**3D 게임2 과제01 보고서**

게임공학과 3학년

2022182018 서가은

상징, 원, 로고, 폰트이(가) 표시된 사진

자동 생성된 설명

**목차**

1. 과제에 대한 목표
2. 가정
3. 실행결과
4. 조작법
5. 구현 내용
6. 추가적으로 구현한 내용

**과제에 대한 목표**

시작화면을 2D 텍스쳐로 만들고, 메뉴는 마우스 휠메시지를 받고 위 아래로 움직일 수 있는 2D 텍스쳐로 만드는 것을 목표로 하였다. 시작화면에서 인게임화면으로 씬 전환하는 것을 만들고, 플레이어가 적 기체와 충돌하기 전에 넉백하는 충돌처리를 추가하고, 플레이어가 총을 쏠 수 있도록 하며, 총에 맞은 적 기체는 폭발하는 이펙트를 넣어주는 것을 목표로 하였다. 또한, 터레인을 구현하고 빌보드 텍스쳐가 시간에 따라 양 옆으로 움직여서 바람에 흩날리는 듯한 효과를 주는 것을 목표로 하였다.

**가정**

LabProject08-0-1을 참고하여 이 프로젝트에서 시작하면 효율적으로 만들 수 있을 것 같았다. LabProject08-0-4를 참고하여 2D텍스쳐를 화면에 띄우고, LabProject08-7을 참고하여 터레인과 빌보드를 만드는 것이 좋을 것 같았다. 또한, 1학기때 만들었던 과제를 참고하여 총알을 발사하는 코드를 추가하면 쉽게 구현할 수 있을 것 같았다.

**실행결과**

비행기, 하늘, 교통, 야외이(가) 표시된 사진

자동 생성된 설명텍스트, 하늘, 구름, 산이(가) 표시된 사진

자동 생성된 설명

ㄴ 시작화면 / 메뉴 스크롤 전

텍스트, 스크린샷, 나무, 잔디이(가) 표시된 사진

자동 생성된 설명스크린샷, 교통, 항공기, 야외이(가) 표시된 사진

자동 생성된 설명

ㄴ 메뉴 스크롤 후 / 인게임 화면

식물, 야외, 스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷, PC 게임, 디지털 합성, 항공기이(가) 표시된 사진

자동 생성된 설명

ㄴ 총알발사 / 적 기체 피격시 폭발 이펙트

**조작법**

방향키 : 플레이어 이동

PgUp : 플레이어 상승

PgDn : 플레이어 하강

Tab : 메뉴 토글키

S : 게임 시작

Esc : 게임 종료

Ctrl : 총알 발사

**구현 내용**

<터레인>

class CTerrainShader : public CStandardShader - Shader.h

CHeightMapTerrain \* m\_pTerrain = NULL; - Scene.h

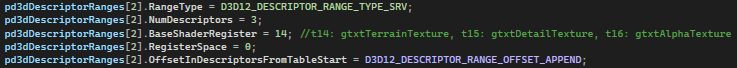
class CHeightMapTerrain : public CGameObject- Object.h

class CHeightMapImage : public CRowFormatImage / class CHeightMapGridMesh : public CMesh - Mesh.h



샘플러스테이트는 기존에 사용했던 를 사용해주었다.

  - Shaders.hlsl

Scene.cpp의 CreateGraphicsRootSignature에서 다음과 같이

디스크립터 랭지를 설정해주고, 텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

루트 파라미터를 설정해주었다.

Scene.cpp의 CScene::BuildObjects에서 m\_pTerrain을 만들어주고, 터레인이 게임 시작시 플레이어 바로 밑에 잘 오도록 포지션을 -1000, -300, -1000만큼 설정해주었다. CScene::Render부분에서도 m\_pTerrain->Render를 추가해주었다.

Object.cpp에서 CHeightMapTerrain 생성자에서 터레인쉐이더를 create해주고, pTerrainMaterial도 셋텍스쳐, 셋쉐이더하여 연결해주었다.

CTexture\* pTerrainTexture = new CTexture(3, RESOURCE\_TEXTURE2D, 0, 1); 루트 파라미터 개수는 1개로, LoadTextureFromFile로 텍스쳐를 읽어들이고,

CScene::CreateShaderResourceViews(pd3dDevice, pTerrainTexture, 0, 5); 루트 파라미터 시작 인덱스는 5로 설정해주었다.

<빌보드 – 꽃, 나무, 풀>

텍스트, 스크린샷, 폰트이(가) 표시된 사진

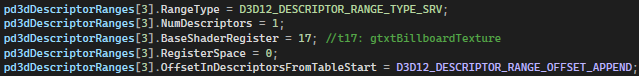
자동 생성된 설명

Shader.hlsl의 VSBillboard에서 시간에 따라 꽃과 나무, 풀이 좌우로 움직이는 애니메이션을 하도록 다음과 같이 설정해주었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

CreateGraphicsRootSignature에서 다음과 같이 디스크립터랭지를 설정해주고, 루트파라미터에 값을 넣어주었다.





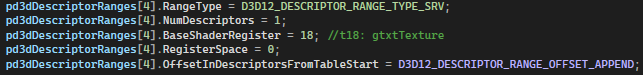
샘플러스테이트는 gssWrap로 설정해주었다.

CScene의 BuildObjects에서 기존에는 CObjectsShader만 있었는데, 여기다 CBillboardObjectsShader도 추가해줄 것이니 m\_nShaders를 2로 설정해주고, m\_ppSHaders[1]에다가 CBillboardObjectsShader로 만든 쉐이더를 넣어주었다.

<시작화면>



Shaders.hlsl에서 VSTextrueToScreen과 PSTextureToScreen를 추가해주었다. 또 다음과 같이 디스크립터 랭지와 루트 파라미터를 설정해주었다.





샘플러 스테이트도 기존의 gssWrap을 사용해주었다.

씬전환으로 시작화면과 인게임 화면을 나누기 위해 Scene.h에서 CScene을 상속받은 StartScene을 추가해주었다.

GameFramework.h에서 CScene \*m\_pScene[2];과 int n\_Scene{}; 멤버 변수를 추가해줘서 씬을 배열로 관리하도록 하였다.

‘s’키를 누르면 씬전환이 일어나도록 GameFramework.cpp에서 OnProcessingKeyboardMessage에서 case’s’:일 때 ChangeScene();함수가 호출되도록 해주었다. ChangeScene은 n\_Scene의 값을 변경해주도록 하였다. CGameFramework::BuildObjects()에서 m\_pScene[0]과 [1] 모두 빌드 오브젝트 해주고, n\_Scene의 값을 다르게 하여 렌더하도록 수정해주었다.

CGameFramework::FrameAdvance()에서 if (dynamic\_cast<StartScene\*>(m\_pScene[n\_Scene]) == nullptr && !is\_visiblity)

if (m\_pPlayer) m\_pPlayer->Render(m\_pd3dCommandList, m\_pCamera);으로 설정해서 StartScene일때는 플레이어가 안보이도록 해주었다.

 - Shader.h

 - Mesh.h

새롭게 쉐이더를 만들어서 StartScene::BuildObjects에서 이 쉐이더를 만들어주고 셋 텍스쳐를 해주었다. 그리고 m\_pDescriptorHeap == nullptr일 때 디스크립터힙을 생성해주고, CreateCbvSrvDescriptorHeaps(pd3dDevice, 1, 100);를 해서 좀 넉넉하게 만들어주었다.

<메뉴화면>

Shaders.hlsl에 다음과 같이 추가해주었다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

CALLBACK함수에 WM\_MOUSEWHEEL를 추가해주고, CGameFramework의 마우스메시지를 처리하는 함수에 WM\_MOUSEWHEEL이 일어나면 gfWheel의 값을 변동하도록 해주었다. 그리고 이 gfWheel에 따라 uv.y값을 조절하여 마우스 휠을 오르락 내리락하면 메뉴 텍스쳐가 움직이는 것처럼 구현하였다. 샘플러 스테이트는 Wrap으로 하여, 휠로 계속 내려서 메뉴 화면 텍스쳐의 끝을 넘어서면 다시 처음 텍스쳐를 렌더하도록 만들어주었다. 메뉴화면은 따로 씬으로 처리하지 않고, CGameFramework::BuildObjects()에서 CScrollMenuShader\* pTextureToScreenShader = new CScrollMenuShader(1);하여 쉐이더를 만들어주었다. 그리고 is\_visiblity라는 bool변수를 주어 TAB키를 누르면 is\_visiblity의 값이 바뀌도록 하여 토글기능을 넣어주었다.

루트 파라미터는 다음과 같이 설정해주었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

CScene의 UpdateShaderVariables에서 다음과 같은 내용을 넣어주었다.



<총알발사>

기존 1학기때 만들었던 bin파일과 코드를 이용하여 구현하였다. case VK\_CONTROL: 메시지가 입력되면 ((CAirplanePlayer\*)m\_pPlayer)->FireBullet(m\_pLockedObject);하여 Ctrl키를 누를때마다 플레이어가 총알을 발사하도록 하였다. 그리고 1학기 과제에서 만든 bin파일을 사용하기 위해 새로운 LoadMeshFromFile()을 만들어주고 해서 CBulletObject 클래스를 만들어주었다. 그리고 CAirplnePlayer의 생성자가 호출될 때 총알메쉬를 읽어들여, 총알을 초기화해주었다. 그리고 Render부분에도 if문에 bool변수인 m\_bActive라는 변수를 주어, 발사된 총알은 그리도록 해주었다. OBB를 이용하여 CheckObjectByBulletCollisions()에서 총알과 오브젝트의 충돌체크를 해주었다. 충돌된 객체의 live를 false로 해주고, 폭발이펙트가 그 객체의 위치에서 그려지도록 해주었다.

<충돌회피>

CheckObjectByObjectCollisions()함수를 사용해서 적기체와 플레이어간의 충돌을 체크해주었다. 충돌하면 플레이어와 충돌된 객체의 정보를 저장하고(m\_pPlayer->m\_pObjectCollided = m\_Objects[j];), 플레이어가 넉백당하는 힘의 크기를 2.0f로 초기화해주었다. Player.cpp의 CPlayer::Update에서 넉백 로직을 넣어주었다.

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 스크린샷이(가) 표시된 사진

자동 생성된 설명

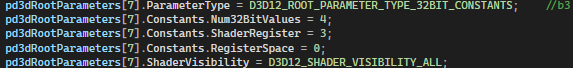
<폭발 효과>

cbuffer cbFrameworkInfo : register(b3)에 float gfNowTime : packoffset(c0.w);를 추가해주었다.

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

알파값이 0.85보다 밑이면 discard해주었다. 시간에 따라 uv좌표를 조정하여 현재 프레임의 텍스쳐 영역으로 이동하였다. 그에 따른 루트파라미터도 재조절해주었다.



CScene::CheckObjectByBulletCollisions()에서 총알과 오브젝트가 충돌하면 총알을 리셋해주고, m\_ExplodeObjects의 포지션을 충돌당한 오브젝트의 위치로 셋해주고, live를 true로 해주었다.

CExplodeObject::Animate에서 다음과 같은 로직을 추가하여 live가 true이면 폭발효과가 나타나도록 해주었다. 그리고 폭발효과가 끝나면 live를 false로 바꿔서 렌더링에서 제외시켜주었다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

m\_NowTime는 CExplodeObject의 멤버변수다. CExplodeObject::UpdateShaderVariable에서 pd3dCommandList->SetGraphicsRoot32BitConstants(7, 1, &m\_NowTime, 3);해주었다.

**추가적으로 구현한 내용**

* bullet.bin은 직접 만들어주었다.
* 폭발 이펙트를 추가해주었다.
* 꽃, 나무, 풀 빌보드가 바람에 흩날리는 것처럼 표현해주었다.

이상으로 보고를 마치겠습니다.