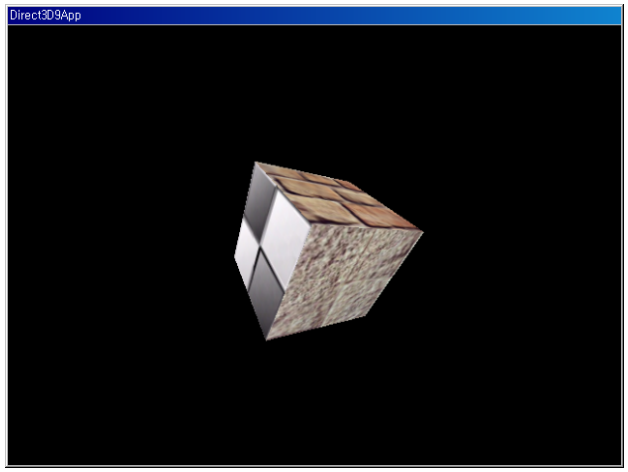
Mesh

305890
2008년 봄학기
5/21/2007
박경신

Overview

- ID3DXMesh 객체의 내부 데이터 구성
- ID3DXMesh의 생성 방법
- ID3DXMesh의 최적화 방법
- ID3DXMesh의 렌더링 방법

Mesh Part I

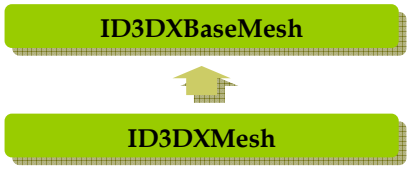


Mesh Part I

- Geometry Information
- Subsets and the Attribute buffer
- Drawing
- Optimizing
- The Attribute Table
- Adjacency Information
- D3DXCreateMeshFVF
- 예제

Geometry Information

- ID3DXMesh 인터페이스는 기능의 상당 부분을 부모인 ID3DXBaseMesh에서 상속 받음



Geometry Information

- ID3DXMesh의 버텍스 / 인덱스 버퍼의 포인터를 얻는 함수

```
HRESULT ID3DXMesh::GetVertexBuffer(LPDIRECT3DVERTEXBUFFER9*
    ppVB);
HRESULT ID3DXMesh::GetIndexBuffer(LPDIRECT3DINDEXBUFFER9*
    ppIB);

IDirect3DVertexBuffer9* vb = 0;
Mesh->GetVertexBuffer( &vb );
IDirect3DIndexBuffer9* ib = 0;
Mesh->GetIndexBuffer( &ib );
```

Geometry Information

- 쓰기나 읽기를 위해 버퍼에 잠금 / 해제 함수

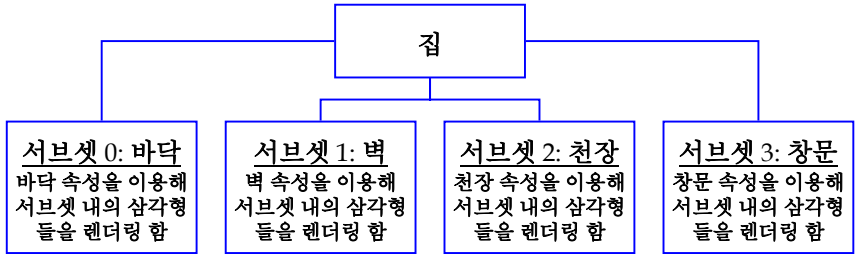
```
HRESULT ID3DXMesh::LockVertexBuffer(DWORD Flags,BYTE** ppData);
HRESULT ID3DXMesh::LockIndexBuffer(DWORD Flags, BYTE** ppData);

HRESULT ID3DXMesh::UnlockVertexBuffer( );
HRESULT ID3DXMesh::UnlockIndexBuffer( );
```

- 기하 관련 정보를 얻기 위한 부가적인 함수들
  - DWORD ID3DXMesh::GetFVF( ); // vertex format
  - DWORD ID3DXMesh::GetNumVertices( ); // num of vertices
  - DWORD ID3DXMesh::GetNumBytesPerVertex( ); // bytes per vertex
  - DWORD ID3DXMesh::GetNumFaces( ); // num of faces

Subset and Attribute Buffer

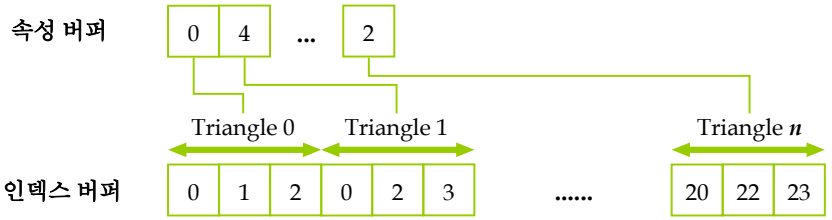
- 서브셋 (Subset)
  - 동일한 속성을 이용해 렌더링할 수 있는 메쉬 내 삼각형들의 그룹
  - 예를 들어, 하나의 기하물체 (집)를 여러 속성 (바닥, 벽, 천정, 창문)으로 렌더링할 때 속성 수만큼의 subset을 구성함
- 속성 (Attribute)
  - 재질, 텍스처, 렌더 상태



< 서브셋으로 분리된 집 >

Subset and Attribute Buffer

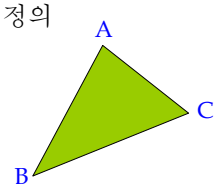
- 속성 ID (Attribute ID)
  - 각각의 서브셋에는 유일한 양의 정수 값을 지정하여 서로 구분
  - 각 삼각형은 자신이 속한 서브셋의 속성 ID를 가짐
- 속성 버퍼 (Attribute buffer)
  - 삼각형의 속성 ID를 저장하는 DWORD 배열
  - 속성 버퍼의 요소 개수 == 메쉬의 삼각형(면) 개수



Subset and Attribute Buffer

- 1:1 대응
  - 속성 버퍼의 항목 i 는 인덱스 버퍼의 삼각형 i 와 대응
  - 삼각형 i 는 인덱스 버퍼의 다음 세 개의 인덱스로 정의

$$A = i \cdot 3$$
$$B = i \cdot 3 + 1$$
$$C = i \cdot 3 + 2$$



- 속성 버퍼에 접근하기 위해, 먼저 버퍼를 잠금

```
DWORD *buffer = 0;
Mesh->LockAttributeBuffer(lockingFlags, &buffer);

// read or write to attribute buffer...
Mesh->UnlockAttributeBuffer( );
```

Mesh Drawing

- DrawSubset
  - Attrib ID 로 지정한 특정 서브셋의 삼각형을 그리는 함수

```
HRESULT ID3DXMesh::DrawSubset(DWORD AttribId);
```

  - 예제: 서브셋 0에 존재하는 모든 삼각형을 그림

```
Mesh->DrawSubset(0);
```

  - 예제: 전체 메쉬를 그리기 위해, 모든 서브셋을 그림

```
for (int i=0; i<numSubsets; i++)
{
   Device->SetMaterial( mtrls[i] );
   Device->SetTexture( 0, textures[i] );
   Mesh->DrawSubset(i);
}
```

Mesh Optimizing

- 최적화 - 좀 더 효과적으로 메쉬를 렌더링 하기 위해 버텍스와 인덱스를 재구성

```
HRESULT ID3DXMesh::OptimizeInplace (
   DWORD Flags,
   CONST DWORD* pAdjacencyIn,
   DWORD* pAdjacencyOut,
   DWORD* pFaceRemap,
   LPD3DXBUFFER* ppVertexRemap);
```

  - Flags – 수행할 최적화의 종류를 알려주는 플래그
  - pAdjacencyIn – 최적화 되지 않은 메쉬의 인접 배열
  - pAdjacencyOut – 최적화 된 메쉬의 인접 정보 배열. 불필요하면 NULL.
  - pFaceRemap – 인덱스 버퍼에서 이동한 면 리맵 정보. i번째 항목은 원래의 i번째 face가 몇 번째로 이동했는지 알려줌. D3DXMesh :: GetNumFaces() 크기. 불필요하면 NULL.
  - ppVertexRemap – 버텍스 버퍼에서 이동한 버텍스 리맵 정보. I번째 항목은 i번째 vertex가 어디로 이동했는지 알려줌. ID3DXMesh :: GetNumVertices() 크기. 불필요하면 NULL.

Mesh Optimizing

- Flags – D3DXMESHOPT flags
  - D3DXMESHOPT_COMPACT – 메쉬에서 이용되지 않은 인덱스와 버텍스를 제거한다.
  - D3DXMESHOPT_ATTRSORT – 속성으로 삼각형을 정렬하고 속성 테이블을 생성한다. 이 플래그는 DrawSubset의 효율을 높여준다.
  - D3DXMESHOPT_VERTEXCACHE – 버텍스 캐시의 히트율을 높인다. (권장)
  - D3DXMESHOPT_STRIPREORDER – 삼각형 스트립이 가능한 길어지도록 인덱스를 재구성한다.
  - D3DXMESHOPT_IGNOREVERTES – 버텍스를 무시하고 인덱스 정보만을 최적화한다.

Mesh Optimizing

- 예제:

```
// 최적화되지 않은 mesh의 adjacency 정보를 얻음
vector<DWORD> adjacencyInfo(Mesh->GetNumFaces()*3);
Mesh->GenerateAdjacency( 0.0f, &adjacencyInfo[0] );

// 최적화된 adjacency 정보를 보관할 배열
vector<DWORD> optimizedAdjacencyInfo(Mesh->GetNumFaces()*3);
Mesh->OptimizeInplace(
    D3DXMESHOPT_ATTRSORT | D3DXMESHOPT_COMPACT |
    D3DXMESHOPT_VERTEXCACHE,
    &adjacencyInfo[0],
    &optimizedAdjacencyInfo[0],
    0,
    0 );
```

Mesh Optimizing

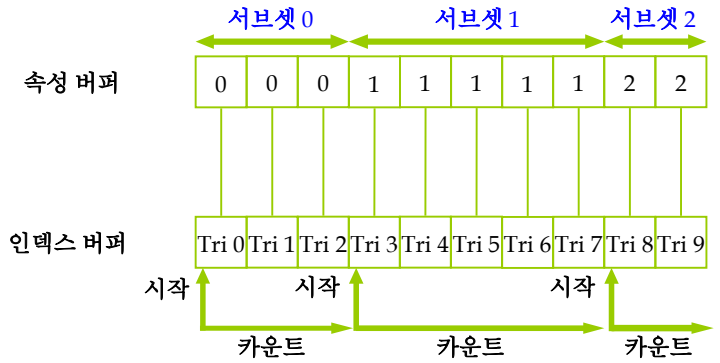
- 비슷한 역할의 함수 – Optimize()
  - ID3DXBaseMesh::CloneMesh 방법과 가장 비슷함
  - 차이점 – 최적화된 버전의 메쉬 객체를 리턴

```
HRESULT ID3DXMesh::Optimize (
    DWORD Flags,
    CONST DWORD* pAdjacencyIn,
    DWORD* pAdjacencyOut,
    DWORD* pFaceRemap,
    LPD3DXBUFFER* ppVertexRemap,
    LPD3DXMESH* ppOptMesh);
```

  - ppOptMesh – 최적화된 메쉬

Attribute Table

- D3DXMESHOPT_ATTRSORT 플래그를 설정하여 메쉬를 최적화
  - 메쉬의 기하정보가 속성에 따라 정렬
  - 속성 테이블 생성



Attribute Table

- D3DXATTRIBUTERANGE 구조체의 배열
  - 테이블의 각 항목은 메쉬의 각 서브셋과 대응
  - 속성 테이블을 검색하면 특정 서브셋 내의 모든 기하 정보를 효율적으로 찾을 수 있음 (선형 검색 불필요)

```
typedef struct _D3DXATTRIBUTERANGE {
    DWORD AttribId;
    DWORD FaceStart;
    DWORD FaceCount;
    DWORD VertexStart;
    DWORD VertexCount;
} D3DXATTRIBUTERANGE
```

  - AttribId – 서브셋 ID
  - FaceStart, FaceCount – 서브셋에 연결된 삼각형들의 시작 위치 (인덱스 버퍼)와 개수
  - VertexStart, VertexCount – 서브셋에 연결된 버텍스들의 시작 위치 (버텍스 버퍼)와 개수

Attribute Table

- 메쉬의 속성 테이블에 접근하는 함수

```
HRESULT ID3DXMesh::GetAttributeTable (
        D3DXATTRIBUTERANGE* pAttribTable,
        DWORD* pAttribTableSize);
```

  - 속성 테이블 내의 속성의 수를 리턴
  - D3DXATTRIBUTERANGE 구조체 배열 return
- 예제:

```
// attribute table의 entry 수를 알아내기
DWORD numSubsets = 0;
Mesh->GetAttributeTable( 0, &numSubsets );
// attribute table 가져오기
D3DXATTRIBUTERANGE table =
                        new D3DXATTRIBUTERANGE[numSubsets];
Mesh->GetAttributeTable( table, &numSubsets);
```

Attribute Table

- 속성 테이블을 직접 지정하는 함수

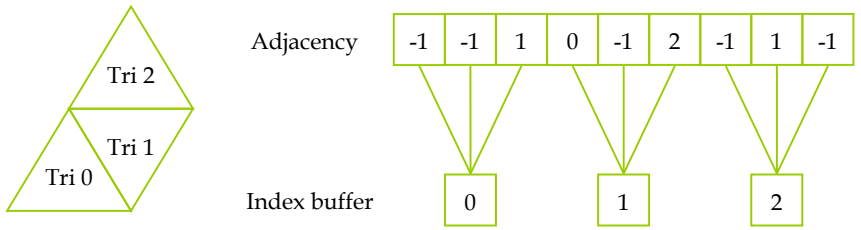
```
HRESULT ID3DXMesh:: SetAttributeTable(
    CONST D3DXATTRIBUTERANGE* pAttribTable,
    DWORD cAttribTableSize);
```

- 예제:

```
// attribute table을 수동으로 지정하기
D3DXATTRIBUTERANGE attributeTable[12];
// fill attributeTable array with data ...
Mesh->SetAttributeTable( attributeTable, 12 );
```

Adjacency Array

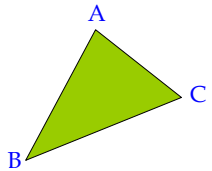
- 인접 배열 (Adjacency array)
  - 최적화와 같은 특수한 메쉬 처리를 위해 필요
  - 주어진 삼각형과 인접한 다른 삼각형에 대한 정보
  - 삼각형을 식별하는 인덱스를 포함하는 DWORD array



Adjacency Array

- i 번째 삼각형은 다음과 같은 인덱스로 구성:

$$A = i \cdot 3$$
$$B = i \cdot 3 + 1$$
$$C = i \cdot 3 + 2$$



- 인접한 삼각형을 갖지 않는 특정한 모서리(edge)
  - DWORD (unsigned 32-bit integer) 이므로 -1를 DWORD에 할당하면 ULONG_MAX가 됨
  - ULONG_MAX == 4,294,967,295 == –1
- 인접 배열은 하나의 삼각형마다 세 개의 인접한 삼각형을 가져야 함
  - 요소 개수 = ID3DXBASEMESH::GetNumFaces( ) * 3

Adjacency Array

- 인접 정보를 생성하는 함수
  - 많은 D3DX의 mesh 생성함수들이 adjacency 정보를 생성해 줌.
  - 수동으로 생성하려면 GenerateAdjacency 함수를 이용

```
HRESULT ID3DXMesh::GenerateAdjacency (
        FLOAT fEpsilon,
        DWORD* pAdjacency);
```

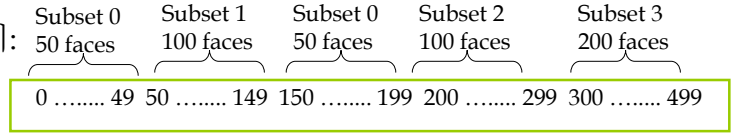
  - fEpsilon – 두 개의 포인트를 동일한 점으로 취급할 거리 근사값
  - pAdjacency – 인접 정보로 채워질 DWORD 배열. Bytes는 3 * ID3DXMesh::GetNumFaces*sizeof(DWORD)

- 예제:

```
vector<DWORD> adjacencyInfo(Mesh->GetNumFaces()*3);
Mesh->GenerateAdjacency( 0.001f, &adjacencyInfo[0] );
```

Subset Example

- 한 메쉬를 다른 크기의 여러 서브셋으로 나누는 예제 (한 subset의 face들이 연속해 있을 필요 없음)
- 예제:




```
DWORD faceCount[] = {50, 100, 50, 100, 200, 0};
//마지막 0은 목록의 마지막임을 표시
DWORD subsetNum[] = {0, 1, 0, 2, 3};
if(FAILED(SetSubsets(pMesh, faceCount, subsetNum)) {
    // handle error
}
```

Subset Example

```
DWORD SetSubsets(ID3DXMesh* pMesh, DWORD *pFaceCount,
                 DWORD *pSubsetNum) {
    // attribute buffer의 최대크기를 알기 위해 사용
    DWORD numFaces = pMesh->GetFaceCount(); // get face count
    DWORD *attribBuf;
    HRESULT hr;
    if (SUCCEEDED (hr=pMesh->LockAttributeBuffer (D3DLOCK_DISCARD,
    &attribBuf))) {
        DWORD faceNum = 0; // initialize face counter
        for (int i=0; pFaceCount[i]; i++) { // loop through the subsets
            // make sure there are enough faces for this subset
            if (faceNum + pFaceCount[i] >= numFaces) { // not enough faces
              pMesh->UnlockAttributeBuffer(); // unlock attribute buffer
              return E_INVALIDARG; // return err
            }
            for (int j=0; j<pFaceCount[i]; j++) {
               attribBuf[faceNum] = pSubsetNum[i]; // set subset number of each face
               faceNum++; // increase face counter
            }
        }
```

Subset Example

```
        pMesh->UnlockAttributeBuffer(); // unlock attribute buffer
        // allocate storage and generate adjacency data
        // 이미 adjacency data가 있다면 다시 생성할 필요 없음
        DWORD *pAdj = new DWORD[numFaces*3];
        if (!pAdj) return E_OUTOFMEMORY;
        if (FAILED(hr = pMesh->GenerateAdjaceny(0.0f, pAdj))) {
            delete pAdj; return hr;
        }
        // optimize the mesh with attribute D3DXMESHOPT_ATTRSORT
        if (FAILED(hr = pMesh->OptimizeInplace(
                D3DXMESHOPT_VERTEXCACHE, pAdj, NULL, NULL, NULL))) {
            delete pAdj; return hr;
        }
        delete pAdj; // de-allocate adjacency data storage
    }
    else
        return hr;
    return S_OK; // return success
}
```

Cloning

- 메쉬의 데이터를 다른 곳으로 복사하는 함수. 주용도는 vertex data를 새로 포맷하거나 바꾸고자 할 때 사용된다.

```
HRESULT ID3DXMesh::CloneMeshFVF (
    DWORD Options,
    DWORD FVF,
    LPDIRECT3DDEVICE9 pDevice,
    LPD3DXMESH* ppCloneMesh);
```

  - Options – 복제된 메쉬를 만드는데 이용될 플래그
  - FVF – 원본 메쉬와 다른 FVF와 옵션을 가질 수 있음
  - pDevice – 복제된 메쉬와 연계될 장치
  - ppCloneMesh – 복제된 메쉬

Cloning

- Options – 복제된 메쉬를 만드는데 이용될 하나 이상의 플래그이다. 자주 이용되는 플래그는 아래와 같다.
  - D3DXMESH_32BIT – 메쉬가 32-bit index를 이용하도록 함
  - D3DXMESH_MANAGED – 메쉬가 managed pool 내에 보관되도록 함
  - D3DXMESH_WRITEONLY – 메쉬 데이터에 쓰기만 허용
  - D3DXMESH_DYNAMIC – 메쉬 버퍼가 동적으로 만들어지도록 함

- 예제:

```
ID3DMesh* clone = 0;
Mesh->CloneMeshFVF(Mesh->GetOptions(), // 원본 메쉬와 동일옵션
        D3DFVF_XYZ | D3DFVF_NORMAL, // cloned mesh FVF
        Device, &clone);
```

Creating Mesh

- Direct3D에서 Mesh object (ID3DXMesh)를 생성하는 방법의 분류
  - Shape creation – 여러 개의 기본요소를 만들어내는 함수
    - D3DXCreateBox, D3DXCreateTeapot, ..
  - Basic mesh creation – 특정 포멧과 크기를 가진 메쉬 객체 생성 함수
    - D3DXCreateMesh, D3DXCreateMeshFVF
  - Mesh file – X file로부터 메쉬 객체를 읽어 들이는 함수
    - D3DXLoadMeshFromX
  - Mesh operations – 기존의 메쉬로 부터 새로운 메쉬 생성 함수
    - OptimizeInplace, Optimize, CloneMeshFVF

Creating Mesh – Shape Creation

- Shape Creation

D3DXCreateBox(LPDIRECT3DDEVICE9 pDevice, FLOAT Width, FLOAT Height, FLOAT Depth, LPD3DXMESH **ppMesh, LPD3DXBUFFER **ppAdjacency);
D3DXCreateCylinder(LPDIRECT3DDEVICE9 pDevice, FLOAT Radius1, FLOAT Radius2, FLOAT Length, UINT Slices, UINT Stacks, LPD3DXMESH **ppMesh, LPD3DXBUFFER **ppAdjacency);
D3DXCreatePolygon(LPDIRECT3DDEVICE9 pDevice, FLOAT Length, UINT Sides, LPD3DXMESH **ppMesh, LPD3DXBUFFER **ppAdjacency);
D3DXCreateSphere(LPDIRECT3DDEVICE9 pDevice, FLOAT Radius, UINT Slices, UINT Stacks, LPD3DXMESH **ppMesh, LPD3DXBUFFER **ppAdjacency);
D3DXCreateTeapot(LPDIRECT3DDEVICE9 pDevice, LPD3DXMESH **ppMesh, LPD3DXBUFFER **ppAdjacency);
D3DXCreateText(LPDIRECT3DDEVICE9 pDevice, HDC hDC, LPCTSTR pText, FLOAT Deviation, FLOAT Extrusion, LPD3DXMESH **ppMesh, LPD3DXBUFFER **ppAdjacency, LPGLYPHMETRICSFLOAT pGlyphMetrics);
D3DXCreateTorus(LPDIRECT3DDEVICE9 pDevice, FLOAT InnerRadius, FLOAT OuterRadius, UINT Sides, UINT Rings, LPD3DXMESH **ppMesh, LPD3DXBUFFER **ppAdjacency);

Creating Mesh – Basic Mesh Creation

- Basic Mesh Creation – 비어 있는 mesh를 만들 수 있음.
  1. 메쉬를 구성할 면 (face)와 정점 (vertex)의 개수를 결정
  2. D3DXCreateMeshFVF에 적절한 크기의 vertex/index/attribute buffer를 할당
  3. 각각의 메쉬 버퍼에 메쉬의 data를 직접 입력

- 주어진 면과 버텍스 개수를 갖는 메쉬 생성 함수

```
D3DXCreateMeshFVF(DWORD NumFaces, // index buffer 크기 결정
                  DWORD NumVertices, // vertex buffer 크기 결정 (1이상)
                  DWORD Options, // D3DXMESH flag
                  DWORD FVF, // FVF flag
                  LPDIRECT3DDEVICE9 pD3DDevice, // IDirect3DDevice9
                  LPD3DXMESH *ppMesh); // ID3DXMesh
```

Creating Mesh – Basic Mesh Creation

- 비슷한 역할 함수: D3DXCreateMesh
  - 차이점 – **FVF** 대신 **D3DVERTEXELEMENT9** 구조체 배열 사용

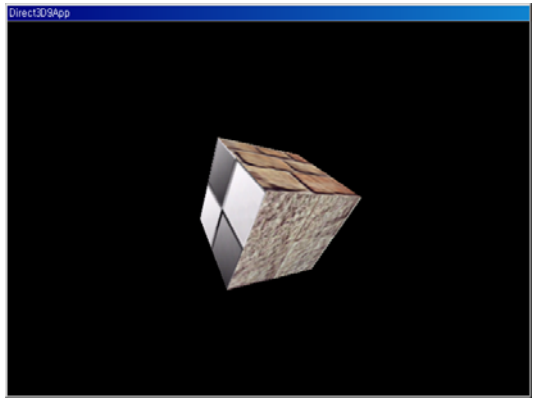
```
D3DXCreateMesh(DWORD NumFaces, DWORD NumVertices,
               DWORD Options,
               const LPD3DVERTEXELEMENT9 *pDeclaration,
               LPDIRECT3DDEVICE9 pD3DDevice,
               LPD3DXMESH **ppMesh);
```

  - pDeclaration
    - D3DVERTEXELEMENT9 구조체 배열로 메쉬 vertex format을 결정

- pDeclaration을 얻는 방법: D3DXDeclarationFromFVF

```
HRESULT D3DXDeclaratorFromFVF (DWORD FVF,
        D3DVERTEXELEMENT9 Declaration[MAX_FVF_DECL_SIZE]);
typedef enum { MAX_FVF_DECL_SIZE = 18 } MAX_FVF_DECL_SIZE;
```

Example: D3DXCreateMeshFVF



1. 빈 메쉬 만들기
   - D3DXMesh
2. 상자의 기하 정보로 메쉬 채움
   - Vertex, Index buffer
3. 메쉬의 각 면이 존재하는 서브셋 지정
   - Attribute buffer
4. 메쉬의 인접 정보 생성
   - Adjacency buffer
5. 메쉬의 최적화
   - Optimize
6. 메쉬의 드로잉
   - Draw subsets

Example: D3DXCreateMeshFVF

- 메쉬 요소의 디버깅과 검사를 위해 메쉬 내용을 파일로 덤프

```
void dumpVertices(std::ofstream& outFile, ID3DXMesh* mesh);
void dumpIndices(std::ofstream& outFile, ID3DXMesh* mesh);
void dumpAttributeBuffer(std::ofstream& outFile, ID3DXMesh* mesh);
void dumpAdjacencyBuffer(std::ofstream& outFile, ID3DXMesh* mesh);
void dumpAttributeTable(std::ofstream& outFile, ID3DXMesh* mesh);

void dumpVertices(std::ofstream& outFile, ID3DXMesh* mesh) {
    outFile << "Vertices:" << std::endl;
    outFile << "---------" << std::endl << std::endl;
    Vertex* v = 0;
    mesh->LockVertexBuffer(0, (void**)&v);
    for(int i = 0; i < mesh->GetNumVertices(); i++) {
            outFile << "Vertex " << i << ": (";
            outFile << v[i]._x  << ", " << v[i]._y
                    << ", " << v[i]._z  << ", ";
            outFile << v[i]._nx << ", " << v[i]._ny
                    << ", " << v[i]._nz << ", ";
            outFile << v[i]._u  << ", " << v[i]._v
                    << ")"  << std::endl;
    }
    mesh->UnlockVertexBuffer();
    outFile << std::endl << std::endl;
}
```

Example: D3DXCreateMeshFVF

```
#include "d3dUtility.h"
#include <vector>
ID3DXMesh* Mesh = 0;
const DWORD NumSubsets = 3;
IDirect3DTexture9* Textures[3] = {0, 0, 0};     // texture for each subset
std::ofstream OutFile; // for mesh data dump
struct Vertex {
    Vertex() {}
    Vertex(float x, float y, float z, float nx, float ny, float nz, float u, float v) {
        _x=x; _y=y; _z=z; _nx=nx; _ny=ny; _nz=nz; _u=u; _v=v; }
    float _x, _y, _z, _nx, _ny, _nz, _u, _v;
    static const DWORD FVF;
};
const DWORD Vertex::FVF = D3DFVF_XYZ|D3DFVF_NORMAL|D3DFVF_TEX1;
void Cleanup() {
    d3d::Release<ID3DMesh*>(Mesh);
    for (i=0; i<3; i++) d3d::Release<IDirect3DTexture9*>(Textures[i]);
}
```

Example: D3DXCreateMeshFVF

```
bool Setup() {
    HRESULT hr = 0;
    hr = D3DXCreateMeshFVF(12, 24, D3DXMESH_MANAGED, Vertex::FVF,
    Device, &Mesh); // 12 triangles & 24 vertices
    if (FAILE(hr)) {
        ::MessageBox(0, "D3DXCreateMeshFVF() – FAILED", 0, 0); return false; }
    Vertex* v  = 0;
    Mesh->LockVertexBuffer(0, (void**) &v);
    v[0] = Vertex(-1.0f, -1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 0.0f, 0.0f); // front
    v[1] = Vertex(-1.0f,  1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 0.0f, 1.0f);
    v[2] = Vertex( 1.0f,  1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 1.0f, 1.0f);
    v[3] = Vertex( 1.0f, -1.0f, -1.0f, 0.0f, 0.0f, -1.0f, 1.0f, 0.0f);
    v[4] = Vertex(-1.0f, -1.0f,  1.0f, 0.0f, 0.0f,  1.0f, 0.0f, 0.0f); // back
    …..
    v[20] = Vertex( 1.0f, -1.0f, -1.0f, 1.0f, 0.0f,  0.0f, 0.0f, 0.0f); // right
    v[21] = Vertex( 1.0f,  1.0f, -1.0f, 1.0f, 0.0f,  0.0f, 0.0f, 1.0f);
    v[22] = Vertex( 1.0f,  1.0f,  1.0f, 1.0f, 0.0f,  0.0f, 1.0f, 1.0f);
    v[23] = Vertex( 1.0f, -1.0f,  1.0f, 1.0f, 0.0f,  0.0f, 1.0f, 0.0f);
    Mesh->UnlockVertexBuffer();
```

Example: D3DXCreateMeshFVF

```
    WORD* i = 0;
    Mesh->LockIndexBuffer(0, (void**) &i); // index data
    i[0] = 0;   i[1] = 1;     i[2] = 2;     i[3] = 0;    i[4] = 2;     i[5] = 3; // front
    i[6] = 4;   i[7] = 5;     i[8] = 6;     i[9] = 4;    i[10] = 6;    i[11] = 7; // back
    i[12] = 8; i[13] = 9;    i[14] = 10;  i[15] = 8;  i[16] = 10;  i[17] = 11; // top
    ….
    i[30] = 20; i[31] = 21; i[32] = 22; i[33] = 20; i[34] = 22; i[35] = 23; // right
    Mesh->UnlockIndexBuffer();
    DWORD* attributeBuffer = 0; // specify the subset
    Mesh->LockAttributeBuffer(0, &attributeBuffer);
    for (int a=0; a<4; a++) attributeBuffer[a] = 0; // first two faces – subset0
    for (int b=4; b<8; b++) attributeBuffer[b] = 1; // next two faces – subset1
    for (int c=8; c<12; c++) attributeBuffer[c] = 2; // last two faces – subset2
    Mesh->UnlockAttributeBuffer();
```

Example: D3DXCreateMeshFVF

```
// optimize the mesh to generate an attribute table
std::vector<DWORD> adjacencyBuffer(Mesh->GetNumFaces() * 3);
Mesh->GenerateAdjacency(0.0f, &adjacencyBuffer[0]);
hr = Mesh->OptimizeInplace(D3DXMESHOPT_ATTRSORT |
      D3DXMESHOPT_COMPACT | D3DXMESHOPT_VERTEXCACHE,
      &adjacencyBuffer[0], 0, 0, 0);
// dump the mesh data to file
OutFile.open("MeshDump.txt");
dumpVertices(OutFile, Mesh);
dumpIndices(OutFile, Mesh);
dumpAttributeTable(OutFile, Mesh);
dumpAttributeBuffer(OutFile, Mesh);
dumpAdjacencyBuffer(OutFile, Mesh);
OutFile.close();
// load textures
D3DXCreateTextureFromFile(Device, "brick0.jpg", &Textures[0]);
D3DXCreateTextureFromFile(Device, "brick1.jpg", &Textures[1]);
D3DXCreateTextureFromFile(Device, "checker.jpg", &Textures[2]);
Device->SetSamplerState(0, D3DSAMP_MAGFILTER, D3DTEXF_LINEAR);
Device->SetSamplerState(0, D3DSAMP_MINFILTER, D3DTEXF_LINEAR);
Device->SetSamplerState(0, D3DSAMP_MIPFILTER, D3DTEXF_POINT);
```

Example: D3DXCreateMeshFVF

```
// disable lighting
Device->SetRenderState(D3DRS_LIGHTING, false);
// set camera
D3DXVECTOR3 pos(0.0f, 0.0f, -4.0f);
D3DXVECTOR3 target(0.0f, 0.0f, 0.0f);
D3DXVECTOR3 up(0.0f, 1.0f, 0.0f);
D3DXMATRIX V;
D3DXMatrixLookAtLH(&V, &pos, &target, &up);
// set projection matrix
D3DXMatrix proj;
D3DXMatrixPerspectiveFovLH(&proj, D3DX_PI * 0.5f,
                           (float)Width / (float) Height, 1.0f, 1000.0f);
Device->SetTransform(D3DTS_PROJECTION, &proj);
return true;
}
```

Example: D3DXCreateMeshFVF

```
bool Display(float timeDelta) {
    if (Device) {
        D3DXMATRIX xRot, yRot, World;
        static float y = 0.0f;
        D3DXMatrixRotationX(&xRot, D3DX_PI * 0.2f);
        D3DXMatrixRotationY(&yRot, y);
        y += timeDelta;
        if (y >= 6.28f) y = 0.0f;
        World = xRot * yRot;
        Device->SetTransform(D3DTS_WORLD, &World);
        Device->Clear(0, 0, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER,
                      0x00000000, 1.0, 0);
        Device->BeginScene();
        for (int i=0; i < NumSubsets; i++) {
            Device->SetTexture(0,  Textures[i]);
            Mesh->DrawSubset(i);
        }
        Device->EndScene();
        Device->Present(0, 0, 0, 0);
    } return true; }
```