3D게임프로그래밍1 과제2 설명문서

게임공학과

2022184015

김해님

1. 접근방식

첫 번째 과제로 작성했던 코드와 따라하기 14(인스턴싱)를 기반으로 코드를 작성하였다.

따라하기 예제에서는 오브젝트 쉐이더를 이용해 하나의 쉐이더가 오브젝트들을 들고 있는 방식으로 구현되어 있었지만, 개인적으로 오브젝트 쉐이더에 대해 이해하기 힘들었다. 특히 오브젝트 쉐이더는 그동안 짜오던 프레임워크와 잘 맞지 않는 형식이라고 생각하였다(항상 씬 안에 게임오브젝트의 std::vector를 가지고 오브젝트를 관리했기 때문에 씬 안에서 오브젝트의 동작 로직을 구현했다. 하지만 오브젝트 쉐이더의 경우 씬이 오브젝트 쉐이더를 가지고 있고, 오브젝트 쉐이더가 게임오브젝트를 들고 있기 때문에 씬에서 게임오브젝트들에 대한 직접적인 접근이 어렵다고 판단했다. 직접적인 접근이 가능하도록 구현할 수도 있겠지만 좋은 구조와 방식이 아니라고 생각했다. – 잘못된 이해일 수도 있지만, 이 이해 상태를 바탕으로 코드를 작성했다.). 폭발 애니메이션(저번 과제의 CExplosiveObject)을 제외하면 그렇게 많은 오브젝트가 한 번에 그려지진 않는다고 판단하여 각 오브젝트가 쉐이더를 들고 있는 형태로 구현할 예정이다(CDiffusedShader를 사용한다). 쉐이더는 BuildObject시에 씬 당 하나만 생성해서 게임오브젝트가 포인터로 들고 있도록 할 것이다.

위에 적은 것처럼 이전 과제에서 프레임 드랍의 문제가 되었던 폭발 애니메이션에 대해 인스턴싱 쉐이더를 이용해서 프레임 드랍을 줄일 예정이다. CGameobject 클래스를 상속받은 CExplosiveObject 클래스를 만들어서 이 오브젝트의 메쉬를 폭발 전에 그릴 메쉬로 설정하고, 폭발이 시작되면 원래 메쉬 대신 인스턴싱 쉐이더의 Render 함수를 호출하도록 할 것이다.

2. 변경점

첫 번째 과제의 프레임워크처럼 따라하기 14의 프레임워크를 변경하였다.

- GameFramework 클래스

GameFramework 클래스는 DirectX와 관련된 설정 및 씬 관리를 담당한다. 기존 따라하기 코드에서는 플레이어(m\_pPlayer)와 카메라 관리를 GameFramework 클래스에서 하도록 짜여 있는데, 여러 개의 씬을 만들어야 하기 때문에 플레이어와 카메라를 각 씬에서 관리하도록 이동하였다.

- CScene 클래스

CScene 클래스는 GameFramework 위에서 동작하는 클래스이다. CScene 클래스는 각 씬의 부모클래스이며, 이 클래스로부터 상속받아 TitleScene, MenuScene 등 각 씬을 생성한다.

씬 안의 게임오브젝트들은 std::vector<CGameObject\*> 컨테이너로 관리하도록 구현했다. 하나하나 Render와 Update를 호출하는 것보다 반복문 돌려서 한 번에 처리하는 게 편하다고 생각했기 때문이다.

- .obj파일으로부터 메쉬 생성

기존에 사용했던 .obj파일을 그대로 이용할 예정이다. 저번에 만들었던 파일 로드 함수와 바운딩박스 생성 함수를 하나로 합쳐서 ObjMesh 클래스의 생성자에서 동작하도록 수정하였다(코드 가독성은 떨어지지만 이 함수를 다른 클래스에서 부를 일이 없어 이렇게 수정했다).

- 추가한 라이브러리

C++의 여러 라이브러리를 추가하였다.

3. 구현

1) 타이틀 씬

첫 번째 과제에서 구현했던 것처럼,