# 3D 게임2 과제03 보고서

게임공학과 3학년

2022182018 서가은

# 목차

I. 과제에 대한 목표
II. 가정
III. 실행결과
IV. 조작법
V. 구현 내용

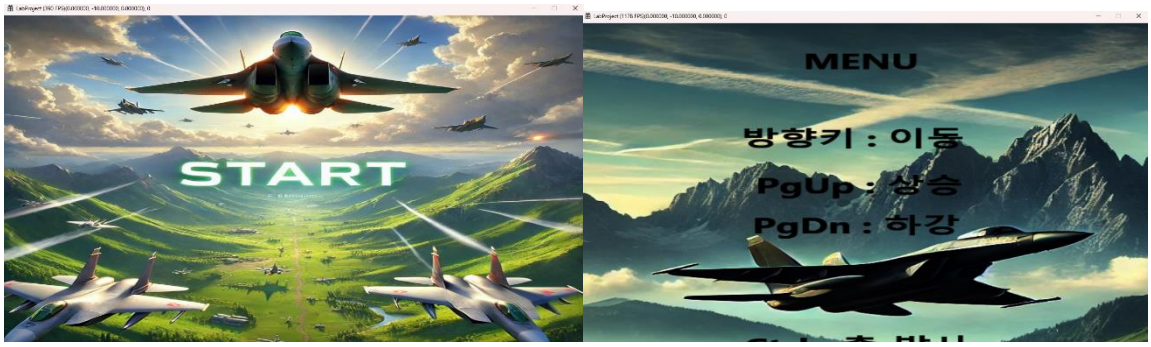
# 과제에 대한 목표

시작 화면, 메뉴, 씬 전환, 적 기체와 충돌 시에 넉백, 총알 발사, 폭발 이펙트, 터레인, 빌보드를 구현한 1차 과제의 내용을 덧붙인 강, UI, 파티클을 구현한 2차 과제에서 출발하여 거울, 환경 매핑, 테셀레이션을 구현하는 것을 목표로 하였다.

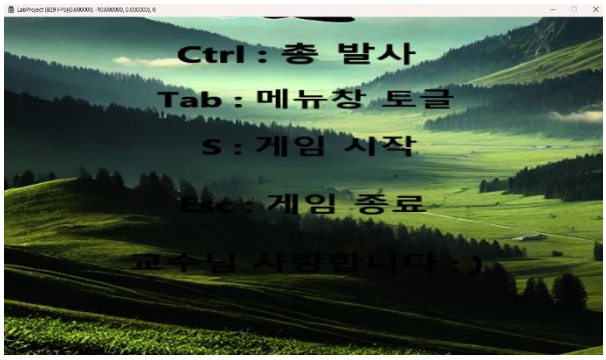
# 가정

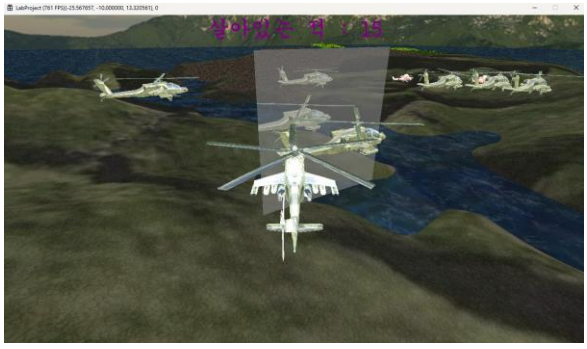
2차 과제에서 출발하여 테셀레이션 예제인 LabProject09-1-0을 참고하면 테셀레이션을 효율적으로 만들 수 있을 것 같았다. 또한, 'DirectX 12를 이용한 3D 게임 프로그래밍 입문' 이라는 책의 예제를 활용하면 거울과 환경매핑을 구현하기 쉬울 것 같았다.

# 실행결과



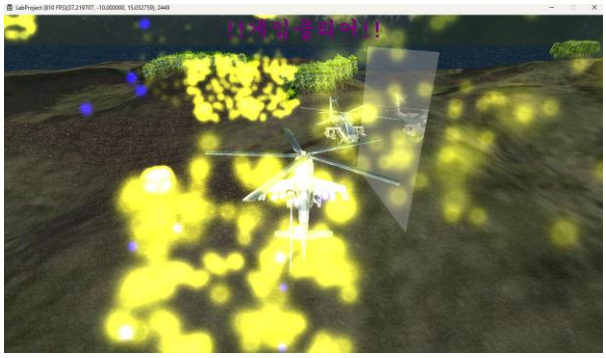

ㄴ 시작화면 / 메뉴 스크롤 전






ㄴ 메뉴 스크롤 후 / 인게임 화면


ㄴ 총알발사 / 적 기체 피격시 폭발 이펙트




ㄴ 적 기체를 모두 파괴했을 시 게임클리어 화면

[테셀레이션 지형 적용 전 / 후 비교 사진]

# 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>

<빌보드 – 꽃, 나무, 풀>

<시작화면>

<메뉴화면>

<총알발사>

<충돌회피 - 넉백>

<폭발 효과>

<강>

<UI>

<파티클>

까지 과제 2 에서 구현한 내용을 바탕으로

<환경 매핑>

<거울>

<테셀레이션>

을 구현하였다.

[환경 매핑]

PSStandard 에서 다음과 같이 수정해주었다.

```
float3 toEyeW = normalize(gvCameraPosition - input.positionW);
float3 r = reflect(-toEyeW, input.normalW);
float4 reflectionColor = gtxtSkyCubeTexture.Sample(gssWrap, r);

return (cColor + reflectionColor * 1.0f);
```

toEyeW는 월드 공간에서 픽셀로부터 카메라 방향을 계산하는 벡터이고, r은 반사벡터이다. reflectionColor는 반사된 벡터(r)를 사용해 큐브 맵에서 샘플링한 결과 색상이다. 최종 결과는 기본 색상과 반사 색상을 더한 값으로 출력하여 환경매핑을 구현하였다.

[거울]

shader.hlsl에 다음과 같이 추가해주었다.

```
float4 PSMirror(VS_TEXTURED_OUTPUT input) : SV_TARGET
{
    float4 cColor = (0.3f, 0.5f, 0.3f, 0.55f);
    return (cColor);
}
```

저 쉐이더는 실제로 렌더링 될 거울를 표현할 쉐이더이다. 버텍스 쉐이더는 기존의 사용하던 버텍스 쉐이더 중 하나를 재활용하여 사용하였다. 거울을 구현하기 위해 class CMirrorShader : public CShader 와 class CReflectShader : public CStandardShader, class CTransparentShader : public CShader를 만들어주었다. CMirrorShader는 스텐실 버퍼에다가 거울이 렌더링될 영역의 스텐실 값을 채우는 역할을 한다. CReflectShader는 거울 안에 비쳐질 오브젝트를 그리기 위해 사용된다. 그리고 CTransparentShader는 실제 렌더링 될 거울을 위해 사용된다. CMirrorShader의 CreateDepthStencilState()에서 기존의 내용을 변형하여StencilEnable를 TRUE로 설정, StencilReadMask와 StencilWriteMask를 0xff로 설정, StencilPassOp를 D3D12_STENCIL_OP_REPLACE, StencilFunc를 D3D12_COMPARISON_FUNC_ALWAYS로 바꾸어 주었다. CMirrorShader의 CreateRasterizerState()에서 기존의 내용을 변형하여 CullMode를 D3D12_CULL_MODE_BACK로 FrontCounterClockwise를 TRUE로 수정해주었다. CMirrorShader의 CreateBlendState()에서 기존의 내용을 변형하여 BlendEnable를 FALSE로, RenderTargetWriteMask를 0으로 수정해주었다. 이런식으로 CReflectShader과 CTransparentShader도 각각의 쉐이더와 맞는 내용으로 수정하여 채워주었다. 게임에서 거울 효과를 표현하기 위해서 총 3가지의 오브젝트를 렌더링하였다. 첫번째로 실제로 존재하는 게임 오브젝트, 두번째로 거울에 비치는 반사 오브젝트, 세번째로 실제로 스텐실 버퍼를 시각화 하는데 표현할 거울 오브젝트를 렌더링하였다. CScene의 BuildObjects()에서 거울 밖의 오브젝트인 m_Airplane를 하나 만들어 주었다. 그리고 CMirrorShader, CReflectShader, CTransparentShader를 m_ppShaders에 추가해주었다. 거울을 표현하는데 렌더링 순서는 첫번째가 CObjectsShader, 두번째가 CMirrorShader, 세번째가 CReflectShader, 네번째로 pTransparentShader이다.

그리고 CMirrorShader의 Render를 다음과 같이 설정해주었다.

```
void CMirrorShader::Render(ID3D12GraphicsCommandList* pd3dCommandList, CCamera* pCamera, int nPipelineState)
{
    pd3dCommandList->OMSetStencilRef(StencilRef: 1);
    OnPrepareRender(pd3dCommandList);
    mMirror->Render(pd3dCommandList, pCamera);
}
```

CReflectShader의 Render도 다음과 같이 만들어 주었다.

```
void CReflectShader::Render(ID3D12GraphicsCommandList* pd3dCommandList, CCamera* pCamera, int nPipelineState)
{
    OnPrepareRender(pd3dCommandList);
    m_ReflectAirplane->Render(pd3dCommandList, pCamera);
    pd3dCommandList->OMSetStencilRef(StencilRef: 0);
}
```

이렇게 해서 CReflectShader 안에 있는 m_ReflectAirplane만 스텐실를 이용한 렌더링을 할 수 있었고, 이후에 그려질 오브젝트들은 기존과 같이 정상적으로 렌더링 될 수 있었다.

[테셀레이션]

LabProject09-1-0의 Shader.hlsl를 이용하여 넣어주었다. 최대한 기존의 변수들을 사용하여 고쳐주었고, cbFrameworkInfo에 uint gnRenderMode : packoffset(c2.z);를 추가해주었다. 기존 b3을 루트 파라미터 타입을 32BIT_CONSTANTS로 해주었기에 Num32BitValues를 10에서 11로 수정해주고, CScene의 UpdateShaderVariables()에서 pd3dCommandList->SetGraphicsRoot32BitConstants(7, 1, &gnRenderMode, 10);를 추가해주었다.
CTerrainTessellationShader를 만들어서 테셀레이션을 적용한 지형을 만들어주었다. CreateVertexShader()와 같은 함수를 기존에 사용하던 함수의 형태로 수정해주고, m_pd3dHullShaderBlob와 m_pd3dDomainShaderBlob를 추가해주었다.
CHeightMapTerrain의 생성자에서 CTerrainShader* pTerrainShader = new CTerrainShader(); 부분을 만들어 두었던 CTerrainTessellationShader* pTerrainShader = new CTerrainTessellationShader();로 수정해주었다. 그리고 Mesh.cpp도 LabProject09-1-0의 내용을 참고하여 수정해주었다. M_pTerrain->SetPosition을 이용하여 위치를 재조정해주었다.

이상으로 보고를 마치겠습니다.