04_ 레벨 디자인 시작하기

<제목 차례>

인천대학교 컴퓨터공학부 박종승

1. 개요

이 장에서는 레벨 디자인을 시작하는 방법을 학습하자.

게임 개발의 일반적인 절차에 대해서 알아보자. 게임 개발 단계는 크게 세 단계(phase)로 나누어 설명할 수 있다. 첫 번째 단계는 만들고자 하는 게임을 간단히 만들어서 게임의 가치가 있는 지를 확인하는 단계인 '프로토타이핑'(prototyping) 단계이다. 두 번째 단계는 만들고자 하는 게임을 제품 수준으로 완성시키기 위한 예행 연습을 수행하는 단계인 '프리프로덕션'(pre-production) 단계이다. 세 번째 단계는 남은 부분들을 모두 만들어서 정식 제작을 진행하는 '프로덕션'(production) 단계이다.
게임 개발 단계 중에서 프로토타입 단계에서는 게임의 가치와 재미를 판단하기 위해 짧은 기간에 게임 자체의 본질적인 요소를 구현하여 게임의 아이디어를 검증한다. 이 단계는 게임의 아이디어인 가설을 효율적으로 검증하기 위한 단계로 게임의 본질적인 재미 요소에 집중해야 하고 게임의 디자인 측면에 집중할 필요는 없다.
게임의 재미 요소를 증명하는 단계인 프로토타이핑 단계를 거친 후에는 프리프로덕션 단계를 진행한다. 프리프로덕션 단계는 게임 제작의 전체 과정의 진행에 대한 작업 공정(workflow)를 증명하는 단계이다. 누가 언제 어디서 무엇을 어떻게 만들 것인지의 작업 공정을 정하고 게임의 한 스테이지를 실제로 만들어본다. 이렇게 해봄으로써 주어진 예산이나 기간을 고려해서 전체 게임을 만들 수 있는지를 검증한다. 이 단계에서는 게임의 주요 기능이 모두 탑재된 스테이지 하나를 최종 제품과 동일한 품질로 만든다. 이 단계를 거치면 막대한 손해가 발행하는 상황을 피할 수 있게 된다.
프리프로덕션 단계 이후에는 동일한 작업 공정으로 게임의 모든 스테이지를 제작하는 프로덕션 단계를 진행한다. 이 단계에서는 인력을 추가하고 예산을 투입하는 등 규모를 확대하여 게임을 완성한다.

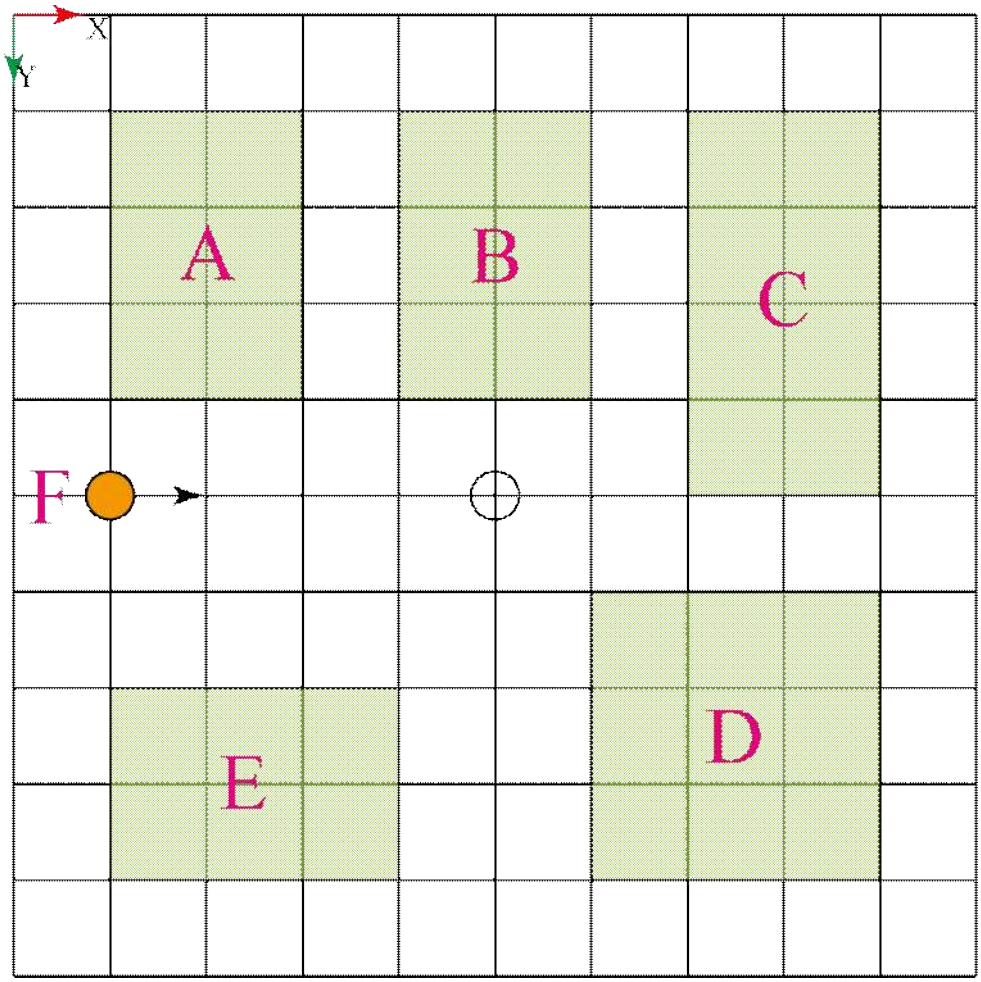
프로토타이핑 단계에서는 게임플레이를 테스트할 수 있는 플레이 가능한 스테이지를 빠르게 만드는 것이 중요하다. 프로토타입 제작 과정에서 레벨 디자이너는 텍스처가 적용되지 않은 상자 등과 같은 간단한 메시를 사용해서 스테이지를 만들고 게임 플레이를 테스트한다. 게임의 본질에 집중하는 것이 좋으므로 메시의 시각적인 요소 등은 모두 생략한다. 이러한 외양에 신경쓰지 않고 게임의 본질을 확인하기 위한 게임플레이만 고려하여 빠르게 맵을 제작하는 과정을 '그레이박싱'(grayboxing)이라고 한다. 그리고, 이렇게 만든 레벨 데이터를 '그레이박스'(graybox)라고 한다.
프로토타이핑을 거친 후에는 그레이박스의 단순한 임시 메시를 고품질의 메시로 교체해야 한다. 그레이박스에서 스테이지에 배치한 임시 메시를 정식 애셋으로 바꾸어 배치하는 과정을 '메싱'(meshing)이라고 한다. 메싱 공정에서는 메시 교체로 인하여 게임플레이에서 충돌 등이 달라지는 문제가 생기지 않도록 주의해야 한다.

이제, 레벨 디자인을 시작하는 방법을 학습하자.

2. 레벨 에디터에서 처음으로 레벨 만들기

레벨 에디터에서 처음으로 레벨을 만들어보자.

간단한 레벨을 제작해보자.
먼저, 만들 레벨의 설계도를 그려보자. 우리는 아래의 그림과 같은 레벨을 만들어보자.



설계도를 작성할 때에 좌표계가 뷰포트에서의 직교 뷰와 일치하도록 작성하는 것이 좋다. 위의 그림은 상단 뷰와 일치하도록 설정하였다.
위의 설계도에서 격자칸 하나를 100x100으로 가정하자.
그리고 각 상자의 높이는 100으로 하자.

이제부터 실제로 실습해보자.

---

1. 새 프로젝트 **Peditcube**를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Peditcube로 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. 현재의 디폴트 레벨의 모습을 살펴보자. 레벨에는 총 7개의 액터가 배치되어 있으며 그 중에서 **Floor**라는 스태틱 메시가 하나 배치되어 있다. 그 외의 액터는 모두 조명과 관련된 액터이다.
Floor 액터는 크기가 1000x1000x50인 직육면체 스태틱 메시이다. 피벗의 위치는 직육면체 윗면의 중심에 있다.
Floor가 선택된 상태에서 **디테일** 탭을 살펴보면 **트랜스폼** 속성에서 **위치**가 (0,0,-0.5)로 되어있다. 액터의 피벗의 위치가 월드에서 (0,0,20)에 있으므로 **Floor**는 월드의 XY평면에서 0.5만큼 아래로 내려가 있다. 우리는 더욱 단순하게 레벨을 구성하기 위해서 **Floor**의 **위치** 속성값을 (0,0,0)으로 수정하자. 즉 위치의 Z값을 -0.5에서 0으로 수정하자.
또한 스케일이 (8,8,8)로 8배 확대되어있는데 우리는 이것을 (1,1,1)로 수정하자.

이제 **Floor**의 윗면이 월드에서 XY평면과 일치하도록 하였다.
Floor가 XY평면의 원점에 위치하고 크기가 1000x1000이므로 설계도의 격자와 일치한다.

3. 이제, 상자 메시 하나를 레벨에 배치하자.

콘텐츠 브라우저에서 **엔진** » **콘텐츠** » **BasicShapes** 폴더에서 **Cube** 스태틱 메시를 찾아보자.
Cube 스태틱 메시는 정육면체 상자 모양으로 크기는 100x100x100이다. 그리고 **Cube**의 피벗은 정육면체의 중심에 있다.
콘텐츠 브라우저에 있는 **Cube** 아이콘을 드래그하여 뷰포트에 드롭하자. **Cube** 스태틱 메시가 레벨에 배치된다.

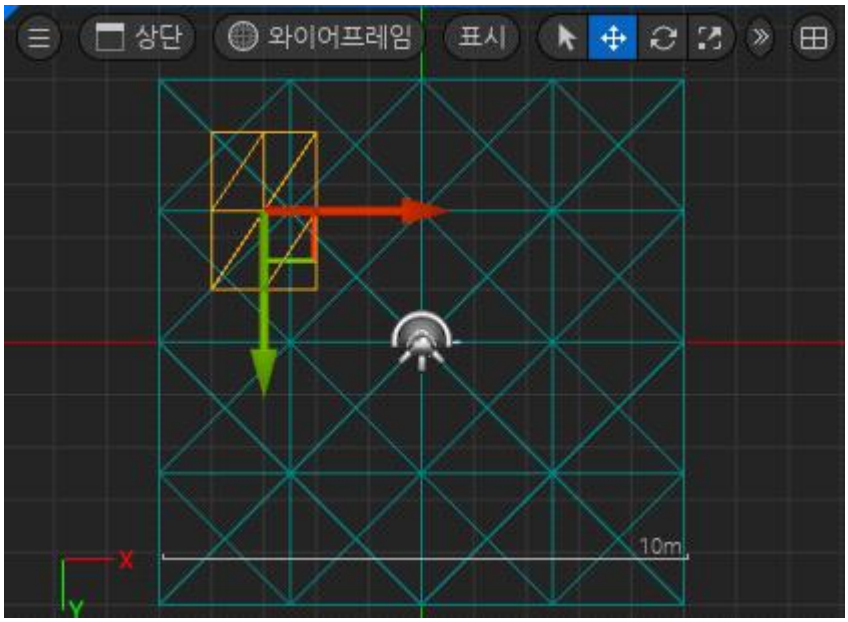
4. 애셋을 뷰포트에 드롭할 때에 **Cube**의 피벗이 **Floor**의 상단면에 일치되도록 배치된다. 즉 **Cube**의 **트랜스폼** 속성에서 **위치**의 Z값이 0이 되도록 배치된다. 따라서 큐브의 절반이 **Floor** 윗면 아래에 잠길 것이다.
Cube 전체가 **Floor** 윗면 위에 놓이도록 하려면 **Cube**의 절반 크기만큼 위로 올려야 한다. 이를 위해서 **Cube**의 Z값이 50이 되도록 하자. **Cube**의 **트랜스폼**의 **위치**에서 Z값을 50으로 입력하자. X,Y값은 무엇이든 그대로 두자.

5. 액터를 바닥에 붙이는 또다른 방법을 연습해보자.
뷰포트에 추가된 **Cube** 액터의 이동 기즈모의 Z축을 드래그하여 **Cube**가 공중에 약간 뜨게 배치하자. 그다음, **End** 키를 누르자. **Cube** 액터가 바닥에 달라붙을 것이다.

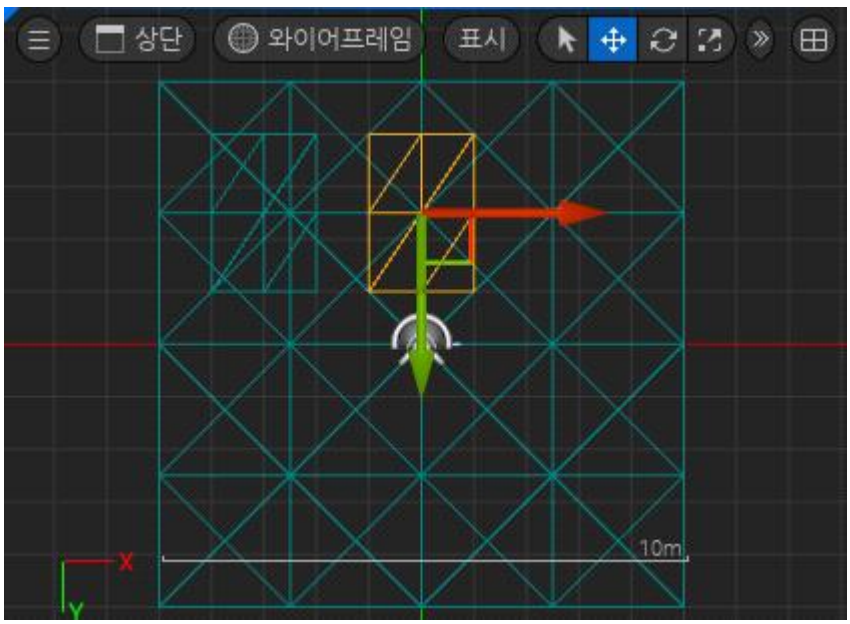
6. 설계도가 **상단** 뷰와 일치되므로 뷰포트를 **상단** 뷰로 전환하자. 뷰포트의 왼쪽 위에 보이는 **원근** 아이콘을 클릭하고 **상단**을 클릭하면 된다.
뷰포트에서 마우스를 스크롤해서 바닥 전체가 보이도록 축소하자.
이제부터, 레벨의 설계도가 되도록 하나씩 배치하자.
가장 먼저, 현재 배치된 **Cube**가 설계도의 상자 A라고 하자.
레벨에 많은 액터가 추가될 것이므로 가급적이면 액터에 구분이 쉬운 이름을 지정해두는 것이 좋다. **아웃라이너**에서 현재 배치된 **Cube**를 선택하고 **F2** 키를 누르면 이름을 수정할 수 있다. 우리는 CubeA로

수정하자.

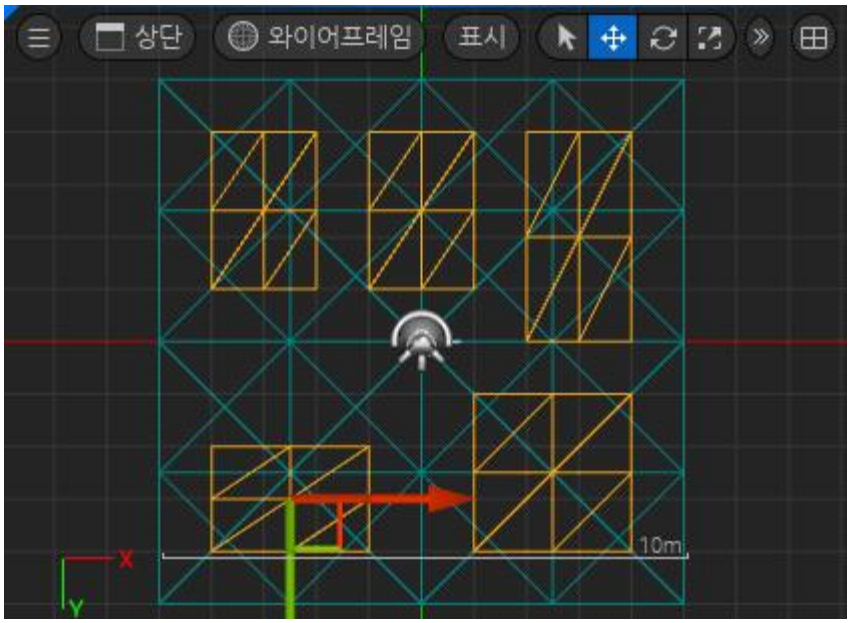
7. 원래 **Cube**의 크기가 설계도의 한 칸에 해당한다. 따라서 상자 A의 크기로 바꾸기 위해 **디테일** 탭에서 **트랜스폼**의 **스케일** 값에 (2,3,1)을 입력하자.
그다음, **Ctrl**+**좌클릭**+**드래그**로 CubeA를 설계도의 상자 A의 위치로 옮기자. **위치**에 (-300,-250,50)을 입력해도 된다.
다음과 같이 보일 것이다.



8. 다음으로, 설계도의 상자 B를 만들자.
뷰포트에서 CubeA의 이동 기즈모의 X축을 **Alt**+**좌클릭**+**드래그**로 오른쪽으로 한 칸 드래그한다. 위치에 (0,-250,0)을 입력해도 된다. 그다음, 액터의 이름을 CubeB로 수정하자. 이제 다음과 같이 보일 것이다.



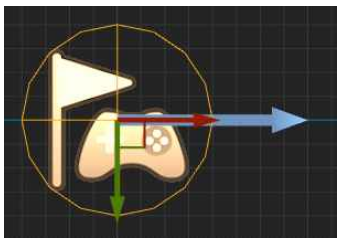
9. 다음으로, 설계도의 상자 C를 만들자.
뷰포트에서 CubeB의 이동 기즈모의 X축을 **Alt+좌클릭+드래그**로 오른쪽으로 한 칸 드래그한다. 위치에 (300,-200,50)을 입력해도 된다. 그다음, 액터의 이름을 CubeC로 수정하자. 그다음, **디테일** 탭에서 **트랜스폼**의 **스케일** 값에서 Y축의 값을 3에서 4로 수정하자. 수정 후에는 스케일이 (2,4,1)이 된다. 그다음, 다시 이동 기즈모의 Y축을 드래그하여 격자에 맞도록 위치를 맞춘다.
상자 D와 E에 대해서도 동일한 방법으로 진행하여 CubeD와 CubeE를 만들자.
CubeD는 위치가 (250,250,50)이고 스케일이 (3,3,1)이도록 하면 된다.
CubeE는 위치가 (-250,300,50)이고 스케일이 (3,2,1)이도록 하면 된다.
이제 5개의 상자가 다음과 같이 배치되었을 것이다.



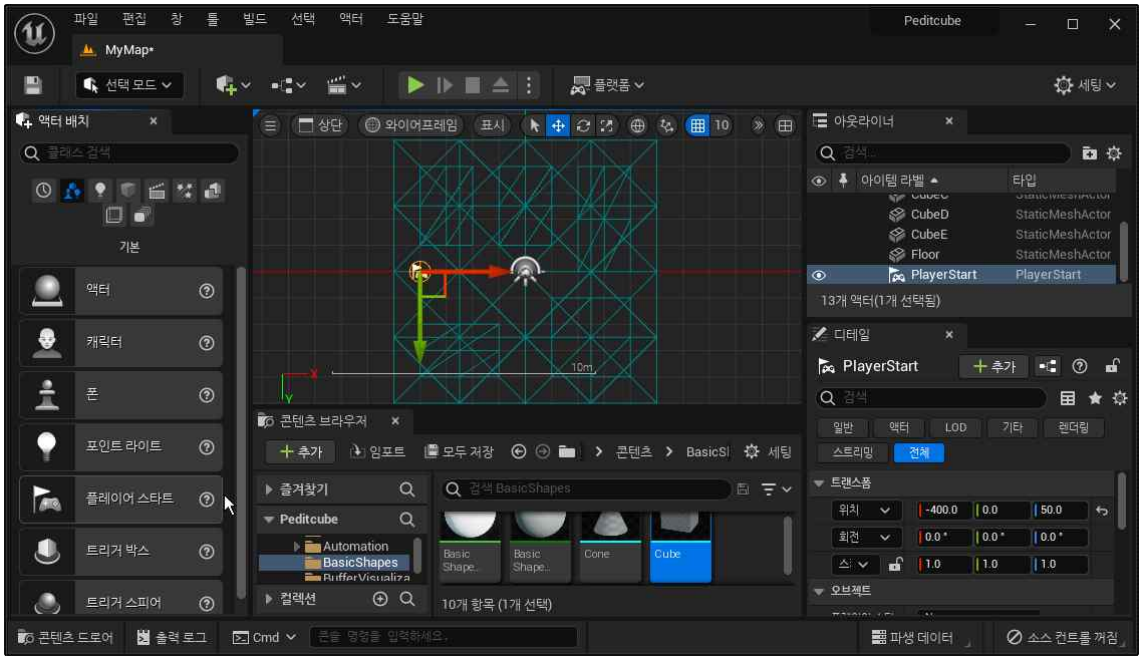
10. 이제 툴바의 녹색 플레이 버튼을 클릭해보자. 뷰포트 화면에 아래와 같이 보일 것이다. 이는 플레이어 시작 위치가 바닥 중간의 위쪽이어서 상자의 윗부분 일부만 보이게 된다.

실행을 시작하면 엔진이 자동으로 **PlayerStartPIE**라는 액터를 생성해준다. 이 액터는 플레이어의 시작 위치가 에디터의 뷰포트에서의 현재 위치로 되도록 해준다. 뷰포트에서의 시점과 동일한 위치에서 플레이해볼 수 있으므로 개발자에게 편리한 기능이다.

11. 개발 단계가 아니라면 플레이어 시작 위치를 고정해주어야 할 것이다.
지금부터 플레이어 시작 위치를 지정해주는 방법을 알아보자.
플레이어 시작 위치를 지정하는 기능의 액터는 **PlayerStart** 액터이다. 이 액터를 미리 레벨에 배치해두면 **PlayerStartPIE**는 생성되지 않는다.
우리는 **PlayerStart** 액터를 레벨에 추가해보자. 먼저 **창** » **액터 배치** 메뉴를 선택하자.
왼쪽에 **액터 배치** 탭이 생길 것이다. 여기서 **기본** 탭에 **플레이어 스타트** 액터가 있다. 이 **PlayerStart** 액터는 게임이 플레이될 때에 플레이어 캐릭터가 등장하는 위치와 방향을 정의하는 특별한 액터이다. 이것을 드래그하여 레벨에 배치하자.
이제 배치된 **PlayerStart** 액터의 **트랜스폼**의 **위치** 속성과 **회전** 속성을 바꾸면 플레이어 시작 위치를 지정할 수 있다. 예를 들어 **위치** 속성값을 (0,0,112)로 하면 XY평면의 원점에 있고 높이는 지면에서 112cm 위에 있게 된다. 또한 **회전** 속성값을 (0,0,0)으로 한다면 전방 방향 기준에서 회전 없음에 해당한다. 전방 방향이 X축이므로 회전 없음은 X축과 일치하는 방향임을 의미한다. 뷰포트에서 **PlayerStart** 액터를 선택하고 화면을 확대해보면 아래 그림과 같이 기즈모와 더불어 하늘색 화살표가 표시되어 있다. 이 방향이 **PlayerStart** 액터가 바라보는 회전 방향에 해당한다.

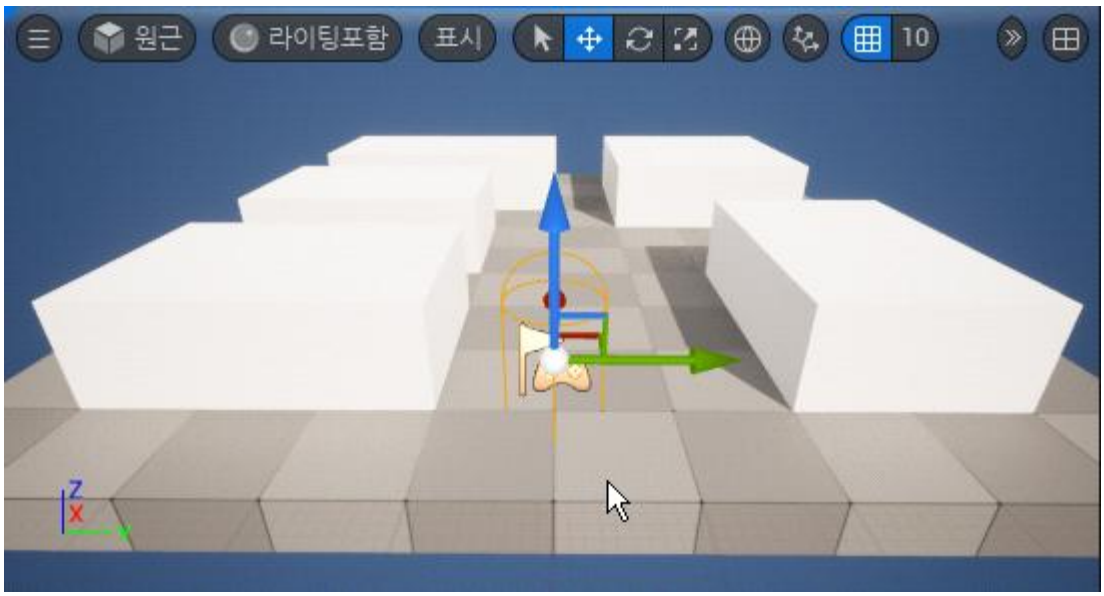


12. 우리는 **PlayerStart** 액터를 설계도의 E의 위치에 배치하자. 디테일 탭에서 **위치** 속성값을 (-400,0,50)으로 수정하자. 즉 X축으로 맨 뒤로 이동하였고 높이는 약간 낮추었다. **회전** 속성은 전방 방향인 X축 방향을 향하도록 그대로 두자.

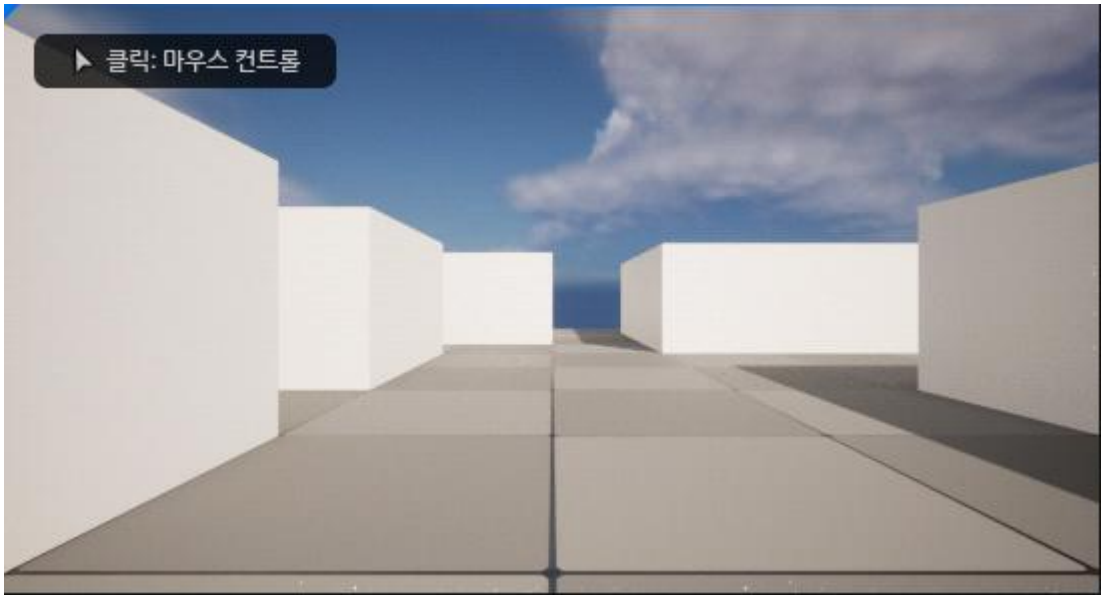


저장하자.

13. 이제 원근 뷰로 바꾸자. 아래와 같이 표시될 것이다.

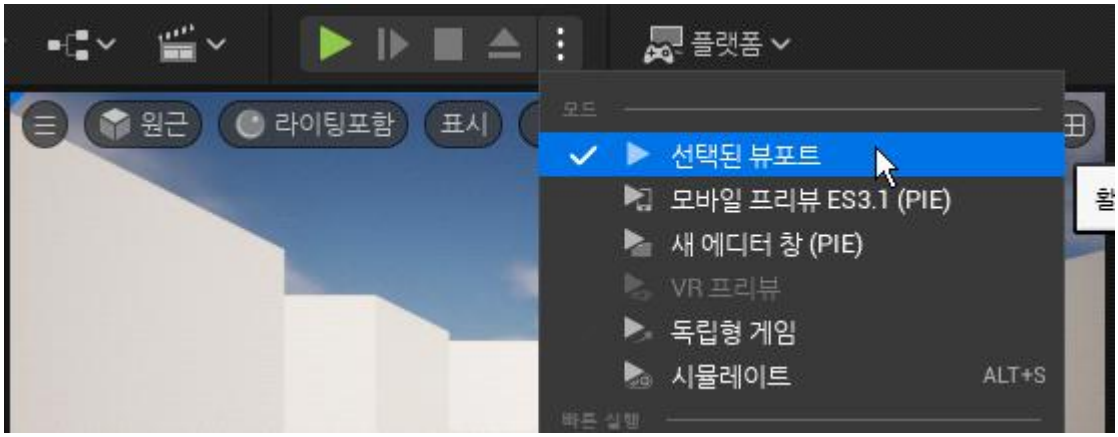


14. 게임을 플레이해보자. 게임을 플레이해보고 실제 시작 위치와 방향이 **PlayerStart** 액터의 위치와 방향과 일치하는지 확인해보자.
게임을 플레이하려면 툴바의 **플레이** 버튼을 클릭하면 된다.
다음과 같이 표시될 것이다.



15. 이제부터, 플레이와 관련된 내용에 대해서 학습해보자.
플레이한 후에 마우스를 움직여보자. 여전히 에디터에서 에디터의 기능을 조작할 수 있음을 알 수 있다. 이와 관련하여, 게임의 플레이 시의 입력 제어권에 대해서 알아보자.
게임이 플레이되더라도 입력 컨트롤은 여전히 에디터에 남아 있다. 즉 게임이 시작하더라도 에디터의 메뉴 등을 조작할 수 있는 상태로 남아있게 된다. 게임이 플레이되면 뷰포트의 화면 왼쪽 상단에 **클릭: 마우스 컨트롤**이라는 메시지가 잠시 표시된다. 입력 컨트롤을 게임 플레이 화면으로 넘기려면 마우스로 뷰포트를 클릭하라는 의미이다. 뷰포트를 클릭하면 그때부터 입력 컨트롤이 게임 화면으로 전달된다.
뷰포트를 클릭하면 뷰포트 화면 왼쪽 상단에 **마우스 커서는 Shift+F1**이라는 메시지가 잠시 표시된다. 이때부터는 마우스 조작이 게임플레이로만 전달되고 에디터의 메뉴 등을 조작할 수 없게 된다. 만약 게임을 종료하지 않고 실행 도중에 마우스 제어권을 다시 에디터로 넘기려면 **Shift+F1** 키를 누르면 된다. 이렇게 입력 제어권을 게임 플레이 중에도 수시로 에디터와 게임플레이 사이를 오가게 할 수 있다.

16. 툴바에 있는 **플레이** 버튼에 대해서 더 알아보자. 플레이 버튼을 클릭하면 기본적으로 **선택된 뷰포트** 모드로 플레이한다. 게임 플레이 후에, 게임을 중지하려면 **ESC** 키를 누르거나 또는 툴바의 중지 버튼을 클릭하면 된다.
선택된 뷰포트 모드가 아닌 다른 모드로 플레이하려면 오른쪽 수직점 모양 아이콘을 클릭하면 된다. 툴바의 **플레이** 버튼은 항상 **선택된 뷰포트** 모드로 플레이하는 것이 아니라 최근 플레이한 모드로 플레이한다.



각 모드에 대해서 하나씩 알아보자.

먼저, **선택된 뷰포트** 모드는 뷰포트에서 게임 화면이 플레이된다. 여러 뷰포트가 있는 경우에는 현재 활성인 뷰포트에서 플레이된다. 이 모드는 디폴트로 사용된다. 빠른 시간에 간편하게 플레이해볼 수 있는 장점이 있다. 대부분의 경우에는 이 모드로 플레이한다. 플레이한 후에 뷰포트를 클릭해야 입력 제어권이 플레이 화면으로 넘어간다.

다음으로, **모바일 프리뷰 ES3.1 (PIE)** 모드에 대해서 알아보자. 이 모드는 모바일 장치에서의 플레이를 시뮬레이션한다. 모바일 장치는 PC와는 다른 OpenGL ES 그래픽 하드웨어 사양을 사용하므로 시뮬레이션을 위해서는 많은 준비를 해야 하므로 창이 뜨는 시간이 오래 걸린다. 별도의 창으로 플레이된다. 마우스는 플레이창과 에디터를 오갈 수 있다. 플레이를 종료하려면 플레이 창의 오른쪽 상단 모서리의 닫기 아이콘을 클릭하면 된다.

다음으로, **새 에디터 창 (PIE)** 모드에 대해서 알아보자. 새 창에서 플레이된다. **선택된 뷰포트** 모드는 플레이 화면의 크기와 비율이 현재 뷰포트의 모양에 따라서 정해지는 단점이 있다. 그러나 **새 에디터 창 (PIE)** 모드에서는 우리가 설정한 화면 해상도와 비율을 그대로 적용하여 플레이된다. 플레이하면 새 창이 뜬다. 이 모드에서도 **선택된 뷰포트** 모드에서와 같이 마우스를 클릭하면 새 플레이 창으로 입력 제어권이 넘어가고 **Shift+F1** 키를 사용하여 입력 제어권이 에디터와 게임플레이 창 사이를 오가도록 있다. 또한, 게임플레이 창이 독립된 창이므로 윈도우의 **Alt+Tab** 키를 사용하여 다른 창으로 오갈 수 있다. 플레이를 종료하려면 **선택된 뷰포트** 모드에서와 같이 **ESC** 키를 누르면 된다.

다음으로, **독립형 게임** 모드는 현재 실행중인 에디터에서 게임을 플레이하는 것이 아니라 완전히 독립된 별도의 프로그램을 실행해서 게임을 플레이하는 모드이다. 이 모드는 게임 실행에 시간이 많이 걸려서 자주 사용하는 모드는 아니다. 게임 배포전에 정식 테스트 단계에서 사용된다. 종료하려면 **Alt+Tab** 키를 사용하여 게임플레이 화면으로부터 마우스 컨트롤을 회수한 후에 플레이 창의 오른쪽 상단 모서리의 닫기 아이콘을 클릭하면 된다. 에디터가 아니므로 **ESC** 키는 동작하지 않는다.

다음으로, **시뮬레이트** 모드는 게임을 플레이하는 모드가 아니라 레벨을 관찰하기 위한 모드이다. 이 모드로 플레이하면 게임에서의 화면은 그대로 나타나지만 플레이어와 관련된 기능은 수행되지 않는다. 대신 마우스와 키보드로 레벨을 자유롭게 돌아다니면서 레벨을 관찰할 수 있다. 따라서 파티클 이펙트나 물리 상태들을 원하는 위치에서 편리하게 확인할 수 있는 장점이 있다. 모드를 종료하려면 **ESC** 키를 누르면 된다.

<참고> 만약 **독립형 게임** 모드 등에서 창을 닫는데 어려움이 있다면 **Tab** 키 바로 위에 배치되어 있는 특수문자 ``` 키를 누른 후에 `quit`을 입력하고 엔터키를 누르면 된다. 이것은 콘솔 명령어 기능을 사용하여 플레이를 종료하는 방식이다.

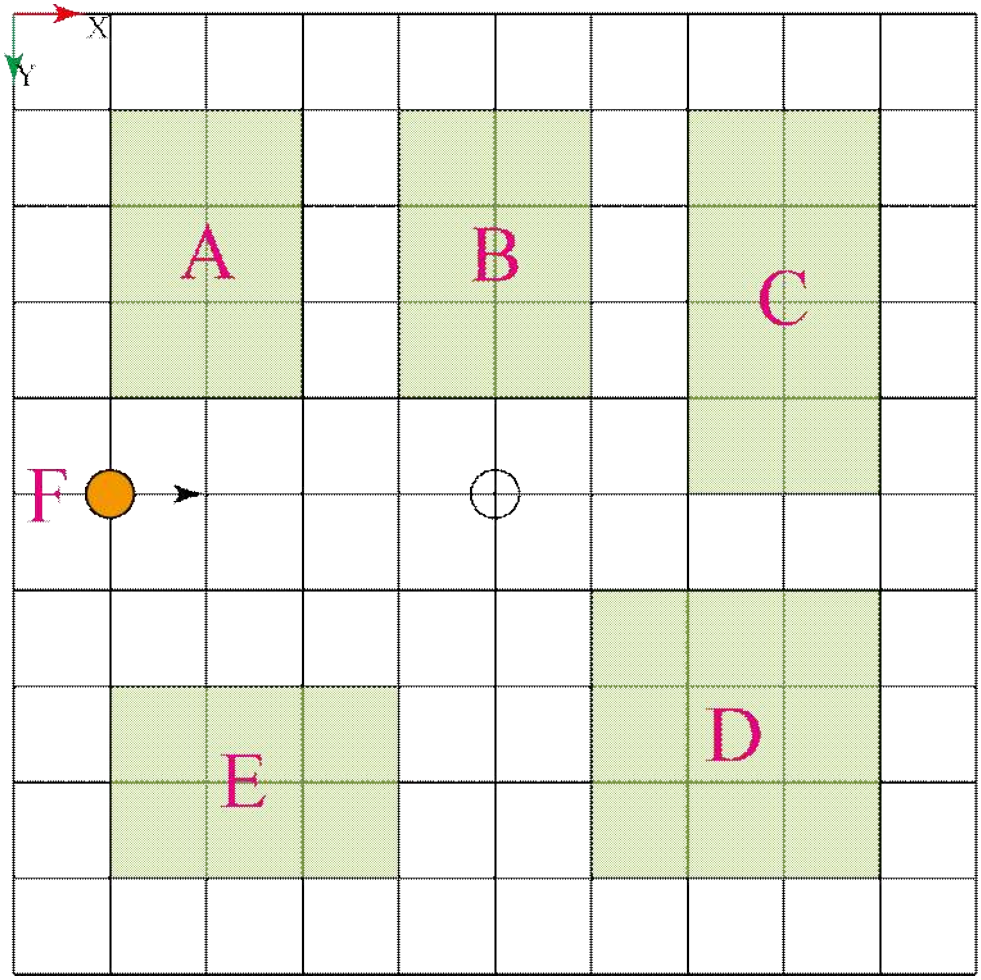
17. 플레이 시의 또다른 유용한 기능에 대해사 알아보자.
플레이 도중에 **F8** 키를 누르면 플레이어가 자신의 폰으로부터 탈출할 수 있다. 탈출한 이후에는 카메라를 자유롭게 이동하면서 플레이 도중에 주위를 관찰할 수 있다. 탈출한 이후에 카메라 이동을 위한 키 조작은 캐릭터 폰의 키조작 방식이 아니라 레벨 에디터에서의 키조작 방식으로 입력해야 한다. **F8** 키를 다시 누르면 폰으로 다시 돌아간다.

지금까지 레벨 에디터에서 레벨 만들기를 학습하였다.

3. 브러시를 사용하여 레벨 만들기

브러시를 사용하여 레벨을 만들어보자.
먼저 **브러시**에 대해서 알아보자.
레벨을 만드는 일반적인 방법은 이전 예제에서와 같이 스태틱 메시를 사용하는 방법이다. 한편, 스태틱 메시를 사용하려면 마야와 같은 외부의 디자인 툴에서 메시를 제작하고 이를 언리얼 에디터로 임포트해야 한다. 메시 제작 절차는 시간이 걸리는 번거로운 작업이다. 또한 외부 디자인 툴을 사용하기 위해서는 툴을 익히기 위한 충분한 학습 시간이 필요하다. 이런 불편을 위해서 언리얼에서는 **브러시**라고 하는 자체적인 레벨 디자인 도구를 제공하고 있다.
브러시는 언리얼에서 레벨을 미리 만들어보는 빠른 프로토타이핑을 위해서 제공하는 도구이다. 따로 학습하지 않아도 누구나 레벨을 쉽게 만들어볼 수 있다.
게임 개발의 프로토타이핑 단계에서는 정식 메시 대신 외양을 무시한 간단한 메시인 그레이박스로 레벨을 제작한다. **브러시**는 이러한 그레이박싱의 용도로 사용될 수 있는 유용한 도구이다.
이 절에서는 **브러시**를 사용하여 레벨을 만들어본다.

브러시로 만들 레벨의 설계도를 그려보자. 우리는 아래의 그림과 같은 레벨을 만들어보자.



위의 설계도 그림은 상단 뷰와 일치하도록 설정하였고 설계도에서 격자칸 하나를 100x100으로 가정하였다. 그리고 각 상자의 높이는 이전 예제에서의 높이의 두 배인 200으로 하자.

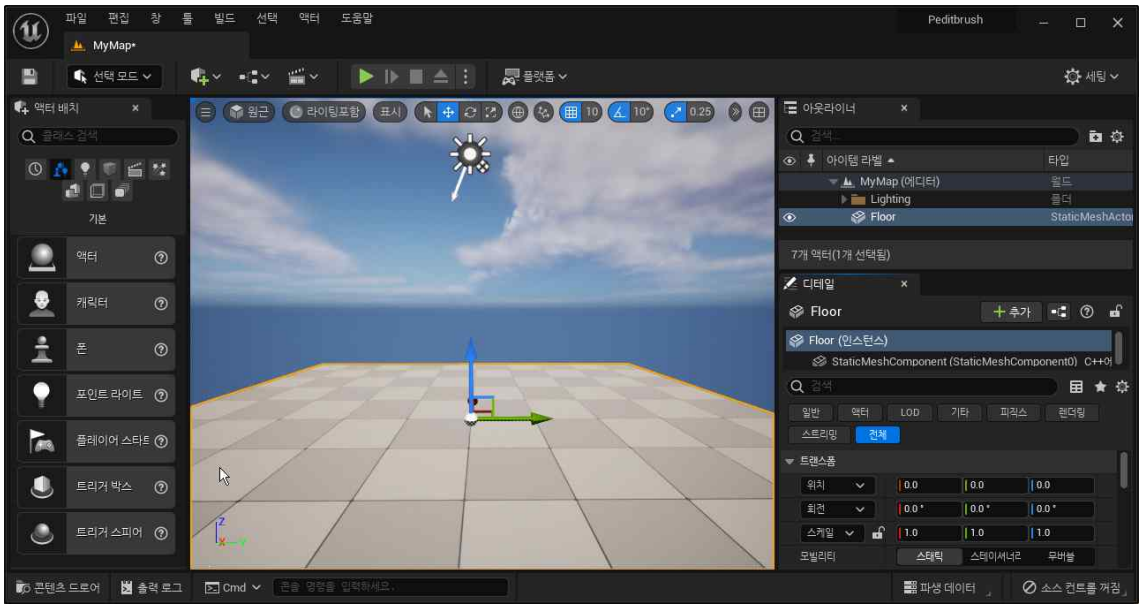
이제부터 실제로 실습해보자.

---

1. 새 프로젝트 **Peditbrush**를 생성하자. 먼저, 언리얼 엔진을 실행하고 **언리얼 프로젝트 브라우저**에서 왼쪽의 **게임** 탭을 클릭하자. 오른쪽의 템플릿 목록에서 **기본** 템플릿을 선택하자. **프로젝트 이름**은 Peditbrush로 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. 현재의 디폴트 레벨에서 있는 스태틱 메시인 **Floor** 액터는 크기가 1000x1000x50인 직육면체 도형이다. 피벗의 위치는 직육면체 윗면의 중심에 있다. 디폴트로, **Floor**의 피벗의 위치가 월드에서 (0,0,20)에 있도록 배치되어 있다. 우리는 **Floor**의 윗면이 월드에서 XY평면과 일치하도록 수정하자. 이를 위해서, **Floor**가 선택된 상태에서 **디테일** 탭에서 **트랜스폼** 속성에서 **위치**의 Z값을 -0.5에서 0으로 수정하자. 그리고 **스케일**을 (8,8,8)에서 (1,1,1)로 수정하자.
이제 **Floor**가 XY평면의 원점에 위치하고 크기가 1000x1000이므로 설계도의 격자와 일치한다.
지금까지의 단계는 이전 예제에서와 동일하게 진행되었다.

3. 이제, **브러시**를 사용하여 레벨을 만들어보자.
메뉴바에서 **창** » **액터 배치** 메뉴를 선택하자. **액터 배치** 탭이 레벨 에디터의 왼쪽에 추가될 것이다. 이번 절에서는 **콘텐츠 브라우저** 탭은 사용하지 않으므로 닫자. 레벨 에디터는 다음과 같은 모습이 될 것이다.



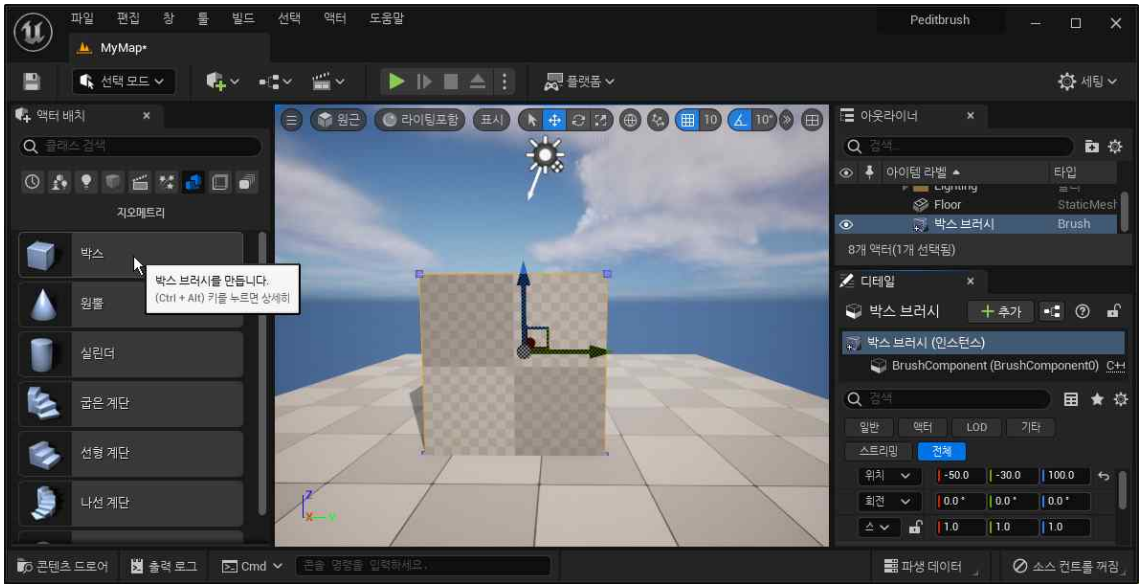
4. **액터 배치** 탭을 살펴보자. 이 탭은 마우스를 드래그하여 액터를 쉽게 레벨에 배치할 수 있도록 한다. 모두 9개의 세부 탭이 있다. 각각 순서대로 **최근 배치됨**, **기본**, **라이트**, **셰이프**, **시네마틱**, **비주얼 이펙트**, **지오메트리**, **볼륨**, **모든 클래스**의 탭이다. 디폴트로 **기본** 탭이 표시된다.
최근 배치됨 탭에는 최근에 레벨에 추가된 액터가 나열된다. **기본** 탭에는 가장 빈번하게 사용하는 주요 액터가 나열된다. **라이트** 탭에는 조명과 관련된 액터가 나열된다.

셰이프 탭에는 기본적인 스태틱 메시 액터가 나열된다. 우리는 이전 예제에서, 레벨에 상자를 배치하기 위하여 **콘텐츠 브라우저**에서 **엔진 콘텐츠** 아래의 **BasicShapes** 폴더에서 **Cube** 스태틱 메시를 드래그하여 배치하였다. 이렇게 하는 대신에, **액터 배치** 탭의 **셰이프** 탭에서 **큐브** 아이템을 드래그하여 레벨 배치해도 정확히 동일한 결과를 얻는다.
시네마틱 탭에는 시네마틱 무비를 제작할 때에 사용되는 시네마틱 카메라 관련 액터가 나열된다. **비주얼 이펙트** 탭에는 시각 효과와 관련된 액터가 나열된다. **지오메트리** 탭에는 브러시 액터가 나열된다. **볼륨** 탭에는 다양한 기능의 볼륨 액터가 나열된다. **모든 클래스** 탭에는 종류를 구분하지 않고 전체 액터가 나열된다.
우리가 사용하려는 브러시 액터는 **지오메트리** 탭에서 나열되어 있다. 이 액터들을 **브러시** 또는 **지오메트리 브러시**라고 한다. '**지오메트리 브러시**'란 언리얼 엔진에서 지원하는 모델링 기능으로 레벨 아이디어의 시안을 빠르게 만드는 방법을 제공한다. 이것의 장점은 외부 DDC 툴로 작성한 별도의 메시 애셋이 없이도 레벨 디자인을 쉽게 진행할 수 있다는 점이다. **지오메트리** 탭을 클릭해보자.

각각의 **브러시** 액터를 레벨에 드래그해보고 모양을 살펴보자.

5. 이제부터, **박스** 브러시를 드래그하여 설계도와 같은 모습이 되도록 배치해보자.
먼저, **박스** 브러시 하나를 레벨에 배치하자. **박스** 브러시는 정육면체 상자 모양으로 크기는 200x200x200이다. 그리고 피벗은 정육면체의 중심에 있다.

브러시는 배치될 때에 이전의 스태틱 메시와는 다르게 브러시의 하단면이 뷰포트 표면의 상단면에 부착되도록 배치된다. 따라서 배치된 브러시의 **트랜스폼**의 **위치** 속성을 보면 Z값이 100으로 되어 있을 것이다.

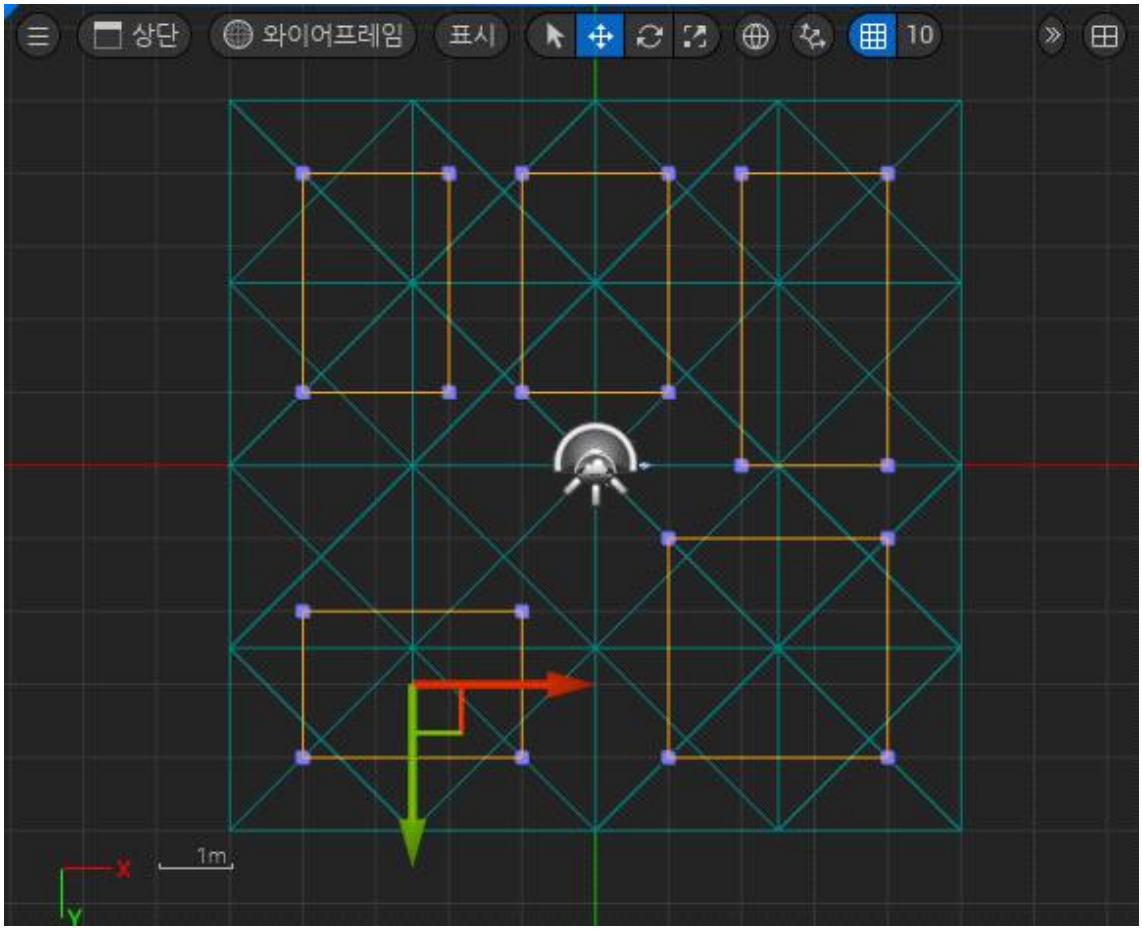


6. 이제부터 이전 예제에서와 거의 유사한 방법으로 배치를 진행하면 된다.
먼저, 설계도가 **상단** 뷰와 일치되므로 뷰포트를 **상단** 뷰로 전환하자. 그리고, 레벨의 설계도가 되도록 하나씩 배치하자. 가장 먼저, 현재 배치된 박스 브러시를 사용하여 설계도의 상자 A를 만들자. 우선, **아웃라이너**에서 현재 배치된 박스 브러시의 이름을 CubeA로 수정하자.

7. 원래 박스 브러시 크기가 설계도의 두 칸에 해당한다. 이전 예제에서는 한 칸이었으므로 스케일 값을 이전 예제에서보다 절반으로 줄여야 한다. 한편, 설계도에서의 높이는 200으로 하였으므로 Z축으로는 스케일할 필요가 없다.
따라서 상자 A의 크기로 바꾸기 위해 **디테일** 탭에서 **트랜스폼**의 **스케일** 값에 (1,1.5,1)을 입력하자. 그다음, **Ctrl+좌클릭+드래그**로 CubeA를 설계도의 상자 A의 위치로 옮기자. **위치** 값에 (-300,-250,100)을 입력하면 된다.

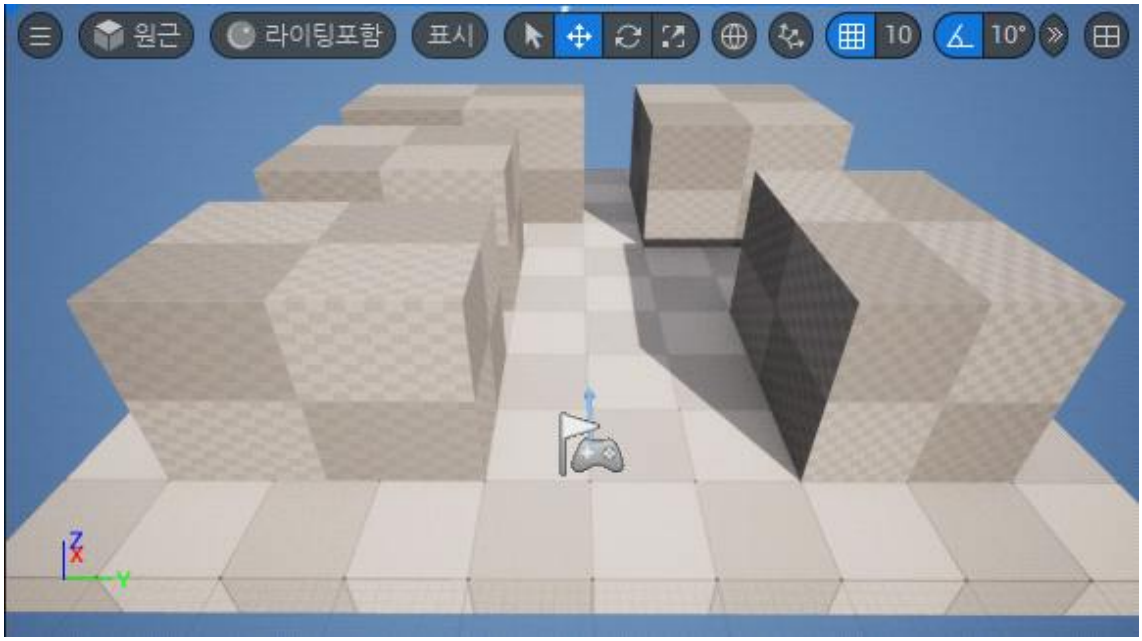
8. 다음으로, 설계도의 상자 B를 만들자. 뷰포트에서 CubeA의 이동 기즈모의 X축을 **Alt+좌클릭+드래그**로 오른쪽으로 한 칸 드래그하여 배치하자. **위치** 값에 (0,-250,100)을 입력하면 된다. 그다음, 액터의 이름을 CubeB로 수정하자.

9. 다음으로, 설계도의 상자 C를 만들자. 뷰포트에서 CubeB의 이동 기즈모의 X축을 **Alt+좌클릭+드래그**로 오른쪽으로 한 칸 드래그하여 배치하자. **위치** 값에 (300,-200,100)을 입력하면 된다. 그다음, 액터의 이름을 CubeC로 수정하자.
그다음, **디테일** 탭에서 **트랜스폼**의 **스케일** 값에서 Y축의 값을 1.5에서 2로 수정하자. 그다음, 다시 이동 기즈모의 Y축을 드래그하여 격자에 맞도록 위치를 맞춘다.
상자 D와 E에 대해서도 동일한 방법으로 진행하여 CubeD와 CubeE를 만들자.
CubeD는 위치가 (250,250,100)이고 스케일이 (1.5,1.5,1)이도록 하면 된다.
CubeE는 위치가 (-250,300,100)이고 스케일이 (1.5,1,1)이도록 하면 된다.
이제 다음과 같이 보일 것이다.



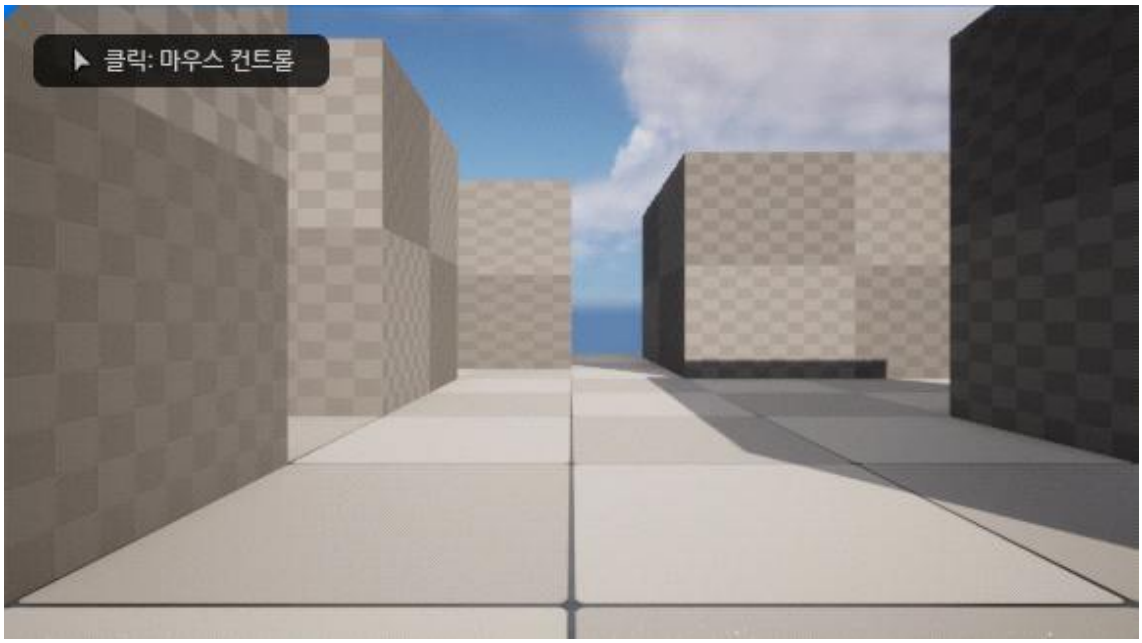
10. **PlayerStart** 액터를 레벨에 추가하자. 먼저 **액터 배치** 메뉴에서 **기본** 탭에 있는 **플레이어 스타트** 액터를 드래그하여 레벨에 배치하자. 추가된 **PlayerStart** 액터가 설계도의 E의 위치에 있도록 디테일 탭에서 **위치** 속성값을 (-400,0,50)으로 수정하자.
저장하자.

11. 이제 **상단** 뷰에서 **원근** 뷰로 바꾸자. 아래와 같이 표시될 것이다.



게임을 플레이해보고 실제 시작 위치와 방향이 **PlayerStart** 액터의 위치와 방향과 일치하는지 확인해 보자.

12. 게임을 플레이해보자. 다음과 같이 표시될 것이다.



지금까지 레벨 에디터에서 브러시를 사용하여 레벨 만들기를 학습하였다.

□