数字媒体技术专业计算机图形学推荐教材

Computer Graphics
Mesh Processing
Theory and Implementation

C#版

计算机图形学

三维模型处理算法初步
理论与实现

赵辉　王晓玲　著

海洋出版社

数字媒体技术专业计算机图形学推荐教材

Computer Graphics
Mesh Processing
Theory and Implementation

C#版

计算机图形学

三维模型处理算法初步
理论与实现

赵辉　王晓玲　著

海洋出版社
2014年·北京

内 容 简 介

“数字媒体技术专业计算机图形学”系列丛书按照计算机图形学教学及从业需求分为5册。本书为其中的三维模型处理算法初步分册。

主要内容： 全书共14章，分别讲述了三维模型的数据结构；三维模型的生成；对偶模型；点、边、面的添加删除等三维模型的基本操作；各种三维模型元素的查找；补洞切割算法；三维模型简化、细分算法；三维模型的几何、拓扑信息计算；三维模型上的莫斯理论应用；三维模型的分段、文件加载的算法。

本书特色： 1. 为新形势下计算机图形学教学及从业需要打造。2. 精练三维模型处理算法的理论知识，便于入门。3. 操作步骤与实现算法对应讲解，层次清晰。4. 选用C#语言编写代码，上手快捷，易于变通。5. 三维模型动画、渲染算法等课程的基础，可以和OpenGL编程并行学习。

读者对象：

- 中高等院校数字媒体技术及软件工程相关专业学生。
- 计算机图形、三维动画、虚拟现实领域从业人士及爱好者。

图书在版编目(CIP)数据

计算机图形学：三维模型处理算法初步：理论与实现（C#版）/赵辉，王晓玲著. —北京：海洋出版社，2014.10

ISBN 978-7-5027-8973-2

Ⅰ. ①计… Ⅱ. ①赵… ②王… Ⅲ. ①三维—计算机图形学 Ⅳ. ①TP391.41

中国版本图书馆CIP数据核字（2014）第233374号

总 策 划：张曌嫘
责任编辑：张曌嫘 张鹤凌
责任校对：肖新民
责任印制：赵麟苏
排 版：申 彪

出版发行：海洋出版社

地 址：北京市海淀区大慧寺路8号（707房间）100081
经 销：新华书店
技术支持：（010）62100050 graphicsresearch@qq.com

发 行 部：（010）62174379（传真）（010）62132549
（010）68038093（邮购）（010）62100077
网 址：www.oceanpress.com.cn
承 印：北京旺都印务有限公司印刷
版 次：2014年10月第1版
2014年10月第1次印刷
开 本：889mm×1194mm 1/16
印 张：13.25（全彩印刷）
字 数：480千字
印 数：1～3000册
定 价：69.00元

前言 Foreword

2008年以来，全国各高等院校纷纷在各自软件工程学科专业的基础上开设了数字媒体技术专业。数字媒体技术专业和计算机科学专业的区别是，前者主要是着重在学习二维图像和三维图形相关的算法和应用开发，而后者还需要学习其他计算机科学相关的知识。由于开设和建立时间短，各学校的数字媒体技术专业的教学工作都还处在摸索阶段，也没有形成统一、成熟的教材体系。根据在数字媒体技术专业多年的教学实践经验，我们总结出本专业要以计算机三维图形学的理论和算法为基础，以三维应用开发为导向进行建设。

我们整合多年一线教学经验与反馈以及当前的三维图形学研究成果，编写了本套丛书。本套丛书涵盖了三维图形学算法的三大方面：模型、动画和渲染。内容根据数字媒体技术专业的教学特点分散到计算机图形学系列的5本图书中：《三维模型处理算法初步：理论与实现（C#版）》、《三维模型处理算法进阶：理论与实现（C#版）》、《三维模型参数化：理论与实现（C#版）》、《OpenGL三维渲染（C#版）》和《GLSL渲染编程》。通过本系列专业基础教材，再加上已有的成熟的计算机基础编程教材以及三维软件使用的教材，就可以完整地覆盖数字媒体技术专业的所有课程。

书里的代码采用的是C#编程语言。C#编程语言是一种结合了C++和Java优点的编程语言。C#语言相对于其他编程语言来说比较容易学习和掌握，但是本套丛书里介绍的原理和算法不仅限于C#语言。读者可以通过示例中的代码，用自己熟悉的编程语言来进行编程。

除了三维模型、动画、渲染的理论和算法实现之外，本套丛书也包含了三维渲染OpenGL和GLSL渲染编程的讲解，通过对渲染实例的讲解介绍如何调用OpenGL和GLSL以得到实时的渲染效果。相比于之前介绍OpenGL编程的图书，本书更加注重通过对实例的讲解，降低学习的难度。GLSL渲染编程需要在OpenGL编程之后学习。OpenGL编程和其他几门课程可以并行进行。三维模型处理算法进阶、三维模型参数化算法的学习需要在掌握三维模型算法初步的知识及操作基础后方可进行。

《计算机图形学——三维模型处理算法初步：理论与实现（C#版）》介绍了三维相关算法的基础和入门知识。本书分为14章，分别讲述了三维模型的数据结构；三维模型的生成；对偶模型；点、边、面的添加删除等三维模型的基本操作；各种三维模

型元素的查找；补洞切割算法；三维模型简化、细分算法；三维模型的几何、拓扑信息计算；三维模型上的莫斯理论应用：三维模型的分段、文件加载的算法。 本书是三维模型动画、渲染算法等课程的基础。学习本书的前修课程是C#、C++或者Java等编程语言。如果学过“数据结构和算法”课程再学习本书效果会更好。本书不需要提前掌握OpenGL渲染，可以和OpenGL编程并行学习。

本书的完成要感谢杨路维、董楠、李子豪、孟旬、孟晗、吴仝、刘乐园、江海博、周安安、邬春鹏、王琪及魏来等同学在算法实现中的辛勤劳动。

本书不仅仅可以作为数字媒体技术专业的专业基础课，还可以作为计算机学科和软件工程学科“数据结构和算法”，“计算机图形学”等课程的教材和参考书。

赵 辉

2014年9月于朝阳区平乐园

北京工业大学软件学院

目录 Contents

1 三维模型数据结构

1.1 三维模型简介.....001
1.2 三维模型的操作.....003
1.3 基于数组的数据结构.....004
1. 以面为中心.....004
2. 共享顶点.....004
3. 基于面连接.....005
4. 基于边连接.....005
5. 邻接矩阵.....005
6. 角表.....006
1.4 半边数据结构.....006
1.5 半边数据结构代码.....008
1. 模型网格类.....009
2. 顶点类.....009
3. 半边类.....012
4. 边类.....012
5. 面类.....014
6. 属性类.....017

2 三维模型的生成

2.1 生成正多边形/圆.....018
2.2 生成锥体.....019
2.3 生成柱体.....020
2.4 生成球面.....020
2.5 生成平面网格.....022
2.6 克隆.....023

3 对偶模型

3.1 对偶模型构造.....025
3.2 对偶模型算法.....026

4 三维模型的基本操作

4.1 添加一个面.....031
4.2 删除一个面.....038
4.3 删除一条边.....039
4.4 删除一个顶点.....041
4.5 分割一个点.....044
4.6 合并一条边.....047
4.7 切换一条边.....052
4.8 其他基本操作.....054
1. 生成噪声.....054
2. 包围框顶点位置.....054
3. 缩放模型.....055
4. 移动模型到中心.....056
5. 把选中的点分组.....056
6. 重新设置序号.....057
7. 改变面的方向.....058

5 点边面查找

5.1 查找一个顶点的邻域.....062
1. 查找一个顶点的一层邻域顶点.....062

2. 查找一个顶点的一层邻域边 062
3. 查找一个顶点的一层邻域面 062
5.2 查找一条边的邻域.....063
1. 查找一条边的一层邻域顶点 063
2. 查找一条边的一层邻域边 064
3. 查找一条边的一层邻域面 066
5.3 查找一个面的邻域.....066
1. 查找一个面的一层邻域顶点 066
2. 查找一个面的一层邻域边 067
3. 查找一个面的一层邻域面 067
5.4 查找一组点、边、面的一层邻域.....068
1. 查找一组顶点的一层邻域半边 068
2. 查找一组顶点的一层邻域顶点 071
3. 查找一组顶点的一层邻域边 071
4. 查找一组顶点的一层邻域面 071
5. 查找一组边的一层邻域顶点 072
6. 查找一组边的一层邻域边 073
7. 查找一组边的一层邻域面 073
8. 查找一组面的一层邻域顶点 074
9. 查找一组面的一层邻域边 074
10. 查找一组面的一层邻域面 074
5.5 查找一组点、边、面的边界点、边、面.....075
1. 查找一组顶点的边界半边 075
2. 查找一组顶点的边界顶点 077
3. 查找一组顶点的边界面 078
4. 查找一组边的边界顶点 078
5. 查找一组边的边界边 078
6. 查找一组边的边界面 079
7. 查找一组面的边界顶点 079
8. 查找一组面的边界边 079
9. 查找一组面的边界面 080
5.6 查找边界.....080
5.7 查找边的分割区域.....081

6 补洞切割

6.1 补洞.....085
6.2 沿平面切割模型.....087
6.3 按三角形面切割模型.....090
6.4 沿选择的边切开模型.....092
6.5 分割模型组件.....095
6.6 分割钝角.....097

7 三维模型简化

7.1 顶点聚类 100
7.2 二次误差度量算法.....104
1. 数学原理 104
2. 算法步骤 106
3. 简化效果 110
7.3 元素删除简化.....111
1. 最小边长合并简化 111
2. 最小面积简化 113
3. 最小高斯曲率简化 113
7.4 简化误差度量 114
7.5 简化的记录.....116

8 三维模型细分

8.1 Loop细分算法 119
8.2 Modified Butterfly细分算法.....122
8.3 Sqrt3细分算法.....126
8.4 细分算法效果比较.....128

9 5-6-7模型

9.1 顶点的价.....130
9.2 3价到4价.....131
9.3 4价到5价.....133
9.4 面分裂.....138
9.5 分割.....141
9.6 简化网格.....143

10 三维模型几何

10.1 面积.....148
1. 三角形面的面积.....148
2. 奥若诺伊（Voronoi）面积.....150
3. 混合面积.....151
10.2 体积.....152
10.3 面的法向.....153
10.4 顶点的法向.....153
1. 相同权重法向.....153
2. 面积权重法向.....154
3. 顶角权重法向.....154
4. 内接球法向.....154
5. 法向对比.....155
10.5 双面夹角.....156
10.6 三角形的角度.....157
10.7 曲率.....157
1. 曲线曲率.....157
2. 曲面曲率.....158
3. 主曲率.....158
4. 高斯曲率.....159
5. 平均曲率.....159
10.8 曲率计算.....159
1. 平均曲率.....160
2. 高斯曲率.....160
3. 主曲率方法一.....162
4. 主曲率方法二.....162
5. 主曲率方法三.....162
6. 效果图.....162

11 三维模型拓扑

11.1 拓扑.....164
11.2 组件数.....164
11.3 亏格.....165
11.4 欧拉公式.....166
1. 欧拉示性数.....166
2. 效果图.....166
3. 欧拉定理.....168
11.5 高斯-博内定理.....169

12 莫斯理论

12.1 莫斯函数.....170
12.2 关键点.....172
12.3 莫斯定理.....175
12.4 莫斯复形.....176
12.5 调和莫斯函数.....179
1. 定义.....179
2. 代码.....179
3. 效果图.....181
12.6 莫斯函数应用.....182

1. 剪开模型 182
2. 莫斯简化 183
3. 模型分段 184

13 三维模型分段算法

13.1 概述 185
1. 优化问题 185
2. 约束条件 186
3. 模型分段属性 186
13.2 区域增长算法 187
1. 以点为中心分段 187
2. 以三角形面为中心分段 190
13.3 K-Means算法 194

14 三维模型文件加载

14.1 OBJ格式文件 196
1. 简述 196
2. 特点 196
3. 结构 196
4. 示例 197
5. 算法步骤 197
14.2 OFF格式文件 199
1. 简述 199
2. 结构 200
3. 示例 200

参考文献

1 三维模型数据结构

三维模型是计算机图形学的基础研究对象。三维模型数据结构就是计算机里表示三维模型的方式。三维模型有不同的数据结构，但是这些数据结构的速度和效率都不一样，某些数据结构在特定的三维模型操作上速度比较快。本章主要介绍以面为中心的数据结构、共享顶点的数据结构、基于面连接的数据结构、邻接矩阵数据结构、角表数据结构，最后着重讲述了半边数据结构，因为对于大多数在三维模型上的操作来说，半边数据结构相对于其他数据结构效率更高、速度更快。

1.1 三维模型简介

在虚拟现实、影视特效、三维游戏、工业设计等应用中都离不开三维模型。三维模型是各种三维应用的核心。在计算机科学领域里，计算机图形学就是研究三维相关算法的科学，包括模型处理、动画、渲染三大部分。计算机图形学的研究需要微积分、线性代数、数值积分、微分几何等数学知识。本书主要讲述三维模型处理的基本算法和实现，这是三维渲染、动画等算法的基础。

三维模型可以有多种表示形式，例如用贝塞尔曲面等参数化曲面来表示，也用点来表示，如图1-1所示。

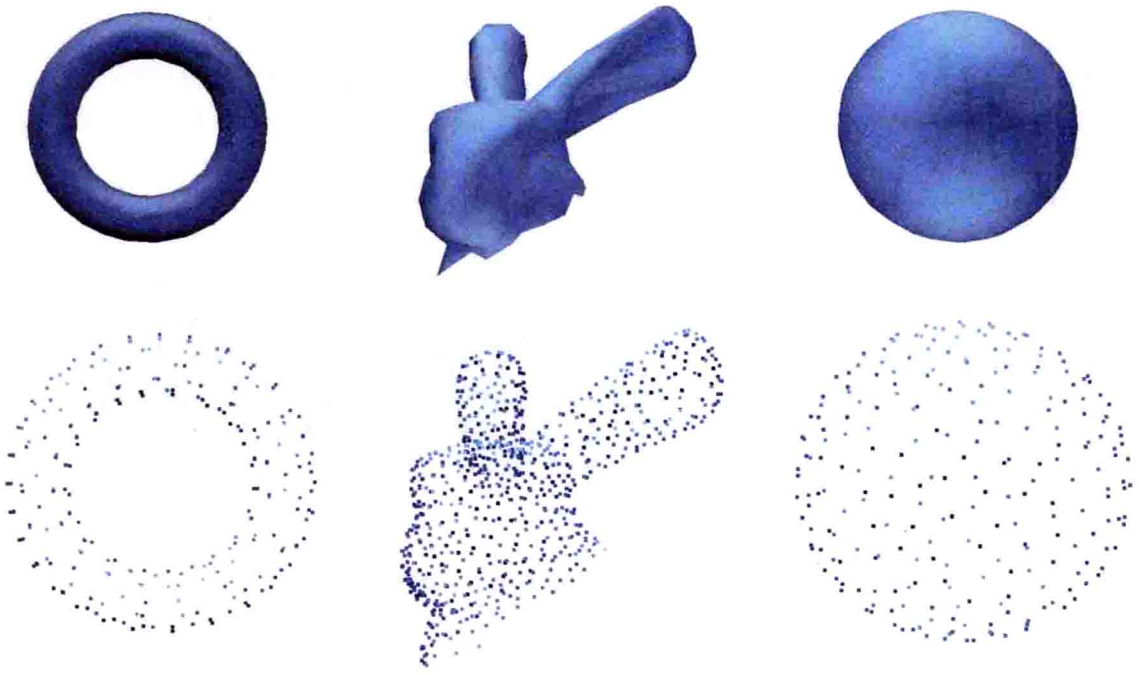


图1-1 三维模型的表示形式（一）

也可以用网格来表示，例如用四边形网格，如图1-2所示。

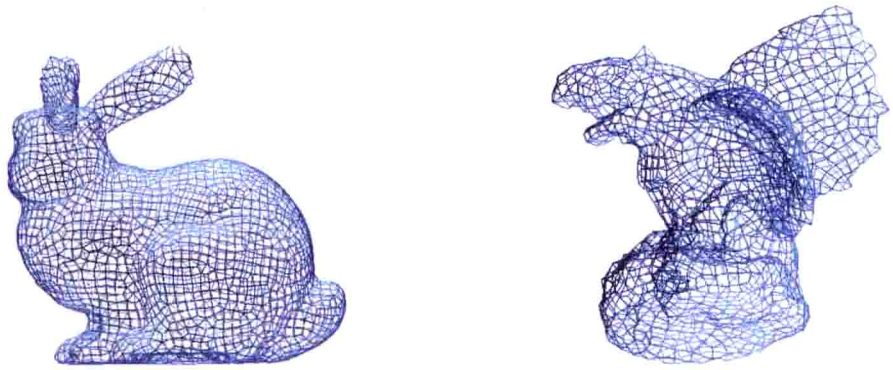


图1-2　三维模型的表示形式（二）

但是大部分模型都是用三角形网格来表示的，如图1-3所示。

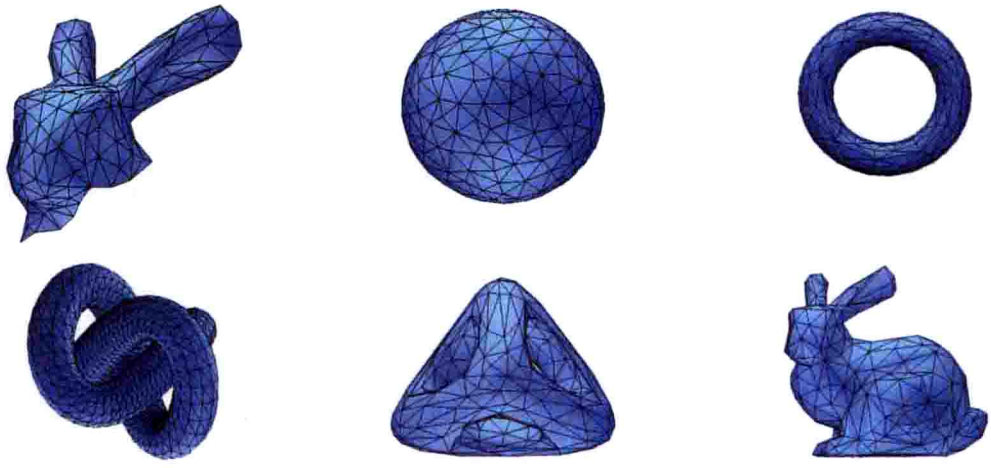
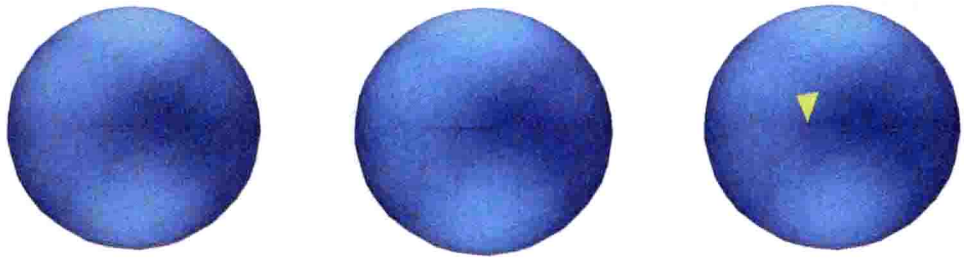


图1-3　三维模型的表示形式（三）

这几种表示方法可以互相转换，例如把参数化表示方法转化为三角形或四边形网格表示方法；三角形网格转换为四边形网格；点表示方法转化为网格表示方法等。本书介绍的三维模型处理算法都是作用于三角形网格上的。

三维模型的三角形网格表示包含三个部分：点、线、面，如图1-4所示。

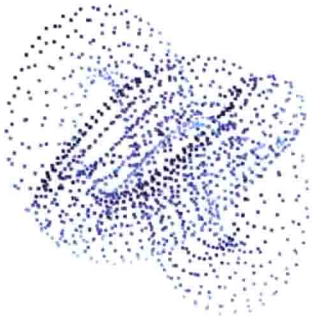


(a) 点　　(b) 线　　(c) 面

图1-4　三维模型的三角形网格表示

点、线、面构成三个集合，数学上一个模型M用（V，E，F）来表示。其中V表示顶点的集合，E表示边的集合，F表示面的集合。三维模型可以分为流形（Manifold）和非流形（Non-Manifold），本书主要介绍流形的三维模型处理算法。有了三维模型之后，就可以把三维模型用各种方式显示出来。图1-5中呈现的是三种不同的显示方法。

(a) 平滑线



(b) 点

(c) 平滑

图1-5 三维模型显示方法

1.2 三维模型的操作

三维网格的信息分为两个部分：几何（Geometry）信息和拓扑（Topology）信息。几何信息由顶点的位置组成，拓扑信息包含点、线、面互相之间的邻接关系。如果一个三维模型的拓扑信息不变，改变几何信息，这个三维模型的形状就会发生变化。

三维模型的点、线、面的数据可以存储在硬盘上，然后在使用时候加载到内存。在内存里，需要一个高效的数据结构来对这些数据进行存储和操作。一般来说，三维网格模型的信息除了最基本的顶点坐标几何信息和连接关系的拓扑信息之外，还有在点、线、面上的各种属性（Attribute）信息，例如顶点的法向（Normal）、颜色（Color）、纹理坐标（Texture Coordinate）等。在设计三维模型数据结构的时候，除了考虑这些信息在内存的存储以外，还需要考虑作用于三维模型上的各种操作（表1-1）。不同的数据结构，操作的时间复杂度和效率是不一样的。一个好的数据结构应该可以有效地存储信息，并且可以使各种操作都能够快速地进行。在计算机科学中，算法都是作用在数据结构上的，因此三维模型的算法是作用于三维模型在内存的数据结构上的。根据算法的不同，可以采用不同的数据结构。

表1-1 三维模型上常见的操作

操作种类	操作前效果图示	操作后效果图示	操作种类	操作前效果图示	操作后效果图示
顶点的删除			合并两个顶点		
边的删除			一个顶点分为两个顶点		
面的删除			模型简化		
补洞			细化		

三维模型数据结构分为两类：基于数组的数据结构和半边数据结构。三维模型数据结构的设计需要考虑前面介绍的各种操作。通常采用半边数据结构来表示三维模型，因为这种数据结构可以有效地处理以上各种操作，假如只考虑其中的某种或者某几种操作的话，也可以选择其他简单的数据结构。

1.3 基于数组的数据结构

基于数组的数据结构是把所有顶点及面的信息都放到数组里面，这样的数据结构优点是占用内存少，缺点是无法得到局部邻接关系的信息，这样对某些三维模型上需要局部邻接关系信息的操作就需要对整个数组进行遍历，因此效率很低。

1. 以面为中心

以面为中心（Face Set）的数据结构按照三角形面的顺序存储每个三角形三个顶点的坐标值。它的优点是简单，缺点是没有保存点、线、面互相之间的连接关系，而且冗余太多。因为每个顶点通常会出现在好几个面里面，在如图1-6所示的数据结构中，每个面都要把此顶点重复保存一次，从而占用了大量内存。

三角形		
顶点坐标	顶点坐标	顶点坐标
[10 20 30]	[40 5 20]	[10 4 3]
…		

图1-6 以面为中心的数据结构

2. 共享顶点

共享顶点（Shared Vertex）的数据结构分为两个部分（图1-7）：一部分是顶点的坐标数组；另一部分是三角形面的数组，每个三角形面里面只存储相对应的顶点数组的序号，而不是具体的顶点坐标。这种数据结构可以表示点之间的连接关系（Connectivity），但是没有局部的邻接关系（Neighborhood），例如从一个给定的面指向另一个相邻的面。

三角形		
顶点索引	顶点索引	顶点索引
2	1	3
…		

顶点
顶点坐标
[40 5 20]
[10 20 30]
[10 4 3]
…

图1-7 共享顶点的数据结构

共享顶点数据结构的效率和速度分析可以参看图1-8所示的实例。在这个实例中，如果要找到面F1的顶点操作，这个操作的时间复杂度是0（1），而找到顶点V3的相邻顶点的操作就需要遍历所有数据。对于判断两点是否相邻的操作也需要遍历所有数据。因而共享顶点的数据结构在很多操作上效率较低。

图1-9是图1-8所示三维模型的共享顶点数据结构。

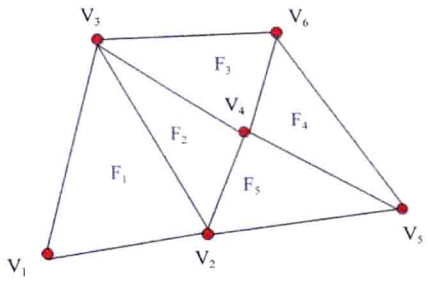


图1-8 简单三维模型

面	三角形		
F_1	2	1	3
…			

顶点	
V_1	[40 5 20]
V_2	[10 20 30]
V_3	[10 4 3]
…	

图1-9　共享顶点的数据结构示例

3. 基于面连接

基于面连接（Face Based Connectivity）的数据结构的组成部分是顶点和面。每个顶点都保存顶点的位置和相邻一个面的索引信息，每个面保存三个顶点的位置和三个邻接面的索引信息。这种数据结构里面没有显示保存边的信息，如图1-10所示。

4. 基于边连接

基于边连接（Edge Based Connectivity）的数据结构有三个部分：顶点、边、面。其中每个顶点保存一个边的索引指向，每个边保存两个顶点和两个相邻面的索引，每个面保存一个边的索引。这种数据结构里面边是没有方向的，如图1-11所示。

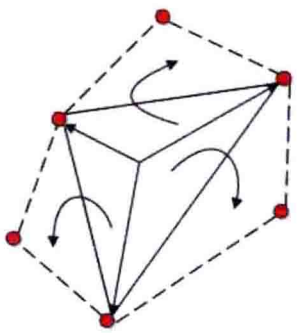


图1-10　面连接数据结构模型

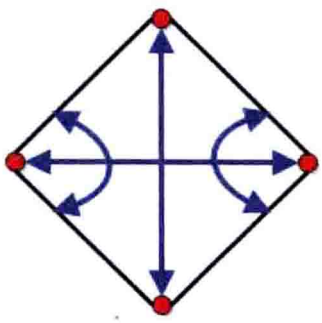


图1-11　边连接数据结构模型

例如，在做模型简化的时候常常需要将一条边删除并用一个顶点代替。这个操作就需要更新与所删除边相关元素的拓扑结构，使用以前的那种方法可能会需要将整个模型的面或点都遍历才能完成操作，这样会浪费大量时间，并占用大量内存。

5. 邻接矩阵

邻接矩阵（Adjacency Matrix）数据结构用矩阵形式来保存数据。这个矩阵命名为A，矩阵里的每个元素为A_{ij}。如果两个顶点V_i和V_j相连，那么矩阵相应元素A_{ij}为1，如图1-12所示的三维模型，V_1和V_3顶点相连，那么如图1-13里面的邻接矩阵数据结构所示A_{31}=1。

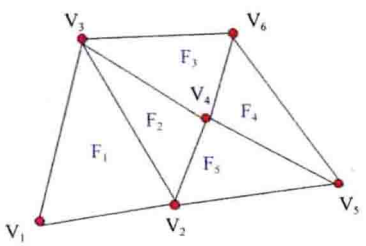


图1-12　邻接矩阵数据结构模型

A=		V_1	V_2	V_3	V_4	V_5	V_6
	V_1		1	1			
	V_2	1			1	1	
	V_3	1	1		1		1
	V_4		1	1		1	1
	V_5		1		1		1
	V_6			1	1	1	

图1-13　邻接矩阵数据结构

邻接矩阵是对称的矩阵。相邻矩阵可以用于非流形（Manifold）的三维网格，但是没有顶点和相邻面的信息。

6. 角表

在角表（Corner Table）数据结构中，一个角是一个顶点和它相邻的三角形面组成的角。组成元素有以下几方面。

① 角（Corner）：用C来表示。

② 与该角相连的三角形面：用C.T来表示。

③ 与该角相连的顶点：用C.V来表示。

④ 在同一个三角形里面（C.T）里面的下一个角（逆时针CCW）：用C.N来表示。

⑤ 前一个角（逆时针CCW）：用C.P来表示。

⑥ 一个角的对角（Opposite Corner）：用C.O来表示。

⑦ 右角（Right Corner）：用C.R来表示，C.R=C.N.O。

⑧ 左角（Left Corner）：用C.L来表示，C.L=C.P.O=C.N.N.O。

如图1-14所示为角表数据结构模型，其角表数据结构如图1-15所示。

如果C=C_1，那么C.T=F_1；C.V=V_1；C.N=C_2；C.P=C_3；C.O=C_6。

如果C=C_3，那么C.R= C_6；

若果C=C_2，那么C.L=C_6。

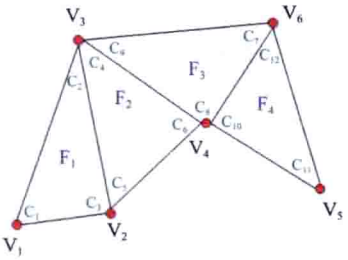


图1-14　角表数据结构模型

角	C.V	C.T	C.N	C.P	C.O	C.R	C.L
C_1	V_1	F_1	C_2	C_3	C_6		
C_2	V_2	F_1	C_3	C_1			C_6
C_3	V_3	F_1	C_1	C_2		C_6	
C_4	V_3	F_2	C_5	C_6		C_7	C_1
C_5	V_2	F_2	C_6	C_4	C_7	C_1	
C_6	V_4	F_2	C_4	C_5	C_1		C_7

图1-15　角表的数据结构

角表数据结构需要保存每个顶点相连的所有角指向，以及该角所有相关角的信息。这种数据结构对大多数操作来说时间复杂度是O（1），但是占用很多内存。

以上都是常用的三维模型的数据结构，总的来说，若只是用来显示而不进行删除、查找等其他操作的话，基于数组的数据结构是比较高效的。

1.4　半边数据结构

半边数据结构（Half-Edge Data Structure）是一个以边为中心的数据结构。它存储了一个三维网格模型所有顶点、边和面的数据以及相邻接关系的信息，并且利用半边来表示了边的方向，使在三维模型上的很多局部操作得以高效的进行。半边数据结构占有内存少，从而弥补了之前许多种数据结构的不足，因此它成为许多应用的首选数据结构，但是它的缺点是只能处理流体（Manifold）模型而不能处理非流体（Non-Manifold）模型。

半边数据结构由5部分组成：顶点（Vertex）、半边（HalfEdge）、边（Edge）、面（Face）和模型（Mesh）。其中半边是有方向（Oriented）的边，而边是没有方向（Non-Oriented）的边，一个没有方向

的边可以视为两个方向相反的半边。半边数据结构的类之间的关系如图1-16所示。

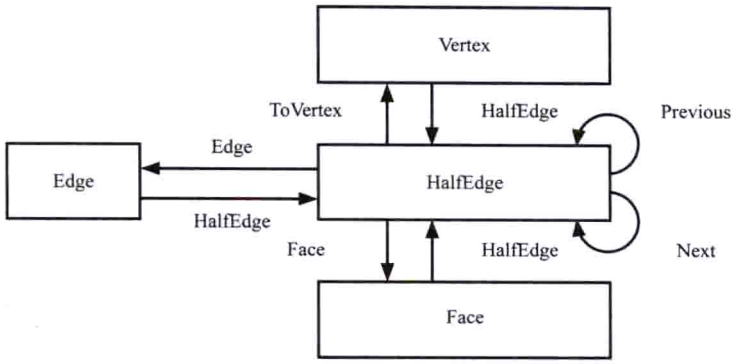


图1-16　半边数据结构的类之间的关系

对于每个顶点来说，半边数据结构除了保存有该顶点的位置等信息，还有一个以此顶点为源点的半边连接指向信息，常常一个顶点可能与若干个半边相连，可以随机选择一个以此为源点的半边作为索引，如图1-17所示。

对于每个半边保存的信息有：此半边的目的顶点（Target Vertex）的指向、半边左侧的面的指向；前一个（Prev）半边指向、下一个（Next）半边指向和点对应的孪生（Twin）半边指向，孪生半边就是和此半边共享一条边，但是方向相反的半边，如图1-18所示。

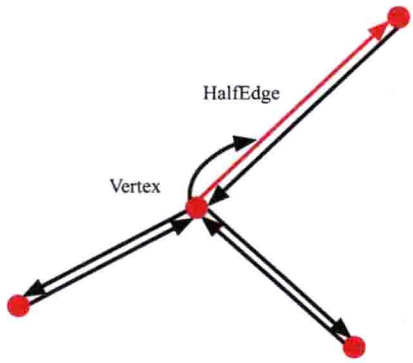


图1-17　半边数据结构保存的顶点信息

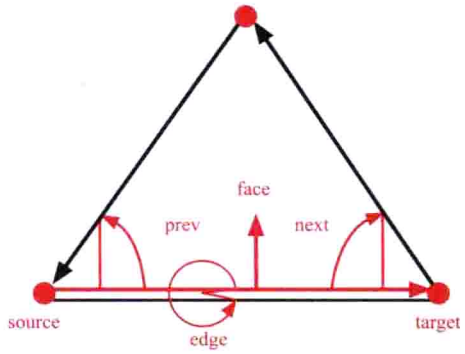


图1-18　半边数据结构保存的半边信息

一个边由两个方向相反的半边组成，在数据结构里面只需要对其中一条进行索引，如图1-19所示。

一个面保存的信息有：一条和此面相邻的半边（Adjacent HalfEdge）指向，一个三角形面有三条边，也就是六条半边。在逆时针方向的三条半边中随机选择一条进行保存，如图1-20所示。

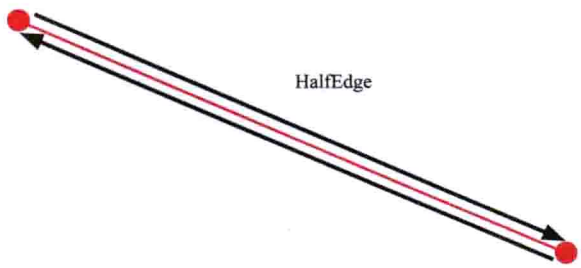


图1-19　半边数据结构保存的半边索引信息

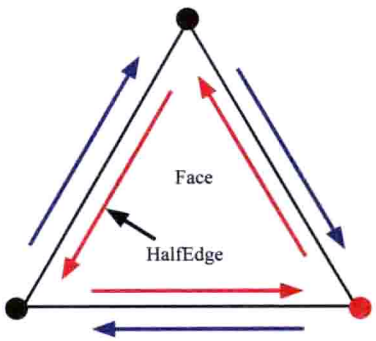


图1-20　半边数据结构保存的面信息

三维模型的数据结构里包含一个顶点的列表、一个半边的列表、一个边的列表及一个面的列表，如图1-21所示。

通过半边数据结构，三维网格上很多操作可以方便、快速有效地进行。如图1-22操作，从一个点，顺序得到此点的所有直接相邻的点，算法思路如下。

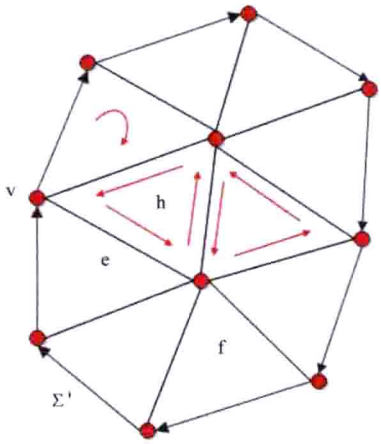


图1-21　半边数据结构保存的信息

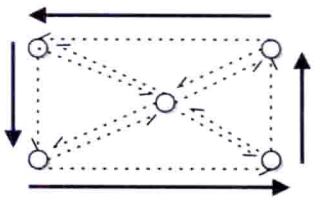


图1-22　操作图示

第一步：通过该点可以直接得到此点的一个相邻的半边。

第二步：通过此半边可以得到一个相邻的顶点，以及此半边的下一个相邻的半边。

第三步：重复上一步，就可以得到所有相邻的顶点。

半边数据结构算法思路示意如图1-23所示。

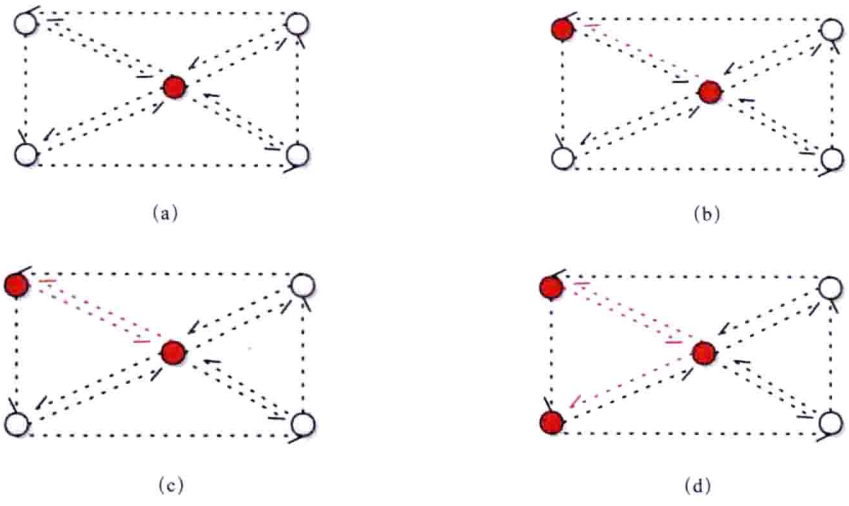


图1-23　半边数据结构算法思路图示

1.5　半边数据结构代码

现今有很多开源的半边数据结构的库，如OpenMesh、CGAL。本书中采用C#语言作为示例语言。因为C#语言结合了C++语言和Java语言的优点，是一种简单、精确、类型安全、面向对象的语言，非常适合作为数据结构和算法的演示。虽然相比较C++来说，C#语言效率比较低，但是本书主要介绍数据结构和算法，用C#语言可以使重点集中在数据结构和算法上，而不是各种指针的处理上面。

半边数据结构在具体实现的时候，可以把半边数据结构作为一个逻辑上的结构看待，也就是点、半边、边、面之间的链接是一个逻辑上的元素。这些元素之上附着的各种具体信息放在对应属性（Traits）