2020182031 이서연 졸업작품 20주차 진행

1. 중간 발표를 보고 느낀 점

사실 다른 팀들 것을 볼 기회는 없었지만 우연히 잠깐 보았던 두 작품이 있었는데 우리의 작품이 외관상으로 너무 이쁘게 나오고 있고 진행도도 빠르게 보이고 있어서 정말 잘 하고 있는 줄 알았다.

발표 전 까지만 해도 엄청 잘한 편일 것이라는 자신이 있었지만 막상 중간발표를 하고 나니 우리의 것은 보이기에는 훌륭하지만 사실 기술적으로 구현되어 있는 것은 적다고 느껴졌다.

벽이 느껴질 정도로 잘한 팀도 있었다. (물론 서버나 게임성 등을 고려하면 누가 더 잘했다고 판단하긴 어렵지만 보이기에는 그렇다.)

체육관을 목표로 열심히 달리고 있지만 요즘 번아웃이 오는 건지 쉬고 싶다는 느낌을 강하게 받는다...

앞으로 3달이 남았는데 그래도 남은 기간 후회없이 열심히 해보려고 한다.

1. Cursor 변경

게임 내에서 마우스 커서가 일반 마우스의 ARROW형태면 게임이라는 느낌이 잘 들지 않는다.

그래서 우리 게임과 분위기가 어울리는 무료 Cursor Asset을 찾아 적용해주기로 하였다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Framework의 Process Input함수인데, 여기서 Cursor Expose는 마우스가 중앙 고정 (플레이어 조종 시 마우스 설정)일 때 true가 된다.

이때는 Set Cursor를 NULL로 하여 마우스 커서가 보이지 않도록 해야 자연스럽다.

그리고 아래 else, 즉 마우스가 자유로운 상태가 되면 Load Cursor함수를 사용하여 미리 리소스에 저장해둔 Cursor이미지를 띄워주고, UI클릭 관련 함수를 이 아래에 넣어 줌으로써, 마우스가 자유로운 상태일 때만 UI를 클릭할 수 있도록 해준다. (플레이어 조종 시에는 UI클릭할 일이 없음)

식물, 텍스트, 나무, 스크린샷이(가) 표시된 사진

자동 생성된 설명

결과적으로 이렇게 반투명하고 예쁜 Cursor가 만들어졌고 추후에 Particle효과도 넣어볼까 싶다.

1. Depth Buffer Shadow 공부 및 구현

Depth Buffer Shadow Mapping은 전에도 언급한적이 있지만 사실 구현방법은 이론적으로는 쉬운데, 카메라 위치에서의 Depth Buffer와 빛의 위치로 카메라를 옮긴 후 그린 Depth Buffer(Shadow Map)을 비교하여 해당 픽셀이 그림자가 진 부분인지 아닌지 알아내는 기법이다.

이론을 알고 있음에도 구현을 하지 못하고 미루고 있던 이유는 이것을 코드로 작성하기가 어렵다고 생각했기 때문이고, 중간 발표 때 많은 팀이 이미 그림자를 구현한 것을 보고 경각심을 느껴 바로 나도 시도해보기로 하였다.

먼저 DirectX 12를 이용한 3D게임 프로그래밍 입문책의 예제 파일들을 보며 공부할 생각이다.

스크린샷, 하늘, 3D 모델링, PC 게임이(가) 표시된 사진

자동 생성된 설명이런 결과가 나오는 예제 코드이고, 이것을 참고하여 만들 것이다.

참고하기에 어려운 사항이 있다면 코드가 우리의 형태와 굉장히 다르게 생겨서 해석하기가 어렵단 것이다.

이 프로그램은 크게 update와 draw를 반복하며 나타나는데, 일단 update부터 보려고 한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

내가 봐야 할 곳은 이곳, 먼저 위의 부분은 Directional Light를 회전시켜주는 코드로 보이고 그 이후의 Material과 Object관련 함수는 내가 이미 구현되어 있는 부분이라서 따로 보지 않으려고 한다.

다음의 Update Shadow Transform함수의 내용을 살펴보겠다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

위의 주석을 보면 하나의 Main Light만을 그림자에 사용한다고 한다. (Directional Light)

그리고 빛의 방향, 위치, ???, ???, ???를 계산해준 뒤 위치는 저장해 둔다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

??? 왜 Bounding Sphere를 계산해서 Projection을 만들지?

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그후 스크린 좌표계를 -1 ~ 1이 아니라 Texture 좌표계인 0 ~ 1로 바꾸어서 View, Projection, ???를 저장한다.

이렇게 Shadow Transform의 Update를 마치면, Main Pass Constant Buffer를 Update해준다.

코드가 길어 올리진 못하지만, Shadow Pass가 아닌 Main Pass에서 사용할 3개의 빛의 정보와 시간, 카메라 정보들을 Constant Buffer로 넘겨준다.

마지막으로 Shadow Pass에서 사용할 Constant Buffer를 Update해주는데 이 때는 Camera의 정보만 바뀌므로 아까 계산한 View와 Projection을 이용하여 정보를 넣어준다. (Shadow Transform은 어디에??)

1. Stream Output을 사용하지 않는 Particle

Particle을 만들어 보려고 하는데 내가 Particle을 사용하고 싶은 곳은 Mage의 공격이나, Map의 반딧불이 같은 효과를 위해서이다. (즉, Particle의 수가 변하지 않고 처음부터 많이 존재해도 된다는 뜻)

그래서 Stream Output을 사용하지 않고, Geometry Shader만을 추가하여 Particle을 만들어보려고 한다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

일단 Particle Shader를 제작해줬는데 가장 간단한 형태의 정보(위치, 속도, 생성시간, 수명)을 넣어주었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Particle의 Vertex Shader이다. 일단 그려지는지 확인만 하기 위하여 움직이지 않도록 설정해 주었고 나중에 여기에 Particle이 움직이는 코드를 넣어줄 것이다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Geometry Shader에서는 미리 사각형과 UV의 Offset을 만들어 두고 Input의 Size로 크기를 키워주고, Camera의 Inverse View를 곱해주는데 이것이 카메라의 반대 방향으로 Vertex를 회전시켜, Billboard의 효과를 내주는 것이다.

Pixel Shader에서는 잘 나오는지 확인만 하기 위해 Texture가 아닌 빨간색을 출력한다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Particle Mesh를 만들기 위한 Vertex로, 필요한 정보들을 Vertex에 담아준다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

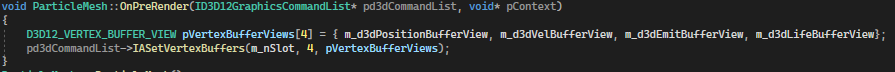
자동 생성된 설명

Particle Mesh를 만들어주는 부분인데, Position, Velocity, Emit Time, Life Time을 넣어주고 따로 Upload Buffer와 View를 생성해 준 뒤, 각각 넣어준다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이와 같은 Random함수를 만들어 적용해 주었다.



그리고 IA Set Vertex Buffers함수를 통해 정보들을 넘겨주면 끝이다.

이제 Shader와 Mesh가 만들어졌으니 Object를 만들 차례이다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Alpha값을 사용하므로 BLEND\_OBJECT로 넣어두며 Particle Texture는 Root Parameter의 20번째에 저장해 두었고, DDS파일도 만들어 두었다.

Particle Mesh의 마지막 인자는 한 Object에 몇 개의 Particle Point를 넘길 지 정해주는 것이고, 내가 현재 Particle을 만드는 것은 Mage의 기본공격을 더 화려하게 꾸며주기 위해서 이므로 50개정도로 일단 넣었다.