到现在为止，我们已经假定我们渲染的模型可以完全按照我们需要的格式使用，并且具有适当的细节量。 实际上，我们很少如此幸运。 建模者和数据捕获设备有其自身的特殊怪癖和局限性，从而导致数据集内以及渲染图内的歧义和错误。 通常需要在存储大小，渲染效率和结果质量之间进行权衡。 在本章中，我们将讨论多边形数据集中遇到的各种问题，以及这些问题的一些修补程序和解决方法。 然后，我们介绍了有效渲染和存储多边形模型的技术。

交互式计算机图形中多边形表示的总体目标是视觉准确性和速度。 “准确性”是一个取决于上下文的术语。 例如，工程师想要以交互的速率检查和修改机器零件，并要求物体上的每个斜面和倒角在任何时候都可见。 将此与游戏进行比较，在该游戏中，如果帧频足够高，则允许在给定帧中出现较小的错误或不准确之处，因为它们可能不会集中在注意力上，也可能在下一帧中消失。 在交互式图形工作中，重要的是要知道要解决的问题的边界是什么，因为这些边界确定可以应用哪种技术。

本章涵盖的领域包括细分，合并，优化，简化和压缩。 多边形可以多种不同的形式到达，通常必须分为更易处理的图元，例如三角形或四边形。 此过程称为“三角剖分”或更笼统地称为“曲面细分”。合并是我们对过程的术语，其中包括将单独的多边形合并到网格结构中，以及获取新数据（例如法线）以进行表面着色。 优化意味着对网格中的多边形数据进行排序，以便更快地进行渲染。 简化就是采用网格并删除其中不重要的特征。 压缩与最小化描述网格的各种元素所需的存储空间有关。

三角剖分可确保正确显示给定的网格描述。 通过允许共享计算并减小内存大小，合并可以进一步改善数据显示并通常提高速度。 优化技术可以进一步提高速度。 通过删除不需要的三角形，简化可以提供更高的速度。 压缩可用于进一步减少总体内存占用量，进而可以通过减少内存和总线带宽来提高速度。

16.1 三维数据源

有多种方法可以创建或生成多边形模型:

1. 直接输入几何描述.
2. 编写创建此类数据的程序.这称为过程建模.
3. 将以其他形式找到的数据转换为表面或体积，例如获取蛋白质数据并将其转换为一组球体和圆柱体。
4. 使用建模程序来构建或雕刻对象。
5. 用同一物体的一张或多张照片重建表面，称为摄影测量法。
6. 使用三维扫描仪，数字化仪或其他感应设备在各个点对真实模型进行采样.
7. 生成在某些空间体积中表示相同值的等值面，例如来自CAT或MRI医学扫描的数据，或在大气中测量的压力或温度样本.
8. 结合上述这些技术。

在建模世界中，有两种主要类型的建模器：基于实体的和基于曲面的。 基于实体的建模者通常出现在计算机辅助设计（CAD）领域，并且通常强调与实际加工过程相对应的建模工具，例如切削，钻孔和刨削。 在内部，它们将具有一个计算引擎，该引擎将严格操纵对象的基础拓扑边界。 为了进行显示和分析，此类建模者具有多方面的功能。 Faceter是一种软件，可以将内部模型表示形式转换为可以显示的三角形。 例如，球体可以在数据库中由中心点和半径表示，而刻面可以将其变成任何数量的三角形或四边形以表示它。 有时，最快的渲染速度是最简单的：使用刻面机时降低所需的视觉精度可以通过生成更少的三角形来提高速度并节省存储空间。

CAD工作中的一个重要考虑因素是所使用的刻面是否设计用于图形渲染。 例如，有限元方法（FEM）的分面器旨在将表面分割成几乎相等面积的三角形。由于这些镶嵌包含许多图形上无用的数据，因此非常适合进行简化。 同样，某些刻面会生成一组三角形，这些三角形非常适合使用3D打印创建现实世界的对象，但缺少顶点法线并且通常不适合用于快速图形显示。

Blender或Maya之类的建模器不是基于内置的坚固性概念。 相反，对象由其表面定义。 与实体建模者一样，这些基于曲面的系统可以使用内部表示和构面来显示诸如样条线或细分曲面之类的对象（第17章）。 它们还可以直接操纵表面，例如添加或删除三角形或顶点。 然后，用户可以手动降低模型的三角形数量。

还有其他类型的建模器，例如隐式曲面（包括“ blobby” metaball）创建系统[**67,558**]，它们可以处理诸如混合，权重和场之类的概念。 这些建模者可以通过生成由某些函数f（x，y，z）= 0的解定义的表面来创建有机形式。然后，使用多边形技术（例如，行进立方体）来创建用于显示的三角形集（第17.3节）.

点云是简化技术的强大候选者。数据通常以固定的间隔进行采样，因此许多采样对所形成表面的视觉感知影响可忽略不计。研究人员花费了数十年的时间研究技术，以过滤出缺陷数据并从点云中重建网格[**137**]。有关此区域的更多信息，请参见第13.9节。

可以对从扫描数据生成的网格执行任意数量的清理或高阶操作。例如，分割技术分析多边形模型并尝试识别单独的部分[**1612**]。这样做可以帮助创建动画，应用纹理贴图，匹配形状以及其他操作。

还有许多其他方式可以生成多边形数据以表示表面。关键是要了解如何创建数据以及出于什么目的。通常，不是为有效的图形显示而专门生成数据。同样，有许多不同的三维数据文件格式，在任何两种格式之间进行转换通常不是无损操作。了解传入数据可能遇到的各种限制和问题是本章的主题。

16.2 细分和三角化

细分是将一个表面分割为一组多边形的过程。 在这里，我们专注于细分多边形表面。 曲面细分在17.6节中讨论。可以出于多种原因进行多边形镶嵌。 最常见的是所有图形API和硬件都针对三角形进行了优化。三角形几乎就像原子一样，因为任何表面都可以由它们制成并渲染。 将复杂的多边形转换为三角形称为三角剖分。

细分多边形时，有几个可能的目标。例如，要使用的算法可以仅处理凸多边形。这种细分称为凸分区。可能需要细分表面（网格化）以使用全局照明技术在每个顶点处存储阴影或相互反射的效果[**400**]。图16.1显示了这些不同类型的细分的示例。细分的非图形原因包括一些要求，例如不使三角形大于某个给定区域，或者使三角形在其顶点处的角度都大于某个最小角度。 Delaunay三角剖分要求每个三角形的顶点形成的每个圆不包含任何剩余的顶点，这将最小角度最大化。尽管此类限制通常是非图形应用程序（例如有限元分析）的一部分，但它们也可以用来改善表面外观。长而细的三角形通常值得避免，因为当在远处的顶点进行插值时，它们会引起伪影。它们也可能无法有效光栅化[**530**]。

大多数镶嵌算法在两个方面起作用。 他们假定多边形中的所有点都在同一平面上。 但是，某些模型创建系统可能会生成变形严重且非平面的多边形面。 这个问题的一个常见情况是翘曲的四边形，几乎在边缘观察。 这可能会形成所谓的沙漏或领结四边形。 参见图16.2。 尽管可以通过创建对角线边缘简单地对这个特定的多边形进行三角剖分，但是更复杂的变形多边形却无法如此轻松地进行管理。

如果可能出现变形的多边形，则一种快速的纠正措施是将顶点投影到垂直于多边形近似法线的平面上。 通常通过计算三个正交的xy，xz和yz平面上的投影面积来找到此平面的法线。 也就是说，通过删除x坐标找到的yz平面上多边形的面积是x分量，xz y和xy z的值。 这种计算平均法线的方法称为Newell公式[**1505,1738**]。

投射到该平面上的多边形可能仍存在自相交问题，其中两个或多个边交叉。因此，需要更加复杂和计算昂贵的方法。Zou等人[**1978**]讨论了基于最小化曲面细分的表面积或二面角的先前工作，并提出了将一组集合中的几个非平面多边形一起优化的算法。

Schneider和Eberly [**1574**]，Held [**714**]，O'Rourke [**1339**]和de Berg等[**135**]分别概述了各种三角剖分方法。 最基本的三角剖分算法是检查多边形上任意两个给定点之间的每个线段，并查看其是否与多边形的任意边相交或重叠。如果是这样，则无法使用线段来分割多边形，因此我们将检查下一对可能的点。否则，使用此线段将多边形分为两部分，并使用相同的方法对这些新的多边形进行三角剖分。该方法非常慢，为O（n3）。

一种更有效的方法是削耳，当以两个过程完成时，即为O（n2）。 首先，对多边形进行遍历以查找耳朵，即查看所有顶点索引为i（i + 1），（i + 2）（模n）的三角形，并检查线段i， （i + 2）不与任何多边形边缘相交。 如果不是，则三角形（i + 1）形成耳朵。 参见图16.3。 依次从多边形中除去每个可用的耳朵，并重新检查顶点i和（i + 2）处的三角形，看看它们现在是否是耳朵。 最终，所有耳朵都被拔掉，然后对多边形进行了三角剖分。 其他更复杂的三角剖分方法是O（n log n），对于典型情况，某些方法实际上是O（n）。 Schneider和Eberly [**1574**]给出了用于夹耳和其他更快的三角剖分方法的伪代码。

除了存储和进一步的计算成本外，将多边形划分为凸形区域也可以比三角剖分更高效。 Schorn和Fisher[]给出了鲁棒性凸度测试的代码。凸多边形可以很容易地由扇形或三角形带表示，如16.4节所述。 某些凹面多边形可以视为扇形（此类多边形称为星形），但是检测这些凹面需要更多的工作[**1339,1444**]。Schneider和Eberly [**1574**]提出了两种凸分区方法，一种是快速脏方法，另一种是最佳方法。

多边形并非总是由单个轮廓组成。 图16.4显示了由三个轮廓（也称为回路或轮廓）组成的多边形。 通过仔细地在循环之间生成连接边（也称为键孔或桥边），可以始终将此类描述转换为单轮廓多边形。 Eberly [**403**]讨论了如何找到定义此类边缘的相互可见的顶点。 也可以颠倒此转换过程以检索单独的循环。

编写强大而通用的三角剖分器是一项艰巨的任务。各种细微的错误，病理情况和精度问题使创建万无一失的代码变得异常棘手。解决三角剖分问题的一种方法是使用图形加速器本身直接渲染复杂的多边形。多边形作为三角形扇形渲染到模板缓冲区。这样，应该填充的区域被绘制了奇数次，而凹坑和孔被绘制了偶数次。通过对模板缓冲区使用反转模式，在第一遍的结尾仅标记填充区域。参见图16.5。在第二遍中，使用模板缓冲区再次绘制三角形风扇，以仅绘制填充区域。通过绘制每个循环形成的三角形，该方法甚至可以用于渲染具有多个轮廓的多边形。主要缺点是必须使用两次通过来渲染每个多边形，并且模板缓冲区会清除每个帧，并且深度缓冲区无法直接使用。该技术对于某些用户交互的显示可能很有用，例如显示动态绘制的复杂选择区域的内部。

16.2.1 着色问题

有时，数据将以四边形网格的形式到达，必须转换为三角形才能显示。 在很长一段时间内，四边形将是凹形的，在这种情况下，只有一种三角剖分的方法。 否则，我们可以选择两个对角线中的任意一个来拆分它。 花一点时间挑选更好的对角线有时会带来明显更好的视觉效果。

有几种不同的方法可以决定如何分割四边形。 关键思想是最小化新边的顶点处的差异。 对于在顶点处没有其他数据的平坦四边形，通常最好选择最短的对角线。 对于每个顶点具有颜色的简单烘焙全局照明解决方案，请选择对角线，其颜色之间的差异较小[**17**]。 参见图16.6。 由一些启发式方法确定的连接两个最小差异拐角的想法通常可用于最大程度地减少伪影。

有时三角形不能正确反映设计者的意图。 如果将纹理应用于扭曲的四边形，则对角线分割均不会保留该意图。 也就是说，在未三角剖分的四边形上进行简单的水平插值，即从左边缘到右边缘插值，也会失败。 图16.7显示了该问题。 出现该问题的原因是，显示时要扭曲施加到表面的图像。 三角形只有三个纹理坐标，因此可以建立仿射变换，但不能建立扭曲。 最多可以剪切而不是扭曲三角形上的基本（u，v）纹理。 Woo等人[1901]进一步讨论了这个问题。 几种解决方案是可能的:

预先扭曲纹理，并使用新的纹理坐标重新应用此新图像。

将曲面细分为更细的网格。 这只会减轻问题。

使用投射纹理使飞行中的纹理发生扭曲[**691,1470**].这具有不希望的表面纹理间距不均匀的效果。

使用双线性映射方案[**691**].这是可以通过在每个顶点上添加其他数据来实现的.

虽然纹理失真听起来像是病理情况，但只要所应用的纹理数据与下面的四边形的比例不匹配，即几乎在任何曲面上，它都会在某种程度上发生。 一个极端情况发生在一个共同的图元上：圆锥体。 当圆锥体带有纹理和刻面时，圆锥体顶端的三角形顶点具有不同的法线。 这些顶点法线不被相邻三角形共享，因此发生着色不连续性[647].