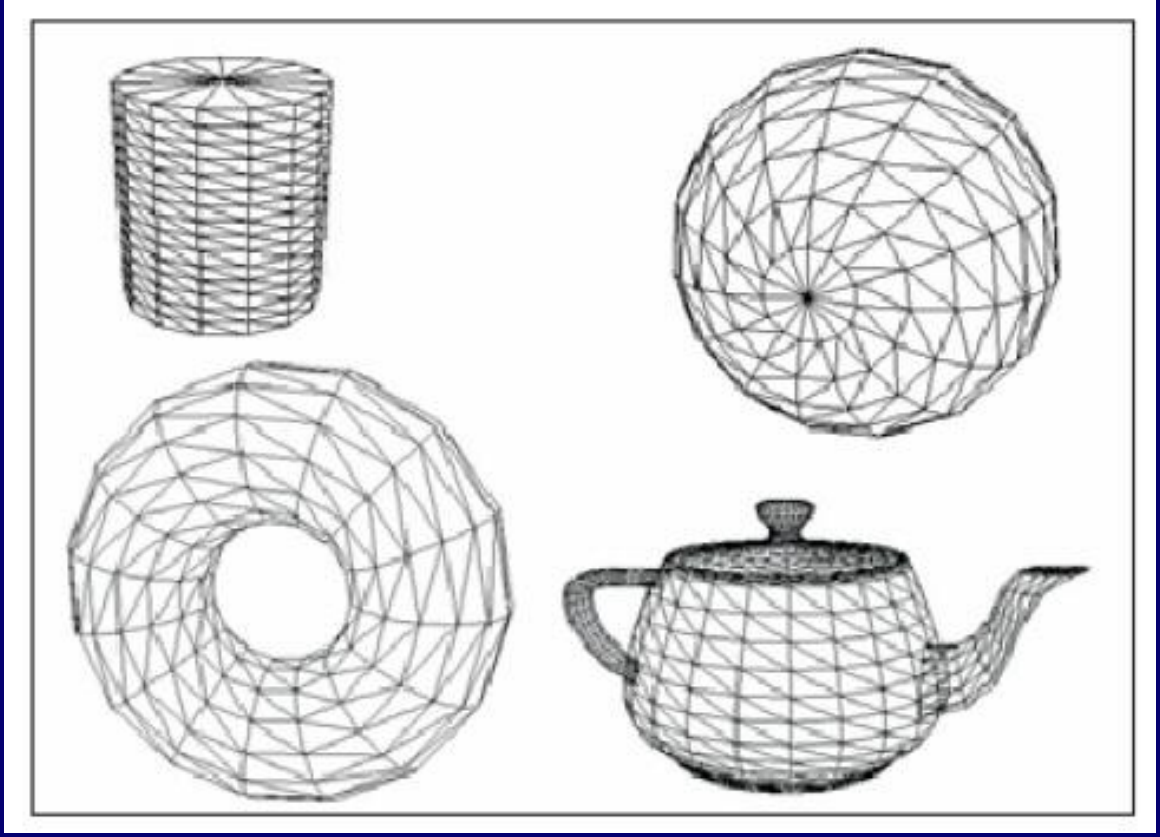
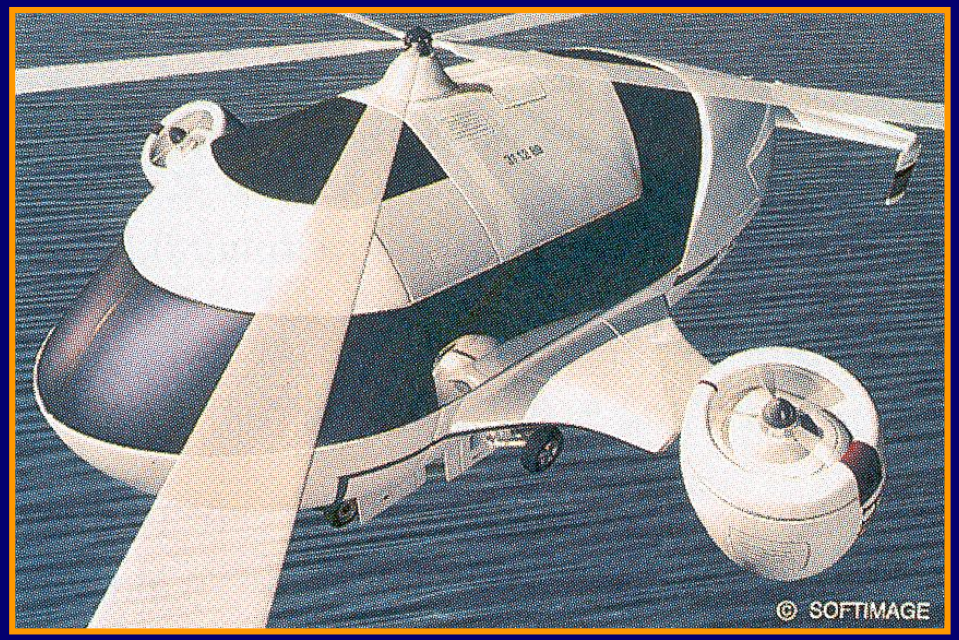
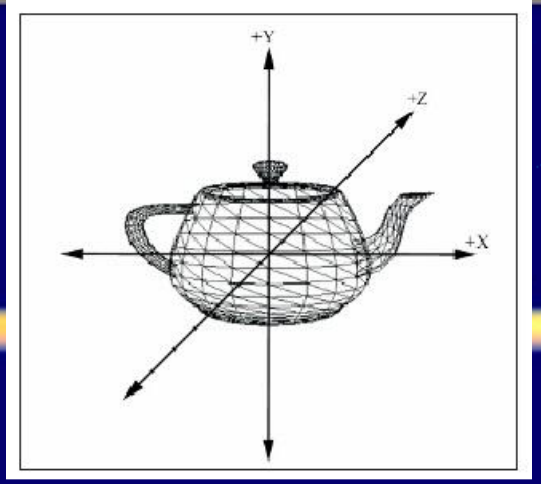
第十一章 网格及三维模型绘制



绘制复杂物体的过程

- 怎样将现实中的一个物体，比如，一只花瓶，一个足球，甚至一架大的战斗机，在电脑屏幕上显示呢？



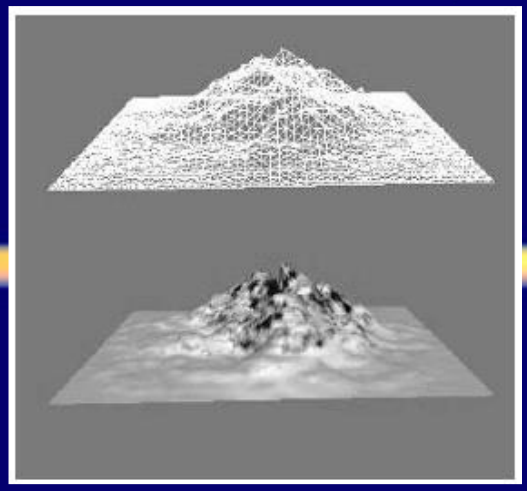


1. 3D建模

- 先把该物体放在一个虚拟的三维坐标系中，该坐标称为局部坐标系(Local Space), 一般以物体的中心作为坐标原点，采用左手坐标系。

  然后，对坐标系中的物体进行点采样, 这些采样点按一定顺序连接成为一系列的小平面, 这些小平面称为图元(Primitive), 3D引擎会处理每一个图元，称为一个独立的渲染单位。这样取样后的物体看起来像是由许许多多的三角形，四边形或五边形组成的，就像网一样，我们称为一个网格(Mesh).

- 这个采样过程又可称为物体的3D建模, 当然现在都有功能非常强大的3D建模工具，例如3D Max。



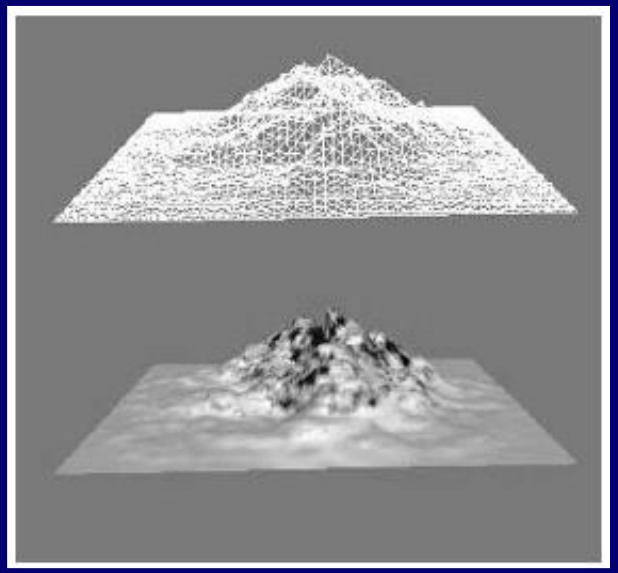
2. 绘制

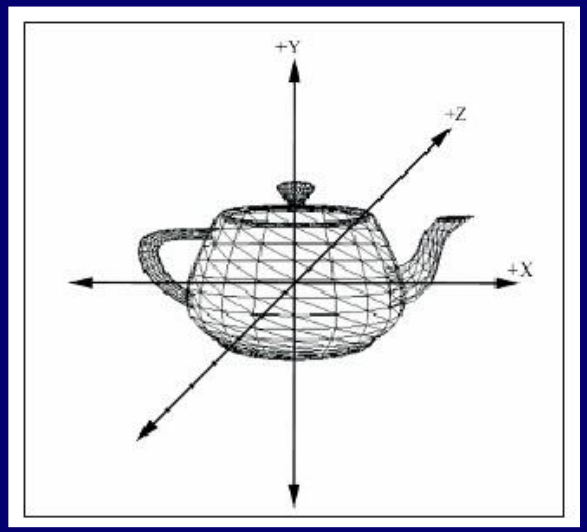
- 我们纪录这些顶点数据和连线情况到一个文件中，3D引擎读取这些数据，依次渲染每一个图元，就能在显示屏幕上再现物体。

- 在D3D中，纪录这些顶点数据和连线情况的文件称为X文件(X File)。它是以X作为文件名后缀的。

D3D中的网格(Mesh)

- 网格也是由一系列共面多边形组成，即由一个个的图元组成，所以有的3D图形学书上，也把网格称为图元链表(Primitive List). 一个物体就可以表示成一个多边形的网格。



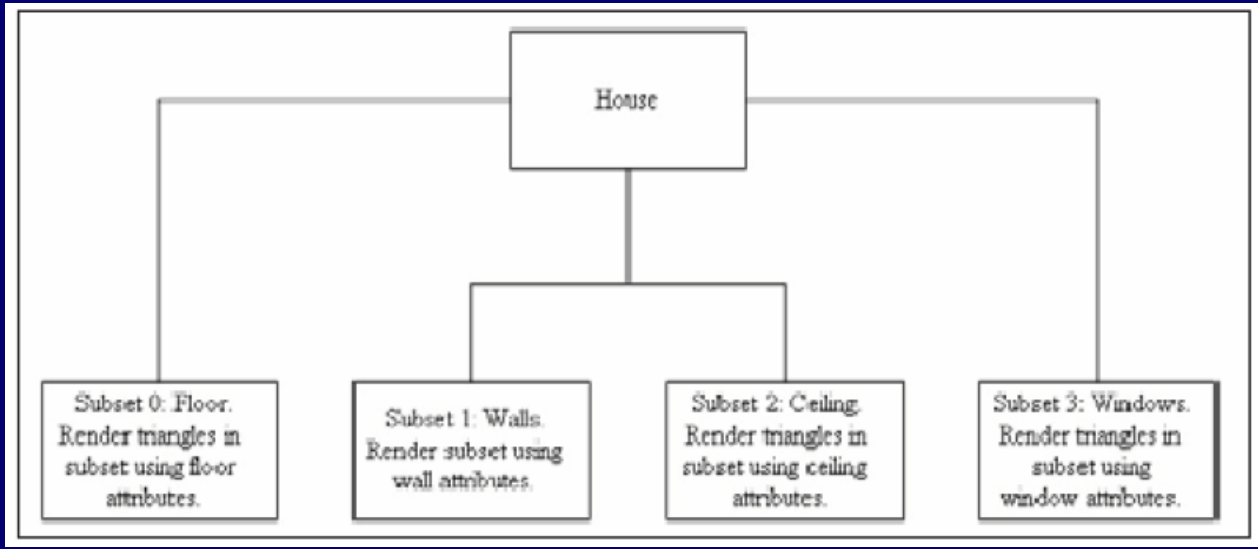


- 1. 网格中包含的内容
  - 网格中主要存储的是对物体的点采样信息，所以一般会有物体采样的顶点个数，顶点数据列表；面个数，面的数据列表的信息。然后，为了增加采样物体的真实性，我们还会加入物体的纹理，材质，灯光等信息。

  - template Mesh {
    <3D82AB44-62DA-11cf-AB39-0020AF71E433> // 给该模块定义的UUID
    DWORD nVertices; // 采样的顶点数
    array Vector vertices[nVertices]; // 顶点数据列表
    DWORD nFaces; // 面个数
    array MeshFace faces[nFaces]; //面的数据列表
    [...] // 一些其他的信息，如：纹理，材质等
    }

- 2. D3D中网格的处理
  - 在微软DirectX8.0 中，提供了几个接口来对网格进行处理，其中最常用的接口是ID3DXMesh。我们可以使用该接口来装载网格数据，获得网格的各种信息等。
  - ID3DXMesh接口的主要功能继承自ID3DXBaseMesh父接口。了解这些是很重要的，其它一些mesh接口如ID3DXPMesh也是继承自ID3DXBaseMesh。

- 一个mesh由一个或数个子集组成
  - 一个子集（subset）是在mesh中的使用相同属性渲染的一组三角形。这里的属性是指材质，纹理和渲染状态。
  - 一座房子mesh可能被分成的几个子集。



- 我们通过给每个子集指定一个唯一非负整数来标识子集。这个值可以是存储在一个DWORD中的任意数值。例如，在图10.1中我们用0，1，2和3来标识子集。

- 在mesh中的每个三角形都与一个属性ID相关联，表示该三角形属于该子集。例如，图10.1中组成地板的三角形具有属性ID0，它表示这些三角形属于子集0。同样，组成墙的三角形具有属性ID1，它表示这些三角形属于子集1。

- 三角形的属性ID存储在mesh的属性缓存中，它是一个DWORD数组.

- 3. 绘制
  - ID3DXMesh接口提供了DrawSubset（DWORD AttribId）方法来绘制AttribId指示的子集中的各个三角形;
  - 要绘制子集0中的所有三角形，我们将这样写：

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

  - 为了绘制整个mesh，我们必须绘制mesh的所有子集, 且有一个相对应的材质和纹理数组，即子集i与材质和纹理数组的第i项对应:

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

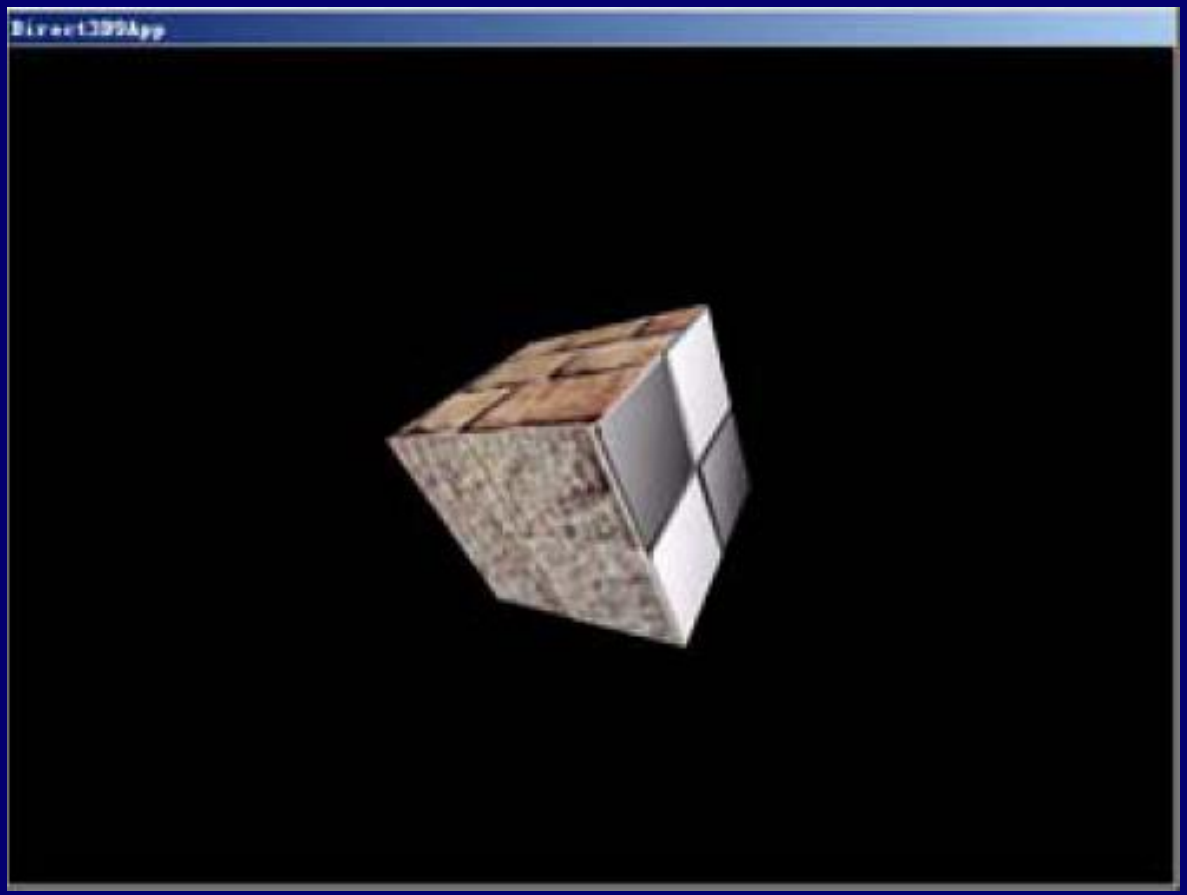
- 4. 创建一个**Mesh**
  - 我们可以使用 D3DXCreateMeshFVF函数来创建一个mesh。
  - 填写上下文数据（需分别向顶点缓存,索引缓存、属性缓存提供顶点、索引、属性数据）

```
HRESULT D3DXCreateMeshFVF(
    DWORD NumFaces,
    DWORD NumVertices,
    DWORD Options,
    DWORD FVF,
    LPDIRECT3DDEVICE9 pDevice,
    LPD3DXMESH* ppMesh
);
```

- NumFaces — mesh将拥有的面数。该值必须大于0。
- NumVertices — mesh将拥有的顶点数。该值必须大于0。
- Options —用来创建mesh的一个或多个创建标志。要了解所有标志信息请查看sdk文档。现在列出一部分：
  - D3DXMESH_32BIT — mesh使用32位索引。
  - D3DXMESH_MANAGED — mesh数据将被放在受控的内存中。
  - D3DXMESH_WRITEONLY — mesh数据只能执行写操作，不能执行读操作。
  - D3DXMESH_DYNAMIC — mesh缓存将是动态的。
- FVF — mesh的顶点格式。

- pDevice — 与mesh相关的设备。
- ppMesh — 输出创建好的mesh。

- Sample



- 创建一个空mesh。
- 用一个立方体几何信息来填充mesh。
- 根据mesh的每个面指定子集。
- 指定属性.
- 绘制mesh。

D3D中的X文件格式

- 1. X文件简介
  - 在D3D中, X文件主要是用来存储网格数据的,当然不光是网格数据,它还用来存储有关纹理,动画及用户定义的对象的一些数据.
  - X文件还是模板驱动(Template-Driven)的,也就是说它存储数据的格式是基于模板的. 这使得这种文件格式具有结构自由,内容丰富,易应用,可移植性高等优点.

- 2. X文件中的模板(Template)
  - 模板定义了数据流是怎样被格式化的，也就是告诉你对于一个3D模型的各种数据，它们是以什么格式存放在数据流中的：
  - 在X文件中，一般会有如下所示的格式：

```
template <template-name> { // 模板名
    <UUID> // 一个全局唯一ID, 用来唯一标识该模板
    <member 1>; // 成员变量1
    ...
    <member n>; // 成员变量n
    }
```

- // 网格模板， 定义网格数据格式
  template Mesh {
  <3D82AB44-62DA-11cf-AB39-0020AF71E433>
  DWORD nVertices; // 网格顶点数
  array Vector vertices[nVertices]; // 网格顶点数据列表
  DWORD nFaces; // 网格面数
  array MeshFace faces[nFaces]; // 网格面的数据列表
  [...] // 可添加其他成员
  }

- // 材质模板，定义材质属性
  template Material {
  <3D82AB4D-62DA-11cf-AB39-0020AF71E433>
  ColorRGBA faceColor; // 材质颜色
  FLOAT power; // 高亮光强度
  ColorRGB specularColor; // 高亮光颜色
  ColorRGB emissiveColor; // 发散光颜色
  [...] // 可添加其他成员
  }

  // 网格面模板，定义网格中的一个面（即图元，还记得网格的概念吗）
  template MeshFace {
  <3D82AB5F-62DA-11cf-AB39-0020AF71E433>
  DWORD nFaceVertexIndices; // 该面包含的定点数
  array DWORD faceVertexIndices[nFaceVertexIndices]; // 面的顶点数据索引列表
  }

- 3. X文件格式
  - D3D的X文件拥有自己的一套完整的语法规则，以下是一个完整X文件，它存储了一个正方形网格的数据。该正方形有两个面，一面为红色，一面为蓝色。

- // A single rectangle with two faces
  Header
  {
  1;
  0;
  1;
  }

- Material Face1Material // Material 1
  {
  1.0; 0.0; 0.0; 1.0;; // 红色
  0.0;
  0.0; 0.0; 0.0; 0.0;;
  0.0; 0.0; 0.0; 0.0;;
  }
- Material Face2Material // Material 2
  {
  0.0; 0.0; 1.0; 1.0;; // 蓝色
  0.0;
  0.0; 0.0; 0.0; 0.0;;
  0.0; 0.0; 0.0; 0.0;;
  }

```
Mesh Face
        {
                4; // 该网格包含四个顶点
                        -1.0; -1.0; 0.0;, // 顶点0坐标
                        -1.0; 1.0; 0.0;, // 顶点1坐标
                         1.0; 1.0; 0.0;, // 顶点2坐标
                        1.0; -1.0; 0.0;; // 顶点3坐标
                2; // 该网格包含两个面
                        4; 0, 1, 2, 3;, // 网格正面，4表示该面有四个
顶点，后面是顶点数据的索引
                        4; 3, 2, 1, 0;; // 网格反面

        MeshMaterialList // 材质列表，为每个面分配一个材质
        {
         2;
        2;
         0, 1;;
         { Face1Material }
        { Face2Material }
         }
}
```

- a. 第1行，x文件的标题
  xof ------ 表明该文件类型是X文件
  0302 ----- 表明该文件大版本号为03， 小版本号为02，该版本号一般不会变
  txt ------ 表明该文件为文本文件，同样"bin"为二进制文件，"tzip"为压缩文本文件，
  "bzip"为压缩二进制文件。
  0064 ----- 表明使用64位的浮点数。同样，0032表示使用32位的浮点数。
  注意： 该标题字符串必须放在X文件的第一行。

  b. 第3-4行，注释
  在X文件中， 符号"//"或"#"用来表示注释。

  c. 第7-12行，该文件的标题模板数据
  该模板数据同最前的标题字符串相对应，用来说明该文件的一些信息。

  d. 第14-30行，定义了两个材质，分别具有红色和蓝色属性。
  在模板的数据中，要注意逗号和分号的使用。这方面的内容可查SDK的文档。

  e. 第32-53行，给出了该网格的模板数据，可参见注释。

读取X文件

- 读取存储在X文件中的mesh数据。这个方法创建一个ID3DXMesh对象，且从X文件中读取几何信息数据填入其中。

```
HRESULT D3DXLoadMeshFromX(
    LPCSTR pFilename,
    DWORD Options,
    LPDIRECT3DDEVICE9 pDevice,
    LPD3DXBUFFER *ppAdjacency,
    LPD3DXBUFFER *ppMaterials,
    LPD3DXBUFFER* ppEffectInstances,
    PDWORD pNumMaterials,
    LPD3DXMESH *ppMesh
);
```

- pFilename — 读取的X文件的文件名。
- Options — 用来创建mesh的一个或多个创建标志。要了解所有标志信息请查看sdk文档。现在列出一部分：
  - D3DXMESH_32BIT — mesh使用32位索引。
  - D3DXMESH_MANAGED — mesh数据将被放在受控的内存中。
  - D3DXMESH_WRITEONLY — mesh数据只能执行写操作，不能执行读操作。
  - D3DXMESH_DYNAMIC — mesh缓存将是动态的。
- pDevice — 与复制mesh有关的设备。
- ppAdjacency — 返回一个ID3DXBuffer包含一个DWORD数组，描述mesh的邻接信息。
- ppMaterials — 返回一个ID3DXBuffer包含一个D3DXMATERIAL结构的数组，存储了mesh的材质数据。我们在下一节介绍mesh材质。
- ppEffectInstances — 返回一个ID3DXBuffer包含一个D3DXEFFECTINSTANCE结构的

- 数组。我们现在通过指定0值来忽略这个参数。
- pNumMaterials — 返回mesh的材质数。
- ppMesh — 返回填充了X文件几何信息的ID3DXMesh对象。

- **X**文件的材质
  - D3DXLoadMeshFromX的第七个参数返回的是mesh包含的材质数，第五个参数返回的是包含着材质数据的一个D3DXMATERIAL结构数组。
  - D3DXMATERIAL结构的定义如下：

```
typedef struct D3DXMATERIAL {
    D3DMATERIAL9 MatD3D;
    LPSTR pTextureFilename;
} D3DXMATERIAL;
```

- D3DXLoadMeshFromX函数读取X文件数据以便在返回的D3DXMATERIAL数组中的第i项与第i个子集相对应。

实例程序

- 3. 实例程序
  - 定义一个ID3DXMesh对象，它被用来存储从X文件中读取的mesh数据，也有一个材质vector和纹理vector,我们用它们来分别存储mesh的材质和纹理。

```
ID3DXMesh*                          Mesh = 0;
std::vector<D3DMATERIAL9>           Mtrls(0);
std::vector<IDirect3DTexture9*>     Textures(0);
```

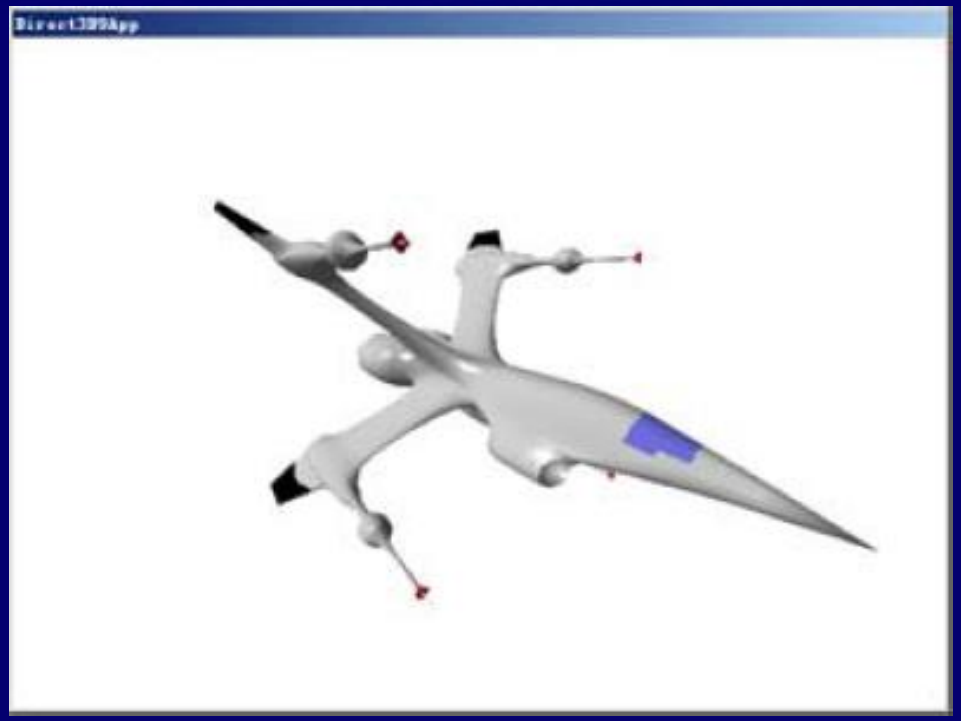
- 首先，我们读取X文件：

```
hr = D3DXLoadMeshFromX(
    "bigship1.x",
    D3DXMESH_MANAGED,
    Device,
    &adjBuffer,
    &mtrlBuffer,
    0,
    &numMtrls,
    &Mesh);
```

- 读取完X文件数据以后，我们必须遍历D3DXMATERIAL数组来读取mesh中所使用的所有纹理：

- 我们使用简单的循环，Mesh便能够被渲染了：

```
for(int i = 0; i < Mtrls.size(); i++)
{
    Device->SetMaterial( &Mtrls[i] );
    Device->SetTexture(0, Textures[i]);
    Mesh->DrawSubset(i);
}
```


Direct3DApp

对于其他类型的文件

- 常用到的有3DSMax,Maya的三维建模工具可以生成3D模型再转换为.X文件。
- 因为.X文件是DirectX定义的格式，并且D3DX库很容易地支持X文件。D3DX库提供了读和写X文件的函数。因此，如果我们使用这种格式就避免了还要自己写程序文件来读/写模型文件了。
- 利用一些已经开发好的针对3DMax,LightWave,Maya软件导出.X文件。

- Maya: May5XFileExpSrc
- 3DS Max: Deep Exporation.exe
- Conv3ds
- Milkshape3D

- 第二次课外作业
  - 练习导入.X文件,可以在Direct3D\Tutorials的Tut06上修改, 在现在的基础上再导入一个模型.
  - 读入两个文件，其中一个文件2.txt保存的是一个物体的顶点信息，文件1.dat保存的是该物体的面片信息，要求将这两个文件的数据读入程序并用D3D绘制出来。