3D 프로그래밍2

과제 01

게임공학과

2013180003 김나단

1. 이번 과제의 목표

3d게임프로그래밍1의 숙제3번을 수정하여 3d게임프로그래밍2 숙제 1번을 수행하고자 하였습니다. Descriptor와 Descriptor Table, 그리고 RootSignature의 사용과 적용이 익숙해지는 것을 목표로 삼았습니다. 또, 이전 과제3번의 미로 맵을 수정하여 맵에서 라이트를 읽어와 포인트 라이트를 생성하는 것을 목표로 삼았습니다.

1. 프로그램에 대한 가정

이번 과제는 3d게임프로그래밍1의 과제의 연장이라고 가정하고 진행하였기에 프로젝트 명이 Homework4로 되어있습니다. 이 점 양해부탁 드립니다.

이번 과제의 목표인 Root Signature 등과 Light에 집중하기 위하여 기존 과제3에서 이번 과제에서 사용하지 않는 기능을 제거하였습니다. 제거된 기능은 다음과 같습니다.

1. 오브젝트 클릭 시 해당 오브젝트를 바라본다.
2. 적 오브젝트 클릭 시 총알이 날아간다.
3. 적 오브젝트가 총알에 맞은 경우 파티클을 생성한다.
4. 플레이어 오브젝트가 적 오브젝트에 닿은 경우 파티클을 생성하고 일시 정지한다.

위의 기능 외에 제거된 기능은 없으며 현 과제를 수행하며 수정되는 부분에 맞춰 기존 기능이 수행되도록 하였습니다. 유지된 주요 기능은 다음과 같습니다.

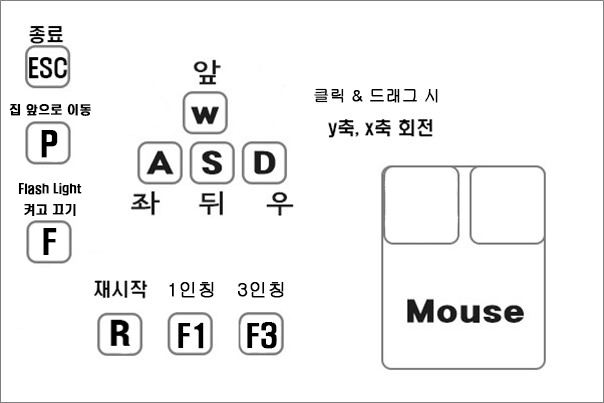
1. P를 누를 경우 집의 입구로 이동한다.
2. 플레이어는 날아다니지 못하는 것을 가정한다.
   1. 스페이스 쉽 시점을 선택할 수 없다.
3. 벽과의 충돌 시 슬라이딩 벡터를 생성하여 미끄러지듯 이동하도록 한다.

추가한 기능과 수정한 기능은 다음과 같습니다.

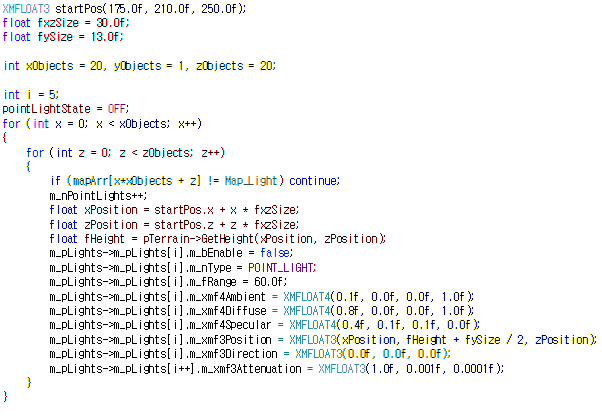
1. 플레이어와 적 오브젝트의 충돌 시 적 오브젝트의 속도는 플레이어의 속도 + 기존 적 오브젝트의 속도를 가지게 한다.
   1. 플레이어에 의해 튕겨져 날아가는 효과를 얻을 수 있다.
2. 적 오브젝트 회전축을 사전에 정한다.
   1. 이전 과제에서 플레이어를 향해 오기 때문에 실시간으로 회전축을 계산했으나 이번 과제에서는 한 방향으로 이동하므로 회전축을 오브젝트 생성 시 속도와 Up Vector를 이용하여 구해 저장해 둔다.
3. 적 오브젝트간 충돌 처리를 추가한다.
   1. 적 오브젝트간 충돌한 경우 서로의 속도, 회전 축을 교환하여 반사를 구현한다.
4. 플레이어의 라이트는 플레이어 오브젝트가 바라보는 방향으로 한다.

이번 과제를 진행한 순서 및 추가 사항은 다음 같습니다.

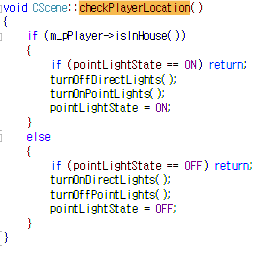
1. Root Signature를 Descriptor와 Descriptor Table을 사용하도록 수정
   1. 이에 필요한 Descriptor Heap, Mapped 변수를 추가
2. Lighting에 필요한 Mesh생성 및 수정
   1. Illuminated- 메쉬 추가
   2. 지형 정점에 Normal저장할 수 있도록 수정
3. 플레이어를 제외한 오브젝트들을 Lighting을 이용하여 그리도록 수정
   1. Input layout 수정 및 CB\_GAMEOBJECT\_INFO 구조체 HLSL과 동일한 구조로 수정
   2. CreateShaderVariable 수정하여 적절하게 값을 넘기도록 수정
   3. Object에서 Material 구조체 제거하고 Index를 저장할 int형 변수 추가
   4. 위 사항에 맞도록 HLSL의 구조체 수정
4. Lights 추가
   1. Directional Light 고정된 상태로 추가
   2. 손전등 용 Spot Light 추가, F키로 켜고 끌 수 있는 기능 추가
   3. 맵에서 Point Light읽어와 사용할 수 있도록 코드 추가
5. 플레이어 Lighting으로 그리도록 수정
   1. Player Info에 메터리얼 index저장할 변수 추가, input Layout 수정, Update Shader Variable 수정
6. 쉐이더에서 중복된 코드 제거
7. Directional Light를 움직일 수 있도록 COrbitObject 추가
   1. Center, Degree, Radius를 지정해주고 Animate시 x, y를 회전
   2. 태양, 달의 경우 거리에 상관 없이 항상 보일 수 있도록 IsVisible함수 오버로딩하여 항상 true리턴
   3. 맵 뒤로 돌아간 경우 라이트를 끌 수 있도록 확인할 함수 추가
8. 최종 확인 및 버그 수정
   1. Input Layout에 Format을 제대로 설정해 주지 않아 색이 안 그려지는 부분이 나오는 버그 수정
   2. 벽과의 충돌체크가 제대로 되지 않는 버그 수정
   3. 태양, 달 오브젝트를 밑에서 본 경우 까맣게만 나오지 않도록 HLSL코드에 Emissive계산 추가
9. 조작 방법



1. 코드 설명
2. Scene
   1. BuildLightsAndMaterials에서 Point Light 배치 시 맵 데이터를 읽어와 배치하였습니다. 또한 지형의 높이 정보를 이용하여 지형 밑으로 묻히지 않도록 하였습니다.

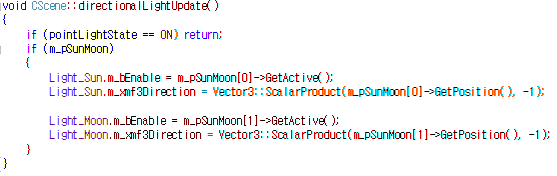


* 1. CreateGraphicsRootSignature에서 플레이어, 카메라, 매터리얼, 라이트를 Descriptor로 설정하였으며, 오브젝트의 경우 Descriptor Table로 설정하였습니다.
  2. AnimateObjects에서 checkPlayerLocation와 directionalLightUpdate함수가 추가 동작하도록 하였습니다. 각 함수의 상세 구현은 다음과 같습니다.
     1. checkPlayerLocation



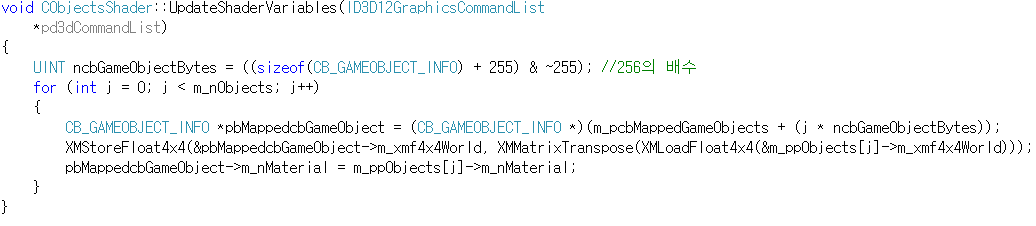
먼저, 플레이어 위치를 검사하여 집 안인 경우 Directional Light를 끄고 Point Light를 활성화 합니다. 집 밖인 경우 반대로 진행합니다. 이때, 각 프레임 별로 중복해서 작업을 하지 않도록 플래그를 설정하여 한번 설정된 이후에는 플래그 검사를 통해 중복 작업처리를 하지 않도록 하였습니다.

* + 1. directionalLightUpdate

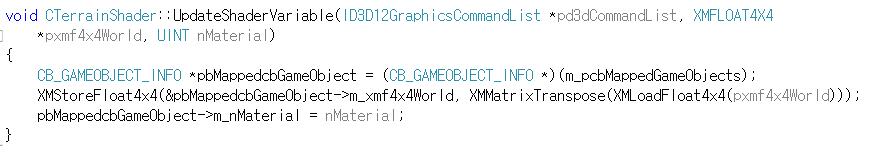


해와 달을 나타낼 오브젝트 배열을 가져와 사용합니다. 이때 m\_pSunMoon의 인덱스가 나타내는 것은 순서대로 해, 달입니다. 이를 이용해 각 위치를 정해줍니다. 특히, Directional Light에 해당하는 해, 달은 위치에 -1을 곱해 해당 방향에서 빛이 오는 느낌이 되도록 하였습니다.

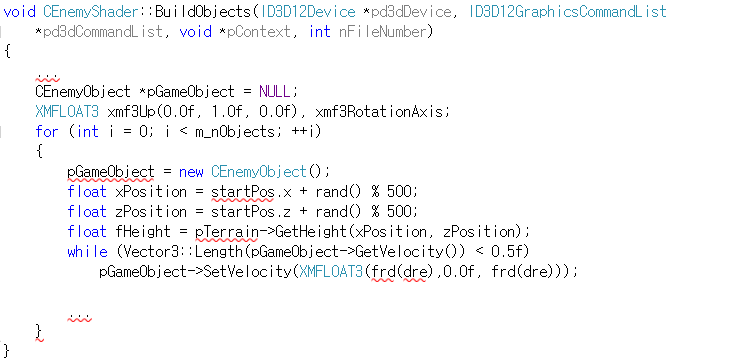
1. Shader
   1. ObjectShader
      1. ObjectShader에서 파생되는 쉐이더들의 중복 함수를 제거하였습니다. 또, Descriptor Table을 사용하기 위하여 CreateShaderVariable ReleaseShaderVariable UpdateShaderVariable의 함수를 그에 맞게 수정하였습니다.
      2. CreateShaderVariable의 경우 업데이트를 위해 Mapping해주고 추가적으로 Descriptor Table에서 사용하기 위한 Descriptor Heap을 생성하고 오브젝트 개 수 만큼 CBV를 생성하였습니다.
      3. Object들 각각이 Material 구조체를 가지고 있는게 부담이 된다고 생각하여 각 오브젝트는 그 오브젝트에서 사용할 재질의 인덱스를 가지고 있도록 하였습니다.



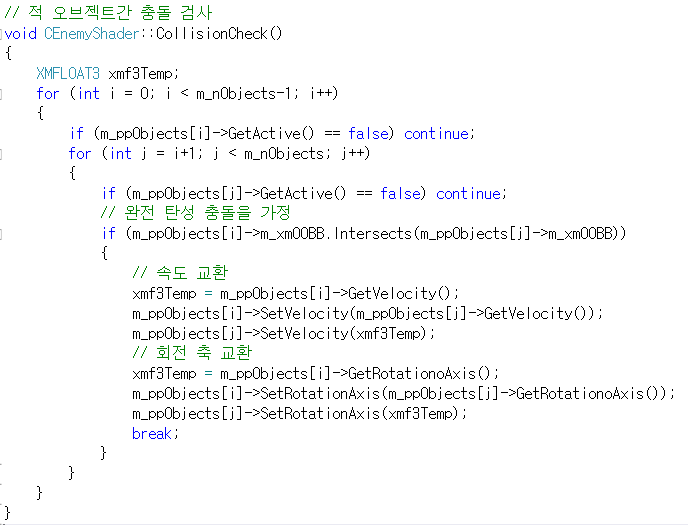
* + 1. Light를 사용하기 위하여 CreateInputLayout을 {POSITION, COLOR} 에서 {POSITION, NORMAL}로 수정하였습니다. 또, 그에 맞게 Format도 기존 COLOR:DXGI\_FORMAT\_R32G32B32A32 에서 NORMAL: DXGI\_FORMAT\_R32G32B32 로 수정하였습니다.
  1. PlayerShader
     1. 기존 VSPlayer / PSPlayer에서 라이팅을 사용하기 위해 VSPlayerLighting과 PSPlayerLighting으로 수정하였습니다.
  2. TerrainShader
     1. 기존에 중복되던 코드를 제거하였습니다. 또 Player와 Terrain같이 오브젝트가 쉐이더를 가지는 경우 외부에서 월드변환 행렬과 재질의 인덱스를 받는 UpdateShaderVariable을 구현하였습니다.



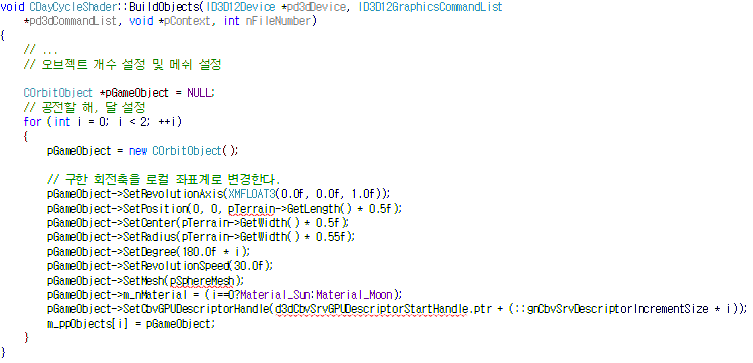
* 1. EnemyShader
     1. BuildObject에서 각 오브젝트의 방향과 회전축을 정해주었습니다.



* + 1. EnemyShader에 오브젝트 간 충돌검사를 추가하였습니다. 한 오브젝트가 여러 오브젝트와 충돌이 발생할 수도 있다고 가정하였으며 충돌 검사를 판단하 즉시 충돌한 오브젝트와 속도, 회전축을 교환합니다. 이것으로 완전 탄성 충돌을 가정한 충돌을 구현하였습니다.



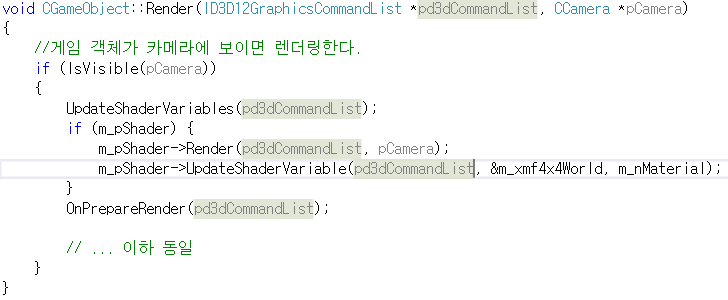
* 1. DayCycleShader
     1. 달, 해의 공전을 위한 쉐이더를 추가하였습니다. 각 오브젝트는 이후 설명할 OrbitObject를 가지고 있으며 GameObject를 상속하여 BuildObject만 수정하는 형태로 구현하였습니다.



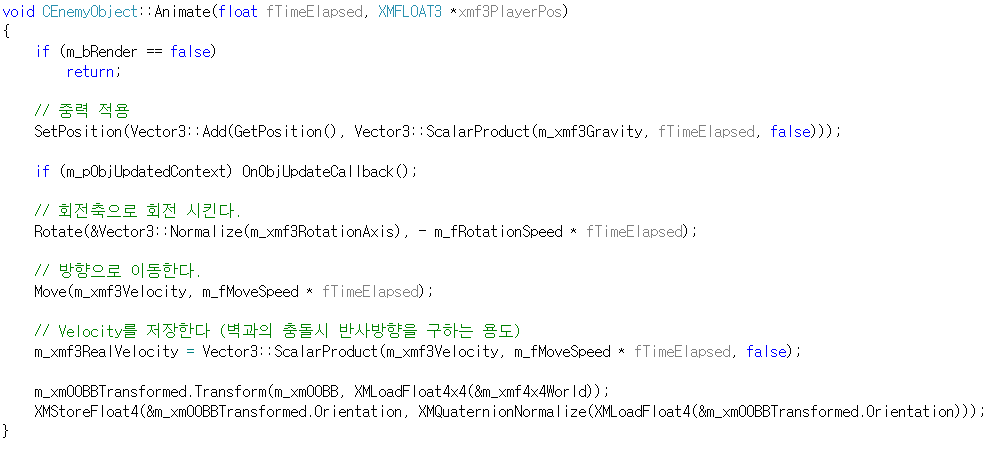
1. GameObject
   1. GameObject
      1. m\_pMaterial 구조체 대신 int형 m\_nMaterial을 가지게 하였습니다.
      2. OnPrepareRender함수를 추가하여 Descriptor Table사용시 Set하는 부분을 넣었습니다.



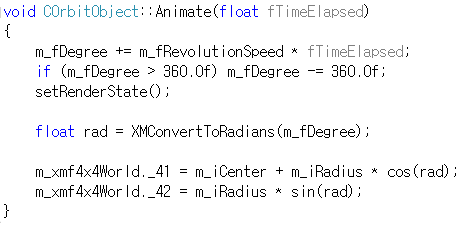
* + 1. Render함수를 Material안의 Shader를 꺼내는게 아닌 기존처럼 가지고 있는 Shader를 이용하여 렌더링 하도록 하였습니다.



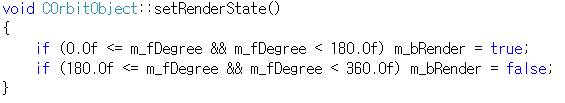
* 1. EnemyShader
     1. 회전 축을 가지고 있으므로 이전의 동적으로 회전축을 구하는 방식에서 고정된 회전축을 이용하여 회전하는 방식으로 변경하였습니다.



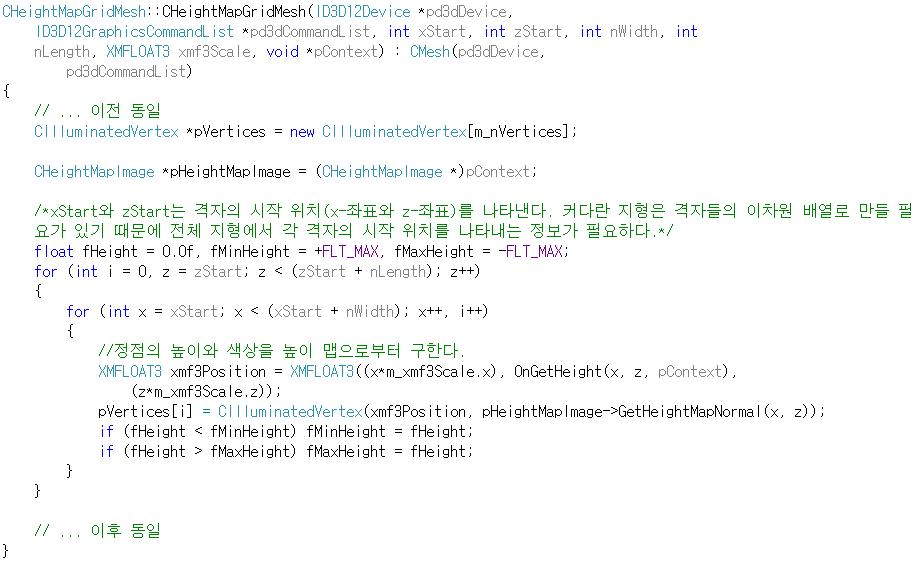
* 1. OrbitObject
     1. 해와 달의 공전을 구현하기 위해 구현한 오브젝트 클래스입니다. 회전 중심 Center, 반지름 Radius, 각도 Degree를 이용하여 회전을 구현하였습니다. 이때 해 달은 Z축으로 회전하며 Z로는 움직임이 없다고 가정하여 2차원 회전으로 구현하였습니다.



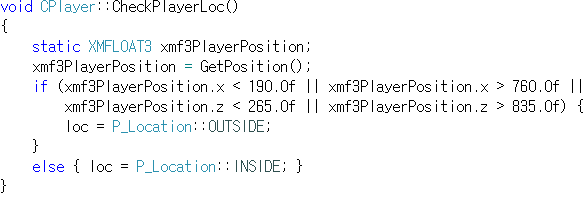
* + 1. 지형의 뒤쪽으로 돌아간 경우 Light를 끄기 위하여 Degree가 180도보다 커진 경우 bool 변수를 false로 하였습니다.



1. Mesh
   1. Light에서 사용할 오브젝트들의 메쉬를 추가하였습니다. CubeMeshIlluminated, ShpereMeshIlluminated는 기존 예제에서 주어진 것을 가져와 사용하였으며 충돌처리를 위해 Vertex와 Index의 배열을 저장하도록 수정하였습니다.
   2. RoofMeshIlluminated는 CubeMeshIlluminated와 동일하게 구현하되 Vertex를 기존 RoofMeshDiffused의 Position을 사용하였습니다.
   3. MeshIlluminated
      1. 예제 코드에 제공된 것을 가져왔습니다.
      2. 충돌 처리를 위해 CheckRayIntersection을 오버라이딩 하였습니다. 구현은 기존과 동일하나 기존 코드의 경우 DiffuedVertex의 배열을 사용하며 현재 Illuminated의 경우 IlluminatedVertex를 사용하기 때문에 제대로 충돌처리가 되지 않는 문제가 있었으며 이것을 해결하기 위해 함수를 오버라이딩 하였습니다.
   4. HeightMapGridMesh
      1. 생성 시 사용하는 정점을 DiffusedVertex에서 IlluminatedVertex로 수정하였습니다.
      2. 생성 시 Vertex에 Normal을 넘겨주도록 수정하였습니다.



* 1. Player
     1. 생성 시 Mesh를 CubeMeshIlluminated로 변경하였습니다.
     2. 플레이어가 집안/밖인지 판단하는 CheckPlayerLoc 함수를 구현하였으며 플레이어 Update시 Move호출 이후 호출하도록 하였습니다.



* + 1. Scene에서 플레이어가 집안에 있는지 밖에 있는지 확인할 수 있도록 isInHouse함수를 추가하였습니다.

C:\Users\ND\AppData\Local\Microsoft\Windows\INetCache\Content.Word\3.png

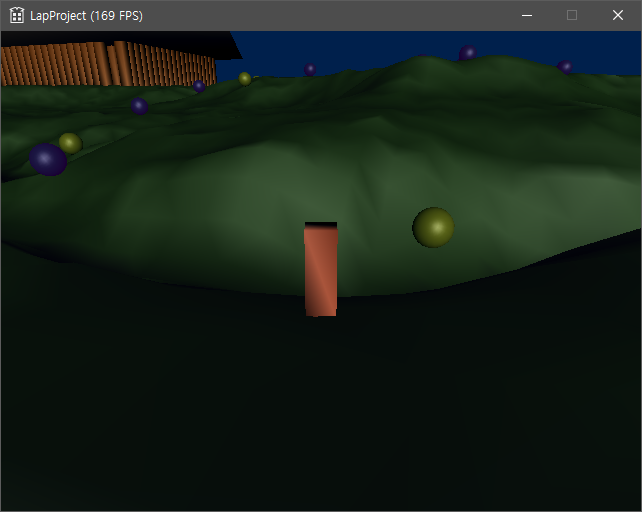
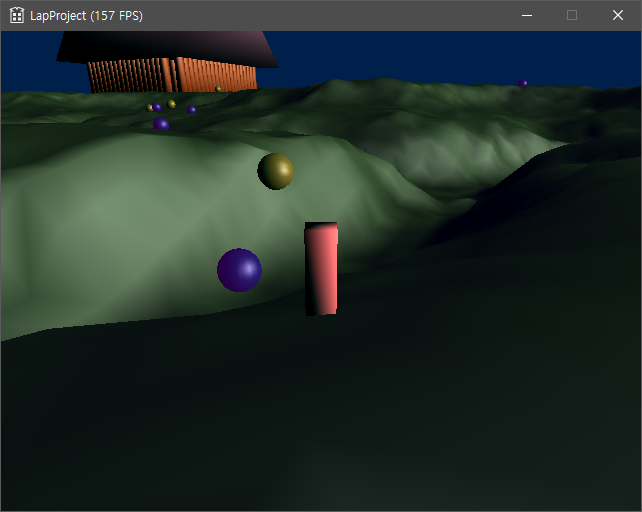
1. Shader.hlsl
   1. 기존 Lighting함수들을 수정하여 재질의 인덱스를 오브젝트의 글로벌 변수에서 가져와 사용하던 것을 플레이어도 Lighting 계산을 할 수 있도록 각 함수가 재질의 인덱스를 받아 계산하도록 수정하였습니다. 또, 해, 달이 자체적으로 빛나도록 하기 위해 색상 계산에 Emmissive를 추가하였습니다.



* 1. 플레이어를 위한 VSPlayerLighting과 PSPlayerLighting을 추가하였습니다.



1. 프로그램 실행 화면
   1. 외부 낮 상태 / 밤 상태



* 1. 건물 내부

