第11章 多边形技术

金小刚 **Email:** *jin@cad.zju.edu.cn*

http://www.cad.zju.edu.cn/home/jin/course.htm

浙江大学CAD&CG国家重点实验室

紫金港校区 信息与控制大楼506

内容简介

- **Tessellation(细化)**

  将多边形分割为更易于处理的三角形、四边形等基本单位的过程。

- **Consolidation (合并)**

  合并包括多边形数据的连接和融合，以及新数据的衍生，例如物体表面法线。

- **Stripification(条带化)**

  条带化是将三角形网格转换成多组多边形条带的过程。

- **Simplification(简化)**

  简化是将数据中不必要或不重要的特征删除的过程。

三维数据的来源

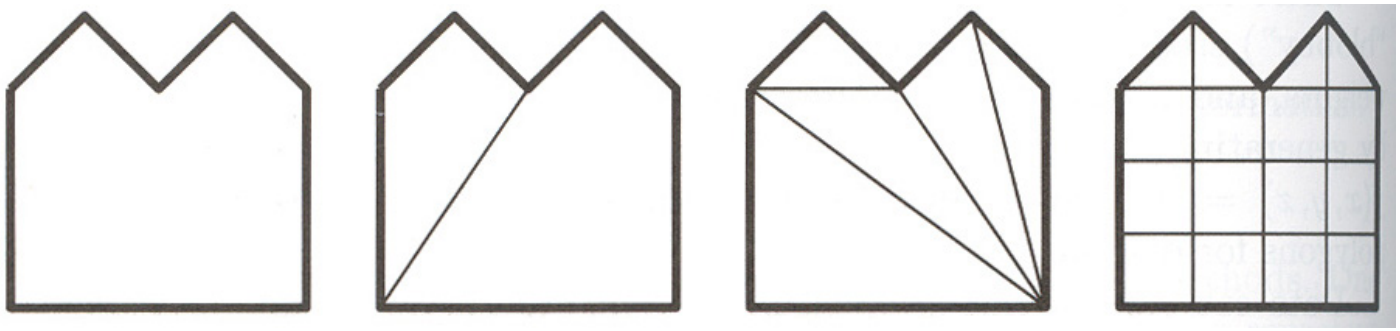
- 以几何描述方式直接输入模型；
- 将其它形式的数据转化为面或体数据。如将蛋白质数据转化为球体或圆柱体；
- 通过编写程序来生成模型**(Procedural Modeling)**；
- 使用建模程序；
- 使用三维数字化仪对真实模型进行采样；
- 基于图像的方法；
- 使用三维扫描仪采集深度信息；

细化与三角剖分

- 进行多边形细化的原因
  - 许多图形**API**和硬件都针对三角形进行了优化；
  - 绘制程序只能对凸多边形进行绘制(这种细化也称为凸分解)；
  - 为使用辐射度技术进行绘制，需要对物体表面进行网格化；
  - 其它非图形学的原因…
    如有限元分析、避免狭长三角形等

细化与三角剖分

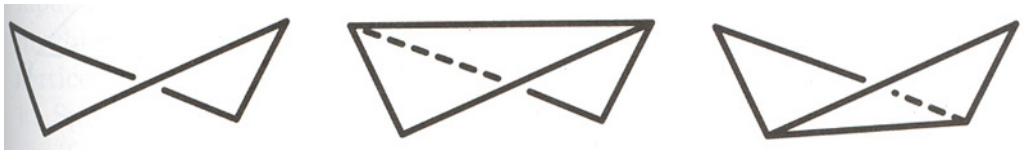
- 不同类型的细化示例



(a)没有细化 (b)细化为2个凸多边形 (c)三角剖分
(d)均匀网格化

细化与三角剖分

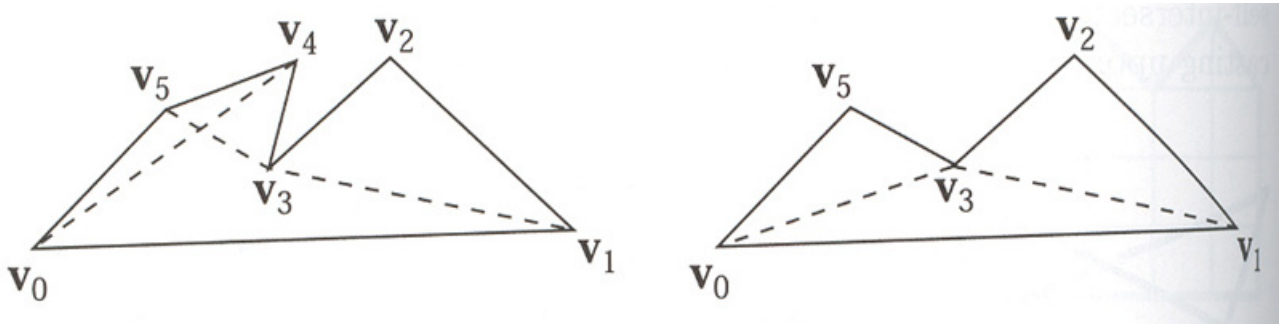
- 表面细化的一个重要步骤是如何最好地将三维多边形投影成二维图形
  - 面积测试：丢弃xyz三个坐标中的一个，使得投影多边形在该坐标平面上具有最大的投影面积，取该坐标平面为投影平面。可以直接计算投影面积来确定，也可以通过丢弃多边形法向中绝对值最大的坐标来确定。
  - 法向测试：根据法向计算公式和多边形是否平坦来决定。为避免自交等问题，可把多边形投影到平均平面上。



接近视线的弯曲四边形，形成病态的蝴蝶结状或沙漏状图形

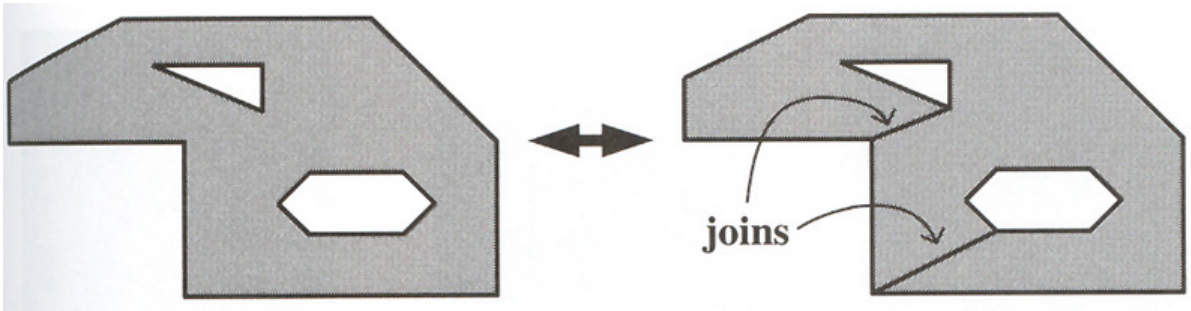
细化与三角剖分

- 三角剖分的方法
  - 基本方法：检查多边形中任意两顶点间的线段，如果不相交、不重叠，那么将多边形分割成两部分，采用相同方法对新生成的多边形继续分割。算法复杂度$O(n^3)$。
  - 耳状剪切(Ear Clipping)：下左图为一在顶点v2、v4、v5处有耳状物的多边形。右图中删除了v4处的耳状物。对邻接顶点v3、v5重复检查，考虑是否形成耳状物。算法复杂度$O(n^2)$。



细化与三角剖分

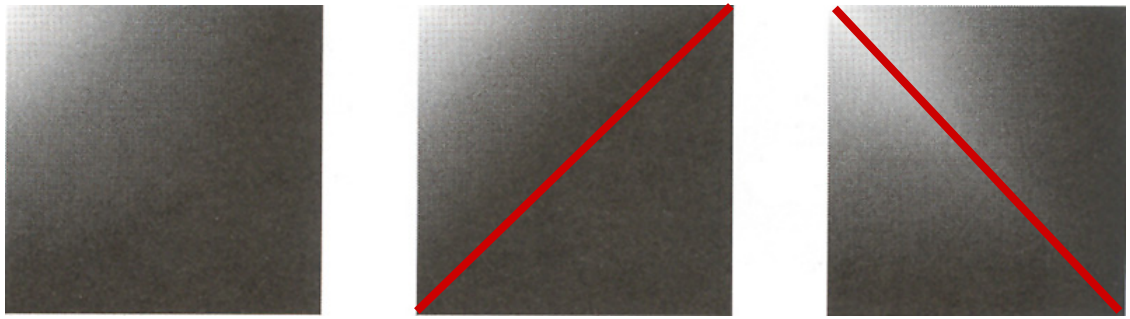
- 存储与绘制
  - 将多边形分解为凸多边形比三角剖分更有效，因为凸多边形可用扇形或者三角形条带表示；
  - 凸多边形分解方法: **(1). Quick and Dirty** 方法；**(2).** 最优化方法。
- 多轮廓线多边形转换为单轮廓线多边形



将具有三个轮廓线的多边形转化为单轮廓多边形

着色问题

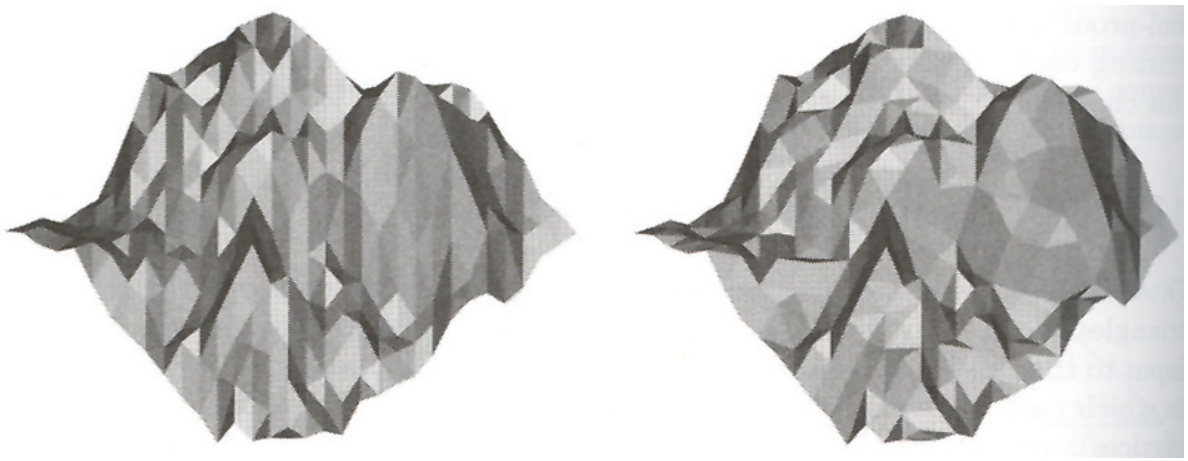
- 四边形的着色问题
  - 为便于显示，通常将四边形分割为三角形，而不同的分割方法会得到不同的显示效果；
  - 分割四边形应采取差异最小化原理；
  - 对于辐射度方法或预光照四边形可选择色彩差异最小的对角线；



(a)四边形绘制 **(b)** 右上角和左下角连接的三角形 **(c)**左上角和右下角连接的三角形

着色问题

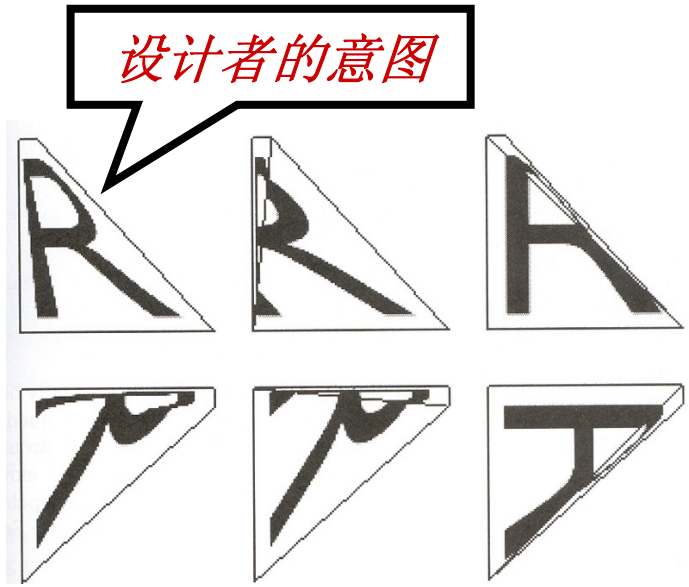
- 四边形的着色问题(续)
  - 对于地形高度场，有多种选择： 连接法向夹角相差最大的两个三角形的对角线；连接面积最接近的两个三角形的对角线；连接高度最接近的顶点；连接高度差异最大的顶点。



图中为高度场网格: (a) 四边形沿同一对角线分割
(b)沿高度接近的对角线分割

着色问题

- 但也有三角形不能反映设计者意图的情形(三角剖分后的纹理映射)。解决方法：
  - 映射前对纹理进行扭曲；
  - 将表面细化为一个更精细的网格，以减轻畸变；
  - 使用投影纹理贴图；
  - 使用双线性插值映射的方法。但通常难以用硬件实现。



左上图是一个贴有“R” 字母纹理的扭曲四边形，右上图为不同的三角剖分，第二行为第一行的旋转

问题根源：一个三角形只有三个纹理坐标，只能进行仿射变换，但不能对纹理进行扭曲变形

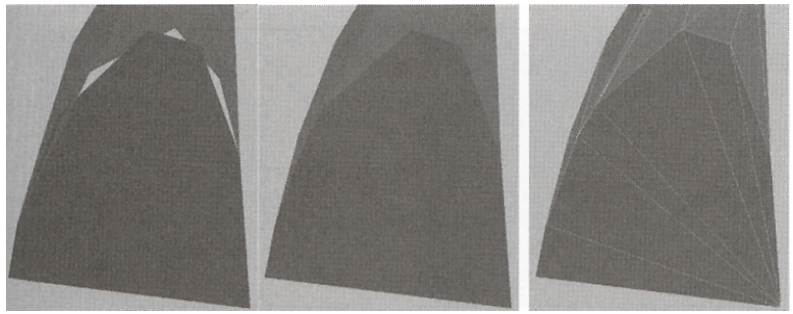
边裂缝与T型顶点

在对物体进行细化时，经常会遇到边裂缝问题

- 边裂缝(Edge Cracking)问题
  - 对曲面网格化的过程可以分解为沿样条曲线进行逐步细分。若两个曲面的共享边界，当一条样条曲线生成的点不能与相邻曲线的点匹配时，会产生边裂缝现象。
- 边缝合(Edge Stitching)
  - 确保沿着共享边的所有顶点均由两个样条曲面共享，就能保证没有裂缝出现。

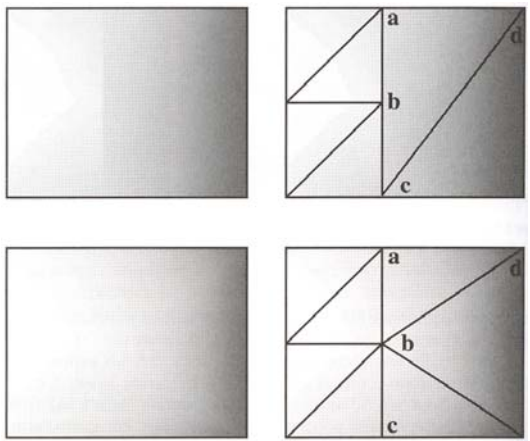
边缘裂缝与T型顶点

- 边缝合(Edge Stitching) (continue)



左图为表面相接处的裂缝，中间为通过边缘点匹配进行修复，右图为校正后的结果

- T型顶点问题
  - 连接曲面时，如果两曲面边相交但没有共享相应顶点，会出现裂缝。



第一行的表面有不连续的阴影效果，其中**b**是**T**型顶点，因为它只属于左侧多边形。第二行将**b**加入右侧，生成多个三角形。

合 并

- 合并(Consolidation)
  - 细化过程结束后，为了提高绘制质量和数据质量，需要搜索和调整细化处理得到的多边形的连接关系，这个过程称为合并。
  - 合并的流程：

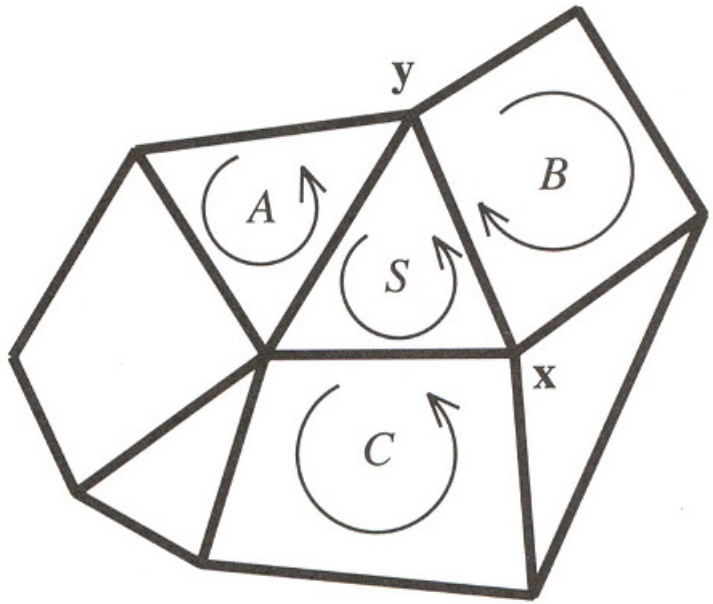
    1. 形成所有多边形的边一面结构，并排序；
    2. 找到相连的多边形组；确定体特性，寻找边界边；
    3. 对每组边一面进行必要的翻转来获得一致性；
    4. 确定网格内部方向，根据需要翻转相应表面；
    5. 找出折叠边界；
    6. 生成多边形网格。

合并

- 具体步骤
  - 将多边形的所有边存为一个列表，其中每条边指向相应的表面，边的两个顶点按照**xyz**坐标从小到大的顺序排列。如果边的长度为**0**，删除边。
  - 在边列表中，找出相同的边，也就是匹配边，这样就可以找到与边相邻的三角形，进而在相邻多边形中找到连接关系，确定边界线。所有相邻的多边形组可以形成一个连续组。

合并

- 具体步骤(continue)
  - 方向一致性。对于每一个连续的多边形组，任意选取一个起始多边形，检查相邻的多边形，判断方向是否一致来决定是否进行翻转。



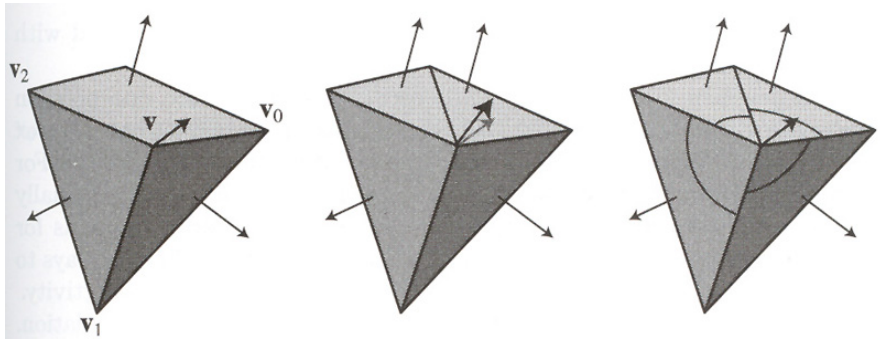
选择起始多边形S并对与它相邻的多边形进行检查。由于多边形S和B所共享边的两个定点遍历顺序一致，都是从顶点x到顶点y，所以需要将多边形B的轮廓边进行翻转。

合并

- 具体步骤(continue)
  - 检查组中所有边是否为两个多边形共享来确定多边形组是否流形。检查多边形组的体积，如果体积为负，翻转所有的环和法线。
  - 一旦方向一致，我们就可以计算顶点的法向。通过为多边形指定平滑组，或者提供一个折叠角(crease angle)，模型的格式就可以提供相应的平滑信息，使用平滑组值显式地定义组中哪些多边形共同组成了一个弯曲的平面。

合并

- 具体步骤(continue)
  - 计算顶点法线。



左图中，对四边形表面法线和两个三角形进行平均得到顶点法线。中图中，对四边形三角剖分，由于每个多边形法线具有相等的权值，导致平均法线的移动。右图中，通过计算两条边之间的二面角来对每条法线的影响进行加权，所以三角剖分没有移动法线

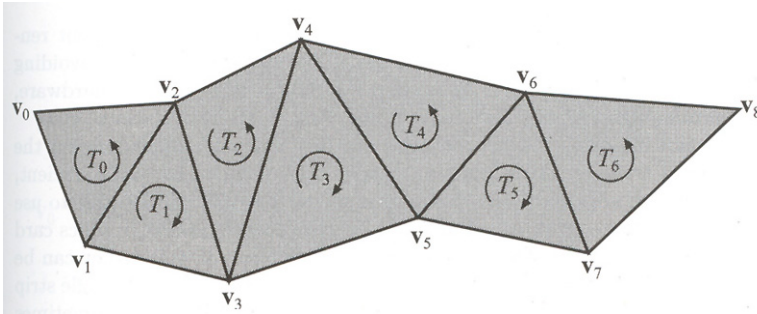
  - 如果边所在的多边形属于不同的光滑组，或者这些边有奇数个相邻多边形，那么这些边就是曲面的边界。
  - 生成网格。

三角形条带、扇形、网格

- 用于提高图形系统性能的一种最常见方法为使发送到图形流水线的每个三角形顶点数少于3个。这样可以减少需要进行变换的点和法向数目，也可以减少光照计算。
- 但是如果应用程序的瓶颈为填充率(每秒可填充的象素数量)时，很难使用这种方法提高系统性能。
- 可以使用三角形条带、三角扇形和网格等方法减少数据量。

三角形条带

- 三角形条带(Triangle Strips)
  - n个顶点的顺序三角形定义为顶点序列：$\{v_0,v_1,\ldots v_{n-1}\}$,其中三角形i为 $\Delta v_i v_{i+1} v_{i+2}$。



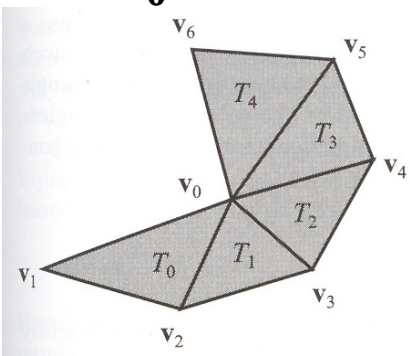
可用一个三角形条带表示一个三角形序列。条带中的第一个三角形决定了所有三角形的方向。在内部可以按照逆时针顺序(如0－1－2、1－3－2、2－3－4、3－5－4，…)遍历顶点，以保证一致的针方向。

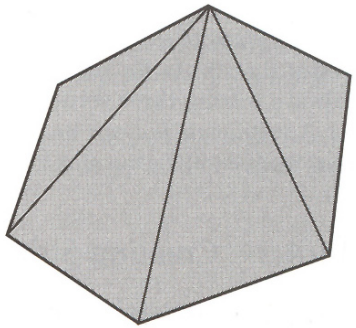
  - 由n个顶点组成的顺序三角形条带具有n－2个三角形，因此三角形条带的长度是n－2。

  - 对于长度为m的顺序三角形条带，发送到图形流水线的平均顶点个数v_a为：$v_a = \frac{3+(m-1)}{m} = 1+\frac{2}{m}$

三角扇形

- 三角扇形（Triangle Fans）
  - n个顶点的三角扇形定义为顶点序列：$\{v_0,v_1,\ldots v_{n-1}\}$,其中顶点$v_0$为中心顶点，三角形i为$\Delta v_0 v_{i+1} v_{i+2}$。



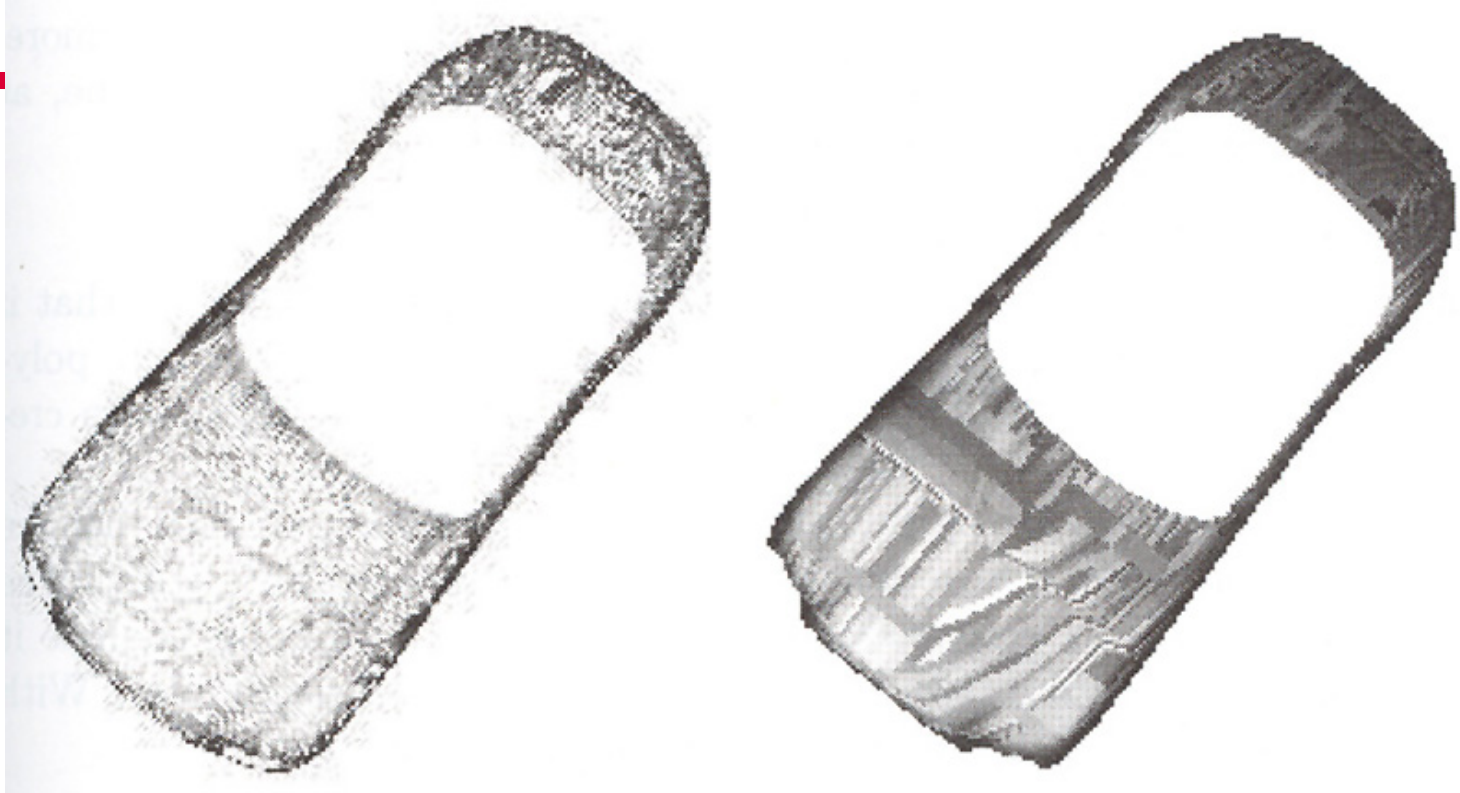


左图所示为三角扇形。对于三角形T_0，需要发送中心顶点v_0，v_1和v_2，对于随后的T_i（i>0），仅需发送v_{i+2}。右图所示，可将任意凸多边形转换为三角扇形。

  - 对于长度为m的三角扇形，发送到图形流水线的平均顶点个数与三角形条带的相同。
  - 任何三角扇形可转换为三角形条带，反之则不一定成立。

三角条带的生成

- 给定一个任意的三角形网格，最好能快速、高效地将其分解为三角形条带。
- 给定任意的三角形网格，如何获得最优的三角形条带问题已被证明为**NP**完全问题，必须使用启发式方法获得最接近的条带数目。
- 我们将给出一种贪婪算法构造顺序三角形条带，并简要介绍对偶图**(Dual Graph)**算法。



左图为线框表示的小车部分表面，右图为对每个三角形条带进行随机着色处理的结果

三角条带的生成

- 背景知识
  - 每个三角形条带的生成算法都是首先为多边形组创建一个邻接数据结构。也就是说，对于多边形的每条边，存贮与之相邻的多边形索引。
  - 与一个多边形相邻的多边形数目称为多边形的度，度是介于0和这个多边形顶点数目间的一个整数。
  - 使用连通平面图的Euler定理可以确定必须发送到图形流水线的顶点平均数目：

$$v - e + f = 2$$

其中v表示顶点数，e表示边数，f表示面数。由于在连通中，每条边有2个面，每个面至少3条边，所以有2e≥3f。如果所有的面都是三角形，那么由2 e＝3f得到f＝2v－4。

三角条带的生成

- 改进的SGI条带生成算法
  - SGI算法仅适用于全部三角剖分的模型。
  - SGI算法是一种贪婪算法。这种算法开始选取的三角形总是具有最低的度。
  - 改进的SGI条带生成算法步骤：

① 选取起始三角形：最初的SGI算法选取具有最低度（必须>0）的三角形。如果存在多个三角形具有相同的最低度，考察相邻三角形的度，如果再次遇到这种情况，从中任意选取。

改进的SGI条带生成算法（续）

② 创建3个不同的三角形条带，每个条带对应起始三角形的一条边。

③ 反方向延伸这些三角形条带：在条带的开始和结束方向尽可能延伸，这是因为彼此分离的三角形无法构成条带，这也是SGI算法试图最小化的。

④ 选取3个条带中最长的，删除其他两个。

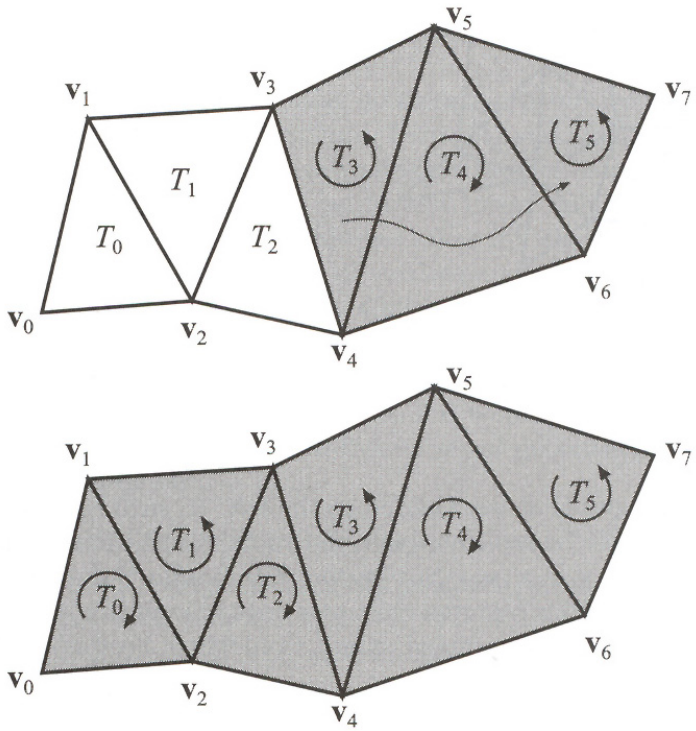
⑤ 不断重复第一步，直到条带中包含所有的三角形。

- 第(2)~(4)步中，确保可以在当前网格中找到一个包含起始三角形的最长条带。

条带延伸示例

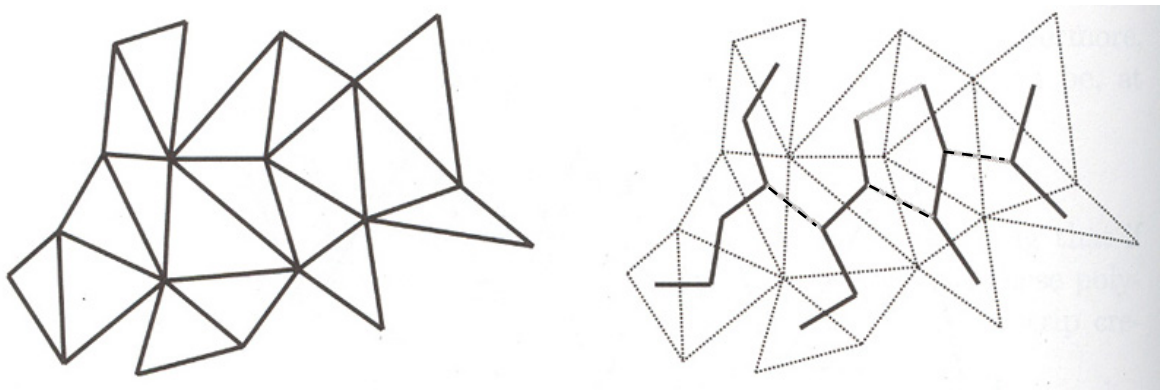
- 为了获得正确的着色效果和背面剔除效果，应保持三角形的方向。

- 三角形条带的第一个三角形决定了三角形的方向。
- 在右上图中，选择T_3作为起始三角形，向右延伸，包含T_4和T_5。
- 在右下图中，条带向左延伸以进一步加长。此时，T_3 不再是条带的起始三角形了。T0（v0v1v2）成为条带的第一个三角形。但是T0为顺时针取向，因此复制条带的第一个顶点v0，生成一个面积为0的三角形，这样整体条带就为逆时针取向。



三角条带的生成

- **Dual Graph算法**
  - 可以通过创建三角形网格的对偶图来生成三角形条带。把相邻三角形面的中心相连生成对偶图的边，这个边图称为“生成树”（**Spanning Tree**），可以用于寻找更好的三角形条带。



左图所示为三角形网格，右图所示是对应的对偶图（灰黑粗线）。图中的黑粗线表示4个不同的三角形条带。

三角条带的生成

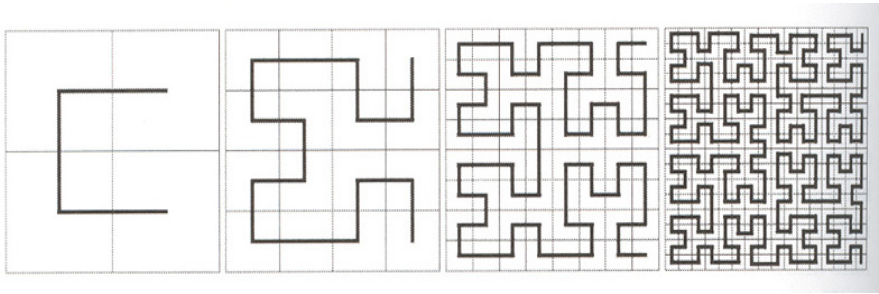
- **Cache－Friendly Stripping**
  - 假设已经生成三角形条带。许多图形卡支持顶点高速缓冲存贮器，但往管线发送条带的顺序不同，其结果性能也不同。
  - 可从某个三角形条带开始，将这个条带的顶点送入顶点高速缓存（**FIFO)**的软件模拟器中。重复检查所有的三角形条带，找出其中能最好利用高速缓冲存储器容量的条带。

三角形网格

- 三角形网格
  - 三角形条带和扇形只允许部分数据共享，而三角形网格允许全部数据共享，因此，三角形网格的效率更高。
  - 三角形网格由一个顶点表和一组轮廓线组成，其中每个顶点包括位置和其它辅助数据，如漫反射颜色、纹理等。每个三角形轮廓线由0∽n－1之间的索引组成，每个索引指向列表中的一个顶点。

三角形网格

- 创建通用的绘制序列
  - 如何为三角形网格生成通用的索引顺序，适用于任意大小的顶点高速缓冲存储器？
  - 背景知识 空间填充曲线：一条空间填充曲线是一条连续曲线，自身没有任何交叉，可以通过访问所有网格单元来填充包含均匀网格的四边形。下图为**Hilbert**曲线示例。



图中细线表示栅格，粗线表示**Hilbert**曲线。曲线可以访问每个栅格单元，而且在任何比例都具有自相似性。

  - 如果使用空间填充曲线来访问三角形网格中的三角形，在顶点高速缓冲器中会有较高的命中率。

一条好的空间填充曲线具有好的空间相关性，即当沿着曲线走时，先前访问过的点最为接近。

三角形网格

- 创建通用的绘制序列
  - **Bogomjakov和Gotsman提出的递归算法：**

    (1). 构造三角形网格的对偶图；

    (2). 为了把网格分成两个大小近似相等、互不相交的子网格，可以使用平衡边切割（**Balanced Edge－Cut**）算法删除对偶图中最小的一组边；

    (3). 以递归方式不断重复这个过程，得到顶点的索引顺序。
  - 当顶点高速缓冲存储器的大小超过**10**的时候，这种算法会优于**Xiang**的三角形条带算法。

顶点与变址缓冲器/数组

- 顶点缓存器（Vertex Buffer）
  - 为新型图形加速卡提供模型数据的最好方式是:用OpenGL顶点数组调用和DirectX顶点缓冲器调用。
  - 顶点缓冲器的思想是在一个连续的内存块中存储数据，这样通过指针驱动器就可以直接对数据访问；
  - 通过顶点缓冲器一次发送大量的数据到图形加速器可以降低系统开销；

顶点与变址缓冲器/数组

- 两种使用顶点缓冲器的方式
  - 静态数据：静态数据通常指模型在其整个生命周期内不变的。可以用图形系统进行优化。在没有硬件支持和光照支持时，可以极大地提高系统性能。
  - 动态数据：在显示方面，动态数据和静态数据很相似，但是由于代价关系，没有优化处理。

顶点与变址缓冲器/数组

- 访问顶点缓冲器
  - 如何访问顶点缓冲器依赖于设备的**DrawPrimitive**方法，可以将数据看作如下格式：

(1) 一系列点；

(2) 一系列不连接的线段，也就是顶点对；

(3) 折线；

(4) 三角形列表 （每三个顶点组成一个三角形）；

(5) 三角扇形；

(6) 三角形条带。

三角形条带通常是表示数据模型的一种最有效方式。要尽可能避免在顶点缓冲器之间进行切换，因为切换操作的开销比较大，最好把具有相同顶点格式的所有三角形放到一个缓冲器中。

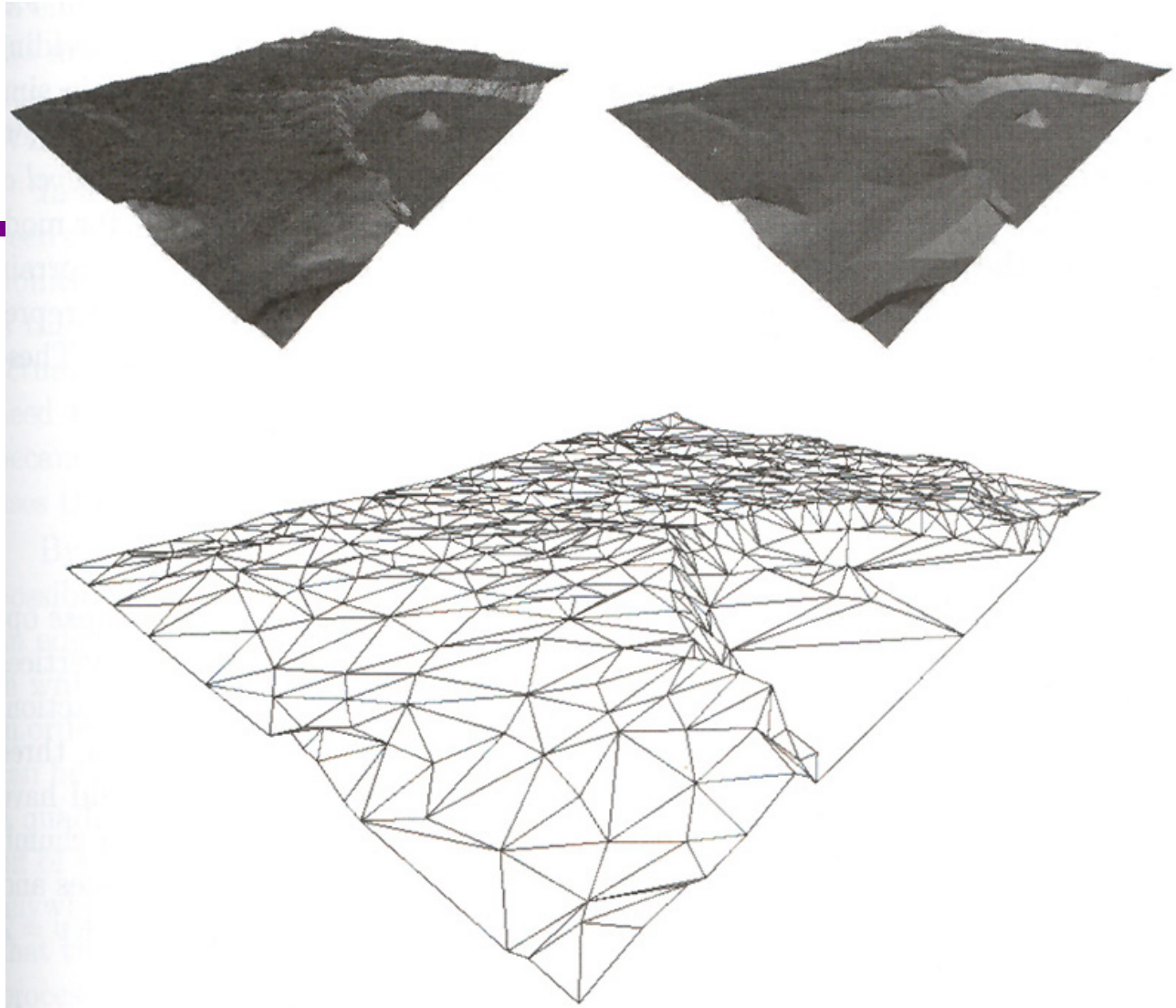
顶点与变址缓冲器/数组

- 访问顶点缓冲器
  - 快速访问顶点缓冲器可以使用索引缓冲器（**Index Buffer**）。索引缓冲器中存放三角形或三角形条带的索引。一个顶点缓冲器可以有一个关联的索引缓冲器。
  - 对顶点和索引缓冲器分类，可以通过判断两者是否驻留内存，也可以通过判断是否为只写类型。如果这两类缓冲器都声明为只读，且不在系统内存，那么就是最高效的。
  - 可以通过顶点数据类型对顶点数据进行分类。

网格简化

- 网格简化

网格简化是在视图保持图形外观不变的情况下尽可能减少精细模型的多边形数目。在实时应用中，通常需要减少存储的顶点数目并将这些顶点发送到图形流水线，使得具有一定的伸缩性。



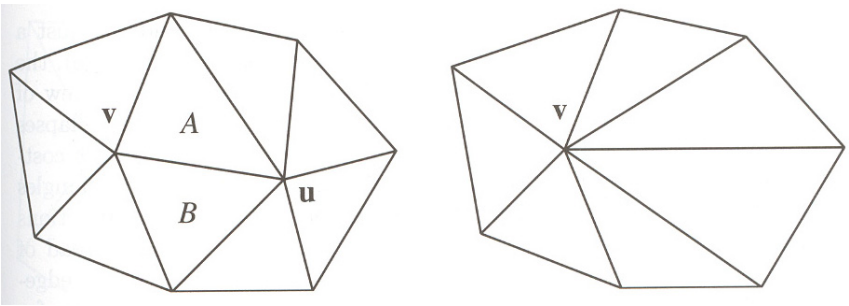
左上图是**Crater Lake**的高度场数据，使用**20**万个三角形绘制；右上图是具有**1000**个三角形的简化模型；下图是简化后的网格形式。

网格简化

- 网格简化方法
  - 静态简化： 在绘制开始之前生成离散的细节层次模型，然后对这些模型进行绘制；
  - 动态简化： 可以给出**LOD**连续模型，而不是离散模型，所以也称为连续细节层次**(Continuous Level of Detail)**算法；
  - 依赖于视点的简化： 这种技术是指模型中细节层次是变化的，地形绘制就是这样的一个例子，在实现的邻接区域需要详细的表示，在较远的地方只需要比较低的细节层次。

动态简化方法

- 边塌陷(Edge Collapse)操作
  - 这种运算是通过将两个点变为一个点的方式来删除一条边。对体模型来说，边塌陷删除2个三角形、3条边和1个顶点。



左图是边uv塌陷前的图形；右图将u塌陷到v，因此删除了三角形A和B以及边uv

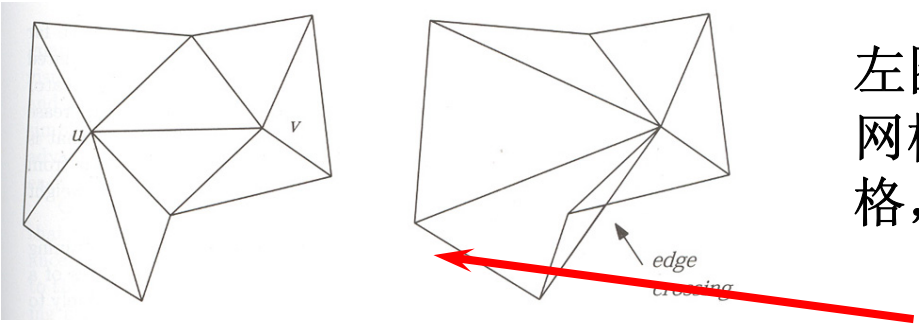
  - 边塌陷的过程可逆，通过顺序存储边塌陷信息，可以从简化模型重构出复杂模型。
  - 该特性在模型的网络传输中非常有用，因为边塌陷后的模型可以以有效的压缩格式进行传输，且在接收数据的时候以渐进方式进行重构和显示。

动态简化方法

- 塌陷策略
  - 子集放置(**Subset Placement**)策略：如上图所示，边**uv**的塌陷仅限于**u**塌陷到**v**或者**v**塌陷到**u**这样两个可能。
  - 最优放置（**Optimal Placement**）策略：不是将一个顶点塌陷为另一个顶点，而是将边的两个顶点收缩到一个新的位置。为了确定最好的位置，可以使用代价函数对每个边塌陷进行分析，对具有最小代价值的边进行塌陷操作。

动态简化方法

- **Garland和Heckbert提出的方法**
  - 思想：任何一对顶点，不仅仅是边顶点，都可形成顶点对并收缩为一个顶点。（可以处理分离表面。虽然这些表面可能非常接近，但实际上并不相连）
  - 为了限制测试的顶点对数量，规定只对在一定距离范围内的顶点进行。



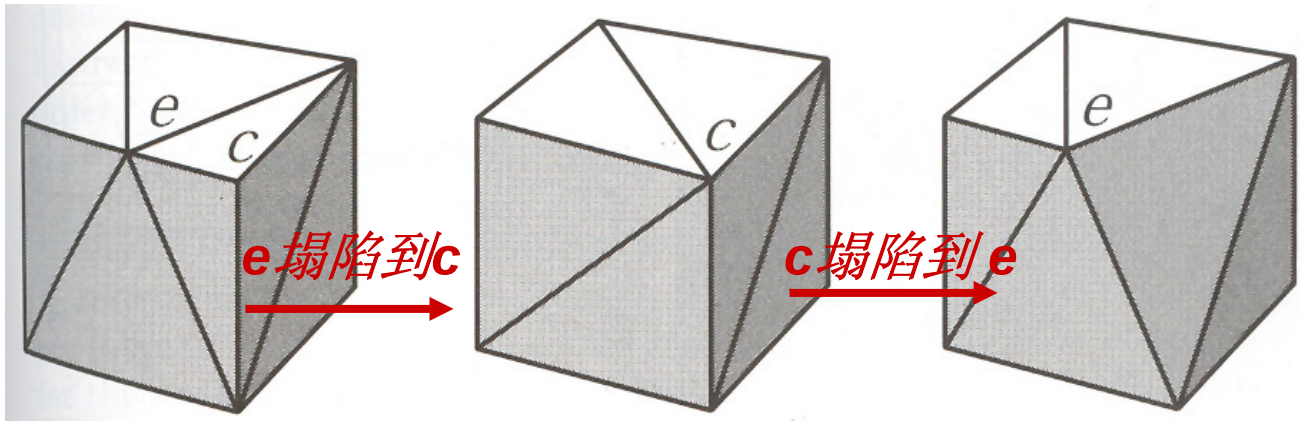
左图是顶点**u**塌陷到**v**前的网格，右图为塌陷后的网格，有边交叉的现象

  - 不论代价函数的值为多少，都要避免不良收缩。可以通过检查在塌陷后邻近的多边形是否改变法线方向来检测。

动态简化方法

- **Garland和Heckbert提出的方法**
  - 原理：对于一个给定顶点，存在一组三角形与之相邻，而且每个三角形都有一个平面方程。移动顶点的代价函数就是这些平面与该顶点新位置之间的距离平方和。为计算新位置**v**和**m**个平面的代价函数，有：

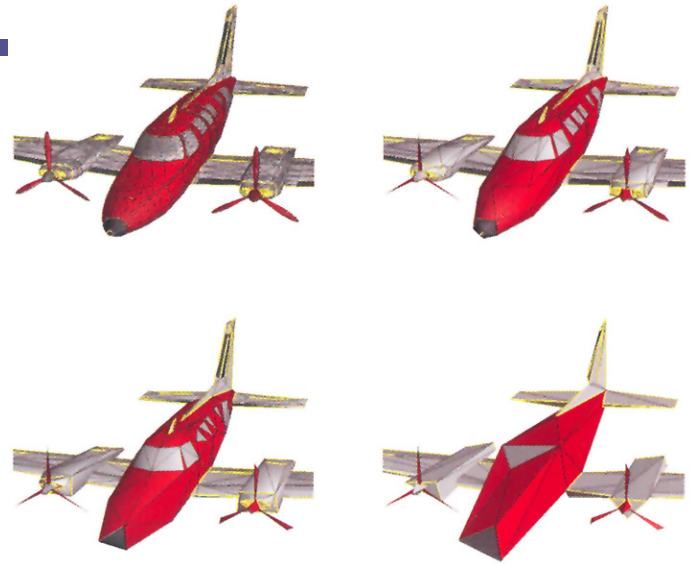
$$c(\mathbf{v}) = \sum_{i=1}^{m} (\mathbf{n}_i \bullet \mathbf{v} + \mathbf{d}_i)^2$$



- 上图为同一条边的两种可能收缩，假设正方体边长为2。因为e收缩到c时，没有离开它们共享的平面，故从e到c的塌陷代价函数值为0。因为c偏移了正方体右平面大小为1的距离，故从c到e的塌陷代价函数值为1。

动态简化方法

- 基于保持表面特征不变的代价函数
  - 例：保持模型的折痕和边界。
  - 例：保留材质变化、纹理边缘、顶点颜色变化等特征。



动态简化方法

- 基于保持表面特征不变的代价函数
  - 最好的塌陷边就是对最终图象变化影响最小、最不容易察觉的边。
  - **Lindstrom**和**Turk**提出图像驱动的塌陷函数，从一组不同的观察方向生成原始模型的图像。对模型所有可能边进行边塌陷操作，将结果与原始图像比较，选择具有最小视觉差异的塌陷边。
  - 优点在于可以直接评估每个塌陷的视觉效果。

简化算法存在的问题和解决办法

- 问题：纹理扭曲。在大部分简化算法中，一个严重问题是简化后纹理与其最初外观之间会有明显的差别。随着边塌陷，纹理到物体表面的映射也会变得有些扭曲。

- 解决方法：将网格分割为若干合理的子网格，并对子网格进行局部纹理参数化。其目的是把物体表面的位置与颜色、凹凸信息分开。

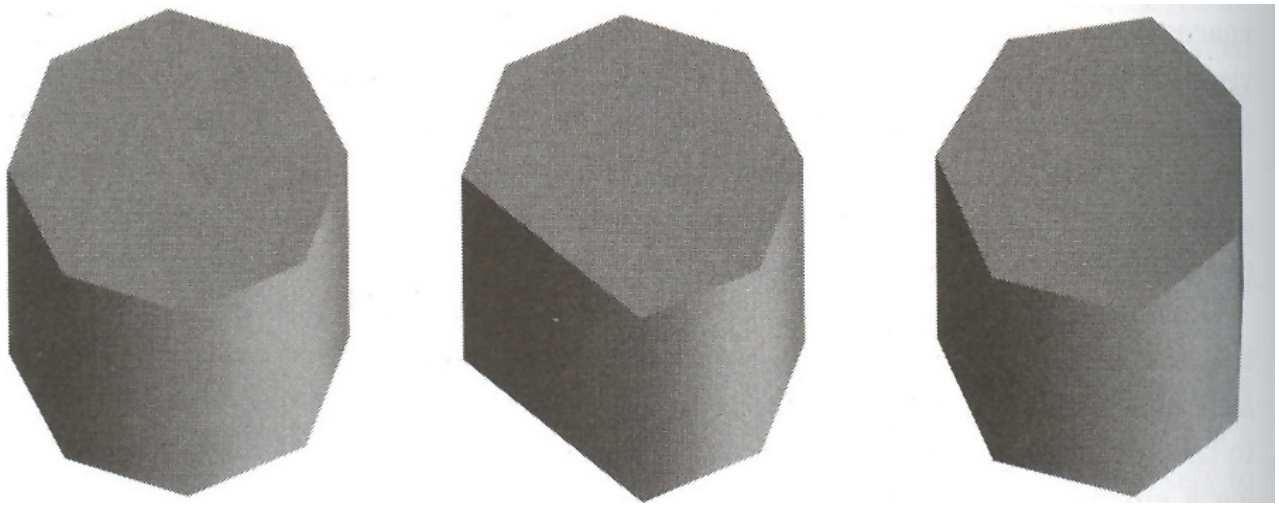
简化算法存在的问题和解决办法

- 问题：在生成**LOD**模型后，如果用一个**LOD**模型突然取代另外一个**LOD**模型，会出现视觉上的跳跃，这个问题称为**LOD**模型的突跃**(Popping)**。

- 解决方法：采用几何渐变（**Geomorph**）来增加或减少层次细节。这可根据复杂模型与简单模型之间的顶点映射来生成平滑过渡。

简化算法存在的问题和解决办法

- 对称性问题：对称变成不对称



左图中的圆柱体包含**10**个面（包括顶面和底面）；中间的圆柱体有**9**个面，其中**1**个面通过自动简化删除了一个面；右面的圆柱体通过造型软件**faceter**重新生成。

基于视点的简化方法

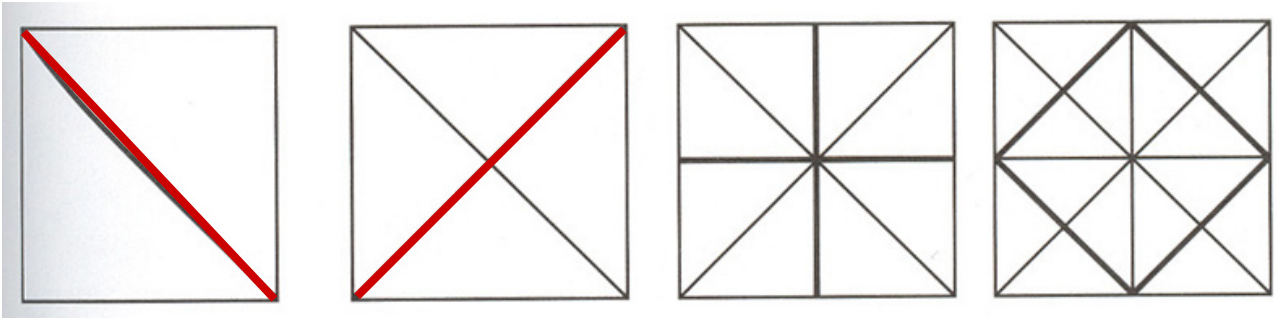
- 地形是一类具有独特性质的模型。该类数据通常用均匀高度场栅格**(Uniform height-field grid)**来表示。
- 可以用与视点无关的简化方法来处理地形数据，直到满足某些中止准则。
- 对于小的表面细节，可以通过颜色或凹凸法向贴图来捕捉。
- 得到的静态网格，通常称为三角化不规则网格**(Triangulated Irregular Network，TIN)**。当地形表面比较小且比较平坦时，这是一种非常有用的表示方式。

基于视点的简化方法

- 对于室外场景来说，经常使用连续**LOD**技术来使得需要处理的顶点数目最小化。其思想是：对于在视线范围内且靠近视点的地形，使用比较详细的层次细节表示。
- 可以使用边塌陷和几何渐变**(Geomorph)**方法，并采用一个基于视点的代价函数。
- 整个地形并不是用一个网格来表示，而是用一些较小的**Tile**来表示。从而可以用锥体剔除和潜在可见集（**Potential Visible Set, PVS**）进行**Tile**的快速剔除。

基于高度场数据的ROAM分层结构简化算法

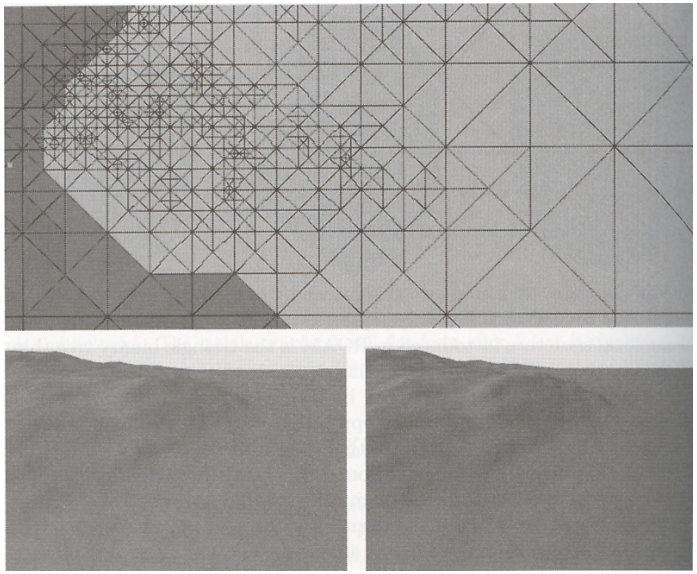
- 基本思想：将分层结构用于地形数据，然后对分层结构进行计算，最后只绘制足以表示地形的那层数据。
- 使用二叉树分层结构，直到高度场数据的极限。



三角形二叉树。左图中，用处在最高层的两个三角形表示高度场。在随后的层次上，将每个三角形一分为二。*粗线显示了分割情况。*

- 误差范围的控制：每个三角形的误差为该三角形关联地形高度场与这个三角形所在平面的最大差异。

基于高度场数据的ROAM分层结构简化算法



与视点相关的地形细分方法ROAM

- 一旦计算出每个三角形的细分层次，就可以进行细化。裂缝的避免和修复是任何一个地形绘制算法的主要部分。也就是说，如果对一个三角形进行细分后，却没有对与之相邻的三角形进行细分，则在两层之间会出现裂缝或T形点，二叉树结构有助于避免这种分割。

ROAM算法的优点

- 可以使用每个三角形的V型效果来区分需要三角形分裂的优先次序，这样可以减少需要绘制的三角形数量。

- 如果视点以均匀、平滑的方式进行变化，则可以使用前一幅画面的细化结果来作为当前帧的初始细分，从而节省遍历时间。

Further Reading

- Schneider, Philip, and David Ederly, ***Geometry Tools for Computer Graphics***, Morgan-Kaufmann Press, 2002