[DIRECT3D状态详解](http://blog.csdn.net/pizi0475/article/details/6121300)

Microsoft® Direct3D®设备是一个状态机。应用程序设置光照、渲染和变换模块的状态，然后在渲染时传递数据给它们。

本节描述图形流水线用到的所有不同类型的状态。

* [渲染状态](http://www.gesoftfactory.com/developer/States.htm#渲染状态)
* [取样器状态](http://www.gesoftfactory.com/developer/States.htm#取样器状态)
* [纹理层状态](http://www.gesoftfactory.com/developer/States.htm#纹理层状态)
* [状态块](http://www.gesoftfactory.com/developer/States.htm#状态块)

**渲染状态**

设备渲染状态控制Microsoft® Direct3D®设备光栅化模块的行为，它们通过改变渲染状态的属性，使用何种类型的着色算法，雾属性和其它光栅化器操作来达到这个目的。

C++应用程序通过调用IDirect3DDevice9::SetRenderState方法控制渲染状态的属性。D3DRENDERSTATETYPE枚举类型指定所有可能的渲染状态，应用程序把一个枚举类型值作为第一个参数传递给**IDirect3DDevice9::SetRenderState**方法。

固定功能顶点处理由**IDirect3DDevice9::SetRenderState**方法和以下设备渲染状态控制。这些控制中的大多数在使用可编程顶点着色器时没有任何作用。

* D3DRS\_SPECULARENABLE
* D3DRS\_FOGSTART
* D3DRS\_FOGEND
* D3DRS\_FOGDENSITY
* D3DRS\_RANGEFOGENABLE
* D3DRS\_LIGHTING
* D3DRS\_AMBIENT
* D3DRS\_FOGVERTEXMODE
* D3DRS\_COLORVERTEX
* D3DRS\_LOCALVIEWER
* D3DRS\_NORMALIZENORMALS
* D3DRS\_DIFFUSEMATERIALSOURCE
* D3DRS\_SPECULARMATERIALSOURCE
* D3DRS\_AMBIENTMATERIALSOURCE
* D3DRS\_EMISSIVEMATERIALSOURCE
* D3DRS\_VERTEXBLEND

另外，固定功能顶点处理流水线使用以下方法设置变换、材质和光照。

* IDirect3DDevice9::SetTransform
* IDirect3DDevice9::SetMaterial
* IDirect3DDevice9::SetLight
* IDirect3DDevice9::LightEnable

**注意**     D3DRS\_SPECULARENABLE控制像素流水线中镜面反射色的加法。D3DRS\_FOGSTART，D3DRS\_FOGEND和D3DRS\_FOGDENSITY控制如何计算雾的起点、终点和像素雾的密度。

更多的信息包含在以下主题中。

**概述**

[**阿尔法混合状态**](http://www.gesoftfactory.com/developer/States.htm#阿尔法混合状态)

[**阿尔法测试状态**](http://www.gesoftfactory.com/developer/States.htm#阿尔法测试状态)

[**环境光状态**](http://www.gesoftfactory.com/developer/States.htm#环境光状态)

[**抗锯齿状态**](http://www.gesoftfactory.com/developer/States.htm#抗锯齿状态)

[**剔除状态**](http://www.gesoftfactory.com/developer/States.htm#剔除状态)

[**深度缓存状态**](http://www.gesoftfactory.com/developer/States.htm#深度缓存状态)

[**雾状态**](http://www.gesoftfactory.com/developer/States.htm#雾状态)

[**光照状态**](http://www.gesoftfactory.com/developer/States.htm#光照状态)

[**轮廓和填充状态**](http://www.gesoftfactory.com/developer/States.htm#轮廓和填充状态)

[**每顶点颜色状态**](http://www.gesoftfactory.com/developer/States.htm#每顶点颜色状态)

[**图元裁剪状态**](http://www.gesoftfactory.com/developer/States.htm#图元裁剪状态)

[**着色状态**](http://www.gesoftfactory.com/developer/States.htm#着色状态)

[**模板缓存状态**](http://www.gesoftfactory.com/developer/States.htm#模板缓存状态)

[**纹理环绕状态**](http://www.gesoftfactory.com/developer/States.htm#纹理环绕状态)

**阿尔法混合状态**

一个颜色的阿尔法值控制它的透明度。启用阿尔法混合允许把一个表面上的颜色、材质和纹理根据透明度混合到另一个表面上。

更多信息请参阅阿尔法纹理混合和纹理混合。

C++应用程序使用D3DRS\_ALPHABLENDENABLE渲染状态启用阿尔法透明混合。Microsoft® Direct3D® API允许多种类型的阿尔法混合。但是，重要的是要注意用户的三维硬件可能不完全支持所有Direct3D提供的混合状态。

已完成的阿尔法混合的类型取决于**D3DRS\_SRCBLEND**和**D3DRS\_DESTBLEND**渲染状态。源和目的混合状态须成对使用。以下示例代码显示了如何将源混合状态设置为D3DBLEND\_SRCCOLOR并将目的混合状态设置为D3DBLEND\_INVSRCCOLOR。

// 本例假设d3dDevice为指向IDirect3DDevice9接口的有效指针。

// 设置源混合状态。

d3dDevice->SetRenderState(D3DRS\_SRCBLEND, D3DBLEND\_SRCCOLOR);

// 设置目的混合状态。

// 设置目的混合状态。

d3dDevice->SetRenderState(D3DRS\_DESTBLEND, D3DBLEND\_INVSRCCOLOR);

改变源和目的混合状态可以使物体在雾很浓或灰尘很多的环境中看起来像发光物体。例如，若应用程序在雾很浓的环境中建模了火焰，能量场，离子束或类似的发光体，则可把源和目的混合状态都设置为D3DBLEND\_ONE。

阿尔法混合的另一种应用是控制三维场景中的光照，也称为光照贴图。将源混合状态设置为D3DBLEND\_ZERO并将目的混合状态设置为D3DBLEND\_SRCALPHA，会根据源的阿尔法信息使场景变暗。源图元被用作光照贴图，对帧缓存中的内容进行缩放，并在适当的时候使之变暗，这就是单色光照贴图。

应用程序可以生成彩色光照贴图，只要把源阿尔法混合状态设置为D3DBLEND\_ZERO，并把目的混合状态设置为D3DBLEND\_SRCCOLOR。

**阿尔法测试状态**

C++应用程序可以用阿尔法测试控制何时把像素被写入渲染目标表面。通过设置D3DRS\_ALPHATESTENABLE渲染状态，应用程序让当前的Direct3D设备根据阿尔法测试函数测试每个像素。如果测试成功，那么就把像素写入表面。如果不成功，那么Direct3D就忽略该像素。应用程序通过**D3DRS\_ALPHAFUNC**渲染状态选择阿尔法测试函数。应用程序可以通过**D3DRS\_ALPHAREF**渲染状态设置一个参考阿尔法值用来和所有像素进行比较。

阿尔法测试常用于在光栅化几乎透明的物体时提高性能。如果正被光栅化的颜色数据比给定像素更不透明（D3DPCMPCAP\_GREATEREQUAL），那么该像素就被写入。否则，光栅化器就完全忽略该像素，这样就节省了将两个颜色混合所需要的处理。以下示例代码检查当前设备是否支持一个给定的比较函数，若支持，则设置比较函数的参数，用来在渲染时提高性能。

// 本示例代码假设pCaps为一D3DCAPS9结构，

// 被先前的一个IDirect3D9::GetDeviceCaps调用填充。

if (pCaps.AlphaCmpCaps & D3DPCMPCAPS\_GREATEREQUAL)

{

    dev->SetRenderState(D3DRS\_ALPHAREF, (DWORD)0x00000001);

    dev->SetRenderState(D3DRS\_ALPHATESTENABLE, TRUE);

    dev->SetRenderState(D3DRS\_ALPHAFUNC, D3DCMP\_GREATEREQUAL);

}

// 如果不支持比较，那么就照常渲染。唯一的缺点是没有性能上的提升。

并不是所有的硬件都支持全部阿尔法测试特性。可以通过调用IDirect3D9::GetDeviceCaps方法检查设备能力，取回设备能力后，根据希望使用的比较函数检查相关D3DCAPS9结构的**AlphaCmpCaps**成员。如果**AlphaCmpCaps**成员只包含D3DPCMPCAPS\_ALWAYS能力或D3DPCMPCAPS\_NEVER能力，那么驱动程序不支持阿尔法测试。

**环境光状态**

环境光是周围的光，从各个方向照射而来。

有关Microsoft® Direct3D®如何使用环境光的信息，请参阅直射光与环境光，和与光照相关的数学。

C++应用程序调用IDirect3DDevice9::SetRenderState方法设置环境光的颜色，并将D3DRS\_AMBIENT枚举类型值作为第一个参数传入。第二个参数是颜色值，默认值为零。

// 本例假设d3dDevice为指向IDirect3DDevice9接口的有效指针。

// 设置环境光

d3dDevice->SetRenderState(D3DRS\_AMBIENT, 0x00202020);

**抗锯齿状态**

抗锯齿是使屏幕上的线和边缘看起来更为平滑的一种方法。Microsoft® Direct3D®支持两种抗锯齿方法：边缘抗锯齿和全屏抗锯齿。

有关这些技术的更多细节，请参阅抗锯齿。

默认情况下，Direct3D不执行抗锯齿。边缘抗锯齿需要渲染第二遍，要启用边缘抗锯齿，应该把D3DRS\_EDGEANTIALIAS渲染状态设置为TRUE，要禁用边缘抗锯齿，应该把D3DRS\_EDGEANTIALIAS设置为FALSE。

要启用全屏抗锯齿，应该把**D3DRS\_MULTISAMPLEANTIALIAS**渲染状态设置为TRUE。要禁用全屏抗锯齿，应该把D3DRS\_MULTISAMPLEANTIALIAS设置为FALSE。

**剔除状态**

Direct3D渲染图元时会剔除背向用户的图元。

C++应用程序使用D3DRS\_CULLMODE渲染状态设置剔除模式，它可以被设置为D3DCULL枚举类型的成员。默认情况下，Direct3D把顶点逆时针排列的面作为背向面剔除。

以下示例代码描述了设置剔除模式的过程，这里把顶点顺时针排列的面作为背向面剔除。

// 本例假设d3dDevice为指向IDirect3DDevice9接口的有效指针。

// 设置剔除状态。

d3dDevice->SetRenderState(D3DRS\_CULLMODE, D3DCULL\_CW);

**深度缓存状态**

深度缓存是消除隐藏线和隐藏面的一种方法。默认的情况下，Direct3D不使用深度缓存。

有关深度缓存的概念，请参阅深度缓存。

C++应用程序用D3DRS\_ZENABLE渲染状态更新深度缓存的状态，用D3DZBUFFERTYPE枚举类型成员指定新的状态值。

若应用程序要阻止Direct3D写入深度缓存，则可在调用IDirect3DDevice9::SetRenderState时用D3DRS\_ZWRITEENABLE枚举类型作为第一个参数，并将第二个参数指定为D3DZB\_FALSE。

以下示例代码显示了如何将深度缓存状态设置为启用z缓存。

// 本例假设d3dDevice为指向IDirect3DDevice9接口的有效指针。

// 启用z缓存

d3dDevice->SetRenderState(D3DRS\_ZENABLE, D3DZB\_TRUE);

应用程序也可以使用D3DRS\_ZFUNC渲染状态控制Direct3D用于深度缓存的比较函数。

Z偏移是把一个表面显示在另一个表面之前的一种方法，即使它们的深度值相同。可以将此技术用于多种效果。一个常用的例子是在墙上渲染影子，影子和墙具有相同的深度值，但是应用程序希望把影子显示在墙上，只要给影子赋一个z偏移值就可以让Direct3D正确地显示它们（请参阅D3DRS\_ZBIAS）。

**雾状态**

雾效果可以赋予三维场景更强的真实感。雾效果不仅可以用于模拟雾，也能减少远处场景的清晰度。这反映了真实世界中的情况，物体离用户越远，它们的细节就越模糊。

有关如何在应用程序中使用雾的更多信息，请参阅雾。

C++应用程序通过设备渲染状态控制雾。D3DRENDERSTATETYPE枚举类型包括许多状态，控制应用程序使用的是像素（查找表）雾还是顶点雾，雾是何颜色，系统使用的雾的公式，及公式的参数。

可以通过将D3DRS\_FOGENABLE渲染状态设置为TRUE启用雾。雾的颜色可以使用D3DRS\_FOGCOLOR渲染状态设置为指定颜色，雾的颜色的阿尔法分量被忽略。

D3DRS\_FOGTABLEMODE和D3DRS\_FOGVERTEXMODE渲染状态控制雾计算使用的公式，它们也间接控制使用何种类型的雾。这两个渲染状态都可以被设置为D3FOGMODE枚举类型的成员。将任何一个渲染状态设置为D3DFOG\_NONE相应地禁用像素雾或顶点雾。如果两个渲染状态都被设置为有效的模式，则系统只启用像素雾。

D3DRS\_FOGSTART和D3DRS\_FOGEND渲染状态控制D3DFOG\_LINEAR模式下雾公式的参数。D3DRS\_FOGDENSITY渲染状态控制指数雾模式下雾的密度。

更多信息请参阅雾的参数。

**光照状态**

使用Microsoft® Direct3D®几何流水线的应用程序可以启用或禁用光照计算。只有包含顶点法向的顶点才能正确地计算光照，不含法向的顶点在所有光照计算中将使用零点积，因此没有法向的顶点得不到光照。

更多信息请参阅光照的数学。

应用程序通过将D3DRS\_LIGHTING渲染状态设置为TRUE启用Direct3D光照，这是默认的设置，应用程序通过将该渲染状态设置为FALSE禁用Direct3D光照。

光照渲染状态与可以对顶点缓存中的顶点执行的光照计算完全无关。在进行顶点处理时IDirect3DDevice9::ProcessVertices方法接收自己的标志用于控制光照计算。

**轮廓和填充状态**

没有纹理的图元会使用它们的材质指定的颜色进行渲染，或者如果为顶点指定了颜色，就使用顶点色。可以将D3DRS\_FILLMODE渲染状态指定为D3DFILLMODE枚举类型的值以选择填充图元的方法。

要启用抖动，应用程序必须把D3DRS\_DITHERENABLE枚举类型值作为第一个参数传给IDirect3DDevice9::SetRenderState，并把第二个参数设置为TRUE，把第二个参数设置为FALSE则禁用抖动。

有时，画一条线中的最后一个像素可能导致与周围图元的重叠。可以使用D3DRS\_LASTPIXEL枚举值控制这种情况。但是，未经深思熟虑最好不要改变这个设置。在有些情况下，禁止渲染最后一个像素可能会导致图元之间的缝隙。

通过设置适当的画线模式可以画物体的轮廓。默认的画线状态是画实线。更多信息请参阅Direct3D扩展（D3DX）线段绘制渲染状态。

**每顶点颜色状态**

当使用弹性顶点格式（FVF）编码时，顶点可以同时包含顶点颜色和顶点法向的信息。默认情况下， Microsoft® Direct3D®在计算光照时使用这些信息。要设置是否把顶点颜色用于光照计算，应该调用IDirect3DDevice9::SetRenderState方法，把D3DRS\_COLORVERTEX作为第一个参数，把第二个参数设置为FALSE禁用顶点颜色光照，或TRUE启用之。

若每顶点颜色被启用，则应用程序可以配置系统从何处取得源顶点颜色信息。D3DRS\_AMBIENTMATERIALSOURCE，D3DRS\_DIFFUSEMATERIALSOURCE，D3DRS\_EMISSIVEMATERIALSOURCE和D3DRS\_SPECULARMATERIALSOURCE渲染状态控制相应环境反射色、漫反射色、放射色和镜面反射颜色成员的来源。每个状态可以被设置为D3DMATERIALCOLORSOURCE枚举类型的成员，该枚举类型定义了一些常数，通知系统是使用当前材质的颜色、顶点的漫反射颜色还是顶点的镜面反射颜色作为指定颜色成员的来源。

**图元裁剪状态**

如果图元的一部分位于视区外，那么Microsoft® Direct3D®可以对此类图元进行裁剪。在C++应用程序中，Direct3D裁剪由D3DRS\_CLIPPING渲染状态控制。可以将该渲染状态设置为TRUE（默认值）启用图元裁剪，或将之设置为FALSE禁用Direct3D裁剪服务。

**着色状态**

Direct3D同时支持平面和高洛德着色，默认情况下使用高洛德着色。要控制当前的着色模式，C++应用程序可以给D3DRS\_SHADEMODE渲染状态指定一个D3DSHADEMODE枚举类型值。

以下C++示例代码显示了把着色状态设置为平面着色模式的过程。

// 本例假设d3dDevice为指向IDirect3DDevice9接口的有效指针。

// 设置着色模式。

d3dDevice->SetRenderState(D3DRS\_SHADEMODE, D3DSHADE\_FLAT);

**模板缓存状态**

应用程序使用模板缓存决定是否把一个像素写入到渲染目标表面。

细节请参阅模板缓存技术。

C++应用程序通过调用IDirect3DDevice9::SetRenderState方法启用或禁用模板缓存，应该将D3DRS\_STENCILENABLE作为第一个参数的值，并相应地把第二个参数设置为TRUE或FALSE启用或禁用模板缓存。

通过调用**IDirect3DDevice9::SetRenderState**，可以设置Microsoft® Direct3D®执行模板测试时使用的比较函数，只需将第一个参数的值设置为D3DRS\_STENCILFUNC，并把D3DCMPFUNC枚举类型值作为第二个参数的值传入。

模板参考值是在模板缓存中的值，模板函数用它进行测试。默认情况下，模板参考值为零。应用程序可以调用IDirect3DDevice9::SetRenderState设置它的值，只需将D3DRS\_STENCILREF作为第一个参数传入，并把第二个参数的值设置为新的参考值。

Direct3D在对每个像素执行模板测试前，会对模板参考值和模板掩码值进行按位与操作，并把得到的结果用模板比较函数与模板缓存的内容进行比较。应用程序可以调用IDirect3DDevice9::SetRenderState设置模板掩码值，只需将D3DRS\_STENCILMASK作为第一个参数传入，并把第二个参数设置为新的模板掩码值。

要设置模板测试失败时Direct3D采取的行动，可以调用**IDirect3DDevice9::SetRenderState**并将D3DRS\_STENCILFAIL作为第一个参数传入，第二个参数必须被设为D3DSTENCILCAPS枚举类型的成员。

应用程序也可以控制当模板测试通过但是z缓存测试失败时Direct3D如何响应，只需调用**IDirect3DDevice9::SetRenderState**，将D3DRS\_STENCILZFAIL作为第一个参数传递，并把第二个参数设为**D3DSTENCILCAPS**枚举类型的成员。

另外，应用程序还可以控制当模板测试和z缓存测试都通过时Direct3D做什么，只需调用**IDirect3DDevice9::SetRenderState**并将D3DRS\_STENCILPASS作为第一个参数，并把第二个参数设为**D3DSTENCILCAPS**枚举类型的成员。

写掩码可以用于修改将要写入模板缓存的值。要设置模板缓存写掩码，只需调用**IDirect3DDevice9::SetRenderState**，并将D3DRS\_STENCILWRITEMASK作为第一个参数传递，并把第二个参数设为写掩码的值。

**纹理环绕状态**

D3DRS\_WRAP0到D3DRS\_WRAP7渲染状态启用或禁用设备的多重纹理级联中各种纹理的u和v环绕。可以把这些渲染状态设置为标志D3DWRAPCOORD\_0，D3DWRAPCOORD\_1，D3DWRAPCOORD\_2和D3DWARPCOORD\_3的组合，相应地启用纹理在第一、第二、第三和第四个方向上的环绕，若使用零值，则禁用所有环绕。默认情况下，所有纹理的所有方向上的纹理环绕都被禁用。

有关的概念综述请参阅纹理环绕。

**取样器状态**

取样状态控制诸如过滤、tiling及寻址等与取样有关的操作。

**取样状态**

SetSamplerState设置取样器状态（包括tessellator中对位移贴图进行取样的状态），为了能够在编译时检测出正在移植Microsoft® DirectX® 8.x的应用程序，这些状态已经被重新命名并以D3DSAMP\_前缀开头。这些状态包括：

* D3DSAMP\_ADDRESSU, D3DSAMP\_ADDRESSV, D3DSAMP\_ADDRESSW
* D3DSAMP\_BORDERCOLOR
* D3DSAMP\_MAGFILTER, D3DSAMP\_MINFILTER, D3DSAMP\_MIPFILTER
* D3DSAMP\_MIPMAPLODBIAS
* D3DSAMP\_MAXMIPLEVEL
* D3DSAMP\_MAXANISOTROPY

在DirectX 9.0中，使用像素着色器2.0版时，虽然纹理坐标的数量仍被限制为8个，但每一趟最多可以支持16个纹理表面。有些纹理层状态与表面相关，有些与坐标集相关，有些与顶点处理相关，而有些则与像素处理相关。IDirect3DDevice9::SetSamplerState方法可以指定纹理过滤、平铺、截取、MIPLOD等状态，最多可以有16个取样器。

C++应用程序通过调用**IDirect3DDevice9::SetSamplerState**方法控制与纹理有关的取样状态。D3DSAMPLERSTATETYPE枚举类型用于指定取样状态。

**相关主题**

* [纹理层状态](http://www.gesoftfactory.com/developer/States.htm#纹理层状态)

**纹理层状态**

纹理层状态控制纹理坐标的生成及纹理坐标的状态，如环绕模式。

C++应用程序通过调用IDirect3DDevice9::SetTextureStageState方法控制与纹理有关的状态。D3DTEXTURESTAGESTATETYPE枚举类型定义了所有与纹理有关的渲染状态，应用程序应该将**D3DTEXTURESTAGESTATETYPE**枚举类型值作为第一个参数传递给**IDirect3DDevice9::SetTextureStageState**方法。

应用程序可以通过调用IDirect3DDevice9::SetTexture方法设置某一层的纹理。

**SetTextureStageState**

SetTextureStageState现在可以设置以下状态。

* 固定功能顶点处理状态。这些状态控制对纹理坐标的操控：D3DTSS\_TEXTURETRANSFORMFLAGS和 D3DTSS\_TEXCOORDINDEX。最多可以设置八个（因为Direct3D总是支持八组纹理坐标）
* 固定功能像素着色器状态（以前的TextureStageState）。D3DTSS\_COLOROP, D3DTSS\_ALPHAOP, D3DTSS\_COLORARG0, D3DTSS\_COLORARG1, D3DTSS\_COLORARG2, D3DTSS\_ALPHAARG0, D3DTSS\_ALPHAARG1, D3DTSS\_ALPHAARG2, D3DTSS\_BUMPENVMAT00, D3DTSS\_BUMPENVMAT01, D3DTSS\_BUMPENVMAT10, D3DTSS\_BUMPENVMAT11, D3DTSS\_BUMPENVLSCALE, D3DTSS\_BUMPENVLOFFSET, and D3DTSS\_RESULTARG。这些状态最多可有MaxTextureBlendStages组可供设置。

D3DTSS\_TEXCOORDINDEX是一个固定功能顶点处理状态。如果使用可编程顶点着色器，那么这个状态被忽略。

应用程序可以使用的纹理取样器的数量由像素着色器的版本决定。

* 固定功能像素着色器：MaxTextureBlendStages/MaxSimultaneousTextures个纹理取样器
* ps.1.1到ps.1.3: 4个纹理取样器。
* ps.1.4：6个纹理取样器。
* ps.2.0：16个纹理取样器。
* 支持Microsoft DirectX® 9.0位移贴图的设备将支持一个额外的取样器（D3DDMAPSAMPLER），它在tesselator对位移贴图进行取样。

有关更多纹理混合的信息，请参阅纹理混合。

**状态块**

一个状态块是一组设备状态、渲染状态、光照和材质的参数、变换状态、纹理层状态和当前纹理信息。状态块是设备当前状态的一个记录，或者是被显式地记录下来的。可以通过单独的一次调用把记录应用于设备。渲染设备可以优化设备状态块以加速应用程序所要求的一般顺序的状态改变。另外，设备状态块也可以使改变设备状态更为方便。

C++应用程序可以在调用IDirect3DDevice9::EndStateBlock方法结束记录一个状态块，以及调用IDirect3DDevice9::CreateStateBlock方法保存一组预定义的设备状态数据时，得到一个状态块句柄。

更多信息包含在以下主题中。

**概述**

[创建预定义的状态块](http://www.gesoftfactory.com/developer/States.htm#创建预定义的状态块)

[记录状态块](http://www.gesoftfactory.com/developer/States.htm#记录状态块)

**创建预定义的状态块**

IDirect3DDevice9::CreateStateBlock方法创建一个新的状态块，它包含全部的设备状态或是仅与顶点或像素处理相关的设备状态。**IDirect3DDevice9::CreateStateBlock**方法接收两个参数，第一个参数代表要在新的状态块中记录的状态信息的类型，第二个参数是返回句柄的地址，用于接收当调用成功时返回的有效状态块的句柄。

第一个参数为D3DSTATEBLOCKTYPE枚举类型，可以从以下三种选择：

* 保存顶点和像素状态。
* 只保存顶点状态。
* 只保存像素状态。

要了解可以保存的状态的全部列表，请参阅D3DSTATEBLOCKTYPE。

一定要检查**IDirect3DDevice9::CreateStateBlock**方法的错误代码，如果该方法失败，很可能是显示模式改变了。为了从此类失败中恢复，应用程序应该重新创建表面，然后重新创建状态块。

**记录状态块**

IDirect3DDevice9接口提供IDirect3DDevice9::BeginStateBlock方法，在应用程序设置设备状态时，该方法会把它们记录到一个状态块中。该方法使系统开始把设备状态的改变记录到一个状态块中，而不是将它们（新的设备状态）应用于设备。可被记录的状态列表请参阅IDirect3DDevice9::BeginStateBlock。

当应用程序完成对状态块的记录后，应该调用IDirect3DDevice9::EndStateBlock方法通知系统以结束记录。该方法返回一个状态块的句柄，应用程序应该将接收状态块句柄的变量的地址作为*pToken*参数传入。应用程序可以根据需要用这个句柄应用该状态块，记录新的状态数据到状态块中，以及在不再需要该状态块时将之删除。

一定要检查**IDirect3DDevice9::EndStateBlock**方法的错误代码，如果该方法失败，很可能是因为显示模式改变了。应该合理地设计应用程序，使之能够从此类失败中恢复，只要重新创建表面并再次记录状态块就可以达到这个目的。