# 6.5 常量缓冲

在上一节的顶点着色器示例中包含如下代码：

cbuffer cbPerObject

{

 float4x4 gWorldViewProj;

};

这段代码定义了一个称为cbPerObject的**cbuffer**对象（constant buffer，常量缓冲）。常量缓冲只是一个用于存储各种变量的数据块，这些变量可以由着色器来访问。在本例中，常量缓冲区只存储了一个称为**gWorldViewProj**的4×4矩阵，它是世界矩阵、观察矩阵和投影矩阵的组合矩阵，用于将顶点从局部空间变换到齐次裁剪空间。在HLSL中，4×4矩阵由内置的**float4x4**类型表示；与之类似，要定义一个3×4矩阵和一个2×2矩阵，可以分别使用**float3x4**和**float2x2**类型。顶点着色器不能修改常量缓冲中的数据，但是通过effect框架（6.9节），C++应用程序代码可以在运行时修改常量缓冲中的内容。它为C++应用程序代码和effect代码提供了一种有效的通信方式。例如，因为每个物体的世界矩阵各不相同，所以每个物体的“WVP”组合矩阵也各不相同；所以，当使用上述顶点着色器绘制多个物体时，我们必须在绘制每个物体前修改**gWorldViewProj**变量。

通常的建议是根据变量修改的频繁程度创建不同的常量缓冲。比如，你可以创建下面的常量缓冲：

cbuffer cbPerObject

{

 float4x4 gWVP;

};

cbuffer cbPerFrame

{

 float3 gLightDirection;

 float3 gLightPosition;

 float4 gLightColor;

};

cbuffer cbRarely

{

 float4 gFogColor;

 float gFogStart;

 float gFogEnd;

};

在本例中，我们使用了3个常量缓冲区。第1个常量缓冲区存储“WVP”组合矩阵。该变量随物体而定，所以它必须在物体级别上更新。也就是，当我们每帧渲染100个物体时，每帧都要对这个变量更新100次。第2个常量缓冲存储了场景中的灯光变量。这里，我们假设要生成灯光动画，所以些变量必须在每帧中更新一次。最后一个常量缓冲存储了用于控制雾效的变量。这里，我们假设场景的雾效变化频率很低（例如，在游戏的一个特定时段中变化一次）。

对常量缓冲进行分组是为了提高运行效率。当一个常量缓冲区被更新时，它里面的所有变量都会同时更新；所以，根据它们的更新频率进行分组，可以减少不必要的更新操作，提高运行效率。