**3D Game Programming 과제4 보고서**

게임공학과 2013180028 이우상

**[ 목표 ]**

DirectX 3D 12의 파이프라인을 이해하고, 텍스처를 이용하여 디테일 매핑, 나무, 꽃, 풀 빌보드를 많이 만든다. 나무, 꽃, 풀은 흙 위에 만든다.

**[ 조작법 ]**

1. F1, F2, F3 키로 플레이어 시점을 바꿀 수 있음

- F1 : 플레이어 1인칭 모드

- F2 : 스페이스 쉽 모드

- F3 : 플레이어 3인칭 모드

2. 화살표 키(↑, ↓, ←, →) 키를 사용하여 앞, 뒤, 좌, 우 방향으로 플레이어를 이동할 수 있음

3. PageUp, PageDown 키를 사용하여 위, 아래 방향으로 플레이어를 이동할 수 있음

4. 마우스 왼쪽 버튼 클릭으로 플레이어를 회전시킬 수 있다.

5. F11 : 전체화면, 창모드 화면으로 전환할 수 있다.

**[ 구현 방법 ]**

1. **지형에 흙 디테일 텍스처 매핑**

//CDiffused2TexturedVertex 클래스 생성자 선언 부분을 수정하고, 멤버 변수를 추가한다.

class CDiffused2TexturedVertex : public CDiffusedVertex

{

public:

CDiffused2TexturedVertex();

CDiffused2TexturedVertex(float x, float y, float z, XMFLOAT4 xmf4Diffuse, XMFLOAT2 xmf2TexCoord0, XMFLOAT2 xmf2TexCoord1, XMFLOAT2 xmf2TexCoord2);

CDiffused2TexturedVertex(XMFLOAT3 xmf3Position, XMFLOAT4 xmf4Diffuse = XMFLOAT4(0.0f, 0.0f, 0.0f, 0.0f), XMFLOAT2 xmf2TexCoord0 = XMFLOAT2(0.0f, 0.0f), XMFLOAT2 xmf2TexCoord1 = XMFLOAT2(0.0f, 0.0f), XMFLOAT2 xmf2TexCoord2 = XMFLOAT2(0.0f, 0.0f));

~CDiffused2TexturedVertex();

// 하이트맵 uv 좌표 변수

XMFLOAT2 m\_xmf2TexCoord0;

// 기존 디테일 매핑 uv 좌표 변수

XMFLOAT2 m\_xmf2TexCoord1;

// 흙 텍스처를 디테일 매핑할 uv좌표 변수를 추가해준다.

XMFLOAT2 m\_xmf2TexCoord2;

};

// CDiffused2TexturedVertex 클래스 생성자 정의 부분을 수정한다.

CDiffused2TexturedVertex::CDiffused2TexturedVertex()

{

m\_xmf3Position = XMFLOAT3(0.0f, 0.0f, 0.0f);

m\_xmf4Diffuse = XMFLOAT4(0.0f, 0.0f, 0.0f, 0.0f);

m\_xmf2TexCoord0 = XMFLOAT2(0.0f, 0.0f);

m\_xmf2TexCoord1 = XMFLOAT2(0.0f, 0.0f);

// 흙 텍스처를 디테일 매핑할 uv좌표를 0으로 초기화

// 셰이더 코드에서 흙 uv좌표를 이용하여 0이면 기존 디테일 텍스처를 이용하여 매핑하 // 고, 0이 아니면, 흙으로 매핑하기 위함이다.

m\_xmf2TexCoord2 = XMFLOAT2(0.0f, 0.0f);

}

CDiffused2TexturedVertex::CDiffused2TexturedVertex(float x, float y, float z, XMFLOAT4 xmf4Diffuse, XMFLOAT2 xmf2TexCoord0, XMFLOAT2 xmf2TexCoord1, XMFLOAT2 xmf2TexCoord2)

{

m\_xmf3Position = XMFLOAT3(x, y, z);

m\_xmf4Diffuse = xmf4Diffuse;

m\_xmf2TexCoord0 = xmf2TexCoord0;

m\_xmf2TexCoord1 = xmf2TexCoord1;

m\_xmf2TexCoord2 = xmf2TexCoord2;

}

CDiffused2TexturedVertex::CDiffused2TexturedVertex(XMFLOAT3 xmf3Position, XMFLOAT4 xmf4Diffuse, XMFLOAT2 xmf2TexCoord0, XMFLOAT2 xmf2TexCoord1, XMFLOAT2 xmf2TexCoord2)

{

m\_xmf3Position = xmf3Position;

m\_xmf4Diffuse = xmf4Diffuse;

m\_xmf2TexCoord0 = xmf2TexCoord0;

m\_xmf2TexCoord1 = xmf2TexCoord1;

m\_xmf2TexCoord2 = xmf2TexCoord2;

}

//CHieghtMapGridMesh 클래스의 생성자 정의 부분을 수정한다.

CHeightMapGridMesh::CHeightMapGridMesh(ID3D12Device \*pd3dDevice, ID3D12GraphicsCommandList \*pd3dCommandList, int xStart, int zStart, int nWidth, int nLength, XMFLOAT3 xmf3Scale, XMFLOAT4 xmf4Color, void \*pContext) : CMesh(pd3dDevice, pd3dCommandList)

{

**. . . (생략)**

float fHeight = 0.0f, fMinHeight = +FLT\_MAX, fMaxHeight = -FLT\_MAX;

for (int i = 0, z = zStart; z < (zStart + nLength); z++)

{

for (int x = xStart; x < (xStart + nWidth); x++, i++)

{

fHeight = OnGetHeight(x, z, pContext);

pVertices[i].m\_xmf3Position = XMFLOAT3((x\*m\_xmf3Scale.x), fHeight, (z\*m\_xmf3Scale.z));

pVertices[i].m\_xmf4Diffuse = Vector4::Add(OnGetColor(x, z, pContext), xmf4Color);

pVertices[i].m\_xmf2TexCoord0 = XMFLOAT2(float(x) / float(cxHeightMap - 1), float(czHeightMap - 1 - z) / float(czHeightMap - 1));

// 여기서 Flatland\_Height 는 하이트 맵 지형의 평평한 부분의 높이이다.

// 하이트 맵 지형에서 Flatland\_Height 보다 낮으면, m\_xmf2TexCoord1의 좌표 값에 써서 기존

// 디테일 텍스처를 매핑하고, m\_xmf2TexCoord2에는 생성자에서 초기화한 0으로 값을 쓴다.

// 하이트 맵 지형에서 Flatland\_Height 보다 크면, m\_xmf2TexCoord1의 좌표 값에 생성자에서

// 0의 값으로 초기화 한 값을 사용하고, m\_xmf2TexCoord2에는 지형의 값을 쓴다.

// 이렇게 하면, Flatland\_Height 값에 따라 일정 높이 부분을 지형에 흙을 디테일 매핑할 수 있다.

if (fHeight < Flatland\_Height)

pVertices[i].m\_xmf2TexCoord1 = XMFLOAT2(float(x) / float(m\_xmf3Scale.x\*0.5f), float(z) / float(m\_xmf3Scale.z\*0.5f));

else

pVertices[i].m\_xmf2TexCoord2 = XMFLOAT2(float(x) / float(m\_xmf3Scale.x\*0.5f), float(z) / float(m\_xmf3Scale.z\*0.5f));

if (fHeight < fMinHeight)

fMinHeight = fHeight;

if (fHeight > fMaxHeight)

fMaxHeight = fHeight;

}

}

**. . . (생략)**

}

//흙 uv 좌표가 추가 되었으므로, 셰이더 코드로 흙 uv좌표를 넘겨줄 CTerrainShader

//클래스의 InputLayout을 만드는 CreateInputLayout( )를 수정한다.

D3D12\_INPUT\_LAYOUT\_DESC CTerrainShader::CreateInputLayout()

{

//UINT nInputElementDescs = 4;

// 셰이더 코드로 넘겨줄 흙 uv 좌표 개수가 1개가 추가 되었으므로,

// InputLayout 개수도 한개 증가한다.

UINT nInputElementDescs = 5;

D3D12\_INPUT\_ELEMENT\_DESC \*pd3dInputElementDescs = new D3D12\_INPUT\_ELEMENT\_DESC[nInputElementDescs];

// 터레인 지형의 위치 정보

pd3dInputElementDescs[0] = { "POSITION", 0, DXGI\_FORMAT\_R32G32B32\_FLOAT, 0, 0, D3D12\_INPUT\_CLASSIFICATION\_PER\_VERTEX\_DATA, 0 };

// 터레인의 색깔 정보

pd3dInputElementDescs[1] = { "COLOR", 0, DXGI\_FORMAT\_R32G32B32A32\_FLOAT, 0, 12, D3D12\_INPUT\_CLASSIFICATION\_PER\_VERTEX\_DATA, 0 };

// 기존 지형의 텍스처 uv

pd3dInputElementDescs[2] = { "TEXCOORD", 0, DXGI\_FORMAT\_R32G32\_FLOAT, 0, 28, D3D12\_INPUT\_CLASSIFICATION\_PER\_VERTEX\_DATA, 0 };

// 기존 지형의 디테일 텍스처 uv

pd3dInputElementDescs[3] = { "TEXCOORD", 1, DXGI\_FORMAT\_R32G32\_FLOAT, 0, 36, D3D12\_INPUT\_CLASSIFICATION\_PER\_VERTEX\_DATA, 0 };

// 흙 디테일 텍스처 uv 추가

pd3dInputElementDescs[4] = { "TEXCOORD", 2, DXGI\_FORMAT\_R32G32\_FLOAT, 0, 44, D3D12\_INPUT\_CLASSIFICATION\_PER\_VERTEX\_DATA, 0 };

D3D12\_INPUT\_LAYOUT\_DESC d3dInputLayoutDesc;

d3dInputLayoutDesc.pInputElementDescs = pd3dInputElementDescs;

d3dInputLayoutDesc.NumElements = nInputElementDescs;

return(d3dInputLayoutDesc);

}

//CHeightMapTerrain 클래스의 생성자를 수정한다.

CHeightMapTerrain::CHeightMapTerrain(ID3D12Device \*pd3dDevice, ID3D12GraphicsCommandList \*pd3dCommandList, ID3D12RootSignature \*pd3dGraphicsRootSignature, LPCTSTR pFileName, int nWidth, int nLength, int nBlockWidth, int nBlockLength, XMFLOAT3 xmf3Scale, XMFLOAT4 xmf4Color) : CGameObject(0)

{

**. . . (생략)**

CreateShaderVariables(pd3dDevice, pd3dCommandList);

// 기존에는 하이트 맵 텍스처와 기존 디테일 텍스처 2개였지만,

// 흙 디테일 텍스처가 추가 되었으므로, CTexture의 생성자 인자 값을 3을 준다.

//CTexture \*pTerrainTexture = new CTexture(2, RESOURCE\_TEXTURE2D, 0);

CTexture \*pTerrainTexture = new CTexture(3, RESOURCE\_TEXTURE2D, 0);

pTerrainTexture->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Base\_Texture.dds", 0);

pTerrainTexture->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Detail\_Texture\_7.dds", 1);

// 흙 디테일 텍스처 파일을 로드한다.

pTerrainTexture->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Sand01.dds", 2);

UINT ncbElementBytes = ((sizeof(CB\_GAMEOBJECT\_INFO) + 255) & ~255); //256의 배수

// 텍스처 개수를 나타내는 변수 추가 : 텍스처가 총 3개이므로 3을 대입

// (기존 하이트 맵 텍스처, 기존 디테일 텍스처, 흙 텍스처)

const int Terrain\_Texture\_Count = 3;

CTerrainShader \*pTerrainShader = new CTerrainShader();

pTerrainShader->CreateShader(pd3dDevice, pd3dGraphicsRootSignature);

pTerrainShader->CreateShaderVariables(pd3dDevice, pd3dCommandList);

// 디스크립터 힙을 만들 때, 텍스처 개수 변수를 인자로 넣어준다.

pTerrainShader->CreateCbvSrvDescriptorHeaps(pd3dDevice, pd3dCommandList, 1, Terrain\_Texture\_Count);

pTerrainShader->CreateConstantBufferViews(pd3dDevice, pd3dCommandList, 1, m\_pd3dcbGameObject, ncbElementBytes);

pTerrainShader->CreateShaderResourceViews(pd3dDevice, pd3dCommandList, pTerrainTexture, 4, true);

// 터레인의 재질을 만들고, 재질에 텍스처를 Set해주고, 디스크립터 힙과 셰이더를 Set.

CMaterial \*pTerrainMaterial = new CMaterial();

pTerrainMaterial->SetTexture(pTerrainTexture);

SetMaterial(pTerrainMaterial);

SetCbvGPUDescriptorHandle(pTerrainShader->GetGPUCbvDescriptorStartHandle());

SetShader(pTerrainShader);

}

//루트 시그니처를 만드는 CScene 클래스의 :CreateGraphicsRootSignature를 수정한다.

ID3D12RootSignature \*CScene::CreateGraphicsRootSignature(ID3D12Device \*pd3dDevice)

{

ID3D12RootSignature \*pd3dGraphicsRootSignature = NULL;

// 디스크립터 레인지를 흙 디테일 텍스처 추가되었으므로 5로 수정

D3D12\_DESCRIPTOR\_RANGE pd3dDescriptorRanges[5];

// 0번은 상수버퍼로 게임오브젝트의 정보를 담는 레인지 설정

pd3dDescriptorRanges[0].RangeType = D3D12\_DESCRIPTOR\_RANGE\_TYPE\_CBV;

pd3dDescriptorRanges[0].NumDescriptors = 1;

pd3dDescriptorRanges[0].BaseShaderRegister = 2; //GameObject

pd3dDescriptorRanges[0].RegisterSpace = 0;

pd3dDescriptorRanges[0].OffsetInDescriptorsFromTableStart = D3D12\_DESCRIPTOR\_RANGE\_OFFSET\_APPEND;

// 1번은 나무, 꽃, 풀 텍스처 레인지 설정

pd3dDescriptorRanges[1].RangeType = D3D12\_DESCRIPTOR\_RANGE\_TYPE\_SRV;

pd3dDescriptorRanges[1].NumDescriptors = 1;

pd3dDescriptorRanges[1].BaseShaderRegister = 0; //t0: gtxtTexture

pd3dDescriptorRanges[1].RegisterSpace = 0;

pd3dDescriptorRanges[1].OffsetInDescriptorsFromTableStart = D3D12\_DESCRIPTOR\_RANGE\_OFFSET\_APPEND;

// 2번은 터레인 지형 레인지 설정

pd3dDescriptorRanges[2].RangeType = D3D12\_DESCRIPTOR\_RANGE\_TYPE\_SRV;

pd3dDescriptorRanges[2].NumDescriptors = 1;

pd3dDescriptorRanges[2].BaseShaderRegister = 4; //t4: gtxtTerrainBaseTexture

pd3dDescriptorRanges[2].RegisterSpace = 0;

pd3dDescriptorRanges[2].OffsetInDescriptorsFromTableStart = D3D12\_DESCRIPTOR\_RANGE\_OFFSET\_APPEND;

// 3번은 터레인 지형 기본 디테일 텍스처 레인지 설정

pd3dDescriptorRanges[3].RangeType = D3D12\_DESCRIPTOR\_RANGE\_TYPE\_SRV;

pd3dDescriptorRanges[3].NumDescriptors = 1;

pd3dDescriptorRanges[3].BaseShaderRegister = 5; //t5: gtxtTerrainDetailTexture

pd3dDescriptorRanges[3].RegisterSpace = 0;

pd3dDescriptorRanges[3].OffsetInDescriptorsFromTableStart = D3D12\_DESCRIPTOR\_RANGE\_OFFSET\_APPEND;

// 4번은 터레인 지형 흙 디테일 텍스처 레인지 설정

pd3dDescriptorRanges[4].RangeType = D3D12\_DESCRIPTOR\_RANGE\_TYPE\_SRV;

pd3dDescriptorRanges[4].NumDescriptors = 1;

pd3dDescriptorRanges[4].BaseShaderRegister = 6; //t6: gtxtSandTexture

pd3dDescriptorRanges[4].RegisterSpace = 0;

pd3dDescriptorRanges[4].OffsetInDescriptorsFromTableStart = D3D12\_DESCRIPTOR\_RANGE\_OFFSET\_APPEND;

// 루트 파라미터를 흙 디테일 텍스처가 추가 되었으므로 7개로 수정

//D3D12\_ROOT\_PARAMETER pd3dRootParameters[6];

D3D12\_ROOT\_PARAMETER pd3dRootParameters[7];

// 플레이어 정보

pd3dRootParameters[0].ParameterType = D3D12\_ROOT\_PARAMETER\_TYPE\_CBV;

pd3dRootParameters[0].Descriptor.ShaderRegister = 0; //Player

pd3dRootParameters[0].Descriptor.RegisterSpace = 0;

pd3dRootParameters[0].ShaderVisibility = D3D12\_SHADER\_VISIBILITY\_VERTEX;

// 카메라 정보

pd3dRootParameters[1].ParameterType = D3D12\_ROOT\_PARAMETER\_TYPE\_CBV;

pd3dRootParameters[1].Descriptor.ShaderRegister = 1; //Camera

pd3dRootParameters[1].Descriptor.RegisterSpace = 0;

pd3dRootParameters[1].ShaderVisibility = D3D12\_SHADER\_VISIBILITY\_ALL;

pd3dRootParameters[2].ParameterType = D3D12\_ROOT\_PARAMETER\_TYPE\_DESCRIPTOR\_TABLE;

pd3dRootParameters[2].DescriptorTable.NumDescriptorRanges = 1;

pd3dRootParameters[2].DescriptorTable.pDescriptorRanges = &pd3dDescriptorRanges[0];

pd3dRootParameters[2].ShaderVisibility = D3D12\_SHADER\_VISIBILITY\_VERTEX;

// 나무, 꽃, 풀 텍스처 정보를 레인지 번호 0번에 설정 되어있으므로 0번에 연결

pd3dRootParameters[3].ParameterType = D3D12\_ROOT\_PARAMETER\_TYPE\_DESCRIPTOR\_TABLE;

pd3dRootParameters[3].DescriptorTable.NumDescriptorRanges = 1;

pd3dRootParameters[3].DescriptorTable.pDescriptorRanges = &pd3dDescriptorRanges[1];

pd3dRootParameters[3].ShaderVisibility = D3D12\_SHADER\_VISIBILITY\_PIXEL;

// 터레인 텍스처 정보를 레인지 번호 1번에 설정 되어있으므로 1번에 연결

pd3dRootParameters[4].ParameterType = D3D12\_ROOT\_PARAMETER\_TYPE\_DESCRIPTOR\_TABLE;

pd3dRootParameters[4].DescriptorTable.NumDescriptorRanges = 1;

pd3dRootParameters[4].DescriptorTable.pDescriptorRanges = &pd3dDescriptorRanges[2];

pd3dRootParameters[4].ShaderVisibility = D3D12\_SHADER\_VISIBILITY\_PIXEL;

// 터레인 기본 디테일 텍스처 정보를 레인지 번호 3번에 설정 되어있으므로 3번에 연결

pd3dRootParameters[5].ParameterType = D3D12\_ROOT\_PARAMETER\_TYPE\_DESCRIPTOR\_TABLE;

pd3dRootParameters[5].DescriptorTable.NumDescriptorRanges = 1;

pd3dRootParameters[5].DescriptorTable.pDescriptorRanges = &pd3dDescriptorRanges[3];

pd3dRootParameters[5].ShaderVisibility = D3D12\_SHADER\_VISIBILITY\_PIXEL;

// 터레인 흙 디테일 텍스처 정보를 레인지 번호 4번에 설정 되어있으므로 4번에 연결

pd3dRootParameters[6].ParameterType = D3D12\_ROOT\_PARAMETER\_TYPE\_DESCRIPTOR\_TABLE;

pd3dRootParameters[6].DescriptorTable.NumDescriptorRanges = 1;

pd3dRootParameters[6].DescriptorTable.pDescriptorRanges = &pd3dDescriptorRanges[4];

pd3dRootParameters[6].ShaderVisibility = D3D12\_SHADER\_VISIBILITY\_PIXEL;

//Shader.hlsl 파일에서 터레인 부분 수정

**. . .(생략)**

Texture2D gtxtTerrainBaseTexture : register(t4);

Texture2D gtxtTerrainDetailTexture : register(t5);

Texture2D gtxtSandTexture : register(t6);

struct VS\_TERRAIN\_INPUT

{

float3 position : POSITION;

float4 color : COLOR;

float2 uv0 : TEXCOORD0; // 터레인 지형 텍스처 uv

float2 uv1 : TEXCOORD1; // 기본 디테일 텍스처 uv

float2 uv2 : TEXCOORD2; // 흙 디테일 텍스처 uv

};

struct VS\_TERRAIN\_OUTPUT

{

float3 position : POSITION;

float4 color : COLOR;

float2 uv0 : TEXCOORD0; // 터레인 지형 텍스처 uv

float2 uv1 : TEXCOORD1; // 기본 디테일 텍스처 uv

float2 uv2 : TEXCOORD2; // 흙 디테일 텍스처 uv

};

VS\_TERRAIN\_OUTPUT VSTerrain(VS\_TERRAIN\_INPUT input)

{

VS\_TERRAIN\_OUTPUT output;

#ifdef \_WITH\_CONSTANT\_BUFFER\_SYNTAX

output.position = mul(mul(mul(float4(input.position, 1.0f), gcbGameObjectInfo.mtxWorld), gcbCameraInfo.mtxView), gcbCameraInfo.mtxProjection);

#else

output.position = mul(mul(mul(float4(input.position, 1.0f), gmtxWorld), gmtxView), gmtxProjection);

#endif

output.color = input.color;

// 터레인 지형 텍스처 uv 정보

output.uv0 = input.uv0;

// 기본 디테일 텍스처 uv 정보

output.uv1 = input.uv1;

// 흙 디테일 텍스처 uv 정보

output.uv2 = input.uv2;

return(output);

}

float4 PSTerrain(VS\_TERRAIN\_OUTPUT input) : SV\_TARGET

{

float4 cBaseTexColor = gtxtTerrainBaseTexture.Sample(gWrapSamplerState, input.uv0);

float4 cDetailTexColor = gtxtTerrainDetailTexture.Sample(gWrapSamplerState, input.uv1);

// 흙 텍스처 색상을 uv2에서 받아와 저장

float4 cSandTexColor = gtxtSandTexture.Sample(gWrapSamplerState, input.uv2);

float4 cColor;

// uv2가 0보다 크면, 흙 텍스처가 매핑되야하는 일정 높이 부분이므로

// uv2가 초기값(0.0)보다 크다.

if (input.uv2.r > 0.0f)

cColor = cSandTexColor;

// uv2가 0이면 흙 텍스처가 매핑 되야하는 좌표가 아니므로

// 기본 디테일 텍스처와 터레인 지형의 색상을 0.5씩 섞는다.

else

cColor = input.color \* saturate((cBaseTexColor \* 0.5f) + (cDetailTexColor \* 0.5f));

return(cColor);

}

1. **나무, 꽃, 풀 빌보드 텍스처 매핑**

// Mesh.h에서 CTexturedRectMesh를 추가한다. CTexturedRectMesh는 사각형 메쉬이다.

// 나무 모델을 사용하지 않고, 사각형 메쉬를 사용하는 이유는 많은 나무, 풀, 꽃을 모두

// 모델을 사용하면 프레임이 나오지 않기 때문에, 사각형으로 폴리곤 수를 대폭 줄일 수

// 있으면서, 나무처럼 보이기위해 사각형 메쉬 1개로 사용한다.

class CTexturedRectMesh : public CMesh

{

public:

CTexturedRectMesh(ID3D12Device \*pd3dDevice, ID3D12GraphicsCommandList \*pd3dCommandList, float fWidth=20.0f, float fHeight=20.0f, float fDepth=20.0f, float fxPosition=0.0f, float fyPosition=0.0f, float fzPosition=0.0f);

virtual ~CTexturedRectMesh();

};

// Mesh.cpp에서 CTexturedRectMesh의 생성자에서 다음과 같이 만든다.

CTexturedRectMesh::CTexturedRectMesh(ID3D12Device \*pd3dDevice, ID3D12GraphicsCommandList \*pd3dCommandList, float fWidth, float fHeight, float fDepth, float fxPosition, float fyPosition, float fzPosition) : CMesh(pd3dDevice, pd3dCommandList)

{

m\_nVertices = 6;

m\_nStride = sizeof(CTexturedVertex);

m\_d3dPrimitiveTopology = D3D\_PRIMITIVE\_TOPOLOGY\_TRIANGLELIST;

CTexturedVertex pVertices[6];

float fx = (fWidth \* 0.5f) + fxPosition, fy = (fHeight \* 0.5f) + fyPosition, fz = (fDepth \* 0.5f) + fzPosition;

**. . . (생략)**

else if (fDepth == 0.0f)

{

**. . . (생략)**

// 깊이 값이 0이고, z포지션이 0일 때,

// 모든 빌보드 텍스처는 이 메쉬를 사용

else

{

pVertices[0] = CTexturedVertex(XMFLOAT3(-fx, +fy, fz), XMFLOAT2(1.0f, 0.0f));

pVertices[1] = CTexturedVertex(XMFLOAT3(-fx, -fy, fz), XMFLOAT2(1.0f, 1.0f));

pVertices[2] = CTexturedVertex(XMFLOAT3(+fx, -fy, fz), XMFLOAT2(0.0f, 1.0f));

pVertices[3] = CTexturedVertex(XMFLOAT3(+fx, -fy, fz), XMFLOAT2(0.0f, 1.0f));

pVertices[4] = CTexturedVertex(XMFLOAT3(+fx, +fy, fz), XMFLOAT2(0.0f, 0.0f));

pVertices[5] = CTexturedVertex(XMFLOAT3(-fx, +fy, fz), XMFLOAT2(1.0f, 0.0f));

}

}

**. . . (생략)**

}

// CGameObject 클래스에서 SetLookAt( )를 정의한다. 빌보드 오브젝트(나무, 꽃, 풀)이

// 플레이어가 움직일 때마다 바라보게 하기 위함이다.

void CGameObject::SetLookAt(XMFLOAT3& xmf3Target, XMFLOAT3& xmf3Up)

{

XMFLOAT3 xmf3Position(m\_xmf4x4World.\_41, m\_xmf4x4World.\_42, m\_xmf4x4World.\_43);

XMFLOAT4X4 mtxLookAt = Matrix4x4::LookAtLH(xmf3Position, xmf3Target, xmf3Up);

m\_xmf4x4World.\_11 = mtxLookAt.\_11; m\_xmf4x4World.\_12 = mtxLookAt.\_21; m\_xmf4x4World.\_13 = mtxLookAt.\_31;

m\_xmf4x4World.\_21 = mtxLookAt.\_12; m\_xmf4x4World.\_22 = mtxLookAt.\_22; m\_xmf4x4World.\_23 = mtxLookAt.\_32;

m\_xmf4x4World.\_31 = mtxLookAt.\_13; m\_xmf4x4World.\_32 = mtxLookAt.\_23; m\_xmf4x4World.\_33 = mtxLookAt.\_33;

}

// CBillboardShader.cpp

// CBillboardShader 클래스를 정의하고 BuildObjects( )를 추가한다.

// 이 함수에서는 위에서 정의한 CTexturedRectMesh(사각형 메쉬)에

// 나무, 꽃, 풀 오브젝트를 생성하고 텍스처를 매핑한다.

// 또, 터레인 지형에서 흙이 매핑된 높이의 x, z좌표를 얻어와

// 그 높이에 나무, 풀, 꽃 빌보드가 위치하게 초기화 한다.

void CBillboardShader::BuildObjects(ID3D12Device \*pd3dDevice, ID3D12GraphicsCommandList \*pd3dCommandList, void \*pContext)

{

CHeightMapTerrain \*pTerrain = (CHeightMapTerrain \*)pContext;

float fTerrainWidth = pTerrain->GetWidth();

float fTerrainLength = pTerrain->GetLength();

int xObjects = 35;

int zObjects = 50;

// 오브젝트의 총 개수 12,000개

m\_nObjects = 12000;

// 풀 텍스처 개수 2개, 꽃 텍스처 2개, 나무 텍스처 5개

const int Grass\_Texture\_Count = 2;

const int Flower\_Texture\_Count = 2;

const int Tree\_Texture\_Count = 5;

int Total\_Texutre\_count = Grass\_Texture\_Count + Flower\_Texture\_Count + Tree\_Texture\_Count;

// 풀의 총 개수인 2개만큼 CTexture 객체를 만들고, 풀 텍스처 2개 dds파일을 로드

CTexture \*ppGrassTextures[Grass\_Texture\_Count];

ppGrassTextures[0] = new CTexture(1, RESOURCE\_TEXTURE2D, 0);

ppGrassTextures[0]->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Grass01.dds", 0);

ppGrassTextures[1] = new CTexture(1, RESOURCE\_TEXTURE2D, 0);

ppGrassTextures[1]->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Grass01.dds", 0);

// 꽃의 총 개수인 2개만큼 CTexture 객체를 만들고, 꽃 텍스처 2개 dds파일을 로드

CTexture \*ppFlowerTextures[Flower\_Texture\_Count];

ppFlowerTextures[0] = new CTexture(1, RESOURCE\_TEXTURE2D, 0);

ppFlowerTextures[0]->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Flower01.dds", 0);

ppFlowerTextures[1] = new CTexture(1, RESOURCE\_TEXTURE2D, 0);

ppFlowerTextures[1]->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Flower02.dds", 0);

// 나무의 총 개수 5개만큼 객체를 만들고, 나무 텍스처 5개 dds파일 로드

CTexture\* ppTreeTextures[Tree\_Texture\_Count];

ppTreeTextures[0] = new CTexture(1, RESOURCE\_TEXTURE2D, 0);

ppTreeTextures[0]->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Tree01.dds", 0);

ppTreeTextures[1] = new CTexture(1, RESOURCE\_TEXTURE2D, 0);

ppTreeTextures[1]->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Tree02.dds", 0);

ppTreeTextures[2] = new CTexture(1, RESOURCE\_TEXTURE2D, 0);

ppTreeTextures[2]->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Tree03.dds", 0);

ppTreeTextures[3] = new CTexture(1, RESOURCE\_TEXTURE2D, 0);

ppTreeTextures[3]->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Tree04.dds", 0);

ppTreeTextures[4] = new CTexture(1, RESOURCE\_TEXTURE2D, 0);

ppTreeTextures[4]->LoadTextureFromFile(pd3dDevice, pd3dCommandList, L"Image/Tree05.dds", 0);

UINT ncbElementBytes = ((sizeof(CB\_GAMEOBJECT\_INFO) + 255) & ~255);

// 텍스처 총 개수 9개만큼 디스크립터 힙 생성

CreateCbvSrvDescriptorHeaps(pd3dDevice, pd3dCommandList, m\_nObjects, Total\_Texutre\_count);

CreateShaderVariables(pd3dDevice, pd3dCommandList);

CreateConstantBufferViews(pd3dDevice, pd3dCommandList, m\_nObjects, m\_pd3dcbGameObjects, ncbElementBytes);

// 풀, 꽃 리소스 뷰를 각각 텍스처 개수만큼 만든다.

for (int i = 0; i < 2; ++i)

{

CreateShaderResourceViews(pd3dDevice, pd3dCommandList, ppGrassTextures[i], 3, true);

CreateShaderResourceViews(pd3dDevice, pd3dCommandList, ppFlowerTextures[i], 3, true);

}

// 나무 리소스 뷰를 나무 텍스처 개수 5개만큼 만듬

for (int i = 0; i < Tree\_Texture\_Count; ++i)

CreateShaderResourceViews(pd3dDevice, pd3dCommandList, ppTreeTextures[i], 3, true);

// 각각 재질을 나무, 풀, 꽃 객체를 만들고, 각 재질에 맞는 텍스처를 Set해준다

CMaterial\* ppGrassMaterials[Grass\_Texture\_Count]{ nullptr };

CMaterial\* ppFlowerMaterials[Flower\_Texture\_Count]{ nullptr };

CMaterial\* ppTreeMaterials[Tree\_Texture\_Count]{ nullptr };

for (int i = 0; i < 2; ++i)

{

ppGrassMaterials[i] = new CMaterial();

ppGrassMaterials[i]->SetTexture(ppGrassTextures[i]);

ppFlowerMaterials[i] = new CMaterial();

ppFlowerMaterials[i]->SetTexture(ppFlowerTextures[i]);

}

for (int i = 0; i < Tree\_Texture\_Count; ++i)

{

ppTreeMaterials[i] = new CMaterial();

ppTreeMaterials[i]->SetTexture(ppTreeTextures[i]);

}

// 풀, 꽃, 나무 각각 높이가 다르므로, 사각형 메쉬를 3개로 만든다.

float GrassHeight = 20.f, FlowerHeight = 30.f, TreeHeight = 100.f;

CTexturedRectMesh\* pGrassMesh = new CTexturedRectMesh(pd3dDevice, pd3dCommandList, 10.0f, GrassHeight, 0.0f, 0.0f, 0.0f, 0.0f);

CTexturedRectMesh\* pFlowerMesh = new CTexturedRectMesh(pd3dDevice, pd3dCommandList, 20.0f, FlowerHeight, 0.0f, 0.0f, 0.0f, 0.0f);

CTexturedRectMesh\* pTreeMesh = new CTexturedRectMesh(pd3dDevice, pd3dCommandList, 50.0f, TreeHeight, 0.0f, 0.0f, 0.0f, 0.0f);

m\_ppObjects = new CGameObject\*[m\_nObjects];

CBillboardObject \*pBillboardObject = NULL;

float fxPitch = 0.25f;

float fzPitch = 0.25f;

CMaterial \*pMaterial = NULL;

CMesh \*pMesh = NULL;

float xPosition = 950.0f;

default\_random\_engine dre;

uniform\_real\_distribution<double> urd\_x(0.f, fTerrainWidth - 100);

uniform\_real\_distribution<double> urd\_z(0.f, fTerrainLength - 100);

// 텍스처 종류

uniform\_int\_distribution<int> uid\_TextureType(0, 2);

uniform\_int\_distribution<int> uid\_Material(0, 1);

uniform\_int\_distribution<int> uid\_TreeMaterial(0, Tree\_Texture\_Count - 1);

// 여기서 부터는 나무, 풀, 꽃 텍스처를 랜덤한 텍스처로 만들게 한다.

// 또한, 그 위치들을 아무 위치가 아니라, 터레인 지형에 흙이 매핑 된 위치에

// 만들어지게한다.

// 흙이 매핑된 높이값은 Flatland\_Height이기 때문에, 랜덤으로 나온 좌표 x, z값에 해당

// 하는 높이와 비교해서 같거나 클 때의 좌표를 찾아낸다.

double x = 0, z = 0;

for (int i = 0; i < m\_nObjects; ++i)

{

x = urd\_x(dre);

z = urd\_z(dre);

while (pTerrain->GetHeight(x, z) < Flatland\_Height)

{

x = urd\_x(dre);

z = urd\_z(dre);

}

pBillboardObject = new CBillboardObject;

// 텍스처, 재질 랜덤 지정

if (uid\_TextureType(dre) == GRASS)

{

pBillboardObject->SetMesh(0, pGrassMesh);

pBillboardObject->SetMaterial(ppFlowerMaterials[uid\_Material(dre)]);

pBillboardObject->SetPosition(x, pTerrain->GetHeight(x, z) + GrassHeight / 2.f, z);

}

else if (uid\_TextureType(dre) == FLOWER)

{

pBillboardObject->SetMesh(0, pFlowerMesh);

pBillboardObject->SetMaterial(ppGrassMaterials[uid\_Material(dre)]);

pBillboardObject->SetPosition(x, pTerrain->GetHeight(x, z) + FlowerHeight / 2.f, z);

}

else if (uid\_TextureType(dre) == TREE)

{

pBillboardObject->SetMesh(0, pTreeMesh);

// 나무는 3개

pBillboardObject->SetMaterial(ppTreeMaterials[uid\_TreeMaterial(dre)]);

pBillboardObject->SetPosition(x, pTerrain->GetHeight(x, z) + TreeHeight / 2.7f, z);

}

pBillboardObject->SetCbvGPUDescriptorHandlePtr(m\_d3dCbvGPUDescriptorStartHandle.ptr + (::gnCbvSrvDescriptorIncrementSize \* i));

m\_ppObjects[i] = pBillboardObject;

}

cout << "오브젝트 빌보드 개수 : " << m\_nObjects << "개 생성완료" << endl;

}

// CBillboardShader의 Render( ) 부분

// 빌보드 오브젝트가 플레이어가 움직일 때마다 바라보게 해야한다.

// 따라서 랜더할 때마다, 빌보드 오브젝트의 SetLookAt함수를 호출해서

// 플레이어를 바라보게 한다.

void CBillboardShader::Render(ID3D12GraphicsCommandList \*pd3dCommandList, CCamera \*pCamera)

{

XMFLOAT3 xmf3CameraPosition = pCamera->GetPosition();

for (int j = 0; j < m\_nObjects; j++)

{

if (m\_ppObjects[j])

m\_ppObjects[j]->SetLookAt(xmf3CameraPosition, XMFLOAT3(0.0f, 1.0f, 0.0f));

}

CObjectsShader::Render(pd3dCommandList, pCamera);

}

// Shader.hlsl에서 빌보드 텍스처의 픽셀셰이더를 수정한다.

float4 PSBillboard(VS\_TEXTURED\_OUTPUT input) : SV\_TARGET

{

float4 cColor = gtxtTexture.SampleLevel(gWrapSamplerState, input.uv, 0);

// 텍스처의 검은색 부분(알파 값이 0인 부분)이 discard를 안쓰면 그대로

// 출력된다. 따라서 이 검은 부분을 없애기 위해 픽셸셰이더에서 discard

// 명령어로 그부분을 출력하지 않게한다.

if (cColor.a <= 0.3f)

discard;

return(cColor);

}