**3D Game Programming 과제2 보고서**

게임공학과 2013180028 이우상

**[ 목표 ]**

정점 하나만 생성하여, 기하셰이더로 사각형을 만들고, 빌보드 나무, 풀, 꽃 텍스처를 매핑한다. 또한 터레인도 기하셰이더를 적용시켜 만든다. 키 버튼 입력으로 기하셰이더를 이용해서 만든 빌보드, 터레인과, 기존에 만들었던 빌보드 오브젝트, 터레인의 차이를 WIREFRAME으로 보여준다.

**[ 조작법 ]**

1. s/S 키를 이용하여, RasterizerState의 FillMode를 SOLID와 WIREFRAME으로 바꾸어 다시 시작하게 한다.
2. g/G 키를 이용하여 기하셰이더를 적용하고, 적용하지 않은 것을 보여준다.

**[ 게임 이미지 ]**

**[ 구현 방법 ]**

1. **s/S, 키를 이용하여 FillMode를 구분 짓게 하는 변수 g\_FillMode를 전역변수로 생성한다.**

**또한 g/G 키를 이용하여 기하셰이더를 적용시킨 것과 적용시키지 않은 것을 보이기 위해, 전역변수 g\_OnGeometry를 생성한다.**

1. **enum으로 타입을 각각 정의하여, FillMode의 SOLID와 WIREFRAME, 기하셰이더를 사용한 유무의 USE, UNUSE를 만든다.**

**// 0번이면 FillMode를 SOLID로 생성, 1번이면 WIREFRAME으로 생성한다.**

enum FILL\_MODE { SOLID, WIRE};

**// false이면 기하셰이더를 적용하지 않고, true이면 기하셰이더를 적용**

enum ON\_GEOMETRY { USE, UNUSE };

**// 처음에는 FillMode를 SOLID, 기하셰이더를 적용 X로 한다.**

int g\_FillMode = SOLID;

bool g\_OnGeometry = UNUSE;

1. CGameFramework의 OnProcessingKeyboardMessage( )에서 s/S 와 g/G 키를 눌렀을 때, 각각 FillMode를 변경시키고, 기하셰이더를 적용시킬 수 있도록 만든다.

case 's':

case 'S':

**// s/S 키를 눌렀을 때, 솔리드 모드이면, 와이어프레임으로 바꿔주고,**

**// 와이어 모드이면, 솔리드 모드로 바꾼다.**

if (g\_FillMode == SOLID)

{

g\_FillMode = WIRE;

cout << "와이어 모드" << endl;

}

else

{

g\_FillMode = SOLID;

cout << "솔리드 모드" << endl;

}

break;

case 'g':

case 'G':

**// g/G 키를 눌렀을 때, 기하셰이더가 적용되었을 때는, 적용을 하지 않겠다고**

**// g\_OnGeometry를 UNUSE로 바꿔주고, 기하셰이더가 적용되지 않았을 때는**

**// USE로 바꾸어 기하셰이더를 적용시킨다.**

if (g\_OnGeometry == USE)

{

g\_OnGeometry = UNUSE;

cout << "기하셰이더 적용 X" << endl;

}

else

{

g\_OnGeometry = USE;

cout << "기하셰이더 적용 O" << endl;

}

break;

1. **CScene의 BuildObject( )에서 기하셰이더를 적용시킨 것과, 적용 시키지 않는 셰이더로 구분해서 만들어 준다.**

**// 기하셰더를 적용시킨다면, CGeometryShader 객체를 생성하여 셰이더를 생성**

if (g\_OnGeometry == USE)

{

CGeometryShader\* pGeometryShader = new CGeometryShader;

pGeometryShader->CreateShader(pd3dDevice, m\_pd3dGraphicsRootSignature);

pGeometryShader->setPositionArray(PositionArray);

pGeometryShader->BuildObjects(pd3dDevice, pd3dCommandList, m\_pTerrain);

m\_ppShaders[0] = pGeometryShader;

}

**// 기하셰더를 적용시키지 않는다면, CBillboardShader 객체를 생성하여 셰이더를 생성**

else if(g\_OnGeometry == UNUSE)

{

CBillboardShader\* pBillboardShader = new CBillboardShader;

pBillboardShader->CreateShader(pd3dDevice, m\_pd3dGraphicsRootSignature);

**// 나무, 꽃, 풀들이 위치할 텍스처 픽셸의 위치**

pBillboardShader->setPositionArray(PositionArray);

pBillboardShader->BuildObjects(pd3dDevice, pd3dCommandList, m\_pTerrain);

m\_ppShaders[0] = pBillboardShader;

}

1. **Terrain에서 기하셰이더를 적용시킨것과 적용시키지 않은 것을 만들기 위해 파이프라인 스테이트를 2개를 만든다. 1개는 기하셰이더를 적용시킨 것, 1개는 적용시키지 않은 것**

**// CTerrainShader의 CreateShader에서 파이프라인 개수를 2개로 하고,**

void CTerrainShader::CreateShader(ID3D12Device \*pd3dDevice, ID3D12RootSignature \*pd3dGraphicsRootSignature)

{

m\_nPipelineStates = 2;

…

**// 0번째 파이프라인이라면, 기하셰이더 코드를 컴파일한다. 1번째이면, 기하셰이더 코드를 컴파일하지 않음**

**// 적용시키**

If(i == 0)

{

d3dPipelineStateDesc[i].GS = CreateGeometryShader(&pd3dGeometryShaderBlob);

}

}

1. **기하셰이더 코드의 함수를 리턴하는 함수를 만들어준다.**

D3D12\_SHADER\_BYTECODE CTerrainShader::CreateGeometryShader(ID3DBlob \*\*ppd3dShaderBlob)

{

**// 기하셰이더를 적용시킨다면, GS\_Terrain 이라는 함수를 컴파일한다.**

return(CShader::CompileShaderFromFile(L"Shaders.hlsl", "GS\_Terrain", "gs\_5\_1", ppd3dShaderBlob));

}

1. **기하셰이더를 적용시켰으면 그에 맞는 픽셸셰이더 함수도 원래의 터레인 픽셸셰이더와 다르게 한 개 더 생성해준다.**

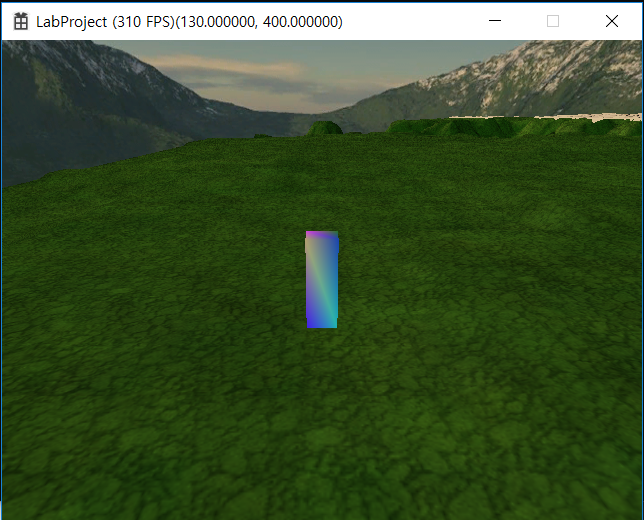
D3D12\_SHADER\_BYTECODE CTerrainShader::CreatePixelShader(ID3DBlob \*\*ppd3dShaderBlob)

{

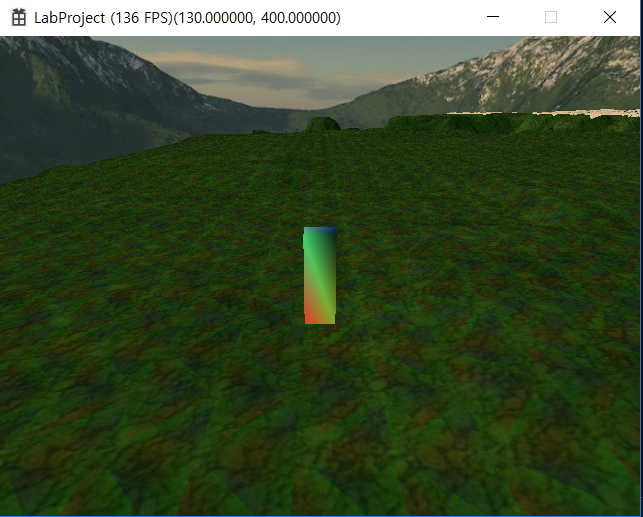
**// 기하셰이더가 적용되었을 때, PS\_GSTerrain 픽셸셰이더 함수를 실행**

return(CShader::CompileShaderFromFile(L"Shaders.hlsl", "PS\_GSTerrain", "ps\_5\_1", ppd3dShaderBlob));

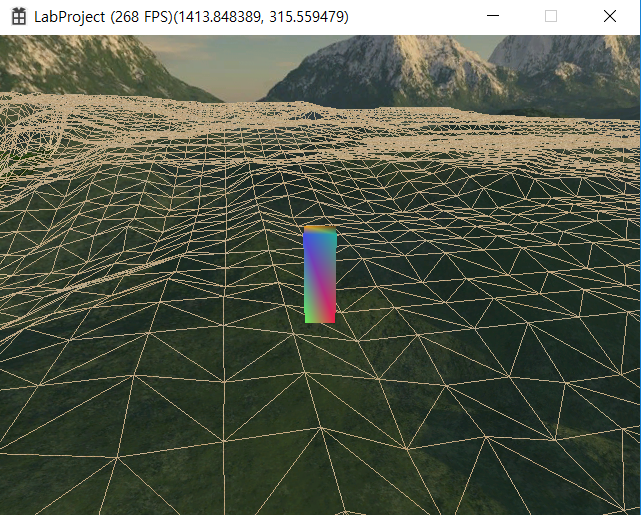
}



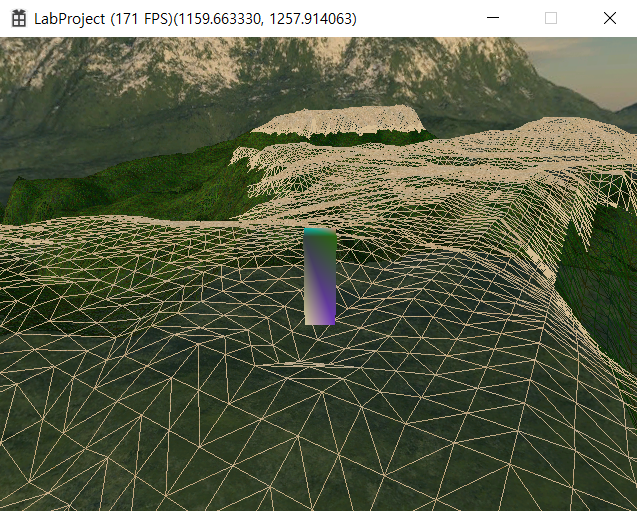
**기하셰이더를 적용하지 않고, 솔리드 모드일 때,**



**기하셰이더를 적용하고, 솔리드 모드일 때**



**기하셰이더를 적용하지 않고, 흙 위에서 와이어프레임 모드일 때**



**기하셰이더를 적용하고, 흙위에서 와이어프레임 모드일 때**