2020182031 이서연 졸업작품 7주차 진행

1. Deferred Lighting 구현

저번주에 막히는 부분이었던 Texture불러오는 부분을 고쳤다.

텍스트이(가) 표시된 사진

자동 생성된 설명

저번주에 Sampler를 만들어 주었다고는 했으나, 만들어 주기만 하고 Sampler를 Shader코드에 연결해주지 않아서 Pixel Shader가 match되지 않아 오류가 발생함에 따라 파이프라인 상태가 만들어지지 않아서 아무것도 그려지지 않은 것이다.

그래서 NumStaticSamplers를 1로 설정해주고 Sampler에 만든 Sampler주소값을 넘겨주었다.

그랬더니 Texture의 색상이긴 하지만 형태가 나타나지 않고 단색으로만 색칠이 되길래 Vertex에서 UV를 확인해 보았는데 코드를 작성하다가 실수로 UV를 2개 생성해주고 첫번째 UV에 값을 넣어주고 두번째 UV를 사용하고 있는 것을 알아냈다.

화살이(가) 표시된 사진

자동 생성된 설명이것이 Texture만 출력하는 모습이다.

또한 Object Shader를 만들어 주기 전에 Render Target을 1개만 만들어서 3개의 정보 중 처음의 1개의 정보만 넘기고 있었는데 모두를 넘겨주기 위하여 3개로 하였다.

이제 GBuffer에 정보가 잘 담기는 것은 모두 확인이 되었다.

이 GBuffer의 내용을 Texture Resource형태로 만들어서 그것으로 디버깅 화면을 만들고, 2PASS Rendering을 하는 것이 나의 이번주 목표이다.

이제 이 3개의 데이터를 합쳐서 테두리와 조명을 적용하기 위해 Screen Shader제작 한 것을 불렀는데

텍스트, 모니터, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

다음과 같이 오류가 나는데 이것은 저기서 넣어준 Render Target View Descriptor Handle이 Descriptor Heap의 위치를 가리키고 있지 않는다는 내용이다.

이를 해결하기 위하여 코드를 하나하나 따라가며 이해하기로 하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

먼저 Constant Buffer View와 Shader Resource View의 Descriptor Heap을 만드는 코드이다.

Descriptor의 개수는 Constant Buffer View와 Shader Resource View의 합이며 현재 CBV는 0개, SRV는 Multiple Render Target의 수인 3개+Depth Stencil Buffer의 1개를 해서 총 4개이다.

CBV SRV Descriptor Heap을 생성해 준 후 CBV Descriptor을 Heap의 맨 앞에 넣어주기 위하여 CBV Descriptor 시작 Handle을 Descriptor Heap의 맨 처음으로 넣어준다. (CPU와 GPU 모두)

그리고 SRV Descriptor 시작 Handle을 CBV Descriptor 시작 Handle에서 CBV개수만큼의 공백을 더해 준 후의 위치로 계산해 준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

다음은 Shader Resource View를 만드는 코드이다.

일단 매개변수 중 Texture은 미리 만들어준 Multiple Render Target개수 만큼의 빈 Texture이고, 루트 파라미터 인덱스는 MRT Texture Array인 6번으로 설정되어 있다.

먼저 Texture의 개수와 타입을 가져온다.

그 후 Texture의 개수만큼 Shader Resource View Descriptor을 만들어서 그곳에 Shader Resource View를 넣어준 후 CPU Descriptor Handle을 지정 해주고 CPU값을 하나 이동 뒤 GPU Descriptor Handle도 지정 해주고 하나 이동해준다.

그리고 루트 파라미터를 6으로 Set해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

그렇게 543번까지 완료한 결과이다.

그리고 Render Target View Descriptor을 하나 생성해 준 후 MRT의 개수만큼 Render Target View Descriptor Handle을 만들어 준다.

그리고 Resource, Render Target View Descriptor, Handle을 조합하여 Render Target View를 만들어 준다.

그리고 이 때 오류는 Create Render Target View함수에서 마지막 인자인 CPU Descriptor Handle이 Descriptor Heap의 위치를 참조하지 않고 있다는 뜻이다.

난 이 오류를 이해할 수 없었다.



왜냐하면 이것이 인자로 들어가는 Descriptor Handle인데 멀쩡히 Heap의 시작지점을 가져오고 있기 때문이다.

텍스트, 모니터, 스크린샷, 화면이(가) 표시된 사진

자동 생성된 설명

이유를 알아냈는데 그것은 Descriptor Heap을 만들어주는 코드에서 Swap Chain Buffer수 만큼만 Descriptor을 만들어주는 것 이였다.

추가적으로 MRT의 Descriptor를 만들어 주지 않았기 때문에 이것을 사용하려고 할 때 빈 곳을 가리키고 있던 것 이였다.

그러자 새로운 오류가 발생했다.

텍스트이(가) 표시된 사진

자동 생성된 설명

여기 아래부분 리소스만큼 Descriptor Handle을 만들고 Clear Render Target View를 할 때 오류가 난다.

The clear values do not match those passed to resource creation. The clear operation is typically slower as a result; but will still clear to the desired value.

지우기 값이 리소스 생성에 전달된 값과 다르다는 뜻이었다.

해결 했는데 이유는 모르겠지만 저 DirectX::Color가 Texture Resource로 Render Target View를 초기화 할 때 사용되면 이런 오류가 뜨는 것 같다.

고치고 나니 또다른 오류가 발생했다.

For the resource format D24\_UNORM\_S8\_UINT, when making a D3D view, the format name for the view can't be R32\_FLOAT. See documentation for the set of valid view format names for this resource format, determining which how the resource (or part of it) will appear to shader.

View를 만들 때 D24\_UNORM\_S8\_UINT는 R32\_FLOAT로 읽을 수 없다는 뜻이다.

내가 Depth Stencil View를 만들 때 DXGI\_FORMAT\_D24\_UNORM\_S8\_UINT형식을 사용한다.

이것은 Depth를 24만큼, Stencil을 8만큼 UINT로 사용하겠단 뜻이다.

하지만 Depth Stencil Shader Resource를 만들 때 DXGI\_FORMAT\_R32\_FLOAT를 사용하였다.

이것은 RGB(색상)으로 R부분 32만큼 읽겠다는 뜻이다.

내 Depth Stencil은 24와 8로 나누어져 있는데 난 32만큼 읽겠다고 한 것이므로 오류가 나는 것이다.

그래서 DXGI\_FORMAT\_R24\_UNORM\_X8\_TYPELESS로 바꿔줬다.

앞에 24만 R로 읽고 뒤8개는 읽지 않겠다는 뜻인데, 이게 무슨 뜻이냐 면 Depth Stencil이라는 똑 같은 리소스를 Depth Stencil의 용도로 쓸 때는 D24 S8로, Shader Resource View의 용도로 쓸 때는 R24로 읽어서 같은 리소스를 다르게 해석하겠다는 뜻이다.

이제 Screen Shader는 잘 나온다.

Multiple Render Target을 만들 때 혹시 Texture쪽에 오류가 생길 까봐 Pixel Shader의 출력 값을 파란색으로 했더니 파란색이 잘 나와서 G Buffer를 불러보니 뜨지 않아서 오류 출력내용을 보니 Pixel Shader와 Root Signature가 맞지 않는다고 한다.

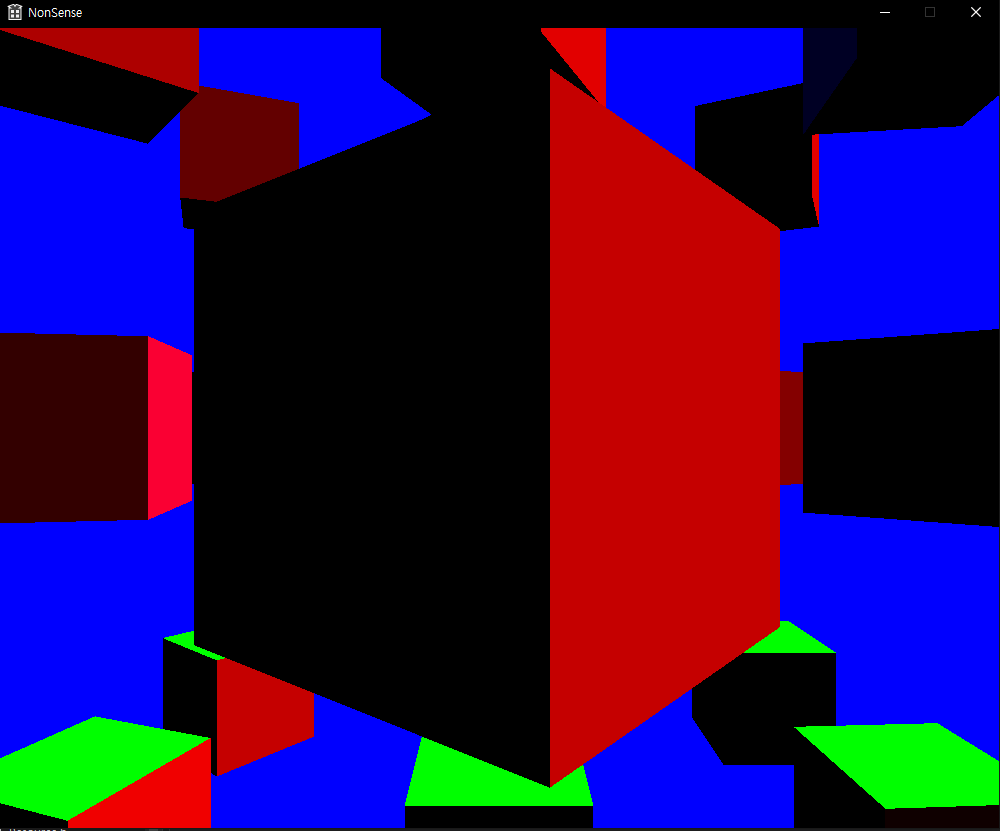
그 이유는 간단했는데 Root Signature에서 MRT의 Num Descriptors의 개수가 딱 MRT개수 만큼만 있었기 때문이다.

우리는 Depth Stencil Buffer까지 옮겨주어야 하기 때문에 거기에 +1을 한 숫자를 넣어주어야 했다.

텍스트이(가) 표시된 사진

자동 생성된 설명이것이 수정 후 Texture만 띄운 것이다.

Screen Shader의 Clear값을 파란색으로 했기 때문에 이것은 Screen Shader가 그려주고 있는 것임을 알 수 있다.

Depth Stencil의 Depth를 불러온 결과 Normal을 불러온 결과

(원래 이렇게 흐린 건가?)

여기서 두가지 문제점이 발생한다.

첫번째는 MRT의 첫번째 Target인 Position값은 전달이 되지 않는다는 점이다.

첫번째 Target이 Swap Chain Back Buffer가 되어 그려지는 것은 알지만 난 이것을 MRT로 전달되기를 바라므로 그 방법을 찾아내야 한다.

원래는 MRT Texture의 0에 Position, 1에 Normal, 2에 Texture, 3에 Depth Stencil이 들어가야 하는데

현재는 0에 Normal, 1에 Texture로 하나씩 앞으로 밀려 있고 2는 비어 있으며 3에 Depth Stencil이 들어간다.

두번째는 Edge다.

교수님의 코드에서는 ID와 Z Depth를 저장하는 float2를 만들어 주었지만 난 공간 활용을 위해 Normal값의 w자리에 Object ID를 넣어주었다.

이 때 Object ID만 뽑아오는 법을 알아내야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

이것이 현재 Edge부분인데 저기서 Object ID를 뽑아오는 방식은 교수님의 방식이다.

Texture의 0번은 Object ID데이터가 존재하는 Texture를 의미한다.

그럼 그 뒤에 Int2(position.xy)부분은 뭘 까? 이 부분을 해석해야 내 ID데이터도 뽑아올 수 있을 듯 하다.

1. 그 외의 것들’

창업 현장 실습 (군 전역 계산기)에서 캘린더 부분 구현을 담당하고 있다.

캘린더는 진작 제작 완료하였고, 날짜별로 선택을 하면 메모를 적을 수 있도록 하여 내가 담당한 부분을 모두 구현 완료하였다.

3D게임프로그래밍 강의를 필요한 부분 모두 시청 완료했다!