# [Direct3D API大全 全部函数](http://blog.csdn.net/pizi0475/article/details/5388234)

//Direct3D 9.0 SDK 开发参考Direct3D 9.0 SDK 文档 (中文版)   
词汇表   
DirectX 8 教程  
[helpcontext(0x00014471)]  
HRESULT \_stdcall CopyRects(  
                [in] Direct3DSurface8\* SourceSurface,   
                [in] void\* FirstElementOfSourceRectsArray,   
                [in] int NumberOfRects,   
                [in] Direct3DSurface8\* DestinationSurface,   
                [in] void\* FirstElementofDestPointArray);  
'## 声明DECLARATIONS ##  
'//Our renderable textures 我们渲染的贴图  
Dim TexSource As Direct3DTexture8  
Dim TexDest As Direct3DTexture8  
Dim TexComb As Direct3DTexture8  
'//The surfaces that will point to them.它们的指针  
Dim SurfSource As Direct3DSurface8  
Dim SurfDest As Direct3DSurface8  
Dim SurfComb As Direct3DSurface8  
'## 初始INITIALISATION ##  
'//Create Our TEXTURE objects创建我们的TEXTURE对象  
Set TexSource = D3DX.CreateTextureFromFileEx(D3DDevice, App.Path & "/texsource.bmp", \_  
                                                                           128, 128, 1, 0, D3DFMT\_R5G6B5, D3DPOOL\_MANAGED, \_  
                                                                           D3DX\_FILTER\_LINEAR, D3DX\_FILTER\_LINEAR, 0, ByVal 0, \_  
                                                                           ByVal 0)  
Set TexDest = D3DX.CreateTextureFromFileEx(D3DDevice, App.Path & "/texdest.bmp", \_  
                                                                           128, 128, 1, 0, D3DFMT\_R5G6B5, D3DPOOL\_MANAGED, \_  
                                                                           D3DX\_FILTER\_LINEAR, D3DX\_FILTER\_LINEAR, 0, ByVal 0, \_  
                                                                           ByVal 0)  
Set TexComb = D3DX.CreateTexture(D3DDevice, 128, 128, 1, 0, D3DFMT\_R5G6B5, D3DPOOL\_MANAGED)  
'## 渲染循环RENDER LOOP ##  
'//渲染时创建一个texture指针  
    Set SurfSource = TexSource.GetSurfaceLevel(0)  
       Set SurfDest = TexDest.GetSurfaceLevel(0)  
       Set SurfComb = TexComb.GetSurfaceLevel(0)  
'我们拷贝1/2的SurfSource面及1/2的 SurfDest面到SurfComb  
       rctSource.Top = 0: rctSource.Left = 0: rctSource.Right = 64: rctSource.bottom = 128  
       ptDest.X = 0: ptDest.Y = 0  
       D3DDevice.CopyRects SurfSource, rctSource, 1, SurfComb, ptDest  
         
       rctSource.Top = 0: rctSource.Left = 64: rctSource.Right = 128: rctSource.bottom = 128  
       ptDest.X = 64: ptDest.Y = 0  
       D3DDevice.CopyRects SurfDest, rctSource, 1, SurfComb, ptDest   
你也可以把 COM 对象就想象成一套为某个主题而设计的一整套库函数。DX 就提供了一套完整的设计3D游戏的库。   
<http://baike.baidu.com/view/1169027.htm>   
使用DirectX的不同组件，你需要链接不同的静态库。例如你要使用DirectDraw组件，你就需要ddraw.lib。   
对于DirectDraw，这个头文件是ddraw.h。//com编程  
//DirectDraw的两个方法  
Dim objDX As New DirectX7  
Dim objDD As DirectDraw7  
Dim objDDPrimSurf As DirectDrawSurface7  
Dim objDDBackSurf As DirectDrawSurface7  
Dim objDDOffSurf As DirectDrawSurface7  
Dim ddsd1 As DDSURFACEDESC2  
Dim ddsd2 As DDSURFACEDESC2  
Dim r1 As RECT  
Private Sub Form\_Load()  
Set objDD = objDX.DirectDrawCreate("C:/Documents and Settings/Administrator/桌面/Noname.png") '建立DirectDraw对象  
'建立协作层  
Call objDD.SetCooperativeLevel(Me.hWnd, DDSCL\_FULLSCREEN Or DDSCL\_ALLOWMODEX Or DDSCL\_EXCLUSIVE)  
objDD.SetDisplayMode 800, 600, 32, 0, DDSDM\_DEFAULT  
End Sub  
  
D3D.H   
<http://wowe1314.blog.163.com/blog/static/2358876200751191213583/>   
在Direct3D编程中，我们要做的工作基本上可以归纳为：   
调用适当的函数获取接口指针；   
调用接口的方法（成员函数）来完成所需功能；   
用完接口后，调用Release方法进行“释放”，注意释放顺序应该和获取它们的顺序相反。   
<http://www.lihuasoft.net/article/show.php?id=2928>   
Microsoft\_DirectX\_9.0c里的 9个DirectX的DLL   
DX9和DX10在渲染流水线上都是有天壤之别的，好在DX高版本开发包运行库中包含了对低版本开发包运行库的实现，所以用DX8开发的程序，DX9运行库也能够很好的支持，在安装有D9运行库的系统上跑DX8开发的程序不需要再安装DX8运行库，但是这个兼容性支持在最近被微软逐渐放弃，有时候DX9的不同更新版本做的程序也不能向下兼容，比如DX9FEB2007SDK,同DX9AUG2006SDK在shader编译规则上也是不同的，2007放弃了VS2.0和PS2.0以下版本shader的支持，同时对于HLSL中#include相对路径引用的默认根目录也是有区别的.openGL的shader扩展不同的厂商有不同的扩展开发包，但是这种情况随着GLSL和openGL2.0的出现有所改观.同时OpenGL是跨平台的而DX不是，这意味着用OpenGL和GNU   C++规则开发的程序可以同时在Linux,unix和安装有GNU环境的Windows上同时运行。从效率上来看，DX由于数据时批量写入显存的，同OpenGL的单条函数写入来讲DX效率上要高一些，不过近来OpenGL也支持了批写入，只是支持批写入的OpenGL放弃了openGL一惯的优势也就是语言架构上的简洁使得函数的数目变得很冗杂。在效果上看DX9同支持GLSL或CG扩展的openGL可以实现相同的显示效果。但是有一点不同是DXUT和D3DX在一些基础绘制上比glu和openGL   ARB   Extend要差一点，比如绘制虚线，DX没有好的函数可以是实现这一功能。但是DX的扩展工具比openGL扩展工具又有多余的优势比如向量计算,GUI控件,mesh优化和曲面展开，PRT预计算等等和性能测试等等上又要强一点。DX10同OpenGL比较就感觉openGL不是同一个数量级上的产品，DX10在渲染流水线和架构上和能够实现的效果上要比DX9和openGL进步的多。要做面向未来的游戏产品尽量还是用DX10吧。   
  
LPDIRECT3D9 D3D主接口   
LPDIRECT3DDEVICE9 D3D硬件主接口   
LPDZRECT3DVERTXBUFFER9 顶点缓冲区接口   
LPD3DVIEWPORT9  视口接口   
LPD3DDISPLAYMODE D3D设备显示模式接口   
LPD3DPRESENT\_PARAMETERS 创建结构用来保存设备的显示模式接口   
LPD3DXVECTOR3  3D向量接口   
LPDIRECT3DTEXTURE9 纹理接口   
ID3DXSprite  精灵接口   
g.pvb  成员函数   
g\_pD3D  成员函数   
g\_pD3D->GetAdapterDisplayMode(D3DADAPTER\_DEFAULT,&d3ddm) 获取显示模式   
g\_pd3dDevice 成员函数   
g\_pd3dDevice->SetRenderState（，BOOL） 是否开启灯光   
g\_pd3dDevice->SetTransform( D3DTS\_WORLD, &(matWorld \* matWorld\_x));//将上面计算出来的旋转矩阵，设置为世界变换矩阵   
g\_pd3dDevice->SetStreamSource( 0, g\_pVB, 0, sizeof(CUSTOMVERTEX) );写入流   
g\_pd3dDevice->SetFVF() 设置FVF   
g\_pd3dDevice->DrawIndexedPrimitive( 画形状, 0, 0, 点个数, 0, 三角形个数 ); 画   
timeGetTime 当前系统的时间   
DIRECT3DTXTURE 文理接口   
BITMAPPEILEHEADER 文件头   
BITMAPINFOHEADER 信息头   
fread 读一个数据块   
biBitcout 每个象素占几个字节   
bicompression 是否被压缩   
fseek 跳，偏移指针   
greatetxture 创建一个空文理   
D3Dcaked\_RECT 锁定结构体   
setTexturestagestata 设置文理操作   
CONSTD3DMATRIX\*  进行变换的变换矩阵   
结构体   
D3DPRESENT\_PARAMETERS 存储D3D设备信息  
下面是用标准的Direct3D方法创建设备，CreateDevice：  
CreateDevice方法创建设备  
HRESULT CreateDevice(  
    UINT                  Adapter,//需要正确的adapter(适配器)  
    D3DDEVTYPE            DeviceType,//设备类型(hal or reference)  
    HWND                  hFocusWindow,//窗口句柄  
    DWORD                 BehaviorFlags,//行为标记(软件/硬件顶点处理和其他设备标记)  
    D3DPRESENT\_PARAMETERS \*pPresentationParameters,//和一个表达参数(一个D3DPRESENT\_PARAMETERS结构的参数)  
    IDirect3DDevice9      \*\*ppReturnedDeviceInterface  
);  
此外，D3DPRESENT\_PARAMETERS结构有众多成员，它们是：  
特定的后备缓存区设置，采样器设备，swap效果，窗口模式，深度模版缓存区设置，  
刷新速度等等。  
DXUTCreateDevice与CreateDevice的区别？  
DXUTCreateDevice( D3DADAPTER\_DEFAULT, TRUE, 640, 480 );  
如果应用程序使用DXUTCreateDevice函数，它不需要调用DXUTSetCallbackDeviceChanging，因为框架将会记得LPDXUTCALLBACKMODIFYDEVICESETTINGS函数。  
给所有这些参数选择有效的设定就是个挑战，那么DXUT框架的DXUTCreateDevice函数简化了这一过程：HRESULT DXUTCreateDevice(  
    UINT AdapterOrdinal  = D3DADAPTER\_DEFAULT,  
    BOOL bWindowed       = TRUE,  
    INT nSuggestedWidth  = 640,  
    INT nSuggestedHeight = 480,  
    LPDXUTCALLBACKISDEVICEACCEPTABLE pCallbackIsDeviceAcceptable     = NULL,  
    LPDXUTCALLBACKMODIFYDEVICESETTINGS pCallbackModifyDeviceSettings = NULL  
);  
大多数基本用法是很简单的，使用默认的参数调用函数：  
DXUTCreateDevice();  
对于这个简单的调用，框架创建设备的默认设定，它可以工作在大部分情况下。  
比使用一个设备创建很多默认设定更好的方法是，应用程序能够通过传递给CreateDevice函数的参数来使设备应用更多的控制。例如，你可能通过SuggestedWidth和nSuggestedHeight参数改变窗口的尺寸:DXUTCreateDevice(  
    D3DADAPTER\_DEFAULT,  
    false,  
    1024,  
    768,  
    NULL,  
    NULL,  
    NULL  
);  
 为了取得更多的控制，应用程序可以使用2个可选的回调函数： LPDXUTCALLBACKISDEVICEACCEPTABLE和LPDXUTCALLBACKMODIFYDEVICESETTINGS.  
 D3DXMATRIX  修改矩阵数组   
CUSTOMVERTEX 设置顶点位置颜色信息   
  
矩阵函数   
D3DXMATRIX \* D3DXMatrixIdentity（POut,pM） 单位矩阵   
D3DXMATRIX \* D3DXMatrixTranspose（上）  矩阵转置   
D3DXMATRIX \* D3DXMatrixInverse（上中间加个FLOAT） 逆矩阵   
D3DXMATRIX \* D3DXMatrixTransformation（）   
D3DXMATRIX\* D3DXMatrixTranslation（输出矩阵，X，Y，Z） 平移变换   
D3DXMATRIX \* D3DXMatrixScaling（上） 缩放变换   
FLOAT D3DXPlaneDotCoord（pp,pv） 点和平面之见的关系   
D3DXPLANE \* D3DXPlaneFromPointNormal（POUT，PPOINT，PNORMAL） 构造子   
D3DXPLANE \* D3DXPlaneFromPoints(Pout,pv1,pv2,pv3) 通过点来描述平面   
D3DXPLANE \* D3DPlaneNormalize(POUT,PP) 标准化一个平面   
D3DXPLANE \* D3DXPlaneTransform(POUT,PP,PM) 平移平面   
D3DXM   
转换函数   
D3DXMATRIX\* D3DXMatrixLookAtLH(输出用于视图转换的矩阵，摄象机的位置，摄象机面向的位置，摄象机的正方向)  视图转换的矩阵   
D3DXMATRIX\* D3DXMatrixOrthoLH(输出用于正交投影的交换矩阵，取景宽，取景高，取景离摄象机的最近距离，取景离摄象机的最远距离) 正交投影变换矩阵   
D3DXMATRIX\* D3DXMatrixPerspectiveFovLH(输出用于透视投影的交换矩阵，摄象机镜头的夹角Y，平截台体的纵横比，近平截面的距离，远平截面的距离) 透视投影的矩阵   
Direct3DCreate9(D3D版本) 创建D3D对象   
设备函数   
SetTransform(变换的类型，变换的变换矩阵) 设置左手或右手坐标   
SetViewport(视口指针) 设置远近距离   
GetClientRect(hWnd,\*RECT) 获取窗口绘图区域   
memcpy（指针，数组，长度） 拷贝   
SetStreamSource(0,G.pvb接口指针，0,长度) 数据流   
GetAdapterDisplayMode(指定显示卡序列号，存储显示模式的指针) 获取显卡的模式   
HRESULT CreateDevice(显卡序列号，D3D设备类型，所属窗口句柄，D3D进行3D运算，存储D3D设备相关信息指针，返回D3D设备借口指针的地址) 创建设备借口   
HRESULT CreateVertexBuffer(顶点缓冲区大小（字节），顶点缓冲区属性，灵活顶点格式，顶点缓冲区内存位置，顶点缓冲区指针地址，保留参数通常为0) 创建顶点缓冲   
HRESULT CreateIndexBuffer(索引缓冲区大小（字节）,顶点缓冲区属性,FMT颜色,顶点缓冲区内存位置,索引缓冲区指针地址，保留参数通常为0)   创建索引缓冲   
HRESULT Lock(加锁内存起始地址，加锁内存大小，返回内存指针地址，加锁属性) 加缩内存   
HRESULT UnLock() 解锁   
HRESULT SetStreamSource(渲染数据流序列号，进行绑定连接的顶点缓冲区指针，进行绑定连接渲染数据流的起始位置，渲染数据流中一个顶点所占的内存大小) 顶点缓冲区和渲染数据流连接   
HRESULT SetFVF(灵活顶点格式) 设置顶点格式   
HRESULT DrawPrimitive(绘制的图元类型，绘制的开始顶点的索引值，绘制的图元数量)  画到后向缓冲区   
HRESULT DrawPrimitiveup() 可以直接画   
HRESULT Preesent(复制源的矩形区域指针，复制目的地的矩形区域指针，D3D设备窗口句柄，最小更新区域指针) 屏幕翻转   
HRESULT SetIndices(使用的索引缓冲区指针) 设置当前绘制的索引数组   
DrawIndexedPrimitive(图元类型，绘制到的索引缓冲区的开始地址，最小的索引数组元素的值，顶点的数目，开始的索引数组元素的值，绘制的数量) 同DrawPrimitive()   
  
绘制函数   
HRESULT DrawPrimitive(基本图元类型，起始顶点，绘制的图元的数量)  图元绘制   
HRESULT Clear(清楚的矩形区域数量，清除的举行区域数组指针，清楚哪个缓冲区，清除后重置的颜色，清除后重置的深度，0-1.0，重置的摸版值) 清空图形绘制区  
Sub Clear(count As Long, recs() As D3DRECT, flags As CONST\_D3DCLEARFLAGS, color As Long, z As Single, stencil As Long)  
DxVBLib.Direct3DDevice7 的成员  
'将整个视界背景设置为蓝色，并清除Z缓冲  
Direct3DDevice.Clear 1, ViewPortRect(), D3DCLEAR\_TARGET, &HFF, 1, 0//清屏函数  
  
HRESULT BeginScene() 开始绘制  
Sub BeginScene()  
    DxVBLib.Direct3DDevice7 的成员  
'开始绘制场景  
Direct3DDevice.BeginScene   
HRESULT EndScene() 结束绘制   
  
纹理函数   
CreateTexture()  创建D3D纹理对象   
LoadBmpTeture（） 装载文理函数   
LoadBmpTexture24Bit (LPDIRECT3DDEVICE9 pDevice,LPCSTR  pSrcFile,LPDIRECT3DTEXTURE9\* ppTexture) 24位纹理   
D3DXCreateTextureFromFile(D3D设备指针，纹理图形文件，存储D3D文理的指针地址) 直接从磁盘获取纹理   
D3DXCreateTextureFromFileEx(D3D设备指针，纹理图形文件，指定纹理宽，高，指定渐进纹理序列级数，纹理使用方式一般为0，指定纹理图形格式，纹理存放的内存类型一般位为0，纹理过滤方式，自动生成的纹理序列过滤方式，设置透明色，图形文件信息存放地址可设置0，调色板存储地址，创建的D3D文理的指针地址) 高级获取纹理   
HRESULT SetTexture(多级纹理的索引0-7，D3D的纹理接口指针) 设置当前要渲染的纹理   
HRESULT SetTextureStageState(多级纹理的索引，纹理渲染状态的类型，纹理渲染状态的值，与类型相对应) 设置纹理的渲染状态   
HRESULT SetSamplerState(指定纹理属性0-7，纹理采样属性类型，设置纹理采样属性) 纹理采样   
HRESULT CheckDeviceFormat(指定显卡序列号,D3D设备类型,指定显示模式格式，缓冲区属性，需要使用查询的格式的设备类型,需要查询的显示格式) 纹理压缩   
HRESULT LockRect(指定加锁的纹理级别，指向D3DLOCKED\_RECT结构，要加锁的RECT区域-0代表整个区域，加锁类型-取0或下表的值) 锁定纹理   
HRESULT UnlockRect(解锁的纹理级别) 解锁纹理  
//二维向量和三维向量的加减   
<http://www.cppblog.com/lovedday/archive/2007/04/27/23060.html>   
C2 = (6.00, 3.00) + (4.00, 2.00) = (10.00, 5.00)   
C3 = (5.00, 2.60, 10.00) + (2.00, 3.40, 3.00) = (7.00, 6.00, 13.00)   
D3 = (5.00, 2.60, 10.00) - (2.00, 3.40, 3.00) = (3.00, -0.80, 7.00)   
向量函数   
D3DXVECTOR3 \* D3DXVer3Length(V) 向量模的计算   
D3DXVECTOR3 \* D3DXVec3Normalize（返回指针，V） 单位化   
D3DXVECTOR3 \* D3DXVec3Add(返回的指针，u,v) 向量加法   
D3DXVECTOR3 \* D3DXVec3Subtract（同上） 减法   
D3DXVECTOR3 \* D3DXVec3Cross(同上) 向量X乘   
D3DXVECTOR3 \* D3DXVec3Lerp(同上) 数乘   
D3DXVECTOR3 \* D3DXVec3Maximize（同上） 取最大值   
D3DXVECTOR3 \* D3DXVec3Minimize（同上） 取最小值   
D3DXVECTOR3 \* D3DXVec3Scale（返回指针，PV，FLOAT） 比例   
FLOAT D3DXVec3Dot(pv1,pv2) 点乘   
参见编程精粹.chm中的COM中模块的导出函数  
Private Type D3DVECTOR  
    x As Single  
    y As Single  
    z As Single  
End Type  
'返回3D向量的规格化向量  
Private Declare Function D3DXVec3Normalize Lib "DX8VB.DLL" Alias "VB\_D3DXVec3Normalize" (VOut As D3DVECTOR, v As D3DVECTOR) As Long  
Private Declare Function D3DXVec3Add Lib "DX8VB.DLL" Alias "VB\_D3DXVec3Add" (VOut As D3DVECTOR, v1 As D3DVECTOR, V2 As D3DVECTOR) As Long  
Private Declare Function D3DXVec3Subtract Lib "DX8VB.DLL" Alias "VB\_D3DXVec3Subtract" (VOut As D3DVECTOR, v1 As D3DVECTOR, V2 As D3DVECTOR) As Long  
Private Declare Function D3DXVec3Length Lib "DX8VB.DLL" Alias "VB\_D3DXVec3Length" (v As D3DVECTOR) As Single  
D3DFVF 自由顶点的格式   
D3DFVF\_DIFFUSE 包含谩反射的信息   
D3DFVF\_NORMAL 包含法线信息   
D3DFVF\_PSIZE 顶点信息指明绘制点的大小   
D3DFVF\_SPECULAR 包含镜面反射的信息   
D3DFVF\_XYZ 包含未经转换的顶点坐标   
D3DFVF\_XYZRHW 包含经过转换的顶点坐标   
D3DFVF\_XYZB1 through D3DFVF\_XYZB5 包含用于骨骼动化的顶点和顶点对骨骼的权重信息   
D3DFVF\_XYZW 包含经过转换和裁剪的顶点坐标   
D3DTRANSFORMSTATETYPE 变换的类型   
  
D3DPRIMITIVETYPE 定义基本图元   
D3DPT\_POINTLIST 一组点的集合   
D3DPT\_LINELIST 一组线的集合   
D3DPT\_LINESTRIP 首尾相连的线段的集合   
D3DPT\_TRIANGLELIST 一组三角形的集合   
D3DPT\_TRIANGLESTRIP 首尾相连的三角形，有两个顶点集合   
D3DPT\_TRIANGLEFAN   组成扇形的一组三角形集合   
D3DPT\_FORCE\_DWORD 未定义的   
D3DDISPLAYMODE 屏幕显示模式   
D3DFMT\_UNKNOWN  未知的象素格式   
D3DFMT\_R8G8B8  24位色，RGB各占8位   
D3DFMT\_A8R8G8B8  32位色，@RGB各占8位   
D3DFMT\_X8R8G8B8  32位色，X为保留8位 RGB各占8位   
D3DFMT\_R5G6B5  16位色，R占5，G占6，B占5位   
D3DFMT\_X1R5G5B5  16位色，保留1位，RGB各占5位   
D3DFMT\_A1R5G5B5  16位色，@占1位，RG5各占5位   
D3DFMT\_A4R4G4B4  16位色，@RGB各占4位   
D3DFMT\_R3G3B2  8位色，R3,G3,B2位   
D3DFMT\_A8  只有8位@   
D3DFMT\_A8R3G3B2  16位色，@8，R3，G3，B2位   
D3DFMT\_X4R4G4B4  16位色   
D3DFMT\_A2B10G10R10 32位色，@占2位，RGB各10位   
D3DFMT\_A8B8G8R8  32位色   
D3DFMT\_X8B8G8R8  32位色   
D3DFMT\_G16R16  32位色，只有红和绿   
D3DFMT\_A2R10G10B10 32位色   
D3DFMT\_A16B16G16R16 64位色   
D3DFMT\_A8P8  8位色，8位表示半透明，8位表示颜色   
D3DFMT\_P8  8位色，用牙色索引值表示   
D3DFMT\_L8  8位色，只表示亮度   
D3DFMT\_L16  16位色，只表示亮度   
D3DFMT\_A8L8  16位色，8位表示半透明，8位表示亮度   
D3DFMT\_A4L4  8位色，4位表示半透明，4位表示亮度   
D3DDEVTYPE\_HAL   硬件抽象层，通过显示硬件来完成图形渲染工作   
D3DDEVTYPE\_NULLREF     
D3DDEVTYPE\_REF   参考光栅器，一般用语测试显示卡不支持的D3D功能   
D3DDEVTYPE\_SW   用语支持第三方的软件   
D3DDEVTYPE\_FORCE\_DWORD  扩展的   
D3DCREATE 3D运算的方式   
D3DCREATE\_ADAPTERGROUP\_DEVICE   
D3DCREATE\_DISABLE\_DRIVER\_MANAGEMENT   
D3DCREATE\_DISABLE\_DRIVER\_MANAGEMENT\_EX   
D3DCREATE\_FPU\_PRESERVE   激活双精度浮点运算或浮点运算异常检测，设置该项会降低系统性能   
D3DCREATE\_HARDWARE\_VERTEXPROCESSING 由D3D硬件进行顶点预算   
D3DCREATE\_MIXED\_VERTEXPROCESSING 由混合方式进行顶点运算   
D3DCREATE\_MULTITHREADED   支持多线程绘制，设置该项会降低系统性能   
D3DCREATE\_NOWINDOWCHANGES     
D3DCREATE\_PUREDEVICE   禁用D3D的GET\*（）函数，禁止D3D使用虚拟设备模拟顶点运算   
D3DCREATE\_SCREENSAVER     
D3DCREATE\_SOFTWARE\_VERTEXPROCESSING 由D3D软件进行顶点运算   
D3DSWAPEFFECT 取值列表   
D3DSWAPEFFECT\_DISCARD    后台缓冲区复制到前台时，清除后台缓冲区内容   
D3DSWAPEFFECT\_FLIP    后台缓冲区内容复制后，保持不变，有多个后台缓冲区时使用   
D3DSWAPEFFECT\_COPY    后台缓冲区内容复制后，保持不变，只有1个后台缓冲区时使用   
D3DSWAPEFFECT\_FORCE\_DWORD   强迫该直作为32位存储，通常不用   
D3DPRESENT 屏幕反转模式列表   
D3DPRESENT\_DONOTWAIT     
D3DPRESENT\_INTERVAL\_DEFAULT 默认的同ONE   
D3DPRESENT\_INTERVAL\_ONE  当屏幕刷新一次时前台后台进行交换   
D3DPRESENT\_INTERVAL\_TWO  当屏幕刷新二次时前台后台进行交换   
D3DPRESENT\_INTERVAL\_THREE 当屏幕刷新三次时前台后台进行交换   
D3DPRESENT\_INTERVAL\_FOUR 当屏幕刷新四次时前台后台进行交换   
D3DPRESENT\_INTERVAL\_IMMEDIATE 图形绘制完成时立即进行交换   
D3DPRESENT\_LINEAR\_CONTENT   
D3DUSAGE 缓冲区属性值列表   
D3DUSAGE\_AUTOGENMIPMAP     
D3DUSAGE\_DEPTHSTENCIL   
D3DUSAGE\_DMAP     
D3DUSAGE\_DONOTCLIP 禁用裁剪，表示顶点缓冲区中的顶点不进行裁剪，当设置该属性时，渲染状态D3DRS\_CLIPPING必须设为FALSE   
D3DUSAGE\_DYNAMIC 使用动态内存分配   
D3DUSAGE\_NPATCHES 使用顶点缓冲区绘制N-patches曲线   
D3DUSAGE\_POINTS  使用顶点缓冲区绘制点   
D3DUSAGE\_RENDERTARGET   
D3DUSAGE\_RTPATCHES 使用顶点缓冲区绘制曲线   
D3DUSAGE\_SOFTWAREPROCESSING 使用软件进行顶点运算，否则使用硬件计算   
D3DUSAGE\_WRITEONLY 只写属性，不能进行读操作，设置该属性可以提高系统性能   
D3DPOOL  缓冲区资源内存位置列表   
D3DPOOL\_DEFAULT  默认的，顶点缓冲区尽可能存在与显存中   
D3DPOOL\_MANAGED  由D3D自动调度顶点缓冲区内存位置（显存和内存）   
D3DPOOL\_SCRATCH  顶点缓冲区位于计算机的临时内存中，这种类型的顶点缓冲区不能直接进行渲染，只能进行内存枷锁，拷贝等操作   
D3DPOOL\_SYSTEMMEM 顶点缓冲区位于内存中   
D3DLOCK  缓冲区加锁   
D3DLOCK\_DISCARD  更新整个缓冲区   
D3DLOCK\_DONOTWAIT   
D3DLOCK\_NO\_DIRTY\_UPDATE 在加锁的过程中系统进行其他操作（默认有Dirty标记)   
D3DLOCK\_NOOVERWRITE 保证不腹稿缓冲区数据，设置该属性可以立即返回内存指针，提高系统性能   
D3DLOCK\_NOSYSLOCK 在加锁的过程中系统可能执行其他操作   
D3DLOCK\_READONLY 设置缓冲区位只读属性   
D3DXVECTOR3 向量算法   
D3DXVECTOR3u(x,y,z);   
D3DXVECTOR3v(x,y,z);   
float 变量=D3DXVec3Dot(u指针,v指针) 点乘   
D3DXMATRIX 矩阵   
D3DXMatrixIdentity 单位矩阵   
D3DXMatrixInverse 逆矩阵   
D3D实现图形变换   
D3DXMatrixTranslation 平移矩阵   
D3DXMatrixLockAtLH 观察矩阵   
D3DXMatrixIdentity  将一个矩阵单位化   
D3DXMatrixRotationY 绕Y轴转   
D3DXMatrixRotationX 绕X轴转   
D3DXMatrixRotationZ 绕Z轴转   
D3DXMatrixScaling 缩放变换   
D3DXMatrixMuLationAxis 围绕任意一个轴旋转   
D3DXMatrixMultiply 组合变换   
D3DUSAGE 纹理使用   
D3DUSAGE\_AUTOGENMIPMAP 自动生成多级渐进纹理序列，该方式在资源处于D3DPOOL\_SYSTEMMEM时无效   
D3DUSAGE\_DEPTHSTENCIL 深度模版缓冲区，只在资源处于D3DPOOL\_default时有效   
D3DUSAGE\_DMAP  该纹理是一个置换纹理   
D3DUSAGE\_DONOTCLIP   
D3DUSAGE\_DYNAMIC   
D3DUSAGE\_NPATCHES   
D3DUSAGE\_POINTS   
D3DUSAGE\_RENDERTARGET 该文理是一个渲染目标缓冲区   
D3DUSAGE\_RTPATCHES   
D3DUSAGE\_SOFTWAREPROCESSING 应用坐标变换   
D3DUSAGE\_WRITEONLY   
D3DTEXTURESTAGESTATETYPE 渲染状态类型   
D3DTSS\_COLOROP   1 文理层的颜色混合方式   
D3DTSS\_COLORARG1  2 颜色混合的第一个参数   
D3DTSS\_COLORARG2  3 颜色混合的第二个参数   
D3DTSS\_ALPHAOP   4 指定纹理层的Alpha透明   
D3DTSS\_ALPHAARG1  5 Alpha混合的第一个参数   
D3DTSS\_ALPHAARG2  6 Alpha混合的第二个参数   
D3DTSS\_BUMPENVMAT00  7 绘制凹凸纹理时   
D3DTSS\_BUMPENVMAT01  8 绘制凹凸纹理时   
D3DTSS\_BUMPENVMAT10  9 绘制凹凸纹理时   
D3DTSS\_BUMPENVMAT11  10 绘制凹凸纹理时   
D3DTSS\_TEXCOORDINDEX  11 该纹理层使用的纹理坐标的索引   
D3DTSS\_BUMPENVLSCALE  22 绘制凹凸纹理的缩放参数   
D3DTSS\_BUMPENVLOFFSET   23 绘制凹凸纹理的平移参数   
D3DTSS\_TEXTURETRANSFORMFLAGS  24 控制纹理坐标的转换标志   
D3DTSS\_COLORARG0  26 指定混合过程的第三个颜色   
D3DTSS\_ALPHAARG0  27 Alpha混合的第三个参数   
D3DTSS\_RESULTARG  28 颜色混合的结果输出寄存器   
D3DTSS\_CONSTANT  32 颜色混合的常量寄存器   
D3DTSS\_FORCE\_DWORD  0x7fffffff 强制转换为32位，通常不用   
D3DSAMPLERSTATETYPE 纹理采样属性   
D3DSAMP\_ADDRESSU  1 包装纹理   
D3DSAMP\_ADDRESSV  2 包装纹理   
D3DSAMP\_ADDRESSW  3 包装纹理   
D3DSAMP\_BORDERCOLOR  4   
D3DSAMP\_MAGFILTER  5 处理放大过滤   
D3DSAMP\_MINFILTER  6 处理缩小过滤   
D3DSAMP\_MIPFILTER  7 多纹理过滤   
D3DSAMP\_MIPMAPLODBIAS  8 多级文理级数偏移值，初试直为0   
D3DSAMP\_MAXMIPLEVEL  9 最大多纹理级别，初试值为0   
D3DSAMP\_MAXANISOTROPY  10 各向异性，初试值为1   
D3DSAMP\_SRGBTEXTURE  11   
D3DSAMP\_ELEMENTINDEX   12   
D3DSAMP\_DMAPOFFSET  13   
D3DSAMP\_FORCE\_DWORD  0x7fffffff 强制转换32位，通常不用   
纹理寻址   
D3DTADDRESS\_WRAP  1 包装纹理寻址   
D3DTADDRESS\_MIRROR  2 镜像纹理寻址   
D3DTADDRESS\_CLAMP  3 夹取纹理寻址   
D3DTADDRESS\_BORDER  4 边框颜色纹理寻址   
D3DTADDRESS\_MIRRORONCE  5 一次镜像纹理寻址   
D3DTADDRESS\_FORCE\_DWORD 0x7fffffff强制转换32位，通常不用   
世界变换   
D3DTS\_WORLD  世界变换