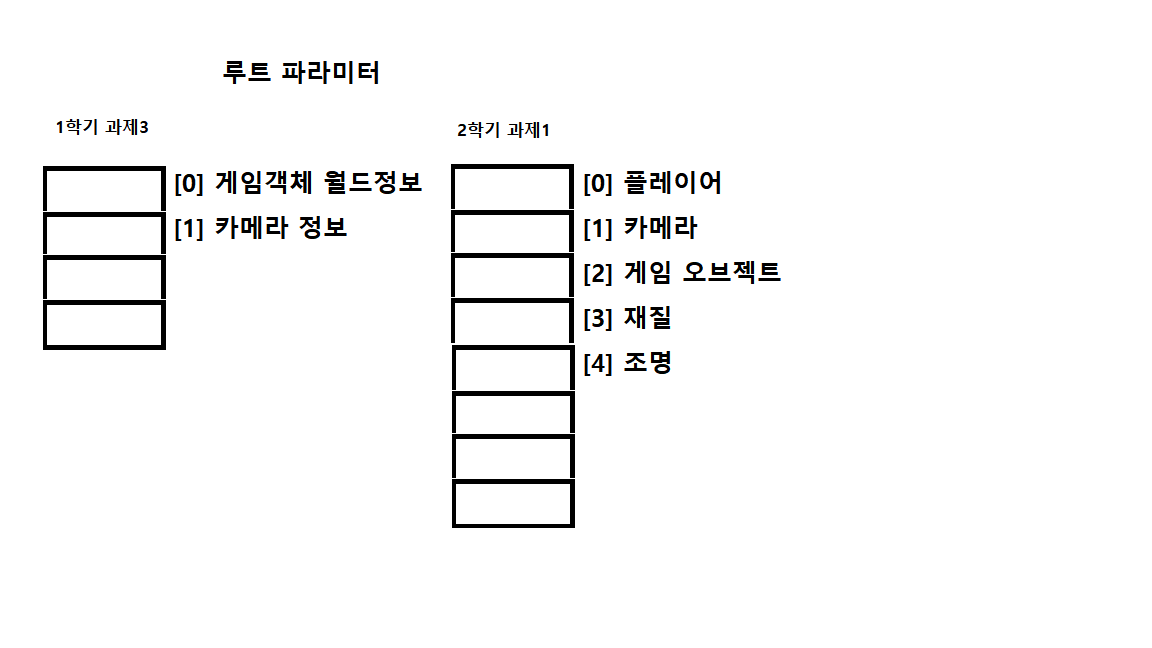
**설명 : 3D게임1 03 의 게임에 조명을 추가 하는 것.**

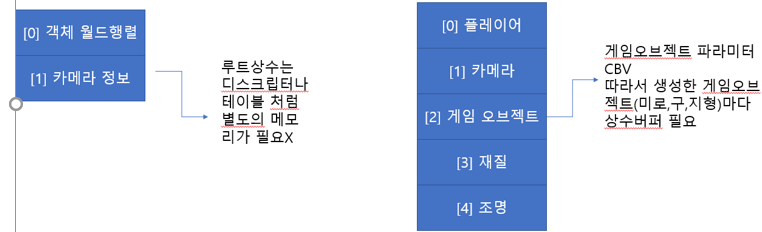
1. **루트 파라미터** – 3D게임1 03과 3D게임2 01의 루트 파라미터 차이점과 해줘야 하는 작업들 위주로 서술하였습니다.
2. **조작법**
3. **구 충돌**
4. **지형**
5. **집 안의 조명**
6. **루트 파라미터**



처음 가정하기에 3D게임1 03에서 사용한 코드에 조명만 추가하면 될거라 생각했지만 그러기엔 문제가 있습니다

3D게임1 03에서의 루트파라미터는 루트상수 2개로 이루어진 반면 조명을 추가하기 위해 루트파라미터가 위의 사진처럼 CBV 4개로 이루어져 있습니다. 루트상수는 값을 바로 쓸것이기 때문에 별도의 메모리가 테이블이나 서술자처럼 필요하지가 않았습니다. 따라서 CreateShaderVariables()에서 별다른 작업이 필요 없는 반면

3D게임2 01는 CBV라 CreateShaderVariables()에서 상수버퍼를 할당할땐 256의 배수로의 업로드힙으로 만드는 작업이 필요하고. 이 상수버퍼를 오브젝트 개수만큼 만드는 작업이 필요했습니다. (할당한 상수버퍼에 오브젝트 개수만큼 곱함) **(관련코드 Shdaer.cpp364~385),** 오브젝트 뿐만 아니라 플레이어, 카메라, 재질, 조명 모두 위와 동일한 작업을 거칩니다.



조명은 씬에 위치하고, 게임오브젝트는 쉐이더에서 위치함으로 조명은 씬에서, 게임오브젝트는 쉐이더에서 CreateShaderVariables() 로 상수버퍼를 할당합니다.

이제 상수버퍼 메모리가 있으면, 이 메모리를 설명하기 위한 디스크립터 (CBV)를 사용합니다.

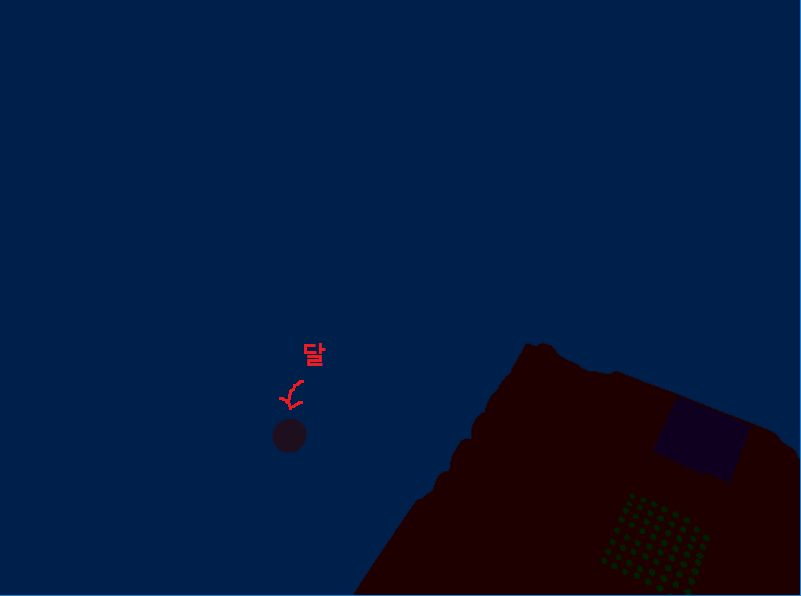
UpdateShaderVariable()함수를 통해 shdaer.cpp나 Scene.cpp에 있는 변수들의 내용을 쉐이더에 memcpy 함수를 통해 전달해주는 작업을 합니다.**(관련코드 Shdaer.cpp 387~432)**

그 후 메쉬를 렌더하기전에 실제로 오브젝트 정보가 어느 주소에 있는지(디스크립터의 목적) 필요한 정보가 어디에있는지 알려주면서 SetGraphicsRootConstantBufferView 함수를 사용해 셋을 하는 작업들이 필요했습니다.

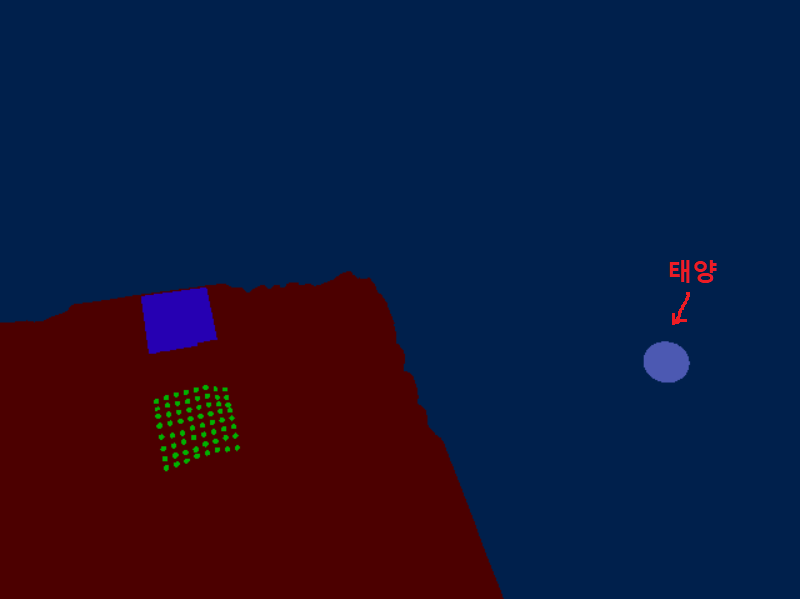
CGameObject::Render 함수에서도 3D게임1 03에선 객체의 월드 정보를 0번 파라미터를 통해 셋했기 때문에 UpdateShaderVariables() 함수가 필요했던 반면 3D게임2 01에선 각각의 루트파라미터마다 따로 Set을 해줄꺼기 떄문에 CGameObject::Render 에 때문에 UpdateShaderVariables() 함수가 필요가 없었습니다.

1. **조작법**

N키 = 달

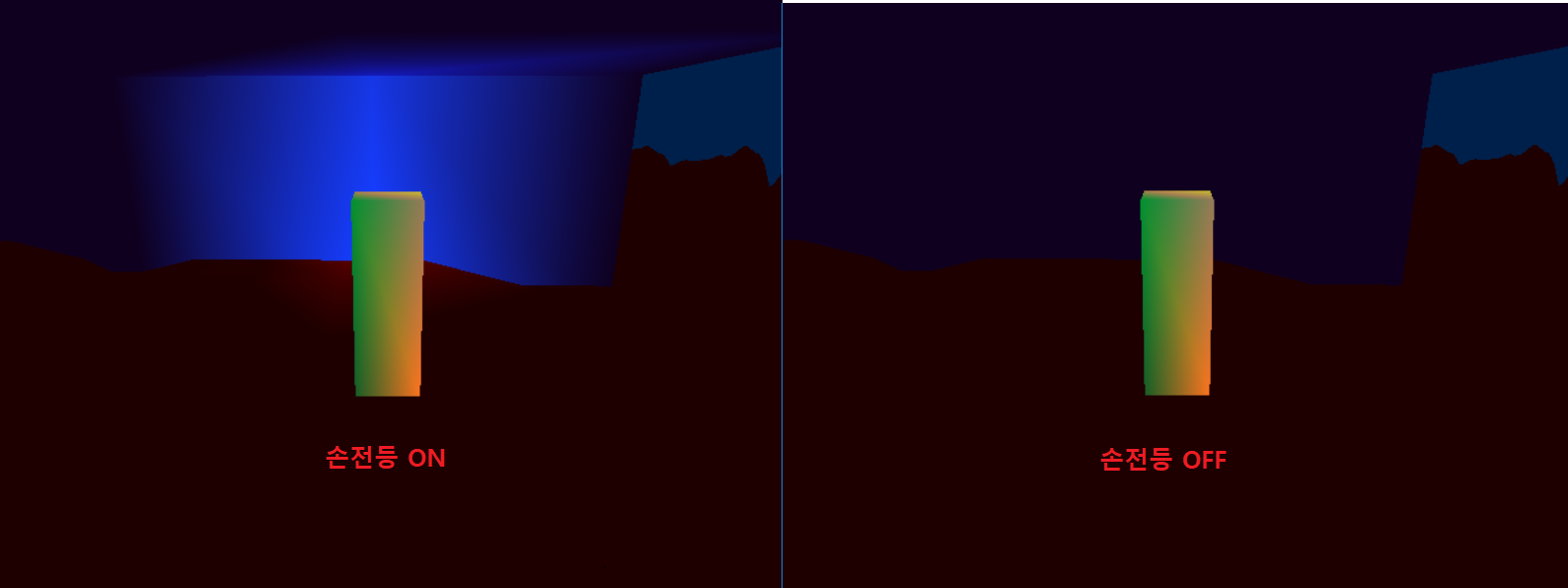


D키 = 낮



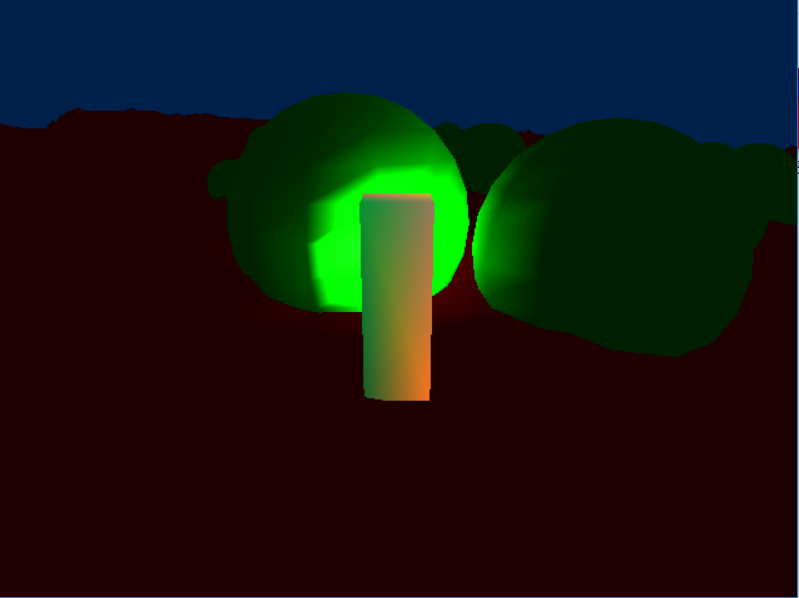
방향키 : 플레이어 이동

Space : 손전등 On/OFF



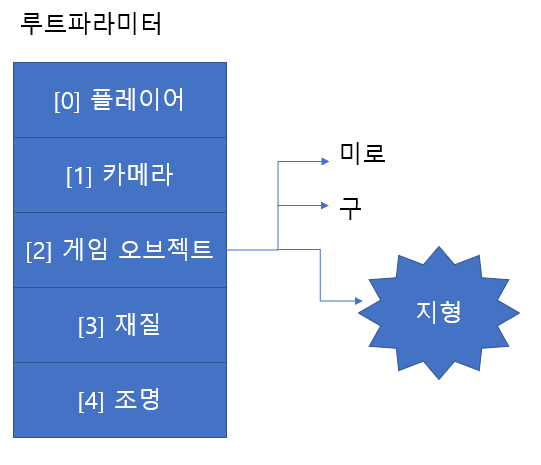
마우스 클릭 : 플레이어 LookVector 변경

1. **구 충돌**

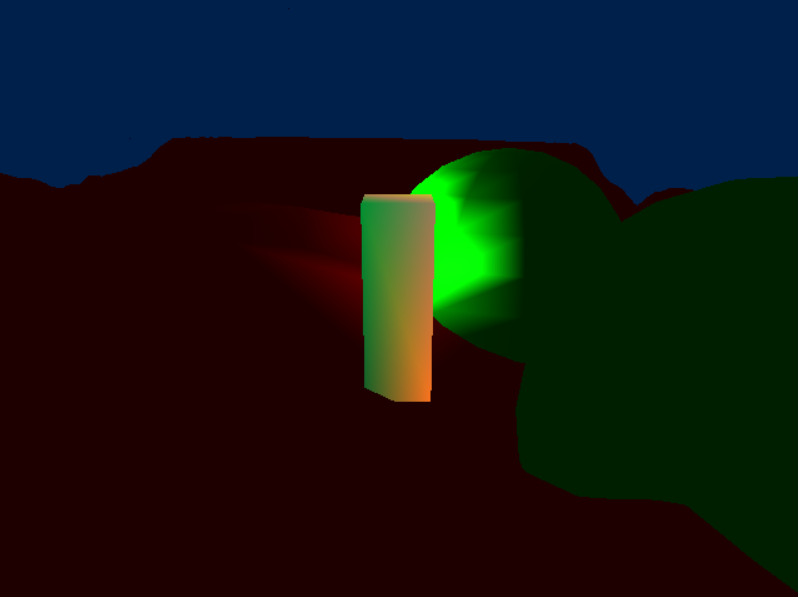
****

구가 있으면 부딪히는 한쪽 구를 옆으로 이동시켰습니다. (겹치게 되지 않기 위해), 충돌 로직은 3D게임1 03에서 썼던 내용을 사용하여 구현하였습니다. (OOBB를 설정하고 서로의 collider 정보를 확인하면서) **(관련 코드 Shader.cpp 1614~ 1640)**

1. **지형**

****

지형을 위한 CTerrainShader 를 정의해주고. CreateInputLayout 에서 Color대신 Normal값을 줘야 합니다. **(관련코드 Shader.cpp 1799)**

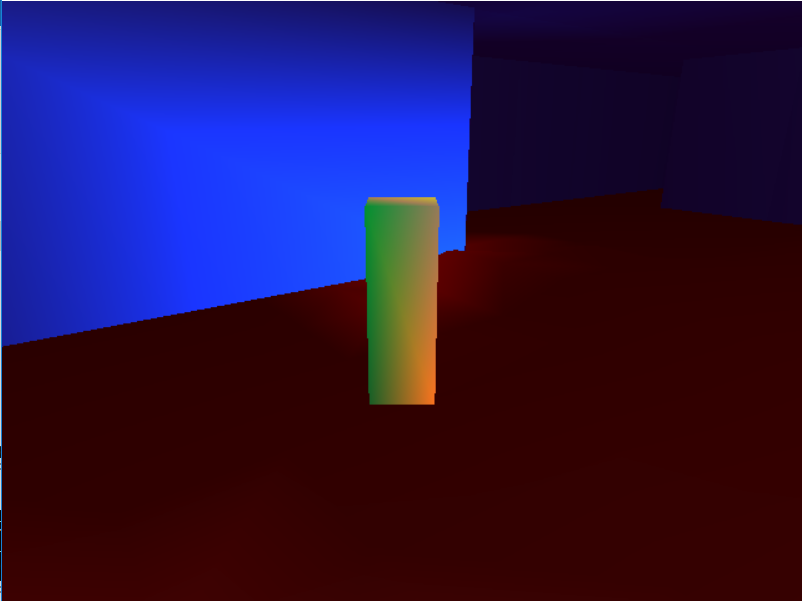
****

지형의 색깔은 1학기 과제 3에선 임의로 정해줬지만 이젠 재질과 빛의 상호작용으로 계산해 주어야 합니다. (따라서 XMFLOAT4 CHeightMapGridMesh::OnGetColor(int x, int z, void \*pContext) 함수에서

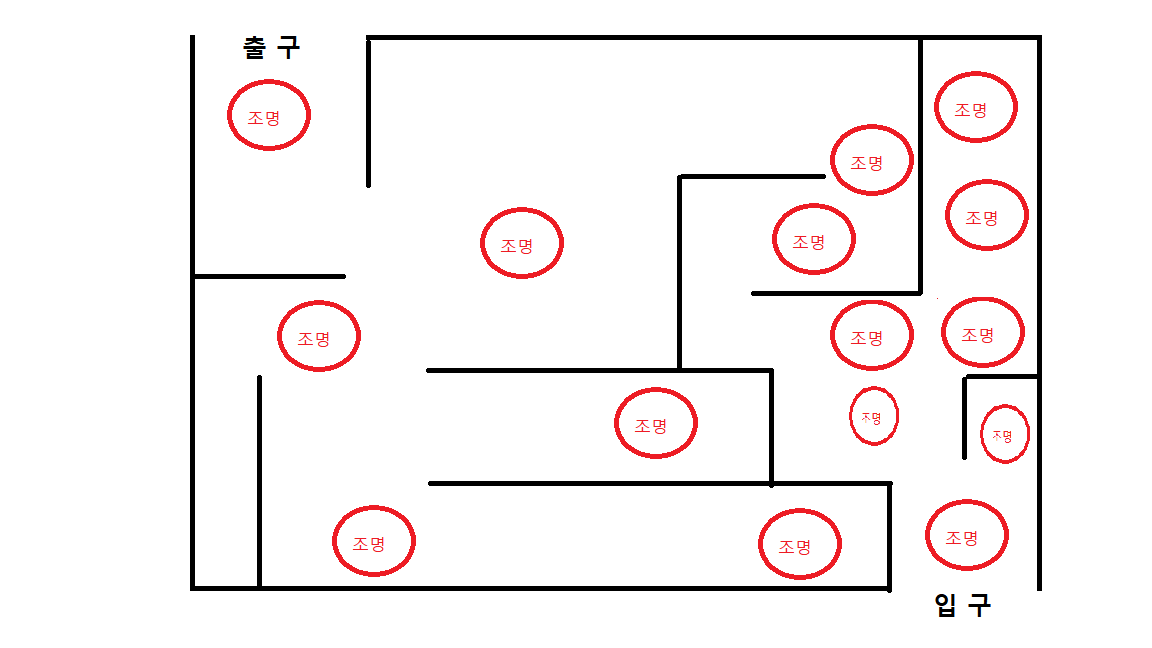
xmf4IncidentLightColor의 인자값을 0으로 하였습니다 **(관련 코드 Mesh.cpp 837줄)**

지형의 Build와 Render 모두 Shdaer.cpp 에서 **(관련 코드 Shdaer.cpp 464줄 및 Shdaer.cpp 1761줄 )** 해주었습니다. (이유는 Terrain 도 루트파리미터의 2번째 엔트리 (게임오브젝트)에 Set 할것이고, 그럴려면 지형을 위한 CreateShaderVariables, UpdateShaderVariables, ReleaseShaderVariables, 을 해주기 위해서입니다.

1. **집 안의 조명**



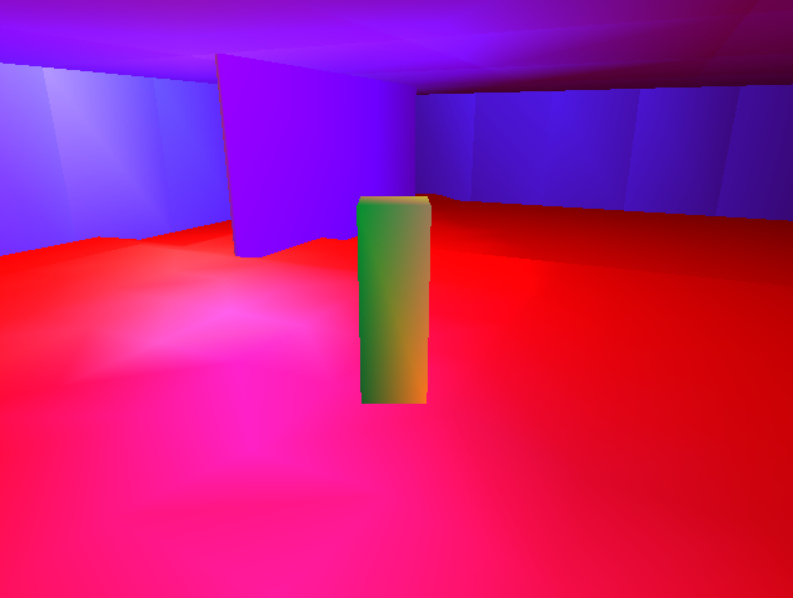
집에 들어오는 느낌을 주기 집에 들어오면 Directional Light의 값을 줄여 주었습니다. **(관련코드 Scene.cpp 561~ 576줄)**



집안의 조명은 총 16개로, 처음에는 다 꺼져있는 상태입니다

플레이어와 조명이 일정 거리 이하로 됬을 때(플레이어의 위치백터와 조명의 위치벡터 사이의 거리를 이용하여) 꺼져있던 조명이 켜지도록 하였습니다 **(관련 코드 Scene.cpp 496 ~ 517)**

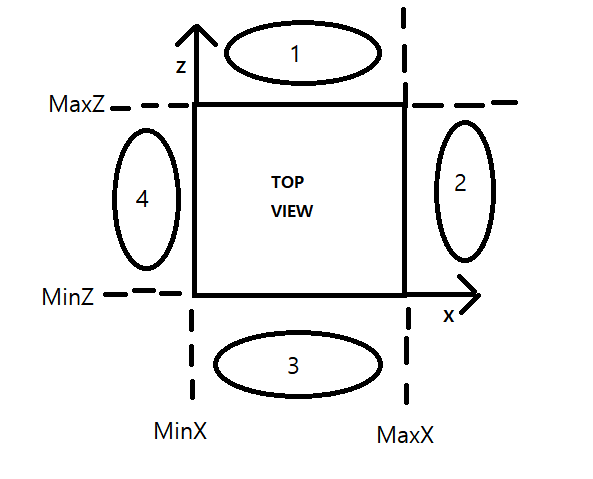
<집안 조명 켜져있을 때>



1학기 과제 03에서는 벽면사이즈가 가로가 300이거나 세로가 300인 사이즈가 있었습니다. 이는 벽의 메쉬에 폴리곤 수가 적어 조명계산이 안되는 경우가 있어서 벽을 여러 개의 오브젝트로 나누었습니다

1. **벽 충돌 체크**

벽들 충돌체크



벽은 직육면체이고 하늘에서 바라보면 위와 같은 그림이 된다.

1번 영역 : (p(플레이어)의 Z좌표 > MaxZ) && (p의 X좌표 > MinX) && (p의 X좌표 < MaxX)

2번 영역 : (p의 x좌표 > MaxX) && (p의 z좌표 > MinZ) && (p의 z좌표 < MaxZ)

3번 영역 : (p의 z좌표 < MinZ) && (p의 x좌표 > MinX) && (p의 x좌표 < MaxX)

4번 영역 : (p의 x좌표 < MinX) && (p의 z좌표 > MinZ) && (p의 z좌표 < MaxZ)

그래서 만약 벽의 oobb와 플레이어의 oobb가 intersects 한 값이 존재 한다면 위와 같은 식으로 충돌처리를 할 수있다. 충돌을 하면 각 1,2,3,4 면 중 하나에 해당하는 노멀벡터로 플레이어를 밀어내도록 하였습니다. **(관련코드 Shdaer.cpp 1523줄~ 1560줄)**