2020182031 이서연 졸업작품 6주차 진행

1. 코드 정리

우리의 코드의 현재 형태는 Render마다 Root Signature과 뷰 포트, Scissor Rect를 Set해주고 Shader의 상수 버퍼에 Camera, Lights, Materials등의 정보를 넘겨주고 있다.

이것은 Render를 위한 사전 준비 단계이며 Render과는 다른 부분으로 분리하기 위하여 OnPrepareRender라는 함수를 만들어 Render를 준비하는 단계로 분류해 주었다.

* 바꾸기 전의 Render함수

텍스트이(가) 표시된 사진

자동 생성된 설명

* 바뀐 후

텍스트이(가) 표시된 사진

자동 생성된 설명

또한 Root Parameter를 설정할 때 고정 상수 값 0 ~ 4를 넣어주고 있었는데 이를 의미적으로 가독성이 생기도록 “stdafx.h”에 다음과 같이 각 상수 값을 정의해 주었다.

#define ROOT\_PARAMETER\_PLAYER 0

#define ROOT\_PARAMETER\_CAMERA 1

#define ROOT\_PARAMETER\_OBJECT 2

#define ROOT\_PARAMETER\_MATERIAL 3

#define ROOT\_PARAMETER\_LIGHT 4

또한 Framework 부분에서 Resource Barrier를 설정 해 줄 때, 모든 인자 값을 넣어주게 되어 코드가 길게 늘어나 있는데 이 또한 함수로 “stdafx.cpp”에 함수를 만들어 주어 관리 하였다.

* 함수화 전

텍스트이(가) 표시된 사진

자동 생성된 설명

* 함수화 후

텍스트이(가) 표시된 사진

자동 생성된 설명

이때 Resource Transition의 두번째 인자는 상태를 바꾸어 줄 리소스가 들어간다.

그리고 세번째 인자는 바꾸기 전의 상태, 네번째 인자는 바꾸고 싶은 상태를 입력하면 된다.

1. 수많은 궁금증

우리 프로그램에 Deferred lighting을 적용하려는 시도를 하면서 내가 아직 예제 프로그램을 이해하지 못했음을 아주 깊이 깨 달았다.

여러 궁금증에 대해 적어 두고 차례차례 답을 풀어나갈 생각이다.

1. 텍스트이(가) 표시된 사진

   자동 생성된 설명이 Laplacian 필터에서 return문을 하나로 합치면 왜 아무것도 안 그려질까?

아래 return을 보면 Edge가 True일 때 하얀색을 return하고 그렇지 않을 땐 투명색을 return해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그렇다는 건 코드를 이렇게 적어도 같은 결과가 나와야 하는데 이렇게 하면 아무것도 return되지 않는다.

아주 단순한 실수였다.

그것은 바로 Line Color에 (0,1,0)을 넣어 줬는데 (0,1,0)은 float3형태가 아니기 때문에 마지막 값인 0만 float3에 모두 들어가고 있던 것이다.

초기화를 해줄 때 float3 Line color = float3(0,1,0); 이라고 해줘야 했다.

1. Pixel Shader들 중 하나인 이것은 float4가 아닌데 색상을 어떻게 아는 것일까?

텍스트이(가) 표시된 사진

자동 생성된 설명

PS\_MULTIPLE\_RENDER\_TARGETS\_OUTPUT의 TARGET0을 보면 Back Buffer로 설정 되어있다.

그리고 f4Scene이라는 변수로 저장되는데 코드를 보면 f4Scene와 f4Color에 f4Illumination과 f4Texture을 곱한 값 (색상)을 넣어주는 것을 알 수 있다.

이때 f4Scene은 Back Buffer로 연결되므로 이 float4값이 그대로 넘어가 Pixel의 색상으로 적용되는 것이고 Color정보도 따로 저장하는 것 이였다.

1. 이 코드는 Deferred Lighting이 되어있는 것인가?

코드를 보면 위의 Pixel Shader에서 빛계산을 해주는데 아무리 보아도 위의 코드는 GBuffer정보가 아닌 Forward Render같다.

저기서 빛정보를 계산해서 넘겨주는데 그렇다면 이 코드는 Deferred Lighting이 아닌 듯하다.

실제로 이 코드는 Forward Lighting이 맞고 Deferred Lighting이 절대 아니다.

위에 코드를 보면 Texture정보가 아닌 Vertex Shader에서 넘어오는 Primitive를 가지고 Lighting을 처리하는 것을 알 수 있다.

이는 결국 오브젝트 Pixel\*Light 계산이 되므로 굉장히 많은 양의 계산을 필요로 하며 저 Illumination Texture을 가지고 2PASS에서 계산을 하더라도 이미 저기서 계산한 것부터 이 코드는 비효율적인 것을 알 수 있다.

즉, 의미 없는 일을 하는 코드인 것인데 교수님이 이것을 알아내라고 일부러 이렇게 만드신 것 같다.

1. 

로 정보를 넣어줄 때 넣어주는 방식은 알겠지만 저 정보 순서를 어떻게 넣어주는 것인가?

이건 2,3번을 알아낼 때 같이 알아내야 했다.

PS\_MULTIPLE\_RENDER\_TARGETS\_OUTPUT의 정보를 그대로 넣어주는 듯하지만 마지막 인자인 Depth는 PS\_MULTIPLE\_RENDER\_TARGETS\_OUTPUT에 없음을 알 수 있다.

실제로 예전 Direct3D는 사용할 수 있는 TARGET이 4개로 한정되어 있었다고 한다.

이때 Depth값은 정보를 넘겨주지 않더라도 Depth/Stencil Buffer에 저장되어 있으므로 TARGET으로 넘겨주지 않고 Depth/Stencil Buffer를 넘겨주어 TARGET을 아끼는 방법을 사용해온다고 한다.

코드내에서 Depth/Stencil Buffer를 옮겨주는 부분을 찾기 너무 어려웠는데 그 이유는

텍스트이(가) 표시된 사진

자동 생성된 설명

Create Resource And Views라는 이 함수가 두개가 있는데 이건 6개의 Render Target Resource를 넘기는 함수이고 또다른 하나는 Depth/Stencil값을 넘겨주는 함수이다.

그러나 이 함수엔 넘기는 기능이 아니라 Render Target Views 서술자 핸들을 만들어주는 기능도 있었다.

즉, 같은 함수지만 하나의 함수에 추가적인 기능이 있었던 것이고, 이를 찾기가 너무 어려워서 하루 종일 걸렸다는 슬픈 이야기..ㅠㅠ

그래도 여기서 7개의 자리를 사용하겠다고 미리 알려준 후 6개의 Render Target을 넣어주고 이후에 호출되는 함수에서 Depth/Stencil값을 마지막 7번째 자리에 넣어준 것이다.

1. Deferred Lighting에 대한 추가 정보

이제 교수님의 예제 코드를 모두 이해하여 Illumination정보가 아닌 Position정보를 넘겨, 2PASS에서 빛계산을 해주는 코드로 바꿨고 Object ID를 사용하여 외곽선 구현 및 색상, 두께도 설정이 가능하다.

그러나 난 여기 까지가 Deferred Lighting이라고 정의하는 줄 알고 다 구현했다고 생각했는데 알고 보니 Deferred Lighting은 이번주 내에 끝낼 수 있을 만큼 만만한 녀석이 아니었다.

빛을 적용할 때도 특정한 방법이 있었고 이 방법을 사용하기 위해 2~3PASS에 걸쳐서 빛처리를 한다는 것 이였다.

빛 처리 전에 빛의 영역을 먼저 2D로 그려준다고 한다. (Spot Light = 삼각형, Point Light = 원)

그리고 빛은 Back Culling이 아닌 Front Culling을 하여 각 Pixel의 Depth값과 비교하여 빛이 닿는 범위를 계산해주어 Diffuse, Specular값을 정보로 넘긴다고 한다.

즉 원래 사용하던 Lighting함수의 빛계산은 하지 않게 된다.

이 과정이 굉장히 생소하고 감이 안 와서 어려울 것 같지만 이렇게 하면 Pixel + Light로 계산양이 훨씬 감소한다고 하여 꼭 구현해야 할 것 같고, 더 공부를 해야 할 것 같다.

1. 그 외

2/2 (목)에 창업 현장 실습 중간 점검을 받으러 학교에 갔고 많은 피드백을 받았다.

1. 3D 게임 프로그래밍1 복습

2023.1.31 – Direct3D Device (part2) – 01

2023.2.1 – Direct3D Device (part2) - 02

2023.2.3 - Direct3D Device (part2) - 03

- 여기 까지의 기록 