3.10 计算着色器

GPU不仅可以用于实现传统的图形管道。在计算股票期权的估计价值和训练深度学习的神经网络等领域有许多非图形用途。以这种方式使用硬件称为GPU计算。CUDA和OpenCL等平台用于将GPU作为大规模并行处理器进行控制，而无需或访问特定于图形的功能。这些框架通常使用带有扩展的C或C++等语言，以及为GPU制作的库。

DirectX 11中引入的计算着色器是GPU计算的一种形式，因为它是一种未锁定到图形管线中某个位置的着色器。它与渲染过程密切相关，因为它由图形API调用。它与顶点、像素和其他着色器一起使用。它利用与管线中使用的相同的统一着色器处理器池。它是一个与其他着色器一样的着色器，因为它具有一些输入数据集并且可以访问缓冲区（例如纹理）以进行输入和输出。Warp和线程在计算着色器中更加明显。例如，每个调用都会获得一个它可以访问的线程索引。还有线程组的概念，在DirectX 11中由1到1024个线程组成。这些线程组由x、y和z坐标指定，主要是为了在着色器代码中方便的使用。每个线程组都有少量的在线程之间共享的内存。在DirectX 11中，这相当于32 kB。计算着色器由线程组执行，从而保证组中的所有线程并发运行[1971]。

计算着色器的一项重要优势是它们可以访问在GPU上生成的数据。将数据从GPU发送到CPU会产生延迟，因此如果处理和结果可以驻留在GPU上，则可以提高性能[1403]。以某种方式修改渲染图像的后处理是计算着色器的常见用途。共享内存意味着采样图像像素的中间结果可以与相邻线程共享。一个被认可的例子是，使用计算着色器来确定图像的分布或平均亮度，其运行速度是在像素着色器上执行此操作的两倍[530]。

计算着色器也可用于粒子系统、网格处理（如面部动画[134]、剔除[1883、1884]、图像过滤[1102、1710]、提高深度精度[991]、阴影[865]、景深[764]，以及可以使用一组GPU处理器的任何其他任务。Wihlidal在文献[1884]中讨论了计算着色器如何比曲面细分中的外壳着色器更有效。其他用途请参见图3.16。

图3.16. (Images from NVIDIA SDK 11 [1301] samples, courtesy of NVIDIA Corporation.)计算着色器示例。左侧图片，计算着色器用于模拟受风影响的头发，头发本身使用曲面细分阶段进行渲染。中间图片，计算着色器执行快速模糊操作。右侧图片，模拟海浪。（图片来自NVIDIA SDK 11[1301]的案例，由NVIDIA Corporation提供。）

我们对GPU渲染管线实现的回顾到此结束。可以通过多种方式使用和组合GPU功能来执行各种与渲染相关的过程。为利用这些功能而调整的相关理论和算法是本书的中心主题。我们现在的重点转向变换和着色。