# 9.4 混合状态

我们已经讨论了混合运算符和混合系数，但是还没有说该如何在Direct3D中使用些值。这些混合参数要通过**ID3D11BlendState**接口来控制。我们可以通过填充一个**D3D11\_BLEND\_DESC**结构体并调用**ID3D11Device::CreateBlendState**方法来创建该接口：

HRESULT ID3D11Device::CreateBlendState(

const D3D11\_BLEND\_DESC \*pBlendStateDesc,ID3D10BlendState \*\*ppBlendState);

**1．pBlendStateDesc**：指向D3D11\_BLEND\_DESC结构体的指针，该结构体用于描述所要创建的混合状态。

**2．ppBlendState**：返回创建后的混合状态接口。

**D3D11\_BLEND\_DESC**结构体的定义如下：

typedef struct D3D11\_BLEND\_DESC {

BOOL AlphaToCoverageEnable; // 默认值：False

IndependentBlendEnable // 默认值：False

D3D11\_RENDER\_TARGET\_BLEND\_DESC RenderTarget[8];

} D3D11\_BLEND\_DESC;

**1．AlphaToCoverageEnable**：当设为true时，表示启用alpha-to-coverage功能。它是一种多重采样技术，在渲染植物的叶子或铁丝网纹理时非常有用。当设为false时，表示禁用 alpha-to-coverage功能。使用alpha-to-coverage需要开启多重采样（即，后台和深度缓冲创建时需要开启多重采样）。在第11章有一个使用alpha-to-coverage的示例。

**2．IndependentBlendEnable**：Direct3D 11支持同时绘制到8个渲染目标。当这个标志设置为true时，表示可以在不同的渲染目标上进行不同的混合处理（不同的混合因子、混合操作，混合开启/关闭等）。如果设置为false，则表示所有渲染目标都使用**D3D11\_BLEND\_DESC::RenderTarget**数组中第一个元素的混合状态，多重渲染目标用于高级的算法，目前为止，我们一次只使用一个渲染目标。

3．**RenderTarget**：包含8个**D3D11\_RENDER\_TARGET\_BLEND\_DESC**元素的数组，第*i*个元素描述了第*i*个多重渲染目标的混合方式。如果**IndependentBlendEnable**设置为false，则所有的渲染目标都使用**RenderTarget[0]**进行混合。

**D3D11\_RENDER\_TARGET\_BLEND\_DESC**结构体的定义如下：

typedef struct D3D11\_RENDER\_TARGET\_BLEND\_DESC{

BOOL BlendEnable; // 默认值:False

D3D11\_BLEND SrcBlend; // 默认值:D3D11\_BLEND\_ONE

D3D11\_BLEND DestBlend; // 默认值Default:D3D11\_BLEND\_ZERO

D3D11\_BLEND\_OP BlendOp; // 默认值:D3D11\_BLEND\_OP\_ADD

D3D11\_BLEND SrcBlendAlpha; // 默认值:D3D11\_BLEND\_ONE

D3D11\_BLEND DestBlendAlpha; // 默认值:D3D11\_BLEND\_ZERO

D3D11\_BLEND\_OP BlendOpAlpha; // 默认值:D3D11\_BLEND\_OP\_ADD

UINT8 RenderTargetWriteMask; // 默认值:D3D11\_COLOR\_WRITE\_ENABLE\_ALL

}D3D11\_RENDER\_TARGET\_BLEND\_DESC;

**1．BlendEnable**：当设为true时，表示启用混合功能；当设为false时，表示禁用混合功能。

**2．SrcBlend**：**D3D11\_BLEND**枚举类型成员，用于为RGB混合指定源混合系数**F**src。

**3．DestBlend**：**D3D11\_BLEND**枚举类型成员，用于为RGB混合指定目标混合系数**F**dst。

**4．BlendOp**：**D3D11\_BLEND\_OP**枚举类型成员，用于指定RGB混合运算符。

**5．SrcBlendAlpha**：**D3D11\_BLEND**枚举类型成员，用于为alpha混合指定源混合系数*F*src。

**6．DestBlendAlpha**：**D3D11\_BLEND**枚举类型成员，用于为alpha混合指定目标混合系数*F*dst。

**7．BlendOpAlpha**：**D3D11\_BLEND\_OP**枚举类型成员，用于指定alpha混合运算符。

**8．RenderTargetWriteMask**：一个或多个下列标志值的组合：

typedef enum D3D11\_COLOR\_WRITE\_ENABLE

{

D3D11\_COLOR\_WRITE\_ENABLE\_RED = 1,

D3D11\_COLOR\_WRITE\_ENABLE\_GREEN = 2,

D3D11\_COLOR\_WRITE\_ENABLE\_BLUE = 4,

D3D11\_COLOR\_WRITE\_ENABLE\_ALPHA = 8,

D3D11\_COLOR\_WRITE\_ENABLE\_ALL =

(D3D11\_COLOR\_WRITE\_ENABLE\_RED|D3D11\_COLOR\_WRITE\_ENABLE\_GREEN|

D3D11\_COLOR\_WRITE\_ENABLE\_BLUE |

D3D11\_COLOR\_WRITE\_ENABLE\_ALPHA)

} D3D11\_COLOR\_WRITE\_ENABLE;

这些标志值用于控制混合之后将哪些颜色分量写入后台缓冲区。例如，通过**D3D11\_COLOR\_WRITE\_ENABLE\_ALPHA**可以屏蔽RGB通道，只将alpha值写入后台缓冲区。这一功能在实现某些高级技术时非常有用。当禁用混合时，由像素着色器返回的颜色会被屏蔽掉。

要将混合状态对象绑定到管线的输出合并器阶段，我们可以调用：

void ID3D11DeviceContext::OMSetBlendState(

ID3D11BlendState \*pBlendState,

const FLOAT BlendFactor,

UINT SampleMask);

**1．pBlendState**：混合状态对象的指针。

**2．BlendFactor**：用于描述RGBA颜色向量的浮点数组。当混合因子指定为**D3D11\_BLEND\_BLEND\_FACTOR**或**D3D11\_BLEND\_INV\_BLEND\_FACTOR**时，Direct3D将以该颜色向量作为混合系数。

**3．SampleMask**：多重采样最多可以支持32个采样源。这个32位整数用于启用和禁用采样源。例如，当第5个二进制位设为0时，表示屏蔽第5个采样源。当然，如果实际使用的多重采样源数量少于5个，那么屏蔽第5个采样源是没有什么实际意义的。当应用程序只使用一个采样源时，该参数只有第1个二进制位有效（参见练习1）。通常，该参数以0xffffffff作为默认值，表示不屏蔽任何采样源。

与其他状态块相同，这里有一个默认的混合状态（禁用混合）；如果以空值来调用**OMSetBlendState**方法，它就会将混合状态恢复为默认值。注意，混合会在每个像素上执行额外的计算工作，所以我们只有在用到混合时才启用它，用完之后应该立即关闭。

下面是创建和设置混合状态的一个例子：

D3D11\_BLEND\_DESC blendDesc = {0};

transparentDesc.AlphaToCoverageEnable = false;

transparentDesc.IndependentBlendEnable = false;

transparentDesc.RenderTarget[0].BlendEnable = true;

transparentDesc.RenderTarget[0].SrcBlend = D3D10\_BLEND\_SRC\_ALPHA;

transparentDesc.RenderTarget[0] = D3D11\_BLEND\_INV\_SRC\_ALPHA;

transparentDesc.RenderTarget[0] = D3D11\_BLEND\_OP\_ADD;

transparentDesc.RenderTarget[0] = D3D11\_BLEND\_ONE;

transparentDesc.RenderTarget[0].DestBlendAlpha = D3D11\_BLEND\_ZERO;

transparentDesc.RenderTarget[0].BlendOpAlpha = D3D11\_BLEND\_OP\_ADD;

transparentDesc.RenderTarget[0].RenderTargetWriteMask[0] = D3D10\_COLOR\_WRITE\_ENABLE\_ALL;

ID3D11BlendState\* TransparentBS;

HR(device->CreateBlendState(&transparentDesc, &TransparentBS));

...

float blendFactor[] = {0.0f, 0.0f, 0.0f, 0.0f};

md3dImmediateContect->OMSetBlendState(TransparentBS, blendFactor, 0xffffffff);

与其他状态块接口一样，我们应该在应用程序初始化时创建它们，然后根据需要在些状态接口之间进行切换。

混合状态对象也可以在effect文件中创建和设定：

BlendState blend

{

// 第一个渲染目标的混合状态

BlendEnable[0] = TRUE;

SrcBlend[0] = SRC\_COLOR;

DestBlend[0] = INV\_SRC\_ALPHA;

BlendOp[0] = ADD;

SrcBlendAlpha[0] = ZERO;

DestBlendAlpha[0] = ZERO;

BlendOpAlpha[0] = ADD;

RenderTargetWriteMask[0] = 0x0F;

// 第二个渲染目标的混合状态

BlendEnable[1] = True;

SrcBlend[1] = One;

DestBlend [1] = Zero;

BlendOp[1] = Add;

SrcBlendAlpha[1] = Zero;

DestBlendAlpha[1] = Zero;

BlendOpAlpha[1] = Add;

RenderTargetWriteMask[1] = 0x0F;

};

technique11 Tech

{

pass P0

{

...

// 在这个pass中使用“混合”。

SetBlendState(blend,float4(0.0f,0.0f,0.0f,0.0f),0xffffffff);

}

}

在混合状态对象中指定的这些值与在C++结构体中指定的值基本相同，只是省去了一些前缀。例如，在effect文件中我们指定**SRC\_COLOR**，而不是 **D3D11\_BLEND\_SRC\_COLOR**。