同济大学

硕士学位论文

虚拟现实场景中多细节层次模型及其显示的研究

姓名：朱萧峰

申请学位级别：硕士

专业：结构工程

指导教师：谢步瀛

20040201

摘　要

虚拟现实（VR）技术正在蓬勃发展，图形的实时生成是虚拟现实系统的一项关键技术。本文主要研究多细节层次模型的自动生成算法，以及应用中的一些相关问题。内容主要涉及以下方面：基于顶点移去准则、边压缩准则、三角形移去准则的 LOD 模型自动生成算法的基本原理，其实现所用的数据结构和程序流程；基于 LOD 的地形渲染技术。本论文的成果将应用于建筑虚拟漫游系统的开发。

第一章绪论，概要的叙述了虚拟现实的发展情况以及 LOD 模型的研究意义内容，并介绍了本论文的主要研究工作。

第二章介绍了编写虚拟现实系统所用的图形 API：DirectX，提出了本论文演示程序的框架结构。

第三章对 LOD 模型自动生成技术作了简要的回顾，并对常用的算法进行了简单的介绍。

第四章到第六章分别针对基于顶点移去准则、边压缩准则、三角形移去准则的 LOD 模型自动生成算法，分析了其原理与基本算法，设计了实现算法所用的数据结构和程序流程，并解决了一些相关的实际问题。

第七章对一些算法中碰到的具体问题进行了探讨，包括误差代价计算方法和连续显示问题。

第八章介绍了基于 LOD 的地形渲染技术。

第九章是本文的总结。介绍了本文的主要研究成果及有待解决的问题。

关键词：LOD，细节层次，虚拟现实，DirectX，顶点移去，边压缩，三角形移去，地形渲染

ABSTRACT

Virtual reality(VR) has been great developed in these years, and the real-time graphic rendering is one of the emphases to this technology. In this dissertation, we study on the arithmetic to generate the models automatically at multiple level of detail(LOD) and solve the related problems in application. The paper addresses the following techniques: the arithmetic for generating models automatically at multiple level of detail based on vertex removal criterion, edge compression criterion, and triangle removal criterion, the data structure and program flow for realizing these arithmetic and terrain rendering technology based on LOD. Achievements of the study will be used to develop the virtual architecture system.

In Chapter 1, we first describe the development situation of virtual reality, then present the importance and reserch content. The realated research works are also described in this chapter.

The graphic API , DirectX, which is used to develop the virtual architecture system is presented in Chapter 2. Then we design the structure of the demo program.

In Chapter 3, we briefly review the automatic generation technology of LOD models, and introduce some common methods in brief.

From Chapter 4 to Capter 6, we study on the principle and basic arithmetic of three simplify methods which are based on vertex removal criterion, edge compression criterion, and triangle removal criterion respetively. We design data structures and program flows to realize them. Some related problems are also solved in these chapters.

In Chapter 7, we probe into some problems met in those arithmetics, including error cost method and continuous rendering.

Terrain rendering technology based on LOD is presented in Chapter 8.

We make a conclusion in Chapter 9. In this chapter, we can see what achievements we have got and what is to be done in the future.

Keywords: LOD(level of detail), virtual reality, DirectX, vertex removal, edge compression, triangle removal, terrain rendering

声　明

本人郑重声明：本人在导师的指导下，独立进行研究工作所取得的成果，撰写成硕士学位论文“虚拟现实场景中多细节层次模型及其显示的研究”。除论文中已经注明引用的内容外，对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本论文中不包含任何未加明确注明的其它个人或集体已经公开发表或未公开发表的成果。

本声明的法律责任由本人承担。

学位论文作者签名：朱高峰

2004 年 3 月 15 日

第一章：绪论

1.1 虚拟现实简介

虚拟现实(Virtual Reality，简称 VR)是一种新的人一机界面，它为用户(参与者)提供了一种具有临场感和多感觉通道的体验，试图寻求一种最佳的人一机通讯方式。虚拟现实也被称为人工现实(Artificial Reality)、电脑空间(Cyberspace)、人工合成环境(Synthetic Environment)、虚拟环境(Virtual Environment，简称 VE)。

VR 技术并非最近几年才出现，它的起源要追溯到计算机图形学之父 Ivan Sutherland 于 1965 年在 IFIP 会议所作的标题为“The Ultimate Display”的报告。在该报告中，Ivan Sutherland 提出了一项富有挑战性的计算机图形学研究课题。他指出，人们可以把显示屏当作一个窗口来观察一个虚拟世界。其挑战性在于窗口中的图像必须看起来真实，听起来真实，而且其中物体的行为也很真实。在该文中，Sutherland 博士给出了我们今天称为虚拟现实概念的经典描述。这一描述包含以下几个要点：

1. 计算机生成的模型世界将向介入者--人，提供视觉、听觉、触觉等多种感官刺激。

2. 计算机生成的虚拟世界应给人一种身临其境的沉浸感。

3. 人能以自然的方式--手势、体姿甚至自然语言--与虚拟世界中的对象进行交互。

这一思想奠定了 VR 研究的基础。

1968 年，Ivan Sutherland 发表了题为“A Head--Mounted 3D Display”的论文，对头盔式三维显示装置的设计要求、构造原理进行了深入的讨论。Sutherland 还给出了这种头盔式显示装置的设计原型，成为三维立体显示技术的奠基性成果。

VR 系统可分为三大类：桌面 VR 系统、沉浸式 VR 系统和分布式 VR 系统。桌面 VR 由于采用标准的 CRT 显示器和立体显示技术，其分辨率较高，价格较便宜。在使用时，桌面 VR 系统设定一个虚拟观察者的位置。桌面 VR 系统通常用于工程 CAD、建筑设计以及某些医疗应用。沉浸式 VR 系统利用头盔显示器

把用户的视觉、听觉和其它感觉封闭起来，产生一种身在虚拟环境中的错觉。分布式 VR 系统则是在沉浸式 VR 系统的基础上将不同的用户通过网络联结在一起，共享同一个虚拟空间，使用户达到一个更高的境界。

VR 技术在各行各业的应用具有很大的潜力和十分广阔的前景。下面介绍 VR 已有的一些应用成果。

(1) 医疗和康复

近年来，计算机在医学界的广泛应用(如病人数据库、手术模拟、远程咨询、数字化 X 射线照片、专家系统等)，改变了人们医疗和康复的模式，特别是 VR 技术在医疗和康复中的应用，将根本改变这一方式。目前，VR 技术在这一领域的具体应用系统及有关研究包括：

- 虚拟解剖训练系统，该系统主要用于教学。
- 外科手术模拟系统。
- 遥控手术，其原型系统 SRI 远程手术模拟系统也已开发成功。
- 手功能诊断，利用数据手套来检查病人手功能失调的程度。如 Greenleaf 医疗系统公司的运动分析系统，就是一个商品化的例子。
- 虚拟训练机，是帮助病人康复训练的 VR 系统。
- 帮助残疾人改变生活方式的有：轮椅虚拟现实系统、会说话的手套、聋哑人电话等。
- 用于新药试制的三维分子结构显示与操纵的 VR 系统，有纽约大学与 Division 公司联合开发的 Grope II 等。

(2) 娱乐、艺术和教学

在 VR 技术发展的早期，娱乐是 VR 技术发展的主要和直接推动力，而现在又成了 VR 系统的主要市场之一。在公共场所获得成功的 VR 游戏项目有：世界上第一个大型 VR 娱乐系统 Battle Tech 中心，娱乐中心于 1990 年在芝加哥开张，可供数十人共同玩战争游戏。此外，还有 Hughes 飞机公司和 LucasArts 娱乐公司等开发的 Mirage VR 模拟器。W.Industries 有限公司于 1991 年开发了世界上第一个采用头盔显示器的娱乐系统，称为 Viruatily System，并获 1992 年 VR 产品奖。

VR 技术在未来的艺术领域将扮演重要的角色。它作为一种新的媒体不仅能变静态艺术为动态艺术，而且将成为联结艺术创作者与欣赏者的重要纽带。目前，

与艺术有关的应用系统有：Videoplace、Mandale、虚拟演员(Virtual Actors)、虚拟博物馆(The Virtual Museum)、虚拟音乐(Virtual Music)等。

VR 技术在教育领域特别是在中小学教育中的作用，目前尚处于研究证实阶段，但其前景是十分诱人的。由 Loffin 及其同事开发的虚拟物理实验室，是一个用于演示牛顿力学及量子物理有关定律的虚拟实验教学系统。

(3) 军事和航空航天

在军事和航空航天领域，模拟和训练特别重要。随着当今技术的高度复杂化，硬件系统的开发周期不断缩短，迫切需要一种灵活的、可升级的、网络化的模拟系统。VR 技术在上述各个方面均能很好地满足这一需要。因此，美国政府充分认识到 VR 技术在保持美国技术领先方面具有的战略意义，并制定了一系列的实施计划。其中应用于军事方面的 VR 系统有：

- 坦克训练网络 SIMNET。
- 虚拟毒刺导弹训练系统。
- 反潜艇作战系统 ASW。在航空航天领域的 VR 系统有：
- NASA 虚拟现实训练系统。
- EVA 训练系统。
- 虚拟座舱。

(4) 商业应用

VR 技术在商业领域最早也是最成功的应用是产品广告宣传。VR 广告比传统广告更易于制作和更改，也更具有感染力。由此，许多公司已认识到率先使用 VR 技术可使他们的产品领导潮流。已有的一些应用包括：

- Maxus 证券交易可视化系统，如哥伦比亚大学研制的 n-Vision 系统。
- 体验广告，这类广告在美国多附着于公共场所的 VR 娱乐设施；在欧洲则开发出不少独立的 VR 广告系统，如 Kingston Micro Electronies、Callscan 、Madrid City Council 等。
- 室内装潢设计，如 Light-space Software 公司开发的灯饰可视化系统 LVS(Lightcape Visualization System)和 LEL 模拟系统(The Living Environmental System)等。

(5) 自动控制和制造业

VR 技术一旦全面应用于自动控制和制造业，必将产生巨大的经济效益。但

是由于该领域固有的技术复杂性，VR 技术在这一领域的应用仍处于早期的研究和开发阶段，比如：

• 机器人辅助设计。

• 脱机编程，利用 VR 技术仿真机器人，编程调试不影响控制过程，大大减少了时间损耗。

• 远程操作，可提供对远程恶劣环境控制系统的精确控制。

在制造业领域，VR 技术可能将零星替代并超过目前广泛使用的 CAD 系统。它可将产品需求分析、时间成本分析和产品设计，甚至将相应的生产线设计集成在一起，以进一步缩短新产品的研制开发周期，降低成本。特别是对于产品定制和售后服务，VR 技术的应用具有更大的优势。

1.2 VR 中真实感图形的实时显示

虚拟现实技术的核心，是介入者的沉浸感，而视觉沉浸感的两个基本的构成要素：真实感和实时性。

真实感是虚拟现实技术对图形显示技术的基本要求之一。只有能绘制出具有足够真实感，能够“以假乱真”的图像，虚拟环境的介入者才会产生足够的沉浸感。反之，粗糙的、缺乏细节的图像会提醒介入者：这是虚拟的世界，是“假”的，不但不会给介入者带来沉浸感，反而会带来“剥离感”，使介入者产生与虚拟环境的对立。

就目前计算机图形学水平而言，只要有足够的计算时间，就能生成准确的象照片一样的计算机图像。例如采用光线追踪(ray tracing)或辐射度方法(radiocity)的光学/物理模拟方法，就可以得到非常逼真的渲染效果。

但 VR 系统要求的是实时显示，所谓实时显示，是指当用户的视点变化时，图形显示速度必须跟上视点的改变速度，否则就会产生迟滞现象，要消除迟滞现象，计算机每秒钟必须生成 10 帧到 20 帧图象。

当场景很简单时，例如仅有几百个多边形，要实现实时显示并不困难，但是，为了得到逼真的显示效果，场景中往往有上万个多边形，有时多达几百万个多边形。此外，系统往往还要对场景进行光照明处理、反混淆处理及纹理处理等等，这就对实时显示提出了很高的要求。

要提高图形显示速度，关键性的在于计算机图形硬件的发展。近几年，计算机3D图形硬件加速技术取得了革命性的进展，无论是图形质量还是显示速度都飞速发展，并从专业领域步入了民用领域。

然而应用模型的复杂程度往往超过当前计算机图形硬件的实时处理能力，考虑到VR对场景复杂度几乎无限制的要求，在VR高质量图形的实时生成要求下，如何从软件着手，减少图形画面的复杂度，已成为VR中图形生成的主要目标。

降低场景的复杂度，即降低图形系统需处理的多边形数目。目前，比较常用的方法有预测计算、脱机计算、场景分块、可见消隐及细节层次模型等。

前面几种由于不在本文的研究范围，在此就不具体介绍。

所谓细节层次(Level of Detail, 简称LOD)模型方法，即为每个物体建立多个相似的模型，不同模型对物体的细节描述不同，对物体细节的描述越精确，模型也越复杂，根据物体在屏幕上所占区域大小及用户视点等因素，为各物体选择不同的LOD模型，从而减少需显示的多边形数目。

1.3 LOD模型的研究内容和意义

一个采用LOD模型的系统中，要解决的问题有以下这些：LOD模型的选择尺度，LOD模型的选择算法，LOD模型的平滑过渡方法以及LOD模型的自动生成。 其中，LOD模型的自动生成是以上问题的基础，本文主要研究LOD模型的自动生成算法。

LOD模型的研究意义在于为解决图形显示质量和图像显示速度之间的矛盾提供了一条可行而有效的方法。它改变了图形越精确越好的片面看法，而是依据视线的主方向、视线在景物表面的停留时间、景物离视点的远近、景物在画面上投影区域的大小和场景中每个图形对象的重要性等因素来决定景物应选择的细节层次。

1.4 三维图形API

计算机三维图形是指将用数据描述的三维空间通过计算转换成二维图像并显示或打印出来的技术，API(Application Programming Interface)即“应用程序接口”是连接应用程序与操作系统、实现对计算机硬件控制的纽带，Direct3D

（DirectX8.0 以后，Direct3D 和 DirectDraw 合并为 DirectGraphics）和 OpenGL 是目前的两大 3D 图形 API。

图 1.1 Direct3D

作为微软 DirectX 技术的组件之一，Direct 3D 也随着 DirectX 的升级而不断更新，同时在微软的全力扶植下，Direct 3D 技术的发展速度极快，DirectX 7：正式支持硬件 T&L（光影变换）、DirectX 8：对 Pixel Shader（像素着色器）Vertex Shader（顶点着色器）的支持、DirectX 9：提供 2.0 版本的可编程顶点和像素着色模式，显卡硬件厂商也纷纷以对最新的 D3D 特效的硬件支持为卖点。遗憾的是，由于平台的局限性等原因，D3D 应用至今仍主要集中于游戏和多媒体方面，专业高端绘图应用方面，老牌的 3D API---OpenGL 仍是主角。

OpenGL 的英文全称是“Open Graphics Library”即“开放的图形程序接口”，它是计算机工业标准应用程序接口，主要用于定义二维三维图形。OpenGL 是一套底层三维图形 API，之所以称之为底层 API，是因为它没有提供几何实体图元，不能直接用以描述场景。但通过一些转换程序，可以很方便的将 AutoCAD、3DS 等图形设计软件制作的 DFX 和 3DS 模型文件转换成 OpenGL 的顶点数据。

图 1.2 OpenGL

1.5 本论文的主要研究工作及组织

一套完整的虚拟建筑漫游系统，主要结构如图 1.3。

本论文所作的工作属于场景管理部分。建好的模型要进入渲染引擎进行渲染，如果多边形数量过于庞大，会给整套系统造成极大的影响，造成性能上的瓶颈。本论文就是要在模型数据进入渲染引擎以前，设计算法自动简化，使得进入渲染引擎的模型数据尽量的减少。

本论文主要研究了三个常见的模型自动简化算法：基于顶点移去准则的 LOD 模型自动生成算法、基于边压缩准则的 LOD 模型自动生成算法以及基于三角形移去准则的 LOD 模型自动生成算法。

然后，本文又介绍了基于 LOD 的地形生成技术，使用四叉树的数据结构对地形进行组织，并介绍了相应的简化算法。

本论文的成果直接用于建筑虚拟漫游系统的开发。

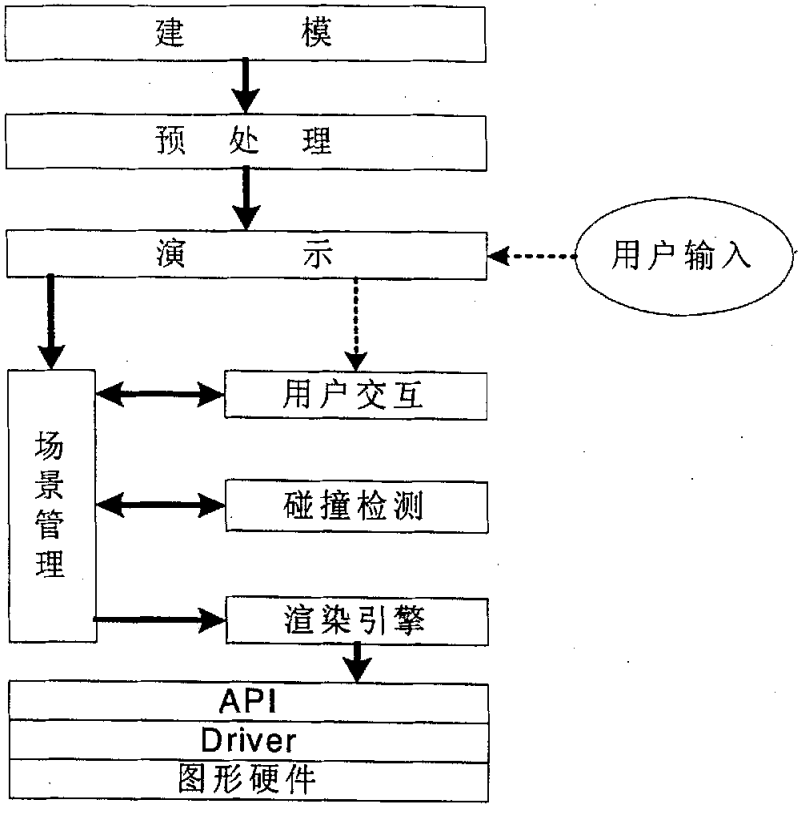


图 1.3 建筑虚拟漫游系统组成

第二章：DirectX图形技术

2.1 DirectX的发展历史

自从 Windows3.1 的 GUI 图形介面登场以后，越來越多的人开始使用 Window。不过，Windows3.1 的图形处理的能力很弱，因此当时游戏厂家主要还是把电脑游戏开发在 MS-DOS 下。

后来，微软在 94 年推出了 WinG。WinG 拥有较高速的图形处理能力，其后又推出了能使得影像高速化的 WinToon，但是这些产品的实用性不大。一直到 Window95 的问世，微软开发了 DirectX1.0，并且根据硬件制造厂商和游戏厂商合作共同更新升级 DirectX 的标准。

图 2.1 DirectX9

随着 DirectX 版本的升级，功能越來越强大，电脑游戏开始抛弃 MS-DOS。发展到现在，DirectX 已经成为游戏的工业标准。

下表是 DirectX 的发展历史：

DirectX 的历史			
Directx 版本	标志性技术	标志性硬件	标志性效果
1.0	——	——	——
2.0	d2d 成熟	trident 9680、s3 virge	2d 动态效果
3.0	d3d 雏形	rival28、i740	简单 3d 效果
5.0	基本 3d 技术	tnt	雾化、阿拉法混合
6.0	成熟 3d 技术	tnt2	双/三线过滤
7.0	t&&l	geforce256、radeon	凹凸映射
8.0	ps、vs	geforse 3、radeon 8500	水波纹
8.1	ps、vs2.0	geforce 4	大纹理水波纹
9.0	ps、vs3.0	nv30、radeon9700	皮毛效果

2.2 DirectX的组成

DirectX 的意思不难理解，Direct 是直接的意思，X 是很多东西，加在一起就是一组具有共性的东西，这个共性就是“直接”。微软将其定义为“硬件设备

无关性”。DirectX 主要由以下几个部分组成：

(1)DirectX Graphics（8.0 以前这部分分成 Direct3D 和 DirectDraw 两部分，分别为 3D 图形部分和 2D 图形部分）

(2)DirecSound（声音相关）

(3)DirectMusic（MIDI 相关）

(4)DirectInput（输入相关）

(5)DirectPlay（网络相关，如 IP 声音通讯）

(6)SirectSetup（Setup 相关）

(7)DirectShow（动画相关）

(8)DirectX Media Objects（媒体数据流相关）

其中，DirectX Graphics 即图形部分是我们的研究重点。

2.3 DirectX 9 中出现的新技术及改进

- **Vertex Shader 和 Pixel Shader**

从 DX8 开始，3D 图形处理技术逐渐统一在 Vertex Shader 和 Pixel Shader。Vertex Shader 被用来描述和修饰 3D 物体的几何形状，同时也用来控制光亮和阴影；Pixel Shader 则用来操纵物体表面的色彩和外观。最新版本 shader 的推出总能为我们带来更强悍的性能和更优异的画质。DX9 采用了 2.0 版的 Vertex Shader 和 Pixel Shader，它们都将支持 64 或甚至 128 位浮点色彩精度。浮点色彩在动态和精度上的增加给图像质量带来质的飞跃，还让很多过去不可能的特效变成现实。

Vertex Shader 2.0 引入了流程控制，增加了条件跳转、循环和子程序。Vertex 程序现在最多可以由 1024 条指令组成（之前只能用 128 条指令），但这只是理论上的数字，拥有循环和跳转指令自然就可以允许更多数量指令的执行。这大大加强了可编程性和效率。

强大的可编程 Pixel Shader 是得以实现具有真实照片和电影质量级别效果的真正精华所在。Pixel Shader 2.0 可以支持高级程序语言和汇编语言，开发者还可以将其汇编代码嵌入较高级的程序语言中。前一版的 Pixel Shader 语言被限制为只能使用最多 6 个材质和 28 条指令，而 2.0 版则将这一上限提升至最大 16 材质

和 160 条指令，也新增了很多强化的运算和操作（如压缩的凹凸贴图等，这意味着能够在游戏中使用更多细致的凹凸贴图效果。）使得编程工作更加轻松和高效。Pixel Shader 的早期版本的另一限制在于一次只能进行一个色彩值的输出。多重渲染物体（Multiple Render Targets）突破了这个限制，允许一次输出 4 个单独的色彩值，这将极大提高 Pixel Shader 程序的效率。

- 浮点型色彩和 **32bit** 帧缓冲格式

目前多数的色彩应用使用各 8bit 长的整型数来表达红、绿和蓝色，这意味着每一原色的取值从 0~255。表面上看，目前的色彩精度对于显示而言也许是足够的，但在运算中就是另外一回事。由于整型数不具有小数部分，因此当它们经过 Pixel Shader 中极其复杂的数学运算后，会出现较大的偏差，这个缺陷将会导致明显的颜色失真。DX9 支持数种新的浮点型颜色格式，其精确度具有极大的提高。

同时，DX9 提供新的精确至每像素 32bit 的帧缓冲格式，能够表现出 4 倍于目前亮度等级的数量，而目前只能提供 256 级不同的亮度表现。人眼可以分辨数百万种不同层次的亮度，亮度等级范围的提升使得人们对数量大增的亮度等级会有完全不同的感觉。如果亮度级别的数量太少，图像看上去要么曝光过度，要么曝光不足，而亮度等级更精确的图像则看上去感觉更加柔和自然。

- **Displacement Mapping**

DX8 开始引入了一种被称为 N-patch 的新的 3D 表面技术，该技术能够通过较少的多边形构造出看上去很圆滑而且感觉自然的物体。DX9 中扩展了 N-patch 技术的概念，使得其更加强大。相较于仅仅对存在的几何图形进行平滑处理，置换贴图（Displacement Mapping，也被称为位移贴图）技术能够添加更多复杂的细节到一个物体上。置换贴图可以通过一张基本纹理、一张光照纹理以及一张最为重要的高程纹理来完成模型外观的构造，模型本身也许只是一个平面或圆柱，但是使用了置换贴图以后，就可以生成一个地形复杂的丘陵或五官俱全的人头。当然，这个“建模”过程也是需要 Vertex Shader 来协助的，并非以往环境凹凸贴图那样只是一个单纯的贴图过程。不同于凹凸贴图，位移贴图技术精确地描述了物体的外形，使得其光线反射和阴影投射的效果更加真实。

- **Gamma** 修正

Gamma 修正可以改善色彩表现，并获得更好的材质光影效果。有些硬件产

品中已经有这项功能，大多是相当高档的产品，但现在它们正逐渐成为主流。DX9 将促使这些高端产品进一步平民化。

2.4 DirectX Graphics 的流水线结构

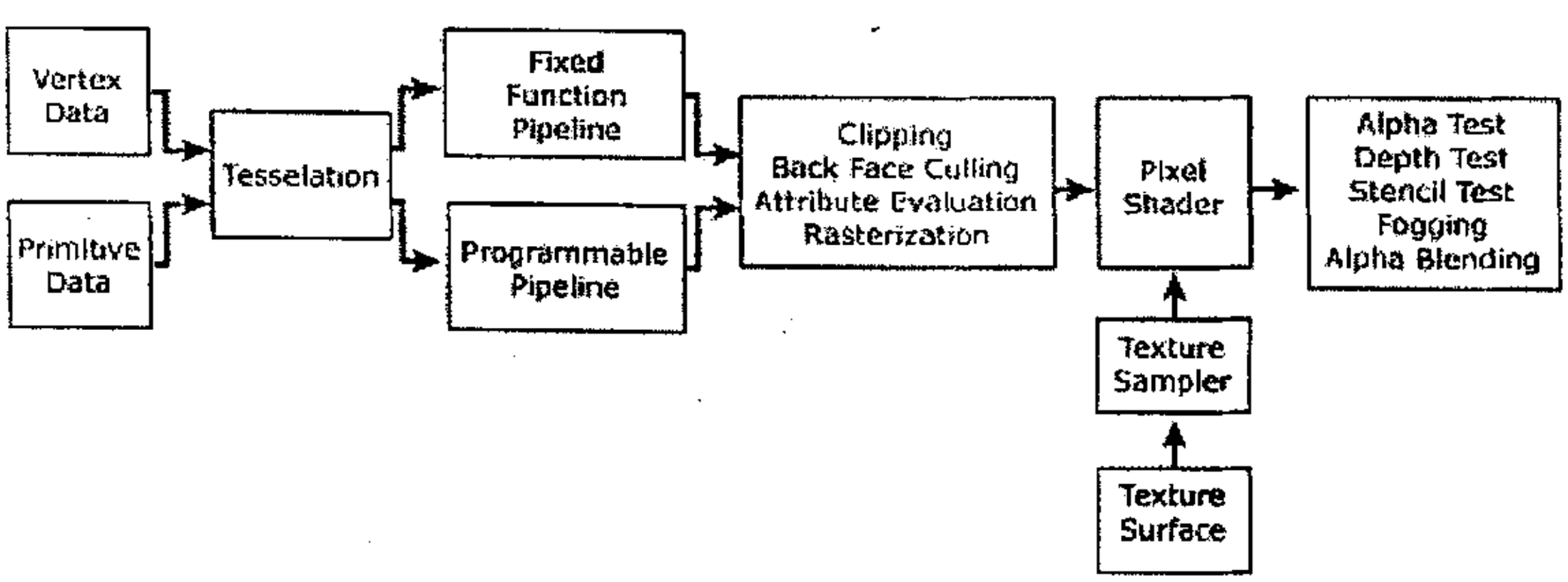


图 2.2 DirectX Graphics 的流水线结构

2.5 使用 COM 组件

DirectX 的功能都是以 COM 组件的形式提供的。COM 是组件对象模型（Component Object Model）的简写，它是一种协议，用来实现软件模块间的二进制连接。当这种连接建立后，两个模块之间就可以通过称为“接口(Interface)”的机制来通信。 DirectX 的所有功能都被归纳到若干个 COM 接口中，用户在使用 DirectX 的功能调用之前，需要先创建并获得相应的 COM 接口。

在 Direct3D 编程中，我们要做的工作基本上可以归纳为：

(1)调用适当的函数获取接口指针；

(2)调用接口的方法（成员函数）来完成所需功能；

(3)用完接口后，调用 Release 方法进行“释放”，注意释放顺序应该和获取它们的顺序相反。

2.6 Direct3D 的程序结构

程序采用标准的 WIN32 编程模式，需要自己编写 WinMain 主函数，并负责创建主窗口。WinMain 的函数大致如下：

```
... ...
//创建主窗口，也就是Direct3D窗口
HWND hWnd = CreateWindow(... ...
//初始化Direct3D
InitD3D();
//建模
InitGeometry();
//进入消息循环
MSG msg;
ZeroMemory( &msg, sizeof(msg) );
while( msg.message!=WM_QUIT )
{
if( PeekMessage( &msg, NULL, 0U, 0U, PM_REMOVE ) )
{ //如果有消息，则处理消息
TranslateMessage( &msg );
DispatchMessage( &msg );
}
else
//如果没有消息，就调用Render
Render();
}
//释放接口
Cleanup();
... ...
```

前面讲到，要使用 DirectX 的功能，必须先创建并获得相应的 COM 接口，首先是一个 IDirect3D9 接口：

```
LPDIRECT3D9 pD3D; //Direct3D对象的接口指针
```

在 InitD3D()中将它创建：

```
pD3D = Direct3DCreate9( D3D_SDK_VERSION )
```

然后是一个设备接口：

```
LPDIRECT3DDEVICE9 pDevice; //设备对象的接口指针
```

并利用上面的 IDirect3D9 接口的 CreateDevice 函数创建它：

```
pD3D->CreateDevice( D3DADAPTER_DEFAULT,
                    D3DDEVTYPE_HAL,hWnd,
                    D3DCREATE_SOFTWARE_VERTEXPROCESSING,
                    &d3dpp, &pDevice);
```

作为 Direct3D 中的渲染部件，设备对象用于顶点变换、光照处理以及矢量图形的光栅化。Direct3D 目前支持两种设备类型：HAL（硬件抽象层 Hardware Abstraction Layer）和 Reference。前者启用硬件三维加速功能，需要显卡支持；后者则以软件模拟方式完成三维处理，虽然速度慢，但可以在任意显卡上运行，一般用于调试程序，其对应的参数为 D3DDEVTYPE_REF。

上述代码使用了软件顶点处理方式，如果显卡支持硬件 T/L，可以选用硬件方式以提高效率，调用参数为：

D3DCREATE_HARDWARE_VERTEXPROCESSING

渲染部分程序结构：

```
void Render()
{
  //用指定颜色清除后备缓存区
  pDevice->Clear(
  0, NULL, D3DCLEAR_TARGET,
  D3DCOLOR_XRGB(0,0,255), //指定使用蓝色
  1.0f, 0);
  //Direct3D规定在渲染前必须调用方法
  //IDirect3DDevice9::BeginScene,
  //结束时要调用IDirect3DDevice9::EndScene。
  pDevice->BeginScene();
  //实际的渲染代码放在此处。
  pDevice->EndScene();
  //交换当前/后备缓存区，刷新窗口
  pDevice->Present(NULL, NULL, NULL, NULL);
}
```

具体的渲染过程大致为：首先根据模型建立顶点缓冲区；然后设置变换矩阵，也就是先计算三个变换矩阵：世界变换矩阵、视角变换矩阵、投影变换矩阵；接

着使用 IDirect3DDevice9::SetTransform()将这三个变换矩阵设置到渲染环境；最后绑定顶点缓冲区至设备数据源并绘制图元，大致代码如下：

```
... ...
pDevice->BeginScene();
SetupMatrices(); //设置变换矩阵
//设置自定义的FVF
pDevice->SetFVF( D3DFVF_CUSTOMVERTEX );
//绑定顶点缓存区至设备数据源
pDevice->SetStreamSource( 0, m_pVB, 0,
    sizeof(CUSTOMVERTEX) );
//绘制图元，其中参数1为图元格式，参数3为三角形数目
pDevice->DrawPrimitive( D3DPT_TRIANGLELIST, 0, 1 );
pDevice->EndScene();
... ...
```

第三章：LOD 显示和简化技术综述

虚拟现实中的物体模型，很大一部分是通过激光扫描测距系统扫描真实三维物体而得到的，为真实反映原物体的表面变化，扫描过程中所采取的采样点非常稠密，导致虚拟现实场景中的多边形数量非常大。而计算机能够实时处理的多边形数量总是有限的，为了保证虚拟现实场景的实时生成，以便进行交互式的控制、和观察，在当前图形硬件处理能力仍然有限的情况下，必须想办法降低场景的复杂度。细节层次（LOD：Level of Detail）显示和简化技术就是在这种背景下提出来的。

3.1 LOD 研究历史

早在计算机发展的初期，人们已经认识到层次模型表示的重要性。Clark 在 1976 年就已提出了用于可见面判定算法的几何层次模型，这是因为三维物体和环境定义中固有的几何结构性质，不仅可用来定义物体的相对运动和所处位置，而且有助于解决图形学中许多其它相关的一些问题。实际场景的复杂性往往超过计算机能进行实时处理的能力，为了在计算机中以较少的时间及较高的图像质量绘制出复杂的场景，Rubin 提出了复杂场景的层次表示方法及相关的绘制算法。Marshall 等人提出的生成三维地形的过程建模方法，也包含了细节层次的思想。然而，前面所述的层次模型一般都是人工建立的。

在多细节层次模型自动生成方面，国际上已开展了一些研究工作。Schroeder 等人提出了基于顶点移去的模型简化方法。该方法首先利用各顶点的局部几何和拓扑信息将各顶点分类，然后根据不同顶点的评判标准决定该顶点是否可以删除。如果可以删除，则采用递归环分割法对删除顶点后所留下的空洞进行三角剖分，否则保留该顶点。Hamann 给出了一种基于三角形移去的模型简化方法。该方法首先对给定模型中的所有顶点计算其曲率，然后由顶点的曲率决定所有三角形的权值，权值最小的三角形被移去，受移去三角形影响的区域则被重新局部三角剖分，对所有新引入的三角形计算其权值，重复以上过程，直到移去了所要移

去三角形的数目为止。

Turk 给出了基于网格重新划分的多边形模型简化方法。该方法首先把一组新顶点分布到原模型上，然后新老顶点形成中间网格，通过必要的拓扑结构一致性检查移去老顶点，并进行局部的三角剖分操作，形成新的与原模型具有相同拓扑结构的网格模型。为了给曲率变化较大的区域分布更多的新顶点，以使重新划分的三角形网格模型较精确地反映出原模型，该算法还根据各顶点的曲率近似计算，用三角形顶点曲率与三角形面积的乘积作为权因子决定新顶点的放置，并根据顶点间的排斥半径将新顶点定位在原多边形内。

Kalvin 给出了面片合并方法自动生成物体的简化模型。该方法首先以原模型中的任一多边形作为“种子”面片，按一组合并条件不断合并种子面片周围的面片，直到周围面片不再满足合并条件。同时，算法根据用户定义的误差给出种子面片的近似面片集合，并根据不断合并的种子面片修改近似面片集合，直到集合为空。

自适应递归划分是曲面曲线生成的常用方法。Schmitt 将其与曲面拟合技术相结合，用不断逼近的曲面片逼近所要简化的物体模型，为规则四边形网格表示物体建立起 Bezier 曲面表示的简化模型。基于此方法，Dehaemer 提出了基于规则四边形网格表示物体的简化多边形模型的自动生成方法。

以上方法都要求生成的简化模型保持拓扑关系和几何关系的正确性。但 Rossignac 认为，简化模型只要其图形效果与原模型一致，则该简化模型成立。为了加快实时图形生成，Rossignac 还提出了一种多分辨率近似法自动生成物体的简化模型。该方法首先赋予各顶点一个权值，给物体特征变化较大处的顶点以较大的权值。然后根据模型的复杂程度、相对大小等将此物体所占空间划分为立方体的的单元。同一单元中的的顶点，以各顶点的权值计算这一单元中所有顶点的代表点，然后依据原模型中各多边形顶点的代表点是否为同一代表点合并各多边形。

另外，Hoppe 采用了能量函数最优化的网格简化方法，并提出了累进网格的生成方法。Eck 等人则利用小波变换技术将多面体模型表示为多分辨率形式，由此可以自动生成一系列连续的原模型的近似模型。该方法产生的近似模型与原模型的误差可控。

最近，国外已有一些学者提出了误差可控的三角形网格模型的简化方法，比

如：Bajaj 等人提出了通过误差传递方法来度量简化模型与原模型的误差程度，从而得到误差可控的简化模型。Klein 等人提出了基于 Hausdorff 距离来度量简化模型与原模型的误差，实现了有效的误差可控的网格简化新方法。

在多细节层次绘制技术研究方面，美国加州大学伯克利分校的 Funkhouse 等人把不同细节层次的场景和不同精度的绘制算法结合起来进行研究，提出了一种自适应显示算法，用该算法生成的图象能满足稳定的帧速率，在虚拟环境中获得成功应用。美国佐治亚技术学院的 Lindstrom 针对规则网格表示的地形模型提出了一种实时连续细节层次绘制方法，在三维地形仿真显示中得到了应用。

国内在以上两方面也开展了一些研究工作。例如，对已有的网格简化算法进行改进或提出新的多面体模型简化方法，给出了虚拟环境中恒帧速率自适应显示算法等等。

3.2 LOD 常见算法简介

对于原始网格模型的不同细节层次的模型的建立，我们假设场景的模型都是三角形网格（在实际应用中，为了绘制方便，三维场景最后一般都被转化为三角形网格），从网格的几何及拓扑特性出发，存在着三种不同基本化简操作：

(1)顶点移去

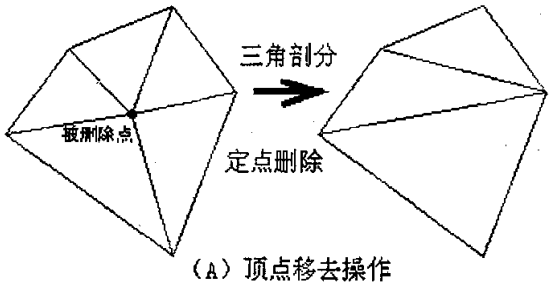
删除网格中的一个顶点，然后对它的相邻三角形形成的空洞作三角剖分，以保持网格的拓扑一致性。如图 3.1(A)。

(2)边压缩

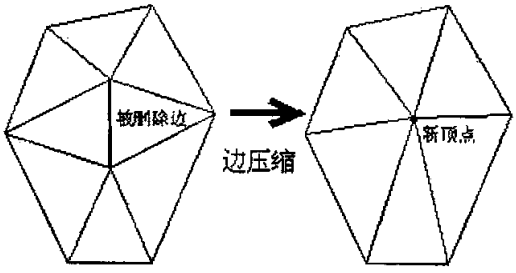
把网格上的一条边压缩为一个顶点，与该边相邻的两个三角形的退化（面积为零），而它的两个顶点融合为一个新的顶点。如图 3.1(B)。

(3)三角形移去

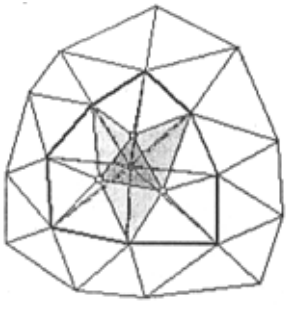
把网格上的一个面片收缩为一个顶点，该三角形本身和与其相邻的三个三角形都退化，而它的三个顶点收缩为一个新的顶点。如图 3.1(C)。



（A）顶点移去操作



（B）边压缩操作



（C）三角形移去操作

图 3.1 三种常见的网格简化算法

利用这些基本操作，我们只要确定每次操作给网格场景带来的误差计算，用这个误差代价计算方法来计算原始网格上的每一个基本元素的误差作为权值，插入到一个按权值增序排列的队列中。然后开始循环进行网格基本化简操作。在每一次循环中，我们选取队首权值最小的操作，执行之，更新变化的网格信息，并重新计算改变了的网格基本元素的误差，插入到队列中，再开始下一个循环，直到队列的最小误差达到用户设定的阈值或者用户希望的化简网格数目已经得到。

第四章：基于顶点移去准则的 LOD 模型自动生成

4.1 一些基本概念

4.1.1 边界顶点

三角形网格边界上的点称为边界顶点，其余顶点为非边界顶点。如图 4.1 所示。

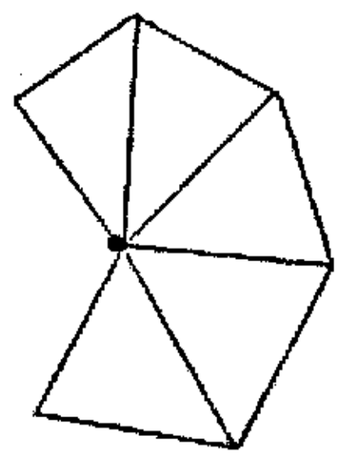


图 4.1 边界顶点

4.1.2 有序三角形环

在三角形网格中任一顶点，与该顶点有关的所有三角形按逆时针方向排列，则构成有序三角形环（对于边界顶点为半环）。

4.1.3 简单顶点

如果与顶点相关的三角形能够构成有序三角形环，并且该顶点所在的边只与两个三角形邻接，那么该顶点称为简单顶点。如图 4.2 所示。

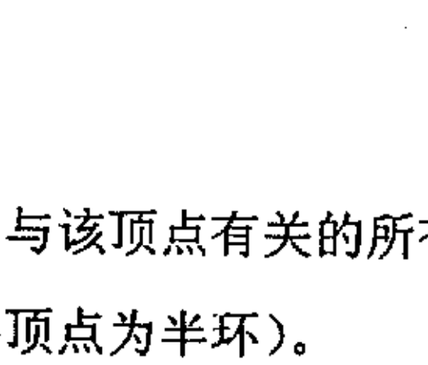


图 4.2 简单顶点

4.1.4 复杂顶点

如果与顶点相关的一个三角形不包含在有序三角形点环中，则该顶点称为复杂顶点。

如图 4.3 所示。

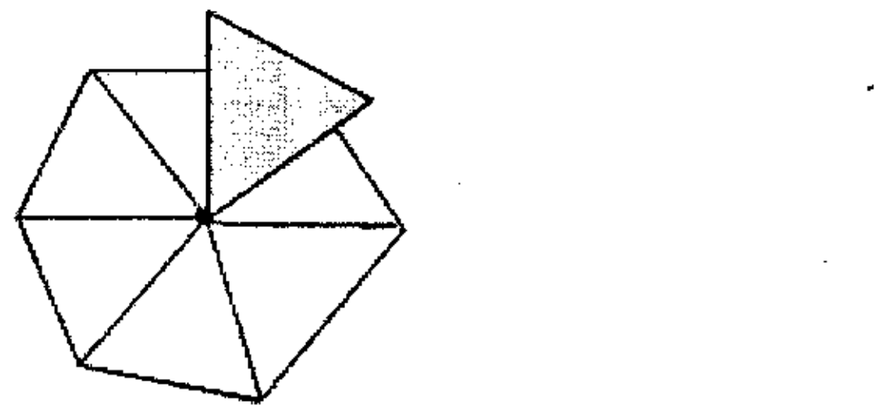


图 4.3 复杂顶点

4.1.5 特征边

两个相邻的三角形，如果其两面角大于某个指定的特征角，那么两个相邻三角形的公共边称为特征边。

4.1.6 内边顶点

对于三角形网格中的简单顶点，可以根据局部几何特性进一步分成内边顶点或角顶点，如果简单顶点被两条特征边使用，则称为内边顶点。

如图 4.4 所示。

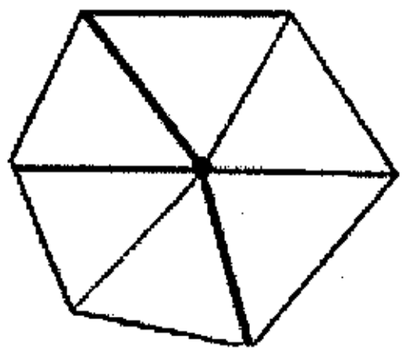


图 4.4 内边顶点

4.1.7 角顶点

如果简单顶点被一条或三角或多条特征边使用，则称为角顶点。

如图 4.5 所示。

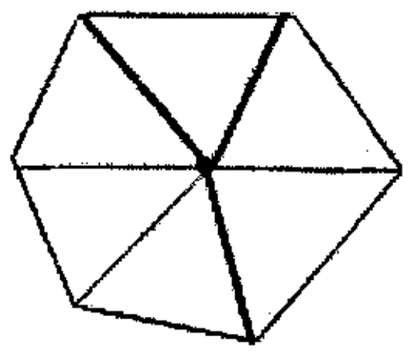


图 4.5 角顶点

4.2 顶点分类

基于顶点移去准则的三角形网格简化算法第一步就是计算确定每个顶点的局部几何和拓扑特征，决定该点是否为可删除点，如果为可删除点，则根据顶点类型选择合适的简化标准。

顶点分类的过程如下：

首先，对于三角形网格中的每一个顶点，求出与之相关的三角形。

然后，根据上面的定义，确定顶点是否为边界顶点，复杂顶点或简单顶点。

对于简单顶点，根据上面的定义，确定相邻三角形的公共边是否为特征边，然后，根据特征边的数量，确定顶点为内边顶点或角顶点或一般的简单顶点。

经过上述过程，所有的顶点都可以归为 5 类：简单顶点、复杂顶点、边界顶点、内边顶点和角顶点。

其中，复杂顶点不作为可删除点，其余的都作为可删除点候选。

4.3 顶点删除

经过前一步的操作，产生了候选顶点，以及与之相关的有序三角形环。接下来，将判断顶点能否移去，如可以移去，则移去与其相关的有序三角形环，最后将剩余区域进行三角化划分。

4.3.1 复杂顶点的判断

前面已经提到，复杂顶点不作为可删除点，将该顶点的可删除标志置为假。

4.3.2 边界顶点或内边顶点的判断

如果顶点为边界顶点或内边顶点，用如下方法决定是否删除：

顶点 v 所在的边界边或特征边的两个顶点 s、e 连成一条直线，计算顶点 v 到直线 se 的距离 d，用这个距离作为判断依据，如图 4.6 所示：

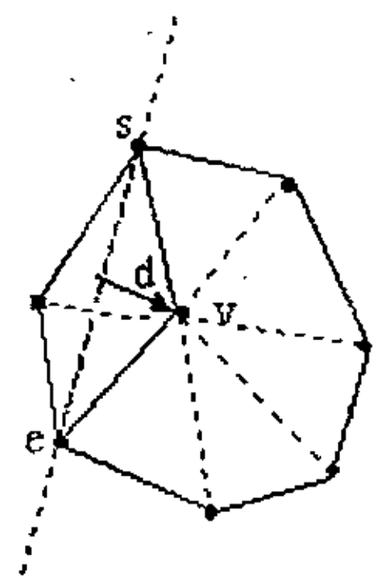


图 4.6 内边顶点判断是否删除

如果距离 d 小于指定的简化标准阈值，那么置该顶点的删除标志为真。否则，置该顶点的删除标志为假。

4.3.3 简单顶点或角顶点的判断

如果顶点为简单顶点或角顶点，用如下方法决定是否删除：

首先定义与该顶点相关的有序三角形环的平均平面。

设三角形环的每个三角形法向量为 $\overline{n_i}$，中心为 $\overline{x_i}$，面积为 A_i，那么，平均平面的法向量和中心这样计算：

$$\overline{N}=\frac{\sum \overline{n_i}A_i}{\sum A_i},\qquad \overline{n}=\frac{\overline{N}}{|\overline{N}|},\qquad \overline{x}=\frac{\sum \overline{x_i}A_i}{\sum A_i}$$

判断是否删除的依据就是顶点到这个平均平面的距离 d。

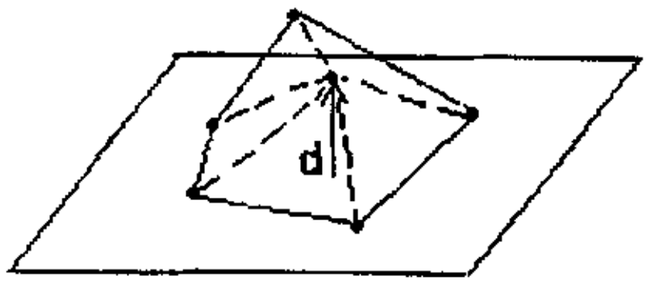


图 4.7 顶点到平均平面的距离 d

如果距离 d 小于指定的简化标准阈值，那么置该顶点的删除标志为真。

否则，置该顶点的删除标志为假。

4.4 区域的三角划分

4.4.1 有序顶点环的平面投影

经过上面的步骤，有些顶点被删除了，同时删除了与它相关的三角形环，这样，就在原有区域形成了一个空洞，为了网格的完整，必须将这个区域重新进行三角划分。

空洞周围的顶点组成一个环，称为有序顶点环。有序顶点环上的顶点都在平均平面附近，所以可以把这些点都投影到平均平面上，把三维问题转化为二维平面问题。

求出有序顶点环上每一点在平均平面上的投影。并将这些投影点保存在一个单项的有序链表里，形成一个平面上的多边形。

4.4.2 判断顶点的凹凸性

接着，我们需要判断每一个顶点的凹凸性。

在有序顶点环的投影组成的多边形上，所有的点按逆时针方向排列，平均平面的法向量$\vec{n}$符合右手法则。

在多边形上按逆时针方向取连续的 3 点，设为 P、Q、R，令$\vec{a}=PQ$，$\vec{b}=QR$，$\vec{c}=\vec{a}\times\vec{b}$。

当$\vec{c}\cdot\vec{n}<0$时，Q 为凹点。

当$\vec{c}\cdot\vec{n}>0$时，Q 为凸点。如图 4.8 所示：

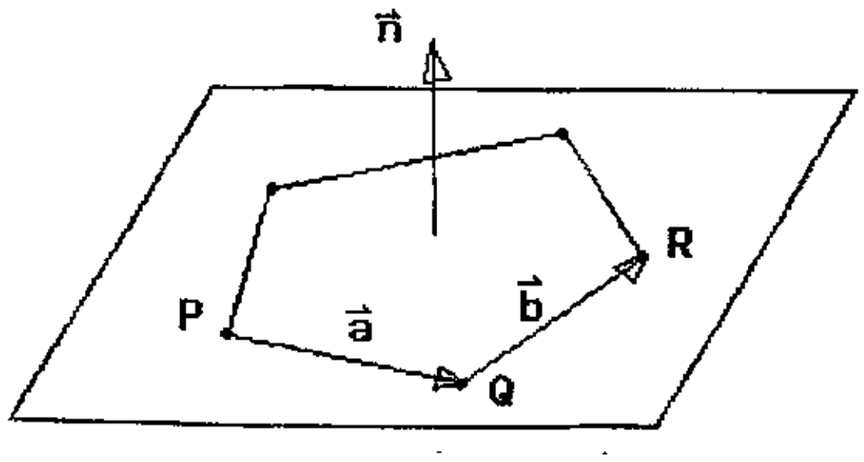


图 4.8 判断顶点 Q 的凹凸性

4.4.3 三角分割

如果存在凹点，设第一个凹点为C，取它前面的两个凸点B、A，如果三角形ABC不包含其它顶点，则在链表中删去B点，并保存该三角形到网格中。重新判断A、C两点的凹凸性。

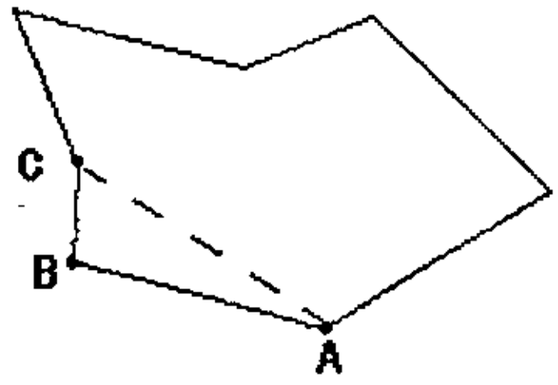


图4.9 三角分割，C为第一个凹点。

重复上述操作，直到链表中不存在凹点。

完成上述步骤后，已经不存在凹点，接下去顺序取出三个点A、B、C，保存该三角形到网格，在链表中删除B点；然后从C点开始重复操作，直到最后链表中只存在三个点。保存最后这个三角形到网格，清空链表。

三角划分完毕。

4.5 算法在程序中的实现

上述基于顶点移去的算法在思路上并不是非常复杂，然而，要实现起来也并不是很容易。

4.5.1 数据结构

要实现一个有效的三角形网格模型的自动简化程序，必须设计出一个合理的数据结构，因为在上述算法中，包含了对顶点及三角形数据的大量操作。所用的数据结构，既要满足算法实现的方便性，又要保证高效。

首先建立一个顶点表，记录顶点的相关信息。

然后是一个三角形表，每一个三角形用顶点的索引表示，根据索引就可以到顶点表中查找包含的顶点的信息。

最后是一个记录每个顶点相关的三角形的索引表，由于每个顶点相关的三角

形数量不定，因此还需有一个表记录相关的三角形的数量。

这样，一共有 4 个表，如图 4.10 所示：

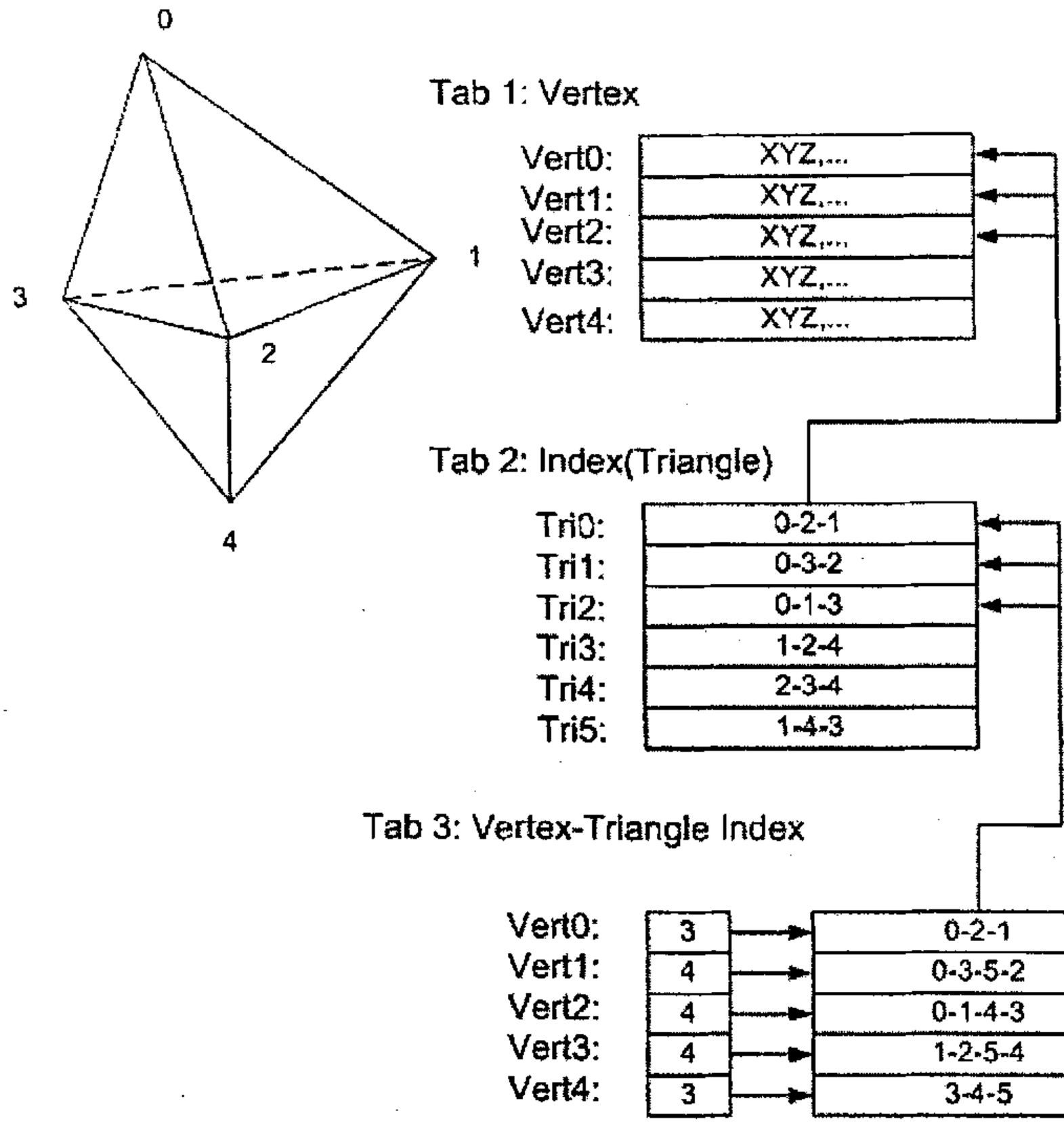


图 4.10 本章算法所用的数据结构

4.5.2 程序流程

图中只是一个简单的模型，一共有 4 个顶点，6 个三角形。

假设要删除顶点 0，从 Tab3 中查出与顶点 0 相关的三角形有 3 个，分别为 0、1、2，然后到 Tab2 中将这些三角形标志为删除。

另外开一个数组，用于记录被删除顶点 0 的有序顶点环，在删除三角形 0 的时候，将他的顶点中除了被删除顶点 0 以外的都记录到该数组中，并且按逆时针方向记录，这里依次为 2、1。然后是删除三角形 2，记录除了被删除顶点 0 以外

的顶点，由于顶点 1 已经被记录过，所以不再记录，记录顶点 3，依此类推。

最后得到的数组就是被删除顶点的相关有序顶点环。将该有序顶点环按照前面提到的方法进行三角划分，得到的新的三角形加入到 Tab2，并调整 Tab3 中每个顶点的相关三角形记录。

第五章：基于边压缩准则的 LOD 模型自动生成

5.1 边压缩操作的基本算法

边压缩操作是将一条边压缩为一个顶点，同时删去相关的三角形，思路比较简单，要解决的关键问题是如何判断一条边是否要删去。

这里使用的是一种误差代价的计算方法，也就是每一条边都做一个评估，估算删除它会产生多大的误差，在每一次叠代中，压缩误差代价最小的边，并重新估算相邻的边。

这个方法的基本操作是边的压缩，用下面这个式子来简单的描述：

$$(v_1, v_2) \rightarrow v$$

主要步骤如下：

(1)把顶点 v_1, v_2 移到 v。

(2)把所有 v_2 出现的地方都用 v_1 替换，比如所相关的三角形中。

(3)删去 v_2 和相关的三角形。

如图 5.1 所示：

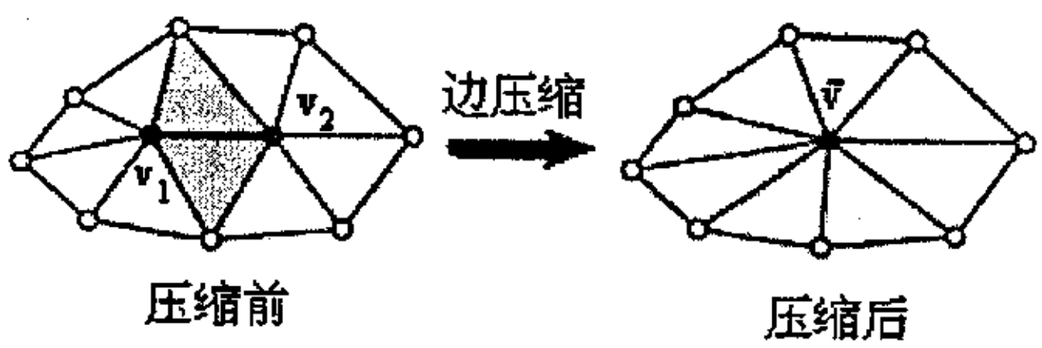


图 5.1 边压缩的基本操作

5.2 误差的评估方法

概念上来讲，我们把每一个顶点和它附近的一组平面联系起来，该顶点的误差就用这个点到这组平面中每个平面的距离的平方和来表示。

当两个顶点压缩到一个点时，目标点的相关平面组就是原始点的两组平面的和。

每一个平面都可以写成这样的方程式：

$$n^T v + d = 0$$

其中，$n = [n_x, n_y, n_z]^T$ 是平面的发向量，d 是常量。

那么，点 $v = [x, y, z]^T$ 到该平面的距离的平方为：

$$D^2 = (n^T v + d)^2 = (n^T v + d)(n^T v + d) = v^T (nn^T) v + 2dn^T v + d^2$$

这是一个二次式，令

$$Q = (A, b, c) = (nn^T, dn, d^2)$$
$$Q(v) = v^T A v + 2b^T v + c$$

因为

$$Q_1 + Q_2 = (A_1 + A_2, b_1 + b_2, c_1 + c_2)$$

所以

$$Q_1(v) + Q_2(v) = (Q_1 + Q_2)(v)$$

所以，要计算顶点到一组平面的距离的平方和，我们只要将所有的二次式相加，最后得到一个二次式。

两个顶点压缩为一个顶点后，相应的二次式也是原来两个点的二次式的和。因此，可以用 $Q(v) = Q_1(v) + Q_2(v) = (Q_1 + Q_2)(v)$ 来定义一次边压缩操作 $(v_1, v_2) \to v$ 的误差。

5.3 压缩后目标点的位置

将两个顶点压缩为一个顶点，需要考虑压缩后的目标点位于何处。这里有两个方法可以选择：

最简单的是在原来的两个顶点中选择一个作为目标顶点，也就是将其中一个顶点收缩到另一个顶点。具体压缩那一个也很简单，只要分别计算 $Q(v_1)$ 和 $Q(v_2)$，找出其中较小的那个，压缩相应的顶点就可以了。

要获得更好的效果，上面的方法是不够的，我们需要找出最优的位置。

压缩后的 $Q(v) = Q_1(v) + Q_2(v) = (Q_1 + Q_2)(v)$，我们要做的就是找到 v，使得

$Q(v)$具有最小值。

当$\partial Q/\partial x = \partial Q/\partial y = \partial Q/\partial z = 0$时，$Q(v)$取得最小值。

计算可得，取得最小值时：

$$v = -A^{-1}b$$

相应的最小值为：

$$Q(v) = -b^T A^{-1} b + c$$

有时候矩阵 A 无法求逆矩阵，这时候就用第一种方法，即在两个端点中选择一个最为目标点。

上述两种方法的取舍取决于实际应用，寻找最优位置的方法产生的结果更接近于原始模型，三角形的形状也更均匀。但是如果对于模型的近似度没有很高的要求，第一种方法也完全可以使用，虽然在效果上要差一点，却可以使程序更简单，速度更快，内存占用更小。

5.4 具体实现方法

5.4.1 误差估算

在上面提到的误差估算方法中，所谓的和顶点相关的一组平面是概念上的，并没有具体定义。

考虑到实际效果以及实现的方便性，这里沿用顶点移去准则算法中的平均平面的概念，也就是每一个顶点，有和它相关的一个有序三角形环。

设三角形环的每个三角形法向量为$\overrightarrow{n_i}$，中心为$\overrightarrow{x_i}$，面积为A_i，那么，平均平面的法向量和中心这样计算：

$$\overrightarrow{N} = \frac{\sum \overrightarrow{n_i} A_i}{\sum A_i}, \qquad \overrightarrow{n} = \frac{\overrightarrow{N}}{\left|\overrightarrow{N}\right|}, \qquad \overrightarrow{x} = \frac{\sum \overrightarrow{x_i} A_i}{\sum A_i}$$

所以这样定义边压缩的误差代价：

对于一条边来说，分别计算两个顶点到各自的平均平面的距离的平方，然后相加，所得结果作为压缩该边的误差代价。

5.4.2 目标点位置

同样基于方便性的考虑，决定目标点的位置的时候，这里采用了第一种简单的办法，也就是在分别计算两个端点的误差代价，取出误差较小的一个，将其压缩到另一个端点的位置，并同时删去退化的三角形。

5.4.3 数据结构

数据结构的设计同样要考虑效率与方便性，在上一章基于顶点移去的算法的数据结构的基础上作适当的调整

因为有很多需要是相同的，同样需要一个记录所有顶点信息顶点表；同样每个三角形的顶点信息也需要记录，和上面一样使用一个顶点的索引表来记录三角形；由于要计算每个顶点到相关有序三角形环的距离，所以仍然保留了每个顶点的相关三角形表以及记录每个顶点相关三角形数量的表。

同时，由于现在的算法主要是在对边进行操作，因此，还需要相应的表来记录边的信息。和三角形一样，使用顶点的索引表来记录边。

由于使用误差估算，使用一个数组用于记录点的压缩误差，数组的索引和顶点表保持一致。使用一个链表来记录边的压缩误差，每一个节点记录边的索引号和压缩误差值，之所以不用数组，是因为牵涉到排序。由前面的讨论可知，边的误差等于两个端点的误差的和。两个端点合并后新的顶点的误差等于原来两个顶点的误差的和。

最后的数据结构如图 5.2 所示：

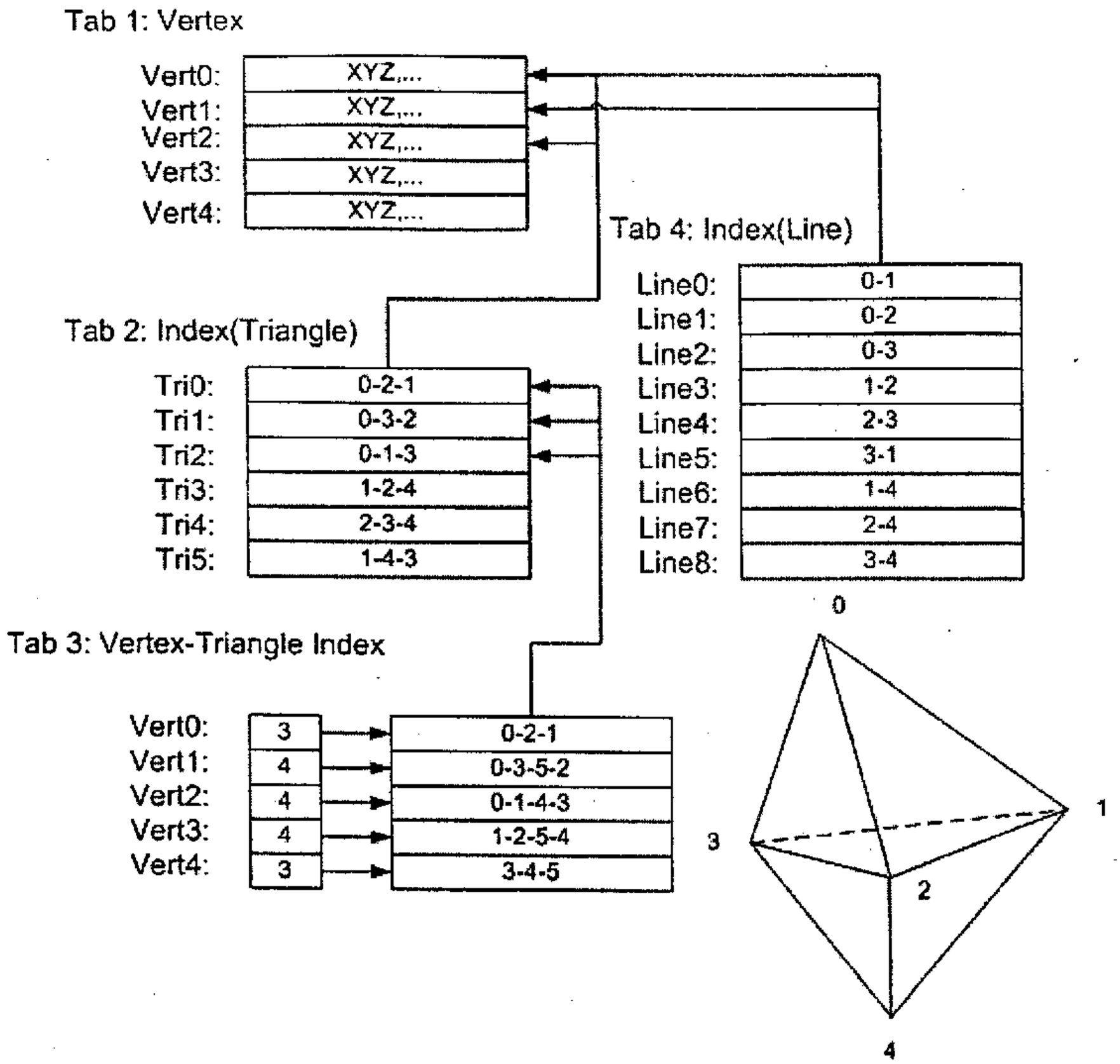


图 5.2 本章算法所用的数据结构

5.4.4 程序流程

首先遍历中所有的顶点，根据 Tab3 中的索引表，计算出每个顶点到它的有序三角形环的平均平面的距离，将这个距离的平方放入记录顶点压缩误差的数组。

然后，遍历边表，在 Tab4 中查出每条边对应的两个顶点，到记录顶点压缩误差的数组中读取这两个顶点的压缩误差，相加后插入记录边的压缩误差的链表里。每次插入的时候，都要保证链表头的压缩误差最小，依次递增。

(1)取该表的表头，读出相应的边的索引号，到 Tab4 中查出该边的两个端点的索引号。

(2)用这两个索引号，到记录顶点压缩误差的数组中读取这两个顶点的压缩误差，并作比较。假如为上图中的0、1两点，0的压缩误差小于1的压缩误差。

(3)查找Tab3，寻找所有包含顶点0的三角形，这里是0-2-1。

(4)转到Tab2，将这三个三角形的顶点0替换为顶点1。

(5)替换的时候注意，Tri0和Tri2已经包含了顶点1，就不要替换了，直接将该三角形置为已删除。

(6)删除该三角形的同时，删除该三角形中顶点 0 和第三个顶点连成的边，这里是边0-2。

(7)在Tab4中将Line0置为已删除。

(8)调整Tab3，顶点0的有序三角形环和顶点1的有序三角形环合并，并去掉Tri0和Tri2。

(9)重新计算顶点 1 的有序顶点环上每一个顶点的压缩误差，更新相应的数组。同时也计算相应改变的边的压缩误差，每次发现改变的时候，调整在记录边的压缩误差的链表中的位置，使该链表始终是递增的。

(10)重新回到第一步，直到剩下的三角形数量达到目标值。

5.5 实现的演示程序

本章的算法已经用程序实现，下面就是程序运行过程中的截图，所用的模型来自于 3ds max 5，导出为dxf格式后用程序读入。

第一个模型是一驾螺旋桨的飞机。

图5.3是初始状态，整个模型有4133个顶点，8124个三角形，12212条边。

图5.4和图5.5是分别简化到1000个顶点和300个顶点后的截图。

对比这三副图我们可以看出，随着三角形数量的减少，渲染的速度也得到了提高，而且相当的明显。

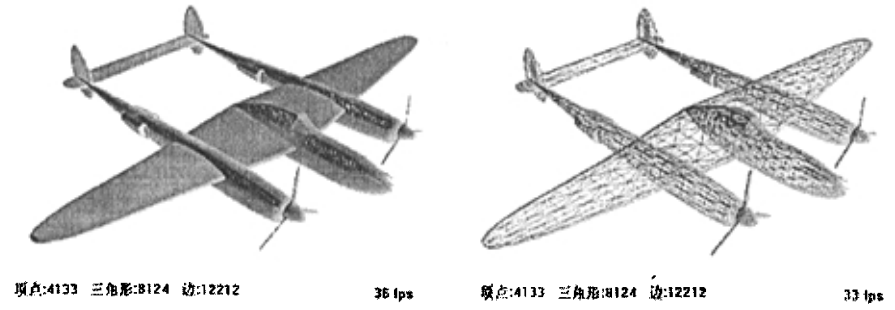


图 5.3 初始状态的模型，左边为加了材质的渲染图，右边为线框图

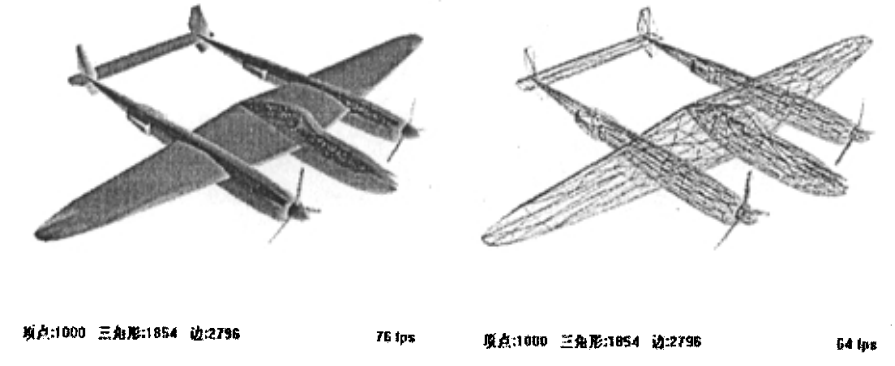


图 5.4 简化到 1000 个顶点后的模型，左边为加了材质的渲染图，右边为线框图

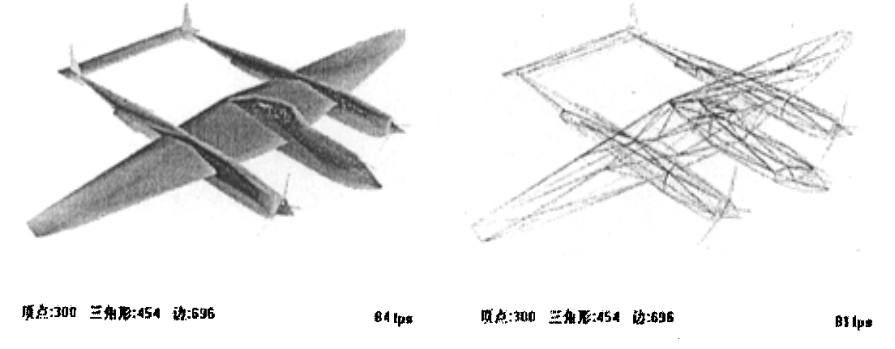


图 5.5 简化到 300 个顶点后的模型，左边为加了材质的渲染图，右边为线框图

第二个模型是一个卡通人物。

图 5.6 是初始状态，整个模型有 4538 个顶点，9048 个三角形，13572 条边。图 5.7 和图 5.8 是分别简化到 1000 个顶点和 300 个顶点后的截图。

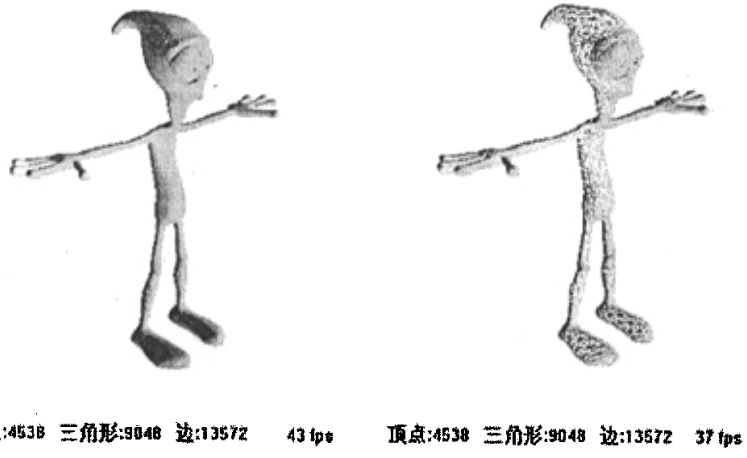


图 5.6 初始状态的模型，左边为加了材质的渲染图，右边为线框图

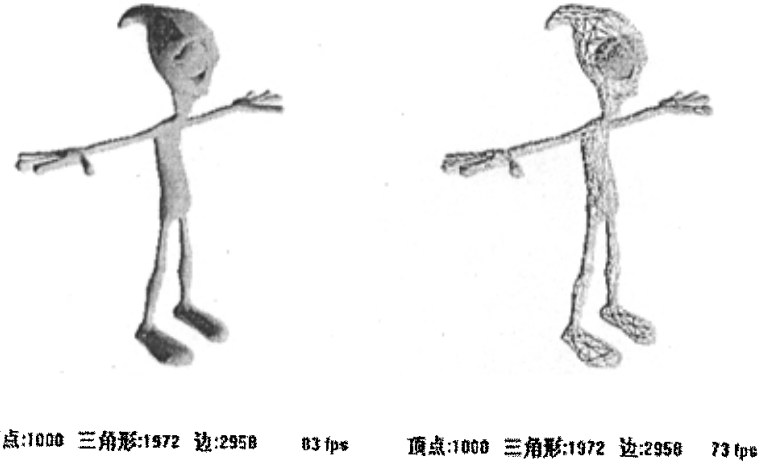


图 5.7 简化到 1000 个顶点后的模型，左边为加了材质的渲染图，右边为线框图

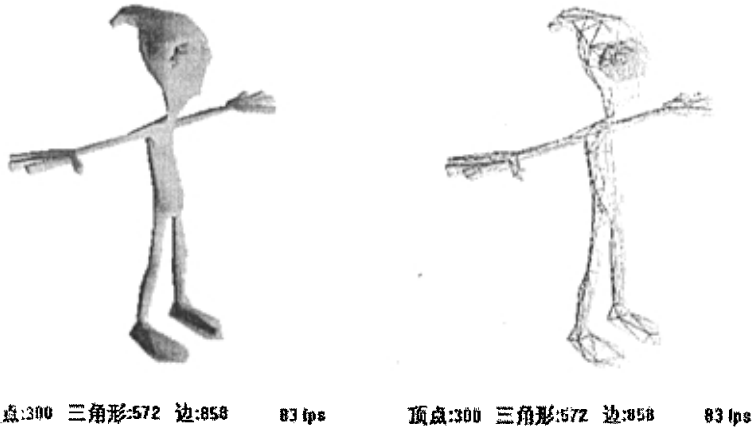


图 5.8 简化到 300 个顶点后的模型，左边为加了材质的渲染图，右边为线框图

第六章：基于三角形移去准则的 LOD 模型自动生成

6.1 算法描述

与前两个算法不同的是，这次要移去的是一个三角形，三角形移去后，由此产生的空洞和第一种算法一样，重新进行三角划分。

简化的步骤主要如下：

(1)计算可移去的三角形。

(2)移去三角型。

(3)进行局部三角化。

由此可见，这里要做的第一步就是确定三角型可移去判断方法。

联想到在基于顶点移去的算法中，使用顶点到平均平面的距离作为判断准则，我们可以建立类似的方法，作为三角形可移去的判断准则。

6.2 一些基本概念

6.2.1 三角形环和三角形板

在三角形网格中，对于任意一个三角形，所有与该三角形至少共享一个顶点的三角形（包括该三角形本身）的集合，称为该三角形的三角形板。除去该三角形，成为该三角形的三角形环。

6.2.2 相邻三角形

对于三角形网格中的任意两个三角形，如果这两个三角形有一条公共边，那么称这两个三角形为相邻的。

6.2.3 连通与非连通

在一个三角形环中，对于任意两个该三角形环中的三角形 T_{l_1}、T_{l_m}，存在三

角形T_{i_2}，...，$T_{i_{m-1}}$也属于该三角形环，满足条件：T_{i_i}和$T_{i_{i+1}}$是相邻的，其中 i=1，2...，m-1，那么称该三角形环是连通的，否则，是非连通的。

6.2.4 平均平面

设三角形板中每个三角形的法向量为$\overrightarrow{n_k}$，中心为$\overrightarrow{x_k}$，面积为A_k，那么，由(2.4)式定义的法向量n和中心x所构造的平面定义为三角板的平均平面。

$$\overrightarrow{N}=\frac{\sum\overrightarrow{n_k}A_k}{\sum A_k},\qquad \bar{n}=\frac{\overrightarrow{N}}{|\overrightarrow{N}|},\qquad \bar{x}=\frac{\sum\overrightarrow{x_k}A_k}{\sum A_k}$$

6.2.5 边界多边形

由三角形板中所有边界边组成的多边形称为三角形板的边界多边形。边界多边形的顶点集合称为三角形板的边界顶点集。

6.3 网格简化算法

前面已经讲到，基于三角形移去准则的网格简化算法主要有以下三个步骤：

(1)计算可移去的三角形。

(2)移去三角型。

(3)进行局部三角化。

具体过程如下：

对于三角形网格中的每个三角形：

(1)求与它相关的三角形板。

(2)求出三角形板的平均平面，计算三角形板中每个顶点到平均平面的距离，并求出这些距离的平方和 D。

(3)如果 D 小于指定的简化标准，那么，该三角形可删除。

(4)对于可以删除的三角形，算出其对应的边界多边形。

(5)对该边界多边形进行局部三角化。

重复以上操作，如果三角形数量没有达到简化目的，则降低简化标准，重复

以上步骤，直到三角形数量达到简化目标。

6.4 边界多边形的局部三角化

在基于顶点移去的网格简化算法中，已经给出了一种多边形局部三角化的方法，这里，对它稍作调整。

(1)对边界多边形中每一个顶点，求出它在平均平面上的正交投影，并用循环单向链表保存顶点，顺序连接投影点得到一平面多边形。

(2)计算出多边形顶点链表中每一结点的凸凹性。多边形上，所有的点按逆时针方向排列，平均平面的法向量 $\bar{n}$ 符合右手法则。

在多边形上按逆时针方向取连续的 3 点，设为 P、Q、R，令 $\bar{a} = PQ$，$\bar{b} = QR$，$\bar{c} = \bar{a} \times \bar{b}$。当 $\bar{c} \cdot \bar{n} < 0$ 时，Q 为凹点。当 $\bar{c} \cdot \bar{n} > 0$ 时，Q 为凸点。如图 6.1 所示：

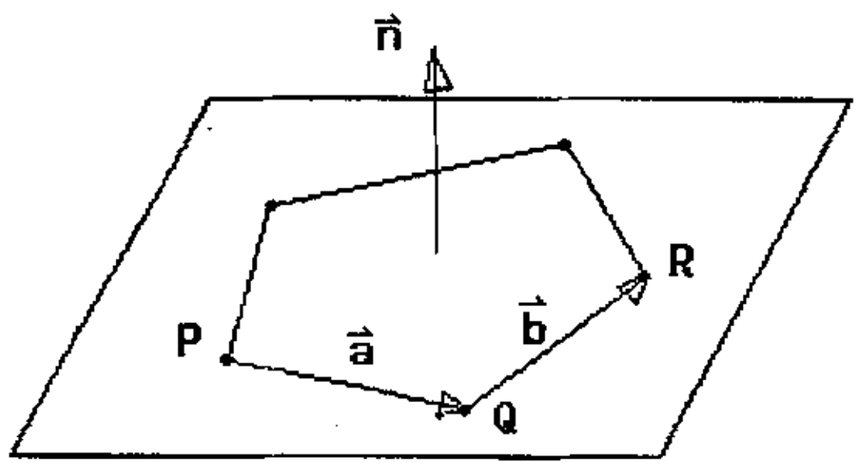


图 6.1 判断 Q 点的凹凸性

(3)在循环链表中，按照顺序取三个点 A、B、C，要求满足 B 为凸点，并且三角形 ABC 不包含多边形上的其他顶点。计算该三角形的最小内角。遍历链表，计算所有满足条件的三角形的最小内角。取出其中最大的，保存相应的三角形，并从链表中删除 Q 点。

(4)如果链表中还存在三个以上的节点，则重新计算每个点的凹凸性，并重复 C 部分的操作。

(5)链表中的最后三个节点，放入三角形网格中。

6.5 算法的实现

基于三角形移去准则的网格简化算法的基本思路和基于顶点移去的网格简化算法基本一致，都是在判断是否能移去后，移去相应的几何图元，然后对留下

的空洞进行三角划分。

6.5.1 数据结构

对于数据结构的设计，同样要考虑效率与实现的方便性。

首先，仍然是一个顶点表。

然后，记录三角形的链表，三角形的顶点用顶点的索引。

最后，记录每个顶点相关的三角形，使用两个链表，第一个链表记录每个顶点相关的三角形的数量，第二个表记录的是顶点相关的三角形的索引。

因此，最后的数据结构和基于顶点移去准则的网格简化算法所用的数据结构基本一致。如图 6.2 所示：

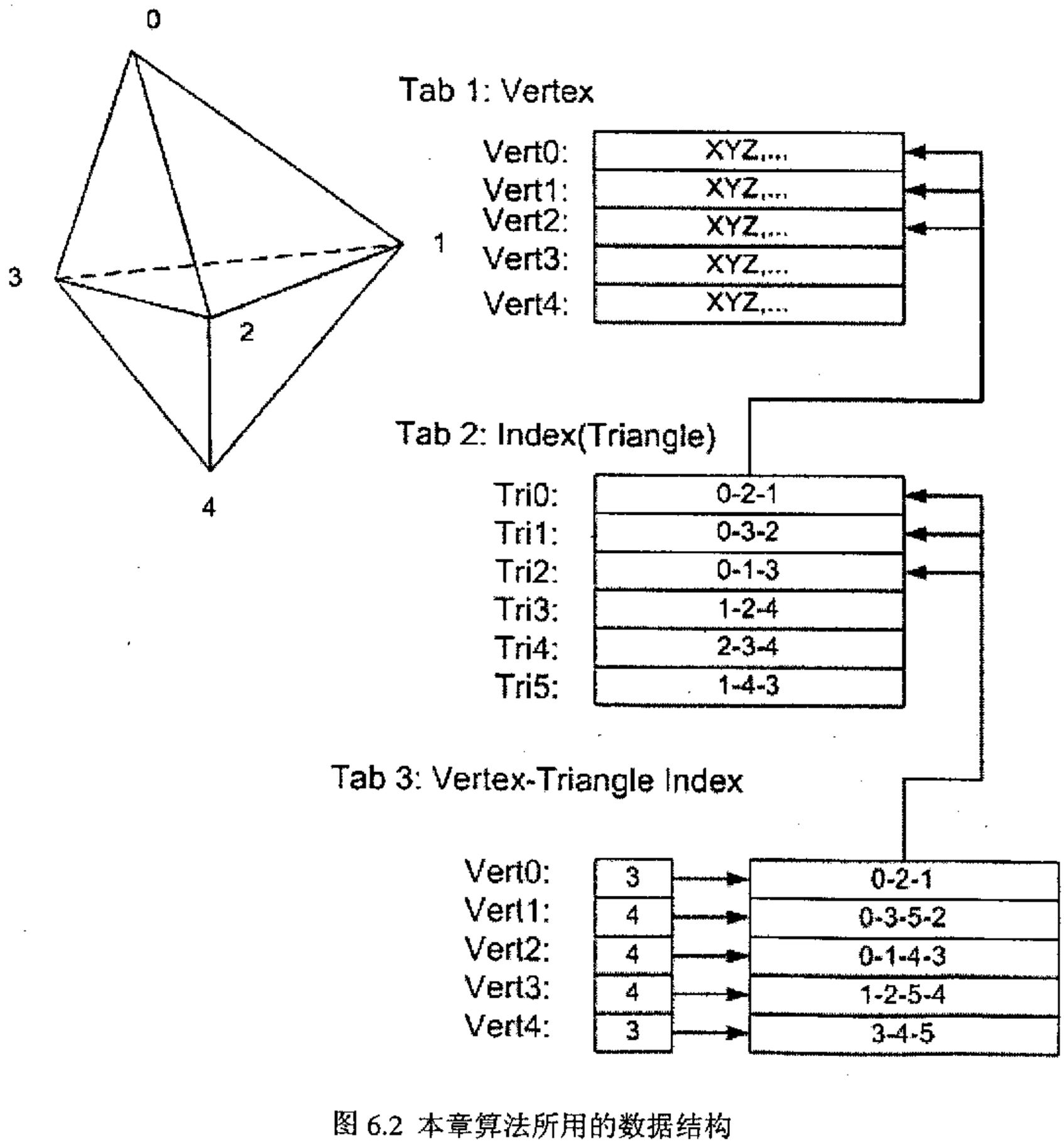


图 6.2 本章算法所用的数据结构

6.5.2 程序流程

遍历 Tab2 中的三角形：

(1)求出三角形相关的三角形板。

根据 Tab2 读出当前三角形的三个顶点，转 Tab3；

对于每一个顶点，在 Tab3 中读出与它相关的三角形，将三组数据合并，并去掉重复部分，得到与三角形相关的三角形板。

同时，遍历三角形板中的每个三角形，根据 Tab2，得到该三角形板包含的所有顶点。

(2)计算得到的三角形板的平均平面。

根据上面给出的式子，计算出该三角形板的平均平面。

(3)计算三角形板中每个顶点到平均平面的距离的平方和 D。

在步骤 A 中，已经得到了三角形板中包含的所有顶点，所以，这里的计算也很方便了。

(4)将 d 和设定的简化标准进行比较。

若 d 小于设定的简化标准，那么当前三角形可删除。

(5)计算边界多边形。

边界多边形用一个单向循环链表来记录。

边界多边形的求法和步骤 A 中求三角形板的所有顶点类似，只是把原三角形的三个顶点去掉，剩余顶点按逆时针方向排列。

(6)边界多边形的局部三角化

三角化的算法上面已经介绍了。

三角化过程中，对于 Tab2，删除三角形板上所有的三角形，保存新增的三角形；对于 Tab3，删除当前三角形的三个顶点，对于边界多边形上的每一个顶点，重新调整相关的三角形。

继续三角形链表中的下一个节点，直到结束。

结束。

第七章：网格简化算法中的一些相关问题

7.1 误差代价计算方法

7.1.1 简述

误差代价计算方法，在基于边压缩的网格简化算法中已经提到过。这里在系统的讨论一下。

无论何种简化方法，包括顶点移去、边压缩和三角形移去，每一次操作，都会对初始模型带来一定的误差，也就是出现失真。但是，对于不同的操作对象，带来的误差也是不同的。

简化的目的，除了要降低多边形的数量，另一个重要的目标就是尽量减少由于简化而带来的失真。

因此，在简化操作过程中，尽量选择移去后造成的误差较小的对象进行操作，可以较好的达到这个目的。同时，不会对计算量产生过大的影响。

采用这种方法，首要问题就是误差代价如何衡量。

7.1.2 误差代价的衡量值

对于上面提到的三种算法，分别这样处理：

(1)顶点移去操作：

对于每个顶点，定义了它的相关的三角形环，根据这个三角形环，可以求出它的平均平面；计算该顶点到这个平均平面的距离的平方，作为该顶点的误差代价衡量值。

(2)边压缩操作：

首先和上面的方法一样，求出每个顶点到它的平均平面的距离；对于每条边，求它的两个顶点的误差代价的平方和，作为该边的误差代价衡量值。

(3)三角形移去操作：

对于每个三角形，可以定义它的三角形板和三角形环，接着，就可以得到三

角形板的平均平面；计算该三角形的三个顶点到平均平面的距离，并求出三个距离的平方和，作为该三角形的误差代价衡量值。

7.1.3 使用方法

确定了误差代价，就可以将原始网格上的每一个基本元素插入到一个按照递增排列的链表里。然后开始循环进行网格基本化简操作

在每次的循环里，我们取第一个误差代价最小的元素进行操作，执行后，更新网格信息，并重新计算变化了的网格基本元素的误差代价，插入到链表里，并保证链表仍然是递增排列，再开始下一轮循环，直到满足结束条件为止。

结束的条件，可以有两种：

(1)最小误差达到设定的阈值

(2)三角形数量达到设定的数量

7.2 连续显示问题

通过前面介绍的方法，我们可以建立起不同细节层次的模型，这些模型具有一定层次的细节，并且，它们的误差是逐步递增的，这样的模型可以很好的用于层次细节的显示中。

在实际显示中，我们根据实际需要，选择不同细节层次的模型，于是，一个现实的问题就出现了，在不同细节层次的模型切换的时候，会产生跳跃感，也就是模型发生了突变，这种现象，极大地影响了虚拟现实场景的真实感。

为了避免这种现象的发生，必须想办法让不同细节层次的模型之间实现平滑过渡，也就是细节层次模型连续显示的问题。

要建立起相邻细节层次的三角形网格模型的平滑过渡，基本的方法就是通过插值对应网格基本元素的位置来实现光滑过渡，问题的关键是如何得到两个相邻层次的多边形网格模型的基本元素之间的对应关系。

针对基于边压缩准则的 LOD 模型自动生成算法，作如下分析：

由于边压缩的基本操作是$(v_1,v_2)\rightarrow v$，所以，只要建立起v_1,v_2和v的对应关系，然后，利用简单的线性插值，就可以达到较好的效果。

假设压缩前的时刻为t_0，压缩后的时刻为t_1，那么在压缩过程中的某时刻$t(t_0 < t < t_1)$，v_1,v_2的位置分别可以这样计算：

$$v_1' = v_1 + \frac{t - t_0}{t_1 - t_0}(v - v_1)$$

$$v_2' = v_1 + \frac{t - t_0}{t_1 - t_0}(v - v_2)$$

第八章 基于 LOD 的地形渲染技术

室内场景是有限的，观察着的活动是有约束的，因此，描述场景所用的多边形数量也是有限的。室外场景和室内场景不同，观察者当然不希望受到约束，比如观察者坐飞机浏览，理论上他可以坐着飞机朝任何一个方向飞去，并且可以飞无限远。

这样，就对室外场景的描述提出了很高的要求，场景的无限广大与计算机处理多边形能力有限之间的矛盾凸现出来。室外场景实时性渲染的主要对象是地形的渲染，必须有一种技术可以降低地形渲染的开销。

8.1 LOD 地形渲染技术

显然，LOD 是一个极好的思路。虽然地形无限大，但是观察者能够看到的总是有限的，而且更重要的是，观察者也并不是对于地形上的每一个局部都有同样大的兴趣，我们完全可以把他可能产生兴趣的那部分画得很详细，而不太可能注意的部分画得很简单。

LOD 技术根据一定的规则简化地形的细节，根据需要来选择不同细节程度的地形表达方式。如离观察者较近的选择较高的细节程度，离观察者较远的选择较低的细节程度。

LOD 算法对场景的处理比较复杂，但是它让我们可以足够自由的去控制我们的场景渲染，更加方便的使用显卡的硬件加速功能。而且可以很容易的在场景中组合其他的物体。

地形的渲染通常分为动态和静态两种。

静态地形的细节通常在事前就计算好了。静态地形可以是均匀的，也可以是不均匀的。不均匀细节的静态地形也有许多的优点：如平原的地貌可以使用较低的细节，而起伏频繁的地方使用较高的细节等级。

动态的地形是视点相关的。随着视点的移动，地形网格将被更新。相对于静态地形来说这是一种更为先进的算法。这种方式建立起来的场景更加符合人的视觉特性：即看到的细节是变化的。动态地形网格的建立和更新要耗费额外的时间，

我们所有要做的就是在精度和速度之间选择一个合适的平衡点。

地形作为一种特殊的几何物体，我们在运用 LOD 法则的时候有一些特殊的技巧。因为地形通常是一个规则的矩形网格。其简化模式可以有两种：规则的简化和非规则的简化，规则的简化通常是对这个矩形网格采用自顶向下(Up-to-Down)、分而治之的策略，典型的有四叉树和二叉树，它们从场景的最低细节层度开始，按需要不断的提高细节。非规则的简化通常是采用自底向上（Down-to-Up）的方法来处理的。它的实现则通常比较少。

8.2 基本思想

首先，简单起见，规定渲染的地形必须是正方形的区域，大小是 $(2^n+1)\times(2^n+1)$，同时采样间隔是均匀的。

我们采用四叉树的概念来描述一个多分辨率地形。每一个节点代表了一个正方形。从整个完整的地形出发，递归的把地形不断的分隔成四个相等的区域，分割的深度越大，得到的分辨率就越高。即分割深度每提高一层，采样密度提高一倍。如图 8.1 所示：

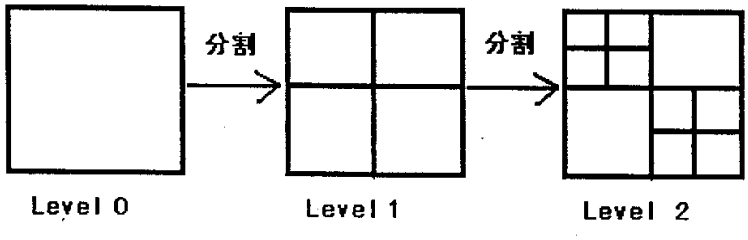


图 8.1 地形的四叉树表示

四叉树的每个节点，要记录的信息包括：一个中心点、四个边点、四个角点，一共 9 个点。

如图 8.2 所示：

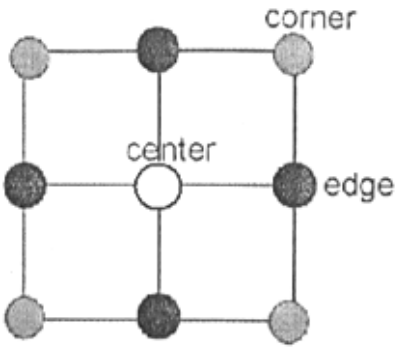


图 8.2 每个节点记录的顶点信息

8.3 数据的组织

就像二叉树可以用顺序结构（一维数组）来存储一样，四叉树也可以用顺序结构来存储，不同的是，现在用的是二维数组。

这里要用到两个二维数组：一个用来记录全分辨率的地形数据，包括每个节点的九个顶点的信息，可以通过索引直接读取；另一个数组和它大小相同，用来记录四叉树节点的状态，如果一个节点要继续分割，就把相应的位置标记为 1，否则就是 0。没有访问到的地方不标记，数值不确定。

$$\begin{pmatrix} ? & ? & ? & ? & ? & ? & ? & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? & 0 & ? \\ ? & ? & 0 & ? & ? & ? & 1 & ? & ? \\ ? & ? & ? & ? & ? & 0 & ? & 0 & ? \\ ? & ? & ? & ? & 1 & ? & ? & ? & ? \\ ? & 0 & ? & 0 & ? & 0 & ? & 1 & ? \\ ? & ? & 1 & ? & ? & ? & 1 & ? & ? \\ ? & 0 & ? & 0 & ? & 0 & ? & 1 & ? \\ ? & ? & ? & ? & ? & ? & ? & ? & ? \end{pmatrix}$$

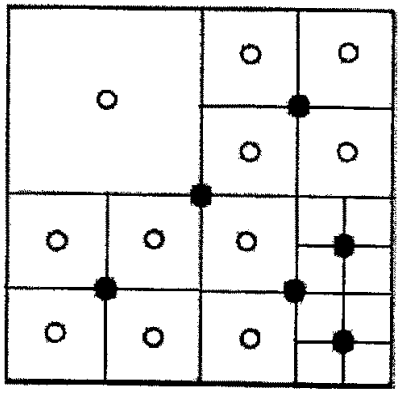


图 8.3 记录节点状态的二维数组和相应的地形网格

图 8.3 中，左边是标志数组，右边是它代表的地形。黑点表示该节点要继续分割，白点表示该节点不需要继续分割。

8.4 判断节点是否分割的条件

我们需要建立一个节点评价系统，用来决定一个节点是要继续分割，还是不用再分割可以送入渲染系统了。

决定一个节点的细节层次，需要考虑的因素主要有两点。首先是节点离视点的远近；然后就是地形本身的起伏状况。离视点越近，起伏越大，就需要显示得越具体。

(1)视点相关性

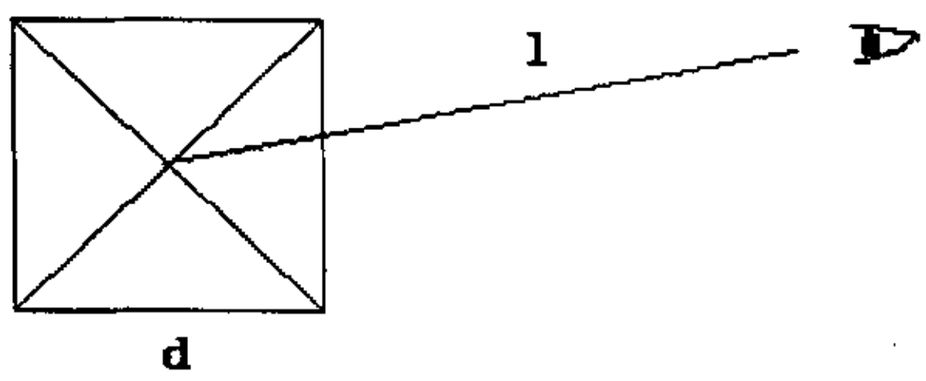


图 8.4 节点的视点相关性

离视点越近，细节需要越多，同时，我们还要考虑节点本身的大小，节点越大，在视野中占的比例也越大，所需细节就越多。

综合这两个因素，可以将下面这个式子作为节点是否分割的一个判断条件：

$$\frac{l}{d} < C \quad \text{(C 是一个设定值，可以调节，以控制最后的细节程度)}$$

其中 l 是节点的中心位置到视点的距离，d 为节点的边长，当满足这个公式时，该节点就要继续分割。C 越大，地形的细节就越多，反之越少。

(2)地形粗糙度

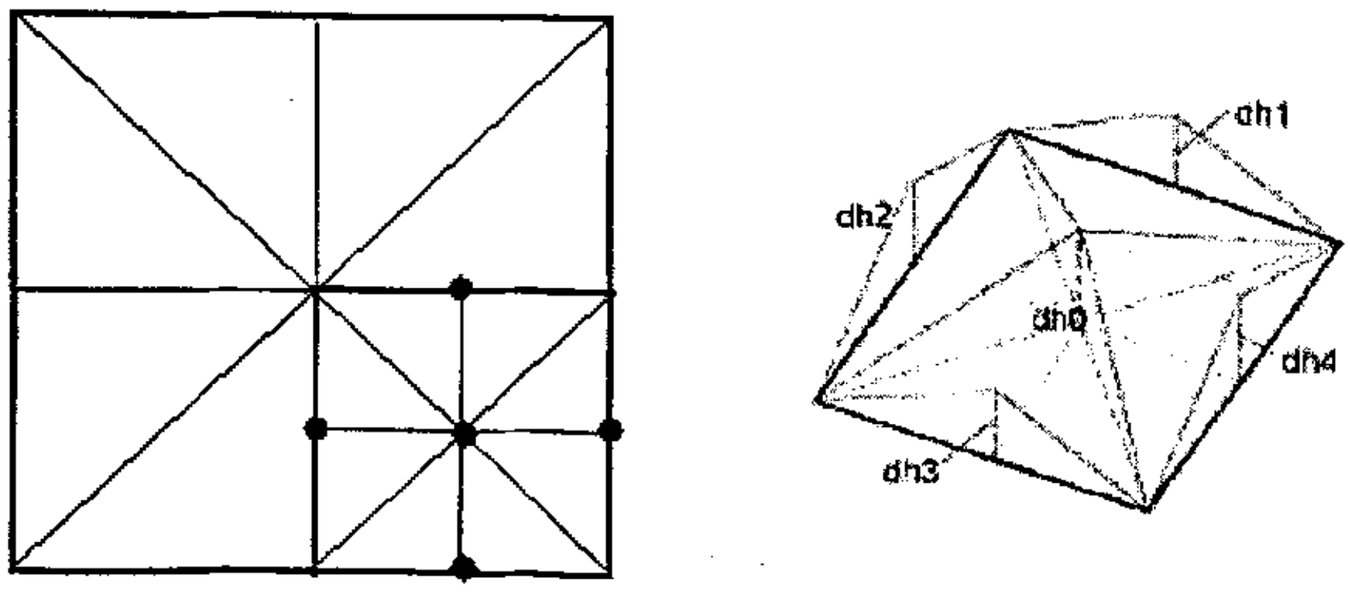


图 8.5 节点分割前后的变化，用来衡量地形的粗糙度

从图 8.5 中可以看出，一个节点分割后，子节点中的黑点高度在划分前后改变了，包括子节点的中心点和四个边点。设为 dh0、dh1.....dh4，这些值的大小，反映了子节点区域的地形粗糙程度，计算方法为所在边的两个角点的平均值与边点的差。

同时，我们还要考虑这个节点的四个子节点的粗糙程度 dh5~dh8（分辨率达到最大的时候不用考虑）。

取这九个值中的最大值除以节点的大小作为这个节点粗糙程度的衡量值，即

$$r = \max(dh0, dh1, ..., dh8)/d$$

由于这些值是不变的，因此可以事先计算好，同样保存在一个二维数组里。

这样，就可以得出这样的式子作为是否分割的判断条件：

$$\frac{l}{r} < C_2$$

综合上面的两个式子，可以得到最终的节点评价公式：

$$f = \frac{l}{d \times r \times C \times C_2} < 1$$

当满足 $f < 1$ 时，节点需要分割。

8.5 网格的渲染方式

前面已经提到，一个节点由一个中心点和若干个围绕中心点的点组成，于是可以用三角形扇的方式来绘制。

在绘制的时候，如果没有事先处理，在不同分辨率的节点拼接处，会产生裂缝，消除这个裂缝的办法是去掉一条边，如图 8.6 所示：

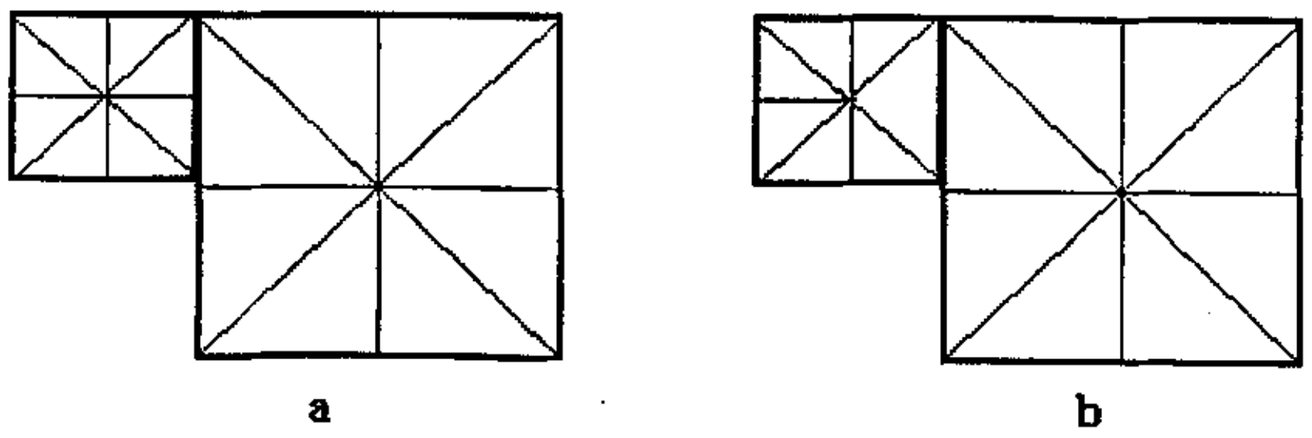


图 8.6 裂缝的避免办法

这样就提出了一个要求，就是拼接处的两个层次的差距不能大于 1。

在渲染的时候，中心点和角点没有问题，遇到边点的时候，必须做一个检察，检查相邻的节点是否被激活，如果没有激活，说明此处存在一个裂缝，要把该边点跳过。

8.6 网格的生成

根据上面提出的要求，下面开始设计网格的生成步骤。

网格的生成采用广度优先的原则，也就是一次性生成所有属于同一细节层次的节点。

这里用到两个队列，第一个队列就是正在处理的层次的所有的节点，另一个队列则保存着处理当前层节点后生成的所有下一个层次的节点。当前层分割一个节点，生成了四个下一层的节点后，就放入到这个队列里。

当前层所有节点都处理完的时候，清空第一个队列，并和第二个队列交换角色。如此不断反复。

我们从第一层根节点开始，也就是最大的一个节点，如果地形没有做分割，就是整个地区了。好，把它放入第一个队列。这一层只有这一个节点，放入完毕。

开始遍历第一个队列。

取出队列的第一个节点。

节点的处理也很简单。首先判断它是否要分割。

判断的条件前面讨论过，$f = \frac{l}{d \times r \times C \times C_2} < 1$，当满足 $f < 1$ 时，节点需要分割，否则不分割。式中，d 可以很方便的计算出，r 是事先算好的，只要从数组中读取，C、C2 是指定的参数。所谓的分割，就是把当前节点的四个子节点放入第二个队列中，如果当前节点不需要分割，则直接送入渲染引擎绘制出来。

第一个队列遍历结束后，清空。和第二个队列简单的交换。也就是进入下一层了，继续做上面提到的那些事情。

直到遍历完一个队列交换后，发现当时的第一个队列没有节点了，结束。

不要忘了上面提到的那个要求，也就是拼接处的两个层次的差距不能大于 1。解决这个问题的办法是，在分割一个节点的时候做一个判断，也就是在同一层中，

和它邻接的四个节点是否已经被激活，如果已经激活，那么当前节点可以继续分割；否则，不能分割。

第九章 总结

本论文主要讨论了多细节层次模型自动生成和显示技术的基本原理、主要方法和作者在攻读硕士学位期间所做的一些研究和开发工作，主要包括以前几个方面：

(1) 基于顶点移去准则的 LOD 模型自动生成算法及其具体实现方法。

(2) 基于边压缩准则的 LOD 模型自动生成算法及其具体实现方法。

(3) 基于三角形移去准则的 LOD 模型自动生成算法及其具体实现算法。

(4) 针对地形的特殊性，采用四叉树实现了地形的 LOD 自动简化方法。

本论文的研究成果，最直接的，可以用来对各类模型进行自动的简化。很多虚拟建筑漫游系统中要用到的模型素材，往往过于复杂，有些在整个场景中也许并不怎么起眼，却又大量的三角形，占用了很多系统的资源，那么就可以直接用我们的成果对其自动的简化，而且保持较好的效果，和原始形状基本一致。

虚拟现实技术当前是计算机领域的一大热门，其发展速度可谓一日千里，无论是软件还是硬件，都使得虚拟现实技术越来越成熟，产生的效果越来越逼真，而且，虚拟现实正越来越走进人们的日常生活。

但是，目前的虚拟现实技术离真正的给人身临其境的沉浸感还有一段距离，至少，我们还是可以很容易的区分哪个是真实的世界，哪个是计算机虚拟的场景，这虽然看上去很正常，但是却是每一个研究虚拟现实的人所极力想消除的，虚拟现实的最高境界，就是让人无法区分虚拟与现实。

虚拟现实技术是许许多多技术的综合，我们所做的工作，就像大海中的一滴水，希望可以为虚拟现实技术的发展作出一定的贡献。

我们所做的课题，也仅仅是整个 LOD 模型研究课题中的一部分，这个课题还有许多有待继续研究的问题。

关键性的问题就是一般网格的 LOD 实时连续生成技术。在本文中，虽然给出了多边形网格的自动生成算法，但是除了地形网格由于其特殊性，可以做到实时连续生成以外，其他的一般网格，只能在事先生成不同的细节层次模型，然后在渲染的时候根据需要选用不同层次的模型。

这样做的弊端就是不能做到视点相关，因为在一个自由度较高的系统中，事先是不可能知道用户在浏览的时候视点会如何移动的，就算事先对浏览者的视线作出某些限制，由于可能的情况依然很多，将导致事先要生成大量的模型数据，这是违反我们研究 LOD 的初衷的。

相信以后一定会有更多更好的方法，达到更好的效果。

参考文献

[1] 孙家广、杨长贵编著《计算机图形学(新版)》清华大学出版社 1995

[2] 曾建超、俞志和《虚拟现实的技术及其应用》清华大学出版社 1996

[3] 杨宝民 朱一宁《分布式虚拟现实技术及其应用》科学出版社 2000

[4] 曾建超 俞志和《虚拟现实的技术及其应用》清华大学出版社 1996

[5] 周昌伟《三维场景建模与虚拟漫游系统的研究和实践》华中科技大学硕士学位论文

[6] 张茂军《虚拟现实系统》科学出版社 2001

[7] 潘李亮《基于 LOD 的大规模真实感室外场景实时渲染技术的初步研究》 2003

[8] 马小虎《虚拟现实中多细节层次模型的研究》浙江大学博士学位论文 1997

[9] 刘学慧《虚拟现实中三维复杂几何形体的层次细节模型的研究》 1998

[10] 马小虎、周昆、潘志庚、石教英《交互式可视化中的一种多细节层次模型自动生成方法》上海交大学报 1996. 10

[11] 王宏武、董士海《一个与视点相关的动态多分辨率地形模型》

[12] 刘学慧、吴恩华《基于图象空间判据的地表模型加速绘制技术》

[13] 周晓云、刘慎权《基于特征角准则的多面体模型简化方法》计算机学报 1996，18(增刊)

[14] 潘志庚、马小虎、石教英《虚拟环境中多细节层次模型自动生成算法》软件学报，1996,7(9)

[15] 彭博《游戏编程指南》2002

[16] Sergei Savchenko 编著、刘长松 程连冀 译《3D 图形编程指南 》1998

[17] 刘学慧、吴恩华《虚拟现实的图形生成技术》中国图象图形学报 1997, 2(4)

[18] 刘文炜《基于虚拟环境技术的实时显示算法的研究与实现》博士论文，中国科学院计算技术研究所, 1996

[19] 丁永祥、夏巨谌等《任意多边形的 Delaunay 三角剖分》计算机学报 1994, 17(4)

[20] 陈向平、应道宁《统一于 NIP 的多边形三角剖分算法》计算机学报 1989, 12(3)

[21] 闵卫东、唐泽圣《二维任意域内点集的 Delaunay 三角划分生成算法》计算机学报 1995, 18(5)

[22] Alan Watt “3D Computer Graphics”, 2nd Edition Addison-Wesley 1993

[23] M.E.Algorri and F.Schmitt "Mesh Simplification" In Proceedings of EUROGRAPH'96 1996

[24] M.J.DeHaemer and M.J. Zyda "Simplification of objects rendered by polygonal approximations" Computers and Graphics 1991, 15(2)

[25] H.hoppe, T.DeRose, T.Duchamp, J.McDonald,and W.Stuetzle "Mesh optimization" In Proceedings of SIGGRAPH'93 1993

[26] A.D.Kalvin, R.H.Taylor "Superfaces: Polygonal Mesh Simplification with Bounded Error" IEEE Computer Graphics and Applications 1996

[27] G.Turk "Re-tiling polygonal surfaces" Computer Graphics 1992, 26(2)

[28] H.Hoppe "Progress Meshes" In Proceedings of SIGGRAPH'96 1996

[29] M. Isenburg, P. Lindstrom, S. Gumhold, and J. Snoeyink "Large Mesh Simplification using Processing Sequences" August 12, 2003

[30] James H Clark "Hierarchical geometric models for visible surface algorithms" Communication of ACM 1976,19(10)

[31] Greg Turk. Re-tiling polygonal surfaces "Computer Graphics" Proceedings of SIGGRAPH'92 1992,26(2)

[32] D.Berman, J.Bartell, and D.Salesin "Multiresolution painting and compositing" In Proceedings of SIGGRAPH'94 1994

[33] A.Certain, J.Popovic, T.DeRose, T.Duchamp, D.Salesin, and W. Stuetzle "Interactive multiresolution surface viewing" In proceedings of SIGGRAPH'96 1996

[34] Michael Garland Paul S. Heckbert "Simplifying Surfaces with Color and Texture using Quadric Error Metrics"

[35] Stefan R¨ottger Wolfgang Heidrich, Philipp Slusallek, Hans-Peter Seidel "Real-Time Generation of Continuous Levels of Detail for Height Fields"

[36] Microsoft Co. "DirectX Documentation for C++" 2002

[37] Discreet -Autodesk，Inc. "3ds max 5 User Reference" 2002

[38] M.F.Cohen "Interactive spacetime control for animation" In Proceedings of SIGGRAPH'92 1992

[39] D.Doo and M.Sabin "Behaviour of recursive division surfaces near extraordinary points" Computer Aided Design 1978, 10(6)

[40] Florian , Patrick RoBbach "Managing the complexity of digital terrain models" Computers and Graphics 1994,18(6)

[41] Greg Turk "Re-tiling polygonal surface" Computer Graphics (Proceedings of SIGGRAPH'92) 1992,26(2)

致　谢

今年是公元 2004 年，自 2001 年 9 月开始攻读硕士学位以来，在同济的这段时光成了人生最美好的回忆。

首先要衷心感谢导师谢步瀛教授。还记得当初第一次见谢老师的时候，平实可亲的样子一生难忘。一起度过的岁月里，老师的悉心教导，真切关怀，让人倍感温暖。谢老师是我们尊敬的师长，更是我们知心的朋友。如今分别在即，愿谢老师身体健康，笑口常开。

还要感谢刘政老师、董冰老师、陈星铭老师和王婉老师，你们的平易近人和严谨治学是我走上工作岗位后的榜样。

远在他乡，难忘两鬓苍苍的父母，二十多年了，父母的养育之恩如何能说尽。父母的爱，世界上没有任何一种语言可以表达，也没有任何一种方式可以回报。只能说一声：爸妈，辛苦了！

论文的完成，得到了同门师兄弟的大力相助，难以忘记和你们一起学习生活的日子，这里要谢过：博士研究生王祎、博士研究生江锡虎、硕士研究生张岩、硕士研究生欧泉、硕士研究生周云岗、还有已经毕业的硕士赵亚鑫、硕士李强、金玉芬。

感谢我的舍友和同学，两年多一起风风雨雨，同甘共苦，他们是：周争考、周峰、张涛、党兴勇、陈愈……

最后把此文献给贺俊，你的支持、鼓励和关心，是我莫大的动力。

朱萧峰

2004 年 3 月