Mesh Part 1

LPD3DXMESH mesh = nullptr;

D3DXCreateTeapot(device, &mesh, 0);

// mesh에서 vertex,index buffer를 가져오는 방법

LPDIRECT3DVERTEXBUFFER9 vb = nullptr;

mesh->GetVertexBuffer(&vb);

LPDIRECT3DINDEXBUFFER9 ib = nullptr;

mesh->GetIndexBuffer(&ib);

// 버퍼의 읽기, 쓰기를 위해 잠그는 방법 및 해제하는 방법

LVertex \* lvb = nullptr;

mesh->LockVertexBuffer(0, (void\*\*)lvb);

mesh->UnlockVertexBuffer();

LVertex \* lib = nullptr;

mesh->LockIndexBuffer(0, (void\*\*)lib);

mesh->UnlockVertexBuffer();

// FVF 리턴

mesh->GetFVF();

// 버텍스 버퍼 내의 버텍스 수 리턴

mesh->GetNumVertices();

// 버텍스 당 바이트 수 리턴

mesh->GetNumBytesPerVertex();

// 메쉬 내의 삼각형 수를 리턴

mesh->GetNumFaces();

// 속성 버퍼에 접근하는 방법

DWORD \* buffer = nullptr;

DWORD lockFlag = 0;

mesh->LockAttributeBuffer(lockFlag, &buffer);

// 속성 버퍼를 읽거나 쓰는 작업

mesh->UnlockAttributeBuffer();

// 드로잉 - ResourceManager.cpp - C3DModel class - Draw function 참고

// Optimize - 효과적으로 메쉬를 렌더링하기 위한 과정

vector<DWORD> adjacencyInfo;

vector<DWORD> opAdjInfo;

adjacencyInfo.resize(mesh->GetNumFaces());

opAdjInfo.resize(mesh->GetNumFaces());

mesh->GenerateAdjacency(0.0f, &adjacencyInfo[0]);

mesh->OptimizeInplace(

D3DXMESHOPT\_ATTRSORT |

D3DXMESHOPT\_COMPACT |

D3DXMESHOPT\_VERTEXCACHE,

&adjacencyInfo[0],

&opAdjInfo[0],

0, 0);

// 메쉬 속성 테이블에 접근 하는 방법

/\*

D3DXMESHOPT\_ATTRSORT 최적화를 수행하면 기하정보 정렬과는 별도로 속성 테이블이 만들어진다.

속성 테이블은 D3DXATTRIBUTERANGE 구조체 배열로, 테이블의 각 항목은 메쉬의 각 서브셋과 대응되며 서브셋의 기하정보들이 보관되는 버텍스/인덱스 버퍼 내의 메모리 블록을 지정한다.

\*/

// 1. 속성 테이블 내의 요소의 수를 얻는 방법

// 첫번째 인자에 0을 넣는다

DWORD numSubsets = 0;

mesh->GetAttributeTable(0, &numSubsets);

// 2. 속성 테이블을 채우기

D3DXATTRIBUTERANGE \* table = new D3DXATTRIBUTERANGE[numSubsets];

mesh->GetAttributeTable(table, &numSubsets);

// 속성 테이블을 직접 지정하는 것도 가능

D3DXATTRIBUTERANGE attributeRange[12];

// 속성 채우기

mesh->SetAttributeTable(attributeRange, 12);

// 근접정보

// 최적화와 같은 특수한 메쉬 처리를 위해서는 주어진 삼각형과 인접한 다른 삼각형에 대한 정보가 필요하다

// 메쉬의 인접 배열은 바로 이러한 정보를 보관한다.

인접 배열은 DWORD의 배열이며 각 항목은 메쉬 내의 삼각형을 식별하는 인덱스이다.

예를 들어 항목 i를 이용해 다음과 같은 인덱스로 구성되는 삼각형을 정의할 수 있다.

A = i \* 3

B = i \* 3 + 1

C = i \* 3 + 2

특정한 면이 인접한 삼각형을 가지고 있지 않음을 의미하는 항목(ULONG\_MAX = 4294967295)도 있다.

DWORD 이므로 -1을 이용해도 같은 값을 가진다.

삼각형은 세 개의 변을 가지므로 결국 세 개의 인접한 삼각형을 가질 수 있다

따라서 인접 배열은 메쉬 내 모든 삼각형들의 인접한 세 삼각형을 위해

ID3DXMESH::GetNumFaces() \* 3 만큼의 요소를 가져야 한다

인접정보를 생성하는 함수

HRESULT ID3DXMESH::GenerateAdjacency(

FLOAT fEpsilon,

DWORD\* pAdjacency

);

ex)

DWORD adjacencyInfo[mesh->GetNumFaces() \* 3];

mesh->GenerateAdjacency(0.001f,adjacencyInfo);

복제

메쉬를 복사한다

LPD3DXMESH\* clone = nullptr;

mesh->CloneMeshFVF(

mesh->GetOptions(),

D3DFVF\_XYZ | D3DFVF\_NORMAL,

device,

&clone);

메쉬 만들기(D3DXCreateMeshFVF)

이 메서드를 이용하여 비어있는 메쉬를 만드는 것도 가능하다. 빈 메쉬를 만들기 위해서는 메쉬를 구성할 면과 버텍스의 수를 지정하고, 적절한 크기와 버텍스, 인덱스, 속성 버퍼를 할당하면 된다.

메쉬 버퍼가 할당된 다음데는 메쉬의 데이터를 우리가 직접 채워넣어야 한다.

버텍스와 인덱스, 속성 데이터를 버텍스 버퍼와 인덱스 버퍼, 속성 버퍼에 각각 저장해야 한다.

HRESULT D3DXCreateMeshFVF(

DWORD NumFaces, // 메쉬가 가질 면의 수

DWORD NumVertices, // 버텍스의 수

DWORD Options, // 생성 플래그

DWORD FVF, // FVF

LPDIRECT3DDEVICE9 pDevice, // 장치

LPD3DXMESH\* ppMesh); // 반환 패러미터

HRESULT D3DXCreateMesh(

DWORD NumFaces, // 메쉬가 가질 면의 수

DWORD NumVertices, // 버텍스의 수

DWORD Options, // 생성 플래그

CONST LPD3DVERTEXELEMENT9\* pDeclaration, // 17장에서 다룸

LPDIRECT3DDEVICE9 pDevice, // 장치

LPD3DXMESH\* ppMesh); // 반환 패러미터

아래의 함수는 입력받는 FVF에 대응되는 D3DVERTEXELEMENT9 구조체의 배열을 리턴한다

HRESULT D3DXDeclaratorFromFVF(

DWORD FVF,

D3DVERTEXELEMENT9 Declaration[MAX\_FVF\_DECL\_SIZE] // 출력 포멧

);