 **엔진** 폴더를 선택하고 **SM_Template_Map_Floor**를 검색하여 스태틱 메시를 찾고 이를 레벨에 드래그하여 배치하자. 배치된 액터의 이름을 **Floor**로 수정하자. 위치를 (0,0,-0.5)로 하고 스케일을 (8,8,8)로 수정하자.

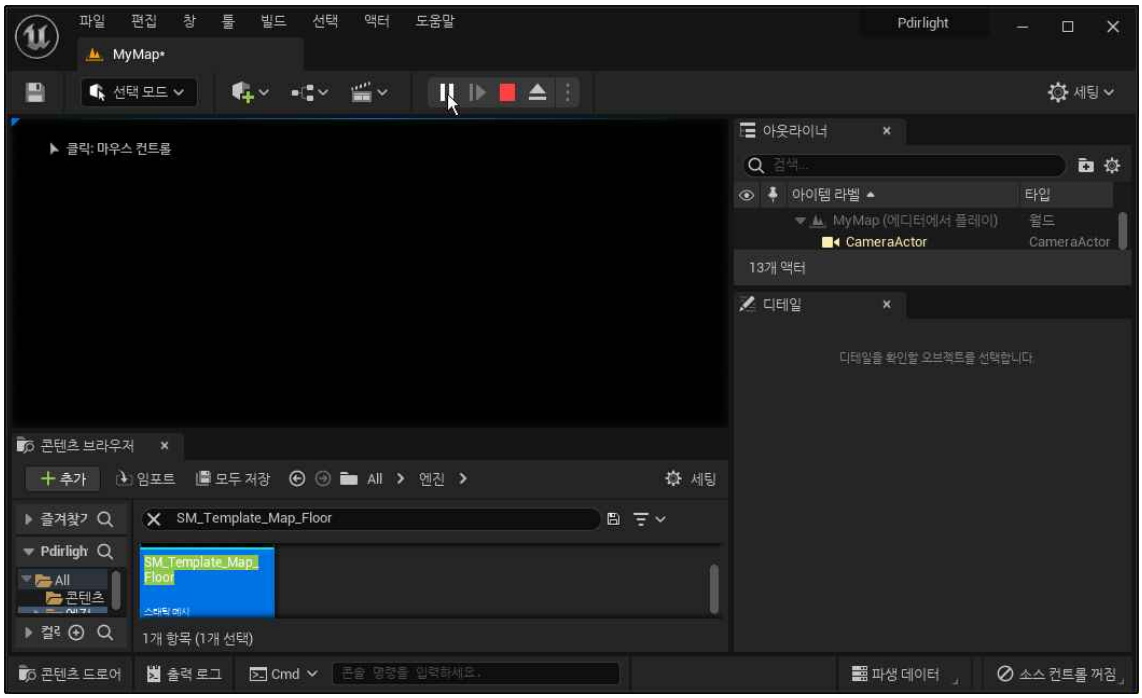


4. 다음으로, **Floor**의 디테일 탭의 **머티리얼** 카테고리에서 **엘리먼트 0**의 머티리얼이 **DefaultMaterial**로

되어 있을 텐데, 이것을 **MI_ProcGrid**로 바꾸자. 격자가 더 촘촘하게 보일 것이다.



5. 플레이해보자. 아직 아무것도 보이지 않는 깜깜한 화면만 보일 것이다.

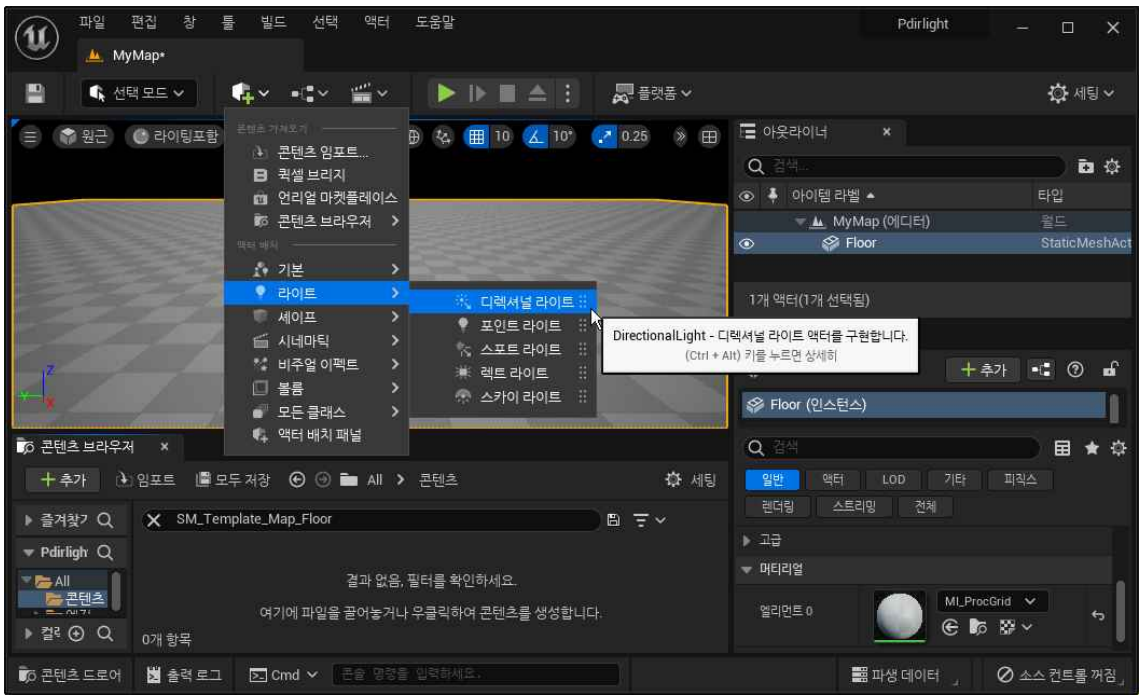


6. 이제 라이트를 추가해보자.

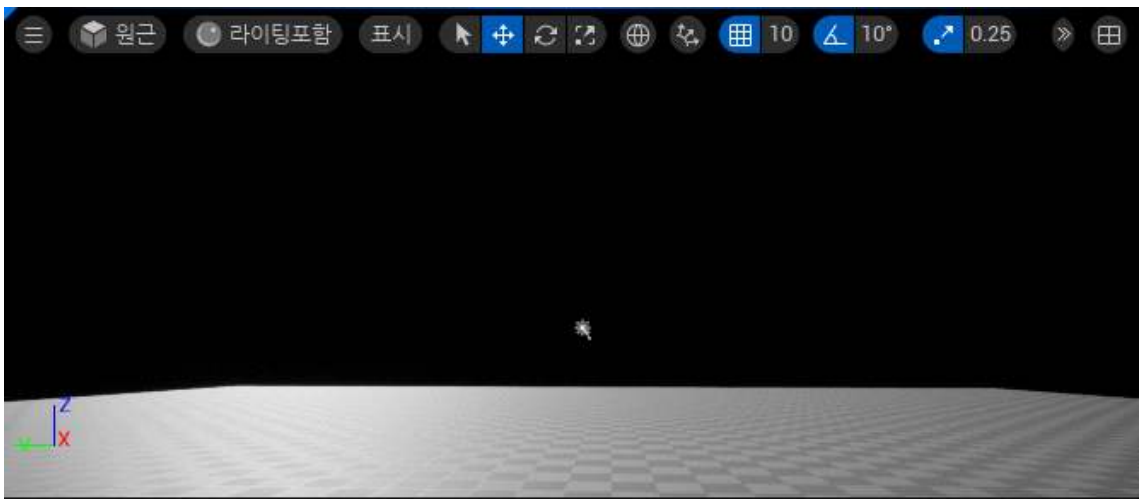
라이트를 추가하는 방법은 **액터 배치** 탭에서 **라이트** 탭에 나열되어 있는 라이트 액터를 드래그하여 추가하면 된다. 또다른 방법으로, 툴바의 **액터 배치** 아이콘을 클릭하고 드롭다운 메뉴에서 **라이트** 아래에 있는 라이트 액터를 드래그하여 추가하면 된다.

우리는 툴바의 **액터 배치** 아이콘을 클릭하고 **라이트** 아래의 **디렉셔널 라이트** 액터를 드래그하여 추가

하자.

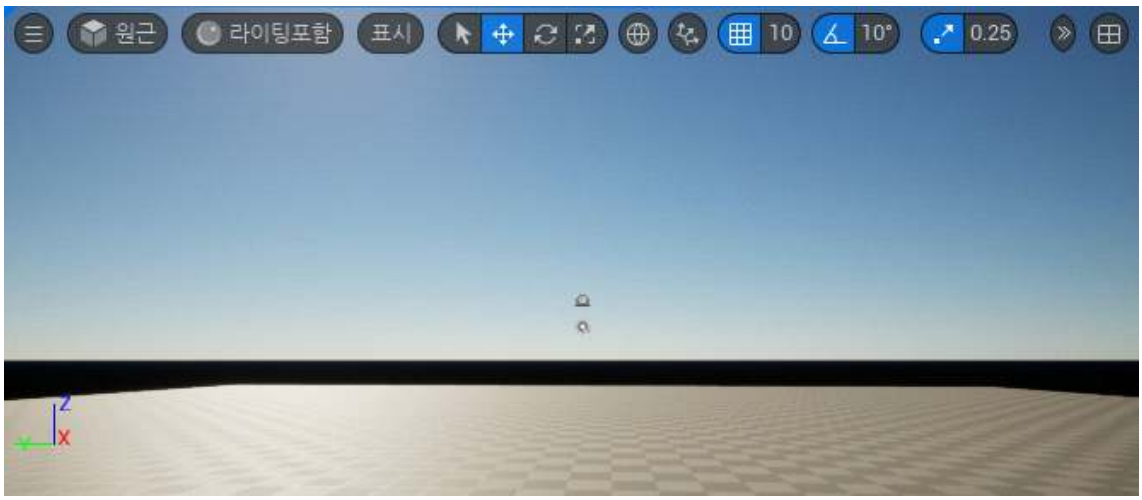


7. **디렉셔널 라이트** 액터가 추가되었을 것이다. 이름은 그대로 **DirectionalLight**로 두자. 디테일 탭에서 위치는 (0,0,400)로 수정하자. 회전은 (112,-38,-31)으로 수정하자. 이렇게 하는 이유는 **Basic** 템플릿 레벨에서의 디렉셔널 라이트와 유사하게 맞추기 위함이다. **모빌리티**는 **무버블**로 수정하자.
그다음, 디테일 탭에서 **애트머스피어 및 클라우드** 영역의 **애트머스피어 썬 라이트**(**Atmosphere Sun Light**)를 확인해보자. 체크되어 있어야 한다. 디폴트로 체크되어 있을 것이다.
Atmosphere Sun Light에 체크되어 있으면 이 라이트는 태양의 역할을 한다. 이 옵션은 디렉셔널 라이트에만 있다. 체크된 경우에는 이 액터를 회전시켜 태양의 위치를 정의한다. 두 개 이상의 디렉셔널 라이트에 지정되어 있는 경우에는 첫 번째로 탐색되는 디렉셔널 라이트만 태양의 위치로 사용된다.
이제, 플레이해보자. 이제는 바닥이 보일 것이다.

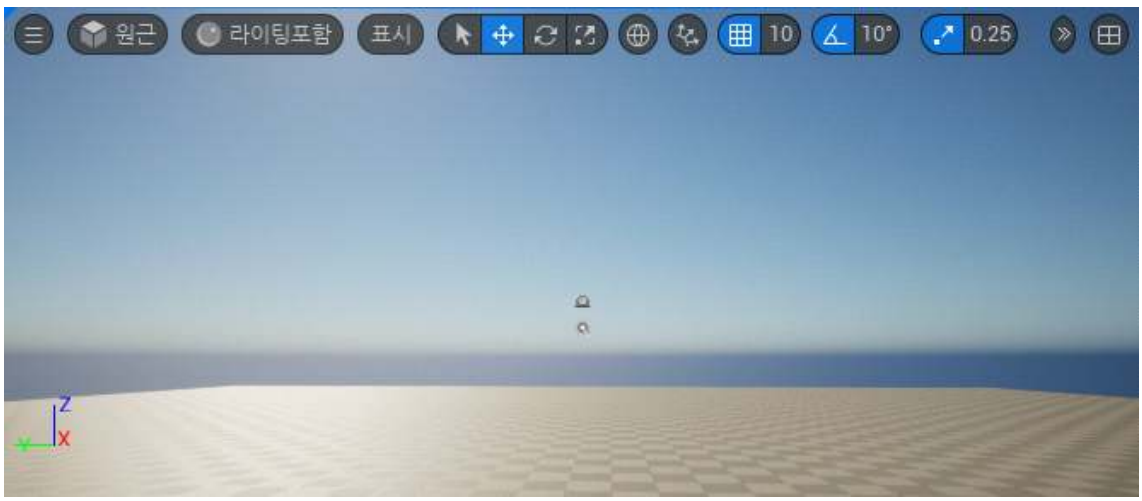


8. 그러나 하늘은 여전히 깜깜할 것이다. 하늘이 깜깜하게 보이는 것을 해결해보자.
대기에서의 빛의 산란 효과를 표현하면 하늘도 보이게 된다.

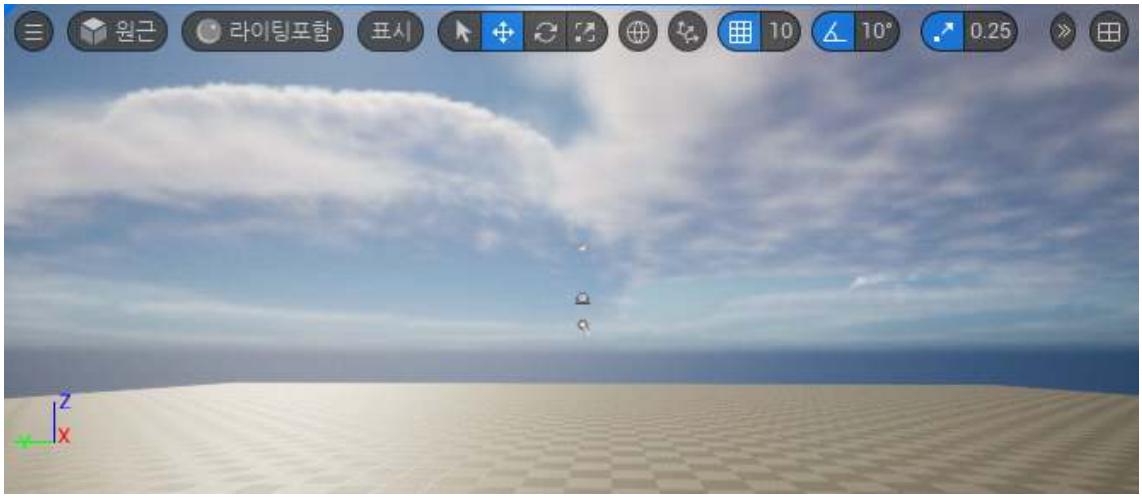
먼저, 하늘에 대기를 추가하자. **액터 배치** 탭에서 **비주얼 이펙트** 탭을 살펴보자. 이 탭에 있는 **스카이 애트머스피어** 액터를 드래그해서 레벨에 배치하자. **SkyAtmosphere** 액터가 추가될 것이다. 위치는 (0,0, 500)으로 수정하자. 사실 위치값과 회전값은 사용되지 않는다.
이제, 지면 위로 파란 하늘이 보일 것이다.



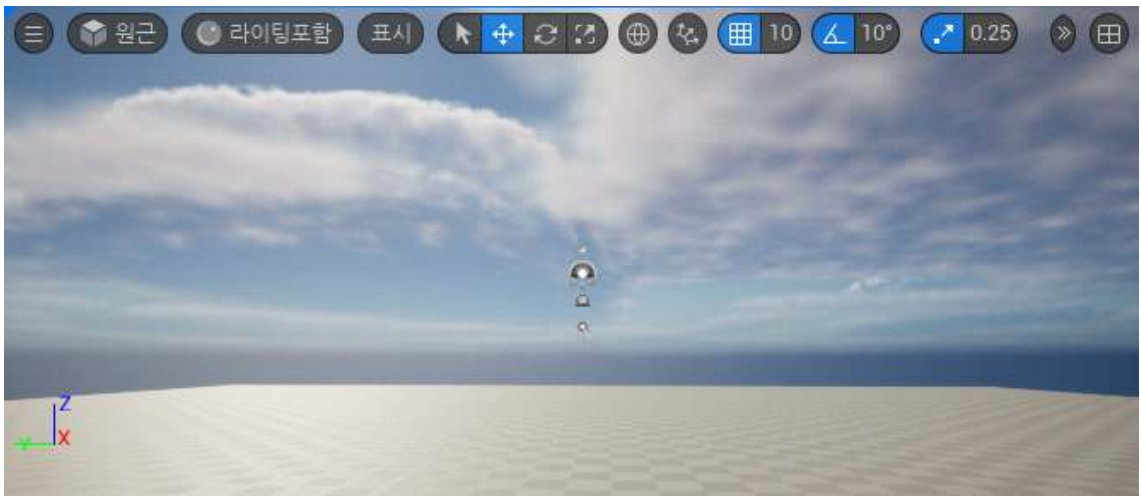
9. 그다음, 포그 효과도 추가하자. **액터 배치** 탭에서 **비주얼 이펙트** 탭에 있는 **익스포넨셜 하이트 포그** 액터를 드래그해서 레벨에 배치하자. **ExponentialHeightFog** 액터가 추가될 것이다. 위치는 (-5600,-50,-6850)으로 수정하자.
이제, 포그 효과로 인하여 지면의 수평선 이하의 영역에서도 깜깜하지 않게 보일 것이다.



10. 그다음, 구름도 추가하자. **액터 배치** 탭에서 **비주얼 이펙트** 탭에 있는 **볼류메트릭 클라우드** 액터를 드래그해서 레벨에 배치하자. **VolumetricCloud** 액터가 추가될 것이다. 위치는 (0,0,700)으로 수정하자.
이제, 움직이는 구름 효과가 추가되어 하늘이 더 자연스럽게 보일 것이다.



11. 간접광을 포함한 전체적으로 라이팅 효과를 추가해보자.
액터 배치 탭에서 **라이트** 탭에 있는 **스카이 라이트** 액터를 드래그해서 레벨에 배치하자. **SkyLight** 액터가 추가될 것이다. 간접광 효과로 인해 바닥이 더 밝아질 것이다.
디테일 탭에서 위치는 (0,0,600)로 수정하자. **모빌리티**는 **무버블**로 수정하자.



지금까지, **빈 레벨**에서 시작하여 **베이직(Basic)** 템플릿 레벨과 비슷하게 되도록 만들어보았다.

이 절에서는 디폴트 레벨에 배치되어 있는 라이팅 기능에 대해서 학습하였다.

<참고> 엔진에는 하늘을 표현하는 스태틱 메시인 **SM_SkySphere**가 있다. 이 메시는 하늘을 위한 구체 메시와 머티리얼을 제공한다. 하늘 표현에 대한 또다른 고급 기능을 사용할 경우에 이 메시를 함께 사용할 수 있다.

<참고> 엔진에는 또한 하늘의 시각적인 모습을 잘 표현해주는 블루프린트 액터인 **BP_Sky_Sphere**가 있다. 이 액터는 태양뿐만 아니라 구름이나 별과 같은 하늘의 시각적인 모습을 표현해준다.
액터의 디테일 탭에서 디폴트 영역에서 **Directional Light Actor** 속성값이 **없음**으로 되어 있는데 이것을 레벨에 배치된 디렉셔널 라이트 액터로 수정해주어야 한다. 이렇게 하면 태양의 위치를 나타내는 디렉셔널 라이트와 연결되어 그 방향을 읽어와서 그에 맞도록 하늘의 모양을 표현한다.
한편, 우리는 이 액터를 사용하지 않고도 이미 멋진 하늘을 이미 표현하였다. 따라서 우리는 이 액터를 추가로 사용할 필요는 없다.

3. 다양한 라이트를 사용한 건물 만들어보기

이 절에서 다양한 종류의 라이트를 사용해서 건물 내부의 라이팅 방법에 대해서 학습한다.
디폴트로 추가되는 디렉셔널 라이트와 더불어 포인트 라이트와 스포트 라이트를 추가해볼 것이다.

<참고> 이 절의 예제는 다음의 문서를 참조하였다.
https://docs.unrealengine.com/4.27/BuildingWorlds/LightingAndShadows/QuickStart/

이제부터 예제를 통해서 학습해보자

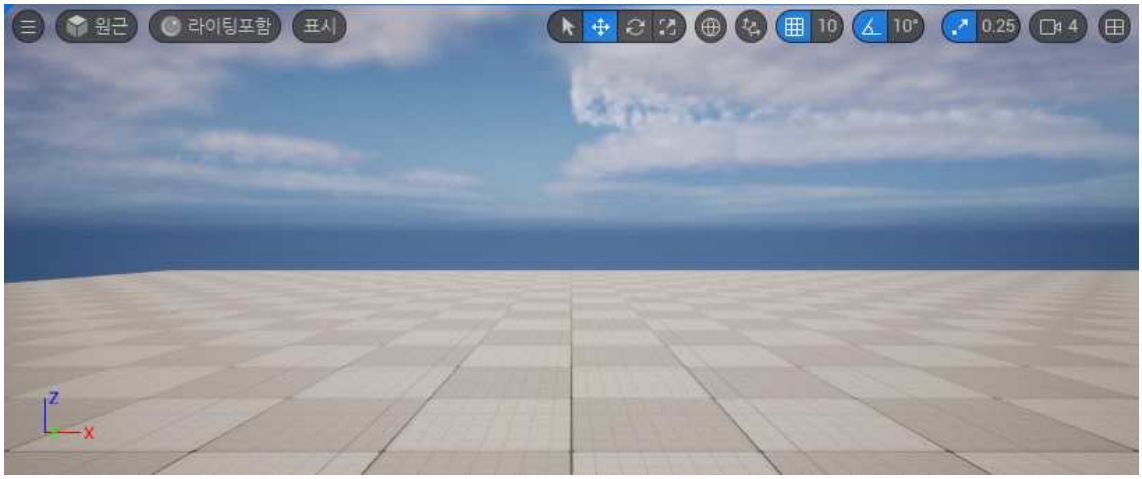
---

1. 새 프로젝트 Phouselight를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Phouselight로 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.
그리고, **아웃라이너**에서 **Floor**를 선택하고, **디테일** 탭에서 **스케일**을 (8,8,8)에서 (1,1,1)로 수정하자.

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/Architecture/Floor_400x400.uasset
StarterContent/Architecture/Wall_400x400.uasset
StarterContent/Architecture/Pillar_50x500.uasset
StarterContent/Architecture/Wall_Door_400x400.uasset
StarterContent/Architecture/Wall_Window_400x400.uasset
StarterContent/Materials/M_Basic_Floor.uasset
StarterContent/Materials/M_Basic_Wall.uasset
```

3. 기존에 배치된 액터의 배치를 수정하자.
먼저, 바닥이 너무 작으니 충분히 크게 하자. **Floor** 액터의 위치를 (0,0,-0.5)에서 (0,0,-20)으로 수정하자. 그리고, 스케일을 (3.5, 3.5, 3.5)로 수정하자.
그다음, **Player Start** 액터의 위치를 기존의 (0,0,92)에서 (-300,300,92)로 수정하자. 회전을 (0,0,0)에서 (0,0,-90)으로 수정하자. 이렇게 하면 우리가 만들 건물 내부에서 방 쪽을 바라보며 배치된다.



4. 이제부터 건물을 만들어보자.

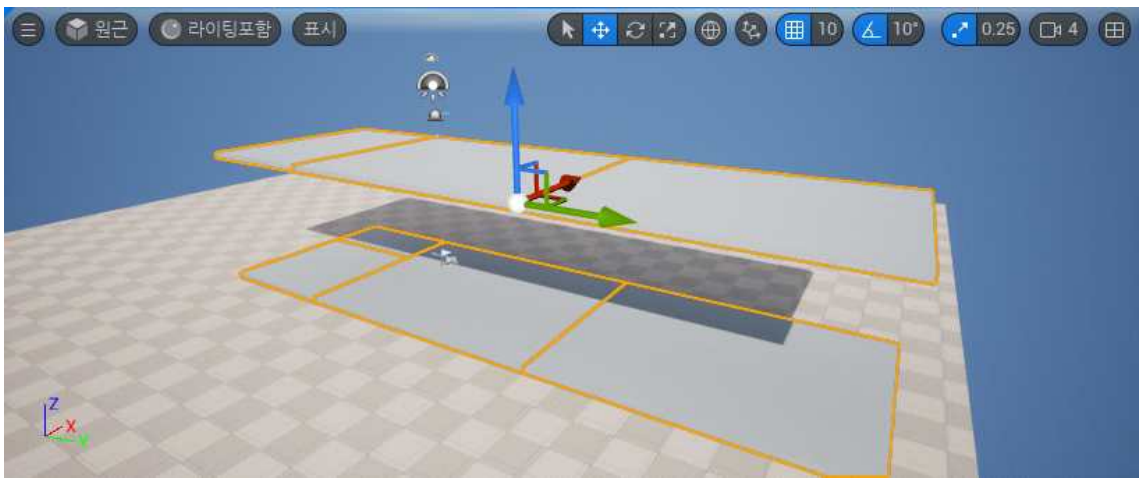
콘텐츠 브라우저에서 스태틱 메시인 **Floor_400x400**를 드래그하여 레벨에 배치하자.

아웃라이너에서 배치된 **Floor_400x400**를 선택하고 **Ctrl+C**를 누르고 **Ctrl+V**를 눌러 추가로 6개의 복사본을 만들자. 총 7개가 배치된다. 각 메시에 대해서 다음과 같이 트랜스폼을 수정하자. 채워져 있지 않은 부분은 디폴트로 두면 된다. 디폴트로 회전은 (0,0,0)이고 스케일은 (1,1,1)이다.

메시 액터	위치	회전	스케일
Floor_400x400	(-600, 0, 0)		(1.5, 2, 1)
Floor_400x401	(-600, 800, 0)		(1.5, 2, 1)
Floor_400x402	(-200, -400, 0)		(0.5, 1, 1)
Floor_400x403	(-600, -400, 0)		
Floor_400x404	(-600, -400, 420)		(1.5, 1, 1)
Floor_400x405	(-600, 0, 420)		(1.5, 2, 1)
Floor_400x406	(-600, 800, 420)		(1.5, 2, 1)

아웃라이너에서 배치된 7개의 액터를 선택하고 아웃라이너의 검색창 오른쪽에 있는 새 폴더 생성 아이콘을 클릭하자. 새 폴더의 이름을 **Floor**로 수정하자. 그리고 폴더를 접자.

5. 이제 레벨이 아래와 같이 보일 것이다.



6. 콘텐츠 브라우저에서 스태틱 메시인 **Wall_400x400**를 드래그하여 레벨에 배치하자.

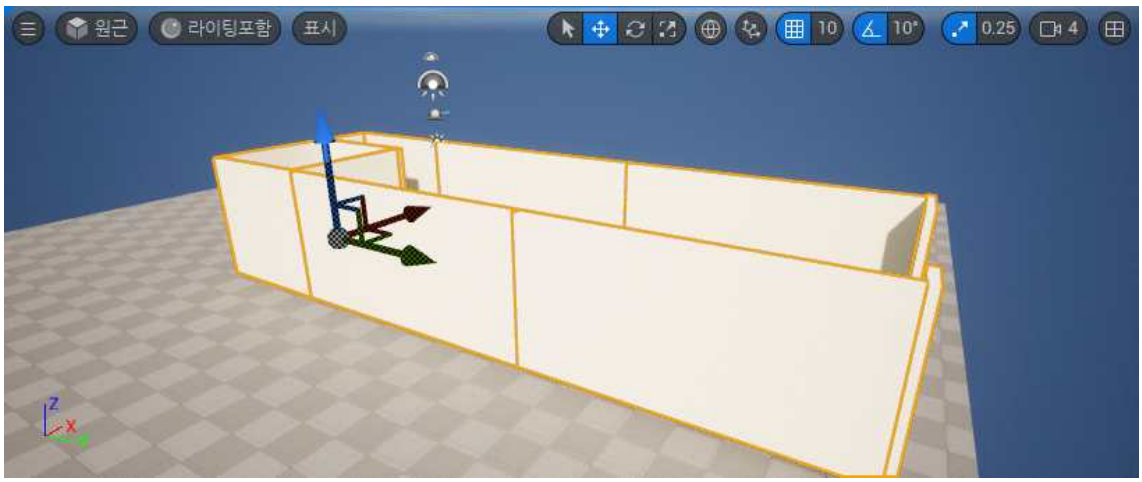
아웃라이너에서 배치된 **Wall_400x400**를 선택하고 **Ctrl+C**를 누르고 **Ctrl+V**를 눌러 추가로 11개의 복사

본을 만들자. 총 12개가 배치된다. 각 메시에 대해서 다음과 같이 트랜스폼을 수정하자.

메시 액터	위치	회전	스케일
Wall_400x400	(30, 1600, 0)	(0, 0, 180)	(0.375, 1, 1)
Wall_400x401	(-510, 1600, 0)	(0, 0, 180)	(0.25, 1, 1)
Wall_400x402	(-600, 790, 0)	(0, 0, 90)	(2, 1, 1)
Wall_400x403	(-600, 0, 0)	(0, 0, 90)	(2, 1, 1)
Wall_400x404	(-600, -400, 0)	(0, 0, 90)	
Wall_400x405	(-600, -400, 0)		
Wall_400x406	(-200, -400, 0)		(0.5, 1, 1)
Wall_400x407	(0, 0, 0)	(0, 0, -90)	
Wall_400x408	(0, 0, 0)	(0, 0, 90)	(2, 1, 1)
Wall_400x409	(0, 790, 0)	(0, 0, 90)	(2, 1, 1)
Wall_400x410	(-600, -10, 0)		
Wall_400x411	(-190, -400, 0)	(0, 0, 90)	

아웃라이너에서 배치된 12개의 액터를 선택하고 아웃라이너의 검색창 오른쪽에 있는 새 폴더 생성 아이콘을 클릭하자. 새 폴더의 이름을 **Wall**로 수정하자. 그리고 폴더를 접자.

7. 이제 **Floor** 폴더를 보이지 않게 하고 **Wall** 폴더만 보이도록 하면 레벨이 아래와 같이 보일 것이다.

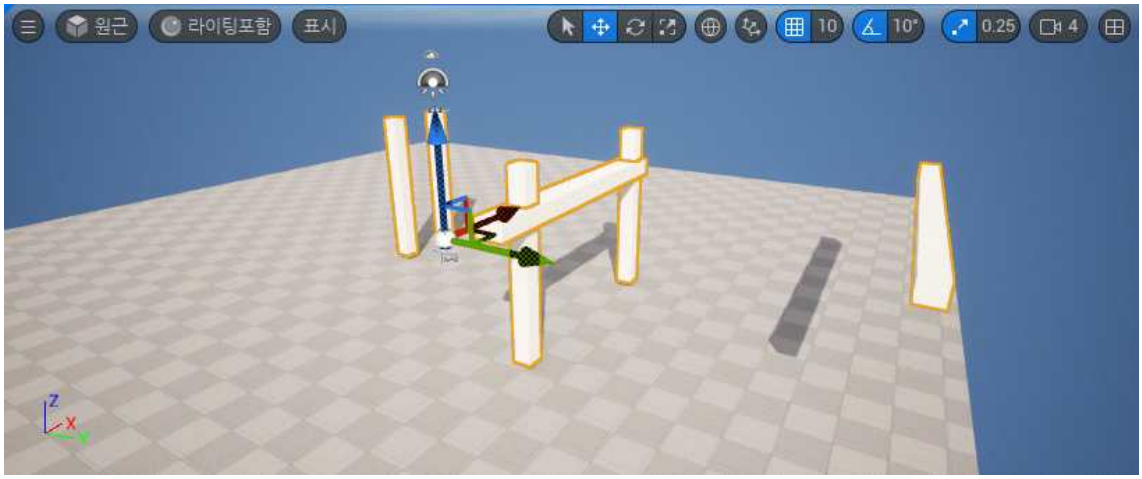


8. 콘텐츠 브라우저에서 스태틱 메시인 **Pillar_50x500**를 드래그하여 레벨에 배치하자.
아웃라이너에서 배치된 **Pillar_50x500**를 선택하고 **Ctrl+C**를 누르고 **Ctrl+V**를 눌러 추가로 5개의 복사본을 만들자. 총 6개가 배치된다. 각 메시에 대해서 다음과 같이 트랜스폼을 수정하자.

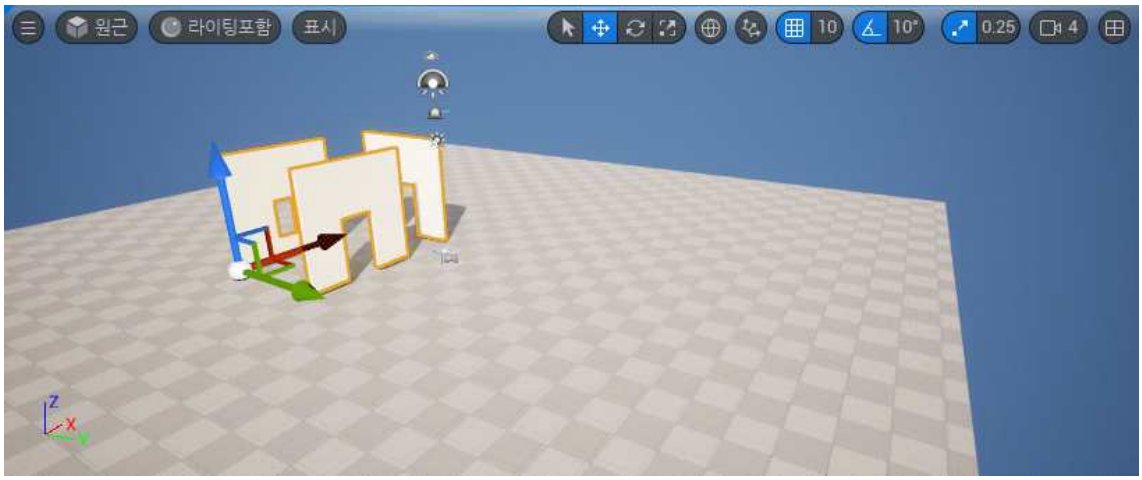
메시 액터	위치	회전	스케일
Pillar_50x500	(-600, 1600, 380)	(0, -90, 0)	(1, 1, 2)
Pillar_50x501	(-710, 800, 380)	(0, -90, 0)	(1, 1.5, 1.5)
Pillar_50x502	(-580, 800, 0)		
Pillar_50x503	(0, 800, 0)		
Pillar_50x504	(-190, -10, 0)		
Pillar_50x505	(0, 0, 0)		

아웃라이너에서 배치된 6개의 액터를 선택하고 아웃라이너의 검색창 오른쪽에 있는 새 폴더 생성 아이콘을 클릭하자. 새 폴더의 이름을 **Pillar**로 수정하자. 그리고 폴더를 접자.

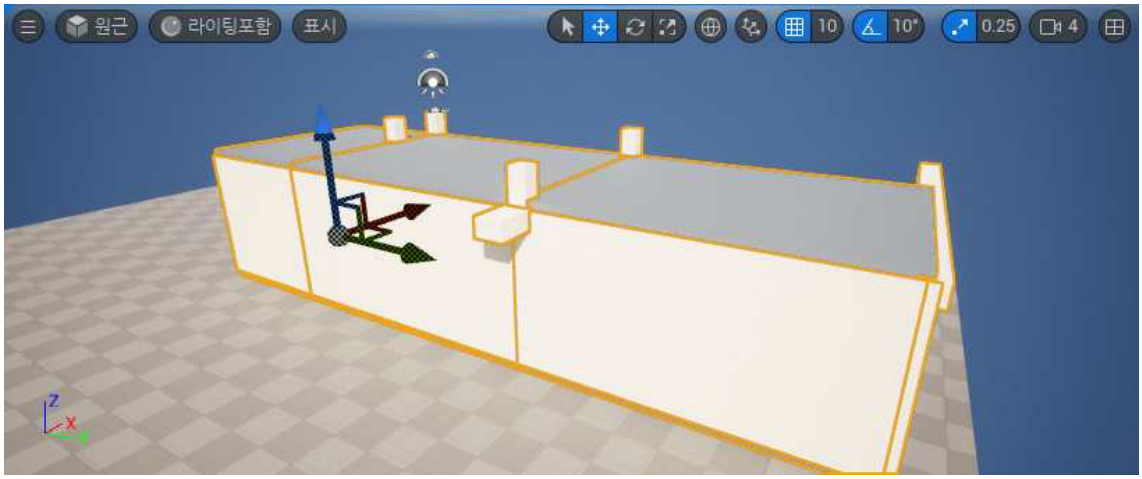
9. 이제 **Floor** 폴더와 **Wall** 폴더를 보이지 않게 하고 **Pillar** 폴더만 보이도록 하면 레벨이 아래와 같이 보일 것이다.



10. 두 개의 벽 부분을 문으로 바꾸고 하나의 벽 부분을 창문으로 바꾸자.
아웃라이너에서 **Wall** 폴더 아래의 **Wall_400x407**를 선택하자. 디테일 탭에서 **Static Mesh** 속성을 찾고 벽 메시인 **Wall_400x400**로 지정되어 있는 것을 문 메시인 **Wall_Door_400x400**로 수정하자.
아웃라이너에서의 **Wall_400x410**에 대해서도 동일한 방법으로 벽 메시를 문 메시로 바꾸자.
아웃라이너에서 **Wall_400x405**를 선택하고, 벽 메시를 이번에는 창문 메시인 **Wall_Window_400x400**로 수정하자.
수정된 세 메시만 보이도록 하면 아래와 같이 보일 것이다.



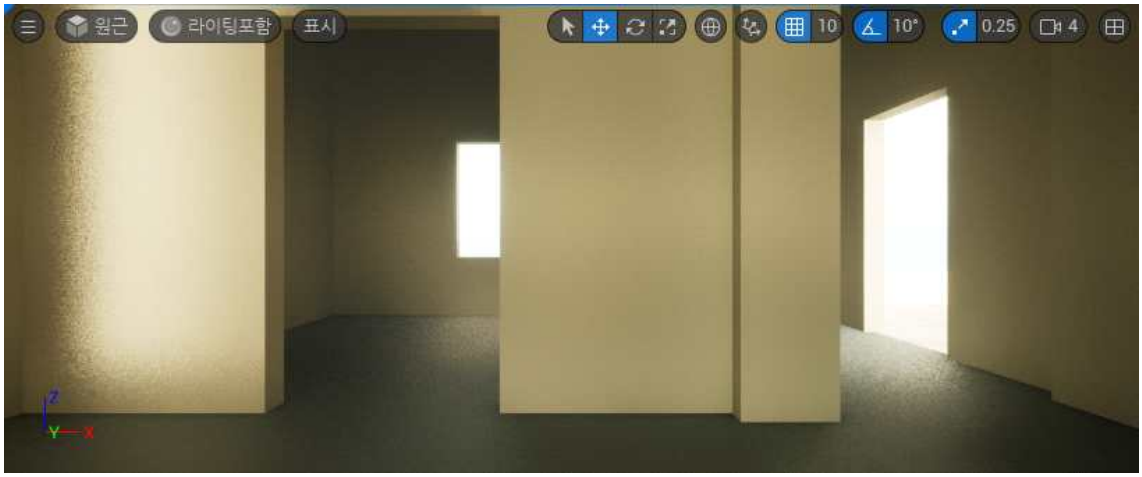
11. 이제 라이팅을 테스트하기 위한 건물이 완성되었다. 전체 모습이 아래와 같이 보일 것이다.



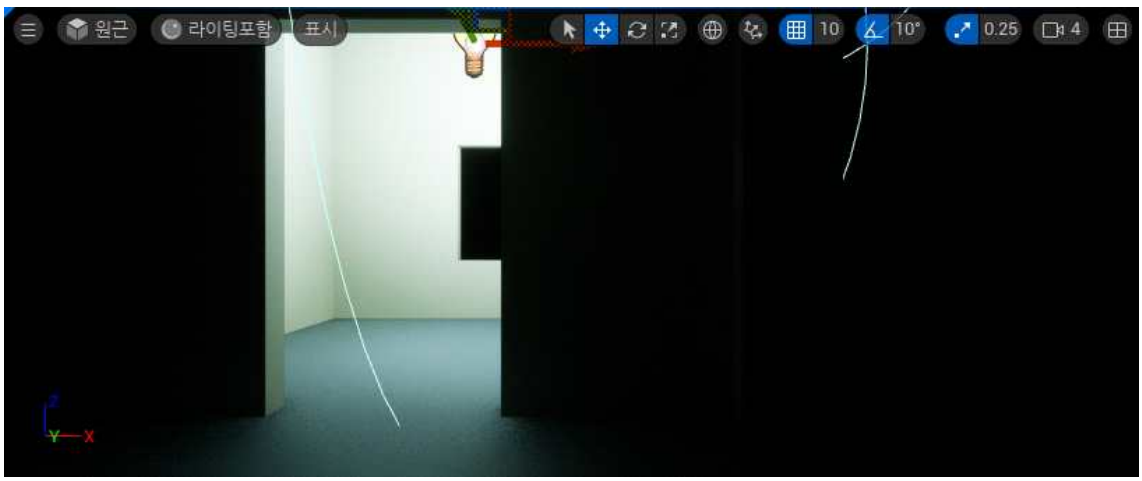
플레이해보자.

12. 이제부터 라이트를 추가하여 라이팅을 완성해보자.
레벨에는 이미 **디렉셔널 라이트**가 배치되어 있다.
디렉셔널 라이트(Directional Light)에 대해서는 이미 이전 절에서 다루었다.
컬러만 약간 수정해보자.
아웃라이너에서 배치되어 있는 **DirectionalLight**를 선택하자.
디테일 탭에서 라이트 영역의 **라이트 컬러**(**Light Color**)를 찾아서 디폴트 값인 (255,255,255)를 (255,232,181)로 수정하자. 약간 따뜻한 느낌의 컬러가 될 것이다.

13. 플레이해보자. 건물 내에 작은 방의 문 앞에서의 장면을 보여준다.



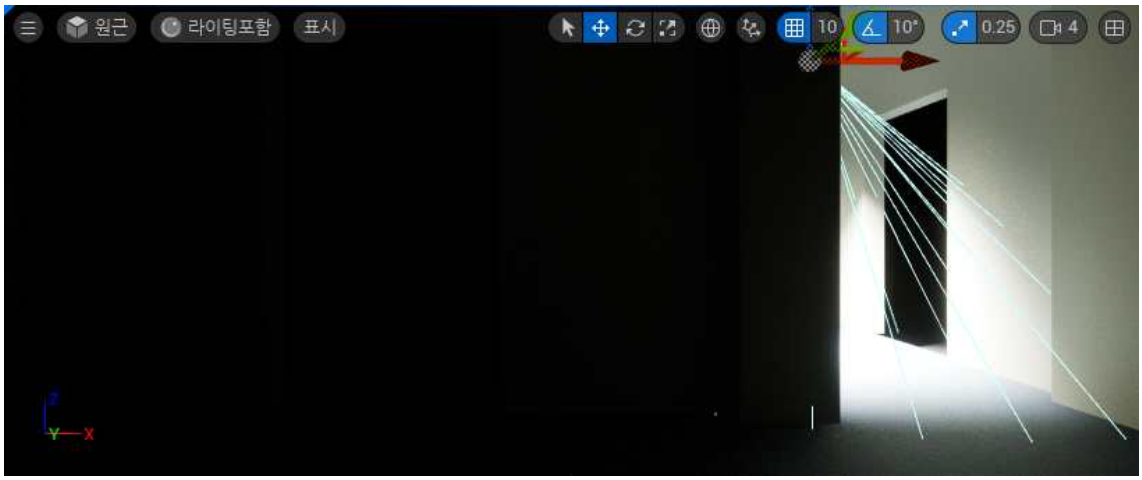
14. 건물 내의 작은 방 안에 포인트 라이트를 추가해서 방 안을 밝게 비추도록 해보자.
이제부터 **포인트 라이트**(Point Light)를 추가해보자.
액터 배치 탭에서 **라이트** 탭을 클릭하고 **포인트 라이트**를 드래그해서 배치하자. 위치를 (-387, -208, 270)으로 하자. 작은 방 중간의 위에 배치될 것이다.
그리고, 디테일 탭에서 라이트 영역의 강도(**Intensity**) 속성이 있다. 이것은 라이트의 밝기를 조절한다. 강도는 광원의 단위 면적당 밝기인 광도(luminous intensity)를 의미하며, 광도의 단위는 칸델라(candela; cd)이다. 디폴트값인 8에서 12로 수정하자.
그다음, 그 아래의 **라이트 컬러**(**Light Color**) 속성값을 디폴트인 (255,255,255)에서 (206,248,255)로 수정하자.
그다음, 그 아래의 **어테뉴에이션 반경**(Attenuation Radius) 속성이 있다. 이것은 라이트가 영향을 미치는 반경을 조절한다. 건물 외부로 줌아웃하고 이 속성값을 조절해보자. 구체가 표시될 것이다. 이 구체가 라이트가 영향을 주는 범위를 의미한다. 디폴트값인 1000을 350으로 수정하자. 작은 방 안의 범위에 잘 맞도록 줄였다.
DirectionalLight를 숨기고 **PointLight**만 보이도록 하면 아래와 같이 보일 것이다.



지금까지 포인트 라이트에 대해서 알아보았다.

<참고> 포인트 라이트에 대한 문서는 다음의 링크를 참조하자.
https://docs.unrealengine.com/point-lights-in-unreal-engine/

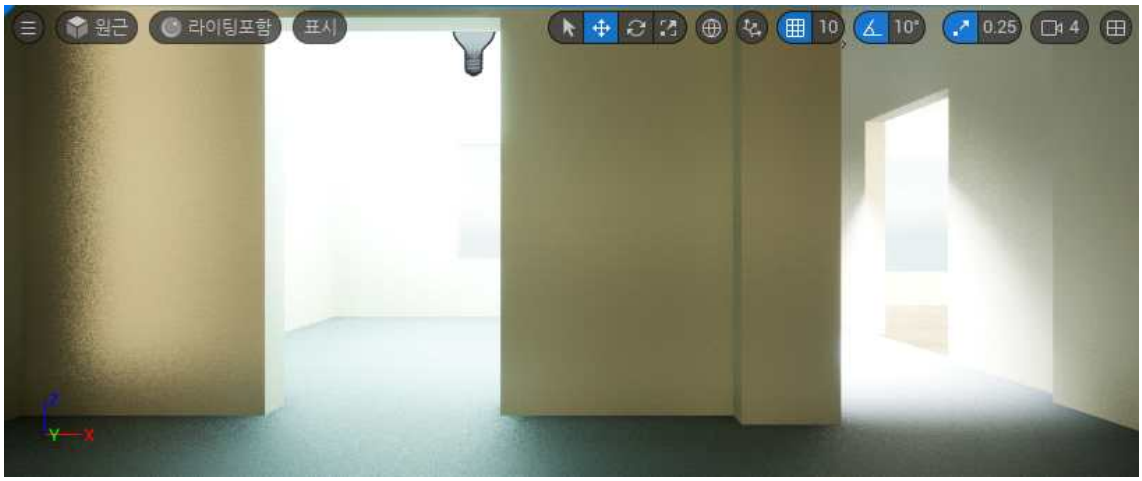
15. 이제부터 **스포트 라이트**(Spot Light)를 추가해보자.
건물의 입구에 달아보자.
액터 배치 탭에서 **라이트** 탭을 클릭하고 **스포트 라이트**를 드래그해서 배치하자. 위치를 (-90, -200, 250)으로 하자. 회전은 (0, -90, 0)으로 두자. 입구 통로 위에 배치될 것이다.
그리고, 디테일 탭에서 **라이트** 영역의 **강도**(**Intensity**) 속성값을 디폴트값인 8에서 50으로 수정하자.
그다음, 그 아래의 **Light Color** 속성값을 디폴트인 (255,255,255)에서 (255,230,169)로 수정하자.
그다음, 그 아래의 **어테뉴에이션 반경** 속성값을 디폴트값인 1000을 500으로 수정하자.
DirectionalLight와 **PointLight**를 숨기고 **SpotLight**만 보이도록 하면 아래와 같이 보일 것이다.



지금까지 스포트 라이트에 대해서 알아보았다.

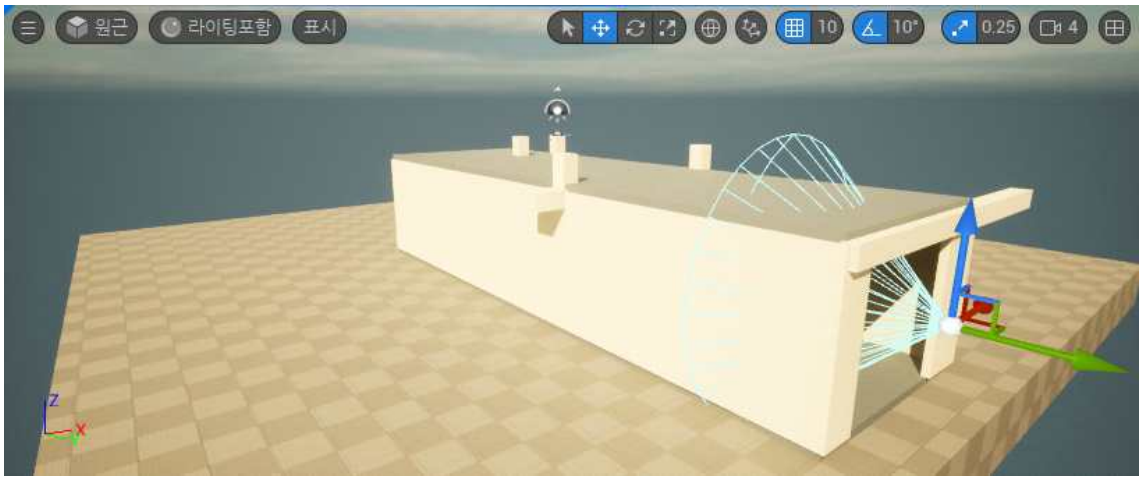
<참고> 스포트 라이트에 대한 문서는 다음의 링크를 참조하자.
https://docs.unrealengine.com/spot-lights-in-unreal-engine/

16. **DirectionalLight**와 **PointLight**와 **SpotLight**를 모두 보이도록 하면 아래와 같이 보일 것이다.

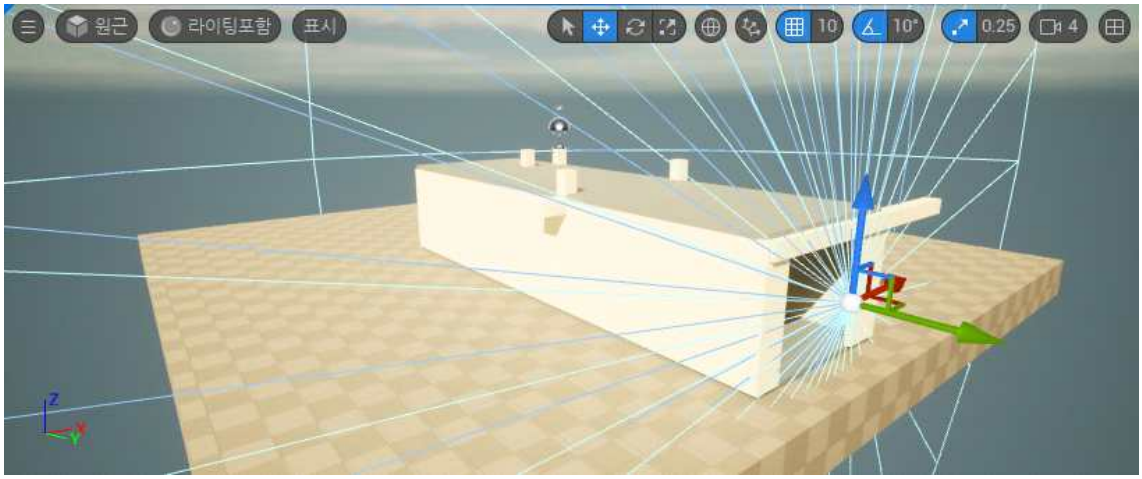


17. 스포트 라이트를 배경 조명의 용도로도 사용할 수 있다.
배경 조명으로는 이전의 디렉셔널 라이트에서 다루었던 **스카이 라이트**(Sky Light)를 흔히 사용한다. **스카이 라이트**는 실외 환경에서의 큰 영역에서 사용하기에 적합하다. 한편, 실내의 작은 영역의 경우에는 **스포트 라이트**를 배경 조명으로 사용하면 된다. **스포트 라이트**를 사용하면 라이팅 제어를 더

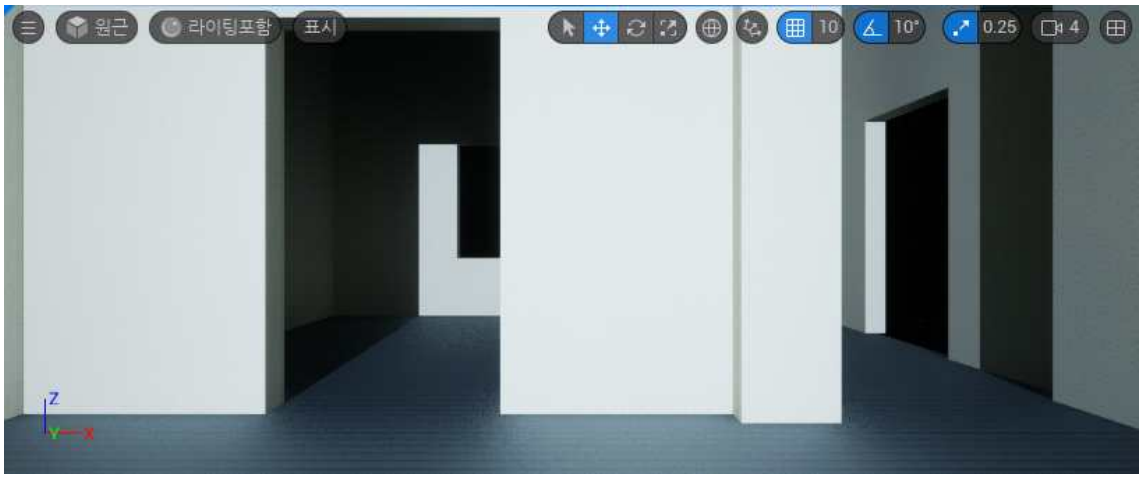
효율적으로 할 수 있다.
이제부터, 배경 조명을 위한 스포트 라이트를 사용해보자.
먼저, 이미 추가된 스포트 라이트를 선택하고 **Ctrl+C**를 누르고 **Ctrl+V**를 눌러 복제본을 만들자. 그다음, 이름을 이름을 **SpotLightAmbient**로 수정하자.
그다음, 위치를 (-350, 1700, 220)으로 수정하고, 회전을 (90,0,-90)으로 수정하자. 이렇게 배치하면 건물 뒤쪽의 크게 뚫려있는 문의 외부에서 건물 안쪽으로 비추게 된다.



18. 건물 안쪽을 잘 비추도록 **SpotLightAmbient**의 속성값을 수정해보자.
디테일 탭에서 **라이트** 영역의 속성값들을 수정하자.
먼저, **강도(Intensity)**를 50에서 3으로 수정하자. 배경 조명이므로 약하게 하자.
그다음, **라이트 컬러(Light Color)**를 (255,255,255)에서 (202, 224, 255)로 수정하자. 약간 연한 파란색으로 된다. 이렇게 하면 레벨의 기본 라이팅 컬러와 다르게 대비 효과를 주어 방의 라이팅이 더 자연스럽게 보인다.
그다음, **어테뉴에이션 반경**을 500에서 5000으로 충분히 크게 하자.
그다음, **내부 원뿔 각도(Inner Cone Angle)**를 0에서 70으로 수정하고, **외부 원뿔 각도(Outer Cone Angle)**를 44에서 80으로 수정하자. 이렇게 하면 중심이 강한 빛이 아니라 전체적으로 균일한 강도의 빛을 내보낸다.
그다음, 디테일 탭에서 **라이트** 영역의 마지막의 **고급** 영역을 펼치자.
그다음, **역 제곱 감쇠 사용(Use Inverse Squared Falloff)** 옵션에 체크되어 있는 것을 클릭하여 체크해제하자. 이 옵션은 현실에서의 빛 작용을 가장 근접하게 흉내 내는 라이트 감쇠 유형이다. 광원에서 멀어지면 빠르게 어두워지도록 한다. 이것을 끄면 감쇄가 덜 되므로 빛이 멀리까지 도달한다.
지금까지 스포트 라이트에 배경 조명으로 사용하는 방법을 알아보았다.



19. 새로 추가한 스포트 라이트만 켰을 때의 모습을 보면 아래와 같다.



20. 플레이해보자.

이 절에서 다양한 종류의 라이트를 사용해서 건물 내부의 라이팅 방법에 대해서 학습하였다.

□