**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. 흙 디테일 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 경계로 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 텍스처를 디스크립터 힙에 정보를 담는 경계로 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;

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;

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;

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;

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

D3D12\_ROOT\_PARAMETER pd3dRootParameters[7];

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;

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;

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;

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;

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;

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;