2020182031 이서연 졸업작품 4~5주차 진행

1. Deferred Lighting 공부

3D 게임 프로그래밍2의 강의자료 중 지연조명 (Deferred Shading)자료를 보며 지연조명에 대하여 자세히 공부했다.

Deferred Lighting은 G-buffer에 먼저 Depth, Stencil, Normal, Specular Intensity, Specular Power, Motion Vector, Albedo 등을 저장하여 지연 결합을 한 후 거기서 추가로 후처리(Post Processing)도 할 수 있다.

Shader Programming을 담당한 이상 여러 효과 구현을 위해 꼭 구현해야 하는 부분인 것 같다.

1. 소스코드 분석

프로그램을 돌려보면 수많은 정육면체가 돌아가고 있고, Texture와 조명이 존재한다.

그러다 특정 키를 누르면 각 mode에 맞는 사진을 찍듯이 필터가 적용되어 Texture로 반환되는 듯하다.

텍스트이(가) 표시된 사진

자동 생성된 설명

코드를 보면 알 수 있듯이 S모드가 기본으로 육면체들이 돌아가는 화면이며, Scene과 Player를 그려준다.

그 전에 깊이 스텐실 버퍼를 초기화 하고 Laplacian Shader의 OnPrepareRenderTarget 함수를 부른다.

텍스트이(가) 표시된 사진

자동 생성된 설명

OnPrepareRenderTarget 함수를 보면 처음에 Texture의 개수 + Render Target의 개수만큼 Descriptor Handle을 만든 후 Render Target들을 노란색으로 초기화 시켜준다.

그 후 Texture 개수만큼 해당 번호의 Descriptor Handle을 불러온 후 하늘색으로 초기화 시켜주고 위에서 만든 Descriptor Handle들에 추가해 준다.

즉 현재 Render Target만큼 노란색으로 초기화, Texture 개수만큼 하늘색으로 초기화 해준 뒤 둘을 합쳐 새로운 Render Target으로 지정해준다.

그 후 Scene을 그려주는데 Scene엔 Object Shader 한 개를 그려주므로 알아볼 필요가 있다.

먼저 Object Shader은 Texture Shader -> Illuminated Texture Shader -> Object Shader 순으로 상속되어 있는 구조이다.

그렇기 때문에 가장 위에 있는 Texture Shader를 보았다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Texture Shader은 다음과 같은 Vertex Shader와 Pixel Shader을 사용한다.

간단한 코드였다. Position과 UV를 입력 받은 후 Position값을 계산해주고 UV는 그대로 return해준다.

그 UV를 사용하여 Texture를 Sampling하여 색상 값으로 return해준다.

다음은 Illuminated Texture Shader의 Shader Code이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

PositionW은 월드 좌표의 Pixel의 위치, normalW은 월드 좌표, normalV는 뷰에서의 Normal이다.

코드를 살펴보니 최종적으로 사용할 Object Shader에서는 VS\_TEXTURED\_LIGHTING을 Vertex Shader로 사용하고 Multiple Render Target을 사용한 Pixel Shader를 만들어준다.

즉, 위에서 본 Textured Shader과 PS\_TEXTURED\_LIGHTING은 사용하지 않는다는 뜻 이였다.

Vertex Shader의 INPUT으로 데이터를 넣어주는 방법을 몰랐는데 재우 님 에게 물어보았다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Object Shader의 Build Object를 보면 Cube Mesh Illuminated Textured를 사용한다는 것을 알 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Cube Mesh Illuminated Textured의 생성자를 보면 Position과 UV 좌표를 지정해준다.

그 후 Calculate Vertex Normal함수를 사용하여 Normal값을 구해 변수에 저장해주고

Position, Normal, UV를 가지는 Illuminated Textured Vertex 클래스에 넣어 준 후 Vertex Buffer에 이 Illuminated Textured Vertex를 넣어준다.

후에 Get GPU Virtual Address함수로 Vertex Buffer View의 Buffer 위치를 만든 Buffer로 넣어준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

만들어진 Vertex Buffer View는 Mesh를 Render할 때 IA Set Vertex Buffers함수로 넣어준다.

VS\_TEXTURED\_LIGHTING\_OUTPUT을 얻었으면

텍스트이(가) 표시된 사진

자동 생성된 설명

PS\_MULTIPLE\_RENDER\_TARGETS\_OUTPUT을 얻어 낼 수 있다.

내용을 보면 VS\_TEXTURED\_LIGHTING\_OUTPUT과 Primitive ID를 입력 받는 걸 볼 수 있다.

입력 받은 UV와 Primitive ID를 이용하여 Texture를 Sampling한다.

Light.hlsl에 있는 Lighting함수를 입력 받은 position과 normal값을 이용하여 Illumination 즉, 빛의 색을 구해낸다.

Scene은 Swap Chain Back Buffer이고 이곳에 빛 x Texture를 하여 빛처리가 된 화면을 그려준다.

Normal은 입력 받은 월드 Normal을 이용하여 구해준다.

또한 Camera Normal은 아까 뷰의 Normal이였던 NormalW를 이용하여 구해준다.

ObjectIDzDepth의 앞부분엔 Object ID, 뒷부분엔 zDepth값을 넣어주고 return해준다.

이렇게 return해준 값에서 색상을 어떻게 알아내서 사용하는 건지 잘 모르겠다.

텍스트이(가) 표시된 사진

자동 생성된 설명

다시 위로 올라가보면 S모드가 아닐 때는 Scene과 Player를 그리지 않고

Laplacian Shader를 Render해주는 것을 알 수 있다.

이상한 점이 있다면 Laplacian Shader에는 Build Object가 없다.

즉, 무엇을 그려줄지를 명시해두지 않았다는 것이다.

텍스트, 스크린샷, 화면, 은색이(가) 표시된 사진

자동 생성된 설명

이것이 Laplacian Shader가 사용하는 Vertex Shader이다.

인자로 int형인 Vertex ID를 받는데 이것이 무엇을 뜻하며 어디서 넣어주는 지도 알아내야 한다.

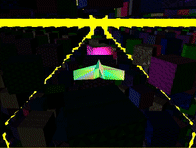
텍스트이(가) 표시된 사진

자동 생성된 설명

그리고 이것이 Pixel Shader이다.

Texture2D gtxtInputTextures[7] : register(t1); //Color, NormalW, Texture, Illumination, ObjectID+zDepth, NormalV, Depth 가 사전에 정의되어 있는데, 저 7개의 순서대로 정보가 담긴 Texture 배열이다.

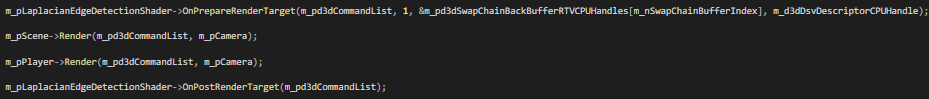
이 정보들도 어디서 어떻게 들어가는지 알아내야 한다.

 이것이 프로그램을 실행한 결과이다.

노란 배경일때가 S모드, 즉 Scene과 Player를 그려주어서 움직이는 상태이고, 그 외가 순서대로

Texture, 조명, Normal, 깊이, View Normal, Object유무, Laplacian Edge를 나타내는 모드이다.

S모드 일 때는 모두 움직이다가 Laplacian Shader로 바꾸면 사진을 찍듯이 세상이 멈추며 filter가 씌워지고, 다시 S모드가 되면 움직인다.



아무래도 Laplacian Shader에서 그려줄 내용을 여기 위아래에서 해주는 것 같지만 코드를 봐도 어려워서 잘 모르겠다 분석이 오래 필요할 것 같다.

S모드일때의 화면정보를 Laplacian Shader에게로 보내는 코드를 찾아보아야 할 듯하다.

이번주에 Multiple Render Target과 이를 이용해서 Deferred Lighting, 외곽선을 구현해보고 싶었지만 생각보다 코드가 어려웠던 점 때문에 구현은 힘들 것 같다.

-5주차-

1. 3D 게임 프로그래밍1 복습

3D게임 프로그래밍1인 기본적인 다이렉트X 12 구조를 알기 위하여 오늘(23.1.17)올라온 TU\_OCW 3D 게임 프로그래밍1 강의를 약 2주간에 걸쳐서 정주행 할 생각이다. (강의 당 2시간 총 33강의)

2023.1.17 - 1주차 1차시 3D 그래픽스 기초 (수학적 기초 개념 – 01)

2023.1.18 – 1주차 2차시 3D 그래픽스 기초 (수학적 기초 개념 - 02)

2023.1.18 – 1주차 3차시 3D 그래픽스 기초 (변환 파이프라인 – 01)

( 여기까지 WORD로 정리하면서 공부하였으나 실수로 파일이 모두 날아감 ㅠㅠ )

1. 3D 게임 프로그래밍1 복습

2023.1.21 - Direct3D 그래픽스 파이프라인 (part1) - 01

2023.1.21 - Direct3D 그래픽스 파이프라인 (part1) - 02

(여기까지 WORD로 정리하면서 공부하였으나 파일이 모두 날아감 ㅜㅜ)

2023.1.23 - Direct3D 그래픽스 파이프라인 (part1) - 03

2023.1.23 - Direct3D 그래픽스 파이프라인 (part1) – 04

 Direct3D 그래픽스 파이프라인 03, 04을 보며 정리한 내용이다.

2023.1.24 – Direct3D 그래픽스 파이프라인 (part2) - 01

1. 예제 코드 해석 완료

저번 주차에 가졌던 의문 중에 첫번째로 이것

텍스트이(가) 표시된 사진

자동 생성된 설명

인강을 보다가 알게 된 사실로 SV\_VertexID는 매 Vertex마다 1씩 늘려가며 번호를 매겨주는 시멘틱임을 알게 되었다. ( 위의 정리본에 적혀있음 )

텍스트이(가) 표시된 사진

자동 생성된 설명

이것도 알게 되었는데

1. 그 외의 것들

이번 주는 창업 현장 실습을 하는데 많은 시간을 들였기 때문에 많은 일을 하지 못했다.

그리고 2박3일간 여행을 가서 방학 기분을 냈다 ^-^