2020182031 이서연 졸업작품 29주차 진행

1. 대기실 Scene 구현

이제 Lobby Scene에서 방에 입장했을 때 해당 방에서 게임을 시작하기 전, Player들이 모여서 각자 역할군을 선택하고 Player를 기다리는 대기실 Scene을 만들어주려고 한다.

텍스트, 나무, 표지판, 여러개이(가) 표시된 사진

자동 생성된 설명기획했던 대기실 Scene 구상도

사실 대기실은 저렇게 Map의 특정 부분에 Camera를 고정해두고, Player들이 자유롭게 이동할 수 있도록 만들려고 했으나, 그것이 매우 어렵단 것을 알았다.

Camera를 고정해두고 움직이려면, 마우스를 제외하고 WASD로만 움직여야 하는데, 그에 따른 Player의 회전도 자동으로 해주고, 충돌처리도 따로 해주어야 하는데, 넉넉한 시간이 있을 지 모르겠다.

일단 최소한의 기능만이라도 만들어 두기 위해서 UI부터 만들어보려고 한다.

Stage Scene을 복사해와서, 모든 Character들을 지워주고, 빈 Map을 불러와주었다.

그리고, Lobby Scene에서 아무 방이나 입장하기 버튼을 누르면 Room Scene으로 이동되게 만들었다.

그리고 다음과 같이 Class선택 UI를 만들어 주었다.

또한, 방 나가기, 준비하기, 시작하기 버튼 UI도 제작해주었고, 이제 Scene에 배치해준다.

텍스트, 만화 영화, 나무, 일러스트레이션이(가) 표시된 사진

자동 생성된 설명

다음과 같이 배치를 해주었고, 배경을 그냥 2D UI로 할지 계획대로 할지는 일정에 따라 바뀔 예정이므로 일단 비워 두었다.

Room Scene은 서버와 많은 연관이 있는 부분이므로 클라이언트에서 딱히 많이 만들어 둘 수가 없다.

물론 Class를 선택하면 해당 Class로 게임을 시작하게 할 수는 있지만, 어차피 서버와 연결할 때 소켓구조로 크게 바뀌기 때문에 굳이 구현할 필요가 없다.

방 나가기를 누르면 Lobby Scene으로 돌아가는 정도로만 구현해 두었다.

1. 그림자 공부

그림자에 대한 공부와 시도는 수시로 계속 해왔고, 보고서에도 적은 적이 많다.

그리고 현재까지도 많은 다른 졸작팀에서 나에게 그림자를 구현했냐고 물어보았고, 나 또한 물어보았을 때 다들 그림자로 많이 애먹고 있다는 사실을 알게 되었다.

늘 드는 생각이지만 그림자의 이론이 어렵긴 하지만 크게 어려운 것은 아니고, 구현을 할 때 고려해야 할 점과 코드로 옮기는 데에 수많은 변수들이 존재하기 때문에 쉽지가 않다.

우리의 게임의 Light는 Directional Light하나뿐이기에 그림자를 구현하면 잘 나타날 것인데, 너무 어려워서 여러 예제를 보며, 비슷한 알고리즘으로 추가해보면서 구현과 공부를 동시에 하려고 한다.

그래서 난 Multi Render Target의 5번째 Buffer에 Light시점에서의 Depth Buffer를 출력해보려고 한다.

일단 Light 시점에서 바라보는 Depth Buffer를 저장 할 Shadow Map을 제작해 준다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

복잡해 보이지만 Shadow Map의 Class를 만들어준다.

Shadow Map은 Width, Height, Viewport, CPU SRV, GPU SRV, CPU DSV를 가지는 Resource인 셈이고 그렇기 때문에 Framework에서 Shadow Map을 위한 Create Shadow Map함수를 만들어 주어야 한다.

Framework에서 전체 Render의 순서의 맨 앞에 Shadow Map Render를 두는데, 이 때 Scene을 한 번 더 Render하므로 총 2번의 Render가 발생하고, 그래서 큰 비용이 들게 되지만 그림자는 어쩔 수 없다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

먼저 Framework에 아까 만든 Shadow Map Class를 하나 추가해 준 후, Create의 마지막에 Create Shadow Map함수를 만들어 넣어준다.

그리고 원래는 Depth Stencil Buffer가 1개였지만, 이제 Light시점, Camera시점 총 2개를 사용해야 하므로 2로 늘려준다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그리고 Shadow를 위한 SRV Descriptor Heap을 생성해, CPU, GPU Handle을 따로 만들어주고, Shadow Map의 크기는 2048x2048로 크게 만들어준다.

그리고 Shadow Map을 그려주기 위한 Shader를 따로 만들어주어야 하지만, Create Shadow Map함수에서 만들어 주었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그림자는 사실 Position의 값만 알면 되므로, Position의 정보만 필요로 하고, VS와 PS는 나중에 볼 예정이고 일단 이렇게 설정해주었다.

그리고 쭉 Pipeline Object Description을 작성해주는데, Rasterizer Depth Bias를 늘려주고, Blend State는 다른 Shader들과 같게 해주었다. (Alpha와 Blend Enable 모두 False)

Depth Stencil State도 기본 값으로 맞춰주었다.

이렇게 하면 일단 Shadow Map을 만들기는 하였으나, 이제 이것을 Debug 화면에 띄우는 것과, Camera의 위치를 Light로 바꾼 뒤, 한 번 더 Render해주는 작업을 해주어야 한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이 부분이 일단 내가 만들어 둔 Shadow Map Render 부분이다.

Viewport를 설정해주고, Shadow Map에 Depth값을 써서 Resource로 사용하기 위해 Resource Transition을 이용하여 Read에서 Write로 바꾸어 준다.

그 후 Depth Stencil View를 초기화 시켜주고, Set Render Target에서 RTS Descriptor의 개수와 Range등이 0, NULL, False로 되어있는 것을 볼 수 있는데, 이것은 실제 Render Target에는 그려주지 않고, 맨 뒤에 Shadow Map의 DSV에 Depth값만 그려주겠다는 뜻이다.

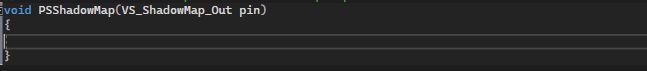
그 이유는 Shadow Map은 화면에 그려줄 필요 없이 Depth값만 필요하기 때문이다.

그렇게 설정해 준 후 모든 Object를 Render하고 Resource Transition을 이용해 다시 원래 상태로 바꾸어 주면 되는데 이 때 사용하는 Shader는 다음과 같다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

일단 Vertex Shader는 그냥 World, View, Projection을 곱해주어 배출하는데, 이 때 View는 Player시점의 Camera가 아니라 Light위치의 View이다.



당황스러운 Pixel Shader다.

텅 비어 있지만 위에서 설명했듯이 Pixel Shader는 의미가 없기에 텅 비게 두었다.

이렇게 Shadow Map을 아직 출력하진 못했지만 출력하기 위한 기초를 다져 두었고, Light 시점으로 Camera를 옮긴 뒤, 모든 Object의 Position값을 옮길 수 있다면 Shadow Map은 완성이 된다.

완성된 Shadow Map을 기존 Depth Buffer와 비교를 통해 그림자의 위치를 알아내고 그 위치를 여러 행렬변환을 해주어 그려주면 그림자가 완성된다.

다음주에 꼭 그림자를 완성할 것이다.

1. 여러 버그 수정
2. 갑자기 Far Monster가 무기를 던지면, 프로그램이 터지는 오류가 발생하였고, 해당 유형의 Monster를 만들어 줄 때 다른 Model을 사용한 것을 발견하고 고쳤더니 해결되었다.
3. Button UI의 Mouse On판정이 살짝 왼쪽 위로 치우쳐져 있어서 원하는 곳을 클릭하기가 조금 애매했는데, 적당한 값을 Mouse Cursor위치에 더해줘서, 오차범위를 없앴다.