**10 混合**

混合技术使我们可以将当前要光栅化(源像素)的像素与之前已光栅化至后台缓冲区的像素(目标像素)相融合。该技术可用于渲染如水与玻璃之类的半透明物体。

**10.1 混合方程**

设Csrc为像素着色器输出的当前正在光栅化的第i行，第j列像素(源像素)的颜色值，再设Cdst为目前在后台缓冲区中与之对应的第i行，第j列像素(目标像素)的颜色值。

若不采用混合技术，Csrc将直接覆写Cdst。若采用了混合技术，则Csrc与Cdst将融合在一起得到新颜色值C后再覆写Cdst。

Direct3D使用下列混合方程来使源像素颜色与目标像素颜色相融合：

RGB方程：C = Csrc x Fsrc 田 Cdst x Fdst

alpha分量方程：A = Asrc x Fsrc 田 Adst x Fdst

这两组方程本质上是相同的，但区别在于混合因子有所差异。将RGB分量与alpha分量分离开来的动机也比较简单，就是希望能独立地处理两者，尽可能多地产生不同的混合变化效果。

**10.2 混合运算**

D3D12\_BLEND\_OP枚举项将用作混合方程中的二元运算符田。

这些运算符也同样适用于alpha混合运算。我们还能同时为RGB和aplha这两种运算分别指定不同的运算符。

Direct3D从最近几版开始加入了一项新特性，通过逻辑运算符D3D12\_LOGIC\_OP对源颜色和目标颜色进行混合，用以取代上述传统的混合方程。

不能同时使用传统混合方程与逻辑运算符这两种混合手段，两者只能择其一。

**10.3 混合因子**

D3D12\_BLEND枚举类型

**10.4 混合状态**

混合状态也是PSO(流水线状态对象)的一部分。

到目前为止，我们一直使用的都是默认的混合状态，并没有启用混合技术。

为了配置非默认混合状态，我们必须填写D3D12\_BLEND\_DESC结构体。

(D3D12\_BLEND\_DESC各参数含义)

(D3D12\_RENDER\_TARGET\_BLEND\_DESC各参数含义)

(创建和设置混合状态例子)

**10.5 混合示例**

**10.5.1 禁止颜色的写操作**

把源像素的混合因子设置为D3D12\_BLEND\_ZERO，将目标混合因子配置为D3D12\_BLEND\_ONE，再令混合运算符为D3D12\_BLEND\_OP\_ADD即可。

更简单方法，将成员D3D12\_RENDER\_TARGET\_BLEND\_DESC::RenderTargetWriteMask设置为0。

**10.5.2 加法混合与减法混合**

如果希望令源像素与目标像素实现加法运算，那么就将源混合因子与目标混合因子同设为D3D12\_BLEND\_ONE，再把混合运算符设置为D3D12\_BLEND\_OP\_ADD。

还可以继续使用上述混合因子，唯令D3D12\_BLEND\_OP\_SUBTRACT来取代其中的加法混合运算符，以此来达到从目标像素中减去源像素的这一目的。

**10.5.3 乘法混合**

设源混合因子为D3D12\_BLEND\_ZERO，目标混合因子为D3D12\_BLEND\_SRC\_COLOR，再将混合运算符置为D3D12\_BLEND\_OP\_ADD。

**10.5.4 透明混合**

设源alpha分量as为一种可用来控制源像素不透明度的百分比。

基于源像素的不透明度，将源像素与目标像素进行混合。为了实现此效果，设源混合因子为D3D12\_BLEND\_SRC\_ALPHA，目标混合因子为D3D12\_BLEND\_INV\_SRC\_ALPHA，并将混合运算符置为D3D12\_BLEND\_OP\_ADD。

使用此混合方法时，还应当考虑物体的绘制顺序。对此，我们遵循以下规则：

首先绘制无须混合处理的物体。接下来，再根据混合物体与摄像机的距离对它们进行排序。最后，按由远到近的顺序通过混合的方式绘制这些物体。

**10.5.5 混合与深度缓冲区**

在使用加法/减法/乘法运算进行混合时，会涉及深度测试这一问题。

进行混合时，不能开启深度测试。通过禁止向深度缓冲区的写操作来禁用物体之间的深度测试。禁用深度值写入操作，但深度值读取与深度检测仍然是开启的。

**10.6 alpha通道**

源alpha分量能够用于在RGB混合的过程中控制像素的透明度。

我们往往可以在常见的图像编辑软件中添加alpha通道，接着再将图像保存为支持alpha通道的格式，比如DDS。

**10.7 裁剪像素**

有时候，我们希望彻底禁止某个源像素参与后续的处理。这可以通过HLSL的内置函数clip(x)来实现。

此函数仅供像素着色器调用，若x<0,则当前这一像素将从后面的处理阶段中丢弃。

(使用clip的例子)

alpha测试的开销并不小，有必要情况下再使用。通过混合操作也能实现相同效果，但是使用clip函数更为有效。

**10.8 雾**

(代码见书本)