7.9 不规则Z-Buffer阴影

由于多种原因，各种阴影贴图方法很受欢迎.它们的成本是可预测的，并且可以很好地适应不断增加的场景大小，最坏的情况是与图元数量成线性关系.它们依靠光栅化来定期采样灯光的世界视野，从而将它们很好地映射到GPU上.但是，由于这种离散采样，出现了问题，因为眼睛看到的位置与光线看到的位置不是一一对应的.当光对表面的采样频率低于眼睛时，会出现各种混叠问题.即使采样率是可比较的，也存在偏差问题，因为在与眼睛所见位置稍有不同的位置采样了表面。

阴影体积提供了一种精确的分析解决方案，因为光线与表面的相互作用会产生一组三角形，这些三角形定义了给定位置是被照明还是处于阴影中。 当在GPU上实现时，算法的不可预测的成本是一个严重的缺点。近年来[1648]所探索的改进令人着迷，但尚未获得在商业应用中采用的“存在证据”。

从长远来看,另一种分析阴影测试方法可能具有潜力：光线跟踪。在11.2.2节中有详细描述，基本思想很简单，尤其是对于阴影。 光线从接收器位置射向光。 如果发现任何阻挡光线的物体，则接收器处于阴影中。 快速射线追踪器的大部分代码专用于生成和使用分层数据结构，以最大程度地减少每条射线所需的对象测试数量。 在动态场景的每个框架上构建和更新这些结构是数十年的历史，也是一个持续的研究领域.

另一种方法是使用GPU的光栅化硬件来查看场景，但不只是z深度，还存储了有关每个光网格单元中遮挡物边缘的附加信息[1003，1607]。例如，想象一下在每个阴影贴图纹理元素上存储与网格单元重叠的三角形列表。这样的列表可以通过保守光栅化来生成，如果三角形的任何部分与像素重叠，而不仅仅是像素的中心，则三角形会生成一个片段（第23.1.2节）。这种方案的一个问题是通常需要限制每个纹理像素的数据量，这反过来可能导致确定每个接收器位置的状态不准确。考虑到现代GPU的链表原理[1943]，每个像素存储更多数据当然是可能的。但是，除了物理内存限制外，每个纹理元素在列表中存储可变数量的数据的问题在于，GPU处理可能变得非常低效，因为单个扭曲可能具有一些片段线程，需要检索和处理许多项目，而其余的线程则处于空闲状态，无事可做。构建着色器以避免动态“if”语句和循环导致的线程发散对性能至关重要.

在阴影图中存储三角形或其他数据并针对它们测试接收器位置的另一种方法是翻转问题，存储接收器位置，然后针对每个三角形测试三角形。 这种保存接收器位置的概念最早由Johnson等人[839]和Aila和Laine [14]提出，被称为不规则z缓冲区（IZB）。 名称有点误导，因为缓冲区本身具有法线的阴影贴图，规则形状。相反，缓冲区的内容是不规则的，因为每个阴影贴图纹理像素将在其中存储一个或多个接收器位置，或者可能根本没有。参见图7.30。

使用Sintorn等人[1645]和Wyman等人[1930，1932]提出的方法，多遍算法创建了IZB并测试了其内容在光线下的可见性。 首先，从眼睛渲染场景，以找到从眼睛看到的表面的z深度。 这些点将转换为灯光的场景视图，并由此设置灯光视锥的紧密边界。 然后将这些点放置在灯光的IZB中，每个点都放在其对应纹理像素的列表中。 请注意，有些列表可能是空的，是灯光可以看到的空间，但是眼睛看不到任何表面。 遮罩保守地光栅化到灯光的IZB，以确定是否隐藏了任何点，等等。 保守栅格化可确保即使三角形不覆盖光纹素的中心，也将针对可能重叠的点进行测试。

可见性测试在像素着色器中进行。 测试本身可以可视化为射线追踪的一种形式。 从图像点的位置到光源会产生光线。如果一个点在三角形内并且比三角形平面更远，则该点被隐藏。光栅化所有遮挡物后，将可见性结果用于对表面进行着色。此测试也称为视锥跟踪，因为可以将三角形视为定义了视锥的视图，该视锥用于检查点是否包含在视锥中。

仔细的编码对于使此方法与GPU兼容至关重要。 Wyman等人[1930，1932]指出，它们的最终版本比初始原型快两个数量级。 这种性能提升的一部分是直接的算法改进，例如剔除表面法线背对光（因此始终不照明）的图像点，并避免为空纹理像素生成碎片。 其他性能提升来自改善GPU的数据结构，以及通过在每个texel中使用较短的相似长度的点列表来最小化线程差异。 图7.30显示了一个带有长列表的低分辨率阴影图，用于说明目的。 理想情况是每个列表一个图像点。 分辨率越高，列表越短，但封堵器生成的片段数量也越多。

从图7.30的左下方图像可以看出，由于透视效应，接地面上可见点的密度在左侧比右侧明显更高。使用级联阴影贴图可通过将更多的光照贴图分辨率聚焦在更靠近眼睛的位置来帮助减小这些区域中的列表大小。

这种方法避免了其他方法的采样和偏差问题，并提供了完全清晰的阴影。出于审美和感官上的原因，通常需要柔和的阴影，但附近的遮挡物（例如Peter Panning）可能会有偏斜问题。 Story and Wyman [1711，1712]探索混合阴影技术。核心思想是使用遮挡物距离混合IZB和PCSS阴影，当遮挡物靠近时使用硬阴影结果，而在更远的地方使用软阴影结果。参见图7.31。阴影质量通常对于附近的物体最为重要，因此，仅对选定的子集使用此技术，就可以降低IZB成本。 此解决方案已成功用于视频游戏中。本章从这样的图像开始，如第224页的图7.2所示。

7.10 其它应用

将阴影贴图视为定义一定的空间，将光与暗分离，也可以帮助确定对象的哪些部分需要阴影。 Gollent [555]介绍了CD Projekt的地形阴影系统如何为每个区域计算仍被遮挡的最大高度，然后可以将该最大高度用于阴影，不仅可以遮挡地形，还可以遮挡场景中的树木和其他元素。为了找到每个高度，为太阳绘制可见区域的阴影图。然后检查每个地形的高度场位置是否有阳光可见性。如果处于阴影中，则通过将世界高度增加一个固定的步长，直到看到太阳，然后执行二进制搜索，来估计太阳首次可见的高度。换句话说，我们沿着一条垂直线行进，并进行迭代以缩小与阴影图的表面相交的位置，从而将明暗分开。内插相邻的高度以在任何位置找到此遮挡高度。在图7.32中可以看到用于地形高度场的软阴影的此技术的示例。在第14章中，我们将看到更多使用光线行进穿过明暗区域的方法。

值得一提的最后一种方法是渲染屏幕空间阴影。由于分辨率有限，阴影贴图通常无法在小要素上产生准确的遮挡。当渲染人脸时，这尤其成问题，因为我们特别容易注意到它们上的任何视觉伪像。例如，渲染发光的鼻孔（当不需要时）看起来很刺耳。虽然使用分辨率更高的阴影图或仅针对目标区域的单独阴影图可以提供帮助，但另一种可能性是利用已经存在的数据。在大多数现代渲染引擎中，从摄像机角度来看的深度缓冲区（来自较早的预渲染）在渲染期间可用。存储在其中的数据可以视为高度场。通过迭代采样该深度缓冲区，我们可以执行光线行进过程（第6.8.1节），并检查朝向光的方向是否被遮挡。尽管代价高昂，因为它涉及到重复采样深度缓冲区，但这样做可以为剪切场景中的特写镜头提供高质量的结果，在剪切场景中，通常需要花费额外的毫秒时间。该方法是由Sousa等人[1678]提出的，如今已广泛用于许多游戏引擎[384，1802].

总结整个章节，到目前为止，某种形式的阴影映射是用于将阴影投射到任意表面形状上的最常用算法。 当在大面积（例如室外场景）中投射阴影时，级联阴影贴图可提高采样质量。 通过SDSM为近平面找到合适的最大距离可以进一步提高精度。 百分比更接近过滤（PCF）使阴影更柔和，百分比更接近软阴影（PCSS）及其变体使接触硬化，不规则的z缓冲区可提供精确的硬阴影。 过滤后的阴影图可提供快速的软阴影计算，并且在遮挡物远离接收器（如地形）时，效果特别好。 最后，尽管成本高昂，但屏幕空间技术仍可用于提高精度。

在本章中，我们重点介绍了应用程序中当前使用的关键概念和技术。每个都有自己的优势，选择取决于世界的大小，构图（静态内容还是动画），材质类型（不透明，透明，头发或烟雾）以及灯光的数量和类型（静态或动态；局部或远距离；点） ，斑点或面积），以及诸如基础纹理可以隐藏任何伪像的程度之类的因素。 GPU的功能不断发展和完善，因此我们希望在未来的几年中会继续出现与硬件良好匹配的新算法。例如，第19.10.1节中描述的稀疏纹理技术已应用于阴影贴图存储，以提高分辨率[241、625、1253]。 Sintorn，Kampe等人[850，1647]以一种创新的方法探索了将光的二维阴影贴图转换为三维三维像素集的想法（小盒子；请参见第13.10节）。使用体素的优点是可以将其分类为亮或暗，因此需要的存储量最少。高度压缩的稀疏体素八叉树表示法可存储大量灯光和静态遮挡物的阴影。 Scandolo等人[1546]将其压缩技术与使用双阴影图的基于间隔的方案相结合,从而提供了更高的压缩率.Kasyan [865]使用体素圆锥追踪（第13.10节）从区域光生成柔和阴影.有关示例，请参见图7.33.更多的圆锥形阴影显示在第585页的图13.33中。