扩展光栅器

我们将以与第一部分相同的方式结束本书的第二部分：对我们在前几章中开发的光栅器进行一组可能的扩展。

法线贴图

在第 13 章（阴影）中，我们看到了表面的法线矢量如何对其外观产生重大影响。例如，正确选择法线可以使多面对象看起来平滑弯曲;这是因为正确选择法线会改变光线与表面相互作用的方式，进而改变我们的大脑猜测物体形状的方式。遗憾的是，除了使表面看起来平滑弯曲之外，通过插值法线我们能做的不多。

在第 14 章（纹理）中，我们看到了如何通过在表面上“绘画”来向表面添加虚假细节。这种称为纹理映射的技术使我们能够对表面的外观进行更精细的控制。但是，纹理映射不会改变三角形的形状 - 它们仍然是平坦的。

*法线贴图*结合了这两种想法。我们可以使用法线来改变光与表面相互作用的方式，从而改变表面的表观形状;我们可以使用属性映射将属性的不同值分配给三角形的不同部分。通过结合这两个想法，法线贴图允许我们在像素级别定义表面法线。

为此，我们将*法线贴图*关联到每个三角形。法线贴图类似于纹理贴图，但其元素是法线向量而不是颜色。在渲染时，我们不是像 Phong 着色那样计算插值法线，而是使用法线贴图来获取要渲染的特定像素的法线矢量，就像纹理映射获取该特定像素的颜色一样。然后我们使用此向量来计算该像素处的照明。

图 15-1 显示了应用了纹理贴图的平面，以及同时应用法线贴图时不同光照方向的效果。

图 15-1 中的所有三个图像都是具有纹理的平面正方形（即两个三角形）的渲染图，如 （a） 所示。当我们添加法线贴图和适当的每像素阴影时，我们会产生额外几何细节的错觉。在 （b） 和 （c） 中，钻石的阴影取决于入射光的方向，我们的大脑将其解释为钻石具有体积。

有几个实际注意事项需要牢记。首先，法线贴图中向量的方向相对于它们所应用的三角形的表面。用于此目的的坐标系称为*切线空间*，其中两个轴（通常X和Z） 与曲面相切（即嵌入其中），其余向量垂直于曲面。在渲染时，三角形的法线向量（以相机空间表示）根据法线贴图中的向量进行修改，以获得可用于照明方程的最终法线向量。这使得法线贴图独立于场景中对象的位置和方向。

其次，一种非常流行的法线贴图编码方法是作为纹理，映射X,Y和Z自R,G和B值。这使法线贴图具有非常典型的紫色外观，因为紫色（红色和蓝色的组合，但没有绿色）编码表面的平坦区域。图 15-2 显示了图 15-1 中示例中使用的法线贴图。

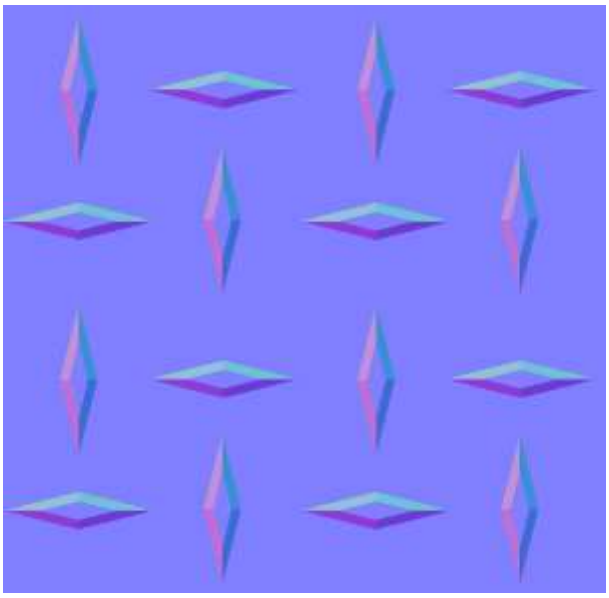

图 15-2：用于图 15-1 中示例的法线贴图，编码为 RGB 纹理

虽然这种技术可以大大提高场景中表面的感知复杂性，但它并非没有限制。例如，由于平面保持平坦，因此无法更改对象的轮廓。出于同样的原因，当从极端角度或近距离查看法线贴图表面时，或者当法线贴图表示的要素与表面大小相比太大时，错觉会崩溃。这种技术更适合微妙的细节，例如皮肤上的毛孔、灰泥墙上的图案或橙皮的不规则外观。因此，该技术也称为*凹凸贴图*。

环境映射

我们开发的光线追踪器最引人注目的特点之一是能够显示相互反射的物体。可以在我们的光栅器中创建相对令人信服但有些虚假的反射实现。

想象一下，我们有一个代表房子里房间的场景，我们想渲染一个放置在房间中间的反射对象。对于代表该物体表面的每个像素，我们知道它所代表的点的 3D 坐标，该点的表面法线，并且由于我们知道相机的位置，我们还可以计算到该点的视图矢量。我们可以相对于表面法线反射视图向量以获得反射向量，就像我们在第 4 章（阴影和反射）中所做的那样。

此时，我们想知道来自反射矢量方向的光的颜色。如果这是一个光线追踪器，我们只会沿着那个方向追踪一条光线并找出答案。但是，这不是光线追踪器。怎么办？

*环境映射*为这个问题提供了一个可能的答案。假设在渲染房间内的物体之前，我们将摄像机放在中间，并渲染场景六次——每个垂直方向（上、下、左、右、前、后）渲染一次。您可以想象摄像机位于一个假想的立方体内，立方体的每一侧都是其中一个渲染的视口。我们将这六个渲染保留为纹理。我们将这组六种纹理称为立方体*贴图*，这就是为什么这种技术也称为*立方体映射*。

然后我们渲染反射对象。当我们到达需要反射颜色的地步时，我们可以使用反射矢量的方向来选择立方体贴图的纹理之一，然后使用该纹理的纹素来获得在该方向上看到的颜色的近似值 - 所有这些都无需追踪一条光线！

这种技术有一些缺点。立方体贴图从单个点捕获场景的外观。如果我们渲染的反射对象不在那个点，反射对象的位置将不会完全符合我们的预期，因此很明显这只是一个

近似值。如果反射物体在房间内移动，这一点尤其明显，因为反射场景不会随着物体的移动而改变。

这种限制也表明了该技术的最佳应用：如果“房间”足够大并且离对象足够远，也就是说，如果对象的移动相对于房间的大小很小，则真实反射和预渲染环境贴图之间的差异可能会被忽视。例如，这对于代表深空中反射宇宙飞船的场景非常有效，因为“房间”（遥远的恒星和星系）对于所有实际目的来说都是无限遥远的。

另一个缺点是，我们被迫将场景中的对象分为两类：作为“房间”一部分的静态对象（在反射中看到）和动态对象（可以反射）。在某些情况下，这可能很清楚（墙壁和家具是房间的一部分;人不是），但即便如此，动态对象也不会反映在其他动态对象上。

值得一提的最后一个缺点与立方体贴图的分辨率有关。而在光线追踪器中，我们可以追踪非常精确的反射，在这种情况下，我们需要在精度（更高分辨率的立方体贴图纹理产生更清晰的反射）和内存消耗（更高分辨率的立方体贴图纹理需要更多的内存）之间进行权衡。实际上，这意味着环境贴图不会产生与真实光线追踪反射一样清晰的反射，尤其是在近距离观察反射对象时。

阴影

我们开发的光线追踪器具有几何正确、非常清晰的阴影。这些是对核心算法的非常自然的扩展。光栅器的体系结构使实现阴影稍微复杂一些，但并非不可能。

让我们从正式确定我们试图解决的问题开始。为了正确渲染阴影，每次我们计算像素和光源的照明方程时，我们都需要知道像素是否实际被光线照亮，或者相对于该光线，它是否在物体的阴影中。

使用光线追踪器，我们可以通过追踪从表面到光线的光线来回答这个问题;在光栅器中，我们没有这样的工具，因此我们将不得不采取不同的方法。让我们探讨两种不同的方法。

模具阴影

*模具*阴影是一种渲染边缘非常明确的阴影的技术（想象一下在阳光明媚的日子里物体投射的阴影）。这些通常被称为*硬阴影*。

我们的光栅器在一次*传递*中渲染场景;它遍历场景中的每个三角形并将其渲染到画布上，每次都计算完整的照明方程（基于每个三角形、每个顶点或每个像素，具体取决于着色算法）。在此过程结束时，画布包含场景的最终渲染。

我们将首先修改光栅器，以分多个通道渲染场景，场景中的每个光源（包括环境光）对应一个*通道*。像以前一样，每个通道都经过每个三角形，但它计算照明方程时仅考虑与该通道相关的光。

这为我们提供了一组由每个灯分别照亮的场景图像。我们可以将它们*组合*在一起，即逐个像素地添加它们，从而获得场景的最终渲染。此最终图像与单程版本生成的图像相同。图 15-3 显示了参考场景的三个光线通道和最终合成。

这使我们能够将“使用来自多个光源的阴影渲染场景”的目标简化为“多次使用单个光源的阴影渲染场景”。现在我们需要找到一种方法来渲染由单个光源照亮的场景，

同时使该光源阴影中的像素完全变黑。

为此，我们引入了*模板缓冲区*。与深度缓冲区一样，它具有与画布相同的尺寸，但其元素是整数。我们可以将其用作渲染操作的模具，例如，仅当模板缓冲区中的相应元素的值为零时，才修改渲染代码以在画布上绘制像素。

如果我们可以设置模具缓冲区，使发光像素的值为零，阴影中的像素具有非零值，则可以使用它来仅绘制被照亮的像素。

创建卷影卷

为了设置模板缓冲区，我们使用称为*影子卷*的东西。阴影体积是一个 3D 多边形，“包裹”在光源阴影中的空间体积周围。

我们为每个可能在场景中投射阴影的对象构建一个阴影体积。首先，我们确定哪些边缘是对象轮廓的一部分;这些是正面和背面三角形之间的边（我们可以使用点积对三角形进行分类，就像我们在第 12 章（隐藏表面去除）中对背面剔除技术所做的那样）。然后，对于这些边缘中的每一个，我们将它们从光的方向拉伸出来，一直延伸到无穷大——或者，在实践中，延伸到场景之外的一个非常大的距离。

这为我们提供了阴影体积的“侧面”。体积的“正面”由对象本身的正面三角形构成，体积的“背面”可以通过创建一个多边形来计算，该多边形的边缘是拉伸边的“远”边。

图 15-4 显示了以这种方式为立方体相对于点光源创建的阴影体积。

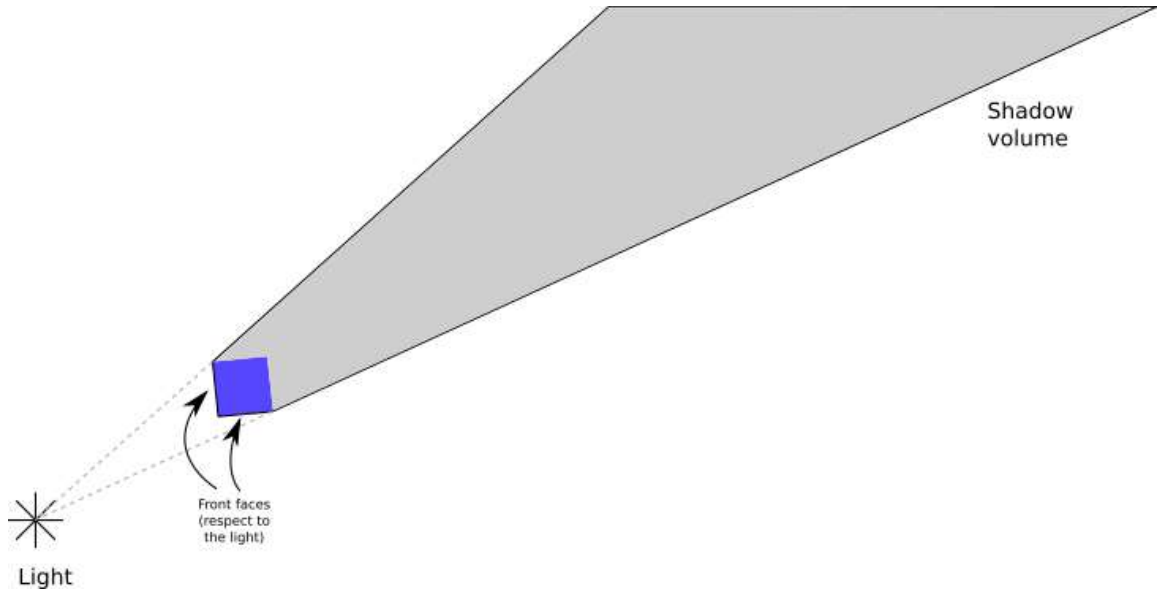

图 15-4：立方体相对于点光源的阴影体积

接下来，我们将了解如何使用阴影体积来确定画布中的哪些像素相对于光源处于阴影中。

计算阴影体积射线交集

想象一下，一条光线从摄像机开始进入场景，直到它击中表面。在此过程中，它可能会进入和离开任意数量的影子卷。

我们可以使用从零开始的计数器来跟踪这一点。每次光线进入阴影体积时，我们都会递增计数器;每次它离开时，我们都会减少它。当光线照射到表面时，我们停下来看着柜台。如果为零，则表示光线进入的阴影体积与它离开的阴影体积一样多，因此必须照亮该点;如果它不为零，则表示光线至少在一个阴影体积内，因此该点必须在阴影中。图 15-5 显示了其中的一些示例。

但是，这仅在相机本身不在阴影体积内时才有效！如果光线从阴影体积内部开始，并且在到达表面之前没有离开它，我们的算法将错误地得出它被照亮的结论。

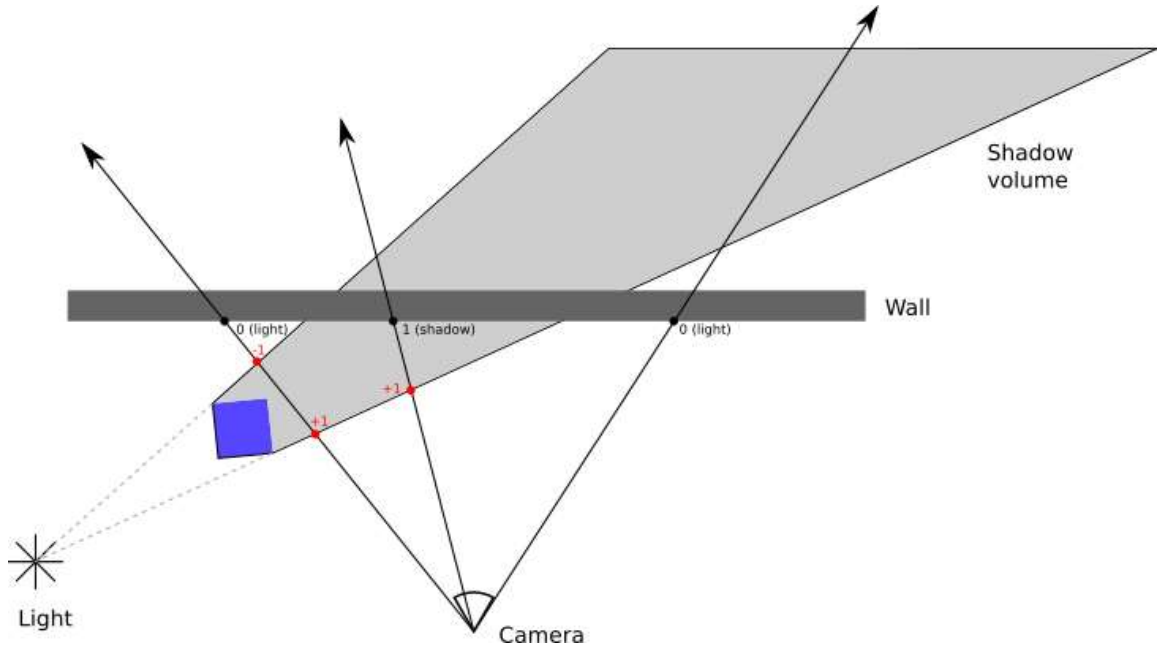

图 15-5：计算光线和阴影体积之间的交点可以告诉我们沿光线的点是被照亮还是在阴影中。

我们可以检查这种情况并相应地调整计数器，但计算一个点内部有多少阴影体积是一项昂贵的操作。幸运的是，有一种方法可以克服这种限制，它更简单、更便宜，尽管有点违反直觉。

光线是无限的，但阴影体积不是。这意味着光线总是在阴影体积之外开始和结束。反过来，这意味着光线进入阴影体积的次数总是与离开阴影体积的次数一样多;整个射线的计数器必须始终为零。

假设我们在光线照射到表面*后*跟踪光线和阴影体积之间的交点。如果计数器的值为零，则在光线照射到表面*之前*，该值也必须为零。如果计数器具有非零值，则它在曲面的另一侧必须具有相反的值。

这意味着在光线照射到表面之前计算光线和阴影体积之间的交点相当于计算光线之后的交点——但在这种情况下，我们不必担心相机的位置！图 15-6 显示了此技术如何始终产生正确的结果。

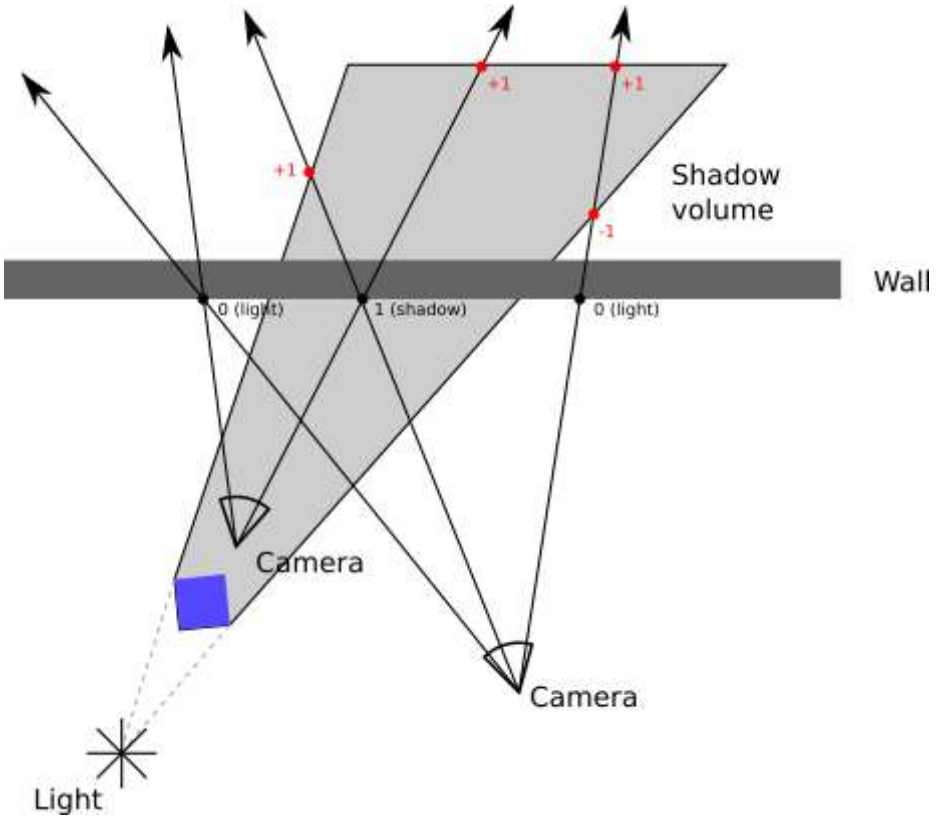

图 15-6：无论相机是在阴影体积内部还是外部，计数器对于接收光线的点的值为零，对于阴影中的点，计数器的值为非零值。

设置模具缓冲区

我们正在使用光栅器，而不是光线追踪器，因此我们需要找到一种方法来保留这些计数器，而无需实际计算光线和阴影体积之间的任何交集。我们可以通过使用模板缓冲区来做到这一点。

首先，我们将场景渲染为仅由环境光照明。环境光不会投射阴影，因此我们可以在不对光栅器进行任何更改的情况下执行此操作。这为我们提供了构成最终渲染所需的图像之一，但它也为我们提供了场景的深度信息，从摄像机看到的深度信息包含在深度缓冲区中。我们需要保留此深度缓冲区以备后续步骤使用。

接下来，对于每个灯，我们按照以下步骤操作：

1. 将阴影体积的背面“渲染”到模具缓冲区，每当像素*未通过*深度缓冲区测试时，就会增加其值。这将计算光线在到达最近的表面后离开阴影体积的次数。

2. 将阴影体积的正面“渲染”到模具缓冲区，每当像素*未通过*深度缓冲区测试时，其值就会递减。这将计算光线在击中最近的表面后进入阴影体积的次数。

请注意，在“渲染”步骤中，我们只对修改模具缓冲区感兴趣;无需将像素写入画布，因此无需计算照明或纹理。我们也不会写入深度缓冲区，因为阴影体积的侧面实际上并不是场景中的物理对象。相反，我们使用在环境光照通道期间计算的深度缓冲区。

执行此操作后，模具缓冲区对照亮的像素具有零，对于阴影中的像素具有其他值。因此，我们正常渲染场景，由与此通道对应的单个光源照亮，仅调用模板缓冲区值为零的像素。`PutPixel`

对每盏灯重复此过程，我们最终会得到一组与每盏灯照亮的场景相对应的图像，并正确考虑阴影。最后一步是通过逐像素地将所有图像添加到场景的最终渲染中。

使用模板缓冲区渲染阴影的想法可以追溯到 1990 年代初，但第一个实现有几个缺点。这里描述的深度失败变体在1999年和2000年被独立发现过几次，最著名的是John Carmack在制作*Doom 3*时发现的，这就是为什么这个变体也被称为*Carmack的逆转*。

阴影映射

在光栅器中渲染阴影的另一种众所周知的技术称为*阴影映射*。这将渲染边缘不太明确的阴影（想象一下阴天对象投射的阴影）。这些通常被称为*柔和阴影*。

重申一下，我们试图回答的问题是，给定表面上的点和光源，该点是否从该光接收照明？这相当于确定光线和点之间是否有物体。

使用光线追踪器，我们追踪了从点到光的光线。从某种意义上说，我们是在问这个点是否可以“看到”光，或者，等价地，光是否可以“看到”这个点。

这就引出了阴影映射的核心思想。我们从光线的角度渲染场景，保留深度缓冲区。与上面描述的环境贴图创建方式类似，我们渲染场景六次，最终得到六个深度缓冲区。这些深度缓冲区，我们称之为*阴影贴图*，让我们确定光线在任何给定方向上可以“看到”的最近表面的距离。

定向光源的情况稍微复杂一些，因为它们没有要渲染的位置。相反，我们需要从*某个方向*渲染场景。这需要使用*正交*投影而不是我们通常的透视投影。通过透视投影和点光源，每条光线都从一个点开始;使用正交投影和定向光源，每条光线彼此平行，共享相同的方向。

当我们想要确定一个点是否在阴影中时，我们计算从光到该点的距离和方向。我们使用方向在阴影图中查找相应的条目。如果此深度值小于从点到光源的距离，则表示存在比我们照亮的点更靠近光源的表面，因此该点位于该表面的阴影中;否则，光可以无障碍地“看到”该点，因此该点被光照亮。

请注意，阴影贴图的分辨率有限，通常低于画布。根据点和光源的距离和相对方向，这可能会导致阴影看起来是块状的。为了避免这种情况，我们还可以对周围深度条目的深度进行采样，并确定该点是否位于阴影的边缘（如周围条目中的深度不连续性所证明的那样）。如果是这种情况，我们可以使用类似于双线性滤波的技术，就像我们在第 14 章（纹理）中所做的那样，得出一个介于 0.0 和 1.0 之间的值，表示该点从光线中可见*的程度*，并将其乘以光的照明;这使得通过阴影映射创建的阴影具有其特征性的模糊外观。避免块状外观的其他方法包括以不同的方式对阴影贴图进行采样 - 例如，查看*百分比更接近过滤*。

总结

与第 5 章（扩展光线追踪器）一样，本章简要介绍了您可以自己探索的几个想法。这些扩展了前几章开发的光栅器，使其功能更接近光线追踪器的功能，同时保持其速度优势。总是需要权衡取舍，在这种情况下，它以不太准确的结果或增加的内存消耗的形式出现，具体取决于算法。