24_ 머티리얼 기초

<제목 차례>

인천대학교 컴퓨터공학부 박종승

1. 개요

이 장에서 머티리얼의 기초를 학습한다.

머티리얼은 객체의 표면을 어떻게 색칠할 것인가를 표현하는 애셋이다. 머티리얼이 없다면 객체는 다각형들의 메시일 뿐이다. 시각적인 화려함이나 사실적인 표현은 모두 머티리얼로부터 결정된다. 동일한 바닥면이라고 하더라도 어떤 머티리얼을 적용하는가에 따라서 나무바닥이 될 수도 있고 빙판이 될 수도 있고 금속판이 될 수도 있다. 또한 표면의 얼룩이나 스크래치도 얼마든지 나타낼 수 있다.

언리얼에서는 머티리얼의 물리적인 특성을 사용해서 렌더링하는 기법인 **물리 기반 렌더링**(physically based rendering)이라는 최신 기법을 사용하고 있다. 이것은 물체의 물리적인 특성을 재질로 표현하고 그 재질에 기반하여 렌더링하는 기법이다.

머티리얼과 혼동하기 쉬운 텍스처에 대해서 알아보자.
텍스처는 단순히 픽셀들의 이차원 배열인 이미지 자체를 의미한다. 이미지 파일로부터 읽어들인 이미지 데이터를 텍스처라고 이해하자. 한편, 머티리얼은 객체의 표면을 칠하는 방법을 정의하는 애셋이다. 머티리얼은 내부적으로 하나 또는 여러개의 텍스처를 활용해서 컬러나 깊이나 투명도 등을 계산한다.

텍스처와 연관되어 UV라는 용어가 자주 등장한다.
텍스처는 픽셀의 이차원 배열이다. 이 배열의 좌표 공간은 정수 배열 인덱스가 아닌 [0,1]의 실수값으로 정규화되어 있다. 왼쪽 하단이 좌표 공간의 원점이고 오른쪽 방향이 U 좌표계이고 위쪽 방향이 V 좌표계이다. 이러한 텍스처의 좌표 공간을 UV 공간이라고 한다.

<참고> 머티리얼의 다양한 학습 자료에 대해서는 다음의 문서를 참조하자.
https://docs.unrealengine.com/unreal-engine-materials/
머티리얼의 튜토리얼 자료에 대해서는 다음의 문서를 참고하자.
https://docs.unrealengine.com/unreal-engine-materials-tutorials/
머티리얼 함수에 대해서는 다음의 링크를 참조하자.
https://docs.unrealengine.com/material-functions-in-unreal-engine/

2. 머티리얼 처음으로 만들어보기

이 절에서 머티리얼을 처음으로 만들어보는 것에 대해서 학습한다.

메인 머티리얼 노드에 대해서 알아보자.
머티리얼을 만드는 것은 머티리얼 그래프를 만드는 것이다. 머티리얼 그래프에서는 **메인 머티리얼 노드**(Main Material Node)라는 최종 노드가 가장 오른쪽에 있다. 머티리얼 그래프는 **메인 머티리얼 노드**의 입력을 만들기 위한 머티리얼 표현식(Material Expression) 노드들의 노드 네트워크이다. 여기서 머티리얼 표현식 노드는 메인 머티리얼 노드 이외의 모든 노드를 지칭한다. **메인 머티리얼 노드**의 입력에 어떤 것을 넣어주는가에 따라서 머티리얼의 모든 것이 결정된다. 참고로, 어떤 문서에서는 **메인 머티리얼 노드**를 **베이스 머티리얼 노드**라고도 한다. 이 두 용어는 동일한 의미이다.

머티리얼을 사용해서 표면을 렌더링하는 일은 GPU에서 실행된다. GPU에서의 작업을 위한 프로그램은 셰이딩 언어로 명시된다. 셰이딩 언어에는 마이크로소프트의 DirectX API에서 사용되는 셰이더 언어인 HLSL(High Level Shader Language)가 있고, OpenGL API에서 사용되는 셰이딩 언어인 GLSL(OpenGL Shading Language)이 있고, 엔비디아의 셰이더 언어인 Cg(C for Graphics)가 있다. 이들은 모두 서로 협력하여 개발하였기 때문에 거의 유사하다.
언리얼 엔진에서 머티리얼 그래프에서 사용되는 머티리얼 표현식 노드들은 바로 HLSL 셰이더 언어의 문장에 해당한다. **메인 머티리얼 노드**는 이들 머티리얼 표현식 노드 네트워크의 결과를 렌더링하는 기능을 수행한다.

머티리얼의 노드 네트워크에 대해서 알아보자.
머티리얼의 노드 그래프는 블루프린트의 노드 그래프와는 전혀 다른 별개의 그래프이다. 머티리얼 표현식 노드는 이전의 블루프린트 노드와 관련이 없는 별개의 노드들이다.
블루프린트의 노드 그래프는 C++ 클래스 코딩을 대신하는 목적이고 머티리얼의 노드 그래프는 셰이더 HLSL 코딩을 대신하는 목적이다. 따라서 블루프린트 노드 그래프는 C++ 코드로 바뀌게 되는 반면에 머티리얼 그래프는 HLSL 코드로 바뀌게 된다.
머티리얼 표현식 노드는 HLSL에서의 문법이나 함수와 관련된다. 머티리얼 표현식 노드는 셰이더의 기본 개념을 내포하고 있기 때문에 셰이더에 익숙하지 않다면 복잡하게 느껴질 수 있다. 우리는 쉬운 수준에서부터 하나씩 배워볼 것이다.

<참고> 메인 머티리얼 노드를 사용하는 방법에 대해서는 다음의 문서를 참조하자.
https://docs.unrealengine.com/using-the-main-material-node-in-unreal-engine/

<참고> 메인 머티리얼 노드의 입력핀에는 베이스 컬러(Base Color), 메탈릭(Metallic) 스페큘러(Specular), 러프니스(Roughness), 이미시브 컬러(Emissive Color), 노멀(Normal) 등을 포함하여 많은 입력핀들이 있다. 입력핀에 대한 자세한 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/material-inputs-in-unreal-engine/

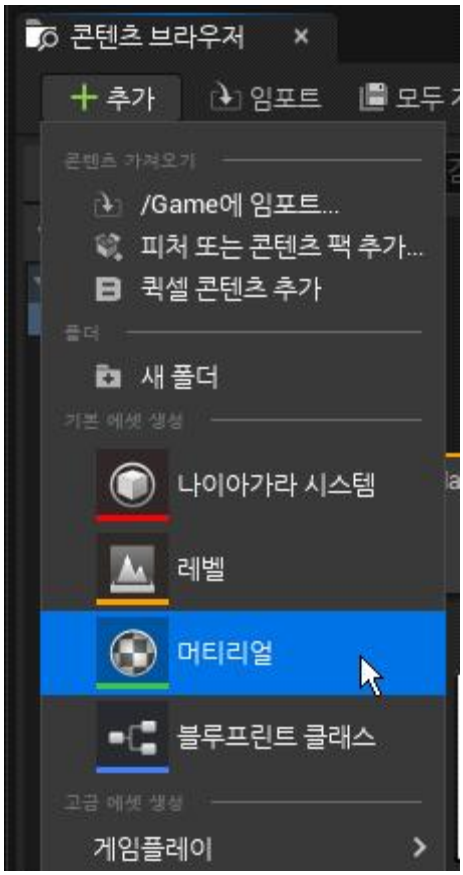
이제부터 예제를 통해서 학습해보자

---

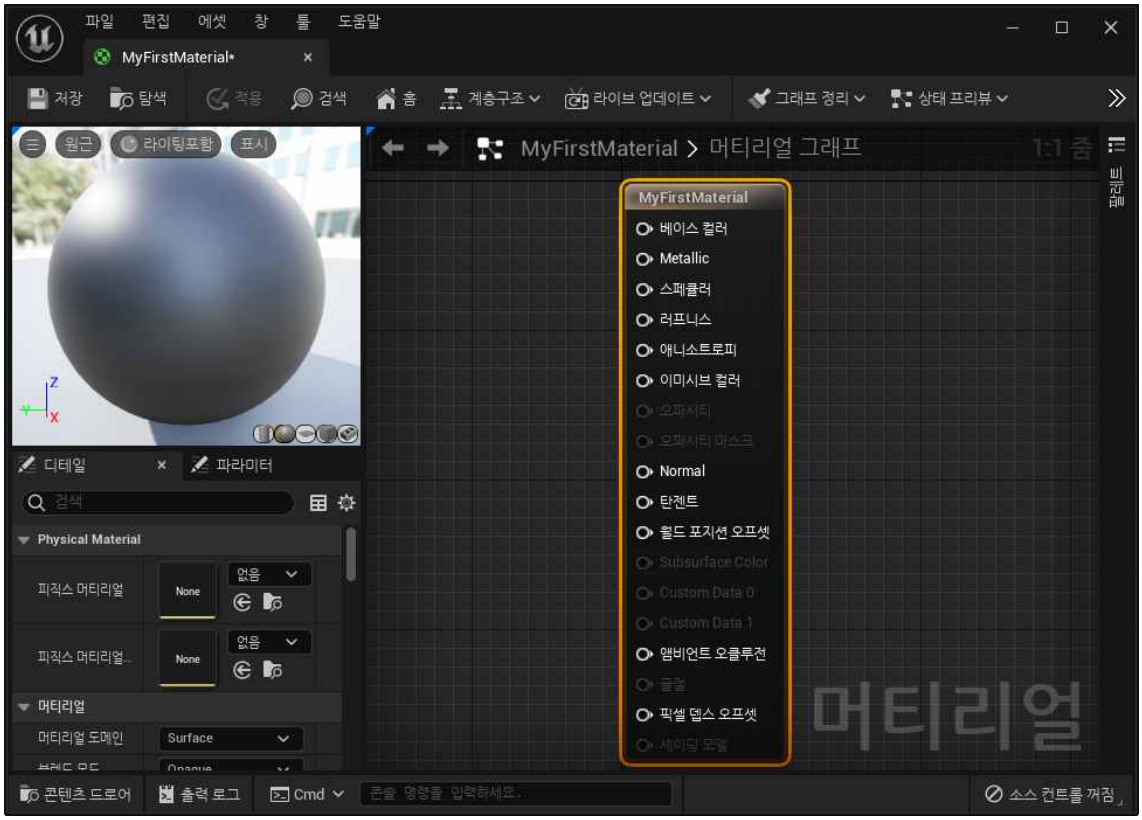
1. 새 프로젝트 **Pmtlfirst**를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서

왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Pmtlfirst로 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. 머티리얼 에셋을 생성하자.
콘텐츠 브라우저에서 **+추가** 버튼을 클릭하고 드롭다운 메뉴에서 **기본 에셋 생성** 영역의 **머티리얼**을 선택하자. 또다른 방법으로 **콘텐츠** 폴더에서 우클릭하고 드롭다운 메뉴에서 **콘텐츠 추가/임포트** 아래의 **머티리얼**을 선택해도 된다. 또다른 방법으로 콘텐츠 폴더를 선택하고 오른쪽 탭의 빈 곳에서 우클릭하고 드롭다운 메뉴에서 **머티리얼**을 선택해도 된다.
생성된 머티리얼의 이름을 디폴트 이름인 **NewMaterial**에서 **MyFirstMaterial**로 수정하자.



3. **콘텐츠 브라우저**에서 생성된 머티리얼 애셋 **MyFirstMaterial**를 더블클릭하여 머티리얼 에디터를 열자. 머티리얼을 생성하면 **메인 머티리얼 노드** 하나가 배치된다. 우리는 머티리얼 표현식 노드 네트워크를 작성하여 **메인 머티리얼 노드**의 각 입력에 적절한 값을 넣어주면 된다.



이제부터, 머티리얼 에디터의 레이아웃을 살펴보자.

먼저, **머티리얼 그래프** 탭을 알아보자.

머티리얼 에디터의 중간에 위치한 격자 탭이 머티리얼 그래프 탭이다. 여기서 노드 네트워크를 작성하여 셰이더 처리를 위한 그래프를 작성한다. 머티리얼 그래프 탭에는 디폴트로 **메인 머티리얼 노드**가 배치되어 있다. 모든 머티리얼에는 반드시 하나의 **메인 머티리얼 노드**가 있어야 한다. 머티리얼 그래프 탭에서 작성하는 모든 노드 네트워크는 결국 **메인 머티리얼 노드**의 입력핀에 연결되기 위한 목적으로 작성된다.

다음으로, **디테일** 탭을 살펴보자.

메인 머티리얼 노드를 선택하면 왼쪽 아래의 디테일 탭에서 모든 머티리얼 속성이 표시된다. 이 속성은 셰이더에서의 주요 속성값을 지정하는 매우 중요한 속성이다. 한편, **메인 머티리얼 노드**가 아닌 그 외의 머티리얼 표현식 노드를 선택한 상태에서는 선택된 노드의 세부 정보가 디테일 탭에 표시된다.

다음으로, 머티리얼 그래프 탭에서 노드를 배치하는 방법을 알아보자.

가장 기본적인 방법은 액션선택 창을 사용하는 것이다. 격자판의 빈 곳에서 우클릭하면 액션선택 창이 뜬다. 이 창에서 나열되는 목록에서 선택하거나 검색하여 노드를 찾아 배치하면 된다. 노드를 배치하는 또다른 방법은 팔레트 탭을 사용하는 것이다. 머티리얼 에디터의 우측에 팔레트 아이콘이 있다. 이 아이콘을 클릭하면 열렸다가 자동으로 닫히는 서랍인 드로어 형태로 팔레트 탭이 나타난다. 이 팔레트 탭에서 배치할 노드를 찾아 드래그하여 그래픽 탭에 끌어 놓으면 노드가 배치된다.

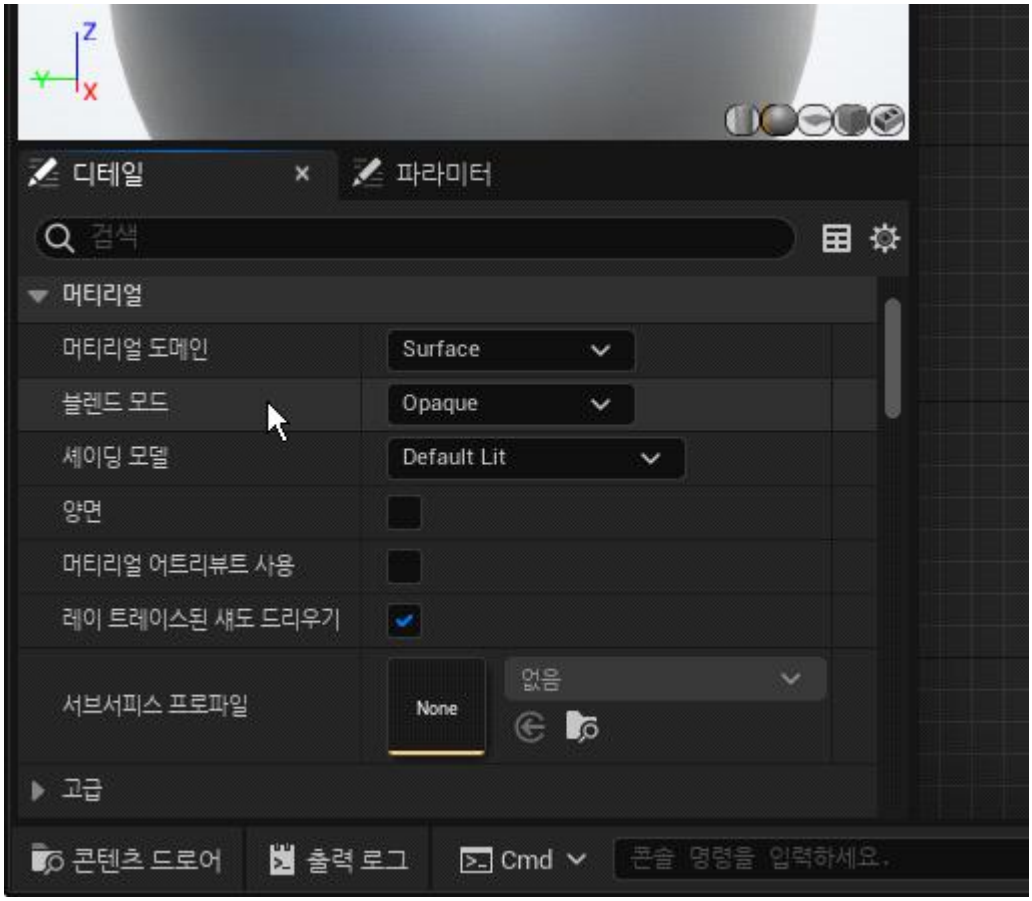
다음으로, 그래프 탭에서의 조작 방법을 알아보자. 그래프에서의 조작 방법은 이전과 거의 동일하다. 마우스 **우클릭+드래그**하면 격자판을 패닝할 수 있다. 또한 **휠버튼+스크롤**하면 격자판을 줌인이나 줌아웃할 수 있다. 그리고, 노드 위에서 **좌클릭+드래그**하면 노드를 이동시킬 수 있다.

다음으로, **프리퓨** 뷰포트에 대해서 알아보자.

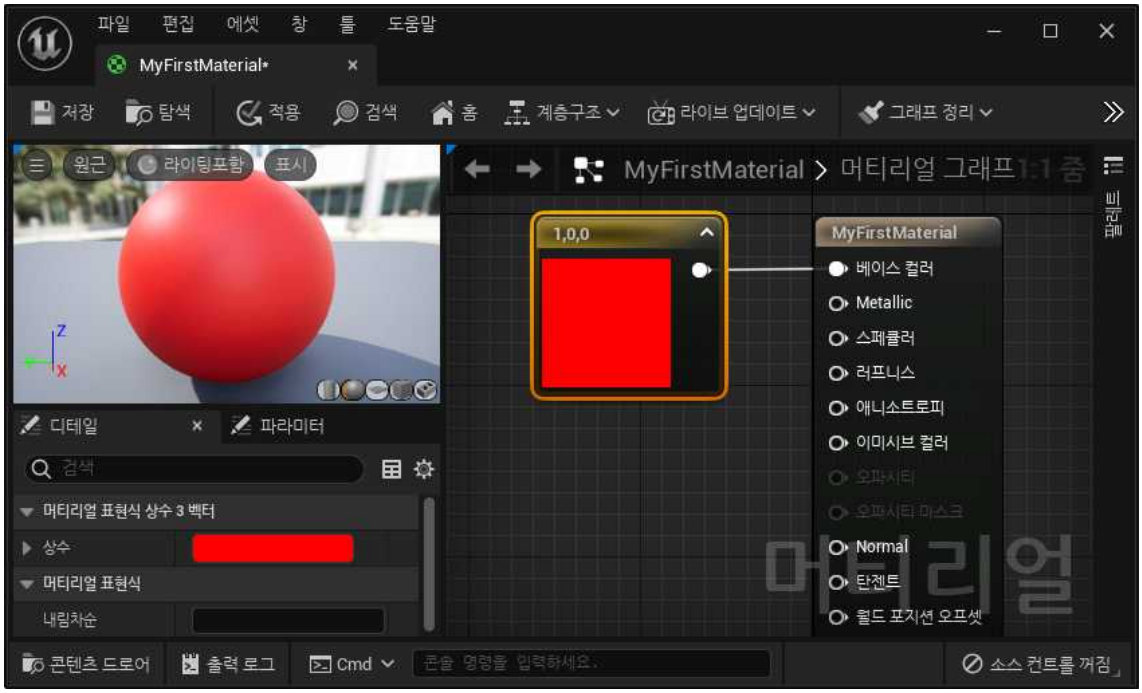
왼쪽 위의 프리뷰 뷰포트에서는 현재까지 완성된 셰이더가 어떤 모습인지를 확인할 수 있다. 프리뷰 뷰포트에서의 조작법은 이전의 뷰포트에서의 조작법과 동일하다. 각 마우스 버튼을 드래그하면서 조작해보자. 그리고, **L+좌클릭+드래그**로 프리뷰 라이트의 위치를 바꾸어볼 수도 있다.
프리뷰 뷰포트의 오른쪽 하단에 여러 아이콘이 있다. 이들을 클릭해서 프리뷰 오브젝트의 모양을 바꿀 수 있다. 원기둥, 구체, 평면, 큐브 프리미티프로 바꿀 수 있고, 가장 오른쪽의 아이콘은 현재 콘텐츠 브라우즈에서 선택한 메시로 바꿀 수 있다.
다음으로 툴바에 대해서 알아보자.
툴바의 **적용** 버튼을 클릭하면 현재 머티리얼 에디터의 변경 사항을 원본 머티리얼과 월드에서 그 머티리얼을 사용한 곳에 적용시킨다. 에디터에서 플레이중인 경우에도 이 **적용** 버튼을 클릭하면 수정사항이 바로 적용된다.
툴바에는 약간 왼쪽에 **홈** 버튼이 있다. 이 **홈** 버튼을 클릭하면 현재 격자판에서의 기준이 되는 노드로 이동한다. 디폴트 머티리얼 그래프에서는 **메인 머티리얼 노드**가 기준이 되는 노드이다.

<참고> 머티리얼 에디터의 인터페이스에 대해서는 다음의 링크를 참조하자.
https://docs.unrealengine.com/unreal-engine-material-editor-ui/

4. 먼저 **메인 머티리얼 노드**에 대해서 알아보자.
메인 머티리얼 노드는 입력핀에 주어진 값들을 사용하여 최종적으로 머티리얼을 렌더링하는 역할을 수행한다.
흰색 입력핀은 머티리얼에 영향을 주는 유효한 입력핀이라는 의미이고 회색 입력핀은 머티리얼에 영향을 주지 않는 입력핀이라는 의미이다. 회색 핀에는 노드 네트워크로부터 입력값이 주어지더라도 그 값은 사용되지 않고 무시된다.
메인 머티리얼 노드에 대해 명시해둔 속성값에 따라서 어떤 입력핀이 활성으로 사용될 지가 정해진다.
메인 머티리얼 노드의 속성값은 메인 머티리얼 노드를 선택하면 왼쪽 디테일 탭에 나열된다. 이들 속성값은 현재 머티리얼의 속성값에 해당하는 매우 중요한 속성값들이다. 특히 **Blend Mode** 속성이나 **Shading Model** 속성은 매우 중요한 속성으로 선택된 속성값에 따라서 입력핀의 활성이 바뀌게 된다.



5. 이제 처음으로 머티리얼 그래프에 머티리얼 표현식 노드를 추가해보자.
가장 먼저, 메인 머티리얼 노드의 **베이스 컬러**(**Base Color**) 입력핀에 빨간색 컬러를 연결해서 빨간색 머티리얼을 만들어보자. **베이스 컬러** 입력핀은 머티리얼의 전체적인 기본 컬러를 결정한다. 빨간색 컬러값을 표현할 수 있는 **Constant3Vector** 노드를 배치해보자. 격자판에서 우클릭하고 액션선택 창에서 **Constant3Vector**를 검색하여 배치하자. 또다른 방법으로, 오른쪽 모서리의 **팔레트** 탭에서 **Constants** 아래에 있는 **Constant3Vector**를 드래그해서 배치하면 된다. 또다른 방법으로, 격자판에서 단축키인 **3+좌클릭**하면 바로 **Constant3Vector** 노드가 배치된다.
그다음, 배치된 **Constant3Vector** 노드를 더블클릭하자. **색상 선택 툴**(Color Picker)이 뜬다. 여기에서 빨간색을 선택하자. 빨간색을 선택하는 방법에는 여러 가지 방법이 있다. **색상 선택 툴**을 조작해가면서 익혀보자. 빨간색을 선택하면 노드의 값은 (1,0,0)에 된다.
그다음, **Constant3Vector** 노드를 메인 머티리얼 노드의 **베이스 컬러** 입력핀에 연결하자. 와어어의 연결은 블루프린트 노드 그래프에서와 같이 한 노드의 핀을 **좌클릭+드래그**하고 다른 노드의 핀에 놓으면 연결된다.
입력핀에 연결하는 순간 왼쪽 위의 프리퓨 뷰포트에 바뀐 결과가 표시될 것이다.



상수값을 표현하는 방법에 대해서 알아보자.

머티리얼 에디티에서는 상수값을 표현하기 위해서 **Constant**, **Constant2Vector**, **Constant3Vector**, **Constant4Vector** 노드를 제공한다. 이들은 각각 실수값이 1,2,3,4개 있는 상수이다. 상수값이 1개 있는 것을 스칼라라고 하고 상수값이 2,3,4개 있는 것을 벡터라고 한다.

벡터의 경우는 주로 좌표값이나 컬러값을 실수로 표현한다. 컬러값의 경우에는 값을 [0,1]의 정규화된 값으로 표현한다. 값의 순서는 3차원 벡터인 경우에는 R,G,B의 순서이고 4차원 벡터인 경우에는 R,G,B,A의 순서이다.

단축키에 대해서 알아보자.

많이 사용되는 노드에 대해서는 격자판에서 바로 배치할 수 있는 단축키가 배정되어 있다. **단축키+좌클릭**하면 해당 노드가 바로 배치된다.

단축키에는 **1,2,3,4,U,T,S,V,A,D,M**이 있다. 단축키 **1,2,3,4**는 각각 **Constant**, **Constant2Vector**, **Constant3Vector**, **Constant4Vector** 노드를 배치한다. 단축키 **U**는 **TextureCoordinate** 노드를 배치한다. 단축키 **T**는 **TextureSample** 노드를 배치한다. 단축기 **S,V**는 **ScalarParameter** 노드와 **VectorParameter** 노드를 배치한다. 단축키 **A,D,M**은 각각 **Add**, **Divide**, **Multiply** 노드를 배치한다.

6. 우클릭하고 액션선택 창에서 **Constant**를 검색하여 배치하자. 또다른 방법으로, 팔레트 탭으로 가서 **Constant**를 찾아 드래그하여 배치해도 된다. 또다른 방법으로, 격자판에서 단축키 **1+좌클릭**으로 바로 배치해도 된다.

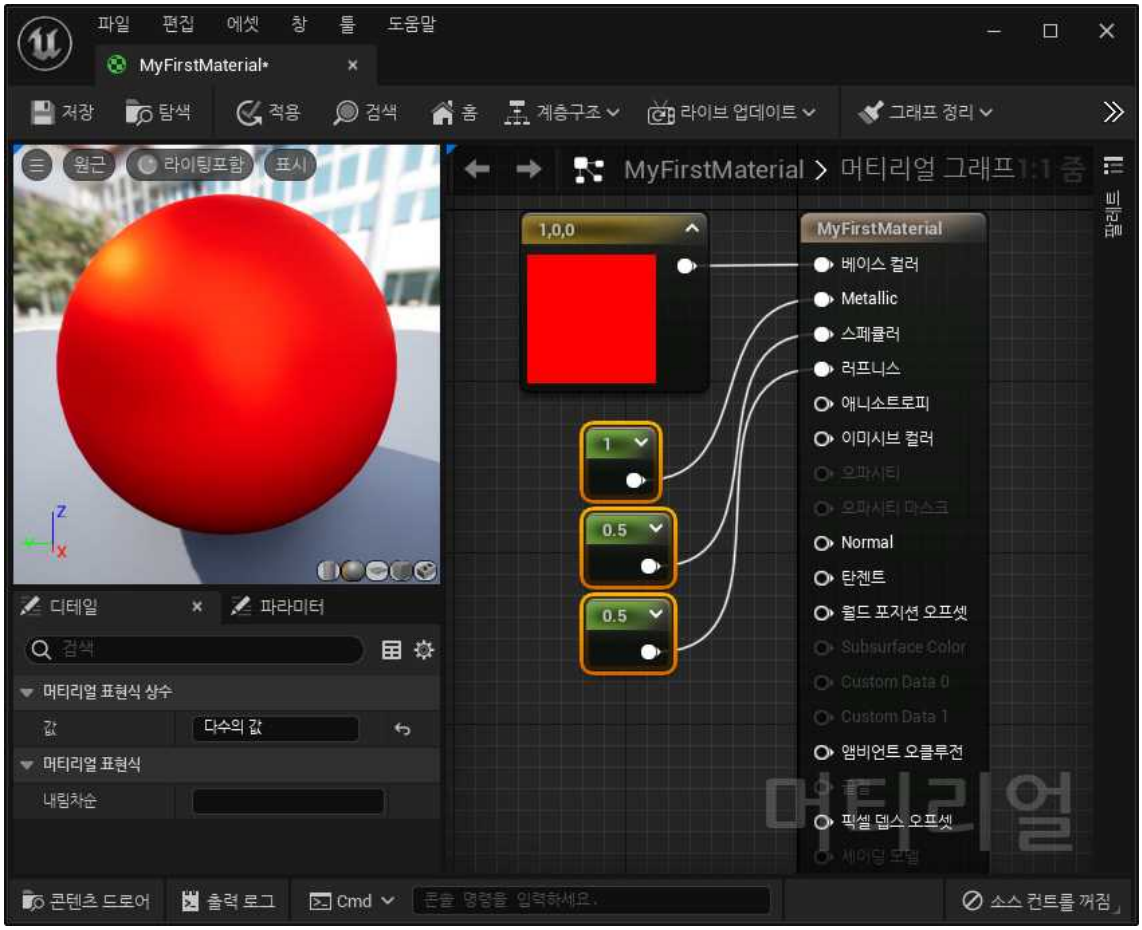
그다음, 배치된 노드를 메인 머티리얼 노드의 **Metallic** 입력핀에 연결하자.

그다음, 배치된 노드를 선택하고 디테일 탭에서 **값(Value)** 속성값에 1을 입력하자. **Metallic** 입력핀에는 [0,1] 범위의 실수값을 넣으면 된다. 속성값은 객체가 비 금속성이면 0에 가깝게 금속성이면 1에 가깝게 입력하면 된다.

그다음, 동일한 **Constant** 노드를 하나 더 배치하자. 단축키 1+좌클릭으로 배치하자. 디테일 탭에서 **값(Value)**에 0.5를 입력하자. 노드를 메인 머티리얼 노드의 **스페큘러**(Specular) 입력핀에 연결하자.

그다음, 동일한 **Constant** 노드를 하나 더 배치하자. 이번에는 기존의 배치된 **Constant**를 선택하고

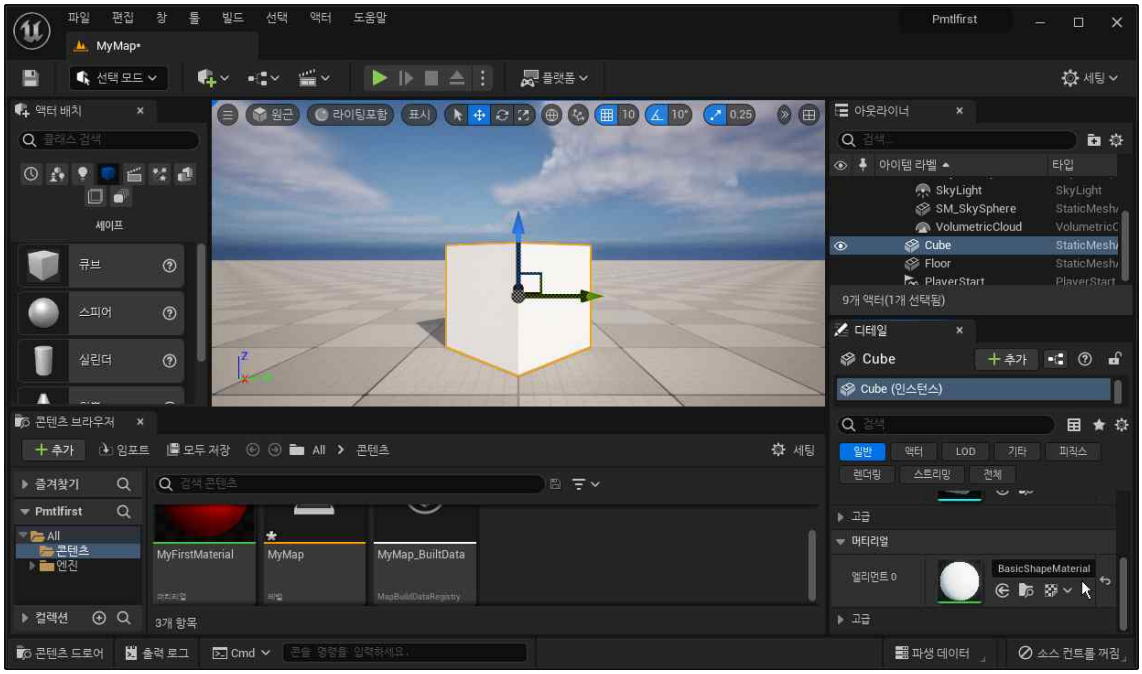
Ctrl+C를 누르고 Ctrl+V를 눌러 복제하여 배치해보자. 디테일 탭에서 값(Value)이 0.5가 되도록 확인하자. 노드를 메인 머티리얼 노드의 러프니스(Roughness) 입력핀에 연결하자.



이제 머티리얼을 완성하였다. 툴바의 저장 버튼을 클릭하자.
툴바의 저장 버튼을 클릭하면 머티리얼 애셋 파일로 저장되고 동시에 수정된 내용이 반영되어 렌더링된다. 아직 저장하지 않고 수정된 내용을 반영하여 렌더링해서 확인해보려면 적용 버튼을 클릭하면 된다. 우리는 앞으로 머티리얼을 수정한 후에 수정된 내용을 확인하려면 저장 버튼이나 적용 버튼을 클릭하자.

<참고> 각 머티리얼 표현식 노드의 오른쪽 상단 모서리에 꺽쇠 아이콘이 있다. 위로 향하는 꺽쇠를 클릭하면 노드가 단순화되고 아래로 향하는 꺽쇠를 클릭하면 노드가 상세화되면서 프리뷰 모양을 보여준다.

7. 이제 머티리얼을 객체에 적용하는 방법에 대해서 알아보자.
레벨 에디터로 가자.
메뉴바에서 창 » 액터 배치를 선택하여 액터 배치 탭을 보이도록 하자. 액터 배치 탭에서 셰이프 아이콘을 클릭하고 큐브를 드래그해서 레벨에 대치하자. 배치된 큐브의 위치를 (300,0,50)으로 하고 회전을 (0,0,45)로 하자.
배치된 큐브를 선택하고 디테일 탭에서 머티리얼 속성을 찾아보자. 머티리얼 영역에 엘리먼트 0의 속성이 있다. 속성값으로 BasicShapeMaterial이 지정되어 있음을 알 수 있다.



메시의 머티리얼 속성에 대해서 알아보자.

모든 메시는 반드시 머티리얼을 가져야 한다. 머티리얼이 없다는 것은 렌더링할 수 없다는 것과 마찬가지의 의미이므로 모든 메시는 사용 여부와 무관하게 반드시 머티리얼을 가져야 한다. 머티리얼이 없는 메시는 있을 수 없다는 것을 기억하자.

만약 머티리얼이 준비되지 않은 메시가 있다면 엔진은 엔진에서 제공하는 회색 격자 패턴 모습의 **WorldGridMaterial**이나 흰색 모습의 **BasicShapeMaterial** 등을 디폴트로 지정해준다.

한편, 하나의 메시는 여러 머티리얼을 가질 수도 있다. 메시의 일부에 대해서는 한 머티리얼을 사용하고 메시의 다른 일부에 대해서는 다른 머티리얼을 사용하는 것이 가능하다. 이렇게 하나의 메시에 여러 머티리얼이 있을 수 있기 때문에 엔진에서는 머티리얼을 위한 여러 슬롯을 두고 있다. 이러한 슬롯을 **엘리먼트** 슬롯이라고 부른다. 슬롯의 인덱스는 0에서부터 시작한다. 첫 번째 머티리얼을 **엘리먼트 0** 슬롯에 저장하여 관리하고, 두 번째 머티리얼은 **엘리먼트 1** 슬롯에 저장하여 관리하는 식이다.

<참고> 언리얼에서는 최대 머티리얼의 개수에는 제한이 없다. 그러나 FBX 파일에서는 최대 64개로 제한하고 있고 다른 프로그램에서도 최대 개수의 제한이 있을 수 있다. 그러나 제한하더라도 충분히 큰 수이므로 불편할 일은 없을 것이다. 단순한 형태의 메시는 하나의 머티리얼로도 충분하다. 일반적인 메시의 경우에도 두 세개 정도의 머티리얼을 가진다.

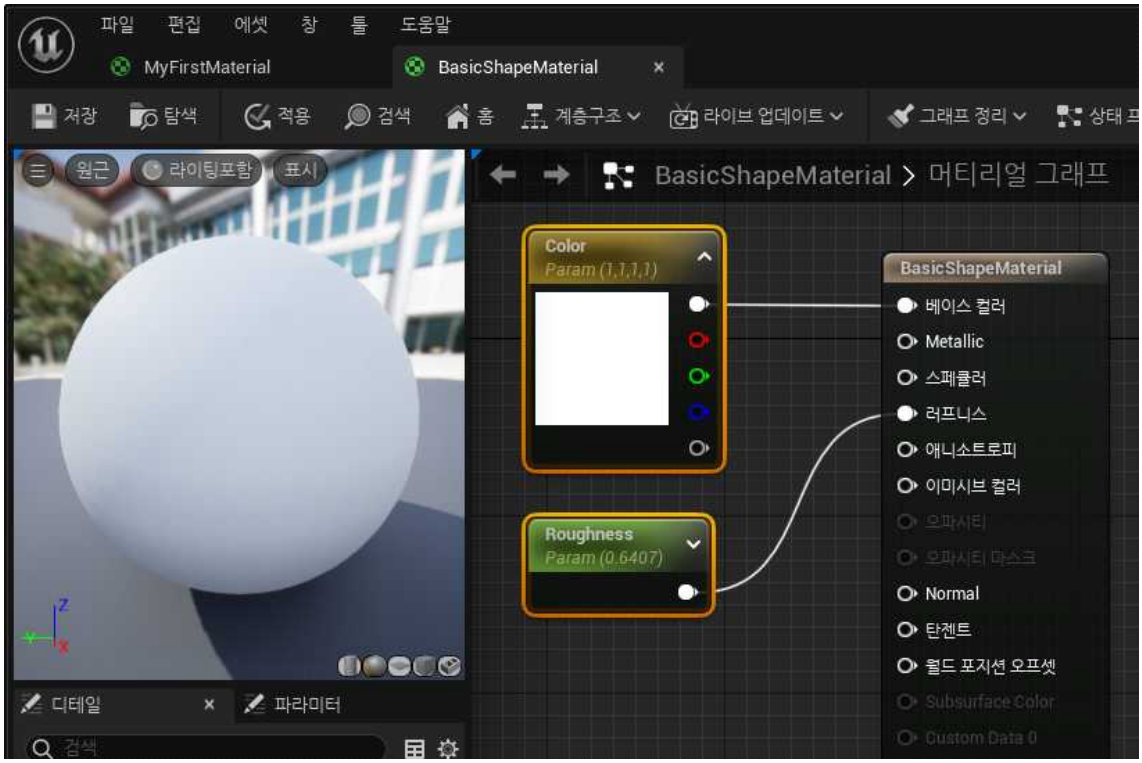
<참고> **액터 배치** 탭에서 **지오메트리** 아이콘을 클릭하고 추가할 수 있는 **브러시** 액터들은 메시와는 다르게 각 면마다 하나의 머티리얼을 지정할 수 있도록 되어 있다. 배치된 브러시의 면을 선택한 후에 엘리먼트 0 슬롯에 원하는 머티리얼을 지정하면 된다.

8. 디테일 탭에서 머티리얼 영역에 **엘리먼트 0**의 속성의 오른쪽에 흰색 구체가 보인다. 이것은 현재의 속성값 **BasicShapeMaterial**의 프리뷰 모습이다. 흰색 구체 아이콘을 더블클릭해보자. 머티리얼 에디터가 열린다.

머티리얼 에디터에서 머티리얼 그래프를 살펴보자. 매우 간단하게 구성되어 있다.

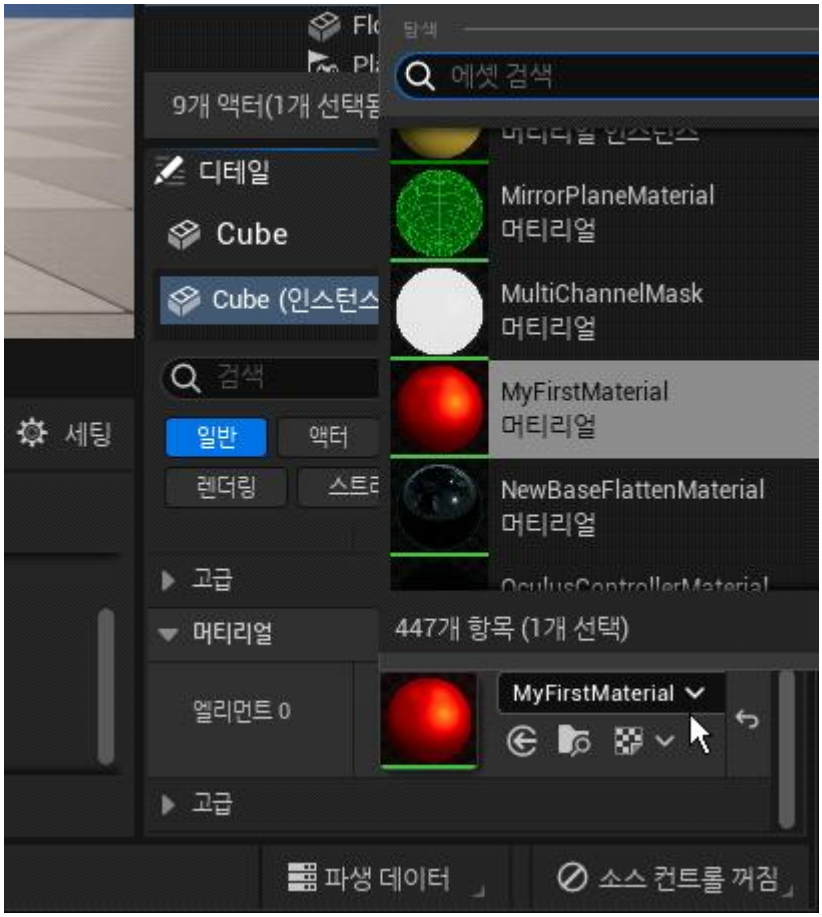
흰색 컬러를 표현하는 **Constant3Vector** 노드가 메인 머티리얼 노드의 베이스 컬러 입력핀에 연결되어

있고, 0.6407의 실수값을 가지는 **Constant** 노드가 러프니스 입력핀에 연결되어 있다.
이제 **BasicShapeMaterial**의 머티리얼 에디터를 닫자.

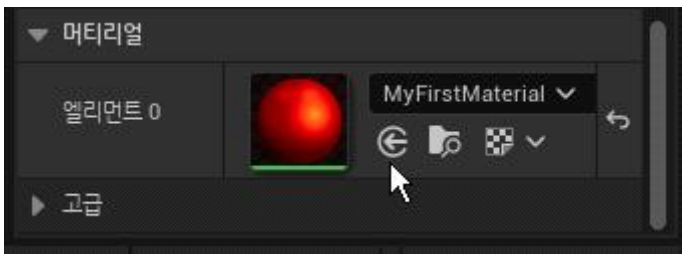


<참고> 러프니스는 표면의 거칠고 부드러운 정도를 제어한다. 표면이 거칠면 반사된 빛이 여러 방향으로 퍼지므로 리플렉션이 희미하거나 선명한 정도나 스페큘러 하이라이트가 좁고 집중되어 있는지 또는 넓게 퍼져있는지에 정도에 영향을 미친다. 러퍼니스 값은 [0,1] 사이의 값으르 0이 완전히 부드러운 거울 반사에 대항하고 1이 완전히 거친 무광 또는 난반사(diffuse)에 해당한다.

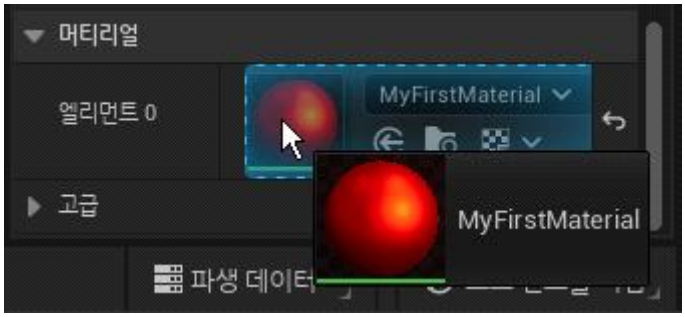
9. 레벨 에디터에서 **Cube** 액터를 선택하고 디테일 탭에 가자. **머티리얼** 영역에 **엘리먼트 0**의 속성값을 바꾸는 방법에 대해서 알아보자.
현재 지정되어 있는 속성값인 **BasicShapeMaterial**를 클릭하고 드롭다운 메뉴에서 **MyFirstMaterial**을 검색하여 선택하자.



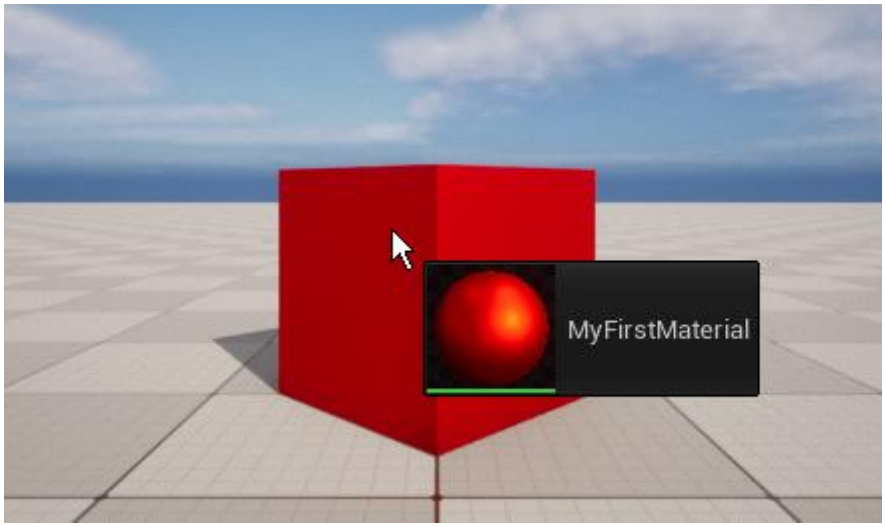
10. 이번에는 다른 방법으로 머티리얼을 바꾸어보자. 먼저 **Ctrl+Z**를 눌러 이전의 속성값 지정을 취소하고 속성값을 원래대로 **BasicShapeMaterial**로 되돌려놓자.
이번에는 먼저 콘텐츠 브라우저에서 **MyFirstMaterial** 애셋을 좌클릭하여 선택해두자. 그다음, 배치된 큐브 액터의 디테일 탭에서 **엘리먼트 0**의 속성값의 흰색 구체 모양 옆의 왼쪽 화살표 아이콘을 클릭하자. 이전과 마찬가지로 우리의 머티리얼이 적용되는 것을 확인할 수 있다.



11. 이번에는 또다른 방법으로 머티리얼을 바꾸어보자. 먼저 **Ctrl+Z**를 눌러 이전의 속성값 지정을 취소하고 속성값을 원래대로 **BasicShapeMaterial**로 되돌려놓자.
이번에는 레벨에서 큐브 액터가 선택된 상태로 두고, 콘텐츠 브라우저에서 **MyFirstMaterial** 애셋을 좌클릭하고 드래그하기 시작하자. 드래그가 시작되면 디테일 탭의 **엘리먼트 0**의 속성값 부분이 녹색 점선으로 하이라이트될 것이다. 그곳에 드롭할 수 있음을 의미한다. 드롭해보자. 이전과 마찬가지로 우리의 머티리얼이 적용되는 것을 확인할 수 있다.



12. 이번에는 또다른 방법으로 머티리얼을 바꾸어보자. 먼저 **Ctrl+Z**를 눌러 이전의 속성값 지정을 취소하고 속성값을 원래대로 **BasicShapeMaterial**로 되돌려놓자.
이번에는 콘텐츠 브라우저에서 **MyFirstMaterial** 애셋을 좌클릭하고 드래그하여 레벨의 큐브 위에 바로 드롭해보자. 이전과 마찬가지로 우리의 머티리얼이 적용되는 것을 확인할 수 있다.



지금까지 머티리얼을 메시에 적용하는 다양한 방법을 알아보았다. 어느 것을 사용해도 좋으니 편리한 방법을 선택하여 사용하면 된다.

13. 이제까지는 레벨 에디터에서 머티리얼을 수정하는 방법에 대해서 알아보았다. 레벨 에디터에 배치된 액터의 머티리얼을 바꾸면 그 레벨에서만 바뀐 머티리얼이 유지된다. 즉 메시의 사본에만 영향을 주고 메시의 원본은 변하지 않는다.
메시의 원본을 바꾸려면 스태틱 메시 에디터나 스켈레탈 메시 에디에서 머티리얼을 지정하면 된다. 콘텐츠 브라우저에서 메시 에셋을 더블클릭하면 메시 에디터가 열린다. 또는 배치된 아웃라이너에서 해당 액터를 선택하고 Ctrl+E를 입력해도 메시 에디터가 열린다.
우리는 아웃라이너에서 Cube 액터를 선택하고 Ctrl+E를 입력해보자. 스태틱 메시 에디터가 열릴 것이다.
스태틱 메시 에디터나 스켈레탈 메시 에디터에서 머티리얼을 수정하는 방법은 이전과 동일하다. 스태틱 메시 에디터의 경우에는 오른쪽의 디테일 탭에서 **머티리얼 슬롯**(**Material Slots**) 영역에 메시의 머티리얼들이 나열되어 있다. 아래의 그림과 같이 스태틱 메시 에디터에서의 머티리얼 슬롯에 드롭해서 머티리얼을 수정할 수 있다.
스켈레탈 메시 에디터의 경우에는 왼쪽의 애셋 디테일 탭에서 **Material Slots** 영역에 메시의 머티리얼들이 나열되어 있다.

메시 에디터에서 머티리얼을 수정하면 메시 애셋 자체의 머티리얼이 수정된다. 우리는 테스트만 해보고 실제로 저장은 하지 않도록 하자.

이 절에서는 머티리얼을 처음으로 만들어 학습하였다.

3. 머티리얼 파라미터와 머티리얼 인스턴스

이 절에서 머티리얼 파라미터와 머티리얼 인스턴스에 대해서 학습한다.

먼저. **머티리얼 파라미터**(Material Parameter)에 대해서 알아보자.
머티리얼을 생성하고 이를 원하는 렌더링이 되도록 수정하는 작업은 시간이 많이 소모된다. 머티리얼 그래프를 수정하면서 이를 적용해보는 작업을 반복해야 하기 때문이다. 특히 수치값을 계속 조정하면서 확인하는 작업을 반복해야 하는 경우가 많다.
이러한 수치값 조정의 경우에 머티리얼을 컴파일하지 않고도 조정하는 수치값이 바로 렌더링에 적용할 수 있도록 하여 머티리얼 개발을 편리하게 하는 기능이 머티리얼 파라미터이다.

다음으로, **머티리얼 인스턴스**(Material Instance)에 대해서 알아보자.
여러 머티리얼을 제작할 때에 많은 부분들이 공통적이고 일부만 다른 경우가 흔하다. 베이스 컬러와 러프니스만 다르고 나머지는 모두 동일한 머티리얼을 여러개를 제작한다고 하자. 약간씩만 다른 머티리얼을 모두 제작하는 것은 비효율적이다. 이런 경우에 모체가 되는 머티리얼을 하나만 일반적인 방법으로 제작한다. 이때, 변할 수 있는 베이스 컬러와 러프니스 부분에 대해서는 머티리얼 파라미터로 해둔다. 그다음, 이 표준 머티리얼을 상속받아서 특별한 머티리얼 유형인 머티리얼 인스턴스를 만들면 된다. 머티리얼 인스턴스에서는 상속받은 머티리얼 파라미터를 수정해서 다른 머티리얼처럼 사용할 수 있다. 머티리얼 인스턴스를 만들 때에는 부모인 표준 머티리얼을 컴파일하거나 수정할 필요가 없다.

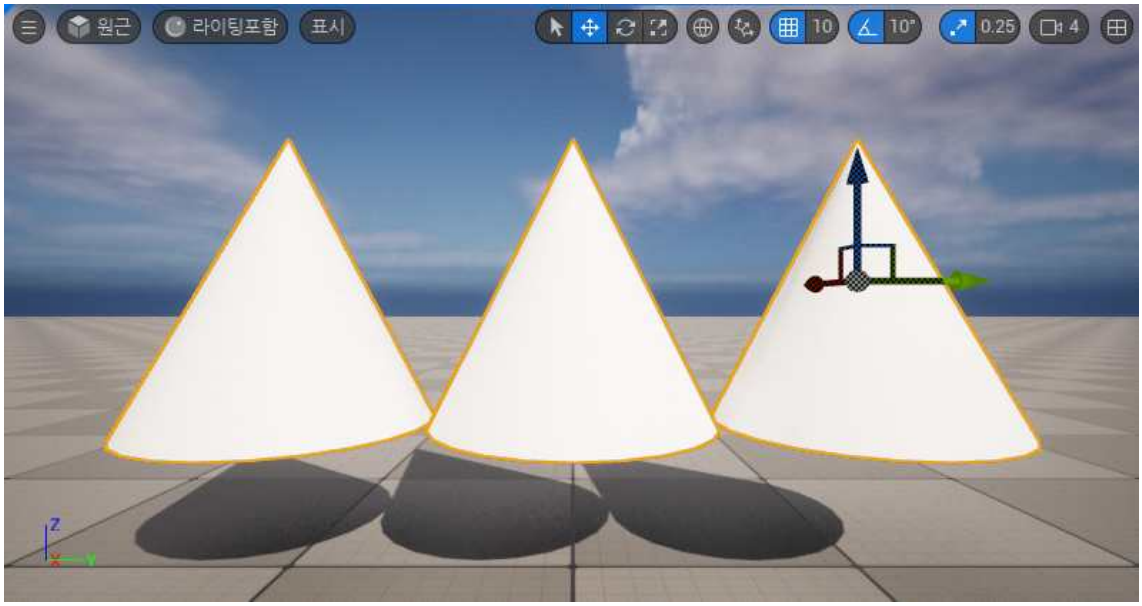
<참고> 머티리얼 인스턴스에 대한 자세한 내용은 다음의 문서를 참조하자.
https://docs.unrealengine.com/instanced-materials-in-unreal-engine/
머티리얼 인스턴스를 생성하고 사용하는 방법에 대한 내용에 대해서는 다음의 문서를 참조하자.
https://docs.unrealengine.com/creating-and-using-material-instances-in-unreal-engine/

이제부터 예제를 통해서 학습해보자

---

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

2. 레벨 에디터로 가자. **액터 배치** 탭에서 **셰이프** 아이콘을 클릭하고 **원뿔**을 드래그해서 레벨에 배치하자. 세 개의 원뿔을 배치하자. 이름을 각각 **ConeLeft**, **ConeMiddle**, **ConeRight**라고 하자. 배치된 **원뿔**의 **위치**를 각각 (200,-100,100), (200,0,100), (200,100,100)으로 하자.



3. 머티리얼 에셋을 생성하자. **콘텐츠 브라우저**에서 **+추가** 버튼을 클릭하고 **머티리얼**을 선택하자. 생성된 머티리얼의 이름을 **MyFirstMaterial**로 수정하자. **MyFirstMaterial**를 더블클릭하여 머티리얼 에디터를 열자.

이전 예제에서 만들었던 붉은색 머티리얼을 이번에는 머티리얼 파라미터로 만들어보자.

먼저, 머티리얼 그래프의 격자판에서 단축키인 **3+좌클릭**하여 **Constant3Vector** 노드를 배치하자.

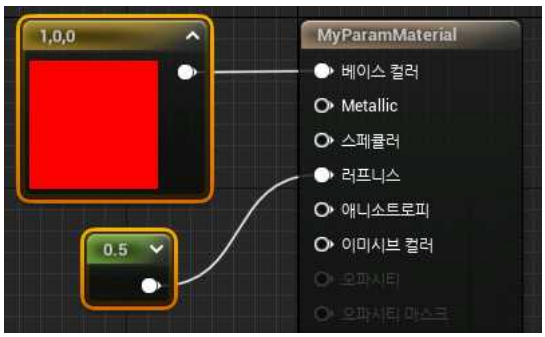
그다음, **Constant3Vector** 노드의 디테일 탭에서 **Constant** 속성의 R,G,B값을 1,0,0으로 수정하자.

그다음, **Constant3Vector** 노드를 메인 머티리얼 노드의 **베이스 컬러** 입력핀에 연결하자.

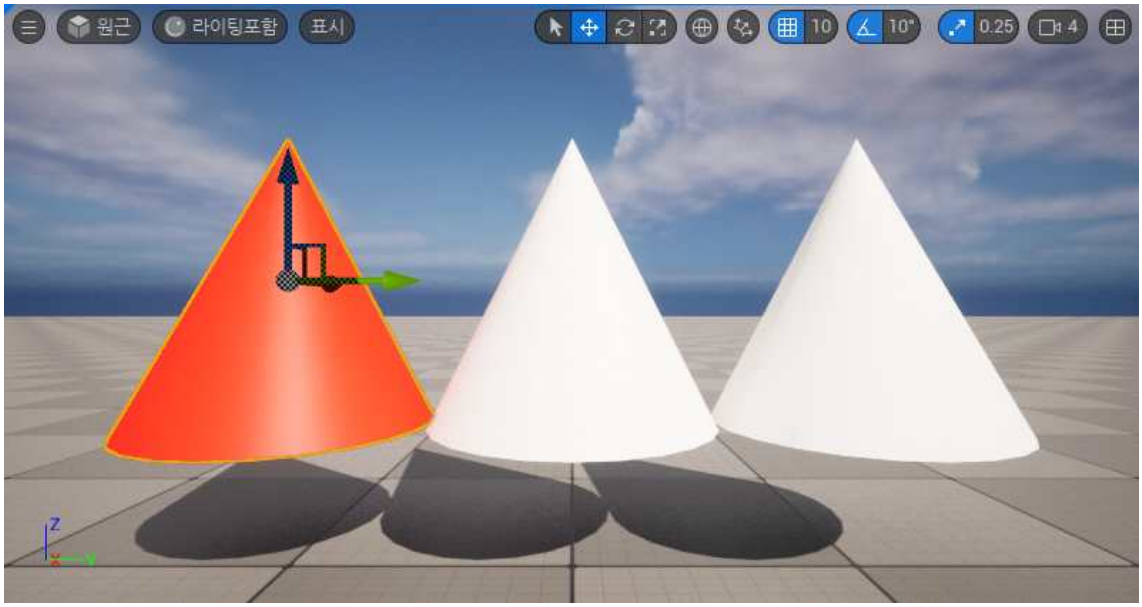
그다음, **1+좌클릭**하여 **Constant** 노드를 배치하자.

그다음, **Constant** 노드의 디테일 탭에서 **값(Value)** 속성값을 0.5로 수정하자.

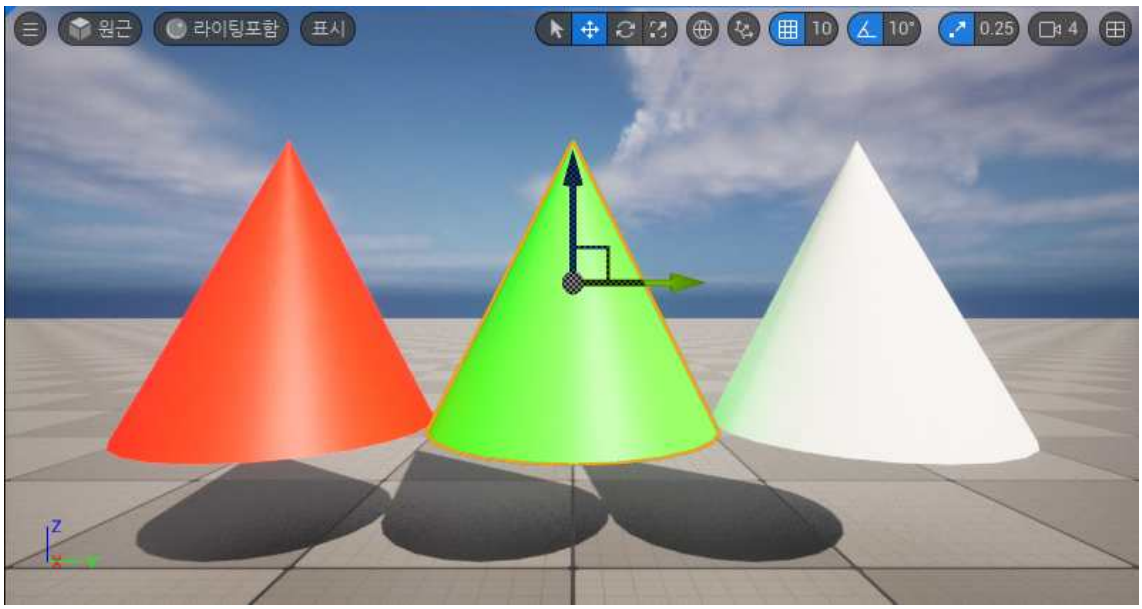
그다음, **Constant** 노드를 메인 머티리얼 노드의 **러프니스** 입력핀에 연결하자. 이제 간단한 붉은색 머티리얼이 완성되었다. 툴바의 저장 버튼을 클릭하여 저장하자.



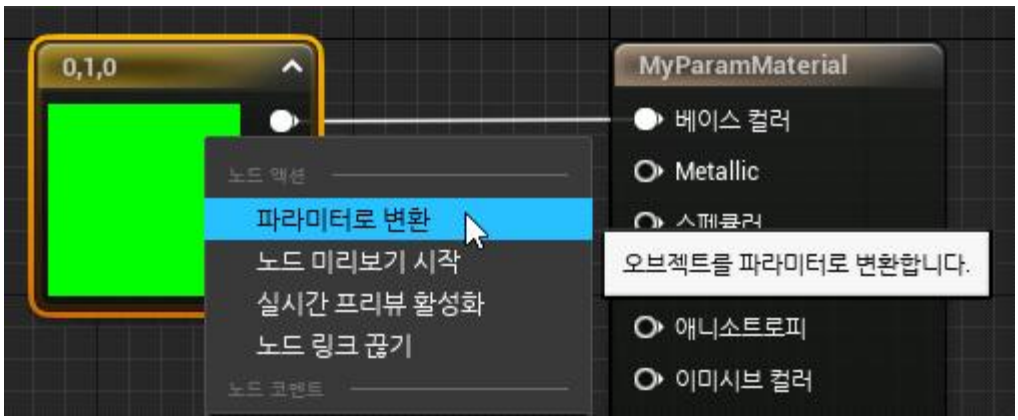
4. 레벨 에디터로 가자. 콘텐츠 브라우저에서 **MyFirstMaterial**를 드래그하여 배치된 **ConeLeft**에 드롭하자. 왼쪽 원뿔이 빨간색으로 바뀔 것이다.



5. 콘텐츠 브라우저에서 **MyFirstMaterial**을 선택하고 **Ctrl+C**를 누르고 **Ctrl+V**를 눌러 머티리얼을 복사하자. 복사된 머티리얼의 이름을 **MyParamMaterial**로 수정하자.
그다음, **MyParamMaterial**를 드래그하여 배치된 **ConeMiddle**에 드롭하자. 원뿔이 빨간색으로 바뀔 것이다.
그다음, **MyParamMaterial**를 더블클릭하여 머티리얼 에디터를 열자. 그리고, 레벨 에디터와 머티리얼 에디터를 동시에 볼 수 있도록 창을 배치하자.
그리고, **Constant3Vector** 노드의 값을 (0,1,0)으로 수정해보자. 머티리얼 에디터에서는 프리뷰가 녹색으로 바뀔 것이다. 한편, 레벨 에디터를 관찰해보자. 레벨 에디터에서는 **ConeMiddle**가 여전히 빨간색으로 남아있을 것이다.
그다음, 툴바의 **저장** 버튼이나 **적용** 버튼을 클릭하자. 이제, 머티리얼을 컴파일하고 레벨에 적용된다. 레벨에서의 중간의 원뿔이 녹색으로 바뀔 것이다.
표준 머티리얼의 경우에는 머티리얼이 다시 컴파일되어야 레벨에 적용되는 것을 확인하였다.



6. 이제, 배치된 두 머티리얼 표현식 노드를 머티리얼 파라미터 노드로 바꾸어보자.
머티리얼 에디터에서 **Constant3Vector** 노드 위에서 우클릭하고 드롭다운 메뉴에서 **파라미터로 변환**을 선택하자. 이름을 **ParamBaseColor**로 바꾸자.
머티리얼 표현식 노드의 경우에는 노드를 다른 곳에서 언급하거나 참조할 필요가 없으므로 노드의 이름 항목이 없다. 그러나 머티리얼 파라미터 노드의 경우에는 파라미터 이름인 **파라미터 이름(Parameter Name)** 속성이 있다. 다른 곳에서 참조할 수 있도록 이 속성값에 의미있는 이름을 지정해주어야 한다. 디폴트로 정해지는 이름을 그대로 두지 말고 의미있는 이름으로 바꾸어 주도록 하자. 이름을 바꾸기 위해서는 노드의 이름 부분을 클릭하거나 또는 디테일 탭에서 **파라미터 이름(Parameter Name)** 속성값에 입력하면 된다.



<참고> **Constant3Vector** 노드 뿐만 아니라 다양한 종류의 머티리얼 표현식 노드에 대해서 머티리얼 파라미터 노드로 전환할 수 있다. 머티리얼 파라미터 노드로 전환이 가능한지를 확인하는 가장 쉬운 방법은 배치된 머티리얼 표현식 노드에서 우클릭하고 드롭다운 메뉴에서 **파라미터로 변환**을 선택 옵션이 보이는지를 확인하는 방법이다. 이 옵션이 보인다면 머티리얼 파라미터 노드로 전환이 가능하다는 의미이다.

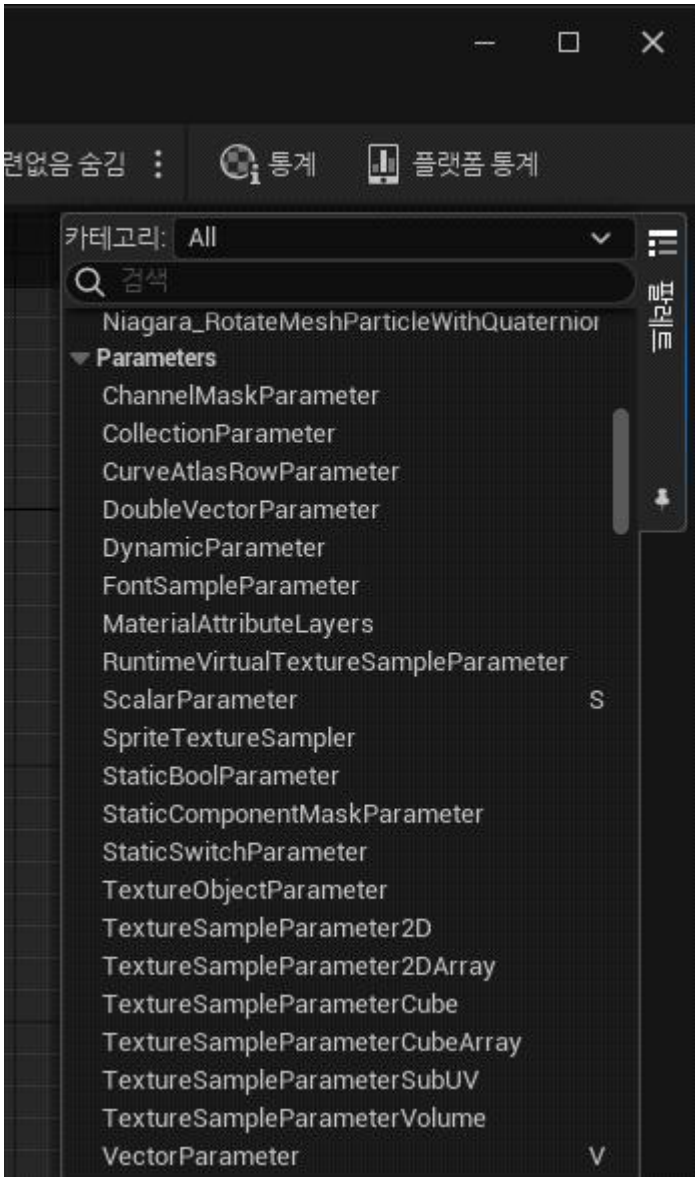
7. 그다음, **Constant** 노드도 동일한 방법으로 **파라미터로 변환**을 선택하고 이름을 **ParamRoughness**로 바꾸자.

저장하자. 이제 머티리얼 파라미터 노드가 포함된 머티리얼을 완성하였다.

<참고> **ParamBaseColor** 노드의 출력핀을 보면 흰색, 빨간색, 녹색, 파란색, 회색의 핀이 있다. 가장 위의 흰색 핀은 RGBA의 모든 채널이 합쳐진 컬러이다. 그 아래의 핀은 각각 R,G,B,A의 단일 채널에 해당한다.

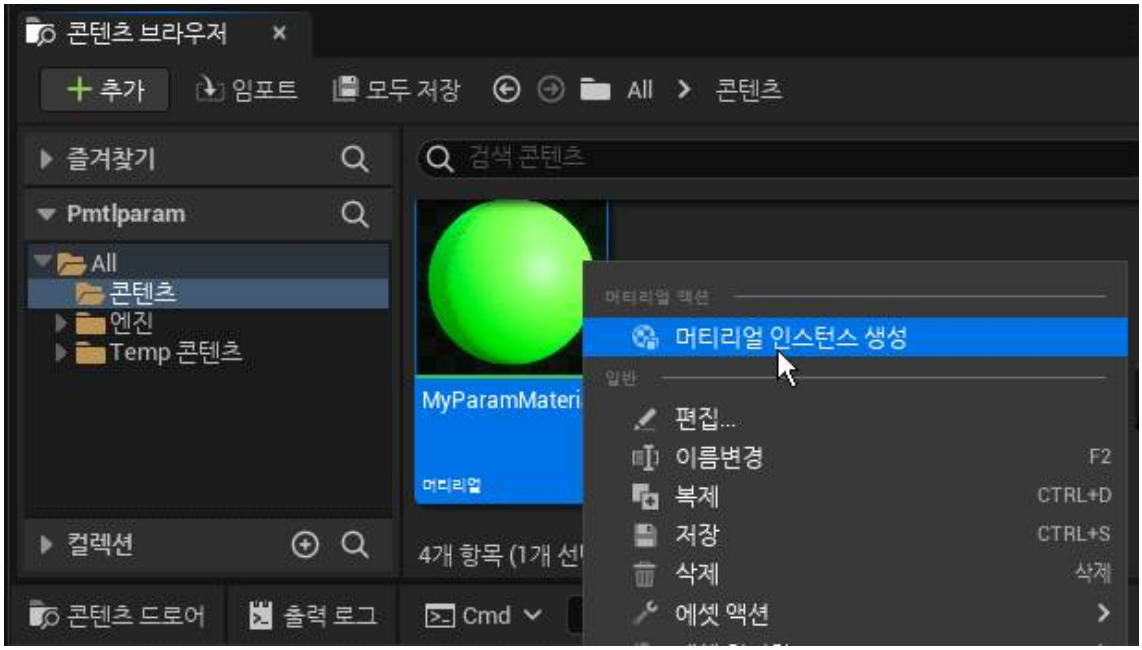
8. 이제 레벨 에디터에서 테스트해보자.
레벨 에디터와 머티리얼 에디터를 동시에 볼 수 있도록 창을 배치하자.
그리고, **MyParamMaterial**의 머티리얼 에디터에서 **ParamBaseColor**로 노드를 더블클릭하자. **색상 선택 툴**이 뜰 것이다. 컬러를 계속 바꾸어보자. 레벨 에디터에서 컬러가 즉시 적용되는 것을 확인할 수 있다.
그다음, 레벨 에디터에서 플레이해보자. 그리고, 다시 머티리얼 에디터에서 **색상 선택 툴**로 컬러를 계속 바꾸어보자. 플레이 중에도 컬러가 바뀌는 것을 확인할 수 있다.
그다음, 머티리얼 에디터에서 **ParamRoughness** 노드를 선택하고
그리고 디테일 탭에서 **Default Value**의 속성값을 0에서 1 사이에서 계속 바꾸어보자. 레벨 에디터를 관찰하면서 **러퍼니스**가 어떤 영향을 미치는지 살펴보자.

9. 위에서는 기존에 배치된 머티리얼 표현식 노드에서 **파라미터로 변환** 메뉴를 선택해서 머티리얼 파라미터 노드를 만들었다. 이번에는 머티리얼 파라미터를 추가하는 또다른 방법을 알아보자.
먼저, 팔레트 탭을 사용하는 방법이다. 팔레트 탭을 열고 **Parameters** 영역에 나열된 노드들을 살펴보자. 모든 머티리얼 파라미터 노드가 나열되어 있다. 이 중에서 **VectorParameter**와 **ScalarParameter**를 하나씩 배치하자. 이 두 노드에 대해서는 단축키가 배정되어 있다. 단축키 **V+좌클릭**과 **S+좌클릭**을 하면 바로 배치된다.
팔레트 탭을 사용하는 대신 액션선택 창을 사용해도 된다. 격자판에서 우클릭하고 액션선택 창에서 **Parameters** 영역의 나열된 노드 중에서 선택하거나 또는 검색하여 선택하여 노드를 배치하면 된다.
팔레트 탭이나 액션선택 창을 통해 노드를 배치한 후에는 이전과 같이 하면 된다. 즉, 각각 이름을 **ParamBaseColor**와 **ParamRoughness**로 바꾸고 메인 머티리얼 노드의 **베이스 컬러** 입력핀과 **러프니스** 입력핀에 연결하자. 그리고, 각각의 노드를 클릭하고 디폴트 값을 (0,1,0,1)과 0.5로 수정하여 이전과 동일하게 만들 수 있다.
이번 단계에서 진행한 것은 연습으로만 진행하고 모두 취소하자.



<참고> 표준 머티리얼의 경우에는 베이스 컬러에 3차원 벡터를 입력하였지만 이번에는 4차원 벡터를 입력하였다. 벡터를 표현하는 머티리얼 파라미터 노드에는 4차원 벡터에 대한 노드만 있기 때문이다. 3차원 벡터를 취하는 베이스 컬러에 4차원 벡터가 연결되는 경우에는 필요없는 마지막 성분은 무시하고 입력된다.

10. 지금까지 **머티리얼 파라미터**에 대해서 알아보았다.
지금부터는 **머티리얼 인스턴스**에 대해서 알아보자. **머티리얼 파라미터**의 강력한 장점은 **머티리얼 인스턴스**에서 나타난다.
콘텐츠 브라우저에서 **MyParamMaterial** 위에서 우클릭하고 드롭다운 메뉴에서 **머티리얼 인스턴스 생성**을 클릭하자. 생성된 머티리얼 인스턴스의 이름은 디폴트로 정해지는 **MyParamMaterial_Inst**로 두자.

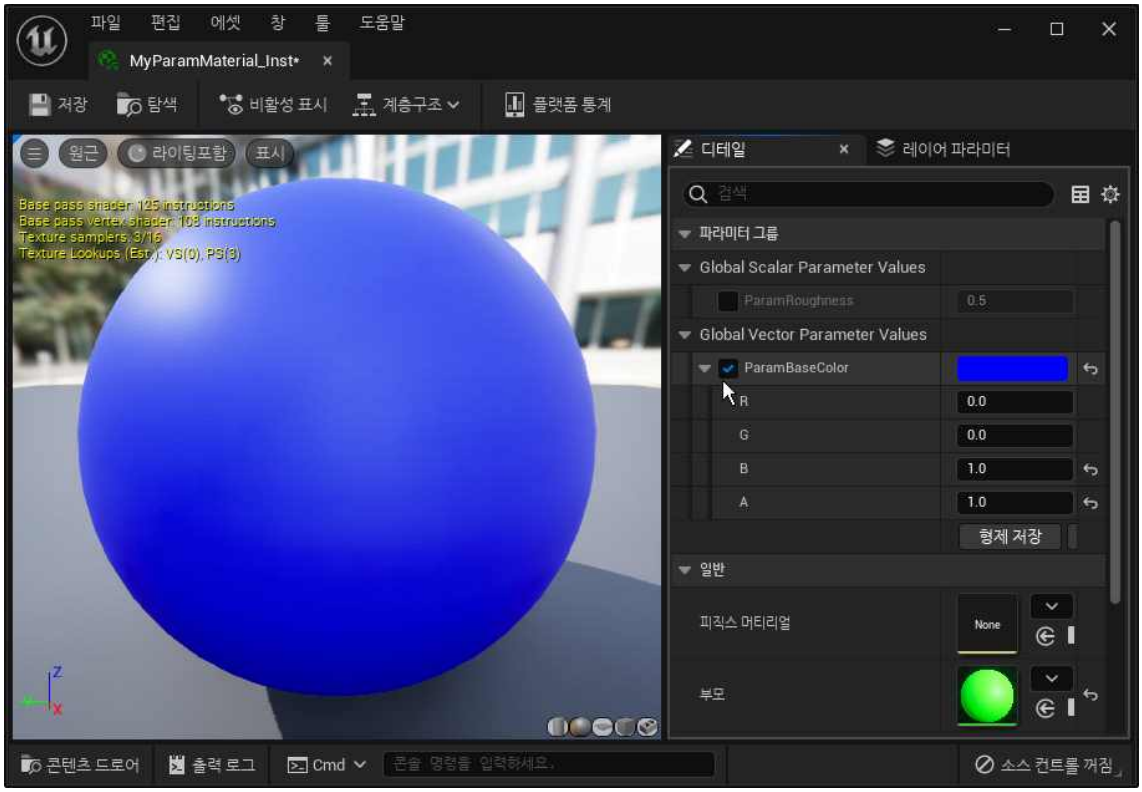


콘텐츠 브라우저에서 **MyParamMaterial_Inst**를 드래그하여 배치된 **ConeRight**에 드롭하자. 오른쪽 원뿔이 녹색으로 바뀔 것이다.

11. 생성된 머티리얼 인스턴스인 **MyParamMaterial_Inst**로를 더블클릭하여 머티리얼 인스턴스 에디터를 열자. 머티리얼 인스턴스 에디터는 기존의 머티리얼 에디터보다는 간단하게 보인다. 프리뷰에서의 모습은 이전의 머티리얼과 동일하게 녹색으로 표시될 것이다.

오른쪽에 디테일 탭을 보자. **파라미터 그룹** 영역의 **Global Scalar Parameter Values**과 **Global Vector Parameter Values** 영역의 아래에 **ParamRoughness**와 **ParamBaseColor**가 나타나 있다. 이 속성이 바로 부모 머티리얼에서의 머티리얼 파라미터 노드에 해당한다. 즉, 머티리얼 인스턴스에서 부모 머티리얼의 파라미터 노드의 값을 바꿀 수 있는 것이다.

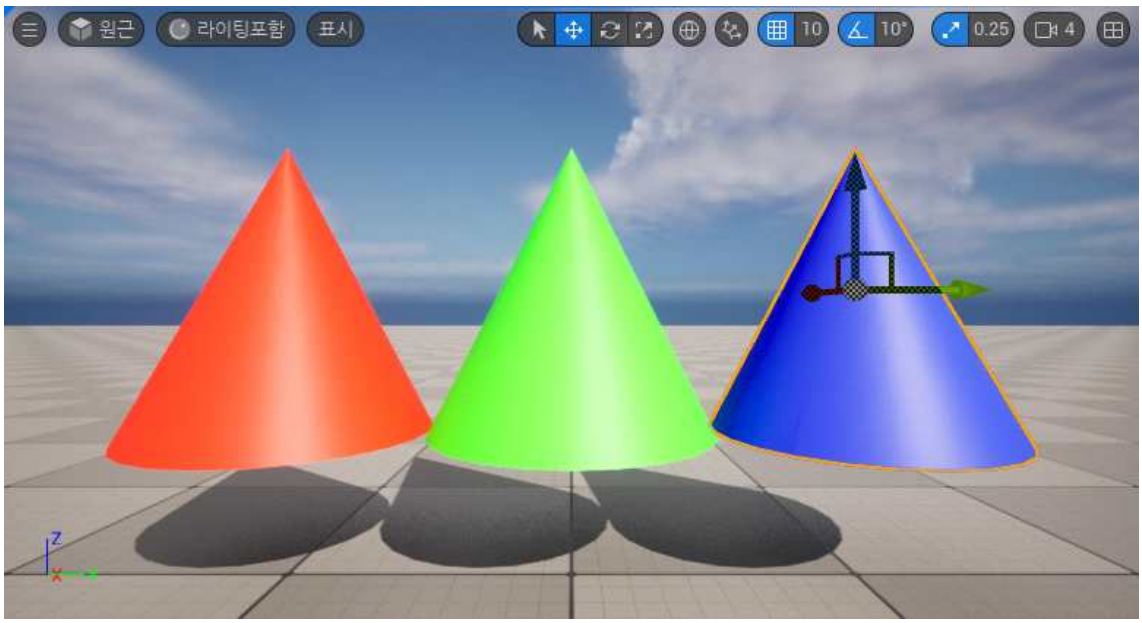
우리는 컬러를 파란색으로 바꾸어보자. **ParamBaseColor**의 체크박스에 체크하자. 이제 디폴트 값을 수정할 수 있다. R,G,B,A의 값을 0,1,0,1에서 0,0,1,1로 수정하자. 파란색으로 바뀔 것이다.



12. ParamBaseColor의 값을 바꾸면서 레벨 에디터의 뷰포트를 확인해보자. 오른쪽 원뿔의 컬러가 즉시 바뀌는 것을 확인할 수 있다. 즉, 부모 머티리얼에서 확인하였던 머티리얼 파라미터 노드의 기능이 그대로 작동되고 있음을 확인할 수 있다.

이제 저장하자.

우리는 아주 간단한 방식으로 머티리얼 인스턴스를 만들고 이를 머티리얼과 동일하게 사용하였다. 머티리얼 그래프 탭에서 노드 네트워크를 만드는 과정을 모두 생략하고 단지 디테일 탭에서 일부 값만 수정하여 머티리얼 인스턴스를 간편하게 생성하였다.



이 절에서는 머티리얼 파라미터와 머티리얼 인스턴스를 학습하였다.

4. 베이스 컬러 입력 사용하기

지금부터는 각 메인 머티리얼 노드의 각 입력핀에 대해서 학습한다.
이 절에서 베이스 컬러 입력핀에 대해서 학습한다.

먼저 물체 표면의 모습에 대해서 기본 개념을 학습해보자.
표면이 어떤 컬러로 보이는 지에 대해서 표면의 빛의 성분을 세 성분으로 분리하여 고려한다. 세 성분은 **디퓨즈**, **앰비언트**, **스페큘러**이다. 이는 물리학적인 접근이 아니라 구현을 위한 컴퓨터학적인 접근이다. 또한 고전적인 접근에서의 용어이고 PBR에서는 점점 사라지고 있는 용어이다. 그러나 반드시 알고 있어야 하는 용어이므로 알아보도록 하자.
물체의 표면이 어떤 컬러로 보이는지에 대해서는 기본적으로는 물체의 표면의 고유한 컬러 특성에 따라 정해진다. 이것을 **디퓨즈**(diffuse) 컬러 성분이라고 한다. 물체의 재질 특성이 빨간색 표면이라면 빨간색을 반사하게 된다. 빛이 표면에 입사된 후에 해당하는 하나의 반사각으로만 반사하는 것이 아니라 모든 각도로 분산되어 반사된다. 따라서 전면에서의 모든 각도에서 보이게 된다. 그러나 빛이 입사하는 각도와 표면이 향해있는 각도에 따라서 카메라로 들어오는 빨간색의 강도가 달라진다. 그러나 빛이 모든 방향으로 반사되므로 카메라가 표면을 바라보는 각도와는 무관하다.
한편, 환경에서의 빛은 수많은 반사를 통해서 다시 입사되므로 광원의 정확한 출처를 밝히기가 어렵다. 이러한 빛들에 대해서 전체적으로 물체를 덮고 있는 빛으로 단순하게 **앰비언트**(ambient) 컬러 성분으로 처리한다. 이 성분은 각도에 상관없이 일정한 강도와 컬러로 표현한다.
표면이 매끄러운 재질의 경우에는 빛이 입사되면 한 반사 각도로 좁은 범위로 강하게 반사된다. 관찰자에게는 표면의 일부 영역이 밝게 반짝이는 것으로 보이게 된다. 이러한 성분을 **스페큘러**(specular) 컬러 성분이라고 한다. 빛이 입사하는 각도와 물체의 표면 각도뿐만 아니라 추가적으로 카메라의 시선 방향도 연관되어 결정된다. 반사되는 성분의 흰색으로 보인다.
용어에 있어서, 디퓨즈를 분산광으로 앰비언트를 주변광으로, 스페큘러를 반사광으로 지칭하기도 한다.
디퓨즈, **앰비언트**, **스페큘러**의 세 성분 중에서 가장 중요한 성분은 **디퓨즈** 성분이고 가장 덜 중요한 성분은 **스페큘러** 성분이다. 특히 **디퓨즈** 성분은 매우 중요한 성분으로 재질이 보이는 모양의 대부분을 결정한다.
메인 머티리얼 노드의 입력핀 중의 하나인 **베이스 컬러** 입력핀은 머티리얼의 전반적인 컬러를 정의한다. **베이스 컬러**는 원칙적으로 **디퓨즈** 성분을 나타내고 **스페큘러** 성분은 제외된다. 우리는 앞으로 **베이스 컬러**는 **디퓨즈** 성분과 동일하다고 생각하자.

이 절에서는 텍스처에 대해서도 처음으로 다룰 것이다.
텍스처에 대해서 간단하게 알아보자. 텍스처는 단순히 픽셀의 2차원 배열 데이터인 이미지 데이터이다. 텍스처의 각 픽셀은 컬러값을 표현하기도 하고 깊이 정보나 노말 정보와 같은 다른 정보를 표현하기도 한다. 따라서 텍스처는 범용으로 사용되는 데이터 구조로 생각해야 한다.
머티리얼에서는 텍스처를 사용해서 복잡한 기능의 구현에 활용한다.
텍스처를 엔진에 임포트하게 되면 **텍스처** 타입의 애셋으로 있게 된다. 머티리얼에서 텍스처를 사용하더라도 텍스처가 머티리얼 내부에 포함되지 않는다.
텍스터는 머티리얼 그래프에서의 **TextureSample** 노드와 같은 특수 머티리얼 표현식 노드를 통해서 머티리얼로 들어온다. 언리얼에서는 머티리얼 내에 텍스처를 포함시킬 수 있는 방법이 없고 항상

머티리얼과 별개로 존재한다.

이제 컬러 정보를 머티리얼에 제공하는 방법에 대해서 알아보자.
물체 표면은 보통 단색이 아니고 무늬나 얼룩이나 스크래치 등이 있는 불규칙한 경우가 흔하다. 이러한 표면의 컬러 정보는 일일이 각 지점에 컬러 정보를 지정하는 대신에 텍스처를 사용하여 표면의 컬러 정보를 표현한다. 텍스처의 해상도가 낮아도 샘플링하여 모든 표면의 지점에 매핑되도록 한다.
텍스처가 표면의 컬러 특성을 표현한다면 텍스처의 각 픽셀값이 RGB로 해석되어 표면의 컬러 성분을 계산하는데 사용된다. 이러한 표면의 기본 컬러 정보는 **메인 머티리얼 노드**의 **베이스 컬러** 입력핀에 연결하면 된다. 따라서 표면의 컬러 정보를 표현하는 텍스처를 **베이스 컬러** 입력핀에 연결해주면 표면의 컬러 정보 입력이 완료된다. 표면이 실제로 어떻게 보일 지에 대해서는 조명의 위치와 각도, 물체의 각도, 카메라의 위치와 각도 등의 여러 요소들에 따라서 달라지면 이는 렌더링 엔진이 모두 처리해준다.
텍스처 데이터가 표면의 노말 방향 정보를 표현할 수도 있다. 표면의 각 지점의 컬러를 일일이 표현하는 것이 어려운 일인 것과 같이 컬러 이외의 다른 정보를 일일이 표현하는 것이 어려운 일이다. 표면의 노말 정보는 최종 컬러를 결정하기 위해서 디퓨즈 성분이나 스페큘러 성분을 계산하는데 사용된다. 텍스처가 노말 정보를 표현하는 경우에는 텍스처 픽셀은 RGB가 아니라 노말 벡터인 XYZ으로 해석된다.

참고로, PBR 개념이 등장하기 전부터 사용해온 용어로 인한 혼란이 있을 수 있다.
텍스처를 제작할 때에 텍스처 내용에 따라서 디퓨즈 텍스처 맵, 노말 텍스처 맵, 스페큘라 텍스처 맵 등의 용어를 흔히 사용하였다. 실제로 디퓨즈 텍스처 맵은 조명의 영향을 배제하고 베이스 컬러의 개념으로 제작하는 경우가 흔하다. 따라서 디퓨즈 텍스처가 주어진 경우에는 이를 **베이스 컬러** 입력핀에 연결해주면 된다.
텍스처 애셋의 이름은 보통 접두사 **T_**로 시작한다. 그리고 접미사 **_D**를 붙이는 경우에는 텍스처가 디퓨즈 컬러 성분을 표현하고 있음을 의미한다. 텍스처 애셋의 이름에 접미사 **_N**가 붙은 경우에 텍스처가 노말 성분을 표현하고 있음을 의미한다.

이 절의 예제에서는 다섯 개의 메시 액터를 레벨에 배치한다. 그리고 다섯 개의 머티리얼을 만들어 각 메시 액터에 지정할 것이다. 모든 머티리얼은 **베이스 컬러** 입력핀만을 사용할 것이다.
첫 번째로 만들 머티리얼인 **M_BaseColor_Color**은 빨간색을 베이스 컬러 입력핀에 지정한다.
두 번째는 머티리얼 인스턴스인 **M_BaseColor_Color_Inst**이며, 첫 번째 머티리얼을 상속하여 머티리얼 인스턴스를 만든 후에, 빨간색을 녹색으로 바꾸어 인스턴스를 완성한다.
세 번째는 머티리얼 **M_BaseColor_Texture**이며, 잔디 텍스처를 베이스 컬러 입력핀에 지정한다.
네 번째는 머티리얼 인스턴스인 **M_BaseColor_Texture_Inst**이며, 세 번째 머티리얼을 상속하여 머티리얼 인스턴스를 만든 후에, 잔디 텍스처를 벽돌 텍스처로 바꾸어 인스턴스를 완성한다.
다섯 번째는 머티리얼 **M_BaseColor_Texturecomp**이며, 두 개의 텍스처를 사용한다. 간단한 사칙 연산으로 두 텍스처를 혼합하여 그 결과를 **베이스 컬러** 입력핀에 지정한다.

<참고> 에픽 게임즈에서 학습용으로 **Content Examples**라는 프로젝트를 제공한다. 이 프로젝트는 Epic Games Launcher를 실행하고 **학습** 탭에 가면 찾을 수 있다. **Content Examples** 프로젝트에는 머티리얼 뿐만 아니라

엔진의 전 영역에 대한 다양한 종류의 예제가 포함되어 있다. 이 프로젝트에 대한 설명 문서는 다음의 링크를 참조하자.

https://docs.unrealengine.com/content-examples-sample-project-for-unreal-engine/

Content Examples 프로젝트에는 여러 머티리얼에 대한 예제가 포함되어 있다. 여러 예제 중에서 머티리얼 예제에 대한 설명 문서는 다음의 링크를 참조하자.

https://docs.unrealengine.com/4.27/Resources/ContentExamples/Materials/

머티리얼에 대한 여러 예제 중에서 가장 기본적인 첫 번째 예제는 메인 머티리얼 노드의 각 입력핀에 대한 예제이다. 이 예제의 설명 문서는 다음의 링크를 참조하자.

https://docs.unrealengine.com/4.27/Resources/ContentExamples/MaterialNodes/

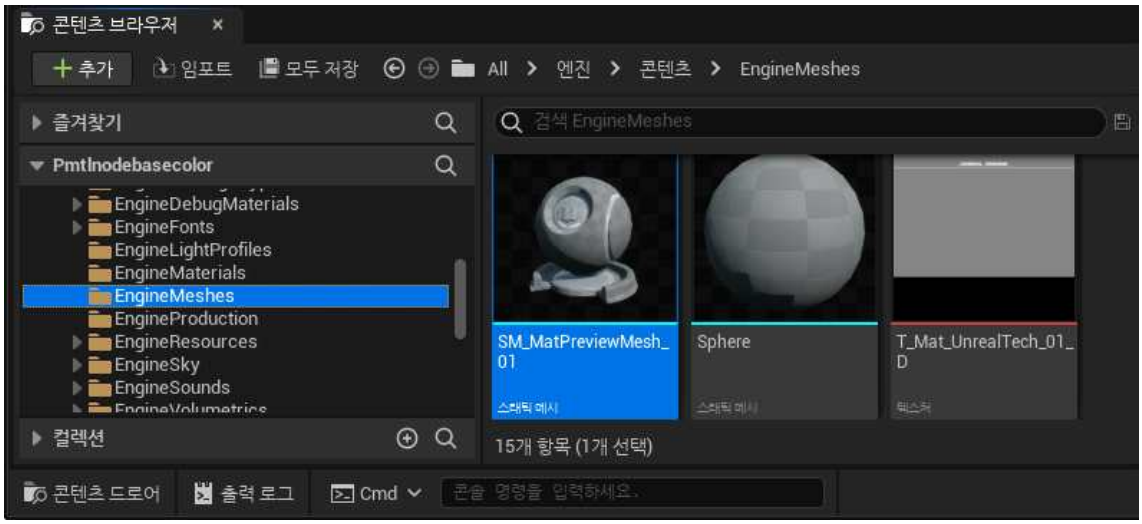
이 절은 메인 머티리얼 노드의 각 입력핀에 대한 예제를 참조하였다.

이제부터 예제를 통해서 학습해보자

---

1. 새 프로젝트 **Pmtlnodebasecolor**를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Pmtlnodebasecolor로 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. 콘텐츠 브라우저에서 왼쪽 탭에서 **엔진** 폴더를 선택하고 오른쪽 탭의 검색상자에서 **SM_MatPreviewMesh_01**를 검색하자. 스태틱 메시가 검색될 것이다. 또다른 방법으로, 왼쪽 탭에서 **엔진** » **콘텐츠** 아래에 있는 **EngineMeshes** 폴드로 이동하자. 오른쪽 탭에서 **SM_MatPreviewMesh_01**를 찾을 수 있을 것이다.



3. 찾은 **SM_MatPreviewMesh_01** 애셋을 더블클릭하자. 스태틱 메시 에디터가 열릴 것이다.

우리는 앞으로 이 메시를 머티리얼을 테스트하기 위한 예제로 사용할 것이다.

오른쪽 디테일 탭을 보면 머티리얼 슬롯이 2개가 있다. 각 머티리얼의 메시에서의 해당 영역을 살펴보자. **엘리먼트 0**과 **엘리먼트 1**의 아래에 있는 **하이라이트** 체크박스와 **단독 표시** 체크박스를 클릭해보자. **하이라이트**는 해당 머티리얼이 적용되는 메시의 외곽선을 표시하고, **단독 표시**는 해당 머티리얼이 적용되는 메시만 화면에 보이게 한다.

각 엘리먼트에는 디폴트 머티리얼이 지정되어 있다. **엘리먼트 0**에는 **WorldGridMaterial** 머티리얼이 지정되어 있고 **엘리먼트 1**에는 **M_Mat_UnrealTech** 머티리얼이 지정되어 있다. 우리는 앞으로 우리의 커스텀 머티리얼을 제작하고 이를 **엘리먼트 0**에 지정해서 테스트해볼 것이다. **엘리먼트 1**은 그대로 둘 것이므로 우리는 앞으로 **엘리먼트 0**에 해당하는 메시 영역에만 관심을 가지면 된다.

스태틱 메시 에디터를 닫자. 엔진의 애셋이므로 수정 내용이 있어도 저장하지 말자.

<참고> **SM_MatPreviewMesh_01** 스태틱 메시, **WorldGridMaterial** 머티리얼, **M_Mat_UnrealTech** 머티리얼은 모두 엔진 콘텐츠 폴더에 있는 애셋들이다. 따라서 이들 원본 애셋을 직접 수정하지 않도록 해야 한다. 만약 수정하려면 자신의 프로젝트 폴더에 사본을 복사한 후에 복사된 사본을 수정하자.

4. 이제 메시를 배치하자.

메시를 배치하기 전에 먼저, **PlayerStart** 액터를 선택하고 위치를 (-450,0,92)로 수정하자. 원래의 위치인 원점 위치보다 뒤로 물러나서 시작하므로 시야가 더 넓어질 것이다.

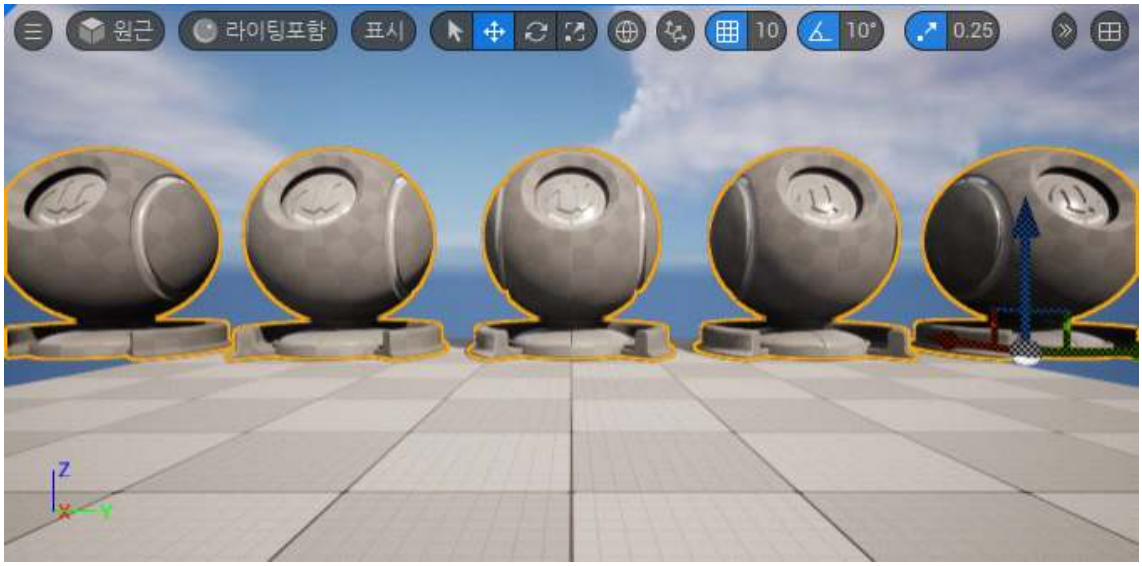
그리고, 뷰포트를 **상단** 뷰로 바꾸자.

그다음, 콘텐츠 브라우저에서 스태틱 메시 **SM_MatPreviewMesh_01**를 레벨에 배치하자.

배치된 액터의 이름을 **MeshLeftmost**로 수정하자. 위치는 (300,-600,0)으로 수정하고 회전은 (0,0,90)으로 수정하자.

배치된 액터를 Y축 방향으로 **Alt+드래그**로 4개의 복사본을 만들어 모두 5개의 액터가 되도록 하자. 배치된 복사본의 이름을 각각 **MeshMiddleleft**, **MeshMiddle**, **MeshMiddleright**, **MeshRightmost**로 수정하자. 위치는 (300,-300,0), (300,0,0), (300,300,0), (300,600,0)이 되도록 하자.

이제 뷰포트를 다시 **원근**으로 되돌려놓자.



5. 첫 번째 머티리얼을 생성하자.

먼저, **콘텐트 브라우저**에서 **콘텐츠** 폴더를 선택하자. **+추가**를 클릭하고 **머티리얼**을 선택하여 새 머티리얼을 생성하자. 이름을 **M_BaseColor_Color**로 수정하자.

그다음, 생성된 머티리얼 애셋을 더블클릭하여 머티리얼 에디터를 열자.

그다음, **3+좌클릭**하여 **Constant3Vector** 노드를 배치하자. 그리고, **디테일** 탭에서 디폴트 값을 (1,0,0)으로 수정하자.

그 다음, 노드를 메인 머티리얼 노드의 **베이스 컬러** 입력핀에 연결하자. 이제 빨간색 머티리얼이 완성되었다. 저장하자.

콘텐츠 브라우저에서 **M_BaseColor_Color**를 드래그하여 레벨의 **MeshLeftmost**에 드롭하자. 가장 왼쪽의 메시가 빨간색으로 바뀔 것이다.

이제, 배치된 **Constant3Vector** 노드를 파라미터 노드를 바꾸자. 노드를 우클릭하고 풀다운 메뉴에서 **파라미터로 변환**을 선택하자. 이름을 **BaseColor**로 수정하자. 이제 다음과 같은 모습이 될 것이다.

저장하자. 메시의 모양은 컬러가 바뀌지 않았으므로 그대로일 것이다. 그러나 파라미터 노드의 장점을 가지게 되었다.

6. 두 번째로, 바로 전에 생성된 머티리얼로부터 머티리얼 인스턴스를 생성하자.

콘텐트 브라우저에서 **M_BaseColor_Color** 위에서 우클릭하고 드롭다운 메뉴에서 **머티리얼 인스턴스 생**

성을 선택하자. 이름은 **M_BaseColor_Color_Inst**로 두자.
그다음, 생성된 머티리얼 애셋을 더클클릭하여 머티리얼 인스턴스 에디터를 열자.
그다음, 오른쪽 디테일 탭에서 **파라미터 그룹** 영역의 **BaseColor** 속성의 체크박스를 클릭하고 값을 (0,1,0,1)로 수정하자. 빨간색에서 녹색 머티리얼로 바뀔 것이다.
저장하자.
콘텐츠 브라우저에서 **M_BaseColor_Color_Inst**를 드래그하여 레벨의 **MeshMiddleleft**에 드롭하자. 중간 왼쪽의 메시가 녹색으로 바뀔 것이다.

7. 세 번째 머티리얼을 생성하자.
콘텐트 브라우저에서 **+추가**를 클릭하고 **머티리얼**을 선택하여 새 머티리얼을 생성하자. 이름을 **M_Base Color_Texture**로 수정하자.
그다음, 생성된 머티리얼 애셋을 더블클릭하여 머티리얼 에디터를 열자.
그다음, **T+좌클릭**하여 **TextureSample** 노드를 배치하자. 그리고, **디테일** 탭에서 **머티리얼 표현식 텍스처 베이스** 영역의 **Texture** 속성값에 **T_ground_Moss_D** 텍스처를 검색하여 지정하자.
그다음, 노드의 RGB 출력핀을 메인 머티리얼 노드의 **베이스 컬러** 입력핀에 연결하자. 이제 이끼 텍스처 머티리얼이 완성되었다. 저장하자.
콘텐츠 브라우저에서 **M_BaseColor_Texture**를 드래그하여 레벨의 **MeshMiddle**에 드롭하자. 중간의 메시가 이끼 모습으로 바뀔 것이다.
이제, 배치된 **TextureSample** 노드를 파라미터 노드를 바꾸자. 노드를 우클릭하고 드롭다운 메뉴에서 **파라미터로 변환**을 선택하자. 이름을 **BaseColor**로 수정하자.

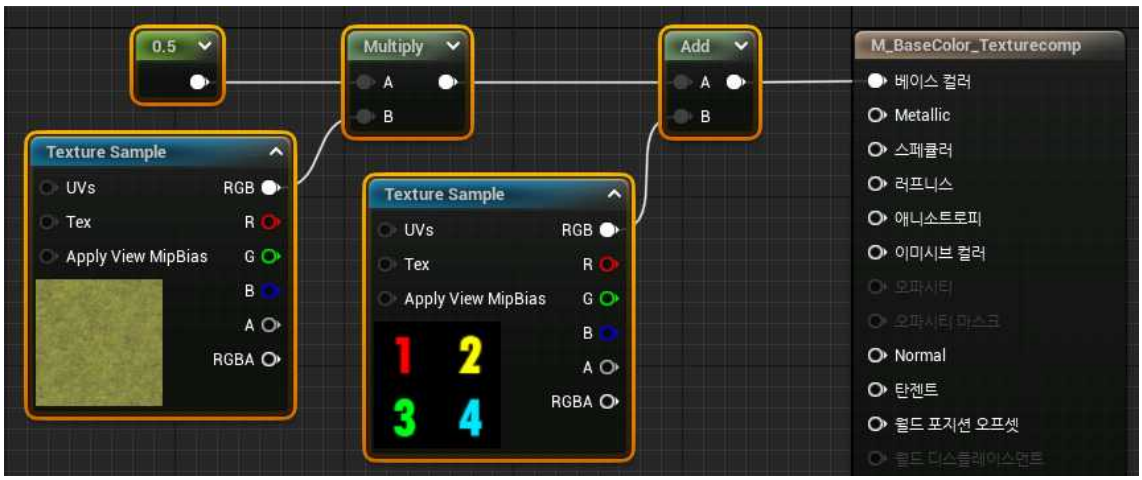
저장하자.

<참고> 어떤 노드의 도움말을 보고 싶을 때에는 노드를 선택하고 **Ctrl+Alt** 키를 누르면 된다. 누르고 있는 동안에만 도움말이 툴팁 형식으로 표시된다. 누르고 있는 상태에서 마우스로 툴팁 하단의 **원본 문서 확인** 링크를 클릭하면 도움말 문서 페이지를 보여준다.

8. 네 번째로, 바로 전에 생성한 머티리얼로부터 **머티리얼 인스턴스**를 생성하자.
콘텐트 브라우저에서 **M_BaseColor_Texture** 위에서 우클릭하고 드롭다운 메뉴에서 **머티리얼 인스턴스 생성**을 선택하자. 이름은 **M_BaseColor_Texture_Inst**로 두자.
그다음, 생성된 머티리얼 애셋을 더클클릭하여 머티리얼 인스턴스 에디터를 열자.
그다음, 오른쪽 디테일 탭에서 **파라미터 그룹** 영역의 **BaseColor** 속성의 체크박스를 클릭하고 값을 엔진의 텍스처인 **DefaultDiffuse**로 수정하자. 이끼 텍스처에서 바닥타일 텍스처 머티리얼로 바뀔 것이

다.
저장하자.
콘텐츠 브라우저에서 **M_BaseColor_Texture_Inst**를 드래그하여 레벨의 **MeshMiddleright**에 드롭하자. 중간 오른쪽의 메시가 바닥타일 모습으로 바뀔 것이다.

9. 다섯 번째 머티리얼을 생성하자.
이번에는 이끼 텍스처와 더불어 까만 배경에 컬러 숫자가 있는 텍스처 두 개를 함께 사용해보자. 이끼 텍스처가 약간 어두워지도록 각 픽셀에 0.5를 곱하자. 그리고 그 결과에 컬러 숫자 텍스처를 그대로 더하자. 숫자 부분이 약간 어두워진 이끼 텍스처와 합쳐져서 나타나도록 해보자.
먼저, **콘텐트 브라우저**에서 **+추가**를 클릭하고 **머티리얼**을 선택하여 새 머티리얼을 생성하자. 이름을 **M_BaseColor_Texturecomp**로 수정하자.
그다음, 생성된 머티리얼 애셋을 더블클릭하여 머티리얼 에디터를 열자.
그다음, 격자탭에서 **1+좌클릭**하여 **Constant** 노드를 배치하자. 디테일 탭에서 **값**에 0.5를 지정하자.
그다음, **T+좌클릭**하여 **TextureSample** 노드를 배치하자. 그리고 **Texture** 속성값에 **T_ground_Moss_D** 텍스처를 검색하여 지정하자.
그다음, **M+좌클릭**하여 **Multiply** 노드를 배치하자. 그리고 **Constant** 노드의 출력핀과 **TextureSample** 노드의 **RGB** 출력핀을 **Multiply** 노드의 입력핀에 연결하자.
그다음, 또하나의 **TextureSample** 노드를 배치하자. 그리고 **Texture** 속성값에 **flipbook** 텍스터를 검색하여 지정하자.
그다음, **A+좌클릭**하여 **Add** 노드를 배치하자. 그리고 **Multiply** 노드의 출력핀과 두 번째 **TextureSample** 노드의 **RGB** 출력핀을 **Add** 노드의 입력핀에 연결하자.
그다음, **Add** 노드의 출력핀을 메인 머티리얼 노드의 **베이스 컬러** 입력핀에 연결하자. 이제 이끼 텍스처에 숫자가 새겨진 머티리얼이 완성되었다. 저장하자.

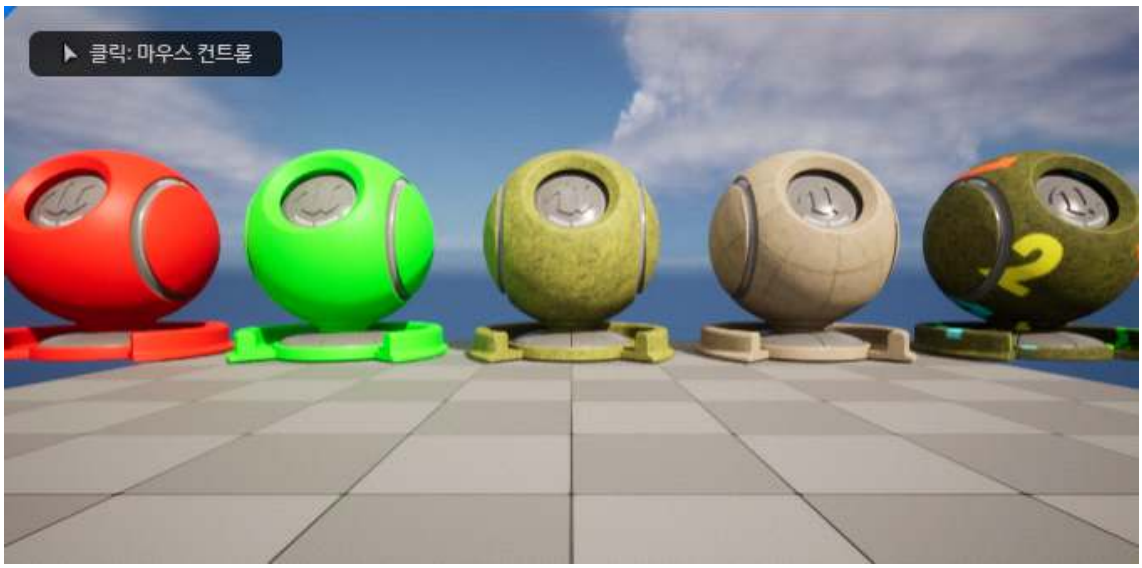


콘텐츠 브라우저에서 **M_BaseColor_Texturecomp**를 드래그하여 레벨의 **MeshRightmost**에 드롭하자. 가장 오른쪽의 메시가 이끼 텍스처에 숫자가 새겨진 모습으로 바뀔 것이다.

<참고> 머티리얼 표현식 노드에서의 산술 노드인 **Add** 노드나 **Multiply** 노드에 대해서 살펴보자. 이들 노드는 입력핀으로 단일 스칼라 값이나 벡터 값이나 텍스트 값 등의 다양한 유형을 입력받을 수 있다.
만약 산술 노드의 두 입력핀에 대한 유형이 동일한 경우에는 각 요소별 연산 결과를 출력한다. 예를 들어 두 벡터의 곱셈은 두 벡터의 각 요소별 곱을 구해 출력 벡터를 계산하는 것을 의미한다. 두 텍스처의 곱셈도 두 텍스처의 각 픽셀별 곱을 구해 출력 텍스처를 계산한다.

만약 산술 노드의 두 입력핀에 대한 유형이 다른 경우에는 단일 값의 입력값을 반복 적용하여 연산 결과를 출력한다. 예를 들어 스칼라와 벡터의 곱셈은 스칼라를 벡터의 각 요소에 곱하여 출력 벡터를 계산한다. 벡터와 텍스처의 곱셈도 벡터를 텍스처의 각 픽셀에 곱해 출력 텍스처를 계산한다.

10. 전체적으로 다음과 같은 모습이 될 것이다.



이 절에서는 **베이스 컬러** 입력핀에 대해서 학습하였다.

□