2020182031 이서연 졸업작품 2주차 진행

소스코드에 Object와 조명을 띄워 놓아야 외곽선이든 Toon Shading이든 구현이 가능 할 것 같아서 계획 순서를 조금 바꾸게 되었다.

Toon Shading과 외곽선에 대한 자료조사는 1주차에 거의 끝낸 상태이므로 코드를 구현하기 전까지 그림자 공부와 2D 리소스 제작 등 당장 할 수 있는 것을 하기로 하였다.

1. 캐릭터 Asset 얼굴 Retouch

 하늘, 물이(가) 표시된 사진

자동 생성된 설명

↑ 원본 전사 캐릭터

벡터그래픽이(가) 표시된 사진

자동 생성된 설명 하늘, 물이(가) 표시된 사진

자동 생성된 설명

↑ 수정된 전사 캐릭터

여러개이(가) 표시된 사진

자동 생성된 설명 하늘, 장난감, 인형이(가) 표시된 사진

자동 생성된 설명

↑ 원본 마법사 캐릭터

텍스트, 벡터그래픽이(가) 표시된 사진

자동 생성된 설명 하늘, 장난감, 인형이(가) 표시된 사진

자동 생성된 설명

↑ 수정된 마법사 캐릭터

캐릭터들의 얼굴을 게임 분위기에 더 잘 맞는 귀여운 얼굴로 Retouching 해 주었다.

1. 소스코드에 육면체 띄우기

이용희 교수님의 따라하기 과제를 하면서 육면체를 띄우고 플레이어와 카메라도 만든 후 보이는 객체만 그려지도록 절두 체 Culling을 구현해 놓았다.

조명을 만들려고 하는데 따라하기 도중 알 수 없는 오류가 발생하여 해결하다가 어디서부터 꼬인 것 인지 알 수 없어서 플레이어와 카메라부터 다시 만들기로 하였다.

그러다가 재우 님이 Root parameter에 값 넘겨주는데 문제가 있음을 발견하고 고쳐 주셨다.

결과적으로 많은 육면체들과 몇 개의 조명이 나오는 화면이 출력된다.

옅은, 켜진이(가) 표시된 사진

자동 생성된 설명 

↑ 현재 출력되는 화면+ ↑ 테스트를 위해 바꾼 화면

왼쪽 그림과 같은 화면이 출력되는 코드였으나 Toon Shading을 더 깔끔하게 확인하기 위하여 객체 수를 줄이고 색상을 하얀색으로 바꾸어 주어서 오른쪽 그림과 같은 화면이 되었다.

1. Toon Shading

Lighting의 Diffuse Factor을 round함수 (반올림)를 이용하여 0.0~1.0이 아니라 내가 원하는 비율로 나누어 주기 위하여 을 정의해주었다. (현재 Toon Shading 값 5)

round함수는 정수 단위로 반올림해주므로 실수로 바꿔주기 위하여

round (Diffuse\*TOON\_SHADING)/TOON\_SHADING 식을 사용하여 값을 반올림해 줄 생각이다.

1/10 조명이 잘 나오고 있나 확인해보기 위하여 Spot Light 하나만 설치해 두고 모든 빛 계산을 무시하고 1,0,0,1을 return 해주고 material의 색상도 추가해주지 않았는데 그럼 빛의 범위안의 모든 픽셀이 1,0,0,1로 나와야 한다고 생각하는데 아래 그림처럼 나오는 것이다.

텍스트, 봉투, 문구, 명함이(가) 표시된 사진

자동 생성된 설명

이렇게 나오는 이유만 알아낸다면 쉽게 구현 할 수 있을 것 같다.

이유를 알아냈다!

바로 Lighting 처리를 Vertex Shader에서 하고 있었던 것이 문제였다.

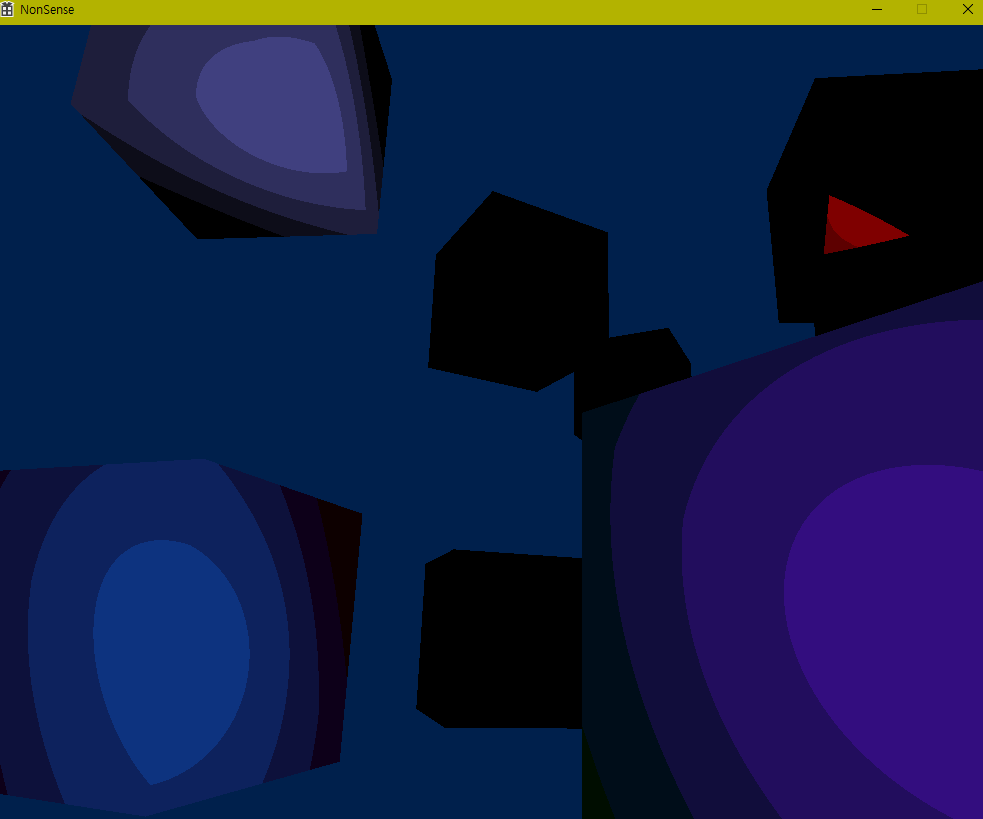
Vertex에선 내가 원하던 처리가 되었지만 Pixel 단위로 계산이 되지 않기 때문에 정점을 기준으로 픽셀이 정해지고 있어서 저렇게 부드럽게 그려지는 것 이였다.

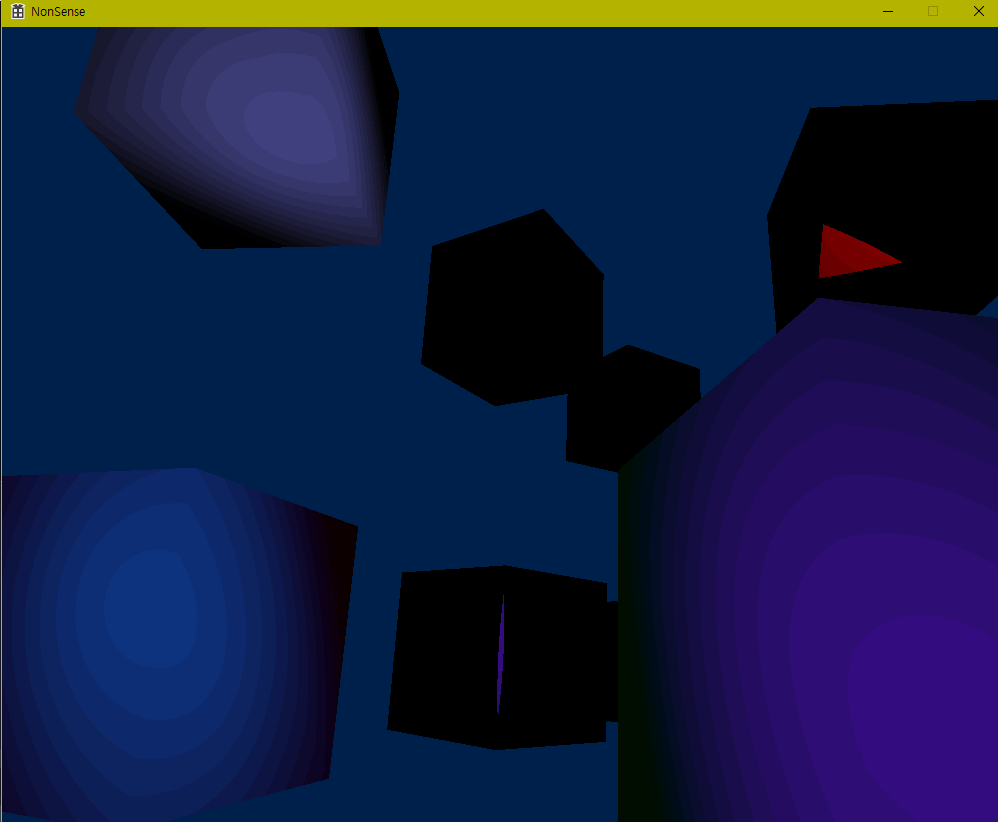
그래서 정점 조명을 사용하지 않고 Pixel Shader에서 Lighting 함수를 불러주었더니 오류가 났다.

그 이유는 가 활성화 되어있어서 Pixel Shader가 Root Signature에 접근하니 오류가 나는 것 이였다.

그래서 이걸 꺼주고 식을 계획대로 바꾸고 실행해주니 정상적으로 나오기 시작했다.

그런 김에 Object들 색도 원래대로 돌려놔 주었고 결과는 아래와 같다.

 텍스트, 옅은, 어두운이(가) 표시된 사진

자동 생성된 설명 

↑ TOON\_SHADING 값이 3일 때 ↑ TOON\_SHADING 값이 5일 때 ↑ TOON\_SHADING 값이 10일 때

하지만 육면체로는 잘 되고 있는지 확인이 어려우므로 구체를 띄워 보기로 하였다.

1. Deferred Rendering 공부

Toon Rendering을 구현하다 보니까 Vertex Shader가 아닌 Pixel Shader에서 조명을 처리하게 되었다. Pixel Shader에서 조명을 처리하면 Pixel단위로 계산이 되기 때문에 정교한 조명처리가 가능해지지만 그만큼 계산양이 매우 많아진다. (Vertex Shader는 정점x조명, Pixel Shader는 픽셀x조명)

사실 Non-Sense에서 조명을 많이 사용할지는 아직 모르겠지만 Shader Programming을 담당한 이상 이런 계산양을 줄이기 위한 Deferred Rendering을 꼭 넣어야 할 것 같다고 생각했다.

지금처럼 조명 계산과 렌더링을 함께하는 것을 Forward Rendering이라고 하며 이런 경우 카메라에 보이지 않는 부분들도 계산하게 되어 상당히 비효율 적이다.

Deferred Rendering이란 간단하게 말해서 3D장면을 2D로 바꾼 뒤 그 정보들을 이용하여 Lighting처리를 하는 것인데 이러면 화면에 보이는 픽셀만 처리하게 되어 효율적인 조명처리가 가능해진다.

다만 문제가 있다면 투명한 물체를 처리할 수 없다는 점이 있는데 그 이유는, 2D Texture에 색상정보를 저장 할 때 투명한 물체를 처리하려면 불투명 물체 A의 색상과 반투명 물체 B의 색상이 모두 2D Texture에 저장되어 있어야 하는데 둘 다 저장을 할 수 없기 때문이라고 한다.

그 해결책으로 불투명 물체만 Deferred Rendering을 하고 그 후에 Forward Rendering으로 투명 물체를 그려주는 방법이 있다.

또한 Deferred Rendering의 이런 단점을 보완한 Forward + Rendering이란것도 있었다.

이건 화면을 타일로 나누어 타일마다의 빛의 양을 계산하고 적용해주는 느낌이 였는데 우리는 계획 상 투명한 물체가 없을 것 같아 Deferred Rendering을 공부하여 적용하기로 결심했다.

1. Texture 공부

Deferred Rendering이나 외곽선을 그리려면 먼저 Texture에 대한 이해가 필수이다.

Deferred Rendering은 색상, 깊이, 법선 등의 데이터를 각각 다른 버퍼에 저장하여 사용하고 외곽선은 스텐실 값을 버퍼에 저장해야 하기 때문에 그것을 Texture로 저장하는 방법을 알아내려면 추가적으로 공부를 해야 한다.

그래서 3D 게임 프로그래밍2 Multiple Render Target 예제를 보며 Texture와 외곽선에 대해 익혀야 한다.