# 10.3 深度/模板状态块

当创建**ID3D11DepthStencilState**接口时，第一步是要填充一个**D3D11\_DEPTH\_STENCIL\_DESC**实例：

typedef struct D3D11\_DEPTH\_STENCIL\_DESC{

 BOOL DepthEnable;//默认True

 // 默认:D3D11\_DEPTH\_WRITE\_MASK\_ALL

 D3D11\_DEPTH\_WRITE\_MASK DepthWriteMask;

 // 默认:D3D11\_COMPARISON\_LESS

 D3D11\_COMPARISON\_FUNC DepthFunc;

 BOOL StencilEnable;// 默认:False

 UINT8 StencilReadMask;// 默认:0xff

 UINT8 StencilWriteMask;// 默认:0xff

 D3D11\_DEPTH\_STENCILOP\_DESC FrontFace;

 D3D11\_DEPTH\_STENCILOP\_DESC BackFace;

}D3D11\_DEPTH\_STENCIL\_DESC;

## 10.3.1 深度设置

**1．DepthEnable**：当设为true时，表示启用深度测试；当设为false时，表示禁用深度测试。当禁用深度测试时，绘图顺序非常重要，因为在这种情况下障碍物后面的像素片段也会被绘制出来（回顾4.1.5节）。如果禁用深度测试，那么无论**DepthWriteMask**设定何值，深度缓冲区中的元素都不会被更新。

**2．DepthWriteMask**：可设为**D3D11\_DEPTH\_WRITE\_MASK\_ZERO**或**D3D11\_DEPTH\_WRITE\_MASK\_ALL**。这两个标志值不能同时使用。当**DepthEnable**设为true时，**D3D11\_DEPTH\_WRITE\_MASK\_ZERO**表示禁用深度缓冲区的写入功能，但深度测试依然有效。**D3D11\_DEPTH\_WRITE\_MASK\_ALL**表示启用深度缓冲区的写入功能；当深度/模板测试都通过时，新的深度值会被写入深度缓冲区。

**3．DepthFunc**：一个用于描述深度测试函数的**D3D11\_COMPARISON\_FUNC**枚举类型成员。我们一般使用**D3D11\_COMPARISON\_LESS**实现普通的深度测试，如4.1.5节所述。也就是，当像素片段的深度值比之前写入后台缓冲区的像素的深度值小时，接受该像素片段（即，将该像素片段写入后台缓冲区，将该像素片段的深度值写入深度缓冲区）。另外， Direct3D支持自定义的深度测试函数。如果有必要的话，你可以自定义深度测试函数。

## 10.3.2 模板设置

**1．StencilEnable**：当设为true时，表示启用模板测试；当设为false时，表示禁用模板测试。

**2．StencilReadMask**：在模板测试时使用的掩码：

if( StencilRef & StencilReadMask ⊴ Value &StencilReadMask)

 accept pixel

else

 reject pixel

默认掩码不屏蔽任何二进制位：

#define D3D11\_DEFAULT\_STENCIL\_READ\_MASK (0xff)

**3．StencilWriteMask**： 当更新模板缓冲区时，我们可以通过掩码来屏蔽某些二进制位，不让它们存入模板缓冲区。例如，当你希望屏蔽前4位数据时，可将掩码设为0x0f。默认掩码不屏蔽任何二进制位：

#define D3D11\_DEFAULT\_STENCIL\_WRITE\_MASK (0xff)

**4．FrontFace**：一个已填充的**D3D11\_DEPTH\_STENCILOP\_DESC**结构体，它告诉模板缓冲区如何处理朝前的三角形。

**5．BackFace**：一个已填充的**D3D11\_DEPTH\_STENCILOP\_DESC**结构体，它告诉模板缓冲区如何处理朝后的三角形。

typedef struct D3D11\_DEPTH\_STENCILOP\_DESC {

 D3D11\_STENCIL\_OP StencilFailOp;// Default:D3D11\_STENCIL\_OP\_KEEP

 D3D11\_STENCIL\_OP StencilDepthFailOp;// Default:D3D11\_STENCIL\_OP\_KEEP

 D3D11\_STENCIL\_OP StencilPassOp; // Default:D3D11\_STENCIL\_OP\_KEEP

 D3D11\_COMPARISON\_FUNC StencilFunc; // Default:D3D11\_ COMPARISON\_ALWAYS

} D3D11\_DEPTH\_STENCILOP\_DESC;

**1．StencilFailOp**：**D3D11\_STENCIL\_OP**枚举类型成员，它描述了当一个像素片段的模板测试失败时，应如何更新模板缓冲区。

**2．StencilDepthFailOp**：**D3D11\_STENCIL\_OP**枚举类型成员，它描述了当一个像素片段的模板测试成功而深度测试失败时，应如何更新模板缓冲区。

**3．StencilPassOp**：**D3D11\_STENCIL\_OP**枚举类型成员，它描述了当一个像素片段的模板测试和深度测试均成功时，应如何更新模板缓冲区。

**4．StencilFunc**：**D3D11\_COMPARISON\_FUNC**枚举类型成员，指定模板测试时使用的比较函数。

typedef enum D3D11\_STENCIL\_OP

{

 D3D11\_STENCIL\_OP\_KEEP = 1,

 D3D11\_STENCIL\_OP\_ZERO = 2,

 D3D11\_STENCIL\_OP\_REPLACE = 3,

 D3D11\_STENCIL\_OP\_INCR\_SAT = 4,

 D3D11\_STENCIL\_OP\_DECR\_SAT = 5,

 D3D11\_STENCIL\_OP\_INVERT = 6,

 D3D11\_STENCIL\_OP\_INCR = 7,

 D3D11\_STENCIL\_OP\_DECR = 8,

} D3D11\_STENCIL\_OP;

**1．D3D11\_STENCIL\_OP\_KEEP**：不更新模板缓冲区；也就是，当前值保持不变。

**2．D3D11\_STENCIL\_OP\_ZERO**：将模板缓冲区元素设为0。

**3．D3D11\_STENCIL\_OP\_REPLACE**：以模板测试中的模板参考值（**StencilRef**）替换模板缓冲区元素。注意，当我们将深度/模板状态块绑定到渲染管线上时（参见10.3.3节），**StencilRef**值就已经被确定下来了。

4．**D3D11\_STENCIL\_OP\_INCR\_SAT**：递增模板缓冲区元素。如果递增之后的值大于最大值（比如，255是8位模板缓冲区的最大值），则将其舍入为最大值。

**5．D3D11\_STENCIL\_OP\_DECR\_SAT**：递减模板缓冲区元素。如果递减之后的值小于 0，则将其舍入为0。

**6．D3D11\_STENCIL\_OP\_INVERT**：反转模板缓冲区元素的二进制位。

**7．D3D11\_STENCIL\_OP\_INCR**：递增模板缓冲区元素。如果递增之后的值大于最大值（比如，255是8位模板缓冲区的最大值），则将其折反为0。

**8．D3D11\_STENCIL\_OP\_DECR**：递减模板缓冲区元素。如果递减之后的值小于0，则将其折反为最大值。

**注意**：我们一般不使用**BackFace**参数，因为当启用背面消隐时，Direct3D根本不会渲染朝后的多边形。不过，有时我们会为了实现某些绘图算法或绘制透明几何体（比如在绘制铁丝立方体时，我们希望看到铁丝网立方体的背面）而渲染朝后的多边形。在这种情况下应该使用**BackFace**参数。

## 10.3.3 创建和绑定深度/模板状态

在填充**D3D11\_DEPTH\_STENCIL\_DESC**结构体之后，我们可以调用如下方法获取一个指向**ID3D11DepthStencilState**接口的指针：

HRESULT ID3D11Device::CreateDepthStencilState(

 const D3D11\_DEPTH\_STENCIL\_DESC \*pDepthStencilDesc,

 ID3D11DepthStencilState \*\*ppDepthStencilState);

**1．pDepthStencilDesc**：一个已填充的**D3D11\_DEPTH\_STENCIL\_DESC**结构体的指针，该结构体描述了所要创建的深度/模板状态块。

**2．ppDepthStencilState**：返回创建后的**ID3D11DepthStencilState**接口的指针。

在创建**ID3D11DepthStencilState**接口后，我们使用如下方法将它绑定到管线的输出合并器阶段：

void ID3D11DeviceContext::OMSetDepthStencilState(

 ID3D11DepthStencilState \*pDepthStencilState,

 UINT StencilRef);

**1．pDepthStencilState**：深度/模板状态块的指针。

**2．StencilRef**：模板测试使用的32位模板参考值。

与其他状态块相同，深度/模板状态也有一个默认值（它使用普通的深度测试，并禁用模板测试）。通过给**OMSetDepthStencilState**方法的第1个参数传递一个空值就可以将深度/模板状态恢复为默认值。

// 恢复为默认值

md3dImmediateContext->OMSetDepthStencilState(0, 0);

## 10.3.4 effect文件中的深度/模板状态

在effect文件中可以直接定义和设置深度/模板状态：

DepthStencilState DSS

{

 DepthEnable = true;

 DepthWriteMask = Zero;

 StencilEnable = true;

 StencilReadMask = 0xff;

 StencilWriteMask = 0xff;

 FrontFacetencilFunc = Always;

 FrontFaceStencilPass = Incr;

 FrontFaceStencilFail = Keep;

 BackFaceStencilFunc = Always;

 BackFaceStencilPass = Incr;

 BackFaceStencilFail = Keep;

} ;

...

technique11 Tech

{

 pass P0

 {

 SetVertexShader(CompileShader(vs\_5\_0,VS()));

 SetGeometryShader(NULL);

 SetPixelShader(CompileShader(ps\_5\_0,PS()));

 SetDepthStencilState(DSS, 0);

 }

}

在深度/模板状态对象中指定的些值与在C++结构体中指定的值基本相同，只是省去了一些前缀。例如，我们在效果代码中指定是**INCR**，而不是**D3D11\_STENCIL\_OP\_INCR**。顺便提一句，我们指定的状态值是不区分大小写的；例如，INCR等价于Incr。