计 算 机 图 形 学 基 础

唐泽圣　周嘉玉　李新友　编著

清 华 大 学 出 版 社

(京)新登字 158 号

内 容 简 介

本书是作者在多年教学工作的基础上,参考了国内外最新版本的教材,编写成适合高校本科生、研究生“计算机图形学基础”课程的教材。

全书共分九章,各章附习题,最后附有上机实验题。第一章为绪论,以下八章分别论述了图形系统、二维图形生成技术,图形的裁剪及几何变换,图形学中的交互技术与用户接口,三维形体的表示和输出流水线、消隐及真实感图形生成技术,最后为反映计算机图形学与图象处理结合方面的知识,讨论了数字图象处理技术。

图书在版编目(CIP)数据

计算机图形学基础/唐泽圣等编著. —北京: 清华大学出版社, 1995
ISBN 7-302-02016-7

Ⅰ.计… Ⅱ.唐… Ⅲ.计算机图形学-基础理论 Ⅳ.TP391.4

中国版本图书馆 CIP 数据核字(95)第 19580 号

出版者: 清华大学出版社(北京清华大学校内,邮编 100084)
印刷者: 北京海淀区清华园印刷厂
发行者: 新华书店总店北京科技发行所
开 本: 787×1092 1/16 印张: 20 16K 彩色插页 1 字数: 455 千字
版 次: 1995 年 12 月第 1 版 1996 年 7 月第 2 次印刷
书 号: ISBN 7-302-02016-7/TP·933
印 数: 6001—14000
定 价: 18.00 元

前　　言

自从 80 年代初开始，国际上著名大学纷纷为本科大学生及研究生开设“计算机图形学”课程，我国各大学开设这一课程也已经有了十年以上的历史。近年来，国内外先后出版了多本教材。但是，由于计算机系统的硬、软件性能和图形功能的发展十分迅速，教材出版后不久，其内容就会落后于技术的发展，难于满足教学的要求。

为此，我们在多年教学工作的基础上，并参考国外最新版本的教材，编写了这本书，作为高校本科生、研究生“计算机图形学基础”课程的教材，在编写过程中，我们力求体现下述原则：

(1) 根据教学大纲要求，着重介绍计算机图形学的基本内容，为进一步深入学习这一领域的知识打下基础。

(2) 尽量反映 90 年代以来计算机图形学的最新发展所需要的基本知识，为掌握这些最新成果准备条件。

(3) 贯彻理论与实践相结合的原则，为同学们上机实践提供指导。

根据以上原则，本书在第一章绪论，第二章图形系统，第三章二维图形生成技术，第四章图形的裁剪及几何变换之后，第五章着重介绍了计算机图形学中的交互技术与用户接口，第六章介绍了三维形体的表示，第七章介绍了三维形体的输出流水线，第八章是消隐及真实感图形生成技术。为了反映计算机图形学与图象处理的结合日益密切这一趋势，增加了第九章数字图象处理技术初步，着重介绍与计算机图形学密切有关的图象处理技术。在本书中，除每章之后附有习题以外，在本书的最后，还给出了上机实验题。

本书的第一章、第六章、第九章是由唐泽圣同志编写的，第二章、第三章、第四章是由周嘉玉同志编写的，第五章、第七章、第八章是由李新友同志编写的，全书由唐泽圣同志统编定稿，在本书的编写过程中，清华大学计算机系 CAD 中心的杨微同志在计算机上绘制了书中的大部分插图，徐少奕同志为书中的程序实例及上机实验题编制和实现了全部 C 语言程序。在此，我们对他们表示衷心的感谢。

欢迎读者在阅读本书的过程中，对本书存在的缺点和问题提出批评和意见。

编　者

1994 年 6 月

目　　录

第一章　绪　　论

一、计算机图形学与图象处理

计算机图形学的发展已经有30年的历史了。它的基本含义是使用计算机通过算法和程序在显示设备上构造出图形来。也就是说,图形是人们通过计算机设计和构造出来的,不是通过摄象机或扫描仪等设备输入的图象。所设计和构造的图形可以是现实世界中已经存在的物体的图形,也可以显示完全虚构的物体。因此,计算机图形学是真实物体或虚构物体的图形综合技术。

与此相反,图象处理是景物或图象的分析技术,它所研究的是计算机图形学的逆过程。包括图象增强、模式识别、景物分析、计算机视觉等,并研究如何从图象中提取二维或三维物体的模型。图1-1简要的表示出计算机图形学和图象处理的区别和联系。

图　1-1

尽管计算机图形学和图象处理所涉及的都是用计算机来处理图形和图象,但是长期以来却属于不同的两个技术领域。近年来,由于多媒体技术、计算机动画、三维空间数据场显示及纹理映射等的迅速发展,计算机图形学和图象处理的结合日益紧密,并相互渗透。

例如,将计算机生成的图形与扫描输入的图象结合在一起,构造计算机动画;用菜单或其它图形交互技术来实现交互式图象处理;通过交互手段,由一幅透视图象中提取出对称物体的三维模型并进行修改;也可由一幅图象直接变换为另一幅图象从而代替了图形的综合等等。计算机图形学与图象处理相结合,加速了这两个相关领域的发展。

二、计算机图形学的发展简史

计算机图形学的发展历史应追溯到50年代末至60年代初期。当时的计算机主要应用于科学计算,使用尚不普及,但已开始配置了图形显示器。例如,美国麻省理工学院旋风(Whirlwind)计算机就配置了由计算机驱动的阴极射线管式的图形显示器,但不具备人-机交互功能。50年代末期,美国麻省理工学院林肯实验室研制的SAGE空中防御系统,就已具有指挥和控制功能。这个系统能将雷达信号转换为显示器上的图形,操作者可以借用光笔指向屏幕上的目标图形来获得所需要的信息,这一功能的出现预示着交互式图形生成技术的诞生。

1963年,美国麻省理工学院的I. E. 萨瑟兰德(I. E. Sutherland)在他的博士论文中提出了SKETCHPAD系统。这一系统用的是TX-2型计算机及阴极射线管式图形显示器,它可用光笔在图形显示器上实现选择、定位等交互功能。计算机还可以跟踪光笔,从原来所在的点到所指定的点画出直线,或在给定圆心和半径后画出圆等。这一系统还引入了分层存储符号和图素的数据结构,因而一幅完整的图可以通过分层调用若干子图来产生。这些基本概念和技术直至今日还是有用的。因此,I. E. 萨瑟兰德的SKETCHPAD系统被公认为对交互图形生成技术的发展奠定了基础。

30年来,交互式计算机图形学有了飞速的发展。在本章中,主要介绍支持计算机图形生成技术的硬、软件环境的发展和变化。

在60年代,计算机主要以大型机的形式出现,大多用于科学计算及事务管理。因而,以大型计算机连接图形终端成为计算机图形生成技术的运行环境。例如,美国通用汽车公司用于计算机辅助汽车设计的DAC系统,美国CDC公司开发的Digigraph计算机辅助设计系统等。进入70年代以后,由于集成电路技术的发展,计算机硬件性能不断提高,体积缩小。价格降低,特别是廉价的图形输入、输出设备及大容量磁盘等的出现,以小型计算机及超级小型机为基础的图形生成系统开始进入市场并形成主流。由于这种系统比起大型计算机来,价格相对便宜,维护使用也比较简单,因而,70年代以来,计算机图形生成技术在计算机辅助设计、事务管理、过程控制等领域得到了比较广泛的应用。

进入80年代以后,工程工作站的出现,极大地促进了计算机图形学的发展。比起小型计算机来,工程工作站在用于图形生成上具有显著的优点。首先,工程工作站是一个用户使用一台计算机,交互作用时,响应时间短;其次,工作站连网后可以共享资源,如大容量磁盘,高精度绘图仪等;而且它便于逐步投资、逐步发展、使用寿命较长。因而,工作站已经取代小型计算机成为图形生成的主要环境。目前,工程工作站正朝着提高速度,扩大容量,提高分辨率,具有标准化的友好的用户界面以及采用多媒体技术等方向发展。将整数运算速度提高到每秒10亿次以上,内存容量扩大到1000兆字节以上的日子,已经为期不远了。

80 年代后期以来,微型计算机的性能迅速提高,以 486 微机为例,整数运算速度已达两千万次以上,内存容量达 64 兆字节,磁盘容量可达 1000 兆字节,而且可以配高分辨显示器及窗口管理系统,并在网络环境下运行,因此,已经成为计算机图形生成技术的重要环境。由于微机系统的价格仍较工作站便宜,因而受到广大用户的欢迎,更易于普及和推广。随着微机性能的进一步提高,可以预计,它将在计算机图形技术的应用中扮演更为重要的角色。

除了计算机系统本身的性能以外,推动计算机图形学不断前进的另一重要因素是图形显示设备的发展。自 60 年代中期出现并得到推广应用的显示设备称为矢量显示器,在这里,矢量是划线的同义词。一个典型的矢量显示器包括一个显示处理器,它作为输入/输出外围设备连接在中央处理器上,一个缓冲存储器和一个阴极射线示波器(CRT)。缓冲存储器中存放着计算机产生的显示程序,它包括带有端点坐标的画线命令以及绘制字符的命令等。矢量显示器的实质是电子束按照显示命令给定的顺序作偏转运动,因此,也称为随机扫描显示器。

为了避免屏幕图象的闪烁,显示处理器必须以每秒 30 次以上的速度周期性的扫描显示程序,刷新萤光屏。在 60 年代,具有这种刷新能力的矢量显示器是很昂贵的,因而影响了它的广泛使用。

60 年代后期出现了存储管理式显示器,它不需要缓冲存储器和刷新过程,消除了闪烁问题,而价格却比矢量显示器便宜一个数量级。这就使得交互式图形生成技术的广泛应用成为现实。将这种显示器连接在小型计算机上,成为 70 年代计算机图形系统的典型模式。但存储管理显示器也存在一个缺点,即不具备局部修改的能力。因而难以进行动态显示。

70 年代初期,基于电视技术的光栅扫描显示器的出现极大地推动了计算机图形学的发展。光栅扫描显示器将被显示的图象以点阵形式存储在刷新缓存中,由视频控制器将其读出并在屏幕上产生图象。光栅扫描显示器较之随机扫描显示器有许多优点。一是规则而重复的扫描比随机扫描容易实现,因而价格便宜;二是可以显示用颜色或各种模式填充的图形,这对于生成三维物体的真实感图形是非常重要的。三是刷新过程与图形的复杂程度无关,只要基本的刷新频率足够高,就不会因为图形复杂而出现闪烁现象。由于光栅扫描显示器具有许多优点,因而直至今日仍然成为图形显示的主要方式,工程工作站及微型计算机都采用这种光栅扫描显示器。

随着计算机图形显示器从专用的图形输出设备发展为标准化的人-机通信接口,就给图形显示软件提出了一个要求,要求它由低层次的与设备有关的软件包转变为高层次的与设备无关的软件包。与设备有关的图形软件包原本是制造厂为他们的专用显示设备提供的,而现在却要求它能驱动多种不同的显示设备。图形软件标准化的这一问题是在 70 年代中期提出的,并由美国计算机学会计算机图形学专门组(ACM SIGGRAPH)在 1979 年提出了 3D CORE 图形软件标准。后来经国际标准化组织批准的第一个图形软件标准是 GKS,这是一个二维图形标准。它的三维扩充 GKS 3D 在 1988 年被批准为三维图形软件标准。与此同时,美国国家标准化委员会也提出了一个程序员的层次式交互图形系统 PHIGS(Programmer's Hierarchical Interactive Graphics System),正如其名字表示的那

样,它支持三维图象的层次嵌套结构,而 CKS 只支持逻辑上相关图象的组合,称为图段,但并非嵌套结构。目前图形软件标准正在朝着高性能、开放式和高效率的方向发展。

三、计算机图形学的应用

正如前述 30 年来,计算机图形系统的硬、软件性能日益提高,而价格却逐步降低,这必然促使计算机图形生成技术的应用日益广泛。目前,已经应用于工业、科技、教育、管理、商业、艺术、娱乐等多个领域。近十年来,我国在计算机图形学的应用上也取得长足进展,在本节中所介绍的图形,均由国内自主开发的软件制作而成。

计算机图形学的代表性应用有:

1) 计算机辅助绘图及设计

在计算机辅助设计(CAD)中,交互式图形生成技术用来设计机械、电子设备等产品的元部件和系统以及工程建筑。例如:汽车本身、飞机和船体的外壳、模具、大规模集成电路、光学系统、民用建筑、服装、玩具等。有时候,用户仅仅需要绘制出设计对象的精确图形,更常见的是,在计算机中构造出设计对象的模型,然后对它的机械性能、电性能或热性能进行分析计算并用图形显示其结果。图 1-2 及图 1-3 是计算机辅助设计软件设计出的机械零件及茶壶、茶杯图象,(见彩图)。

2) 事务管理中的交互式绘图

应用图形学最多的领域之一是绘制事务管理中的各种图形,如统计数据的二维及三维图形,直方图、线条图,表示百分比的扇形图。又如工作进程图,库存和生产进程图以及大量的其它图形。所有这些都以简明的形式呈现出数据的模型和趋势以增加对复杂现象的理解并促进决策的制定。

3) 科学计算可视化

随着科学技术的进步,人类面临着越来越多的数据需要进行处理。这些数据来自高速计算机,人造地球卫星,地震勘探,计算机层析成象和核磁共振等途径。科学计算可视化就是应用计算机图形生成技术将科学及工程计算的中间结果或最后结果以及测量数据等在计算机屏幕上以图象形式显示出来,使人们能观察到用常规手段难以观察到的自然现象和规律,实现科学计算环境和工具的进一步现代化。科学计算可视化可广泛应用于计算流体力学、有限元分析、气象科学、天体物理、分子生物学、医学图象处理等领域。图 1-4 及图 1-5 是由三维空间数据场显示软件产生的三维气象图象及人的头骨图象(见彩图)。

4) 过程控制

在过程控制中,常常将计算机与现实世界中的其它设备连成一个系统。计算机图形显示设备常用来显示系统中关键部位的状态。如炼油厂、发电厂和电力系统的状态显示器可显示出由传感器送来的压力、温度、电压、电流等数据,从而使操作人员可对异常情况作出反应。机场的飞行控制人员从雷达显示器上观察到计算机产生的标志及状态信息,因而可以更快、更准确的管理空中交通。

5) 计算机动画及广告

由于计算机图形系统的硬件速度提高,软件功能增强,因而利用它来制作计算机动画已经是相当普遍的了。为了避免画面闪烁,放映一秒钟的动画,就需制作 24 幅画面,因而

制作较长时间的动画,工作量是相当大的。但是,利用计算机制作动画恰恰可以在两幅关键画面之间自动插入中间画面,从而大大节约了时间,提高了动画制作的效率。目前,利用计算机图形生成技术制作的动画作为电视节目的片头或广告,在国内已经屡见不鲜了。

6) 计算机艺术

将计算机图形学与人工智能技术结合起来,可构造出丰富多彩的艺术图象,这是近年来计算机图形学的又一个重要应用领域。利用专家系统中设定的规则,可以构造出形状各异的多种图案并实现合理的颜色配置。图 1-6 是由智能 CAD 系统创作的美术图案(见彩图)。此外还可以利用计算机图形学技术生成盆景和书法等。

7) 地形地貌和自然资源的图形显示

应用计算机图形生成技术产生高精度的地理图形或其它自然资源的图形是另一个重要的应用领域。包括地理图、地形图、矿藏分布图、海洋地理图、气象图,植被分布图等。目前,地理信息管理系统已经在先进国家中得到广泛的应用。它就是建立在地理图形基础之上的信息管理系统。图 1-7 是表示我国地形的三维立体图形(见彩图)。

8) 办公自动化及电子出版系统

随着微型计算机及桌上印刷设备的发展,计算机图形学及人机交互技术在办公自动化及电子出版系统中得到广泛的应用。昔日需要提交给专门的印刷机构出版的资料,现在可以在办公室内印刷了,办公自动化及电子出版系统可以产生传统的硬拷贝文本,也可以产生电子文本,包括正文、表格、图形及图象等内容。

国外资料表明,在整个 90 年代,计算机图形生成技术的应用仍将不断发展,计算机图形系统的市场将以每年 15% 至 20% 的递增速度不断扩大。1990 年为 160 亿美元,1995 年将达到 360 亿美元,至 2000 年将达到 800 亿美元。但是,在应用领域的分布方面将会有些变化。整个 80 年代,计算机辅助设计/制造成为计算机图形技术的最主要应用领域,大约占整个计算机图形技术应用的二分之一左右。显然,这一领域的应用仍将继续发展。但是,由于多媒体技术,科学计算可视化,计算机动画等领域的迅猛发展,至 1995 年,计算机辅助设计/制造在计算机图形技术的应用中所占的份额将降低至三分之一左右,与此相反,以图形或图象来表示和说明客观世界的各种各样的应用将上升至二分之一左右。

四、计算机图形学的发展动向

前面已经提到,计算机图形学是通过算法和程序在显示设备上构造出图形的一种技术。这和用照像机摄制一幅照片的过程比较类似。当用照相机摄制一个物体、比如说一幢建筑物的照片时,首先在现实世界中必须有那么一幢建筑物存在,才能通过照相的原理拍摄一张照片。与此类似,要在计算机屏幕上构造出三维物体的一幅图象,首先必须在计算机中构造出该物体的模型。这一模型是由一批几何数据及数据之间的拓扑关系来表示的。这就是造型技术。有了三维物体的模型,在给定了观察点和观察方向以后,就可以通过一系列的几何变换和投影变换在屏幕上显示出该三维体的二维图象。为了使二维图象具有立体感,或者尽可能逼真地显示出该物体在现实世界中所观察到的形象,就需要采用适当的光照模型,尽可能准确地模拟物体在现实世界中受到各种光源照射时的效果,这些就是计算机图形学中的画面绘制技术。三维物体的造型过程、绘制过程等都需要在一个操作方

便、易学易用的用户界面下工作,这就是人机交互技术。多年来、造型技术、绘制技术及人机交互技术构成了计算机图形学的主要研究内容。当前,仍然在这三个方面不断地向前发展。

1) 造型技术的发展

计算机辅助造型技术以所构造的对象来划分,可以分为规则形体造型和不规则形体造型。规则形体指的是可以用欧氏几何进行描述的形体,例如: 平面多面体、二次曲面体、自由曲面体等,统称为几何模型。构造几何模型的理论、方法和技术称为几何造型技术,它是计算机辅助设计的核心技术之一,因而早在 70 年代国际上就进行了广泛而深入的研究,目前,已有商品化的几何造型系统提供给用户使用。近年来,由于非均匀有理 B 样条(Nonuniform Rational B Spline)具有可精确表示圆锥曲线的功能,以及对控制点进行旋转、比例、平移及透视变换后曲线形状不变的特点,因而为越来越多的曲面造型系统所采用。同时,将线框造型、曲面造型及实体造型结合在一起,并不断提高造型软件的可靠性也是造型技术的重要研究方向。

虽然几何造型技术已得到广泛应用。但是,它只是反映了对象的几何模型,而不能全面反映产品的信息,如产品的形状、公差、材料等,从而使得计算机辅助设计/制造的一体化难于实现。在这样的背景下,就出现了特征造型技术,它将特征作为产品描述的基本单元,并将产品描述成特征的集合。例如,它将一个机械产品用形状特征、公差特征、技术特征三部分来表示,而形状特征的实现又往往是建立在几何造型的基础之上的。目前,特征造型技术在国内外均处于起步阶段。

近几年来,主要是由于发展动画技术的需要,提出了基于物理的造型技术。在几何造型中,模型是由物体的几何数据和拓扑结构来表示的。但是,在复杂的动画技术中,模型及模型间的关系相当复杂,不仅是静态的、而且是动态的。这时,靠人来定义物体的几何数据和拓扑关系是非常繁杂的,有时甚至是不可能的。在这种情况下,模型就可以由物体的运动规律自动的产生,这就是基于物理的造型技术的基本概念。显然,它是比几何造型层次更高的造型技术。目前,这种基于物理的造型技术不仅可在刚体运动中实现,而且已经用于柔性物体。

与规则形体相反,不规则形体是不能用欧氏几何加以定义的,例如,山、水、树、草、云、烟、火以及自然界中丰富多彩的其它物体。如何在计算机内构造出表示它们的模型,是近年来研究工作的另一个特点。与规则形体的造型技术不同,不规则形体的造型大多采用过程式模拟,即用一个简单的模型及少量的易于调节的参数来表示一大类物体,不断改变参数,递归调用这一模型就能一步一步地产生数据量很大的物体,因而这一技术也称为数据放大技术。近年来,国际上提出的基于分形理论的随机插值模型、基于文法的模型以及粒子系统模型等都是应用这一技术的不规则形体造型方法,并已取得了良好的效果。

2) 真实图形生成技术的发展

真实图形生成技术是根据计算机中构造好的模型生成与现实世界一样的逼真图象。在现实世界中,往往有多个不同的光源,在光源照射下,根据物体表现的不同性质产生反射和折射、阴影和高光,并相互影响、构造出丰富多采的世界。早期的真实图形生成技术用简单的局部光照模型模拟漫反射和镜面反射,而将许多没有考虑到的因素用一个环境光

来表示。80 年代以后,陆续出现了以光线跟踪方法和辐射度方法为代表的全局光照模型,使得图象的逼真程度大为提高,但是却又带来了另一个问题,这就是计算时间很长。目前,在许多高档次的工作站上,已经配备了由硬件实现光线跟踪及辐射度方法的功能,从而大大提高了逼真图形的生成速度。

3) 人-机交互技术的发展

直至 80 年代初期,在设计计算机图形生成软件时,一直将如何节约硬件资源——计算时间和存储空间——作为重点,以提高程序本身的效率作为首要目标。随着计算机硬件价格的降低和软件功能的增强,提高用户的使用效率逐渐被认为是首要目标。为此,如何设计一个高质量的用户接口成为计算机图形软件的关键问题。

一个高质量的用户接口的设计目标应该是:易于学习、易于使用、出错率低、易于回忆起如何重新使用这一系统并对用户有较强的吸引力。80 年代中期以来,国际上出现了不少符合这一目标的人-机交互技术。例如:屏幕上不仅可以开一个窗口而且可以开多个窗口;从以键盘实现交互发展到以鼠标器实现交互;将菜单放在屏幕上而不是放在台板上;不仅有静态菜单而且有动态菜单;不仅用字符串作为菜单而且用图标作为菜单;图标可以表示一个对象,也可以表示一个动作,从而使菜单的含义一目了然。

如何在三维空间实现人-机交互一直是计算机图形技术的一个研究热点。近年来,虚拟环境技术的出现使三维人-机交互技术有了重要进展。所谓虚拟环境是指完全由计算机产生的环境,可是却具有与真实物体同样的外表、行为和交互方式。目前,典型的方法是用户头戴立体显示眼镜,头盔上装有一个敏感元件,反映头部的位置及方向,并相应改变所观察到的图象。手带数据手套实现三维交互,并有一个麦克风用来发出声音命令。

以上是关于计算机图形生成技术发展动向的简单介绍。由于本书是交互式计算机图形学的大学本科教材,上述发展动向中的大部分在本书中难于作详细介绍。不过,我们相信,在绪论中就这一问题作简单介绍对读者是有益的。

思 考 题

1. 试比较计算机图形学与图象处理技术的相同点和不同点。
2. 试描述您所熟悉的计算机图形系统的硬软件环境。
3. 试列举计算机图形学的三个应用实例。

第二章　图形系统

第一节　系统的组成

计算机图形系统应由硬件设备及相应的程序系统(即软件)两部分组成。

硬件包括主计算机,图形显示器以及鼠标器和键盘等基本交互工具,还有图形输入板、绘图机、图形打印机等图形输入输出装置,以及磁盘、磁带等存储设备。软件包括操作系统、高级语言、图形软件和应用软件。

硬件设备是计算机图形学存在与发展的物质基础,其本身又是计算机科学技术高水平发展和应用的结果。

第一台图形设备是 1950 年美国麻省理工学院(MIT)的“whilwind 1”(“旋风 1 号”)计算机的一个配件——图形显示器。它只能显示简单的图形,类似一台示波器。计算机科学技术经过 30 多年的长足进步,才有了如今日臻完善的计算机图形系统。

现代计算机图形系统与一般计算机系统最主要的差别是具有图形的输入、输出设备以及必要的交互工具。作为系统核心的主计算机,在运算速度和存储容量上均有较高的要求。一般运算速度至少为每秒数百万次,内存容量至少为数百千字节或数兆字节。这就不难理解,为什么直到 80 年代,计算机图形系统才得以更普遍的推广,因为这时个人计算机已达到这样的技术水平,可作为普及型图形工作站的主机。

严格说来,使用系统的人也是这个系统的组成部分。在整个系统运行时,人始终处于主导地位。可以说,一个非交互式计算机图形系统只是通常的计算机系统外加图形设备;而一个 交互式计算机图形系统则是人与计算机及图形设备协调运行的系统,如图 2-1 所示。

图 2-1　图形系统一般结构示意图

一、图形系统的基本功能

作为一个图形系统,至少应具有计算,存储、输入、输出、对话等五个方面的基本功能(见图 2-2)。

图 2-2　图形系统基本功能框图

1. 计算功能

图形系统应能实现设计过程中所需要的计算、变换、分析等功能。如直线、曲线、曲面等几何因素的生成,坐标的几何变换,线段、形体间的求交、裁剪计算以及点的包含性检查等。

2. 存储功能

在图形系统的存储器中,应能存放所设计的各种形体的几何数据及形体之间的相互关系,并可实现对有关数据的实时检索以及保存对图形的删除、增加、修改等信息。

3. 输入功能

图形系统应能将所设计形体的几何参数(例如大小、位置等)以及各种命令输入到系统中去。

4. 输出功能

图形系统应能在显示屏幕上显示出设计过程当前的状态,经过删、增、改后的结果。在得出满意的设计结果或有其它输出要求时,应能通过绘图仪、打印机等设备实现硬拷贝输出,以便长期保存。

5. 对话功能

图形系统应能通过图形显示器及其它人-机交互设备直接进行人-机通信。利用定位、拾取等手段,输入或获取各种参数,同时应能领会人的意图,接收各种命令,实现删、增、改等操作,并能观察设计结果。

如果需要这样一个具备基本功能的系统,首先就要对硬件设备和软件结构作出合理的选择。此外,还应考虑如下一些要求:

(1) 系统怎样与其它方面(如生产、设计以及人们的经验等)最佳地相互配合;

(2) 经济因素;

(3) 系统安装、运行、维护、管理的条件;

(4) 用户接口。

二、图形系统的分类

计算机图形系统根据其功能的强弱,也即所配置的硬件规模、软件丰富程度,以及价格的高低,大体可分为四类:

1. 以大型机为基础的图形系统

这种系统在发达国家多用于飞机制造、汽车制造等大型企业(也是应用计算机辅助设计技术最早的企业)。它以大型计算机为基础,具有容量庞大的存储器和极强的计算功能,并且具有大量的显示终端及高精度、大幅面的硬拷贝设备。这种图形系统还往往拥有自行开发的、功能齐全的、且不外传的应用软件系统。

例如,美国第三大汽车公司(CHRYSLER 汽车公司)就拥有庞大的计算机系统来实现计算机辅助设计、实体造型、结构分析、运动模拟、工程和科学计算、项目管理、生产过程控制等。早在十年前,该公司就拥有 CDC 公司的 CYBER 系列大型计算机 27 台,还有一些 IBM 公司及 DEC 公司的计算机,共有 151 个磁盘驱动器,磁盘总容量高达 195 千兆字节,拥有各种终端 1176 个,其中 550 个用于 CAD/CAM。该公司还自行开发了一整套图形应用软件,这一庞大的软件系统是该公司用来实现产品更新、满足市场需求和在竞争中求得生存的重要保证。

2. 以中型机或超级小型机为基础的图形系统

70 年代后期,随着具有分时操作系统的 32 位超级小型机的出现,很快推出了以它为基础的图形系统。与第一类系统不同,这一类系统是以商品形式出现的,它们在 70 年代末至 80 年代初,成为计算机辅助设计市场上的主流产品。

这种系统以 32 位超级小型机作为处理机,并配有较大容量的内存和外存,配有高精度、大幅面的图形输入输出设备。这类系统还常配备具有较强功能的图形支撑软件和应用软件随机出售。但这类系统价格较贵,一般每套需要几十万美元。如美国的 Intregraph, Applicon 及 computer Vision 等公司推出的产品均属此类系统,然而,此类系统目前大多已被淘汰。

3. 以工程工作站为基础的图形系统

80 年代初期以来,出现了工程工作站和以它为基础的图形系统。工作站是具有高速的科学计算、丰富的图形处理、灵活的窗口及网络管理功能的交互式计算机系统。它与具有分时系统的超级小型机不同:一个用户使用一台计算机,并具有联网功能。由于联网后可以共享资源,便于逐步投资、逐步发展等优点,因而受到了广泛欢迎。这类系统发展十分迅速,已经取代了第二类图形系统。美国的 Sun, SGI, HP, DEC, IBM 等公司的产品均属此类。

工程工作站大多采用 32 位或 64 位字长的中央处理器(CPU);广泛采用精减指令(RISC),超标量、超流水线等技术;内存至少 8MB,可扩充到 100MB 以上,高速缓存大多在 32KB 以上;自带外存,磁盘容量在 600MB 以上;运算速度在 20MIPS 和 5MFLOPS 以上。具有 UNIX 操作系统和 X 窗口管理系统,还配有用户界面开发工具,如 Motif 或 Openlook);不仅有字符处理功能,更要有图形处理功能,图形显示器的分辨率在 1024×900 以上,一般具有 8 个位面(可显示 256 种颜色),有的具有 100 个位面以上;可以在网

络的任何地方(近程或远程)存取信息,具有无盘节点和有盘节点的形式;此类系统不仅可用于办公室自动化、文字处理、文本编辑等,更主要的是用于工程和产品的设计与绘图,运动模拟、动画和科学计算可视化等领域。

在以工作站为基础的图形系统中,制造商一般只提供硬件及系统软件,应用软件多由专门从事系统集成的公司或其它公司开发,由用户选购。

这类工作站的价格视档次高低不同而异,低档的每台不到 1 万美元,高档的十几万美元。

4. 以微型机为基础的图形系统

近年来,由于微型机的性能日益提高,价格日益降低,因而使得以微机(例如 386,486 微机)为基础的图形系统在技术上成为可能,日益得到广泛应用。

这类系统以常见的高档微机为基础,配上浮点运算部件,并配以中、低分辨率的图形显示器以及交互设备,廉价的绘图仪及打印机等,单价低于几千美元。

目前,以微型机为基础的图形系统多用作二维图形的计算机辅助绘图,应用软件也多是二维的。但是,随着微型机性能的进一步提高,用微型机实现三维形体的设计及显示也是完全可能的。

第二节　图形硬件设备

与一般计算机系统一样,计算机图形系统也具有中央处理机、外存储器、打印机、键盘等设备。此外,还必须有图形的输入设备和输出设备,以保证图形系统基本功能的实现。根据不同的应用领域要求,可以有各种不同的设备配置。比如,若仅用字符产生的图形即可满足应用要求,那么就可以在与通用计算机相联的打印机或显示屏幕上,用“PRINT”或“WRITE”语句输出字符来构成图形(见图 2-3)。如果配有绘图仪,就可以产生如图 2-4 所示的图形。在图形显示器上,我们可以利用画线命令交互地建立和处理图形。如果还有更进一步的应用要求,那就必须配置其它相应的硬件设备。

一、显示设备

显示设备是最终产生图形显示效果的部件。虽然已有各种各样的图形显示装置和许多新的显示技术和显示设备出现,但是目前占统治地位的仍是阴极射线管(CRT)显示器,它的核心部件是 CRT,估计这种状况还要持续多年。

1. CRT

(1) 单色 CRT

单色 CRT 的构造见图 2-5,它利用电场产生高速的聚焦电子束,偏转到屏幕表面的不同部位,以产生可见图形。CRT 主要由 3 部分组成:电子枪、偏转系统和荧光屏。

① 电子枪

电流通过灯丝产生热量,即对阴极加热而发射出电子束。在聚焦极上加上一定的正电压,使电子束聚焦,形成很细的电子束。再由加速极(通常不止一个)加上正电压对电子束加速,使它具有足够的能量射向荧光屏。靠近阴极有一控制极,加上负电压后能控制电子

图 2-3　采用不同的字符组合，在打印机上产生的图形

图 2-4　笔式绘图仪输出的图形

束的强弱，甚至使电子束截止。也就是说，控制板能够控制电子束打在荧光屏上所产生的光的强弱。

② 偏转系统

偏转系统是 CRT 中最关键的部分，有了它才能在屏幕上描绘出图形。我们可以用静电场，也可以用磁场来控制电子束的偏转。用静电场产生偏转时，垂直和水平两套平板放在阴极射线管的管颈内部(见图 2-5)。磁偏转系统则是外部偏转系统，它有两个线圈绕在管颈上 (见图 2-6)，当电子束通过线圈时，一个线圈的磁场使电子束产生水平偏转，另一

个使之产生垂直偏转。

图 2-5　CRT 剖面示意图

图 2-6　磁偏转系统 CRT 剖面图

偏转系统最重要的特性是灵敏度,它反映了偏转信号所能产生的偏转角度的大小。如果一个 CRT 的灵敏度低,为获得较大的偏转角度,只能加长管子的长度,其结果,显示器就非常笨重。静电偏转技术的灵敏度由式(2-1)给出:

$$\tan\alpha = LV_d/(2DV_a) \tag{2-1}$$

式中　α为偏转角度(见图 2-7);

V_a 为加速电压;

V_d 为偏转电压;

L 为偏转板的长度;

D 为板间距离。

显然,当偏转角 α给定以后,V_a 愈大(即电子束速度愈高),所要求的偏转电压也愈高。磁偏转的灵敏度由式(2-2)决定:

图 2-7　偏转灵敏度示意图

$$\tan\alpha = BL/\sqrt{2KV_a} \tag{2-2}$$

式中　B 为磁场强度;

L 为偏转场的有效长度;

K 为常数。

同样,高速电子要求较强的磁场,也即要求较大的电流。但是,灵敏度与 $\sqrt{V_a}$ 成反比,而不是与 V_a 成反比。所以,一般说来,磁偏转方法允许用较高速的电子束,因而能产生亮度较高的图象。另外,磁偏转系统可以使电子束会聚成更小的光点,且具有结构简单及所需电功率较小等优点,所以大多数 CRT 都用磁偏转系统。

③ 荧光屏

荧光屏上涂有荧光粉,电子束打在荧光屏上,荧光粉就会发光而形成光点。除了颜色不同外,各类荧光物质之间的主要区别在于荧光物质的余辉时间。通常,由电子束轰击荧光层某点所产生的荧光,当电子束离开该点后,其亮度值随时间按指数规律衰减(见图 2-8)。余辉时间就是指光亮度值衰减到初始值的 1/10 所需的时间。各种荧光物质的余辉时间差别很大,可以从几微秒到几秒,而用于图形设备的大多数荧光物质的余辉时间一般

为几十到几百毫秒。因此,为了得到一幅稳定的、不闪烁的画面,一种方法就是使电子束不断重复地描绘出原来的图形。如果重复频率足够快,某点的亮度值对观察者来说就好像是一个常数,如图 2-9 所示。余辉时间是决定产生不闪烁图形所需刷新速率的主要因素。余辉越长,所需的刷新速率就越低。一般短余辉的荧光物质适用于动画显示;而长余辉的荧光物质则适用于显示复杂的、静态的画面。

图 2-8　荧光物质亮度与时间关系图

图 2-9　电子束重复扫描得到恒定的亮度

单色 CRT 显示图象的质量取决于:设备固有的单个光点直径的大小以及“可寻址能力”(addressability)。可寻址能力可以理解为单位长度内,能够利用的单个光点的数目,它与用于定位的计算机字长的位数以及数模转换器等因素有关。图 2-10 表示出不同的可寻址能力,即当点间距分别等于光点直径,或为光点直径的 1/2, 1/3, 1/4 时图形的显示效果。所以,通常总是希望点的直径大于点间距,以得到较光滑的图形。

一个 CRT 在水平或垂直方向上能够识别出的最大光点数称为分辨率。这是衡量 CRT 的重要指标,它取决于所用荧光物质的类型以及聚焦和偏转系统。有时我们也把分辨率理解成所能识别的光点的最小距离。显然,点数愈多,距离愈小,分辨率愈高,显示的图形也就愈精确。分辨率与光点直径大小有关,但是不可能大于可寻址能力。分辨率也与各个点的亮度曲线形状有关。在图 2-11 中,(a)图的各点亮度曲线边缘很陡,(c)图的则较平坦。因此,(b)图的点的分辨率要比(d)图的点的分辨率高得多。

图 2-10　光点尺寸与点间距有不同比值时所产生的效果

图 2-11　点的亮度曲线交叉部分的亮度对分辨率的影响

(2) 彩色 CRT

CRT 之所以能显示不同颜色的图形是由于使用了能发出不同颜色的荧光粉的结果。产生彩色显示的基本方法有两种: 一是射线穿透法, 二是影孔板法

射线穿透法显示彩色图形已经用于随机扫描显示器中, 它是在屏幕上涂有两层荧光粉, 一般是红色和绿色。所显示的颜色取决于射线穿透荧光层的深浅: 速度低的电子只能激励外层的红色荧光粉, 高速电子可以穿透红色层而激励内层的绿色荧光粉, 中速电子则可以使所激发出的红光和绿光组合而产生两种附加的颜色, 即橙色和黄色。因此, 电子的速度决定了屏幕上某点的颜色, 这可以由射线的加速电压来控制。射线穿透法是一种廉价的产生颜色的方法, 但它只能产生四种颜色, 而且图形的质量也不如其它方法好。

影孔板法广泛用于光栅扫描系统中, 它能产生比射线穿透法范围宽得多的色彩。这种 CRT 屏幕的内部涂有很多组呈三角形的荧光粉, 每一组有三个荧光点, 当某组荧光粉被激励时, 分别发出红、绿、蓝三个基色。这种类型的 CRT 有三个电子枪, 分别与三基色相对应。紧挨屏幕后面放有影孔板栅网, 上面有很多小孔, 与屏幕上的三元组一一对应(见图 2-12)。三束电子经偏转聚焦成一组射线, 穿过影孔板上的孔, 激活屏幕上的一个三元组, 出现一个色点。这些小孔与三元组和电子枪精确地排列成一条直线, 使得三元组中每个点

仅仅受到一个电子枪所发出的电子的作用。这样，每一电子束的电子数目就控制着三元组所产生的红、绿、蓝三种光的量。于是，可以根据混合色所需的各种成分，以不同的强度激发红、绿、蓝三个荧光点。因而可以在一个三元组上产生范围很宽的色彩等级。例如，关闭红色和绿色电子枪，就能得到蓝色；以相同强度的电子束去激发全部三个荧光点，就会得到白色。在廉价的彩色显示器中，电子束只有发射和关闭两种状态，因此，只能有八种颜色；而比较复杂的系统，则可以发射中间等级强度的电子束，因而能产生多达几百万种的颜色。

图 2-12　影孔板彩色 CRT 原理图

影孔板彩色 CRT 中的影孔板和三元组对其分辨率有较大的影响。一般高分辨显象管三元组的平均径距(pitch)约为 0.21mm，家用电视显象管约为 0.6mm。因为聚焦的电子束不能保证精确地穿过影孔板上对应孔的中心，所以，电子束的直径必须为三元组径距的 7/4 倍。例如，若影孔板的径距为 0.25mm(0.01 英寸)，则电子束的直径就为 0.45mm(0.018 英寸)，其分辨率约为 1/0.018 英寸= 55/英寸，对于 19 英寸的彩色 CRT，宽 15.5 英寸，高 11.6 英寸，那么可能的分辨率为 15.5 英寸× 55/英寸≈850 及 11.6 英寸× 55/英寸≈640。这与可寻址能力为 1280× 1024 或 1024× 800 的典型显象管相对应。

影孔板的径距对影孔板 CRT 的分辨率是很大的限制。随着径距减小，分辨率会提高。但是径距愈小，管子生产就愈困难。小径距的影孔板很脆，难于安装，也容易因电子束加热而弯曲。

影孔板也限制了 CRT 的亮度。一般来说，只有 20%的电子束轰击荧光屏，其余则轰击影孔板。因此，与单色 CRT 相比，能产生光的电子要少得多。可以增加电子束电流来增强亮度，但这又引起聚焦的困难，且也使影孔板更热而更为弯曲。

大多数影孔板 CRT 的尺寸为 15—21 英寸，其表面有点弯曲，对观察者而言，会产生光学畸变。有些平面式 CRT 也能生产，包括 29 英寸径距为 0.31mm 的管子，当然，其价

格也较高。

CRT 的刷新频率是指每秒重画图象的次数。为了得到稳定的画面,通常刷新频率应为 30—50 帧/秒。随着刷新频率的降低,会出现闪烁。使图象不闪烁的刷新频率称为临界停闪频率 CFF(critical fusion freqwency)。决定 CFF 的因素有:荧光屏的余辉时间,图象亮度及环境光的亮度,发射光的波长以及观察者本身,据分析调查,不同的观察者之间,CFF 可有 20Hz 之差。

(3) 直视型存储管 DVST(Direct-View Storage Tube)

这种存储管的写电子枪和普通的 CRT 一样,经聚焦的电子束在 X,Y 两个方向上偏转,其定位和偏转有随机性,但电子束不是直接写在荧光屏上,而是写在荧光屏前面的存储栅上(见图 2-13)。存储栅是一个很细的金属栅网,上面有介质,由写电子枪射出的高能量电子束将栅网上介质的电子轰击出来,轰击出来的电子比写上去的电子要多,所以栅网上遭电子束轰击的地方呈正电荷,电子束书写的轨迹在栅网上形成正电荷轨迹(也就是电子束描绘的轨迹)。在直视型的存储管内还有一个电子枪,叫第二电子枪,或称读出电子枪(泛流枪)。它射出的低能量漂浮电子流(泛流电子)大面积向收集极流去,收集极使这些电子均匀散开,流向存储栅。存储栅上呈现正电荷的地方吸引电子,使电子通过,通过的电子轰击荧光屏而发光,而其它位置则不通过电子。所以存储栅既存储图形,又控制电子通过,并在屏幕上形成图形。

图 2-13　直视型存储管原理图

由 DVST 显示图形不需要附加存储器和有关电路,能在几小时内显示不闪烁的复杂图形。它可在较低的传输速率下工作,如 300 或 1200 波特率,并具有价格便宜的优点。但是这种显示器不能做选择性修改,所以难以进行动态显示。近年有些直视型存储管提供了某种“写入通过”模式,它降低了写电子束的能量,使写电子不是保存在存储栅上,而是直接写在屏幕上,因而具有一定的动态性能。但是,由于写电子束的速度低,故所显示的直线段数目,也即图形的复杂性受到限制。美国 Tektronix 公司的 4014、4114 等显示器均是这种类型的显示器。

2. 其它类型的显示器件

(1) 等离子板显示器(Plasma Panel)

这种显示器由三层玻璃制成的单个集成板组成(见图 2-14)。前面一层的内表面有很细的电导体垂直线条,中间一层有许多小孔(氖泡),后面一层的内表面有很细的电导体水平线条。为了接通某一氖泡,要相应地调节两端电压,使其电位差足够大(一般约为 120V 左右),以“点燃”氖泡,使其发光。一旦开始发光,只需较低的电压(约 90V)即可维持。为了“关闭”氖泡,就要使其两端电压暂时低于维持电压。氖泡的开关时间约为 20 μs。

图 2-14　等离子板显示器原理图

等离子板显示器的主要优点是重量轻,且不需要刷新缓存。这种显示器已用于显示动态图形,但是其分辨率不高,且价格较贵。

(2) 液晶显示器件 LCD(Liguid-Crystal Display)

一般由六层组成,各层被压在一起,形成薄板。其中,液晶层约 0.0005 英寸,液晶材料由长晶体分子组成。通常这些单个分子呈螺旋形,因此通过它的偏振光被转了 90 度。进入前层的光是垂直的,当光线通过液晶后,偏振方向转了 90 度,成为水平,此时可通过后面的水平极化板,经反射后,又通过此两极化板及晶体。

当晶体在电场中时,其上所有分子均呈同一方向,这样就无偏振效应。因此,在电场中的晶体并不改变发射光的极化方向,所以光线保持其垂直偏振方向,也就无法通过后面的极化板,此光被吸收,所以观察者看到的是一暗点。

为了保存屏幕上的图形,可以借助辅助的存储器,系统重复地“扫视”存储器,连续地施加一定的“点燃”电压,以产生光脉冲。

LCD 的优点是:低价、重量轻、尺寸小且消耗功率低。其缺点是只反射入射光,而不产生自己的光,是被动的显示装置。近些年,利用激活板已克服了这种缺点。

(3) 电子发光显示器

其结构类似于 LCD 及等离子板显示器的结构,在前后板之间是一 500 毫微米薄层的电子发光材料层。当它在 10^5 v/cm 强电场中时可发射光线。这种显示器较亮,且通断迅速。典型尺寸为 6× 8 英寸及 12× 16 英寸等,每英寸可取址点为 70。缺点是电功率消耗大于 LCD,且价格偏高。

以上几种显示器件的性能比较见表 2-1。

表 2-1　常用显示器件性能比较表

性能项目	CRT	等离子板	液晶	电子发光
功耗	中等	中等	很低	尚低
屏幕尺寸	很好	很好	中等	好
重　量	不好	很好	很好	很好
深度	不好	好	很好	很好
不平度	尚好	很好	很好	好-很好
亮　度	很好	很好	尚好	很好
寻址能力	好-很好	好	尚好	好
反差	好-很好	好	中等	好
每点亮度等级	很好	中等	中等	中等
观察角度	很好	好-很好	不好	好
彩色能力	很好	中等	好	好
价　格	低	高	低	中等偏高

3. 随机扫描显示器

随机扫描显示器的基本结构见图 2-15。要显示的图形由计算机加工成为显示器的显示指令,即所谓的显示档案或显示文件。显示指令经接口电路送到显示器的缓冲存储器,而固定存储器中则存放各种常用字符、数字等显示指令。图形控制器取出缓冲存储器或固定存储器中的显示指令,依次执行。显示指令中的亮度、位移量等数字信息经线产生器化为控制电子束偏转和明暗的物理量,即电压或电流。再由管头控制电路使电子束以所需的亮度偏转到所需的位置。为了得到稳定的图形,还需要不断地重复扫描显示档案,即所谓的刷新。刷新频率通常为每秒 25—50 次,即每秒 25—50 帧左右。

图 2-15　随机扫描显示器结构

在随机扫描显示器中,电子束根据需要可在荧光屏的任意方向上连续扫描,没有固定扫描线和规定扫描顺序的限制,也就是说,电子束的定位及偏转具有随机性,故称随机扫描。此时电子束像一支快速移动的画笔,实际勾划出要显示的图形。随机扫描所描绘的图形只能是单线条图形,故随机扫描又称为矢量扫描或轨迹扫描,随机扫描显示器又称为画线显示器,其显示的图形质量很好。

由于这种显示器一般使用的都是低余辉的荧光粉,若刷新频率低于 25—30 帧/秒时,图形就会出现闪烁。当扫描速度一定时,图形的复杂性,也即画线的总长度就受到限制。目前国外最长的可画 250 米左右,国内最长的可达 120 米左右。

刷新式的随机扫描显示器可以不断改写缓存中的内容,因而具有局部修改及高度的动态性能。另外,它还具有较高的分辨率,明显的对比度等优点。而且,由于使用较早,有丰富的软件支持,并早已应用于军事和许多领域的 CAD。但因其价格昂贵,故一直未能广泛普及。

4. 光栅扫描显示器

光栅扫描显示器的基本结构如图 2-16 所示,控制器部分结构见图 2-17。

图 1-16　光栅扫描显示器基本结构

图 2-17　光栅扫描显示器控制器原理图

这种显示器的 CRT 屏面可分为 m 行扫描线,每一行又可分为 n 个小点。这样,整个屏面就被分成 m× n 个小点,我们称为象素(pixel)。显示屏上的每个象素都对应帧缓冲存

储器中的若干位,最简单的黑白图象每个象素只需要一位。若该位为 0,表示该象素为暗;若该位为 1,表示该象素为亮(反过来也可)。这样的图象我们称为 MASK 图象,或二值图象。如果每个象素用 i 位表示它的灰度,也即有 i 个位平面的帧缓存,那么就能产生 2^i 级灰度等级或颜色种类。也就是说,光栅扫描显示器的帧缓存中,存放的不是显示指令,而是对应各个象素的亮度或色彩信息,这种信息常常称为位图(Bitmap)。

计算机将要显示的图形、图象转化为位图,经过接口电路送入帧缓存,而图形控制器控制电子束依照固定的扫描线和规定的扫描顺序,自上而下,从左到右扫描整个屏面。与此同时,把一帧画面中每个象素的值从帧缓存中读出。读出时,帧缓存的地址码的生成要与光栅扫描同步,每读出一个单元,电子束恰好扫过一个象素。读出的值可控制电子束的能量大小,并决定象素的亮度。每当扫描完一帧图象时,显示控制器向计算机申请中断,使计算机能利用帧回扫的时间(大约 1.3ms)去修改帧缓存中的内容,以实现显示画面的修改。

为了得到稳定的画面,光栅扫描显示器每秒也要刷新 30 次或 60 次。也即在 1/30—1/60 秒内,要把一帧画面的每个象素的值从帧缓存中取出,不管多简单的图形,每次都要扫遍全帧。为此,必须要有高速大容量的存储器。例如,对于一台中分辨率的显示器,具有 640×480 个象素,若每秒刷新 60 次,则存取一个象素的时间为 1/(640×480×60) = 54ns。而一般对于典型的 RAM(随机存储器)片子,其存取周期为 200ns。因此,控制器必须在一个存取周期中,访问多个象素值(比如,能取 16 个象素值的控制器),当然,最终应有适应光栅显示器需要的 RAM 结构。

很多光栅扫描显示器还采用电视系统的隔行扫描方式。隔行扫描把一帧完整的画面分为两场显示,第一场含偶数扫描线,第二场含奇数扫描线。刷新周期也分为两部分,若每一场用 1/60 秒,则显示一帧画面为 1/30 秒。这样,计算机在每一场回扫期间都可以利用中断修改帧缓存的内容,而画面的信息量并没有减少,因为两场合为一帧显示,从而保证了图象的质量。由于画面的更新仍为 60Hz,因此降低了闪烁效应。同时,因为在每一场的 1/60 秒内,从帧缓存中读出的信息量比逐行扫描降低一半,因而可降低对帧缓存存取速度及设备线路通频带的要求,从而使设备的复杂程度及成本都大大降低。

对彩色显示器要分别控制三个原色:红、绿、蓝。为了使三原色按不同的比例合成各种色彩,每种原色也要有不同的灰度,如果每象素的各原色要有 256 种灰度,则每一个原色就要在帧缓存单元中占据 8 位,因此帧缓存每个单元就要有 24 位,于是帧缓存就很大,相应的价格就比较高。为此,我们可以采用彩色表技术,也即,使帧缓存每个单元的位数不增加,却能具有在很大范围内挑选颜色的能力。如图 12-18 所示,此时由帧缓存中读出来的值并不是相应象素的彩色值,而是彩色表中的一个编号(入口地址),按这个编号在彩色表中取出的数才是该象素的彩色值。

实际应用中可以根据需要,灵活地建立各种合适的彩色表,虽然彩色表中颜色的种类受帧缓存单元位数的限制,但它的内容可以自由设置,显示器能显示的颜色都可被彩色表选用。虽然对一帧图象来说,所用的彩色种类没有增加,但可方便地修改彩色表,使广泛的色彩出现在各帧画面中,因此,总的来说扩大了色彩的范围。

光栅扫描显示器最突出的优点是:它不仅可以显示物体的轮廓线、特征线等所谓的

图 2-18　有颜色查找机构的光栅显示器

线框图形,而且由于其每一象素点的灰度或色彩可以控制,因而可以显示被多种灰度和色调的象素所填充的所谓面图形。这就使得输出具有真实感的立体图形成为可能。另外,它可以和电视兼容,价格远低于随机扫描显示器。自 80 年代以来,得到了迅速发展,其应用领域也不断扩大,已成为显示器市场的主流。光栅扫描显示器的缺点是:从应用程序中将图形的描述转换成帧缓存中象素信息的过程——即扫描转换——比较费时,相应的软件比较复杂。另外,在显示斜线时,还存在线条边缘的阶梯效应,解决起来比较麻烦。

5. 显示处理机(图形处理器)

在图形硬件系统中,为了减轻主机负担,加快图形处理速度,一般都有两个以上的处理部件。除了中央处理机(CPU)以外,还有一个专用的显示处理机(DPU),用来与 CPU 交互作用和控制显示设备的操作,如图 2-19 所示。

图 2-19　图形系统功能简图

(1) 随机扫描系统的 DPU

图形系统中的 DPU 组成差别很大,不同规模的画线显示器决定了它们的 DPU 的复杂程度。最简单的一种 DPU 如图 2-20 所示。

这种 DPU 可以设有缓冲存储器,也可以不设。不设缓存时,所用的显示文件要存在主机内存储器的某个区域内,由主机 CPU 运行程序,形成 DPU 的显示文件,并由主机 CPU 把显示文件起始地址送入 DPU 的指令计数器。DPU 按这个起始地址从内存中依次读出显示指令,并把它送入指令寄存器中,然后对操作码译码,在控制逻辑的参与下执行指令。DPU 的控制逻辑还要给它的指令计数器作加‘1’操作,以便顺序取出下一条显示指

图 2-20　最简单的 DPU

令。这种 DPU 只能处理下列几种显示指令:

· 取 x 坐标数据到 x 寄存器;

· 取 y 坐标数据到 y 寄存器, 并显示(x, y)点;

· 转移指令。

图 2-21 所示的 DPU 具有缓冲存储器, 它增加了加(减)法器及其它专用硬件寄存器, 因而功能增强了。

(2) 光栅扫描系统的 DPU

目前低档的图形系统大多采用前面所述的简单的光栅扫描系统, 其图形显示所需的扫描转换是靠 CPU 直接向帧缓存写入每个象素点的数据来实现的, 即由 CPU 计算出每个象素点坐标所对应的帧缓存地址, 并赋以亮度或颜色值。其每一步虽然很简单, 但是要重复很多次。而且, 这种结构随着取址能力或显示器刷新速率的增加, 视频控制器对存储器的存取次数也增加, 这就降低了 CPU 的存取周期数, CPU 的速度就下降了。具有独立显示处理机的光栅扫描系统可克服上述缺陷。

光栅扫描系统中的显示处理机专门用来将输出图素扫描转换成象素位图, 同时执行一些诸如象素或象素块的移位、拷贝、修改等光栅操作。如图 2-22 所示, 具有独立显示处理机的光栅扫描系统有三个存储器:

· 系统存储器—— 保存数据及在 CPU 上执行的程序, 如应用程序、图形子程序包及操作系统等;

· 显示处理机存储器—— 保存数据及扫描转换和光栅操作的程序;

· 帧缓冲存储器—— 保存由扫描转换和光栅操作建立起来的能显示的图象。

简单的显示处理机只执行某些有可能实现的与图形有关的操作; 而较强功能的显示处理机则可实现裁剪、窗口一视图变换, 还有与拾取有关的逻辑及当拾取到某一图素时的反馈等交互操作。不少显示处理机具备局部段存储器, 也称显示表存储器, 显示指令按段名分类存于显示处理机的存储器中。显示处理机利用这些被保存的段可以进行变换及重画, 可以把段拖到新的位置, 还可对段进行创建、删除、编辑及使其可见或不可见等操作。

图 2-21　一种实用的 DPU 结构

图 2-22 所示的光栅扫描系统是通过系统总线上的 I/O 通道,在系统与帧缓存之间传输图象,但是这种传输对于诸如动画、拖动以及窗口、菜单的隐匿、弹出等实时操作实在太慢。在图 2-23 所示的具有单地址空间(SAS)的光栅扫描系统结构中,显示处理机、CPU 及视频控制器都与系统总线相连,因此都能访问系统内存。但更通用的 SAS 光栅扫描系统结构示于图 2-24 中,系统存储器所提供的那部分可以直接被视频控制器访问,而不必始终占用系统总线。

(3) 图形处理器的各个发展阶段。

① 第一代单片图形处理器

1984 年,日立公司率先推出了第一代单片图形处理器 HD—63484,并称之为高级 CRT 控制器。它是在该公司原先的 CRT 控制器 6845 基础上增加了一个专用画图处理器,利用硬逻辑来解释 CPU 送来的画图命令,然后产生出所需的各种几何图形。63484 首

图 2-22　具有独立显示处理机的光栅扫描系统

图 2-23　单地址空间光栅扫描系统结构

次实现了位组传输功能和硬件开窗功能,进一步减轻了 CPU 的图形显示负担。

1986 年,德克萨斯仪器公司推出了图形系统处理器 TMS34010,1987 年大批投产。这是第一种完全可编程的图形处理器。它除了能执行专用的画图指令外,还能执行一套精简的通用指令。

在 TMS34010 问世之后不久,英特尔公司推出了图形协处理器 82786。它采取了另一种设计思想,其中有两个由微码控制的高性能处理器:图形处理器和显示处理器,且由 CPU 提供的两种表结构来驱动。图形处理器接受的表结构是各种画图命令和数据操作命令的集合;显示处理器接受的表结构是各种显示命令和开窗命令的集合。82786 也采用硬件开窗技术,且比 63484 更为灵活。

图 2-24　更通用的单地址空间光栅扫描系统结构

② 第二代单片图形处理器

日本电气公司推出的 72120 是第一种第二代单片图形处理器,其中除了图形和显示处理器外,还有一个预处理器,以提高速度。它有一套很丰富的画图命令还可进行各种变换。另外,它的位组传输能力可以使开窗性能达到 82786 的水平。它不仅能满足普通显示的需要,还能满足图象显示的需要。

德克萨斯仪器公司于 1988 年推出的 34020 其性能五倍于原先的 34010。

日立公司的 GDP 也是 1988 年推出的,被认为是功能最强的单片图形处理器。它含有定时、画图和显示三个处理器,都由微码进行控制,其性能达到了高档二维图形工作站的水平。

③ 多片图形处理器

AMD 公司推出的 9560 四象点数据流管理器(QPDM),每个 9560 可以处理 4 个位平面的帧缓冲存储器,有利于提高速度和增加灵活性。

美国国家半导体公司推出了包括十几种芯片的图形处理器。这套芯片叫做高级图形芯片组(ADCS),除了视频时钟发生器、视频移位寄存器和视频 RAM 控制器之外,还有 DP8510/1 位组传输处理器和 DP8500 光栅图形处理器(RGP)。DP8500 很像现在生产的单片图形处理器,其中高性能画图处理器能执行硬逻辑的画图例行程序和用户直接编程的通用处理指令。

④ 通用微处进器用作图形处理器。

有的图形系统生产厂家目前仍然喜欢用通用微处理器来代替图形处理器。这种做法的一个好处是系统 CPU 和专用于图形显示的从处理器有一个共同的编程环境。如福兰第公司的下一代 VARS 图形控制器是利用 AMD 公司的 29000 精减指令(RISC)微处理器做成的。泰克公司决定把莫托洛拉公司的 88000RISC 微处理器作为它今后的图示系统的标准处理器。

⑤ 流水线多处理器结构的图形机

从原理上说,代替单片图形处理器的最佳方案是流水线多处理器图形机(Graphics

Engine)。这是因为图形处理和显示的整个过程本身具有流水线的性质。具体来说,每个高级图示命令都需要经过逐步进行几何变换的过程,最后才形成位图形式的输出格式。如果这整个流水线的每一步都由一个处理器去完成,图形处理和显示的整体速度就必然很高。这是上述其它方法所无法比拟的。

典型的流水线多处理器图形机包括三个独立的处理器:显示表处理器或命令处理器、几何处理器以及显示控制器或显示处理器。但在实际的高档系统中,处理器的数目远不止三个,流水线上的功能分得很细,每个功能由一个微处理器去承担。

比如 Silicon Graphics 公司推出的 IRIS 4D 超级工作站使用的图形机包括 38 个定制和半定制处理器,它们由 MIPS 公司生产的 RISC 通用微处理器进行统一控制。泰克公司最新的 50 万向量/秒的三维工作站也是利用图形机做成的,包括用于整体控制的通用 68020 微机处理器和三级流水线的门阵列,每个门阵列里也采用流水线结构。

图形机的性能远高于单片图形处理器。如德克萨斯仪器公司的 88 位片处理器速度为 20MIPS,由 9 个芯片所组成的简单三级流水线的图示性能比单片图形处理器 34010 高出 100 多倍。单片图形处理器可以被组织到图形机中,如 34010 配上得克萨斯仪器公司的 8837 浮点处理器之后,其速度是单个 34010 的 10 倍至 50 倍。日立公司设计的多处理器图形板使用 H32 通用微处理器作为整体控制器,用一个或几个 FPU32 浮点处理器来进行坐标交换和裁剪,用 GDP 作为画图处理器和定时控制器。尽管处理器数目还不太多,但比图形板性能已很惊人:在使用一个浮点处理器时为 4 万向量/秒,在使用多个浮点处理器时达到 9 万至 20 万向量/秒。

⑥ 阵列结构的图示系统

为了适应图形处理与显示的需要,因莫斯公司推出了最新的多端口单片机 T800。该芯片含有 10MIPS 的 RISC 处理器和 1.5MFLOPS 的浮点处理器,还有 4K 字节的静态随机存储器。它的指令系统中增加了许多适于图表处理的高效指令。因此,T800 可作为独立的图形处理器来使用。

多端口单片机的真正实力在于大量的这种芯片可以组成高度并行的处理阵列。如何把这种阵列应用于图示系统,是对图示系统设计师的新挑战。这是因为并行处理阵列的工作方式显然与现有所有图形系统的流水线工作方式完全不同。

二、硬拷贝设备

1. 点阵式打印机(Dot-matrix Printer)

点阵式打印机的工作原理是利用打印头将成型字符通过色带印在纸上。打印头通常有 7～24 针,打印头每次相对纸走一步,纸向前走行。因此,这类打印机是光栅输出设备,需要有扫描转换事先把矢量图象转换成打印机用信号。点阵式打印机的可取址能力可以不受打印机头上针间距的限制,因为可以用两列针使其在垂直方向上补上半个针间距,以提高分辨率,如图 2-25 所示。利用彩色色带可以产生彩色硬拷贝。

2. 笔式绘图仪(Pen Plotter)

笔式绘图仪是矢量型设备,绘图笔相对纸作随机移动。在笔式绘图仪上,一个电脉冲通过驱动电机与传动机构使画笔移动的距离称为步距,或称脉冲当量。步距愈小,画出的

图就愈精细。一般国产绘图仪的步距为 0.1～0.00625mm,国外绘图仪的步距有的可达 0.001mm。在实际应用中,0.1mm 的步距可以满足一般图形的要求;0.05mm 的步距可以使人的肉眼觉察不出图形的阶梯状波动;而 0.00625mm 的步距可以满足一般精密绘图的需要。

图 2-25　用两列针的点阵式打印机头

图 2-26　滚筒式绘图仪结构示意图

笔式绘图仪可以分为滚筒式和平板式两种。

(1) 滚筒式绘图仪

滚筒式绘图仪(Drum Plotter)是用两个电机分别带动绘图纸和绘图笔沿 X,Y 轴向运动。一种滚筒式绘图仪是把有孔的纸固定在有突出针的滚筒上,滚筒可向前向后转动,如图 2-26 所示。另一种称为桌面绘图仪(desk-top),纸沿某一轴向在某一范围内前后运动,笔相对纸作另一方向运动,见图 2-27。

图 2-27　桌面(desk-top)绘图仪

滚筒式绘图仪的结构比较简单,价格相对便宜一些,但是精确度和速度不可能太高。

(2) 平板式绘图仪

这种平板式绘图仪(Flatbed Plotter)的绘图笔在纸上可作 X, Y 两个方向的运动,绘图纸可以用静电方式、真空吸附或直接铺在绘图机平板上。

机械传动的平板式绘图仪,驱动装置一般为钢丝绳或齿轮条箱,在 X, Y 两个方向上用电机拖动笔架运动。这种绘图仪速度低,一般每分钟仅 7—15m,而且精度低,寿命短,价格相对便宜。其结构示意图见图 2-28。

图 2-28　机械传动绘图仪结构示意图

具有平面电机的平板式绘图仪采用两轴同时驱动的单向脉冲电机。动子重量比较轻,与定子之间采用空气轴承,所以速度快,每分钟可达 120m。这种绘图仪的精度较高,而且维修方便,但价格比较昂贵,如 XYNETICS 绘图仪精度为 0.025mm,能绘 A0 图纸,其结构示意图如图 2-29 所示。

图 2-29　平面电机式绘图仪结构示意图

笔式绘图仪一般都有一个微处理机,用以接收诸如“画线”,“定位”,“画圆”,“画字符”,“设置线型”,及“选笔”等命令。微处理机把输出图素分解成横平、竖直及对角线等八个任意方向的增量运动,位置传感器和伺服电机反馈系统执行该运动命令,同时利用电磁原理抬笔或落笔。

绘图速度取决于绘图笔移动的加速度和速率。这里,加速度与绘图机笔头的质量有关。

3. 静电绘图仪

静电绘图仪(Electrostatic Plotter)是打印机与笔式绘图仪的结合,它的运动部分很少,只有供纸和调色盒是机械驱动,其余都是电子线路。它的原理是:事先使白纸或黑纸上带有负电荷,而吸有调色剂的针尖带有正电荷,当由程序控制的电压按阵列式输出并选中某针尖时,就将调色剂附着到纸上,产生极小的静电点。

静电绘图仪能输出具有明暗度的面图形,分辨率较高,可达每毫米 4～8 个点,其速度比笔式绘图仪的速度高,是高性能打印机速度的两倍。这类绘图仪运行可靠,噪声小,但是线条有锯齿状,且用纸特殊而价格昂贵。其结构如图 2-30 所示。

图 2-30　静电绘图仪结构示意图

4. 激光打印机

激光打印机(Laser Printer)有一个涂有硒的、能旋转的鼓,且上面带有正电荷。由信号控制的激光束被扫描到鼓上,激光束射到的部分失去电荷,而其余部分仍保留正电荷。带有负电荷的调色剂则粘附到鼓的正电荷区,从而形成黑色空拷贝。上述过程重复三次,每次一种基色,将会产生彩色拷贝。单色激光打印机部分结构见图 2-31 所示。激光打印机可以有一微处理机,进行扫描转换及控制打印机。

5. 喷墨绘图仪

喷墨绘图仪(Ink-jet Plotter)的喷墨装置多数情况是安装在类似打印机的机头上,纸则绕在滚筒上并使之快速旋转,喷墨头则在滚筒上缓慢运动,并且把青色、品红、黄色,有时是黑色墨喷到纸上。所有颜色是同时附在纸上,这与激光打印机及静电绘图仪不同。某些喷墨绘图仪可以接收视频及数字信号,因此可用于光栅显示屏幕的硬拷贝,此时图象分辨率受到视频输入分辨率的限制。旋转式喷墨绘图仪的结构示意图见图 2-32。

6. 热转换打印机

热转换打印机(Thermal-transfer Printer)是另一种光栅式硬拷贝设备,其原理与静电绘图仪类似。信号有选择地加热一些非常细的、相互分离的热管尖上(一般每英寸有

图 2-31　激光打印机结构示意图　　图 2-32　旋转式喷墨绘图仪结构图

200 个),然后把蜡纸上的颜色转换到绘图纸上。蜡纸和绘图纸是相互压在一起的,且同时覆盖在热管尖的带条上。对于彩色打印输出,蜡纸分别是一卷卷彩色条带,每一卷长度与绘图纸相同。由于管针加热与冷却速度很快,所以可以在一分钟内得到一幅彩色硬拷贝图片。某些热转换打印机也接收视频信号及数字化位图输入,从而可以产生视频图象的硬拷贝。

热化染色转换打印机(Thermal sublimation dye transfer printer)与热转换打印机工作原理类似,只是这种热化和染色转换,每种颜色可以有 256 亮度等级,从而可得到高质量全彩色图象,分辨率可达每英寸 200 个点。转换过程比蜡纸转换要慢,但质量接近照相。

7. 摄象机

摄象机(Camera)是另一种硬拷贝设备,它拍下显示在 CRT 屏幕上的图象。可以有两种基本技术得到彩色胶片。一种是摄象机直接从彩色 CRT 上得到彩色图象,其图象质量受到影孔板及光栅扫描的限制。另一种是通过彩色滤波器拍摄黑白 CRT 上的图象,按序显示图象的不同颜色分量。这种技术可产生很高质量的光栅或矢量图象。图 2-33 表示的是用彩色滤波器产生彩色图象的摄象机结构图。

图 2-33　采用彩色滤波器的摄象机结构

输入到摄象机中的信号可以是光栅视频信号,或是一张位图,或是矢量类型结构的信号。不论是可直接驱动彩色 CRT 的视频信号,还是红、绿、蓝成分的单色信号,均可通过滤波器分时显示。在任意情况下,在整个记录周期,视频信号必须是常数。对于低灵敏度的胶片,周期可达 1 分钟。高速、高分辨率的位图或矢量系统价格很贵,因为 CRT 和电子

线路本身必须仔细地设计和校准。随着速度和分辨率的降低,成本也急剧地下降。

上述各硬拷贝设备的性能比较见表 2-2 及表 2-3。

表 2-2　单色硬拷贝设备性能比较表

性能项目	笔式绘图仪	点阵打印机	静电绘图仪	激光打印机	喷墨绘图仪	热转换打印机	摄象机
每点亮度等级	2	2	2	2	≥2	2	很高
可寻址能力(每英寸点数)	1000	≤250	≤400	≤1500	≤200	≤200	≤800
点的大小(× 10^{-3}英寸)	6—15	10—18	8	5	8—20	7—10	6—20
相对成本幅度	低—中等	很低—低	中等	中等—高	低—中等	低—中等	低—高
每幅图象相对成本	低	很低	中等	中等	低	中等	高
图象质量	低—中等	低—中等	中等	高	中等	中等	高
速度	低	中等	高	中等—高	中等	中等	低

表 2-3　彩色硬拷贝设备性能比较表

性能项目	笔式绘图仪	点阵打印机	静电绘图仪	激光打印机	喷墨绘图仪	热转换打印机	摄象机
每点颜色等级	≤16	8	8	8	≥8	≥8	很多
可寻址能力(每英寸点数)	1000	≤250	≤400	≤1500	≤200	≤200	≤800
点的大小(× 10^{-3}英寸)	15—6	18—10	8	5	20—8	10—7	20—6
相对成本幅度	低—中等	很低	中等—高	中等—高	低—中等	中等	中等—高
每幅图象相对成本	低	很低	中等	中等	低	中等	高
图象质量	低—中等	低	中等	高	中等	中等—高	中等—高
速度	低	低—中等	中等	中等	中等	低—中等	低

三、输入设备

图形输入设备可将用户的图形数据及各种命令等转换成电信号,并传递给计算机。从逻辑上看,可分为六种功能,即定位(locator)、笔划(stroke)、送值(valuator)、选择(choice)、拾取(pick)及字符串(string),也可称为六种逻辑设备。这里所谓逻辑设备,是指按逻辑功能定义的设备,并非具体的物理设备。一种逻辑设备对应于一种或一类特定的物理设备,而实际的物理设备又往往是某些逻辑输入功能的组合,详见表 2-4。

表 2-4　输入设备功能分类

逻辑功能	相应典型物理设备	基本功能
定位	鼠标器、姆指轮、图形输入板等	输入一个点坐标
笔划	鼠标器、姆指轮、图形输入板等	输入一系列点坐标
送值	数字键盘	输入一个整数或实数
选择	光笔选菜单项、功能键	根据一个正整数得到某一种选择
拾取	光笔	拾取一个显示着的图素
字符串	文字键盘	输入一串字符

1. 定位器

定位器(Locator)用以指示一个位置,其输入量是 x,y 坐标。

(1) 坐标数字化仪(digitizer)。它是常见的定位设备,其中全电子式坐标数字化仪由于精度高,使用方便,得到普遍应用。这种设备利用电磁感应原理:在台板的 x,y 方向上有许多平行的印刷线,每隔 200 μm 一条。游标中装有一个线圈,当线圈中通有交流信号时,十字叉丝的中心便产生一个电磁场,当游标在台板上运动时,台板下的印制线上就会产生感应电流。印制板周围的多路开关等线路可以检测出最大信号的位置,即十字叉线中心所在的位置,从而得到该点的坐标值。全电子式坐标数字化仪外形图如图 2-34 所示。

(2) 图形输入板

图形输入板(Tablet)这是一种在结构和原理上都类似于坐标数字化仪的输入设备,只是面积较小而已,常见的面积为 280mm× 280mm。这种设备配有特制的指示笔(Stylus),操作时,靠笔上的压力敏感开关接通电路,输入所在位置。有的输入板所配的手动指示器(hard cursor)上有几个按键,可用来输入命令和接受坐标(见图 2-35)。

图 2-34　全电子式坐标数字化仪

图 2-35　带有按键的输入板

(3) 手动盒亦称鼠标器(mouse)。常用的一种鼠标器是机械式的,见图 2-36,这是一个手持的底部有轮子的设备,当其在平面上移动时,轮子就随之转动,而连接在轮子上的电位器可在两个正交的方向上检测出这一相对运动。另一种手动盒是基于电磁感应原理

的,就像坐标数字化仪的游标一样,当手动盒在布满印制线的台板上运动时,就可以根据计数脉冲检测出相对位置。目前还有一种鼠标器,采用了红外遥控技术,如日本产的WACOM WT-400数字化仪上配置的鼠标器就是这种结构。鼠标器上一般都有几个按键可用来输入命令或其它信息。

图 2-36　手动盒(鼠标器)

(4) 其它定位设备。

跟踪球(track ball)(有时也叫水晶球(crystal ball))、操纵杆(Joystick)等,都是将机械运动转换为电位器输出,再转换成数字量的定位设备。控制开关是操纵杆的一种变形,可以作八个方向运动。接触控制板(touch panel)是将透明的控制板安装在 CRT 表面上,当用户用手指触及它时,所触及的位置就被检测到了。声学输入板(sonic tablet)是利用声音将指示笔和输入板两边的两个条形话筒结合起来实现定位的,如图 2-37 所示,接触定位时,在笔尖上产生一个电火花,只要测出到达每个话筒所需的时间,即可知道该点的相对位置。三维声笔(sonic pen)则是利用三个相互垂直的条形话筒来确定指示笔所在的空间位置。

图 2-37　二维声学输入板

2. 拾取器

拾取器(pick)用以拾取显示屏幕上的一个形体、图组或图素。

光笔是唯一的一种本来就具有拾取功能的设备,其原理图示于图 2-38。光笔的输出一般接到显示处理机(DPU)的控制逻辑上,当光笔工作时,DPU 就转去执行用来显示检测到的输出图素的指令。光笔的视野可以通过在透镜前放置不同大小的光圈来控制,或者通过调整透镜的焦距来控制。光笔上的开关可以控制其工作或不工作。有些光笔把光电倍增管放在它的圆柱形笔杆里,从而节省了光导纤维。由于光笔的使用不太方便,现在越来越多的系统采用图形输入板或手动盒来模拟光笔的拾取功能。

图 2-38　光笔原理图

3. 定值器

定值器(valuator)这是提供标量值的设备。多数定值器采用旋转式电位器(刻度盘),这特别适合于用来控制目标的旋转。另一些定值器采用滑线式电位器,它用线性位移代替旋转,适合于给定非角度的数值。

4. 键盘

键盘(keyboard)用来输入字符或字符串,典型的设备是字母数字键盘。

5. 按键

按键(button)它可从一组动作或功能中作出选择。常用的设备是已编程的功能键盘(PFK)。也有的用弦键盘(chord keyboard)来实现,它由若干形如薄琴键的按键组成,当同时按下某几个键时即可组成不同的功能。

6. 其它设备

如语音识别器,可以用来输入语音命令及数值等。

第三节　图形软件系统

一、图形软件的组成

图形软件系统应该具有良好的结构,要有合理的层次结构和模块结构。应把整个图形软件分为若干层次,每一层又分为若干模块,使得整个系统容易设计、调试和维护,便于扩充和移植。图 2-39 表示出图形软件的各层次。

(1) 零级图形软件

是最底层的软件,主要解决图形设备与主机的通讯、接口等问题,又称设备驱动程序,是一些最基本的输入、输出子程序。由于使用频繁,程序质量要求尽可能高,因此常用汇编语言,甚至机器语言,或按近机器语言的高级语言编写。零级图形软件是面向系统的,而不是面向用户的。

(2) 一级图形软件

又称基本子程序,包括生成基本图形元素、对设备进行管理的各程序模块。它可以用汇编语言编写,也可以用高级语言编写,要从程序的效率与容易编写、调试、移植等要求全面考虑。一级图形软件既面向系统又面向用户。

图 2-39　图形软件层次

(3) 二级图形软件

也称功能子程序,是在一级图形软件基础上编制的。其主要任务是建立图形数据结构(图形档案),定义、修改和输出图形;以及建立各图形设备之间的联系,要具有较强的交互功能。二级图形软件是面向用户的,要求使用方便、容易阅读,便于维护和移植。

(4) 三级图形软件

这是为解决某种应用问题的图形软件,是整个应用软件的一部分。通常由用户编写或系统设计者与用户一起编写。

一般把零到二级图形软件称为基本图形软件,或称支撑软件,而把三级或三级以上图形软件称为应用图形软件。

二、基本图形软件

1. 基本图形软件的内容

基本图形软件作为图形系统的支撑软件,其功能可根据需要而有所不同,但其基本内容一般应包括:

- 系统管理程序;
- 定义和输出基本图素及复合图素图形的程序;
- 图形变换,包括几何变换、开窗、裁剪等程序;

· 实时输入处理程序;

· 交互处理程序

2. 建立基本图形软件的方法

通常可采用以下三种方法来建立基本图形软件:

(1) 图形程序包

这是以某种高级语言为基础,加上扩充处理图形功能的子程序包。这种方法实现起来比较容易,一般不用或很少修改原来高级语言的编译程序,容易调试,便于修改及扩充。但是由于一般高级语言,如 FORTRAN, BASIC, PASCAL 等并不是为处理图形而设计的,因此,用其来处理图形,总有不合适之处。

(2) 修改高级语言

这是在某种高级语言基础上,经修改其编译系统实现的,一般是修改其编译系统,扩充一些处理图形的语言和数据类型,改变原高级语言某些不适应图形处理的缺点。但是,修改某语言的编译系统并非易事,而且难以彻底解决高级语言不适应图形处理的问题。

(3) 专用高级图形语言

这是一种从语句、数据结构到输入、输出等各方面都按照处理图形的需要来设计的一种高级图形语言,它必须有自己独立的编译器。当然,设计和实现这种编译器是十分困难的,因为这既要具备一般高级语言的功能,又要扩充许多新的图形处理功能,而扩充到什么地步也难确定,因为应用领域中,图形系统要涉及的问题是各不相同的。

前两种方法都有大量可以利用的软件资源、程序库以及有关资料,这对一些高级语言已十分熟悉的人来说,要学习的新东西不多。目前,国内外通用的图形软件系统绝大多数都采用前两种方法,特别是图形软件包的方法。但是从长远的观点看,设计一种性能好的专用高级图形语言还是十分重要的。

三、图形软件与其它软件资源的联系

1. 高级语言的选择

图形程序包是以某种高级语言为基础,选择哪一种高级语言,应考虑下面几种因素:

(1) 图形处理系统主要应用于工程技术领域,因此应该选择在这些领域中较为通用的高级语言;

(2) 图形软件包是模块结构,因此最好选择模块化结构的高级语言;

(3) 图形处理的应用范围很广,需要各方面的软件支持,因此要选择支持性软件(如数学软件包、数据库管理等)比较丰富的高级语言;

(4) 图形处理常常要采用相当复杂的数据结构,因此要选择数据类型比较灵活和丰富的高级语言;

(5) 图形处理要和许多图形外部设备打交道,因此要选择输入输出功能比较强的高级语言;

(6) 图形处理要求响应速度快,因此又要选择目标程序质量比较高的高级语言。

显然,完全满足这些要求的高级语言并不存在,这也是需要研制专门处理图形的高级语言的理由,因此,只能从目前较通用的高级语言中选择:

· BASIC　简单易学,有会话功能,应用普遍,但是计算功能较弱,一般只适用于微机图形系统。

· FORTRAN　具有模块结构,与图形软件包的联接非常方便,有丰富的支持软件,而且工程界通用,应用非常普遍。但是数据类型不够丰富,处理复杂的图形对象也不够方便。

· PASCAL　是结构化的程序设计语言,有丰富的数据结构,便于表达复杂的图形结构,但是在工程界不够普及。

· C 语言　与 PASCAL 相仿,由于接近机器语言,效率较高,有着良好的发展前景。

2. 操作系统的选择

操作系统是计算机最基本的系统软件,它的功能强弱大大影响其它系统软件功能的强弱。图形处理系统带有许多图形输入输出设备,而通常操作系统的设备管理并不管理这些设备,要管理这些设备,可有三种途径:

(1) 研制包括管理图形设备在内的新的操作系统,即图形操作系统;

(2) 修改现有操作系统,扩充它的外部设备表以及相应的设备驱动程序;

(3) 作为用户设备,由图形软件包自己管理。

当然,研制专门的图形操作系统,既是十分有意义的事,又是一种相当困难的事,所以通常都是以现有的操作系统为基础建立图形处理系统。选择的标准除一般功能的考虑外,就是要容易扩充图形设备,或者允许作为用户设备由图形软件包自己管理。当前,DOS、VMS、UNIX 是常常被用来作图形处理系统基础的操作系统,特别是 UNIX,由于灵活方便,功能齐全和易于移植、扩充,它已被越来越多的图形处理系统所采用。

四、实用图形软件包简介

1. IBM-PC 基本汇编指令驱动程序

(1) 设置屏幕状态 SETCRT(MODE,BCOL,FCOL)

其中,参变量MODE——屏幕模式,例如"04"为 320×200 图形方式;

BCOL——屏幕底色,例如"1"为蓝色;

FCOL——调色板号,例如"0"为绿/红/黄色

程序如下:

```
FRAME    STRUC
SAVEBP   DW
SAVERET  DD
FCOL     DD
COL      DD
MODE     DD
FRAME    ENDS
;
CSEG     SEGMENT   'CODE'
DGROUP   GROUP     DATA
         ASSUME    CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP
         PUBLIC    SETCRT
```

```
SETCRT    PROC     FAR
          PUSH     BP
          MOV      BP,SP
          LES      BX,[SP]+MODE
          MOV      AX,ES:[BX]
          MOV      AH,00H
          INT      10H
          LES      BX,[BP]+COL
          MOV      AX,ES:[BX]
          MOV      BX,AX
          MOV      AH,11
          MOV      BH,00H
          INT      10H
          LES      BX,[BP]+FCOL
          MOV      AX,ES:[BX]
          MOV      BX,AX
          MOV      AH,11
          MOV      BH,01H
          INT      10H
          POP      BP
          RET      12
SETCRT    ENDP
;
CSEG      ENDS
          END
```

(2) **绘制一个点** POINT(INY,INX,COL)

{即在屏幕的某行、某列位置上,画具有某一颜色值的点}

参变量: INY——**点的** Y **坐标,即行号**;

INX——**点的** X **坐标,即列号**;

COL——**点的颜色值**。

程序

```
FRAME     STRUC
SAVEBP    DW
SAVERET   DD
COL       DD
INX       DD
INY       DD
FRAME     ENDS
;
CSEG      SEGMENT   'CODE'
DGROUP    GROUP     DATA
          ASSUME    CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP
POINT     PROC      FAR
          PUBLIC    POINT
          PUSH      BP
          MOV       BP,SP
```

```
        LES     BX,[BP]+ INX
        MOV     DX,ES:[BX]
        LES     BX,[BP]+ INY
        MOV     CX,ES:[BX]
        LES     BX,[BP]+ COL
        MOV     AX,ES:[BX]
        MOV     AH,0CH,AX
        INT     10H
        POP     BP
        RET     12
POINT   ENDP
CSEG    ENDS
        END
```

(3) **设置屏幕为字符方式** REDOS

REDOS **功能是当绘图结束时,接受键盘字符,改变屏幕状态为黑底白字字符方式。**

程序:

```
CSEG    SEGMENT     'CODE'
DGROUP  GROUP       DATA
        ASSUME      CS:CSEG,DS:DGROUP,ES:DGROUP,SS:DGROUP
REDOS   PROC        FRA
        PUBLIC      REDOS
        PUSH        BP
        MOV         BP,SP
        MOV         AH,11
        MOV         BH,01
        MOV         BL,00
        INT         10H
        MOV         AH,1
        INT         21H
        MOV         AH,0
        MOV         AL,02
        INT         10H
        POP         BP
        RET
REDOS   ENDP
CSEG    ENDS
        END
```

2. Microsoft C/C^{++} 7.0 **运行库图形例程**

Microsoft **运行库中有** 550 **多个函数和宏定义,其所包含的一系列图形例程提供了各种不同的图形函数,即: 低级图形基本函数、字形函数和表示图形(显示曲线图形、饼图等)函数。图形函数是由** GRAPHICS.LIB **和** PGCHART.LIB **两个库提供的,这两个库必须显式地与用户程序连接。** GRAPHICS.LIB **库提供了对低级图形和字符字形例程的支持;** PGCHART.LIB **库则支持表示图形例程。**

低级图形和字形函数是在头文件 GRAPH.H **中声明的。根据生成和处理图形对象所**

涉及的任务不同,而将该库分为 8 类,如下表所示:

类　别	任　务
配置方式和环境	为硬件选择合适的显示方式,并建立输出图象和显示图象的内存区域
设置坐标	指定屏幕上的逻辑原点及活动显示区域
设置低级图形调色板	指定低级图形例程的调色板映象
设置属性值	指定低级图形例程前景、背景色、填充标志以及线型
产生图形输出	画图并加以填充
产生文本输出	在屏幕上显示文本
传送图象	将图象存于内存中,并进行检索
显示字形	用与 Microsoft Windows 兼容的字符字形显示文本

下面进一步列出附录中上机实验题所涉及到的各类函数。详细解释请参见附录 B 中 12# 参考文献。

① 配置方式和环境

该类例程用于建立图形或文本的操作方式,确定当前环境以及控制光标显示。

例　程	作　用
- clearscreen	清屏后用当前背景色填充屏幕
- getactivepage	取当前活动页号
- getbkcolor	获取当前背景色
- getvideoconfig	获取当前图形环境的状态
- getvisualpage	获取当前可视页数
- grstatus	获取最新图形函数调用的状态
- setactivepage	设置用于输出图象的活动页所占用的内存区域
- setbkcolor	设置当前的背景色
- settextrows	设置文本行的行数
- setvideomode	选择显示屏幕的操作方式
- setvideomoderows	设置显示方式及用于文本操作的行数
- setvisualpage	为当前可视页设置内存区域

② 设置坐标

此类例程设置当前文本或图形位置,并在不同图形坐标系之间进行象素坐标转换。Microsoft 图形函数可以识别三类坐标:

· 定点物理坐标

· 应用程序定义的观察坐标

· 包括浮点值的窗口坐标

该类函数的作用是建立窗口和观察坐标系,并且在物理坐标系、观察坐标系和窗口坐标系之间进行坐标转换。

缺省的观察坐标系与物理屏幕坐标系是一致的,物理坐标系的原点(0, 0)总是在显示

的左上角位置,x 轴正向是从左到右的方向,y 轴正向是自上而下。物理上的垂直和水平方向依赖于硬件的显示配置和选用的方式。

例　　程	作　　用
- getcurrentposition	确定观察坐标系中的当前位置
- getcurrentposition- w	确定窗口坐标系中的当前位置
- getphyscoord	将观察坐标转换为物理坐标
- getviewcoord	将物理坐标转换为观察坐标
- getviewcoord- w	将窗口坐标转换为观察坐标
- getviewcoord- wxy	将- wxycoord 结构表示的窗口坐标转换成观察坐标
- getwindowcoord	将观察坐标转换为窗口坐标
- setcliprgn	将图形输出限定在屏幕上指定区域
- setvieworg	确定观察坐标系的原点位置
- setviewport	将图形输出确定在屏幕上指定区域,并且将观察坐标系原点定位于该区域的左上角坐标
- setwindow	定义浮点窗口坐标系统

③ 设置低级图形调色板

低级调色板例程用来选择或重新变换调色板颜色。

例　　程	作　　用
- remapallpalette	改变当前调色板中所有颜色指数
- remappalette	改变当前调色板中单个颜色指数
- selectpalette	选择预定义的调色板

④ 设置属性值

用于画线、弧、椭圆和其它基本图形的低级输出函数并不指定颜色或线型信息,相反,低级图形函数使用由下列函数独立设置的属性值。

例　　程	作　　用
- getarcinfo	确定最新绘制的弧或饼状图终止点的视口坐标
- getcolor	获取当前颜色属性
- getfillmask	获取当前填充掩码
- getlinestyle	给出当前线型
- getwritemode	给出当前逻辑画线方式
- setcolor	设置当前颜色属性
- setfillmask	设置当前填充掩码
- setlinestyle	设置当前线型
- setwritemode	设置逻辑画线方式

⑤ 产生图形输出

图形输出函数使用指定的坐标系列,画出不同的图形。该类函数一般使用当前或缺省的线型、填充掩码、写入方式、背景色及前景色等属性。

每个函数的名字都显示了其任务或所绘图形,如下表所示:

例　　程	作　　用
_arc,_arc_w,_arc_wxy	画圆弧
_ellipse,_ellipse_w,ellipse_wxy	画椭圆或圆
_floodfill,_floodfill_w	用当前颜色填充一定的屏幕区域
_getcurrentposition_getcurrentposition_w	给出函数_lineto 和_outgtext 使用的当前图形输出位置
_getpixel,_getpixel_w	象素颜色
_lineto,_lineto_w	用线段连接当前图形输出点和指定点
_moveto,_moveto_w	将当前图形输出点移到指定点
_pie,_pie_w,_pie_wxy	画扇形图
_polygon,_polygon_w,_polygon_wxy	画多边形或扫描填充一个多边形
_rectangle,_rectangle_w,_rectangle_wxy	画矩形或扫描填充一个矩形
_setpixel,_setpixel_w	设置象素颜色

该类例程中大多都有几种不同形式可用,这可以根据名字判断,没有后缀的输出函数使用观察性坐标系;_w 结尾的函数以双精度数为参数,且使用窗口坐标系;_wxy 结尾的函数使用结构_wxycoord 来定义坐标,也使用窗口坐标系。

像弧或椭圆那样的圆形图,是定在一个矩形范围内的,这个矩形用两个点(即矩形对角的两个点)指定矩形的中心便是图形的中心,矩形的边界决定图形的大小。

⑥ 产生文本输出

下面的例程用于图形方式和文本方式的文本输出,与标准控制 I/O 库例程不同的是,该类例程能识别文本窗口边界,并使用当前文本颜色。

例　　程	作　　用
_displaycursor	在图形例程出口设置光标开关
_gettextcolor	给出当前文本颜色
_gettextcursor	给出当前光标属性(仅用于文本方式)
_gettextposition	给出当前文本输出位置
_gettextwindow	求当前文本窗口边界
_outmem	打印内存缓冲区中一定长度的文本
_outtext	自当前文本位置起向屏幕输出文本字符串
_scrolltextwindow	将当前文本窗口上卷或下卷
_settextcolor	设置当前文本颜色
_settextcursor	设置当前光标属性(仅用于文本方式)
_settextposition	重定位当前文本位置
_settextwindows	定义当前文本显示窗
_wrapon	控制所输出文本是否绕转到下一行

例程_outtext 和_outmem 不提供格式化操作。用户希望输出整数或浮点数时,必须在调用它之前将值转换为字符串变量(通过 Sprinf 函数)

例程_outtext 可以识别\n(换行符)和\r(回车符)序列,_outmem 例程将它的作为可打印的图形字符加以处理。

⑦ 传送图象

该类函数通过应用程序分配的缓冲区,在内存和显示屏之间传送屏幕图象,还可以确定存放给定图象所需要的缓冲区的字节数。

以_w 或_wxy 结尾的函数使用窗口坐标; 其它函数使用观察坐标。

例　　程	作　　用
_getimage,_getimage_w,getimage_wxy	将屏幕图象存入内存
_imagesize,_imagesize_w,imagesize_wxy	求存放图象所需要的缓冲区大小(以字节表示)
_putimage,_putimage_w	从内存中搜索一个图象并显示它

有时,函数_getimage 存放图象需要的缓冲区会大于 64K(65,534 字节),此时用函数_halloc 分配大于 64K 的缓冲区。

⑧ 显示字形

下面所列出的函数用于控制屏幕上基于字形的字符输出。

例　　程	作　　用
_getfontinfo	获取当前字形特征
_getgtextextent	测定当前字形中指定文本的象素宽度
_getgtextvector	给出字形文本输出的位置
_outgtext	向屏幕上指定象素位置输出当前字形的文本
_registerfonts	初始化字形库
_setfont	寻找匹配一系列专门特征的单个字形,并将该字形设置为_outgtext函数使用的当前字形
_setgtexvector	设置字形文本输出的当前位置
_ungisterfonts	释放由_registerfont 分配的内存

第四节　图形软件标准

随着计算机图形学应用领域的不断扩大,各种图形软件日益增多,图形设备也是五花八门,如果没有一个软件标准,对应用软件的开发和移植等工作将造成困难。

从 70 年代起,国际上开发了很多通用的图形软件,它们共同的特点是将软件的核心部分与计算机的操作系统及显示终端分开,使前者不受后者的牵扯。在图形元素处理的流程中,只有在显示前的一步,才把要显示的图形元素用汇编语言或其它语言变成显示终端的命令,使图形能显示出来。这样做可使图形软件包在不同的计算机和图形设备之间进行

移植。如果能将图形软件包的使用方法标准化,则应用软件的移植将更方便。

为了在国际范围内开展图形标准化的工作,统一交互图形编程的某些准则,国际信息协会 IFIP 于 1974 年 8 月召开 5.2 组工作会议,讨论了这一议题。经过一系列专家会晤,逐步明确了图形软件标准的问题所在和主要措施。

美国计算机协会 ACM 的计算机图形特殊兴趣小组 SIGGRAPH 于 1974 年成立了图形标准计划委员会 GSPC,并于 1977 年提出了三维功能的图形系统"Core Systew",1978 年又对该系统作了进一步修改。

西德标准局 DIN 于 1977 年提出了 GKS 二维绘图系统。

英国标准化研究所 BSI 于 1978 年建立了工作组 OIS/5/WG5 Computer Graphics,他们对 GKS 作了审议,并提出了不少意见,他们是图形标准化的主要贡献者之一。

挪威也提出以该国的 IDIGS 图形系统作为国际标准的起点。

国际标准化组织 ISO 下属的图形工作组从 1977 年起多次开会,反复比较了 GKS 和 CORE 版本,提出了不少问题,并尽力使这两种版本相互靠拢,由于 GKS 版本修改快,接受 ISO 工作组意见多,提供讨论的文本最完整,因此,于 1982 年被 ISO 工作组通过为国际标准草案。1985 年正式批准为二维图形国际标准。此后,美国国家标准局也采用 GKS 为国家标准。

目前已经制订或正在制订的一些图形标准都是接口标准。这些标准的功能旨在使图形系统中两部分之间的接口标准化,可以分为两类:

· 数据接口标准——用以确定系统中各界面之间数据传递和通讯的标准;

· 子程序接口标准——规定应用程序调用子程序的功能及格式的标准。

这些接口标准在图形系统中所处位置不同,起的作用及所提供的服务也不同(详见图 2-40),现简要说明如下:

(1) 基本图形交换规范 IGES(Initial Graphics Exchange Specification)

由于图形系统的广泛应用,越来越多的用户需要把它们的数据在不同的图形系统之间交换。过去常采用的方法是把一个系统产生的数据文件经过一个翻译器变成另一个系统能识别的数据文件,对多个图形系统就需要编写多个翻译器。IGES 就是为了解决数据在不同的图形系统间传送的问题,它定义了一套表示图形系统中常用的几何和非几何数据的格式以及相应的文件结构。1981 年,IGES 成为美国国家标准化委员会(ANSI)标准。

IGES 的作用是在不同的图形系统之间交换数据,其关系如图 2-41 所示。如数据要从系统 A 传送到系统 B,必须由系统 A 中的 IGES 前处理器把这些传送的数据格式转换成 IGES 格式,而 IGES 的实体数据再由系统 B 中的 IGES 后处理器把 IGES 转换成该系统内部的数据格式。把系统 B 的数据转送给系统 A 也需相同的过程。

IGES 中的基本单元是实体,它分为三类:

① 几何实体如点、直线、圆弧、样条曲线、曲面等;

② 描述实体如尺寸标准、绘图说明等;

③ 结构实体如结合项、图组、特性等。

IGES 的文件格式是由 ASCII 码、记录长度为 80 个字符的顺序文件组成。文件分成五个节,如图 2-42 所示,开始节存放用户可阅读的定义信息;整体节和结束节存放数据的

图 2-40　图形软件标准

图 2-41　不同系统通过 IGES 交换数据

原始信息和文件本身的信息; 实体信息存放在目录入口(DE) 和参数数据(PD) 节中, 在 DE 和 PD 节中还存放实体的有关指针及相互关系。

图 2-42　IGES 的文件结构

IGES 的出错处理是十分重要的, 因为在不同系统间转换数据会产生错误, 如重字符、错字符等, 开发的转换器会对 IGES 文件产生不正确的解释。这些错误可能发生在产生 IGES 文件或读入 IGES 文件的过程中, 这就要求 IGES 转换程序能报告并校正有关的

错误。一般 IGES 出错处理的过程如图 2-43 所示。

图 2-43　IGES 出错处理过程

(2) 图形核心系统 GKS(Graphics Kernal System)

GKS 提供了在应用程序和图形输入输出设备之间的功能接口，是一个子程序接口标准。它只定义了一个独立于语言的图形系统核心，在具体应用中，必须以符合使用语言约定的方式，把 GKS 嵌入到相应的语言之中。GKS 在图形应用中的地位如图 2-44 所示。图中每个层次可以调用下一层次的功能。对于应用程序员来说，通常使用面向应用层、依赖语言的接口层以及操作系统等资源。

图 2-44　GKS 在应用中的地位

GKS 作为一个系统核心，其意义还在于它提供的图形功能和各种特殊的图形设备是无关的。GKS 中系统可调用的图形输出和输入设备均称为图形工作站，它是一个抽象的物理设备。GKS 共有六种工作站：输入、输出、输入输出、独立图段存储(WISS)、元文件输出(MO)、元文件输入(MI)工作站、输入工作站至少有一个逻辑输入设备。输出工作站仅有输出功能，GKS 允许输出图素在不同的工作站之间变

换和传送。输入输出工作站既有输出功能、又有输入功能,既交互处理功能。WISS、MO 和 MI 工作站是 GKS 特置的用来暂时或永久地存储图形信息的,为了便于控制,也将它们作为工作站看待。GKS 提供的每类工作站都对应一个工作站描述表,用来描述工作站功能和特性。

GKS 提供一系列交互和非交互式图形设备的全部基本图形处理功能,大致可分为以下十类:

① 控制功能: 执行打开、关闭 GKS,以对使工作进入或退出活动状态和删除工作站等;

② 输出功能: 确定输出图形的类型;

③ 输出属性: 设定图素的各种属性以及各种因素在工作站上的表现方式;

④ 变换功能: 实现规范化变换和工作站变换;

⑤ 图组功能: 对图形进行生成、删除、复制以及实现图组属性控制;

⑥ 输入功能: 对各种输入设备初始化,设定设备方式,确定输入控制方式;

⑦ 询问功能: 查询 GKS 描述表、状态表、出错表、工作站描述表、图素表等;

⑧ 实用程序: 实现坐标变换等;

⑨ 元文件处理;

⑩ 出错处理。

GKS 包括下列几处基本图素:

· 线元素—— 折线集(polyline);

· 点元素—— 符号集(polymarker);

· 字符元素—— 文本(text);

· 光栅元素—— 区域填充(fill area)及单元阵列(cell array)。

在 GKS 中,图素能以图段方式组合,并以图段为单位进行图形输出,各图段由应用程序定义,GKS 根据所指定的图段名进行识别与选择,而且还可对图段进行变换、可见性、改名、检测及删除等一系列操作。

GKS 采用元文件在图形系统之间传送图形信息,其功能包括 GKS 元文件的生成、读入和解释等。GKS 元文件结构及其读、写过程如图 2-45 所示。

图 2-45　GKS 文件结构

GKS 是一个二维图形的标准,它不适用于三维图形的处理,为此发展了 GKS-3D 以适应三维图形的要求。GSK-3D 是一个完全的三维系统,即其所有图段、图素数据均是三维格式,即使原来的 GKS 中的二维数据也要转换成三维的格式。GKS-3D 的功能包括三维输入、三维图素、三维几何属性、具有视图操作的三维变换以及隐藏线、面的消除等。GKS-3D 和 GKS 是完全兼容的,它们在功能上可以混合使用,但在用 GKS-3D 处理 GKS 定义的功能之前,需要把 GKS 定义的内容转换成三维的格式,也就是说,GKS 的功能只是在 GKS-3D 某个具体平面上的实现。

(3) 程序员级层次结构图形系统 PHIGS(Programmer's Hierarchical Interactive Graphics System)

PHIGS 是 ANSI 在 1986 年公布的计算机图形软件系统标准,是为应用程序员提供的控制图形设备的子程序接口标准。与 GKS 不同,PHIGS 的图形数据按照层次结构组织,使多层次的应用模型能方便地利用 PHIGS 进行描述,而且,PHIGS 提供了动态修改和绘制显示图形数据的手段。

PHIGS 的标准功能可划分为九个程序模块来分别实现,各模块相对独立,一个模块仅通过系统的公共数据结构与其它模块间接连接。各模块调用的公共子程序集中在一个公共子程序模块中,从而使整个系统的逻辑结构清晰,且没有重复的程序功能,便于逐个模块地进行程序开发,并可利用已经测试通过的程序模块对正在调试的程序模块进行验证,也为整个 PHIGS 的开发提供了方便。PHIGS 的程序功能模块结构如图 2-46 所示。

图 2-46　PHIGS 的程序模块结构

PHIGS 的所有图形数据都被组织在称之为结构的单元中,结构之间可以通过层次调用发生联系。每个结构由若干个元素组成,其中包括图形元素(如直线、标号元素、模型变

换矩阵元素、观察选择元素、应用数据元素和结构调用元素等。结构元素不能在结构之外存在,每个结构都有一个应用程序给定的结构标识名与之唯一对应。多个结构可以通过在结构中的结构调用元素形成一个结构网格。但 PHIGS 标准规定结构网络中不能存在环路,即如果结构 A 直接或间接地调用了结构 B,那么结构 B 不能再调用结构 A。

应用程序可能通过四种方式创建一个 PHIGS 结构:

· 调用一个不存在的结构,并把结构调用元素插入到当前打开的结构中;

· 打开一个不存在的结构;

· 一个不存在的结构被登录到一个工作站上;

· 在改变结构标识符时引用了一个结构的标识名。

PHIGS 还提供了一种非常有效的结构编辑手段,这种编辑手段与字符文件的编辑十分相似。如删除已有的结构元素,插入新的结构元素和存取结构元素等。当然,只有创建了的结构才能对其进行编辑加工。PHIGS 的结构编辑命令主要有:

① Open Structure

用来打开一个结构进行编辑。该结构可被称之为当前结构,在同一时间只能有一个结构被打开。此时产生指向当前结构最后一个元素的元素指针,该指针的有效范围 0 到 n,其中 0 表示结构的开头,即第一个元素之前,n 表示结构中的最后一个元素。当一个结构打开后,应用程序可调用定义结构元素的功能插入更多的元素到该结构中。每个元素都被插入到结构中当前元素指针的下一个位置,然后移动元素指针使其指向新插入的元素。应用程序也可以调整元素指针的位置进行各种编辑操作。

② Close Structure

结束对当前结构的编辑。

③ Inquire Open Structure

返回当前结构的标识名。

④ Copy Structure

将一个指定结构中的内容拷贝到当前结构的元素指针位置之后,元素指针随后指向被拷贝的最后一个元素。

⑤ Empty Structure

删除该结构中的所有元素,并不影响该结构的存在性以及其它结构对该结构的调用,但此时删除了该结构对后继结构的调用关系。

⑥ Delete Structure

删除该结构和所有其它结构对该结构的调用元素。

⑦ Delete All Structure

删除包括根结构在内的结构网格中的全部结构。

⑧ Execute Structure

在当前结构中插入一个结构调用元素。

此外,PHIGS 还具有改变结构标识名,改变结构间的调用关系,询问结构的有关参数等功能。除了对结构的操作外,PHIGS 还可直接对结构中的元素进行有关操作,例如:将元素指针移到指定位置;通过给定位移量,相对地向前或向后移动元素指针;将元素指

针改在特定的标号元素处; 删除元素指针所指的元素等。

由于 PHIGS 对结构、元素都有丰富的编辑功能,从而用 PHIGS 便于构造高度交互性和动态性的图形系统。

PHIGS 与 GKS-3D 功能类似,但存在以下几方面的差别:

· 数据结构

GKS-3D 提供了单层、平面的图形数据结构,其图段用来表示的是图像信息而不是图形的构造信息,图段数据经过坐标规格化变换后,不再是定义该图段的坐标空间的数据。而 PHIGS 的结构始终是在造型空间中定义的数据。

· 可修改性

GKS 产生的图段,其内容不能修改,但影响图段整个特征的某些属性,如可见性、高光、可检测性、图段的几何变换等则是可以修改的。而 PHIGS 中的任何结构、结构中的任何一部分元素则可以在任何时候进行修改。

· 属性的存储

GKS 把图素属性和图素放在一起存入图形数据结构中,为了修改某图段中的一个图素的属性,必须去除该图素的旧属性,重新生成一个新属性。在 PHIGS 中,只有当遍历一个结构并要显示该结构时,其中的图素才变成输出图素,此时,那些属性结构元素才连接到输出图素上,因此,图形数据的修改是容易的。

· 输出流水线

GKS-3D 设置了三种不同的坐标系,首先在用户坐标系(WC)中定义图素,经规格化变换后到规格化设备坐标系(NDC),这是与设备无关的二维直角坐标系,取值范围在 0.0 到 1.0 之间。然后在 NDC 中,经图段变换、规格化裁剪、视图变换及视图映象等操作后,转换到与设备有关的设备坐标系(DC)中进行输出。

PHIGS 则定义了 5 种坐标系,其输出流水线为: 造型坐标系(MC),经局部、整体变换到用户坐标系(WC),经视图变换到观察坐标系(UVN),再经观察投影变换后到规格化的投影坐标系(NPC),最后经工作站映象到设备坐标系(DC)输出。

(4) 计算机图形设备接口 CGI(Computer Graphics Interface)

CGI 是 ISO TC 97 提出的较形设备接口标准草案,与 ANSI 1985 年公布的 VDI (Virtual Device Inferface)标准一致。CGI 的目的是提供一种与设备无关的方法,即可视为图形设备驱动程序的一种标准,属子程序接口标准。

CGI 有 5 部分功能:

① 控制、查询及出错处理;

② 输出图素及其属性;

③ 图段定义及处理;

④ 输入及响应处理;

⑤ 光栅图形处理。

CGI 的几何图素和结构图形都是在虚拟的设备坐标空间(VDC)、即二维笛卡尔坐标系中定义的。用户可用整型数、实型数在虚拟设备空间的范围内定义一个矩形窗口,用户定义的窗口将映象到物理显示设备的一个子矩形区域中。

CGI 的设备控制功能很强。它具有初始化设备和终止设备执行、重新设置设备属性和控制其约定状态的功能。它还为用户提供了从图形缓存输出图形并不断修改图形的手段，也可以强迫图形设备连续修改图形。CGI 还具有软拷贝设备和硬拷贝设备模拟功能。

(5) 计算机图形元文件 CGM(Computre Graphics Metafile)

CGM 是 ANSI 1986 年公布的标准，1987 年成为 ISO 标准。它是一套与设备无关的语义、词法定义的图形文件格式，提供了随机存取、传送、简洁定义图象的手段。CGM 是一个静态的图形生成元文件，即它不能产生所定义图形的动态效果，例如不能实现动态的几何变换。通用性是 CGM 的关键属性，即 CGM 能广泛地适用于各种设备及应用系统。

CGM 标准主要由两部分组成，其一是功能规格说明，以抽象的词法描述了相应的文件格式，包括元素标志符、含义的定义以及参数描述。其二描述了 CGM 的三种标准编码形式，即字符、二进制和清晰的正文编码。

CGM 的结构如图 2-47 所示。每个图形元文件由一个元文件描述体和若干个逻辑上独立的图形描述体顺序组成。每个图形描述体由一个图形描述单元和一个图形数据单元构成。

图 2-47　CGM 的结构

习　　题

1. 计算机图形系统与一般计算机系统最主要的差别是什么?
2. 在计算机图形系统中，对主计算机有什么特殊要求?
3. 画出交互式计算机图形系统的一般结构。
4. 图形系统的基本功能是什么?
5. 选择图形系统应考虑哪些原则?
6. 图形系统根据什么原则分类? 一般可分为几类? 说出每一类的主要特点。
7. 图形硬件设备主要包括哪些? 请按类别举出典型的物理设备?
8. CRT 中偏转系统最重要的特性是什么? 为什么在图形系统中多采用磁偏转系统?
9. 影响单色 CRT 图象显示质量的因素是什么? 通常多采用什么方法得到较光滑的图形?
10. 简述产生彩色显示的两种基本方法。

11. 为什么影孔板彩色 CRT 分辨率一般要比单色 CRT 的分辨率低？提高其分辨率受到什么因素的限制？
12. 直视型存储管与一般 CRT 在结构上有什么不同？用其显示图形有什么优缺点？
13. 试比较几种常用显示器件的主要性能。
14. 简述随机扫描显示器和光栅扫描显示器的简单工作原理和各自的特点。
15. 有两台光栅扫描显示器，它们的产品介绍都称可以有 4096 种颜色，但其中一台在显示时却只能产生 16 种颜色，试问两者的配置有何不同？
16. 光栅扫描显示系统中，显示处理机的主要功能是什么？
17. 具有独立显示处理机的光栅扫描系统，一般有哪几个存储器？
18. 图形系统有哪几种硬拷贝设备？
19. 根据工作原理分类，绘图仪可分为哪两类？各有什么优缺点？
20. 激光打印机的简单工作原理是什么？
21. 试比较几种单色硬拷贝机的主要性能。
22. 输入设备可有哪六种逻辑功能？请举出各自对应的物理设备。
23. 画出图形软件的层次结构及主要组成。
24. 建立基本图形软件可采用哪三种方法？各有什么特点？
25. 基本图形软件应包含哪些基本内容？
26. 图形软件主语言的选择应考虑哪些因素？
27. 操作系统一般可采用什么方法来管理图形系统的外部设备？
28. 驱动程序起什么作用？有几种实现方法？
29. 为什么要制定图形软件标准？可分为哪两类？
30. 请列举出当前已成为国际标准的几种图形软件标准，并简述其主要功能。

第三章　二维图形生成技术

在这一章里,主要叙述一些能在指定输出设备上,根据坐标描述构造二维几何图形的方法。我们知道,一幅图最简单的几何成分是点和直线,此外还有曲线、多边形区域以及字符串等。我们将着重讨论生成这些图素的基本技术和算法。

为在输出设备上输出一个点,就要把应用程序中的坐标信息转换成所用输出设备的相应指令。对于一个CRT监视器来说,输出一个点就是要在指定的屏幕位置上开启(接通)电子束,使该位置上的荧光点辉亮。对于黑白光栅显示器来说,则是要将帧缓存中指定坐标位置处的值置为"1",然后,当电子扫视每一条水平扫描线时,一旦遇到帧缓存中值为"1"的点,就发射一亮光,即输出一个点。对于随机扫描显示器,画点的指令保存在显示文件中,该指令把坐标值转换成偏转电压,并在每一个刷新周期内,使电子束偏转到相应位置。

在图形设备上输出一条直线,是通过在应用程序中对每一条直线端点坐标的描述,由输出设备将一对端点间的路径加以描绘来实现的。对于水平线或垂直线,只要有了驱动设备使之动作的指令,一般都能准确地画出。但是对于任意斜率的直线,就要考虑算法了。因为大多数图形设备,都只提供驱动x方向和y方向动作的信号。这两个方向的信号用来指示绘图笔动作或电子束的偏移,或控制应赋值象素的地址。

对于曲线及各种复杂的图素,我们都可以将其离散成许多小直线段,连接各直线来逼近欲生成的曲线或其它复杂图形。因此,直线的生成算法是二维图形生成技术的基础。

第一节　直线图形

一、生成直线的常用算法

直线生成算法的任务就是要给出一个判断方法,即判别规则。根据该规则决定图形设备什么时候该动作,什么时候不该动作。各种算法的差别主要在于产生这些判别规则的方法和过程不同,发送的命令不同,当然,各自更为适应的设备对象也有所不同。

1. 逐点比较法

这是绘图仪经常采用的一种方法。

所谓逐点比较法,就是在绘图过程中,绘图笔每画一笔,就与规定图形进行比较,然后决定下一步的走向,用步步逼近的方法画出规定的图形。下面讨论这种算法的判别规则。

我们以直线起点为坐标原点,并且约定直线在四个象限中画笔的走向如图3-1所示。

(1) 偏差计算

我们先以第一象限为例推导出偏差计算公式,然后再扩展到其它象限。

① 一般公式

如图3-2所示,设要画的线段为OA,而画笔的当前位置为M。我们以OM,OA斜率

图 3-1　各象限画笔走向　　图 3-2　根据直线斜率计算偏差

的大小来计算偏差。设 OA, OM 与 x 轴正向的夹角分别为 α和 β, 点 A 和 M 的坐标分别为(x_A, y_A), (x_M, y_M), 则偏差值 d 为:

$$d = \mathrm{tg}\beta - \mathrm{tg}\alpha = \frac{y_M}{x_M} - \frac{y_A}{x_A}$$

$$= \frac{y_M x_A - y_A x_M}{x_M x_A}$$

当 $d < 0$ 时, 表示笔在 OA 线段的下方, 根据约定, 此时应走 $+y$ 一步; 当 $d \geq 0$ 时, 表示笔在 OA 线段的上方(或在 OA 线上), 则应走 $+x$ 一步。

这种算法只需判断偏差值 d 的正负, 其值的大小并不重要。对于第一象限, 因为分母 x_M, x_A 永远为正, 所以只需判断分子项的正负即可。我们得到偏差的判断公式为:

$$F_M = y_M x_A - y_A x_M \qquad (3\text{-}1)$$

② 递推公式

用式(3-1)来计算偏差时, 由于每次都要计算两次乘法, 一次减法, 计算工作量还是很大的。如果我们设法用前一点的偏差来推算走步方向以及走步后的偏差, 则偏差计算就可以大大简化, 也更适于计算机实现。现在仍以第一象限为例, 简叙这种递推过程。

如图 3-3 所示, 设笔的当前位置为 $M_1(x_1, y_1)$, 此时 $F_1 = y_1 x_A - y_A x_1 < 0$, 应走 $+y$ 一步到 M_2, 即:

$$x_2 = x_1$$

$$y_2 = y_1 + 1$$

此处 $+1$ 表示向正方向走一个设备单位步长。这里, 我们假设 x, y 坐标值, 均已转换为当前设备单位。M_2 处的偏差为

$$F_2 = y_2 x_A - y_A x_2 = y_1 x_A + x_A - y_A x_1 = F_1 + x_A$$

若 $F_2 \geq 0$, 则应走 $+x$ 一步到 M_3, 即

$$x_3 = x_2 + 1$$

$$y_3 = y_2$$

M_3 处的偏差为

$$F_3 = y_3x_A - y_Ax_3 = y_2x_A - y_Ax_2 - y_A = F_2 - y_A$$

这样递推下去,就可得出第 i 步的结果:

如 $F_i \geq 0$,则走+ x 一步,此时 $F_{i+1} = F_i - y_A$;

如 $F_i < 0$,则走+ y 一步,此时 $F_{i+1} = F_i + x_A$。

由于偏差 F_i 的推算只用到了终点坐标值 x_A, y_A,而与中间点的坐标值无关,且只需要进行加减运算,因而大大减少了计算量。

图 3-3 第一象限线段的递推过程

图 3-4 第二、三、四象限中线段的偏差判别

③ 任意象限中的偏差计算公式

图 3-4 表示出直线在第二、三、四象限时 α和 β角的含义。

对于第二象限有:

$$\operatorname{tg}\alpha = \frac{|x_A|}{y_A}$$

$$\operatorname{tg}\beta = \frac{|x_M|}{y_M}$$

判别式

$$F = |x_M|y_A - |x_A|y_M$$

如果 $F_i \geq 0$,则走+ y 一步, $x_{i+1} = x_i$, $y_{i+1} = y_i + 1$, $F_{i+1} = F_i - |x_A|$;

如果 $F_i < 0$,则走- x 一步, $x_{i+1} = |x_i| + 1$, $y_{i+1} = y_i$, $F_{i+1} = F_i + y_A$。

对于第三象限有:

$$\operatorname{tg}\alpha = \frac{|y_A|}{|x_A|}$$

$$\operatorname{tg}\beta = \frac{|y_M|}{|x_M|}$$

对于第四象限有:

$$\operatorname{tg}\alpha = \frac{x_A}{\text{©}y_A\text{©}_l'}$$

$$\operatorname{tg}\beta = \frac{x_M}{\text{©}y_M\text{©}_l'}$$

现将各象限运算公式归纳如表 3-1。

表 3-1　偏差运算公式

	$F_i \geq 0$	$F_i < 0$
第一象限 第三象限	走 $\begin{matrix}+x\\-x\end{matrix}$，$F_{i+1} = F_i - \text{©}y_A\text{©}_l'$	走 $\begin{matrix}+y\\-y\end{matrix}$，$F_{i+1} = F_i + \text{©}x_A\text{©}_l'$
第二象限 第四象限	走 $\begin{matrix}+y\\-y\end{matrix}$，$F_{i+1} = F_i - \text{©}x_A\text{©}_l'$	走 $\begin{matrix}-x\\+x\end{matrix}$，$F_{i+1} = F_i + \text{©}y_A\text{©}_l'$

(2) 终点判断

设绘图机的步距为 Δt, 直线在 x, y 方向的增量分别为 Δx 和 Δy。按照上述运算方法, 绘图笔从直线的起点画到终点, 在 x 方向上应走 $\left|\frac{\Delta x}{\Delta t}\right|$ 步, 在 y 方向上应走 $\left|\frac{\Delta y}{\Delta t}\right|$ 步。因为对于任一条给定的直线, 各种方法所产生的 x 方向和 y 方向的走步信号总和在同一台设备上是相同的, 所以我们取 $n = \left|\frac{\Delta x}{\Delta t}\right| + \left|\frac{\Delta y}{\Delta t}\right|$ 作为终点判断的控制数, 并将此数存入计数器内。在 x 或 y 方向上每走一步, 计数器减 1, 当计数器减至零时, 作图停止。

也可以取 $n = \max\left|\frac{\Delta x}{\Delta t}\right|, \left|\frac{\Delta y}{\Delta t}\right|$ 作为长度控制数(即选取 x 或 y 方向上增量较大的方向作为计长方向), 并将 n 存入计数器中。在计长方向上每走一步, 计数器减 1, 直至 n 为零止。

2. 数值微分法

数值微分法即 DDA 法(Digital Differential Analyzer), 这是一种基于直线的微分方程来生成直线的方法。

设一直线段由 (x_1, y_1) 至 (x_2, y_2), 此直线的参数方程可表示为:

$$x = x_1 + (x_2 - x_1)t$$

$$y = y_1 + (y_2 - y_1)t$$

其中参数 t 由 0 变化到 1 时, x 从 x_1 变到 x_2, y 从 y_1 变到 y_2, 如果从 (x_1, y_1) 到 (x_2, y_2) 分成 n 步, 每一步的 t 变化 dt, 则第 i 步时

$$t_i = i\,¡¤dt, \qquad (i = 1, 2, \cdots, n)$$

对应

$$x_i = x_1 + (x_2 - x_1)\,¡¤t_i$$

$$y_i = y_1 + (y_2 - y_1)\,¡¤t_i$$

而

$$x_{i+1} = x_1 + dx(t_i + dt) = x_i + dx\,¡¤dt$$

$$y_{i+1} = y_1 + dy(t_i + dt) = y_i + dy\,¡¤dt$$

这里 $dx = x_2 - x_1$, $dy = y_2 - y_1$, 若令斜率 $k = \frac{dy}{dx}$

$$x_{inc} = dx \cdot dt$$

$$y_{inc} = dy \cdot dt$$

则

$$x_{i+1} = x_i + x_{inc}$$

$$y_{i+1} = y_i + y_{inc}$$

DDA 法的基本原理是: 在直线当前位置(x_i, y_i)上, 分别加上两个小增量 x_{inc}, y_{inc}, 从而求出下一点的坐标(x_{i+1}, y_{i+1})。这里 dt 是某个很小的量, 其取值方法的不同, 决定了有 2 种不同的 DDA 法。

(1) 简单 DDA 法

对于简单的 DDA 法, 我们通常选取:

$$dt = \frac{1}{\max(|dx|, |dy|)} \tag{3-2}$$

由于 x、y 方向的步长分别为:

$$x_{inc} = dx \cdot dt$$

$$y_{inc} = dy \cdot dt$$

所以, 在 x 或 y 变化大的方向, 即计长方向上, 步长始终为 1, 而另一方向上的步长则小于 1, 其是否走步, 视其有无整数部分溢出, 也即:

若 $|k| \leq 1$, 则取 $x_{inc} = \pm 1$, $y_{inc} = \pm k$, $y_{i+1} = y_i \pm k$;

若 $|k| > 1$, 则取 $y_{inc} = \pm 1$, $x_{inc} = \pm 1/k$, $y_{i+1} = y_i \pm 1/k$。

简单 DDA 法可以用一套寄存器和加法器来实现, 如图 3-5 所示, 其中小数部分初值为 0.5, 非计长方向、小于单位步长的增量信号(x_{inc}或y_{inc})加到加法器的输入端, 当一旦有整数部分溢出时, 则驱动该方向走步。

图 3-5　由硬件实现简单 DDA 法的原理图

(2) 对称 DDA 法

对于对称 DDA 法,我们可以找到如下关系式:

$$2^{m-1} < \max(|dx|, |dy|) < 2^m$$

则令

$$dt = 2^{-m} \tag{3-3}$$

所以

$$x_{inc} = dx \times 2^{-m} < 1$$

$$y_{inc} = dy \times 2^{-m} < 1$$

即在 x 和 y 两个方向上,步长均小于 1。若用硬件来实现对称 DDA 法,则应用 2 套寄存器和加法器,将 x_{inc}和 y_{inc}分别加到 x 和 y 的小数部分,哪个方向上有整数溢出,就驱动该方向走步。

用 DDA 法产生的直线比较精确,而且逻辑简单,易于用硬件实现。但是,在求 x_{inc}或 y_{inc}时,要作除法运算,影响了算法的效率。

3. Bresenham 直线生成算法

由 Bresenham 提出的直线生成算法实际上也是简单的 DDA 算法。即在某一计长方向上,每次必变化 1 个单位步长或一个象素单位,另一方向上的变化量可通过计算得到。这种算法只用到整型数计算,不必作四舍五入操作,而且在计算判别式时,计算量很小,因而得到广泛的应用。下面我们介绍在光栅扫描系统上实现的这种算法。

先假设直线的斜率在 0～1 之间,如图 3-6 所示。设 $P_{i-1}(x_{i-1}, y_{i-1})$是已选定的离直线最近的象素,现在要决定下一个象素是 T_i还是 S_i。由图 3-6 可以直观地看出:

若 $s < t$,则 S_i比较靠近直线,应选 S_i;

若 $s \geq t$,则应选 T_i。

假设直线从(x_1, y_1)到(x_2, y_2),经变换后可表示为从$(0, 0)$到(dx, dy),其中 $dx = x_2 - x_1$, $dy = y_2 - y_1$。此时直线方程可简化为:

$$y = \frac{dy}{dx} \times x$$

图 3-6 Bresenham 算法的几何图形

设 P_{i-1}的坐标为(r, q),则 S_i 的坐标为$(r+1, q)$, T_i 的坐标为$(r+1, q+1)$。因此,

$$s = \frac{dy}{dx}(r+1) - q$$

$$t = (q+1) - \frac{dy}{dx}(r+1)$$

则:

$$s - t = 2\frac{dy}{dx}(r+1) - 2q - 1$$

即:

$$dx(s - t) = 2(r \times dy - q \times dx) + 2dy - dx$$

由于 dx 为正,故我们可以选 $dx(s-t)$的正、负作为选择 S_i和 T_i的标准。若令其为 d_i,

则有:

$$d_i = 2(r \times dy - q \times dx) + 2dy - dx$$

将 $r = x_{i-1}$, $q = y_{i-1}$ 代入, 则有:

$$d_i = 2x_{i-1} \times dy - 2y_{i-1} \times dx + 2dy - dx \tag{3-4}$$

将每一下标加 1, 即以 $i+1$ 代 i, 则得:

$$d_{i+1} = 2x_i \times dy - 2y_i dx + 2dy - dx$$

从 d_{i+1} 中减去 d_i, 得到:

$$d_{i+1} - d_i = 2dy(x_i - x_{i-1}) - 2dx(y_i - y_{i-1})$$

因为 $x_i - x_{i-1} = 1$, 重写上式得:

$$d_{i+1} = d_i + 2dy - 2dx(y_i - y_{i-1})$$

如果 $d_i \geq 0$, 则选 T_i, $y_i = y_{i-1} + 1$, 且

$$d_{i+1} = d_i + 2(dy - dx) \tag{3-5}$$

如果 $d_i < 0$, 则选 S_i, $y_i = y_{i-1}$, 且

$$d_{i+1} = d_i + 2dy \tag{3-6}$$

这样, 我们就有了一个迭代的方法, 即由前一个 d_i 计算出下一个 d_{i+1}, 并对 S_i 及 T_i 进行选择。初值的计算可由式(3-4)求出, 因为此时 $i = 1$, 且 $(x_0, y_0) = (0, 0)$, 于是

$$d_1 = 2dy - dx \tag{3-7}$$

我们看到式(3-5)、式(3-6)及式(3-7)所需的计算量是很小的, 它只包括加法、减法和左移(乘 2)操作。这也是本算法得到广泛应用的原因。

当斜率大于 1 时, 我们可以将 x、y 交换, 即以 y 方向作为计长方向, 每次变化 1 个单位步长或 1 个象素单位。对于任意走向的直线, 则可以根据其端点的位置来控制其增量的正负。下面就是实现上述算法思想的 C 语言程序。

```
int Bres-Line(int x1, int y1, int x2, int y2, int color)
{
    int oldcolor, iTag;
    int dx, dy, tx, ty, inc1, inc2, d, curx, cury;

    oldcolor= setcolor(color);
    setpixel(x1, y1);

    if (x1= = x2 && y1= = y2)
      {
       setcolor(oldcolor);
       return(1);
      }

    iTag= 0;
    dx= abs(x2- x1);
    dy= abs(y2- y1);
    if (dx< dy)
      {
        iTag= 1;
        iSwap(&x1, &y1);
```

```
            iSwap(&x2, &y2);
            iSwap(&dx, &dy);
        }

    tx= (x2- x1)> 0? 1: - 1;
    ty= (y2- y1)> 0? 1: - 1;
    curx= x1;
    cury= y1;
    inc1= 2* dy;
    inc2= 2* (dy- dx);
    d= inc1- dx;
    while (curx! = x2)
      {
        if (d< 0)
           {
            d+ = inc1;
           }
        else
           {
            cury+ = ty;
            d+ = inc2;
           }
        if (iTag)
           setpixel (cury, curx);
        else
           setpixel (curx, cury);
        curx + = tx;
      }
    setcolor (oldcolor);
    return (0);
}

iSwap(int * a, int * b)
{
    int tmp;

    tmp= * a;
    * a= * b;
    * b= tmp;
}
```

二、直线属性

直线的属性包括线型、线宽和线色。在直线生成程序中,通常可根据用户选择的一些特定参数值来产生具有相应属性的直线。这些参数应全程有效,实用系统中,可把常用属性值设置为默认值,如某种颜色的细实线。

1. 线型

最常见的线型有实线、虚线和点划线等。例如,可以自己编写函数 setlt(lt)来设置直

线的线型,将 lt 选择为 1, 2, 3, 分别对应所需要的线型。对于虚线和点划线,当然还应规定各段明暗部分的长度,以便计算相应坐标,控制其输出。长度定义一般是在用户坐标系下进行的。通常,线型的默认值是实线。

2. 线宽

直线的宽度一般是在设备坐标系下定义的。我们可以把设备能产生的最小直线宽度(通常认为是 1 个光点宽度、绘图笔的画线宽度,或是 1 个象素的宽度)作为基准,再定义二倍宽或四倍宽的直线。产生线宽的功能是比较容易实现的,只要有了输出基本宽度的直线生成程行,重复画几次,便可得到几倍于基本宽度的直线。不过,此时还应考虑直线的斜率,以便决定在端点处是沿 x 方向还是沿 y 方向增减坐标重复画线。当然,对于粗线在连接处还应作特殊处理,才能达到满意的效果。当自己编写 setlw(lw)函数设置线宽时,lw 可选取 1, 2, 3, 分别对应不同的线宽。线宽的默认值常选取基本宽度的直线。

3. 线色

一般光栅扫描系统可以提供多种颜色,通常用 0, 1, 2, …等整数值表示不同的颜色。例如,一个能产生 8 种颜色的光栅系统,其颜色代码与保存在帧缓存中的颜色值的对应关系如表 3-2 所示。这里,每个象素点用 3 个二进制位表示其颜色,每一位分别控制红、绿、蓝电子枪。例如,编写函数 setlc(lc)来设置线色时,lc 即颜色代码号,可由应用程序选择。若帧缓存中每个象素的颜色用更多的二进制位表示,那么颜色的种类也增加。例如,每个象素占 6 位,红、绿、蓝电子枪分别由 2 位来控制,则每个电子枪可以有 4 种颜色等级,总共可产生 64 种颜色。

表 3-2　帧缓存保存颜色代码的方法

颜色代码	帧缓存中保存的颜色值			所显示的颜色
	红	绿	蓝	
0	0	0	0	黑
1	0	0	1	蓝
2	0	1	0	绿
3	0	1	1	青
4	1	0	0	红
5	1	0	1	紫
6	1	1	0	黄
7	1	1	1	白

如果电子枪的颜色等级不是由保存在帧缓存中的值来控制,而是由保存在彩色查寻表中的值控制,帧缓存中的值只作为查寻表的入口指针,那么在查寻表的位数 W 大于帧缓存的位平面数 N 时,就可以大大增加选择颜色的余地,而无需增加帧缓存的容量。图 3-7中,N = 6, 有 64 个查寻表入口地址, W = 12, 因此总共可以产生 4096 种颜色。用户可事先填写查寻表中的内容以选取不同的颜色组合。若用 sctct(ct, lc)子程序选择颜色,其

中 ct 即查寻表的地址,lc 为选中的颜色代码返回参数。

图 3-7　用彩色查寻表控制颜色

对于不具备彩色功能的显示器,应用程序中选择颜色的子程序功能就被转换成确定直线上各点的亮度等级(或称灰度等级)。亮度等级常用 0—1 之间的数来表示。在帧缓存中,保存亮度等级与保存彩色代码相类似,这时,应用程序中所指定的亮度值被转换成相应的二进制代码。如果光栅中每个象素用 1 位表示,那么只可能有黑或白两级灰度。如果每个象素用 3 位表示,就可以提供 8 种不同等级的亮度。在高质量的系统中,帧缓存中每个象素可能占 8 位或更多位,因而可表示多级灰度。表 3-3 所表示的是由用户所定义的四级灰度值及其被转换成的相应代码。此例中,凡是 0.33 左右的亮度值,均以二进制代码“01”表示,它对应于深灰色的亮度等级。

表 3-3　帧缓存中保存的灰度等级

亮度值代码	帧缓存中		所显示的灰度等级
	亮度值	二进制值	
0.0	0	0 0	黑
0.33	1	0 1	深灰
0.67	2	1 0	浅灰
1.0	3	1 1	白

第二节　二 次 曲 线

二次曲线是指那些能用二次函数

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

来表示的曲线,包括圆、椭圆、抛物线、双曲线等。本节将介绍几种拟合二次曲线的方法。基本的技术是将曲线离散成小直线段,通过连接各直线段来逼近所要的曲线。

一、圆弧和椭圆弧的拟合法

1. 逐点比较法插补圆弧

这种方法就是在输出圆弧的过程中,每走完 1 个单位长度以后,就与应画的圆弧进行比较,根据比较的结果,然后再决定下一次的走向,这样就可以一步一步地逼近所画的圆弧。为了便于比较,首先需要约定当前轨迹点处于不同象限时的运动走向。当然,这种走向与顺时针还是逆时针画圆也有关系。图 3-8 及图 3-9 分别表示了这种约定。下面的问题就是要推导一个圆弧插补的判别公式(也称之为判别函数),用其判断轨迹点的位置是位于圆内还是圆外。然后,再根据上述约定就可决定应沿 x 方向走还是沿 y 方向走。这样画出的圆弧肯定是阶梯形的,但当单位步长比较小(如 0.1mm)时,人眼所看到的圆弧仍然是光滑的。

图 3-8　顺圆走向　　图 3-9　逆圆走向

(1) 判别函数

我们先以图 3-10 第 I 象限画逆圆为例推导判别函数的表达式,然后再推广到其它象限及画顺圆的情况。设圆心为 $C(x_c, y_c)$,起始点 $S(x_s, y_s)$ 在圆上,因而可求出半径 $R_s = \sqrt{(x_s - x_c)^2 + (y_s - y_c)^2}$,另有一点 $M(x_m, y_m)$,其到圆心的距离为 $R_m = \sqrt{(x_m - x_c)^2 + (y_m - y_c)^2}$。我们可以取 $R_m - R_s$ 作为判别函数,但为避免开方,我们取 $F_m = R_m^2 - R_s^2$,作为判别函数。显然,当 $F_m > 0$ 时,点 M 在圆外;$F_m = 0$ 时,点 M 在圆上;$F_m < 0$ 时,点 M 在圆内。所以,根据 F_m 的正负就可以确定走向。

图 3-10　圆弧的插补

(2) 递推公式

因为起始点 S 在圆上,所以 $F_0 = 0$,根据约定,我们走一步 $-\Delta x$,则 $x_1 = x_s - \Delta x$,$y_1 = y_s$,且

$$F_1 = (x_1 - x_c)^2 + (y_1 - y_c)^2 - R_s^2$$

$$= F_0 - 2\Delta x(x_s - x_c) + \Delta x^2$$

由于 $F_0 = 0$, $\Delta x^2 > 0$, 以及 I 象限内 $x_s - x_c > 0$, 且 $x_s - x_c > \Delta x$, 所以 $F_1 < 0$, 应走 $+\Delta y$ 一步, 并有 $x_2 = x_1$, $y_2 = y_1 + \Delta y$, 因而有:

$$F_2 = (x_2 - x_c)^2 + (y_2 - y_c)^2 - R_s^2$$

$$= F_1 + 2\Delta y(y_1 - y_c) + \Delta y^2$$

如此推导下去, 对于第 i 步($i = 0, 1, 2, \cdots, n$), 如果 $F_i \geq 0$, 则走一步 $-\Delta x$, $x_{i+1} = x_i - \Delta x$, $y_{i+1} = y_i$。

$$F_{i+1} = F_i - 2\Delta x \cdot (x_i - x_c) + \Delta x^2;$$

如果 $F_i < 0$, 则走一步 $+\Delta y$, $x_{i+1} = x_i$; $y_{i+1} = y_i + \Delta y$。

$$F_{i+1} = F_i + 2\Delta y \cdot (y_i - y_c) + \Delta y^2$$

(3) 一般运算规律

我们从顺、逆圆运动走向的约定中可以看到, 对于逆圆的Ⅰ、Ⅲ象限及顺圆的Ⅱ、Ⅳ象限, 沿运动走向的 x 绝对值是不断减小的, 而 y 绝对值是不断增加的, 因而其插补运算规律相同。对于逆圆Ⅱ、Ⅳ象限, 其 x 绝对值不断增加, y 绝对值不断减少, 插补运算规律与顺圆Ⅰ、Ⅲ象限相同。所以, 圆弧逐点插补运算规律可以归纳成表 3-4, 其中 $x_i = |x_i - x_c|$, $y_i = |y_i - y_c|$。

表 3-4 圆弧逐点插补运算规律表

象限	$F_i \geq 0$		$F_i < 0$	
	移动量	运算公式	移动量	运算公式
逆Ⅰ	$-\Delta x$	$F_{i+1} = F_i - 2\Delta x \cdot x_i + \Delta x^2$	$+\Delta y$	$F_{i+1} = F_i + 2\Delta y \cdot y_i + \Delta y^2$
逆Ⅲ	$+\Delta x$	$\lvert x_{i+1}\rvert = \lvert x_i\rvert - \Delta x$	$-\Delta y$	$\lvert x_{i+1}\rvert = \lvert x_i\rvert$
顺Ⅱ	$+\Delta x$	$\lvert y_{i+1}\rvert = \lvert y_i\rvert$	$+\Delta y$	$\lvert y_{i+1}\rvert = \lvert y_i\rvert + \Delta y$
顺Ⅳ	$-\Delta x$		$-\Delta y$	
逆Ⅱ	$-\Delta y$	$F_{i+1} = F_i - 2\Delta y \cdot y_i + \Delta y^2$	$-\Delta x$	$F_{i+1} = F_i + 2\Delta x \cdot x_i + \Delta x^2$
逆Ⅳ	$+\Delta y$	$\lvert x_{i+1}\rvert = \lvert x_i\rvert + \Delta x$	$+\Delta x$	$\lvert x_{i+1}\rvert = \lvert x_i\rvert + \Delta x$
顺Ⅰ	$-\Delta y$	$\lvert y_{i+1}\rvert = \lvert y_i\rvert - \Delta y$	$+\Delta x$	$\lvert y_{i+1}\rvert = \lvert y_i\rvert$
顺Ⅲ	$+\Delta y$		$-\Delta x$	

(4) 终点判断

通常多采用简单的终点判断, 即每走一步 x 或 y, 都与终点坐标去比较。令终点坐标为(x_e, y_e), 我们可以假设, 当 $|x_i - x_e| < \delta$ 且 $|y_i - y_e| < \delta$ 时(δ 为给定的某一正整数, 如可取为 1), 就认为已到达终点。为了避免起点与终点靠得很近, 而又需要画大圆弧时所引起的误判, 我们可以先根据判别函数和走向, 在连续走几步后, 再将当前点与终点坐标进行比较, 只要未到终点, 就继续执行运算规则。

2. 角度 DDA 法产生圆弧

若已知圆心坐标为(x_c, y_c), 半径为 r, 则以角度 t 为参数的圆的参数方程可写为:

$$x = x_c + r \cdot \cos t$$

$$y = y_c + r \cdot \sin t$$

当 t 从 0 变化到 2π时,上述方程所表示的轨迹是一整圆;当 t 从 t_s 变化到 t_e 时,则产生一段圆弧。由于我们定义角度的正方向是逆时针方向,所以圆弧是由 t_s 到 t_e 逆时针画圆得到的。

若给定圆心坐标(x_c, y_c),半径 r 及起始角 t_s 和终止角 t_e,要产生从 t_s 到 t_e 这段圆弧的最主要问题是离散化圆弧,即求出从 t_s 到 t_e 所需运动的总步数 n。可令:

$$n = (t_e - t_s)/dt + 0.5$$

其中,dt 为角度增量,即每走一步对应的角度变化。下面的问题就是如何选取 dt。通常,是根据半径 r 的大小来给定 dt 的经验数据。在实际应用中,应对速度和精度的要求加以折衷,并适当调整 dt 的大小。如果用户给定的 $t_e < t_s$,则可令 $t_e = t_e + 2\pi$ 以保证从 t_s 到 t_e 逆时针画圆。如果 n= 0,则令 n= 2πdt,即画整圆。为避免累积误差,最后应使 t= t_e,强迫止于终点。用上述算法思想产生圆弧的 C 语言程序如下所示。

```
arc(int xc, int yc, double r, double ts, double te)
{
    double rad, tsl, tel, deg, dte, ta, ct, st;
    int x, y, n, i;

    rad= 0.0174533;
    tsl= ts* rad;
    tel= te* rad;
    if (r< 5.08)
      deg= 0.015;
    else
      if (r< 7.62)
        deg= 0.06;
      else
        if (r< 25.4)
          deg= 0.075;
        else
          deg= 0.15;
    dte= deg* 25.4/r;
    if (tel< tsl)
      tel + = 6.28319;
    n= (int)((tel- tsl)/dte+ 0.5);
if (n= = 0)
  n= (int)(6.28319/dte+ 0.5);
    ta= tsl;
    x= xc+ r* cos(tsl);
    y= yc+ r* sin(tsl);
    moveto(x, y);
    for (i= 1; i< = n; i+ + )
      {
        ta + = dte;
        ct = cos(ta);
        st = sin(ta);
        x = xc+ r* ct;
        y= yc+ r* st;
```

```
            lineto(x,y);
          }
        x= xc+ r* cos(tel);
        y= yc+ r* sin(tel);
        lineto(x,y);
        return(0);
    }
```

在平面上给定 3 个点的坐标 $P_1(x_1,y_1)$, $P_2(x_2,y_2)$, $P_3(x_3,y_3)$, 我们就可以产生从 P_1 到 P_3 的一段圆弧。这里的关键是要求出圆心坐标和半径, 以及起点 P_1 和终点 P_3 所对应的角度 t_s 和 t_e。根据 3 个点坐标, 我们不难求出圆心坐标为:

$$x_c = (A \times (y_3 - y_2) + B \times (y_2 - y_1))/(2 \times E)$$

$$y_c = (A \times (x_2 - x_3) + B \times (x_1 - x_2))/(2 \times E)$$

其中:

$$A = (x_1 + x_2) \times (x_1 - x_2) + (y_1 + y_2) \times (y_1 - y_2)$$

$$B = (x_3 + x_2) \times (x_3 - x_2) + (y_3 + y_2) \times (y_3 - y_2)$$

$$E = (x_1 - x_2) \times (y_3 - y_2) - (x_2 - x_3) \times (y_2 - y_1)$$

如果 $E = 0$, 也即

$$\frac{y_3 - y_2}{x_3 - x_2} = \frac{y_2 - y_1}{x_2 - x_1}$$

则应画从 $P_1(x_1,y_1)$ 到 $P_3(x_3,y_3)$ 的一条直线。否则, 由上式求出圆心坐标后, 可求出半径:

$$r = \sqrt{(x_1 - x_c)^2 + (y_1 - y_c)^2}$$

起点和终止点对应的角度可分别由下式求出:

$$t_s = \begin{cases} 90 \text{ 度}, & \text{当 } x_1 = x_c \text{ 时} \\ \mathrm{tg}^{-1}\dfrac{y_1 - y_c}{x_1 - x_c} \times 57.2958 \text{ 度}, & \text{当 } x_1 \neq x_c \text{ 时} \end{cases}$$

$$t_e = \begin{cases} 90 \text{ 度} & \text{当 } x_3 = x_c \text{ 时} \\ \mathrm{tg}^{-1}\dfrac{y_3 - y_c}{x_3 - y_c} \times 57.2958 \text{ 度}, & \text{当 } x_3 \neq x_c \text{ 时} \end{cases}$$

由计算机提供的反正切函数所计算出的角度一般只是第 I 和第Ⅳ象限的角度, 因此, 根据判断, 若 P_1(或 P_3) 点相对圆心是位于第Ⅱ和第Ⅳ象限内, 则由上式求出的 t_s(或 t_e) 应再加 180 度。另外, 由于我们是从 P_1 画到 P_3, 即从 t_s 到 t_e, 是逆时针画圆, 所以 P_2 点不一定在圆弧上。同样, 当 $t_e < t_s$ 时, 应使 $t_e = t_e + 360$。

3. 角度 DDA 法产生椭圆弧

在平面上给定椭圆的长、短半轴 a 和 b, 椭圆上的起始点 $S(x_s,y_s)$, 起始点离心角 t_s 及终止点离心角 t_e, 以及长轴与 X 轴正向的夹角 α后, 利用椭圆的参数方程和坐标变换, 用 DDA 法即可从 t_s 到 t_e 画出椭圆弧。首先求出椭圆圆心的坐标(x_c,y_c)。如图 3-11 所示, 在 X 乇Y 坐标系中, S 点坐标为

$$x_s' = a \cdot \cos t$$
$$y_s' = b \cdot \sin t$$

图 3-11　用角度 DDA 法产生椭圆弧

在 X′Y′坐标系中, S 点坐标为:

$$x_s'' = x_2' \cdot \cos\alpha - y_s' \cdot \sin\alpha$$
$$y_s'' = x_s' \cdot \sin\alpha + y_s' \cdot \cos\alpha$$

在用户坐标系 XOY 中, S 点的坐标可表示为:

$$x_s = x_c + x_s''$$
$$y_s = y_c + y_s''$$

其中, (x_c, y_c) 即为欲求的椭圆圆心的坐标。

$$x_c = x_s - x_s''$$
$$y_c = y_s - y_s''$$

下面的问题是求由起始点 S 到终止点 E(即从起始点离心角 t_s 到终止点离心角 t_e)的椭圆弧上各点的 x、y 坐标。同 DDA 法产生圆弧一样, 我们将这段圆弧离散为 n 等分, 角度增量为 dt, 于是:

$$n = (t_e - t_s)/dt + 0.5$$

对于第 i 步, 其对应的离心角为:

$$t_i = t_s + i \cdot dt, \qquad i = 1, 2, \cdots, n.$$

该点坐标为:

$$x_i = x_c + x_i''$$
$$y_i = y_c + y_i''$$

而

$$x_i'' = x_i' \cdot \cos\alpha - y_i' \cdot \sin\alpha$$
$$y_i'' = x_i' \cdot \sin\alpha + y_i' \cdot \cos\alpha$$

及

$$x_i' = a \cdot \cos t_i$$

$$y_i' = b \cdot \sin t_i$$

有了各点的坐标值(x_i, y_i)，就可以从起始点开始，调用两点间画直线程序画出椭圆弧。下面的C语言程序段就是按上述算法思想编写的。

```
ellipse(int xc, int yc, double a, double b, double alp, double ts, double te)
{
    double rad, tsl, tel, alpl, deg, dte, r, ta, a1, a2, b1, b2;
    int x, y, n, i;

    rad= 0.0174533;
    tsl= ts* rad;
    tel= te* rad;
    alpl= alp* rad;
    a1= a* cos(tsl);
    b1= cos(alpl);
    a2= b* sin(tsl);
    b2= sin(alpl);
    r= (a> b)? a: b;

    if (r< 5.08)
      deg= 0.015;
    else
      if (r< 7.62)
      deg= 0.06;
      else
        if (r< 25.4)
          deg= 0.075;
        else
          deg= 0.15;
    dte= deg* 25.4/r;

    if (tel< tsl)
      tel + = 6.28319;
    n= (int)((tel- tsl)/dte+ 0.5);
    if (n= = 0)
      n= (int)(6.28319/dte+ 0.5);
    ta= tsl;
    x= xc+ a1* b1- a2* b2;
    y= yc+ a1* b2+ a2* b1;
    moveto(x, y);
    for (i= 1; i< = n; i+ + )
      {
        ta + = dte;
        a1= a* cos(ta);
        a2= b* sin(ta);
        x= xc+ a1* b1- a2* b2;
        y= yc+ a1* b2+ a2* b1;
        lineto(x, y);
```

```
        }
        a1= a* cos(tel);
        a2= b* sin(tel);
        x= xc+ a1* b1- a2* b2;
        y= yc+ a1* b2+ a2* b1;
        lineto(x,y);
        return(0);
    }
```

4. Bresenham 画圆算法

如图 3-12 所示,假设圆心坐标在坐标系原点,只产生第 I 象限的八分之一圆弧,即从 $S(0,R)$ 到 $E(R/\sqrt{2},R/\sqrt{2})$ 之间的 45 圆弧。

此算法在每一步都选择一个离开实际圆周最近的点 $P_i(x_i,y_i)$ 使其误差项

$$|D(P_i)|=|(x_i^2+y_i^2)-R^2|$$

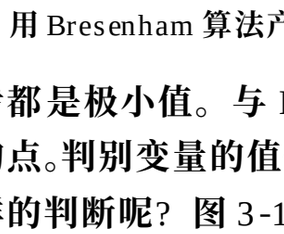

图 3-12　用 Bresenham 算法产生圆弧

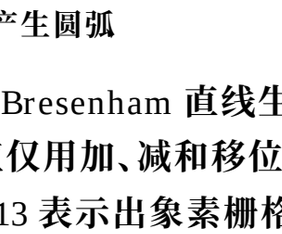

图 3-13　Bresenham 画圆算法的判别点

在每一步都是极小值。与 Bresenham 直线生成算法一样,其基本方法是利用判别变量选择最近的点。判别变量的值仅用加、减和移位运算即可逐点计算出来,其符号用作判断。要作什么样的判断呢? 图 3-13 表示出象素栅格的一小部分,以及这个实际的圆弧穿过此栅格的各种可能的形式(从 A 到 E)。

设 P_{i-1} 已被选为 $x=x_{i-1}$ 时最靠近圆弧的点。现在,当 $x=x_{i-1}+1$ 时,必须决定是 $T_i(x_{i-1}+1,y_{i-1}-1)$ 还是 $S_i(x_{i-1}+1,y_{i-1})$ 更接近实际的圆弧。我们可以令:

$$D(S_i)=[(x_{i-1}+1)^2+(y_{i-1})^2]-R^2$$

$$D(T_i)=[(x_{i-1}+1)^2+(y_{i-1}-1)^2]-R^2$$

它们分别代表从圆心(坐标原点)到 S_i 或 T_i 的距离平方与半径平方的差。如果 $|D(S_i)|\geq|D(T_i)|$ 则 T_i 比 S_i 更接近实际的圆弧;反之,则应选 S_i。现在令:

$$d_i=|D(S_i)|-|D(T_i)| \tag{3-8}$$

显然,如果 $d_i\geq 0$,则选 T_i,且 $y_i=y_{i-1}-1$;如果 $d_i<0$,则选 S_i,且 $y_i=y_{i-1}$。

由于式(3-8)的计算要用到绝对值,所以比较麻烦。通过对图 3-13 所示几种情况的分析,我们发现可以简化这种计算。

对于情况 C,因为 S_i 在圆外,我们有 $D(S_i)>0$,同时因为 T_i 在圆内,故有 $D(T_i)<0$。

如果我们把式(3-8)改写成:

$$d_i = D(S_i) + D(T_i) \tag{3-9}$$

即可直接根据式(3-9)中 d_i 的正负来选择 T_i 或 S_i。

对于情况 A 和 B,由于$|D(S_i)| < |D(T_i)|$,由式(3-8)得 $d_i < 0$,故应选 S_i。从另一方面分析,由于 T_i 在圆内,即 $D(T_i) < 0$,S_i 在圆内(情况 A)或圆上(情况 B),即 $D(S_i) \leq 0$,所以如果也用式(3-9)来计算 d_i,此值也小于零,故也得出选 S_i 的结论。

对于情况 D 和 E,由于$|D(S_i)| > |D(T_i)|$,由式(3-8)知 $d_i > 0$,应选 T_i。而此时 S_i 在圆外,即 $D(S_i) > 0$,T_i 在圆上(情况 D)或圆外(情况 E),即 $D(T_i) \geq 0$,由式(3-9)得 $d_i > 0$,也得出选 T_i 的结论。

因此,我们可以用式(3-9)代替式(3-8)。下面的工作是要推导一个递推公式,以简化 d_i 的计算。根据定义及上述讨论可有:

$$\begin{aligned} d_i &= D(S_i) + D(T_i) \\ &= [(x_{i-1} + 1)^2 + y_{i-1}^2] - R^2 + [(x_{i-1} + 1)^2 + (y_{i-1} - 1)^2] - R^2 \end{aligned}$$

用 $i+1$ 代 i,得

$$d_{i+1} = [(x_i + 1)^2 + y_i^2] - R^2 + [(x_i + 1)^2 + (y_i - 1)^2] - R^2$$

如果 $d_i < 0$,选 S_i,$x_i = x_{i-1} + 1$,$y_i = y_i - 1$,得:

$$d_{i+1} = d_i + 4x_{i-1} + 6 \tag{3-10}$$

如果 $d_i \geq 0$,选 T_i,$x_i = x_{i-1} + 1$,$y_i = y_{i-1} - 1$,得:

$$d_{i+1} = d_i + 4(x_{i-1} - y_{i-1}) + 10 \tag{3-11}$$

对于 $i = 1$,$x_{i-1} = x_0 = 0$,$y_{i-1} = y_0 = R$,则初值为:

$$d_1 = 3 - 2R \tag{3-12}$$

显然,式(3-10)、式(3-11)及式(3-12)的计算量是很小的,因此效率较高。当然,这样仅产生八分之一的圆弧。利用圆周对坐标原点(圆心)及坐标轴的对称性,就可以将 45° 圆弧扩展到整个圆周。如图 3-14 所示,如果点(x, y)在圆周上,则与之对称的另外 7 个点也在圆周上,其坐标是很容易求出的。由此类推,对于圆心为任意坐标值的圆也不难实现。用上述算法实现的 C 语言源程序如下所示。

```
int Bres_Circle(int x0, int y0, double r,
                int color)
{
    int x, y, d;

    x = 0;
    y = (int)r;
    d = (int)(3 - 2* r);
```

图 3-14　一个圆上的 8 个对称点

```
        while (x< y)
          {
           CirPot(x0, y0, x, y, color);
           if (d< 0)
             d + = 4* x+ 6;
           else
            {
             d + = 4* (x- y)+ 10;
             y- - ;
            }
           x+ + ;
          }
        if (x= = y)
          CirPot( x0, y0, x, y, color);
        return( 0);
}

int CirPot(int x0, int y0, int x, int y, int color)
{
        int old color;

        old color= setcolor(color);
        setpixel((x0+ x),(y0+ y));
        setpixel((x0+ y),(y0+ x));
        setpixel((x0+ y),(y0- x));
        setpixel((x0+ x),(y0- y));
        setpixel((x0- x),(y0- y));
        setpixel((x0- y),(y0- x));
        setpixel((x0- y),(y0+ x));
        setpixel((x0- x),(y0+ y));
        setcolor(oldcolor);
        return( 0);
}
```

二、二次曲线的参数拟合法

1．二次曲线的一般参数方程

对于一般的二次多项式，从理论上讲，必定存在着对应的参数方程。假如我们构造出的参数向量方程为：

$$Q(t) = \frac{at^2 + bt + c}{1 + e_1 t + e_2 t^2}, \qquad t \in [0, 1] \tag{3-13}$$

其中 a，b，c 为常数向量，e_1，e_2 为常数。其所对应的代数方程为：

$$\begin{aligned} x(t) &= \frac{a_x t^2 + b_x t + c_x}{1 + e_1 t + e_2 t^2} \\ y(t) &= \frac{a_y t^2 + b_y t + c_y}{1 + e_1 t + e_2 t^2} \end{aligned}, \qquad t \in [0, 1] \tag{3-14}$$

这里，a_x，b_x，c_x，a_y，b_y，c_y 为参数方程的系数。我们可以利用这种形式的参数方程来描述椭

圆、抛物线、双曲线等二次曲线。

通常，我们给定 3 个控制点 P_0, P_1, P_2，并规定曲线的边界条件：

当 $t=0$ 时，曲线过 P_0 点，且切于 $\overline{P_0P_1}$；

当 $t=1$ 时，曲线过 P_2 点，且切于 $\overline{P_1P_2}$。

将边界条件代入式(3-14)，可得到下列关系式：

$$
\begin{aligned}
&c_x = x_0\\
&c_y = y_0\\
&\frac{a_x + b_x + c_x}{1 + e_1 + e_2} = x_2\\
&\frac{a_y + b_y + c_y}{1 + e_1 + e_2} = y_2\\
&b_x - e_1x_0 = k(x_1 - x_0)\\
&b_y - e_1y_0 = k(y_1 - y_0)\\
&\frac{(2a_x + b_x)(1 + e_1 + e_2) - (a_x + b_x + c_x)(e_1 + 2e_2)}{(1 + e_1 + e_2)^2} = l(x_2 - x_1)\\
&\frac{(2a_y + b_y)(1 + e_1 + e_2) - (a_y + b_y + c_y)(e_1 + 2e_2)}{(1 + e_1 + e_2)^2} = l(y_2 - y_1)
\end{aligned}
$$

从而可解出：

$$
\begin{aligned}
&k = 2 + e_1\\
&l = \frac{2 + e_1}{1 + e_1 + e_2}\\
&c_x = x_0\\
&c_y = y_0\\
&b_x = -2x_0 + (2 + e_1)x_1\\
&b_y = -2y_0 + (2 + e_1)y_1\\
&a_x = x_0 - (2 + e_1)x_1 + (1 + e_1 + e_2)x_2\\
&a_y = y_0 - (2 + e_1)y_1 + (1 + e_1 + e_2)y_2
\end{aligned}
\tag{3-15}
$$

因此，只要给定 P_0、P_1、P_2 3 个点坐标，就可以确定曲线的位置。再给定 e_1、e_2，便可确定曲线的形状。一般可以用判别式 $d = e_1^2 - 4e_2$ 来表示曲线的类型：若 $d=0$，为抛物线；$d>0$，为双曲线；$d<0$，为椭圆。如要绘制二次曲线，还应给定步长 dt，使 t 从 0 到 1 变化，求出每点的 x、y 坐标后，再用直线段相连。

2．抛物线的参数拟合

当 $e_1 = e_2 = 0$ 时，式(3-13)变为：

$$
Q(t) = at^2 + bt + c, \qquad t \in [0, 1] \tag{3-16}
$$

所对应的代数方程为：

$$
\begin{aligned}
x(t) &= a_xt^2 + b_xt + c_x\\
y(t) &= a_yt^2 + b_yt + c_y
\end{aligned}
, \qquad t \in [0, 1] \tag{3-17}
$$

若已知三个控制点 $P_0(x_0, y_0)$、$P_1(x_1, y_1)$、$P_2(x_2, y_2)$，则由其决定的抛物线参数方程的系数可由式(3-15)求得：

$$
\begin{aligned}
c_x &= x_0 \\
c_y &= y_0 \\
b_x &= 2(x_1 - x_0) \\
b_y &= 2(y_1 - y_0) \\
a_x &= x_2 - 2x_1 + x_0 \\
a_y &= y_2 - 2y_1 + y_0
\end{aligned}
\tag{3-18}
$$

下面讨论抛物线曲线的两个重要性质。如图 3-15 所示，设参数 $t = 1/2$ 时，曲线上的点为 $P_m(x_m, y_m)$，P_0、P_2 点的连线的中点为 C，其坐标值为 $((x_0 + x_2)/2, (y_0 + y_2)/2)$。

图 3-15　抛物线曲线

性质 1：曲线在 $t = 1/2$ 处的切线平行于 $\overline{P_0P_2}$.

∵
$$
y_x'\big|_{t=1/2} = \left.\frac{y_t'}{x_t'}\right|_{t=1/2} = \left.\frac{2a_y t + b_y}{2a_x t + b_x}\right|_{t=1/2}
$$

$$
= \frac{a_y + b_y}{a_x + b_x}
$$

将(3-18)式代入，则有

$$
y_x'\big|_{t=1/2} = \frac{y_2 - y_0}{x_2 - x_0}
$$

也即证明了，曲线在 $t = 1/2$ 处的切线与 $\overline{P_0P_2}$ 直线平行。

性质 2：P_m 点为 $\overline{P_1C}$ 直线的中点。

∵ $t = 0$，　$x = x_0$，　$y = y_0$，

　$t = 1/2$，$x = x_m$，　$y = y_m$，

　$t = 1$，　$x = x_2$，　$y = y_2$

将其代入式(3-17)中，则得

$$x_0 = c_x$$

$$y_0 = c_y$$

$$x_m = \frac{1}{4}a_x + \frac{1}{2}b_x + c_x$$

$$y_m = \frac{1}{4}a_y + \frac{1}{2}b_y + c_y$$

$$x_2 = a_x + b_x + c_x$$

$$y_2 = a_y + b_y + c_y$$

可以解出

$$\begin{aligned} c_x &= x_0 \\ c_y &= y_0 \\ b_x &= 4x_m - x_2 - 3x_0 \\ b_y &= 4y_m - y_2 - 3y_0 \\ a_x &= 2(x_2 - 2x_m + x_0) \\ a_y &= 2(y_2 - 2y_m + y_0) \end{aligned} \tag{3-19}$$

又因为曲线在 P_0 点的斜率为:

$$y_x'|_0 = \left.\frac{y_t'}{x_t'}\right|_{t=0} = \frac{b_y}{b_x}$$

将式(3-19)代入,得

$$y_x'|_0 = \frac{4y_m - y_2 - 3y_0}{4x_m - x_2 - 3x_0}$$

又 $\because$

$$y_x'|_0 = \frac{y_1 - y_0}{x_1 - x_0}$$

$\therefore$

$$\frac{4y_m - y_2 - 3y_0}{4x_m - x_2 - 3x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

而曲线在 P_2 处的斜率为

$$y_x'|_1 = \frac{2a_y + b_y}{2a_x + b_x}$$

将式(3-19)代入,得

$$y_x'|_1 = \frac{4(y_2 - 2y_m + y_0) + 4y_m - y_2 - 3y_0}{4(x_2 - 2x_m + x_0) + 4x_m - x_2 - 3x_0}$$

又 $\because$

$$y_x'|_1 = \frac{y_2 - y_1}{x_2 - x_1}$$

$\therefore$

$$\frac{4(y_2 - 2y_m + y_0) + 4y_m - y_2 - 3y_0}{4(x_2 - 2x_m + x_0) + 4x_m - x_2 - 3x_0} = \frac{y_2 - y_1}{x_2 - x_1}$$

从而可解出

$$x_m = x_1 + \frac{x_0 + x_2}{2} \Big/ 2$$

$$y_m = y_1 + \frac{y_0 + y_2}{2} \Big/ 2$$

其中,$((x_0+x_2)/2,(y_0+y_2)/2)$即$\overline{P_1P_2}$直线的中点C的坐标。上式即说明了$P_m(t=1/2$处曲线上的点)为$\overline{P_1C}$之中点,即$\overline{P_1P_m}=\overline{P_mc}$。由此可以得出结论:$P_0$、$P_1$、$P_2$与$P_0$、$P_m$、$P_2$三点所构成的抛物线是等价的,前者由式(3-18)确定系数,后者由式(3-19)确定系数,而且后者产生的曲线通过给定的三点。

当给定三个点坐标后,根据式(3-18)或式(3-19)求出相应系数,便可绘制抛物线。由式(3-17)可得抛物线的离散化方程为

$$x_i = a_x t_i^2 + b_x t_i + c_x$$

$$y_i = a_y t_i^2 + b_y t_i + c_y$$

其中,$t_i=i*dt$,$i=1,2,\cdots,N$,N是离散化后所取点的个数,dt为相应的步长。为使t从0到1变化,可令$dt=1/N$。N的选取则可以根据经验,使N正比于D/S,这里S为单位步长,D为图3-15中所示的$\overline{P_mC}$的长度。D反应了抛物线曲率的大小,其值为

$$D = \frac{1}{4}\sqrt{a_x^2 + a_y^2}$$

下面所示为实现上述算法的C语言程序段。

```
Par(int xs,int ys,int xm,int ym,int xe,int ye)
{
    double d,d1,ax,ay,bx,by;
    int n,i;

    ax=(xe-2*xm+xs)*2.0;
    ay=(ye-2*ym+ys)*2.0;
    bx=(xe-xs-ax);
    by=(ye-ys-ay);
    n=sqrt(ax*ax+ay*ay);
    n=sqrt(n*100.0);
    moveto(xs,ys);
    d=1.0/n;
    d1=d;
    for(i=0;i<=n;i++)
      {
        lineto((int)(ax*d1*d1+bx*d1+xs),(int)(ay*d1*d1+by*d1+ys));
        d1+=d;
      }
    lineto(xe,ye);
    return(0);
}
```

用上述离散化公式求得的x_i,y_i,因$i=1,\cdots,N$,共需4N次乘法,4N次加法。若t在0到1闭区内取等步长,即$dt=t_{i+1}-t_i=t_i-t_{i-1}=\cdots$,则我们可以得到递推公式,从而减少计算工作量。若把P_i写成

$$P_i = (at_i + b)t_i + c$$

则有

$$P_i - P_{i-1} = adt(t_i + t_{i-1}) + bdt$$

$$P_{i+1} - P_i = adt(t_{i+1} + t_i) + bdt$$

且

$$(P_{i+1} - P_i) - (P_i - P_{i-1}) = 2adt^2$$

令

$$u = 2adt^2$$

则

$$P_{i+1} = P_i + P_i - P_{i-1} + u$$

也即

$$x_{-1} = (a_x dt - b_x)dt + c_x$$

$$x_0 = c_x$$

$$x_{i+1} = x_i + x_i - x_{i-1} + u_1$$

及

$$y_{-1} = (a_y dt - b_y)dt + c_y$$

$$y_0 = c_y$$

$$y_{i+1} = y_i + y_i - y_{i-1} + u_2$$

其中,

$$u_1 = 2a_x dt^2, \quad u_2 = 2a_y dt^2, \quad i = 1, 2, \cdots, N.$$

因此, 共需 7 次乘法和 6N 次加法。

3. 双曲线的参数拟合

当 $e_1 = 1, e_2 = 0$ 时, 式(3-13)变为

$$Q(t) = \frac{at^2 + bt + c}{1 + t}, \quad t \in [0, 1]$$

所对应的代数方程为

$$x(t) = \frac{a_x t^2 + b_x t + c_x}{1 + t}$$

$$y(t) = \frac{a_y t^2 + b_y t + c_y}{1 + t}$$

若给定三点坐标, 则由其决定的双曲线参数方程的系数可由式(3-15)求得:

$$c_x = x_0$$

$$c_y = y_0$$

$$b_x = -2x_0 + 3x_1$$

$$b_y = -2y_0 + 3y_1$$

$$a_x = x_0 - 3x_1 + 2x_2$$

$$a_y = y_0 - 3y_1 + 2y_2$$

再给定离散的步数 N, 令 $dt = 1/N$, $t_i = idt$, $i = 1, 2, \cdots, N$, 就可求出各点坐标 x_i, y_i。依次连接各点得到的直线集, 即可逼近所要曲线。

4. 椭圆的参数拟合

当 $e_1 = 0, e_2 = 1$ 时, 式(3-13)变为:

$$Q(t) = \frac{at^2 + bt + c}{1 + t^2} \qquad t \in [0, 1]$$

所对应的代数方程为:

$$x(t) = \frac{a_x t^2 + b_x t + c_x}{1 + t^2}$$

$$y(t) = \frac{a_y t^2 + b_y t + c_y}{1 + t^2}$$

若给定三点坐标,则由其决定的椭圆参数方程的系数可由式(3-15)求出:

$$c_x = x_0$$

$$c_y = y_0$$

$$b_x = -2x_0 + 2x_1$$

$$b_y = -2y_0 + 2y_1$$

$$a_x = x_0 - 2x_1 + 2x_2$$

$$a_y = y_0 - 2y_1 + 2y_2$$

第三节　自 由 曲 线

在汽车、飞机、轮船等的计算机辅助设计中,复杂曲线和曲面的设计是一个主要问题。所谓复杂曲线和曲面,指的是形状比较复杂、不能用二次方程描述的曲线和曲面。汽车车身、飞机机翼和轮船船体等的曲线和曲面均属于这一类,一般称为自由曲线和自由曲面。本节我们先讨论自由曲线,自由曲面将在第六章中讨论。

在自由曲线设计中,经常碰到的有以下两类问题。一类是由已知的离散点来决定曲线。另一类问题是已知自由曲线(并且可以显示出来),如何通过交互方式予以修改,使其满足设计者的要求。要解决这两类问题,首先必须研究自由曲线的数学表示形式。完全通过给定点列(称型值点)来构造曲线的方法称为曲线的拟合。求给定型值点之间曲线上的点称为曲线的插值。求出几何形状上与给定型值点列的连接线相近似的曲线称为曲线的逼近,这种曲线不必通过型值点列。

在上一节,我们曾用参数方程来讨论二次曲线的拟合问题。在这一节,参数方程将成为描述自由曲线的主要形式。例如,二维空间曲线上的一个点用 $P = [x(t), y(t)]$ 表示,三维空间曲线上的一个点用 $p = [x(t), y(t), z(t)]$ 表示。

用参数方程来描述自由曲线有什么优点呢?

第一,被描述的自由曲线的形状本质上与坐标系的选取无关。如果我们要通过一系列的点拟合一条曲线,那么,容易看出,曲线的形状决定于这些点本身之间的关系,而和这些点与其所在坐标系的关系无关。参数方程正好突出了自由曲线本身的这一性质。

第二,参数方程将自变量和因变量完全分开,使得参数变化对各因变量的影响可以明显地表示出来。

第三,任何曲线在坐标系中都会在某一位置上出现垂直的切线,因而导致无穷大斜率。而在参数方程中,可以用对参数求导来代替,即: $\frac{dy}{dx} = \frac{dy/dt}{dx/dt}$,从而避免了这一问题。前

面这一关系式还说明,斜率与切线矢量的长度无关,即:

$$\frac{dy}{dx} = \frac{k\,¡¤dy/dt}{k\,¡¤dx/dt} = \frac{l\,¡¤dy/dt}{l\,¡¤dx/dt} = \cdots$$

第四,用参数方程表示曲线时,因为参变量是规范化的,即将其值限制在由 0 到 1 这一闭区间之内(用 $t\in[0,1]$ 表示),因而所表示的曲线总是有界的,不需要另设其它的几何数据来定义其边界。

空间一条自由曲线可以用三次参数方程表示如下:

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z \end{aligned} \qquad (3\text{-}20)$$

其中,t 为参数,且 $0\le t\le 1$。$t=0$ 时,对应曲线段的起点,$t=1$ 时,对应曲线段的终点。

为什么我们用三次参数方程来表示自由曲线呢?这是因为,我们既要使曲线段的端点通过特定的点,又使曲线段在连接处保持位置和斜率的连续性,而三次参数方程是能表示这种曲线的最低阶次的方程。当然,用更高次的参数方程也是可以的,但是计算较为复杂,而且会产生不必要的扭摆。

一、抛物线参数样条曲线

抛物线参数样条曲线完全通过给定的型值点列,其基本方法是:给定 N 个型值点 $P_1, P_2, \cdots, P_N$,对相邻三点 P_i, P_{i+1}, P_{i+2} 及 $P_{i+1}, P_{i+2}, P_{i+3}$,$i=1\cdots N-2$,反复用抛物线算法拟合,然后再对此相邻抛物线曲线在公共区间 P_{i+1} 到 P_{i+2} 范围内,用权函数 t 与 $1-t$ 进行调配,使其混合为一条曲线,可表示为:

$$S = \sum_{i=1}^{N-2} [(1-t)S_i + tS_{i+1}], \qquad t\in[0,1]$$

其中,S_i 即 P_i, P_{i+1}, P_{i+2} 三点决定的抛物线曲线,S_{i+1} 为 $P_{i+1}, P_{i+2}, P_{i+3}$ 三点决定的抛物线曲线,混合后的曲线 S 在 P_{i+1} 到 P_{i+2} 公共段内,是 S_i 的后半段与 S_{i+1} 的前半段加权混合的结果。S 曲线在公共段内具体的参数方程可写为:

$$\begin{aligned} x &= (1-2t_2)(a_{1x}t_1^2 + b_{1x}t_1 + c_{1x}) + 2t_2(a_{2x}t_2^2 + b_{2x}t + c_{2x}) \\ y &= (1-2t_2)(a_{1y}t_1^2 + b_{1y}t_1 + c_{1y}) + 2t_2(a_{2y}t_2^2 + b_{2y}t + c_{2y}) \end{aligned}$$

其中,

$$t_2\in[0,0.5],$$
$$t_1 = t_2 + 0.5 \quad \in[0.5,1]$$

式中:$a_{1x}, b_{1x}, c_{1x}, a_{1y}, b_{1y}, c_{1y}$ 为 S_i 段曲线的系数,由 P_i, P_{i+1}, P_{i+2} 三点决定,参变量为 t_1,在公共段范围内 $t_1=0.5\sim 1$;

$a_{2x}, b_{2x}, c_{2x}, a_{2y}, b_{2y}, c_{2y}$ 为 S_{i+1} 段曲线的系数,由 $P_{i+1}, P_{i+2}, P_{i+3}$ 三点决定,参变量为 t_2,在公共段范围内 $t_2=0\sim 0.5$。

显然,当 $t_1=0.5, t_2=0$ 时,$S=S_i$;

当 $t_1=1.0, t_2=0.5$ 时,$S=S_{i+1}$.

详见图 3-16。

请读者自己证明: 用这种方法拟合的自由曲线, 在 P_2 到 P_{N-1} 各已知点的左、右侧能达到一阶导数连续, 即 $C^{(1)}$ 连续。而当曲线两端没有一定的端点条件限制时, 则曲线两端各有一段曲线不是加权混合的形式, 而只是 S_1 段的前半段和 S_{N-2} 段的后半段, 如图 3-17 所示。

二、Hermite 曲线

1. 参数方程

为了用三次参数方程描述一条自由曲线段, 必须根据给定条件求出式(3-20)中的系数。Hermite 曲线是给定曲线段的两个端点坐标

图 3-16　抛物线参数样条曲线的拟合

图 3-17　抛物线参数样条曲线

P_0、P_1 以及两端点处的切线矢量 R_0、R_1 来描述曲线的。我们将(3-20)式写成矩阵形式:

$$Q(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{matrix} a \\ b \\ c \\ d \end{matrix} \tag{3-21}$$

其 x 分量可表示为:

$$x(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{matrix} a \\ b \\ c \\ d \end{matrix}_x$$

若令:

$$T = [t^3 \quad t^2 \quad t \quad 1]$$

$$C_x = [a \quad b \quad c \quad d]_x^T$$

则

$$x(t) = T \cdot C_x \tag{3-22}$$

另外,

$$x'(t) = [3t^2 \quad 2t \quad 1 \quad 0] \cdot C_x \tag{3-23}$$

将给定边界条件

$$x(0) = P_{0x},\ x(1) = P_{1x},\ x'(0) = R_{0x},\ x'(1) = R_{1x}$$

代入(3-22)、(3-23)式中, 得:

$$P_{0x} = [0 \quad 0 \quad 0 \quad 1] \cdot C_x$$
$$P_{1x} = [1 \quad 1 \quad 1 \quad 1] \cdot C_x$$
$$R_{0x} = [0 \quad 0 \quad 1 \quad 0] \cdot C_x$$
$$R_{1x} = [3 \quad 2 \quad 1 \quad 0] \cdot C_x$$

用矩阵方程可表示为:

$$\begin{bmatrix} P_0 \\ P_1 \\ R_0 \\ R_{1x} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot C_x \tag{3-24}$$

对(3-24)式两端乘以 4×4 矩阵的逆阵, 可得:

$$C_x = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ R_0 \\ R_{1x} \end{bmatrix} \tag{3-25}$$

令:

$$M_h = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{3-26}$$

即 Hermite 矩阵, 为常数, 以及

$$G_h = [P_0 \quad P_1 \quad R_0 \quad R_1]^T \tag{3-27}$$

为 Hemite 几何矢量, 则(3-25)式可表示为:

$$C_x = M_h \cdot G_{hx}$$

而(3-22)式可表示为:

$$x(t) = T \cdot M_h \cdot G_{hx}$$

(3-21)式可表示为:

$$Q(t) = T \cdot M_h \cdot G_h, \qquad t \in [0, 1] \tag{3-28}$$

显然, 只要给定 G_h, 就可在 $0 \le t \le 1$ 范围内求出 $Q(t)$, 对于不同的初始条件, G_h 是不同的, 而 T、M_h 均是相同的。

2. 调合函数

式(3-28)中的 $T \cdot M_h$ 称为调和函数, 如令其为 $F_h(t)$, 则各分量可重写如下:

$$F_{h1}(t) = 2t^3 - 3t^2 + 1$$
$$F_{h2}(t) = -2t^3 + 3t^2$$
$$F_{h3}(t) = t^3 - 2t^2 + t$$
$$F_{h4}(t) = t^3 - t^2$$

这些分量对 $Q(0)$, $Q(1)$, $Q'(0)$, $Q'(1)$ 起作用, 使在整个参数域范围内产生曲线的值, 从而构成赫米特曲线。调和函数有如下重要性质:

(1) 调和函数仅与参数值 t 有关，而与初始条件无关。

(2) 调和函数对于物体空间三个坐标值(x, y, z)是相同的。

(3) 当处于参数域的边界时，调和函数各分量中仅有一个起作用。即 t= 0 时，$F_{h1}(t) = 1$，而 $F_{h2}(t) = F_{h3}(t) = F_{h4}(t) = 0$；当 t= 1 时，$F_{h2}(t) = 1$，而 $F_{h1}(t) = F_{h3}(t) = F_{h4}(t) = 0$。

图 3-18 表示出 Hermite 曲线的调和函数随参数 t 而变化的曲线。以后将会看到，不同类型的三次参数曲线的调和函数也是不同的。

3. 切线矢量

下面再讨论一下两端点处的切线向量对曲线形状的影响。

在图 3-19 中，$Q'(0)$，$Q'(1)$ 表示起始点及终止点处的切线向量。令单位矢量

$$E_0 = \frac{Q'(0)}{|Q'(0)|} \qquad E_1 = \frac{Q'(1)}{|Q'(1)|}$$

图 3-18 Hermite 曲线的调和函数

于是有：

$$|E_0| = \sqrt{E_{0x}^2 + E_{0y}^2 + E_{0z}^2} = 1$$

$$|E_1| = \sqrt{E_{1x}^2 + E_{1y}^2 + E_{1z}^2} = 1$$

如果令：

$$|Q'(0)| = k_0, \qquad |Q'(1)| = k_1,$$

则有：

$$Q'(0) = k_0 E_0, \qquad Q'(1) = k_1 E_1$$

同时可得：

$$G_h = [Q(0) \quad Q(1) \quad k_0E_0 \quad k_1E_1]^T$$

显然，如果切线向量的方向和长度都是已知的，那么它的三个分量也就全部确定了，由式(3-28)即可决定 Hermite 曲线。如果仅仅给出切线的方向，即该切线向量的方向余弦，而未给出切线向量的长度，那么就不能确定 Hermite 曲线。尽管曲线的端点及端点处切线的方向是一定的，由于切线向量的长度不同(随 k_0, k_1 而变化)，Hermite 曲线会具有完全不同的形状。图 3-20 表示出当 k_1= 10，k_0 由 10 变化到 80 时对 Hermite 曲线的影响。图中曲线的参数为：

$$Q(0) = P_0 = [x_0 \quad y_0 \quad z_0] = [4.0 \quad 4.0 \quad 0.0]$$

$$Q(1) = P_1 = [x_1 \quad y_1 \quad z_1] = [24.0 \quad 4.0 \quad 0.0]$$

$$Q'(0) = R_0 = k_0[E_{0x} \quad E_{0y} \quad E_{0z}] = k_0[0.8320 \quad 0.5547 \quad 0.0]$$

$$Q'(1) = R_1 = k_1[E_{1x} \quad E_{1y} \quad E_{1z}] = k_1[0.8320 \quad -0.5547 \quad 0.0]$$

图 3-19　Hermite 曲线的切线向量

图 3-20　切线向量的长度对 Hermite 曲线形状的影响

4．自由曲线的连接

当许多自由曲线段首尾相接构成一条自由曲线时，如何保证曲线连接处具有合乎要求的连续性是一个关键问题。当曲线段 1 的端点和曲线段 2 的端点不相重合时，这两个曲线段是不连续的，实际上它们仍然是两个曲线段（如图 3-21(a)）。如果两个曲线段具有一个公共的端点，那么这两个曲线段是连续的，且在连接处至少为 $C^{(0)}$ 连续（图 3-21(b)）。如果这两个曲线段不仅具有公共的端点，而且在连接处其切线向量共线，那么这两个曲线段就为 $C^{(1)}$ 连续（图 3-21(c)）。$C^{(2)}$ 连续的必要条件之一是两曲线连接处的曲率相等。

假设两条不相连的 Hermite 参数曲线段 1 和 3 的边界条件 G_{h1}、G_{h3} 已经给定，如果要使参数曲线段 2 与曲线段 1、3 之间的连接均为 $C^{(1)}$ 连续，那么曲线段 2 的边界条件 G_{h2} 应该是什么呢？

$$G_{h1} = [Q_1(0) \quad Q_1(1) \quad Q_1'(0) \quad Q_1'(1)]^T$$

$$G_{h3} = [Q_3(0) \quad Q_3(1) \quad Q_3'(0) \quad Q_3'(1)]^T$$

式中,各符号的下标表示参数曲线段的序号。为了保证曲线段 1、2 之间及 2、3 之间的 $C^{(1)}$ 连续,必须有

$$Q_2(0) = Q_1(1), \qquad Q_2(1) = Q_3(0),$$

$$Q_2'(0) = a\frac{Q_1'(1)}{\|Q_1'(1)\|}$$

$$Q_2'(1) = b\frac{Q_3'(0)}{\|Q_3'(0)\|}$$

于是可得

$$G_{h2} = Q_1(1) \quad Q_3(0) \quad a\frac{Q_1'(1)}{\|Q_1'(1)\|} \quad b\frac{Q_3'(0)}{\|Q_3'(0)\|}^T$$

其中,a,b 是正的比例因子。

图 3-21 自由曲线连续性示意图

与此类似,可得到曲线段 i 与曲线段 i- 1 及曲线段 i+ 1 实现 $C^{(1)}$ 连续的条件为:

$$G_{hi} = Q_{i-1}(1) \quad Q_{i+1}(0) \quad a\frac{Q_{i-1}'(1)}{\|Q_{i-1}'(1)\|} \quad b\frac{Q_{i+1}'(0)}{\|Q_{i+1}'(0)\|}^T$$

三、三次参数样条曲线

Hermite 曲线比较简单,易于理解,但要求使用者给出两端点处的切线向量作为初始条件,这是很不方便的,有时甚至是难于作到的。可它却是应用很广泛的三次参数样条曲线的基础。

什么是样条及样条曲线呢? 在汽车、飞机、轮船等的设计中,常常碰到这样一个问题:平面上给出一组离散的有序点列,要求用一条光滑曲线把这些点顺序连接起来。长期以来,绘图员常常用一根富有弹性的均匀细木条或有机玻璃条,用压铁将它压在各型值点处,从而强迫它通过这些点,最后沿这根被称为"样条"的细木条画出所需要的光滑曲线,这就是样条曲线。若把小木条看成弹性细梁,压铁看成作用于梁上的集中载荷,则按上述方法画出的光滑曲线在力学上即可模拟弹性细梁在外加集中载荷作用下的弯曲变形曲线。下面我们可以看到,对于这种弯曲变形曲线分段来说,和 Hermite 曲线是一样的。

1. Hermite 曲线的二阶导数形式

由式(3-28)知,Hermite 曲线可写成:

$$Q(t) = F_{h1}(t)Q(0) + F_{h2}(t)Q(1) + F_{h3}(t)Q'(0) + F_{h4}(t)Q'(1) \qquad (3\text{-}29)$$

其中,$Q(0)=P_0, Q(1)=P_1, Q'(0)=R_0, Q'(1)=R_1$ 对式(3-29)两侧对 t 求导,得:

$$Q'(t) = F_{h1}'(t)Q(0) + F_{h2}'(t)Q(1) + F_{h3}'(t)Q'(0) + F_{h4}'(t)Q'(1)$$

再次对 t 求导,得:

$$Q''(t) = F_{h1}''(t)Q(0) + F_{h2}''(t)Q(1) + F_{h3}''Q'(0) + F_{h4}''(t)Q'(1)$$

式中,

$$F_{h1}''(t) = 12t - 6$$

$$F_{h2}''(t) = -12t + 6$$

$$F_{h3}''(t) = 6t - 4$$

$$F_{h4}''(t) = 6t - 2$$

故有

$$Q''(0) = -6Q(0) + 6Q(1) - 4Q'(0) - 2Q'(1) \tag{3-30}$$

$$Q''(1) = 6Q(0) - 6Q(1) + 2Q'(0) + 4Q'(1) \tag{3-31}$$

将上述二方程联立求解,得出 $Q'(0)$ 及 $Q'(1)$ 如下:

$$Q'(0) = -Q(0) + Q(1) - \frac{1}{6}(2Q''(0) + Q''(1)) \tag{3-32}$$

$$Q'(1) = -Q(0) + Q(1) + \frac{1}{6}(Q''(0) + 2Q''(1)) \tag{3-33}$$

将式(3-32)及式(3-33)代入式(3-29)可得:

$$Q(t) = (1-t)Q(0) + tQ(1) + \frac{1}{6}(-t^3 + 3t^2 - 2t)Q''(0) + \frac{1}{6}(t^3 - t)Q''(1) \tag{3-34}$$

这就是以曲线段两个端点的位置向量及端点处的二阶导数表示的三次参数方程。同时,该方程也给出一组新的调和函数。根据力学原理,经分析可知,由(3-34)式表示的三次参数曲线的 y 坐标值对应于弹性细梁的变形 δ,三次参数曲线的一阶导数对应于弹性细梁的斜率 α,而二阶导数则对应于弹性细梁的曲率。如果用(3-34)式的三次参数曲线方程来描述弹性细梁的变形曲线,则其结果与力学中给出的弹性梁变形方程是完全一致的。因此我们可以用 Hermite 三次参数曲线来描述传统的样条曲线。

2. 连续的三次参数样条曲线

现有几个点组成的一组离散点列 $P_1, P_2, \cdots, P_{i-1}, P_i, P_{i+1}, \cdots, P_n$(见图 3-22),现要求将一系列 Hermite 三次系数曲线段连接起来,使其通过这些点列,构成一条三次参数样条曲线,且在所有曲线的连接处均具有位置、切线向量和二阶导数的连续性。

图 3-22　一组离散点列

我们假定,上述点列中每相邻两点 P_i, P_{i+1}($1 \leq i \leq n-1$),组成 Hermite 曲线的一对起点和终点,那么,n 个点共有 n-1 段曲线,第 i 段曲线的起点和终点分别为 P_i 和 P_{i+1}。

式(3-30)和式(3-31)给出 Hermite 三次参数曲线两个端点处的二阶导数,对于第 i 段曲线而言,此二式中的 $Q(0)$ 即为 P_i,$Q(1)$ 即为 P_{i+1}。由此可得第 i 段 Hermite 参数曲线两个端点处的二阶导数如下:

$$P_i''= -\ 6P_i + 6P_{i+1} - 4P_i' - 2P_{i+1}' \tag{3-35}$$

$$P_{i+1}'' = 6P_i - 6P_{i+1} + 2P_i' + 4P_{i+1}' \tag{3-36}$$

式中 P_i 表示曲线在 P_i 点的切线向量，P_i 表示曲线在 P_i 点的二阶导数。同理，第 $i+1$ 段曲线也有如下类似的两个式子：

$$P_{i+1}''_1 = -\ 6P_{i+1} + 6P_{i+2} - 4P_{i+1}' - 2P_{i+2}' \tag{3-37}$$

$$P_{i+1}''_2 = 6P_{i+1} - 6P_{i+2} + 2P_{i+1}' + 4P_{i+2}' \tag{3-38}$$

因为相邻曲线段的公共端点处二阶导数连续，故式(3-36)和式(3-37)右端应相等，即：

$$6P_i - 6P_{i+1} + 2P_i' + 4P_{i+1}' = -\ 6P_{i+1} + 6P_{i+2} - 4P_{i+1}' - 2P_{i+2}'$$

化简后得：

$$P_i' + 4P_{i+1}' + P_{i+2}' = 3(P_{i+2} - P_i) \tag{3-39}$$

对于 n 个点，可以得出 $n-2$ 个类似的方程，即有 $1\leq i\leq n-2$。不过，这组联立方程中有 n 个未知数。为了求解这组联立方程，还必须给出两个边界条件，即给出此三次样条曲线起点和终点处的切线向量或二阶导数。曲线两端的边界条件有多种形式，主要根据实际问题的物理要求对曲线两端给出的约束条件而定。常用的约束条件有自由端、夹持端、抛物端等三种，现分别叙述如下：

(1) 自由端

在这种情况下，两端点处的二阶导数为零，即 $P_1''= P_n''= 0$。

由式(3-35)可得出

$$2P_1' + P_2' = 3(P_2 - P_1) \tag{3-40}$$

由式(3-36)可得出

$$P_{n-1}' + 2P_n' = 3(P_n - P_{n-1}) \tag{3-41}$$

于是，由式(3-39)、式(3-40)、式(3-41)三式可写出自由端三次参数样条曲线的矩阵表示式如下：

$$\begin{array}{cccccc|c|c}
2 & 1 & 0 & 0 & \cdots & 0 & P_1' & 3(P_2 - P_1) \\
1 & 4 & 1 & 0 & \cdots & 0 & P_2' & 3(P_3 - P_1) \\
0 & 1 & 4 & 1 & 0\cdots & 0 & P_3' & 3(P_4 - P_2) \\
\hline
 & & & & & & \vdots & \vdots \\
\hline
0 & 0 & \cdots & \cdots & 1 & 4 & P_{n-1}' & 3(P_n - P_{n-2}) \\
0 & 0 & \cdots & \cdots & 0 & 1 & P_n' & 3(P_n - P_{n-1})
\end{array} \tag{3-42}$$

(2) 夹持端

根据实际问题的要求，给出两端的切线向量，即 $P_1' = k_1E_1$，$P_n' = k_nE_n$。其中，E_1，E_n 为单位向量。于是，可写出夹持端三次参数样条曲线的矩阵表达式

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ & & & & & \\ 0 & \cdots & 0 & 1 & 4 & 1 \\ 0 & 0 & \cdots & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} P_1' \\ P_2' \\ \vdots \\ P_{n-1}' \\ P_n' \end{bmatrix} = \begin{bmatrix} k_1 E_1 \\ 3(P_3 - P_1) \\ 3(P_4 - P_2) \\ \vdots \\ 3(P_n - P_{n-2}) \\ k_n E_n \end{bmatrix} \tag{3-43}$$

(3) 抛物端

假设曲线的第一段和第 n- 1 段(末段)为抛物线,也就是说,此二段曲线的二阶导数为常数,即 $P_1'' = P_2''$, $P_{n-1}'' = P_n''$。因此,可由(3-35)式及(3-36)式得出如下关系式:

$$P_1' + P_2' = 2(P_2 - P_1)$$

$$P_{n-1}' + P_n' = 2(P_n - P_{n-1})$$

于是,可写出抛物端三次参数样条曲线的矩阵表达式为:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & \cdots & 0 \\ & & & & & & \\ 0 & 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} P_1' \\ P_2' \\ \vdots \\ P_{n-1}' \\ P_n' \end{bmatrix} = \begin{bmatrix} 2(P_2 - P_1) \\ 3(P_3 - P_1) \\ 3(P_4 - P_2) \\ \vdots \\ 3(P_n - P_{n-2}) \\ 2(P_n - P_{n-1}) \end{bmatrix} \tag{3-44}$$

如果给定其它类型的边界条件,也可推导出相应的三次参数样条曲线的矩阵表达式。在得出三次参数样条曲线的矩阵表达式后,即可用“追赶法”或其它方法求解此三对角方程组,得出各型值点处的切线向量 $P_i'(1 \le i \le n)$。将各点的切线向量连同各点的位置向量 $P_i(1 \le i \le n)$依次分段代入 Hermite 三次参数曲线方程(3-29),按照作图需要计算出各线段内的若干插值点,并依次用直线相连,即可画出所求的三次参数样条曲线。

最后还要说明一点,在以上讨论的三次参数样条曲线中,每一曲线段参数值 t 的取值范围均为 0～1。这一最简单的办法只适用于型值点分布比较均匀的场合。当型值点间的间隔极不均匀时,其效果就很差,在间隔较大的一段,曲线相当扁平而靠近弦线,在间隔很小的一段,曲线可能出现扭曲。解决这一问题的办法是:全曲线段端点之间的弦 长等于其参数值 t,即 $t \in [0, L_i]$,其中 L_i 为第 i 段曲线两端点间的弦长。这时,可以推导出曲线段的调和函数为:

$$F_{h1}(t) = 2\frac{t^3}{L_i^3} - 3\frac{t^3}{L_i^2} + 1$$

$$F_{h2}(t) = -2\frac{t^3}{L_i^3} + 3\frac{t^2}{L_i^2}$$

$$F_{h3}(t) = \frac{t^3}{L_i^2} - 2\frac{t^2}{L_i} + t$$

$$F_{h4}(t) = \frac{t^3}{L_i^2} - \frac{t^2}{L_i}$$

若取 $t = dL_i$,则

$$F_{h3}(t) = (\alpha^3 - 2\alpha^2 + \alpha)\,¡¤L_i$$

$$F_{h4}(t) = (\alpha^3 - \alpha^2)\,¡¤L_i$$

均与弦长 L_i 有关。这就是说,以弦长为参数值 t 时,作用于切线向量的调和函数与弦长本身成正比,从而改善了曲线形状。关于这一问题,有兴趣的读者可参阅有关文献。

四、Bezier 曲线

前面讨论过的三次参数样条曲线通过了给定的型值点。但是在工程设计中,开始给出的型值点以及由此而产生的样条曲线有时尚不能满足性能或美观的要求,需要加以修改。参数样条曲线不能直观表示出应该如何修改以及如何控制曲线的形状。如果我们改变了一个或多个型值点,则可能出现并非所希望的曲线形状。因此,样条曲线作为外形设计的工具,还缺少灵活性和直观性。针对这一问题,法国雷诺汽车公司的 Bezier 提出了一种新的参数曲线表示方法,称为 Bezier 曲线。这种方法能使设计者比较直观地意识到所给条件与所产生的曲线之间的关系,能比较方便地通过修改输入参数来改变曲线的形状和阶次。此外,这种方法所用数学工具比较简单,易于为广大设计师和绘图员所接受。

Bezier 曲线是通过一组多边折线的各顶点唯一地定义出来的。在多边折线的各顶点中,只有第一点和最后一点在曲线上,其余的顶点则用以定义曲线的阶次和形状。曲线的形状趋向于多边折线的形状,改变多边折线顶点的位置和改变曲线形状有密切的联系。因此,多边折线又称为特征多边形,其顶点称为控制点。图 3-23 表示出 Bezier 曲线及其特征多边形。

图 3-23 Bezier 曲线及其特征多边形

1. Bezier 曲线的数学表示式

Bezier 曲线的数学基础是在第一个和最后一个端点之间进行插值的多项式调和函数。通常,将 Bezier 曲线段以参数方程表示如下:

$$Q(t) = \sum_{i=0}^{n} P_i B_{i,n}(t), \qquad t \in [0,1] \tag{3-45}$$

这是一个 n 次多项式,具有 n+1 项。其中,P_i($i = 0, 1, \cdots, n$)表示特征多边形 n+1 个顶点的位置向量,$B_{i,n}(t)$是伯恩斯坦(Berstein)多项式,称为基底函数,可表示如下:

$$B_{i,n}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}, \qquad i = 0, 1, \cdots, n \tag{3-46}$$

请注意:当 $i = 0, t = 0$ 时,$t^i = 1, 0! = 1$。

由式(3-45)及式(3-46)可推出一次、二次及三次 Bezier 曲线的数学表示式。

(1) 一次 Bezier 曲线

n= 1, 一次多项式, 有两个控制点, 则:

$$Q(t) = \sum_{i=0}^{1} P_i B_{i,1}(t)$$

$$= (1 - t)P_0 + tP_1, \qquad t \in [0, 1] \tag{3-47}$$

这说明, 一次 Bezier 曲线是连接起点 P_0 和终点 P_1 的直线段。

(2) 二次 Bezier 曲线

n= 2, 二次多项式, 有三个控制点, 则:

$$Q = \sum_{i=0}^{2} P_i B_{i,2}(t)$$

$$= (1 - t)^2 P_0 + 2t(1 - t)P_1 + t^2 P_2, \qquad t \in [0, 1]$$

$$= (P_2 - 2P_1 + P_0)t^2 + 2(P_1 - P_0)t + P_0 \tag{3-48}$$

如令 $a = P_2 - 2P_1 + P_0$, $b = 2(P_1 - P_0)$, $c = P_0$, 则(3-48)式即为: $Q(t) = at^2 + bt + c$, 此式即(3-16)式, 即为一条抛物线。说明二次 Bezier 曲线为抛物线, 其矩阵形式为:

$$Q(t) = [t^2 \; t \; 1] \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix}, \qquad t \in [0, 1]$$

(3) 三次 Bezier 曲线

n= 3, 三次多项式, 有四个控制点, 则有:

$$Q(t) = \sum_{i=0}^{3} P_i B_{i,3}(t)$$

$$= (1 - t)^3 P_0 + 3t(1 - t)^2 P_1 + 3t^2(1 - t)P_2 + t^3 P_3$$

$$= B_{0,3}(t)P_0 + B_{1,3}(t)P_1 + B_{2,3}(t)P_2 + B_{3,3}(t)P_3, \tag{3-49}$$

其中

$$B_{0,3}(t) = (1 - t)^3$$

$$B_{1,3}(t) = 3t(1 - t)^2$$

$$B_{2,3}(t) = 3t^2(1 - t)$$

$$B_{3,3}(t) = t^3$$

称为三次 Bezier 曲线的调和函数, 构成图 3-24 所示的 4 条曲线。这 4 条曲线均是三次曲线, 形成 Bezier 曲线的一组基。任何三次 Bezier 曲线都是这 4 条曲线的线性组合。

式(3-49)可用矩阵形式表示为:

$$Q(t) = [t^3 \; t^2 \; t \; 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}, \quad t \in [0, 1]$$

可记为:

$$Q(t) = T \cdot M_b \cdot G_b$$

其中

图 3-24　三次 Beier 曲线的调和函数

且

$$M_b = \begin{matrix} T = [t^3 & t^2 & t & 1] \\ -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{matrix}$$

为三次 Bezier 曲线系数矩阵。而

$$G_b = [P_0 \quad P_1 \quad P_2 \quad P_3]^T$$

为三次 Bezier 曲线的四个控制点位置矢量。

2. Bezier 曲线的性质

(1) 端点及端点切线

由式(3-45)可得出 Bezier 曲线两端点的值:

t= 0:

$$Q(0) = \sum_{i=0}^{n} P_i B_{i,n}(0)$$

$$= P_0 B_{0,n}(0) + P_1 B_{1,n}(0) + \cdots + P_n B_{n,n}(0)$$

$$= P_0$$

t= 1:

$$Q(1) = \sum_{i=0}^{n} P_i B_{i,n}(1)$$

$$= P_0 B_{0,n}(1) + P_1 B_{1,n}(1) + \cdots + P_n B_{n,n}(1)$$

$$= P_n$$

这说明, Bezier 曲线通过特征多边形的起点和终点。

将式(3-46)对参数 t 求导, 得:

$$B'_{i,n}(t) = \frac{n!}{i!(n-i)!}\{it^{i-1}(1-t)^{n-i} - (n-i)(1-t)^{n-i-1}t^i\}$$

$$= \frac{n(n-1)!}{(i-1)![(n-1)-(i-1)]!}t^{i-1}(1-t)^{(n-1)-(i-1)}$$

$$- \frac{n(n-1)!}{i![(n-1)-i]!}t^i(1-t)^{(n-1)-i}$$

$$= n\,B_{i-1,n-1}(t) - B_{i,n-1}(t)$$

故由式(3-45)可得:

$$Q'(t) = n\sum_{i=0}^{n} P_i[B_{i-1,n-1}(t) - B_{i,n-1}(t)]$$

$$= n\{(P_1 - P_0)B_{0,n-1}(t) + (P_2 - P_1)B_{1,n-1}(t)$$

$$+ \cdots + (P_n - P_{n-1})B_{n-1,n-1}(t)\}$$

$$= n\sum_{i=1}^{n}(P_i - P_{i-1})B_{i-1,n-1}(t)$$

在起始点, $t=0$, $B_{0,n-1}(0)=1$, 其余项均为0, 故有:

$$Q'(0) = n(P_1 - P_0) \tag{3-50}$$

在终止点, $t=1$, $B_{n-1,n-1}(1)=1$, 其余项均为0, 故有:

$$Q'(1) = n(P_n - P_{n-1}) \tag{3-51}$$

对于三次Bezier曲线, $n=3$ 所以

$$Q'(0) = 3(P_1 - P_0)$$

$$Q'(1) = 3(P_3 - P_2)$$

这是一个很重要的性质, 它说明Bezier曲线在始点和终点处的切线方向与特征多边形的第一条边及最后一条边的走向一致。

(2) 对称性

假如保持n次Bezier曲线诸顶点的位置不变, 而把次序颠倒过来, 即下标为i的点(P_i)改为下标为$n-i$的点(P_{n-i}), 则此时曲线仍不变, 只不过曲线的走向相反而已。这一性质可证明如下:

由伯恩斯坦多项式可以导出:

$$B_{i,n}(t) = \frac{n!}{i!(n-i)!}t^i(1-t)^{n-i} = B_{n-i,n}(1-t)$$

记次序颠倒以后的顶点为P_i^*, 则有

$$P_i^* = P_{n-i}, \quad i = 0, 1, 2, \cdots, n$$

此时, 设新的Bezier曲线为$Q^*(t)$, 则

$$Q^*(t) = \sum_{i=0}^{n} P_i^* B_{i,n}(t) = \sum_{i=0}^{n} P_{n-i}B_{i,n}(t)$$

令$n-i=k$, 则$i=n-k$, 且$i=0$时, $k=n$及$i=n$, $k=0$

$$\therefore \quad Q^*(t) = \sum_{k=n}^{0} P_k B_{n-k,n}(t)$$

再将k换成i, 则

$$Q^*(t) = \sum_{i=n}^{0} P_i B_{n-i,n}(t)$$

又 ∵ $$B_{i,n}(t) = B_{n-i,n}(1-t)$$

∴ $$B_{i,n}(1-t) = B_{n-i,n}(t)$$

∴ $$Q^*(t) = \sum_{i=n}^{0} P_i B_{i,n}(1-t)$$

$$= \sum_{i=0}^{n} P_i B_{i,n}(1-t)$$

$$= Q(1-t)$$

(3) 凸包性

∵ $$\sum_{i=0}^{n} B_{i,n}(t) = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$$

$$= [(1-t)+t]^n \equiv 1$$

这一结果告诉我们，当 t 在[0, 1]区间变化时，对于任一 t 值，伯恩斯坦多项式各项之和均为 1，且有：

$$B_{i,n}(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i} \geq 0$$

因此，$B_{i,n}(t)$构成了 Bezier 曲线的权函数。对于某一 t 值，$Q(t)$是特征多边形各顶点 P_i，($i = 0, 1, \cdots, n$)的加权平均，权因子依次是 $B_{i,n}(t)$，($i = 0, 1, \cdots, n$)且均为正数。在几何图形上，这意味着 Bezier 曲线各点均应落在特征多边形各顶点构成的凸包之中，此处凸包是指包含所有顶点的最小凸多边形。用数学归纳法可以证明这一结论。

(4) 几何不变性

曲线的形状仅与特征多边形各顶点的相对位置有关，而与坐标系的选择无关。

3．三次 Bezier 样条曲线

由前面所讨论的 Bezier 曲线段的数学表示式及其性质可以看出，对于形状比较复杂的曲线来说，只用一段三次 Bezier 曲线来描述就不够了。一种办法是增加顶点的个数，从而也就增加了 Bezier 曲线的阶次，但是高次 Bezier 曲线计算比较复杂，而且还有许多问题有待于理论上解决。因此，工程上往往使用另一种方法，即用分段三次 Bezier 样条曲线来描述。将分段的三次 Bezier 曲线连接起来构成三次 Bezier 样条曲线，其关键问题是如何保证连接处具有 C^1 及 C^2 连续性。

设有两 Bezier 曲线段 $Q_1(t)$及 $Q_2(t)$，其多边折线顶点分别为 P_0, P_1, P_2, P_3 及 R_0, R_1, R_2, R_3，而且 $P_3 = R_0$。若要求两曲线段在连接点 $P_3(R_0)$处实现 C^1 连续(如图 3-25 所示)，那么应该具备什么条件呢?

因 $Q_1'(1) = 3(P_3 - P_2)$

$Q_2'(0) = 3(R_1 - R_0)$

为实现 C^1 连续，应使

$$Q_2'(0) = \alpha Q_1'(1)$$

亦即：

$$R_1 - R_0 = \alpha(P_3 - P_2) \qquad (3\text{-}52)$$

式中 α为一比例因子。这就是说,实现 C^1 连续的条件是 P_2, $P_3(R_0)$, R_1 在一条直线上,而且 P_2, R_1 应在 $P_3(R_0)$ 的两侧。

两曲线实现 C^2 连续的充要条件为:

① 在连接处两曲线的密切平面重合。

② 在连接处两曲线的曲率相等。

关于这一问题我们不再作进一步的讨论,有举趣的读者可阅读有关的参考文献。

五、B 样条曲线

以伯恩斯坦多项式为基础的 Bezier 曲线具有许多优点,但是都存在以下两个问题:第一,特征多边形顶点的数量决定了 Bezier 曲线的阶次,即 n 个顶点的特征多边形必然产生 n - 1 次 Bezier 曲线,这是不够灵活的。第二,Bezier 曲线段不具备局部修改的可能性。因为,$B_{i,n}(t)$ 在参数的整个开区间(0, 1)内均不为零,所以曲线段在开区间内任何一点的值均要受到全部顶点的影响,改变其中某一顶点的位置对整个曲线均有影响。

图 3-25

为了克服上述缺点,在 1972 年至 1974 年期间,人们用 B 样条基替换了伯恩基多项式,构造出等距节点的 B 样条曲线。B 样条曲线除保持了 Bezier 曲线的直观性和凸包性等优点之外,还可以进行局部修改,且曲线更逼近特征多边形。此外,曲线的阶次也与顶点数无关,因而更为方便灵活。由于以上原因,B 样条曲线和曲面得到越来越广泛的应用。

1. B 样条曲线的数学表达式

(1) 一般形式

若给定 N = m + n + 1 个顶点(m 为最大段号,n 为阶次),则第 i 段(i = 0, 1, ⋯, m)、n 次等距分割的 B 样条曲线函数可表示为:

$$Q_{i,n}(t) = \sum_{l=0}^{n} P_{i+l} F_{l,n}(t), \quad l = 0, 1, \cdots, n \qquad (3\text{-}53)$$

其中,基底函数

$$F_{l,n}(t) = \frac{1}{n!} \sum_{j=0}^{n-l} (-1)^j C_{n+1}^{j} (t + n - l - j)^n$$

而

$$C_n^j = \frac{n!}{j!(n-j)!}$$

P_{i+l} 为定义第 i 段曲线特征多边形的 n + 1 个顶点。

(2) 三次 B 样条曲线

由于 n = 3,所以 l = 0, 1, 2, 3,此时所对应的基底函数分别为:

$$F_{0,3}(t) = \frac{1}{3!} \sum_{j=0}^{3} (-1)^j C_4^j (t + 3 - j)^3$$

$$= \frac{1}{6}(-t^3 + 3t^2 - 3t + 1)$$

$$F_{1,3}(t)=\frac{1}{3!}\sum_{j=0}^{2}(-1)^j C_4^j(t+2-j)^3$$

$$=\frac{1}{6}(3t^3-6t^2+4)$$

$$F_{2,3}(t)=\frac{1}{3!}\sum_{j=0}^{1}(-1)^j C_4^j(t+1-j)^3$$

$$=\frac{1}{6}(-3t^3+3t^2+3t+1)$$

$$F_{3,3}(t)=\frac{1}{3!}\sum_{j=0}^{0}(-1)^j C_4^j(t-j)^3$$

$$=\frac{1}{6}t^3$$

则第 i 段、三次 B 样条曲线的矩阵形式可表示为:

$$Q_{i,3}(t)=\sum_{l=0}^{3}F_{l,3}(t)P_{i+l}$$

$$=\frac{1}{6}[t^3\ t^2\ t\ 1]\begin{bmatrix}-1&3&-3&1\\3&-6&3&0\\-3&0&3&0\\1&4&1&0\end{bmatrix}\begin{bmatrix}P_i\\P_{i+1}\\P_{i+2}\\P_{i+3}\end{bmatrix}\qquad(3\text{-}54)$$

若令:

$$T=\begin{bmatrix}t^3&t^2&t&1\end{bmatrix}$$

$$M_s=\frac{1}{6}\begin{bmatrix}-1&3&-3&1\\3&-6&3&0\\-3&0&3&0\\1&4&1&0\end{bmatrix}$$

$$G_{si}=\begin{bmatrix}P_i&P_{i+1}&P_{i+2}&P_{i+3}\end{bmatrix}^T$$

式(3-54)可记为:

$$Q_{i,3}(t)=T\ M_s\ G_{si}$$

2. B 样条曲线的性质

(1) 端点性质及连续性

由(3-54)式对 t 一次求导, 可得:

$$Q_{i,3}'(t)=\frac{1}{2}[t^2\quad t\quad 1]\begin{bmatrix}-1&3&-3&1\\2&-4&2&0\\-1&0&1&0\end{bmatrix}\begin{bmatrix}P_i\\P_{i+1}\\P_{i+2}\\P_{i+3}\end{bmatrix}\qquad(3\text{-}55)$$

对(3-55)式再次求导, 可得

$$Q_{i,3}''(t) = [t\ 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 1 & -2 & 1 & 0 \end{bmatrix}\begin{bmatrix} P_i \\ P_{i+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix} \tag{3-56}$$

由式(3-54)、式(3-55)、式(3-56)可得端点性质如下:

$$\begin{aligned} Q_{i,3}(0) &= \frac{1}{6}(P_i + 4P_{i+1} + P_{i+2}) = \frac{1}{3}(P_{i+1}^{*} + 2P_{i+1}) \\ Q_{i,3}(1) &= \frac{1}{6}(P_{i+1} + 4P_{i+2} + P_{i+3}) = \frac{1}{3}(P_{i+2}^{*} + 2P_{i+2}) \end{aligned} \tag{3-57}$$

$$\begin{aligned} Q_{i,3}'(0) &= \frac{1}{2}(P_{i+2} - P_i) \\ Q_{i,3}'(1) &= \frac{1}{2}(P_{i+3} - P_{i+1}) \end{aligned} \tag{3-58}$$

$$\begin{aligned} Q_{i,3}''(0) &= (P_i - 2P_{i+1} + P_{i+2}) = 2(P_{i+1}^{*} - P_{i+1}) \\ Q_{i,3}''(1) &= (P_{i+1} - 2P_{i+2} + P_{i+3}) = 2(P_{i+2}^{*} - P_{i+2}) \end{aligned} \tag{3-59}$$

其中

$$P_{i+1}^{*} = \frac{P_i + P_{i+2}}{2}$$

$$P_{i+2}^{*} = \frac{P_{i+1} + P_{i+3}}{2}$$

为了考查B样条曲线段在连接处的连续性,不仅需要计算出第i段曲线终点处的$Q_{i,3}(1)$, $Q_{i,3}'(1)$及$Q_{i,3}''(1)$,而且还要求出第i+1段曲线始点处的$Q_{i+1,3}(0)$, $Q_{i+1,3}'(0)$及$Q_{i+1,3}''(0)$。由式(3-57)、式(3-58)及式(3-59)可得:

$$\begin{aligned} Q_{i+1,3}(0) &= \frac{1}{6}(P_{i+1} + 4P_{i+2} + P_{i+3}) \\ Q_{i+1,3}'(0) &= \frac{1}{2}(P_{i+3} - P_{i+1}) \\ Q_{i+1,3}''(0) &= (P_{i+1} - 2P_{i+2} + P_{i+3}) \end{aligned} \tag{3-60}$$

比较(3-57)～(3-60)式可知$Q_{i,3}(1) = Q_{i+1,3}(0)$, $Q_{i,3}'(1) = Q_{i+1,3}'(0)$及$Q_{i,3}''(1) = Q_{i+1,3}''(0)$。

以上结果告诉我们,三次B样条曲线在连接处一阶导数、二阶导数都是连续的,也即,三次B样条曲线具有二阶导数的连续性。将此结论加以推广,即可得出n次B样条曲线具有n-1阶导数连续的论断。由于实际工作中用得最多的是三次B样条曲线,很少应用高于三次的B样条曲线,因此,有关高次B样条曲线的问题,我们就不讨论了。式(3-57)～式(3-60)的几何图象如图3-26所示。

(2) 局部性

由(3-54)式可以看出,每一段三次B样条曲线由

图3-26 三次B样条曲线端点特性

四个控制点的位置向量来决定。同时也可看出,在三次 B 样条曲线中,改变一个控制点的位置,最多影响四个曲线段。因而,通过改变控制点的位置就可对 B 样条曲线进行局部修改,这是一个非常重要的性质。

(3) 扩展性

从式(3-54)可以看出,如果我们增加一个控制点,就相应地增加了一段 B 样条曲线。此时,原有的 B 样条曲线不受影响,而且新增的曲线段与原曲线的连接处具有一阶、二阶导数连续的特性,这一点是由 B 样条曲线本身的性质所保证的,不需要附加任何条件。因而要对原有的 B 样条曲线加以扩展是很方便的。图(3-26)表示出这一性质。

3. 三次 B 样条曲线的边界条件和反算拟合

图 3-27　三次 B 样条曲线边界条件

假定 B 样条曲线的控制顶点序列 P_i($i=0,1,\cdots,n$)已经给定,相应的三次 B 样条曲线如图(3-27)所示。如果要使曲线以 P_0 为起始点且切于向量 P_0P_1,同时以 P_n 为终点且切于向量 $P_{n-1}P_n$,那么只需在始端和终端各增加一个顶点 P_{-1} 及 P_{n+1},使得向量 $P_{-1}P_0=P_0P_1$,$P_{n-1}P_n=P_nP_{n+1}$,这样在始端和终端所增加的 B 样条曲线段即可满足上述要求。这一结论可证明如下:

先证明始端特性。

$$\because \quad Q_{i,3}(0)=\frac{1}{6}(P_i+4P_{i+1}+P_{i+2})$$

对于所增加的第 $i=-1$ 段曲线段的起始点为:

$$Q_{-1,3}(0)=\frac{1}{6}(P_{-1}+4P_0+P_1)=P_0$$

又 $\because$

$$Q'_{i,3}(0)=\frac{1}{2}(P_{i+2}-P_i)$$

$$\therefore \quad Q'_{-1,3}(0)=\frac{1}{2}(P_1-P_{-1})=P_1-P_0$$

仿此,可证明终端特性也满足要求。

所谓反算拟合,就是给定一系列的型值点 B_i($i=0,1,\cdots,n-3$),反求三次 B 样条曲线的控制顶点 P_i($i=0,1,\cdots,n-1$),从而决定一条三次 B 样条曲线,使其通过这一系列的型值点($n-2$ 个)。这个问题在实际工程中是很有用的。因为用三次 B 样条曲线来表示所需要的曲线,不仅可以显示输出,而且可以通过改变控制点实现交互修改。

由式(3-57)知

$$Q_{i,3}(0)=\frac{1}{6}(P_i+4P_{i+1}+P_{i+2}),\quad i=0,1,\cdots,n-3$$

在这组联立方程中,左端为三次 B 样条各曲线段的端点,即 B_i,共有 $n-2$ 个,是已知的。而右端为要求的相应控制点,却有 n 个。为此,必须补充两个边界条件,方可由此方程组解出相应的控制点序列 P_i。

第四节　字　　符

字符在计算机图形处理技术中是必不可少的内容,任何图形的标注、说明,都离不开字符。目前常用的字符有两种:一种是ASCII码字符,另一种是汉字字符。在工程绘图中,还有很多面向各种应用的工程符号,如电子线路中的电阻、电容符号;土建工程施工图中的标高、剖切位置符号;机械设计图纸中的加工精度、光洁度符号等,都可以归类于字符。1981年5月,我国国家标准化局发布并实施的"信息、交换用汉字编码字符基本集"中,共收集字符7445个,其中国标一级汉字3755个,国标二级汉字3008个,其余符号682个。每个字符均与1个十进制编码相对应。常用的字符编码有国标(区位)码和电传(电报)码两种。这些字符和符号在计算机内可用点位图(点阵)方式或笔划(矢量)方式来表示。它们的输入输出可以用基于字符发生器原理的硬件实现,也可以用基于编码技术的软件来实现。

一、矢量字符的存储与显示

1. 定义字符

用户定义矢量字符一般需要以下几步:

(1) 写字模

选一定大小的正方形网格,作为字符的界框,网格大小可选16×16,32×32,64×64,128×128等。对每一字符,在选定的网格中,采集其每一笔划两端点(结点)相对网格左下角(0,0)点的(x,y)坐标值。

(2) 确定字符代码

给出定义字符的编号,该编号不能重复,也不能越界。

(3) 按一定结构保存字符结点坐标及由前一点到此结点是否划线的标志。

(4) 保存结束标志。

(5) 将字模存入字符库中。

2. 存储字符

为了节省存储空间,查找迅速,通常可以采取以下几条措施来保存字符信息:

(1) 压缩存储

由于字模中,每一结点都有x、y坐标及划线与否的标志三个信息,如果分别存放,势必占有很大空间。为此,采用压缩存储的方法。例如,对于128×128的网格,可以用2个字节的整数来保存一个结点的三个信息。此时,结点的x、y坐标各占7位,划线与否标志占1位。

(2) 用二进制记录格式存储字模。

(3) 用定长记录存储字模。

这两条都是为了查找迅速而采取的方法。图3-28表示的是一个字符的存储结构。这里,代码号即为记录号。每一字符最多占2个记录,每一记录占48个整型单元。若记录的第一个单元为0,说明该字符只占1个记录;否则,由续记录号指示第二个记录的地址。续

记录可以单独用一个空记录,也可以用那些结点个数较少的字模中的剩余空间。最后,还应保存字符的结束标志(如-1)。

图 3-28　矢量字符的存储结构

3．显示字符

在应用中,用户通常希望在一定条件下,显示一串字符。比如,给定要显示字符串的定位点坐标,通常是字串的左下角点坐标(x_0, y_0);要显示字符的个数 N;字串路径与水平轴正向的夹角 ANG;每个字符代码 CODE,以及字符的宽度 W,高度 H,两字符的间距 S 等。显示的过程就是根据给定条件,把每一字符中结点信息取出,计算其在用户坐标系中的实际坐标。

(1) 求出字串中每个字符界框的左下角点坐标(x, y);

(2) 根据字符代码 CODE,按存储的逆过程,取出该字符中每个结点的坐标(Nx, Ny),显然此(Nx, Ny)是相对字符界框左下角(0, 0)点的坐标值。与此同时,把划线与否标志 NP 也取出。

(3) 进行一系列坐标变换,计算出代码 CODE 字符中每个结点在实际用户坐标系中的坐标:

① 作比例变换,计算出每个结点相对字符界框左下角点坐标(x, y),但是在宽度为 W,高度为 H 的矩形范围内的坐标值(x_s, y_s);

② 作旋转变换,计算出每个结点相对字符界框左下角点坐标(x, y),但与水平轴正向成 ANG 夹角方向的坐标值(x_R, y_R);

③ 作平移变换,计算出每个结点相对于用户坐标系(0, 0)的坐标值(x_u, y_u);

④ 根据 NP 标志,决定由前一点如何到当前点(x_u, y_u),是定位空走,还是划线。

二、点阵字符的存储与显示

在光栅扫描显示系统中,字符是利用掩膜来定义,并将其写入帧缓存保存和显示的。所谓字符掩膜,就是包含表示该字符的象素图案的一小块光栅。掩膜本身仅含一些二进制值,以指出掩膜中的象素是否用于表示字符信息。在简单的黑白显示器中,当象素用于表示字符时,对应二进制值通常取为 1,否则为 0。在彩色显示器中,则用更多象素位数以表示色彩的浓淡和作为查色表的指针。

当指定了某字符掩膜的原点在帧缓存中的坐标(x_0, y_0)以后,就可将此字符掩膜中每个象素相对(x_0, y_0)平移后的值写入帧缓存。读取帧缓存中这些象素值,就可以在屏幕上显示此字符。

如果将保存在帧缓存中某字符掩膜相应的象素值均置成背景色或背景光强,就可以擦除帧缓存中的该字符。

当字符写入帧缓存后,还可对字符掩膜进行修改,以获得不同字体或方向。图 3-29 是几种简单的修改例子。图 3-29(a)是原来的字符掩膜。将掩膜写入帧缓存中两个相邻单元 x_0 和 x_0+1,就可以得到粗体字(见图 3-29(b))。字符也可以旋转(见图 3-29(c))或倾斜,使字符成为斜体,(见图 3-29(d))。

图 3-29　变换后的字符掩膜

在 Microsoft C/C++ 7.0 图形模式环境下,如要在屏幕上输出基于字形的字符,一般要经过以下几步:

(1) 初始化字形库

调用_registerfonts 函数,将事先保存在盘上的.FON 文件中的字形头部信息装入内存。

(2) 装入单个字形

调用_setfont 函数,从现有字形集中寻找与指定特征匹配的单个字形,并将该字形设置为函数_outgtext 所有后继调用中使用的当前字形。选择字形由高到低的优先级为:

① 象素高度;

② 字模;

③ 象素宽度;

④ 固定间隔或比例间隔的字形。

(3) 设置文本颜色

调用_setcolor 函数,决定文本输出的当前色。

(4) 设置文本当前图形位置

由函数_moveto 决定文本输出的当前位置。

(5) 设置字形文本输出的方向

函数_setgtextvector 决定屏幕上字形文本的旋转方向,包括水平、逆转 90 度、旋转 180 度及逆时针旋转 270 度等。

(6) 在图形方式下打印基于字形的文本。

调用_outgtext 函数,即可在屏幕工输出基于字形的文本。文本是以当前字形、当前颜

色,在当前图形位置、以设置的输出方向在屏幕上输出。

第五节　区 域 填 充

光栅系统的突出优点之一是容易保存和显示用某种颜色或图案填充的区域。区域内的填充图案是以颜色或亮度值保存在帧缓存中的。在一个向量式图形系统中,显示这样的区域比较困难,因为它要求在每一个刷新周期内显示出区域内的无数条直线段。因此,目前大多数的区域填充都是在光栅系统中实现的。

一个区域是指一组相邻而又相连的象素,且具有同样的属性。根据边或顶点的简单描述,生成实区域的过程称为区域填充或多边形的扫描转换。

区域填充算法可分为两大类:一是种子填充算法;二是扫描转换填充算法。

种子填充算法首先假定封闭轮廓线内某点是已知的,然后算法开始搜索与种子点相邻且位于轮廓线内的点。如果相邻点不在轮廓线内,那么就到达轮廓线的边界;如果相邻点位于轮廓线之内,那么这一点就成为新的种子点,然后继续递归地搜索下去。种子填充算法只适用于光栅扫描设备。

扫描转换填充算法则是按扫描线的顺序确定某一点是否位于多边形或轮廓线范围之内。这种算法也适用于画线显示器,用以产生剖面线或阴影线。

区域的建立和定义通常可采用两种方式:一是内定义区域(interior-defined),用这种方式定义的区域内部所有象素具有同一种颜色或亮度值,而区域外的所有象素具有另一种颜色或亮度值。将该区域中的全部象素都设置为新值的算法,即填充内定义区域的算法,称为漫水法(flood-fill algorithm)。另一种是边界定义区域(boundary-defined),这种方式定义的区域,其边界上所有象素均具有特定的颜色或亮度值,而在区域内的象素则具有不是新值的某种颜色或亮度值。边界值可以为新值,但区域内不允许有新值。将该区域中的全部象素值都设置为新值的算法称为边界填充算法(boundary-fill algorithm)。

区域的连通情况又有四连通和八连通区域之分:

四连通区域　各象素在水平和垂直四个方向上是连通的。如图 3-30 (a)～(d)所示,其中 3-30(a)和(b)是四连通式内部定义的区域;而图 3-30(c)和(d)是四连通式边界定义的区域。

八连通区域　各象素在水平、垂直及四个对角线方向都是连通的。如图 3-31(a)～(d)所示,其中图 3-31(a)及(b)是八连通式内部定义的区域;而图 3-31(c)和(d)则是八连通式边界定义的区域。

我们看到,一个八连通式区域的边界是四连通式的;而一个四连通式区域的边界则为八连通式的,显然,一个八连通式区域的算法可以用在四连通式的区域上,但是由于它可以"跳过"象素之间的对角线连线,故有可能越界,因而会产生意想不到的结果。

一、种子填充算法

前面已说过,种子填充算法是假设在多边形或区域的内部,至少有一个象素是已知的,然后设法找到区域内所有其它象素,并对它们进行填充。

图 3-30　四连通式区域

图 3-31　八连通式的区域

1．简单的种子填充算法

(1) 漫水法

这是对内定义区域进行填充的算法,此算法所采用的基本方法是,首先在区域内测试一点(x,y)的象素值,看其是否具有原始给定的值,也即决定该点是否在区域内又未被填

充过。如果是,则改变其颜色或亮度值。然后再在其四个方向或八个方向上扩展,继续测试,通过递归调用,实现四连通式或八连通式的区域填充。

(2) 边界填充算法

边界填充算法与漫水法的基本思想是一样的。所不同的是,在测试(x, y)点的象素是否处在区域之内同时又未被访问过时,包括两部分的内容: ① 与边界值相比较,以检测此象素是否为该区域的一部分; ② 与新值相比较,以决定该象素是否已被访问过。这种测试的前题条件是: 在初始状态,区域内没有一个象素已设置为新值。但是,允许新值等于边界值。

我们可以用堆栈的方法,对边界定义的区域进行填充,基本流程是:

① 种子象素压入堆栈;

② 当堆栈非空时,

- 从堆栈中推出一个象素,并将该象素置成所要的值;
- 对于每个与当前象素邻接的四连通或八连通象素,进行上述两部分内容的测试;
- 若所测试的象素在区域内又未被填充过,则将该象素压入堆栈。

图 3-32　填充边界定义的区域

上述简单的种子填充算法过程是很简单的,但却是深度递归的。递归要花费时间,当内存空间有限时,还可能引起栈溢出。图 3-32 所表示的是由顶点(1, 1),(8, 1),(8, 4),(6, 6)及(1, 6)所决定的边界定义多边形区域,种子象素为(4, 3)。当用堆栈方法对其进行填充时,我们发现,有些象素多次被压入堆栈,而且堆栈变得很大。如按照右、上、左、下四连通域填充时,当算法进行到象素(5, 5,)时,堆栈的深度为 23 层,栈内包含很多重复的和不必要的信息。使堆栈极小化的一种算法是: 在任意不间断的扫描线区段中,只取一个种子象素,称为扫描线种子填充算法。

2. 扫描线种子填充算法

上述简单种子填充算法效率低的主要原因,是我们并未考虑象素间的相关性,而是孤立地对一个一个象素进行测试。扫描线种子填充算法的测试对象是一个个象素段。这里,象素段是指区域内相邻象素在水平方向的组合,它的两端以具有边界值的象素为边界,其

中间不包括具有新值的象素。对于区域内的每一象素段，我们可以只保留其最右(或左)端的象素作为种子象素。因此，区域中每一个未被填充的部分，至少有一个象素段是保持在栈里的。扫描线种子填充算法适用于边界定义的区域。区域可以是凸的，也可以是凹的，还可以包含一个或多个孔，见图 3-33 所示。

图 3-33　用扫描线种子填充算法填充边界定义的区域

此算法按下述步骤进行：

(1) 从包含种子象素的堆栈中推出区段的种子象素；

(2) 沿着扫描线，对种子象素的左、右象素进行填充，直至遇到边界象素为止；

(3) 区段内最左和最右的象素记为 x_l 和 x_r，在 $x_l \leq x \leq x_r$ 中，检查与当前扫描线相邻的上、下两条扫描线是否全为边界象素或已填充过；

(4) 如果这些扫描线既不包含边界象素，也不包含已填充的象素，则在 $x_l \leq x \leq x_r$ 中，把每一象素段的最右象素取作为种子象素，并压入堆栈；

(5) 初始化时，向堆栈压入一个种子象素，并在堆栈为空时结束。

用此算法填充图 3-32 所示的区域，堆栈深度最大为 2，从而解决了简单种子填充算法存在的问题。

二、扫描转换填充算法

扫描转换填充算法适用于规则边界的封闭区域,通常是将由顶点定义的多边形的边及其内部用预期的象素值予以填充,因此常称为多边形的扫描变换。与种子填充算法比较,这种算法不需要预先定义区域内部或边界的象素值,因而无需访问刷新缓存,即不必作读象素的操作,所以一般说来效率较高。

1. 简单测试法

有了对多边形轮廓的描述,要对此多边形内的区域进行填充,最本质的问题是要找出位于该多边形边界内的全部象素,即进行多边形对平面上点的包含性检查。然后,再将那些位于多边形内的全部象素的象素值置换成相应的数值。下面是两种常用的包含性检查的方法。

(1) 射线法

这种方法是由被测点向某方向(如 $y=-\infty$)作射线,计算此射线与多边形所有边的交点个数。若交点个数为奇数,则被测点在多边形内部,如图 3-34(a)所示。如果交点个数为偶数(包括 0),则该点在多边形的外部,如图 3-34(b)所示。

图 3-34　射线法交点计数示意图

显然,当射线恰好通过多边形的顶点时,有可能出现判断的错误。为此,我们可采取“左闭右开”(或“上闭下开”)的原则:即在射线左边的边与该射线相交时交点有效,应计数;而在射线右边的边与射线相交时交点无效,不计数。因此,在图 3-35 中,(a)、(b)、(c)、(d)中的$\overline{12}$边及(c)中的$\overline{23}$边与射的交点均有效;而(a)中的$\overline{34}$边,(b)中的$\overline{23}$边和(d)中的$\overline{23}$、$\overline{34}$、$\overline{45}$边虽与射线相交,但因它们均在射线的右边,所以交点无效。当射线与多边形的某条边重合时(如(a)中的$\overline{23}$边),我们规定不作求交运算。这样,在图 3-35 中,被测点 P 可以被判断为在多边形之内,因为过 P 点所作的射线与多边形各边相交的交点个数均为奇数。

(2) 弧长法

这种方法要求多边形由有向边组成,即规定沿多边形各边的走向其左侧(或右侧)为多边形的内部。方法是以被测点为圆心作单位圆,将全部有向边向单位圆作径向投影,并计算其在单位圆上弧长的代数和。若代数和为 0,则被测点在多边形之外,如图 3-36(a)所示;若代数和为 2π 则被测点在多边形之内,如图 3-36(b)所示。对于内部有空洞的多边形,只要按照上述规定来定义多边形的有向边,则可以采用同样的测试方法。

图 3-35　用射线法测试点的包含性

图 3-36　用弧长法测试点的包含性

如果用函数 Inside(polygon, x, y) 来说明被测点(x, y)的位置, 那么, 当我们用上述任一种方法测试出(x, y)在多边形 polygon 之内时, 此函数就回送‘真’值, 这样, 当对平面上所有点都进行这样的测试时, 即可对多边形内部的点进行填充。

简单测试法虽然简单, 但是却慢得不可容忍。因为屏幕上有数以百万计的点要进行测试, 每测试一个点, 都要用射线法去计算交点, 或用弧长法去计算径向投影弧长的代数和, 工作量实在太大。

2. 扫描线算法

(1) (YX)算法

这种算法不是孤立地去测试平面上的点是否在一个多边形之内, 而是利用一条扫描线上的象素存在着相关性这一特征, 将很多相邻的象素放在一起测试, 从而大大减少了测试点的数目。因为在一条扫描线上, 如果某个象素在多边形之内, 那么, 一般地说, 紧挨着的各象素也多半在多边形内。对于边界外的象素也同样存在着这种相关性。只有当多边形的边恰好通过两个象素之间时, 它们在多边形的内、外性质才有所不同, 而这种情况一般说来是有限的。在图 3-37 中, 对于 y= 8 这条扫描线, 可求出交点 a、b、c、d, 因此, 凡是 a< x< b和 c< x< d 之内的象素, 都位于多边形之内。根据这一思想, (yx) 算法的步骤可叙述如下。

① 求交　　求出每一条扫描线与多边形所有各边的交点, 建立起(x, y) 交点表, 若为水平边则跳过。

② 排序　　按 x 及 y 增加(或减少)的顺序,对这些交点进行排序。例如,交点 (x_1, y_1)和(x_2, y_2),当 $y_1 < y_2$ 或 $y_1 = y_2$ 而 $x_1 < x_2$ 时,则(x_1, y_1)先于(x_2, y_2)。

③ 填充　　将每一对交点之间的所有象素置成所需要的象素值。

图 3-37　多边形和扫描线

这个算法的核心是按(x, y)值对交点作一次分类,以便找出象素的相关群。分类的顺序并不是特别重要的,但是却影响位于多边形内的象素出现的顺序,也即影响显示画面出现的顺序。此算法是对每一对交点之间的所有象素进行填充,因此交点的个数必须是偶数才能保证填充的正确性。而当扫描线恰好通过多边形的顶点(称为奇异点)时,不适当地计数就会造成错误。如图 3-37 中 y = 7 这条扫描线,若顶点④(也即交点(7, 7))只计一次,就会出现这种情况。对于顶点①、③、⑤也有类似的情况。对这些奇异点,我们最好多计入一个交点。但对于顶点②、⑥,重复计数又会出现错误。为此,我们提出一种可靠的处理奇异点的方法,它依赖于多边形的拓扑性质。具体地说,它利用多边形相继各边的方向,若多边形的边单调地向上或向下伸展时,则对于每条与两个边的端点相交的扫描线,只需记录一个交点。而当边的方向是由下变成向上或相反时,则必须重复记录一个交点。

还有一种简单的处理奇异点的方法,就是对于那些不需要重复计数的奇异点,可以将其中一条边的 y 坐标值抬高一个象素,以保证仅有一条边与扫描线有交点,确保填充的正确性。在设备分辨率足够高的情况下,这种处理方法可以满足一般的要求,对于图 3-37 中的边 e_2 及 e_5,如使其 y 的最小值分别为 y = 4 和 y = 6,则保证了顶点②及⑥只计一次交点。

(2) Y-X 算法

上边介绍的(YX)算法的第一步,也就是计算交点这一步,工作量是很大的,因为需要对多边形的每一条边都进行测试,看其是否与每一条扫描线相交。但是对于某一条扫描线,我们感兴趣的常常只有少数几条边。而且,与第 i 条扫描线相交的许多条边,一般地说也与第 i+ 1 条扫描线相交。也即在与一条边相交的很多条扫描线范围内,存在着边的相关性。因此,对于每一条扫描线,我们只比较与其相交的那些边。算法执行时要求建立一张有效边表(AET),AET 中的内容随每一条扫描线 y 值的不同而变化。也即,在 AET 中,只保留与当前扫描线相交的所有的边,而且这些边按其与该扫描线交点的 x 坐标的顺序

存放,以便于在一对交点之间填充。当从一条扫描线移到下一条扫描线时,我们利用两条扫描线间隔为 1 的特性,可以简化边与扫描线交点的计算。图 3-38 中,设边$\overline{AB}$的斜率为 m,若其与扫描线 y_i 的交点横坐标为 x_i,则与扫描线 y_{i+1} 的交点的横坐标为:

$$x_{i+1} = x_i + 1/m$$

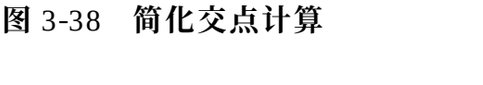

图 3-38　简化交点计算

为了便于建立和修改 AET,首先需要建立一张边表(ET)。边表一般是由一系列的存储桶(bucket)构成的,桶的数目与扫描线数一样多。凡是与每条边两端点中较小的 y 坐标值(y_{min})对应的桶,都记录着待填充多边形各边的有关信息,包括每条边两端点中最大的 y 坐标值(y_{max}),y 值较小的那个端点的 x 坐标值(x_{min})以及该边斜率的倒数(1/m)。图3-39表示出图 3-37 的 6 条边是如何存放的,其中 e_2 及 e_5 边已缩短了一条扫描线的距离,以避免出现双重交点。

图 3-39　边表

当形成了 ET 以后,Y-X 算法按如下步骤执行。

(1) 将 y 值设置为 ET 中所列的最小 y 值,也即第一个非空的存储桶。

(2) 将 AET 初始化,使其为空。

(3) 重复作以下各步,直至 AET 和 ET 都为空:

① 把 ET 中 y 的信息与 AET 合并,同时保存 AET 中按 x 值实现的排序序列;

② 对于扫描线 y,在一对交点之间填充所需要的象素值;

③ 从 AET 中删去 $y > y_{max}$ 的项;

④ 对于仍留在 AET 中的每一项,用 x + 1/m 代替 x;

⑤ 检查并保证 AET 中各项按 x 值的排序;

⑥ 使 y 增 1,成为下一条扫描线的坐标。

图 3-40 表示出在图 3-37 中所示的多边形当扫描线 y= 9 和 y= 10 时 AET 的内容。

图 3-40 有效边表

(a) 扫描线 9; (b) 扫描线 10

由于此算法同时考虑到了扫描线和边的相关性。因此是一个效率较高的填充算法。

三、区域填充属性

区域填充属性包括填充式样、填充颜色和填充图案的类型等,用户可以对这些属性进行选择。

1. 填充式样

这是描述区域内部的类型的,诸如是空心的,实心的,还是有花样图案的。

例如在 Microsoft C/C++ 7.0 环境中,定义区域的函数_ellipse,_polygon,_rectangle 等均有"control"这一参数,当其选为"_GBORDER"时,则表示该区域为空心的,只画边界,不填充区域。当选为"_GFILLINTERIOR"时,则要填充区域。此时,还应视_setfillmask 函数中,定义图案(即掩码)的数组 mask 是否为"NULL",若为"NULL",则以当前设置的颜色填充区域,即区域为实心的;否则,则以所定义的图案填充该区域。

2. 填充颜色

对于一个空心的区域,可以选择区域外框的颜色;而对一个实心的区域,则可以选择内部的颜色。

3. 填充图案

(1) 定义图案

用户可以预先定义各种图案,或称模板,掩码(mask)。

在 Micorsoft C/C++ 7.0 环境下,定义图案的函数为_setfillmask,其中,参数 mask,即掩码数组,它是 8×8 位的数组,每一位代表一个象素。值为 1 的位,将相应象素设置成当前颜色;值为 0 的位,则不改变原象素值。

(2) 填充区域

在预定义了某种图案后,即可对指定的区域进行填充。

例如,调用_floodfill 函数,即可实现对区域的填充,其中,参数 x,y 为起始点坐标,当

起点在区域内部时,就填充内部;如果在区域外部,则填充背景部分。

习　　题

1. 在图形设备上如何输出一个点?为输出一条任意斜率的直线,一般受到哪些因素影响?
2. 为什么说直线生成算法是二维图形生成技术的基础?
3. DDA 法生成直线的基本原理是什么?请画出用硬件实现对称 DDA 法的原理图。
4. 对于 Bresenham 直线生成算法,如何利用对称性通过判别误差变量同时从直线两端向直线中心画直线?又如何消除可能产生的误差?
5. 试对常用的 3 种直线生成算法的复杂性进行比较。
6. 实现直线的线宽为什么要考虑直线的斜率?不同斜率的粗线在连接处会出现怎样的情况?如何进行特殊处理?
7. 试讨论一下如何提高角度 DDA 法圆弧生成算法的效率。
8. 为什么说 Bresenham 画圆的算法效率较高?
9. 给定三个型值点或三个控制点生成的抛物线是同一条吗?为什么?
10. 用参数方程描述自由曲线具有什么优点?为什么通常都用三次参数方程来表示自由曲线?
11. 请证明用抛物线参数样条曲线拟合的自由曲线,在 $P_2 \sim P_{N-1}$ 各已知点的左、右侧能达到一阶导数连续。
12. 请给出 Hermite 形式曲线的曲线段 i 与曲线段 i- 1 及曲线段 i+ 1 实现 $C^{(1)}$ 连续的条件。
13. 什么是样条曲线?三次参数样条曲线在公共端点处满足 $C^{(2)}$ 连续的条件是什么?
14. 简述三次参数样条曲线常用的 3 种边界条件及其含义。
15. 当用三次参数样条曲线拟合型值点间的间隔极不均匀的自由曲线时,如何选择参数值来改善曲线的形状。
16. Bezier 形式曲线具有哪些特性?试用 n 的归纳法证明其凸包性。
17. 在由 B 样条曲线上的 n 个型值点坐标反算其特征多边形的控制点列时,若曲线为封闭的,请写出其边界条件及所求关系式。
18. 定义矢量字符一般需要哪几步?为节省存储空间,通常多采用哪些措施来存储字符?
19. 显示一串字符都要进行哪些坐标计算?
20. 请用图说明,一个四连通区域的边界是八连通的,而一个八连通区域的边界则是四连通的。
21. 简述扫描线种子填充算法的算法思想及流程。已知一区域边界顶点如下所述:(1, 1),(4, 1),(4, 3),(9, 3),(9, 1),(11, 1),(11, 6),(8, 9),(1, 9)及(1, 1)。内部孔边界顶点为:(4, 4),(9, 4),(9, 5),(7, 7),(4, 7)及(4, 4)。种子 S 为(5, 7)。请用扫描线种子填充算法对其进行填充,写出每一条扫描线对应堆栈中的内容。
22. 请列出用 Y-X 扫描线算法实现多边形区域填充所需的程序模块及主要功能。
23. 已知多边形各顶点坐标为:(2, 2),(2, 4),(8, 6),(12, 2),(8, 1),(6, 2)及(2, 2)。在用 y-x 算法对其实现扫描转换时,请写出 ET 及全部 AET 的内容。
24. 请对(YX)及 Y-X 填充算法作比较。指出各自的优、缺点。
25. 区域填充都有哪些属性?
26. 图 3-41(a)所示的是黑、白相间的图案,图 3-41(b)为一封闭的三角形区域。请实现用该图案填充三角形内部的区域。

图 3-41

第四章　图形的裁剪及几何变换

图形的裁剪及几何变换是计算机图形学领域内较为基础的内容之一。通过裁剪，可以截取落在指定区域内的图形；通过变换，可以从简单图形得到复杂图形，可以从某一个图形得到多个其它图形，从而可以减少用户对图形参数的输入量。

第一节　窗口视图变换

一、窗口区和视图区

用户用来定义设计对象的实数域统称为用户域，也称为用户空间。从理论上说，用户域是连续的、无限的。用户可以在用户域中指定任意的区域 w，把他感兴趣的这部分区域内的图形输出到屏幕上，通常称这个区域为窗口区。窗口区一般是矩形区域，可以用其左下角点和右上角点坐标来表示。显然，窗口是用户图形的一部分，而且可以采取措施，嵌套定义窗口。也就是在第一层窗口中再定义第二层窗口，在第 i 层窗口中定义第 i+ 1 层窗口等等，嵌套的层次由图形处理软件规定。

图形设备上用来输出图形的最大区域称之为屏幕域，它是有限的整数域，大小随具体设备而异。任何小于或等于屏幕域的区域都可定义为视图区。视图区由用户在屏幕域中用设备坐标定义，一般也定义成矩形，大多由其左下角点坐标和右上角点坐标来定义。当然，视图区也可以嵌套，嵌套的层次由图形软件决定。

二、窗口区和视图区之间的坐标变换

为了全部、如实地在视图区中显示出窗口区内所定义的物体图形，就必须求出图形在窗口区和视图区之间的映象关系，也就是说，需要根据用户所定义的参数，找到窗口区和视图之间的坐标对应关系。如图 4-1 所示，设矩形窗口的左下点的 x，y 坐标为 w(1)，w(2)，右上角点坐标为 w(3)，w(4)。矩形视图区的两个角点坐标分别为 iv(1)，iv(2) 和 iv(3)，iv(4)。因此，窗口区中的一点(x，y) 与视图区中的一点(ix，iy) 存在如下的对应关系：

$$\frac{ix - iv(1)}{x - w(1)} = \frac{iv(3) - iv(1)}{w(3) - w(1)}$$
$$\frac{iy - iv(2)}{y - w(2)} = \frac{iv(4) - iv(2)}{w(4) - w(2)} \qquad (4\text{-}1)$$

若

$$a_v = \frac{iv(3) - iv(1)}{w(3) - w(1)}$$
$$b_v = iv(1) - a_v ¡¤w(1)$$

$$c_v = \frac{iv(4) - iv(2)}{w(4) - w(2)}$$

$$d_v = iv(2) - c_v \times w(2)$$

则(4-1)式可表示为:

$$\begin{aligned} ix &= a_v x + b_v \\ iy &= c_v y + d_v \end{aligned} \tag{4-2}$$

其中, a_v, c_v 分别反映了视图区和窗口区之间在 x 和 y 方向上的伸缩比, b_v、d_v 则分别反映了定位点在 x 和 y 方向上的偏移量。显然, 如果 $a_v = c_v = 1$, $b_v = d_v = 0$, 则窗口区和视图区的大小相等, 原点相同。若 $a_v \neq c_v$, 则图形会发生畸变。例如, 在窗口区中的一个圆, 在视图区中则显示为一椭圆。因此, 应当采取措施加以避免。

图 4-1　窗口区与视图区的对应关系

如果已知视图区内的一点坐标(ix, iy), 则窗口区内的对应点坐标(x, y)可由下式求出:

$$\begin{aligned} x &= a_w \times ix + b_w \\ y &= c_w \times iy + d_w \end{aligned} \tag{4-3}$$

其中,

$$a_w = \frac{w(3) - w(1)}{iv(3) - iv(1)}$$

$$b_w = w(1) - a_w \times iv(1)$$

$$c_w = \frac{w(4) - w(2)}{iv(4) - iv(2)}$$

$$d_w = w(2) - c_w \times iv(2)$$

如果我们所定义的视图区是规格化的, 也即 iv(1) = iv(2) = 0.0, iv(3) = iv(4) = 1.0, 则有:

$$a_v = \frac{1.0}{w(3) - w(1)}$$

$$b_v = -a_v \times w(1),$$

$$c_v = \frac{1.0}{w(4) - w(2)}$$

$$d_v = -\ c_v \ ¡¤w(2)$$

以及

$$a_w = w(3) - w(1)$$
$$b_w = w(1)$$
$$c_w = w(4) - w(2)$$
$$d_w = w(2)$$

因此,只要用户指定窗口区和视图区的范围,就可利用(4-2)式和(4-3)式实现窗口和视图之间的坐标变换。至于从规格化视图区到实际物理空间视图区的变换,也不难实现。

第二节　二维图形的裁剪

当用户在平面上定义了一个窗口以后,总是希望把落在窗口内的部分图形映象到视图区中,而把在窗口以外的图形运用裁剪方法统统裁掉,不予输出。这种平面上的图形受该平面上的矩形窗口的裁剪称为二维裁剪。

一、裁剪的策略及原理

平面图形在裁剪时,由于组成图形中的每一个基本元素都要经过裁剪,因此裁剪算法直接影响整个图形系统的效率。裁剪的方法很多,其效率的高低常和图形的复杂情况、计算机功能等因素有关,所以要根据实际情况来选择裁剪方法。一般可选用的裁剪策略有:

(1) 图形裁剪与窗口——视图变换的先后。

多数图形裁剪都是在用户坐标系下相对窗口边框进行裁剪,也即是"先裁剪后变换"。这种方法可以避免落在窗口外的图形再去进行无效的窗口——视图变换运算。当然也可以先把用户坐标转化为屏幕坐标,在屏坐标系下相对于视图区进行裁剪,即"先变换后裁剪",对于一些特定算法可以选用这种策略。

(2) 图形生成与裁剪的先后。

"先生成后裁剪",只需要简单的直线段的裁剪算法,但可能造成无效的生成运算。"先裁剪后生成"则可避免对那些最后被裁掉的元素进行无效的生成运算,但却要采用比较复杂的图形,如圆弧、椭圆弧等的裁剪处理。

另外,由于裁剪处理比较费时,对于一些特殊的应用,例如,只需观察图形的全貌,并不真正需要实施裁剪时,则可允许自动调整窗口的大小,使之跳过裁剪。

裁剪的基本目的是判断图形元素是否在所考虑的区域内。如在区域内,则进一步求出在区域内的那一部分。因此,裁剪处理包含两部分内容:

(1) 点在区域内外的判断;

(2) 计算图形元素与区域边界的交点。

当区域为正规的矩形时,判断点在区域的内外非常简单,只要做1～4次比较。即,若点(x,y)满足:

$$\begin{aligned} x_L \le x \le x_R \\ y_B \le y \le y_T \end{aligned} \qquad (4\text{-}4)$$

则该点在 x_L, x_R, y_B, y_T 所决定的矩形区域内。若式(4-4)的不等式中有一个不满足,那么此点就不可见,应予裁掉。当然,如果对于一幅整图中的所有点都用式(4-4)进行比较实现对图形的裁剪,显然是很不合适的。这样的裁剪时间太长,而且使得裁剪出来的点列不一定保持原来图形的画线序列,因而给图形输出造成困难。

交点的计算比较费时,特别是比较复杂的曲线的交点计算更是费时。通常,先作 BOX 重叠测试,即把那些曲线的 BOX 与区域的 BOX 根本不相重叠的求交计算排斥在外。

由此可见,裁剪的原理并不复杂,但由于裁剪处理常常涉及大量的图形元素,所以对裁剪处理的速度有十分高的要求。如果一条线段的裁剪需要 5 毫秒,那么十多条线段所需的裁剪时间就会使人们有所感觉,从而在复杂的图形显示中就会出现不能容忍的等待,而在动态图形显示中会出现扭曲变形的问题。因此,提高裁剪算法的速度是至关重要的。

下面分别介绍几种常用的基本图素裁剪算法。

二、二维线段的裁剪

1. 矢量裁剪法

设窗口的四条边界为 x_L, x_R, y_B, y_T,某条待裁矢量线段为 $\bar{a}$,其起点和终点的 x、y 坐标分别为(A, B)和(C, D),如图 4-2 所示。线段裁剪算法的任务就是要找出该线段落在窗口区内或窗口边界上的起始点和终止点的坐标。矢量裁剪算法思想是:先从(A, B)为始点进行判断或进行求交运算,所得交点坐标(x, y)保存在(x_s, y_s)中,然后再把矢量倒过来,即以(C, D)为始点,再用前面的判断及求交运算程序求得交点坐标(x, y),最后只输出从(x_s, y_s)到(x, y)之间的线段。下面仅以(A, B)作始点为例,简叙矢量裁剪算法的步骤。

图 4-2　线段$\bar{a}$与窗口

(1) 当 $x_L \le A \le x_R$ 时

① 若 $y_B \le B \le y_T$,则 $x = A, y = B$;

② 若 $B < y_B$,同时 $D < y_B$,则与窗口无交。否则用求交公式求得其交点坐标为:

$$x = A - (B - y_B)(A - C)/(B - D)$$

$$y = y_B$$

此时,应判断 x 值的有效性。仅当 $x_L \le x \le x_R$ 时,所求交点方属有效。

③ 若 $B > y_T$,则与②类似。当 $D > y_T$ 时,与窗口无交。否则,需求交,交点坐标为:

$$x = A - (B - y_T)(A - C)/(B - D)$$

$$y = y_T$$

同样,应判断 x 值的有效性。

(2) 当 $A < x_L$ 时

若 $C < x_L$,则与窗口无交,否则需求交,交点坐标为:

$$x = x_L$$

$$y = B - (A - x_L)(B - D)/(A - C)$$

此时应判断 y 值的有效性:

① 若 $y_B \leq y \leq y_T$, 则所求交点有效;

② 若 $y < y_B$ 或 $y < y_T$, 同时 $y_B \leq B \leq y_T$, 则所求交点无效(见图 4-3(a));

③ 若 $y < y_B$ 或 $B < y_B$, 则还可能存在真正的交点, 应返回到前面(1)中的②, 进一步判断求交(见图 4-3(b));

④ 若 $y > y_T$ 且 $B > y_T$, 则与③类似, 应返回到(1)中的③, 也需要作进一步的判断并求交(见图 4-3(c))。

(3) 当 $A > x_R$ 时, 可类似地求出 $\bar{a}$ 与窗口右边框的交点。

图 4-3 线段与窗口边界相交情况

这种方法的缺点是在求交过程中必须使用乘除法, 而且还要判断交点的有效性, 即要判断所求交点是否既落在窗口边界上又落在线段上。若利用直线的参数方程, 则可以提高求交的效率。

设某条直线的端点为(x_1, y_1)及(x_2, y_2), 参数 t_1 的范围是$[0, 1]$; 某窗口边界的端点为(x_3, y_3)及(x_4, y_4), 参数 t_2 的范围也是$[0, 1]$, 则有

$$x = x_1 + pt_1$$

$$y = y_1 + qt_1$$

及

$$x = x_3 + rt_2$$

$$y = y_3 + st_2$$

其中, $p = x_2 - x_1$, $q = y_2 - y_1$, $r = x_4 - x_3$, $s = y_4 - y_3$

如再令: $u = x_3 - x_1$, $v = y_3 - y_1$, 则有

$$t_1 = \frac{us - rv}{ps - rq}$$

$$t_2 = \frac{uq - pv}{ps - rq}$$

若用上式计算出的 t_1, t_2 均在[0, 1]内，则此交点确实就是窗口的有效交点。当然，对于平行于窗口边界的线段，应预先进行判断，特殊处理。

2．编码裁剪法

这种算法是用区域检查的办法有效地识别可以直接接受或直接舍弃的线段，只有不属于这两种情况的线段才需要计算交点。算法以图 4-4 所示的 9 个区域为基础，其中 x_L，x_R，y_B，y_T 为窗口的 4 条边界。任何一条线段的端点，根据其坐标所在的区域，都可以赋予 4 位二进制代码。设最左边的位是第 1 位，则其含义如下：

第 1 位为 1，表示端点在 y_T 上方；

第 2 位为 1，表示端点在 y_B 下方；

第 3 位为 1，表示端点在 x_R 右边；

第 4 位为 1，表示端点在 x_L 左边；

否则，相应位置 0。

显然，如果线段两个端点的 4 位编码全为 0，则此线段全部在窗口内，可直接接受，如果对线段两个端点的 4 位编码进行逻辑与(按位乘)运算，结果为非 0，则此线段全部在窗口之外，可直接舍弃；否则，这一线段既不能直接接受，也不能直接舍弃，它可能与窗口相交。此时，需要对线段进行再分割，即找到与窗口一个边框的交点。根据交点位置，也赋予 4 位代码，并对分割后的线段进行检查：或者接受，或者舍弃，或者再次进行分割。重复这一过程，直到全部线段均被舍弃或被接受为止。

图 4-4　窗口及线段端点的编码

图 4-5　编码裁剪法图解说明

与窗口边界求交次序的选择完全是任意的。但是无论哪种次序，对于有些线段的裁剪，可能不得不重复 4 次计算与 4 条边的交点，如图 4-5 中线段 AF 即是一例。

编码裁剪法的流程框图如图 4-6 所示。其中，直线段的两个端点 $P_1(x_1, y_1)$，和 $P_2(x_2, y_2)$，窗口的 4 条边界为 x_L, x_R, y_B, y_T。在线段再分割求交的过程中，保证 P_1 始终在窗口之

外,得到的交点(边界上的点)为新的 P_1,重复判断 P_1,P_2 的代码,舍弃在窗口外的,显示在窗口内的线段。

图 4-6　编码裁剪算法程序流程框图

3. 中点分割裁剪法

中点分割裁剪法又称对分法,其算法思想是:当一条线段既不能直接接受也不能直接舍弃,欲求其与区域的交点时,预先假设此交点落在线段的中点,如果这估计是错误的,则将直线分为两段,并对该两段再分别加以测试。用这种二分法搜索方式一直进行下去,直

到原来线段的一段被直接接受,而另一段被直接舍弃。中点分割裁剪算法判断线段是否与区域有交,仍可采用前面介绍的编码方法。只是在不得不求其与区域的交点时,不断用对分法求其中点,以增加循环为代价,避免费时的求交运算。设裁剪区域是正矩形,要裁剪的线段为 P_1P_2,如图 4-7 所示,为求其可见部分 AB,算法可分两个过程平行进行,即:

① 从 P_1 出发,找出离 P_1 最近的可见点 A;

② 从 P_2 出发,找出离 P_2 最近的可见点 B。

此两点的连线 AB,即为原线段 P_1P_2 的可见部分。

图 4-7　中点分割裁剪示意图

求 P_1 最近的可见点 P 的算法框图如图 4-8 所示。当然,求 P_2 的最近可见点 P 的算法也是一样的,只要把 P_1 和 P_2 交换就可以了。这里 ε可取为一个象素的宽度。

一个二分法的搜索至多在 $\log_2 M$ 步后结束,这里 M 是所搜索的表的长度。所以上面讲的二分的过程,也将经过 $\log_2 M$ 步后找到其中的一个交点。这里 M 被解释为直线的水平分量和垂直分量中较长的那个分量。对于分辨率为 $2^N \times 2^N$ 的显示器来说,$M = 2^N$,则对于一条可见端点的最长的线,至多经过 N 步的裁剪。中点分割裁剪算法由于计算过程只要做加法和除 2,所以很容易用硬件来实现。另外,如果可用并行方式实现上述对分过程,则又可使裁剪速度大大加快。

图 4-8　中点分隔裁剪算法框图

三、字符的裁剪

字符串可以按下述不同方法进行裁剪

1．矢量裁剪

这种方法把每个字符都看作是一些短直线(笔划)的组合,故每一笔划都必须个别地进行裁剪。显然,这会产生令人满意的结果(如图 4-9 所示),但这样做速度慢,又不能与某些硬件字符发生器(如点阵式字符发生器)相兼容。

图 4-9　按字符的笔划进行裁剪

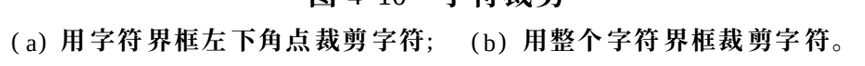

图 4-10　字符裁剪

(a) 用字符界框左下角点裁剪字符;　(b) 用整个字符界框裁剪字符。

2．字符裁剪

这种方法认为每个字符被一个称为字符框的矩形所包围,然后以这个框中的某一点(中心或某一个角)与窗口进行比较,如果这一点在窗口内,则显示此字符。我们也可以用整个字符框,或者为简单起见,用对角线与窗口进行比较,只有当字符框或对角线完全在窗口之内时,才显示该字符。图 4-10 说明了这两种方法的结果。

3．字符串裁剪

这种方法是把整个字符串作为整体来处理:或者全部显示,或者全部不显示。测试时可以选用测试字符串界框上的某一点、测试字符串界框的对角线以及测试整个字符串界框等不同方法。图 4-11 说明用整个字符串界框来决定是否显示字符串时所得到的结果。

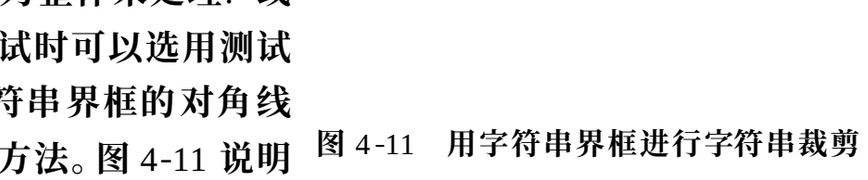

图 4-11　用字符串界框进行字符串裁剪

四、多边形的裁剪

多边形的裁剪是以线段裁剪为基础的,但又不同于线段的裁剪。在线段裁剪中,是把一条线段的两个端点孤立地加以考虑的。而多边形则是由若干条首尾相连的线段连接而成的,其中第 i 条线段的终点必定是第 i+ 1 条线段的起点,也就是说,多边形是由一些有序的线段组成的,裁剪后的多边形仍应保持原多边形各边的连接顺序。另外,多边形是封闭的图形,经裁剪后的图形仍应是封闭的,而不是一些孤立的线段。因此,对于多边形的裁剪应着重考虑以下问题:如何把多边形落在窗口边界上的交点正确地、按序连接成裁剪后的多边形,其中包括决定窗口边界及拐角点的取舍。图 4-12 示出了一些多边形裁剪的例子。下面介绍两种常用多边形裁剪算法。

图 4-12　多边形裁剪的例子

图 4-13　多边形逐边裁剪算法执行过程

(a) 原始多边形;　(b) 右边界裁剪后的多边形;　(c) 下边界裁剪后的多边形;
(d) 左边界裁剪后的多边形;　(e) 上边界裁剪后的多边开形。

1．逐边裁剪法

这个算法是由 I.E．萨瑟兰德(I.E.Sutherland)和 G.W．霍德曼(G.W.Hodgman)在 1974 年提出来的。具体做法是:每次用窗口的一条边界对要裁剪的多边形进行裁剪,把落在窗口外部区域的图形去掉,只保留窗口内部区域的图形,并把它作为下一次待裁剪的多边形。若连续用矩形窗口的 4 条边界对要裁剪的原始多边形进行裁剪,则最后得到的多边形即为裁剪后的结果多边形。图 4-13 说明了这个过程,其中,原始多边形为 ABCDEFGA,经窗口右边界裁剪后,得到 ABHIDJFGA,此多边形即为下一次待裁剪的多边形。经窗口的四条边界裁剪后的多边形为 NKLIDJQPGMN。显然,对于每一条窗口边框,都要计算其与多边形各条边的交点,然后把这些交点按照一定的规则连成线段。而与窗口边界不相交的多边形的其它部分则保留不变。因而,需要两个大数组用来保存原始多边形顶点坐标及某条窗边裁剪后所生成的新多边形的顶点坐标。逐边裁剪算法原理简单,易于用程序实现,但计算工作量比较大,而且需要一个比较大的存储区来存放裁剪过程中待裁剪的多边形。

2．双边裁剪法

此算法是在 1977 年由韦勒(Weiler)和阿瑟顿(Atherton)提出来的。设用户多边形为主多边形 P_s,窗口为裁剪多边形 P_c。同时,设每一多边形按顺时针方向排列,因此,沿多边形的一条边走动,其右边为多边形的内部。算法首先沿 P_s 任一点出发,跟踪检测 P_s 的每一条边,当 P_s 与 P_c 的有效边框相交时:

① 若 P_s 的边进入 P_c,则继续沿 P_s 的边往下处理,同时输出该线段;

② 若 P_s 的边是从 P_c 中出来,则从此点(称为前交点)开始,沿着窗口边框向右检测 P_c 的边,即用 P_c 的有效边框去裁剪 P_s 的边,找到 P_s 与 P_c 最靠近前交点的新交点,同时输出由前交点到此新交点之间窗边上的线段;

③ 返回到前交点,再沿着 P_s 处理各条边,直到处理完 P_s 的每一条边,回到起点为止。

图 4-14 说明了双边裁剪算法的执行过程。

双边裁剪算法的思路清楚,可适用于任何凸的或凹的多边形的裁剪。但是,这种算法由于需要反复求 P_s 的每一条边与 P_c 的 4 条边以及 P_c 的每一条有效边与 P_s 的全部边的交点,因而计算工作量很大。

多边形裁剪的算法很多,多年来,国内外学者都有很多研究,发表过不少文章,有兴趣的读者可以参考有关文献,在此不再多述。

第三节　二维图形几何变换的一般表示

一、二维图形几何变换的一般表示

我们知道,一幅二维图形可以由许多直线连接而成,或者被看作由许多小直线段逼近而成。而一条直线段则可以由始、末端点相连产生。所以,我们可以把一幅二维图形看成是一个点集。如果在 xy 平面内,一个点的坐标用行向量[x y]或列向量 $\begin{matrix}x\\y\end{matrix}$ 表示,那么一

图 4-14　多边形双边裁剪法执行过程图

个图形的点集则可以用 m× 2 或 2× m 的矩阵表示如下:

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_m & y_m \end{bmatrix}, \qquad \begin{bmatrix} x_1 & x_2 & \cdots & x_m \\ y_1 & y_2 & \cdots & y_m \end{bmatrix}$$

这样,对图形的几何变换就可以归结为对点的变换。当然,我们这里所研究的二维几何变换都是能将直线变成直线的变换,因此,无需要将直线上所有的点都进行变换,而仅对直线段的两个端点施加变换即可。

如果我们用 P = [x y]表示 XY 平面上一个未被变换的点,用 P′= [x′y′]表示经某种变换后的新点。那么,对于二维几何变换,一般可分下面 3 种情况来讨论。

对于平移变换,令 T = [T_x T_y],其中 T_x 为点 P 沿 X 方向的平移量,T_y 为沿 Y 方向的平移量,则有:

$$x' = x + T_x, \qquad y' = y + T_y \tag{4-5}$$

相应的向量形式为

$$[x'y'] = [x\ y] + [T_x\ T_y]$$

也即

$$P' = P + T$$

如果我们对平面图形点集中的每一个点都按照式(4-5)进行变换,则可以实现对该图形的平移。图 4-15 表示了对一所房子的轮廓线作平移变换后的结果,其中平移量 T_x = 3, T_y = - 4。

图 4-15　二维图形的平移变换

对于比例变换,则是给定点 P 相对于坐标原点沿 X 方向的比例系数 S_x 和沿 Y 方向的比例系数 S_y,经变换后

$$x' = x \cdot S_x \quad , \quad y' = y \cdot S_y \tag{4-6}$$

若令

$$s = \begin{matrix} S_x & 0 \\ 0 & S_y \end{matrix}$$

则可把式(4-6)写成矩阵形式如下:

$$[x' \quad y'] = [x \quad y] \begin{matrix} S_x & 0 \\ 0 & S_y \end{matrix}$$

即

$$P' = P \cdot S$$

图 4-16 表示的是一所房子的轮廓线以及 $S_x = 1/2$, $S_y = 1/4$ 时作比例变换的结果。显然,我们现在讨论的比例变换是相对原点而言的,故当 S_x, S_y 小于 1 时,房子就变得更小且更接近于原点了。如果比例系数大于 1,则房子将会变得更大并离原点更远。当 $S_x \neq S_y$ 时,也即 x 方向和 y 方向的比例系数不同时,图形会发生形变。至于相对任一点的比例变换,我们将在后面的连续变换中讨论。

图 4-16　二维图形的比例变换

对于旋转变换,先讨论平面上点绕原点的旋转变换。若点 P 相对于原点逆时针旋转 θ 角,那么,从数学上很容易得到变换后的坐标为

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned} \tag{4-7}$$

用矩阵形式表示为

$$[x' \quad y'] = [x \quad y] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

即

$$P' = P \cdot R$$

其中

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

图 4-17 表示一个正方形旋转 45°的情况。关于绕平面上任意一点的旋转变换也将在后面讨论。

图 4-17　一个正方形的旋转变换

二、二维变换的齐次坐标表示

从上面的分析我们看到,对于常用的平移、比例、旋转等几何变换,其矩阵表示分别为:

$$P' = P + T$$
$$P' = P \cdot S$$
$$P' = P \cdot R$$

看得出,对于平移变换的处理方法(加法)和对于比例、旋转变换的处理方法(乘法)是不相同的。为此,我们希望寻求一种一致的或同类的方法来处理这些变换。如果我们采用齐次坐标技术,上述三种变换都可以用乘法来处理。

齐次坐标技术是从几何学中发展起来的,随后在计算机图形学中得到了广泛的应用。正因为采用了齐次坐标技术,图形的变换才可以转化为表示图形的点集矩阵与某一变换矩阵进行矩阵相乘这一单一问题,因而可以借助计算机高速计算功能,很快得到变换后的图形,从而为高度动态的计算机图形显示提供了可能性。

在二维平面中,点 $P(x,y)$ 的齐次坐标表示为 $P(wx, wy, w)$,这里,w 是任一不为 0 的比例系数。显然,(w_1x, w_1y, w_1) 和 (w_2x, w_2y, w_2) 表示的是同一个点 (x,y)。例如(12, 8,

4) 和(6, 4, 2)以及(3, 2, 1)都表示同一个点(3, 2)。这就是说, 二维平面中的点用非齐次坐标表示时, 具有两个分量, 且是唯一的。而用齐次坐标表示, 则具有 3 个分量, 且不唯一, 从广义上说, 齐次坐标技术就是用 n+ 1 维向量表示一个 n 维向量, 而且在 n+ 1 维空间中讨论 n 维向量的变换。通常, 我们总是用 P= [x y 1] 代表齐次坐标表示法中二维平面内一个未被变换的点, 用 3× 3 矩阵表示变换矩阵。若令变换矩阵

$$T = \begin{matrix} a & b & p \\ c & d & q \\ l & m & s \end{matrix}$$

则点 P 经 T 作用后得

$$[X \quad Y \quad H] = [x \quad y \quad 1] \begin{matrix} a & b & p \\ c & d & q \\ l & m & s \end{matrix}$$

$$= [ax + cy + l \quad bx + dy + m \quad px + qy + s]$$

从而可以求出变换后点 P 的二维直角坐标为:

$$x' = X/H = (ax + cy + l)/(px + qy + s)$$

$$y' = Y/H = (bx + dy + m)/(px + qy + s)$$

我们称上述过程为规范化过程。也就是说, 我们在三维空间中进行二维点的变换, 最后仍要回到二维平面中观察其变换结果。变换矩阵中各元素的取值不同, 可以表示不同的变换, 现在分述如下:

1. 平移变换

在变换矩阵中, 取 l= T_x, m= T_y, l 和 m 分别表示点 P(x, y) 沿 X, Y 方向的平移量, 则式(4-5)的平移变换可表示成 P = P · T, 其中

$$T = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{matrix} \tag{4-8}$$

也即

$$[X \quad Y \quad H] = [x \quad y \quad 1] \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{matrix}$$

$$= [x + T_x \quad y + T_y \quad 1]$$

因 H = 1, 规范化后得

$$x' = X = x + T_x$$

$$y' = Y = y + T_y$$

此式即式(4-5), 我们看到, 正是采用了齐次坐标表示法, 平移变换的处理方法由加法变成了乘法。

如果点 P(x, y) 经 T_1 变换后平移了(T_{x1}, T_{y1}), 经 T_2 变换后又平移了(T_{x2}, T_{y2}), 那么将产生什么结果呢? 从直观上看, 应是平移了(T_{x1}+ T_{x2}, T_{y1}+ T_{y2}), 那么用齐次坐标又应如何描述呢? 首先令

$$P' = P \cdot T_1$$

$$P'' = P' \cdot T_2$$

则

$$P'' = P \cdot T_1 \cdot T_2$$

其中

$$T_1 = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_{x1} & T_{y1} & 1 \end{matrix},$$

$$T_2 = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_{x2} & T_{y2} & 1 \end{matrix}$$

而 $T_1 \cdot T_2$ 为

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_{x1} & T_{y1} & 1 \end{matrix} \cdot \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_{x2} & T_{y2} & 1 \end{matrix} = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_{x1}+T_{x2} & T_{x2}+T_{y2} & 1 \end{matrix}$$

这确实说明了实际的平移量是$(T_{x1}+T_{x2}, T_{y1}+T_{y2})$，也就是说，连续的平移变换是平移量相加，但在齐次坐标表示中，可用矩阵相乘来描述。

2. 比例变换

在变换矩阵 T 中，取 $a=S_x$，$d=S_y$，它们分别表示点 $P(x,y)$ 沿 X 和 Y 方向相对原点的比例变换系数，则式(4-6)的比例变换可表示成

$$P' = P \cdot T$$

其中

$$T = \begin{matrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{matrix} \tag{4-9}$$

也即

$$[X \quad Y \quad H] = [x \quad y \quad 1] \begin{matrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{matrix}$$

$$= [S_x x \quad S_y y \quad 1]$$

规范化后得

$$x' = S_x \cdot x$$

$$y' = S_y \cdot y$$

此式即式(4-6)，这说明比例变换仍然可以用矩阵乘法来处理。

连续的比例变换可以通过连续的矩阵乘法来实现。例如，点 $P(x,y)$ 经比例变换 $T_1(S_{x1}, S_{y1})$ 后，再经比例变换 $T_2(S_{x2}, S_{y2})$，那么，最终的比例变换矩阵 $T = T_1 \cdot T_2$，即

$$\begin{matrix} S_{x1} & 0 & 0 & S_{x2} & 0 & 0 \\ 0 & S_{y1} & 0 & 0 & S_{y2} & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{matrix} = \begin{matrix} S_{x1}S_{x2} & 0 & 0 \\ 0 & S_{y1}S_{y2} & 0 \\ 0 & 0 & 1 \end{matrix}$$

3. 旋转变换

旋转变换公式(4-7)可表示成

$$P' = P \cdot T$$

其中

$$T = \begin{matrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{matrix} \tag{4-10}$$

其中 θ角的正负值是这样来确定的,即点 P 在 XY 平面的内绕原点逆时针旋转所形成的角度为正,反之为负。

两个连续的旋转变换应该是角度的相加。设点 P 经旋转变换 T_1(θ 角)后,再经旋转变换 T_2(θ 角),则最终的旋转变换阵 T 应是 $T_1 \cdot T_2$,,即:

$$\begin{matrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{matrix} \begin{matrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{matrix} = \begin{matrix} \cos\theta\cos\theta - \sin\theta\sin\theta & \cos\theta\sin\theta + \sin\theta\cos\theta & 0 \\ -\sin\theta\cos\theta - \cos\theta\sin\theta & -\sin\theta\sin\theta + \cos\theta\cos\theta & 0 \\ 0 & 0 & 1 \end{matrix}$$

$$= \begin{matrix} \cos(\theta+\theta) & \sin(\theta+\theta) & 0 \\ -\sin(\theta+\theta) & \cos(\theta+\theta) & 0 \\ 0 & 0 & 1 \end{matrix}$$

4. 对称变换

对称变换也称反射变换或镜象变换。

(1) 对称于 Y 轴

当变换矩阵中 b= c= 0, a= - 1, d= 1,而 l= m= 0, p= q= 0 及 s= 1 时,我们有

$$T = \begin{matrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \tag{4-11}$$

则点 P 经 T 作用后,所得 P 点的坐标为:

$$x' = -x$$

$$y' = y$$

也即 P 与 P 相对于 Y 轴是对称的,如图 4-18(a)所示。

(2) 对称于 X 轴

当变换矩阵

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-12}$$

时,则点 P 经 T 作用后,所得 P′点的坐标为

$$\begin{aligned} x' &= x \\ y' &= -y \end{aligned}$$

即得到对 X 轴对称的点,见图 4-18(b)。

(3) 对称于原点

当变换矩阵

$$T = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-13}$$

时,则点 P 经 T 作用后,

得

$$\begin{aligned} x' &= -x \\ y' &= -y \end{aligned}$$

也即相对于原点是对称的,见图 4-18(c)。

(4) 对称于直线 y= x

当变换矩阵

$$T = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-14}$$

时,则点 P 经 T 作用后,

得:

$$\begin{aligned} x' &= y \\ y' &= x \end{aligned}$$

即产生相对于直线 y= x 的对称点 P′,如图 4-18(d)。

(5) 对称于直线 y= - x

当变换矩阵

$$T = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4-15}$$

时,点 P 经 T 作用后,

得:

$$\begin{aligned} x' &= -y \\ y' &= -x \end{aligned}$$

即产生了相对于直线 y= x 的对称点 P′,如图 4-18(e)所示。

5. 错切变换

错切变换也称为剪切、错位或错移变换。

(1) 沿 X 轴方向关于 Y 的错切

图 4-18　二维图形的对称变换

(a) 对称于 y 轴;　(b) 对称于 x 轴;　(c) 对称于原点;

(d) 对称于直线 y= x;　(e) 对称于直线 y= - x。

此时变换矩阵

$$T = \begin{matrix} 1 & 0 & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad (4\text{-}16)$$

点 P 经 T 作用后, P 点坐标为:

$$x' = x + cy$$
$$y' = y'$$

即变换前和变换后 y 坐标不变, 而 x 坐标依赖于初始坐标值(x, y)及元素 e 的值呈线性变化。图 4-19(a)说明了矩形 ABCD 经错切变换后变为 $A^* B^* C^* D^*$, c= tgα如 c> 0, 则沿+ x 方向错切; 若 c< 0, 则沿- x 方向错切。

(2) 沿 Y 轴方向关于 X 的错切

当变换矩阵

$$T = \begin{matrix} 1 & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \qquad (4\text{-}17)$$

时, 则点 P 在 T 的作用下变为 P′, 其坐标为:

$$x' = x$$

$$y' = y + bx$$

即变换前后 x 坐标不变,而 y 坐标与初始坐标值(x,y)和元素 b 有关。图 4-19(b)说明矩形 ABCD 经错切变换后变为 $A^* B^* C^* D^*$ 的结果。$b = tg\beta$ 如 $b > 0$,说明点沿+ y 方向错切;若 $b < 0$,则点沿- y 方向错切。

图 4-19　二维图形的错切变换

(a) 沿 x 轴轴关于 y 的错切;　(b) 沿 y 轴关于 x 的错切。

6. 组合交换

所谓二维图形的组合变换,就是在 XY 平面内,对一个已定义的图形,按一定顺序进行多次变换而得到新的图形。通常,我们总是试图将一些复杂的变换转换成一个个上面介绍过的简单变换,也即用各个简单变换矩阵的连乘,表示总体变换的效果。另外,我们在讨论图形变换问题时可以采用前面讨论过的方法,即认为坐标系不动,而图形中的坐标点在变化。实际上,从相对的观点来看,处理图形变换时,也可以使图形不变,而让坐标系作相应的变化。从效果上看,两者是等效的,只是应考虑变换参数是取正值还是取负值。例如,点 P 沿 X 轴正向平移 T_x,相当于坐标系沿 X 轴反向平移 T_x,即坐标系平移了 $-T_x$。又如,点 P 绕原点逆时针旋转 θ角,等价于坐标系统原点顺时针旋转了 θ角等等。因此,在以后讨论的各种变换中,可以根据需要,有时令图形变化,有时令坐标系变化。

现在我们来讨论 XY 平面内,点 P(x,y)绕任意点 $A(x_A, y_A)$ 逆时针旋转 θ角的组合变换表达式。如图 4-20 所示,为了实现点 P 绕任意点 A 作旋转变换,需要连续作如下三个基本的变换:

(1) 平移坐标系,使坐标系原点与任意点 A 重合,形成新的坐标系 AX′Y′。平移变换阵 T_1 为

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_A & -y_A & 1 \end{bmatrix}$$

点 P 在新坐标系下的坐标点为 P′,且 $P' = P \cdot T_1$。

图 4-20　绕任意点的旋转变换

(2) 在新坐标系 AX′Y′下,使点 P 绕 A 点(即坐标系原点)旋转。若逆时针旋转 θ角,则旋转变换阵 T_2 为

$$T_2 = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

点 P 在 T_2 作用下变为 P″,即 P″= P′· T_2。

(3) 再作平移变换。使坐标系 AX′Y′回到原来位置,即与 OXY 坐标系重合。平移变换阵 T_3 为:

$$T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_A & y_A & 1 \end{bmatrix}$$

在 T_3 作用下,点 P′变为 P,即 P = P″· T_3,其中,P′是点 P 绕 A 点旋转后,在 AX′Y′下的坐标点,而 P 才是点 P 作上述变换后,在 OXY 下的最终坐标点。因此,组合变换应为:

$$\begin{aligned} T &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_A & -y_A & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_A & y_A & 1 \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ x_A(1-\cos\theta) + y_A\sin\theta & y_A(1-\cos\theta) - x_A\sin\theta & 1 \end{bmatrix} \end{aligned}$$

显然,这个结果与点 P 绕原点作旋转变换的结果大不一样。

可以用类似的方法实现相对于任意点作比例变换。若任意点 A 的坐标为(x_A, y_A),比例变换系数为 S_x, S_y,则在此情况下的组合变换为

$$\begin{aligned} T &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_A & -y_A & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_A & y_A & 1 \end{bmatrix} \\ &= \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ x_A(1-S_x) & y_A(1-S_y) & 1 \end{bmatrix} \end{aligned}$$

由于一般说来,矩阵相乘是不可互换的,所以组合变换应严格遵照一定的顺序。假定 T_1 和 T_2 分别表示一个基本的平移、比例或旋转变换,那么在一般情况下,$T_1 \cdot T_2 \neq T_2 \cdot T_1$。而在一些特殊情况下,这种互换是允许的。比如,在两个平移变换之间,或在两个比例变换之间,或是两个旋转变换之间,或当 $S_x = S_y$ 时的比例变换与旋转变换之间,可以互换。此时,就不需要关心矩阵乘法的顺序了。

至此,我们看到,二维平面内的一般几何变换,均可以用 3× 3 变换矩阵来描述。变换矩阵

$$T = \begin{bmatrix} a & b & p \\ c & d & q \\ l & m & s \end{bmatrix}$$

从功能上可分为如下 4 部分:

(1) 2× 2 子阵 $\begin{matrix} a & b \\ c & d \end{matrix}$ 用来描述比例、反射、错切及旋转等变换;

(2) 1× 2 行阵[l m]完成平移变换功能;

(3) 元素 s 产生整体的比例变换。它使整个图形沿 X、Y 轴作等比例均匀变换。与局部比例变换系数不同的是: 当$\left|s\right|<1$时, 图形放大; 当$\left|s\right|>1$时, 图形缩小; 若 s< 0, 图形相对原点作对称的比例变换;

(4) 2× 1 列阵 $\begin{matrix} P \\ q \end{matrix}$ 它完成透视变换, 产生投影。

齐次坐标表示技术还可以用来表示无穷远点。一般说来, 二维齐次向量 a b 0 表示的是 ay- bx= 0 直线上的无穷远点。因此, X 轴上的无穷远点可用[1 0 0]表示; Y 轴上的无穷远点可用[0 1 0]表示。广义地说, 附加分量 h= 0 的 n+ 1 维齐次坐标, 表示的是一个 n 维的无穷远点, 或称非固有点。另外, 我们还可以利用齐次坐标技术, 通过适当选择 3× 3 变换阵中的某些元素值, 使无穷远点变换成与之对应的有限远点。例如, 我们可以用一般的 3× 3 变换阵

$$A=\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix}$$

来变换 X 轴上的无穷远点[1 0 0], Y 轴上的无穷远点[0 1 0]及坐标原点[0 0 1], 也即

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix}=\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix}$$

结果, [1 0 0]变换成[a_{11} a_{12} a_{13}], 只要 $a_{13}\neq 0$, X 轴上的无穷远点就不再是无穷远点了。同样[0 1 0]变成[a_{21} a_{22} a_{23}], 只要 $a_{23}\neq 0$, Y 轴上的无穷远点也不再是无穷远点。所以, 我们说, 齐次坐标技术不仅可以表示无穷远点, 而且可以通过矩阵变换, 将无穷远点变换成有限远点。

第四节 三维图形的几何变换

上述有关二维图形几何变换的讨论, 基本上都适合于三维空间, 只不过三维空间的几何变换要复杂得多。从工程设计角度看, 三维空间几何变换直接与显示、造型有关, 因此更重要一些。

与二维图形变换一样, 我们也采用齐次坐标技术来描述空间的各点坐标及各种变换。显然, 三维空间中的点 P(x, y, z), 用齐次坐标表示应是[x y z 1], 而描述三维空间中各种变换的变换矩阵 T 应是 4× 4 的形式, 若

$$T=\begin{matrix} a & b & c & p \\ d & e & f & q \\ g & h & i & r \\ l & m & n & s \end{matrix}$$

则从功能上也可分为如下部分:

(1) 3×3 子阵 $\begin{matrix} a & b & c \\ d & e & f \\ g & h & i \end{matrix}$, 可以产生比例、旋转、错切、对称等变换;

(2) 1×3 行阵$[l \quad m \quad n]$可以产生平移变换;

(3) 3×1 列阵 $\begin{matrix} p \\ q \\ r \end{matrix}$ 产生射影变换;

(4) 元素 s 产生整体的变比变换。

下面我们将分别讨论三维空间中的几何变换。至于三维形体的投影变换, 我们将在第七章中讨论。

1. 变比变换

(1) 局部变比

当变换矩阵

$$T=\begin{matrix} a & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \tag{4-18}$$

时, 则点 $P(x,y,z)$ 在 T 作用下变为

$$\begin{aligned} x' &= ax \\ y' &= ey \\ z' &= iz \end{aligned}$$

当 $a,e,i>1$ 时, P 点坐标相对于原点被放大; 当 $a,e,i<1$ 时, 则相对于原点被缩小。

(2) 整体变比

当变换矩阵

$$T=\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & s \end{matrix} \tag{4-19}$$

时, 点 $P(x,y,z)$ 在 T 作用下变为

$$\begin{aligned} x' &= x/s \\ y' &= y/s \\ z' &= z/s \end{aligned}$$

显然, 当$s<1$ 时, P 点坐标相对原点沿 X, Y, Z 轴作等比例均匀放大; $s>1$ 时, 则缩

小。

2. 平移变换

当变换阵

$$T=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\l&m&n&1\end{bmatrix}\tag{4-20}$$

时，点 P 经平移变换后为

$$\begin{aligned}x'&=x+l\\y'&=y+m\\z'&=z+n\end{aligned}$$

其中，l，m，n 分别为点 P 沿 X 轴、Y 轴、Z 轴方向的平移量。

3. 对称变换

(1) 相对于 XY 平面的对称变换

变换矩阵

$$T=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&-1&0\\0&0&0&1\end{bmatrix}\tag{4-21}$$

在 T 作用下，点 P(x，y，z) 只对 z 坐标取反，而 x，y 坐标不变，即相对于 XY 平面作对称变换。

(2) 相对于 XZ 平面的对称变换

此时变换矩阵

$$T=\begin{bmatrix}1&0&0&0\\0&-1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\tag{4-22}$$

(3) 相对于 YZ 平面的对称变换

$$T=\begin{bmatrix}-1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\tag{4-23}$$

当空间点相对于任一平面作对称变换时，应先将此平面转换成与某一坐标面相重合，并运用上述简单的对称变换，然后再将平面反变换回原来的位置即可。

4. 错切变换

当变换矩阵

$$T=\begin{bmatrix}1 & b & c & 0\\ d & 1 & f & 0\\ g & h & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{4-24}$$

时,点 P(x,y,z)在 T 作用下变为

$$\begin{aligned} x' &= x + dy + gz\\ y' &= y + bx + hz\\ z' &= z + cx + fy \end{aligned}$$

也就是说,当变换矩阵中 3× 3 子阵的非主对角元为非 0 时,将发生错切变换。我们可以根据这些元素所在的列判断出沿哪个坐标轴出现错切。其中如 d,g 不为 0,则沿 X 轴方向有错切;若 b,h 不为 0,说明沿 Y 轴有错切;而 e,f 不为 0,则沿 Z 轴有错切。另外,我们还可以根据这些元素所在的行,判断出是关于哪个变量的错切。比如 b,c 是关于变量 x 的错切;d,f 是关于变量 y 的错切;g,h 是关于变量 z 的错切。

5. 旋转变换

三维空间中某点绕任意轴的旋转变换比较复杂,我们将在后面组合变换中进行讨论。

如果选定 X、Y、Z 三个坐标轴之一作为旋转轴,则实际上各点只在垂直于旋转轴的坐标平面内作二维旋转,此时运用二维旋转公式就可以推导出三维旋转变换矩阵。这里要注意旋转角度的正负号问题,决定旋转角度正负号的因素有 3 个:一是采用的坐标系是右手系还是左手系;二是旋转对象是形体还是坐标系;三是顺时针还是逆时针旋转。通常,我们规定:在右手坐标系下,物体绕某坐标轴逆时针方向旋转为正;那么,当物体不动,而坐标系旋转时,则方向相反。假如物体在左手坐标系下旋转,那么上述规定的方向正好相反。表 4-1 列出了旋转变换时,上述因素对角度 θ取值的约定。

表 4-1 作旋转变换时,角度 θ取值的约定

θ 取值 / 坐标系 / 类型		右手系	左手系
空间点绕某轴	逆转	正	负
	顺转	负	正
坐标系旋转	逆转	负	正
	顺转	正	负

(1) 绕 Z 轴正方向旋转 γ角

变换矩阵

$$T=\begin{bmatrix}\cos\gamma & \sin\gamma & 0 & 0\\ -\sin\gamma & \cos\gamma & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{4-25}$$

点 P(x,y,z)在 T 作用下变为

$$x' = x\cos\gamma - y\sin\gamma$$
$$y' = x\sin\gamma + y\cos\gamma$$
$$z' = z$$

也即点 P 绕 Z 轴在 XY 平面内旋转，z 坐标不变，x、y 坐标的变化与前面讨论的二维旋转变换一样。当然，旋转轴必须是穿过原点的坐标轴才能得到上面的结果。否则，要用到组合变换。

(2) 绕 X 轴正向旋转 α角

此时变换阵

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4-26}$$

点 P 在 T 作用下变为

$$x' = x$$
$$y' = y\cos\alpha - z\sin\alpha$$
$$z' = y\sin\alpha + z\cos\alpha$$

(3) 绕 Y 轴正向旋转 β角

此时点 P 在 ZOX 平面内作二维旋转，y 坐标不变，即

$$y' = y$$
$$z' = z\cos\beta - x\sin\beta$$
$$x' = z\sin\beta + x\cos\beta$$

所以，变换矩阵应为

$$T = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4-27}$$

6. 三维组合变换

现在我们来讨论绕空间任意轴作旋转变换的问题。如图 4-21 所示，已知空间有一任意轴 AB，A 点坐标为(x_A, y_A, z_A)，AB 的方向数为(a, b, c)。现有一点 P(x, y, z)，绕 AB 逆时针旋转 θ角后成为 P(x', y', z')，若旋转变换矩阵为 R_{AB}，则应有

$$[Z \quad Y \quad Z \quad H] = [x, y, z, 1]\cdot R_{AB}$$
$$x' = Z/H$$
$$y' = Y/H$$
$$z' = Z/H$$

下面讨论如何求出 R_{AB}的表达式。

与二维的组合变换一样，我们试图将绕任意轴 AB 旋转的复杂问题转换成一些诸如平移变换、绕某坐标轴旋转变换等简单问题，用各个简单变换矩阵的连乘表示总体变换的效果。因此，解题的途径是多种的。图 4-21 表示的变换顺序为：先平移坐标系，使 A 点成

图 4-21　绕空间任意轴的旋转变换

为新坐标系 AX Y Z 的原点;再使 AB 分别绕 X 轴、Y 轴旋转适当角度,使 AB 与 Z 轴重合;那么绕 AB 转 θ角就变成了绕 Z 轴转 θ角;最后,再作逆变换,使 AB 回到原来的位置,因此,

$$R_{AB} = T_A \cdot R_x \cdot R_y \cdot R_z \cdot R_y^{-1} \cdot R_x^{-1} \cdot T_A^{-1}$$

其中,T_A^{-1}, R_x^{-1}, R_y^{-1} 分别表示平移变换阵 T_A、旋转变换阵 R_x、R_y 的逆变换矩阵。现分别表述如下:

(1) $T_A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_A & -y_A & -z_A & 1 \end{bmatrix}$ 它形成 AX Y Z 坐标系。

(2) 令△ABB$_1$ 绕 X 轴逆时针旋转 α角,变为△AB$_2$B$_3$,也就是使 AB 轴(其长度为 s)转到 AX Z 平面上,α取正值,则

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

其中,$\cos\alpha = \frac{c}{v}$, $\sin\alpha = \frac{b}{v}$, $v = \sqrt{(b^2 + c^2)}$

(3) 在 AX Z 平面内,令 AB$_2$ 绕 Y 轴顺时针转 β角,变为 AB$_4$,也即让 AB(现在为 AB$_4$)与 Z 轴重合,β取负,则

$$R_y = \begin{bmatrix} \cos(-\beta) & 0 & -\sin(-\beta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(-\beta) & 0 & \cos(-\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

其中, $\cos\beta=\frac{v}{s}$, $\sin\beta=\frac{a}{s}$, $s=\sqrt{(a^2+v^2)}$

(4) 根据题意, 要求绕 AB(即现在的 Z 轴)逆时针转 θ角, 则

$$R_x=\begin{bmatrix}\cos\theta & \sin\theta & 0 & 0\\ -\sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

(5) 作逆变换, 回到 AB 原来的位置。

$$R_y^{-1}=\begin{bmatrix}\cos\beta & 0 & -\sin\beta & 0\\ 0 & 1 & 0 & 0\\ \sin\beta & 0 & \cos\beta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$R_x^{-1}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos(-\theta) & -\sin(-\theta) & 0\\ 0 & \sin(-\theta) & \cos(-\theta) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$T_A^{-1}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ x_A & y_A & z_A & 1\end{bmatrix}$$

至此, 绕空间任意轴旋转的复杂问题得以全部解决。

习　　题

1. 给定矩形窗口及视图区的参数, 且两者不为相似形, 如何保证由窗口到视图区的全部不失真映象。
2. 请简述二维图形裁剪的基本原理及可选用的裁剪策略。
3. 对于编码裁剪算法和中点分隔裁剪算法, 讨论一下一条等长的直线段, 哪种分布(以直线端点的外部码与窗口的交点为特征), 其裁剪的时间最长。
4. 请画出用双边裁剪法实现矩形域对任一多边形的裁剪算法流程。
5. 试证明下述几何变换的矩阵运算具有互换性:
   (1) 两个连续的旋转变换;
   (2) 两个连续的平移变换;
   (3) 两个连续的比例变换;
   (4) 当 $S_x=S_y$ 时, 旋转和比例变换。
6. 试用几种不同顺序的简单几何变换, 求出将平面上的任一直线段 $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ 变换成与 X 轴重合的变换阵, 并说明其等效性。
7. 请利用几何变换原理, 编制并上机实现生成直线箭头图形的 C 语言函数, 格式为

   ARROW (IT, x_1, y_1, x_2, y_2, IL, IW)

   其中 IT = 1 画单向箭头; IT = 2 画双向箭头。IL 为箭头部分长度, IW 为箭头部分宽度。(x_1, y_1), (x_2, y_2) 由鼠标器在屏幕上确定。
8. 利用几何变换, 编制并上机实现矩形绕其左下角点旋转的 C 语言函数。要求每隔 36 度画 1 个, 共画

10 个。

9. 采用旋转及比例变换的原理，编制并实现产生正方形螺旋线的 C 语言函数，使每个后继的正方形被旋转某一固定角度 R，且被放大，同时使它完全包含前 1 个正方形。

10. 利用变换原理，实现两个已知圆的内、外公切线的输出。

11. 给定空间一点 P(x, y, z)及任意一平面 θ 平面方程为

$$ax + by + cz = d = 0$$

请用几个简单的几何变换级联，求出 P 对 θ的对称点 P′(x′, y′, z′)的复合变换矩阵。

第五章　交互技术及用户接口

用户接口是人与计算机交互作用的界面。人-机交互由最早的批处理作业方式、问答方式、逐步发展成菜单交互方式。随着计算机图形学的迅速发展和计算机硬件设备性能价格比的极大提高，人-机交互又由菜单交互方式发展成今天的图形交互接口。现在，窗口管理系统、图标、丰富多彩的图形界面，已成为流行的人-机交互环境。即使是对计算机不熟悉的人，也能通过图形交互接口用计算机完成自己的工作。

本章先介绍用户接口中的基本交互任务和基本交互技术。与交互技术有关的交互设备，我们已在第二章中作了详细的介绍，这里就不再重复。在基本交互技术的基础上，我们将进一步介绍各种组合交互技术。接下来，将介绍交互接口的设计，包括交互接口的风格，交互接口的设计步骤和设计原则。最后是一些用户接口软件。通过对这些交互软件的了解，可以进一步加深图形交互接口的概念。根据交互接口软件所处的地位，我们将由低层次的基本图形包中的基本输入控制方式开始介绍，最后是窗口管理系统和交互技术工具箱。侧重点将放在这些软件的设计和实现方法，以及用户的应用程序如何利用这些软件实现自己的用户接口。

第一节　基本交互任务及其技术

交互式图形系统的基本交互任务包括：定位、选择、文字输入、数值输入。这一节着重描述各个基本交互任务，并针对不同类型的交互任务介绍一些交互技术。

一、定位

定位任务是指向应用程序指定一个点的坐标(x，y)或(x，y，z)。实现定位的交互技术通常有两种：一是把屏幕上的光标移到要确定的点，再按一下键；再就是用键盘键入那个点的坐标。定位设备可以是直接的(如光笔)或间接的(如数字化板、鼠标器、游戏棒)，可以是连续的(如鼠标器)或离散的(如光标控制键)，也可以是绝对的(如数字化板)或相对的(如鼠标器)。另外，用键盘上的上下左右键也可以移动光标，用语音识别设备也可以发出相同的命令。还可以将这些技术混合在一起使用，如用鼠标器控制光标移动到定位点附近，再用键盘上的上下左右键一次一个象素地移动光标，进行精确定位。

下面介绍一些在定位中要考虑的基本问题。

1．坐标系统

定位中的一个重要问题是在哪个坐标系中给出反馈。在交互式图形系统中，一般有三个坐标系：屏幕坐标系、用户坐标系及物体坐标系。如果一个定位设备为了拖动一个物体而向右移动，那么这个物体应朝哪个方向运动呢？至少有三种可能性：物体在屏幕坐标系中沿 X 轴正向运动；物体在用户坐标系中沿 X 轴正向运动；物体在物体坐标系中沿 X 轴

正向运动。第一种选择,即物体在屏幕坐标系中沿 X 轴正向运动,是正确的。因为在后两种情况中,沿 X 轴正向运动不一定就是沿屏幕坐标系的 X 轴正向运动。例如,如果视图变换是一个 180 的旋转,则用户坐标系的 X 轴正向与屏幕坐标系的 X 轴正向正好相反,因而定位器的右向运动将会导致物体在屏幕上的向左运动。这样的反馈违背了刺激与响应相吻合的基本原则,即系统对用户动作的响应必须是在同一方向上,响应的幅度应与动作的大小成正比。如果把鼠标器倒着放置,或把台板的方向转 180°也会产生同样的问题。

2. 分辨率

定位任务中所要求的分辨率,随应用程序的不同而不同,一般会从几百变化到几百万。键盘输入坐标值(x, y)时,可以提供无限高的分辨率,因为输入的数字串可以根据需要来决定长度。台板、鼠标器等定位设备的分辨率至少和分辨率为 500～2000 的显示器一样高。在定位设备的分辨率达不到用户程序的要求时,可以用窗口到视口的坐标变换技术将用户坐标系中的某个区域放大,从而使屏幕上的一个象素单位与用户坐标系中任意小的单位对应起来。

3. 网格

在许多定位任务中,一个重要的视觉辅助工具就是在屏幕工作区中用较低的亮度或较淡的颜色显示出网格,以帮助调整定位点的位置,使图素的端点落在网格上。网格帮助用户生成整齐的图形。要实现网格化,应用程序可以在定位点已靠近某个网格点时,简单地将定位器的坐标截断到这个最近的网格点上。网格通常是均匀而规则的,充满整个屏幕,但不规则网格、不同区域不同网格、旋转网格在作图时也很有用。

4. 反馈

不同定位方式需要不同的反馈形式。有两类定位方式:空间定位方式和数值定位方式。对于空间定位方式,例如在两个矩形之间画一条线,或把一个物体放在另外两个物体的中间等,在与附近图素的空间关系中,用户知道定位点可能在哪儿。这时用户最希望得到的反馈是光标点的实际位置。而在数值定位方式中,用户知道的是定位点的坐标值(x, y),因此希望得到的反馈当然是光标点的位置坐标。如果提供的反馈形式不对,用户就得将一种形式的反馈转换成另一种形式。既显示光标又显示数值坐标,就可以提供两种形式的反馈,如图 5-1 所示。

图 5-1　数值反馈

数值反馈表示了被构造体的尺寸,高和宽的值随光标(+)的移动而改变,这样,用户就可以将物体调整到希望的大小

5. 方向性

可以限制某些定位设备作任意方向的移动。比如,PC 鼠标器的侧面有两个键(图 5-2)。一个是水平移动限制键,当这个键被按住后,无论鼠标器如何移动,屏幕上的光标只能上下运

图 5-2　鼠标器上的方向限制键

动,因为鼠标器的 x 坐标的变化被封锁了。这在画竖直线时很有用。另一个键是垂直移动限制键,用来封锁 y 坐标的变化,常用来画水平线。

二、选择

选择任务是指从一个被选集中挑选出一个元素来。在作图系统中,操作命令、属性值、物体种类、物体等都是可能的被选集。

被选集可根据其元素的变化程度被分成固定集和可变集两类。固定集是指被选集的大小是相对固定的,虽然可以变化,但不是经常性的,而且变化不多。如命令、属性值、物体种类。可变集是指被选集的大小变化非常大,而且经常变,如物体集,在作图过程中随时变化。

有些交互技术适合上述两类被选集,而有些则不然。例如,指点技术是在一组可以看得见的元素中用指点元素的方法来达到选择这个元素的目的,不需要关心这个元素的类型,所以指点技术既适合固定集,又适合可变集。功能键可以用来选择命令、属性、物体类型,但很难用来选择物体,因为物体的个数是变的,也多于功能键的数目,不能为每一个物体定义一个功能键。

(一) 可变集的选择技术

适合可变集的选择技术有: 指名和拾取。

1. 用指名方式选择物体

用户直接在键盘上键入被选物体的名字。这个想法比较简单,但如果用户不知道物体的名字就难办了,这在显示成千上万个物体时会经常发生的。但无论如何,这个技术在以下两种情况下是非常有用的:

① 用户知道各种物体名字。用指名方式选择物体可能比拾取更快。特别是在物体很多、作图区很大,用户要通过卷滚屏幕才能把要选的物体找出来显示时更是如此。

② 显示的内容杂乱无章、互相遮挡重叠,用拾取方法很难实现选择。键入被选物体的名字可能是最后一招。

特殊匹配符* 和? 可以用来进行多个选择。指名选择方式对于那些有经验的长期用户是合适的。

指名选择的最佳反馈形式是在键入每一个字母后,立即显示所有能与当前已键入的字串相匹配的物体名。这有两个好处,一是可以帮助用户在不记得全名时回忆起来,再就是在没有二义性的情况下,即只有一个元素与已键入的字串匹配时,用户就可以停止键入剩下的字母了。例如,在物体集

{polyline1, polyline2, polygon, cycle}

中,用户希望选择元素 polygon。当键入 p 时,反馈显示 3 个元素 polyline1, polyline2, polygon。直到键入 poly 时,仍显示这 3 个元素。当再键入 g 后,只显示唯一的元素 polygon。这时用户就无需再继续输入 on,因为要选的元素 polygon 已唯一地出现。

在指名选择中,还有一个技术是拼写校正。当键入的名字与集合中的任何元素名都不匹配时,应显示集合中与这个键入的名字相近的元素名,让用户辨认。

在指名选择方式中,除了用键盘输入元素名字外,还可以用语音识别器,说出元素的

名字、名字的缩写或代码，也可以采用字符识别技术，在台板上用指示笔写出元素名。

2．拾取物体

将屏幕上的光标移到被选择的物体上，再按一下鼠标器的键，即指示要拾取这个物体。这实际上是一种指点方式，指点物体以达到选择的目的。拾取通常是在一个有层次结构的物体集上进行的。图 5.3 显示了房子及其层次结构。墙和房顶构成了房子，而墙的中间又开有一个方窗和一个半圆窗。

图 5.3　房子及其层次结构表示

当光标落在半圆窗内时，用户并不清楚是拾取半圆窗、拾取墙，还是拾取整个房子。因此应该有命令或手段使用户确切地知道所选物体属于哪一层。有两种方法可以帮助用户。一是用 SET- PICK- LEVEL 这样的命令来指定选择对象的层次。例如，若要选择墙，则用 SET- PICK- LEVEL 命令将选择层设为 2。二是用 Up- hierarchy(上一层)和 Down- hierarchy(下一层)这样两个命令改变当前层次。拾取时，被选中的物体中最高一层的物体颜色将会改变。如果这就是用户要选的物体，则用户可以继续他的后续工作。否则，用户发出 Down- hierarchy 命令，向下走一层，这一层中被选中的部分将会改变颜色。如果这还不是用户要选的，则他可以继续用 Down- hierarchy 命令向更下一层走。反之，用 Up- hierarchy 命令可以向反方向走，即向上层走。

(二) 固定集的选择

适合固定集的选择技术有：

① 指名技术：用键盘键入集合元素的名字、名字的缩写、或其它表示元素的代码或序号；或用语音识别设备识别所说的集合元素的名字、缩写字母、代码或序号；或用字符识别设备识别手写的名字、缩写字母、代码或序号。

② 功能键：使用与集合元素相对应的功能键。

③ 菜单技术：在菜单中指点菜单项。

④ 模式识别：对于被选集中的每一个元素，交互系统先定义一个互不重复的运动模式。用户用连续定位设备产生一系列运动，识别器自动地将这一运动系列与这组已定义好的运动模式进行比较匹配，做出选择。

下面我们重点介绍菜单设计中的一些关键问题。

1．菜单顺序

菜单中的元素可以按不同的顺序排列。如按字母顺序排列；按功能分组排列；按使用频度排列，最常用的放在前面；按重要程度排列，最重要的排在前面。这些排列方法可以组合起来使用。如按功能分组的菜单，组内的元素仍按字母顺序排列，而组与组之间按使用

频度排列。图 5.4 列出几种菜单的顺序。

对于同一个系统,所有的菜单都按同一种风格进行排序是非常重要的。实验证明,按功能分组排序的菜单对用户最有帮助。很多窗口系统,如 MS. Windows 的菜单结构就能反映这一实验结果。

2. 菜单层次

在菜单项很多,一次不能全部显示出来时,有两种办法可以解决。一是使用带卷滚条的单层菜单或分页菜单;二是将菜单项组织成多层菜单。

卷滚条技术在很多窗口系统中得到了应用。通过移动卷滚条上的小方块,可以在一个不大的可见域内显示全部菜单元素,让用户挑选。图 5-5 是一个带卷条的菜单。卷滚条既可以用鼠标器直接操作,也可以用在键盘上定义的加速键。用鼠标器操作时,定位选择滚动箭头可以向上或向下移动一行菜单项,上下拖动卷滚方块,可以上下滚动菜单窗口。把光标定位在菜单的可见部分可以进行选择。加速键 PAGE UP 和 PAGE DOWN 分别向上和向下滚动菜单窗口, CTRL+ HOME 和 CTRL+ END 分别卷滚到菜单头和菜单尾,方向键↑和↓在菜单的可见部分选择元素。

图 5.4 三种菜单组织

(a) 按字母顺序排列; (b) 按功能分组,组内元素、组与组仍按字母顺序; (c) 按功能分组,组内按字母顺序,组间按使用频度。

分页菜单如图 5-6 所示,也是一种单层菜单。菜单项按某种顺序排列。屏幕上只显示一页菜单,用 PAGE UP 命令或上键头显示上一页菜单,用 PAGE DOWN 或下键头显示下一页菜单。

图 5-5 带卷滚条的菜单

图 5-6 分页菜单

对于多层菜单,有多种表现形式, MS Windows 的下拉式菜单是一种两层菜单。第一层菜单以菜单条的形式始终显示在窗口上(图 5-7a)。当按住鼠标器的左键,把光标移到菜单条中的某个菜单项上时,这个菜单项对应的子菜单,即第二层菜单就在这个菜单项的下方显示出来。使用键盘上的加速键时,先按 ALT 键选取菜单条,利用左右键移动光标选取菜单条上的菜单项,回车则显示对应这个菜单项的子菜单(图 5-7b)。光标可以在子菜单中上下移动选择子菜单项。

逐级显示是多层菜单的另一种表现形式。图 5-8 是一个三级菜单,(a)、(b)、(c)分别是逐级显示的结果。由每一级被着重显示的菜单项可以看出从第一级开始到光标定位点所在的子菜单为止的选择路径。菜单上有带右键头符号的菜单项和不带右键头的菜单项。右键头符号说明这个菜单项不是最终的菜单项,而是下一层子菜单的名字。

还有一种多层菜单的表现形式是板式菜单(图 5-9),它比逐级显示的菜单要占用更

(a)

(b)

图 5-7　下拉式菜单

(a) 窗口中的菜单条; (b) 拉出的子菜单

图 5-8　弹出式逐级显示菜单

(a) 鼠标器左键按下后, 在光标的位置显示的第一级菜单, 光标可以上下移动选择菜单项; (b) 光标再向右拉出的第二级菜单; (c) 用同样的方法拉出的第三级菜单。

多的屏幕显示区,但可以在每一级菜单中使用分页技术或卷滚条技术。图中最左列是第一级菜单,与这列被选中的菜单项对应的子菜单显示在这列的右边一列。如果这项没有子菜单、则右列为空。

设计多层菜单,关键是要确定菜单的深度(即菜单的层数)和宽度(即每层的菜单项数)。实验表明,使用层次不多的菜单(如二级菜单或三级菜单)会提高选择的速度和正确性。

图 5-9　带分页的板式菜单

加速键技术可以提高熟练用户在多层菜单中选择菜单项的速度。通常每层菜单的每一项都定义一个唯一的名字,用户可以直接键入这个名字。有时在同一层,各项的名字是唯一的,但层与层之间的菜单项的名字可能会重复,这时,用户就要键入整个选择路径的名字。

3．菜单的放置

菜单可以显示在屏幕上或辅助屏幕上,也可以印在台板上,或粘在功能键上。屏幕菜单可以是静态的,自始至终显示在屏幕或窗口的某一固定区域里;也可以是动态的,在用户请求时才显示在屏幕上,用户选择后立即消失。MS Windows 的菜单条是静态菜单,而弹出式菜单、下拉式菜单则是动态菜单。

印在台板上的固定菜单,适合于专用图形系统。如图 5-10 是二维交互绘图系统 ME10 的台板菜单(局部)。用台板菜单,有两个好处,一是用户可以在台板上看到所有的

图 5-10　ME10 的台板菜单(局部)

菜单,二是可以节约主屏幕的显示区域。但用台板菜单或辅助屏幕菜单时,用户不得不经常地把视线从主屏幕移开,视觉上的连续性受到了影响。

弹出式菜单在用户有选择请求时显示在屏幕上,选择任务完成后就从屏幕上消失。菜单一般在以下两种情况下弹出:一是用户有一个明显的动作,如按鼠标器的左键;二是在下一步交互过程中需要选择菜单。菜单通常在光标位置弹出,因为光标位置通常是用户视力集中的地方,可保持视觉的连续性。一个值得重视的特性是菜单弹出时的状态。如果认为上次选中的菜单项在这次更有可能选中的话,那么在菜单弹出时,这项就要着重显示出来,而且光标也要定位在这一项上。如果菜单是按使用频度排序的话,那么在菜单弹出时,

就应该着重显示最常用的菜单项,并使光标定位在这一项上。而且这一项不应放在菜单的顶部,而应放在菜单的正中位置。这样选择其它项时可以减少光标的移动距离。

弹出式菜单节省了宝贵的屏幕空间,是交互接口设计中一个最有价值的工具。用快速光栅操作指令 RasterOp 就可以实现弹出式菜单。

在光标位置到底弹出什么内容的菜单是由光标位置的显示内容决定的。例如,在大多数窗口管理系统中,如果光标在窗口的标题横条中,则弹出控制窗口操作的菜单;如果光标在窗口的外部(即屏幕的背景),则弹出控制窗口管理系统的菜单;如果光标在窗口的内部,则弹出的菜单与应用程序有关。这种联想式菜单一开始会迷惑一些新手,但他们很快就会明白,而且的确很有用。

下拉式菜单和拉出式菜单通常是从屏幕边上或窗口边上的菜单条里拉出的。Apple Macintosh、MS-Windows 都使用下拉式菜单。图 5-11 是一种拉出式菜单。跟下拉式菜单一样,拉出式菜单也是二层菜单。

图 5-11　拉出式菜单

4. 被选项的着重表示

被选项应着重表示。当光标进入菜单区时,光标所在的

图 5-12　菜单被选项的着重表示方式

(a) 反转表示;　(b) 带圆按键的菜单;　(c) 带对号的菜单;

(d) 带移动指针的菜单;　(e) 带专门选择结果域的菜单。

菜单项就应着重显示;当光标在菜单上移动时,光标进入哪个菜单项,哪个菜单项就被着重显示,光标离开这个菜单项,这个菜单项就恢复成原先的正常显示。当用户没有做出任何选择而把光标移出菜单区时,菜单项就都恢复成原先正常的显示。对于弹出式或下拉式菜单,当光标移出时,菜单就自动消失。

着重表示被选项的方式很多。例如高光显示、反转显示。有的菜单项前带圆按键或方按键,按键开关的状态表示选择的结果。有的菜单是带确认号的,有确认号的项表示被选中,这种菜单可以进行多种选择。有带移动指针的菜单,指针随光标移动,指针的位置就是当前所选项。还有带专门选择区的菜单,选择的结果放在选择区中。图 5-12 列出了各种选择结果的着重表示方式。

5. 菜单项的大小和形状

菜单项的大小直接影响选择的速度和正确性。较大的菜单项可以加快选择速度,但要占更大的屏幕域;较小的菜单项可以在一个固定的区域内显示多个,但在选择时会引起更多的失误。因此用小菜单项节省空间还是用大菜单项提高选择速度,减少失误,应该在设计时权衡利弊予以选择。

菜单的形状一般都是长条的,但图 5-13 显示了另一种形状的菜单。这是一种圆饼式菜单。弹出时光标在圆的中心。这种形状的好处在于当光标从中心向要选择的目标移动时,宽度越来越大,从而减少了可能的选择失误。另一个好处在于光标到每一个菜单项的距离是相等的。

图 5-13　有四个菜单项的圆饼菜单

三、文字输入

文字输入任务是指向应用程序输入一个字符串。如键入一个插图说明,向文字处理器输入文字等。应用程序对输入的字串不赋予任何特殊意义,因此,键入一个命令名或菜单项不属于文字输入任务。文字输入中最基本的技能就是会用标准键盘。

1. 字符识别

用户用一个连续定位设备(如台板指示笔)写印刷字符,由计算机来识别。这种手写印刷字的识别要比识别扫描输入的字符容易一些。因为台板记录了写字的顺序、方向、速度和笔划的压力。模式识别算法是将每个字与存好的模板匹配。例如大写字母 A 由三笔组成,两笔朝下,一笔水平。可以训练识别器使之分辨不同风格的字体,从用户写的样本中计算出各个字的参数。

到目前为止,手写体字符的识别还没有简单的算法,因为手写字的变化太大,而且把一个词分割成一个一个的字母也很难。汉字识别工作虽然也有很大的发展,但成熟的识别系统还不多。

2. 从菜单中挑选

把字母、词、词组、短语等以菜单形式显示出来,由用户用选择设备在菜单中挑选,完成文字输入任务。这一技术在某些情况下很有用。首先,如果只输入一个短的字符串,用户的手已在一个定位设备上,则菜单选择会比把手移到键盘上、键入字串后再回到定位设

备上要快得多。第二,如果字符集过大(如汉字集),用键盘无法表示,则用这个方法代替键盘是非常合理的。事实上,汉字的拼音输入法就是一种二级菜单。通常是先输入某个汉字的拼音。所有与这个拼音相同的汉字都显示在屏幕下方的横条里,由用户在其中挑选。

3. 字符输入技术的评价

对于量大的文字输入,唯一可以取代专业打字员的方法是自动扫描识别。图 5-14 显示了各种输入技术的输入速度的实验值。语音输入没有在图表上表示出来。它现在还比较慢,但对那些必须腾出手来做其它工作的人来说是很有吸引力的。

图 5-14　各种文字输入技术的速度(以每分钟击键次数或字数统计)

四、数值输入

量的交互输入工作是指在某个最小值和最大值之间指定一个数值。典型的交互技术有:

① 由键盘输入数值;

② 调节电位器的阻值产生相应的数值;

③ 手写数字的识别或语音识别;

④ 用上下翻转数字的计数器选择数值;

⑤ 用光标移动屏幕上的标度盘或刻度尺上的指针。

电位器是一种数值输入设备。调节电位器的阻值,就可以产生与这个阻值相应的数值,就如调节收音机的音量一样。电位器有两种,一种是旋钮式电位器,一种是杆式电位器。旋钮式电位器在旋钮上带一个指针指示当前的值。虽然杆式电位器的值比旋钮式电位器的值容易读,但大多数图形系统的厂家只提供旋钮式电位器,因为旋钮容易调节。调节电位器时注意方向的一致性是非常重要的。通常顺时针旋转和向上移动表示值的增加。

图 5-15 是一种上下翻转的计数器。把光标移到‘ + ’号上,按鼠标器左键,可以使‘ + ’号下面的数字加 1;光标在‘ - ’号上时按左键可以使‘ - ’号上面的数字减 1。用这种计数器可以得到高精度的数值。可以节省显示空间,但操作较慢。

图 5-16 是一些标度盘和刻度尺。用户把光标移到标度盘或刻度尺的数值指示器上,

按着鼠标器的左键沿着标尺拖动数值指示器(指针或方块)到希望的数值处,然后松开左键。指针或方块所指的数值就是所要输入的值。通常在移动指针或方块的过程中,还用数字显示反馈值。用标度盘或刻度尺输入数值,其值的范围和精度在使用象鼠标器这样的相对定位设备时可以得到提高。

图 5-15　上下翻转计数器

图 5-16　标度盘和刻度尺及其数值反馈

交互输入数值时,反馈的形式非常重要。一般是提供数字反馈,这在用户知道要输入的数值大小时很有用。另一形式的反馈是刻度盘或标尺上的指针,指示数值的大概位置,这在用户使用电位器这样的输入设备时很有用。

第二节　基本交互任务的组合

在基本交互任务的基础上,可以形成以下三种形式的组合交互任务:

① 对话框: 用来设定多个参数;

② 构造: 用来构造需要两个或多个位置的形体;

③ 操作: 修改已有的几何形体的形状,调整物体之间的相对位置。

一、对话框

我们常常要在一个集合中选择多个元素。如在字符属性集中,实体、斜体、空心字等都不是排他的,用户可以选择多个。另外有些集合是相关的,如字体和字型。菜单技术在一个集合中选一个元素是有用的,但不适于作多个选择。例如弹出式菜单在一个选择完成后就消失了,无法做第二个选择。

对话框可以解决上述问题。对于对话框,在用户明确地指示关闭对话框之前,对话框将一直显示在屏幕上。对话框可以保证多个选择。可以提供输入文字和数值的区域,可以立即改变选择。当所有信息都输入到对话框中之后,给出一个命令,一般说来,对话框就消失了,系统就按选择的属性或给定的参数执行命令。有时,也可以用“APPLY”这样的命令在对话框不消失的情况下使用新的参数执行命令。

对话框是用户与系统进行信息交流的窗口。当系统执行的某个命令需要更多的信息

或数据时,就显示对话框,向用户提问,由用户填写或选择对话框中的各项,作为回答。

作为例子,下面列出窗口管理系统 MS Windows 中对话框的一些内容。

① 列表框(图 5-17):对话框中的列表框显示可以选择的栏目。如果可选项太多,无法在列表框中显示出来时,则提供卷滚条。在列表框中只允许选择一项。

图 5-17　带列表框的对话框图

② 下拉式列表框(图 5-18):在对话框太小或太拥挤,不能显示列表框时,采用下拉式列表框。对话框中只显示一个带下键头方块的长条框。当光标移到下键头方块处定位时,在长条框的下方将弹出一个列表框供选择。选择的结果在长条框中显示。选择结束后,弹出的列表框即消失。在下拉式列表框中也只能选择一项。

③ 任选项开关(图 5-19):任选项开关为一些排它项的列表,是一种带圆键的菜单,每次只能从列表中选择一项。用圆键表示选择结果。被选项的圆键中包含一个黑点。

④ 确认号框(图 5-17):确认号框列出了可选项,用确认号表示选择结果。在确认号框中可以选择多个选项。

⑤ 文字输入框(图 5-17):是用户输入文本信息或数字的长条框。当光标移入输入框时,光标变成闪烁的竖条,等待用户输入。

⑥ 命令按钮(图 5-20):对话框中的命令按钮启动一个可以立即执行的程序,这个程序将按对话框中提供的信息执行。通常“关闭”或“Cancel”命令取消当前对话框的参数设置,关闭对话框;“认可”或“OK”命令接收对话框的参数设置,执行用户所选择的程序,并关闭对话框;后跟省略号(…)的命令将打开另一个对话框,以便提供更进一步的信息;后跟两个大于号(>>)的命令将放大当前对话框。

图 5-18　对话框中的下拉式列表框

(a) 列表框弹出之前；　(b) 列表框弹出之后

图 5-19　对话框中的任选项开关

二、构造技术

我们把作图看成是一个构造过程。这个过程实际上是选择作图命令和指定一系列定位点的过程。例如在选择画直线命令后，在作图区先后指定两个点就可以在这两个点之间联一条直线。有如下一些常用的构造技术。

1．橡皮筋技术

图 5-20　对话框中的命令按钮

橡皮筋技术常常用来构造直线段。按下定位器(如鼠标器)的键,得到一线段的起点位置。当光标移动时,线段的终点也在移动。在起点和光标点之间始终有一条联线,就像橡皮筋一样随光标移动。放开键,这条线段就确定下来了。橡皮筋通常是在键按住时出现,可以使操作者在作图过程中就能看到构造的线段,并不断调整其位置。图 5-21 显示了用橡皮筋技术构造直线段的各个过程。

图 5-21　用橡皮筋技术构造直线段的过程

(a) 按定位器键,得到橡皮筋的起点;　(b) 从起点到光标点作直线;

(c) 起点到光标点之间的连线就像橡皮筋一样随光标的移动而伸缩移动,直到放开键;

(d) 光标不再控制线段。

由橡皮筋画线技术可以扩展到橡皮筋矩形,橡皮筋圆、橡皮筋椭圆、橡皮筋折线和橡皮筋多边形等交互技术。

橡皮筋矩形的技术操作方法,是按下键后得到矩形的一个固定的角点,把光标点作为另一个对角点画矩形。按住键移动光标时,这个矩形随光标的移动而改变大小,直到放开键。橡皮筋圆的技术操作方法是按下键后得到圆心点,把光标到圆心的距离作为半径画圆。按住键移动光标时,这个圆随光标的移动而改变其大小,直到放开键。或者拉动一个

方框,作这个方框的内接圆。方框由两个对角点定义,其中一个是定点,一个是随光标移动的动点。橡皮筋椭圆的技术操作方法是用橡皮筋矩形技术拉出一个矩形,作这个矩形的内接椭圆。所有这些构造过程都是先按下定位器的键,再按住键移动光标,并同时观察作为反馈的橡皮筋,最后放开键。

橡皮筋作折线是构造线段的组合。在选择画折线的命令之后,先像画线段一样画第一条线段,再按住键作第二条、第三条。结束时可以按鼠标器的另一个键,也可以不移动光标连续按两下键,或者把光标移出作图区选择下一个作图命令。橡皮筋作多边形时与此类似。当光标定位在起始点时,表示多边形构造结束。有时当光标不在起始点时,给出明显的结束命令后,系统自动地,在起点和最后一个点之间联上线形成封闭多边形。

2. 约束技术

(1) 水平和垂直约束技术　　当需要绘制横平竖直的直线段时,运用水平和垂直约束技术可以避免由于人眼或定位设备带来的误差。此时定位点的坐标是约束函数的输入值。如果用水平约束构造直线,在线段的第一点被确定后,无论光标在屏幕上如何运动,线段的纵坐标始终与起点相同,显示的是一条(x_1, y_1)到(x_2, y_1)的直线。同样,如果用垂直约束构造直线,线段的横坐标始终与起点的相同,显示的是一条从(x_1, y_1)到(x_1, y_2)的竖线段。

(2) 引力场约束技术　　作图时,我们时常要从已有直线的某个端点或从该直线上的某点开始作一条新的直线。如果在定位时使用了网格技术,则很容易使点与点重合;否则的话是很费时的。引力场约束技术可以帮助用户把光标点精确地定位在某个端点或直线上。引力场是一种想象的约束范围。如图 5-22,是直线的引力场,一旦光标进入这个范围,它就被吸引到这条直线上。由于围绕端点的场域比较大,所以使光标与端点的重合特别容易。

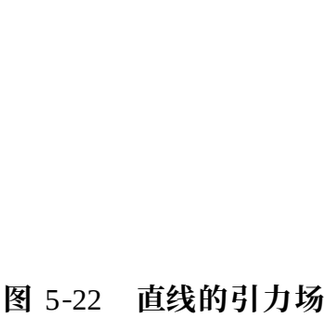

图 5-22　直线的引力场

三、动态操作

在作图过程中,除了利用上述的构造技术生成线段、矩形等物体外,还需要对这些已生成的物体进行动态操作。动态操作包括:物体的拖动、物体的旋转、物体的缩放,以及物体的形变。

1. 拖动

选择拖动物体的命令后,先在作图区用定位设备拾取某个要拖动的对象,再按住键移动光标,则这个被拾取的物体将随着光标的移动而移动,就好像光标在拖动物体一样。放开键后,物体就固定下来,再移动光标对这个物体就不起作用了。图 5-23 显示了拖动物体的过程。

2. 旋转

选择旋转命令后,先在作图区拾取要旋转的物体,这个物体将被着重显示。同时,在物体的中心将显示一个图标代表旋转中心。用光标先将这个图标拖到实际的旋转中心,再按

图 5-23　拖动物体

(a) 将光标移动到要拖动的物体处后按定位器键；　(b) 物体被着重显示，表示已被拾取；

(c) 物体随着光标移动；　(d) 放开键，物体被固定下来。

住定位器的键移动光标时，物体将围绕这个旋转中心随光标的移动而旋转。放开键则物体固定下来，恢复原先的正常显示。图 5-24 显示了旋转过程。

图 5-24　动态旋转过程

(a) 着重显示的被旋转体及旋转中心图标；　(b) 图标被拖到一个新的位置

(c) 物体随光标移动而旋转；　(d) 固定后的物体，图标消失。

3．缩放

动态缩放操作跟旋转操作大致相同。不同的是这里要确定的是缩放中心的位置，而不是旋转中心的位置。移动光标，物体将以这个缩放中心为基准进行缩小或放大。

4．形变

拖动和旋转改变物体的位置，缩放改变物体的大小，往往我们还要拖动物体上的某个点成某条边使物体产生形变。操作柄技术可用于物体的形变。操作柄在物体被拾取后显示出来，通常用小方块表示。

图 5-25a 显示了带八个操作柄的矩形，其中四个在矩形的角点，四个在边上。用光标拖动操作柄时，与操作柄相联的线就变成橡皮筋线，随光标的移动而被拉伸。例如，当拖动某条边上的操作柄时，与这条边相对的边固定不动，整个矩形随光标的移动而在一个方向伸缩(图 5-25b)。当拖动矩形某个角点上的操作柄时，这个角点的对角点固定不动，整个矩形变成橡皮筋矩形，随光标的移动而放大缩小(图 5-25c)。这种整体形变实际上是一种在两主轴方向上缩放比例不等的动态缩放。

局部动态形变操作可以如图 5-26 所示。在多边形各顶点上有一个操作柄，当拖动某

图 5-25　带操作柄的矩形(用于矩形的形变)

个操作柄到新的定位点时,相邻两条边变成橡皮筋,以相邻两端点为固定点随光标的移动而被拉伸。同样,如果在多边形的边上定义了操作柄,则在移动光标拖动这个操作柄时,这条边将以原来的斜率和长度被拖动,与这条边相邻的两条边将变成橡皮筋。

图 5-26　带操作柄的多边形(用于多边形的局部形变)

第三节　用户接口的设计

在介绍了图形交互技术的基础上,我们可以考虑如何设计一个用户接口了。这一节我们主要介绍用户接口的设计目标、设计步骤、用户接口的风格、用户接口的设计原则,以及设计过程中的各个具体环节。

一、用户接口的设计目标

在设计用户接口时,应始终瞄准以下几个重要目标:

1. 提高学习速度。学习速度是一个与用户熟练掌握图形交互系统的使用方法所花时间有关的量。这是一个相对量。只有 10 个交互命令的系统当然要比有 1000 个交互命令的系统容易学,但在为一个功能丰富的系统设计用户接口时,如果违背常理,用 10 个难学的命令去实现的话,可能也不会提高学习速度。

2. 提高使用速度。使用速度是一个与那些熟练的操作员用系统完成某个特定任务所需时间有关的量。学习速度与使用速度是一对矛盾。要同时满足这两个要求,最常用的方法是在用户接口系统中提供一个供初学者学习的基本命令集,最好用菜单实现,易于学习;而整个系统的所有命令,则可由键盘或功能键输入,以提高使用速度。

3. 降低操作失误率。操作失误率是指每次交互操作的平均失误次数。这一指标也影

响到学习速度和使用速度。

4. 增强记忆。交互系统应能帮助用户在长期不使用该系统后,一旦接触,就能尽快地回忆起各个交互操作。

5. 增强对潜在用户的吸引力。这是市场方面的一个目标。

二、用户接口的设计步骤

遵循软件设计的一般规律,用户接口的设计可分为以下 4 个步骤:

1. 概念设计。定义图形系统中必须由用户掌握的基本概念,包括对象定义、对象属性定义、对象关系定义和对象操作定义。例如,在简单的编辑软件中,操作对象是字符、行和文件;文件的属性是文件名;文件由一系列字符行组成,字符行由一系列字符组成;行的操作有插入、删除、移动和拷贝,文件的操作有生成、删除、换名、打印和拷贝。

概念的描述通常是借助于一些类似于用户已熟悉的东西,如打字机、绘图板、工作台、公文夹等。要注意的是,这种类比的方法对理解用户接口的概念虽然有帮助,但过分牵强附会,用这种方法表达图形用户接口的某些更高级的概念是有害的。

2. 功能设计。规定用户接口的详细功能。对于每个操作,说明需要什么参数、会出现什么错误,如何处理可能产生的错误,会产生什么结果、会给出什么反馈等。这里要注意的是,功能设计也称为语义设计,它只定义交互操作的含义,而不是交互操作的顺序或实施这些操作的设备。

3. 交互顺序设计(语法设计)。定义输入和输出的顺序。对于输入,是按一定的规则将各个词组组成一个完整的句子。词组可以看成是通过交互技术输入到系统中的不可分割的最小单元。例如,选菜单操作是一个词组。虽然这一操作可进一步划分成移动鼠标器和按键两个动作,但这两个动作都不能单独向系统提供有意义的信息。在这一节的第四部分输入模式和语法中,我们将具体介绍与输入模式有关的各种语法规则。

对于输出,是定义显示屏的 2D 和 3D 的布局,以及这种显示形式的临时变化。和输入的最小单元是不可分割的一样,输出的最小单元也是不可分割的,否则将失去意义。例如,在电路的辅助设计系统中,电阻符号是有意义的,而组成这个电阻符号的线条就没有什么意义。字符串可以向人们传递某种信息,同样,在图形系统中,符号和图形也能向人们传递某种有意义的信息。

4. 联结设计(词法设计)。定义如何由单词形成一个个有意义的词组。对于输入,单词可以看成是联结到系统上的各种输入设备,联结设计就是选择或设计前面我们介绍过的交互技术。对于输出,单词可以看成是基本图形软件包提供的各种几何形状(如直线、圆、字符)以及它们的属性(如色彩、字体)。联结设计就是把这些图素及其属性组合起来,形成图标和其它符号。

以一个简单的家具布置程序为例说明上述设计步骤。在概念设计阶段,确定操作对象是一间房子和要放到这间房子里的家具;对象之间的关系是房子要装这些家具;对家具的操作有生成、删除、移动、旋转、选择;对房子的操作有存和取。功能设计是对这些操作的详尽描述。

输入的顺序设计将是先选择操作对象,再选择对这个对象实施的操作。输入时的联结

设计将是用鼠标器从菜单中选择操作命令、选择家具、提供家具放置的位置。输出的顺序设计是定义显示屏幕的布局,将屏幕划分成几个不同的区域,确定菜单区、揭示区、出错信息显示区。输出时的联结设计包括字体、线宽、线色、填充域颜色的定义,以及由基本图素形成家具符号的方法。

三、用户接口的风格

在设计用户接口时,应保持以下的风格:

1. 所见即所得

这是交互式图形学的基础。屏幕上用户与之交互的内容,就是交互系统最终要生成的图象。大多数交互式图形系统都具有所见即所得这样的风格。

现在许多文字编辑软件也有所见即所得的接口。例如,在 MS- Windows 的编辑器 WRITE 中,要用黑体打印的文字在屏幕上显示时也是黑体。而在所见非所得的系统中,我们见到的是插在文字之间的控制符。例如文字串

In this sentence, we show @b(bold), @i(italic), and @ub(underlined bold)text.

的输出结果是:

In this sentence, we show bold, *italic*, and underlined bold text.

又比如,带输出控制符的数学表达式

@f(@i(u)@sub(max)- @i(u)@sub(min), @i(x)@sub(max)- @i(x)@sub(min))

会生成如下结果:

$$\frac{u_{max} - u_{min}}{x_{max} - x_{min}}$$

在这样的所见非所得的系统中,用户必须用脑子在控制符和结果之间来回切换,在输入过程中很难确定这些控制符是否会转换成真正需要的输出结果。

所见即所得也受到一些限制。比如,当显示器的分辨率、大小、色彩与硬拷贝设备不同时,很难保证完全匹配。更重要的是,一些系统不能实现完全的具有所见即所得风格的接口,例如,一般文字处理系统中,插在文字间的分页符输出时是没有的,但却在屏幕上显示出来了,这就违背了所见即所得的原则。

2. 直接操作

直接操作是指将可以操作的对象、对象属性、对象关系都在屏幕上显示出来,用鼠标器在它们上面进行某个动作,达到实行某种操作的目的。这样,无需用选菜单或键入命令这种传统的方式实行某个操作了。而是隐含在各种动作中。在屏幕上显示的可能是文字,如对象的名字、属性等,也可能是更为一般的图形或图象,如图标(icon)。

图 5-27 显示了 Macintosh 使用的一些直接操作。这里,磁盘和文件都表示成图标,从一个磁盘拖动文件的图标到另一个磁盘,就实现了文件的拷贝,拖动到垃圾筒就实现了文件的删除。

可以说,直接操作是图形用户接口的独特风格,易学易用。但对熟练用户来说却又显得比较慢。例如,打印文件 A,要先找到文件 A 的图标,定位选择,再拖到打字机图标上。

图 5-27　Macintosh 屏幕

右上是磁盘,它下面是目录,已被打开,显示在左上角的窗口内。窗口内的文件用带文件名的图标表示。光标正将一个文件拖到垃圾筒去。

找文件 A 可能还会用到卷滚条。如果用户知道文件名,用“ pr A ”命令就快多了。所以菜单命令、键盘命令和直接操作并存是当今用户接口的特点。

对话框也是一种直接操作。指点对话框中的某个文字域,就可以在这个域中输入文字或数据;也可以在列表中选择有限的域值。

3．图标

图标(ICON)是操作对象、对象属性、对象关系、动作、或某些概念的图形表示。用户接口的设计者可以选择文字表示某个概念,也可以用图标表示这个概念。图标和文字,哪个更好,很难说得清楚。但要注意的是,是否使用图标与直接操作没有任何关系,对文字也可以进行直接操作。

图标有以下特点:

① 用心设计的图标可以比文字更易于识别其含义。

② 占屏幕空间小。

③ 与语种无关,可以用于各个国家。

设计图标至少有以下三个目标,这三个目标的重要性随应用系统的不同而不同。

① 识别: 如何快速而准确地让用户认识图标,识别其真正的含义。

② 记忆: 一旦学习后,如何很好地记住图标的含义。

③ 辨别: 如何很好地区分不同图标。

用图标表示操作对象很容易。图 5-28 是从各种系统中收集的一组图标,表示办公用品。至于对象的属性,如果能给出一个合适的方式显示它们的值,那也是很容易用图标表示的。例如,调色板、线宽、线型、纹理图案、字体等对象属性,都可以用图表示出来。

动作,也就是命令,也可以用图标表示。有以下三种设计策略:

① 命令图标可以用现实生活中真正实行这个命令操作的物体来表示。如用剪子表示

图 5-28　表示办公用品的图标

图形的剪切、用打字机表示打印输出、用放大镜表示图形的放大缩小。图 5-29 给出了一些这样的图标。

这样表示的图标一般很难学习,因为用户先要认出这个图标是什么,再要理解这个图标表示的物体能做什么。这种两步理解的过程不是用户所希望的,尤其是在一个物体可能有多种动作时,就会产生混乱。

② 命令图标可以用命令执行前后的状态表示。图5-30中用方块表示几何变换前后的情况。这种表示方法在操作对象很简单时很有效。但如果操作对象很复杂、类型很多时,会让用户把应用面很广的命令误认为只能适用于图标上表示的物体。

图 5-29　用现实生活中实行操作的物体表示命令图标

③ 命令图标可以用更抽象的动作图形来表示。图 5-31 中显示了一些典型的例子。这些图标依赖于各种文化知识背景,如用八角形表示停止,用× 表示删除等。

4. 其它对话风格

还有一些既适合图形交互接口,又适合其它软件的对话风格。如菜单、命令语言、自然语言对话、问答式对话等。

① 菜单:广泛应用于图形系统和非图形系统。菜单的优点在于系统的命令都用菜单的形式显示出来了,用户见到的,实际上是他早已熟知的字符串或图标。选择菜单就可以执行命令,而不用去回想一个个命令,再键入计算机。菜单减少了用户的记忆,因此对新手特别有吸引力。

② 命令语言:是一种与计算机实现交互作用的传统方法。这一技术可以容纳大量的命令,而不像菜单那样受被选集元素个数的限制。易于扩充命令,可快速输入命令。学习时间是命令语言的主要问题,需要键盘输入技能是使用命令语言的不利因素,输入的失误率较高也是命令语言容易产生的问题。

③ 自然语言理解:这是作为交互系统的最高目标提出来的。如果计算机能理解我们用日常语言键入的或说出的命令,那么人人都可以用计算机了。然而,目前具有大词汇量

图 5-30　用命令执行前后的状态表示命令图标　　图 5-31　用抽象的动作图形表示命令图标

的语音识别系统必须经过专门训练才能识别特定用户的声音,还会产生失误。键入长句子是一件烦人的事。一般说来,自然语言理解系统不限制命令集的大小,所以如果用户企图要系统做一些不可实现的事,就会降低系统的性能,使用户丧失对系统的信心。这个问题可以通过限制自然语言理解系统的知识域来解决,使用户了解系统的能力,不发出过分的请求。

交互式图形系统的自然语言语音输入方式如果与其它对话方式结合,将会大大地提高交互效率。使手和声音同时或交替工作,就更像真实世界中我们的工作方式:先用手指着某个东西,再谈论它。用户要移动某个物体,可以在指点它的同时说"移到…",再指点另一个地方说"哪儿"。

④ 问答对话:是计算机入门的一种交互方式。计算机在显示器上显示出问题,等待用户回答。用户可用键盘输入答案。通常答案是受限制的,而且就显示在屏幕上,可由用户挑选。这种对话方式的缺点在于不能返回到前面的问题,重新回答,而且在用户回答当前问题时,不知道以后还会出现什么问题。

四、用户接口的设计原则

在设计用户接口时,应遵循以下设计原则:

1. 保持一致性

保持交互系统的一致性,是指在设计系统的各个环节时,应遵从统一的、简单的规则,保证不出现例外和特殊情况。保持系统一致性的基本目的是能让用户由系统某个方面的知识推广到其它方面。当系统不按照某个可以理解的、合乎逻辑的方式进行时,一致性可以使用户免于受挫。实现一致性的最好方法是自顶向下仔细地设计整个系统。

用户接口输入部分的一致性例子有:

① 键盘上的键(如回车、制表、换行、删除),总是保持某一个功能不变,无论在何种情况下键入都有效。

② 全程命令(如 HELP、STATUS、CANCEL)随时可以发出。

③ 常用命令(如 MOVE、COPY、DELETE)可以对系统中的任何类型的操作对象起作用,而且其执行结果是可预料的。

输出部分的一致性例子有:

① 使用同样的符号、同样的颜色表示同一类信息,如总是用红色表示停止,用绿色表示通行。

② 系统状态信息总是在一个合乎逻辑的固定地方显示。

③ 在一个菜单区中菜单项总是显示在同一个相对位置上。

要注意的是,一致性会与系统的其它设计原则相冲突。例如,把一个代表文件的图标拖到垃圾图标上,就执行了删除文件的命令。那么如果把这个文件的图标拖到打字机图标上,会发生什么呢? 是否在执行了打印命令后,为保持与拖到垃圾筒的功能一致而执行删除命令呢? 显然是不能删除的。比保持一致性更高更一般的设计原则是:按用户认为最正常、最合乎逻辑的方式去做,这比保持单纯的一致性更重要。

2. 提供反馈

反馈是人-机交互的一部分,设计用户接口时必须考虑到各种反馈,并在程序中实现。

根据设计过程中的功能设计、顺序设计(语法)和联结设计(词法)的三个步骤,可以给出三级反馈。设计人员必须有意识地考虑到每一级,并明确地决定是否提供反馈,以何种形式给出反馈。

① 最低级的反馈(对应于联结设计)。用户在交互设备上的每个动作都应立即产生明显的反馈。如在键盘上键入字符时应在屏幕上有回显,鼠标器的移动应对应于光标的运动。

② 二级反馈(对应于顺序设计)。当系统接受输入语言(命令、位置、操作对象等)中的每个词组时,应提供反馈,被拾取的物体或被选中的菜单项要着重显示,使用户知道他的动作已被接受。在整个句子输入完毕且被断定是正确时,也要给出反馈,但这种反馈通常是在执行这个命令需要一两秒以上的时间才给出。

③ 功能级反馈(对应于功能设计)。这是最有用且最受用户欢迎的一种反馈形式。它告诉用户,他所发出的命令已执行完了。通常是将操作结果显示出来。在执行某个费时的操作过程中,要给出某种反馈,表明计算机仍在执行中。这样的反馈有多种形式,最能引起用户注意的是,用一个标度尺指示命令完成的百分比,使用户知道何时能完成这个命令的操作。

有必要区分对象域的反馈和控制域的反馈。对象域的反馈向用户表明正在被操作的对象在作图区的哪个位置。有三种表示对象域反馈的方法:

① 把屏幕划分成如图 5-32 所示的三个区域,右上角的小区域显示全图,用一个小方块指示作图区在这个全图中的位置。在双屏系统或窗口管理系统中,可以用一个屏幕或窗口显示全图,用另一个显示局部,这样会更加有效。拖动小方块或改变小方块的大小,就能达到平移和缩放的效果。

图 5-32　用划分屏幕的方法给出对象域的反馈

② 图 5-33 是窗口系统中的卷滚条。卷滚条中可移动的小方块指示窗口中当前显示的内容在整个图形中的位置。

图 5-33　用卷滚条上的小方块给出对象域的反馈

③ 在同一个屏幕上交替显示全图和局部放大图(图 5-34)。显示全图时,可以拉出一个矩形框,选择 OK 后,将详细地显示这个矩形框中的图形。

图 5-34　用交替显示全图和局部图的方法给出对象域的反馈。现在显示的是全图

控制域的反馈向用户显示控制这个交互系统的一些状态、当前值、默认值、菜单、对话框等。最重要的是当前设定值,它总是在反馈域中被显示出来。图 5-35 显示了菜单被选之后,用< 指示当前被选项。

图 5-35　用< 指示当前被选项

图 5-36　SunView 在事件发生地显示反馈信息

反馈的位置很重要,通常是在屏幕上指定一个固定的区域显示反馈信息。但这破坏了视觉上的连续性,因为用户的眼睛必须在绘图区和信息反馈区来回移动。经验告诉我们,用户往往不注意固定区域的信息。采用声音反馈可以避免这个问题。

把反馈信息显示在用户注意力集中的地方,如光标所在的位置,是最好的。现在很多窗口系统,都能在事件发生的地方弹出一个反馈窗口,显示警告信息、确认信息、出错信息等。图 5-36 是 SunView 窗口系统显示反馈信息的方法。

3. 尽量减少失误的可能性

要尽可能地减少用户的操作失误,为此,要注意两点。一是使系统只提供当前有效的菜单命令,引导用户只在有效的范围内工作,不让用户做不能做或不允许做的事。例如:

· 不提供会产生“非法选择”、“无效命令”等反馈信息的菜单项;

· 在没有东西可删除时,不让用户有机会选择“删除”命令;

· 如果操作对象不是一个字符串,就不能让用户对这个操作对象改变字体;

· 如果剪切板是空的,不要让用户做“粘贴”操作;

· 如果没有选择任何东西,就不要让用户做“拷贝”操作;

· 在当前被选的物体不是曲线时,就不要让用户选择曲线光顺的命令。

在上述这些情况下,系统就应该使这些不可用的命令项隐藏起来,如用灰色代替黑色,告诉用户这些菜单项目前不可用(图 5-37)。

图 5-37　不可选的菜单项用灰色表示

再就是尽量避免副作用,不要产生用户不期望的结果。典型的副作用是打印命令删掉了被打印的文件。设计时考虑不全,与用户交流不充分往往会产生副作用。

4. 提供出错恢复

谁都有出错的时候。试想象一下,键盘上如果没有 BACKSPACE 键将会如何呢? 恢复功能强,用户就不会因为害怕出错而缩手缩脚,处处小心。

有四种出错时的恢复方法:复原(UNDO)、中止(ABORT)、取消(CANCEL)和校正(CORRECT)。

① 复原

最严重的错误是出现在功能上。用户错误地执行了一个或多个命令,得到了一个预料之外的结果,这时就需要“复原”到命令执行前的状态。

有两种复原方式:单级复原和多级复原。单级复原只恢复到上一个命令执行前的状态。“复原”本身也是一个命令。所以如果紧接着“复原”命令之后再执行一次“复原”命令的话,就等于没有做恢复工作。

多级复原是在前面执行过的命令栈上进行操作。执行过的命令都存入栈中,根据栈里存放的命令逐一做复原工作。有时限于栈的大小,一个工作段开始时,就将栈清空。对应于“复原”命令,还有一个“重做(Redo)”命令,把复原以后的状态通过重做再恢复过来,以防止用户复原过多。一般是对当前发出的复原命令可以执行重做命令。“复原”和“重做”命令都不存放到命令栈中。

有两种形式的复原。一种是在某个命令执行完、执行结果显示在屏幕上之后,向用户提问是接受还是拒绝,用户在发出一条命令之前,必须做出选择。这种复原形式叫做“明确接受,明确拒绝”,通常加在一些危险的操作后面,强迫用户在确定接受之前再想一下。另一种复原形式可能更为可取,因为在拒绝一个命令的执行结果时才需要采取行动,发出下一条命令就隐含着接受了上一条命令的执行结果。这种形式叫做“隐含接受,明确拒绝”。

不管怎样提供“复原”功能,都需要额外的编程,尤其是对那些要改动数据结构的命令,“复原”工作更难。一种容易的、但不太令人满意的方法是:并不提供“复原”功能,只是在命令执行前让用户确认是否真的要执行。删除文件时常常这样。

② 中止

在命令执行过程中,用户就有可能意识到出现了失误。这时就需要“中止”命令,以便中途打断正在执行的命令。像“复原”一样,“中止”命令也需要把系统恢复到命令执行前的状态,在这个意义上,可以把“复原”和“中止”看成是相同的命令。

③ 取消

在用户为执行某个命令而输入所需要的参数时,会突然发现自己并不想执行这个命令。这时需要“取消”这个命令,回到输入前的状态。窗口系统的对话框、填充板中都有“取消”命令,以保证用户在输入过程中发现错误时,可以恢复原状。“取消”命令也可以看成是一种特殊的“复原”命令,因为它也使系统回复到当前命令的最初状态。

④ 校正

在用户输入参数时,往往会出现失误,这时用户总是希望能简单地把错误校正过来。不同的对话形式决定了校正工作的难易。用对话框或填充板时,最容易修改自己的输入和选择。命令语言输入也可以校正,但必须用 BACKSPACE 键删除到出错的地方,再重新输入。用问答对话方式和菜单方式输入时,就不那么容易校正了。

5. 面向多层次用户

交互式图形系统要面向各种层次的用户,使毫无经验的新手、经验不多的用户,直到熟练用户,都能找到适合自己的交互手段。

使系统能容纳多层次用户的方法是,提供加速键,增加提示信息,提供帮助信息,可扩展功能,及隐藏复杂功能等。

① 加速键

新手通常对菜单、对话框、和其它能提供提示信息的对话形式感到很舒服,因为提示信息会告诉他们该做什么和该学什么。但熟练用户注重的却是操作速度。这就需要用功能键或键盘命令,即用加速键代替较慢的交互手段。

加速技术有:

(i) 用单字母命令代替用鼠标器选择菜单。

(ii) 对菜单项编号从键盘上直接键入菜单项的号码。

(iii) 连击鼠标器上的键。这是最快的一种方法,如按一下键表示选择文件,按两下键则表示打开文件。

(iv) 命令行输入。随着用户操作水平的提高。他们会越来越多地用命令行一次输入一串命令及其参数。

② 提示

提示是建议用户下一步做什么,这与反馈有所不同。熟练用户不希望有太多的提示信息,因为提示信息过多会降低系统的速度、占屏幕空间、甚至烦人。很多交互系统提供多级提示方式,由用户控制提示信息量。

提示有多种形式:

(i) 最直接的是显示信息,明确地向用户解释下一步该做什么,如"输入 X 坐标"。语音合成器可以用来向用户发出口头指令。

(ii) 更巧妙的提示是在功能键盒上,盒上可选择的功能键都被点亮,而不可选择的功能键则变暗。

(iii) 在必须输入一个定位点时,突出显示光标或定位叉丝;在需要输入一个数据时,显示标尺或度盘;在需要输入字符串时,显示带下划线的闪烁光标。

(iv) 对于具有直接操作风格的图形接口,可以操作的图标就是一种很好的提示。

③ 帮助

帮助信息可以让用户对系统的基本概念、典型任务、各种命令使用方法、交互技术等有更好的了解。最理想的是,帮助信息能在交互过程中按某一固定方式随时被调出来显示,且能容易地返回到调出帮助信息时的命令执行状态。帮助信息应与当时的操作状态相对应。如果在系统等待命令输入时调用帮助信息,则列出系统中当前能输入的所有命令。如果在一个命令键入后调用帮助信息,则显示有关这条命令的详细信息。

一个简单的调用帮助信息的办法是在屏幕上指点希望取得帮助信息的对象,再按 HELP 键。这个对象可能是菜单项、状态指示器、窗口标题栏、卷滚条、或已生成的物体。

即使有提示和菜单,提供帮助信息还是必要的,因为它为用户提供了一个更多更详细地了解系统的机会。就是特别熟练的用户,在系统庞大而复杂时也会忘记某些细节。

④ 可扩展

使用户接口具有扩展性,是指允许用户在原有命令集中增加新的命令,以扩充用户接口的功能。宏定义是一种扩充功能的方法。在宏定义中,用户的操作过程保存在一个跟踪文件中,可以调出来按顺序执行。用户用编辑命令可以建立这种宏文件。

⑤ 隐藏复杂功能

把复杂功能隐藏起来不让使用,是新手们学习系统基本功能的很好方法。这样,不用指定选项,不用学习不常用的特殊命令、不用经历复杂的启动过程就可以开始实际的工作。

有些命令常常会有很多可选参数项。对于这种命令,就需要简化,提供合理的默认值代替任选参数项。简化后的命令可以做成一个小的"初学者工具箱"。例如在画圆形百分图的程序中,用户指定几个必要的参数后就可以按默认值画出百分图。但如果用户对这个百分图不满意,那他可以修改圆饼的半径、每个扇形的颜色或纹理,可以加注释、改变字体和文字大小,可以改变数字显示位置等。这些在开始画百分图时不应强迫用户去做。

隐藏复杂命令的另一个策略是:复杂的、高级的命令只有通过键盘或功能键才能执行。这样可保持菜单更小、系统更简单。

6. 尽量减少要记忆的内容

要尽量减少用户不必要的记忆。如在绘图系统中,用数字代号表示物体,而不要用名字。拾取物体或图标就不需要记忆。重要的是尽可能地唤醒用户的识别而不是记忆。

五、输入模式和语法

在图形系统中,模式就是一种状态,在这种状态下,用户可以完成交互操作中的某些任务。例如:

① 命令输入模式:在命令输入模式下,只有那些对当前被选择的对象可以执行的命令才允许执行。

② 对话框模式:在对话框模式下,只允许填写对话框或执行对话框中的命令。

输入模式分有害模式和有用模式两种:

要防止系统进入有害模式,因为当系统进入有害模式后,运行时间会很长,甚至出不来。有害模式没有什么用,与任何操作对象无关,用户不知道现在是处在何种状态,不知道如何转到自己希望的模式去。有害模式只能把用户搞糊涂。

有用模式为用户提供的是简单明了的命令集或菜单,提供提示或帮助信息,可以减少用户的记忆。命令或菜单按某种方式有机地组织起来,让用户用起来得心应手。有用模式会明确地显示当前的状态,提供反馈信息,显示当前哪些命令可用,有明显的、简单的、快速的方法从一种模式退出来进入另一种模式。窗口系统提供了视觉效果非常好的有用模式,因为每个窗口代表一种模式,激活窗口的操作就能从一种模式切换到另一种模式。

对用户接口的输入模式产生主要影响的是命令语言的语法。下面介绍三种语法及其相应的输入模式。

1. 前缀语法。

以下形式的语法为前缀语法:

命令,参数 1,参数 2,…,参数 n。

在前缀语法中,一旦指定了命令名,用户就被封锁在这种命令状态,按指定的顺序输入各个参数,执行命令。这时的错误校正机制非常重要,因为否则的话,选择命令的用户就不得不进行可能会是很长的参数输入。

在前缀语法中,可以把参数从命令中分离出来,先发出不带参数的命令,再输入任意组参数,达到一个命令多次执行的目的。例如删除物体的命令:

```
Delete- object          (建立命令模式)
object 1
object 2
   ¦
object n
其它命令                (建立另一个命令模式)
```

Delete- object 建立了一个删除命令模式,在这种模式下,被选择的物体都将被删除,直到建立另一个命令模式。

2. 后缀语法

把前缀语法中的参数提出来,用设置参数的命令在选择执行命令之前,预先设置参数,就构成了后缀语法。如前缀语法的命令:

```
Draw- line, point, point, line- style, line- thickness
```

可以重构为:

```
Set- attributes attribute- values    (设置属性值,此命令只在当前属性值不合
                                      适时执行)
Draw- line point, point
```

或者更进一步:

```
Set- attributes   attribute- values   (设置属性)
Select- point     point               (选择起点)
Select- point     point               (选择终点)
Draw- line                            (画线)
```

后缀语法在用定位设备交互时特别方便,因为人们习惯于先指定操作对象,再确定要做什么。

在后缀语法中,当前被选对象这个概念非常重要,命令总是施加在当前被选对象上,而且对同一个当前被选对象,可以执行多个命令。

完全不带参数的命令并不是最理想的。如在上述的画线命令中,先指定两点,再告诉系统用直线联起来,就失去了橡皮筋交互画线的能力。

3. 混合语法

在混合语法中,无论用户是先发命令再指定操作对象,还是先指定操作对象,再发命令,都能执行命令。如:

```
Set- attributes   attribute- values   (设置属性)
Select            object1             (把属性值赋给物体 1)
Set- attributes   attribute- values   (设置属性)
```

Select	object2	(把属性值赋给物体 2)
Select	object3	(选择物体 3)
Set- attributes	attribute- values	(把属性值赋给物体 3)

在上例混合语法中,命令模式和当前被选对象模式是交替的。不能全假设为当前被选对象模式,否则的话,第二次 Set- attributes 命令会对 object1 起作用。也不能全假设为命令模式,否则的话,在第三次 Set- attributes 命令之后,还要用户选择物体对其赋值,而不是把属性值赋给 object3 了。

在混合语法中,增加 Do- it(执行)命令,会使命令模式和当前被选对象模式之间的切换显得更加清楚。

Select- object	object	(建立当前被选对象)
Set- attributes	attribute- values	(建立命令模式)
Do- it		(执行设置属性命令)
Copy		(建立命令模式)
Select- object	object	(建立当前被选对象)
Do- it		(执行拷贝命令)
Do- it		(再次执行拷贝命令)

能使语法随模式的改变而改变是最理想的。例如,在没有当前被选对象时发命令就是前缀语法,而在有当前被选对象时发命令就是后缀语法。如:

Set- attributes	attribute- values	(在此之前没有当前被选对象,则保留属性,等待选择对象)
Select- object	object	(选择对象后,设置属性值)
Select- object	object	(在此之前没有命令,则建立当前被选对象,等待命令)
Set- attributes	attribute- values	(发命令后设置属性值)。

在这种随模式改变而改变的语法分析中,必须向用户反馈当前的模式,让用户清楚地了解执行命令后会产生什么结果。有些用户可能会被这种交互方式搞糊涂,因为这毕竟是没有遵守一致性的原则。

六、视觉效果设计

用户接口视觉效果设计得好坏,直接影响用户对这个接口的最初印象和长期使用。视觉效果设计涉及用户接口的各个图形元素;如整个屏幕的布局、菜单和参数板的设计、颜色的使用、信息编码和信息布置。好的视觉效果应具有简单明快、一致、有吸引力等特点。

1. 简单明快

达到简单明快这一效果的最重要途径是,对显示信息在视觉方面进行组织和安排,以加强和强调它们的潜在逻辑关系。有以下一些视觉规则,可以用来组织和安排显示信息:

① 相似性: 具有相同属性的两个视觉刺激应归属于同一类。

② 相近性: 距离相近的两个视觉刺激应归属于同一类。

③ 封闭性: 对一组几乎形成一个闭域或可以看成是包围了某个域的视觉刺激,看成是一个域。

④ 连续性：观察一些彼此相交的曲线，看到的是一根根光滑连续的曲线。

图 5-38、图 5-39、图 5-40 和图 5-41 分别显示了如何利用这些视觉规则加强被显示信息的内在逻辑关系。要记住的是，使用这些视觉规则的目的是通过强调这种内在的逻辑关系，达到简单明了的视觉效果。另一个目的是：减少用户视线的改变，减少用户手的运动。这些目标可能是矛盾的，这就要求设计者们寻求最佳方案。

图 5-38　视觉规则

(a) 均一分布的方阵；　(b) 由相近性可以看出水平组；　(c) 由相近性可以看出竖直组；

(d) 由相似性可以看出水平组；　(e) 相近性和相似性的组合，更加强了水平组；

(f) 由封闭性可以看出一个方块点阵，虽然少了两个点。

图 5-39　视觉规则

(a) 两条相交的线；　(b)、(c) 两种可能的解释，由连续性可以看出是(b)；　(d) 两个重叠的窗口；

(e)、(f) 两种可能的解释，由连续性可以看出是(e)。

2．视觉编码

在用户接口设计中，编码是指使不同显示对象产生视觉差异。计算机图形学中广泛使用的编码技术有：颜色、形状、大小或长度、字样、方向、亮度、纹理、线宽、线型等。

对一个特定的编码技术，确定其编码等级是编码技术的基本点。码值越多，用户越容易对相邻的编码产生混淆。据统计，最多用 10 种颜色、15 种几何形状、6 种区域大小、6 种

图 5-40　视觉规则

(a) 没有进行视觉效果设计的参数板布局;

(b) 用封闭规则进行分组,各部分的逻辑关系就明显了

图 5-41　菜单组织

(a) 看上去无结构、无逻辑关系;　(b) 按相近性分组;

(c) 在相近性分组的基础上,相似性显示了二级逻辑关系。

长度、24 种角度的编码值时,用户识别显示信息的出错率会降低到 5%。对每个编码加注释,说明其含义,可以降低出错率。

在特别重要的场合,往往采用多重编码的方法,即用两个或两个以上的编码方法表示同一个显示对象。如图 5-42,是三重编码:用线宽、几何形状、内域的填充图案来区分显示对象。在这个例子中,任何一种编码都足以区分这三个显示对象。颜色编码通常与其它编码技术一同使用,这样可以照顾有色盲的用户。

除了要确定编码等级,还要了解被编码的显示信息是有名信息、有序信息,还是可度量信息。有名信息是指那些可以用指名方式来区分的信息,它们之间没有大小的概念。有

图 5-42　三重视觉编码:线宽、几何形状、填充图案

序信息是指那些有一定顺序、有大于或小于关系的信息,它们之间没有距离的概念。可度量信息是指那些如温度、高度、重量、数量等可以度量的信息。

① 有名信息的编码: 颜色、亮度、形状、大小、线型、字样、断面线图案等编码技术都适合于有名信息。对于数量一定的有名信息,如几何造型系统中的基本体素,正方体、圆柱、圆锥、圆台、环、正多面体,用颜色编码是最易区别的。对于有名信息,编码是无序的,用户不能从中推断出哪个重要,哪个不重要。用字样编码时,要注意在同一画面上不能用太多的字样。一般用不多于两三种字样时,不至于产生混乱的画面。

② 有序信息的编码: 线型、密度可变的填充图案、字体的大小等编码技术适合于有序信息。这些编码不必连续改变,但至少必须有一个显而易见的顺序。

③ 可度量信息的编码:可度量信息必须由一种可连续变化的编码技术来显示。标度尺上的定位点、长短、两线夹角或线的倾角、面积大小等编码技术适合于可度量信息。

与视觉编码很接近的是如何把用户的注意力吸引到某个特殊的显示对象上。如吸引到出错或警告信息上、吸引到当前被拾取的物体上,或吸引到当前选择的命令上。吸引注意力的方法有:用特殊的颜色或形状、闪烁、反转显示、光标转动等。改变颜色比改变形状、大小、灰度更能吸引用户的注意力。

3. 布局原则

三个基本布局原则是保持平衡、网格化、成比例。图 5-43 显示了两种不同的布局方法,图(a)具有均衡性,绘图区在屏幕中央,而图(b)就左轻右重,失去平衡。而且(b)在右上角也设计得不规范,卷滚条上箭头的底线与菜单项的分格线没有错开,会产生错觉。

图 5-43　屏幕布局

(a) 保持平衡;　(b) 过分强调右侧。

图 5-44 显示了如何利用各区域之间的空间以及网格化原则。在(b)和(c)中,三个区

域的边界都约束到网格上，看上去很干净、整齐，而(d)就没有这种感觉。

图 5-44　四种布局设计

(a) 原始设计；　(b) 加框；　(c) 强调边框以区分三个域；

(d) 边框线没有调整到网格上。

成比例原则是使矩形区域两边的长度成一个合适的比例，这样会更美观。常用的边长比有 1∶ 1, 1∶ 2, 平方根比 1∶ 1.414, 黄金分割比 1∶ 1.618。

第四节　用户接口软件

前面我们重点讨论了用户接口中的各种交互技术，以及用户接口的设计方法。这里我们将介绍各种用户接口软件。图 5-45 显示了不同层次上的用户接口软件，以及它们之间

图 5-45　用户接口软件

的调用关系。其中,应用程序可以调用各层的接口软件:操作系统、窗口管理系统和图形软件包、图形工具箱、用户接口管理系统;交互设计工具是专门为非编程人员设计窗口、菜单、对话框和对话顺序而提供的。

一、基本输入控制方式

图形软件包一般向用户提供三种基本的输入控制方式:请求方式、采样方式和事件方式。可以用如下命令设置一个输入设备的输入控制方式:

```
setInputMode(device, mode)
```

这里,device 为逻辑输入设备,mode 为三种输入控制方式之一。

有六种逻辑输入设备,以下是六种逻辑输入设备名,以及由其输入的内容:

逻辑输入设备名	由逻辑设备输入的内容
定位器	用户坐标系中一个点的坐标
拾取器	图形段名
选择器	指示被选对象的整数
数值输入器	实数值
字符输入器	字符串
连续定位器	用户坐标系中点坐标系列

1. 请求输入方式

请求输入方式按以下步骤工作:

① 应用程序向输入设备发出输入请求命令,如希望从定位器中获得一个定位坐标时,可用如下请求命令:

```
requestLocator (device,  &x, &y)
```

② 等待用户输入数据。

③ 用户开始输入操作,如移动鼠标器到定位点,再按一下键。

④ 在用户输入操作完成后,应用程序得到输入数据,如定位点的坐标(x, y)。

这里需要注意的是,即使设备被置成了请求输入方式,但在应用程序未发出输入请求之前,用户是不能进行输入操作的。否则的话,输入的数据将不被应用程序接收而丢失。所以,一般在应用程序中,在请求输入命令发出之前先显示一条提示信息,提示用户开始输入操作。

请求方式一次只能对一个输入设备进行,而且只支持那些由 RS-232 端口联到计算机上的老式图形终端或其它交互设备。

2. 采样输入方式

一个输入设备一旦被设置成了采样输入方式,用户就可以立即向这个设备进行输入。应用程序按一定的周期反复测试输入设备,读取数据。采样方式允许用户同时使用多个设备。以下是采样例程序:

```
endStatus= FALSE;
    do {
        status= FALSE;
```

```
        samplePick(device, &status, &segmentName);
            /* status= TRUE 说明拾取成功,
               segmentName 是被拾取对象的名字 * /
        if (status= = TRUE)  执行拾取对象处理子程序;
        status= FALSE;
        sampleString (device, &status, string);
            /* status= TRUE 说明字符串输入完了,
               string 是输入的字符串 * /
        if (status= = TRUE)  执行字符串处理子程序;
    }while (endStatus= = FALSE);  /* endStatus 会在拾取对象处理子程序或字符串处理子
                                     程序中被设置成 TRUE* /
```

上述这种采样过程是很危险的,因为在对第一个被拾取的对象进行处理的过程中,用户可能又做了另外几次输入操作,这些数据由于应用程序没来得及接收而丢失了。在采样方式中,用户的输入顺序无法保持,上例中,用户可以先拾取图形,再输入字符串,也可以先输入字符串,再拾取图形。

由于应用程序不停地测试输入设备,随时读入用户输入的数据,所以应用程序可以对用户的动作给出实时反馈,这是采样方式的优点。

3. 事件输入方式

在事件输入方式中,输入设备与应用程序是异步工作的。应用程序先将输入设备置成事件方式,并允许由设备输入。一旦设备是可输入的,不管应用程序是否准备好接收数据,用户都可以在这个输入设备上进行输入操作了。任何一个输入操作的数据都将作为事件按其发生的次序被送到一个输入队列中。应用程序随时可以检查这个输入队列,处理队列中的事件,或删除队列中的事件。事件输入方式避免了请求方式和采样方式中存在输入数据丢失的问题。

检查事件队列,可用如下函数:

```
waitEvent(timeout, &deviceClass, &device)
```

这里,timeout 是最长等待事件的时间。当事件队列为空时,应用程序将被挂起,直到最长等待时间已过或有一个事件进入事件队列。这时,若是因为最长等待时间已过而返回,则参数 deviceClass(设备类型)为空。否则的话,将刚进入事件队列中的那个事件传送到当前事件记录中去,并将这个事件发生的设备类型和设备名分别作为参数 deviceClass 和 device 的值,返回给应用程序。

当队列不为空时,无需挂起应用程序,而是直接将队列中排在第一的事件传送到当前事件记录中去,并返回这个事件的 deviceClass 和 device 的值。

参数 timeout 可以置成 0,表明在发现队列为空时将不等待,立即返回。

用下述命令可以从当前事件记录中取得输入的数据:

```
getEvent(&event)
```

参数 event 中存放返回的输入数据。event 为一事件记录类型,记录事件发生的设备类型、设备名、发生时间、事件类型、用户输入的数据等。

从当前事件记录中取得数据,还可以针对不同的设备用不同的命令。例如,对于拾取设备,可用命令

getPick(&segmentName),

而对于文字输入设备,可用命令

getString(&string)。

可以把采样方式中的例程序改写成事件方式的例程序:

```
endStatus= FALSE;
do {
    waitEvent(timeout, &deviceClass, &device);
    switch(deviceClass) {
        case  PICKDEVICE:
            getPick(&segmentName);
            处理拾取对象的子程序;
            break;
        case STRINGDEVICE:
            getString(&string);
            处理字符串的子程序;
            break;
        default:
        break;
    }
  }while (endStatus= = FALSE);
```

与请求方式不同的是,事件方式下的输入是异步的,即在用户输入数据的同时,应用程序仍可以不停地执行。这种方式又叫做超前输入方式,即用户可以在应用程序需要数据之前开始输入。这种超前输入的能力,可以提高交互输入的速度。

二、窗口管理系统

窗口管理系统具有现代用户接口许多重要特征:屏幕被划分成多个可重叠的矩形区域,即窗口;每个窗口可以对应一个进程;可以在任何一个窗口中运行应用程序,应用程序可以把结果显示在任何一个窗口中;窗口的大小可以改变;可以移动窗口;屏幕上可以有弹出式或拉出式菜单,可以有对话框,可以有各种图标。图 5-46 是 MS Windows 启动后的窗口界面。

和操作系统是一个资源管理器一样,窗口管理系统也是一个资源管理器,只是资源的类型不同而已。窗口管理系统管理的资源主要是两类:一类是用来显示输出的屏幕,另一类则是用来输入的交互设备。窗口管理系统将屏幕资源合理地分配给那些要使用屏幕的应用程序,使得这些程序之间的显示互不干扰;将输入设备分配给那些要交互输入的应用程序,并把由交互输入设备得到的数据分送到相关程序的事件队列中去。

窗口管理系统由两个主要部分组成:

① 窗口管理器。用户可以使用窗口管理器在屏幕上交互地生成或删除窗口、打开或关闭窗口、移动或缩放窗口、弹出或隐藏窗口。

② 窗口系统。它是一个基本的功能部件,真正地实现窗口的生成、删除、打开、关闭、移动、缩放、弹出、隐藏等功能。

窗口管理器是建立在窗口系统之上的,它与窗口系统的关系犹如命令解释器与操作

图 5-46　MS Windwos 窗口界面

系统内核的关系。建立在窗口系统之上的,除窗口管理器外,还有更高级的图形软件包和用户开发的各种应用程序。我们通常把窗口系统称作服务程序(server),而把建立在窗口系统之上的各种应用程序,包括窗口管理器本身,都称作客户程序(client)。图 5-47 表现了窗口系统与应用程序之间的关系。

图 5-47　窗口系统与应用程序的关系

在同一个窗口系统上,可以建立多个窗口管理器,每个窗口管理器可以有其独特的显示风格、外观和窗口操作方法。比如在 X- Window 窗口系统上,有 Motif、Xsight、

OpenDeskTop 等窗口管理器。

一般说来,窗口管理器和窗口系统在同一台计算机上运行。但如果对窗口系统的编程接口定义得很清楚,且是通过内部进程通讯功能实现的话,那么窗口管理器和窗口系统也可分别驻留在两台用高速网联结起来的计算机上运行,从而实现客户-服务器式的窗口管理系统。这种运行方式的直接好处在于,当应用程序是一个大型程序,需要超级计算机运行时,它就可以驻留在超级计算机上运行,而作为它的服务程序——窗口系统则运行在用高速网与这台超级计算机相联的工作站上,用户在工作站上与应用程序交互。

在设计客户-服务器式窗口管理系统时,为了减小通讯延迟,必须认真地考虑窗口管理器与窗口系统之间的接口,尽可能做到如下两点:

① 在客户程序和服务程序之间用异步通讯,而不要用同步通讯。这样,在客户程序向服务程序发送一个消息时,客户程序不用等待服务程序回答是否接收到消息,就可以继续向下运行,也可以再发送下一个消息。

② 把消息集中起来传送,以减少传送的次数。大多数网络通讯协议中对传送块的大小都有一个下限,通常是 16 到 32 个字节。一般地,一次传送一个大的数据块所花的时间,要比把这个数据块分成多个小块多次传送所花的时间少。

下面,我们着重介绍一下窗口系统的输出处理和输入处理。

1. 窗口系统的输出处理

屏幕是窗口系统分配给客户程序的输出资源。窗口系统必须管理这个屏幕资源,保证各个客户程序之间的输出显示互不干扰。对于不同的窗口系统,其资源的分配策略也大不相同,主要的不同点在于如何显示一个窗口中由于放大、弹出、卷滚操作而重新显露出来的区域。图 5-48 中,阴影区域是在放大、弹出、或卷滚操作后重新显露出来的区域,这个区域的内容需要重新显示。

图 5-48　阴影区域代表在窗口操作要重画的部分

(a) 放在操作;　(b) 弹出操作;　(c) 卷滚操作。

有两种显示重新显露区域的策略:

(1) 窗口系统不负责绘制窗口中显露出来的那部分区域里的图形,而是发一个“窗口显露”事件给负责这个窗口显示的客户程序,由客户程序绘制这个区域的图形。窗口系统不保存窗口中被遮挡部分的内容,也不保存窗口以外区域的内容。输出的图形将按窗口的可见域裁剪。比如显示一条直线时,只显示直线在窗口中可见部分,其余部分全被裁剪掉,而且不做任何保存。

负责窗口显示的客户程序在接收到由窗口系统发来的“窗口显露”事件后,重新显示窗口中的图形。有三种显示方法:

① 在用户坐标系中建立一个与窗口系统中的窗口相对应的裁剪窗口,整个窗口区域,不管是新显露的区域还是原来就可见的区域,都重画一遍。

② 只建立一个与显露部分相对应的小裁剪域,只重画这个区域的图形。

③ 预先保存被遮挡区域的点位图,在窗口显露出来时,不再重画,而是用 PixBlt 命令将保存的点位图显示在屏幕窗口上。

像这样把窗口显示的任务交给客户程序的策略无疑增加了客户程序的负担,但的确是一个行之有效的方法,可以保证窗口系统适应于具有不同数据模型(如点位图、有结构的层次图、更复杂的关系模型)的应用程序。在窗口本身不能提供一种适合各种应用程序需求的数据模型时,最好的办法就是把任务交给应用程序,让应用程序来进行有效的数据存储和提供快速重画能力。

(2) 窗口系统保存窗口中被遮挡的部分。这样在窗口放大、弹出、或卷滚时,客户程序就不必显示新显露出来的部分。有些窗口系统允许客户程序选择是否要窗口系统保存这部分内容。如果要求窗口系统负责重画新显露的区域,那么窗口系统应该保存多少内容?典型的是尽可能多地保存窗口,通常是全屏幕。有些窗口系统则不惜存储空间,保存一个比显示屏幕本身还要大的点位图。“窗口显露”事件仍还要发送给客户程序,因为在卷滚操作时,客户程序要负责重画不在点位图中的那部分显露区域;在窗口中的图形被缩放时,客户程序也要负责重画工作。

窗口系统为每一个窗口保存一个后备点位图,当窗口被遮挡的部分被显露出来后,这部分的内容就从后备点位图中用 PixBlt 命令拷贝到屏幕窗口上。这种双窗口的方法在窗口的内容更新时可能较慢,因为对于被遮挡的部分,客户程序也要将其画到后备点位图中去,而实际显示的可见区域可能只占后备点位图的很小一部分。窗口系统在显示时,只把点位图中可见部分的内容拷贝到屏幕窗口上。避免双重存贮的办法是把窗口划分成多个矩形域,不可见域的图形存到后备点位图中,而可见部分只存放在屏幕的帧缓存中。在窗口的某个可见部分要被遮挡时,再将这部分内容拷贝到后备点位图中去,这样会比更新窗口,又更新后备点位图要快得多。

对窗口的缩放操作,其效果有两种可能,一是窗口中原有的图形尺寸不变,而在窗口放大时,用户看到更多的图形,在缩小时看到的少一些。另一种是窗口中原有的内容不变,而是随着窗口的放大而变大,随着窗口的缩小而变小。

有些窗口系统使用有层次结构的窗口,即一个窗口中有多个子窗口,子窗口可能还有子窗口。子窗口常包含在父窗口内。层次窗口可以用来实现对话框、参数板、菜单。例如,

整个对话框可以看成是一个父窗口,而对话框中的每一个域、按键盒等是互相独立的子窗口,按键盒中的各个按键又是按键盒的子窗口。图 5-49 显示了一个有层次结构的窗口。

(a) (b)

图 5-49

(a) 设置图象属性的对话框; (b) 对话框的层次结构

对于有层次结构的窗口,其资源分配中有很多微妙之处。比如当父窗口的大小改变时,要确定子窗口受何影响。当一个主程序创建的子进程又生成了一个新窗口时,要确定这个新窗口与主程序所在窗口的关系。

客户程序可以调用的窗口系统子程序有:

createWindow (name)	生成一个新窗口,它将成为当前窗口
setPosition(Xmin, Ymin)	设置当前窗口的位置
setSize(height, width)	设置当前窗口的大小
selectWindow(name)	选择窗口,使其成为当前窗口
hideWindow	使当前窗口不可见,并显露所有被当前窗口遮挡的窗口
showWindow	使当前窗口完全可见,并在所有窗口之上。
getPosition(&x, &y)	获取当前窗口的位置
getSize(&height, &width)	获取当前窗口的大小
bringToTop	使当前窗口在所有窗口之上
sendToBottom	使当前窗口在所有窗口之下
destroyWindow	删除当前窗口

窗口管理系统需要管理的另一个输出资源是颜色查找表。假设有一个 8 个位平面的显示系统,有两个客户程序要在这个显示系统的屏幕上输出图形。这时窗口系统就要分别为它们分配一个颜色查找表。如果均分的话,那么每个客户程序可以得到一半(128 项颜色)。每个客户程序能够分配到多少颜色项与客户程序的个数有关。窗口系统通常要定义一个上限,以便确定每个客户程序能分配到多少颜色项,但如果实际上只有很少几个客户程序同时运行时,就又浪费了颜色表。

考虑到某个时刻只会有一个客户程序在运行,所以有的窗口系统就把查找表全部让

给光标所在的窗口使用。这样做是可行的，但整个屏幕的显示在查找表从一个客户程序交换到另一个客户程序时，会发生奇怪的变化。

还有一种方法，是分派颜色而不是分配查找表。如果某个客户程序要用纯蓝色，这时查找表中的某一项又恰好是这个颜色的话，那么就把这一项的索引地址分配给这个客户程序使用。如果查找表中没有纯蓝色，而且查找表中还有空项，则分配一个空项给纯蓝色。否则的话，把查找表中最接近纯蓝色的项分配给这个客户程序。显然，对于这个方法，如果查找表已满，而各项的颜色与所要求的颜色相去甚远，就不允许客户程序改变颜色查找表，也不是很好的。但除此之外，也没有什么令人满意的方法。

2．窗口系统的输入处理

窗口系统的输入处理主要包括输入设备管理和事件管理。联结到计算机上的各种输入设备以及由这些设备所产生的事件是窗口系统要管理的输入资源。窗口系统从输入设备中获取事件，把事件与相应的客户程序联系起来，并把事件发送给与之相联系的客户程序。因此在分配事件之前，窗口系统必须知道哪个事件是要发送给哪个客户程序的。

除了输入设备可以产生事件外，窗口系统本身也会产生一些特殊事件。例如，当光标进入某个窗口时，会产生“进入窗口”事件；当光标离开某个窗口时，会产生“离开窗口”事件。这两个事件发送给用户接口程序，让用户接口程序着重显示当前窗口。当窗口放大、弹出、卷滚时，窗口的内容需要重画。这时会产生“窗口被毁”事件，通知用户程序进行重画工作。“进入窗口”和“离开窗口”事件是用户操作的直接反应，而“窗口被毁”事件则是在客户程序要求改变窗口时产生的二级事件。上述这些事件也都要送到客户程序中去。有些事件还要送到多个客户程序中去，如用户关闭窗口产生的“窗口被毁”事件，就要送到被这个窗口遮挡的所有窗口的客户程序中去。

有如下类型的事件：

KeyPress	按下键盘上的键
KeyRelease	释放键盘上的键
ButtonPress	按定位器(如鼠标器)上的键
ButtonRelease	释放定位器上的键
Motion	移动光标
LeaveWindow	光标从窗口离开
EnterWindow	光标进入窗口
WindowExpose	窗口被显露出来
ResizeRequest	要求改变窗口大小
Timer	计时器计时已到

对于多层次窗口，子窗口可以像父窗口一样由用户操作，因而也可以与各种事件联系起来。与某个事件相联系的窗口通常是光标所在的、最低级的、并且允许这个事件发生的窗口。不同类型的事件可以与不同窗口相联系，事件队列中记录的每一个事件都把与之相联系的窗口作为它记录的一部分。在一个窗口中发生的事件可以向包含这个窗口的父窗口报告，但要做到这一点，客户程序必须使最外层到最低层的所有窗口都允许这一事件。只有这样，同一个事件，才会被送到多个窗口的事件队列中去。

比如弹出式菜单，可以看成是一个主窗口，而一个个菜单项是子菜单。当光标由菜单

项 i 移出,并移进菜单项 i+ 1 时,窗口 i 的“离开窗口”事件将先送入事件队列,窗口 i+ 1 的“进入窗口”事件接着放入这个队列。客户程序处理第一个事件时,使菜单项 i 的着重显示恢复正常;处理第二个事件时,着重显示菜单项 i+ 1,形成光标进出菜单项的反馈。

向客户程序分配事件,有两种基本方法。这两种方法已被窗口系统广泛使用。

① 定点发送

寻找事件发生时光标所在的窗口,将事件直接送到创建这个窗口的客户程序,即送到这个窗口的所有者。

为了实现定点发送,窗口系统必须维护一个数据链表,链表的各节点记录每个窗口的边界。当一个事件出现时,立即搜索这个数据链表,找到包含光标的可见窗口。如果数据链是有序的,最上层的窗口排在链表的前头,那么只要找到第一个包含光标的窗口就可以了。对于多层次窗口,需要更复杂的数据结构和搜索方法。

② 定向发送

一个客户程序通知窗口系统将某类事件发送给另一个客户程序。例如,窗口管理器提供这样一个命令,允许用户把所有的键盘输入事件发送给拥有某个窗口的客户程序,实现定向输入,而不管光标是否在这个窗口中。窗口管理器接收到这样的定向输入命令后,通知窗口系统把键盘输入事件发送给这个窗口的客户程序。定向发送是发送键盘事件的常用方法,但即使是鼠标器的按键事件,也可以重定向。

图 5-50 显示了窗口系统与非窗口系统处理事件的方式。在非窗口系统中(图 5-50(a)),应用程序一接到事件就将事件分配到相应的事件处理过程。事件分配工作由应用程序本身来完成。以下是典型的事件分配程序:

```
repeat
    waitEvent (timeout, &deviceClass, &device);
    case deviceClass of
    class1:
        case device of
            device1: procedure A;
            device2: procedure B;
    class2:
        ┆
```

而在窗口管理系统中(图 5-50(b)),事件的分配工作由窗口系统来完成,用户的应用程序只要登记和处理事件就可以了。

登记是指应用程序在创建某个窗口时,向窗口系统登记处理事件的过程,以及说明在哪些事件发生时,这个处理事件的过程才被调用。Xview 的例程序如下:

```
XV_set (window,    /* window 为已生成的窗口名* /
        WIN_EVENT_PROC, user_event_proc,
        WIN_CONSUME_EVENTS,
            LOC_WINENTER,
            WIN_MOUSE_BUTTONS,
            NULL,
        NULL);
```

例程序中,use_event_proc 为处理事件的过程名,当 LOC_WINENTER(进入窗口事件)、

WIN_MOUSE_BOTTONS(鼠标器按键事件)发生时,这个过程将被调用。

典型的事件处理过程如下:

```
void user_event_proc (window, event)
WINDOW window;
EVENT * event;
{
   具体的事件处理内容;
}
```

让窗口系统承担事件的检测和分派工作的直接好处是把用户从繁锁的事件分配中解放出来,专门考虑事件处理程序。

图 5-50

(a) 非窗口系统的事件处理方式; (b) 窗口系统的事件处理方式。

窗口系统提供的事件输入子程序有:

EnableEvents(eventList)　　允许事件发生

WaitEvent(timeout, eventType, eventRecord, window)

从事件队列中取下一个事件

SetInputFocus(window, eventList)　　事件定向

三、交互技术工具箱

用户接口的外观和给人的感受很大程度上取决于它们所使用的各种交互技术。设计和实现一组好的交互技术不是一件容易的事。这里介绍三种交互技术工具箱。用交互技术工具箱,可以实现统一的外观、统一的交互风格,可以节省编程时间。

1. 用户接口子程序库

用户接口子程序库是为应用程序设计者提供的、用来设计各种交互技术的子程序集。各个窗口管理系统都会提供用户接口子程序库。现在广泛使用的子程序工具箱有:Andraw 窗口管理系统的工具箱,Macintosh 工具箱,Ms- Windows 的工具箱,X 窗口管理系统的 OSF/Motif 和 InterViews 工具箱,在 X 和 NeWS 两个窗口系统上实现的 OPENLOOK 工具箱,Presentation 窗口管理系统的工具箱和 SunView 窗口管理系统的工具箱。

工具箱中的子程序不仅可以由应用程序调用,也可以在窗口管理器中调用。在多个应用程序中使用同一个工具箱,就可以保证应用程序、窗口管理系统的外观和交互风格相一致。例如,对话框的外形、布局、颜色,对话框中的文字输入域、列表框、按键等,在各个应用程序之间都具有同样的风格。

就像图 5-45 所示的那样,工具箱是在窗口管理系统之上实现的,但如果没有窗口管理系统,也可以直接在图形包上实现。然而工具箱中的元素,如菜单、对话框、卷滚条等在窗口中实现是比较方便的。

在 X 窗口系统中,交互技术叫做 widget。典型的 widget 有对话框、文件选择框、警告框、帮助框、列表框、文字输入框、信息框、圆按键、圆按键盒、选择键、选择键盒、开关按键、开关按键盒、固定菜单、弹出式菜单、卷滚条、窗口。每个 widget 都当成一个窗口来实现。X 窗口系统是一个多层次的窗口系统,因而,可以把圆按键看成是圆按键盒的子窗口。复杂的对话框会含有多个子窗口。

在子程序工具箱中,还有事件通知器(notifier),在事件发生时调用事件处理程序(callback)。在有些情况下,事件处理过程也是工具箱的一部分,如着重显示当前菜单项的过程,关闭一个窗口的过程,事件处理过程主要由用户程序提供,如菜单命令的执行过程、检查字符输入域中字符串的有效性等程序。图 5-51 是一个对话框的局部及其相关的事件处理过程。

图 5-51　对话框及其事件处理过程

其中,CopyProc 在文字串输入结束后被调用,检查输入的文字串是否是有效的数字;

PaperSourceProc、CoverPageProc 记录当前所选的圆键,并保证只有一个圆键被选中。

要注意的是,在上述 widget 中,既有高层次的工具,又有低层次的工具,一些 widget 可能是由另一些 widget 组成的。例如,一个对话框可能会由几个圆键盒、开关盒、文字输入域组成。因此,工具箱中还包括由低级 widget 组成高级 widget 的子程序。以下是生成图 5-52 对话框的 SunView 例程序。

```
frame= (Frame)xv_create (NULL,
            FRAME,                  /* 指明生成对话框的外框* /
            WIN_SHOW,               TRUE,
            FRAME_LABEL,            "sample",
            XV_WIDTH,               200,
            XV_HEIGHT,              100,
            NULL);
panel= (Panel) xv_create (frame,
            PANEL,                  /*  指明生成对话板* /
            NULL);
(void) xv_create (panel,
            PANEL_BUTTON,           /* 指明生成按键* /
            PANEL_LABEL_STRING,     "Quit",
            PANEL_NOTIFY_PROC,      quit_proc,
            NULL);
```

图 5-52　一个只含 Quit 按键的对话框

2. 资源描述语言

MS_Windows 用资源描述语言定义对话框或菜单。资源描述语言中的语句对每一个资源命名,并加以描述。图 5-53 显示了一个对话框及其相应的资源描述文件 .RC。资源编译器 RC 将资源描述文件 .RC 编译成资源代码 .RES,再加入到用户程序的可执行代码中去。整个编译过程如图 5-54 所示。

3. 资源编辑器

虽然用子程序库或资源描述语言可以简化编程,保证用户接口具有一致的风格,但用程序实现一个对话框或窗口,还是很难的。一种更好的方法是使用资源编辑器。先用资源编辑器交互地设计构造窗口布局,再转换成程序码,与应用程序联结成可运行的目标代码。使用资源编辑器,不需要任何编程技能,所以,即使是窗口界面的设计人员,也能用这个编辑器设计对话接口。

资源编辑器是一个典型的交互设计工具。图 5-55 所示的,是 MS_Window 3.X 的对

图 5-53　对话框及其资源描述文件

图 5-54　资源编译流程

话框编辑器。对话框编辑器是一种在屏幕上交互设计和测试对话框的工具。

对话框编辑器可以创建或修改以下文件:

· 对话框描述文件(.DLG): 含有资源编译器可解释的对话语句和控制语句;

· 资源文件(.RES): 含有多个对话框资源和其它资源(如图标、光标形状);

· 头文件(.H): 可插入到 C 程序或资源描述文件(.RC)开头的文件,常包含一些常数定义、参变量定义、过程名定义等。

图 5-55　对话框编辑器

图 5-56　对话资源的编辑和编译流程

资源编译器对上述文件进行编译,并与应用程序联结生成可运行的代码。编译流程如图 5-56 所示。

习　　题

1. 列出 MS- Windows 3. X 所用的各种交互任务,并按四种基本交互任务分类。如果有不能归到四种交互任务的,请将其进一步分解。
2. 用程序实现固定菜单。当光标在菜单上移动时,要着重显示光标所在的菜单,作为光标移动的反馈。
3. 用程序实现 UNIX 命令解释器。先提示用户输入命令,在用户输入一个命令后,与命令集中的元素匹配,给出“不匹配”,“匹配”、以及“与某个命令相接近”等反馈信息。
4. 在光栅显示器上用程序实现一个显示文件系统子目录的对话框。
5. 在光栅显示器上用程序实现用橡皮筋作直线的过程。
6. 你所用的汉字编辑系统具有所见即所得的图形接口风格吗? MS- Windows 3. X 所带的文字编辑器 WRITE 是一个所见即所得的图形接口,请试用。
7. 你所用的窗口管理系统中有哪些操作是直接操作? 有哪些命令你认为是可以改成直接操作的?
8. 请用窗口系统提供的图标设计工具设计一个表示可执行程序的图标和一个表示执行程序的图标。
9. 列出你所用窗口管理系统的各种反馈方式。
10. 列出 10 个视觉编码方式。
11. 考察你所用的各种图形交互接口的出错恢复功能。有几种方式? 是否满足你的需要?
12. 请设计一个具有后缀文法的用户接口,实现 UNIX 命令集的解释。
13. 设计适合多层次窗口条件下定点发送事件的数据结构。
14. 考察两到三个窗口管理系统,回答下述问题:
    a) 是否具有多层次窗口?
    b) 是否是客户- 服务器方式? 如果是,是否可以允许客户程序驻留在网络的另一个节点上?
    c) 是否提供定点发送事件和定向发送事件?
    d) 窗口管理系统与图形包是集成在一起,还是将图形调用直接传到图形硬件设备上?
    e) 是否保持一致的交互风格?
    f) 是否面向多层次用户?
15. 分别用 MS- Windows 3. X 提供的三种交互技术工具箱实现同一个对话框。

第六章　三维形体的表示

在计算机屏幕上产生一幅有立体感的图象的过程和用照像机拍摄一张照片的过程十分相似。当我们想拍摄某一物体的照片时，其先决条件是该物体必须是现实存在的。同样，要想在计算机屏幕上产生一个三维形体的图象，首先必须在计算机内构造并表示出该三维形体的模型，然后再用本书中其它章节讨论的投影变换及真实图形生成技术等在屏幕上产生出图象。因此，在计算机内表示、构造三维形体并进行运算的技术，即几何造型技术，是计算机图形学的一个重要组成部分。由于本书篇幅所限，不能对几何造型作全面论述，在本章中仅仅讨论三维形体的表示问题，因为这是几何造型的基础。有了这一基础，简单的形体也就可以构造了。另外，由于三维空间中点、线、平面等的表示比较简单，或者在其它课程中已经讨论过了，因此，本章集中讨论最主要的两个问题，即三维空间中曲面的表示和实体的表示问题。

第一节　曲面的表示

第三章第三节中曾讨论到一条自由曲线可以由一系列的曲线段连接而成。与此类似，一自由曲面也可以由一系列的曲面片拼合而成，因此，曲面片是曲面的基本单元，一个曲面片是以曲线为边界的点的集合，这些点的坐标(x, y, z)均可用双参数的单值函数表示，即

$$x = x(u,w),\quad y = y(u,w),\quad z = z(u,w)$$

因此，我们首先讨论的是曲面片的数学表示形式及其性质。

和自由曲线一样，我们可用三次参数方程来表示曲面片，即

$$\begin{aligned} Q(u,w) = {} & a_{33}u^3w^3 + a_{32}u^3w^2 + a_{31}u^3w + a_{30}u^3 \\ & + a_{23}u^2w^3 + a_{22}u^2w^2 + a_{21}u^2w + a_{20}u^2 \\ & + a_{13}uw^3 + a_{12}uw^2 + a_{11}uw + a_{10}u \\ & + a_{03}w^3 + a_{02}w^2 + a_{01}w + a_{00} \end{aligned}$$

或写为

$$Q(u,w) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}u^iw^j \qquad (6\text{-}1)$$

式中：u, w 为参数；$u, w \in [0,1]$。此参数方程共有 16 个系数向量，每一系数向量都有 3 个独立的分量，因而总共有 48 个自由度。由于两个参数向量都有三次方项，因而称为双三次曲面片，图 6-1 所示为双三次曲面片的一例。

在式$(6\text{-}1)$中，如令 u, w 均等于零，则得 $Q(0,0) = a_{00}$，这是曲面片的一个顶点，用 P_{00} 表示。同理，如令 $u=0, w=1$；$u=1, w=0$ 及 $u=1\ w=1$，则可得到曲面片的其它三个顶点 $Q(0,1)$，$Q(1,0)$ 及 $Q(1,1)$，分别用 P_{01}，P_{10} 及 P_{11} 表示。与此类似，如果我们令 u, w 两个参

图 6-1

数之一等于其极限 0 或 1, 而另一参数在[0, 1] 之间变化, 就可得到 4 条边界曲线。如果我们令 u= u_i, 而 w 在[0, 1] 之间变化, 就可得到曲面上的一条曲线。与此相反, 如果令 w= w_j, 而 u 在[0, 1] 之间变化, 就可得到曲面上的另一条曲线。显而易见, 两条曲线的交点即为 $Q(u_i, w_j)$, 见图 6-1。

与第三章中三种不同形式的曲线相对应, 双三次曲面片也有三种不同的形式, 现分别讨论如下。

一、孔斯(Coons) 曲面

由前(3-28) 式可知 Hermite 曲线可表示为

$$Q(t) = TM_hG_h = F_h(t)G_h$$

其中, $F_h(t)$ 称为调和函数, G_h 为 Hermite 几何矢量。如果我们用 Hermite 曲线的方式来给定曲面片四条边界曲线的初始条件, 就可以得到如下的表示式

$$Q(u,0) = F_h(u)[P_{00} \quad P_{10} \quad P_{00}^u \quad P_{10}^u]^T \tag{6-2}$$

$$Q(u,1) = F_h(u)[P_{01} \quad P_{11} \quad P_{01}^u \quad P_{11}^u]^T \tag{6-3}$$

$$Q(0,w) = F_h(w)[P_{00} \quad P_{01} \quad P_{00}^w \quad P_{01}^w]^T \tag{6-4}$$

$$Q(1,w) = F_h(w)[P_{10} \quad P_{11} \quad P_{10}^w \quad P_{11}^w]^T \tag{6-5}$$

式中 $F_h(u)$, $F_h(w)$ 为 Hermite 曲线的调和函数, P_{00}^u 表示在 u= 0、w= 0 这一点的 $\frac{Q(u,w)}{u}$。P_{00}^w 表示在 u= 0、w= 0 这一点的 $\frac{Q(u,w)}{w}$, 余类推。

式(6-2) 至(6-5) 只给定了 12 个向量作为初始条件, 为了全部确定式(6-1) 中的 16 个未知系数向量, 尚需给定其它 4 个向量作为初始条件。这 4 个向量是位于 4 个顶点的扭矢, 其数学表示式为

$$P_{00}^{uw} = \frac{\partial^2 Q(u,w)}{\partial u\,\partial w}\ (\text{在顶点 } u=0, w=0 \text{ 处})$$

$$P_{10}^{uw} = \frac{\partial^2 Q(u,w)}{\partial u\,\partial w}\ (\text{在顶点 } u=1, w=0 \text{ 处})$$

$$P_{01}^{uw} = \frac{\partial^2 Q(u,w)}{\partial u\,\partial w}\ (\text{在顶点 } u=0, w=1 \text{ 处})$$

$$P_{11}^{uw} = \frac{\partial^2 Q(u,w)}{\partial u\,\partial w}\ (\text{在顶点 } u=1, w=1 \text{ 处})$$

这就是式(6-1)在四个顶点处的混合偏导数,可由该式求出,即

$$\frac{\partial^2 Q(u,w)}{\partial u\,\partial w} = 9a_{33}u^2w^2 + 6a_{32}u^2w + 3a_{31}u^2 + 6a_{23}uw^2 + 4a_{22}uw + 2a_{21}u + 3a_{13}w^2 + 2a_{12}w + a_{11} \tag{6-6}$$

于是,可得 4 个顶点处的扭矢为

$$P_{00}^{uw} = a_{11} \tag{6-7}$$

$$P_{10}^{uw} = 3a_{31} + 2a_{21} + a_{11} \tag{6-8}$$

$$P_{01}^{uw} = 3a_{13} + 2a_{12} + a_{11} \tag{6-9}$$

$$P_{11}^{uw} = 9a_{33} + 6a_{32} + 3a_{31} + 6a_{23} + 4a_{22} + 2a_{21} + 3a_{13} + 2a_{12} + a_{11} \tag{6-10}$$

与此类似,可由式(6-1)求出其余 12 个向量与未知系数向量的关系。它们是

$$P_{00} = a_{00} \tag{6-11}$$

$$P_{10} = a_{30} + a_{20} + a_{10} + a_{00} \tag{6-12}$$

$$P_{01} = a_{03} + a_{02} + a_{01} + a_{00} \tag{6-13}$$

$$P_{11} = a_{33} + a_{32} + a_{31} + a_{30} + a_{23} + a_{22} + a_{21} + a_{20} + a_{13} + a_{12} + a_{11} + a_{10} + a_{03} + a_{02} + a_{01} + a_{00} \tag{6-14}$$

$$P_{00}^{u} = a_{10} \tag{6-15}$$

$$P_{00}^{w} = a_{01} \tag{6-16}$$

$$P_{10}^{u} = 3a_{30} + 2a_{20} + a_{10} \tag{6-17}$$

$$P_{10}^{w} = a_{31} + a_{21} + a_{11} + a_{01} \tag{6-18}$$

$$P_{01}^{u} = a_{13} + a_{12} + a_{11} + a_{10} \tag{6-19}$$

$$P_{01}^{w} = 3a_{03} + 2a_{02} + a_{01} \tag{6-20}$$

$$P_{11}^{u} = 3a_{33} + 3a_{32} + 3a_{31} + 3a_{30} + 2a_{23} + 2a_{22} + 2a_{21} + 2a_{20} + a_{13} + a_{12} + a_{11} + a_{10} \tag{6-21}$$

$$P_{11}^{w} = 3a_{33} + 2a_{32} + a_{31} + 3a_{23} + 2a_{22} + a_{21} + 3a_{13} + 2a_{12} + a_{11} + 3a_{03} + 2a_{02} + a_{01} \tag{6-22}$$

图 6-2 示出了双三次曲面片及其 12 个向量。

显而易见,将式(6-7)至式(6-22)联立求解,即可求出 16 个系数向量,从而得出用 4 个顶点向量、8 个切线向量及 4 个扭矢向量表示的双三次曲面片方程。但是,这种方法相当繁琐,且几何意义不够明显。为了更好地说明这一问题,我们先讨论一下扭矢的几何意义。

图 6-2

一个双三次曲面片是由参数空间相互正交的两组曲线集组成的，这两组曲线集分别由参数 u 及 w 来定义。一组曲线包括 u= 0 及 u= 1 这两条边界曲线及无穷多条由 u= i 决定的中间曲线。与此相似，另一组曲线包括 w= 0 及 w= 1 这两条边界曲线以及无穷多条由 w= j 决定的中间曲线。

由式(6-2)至(6-5)，可以定义 4 条边界曲线，但是如何来定义中间曲线呢?

首先，在式(6-4)及式(6-5)中令 w= j，即可分别求出曲线 Q(u, j) 的两个端点 P_{0j} 及 P_{1j}。但是，又如何求出这两个端点处的切线 P^{u}_{0j} 及 P^{u}_{1j} 呢? 一般说来，P^{u}_{ow} 及 P^{u}_{1w} 是随着 w 的变化而变化的，其变化率为

$$\frac{(P^{u}_{0w})}{w} = \frac{^{2}Q(u,w)}{u\ w}\Bigg|_{u=0,\,w=w}$$

$$\frac{(P^{u}_{1w})}{w} = \frac{^{2}Q(u,w)}{u\ w}\Bigg|_{u=1,\,w=w}$$

在 4 个顶点处，这一变化率就是前面讨论过的扭矢 P^{uw}_{00}，P^{uw}_{10}，P^{uw}_{01} 及 P^{uw}_{11}。

在 Hermite 曲线中，我们用两个端点的位置向量及端点处的切线向量来决定一条曲线。与此类似，我们可以用两个端点处的切线向量及扭矢向量来决定边界曲线上中间各点的切线向量。即

$$P^{u}_{0w} = F_h(w)[P^{u}_{00}\quad P^{u}_{01}\quad P^{uw}_{00}\quad P^{uw}_{01}]^{T}$$

$$P^{u}_{1w} = F_h(w)[P^{u}_{10}\quad P^{u}_{11}\quad P^{uw}_{10}\quad P^{uw}_{11}]^{T}$$

$$P^{w}_{u0} = F_h(u)[P^{w}_{00}\quad P^{w}_{10}\quad P^{uw}_{00}\quad P^{uw}_{10}]^{T}$$

$$P^{w}_{u1} = F_h(u)[P^{w}_{01}\quad P^{w}_{11}\quad P^{uw}_{01}\quad P^{uw}_{11}]^{T}$$

因此，虽然扭矢与双三次曲面片 4 条边界曲线的形状无关，但它却影响边界曲线上中间各点的切线向量，从而影响整个曲面片的形状。图 6-3 表示出扭矢对曲面片形状的影响，但并不影响曲面片的 4 条边界曲线。

图 6-3

下面,我们可以推导出由顶点向量、顶点处的切线向量以及扭矢表示的双三次曲片面方程了。如图 6-4 所示,曲面片上一点 $Q(i,j)$ 位于 $u=i$、w 在 $[0,1]$ 之间变化及 $w=j$、u 在 $[0,1]$ 之间变化所形成的两条曲线的交点上。求 $Q(i,j)$ 的问题可以转化为求 Hermite 曲线(例如 $Q(i,w)$)上参数值 $w=j$ 的那一点,而这一条 Hermite 曲线又是由 P_{i0}, P_{i1}, P^{w}_{i0}, P^{w}_{i1} 所决定的。

图 6-4

于是,我们有

$$P_{i0}=F_{h1}(u_i)P_{00}+F_{h2}(u_i)P_{10}+F_{h3}(u_i)P^{u}_{00}+F_{h4}(u_i)P^{u}_{10}$$

$$P_{i1}=F_{h1}(u_i)P_{01}+F_{h2}(u_i)P_{11}+F_{h3}(u_i)P^{u}_{01}+F_{h4}(u_i)P^{u}_{11}$$

$$P_{i0}^{w} = F_{h1}(u_i)P_{00}^{w} + F_{h2}(u_i)P_{10}^{w} + F_{h3}(u_i)P_{00}^{uw} + F_{h4}(u_i)P_{10}^{uw}$$

$$P_{i1}^{w} = F_{h1}(u_i)P_{01}^{w} + F_{h2}(u_i)P_{11}^{w} + F_{h3}(u_i)P_{01}^{uw} + F_{h4}(u_i)P_{11}^{uw}$$

由此可以得出:

$$\begin{aligned} Q(i,j) = P_{ij} &= F_{h1}(w_j)P_{i0} + F_{h2}(w_j)P_{i1} + F_{h3}(w_j)P_{i0}^{w} + F_{h4}(w_j)P_{i1}^{w} \\ &= F_{h1}(w_j)[F_{h1}(u_i)P_{00} + F_{h2}(u_i)P_{10} + F_{h3}(u_i)P_{00}^{u} + F_{h4}(u_i)P_{10}^{u}] \\ &\quad + F_{h2}(w_j)[F_{h1}(u_i)P_{01} + F_{h2}(u_i)P_{11} + F_{h3}(u_i)P_{01}^{u} + F_{h3}(u_i)P_{11}^{u}] \\ &\quad + F_{h3}(w_j)[F_{h1}(u_i)P_{00}^{w} + F_{h2}(u_i)P_{10}^{w} + F_{h3}(u_i)P_{00}^{uw} + F_{h4}(u_i)P_{10}^{uw}] \\ &\quad + F_{h4}(w_j)[F_{h1}(u_i)P_{01}^{w} + F_{h2}(u_i)P_{11}^{w} + F_{h3}(u_i)P_{01}^{uw} + F_{h4}(u_i)P_{11}^{uw}] \end{aligned}$$

将上式中参数 u, w 的下标 i, j 去掉, 并改写为矩阵形式, 即得到 Hermite 曲面的一般表示:

$$Q(u,w) = [F_{h1}(u) \quad F_{h2}(u) \quad F_{h3}(u) \quad F_{h4}(u)] \times \begin{bmatrix} P_{00} & P_{01} & P_{00}^{w} & P_{01}^{w} \\ P_{10} & P_{11} & P_{10}^{w} & P_{11}^{w} \\ P_{00}^{u} & P_{01}^{u} & P_{00}^{uw} & P_{01}^{uw} \\ P_{10}^{u} & P_{11}^{u} & P_{10}^{uw} & P_{11}^{uw} \end{bmatrix} [F_{h1}(w)F_{h2}(w)F_{h3}(w)F_{h4}(w)]^{T} \tag{6-23}$$

如令

$$B = \begin{bmatrix} P_{00} & P_{01} & P_{00}^{w} & P_{01}^{w} \\ P_{10} & P_{11} & P_{10}^{w} & P_{11}^{w} \\ P_{00}^{u} & P_{01}^{u} & P_{00}^{uw} & P_{01}^{uw} \\ P_{10}^{u} & P_{11}^{u} & P_{10}^{uw} & P_{11}^{uw} \end{bmatrix}$$

并将上式简化, 则得

$$Q(u,w) = F_h(u)BF_h(w)^{T} \tag{6-24}$$

式中, $F_h(u)$, $F_h(w)$ 分别是变量 u 及 w 的 Hermite 调和函数。故

$$F_h(u) = [u^3 \quad u^2 \quad u \quad 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = UM_h$$

$$F_h(w) = [w^3 \quad w^2 \quad w \quad 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = WM_h$$

于是有

$$Q(u,w) = UM_hBM_h^{T}W^{T} \tag{6-25}$$

上式中的 B 称为角(顶)点矩阵, 它的排列非常整齐, 矩阵中的 16 个元素可以分为 4 组, 左上角 4 个元素代表 4 个角点的位置向量, 右上角和左下角分别代表边界曲线在 4 个角点处的两组切线向量, 右下角的一组则为角点扭矢。

需要说明的是, 这种 Hermite 双三次曲面是 Coons 曲面的一种形式, 之所以这样命名

是因为后来 Steven Coons 在这个问题上作了很多工作。双三次 Coons 曲面的主要缺点是必须给定矩阵 B 中的 16 个向量,才能唯一确定曲面片的位置和形状,而要给定扭矢是相当困难的,因而使用起来不太方便。

如果我们令所有的扭矢为零,则 6-24 式中的几何系数矩阵 B 变为

$$B=\begin{matrix} P_{00} & P_{01} & P^{w}_{00} & P^{w}_{01} \\ P_{10} & P_{11} & P^{w}_{10} & P^{w}_{11} \\ P^{u}_{00} & P^{u}_{01} & 0 & 0 \\ P^{u}_{10} & P^{u}_{11} & 0 & 0 \end{matrix}$$

这种形式的曲面片称为 Ferguson 曲面片。这种曲面片在连接边界上只能实现 C^1 连续,但是,它易于构造,在许多应用场合也能满足要求。

二、贝塞尔(Bezier)曲面

前已述及,Bezier 曲线段是由它的特征多边形顶点来决定的,而 Bezier 曲面片则是由特征多面体的顶点来决定的,其数学表示式如下:

$$Q(u,w)=\sum_{i=0}^{m}\sum_{i=0}^{n}P_{ij}B_{i,m}(u)B_{j,n}(w) \qquad (6\text{-}26)$$

$$u,w\in[0,1]$$

其中,$P_{ij}(i=0,1,\cdots,m;j=0,1,\cdots,n)$ 是特征多面体各顶点的位置向量,共计$(m+1)\times(n+1)$个顶点。$B_{i,m}(u)$及$B_{j,n}(w)$是伯恩斯坦多项式,其定义如式(3-46)所示。显然,m 与 n 不一定相等。如 $m=n=3$,则由 4×4 个顶点构造特征多面体,其相应的 Beizer 曲面片称为双三次 Bezier 曲面片。

与双三次 Coons 曲面一样,双三次 Bezier 曲面片也可以用矩阵形式表示为

$$Q(u,w)=F_b(u)PF_b(w)^T \qquad (6\text{-}27)$$

其中,

$$P=\begin{matrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \\ P_{41} & P_{42} & P_{43} & P_{44} \end{matrix}$$

$$F_b(u)=[F_{b1}(u)\quad F_{b2}(u)\quad F_{b3}(u)\quad F_{b4}(u)]$$

$$=[(1-u)^3\quad 3u(1-u)^2\quad 3u^2(1-u)\quad u^3]$$

$$=[u^3\quad u^2\quad u\quad 1]\begin{matrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{matrix}$$

$$=UM_b \qquad (6\text{-}28)$$

$$F_b(w)=[F_{b1}(w)\quad F_{b2}(w)\quad F_{b3}(w)\quad F_{b4}(w)]$$

$$=[(1-w)^3\quad 3w(1-w)^2\quad 3w^2(1-w)\quad w^3]$$

$$= [w^3 \quad w^2 \quad w \quad 1] \begin{bmatrix} 1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$= WM_b \tag{6-29}$$

故有,

$$Q(u,w) = WM_bPM_b^TW^T \tag{6-30}$$

图 6-5 **为双三次** Bezier **曲面片之一例。**

图 6-5

矩阵 P **表示出双三次** Bezier **曲面片特征多面体** 16 **个控制顶点的位置向量。显而易见,这** 16 **个控制点中只有** 4 **个顶点** $P_{11}, P_{14}, P_{41}, P_{44}$**位于** Bezier **曲面上。**P **矩阵中周围的** 12 **个控制点定义了** 4 **条三次** Bezier **曲线,即边界曲线。其余的** 4 **个点** $P_{22}, P_{32}, P_{23}, P_{33}$**与边界曲线无关,但影响曲面片的形状,其作用与双三次** Coons **曲面中的扭矢是一样的。这一点可进一步说明如下:**

一曲面片可用双三次 Coons **曲面表示为**

$$Q(u,w) = UM_hBM_h^TW^T$$

如用 Bezier **曲面表示,则为**

$$Q(u,w) = UM_bPM_b^TW^T$$

如二者表示的是同一曲面,则有

$$UM_hBM_h^TW^T = UM_bPM_b^TW^T$$

故

$$M_hBM_h^T = M_bPM_b^T$$

于是有

$$P = M_b^{-1}M_hBM_h^T[M_b^T]^{-1}$$

令

$$D = M_b^{-1} M_h = \begin{matrix} 0 & 0 & 0 & 1 & 2 & -2 & 1 & 1 \\ 0 & 0 & 1/3 & 1 & -3 & 3 & -2 & -1 \\ 0 & 1/3 & 2/3 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{matrix}$$

$$= \begin{matrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1/3 & 0 \\ 0 & 1 & 0 & 1/3 \\ 0 & 1 & 0 & 0 \end{matrix}$$

故 $D^T = M_h^T [M_b^T]^{-1}$

于是有

$$P = DBD^T = \begin{matrix} 1 & 0 & 0 & 0 & P_{00} & P_{01} & P_{00}^w & P_{01}^w & 1 & 1 & 0 & 0 \\ 1 & 0 & 1/3 & 0 & P_{10} & P_{11} & P_{10}^w & P_{11}^w & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1/3 & P_{00}^u & P_{01}^u & P_{00}^{uw} & P_{01}^{uw} & 0 & 1/3 & 0 & 0 \\ 0 & 1 & 0 & 0 & P_{10}^u & P_{11}^u & P_{10}^{uw} & P_{11}^{uw} & 0 & 0 & 1/3 & 0 \end{matrix}$$

$$= \begin{matrix} P_{00} & P_{00} + \frac{1}{3}P_{00}^w & P_{01} - \frac{1}{3}P_{01}^w & P_{01} \\ P_{00} + \frac{1}{3}P_{00}^u & P_{00} + \frac{1}{3}(P_{00}^u + P_{00}^w) + \frac{1}{9}P_{00}^{uw} & P_{01} + \frac{1}{3}(P_{01}^u - P_{01}^w) - \frac{1}{9}P_{01}^{uw} & P_{01} + \frac{1}{3}P_{01}^u \\ P_{01} - \frac{1}{3}P_{10}^u & P_{10} - \frac{1}{3}(P_{10}^u - P_{10}^w) - \frac{1}{9}P_{10}^{uw} & P_{11} - \frac{1}{3}(P_{11}^u + P_{11}^w) + \frac{1}{9}P_{11}^{uw} & P_{11} - \frac{1}{3}P_{11}^u \\ P_{10} & P_{10} + \frac{1}{3}P_{10}^w & P_{11} - \frac{1}{3}P_{11}^w & P_{11} \end{matrix} \tag{6-31}$$

(6-31)**式表示出双三次** Bezier **曲面片特征多面体控制顶点矩阵** P **同双三次** Coons **曲面的角点信息矩阵** B **中各元素之间的关系。它的几何形态如图** 6-6 **所示。**

图 6-6

在(6-31)式中,作为双三次 Bezier 曲面片的控制顶点矩阵 P,它周围的 12 个控制点定义了 4 条三次 Bezier 曲线,即边界曲线。这 12 个点中,4 个角点是 P_{00}, P_{01}, P_{10}, P_{11}。这就是双三次 Coons 曲面的 4 个角点。其余 8 个顶点均可根据三次 Bezier 曲线的端点性质,由其相应的角点向量及角点切线向量得出,这一点在(6-31)式中看得很清楚。从这一矩阵的中心 4 个元素可以看出,以 $P_{11}P_{12}$和 $P_{11}P_{21}$为邻边作平行四边形时,由 P_{11}的对角顶点到 P_{22}所作的向量,恰巧等于$\frac{1}{9}P_{00}^{uw}$。在其余 3 个角点处也有类似的关系,这些关系都清楚地标明在图 6-6 中。

因此,我们看到了这样一个有趣的事实:双三次 Coons 曲面片的 4 个角点扭矢,同双三次 Bezier 曲面片特征多面体中央 4 个顶点 P_{22}, P_{23}, P_{32}, P_{33}有密切的关系。一旦给定了双三次 Bezier 曲面片的 4 条边界,调整顶点 P_{22}, P_{23}, P_{32}, P_{33}的位置,也就等于在双三次 Coons 曲面中变动 4 个扭矢。

三、B 样条曲面

前已述及,Bezier 曲面是 Bezier 曲线的拓广。与此类似,B 样条曲面则是 B 样条曲线的拓广。(3-53)式曾给出一段 n 次 B 样条曲线的数学表示式。在这里,我们给出一块 m×n 次 B 样条曲面片的数学表示式如下:

$$Q(u,w)=\sum_{i=0}^{m}\sum_{j=0}^{n}P_{ij}F_{i,m}(u)F_{j,n}(w)$$

$$u,w\in[0,1] \qquad (6\text{-}32)$$

式中,P_{ij}($i=0,1,\cdots,m$; $j=0,1,\cdots,n$)是定义此曲面片的顶点位置向量阵列,共计$(m+1)(n+1)$个顶点。$F_{i,m}(u)$, $F_{j,n}(w)$为 B 样条基底函数,u, w 为参数。显然,m 与 n 不一定相等,如 $m=n=3$,则由 4×4 个顶点构成特征多面体,其相应的 B 样条曲面片称为双三次 B 样条曲面片,这是用得最广的一种。

双三次 B 样条曲面片也可以用矩阵形式表示为

$$Q(u,w)=UM_sPM_s^TW^T \qquad (6\text{-}33)$$

其中

$$M_s=\frac{1}{6}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 0 & 3 & 0\\ 1 & 4 & 1 & 0\end{bmatrix}$$

其它符号的含意前已述及,不再重复。

图 6-7

图 6-7 表示一个双三次 B 样条曲面片及其相应的特征多面体顶点网格。

四、曲面片的连接

在讨论了三种不同形式的曲面片以后,如何将曲面片连接起来,组合成完整的曲面呢?曲面片的连接应遵循什么条件呢?

曲面片互相连接时,一般要求连接处具有 C^1 连续的性质。为了实现这一要求,相连接的曲面片不仅应该具有公共的边界曲线,而且在边界曲线上的任何一点,两个曲面片跨越边界的切线向量应该共线,而且两切线向量的长度之比应为常数。

图 6-8 表示出两个双三次 Coons 曲面片 $S(u,w)$ 和 $T(u,w)$ 相连接。$S(u,w)$ 的四个顶点为 $P_{00}, P_{01}, P_{10}, P_{11}$; $T(u,w)$ 的四个顶点为 $Q_{00}, Q_{01}, Q_{10}, Q_{11}$。那么,实现 C^1 连续的条件为:

(1) $S(1,w) = T(0,w)$

(2) $Q_{00}^{u} = aP_{10}^{u}$, $Q_{00}^{uw} = aP_{10}^{uw}$, $Q_{01}^{u} = aP_{11}^{u}$, $Q_{01}^{uw} = aP_{11}^{uw}$

图 6-8

当这些条件满足时,边界曲线 $S(1,w)$ 上任何一点都有 $Q_{0w}^{u} = aP_{1w}^{u}$。

当两个相连接的曲面片为 Bezier 曲面片时(图 6-9),根据 Bezier 曲线的端点性质,实现 C^1 连续的条件为:

(1) $P_{4i} = Q_{1i}$ $i = 1, 2, 3, 4$

(2) $P_{4i} - P_{3i} = \chi(Q_{2i} - Q_{1i})$ $i = 1, 2, 3, 4$

图 6-9

与三次 B 样条曲线相似,双三次 B 样条曲面的优点是极其自然地解决了曲面片之间

的连续性问题。只要将曲面片的特征多面体顶点网格沿某一方向延伸一排，即可以产生另一个曲面片。双三次 B 样条曲面的基本性质自然地保证了二者之间实现 C^2 连续，无需附加其它条件。

第二节　实体的表示

在早期的计算机图形生成技术中，三维物体大多是用线框模型来表示的。线框模型由定义一个物体边界的直线和曲线组成，每一条直线和曲线都是单独构造出来的，并不存在面的信息。直至 80 年代前期，国际上商品化的交互式二、三维图形软件系统中所谓的三维图形功能大多属于这一类。单一的线框造型存在着几个缺陷。第一，用三维线框模型表示三维物体常常具有二义性。图 6-10 表示的是打了一个方孔的长方体，但无论从三个方向中的哪个方向打一个方孔，其结果都如图 6-10 所示。因此，对这一线框模型，可以有三种不同的理解。第二，三维线框模型也易于构造出无效形体。例如，在图 6-11 中，由于 A 点

图　6-10

图　6-11

的位置不恰当，造成同一物体各面之间的相互穿透。这是因为不存在面的信息，而无法检查出这一问题。第三，三维线框模型表示不出曲面的轮廓线，因而也就不能正确表示曲面信息，如图 6-12 所示。因此，线框模型尽管具有比较简单和运算速度较快等优点，但并没有表示出一个三维物体的全部信息，在许多场合不能满足要求，因而应用并不广泛，在本章中也就不作过多讨论，本章的重点是讨论三维物体的实体模型表示。

图　6-12

一、实体的定义及正则形体

要想在计算机内表示、构造一个实体，首先必须对什么是实体有一个确切的定义，并据此检查所构造的实体的有效性。

直至 70 年代末期，关于三维物体的表示和构造并未建立起严密的理论。多数情况下

靠用户来检查物体模型的有效性、唯一性和完备性。随着模型复杂程度的提高,以及实体模型作为计算机辅助设计中某些应用的输入而加以运算和处理,使得通过人的干预来检查模型的有效性变得越来越困难。因此,对实体及其有效性作一个严格的定义,就成为十分必要的了。

目前,一般认为美国 H·B·沃尔克尔(H.B.Voelcker)及 A·A·G·雷契切(A.A.G.Requicha)等人的工作为实体的表示、构造及运算等奠定了必要的理论基础。

Voelcker 及 Requicha 等基于点集拓扑的理论,认为三维空间中的物体是空间中点的集合。并且从点集拓扑的邻域概念出发,通过定义点集的闭包给出正则集的定义。一个开集的闭包指的是该开集与其所有边界点的集合的并集,其本身是一个闭集。组成一个三维物体的点的集合可以分为内部点和边界点两部分。由内部点构成的点集的闭包就是正则集。三维空间中点集的正则集就是三维正则形体,也就是有效的实体。

下面,我们通过图 6-13 所示二维物体的例子进一步说明上述定义。

图 6-13

图 6-13(a)表示,任何一个物体可以定义为点的集合,并且可分为内部点及边界点,边界点可以属于或不属于物体的一部分。该图中,内部点用浅灰色表示,边界点中属于物体的那部分用黑色表示,其余的边界点用深灰色表示。此图表示的物体有悬边及悬点。

图 6-13(b)表示 6-13(a)的闭包,(b)中物体所有的边界点都是物体的一部分。

图 6-13(c)表示 6-13(a)中物体的内部点集,此时,悬边和悬点都去掉了。

图 6-13(d)表示 6-13(c)中物体内部点集的闭包,是一个正则形体。

从图 6-13 可以清楚的看出关于正则形体定义的含义。形象地说,正则形体是由其内部的点集及紧紧包着这些点的表皮组成的。

一个有效的实体应具有的性质如下:

(1) 刚性。一个实体必须有不变的形状,即形状与实体的位置及方向无关。

(2) 维数的一致性。三维空间中,一个实体的各部分均应是三维的,也就是说,必须有连通的内部,而不能有悬挂的或孤立的边界。

(3) 有限性。一个实体必须占有有限的空间。

(4) 边界的确定性。根据实体的边界能区别出实体的内部及外部。

(5) 封闭性。经过一系列刚体运动及任意序列的集合运算之后,仍然保持有效的实体。

我们在这一节里所讨论的是符合正则形体定义的实体,即有效的实体,在以后的叙述中,均简称为实体。

一个实体的表面必须具有的性质如下:

(1) 连通性。位于实体表面上的任意两个点都可用实体表面上的一条路径连接起来。

(2) 有界性。实体表面可将空间分为互不连通的两部分，其中一部分是有界的。

(3) 非自相交性。实体的表面不能自相交。

(4) 可定向性。表面的两侧可明确定义出属于实体的内侧或外侧。

(5) 闭合性。实体表面的闭合性是由表面上多边形网格各元素的拓扑关系决定的。即每一条边具有两个顶点，且仅有两个顶点；围绕任何一个面的环具有相同数目的顶点及边；每一条边连接两个或两个以上的面等等。

以上这些性质广泛应用于实体的表示、构造及运算中。

下面，我们进一步讨论实体表面的二维流形性质。

所谓二维流形指的是对于实体表面上的任何一点，都可以找到一个围绕着它的任意小的邻域，该邻域在拓扑上与平面上的一个圆盘是等价的。这意味着，在邻域的点集和圆盘之间存在着连续的一对一的对应关系(如图 6-14(a)，(b)所示)。与此不同，如果实体表面上的一条边所连接的面多于两个，那么，这条边上任意一个点的小邻域都包含着来自这些面上的点。因此，在拓扑上与平面圆盘是不等价的，这就是非二维流形(图 6-14(c))。显然，非二维流形的物体仍然具有正则形体的性质，因而是正则形体。但是，如图 14(c) 的情况。在物理上是难于实现的，因而现有的造型系统大多数不包括非二维流形的物体。我们下面讨论的也只限于表面具有二维流形性质的实体。

图　6-14

二、正则集合运算及集合成员分类

前已述及，一个有效的实体经过一系列的集合运算之后，应仍然保持为一个有效的实体。怎样才能实现这一要求呢?

为简单起见，我们以二维平面上的物体为例，来讨论这一问题。如图 6-15 所示，设二维平面上有物体 A 及 B(图 6-15(a))，将它们放到图 6-15(b)的位置上。如果以通常的集合运算规则求 A∩B，则得到图 6-15(c) 中的结果。按照正则形体的定义及应满足的性质进行检验，图 6-15(c) 中的结果不符合正则形体的条件，或者说，不是有效的二维形体，因为它有一条悬边，不具有维数的一致性。去掉这条悬边，所得到的图 6-15(d) 中的结果才是一个有效的二维形体。为此，我们把能够产生正则形体的集合运算称为正则集合运算，其相应的正则集合算子以$\cup^*$(正则并)、$\cap^*$(正则交)、$-^*$(正则差)表示。

如何才能实现正则集合运算呢? 基本上有两种方法: 其一是先按照通常的集合运算

图　6-15

求出结果,然后再用一些规则加以判断,删去那些不符合正则形体定义的部分,如悬边、悬面等,从而得到正则形体。这是一种间接的方法。其二是定义出正则集合算子的表达式,用它直接得出符合正则形体定义的结果。下面将分别予以讨论。

用间接方法产生正则形体是基于点集拓扑的邻域概念的。如果 P 是点集 S 的一个元素,那么点 P 的以 R(> 0)为半径的邻域指的是围绕点 P 的半径为 R 的小球。显然,P 的邻域描述了集合 S 在点 P 附近的局部几何性质。当且仅当 P 的邻域为满时,P 在 S 之内;当且仅当 P 的邻域为空时,P 在实体之外;当且仅当 P 的邻域既不为满,也不为空时,P 在实体的边界上。这一性质可以用来对集合运算的结果进行检查。如图 6-16 所示,在集合运算 A∩B 的结果图形上,取点 P 及 R。点 P 的邻域为点 P_A 的邻域和点 P_B 的邻域的交集,现为空集,因而 P 点不在 $A \cap^* B$ 之内。点 R 的邻域为点 R_A 的邻域和点 R_B 的邻域的交集,既不为满也不为空,因而 R 点在 $A \cap^* B$ 的边界上。根据点的性质可以判断该点所在边的性质,从而去掉悬边,得到正则形体。

图　6-16

用直接方法产生正则形体是建立在集合成员分类的基础之上的。前已述及,符合正则形体定义的实体,是三维空间中点的正则集,它可以用它的边界面及内部来表示,即

$$G = \{bG, iG\}$$

式中,G 为符合正则形体定义的实体,bG 为 G 的边界面,iG 为 G 的内部。如果 bG 也符合

上述关于正则形体表面的性质，那么，bG 所包围的空间就是 iG。这就是说，一个实体可以用它的边界面来定义；或者说，已知一个有效实体的边界面，即可定义一个实体。

在三维空间中，给定一个实体 G 后，空间点集就被分为三个子集，一是该实体的内部点集，二是该实体边界上的点集，三是该实体之外的点集。为了产生正则形体，需要决定一个特定的点或点的集合属于哪一个子集，这就是集合成员分类问题。而我们感兴趣的往往是一个有界面相对于一个实体的分类问题。

若给定一个实体 G 及一个有界面 S，那么，S 可能被 G 分割为三部分，即位于 G 内的面，位于 G 外的面以及位于 G 的边界上的面。如 S 相对于 G 的分类函数以 C(S, G) 表示，则有

$$C(S, G) = \{S \text{ in } G, S \text{ out } G, S \text{ on } G\}$$

其中

$$S \text{ in } G = \{p \mid p \in s, p \in iG\}$$

$$S \text{ out } G = \{p \mid p \in s, p \notin G\}$$

$$S \text{ on } G = \{p \mid p \in s, p \in bG\}$$

我们以 -S 表示有界面 S 的反向面，即 -S 和 S 是同一个有界面，只是有界面 -S 上任何一点的法向均和有界面 S 上该点的法向相反。也就是说，如有界面 S 在 P 点的法向为 $N_p(S)$，那么，有界面 -S 在 P 点的法向就是 $-N_p(S)$。于是 S on G 可以进一步分为两种情况，即

$$S \text{ on } G = \{S \text{ shared } (bG), S \text{ shared } (-bG)\}$$

其中：

$$S \text{ shared } (bG) = \{P \mid p \in s, p \in bG, N_p(s) = N_p(bG)\}$$

$$S \text{ shared } (-bG) = \{P \mid p \in s, p \in bG, N_p(s) = -N_p(bG)\}$$

因此，S 相对于 G 的分类函数 C(S, G) 可以修改为

$$C(S, G) = \{S \text{ in } G, S \text{ out } G, S \text{ shared } (bG), S \text{ shared}(-bG)\}$$

如设 A, B 为三维空间的两个实体，〈OP〉为一个集合运算算子，R 是集合运算的结果，则

$$R = A\langle OP\rangle B$$

前面已经说过，如果运算结果 R 仍然是正则的，则〈OP〉为正则集合算子。

由于一个实体可以由它的边界面来表示，如果我们得到了它的边界面，也就定义了该实体。在边界面的分类基础上，我们可以得出关于边界面的三个正则集合算子的表达式：

$$b(A \cup^* B) = \{bA \text{ out } B, bB \text{ out } A, bA \text{ shared } bB\}$$

$$b(A \cap^* B) = \{bA \text{ in } B, bB \text{ in } A, bA \text{ shared } bB\}$$

$$b(A -^* B) = \{bA \text{ out } B, -(bB \text{ in } A), bA \text{ shared } -(bB)\}$$

根据这组表达式，即可定义出新的实体 $A \cup^* B$, $A \cap^* B$ 及 $A -^* B$。

这组表达式的正确性可以严格证明，在此我们就不多述了。有兴趣的读者可参阅有关文献。这里，我们仍然用上面的例子来说明其正确性。图 6-17 表示出两个物体 A 和 B 在正则集合算子 $\cup^*$, $\cap^*$ 及 $-^*$ 的作用下所得出的结果。凡是被上述表达式选中的边界在图中用实线表示，未选中的边界则用虚线表示。一般约定，边界面上任一点的法向指向物体

外部。显然,这是一种利用正则集合算子产生正则形体的直接方法。

图 6-17

三、实体的边界表示

前面已经讨论了用实体的边界来表示一个实体的理论基础。在本小节中,主要讨论边界表示正确性的检验及边界表示的数据结构。

平面多面体是实体中最常见、应用得最广泛的一种,它也可以用来近似的表示曲面体。因此,在本小节中,我们将重点讨论平面多面体的边界表示。

1. 多面体及欧拉(Euler)公式

平面多面体是多面体中最常见的一种。指的是表面由平面多边形构成的三维物体。在具有二维流形性质的平面多面体中,每一条边连接两个面,且仅连接两个面。简单多面体指的是那些经过连续的几何形变可以变换为一个球的多面体,也就是与球具有拓扑等价的那些多面体(如图 6-18 所示)。

图 6-18

前面讲过,一个实体的表面必须满足闭合性,即表面上多边形网格各元素之间的拓扑关系必须满足一系列的条件。那么,有没有比较简单的方法进行检验呢?有的,这就是欧拉公式。

我们先讨论简单多面体的情况。令简单多面体的顶点数,边数和面数分别用 V, E, F, 表示,则存在如下关系:

$$V - E + F = 2$$

这就是有名的欧拉公式,说明了一个简单多面体中顶点数、边数及面数的关系。图 6-19 表示出欧拉公式应用于几个简单多面体的实例。可以用多种方法来证明欧拉公式的正确性,有兴趣的读者可参考有关文献。

这里必须强调一点,欧拉公式只是检查实体有效性的一个必要条件,而不是充分条件。举一个简单的例子就可说明这一点,图 6-20 表示的是带有一个悬面的立方体,其顶点数(V)为 10,边数(E)为 15,面数(F)为 7,符合欧拉公式,但却不是一个有效实体。为了检查一个三维物体是不是有效实体,还需要附加一些条件:如每一条边必须连接两个点;

图 6-19

一条边被两个面、且仅被两个面所共享; 至少要有三条边交于一个顶点等。

对于非简单多面体, 欧拉公式是否成立呢? 让我们先来看一个例子。图 6-21 表示出一个立方体具有一个贯穿的方孔和未贯穿的方孔, 这时, V= 24, E= 36, F= 15。显然, V- E + F≠2。这就是说, 欧拉公式不适用于非简单多面体。但是, 对欧拉公式加以扩展, 就可以适用于图 6-21 的这种非简单多面体了。

图 6-20

图 6-21

如令 H 表示多面体表面上孔的个数, G 表示贯穿多面体的孔的个数, C 表示独立的、不相连接的多面体数, 则扩展后的欧拉公式为

$$V - E + F - H = 2(C - G)$$

对于图 6-21 所示的非简单多面体, 扩展后的欧拉公式是适用的。与前面一样, 扩展后的欧拉公式仍然只是检查实体有效性的必要条件, 而不是充分条件。

最后还要说明一点, 欧拉公式不仅适用于由平面多边形组成的多面体, 也适用于由曲面片组成的多面体, 但必须与球是拓扑等价的, 如图 6-22 所示。

图 6-22

2. 边界表示的数据结构

用实体的边界来表示实体,在计算机内是如何实现的呢?这就是边界表示的数据结构问题,我们仍以平面多面体为例来讨论这一问题。

边界表示的概念不仅能用来表示三维空间的平面多面体,而且可延伸到表示一个面、一条边。即一个平面多面体可用构成边界的一系列平面多边形表示,一个多边形又可用构成其边界的一系列边来表示,而一条边又可用两个点来表示。那么,如何在计算机中将面、边、点信息合理地组织起来以有效地满足实体的构造、运算及显示的需要呢?

要用实体的边界信息表示一个实体,必须正确地表示出实体边界的拓扑信息及几何信息。所谓拓扑信息,指的是面、边、点之间的连接关系、邻近关系及边界关系,而几何信息则指的是面、边、点的位置及大小等几何数据。由于使用者要频繁地对实体的面、边、点进行查找或修改,并且希望尽快地知道这些操作的影响及结果,因此,所设计的数据结构是否便于对实体进行面、边、点的存放,查找或修改,是一个十分关键的问题,必须妥善的加以解决。

可是,这并不是一个很容易解决的问题。如果我们简单地用一系列的顶点来表示一个多边形,即

$$P = \{(x_1, y_1, z_1), (x_2, y_2, z_2), \cdots, (x_n, y_n, z_n)\}$$

而顶点坐标是按沿多边形边界的顺序存放的,相邻两顶点之间代表一条边。那么,对于一个简单的多边形,这种表示方法是节约空间的。但是,对于一个多面体边界上的一系列多边形来说,这种表示方法使用了过多的空间,这是由于被多个多边形共享的顶点多次重复存储的缘故。此外,这种表示方法没有表示出共享边及共享点的信息,如果我们想通过交互方式拖曳一个顶点及共享这个顶点的所有的边,那么我们必须找到共享这个顶点的全部多边形,这就需要将一个多边形的顶点坐标与所有其它多边形的顶点坐标相比较,显然,这将使用过多的计算时间。

如图 6-23 所示,从原理上来说,实体的面、边、点之间共有 9 种不同类型的拓扑关系。

图 6-23

即 V→{V}, V→{E}, V→{F}, E→{V}, E→{E}, E→{F}, F→{V}, F→{E}, F→{F}。在这里符号“→”表示指针,即可以从它的左端求出它的右端。例如, V→{E}表示由一个顶点找出相交于此顶点的所有边,而 F→{E}则表示由一个面找出该面所有的边,如此等等。这 9 种不同类型的拓扑关系也可以用图 6-24 来表示,图中,由结点 X 指向结点 Y 的带箭头的弧,表示一个查找过程,即“给定 X,求出 Y”。

显而易见,在这 9 种不同类型的拓扑关系中,至少必须选择两种才能构成表示一个实体所需的完全的拓扑信息。因此,可能采取的数据结构形式,按所给定的拓扑关系来分,共有 C_9^2, C_9^3, …, C_9^9 等 8 类,共计 502 种($C_9^2+C_9^3+\cdots+C_9^9=502$)。如果将此 8 类数据结构形式表示为 C_9^m(m= 2, 3, …, 9),那么,当 m 小时,所存储的拓扑关系少,因此所需的存储空间小,但查找时间较长。当 m 大时,所存储的拓扑关系多,因而所需的存储空间大,但查找时间较短。因此,要根据对实体的操作所提的要求及计算机系统资源的状况,妥善选择一种数据结构,求得时间和空间上的合理折衷,以提高整个系统的效率。

在国际、国内研究开发的三维实体造型系统中,采用得较多的是翼边数据结构及对称数据结构或者它们的变形。现简述如下:

(1) 翼边数据结构

翼边数据结构最早是由美国斯坦福大学的 B·G·波姆嘎特(B. G. Baumgart)等人提出来的。其结构如图 6-25 所示。这种结构以每一条边为核心,分别指向我们从外面观察这一多面体时所见到的这条边的上下两个顶点,左右两个邻面以及上下左右四条邻边。每一个顶点都有一个指针,反过来指向以该顶点为端点的某一条边。每一个面也有一个指针,反过来指向它的一条边。通过翼边结构,可以方便地查找各元素之间的连接关系。但翼边结构存储的信息量大,存储内容重复,但却获得了较高的查找速度。

(2) 对称数据结构

图 6-24　　图 6-25　　图 6-26

对称数据结构的拓扑关系如图 6-26 所示。显然,在这种数据结构中,显式地存放了 {Fi}→{Ei}, {Ei}→{Fi}, {Ei}→{Vi}及{Vi}→{Ei}4 种拓扑关系。这种对称数据结构的具体实现可如图 6-27 所示,其中 VGi 表示围绕点 Vi 的所有边。与此类似,FGi 表示围绕一个面 Fi 的所有边。可以证明,这种数据结构的空间复杂性近似为 6E, E 为该实体所有的

边数。

图 6-27

从上面的数据结构可以看出,用边界表示实体时,数据量较多,所使用的存储空间较大。但是,有一个突出的优点,那就是实体的面、环、边、点的信息都直接表示出来了,因而,集合运算的结果可以继续参加集合运算,而且也便于显示或绘图输出。因此,这种方法广泛地用来表示三维实体。

四、扫描表示法

这也是一个得到广泛应用的表示三维体的方法。它的基本原理是很简单的,即空间中的一个点,一条边或一个面沿着某一路径扫描时,所形成的轨迹将定义一个一维的、二维的或三维的物体。在这里,表示或形成一个物体需要两个要素,其一是作扫描运动的物体,其二是扫描运动的轨迹。在三维形体的表示中,应用得最多的是平移扫描体和旋转扫描体两种。

图 6-28 中的几个例子说明了平移扫描体和旋转扫描体的原理。在图 6-28(a)中,扫描体是一条曲线,扫描轨迹是一条直线,该曲线沿着直线扫描的结果得到一个曲面。在图 6-28(c)中,扫描体是一个平面,平面上有四个孔,扫描轨迹是与该平面相垂直的一个直线段,当该平面沿着该直线段扫描时,得到一个具有四个孔的三维实体。这是两个平移扫描的实体。这两个例子具有一个共同的特点,即扫描得到的物体都有相同的截面,这是简单的平移扫描法的特点。

图 6-28(b)中,扫描体是一条曲线,它绕一个旋转轴作旋转扫描,得到的是一个曲面。

图　6-28

在图 6-28(d)中,扫描体是一个平面,它绕一个旋转轴作旋转扫描,得到一个三维实体。这是两个由旋转扫描法构成物体的例子,都是轴对称的物体,这是旋转扫描法的特点。另外,需要说明的是,旋转轴本身并不一定是所构造的物体的一部分。

如果在扫描过程中,允许扫描体的截面随着扫描过程变化,那我们就可以得到不等截面的平移扫描体和非轴对称的旋转扫描体,这种方法统称为广义扫描法,图 6-29 所示为用不等截面的圆盘作平移扫描构成的圆台。

图　6-29

在扫描表示法中,由于三维空间的实体和曲面可分别由二维平面及曲线通过平移扫描或旋转扫描来实现,因此,只需定义二维平面及曲线即可。这对于许多领域的工程设计人员来说都是很方便的。例如,建筑设计师们就是先设计建筑物的平面图,然后通过平移扫描构造建筑物的模型的。因此,在三维物体的表示中,扫描表示法往往是必不可少的输入手段,应用颇为广泛。

五、构造的实体几何法

构造的实体几何法(Constructive Solid Geometry),简称 CSG 法,是在实体的表示、构造中得到广泛应用的一种方法。它首先是由美国的 H·B·沃尔克尔及 A·A·G·雷

契切等人提出来的。它的基本思想是将简单的实体(又称体素)通过集合运算组合成所需要的物体。因此,这种方法称为构造的实体几何法。在这一方法中,常用的体素有长方体、圆柱体、圆锥体、圆台体、环、球等。在某些功能较强的系统中,还可通过扫描表示法产生一些实体,这些实体也作为体素在 CSG 法中应用。

在构造的实体几何法中,集合运算的实现过程可以用一棵二叉树(称为 CSG 树)来描述。如图 6-30 所示,二叉树的叶子结点表示体素或者几何变换的参数,其非终端结点表示施加于其子结点的正则集合算子(正则并、正则交和正则差)或几何变换的定义。显然,二叉树根结点所表示的就是集合运算的最终结果。如果这一方法中所采用的体素是正则的,而所采用的集合运算算子也是正则的,那么,所得出的结果也必然是正则几何形体,即有效的实体。

图 6-30

在构造的实体几何法中,体素也可以用半空间的集合运算组成。一个无边界的面可将三维空间分割成两个无边界的区域,每一个区域均称为半空间。一组半空间经过交运算后即可构成一个封闭的三维实体。于是,一个复杂的物体可以定义为一系列有向面求交以后再求并所得的结果,即

$$F = \bigcup_{j=1}^{m}\left(\bigcap_{i=1}^{n} f_{ij}\right)$$

在构造的实体几何法中,如果体素设置比较齐全,那么,通过集合运算,可以构造出多种不同的符合需要的实体。覆盖域比较宽是这种方法的优点,此外,用户可以通过输入或调用简单体素来直观而方便地构造三维实体。因而,这种方法在现有的大多数实体造型系统中作为输入手段而得到广泛应用。

这一方法也有它的局限性。当用户输入体素时,主要是给定体素的有关参数(如长方体的长、宽、高,圆柱的底面圆心、半径及高等),然后由系统给出该体素的表面方程,当进行求交运算时,可以通过表面方程求交,但是集合运算的中间结果却难于再用简单的代数

方程表示,因而不能继续参与集合运算。此外,这种表示方法用于输出也很不方便。因此,单纯使用结构的实体几何法存在着难于克服的困难,为了解决这一问题,往往在用结构的实体几何法实现实体的定义和输入以后,将其转换为边界表示,再进行集合运算和显示输出,这是实体造型系统中常用的一种方法。

六、八叉树表示法

八叉树表示法是占有空间计数法的一种,占有空间计数法将实体所在空间进行分割,一般是分割成由立方体组成的网格。于是,一个实体可以由它所占用的立方体序列来表示。当分割后的立方体越来越小时,就逐步接近用空间点的集合来表示实体了。

八叉树表示法是一种层次结构的占有空间计数法,它是由图象处理中的四叉树法扩展而来的。如图 6-31 所示,我们可将实体所在空间用一个立方体(一般称为宇宙)来表示。

图 6-31

如果该立方体完全被实体所占有,那么该立方体可表示为“满”,如果该立方体与实体完全不相交,则该立方体表示为“空”。如果该实体占有立方体的部分空间,那么就将该立方体等分为 8 个小立方体,图 6-31(a)就是这样一种情况,等分后的 8 个小立方体,可按一定规律给以编号(0, 1, 2, …, 7)(如图 6-31(b)所示),然后再按上述规则进行检查,确定为“满”、“空”或再次 8 等分。如此进行下去,直到全部小立方体均为“满”、“空”,或者是设备

分辨率允许的最小立方体为止。因此,这是一种递归分割的过程。

这种递归分割的实体表示形式,可用八叉树结构加以描述。如图 6-31(c)表示,树的根结点表示实体所在空间,即“宇宙”。其 8 个子结点中,黑结点表示“满”的子空间,白结点表示“空”的子空间,需要再次分割的子空间用圆形结点表示,形成一棵八叉树,那么,最多可能递归分割到多少层呢?假如显示此物体的图形显示器的分辨率为 1024×1024(即 $2^{10}\times 2^{10}$)若八叉树的根为第 0 层,那么递归分割 10 次以后,在第 11 层就得到最大可能的空间阵列,共有 $2^{10}\times 2^{10}\times 2^{10}$ 个结点。如果分割到这一象素级精度以后,仍然出现部分小立方体被实体占有的情况,那么可以按照统一的规则将此小立方体设置为“满”或“空”,相应的叶子结点为黑或白。因此,八叉树只是空间实体的一个近似表示。

用八叉树结构表示空间实体,具有许多优越性,概括说起来有如下一些。

(1) 可以用统一而简单的形体(即立方体)来表示空间任意形状的实体,因而,数据结构简单划一。

(2) 易于实现实体之间的集合运算,如交、并、差等。

(3) 易于检查空间实体之间是否碰撞,计算出两个实体之间的最小距离也不困难。

(4) 易于计算物体的性质,如物体的体积、质量、重量、转动惯量等。

(5) 由于各小立方体在数据结构中总是排好序的,故易于实现消隐及显示输出。

用八叉树表示空间实体,也有如下一些不足之处:

(1) 八叉树表示所需存储容量较大。D·J·R·米格(D. J. R. Meagher)曾分析过八叉树结构对存储容量的需求。假如设备分辨率为 1024×1024,并且平均说来,在八叉树的每一层中,只有 4 个结点需要再次分割为 8 个子结点,那么八叉树中需要存储的全部结点数为 2.7×10^6 个,这当然是一个相当大的数字。但是,采取一些改进办法,例如,只存黑结点,不存白结点,根据需要确定分割精度,而不一定分割至象素级等,都可以大大减少对存储空间的需求。另一方面,当今的计算机或工作站的存储空间不断扩充,所以,这并非是一个不可克服的困难。

(2) 在八叉树表示中,难于实现在其它表示形式下易于实现的某些几何变换,如旋转任意角度,具有任意比例系数的比例变换等。

(3) 八叉树表示只是空间实体的近似表示,将八叉树表示转换为精确的边界表示是非常困难的,因而难于用这种表示的结果实现绘图输出。这是它的主要弱点。

由于八叉树表示存在着如上的不足之处,因而限制了它的应用范围。在实体造型系统中,一般将它用作一种辅助表示形式。此外,由于八叉树表示具有层次结构的特性,因而常用来管理空间实体,以实现图形显示过程的加速。

习　　题

1. 设一 Coons 曲面片的系数矩阵为

$$B_x=\begin{matrix}0&10&16&0\\0&18&24&0\\0&8&8&0\\0&8&8&0\end{matrix}\qquad B_y=\begin{matrix}0&0&0&0\\10&10&0&0\\10&10&0&0\\10&10&0&0\end{matrix}\qquad B_z=\begin{matrix}10&0&0&-16\\8&0&0&-14\\-2&0&0&2\\-2&0&0&2\end{matrix}$$

试分析该曲面片的特点。

2. 试计算出上述曲面片中，参数值为 u= 0.5, w= 0 及 u= 0.5, w= 1 的点的坐标值。
3. 试编写一个算法，计算出 Coons 曲面片上具有参数值 u 及 w 的点的坐标值。

   该算法表示为 PSRF(CI, U, W, P)

   其中，CI(4, 4, 3)是定义此曲面片的系数矩阵；

   U 是参数 u 的数值；

   W 是参数 w 的数值；

   P(3)是该点的坐标值。
4. 试编写一算法，计算出 Coons 曲面片上某一点处的单位法矢。该算法表示为 UNV(CI, UW, N)。

   其中，CI(4, 4, 3)是定义此曲面片的系数矩阵；

   UW(2)　是某一点处的参数值 u 及 w；

   N(3)　是该点处的单位法矢。
5. 试编写一算法，计算出 Bezier 曲面片上某一点处的坐标值。该算法表示为 BZSRF(M, N, CI, IFLG, U, W, P)。

   其中，

   M 是沿 u 为常数的曲线上的控制点数。

   M∈[3, 6]；

   M 是沿 w 为常数的曲线上的控制点数。

   N∈[3, 6]；

   CI(M, N, 3)是定义特征多边形的控制顶点的坐标输入矩阵；

   IFLG 定义曲面是开曲面或闭曲面的数学；

   如为开曲面，则 IFLG= 0；

   如 u 为常数的曲线是闭合的，则 IFLG= 1；

   如 w 为常数的曲线是闭合的，则 IFLG= 2；

   U, W 是参数 u, w 的输入值；

   P(3)是该点的坐标值 x, y, z；

图　6-32

6. 如果我们要构造 2×2 个双三次 B 样条曲面片组，问需要多少个控制点？如要构造 3×3 及 4×4 的双三次 B 样条曲面片组呢？

7. 图 6-32 表示出二维图形 A, B 的相互位置, 按本章讲授的公式求出 $A \cup^* B$, $A \cap^* B$, $A -^* B$ 及 $B -^* A$, 并以图表示之。
8. 形体的拓扑信息和几何信息各包含哪些内容? 各起什么作用? 举例说明之。
9. 试写出判断一点 P 是否在一个多边形 Q 内, 多边形 Q 外或者在多边形 Q 的边上的算法。
10. 试写出判断一条线段 P_1P_2 是否在一个多面体内的算法。
11. 试写出判定空间任意位置的两个长方体是否相交的算法。
12. 试写出用八叉树表示一个多面体的算法。
13. 试写出定义平面多边形后, 经平移和旋转扫描产生三维实体的算法。
14. 试以圆柱体为例, 写出将它的 CSG 表示转换成边界表示的算法。
15. 试比较实体的边界表示, 扫描表示、CSG 表示及八叉树表示的优缺点。并说明各适用于什么应用场合?

第七章　三维形体输出流水线

在第六章，我们介绍了三维形体的表示方法。这一章，我们将介绍三维空间中的投影变换及空间转换。

首先要介绍的是平面几何投影技术。大家知道，用来显示图形的输出设备都是二维的，而被输出的是三维形体。这样，在三维图形输出过程中，或者说在三维输出流水线中，就需要将三维坐标表示的几何形体变换成二维坐标表示的图形。这一过程就是投影变换。然后，针对二维图形，用前面介绍的二维输出技术在输出设备上显示出来。

观察空间的定义和转换将使三维输出过程规范化。在这一章里，我们将介绍如何定义观察坐标系、裁剪空间、规范化投影空间、图象空间，以及如何经过一系列空间转换将三维用户坐标系下的几何形体转换到图象空间，最后用正投影产生二维图形。

整个三维形体的输出流水线如上页图所示。

与二维图形的裁剪一样，三维形体也需要裁剪。在上述输出流水线中，裁剪是在投影之前进行的，因而是三维裁剪；如果安排在投影之后，则变成了二维裁剪。

三维裁剪也可以在规范化投影空间中进行。裁剪后的形体再向图象空间进行转换时，可以节省转换时间。但在规范化投影空间进行三维裁剪要比在图象空间进行裁剪慢得多。

第一节　平面几何投影

一、投影与投影变换的定义

投影是将 n 维的点变换成小于 n 维的点。本节将专门讨论三维到二维的平面几何投影。

在三维空间中，选择一个点，记这个点为投影中心；不经过这个点再定义一个平面，记这个平面为投影面；从投影中心向投影面引任意多条射线，记这些射线为投影线。穿过物体的投影面将与投影面相交，在投影面上形成物体的像，这个像记为三维物体在二维投影面上的投影。像这样将三维空间中的物体变换到二维平面上的过程称为投影变换。图7-1表示了同一条直线段 AB 的两种不同的投影。由于直线的平面投影本身仍是一条直线，所以，对直线段 AB 作投影变换时，只需对线段的两个端点 A 和 B 作投影变换，连接两个端点在投影面上的投影 A 和 B 就可得到整个直线段的投影 A Ｂ ′。

图 7-1　线段 AB 及其投影

二、平面几何投影的分类

平面几何投影(以下简称投影)可分为两大类，即透视投影和平行投影。它们的本质区别在于透视投影的投影中心到投影面之间的距离是有限的，而平行投影的投影中心到投影面之间的距离是无限的。图 7-1 表示了这两类投影。当投影中心在无限远时，投影线互

相平行,所以定义平行投影时,给出投影线的方向就可以了,而定义透视投影时,需要明确地指定投影中心的位置。

图 7-2 显示了各类投影之间的逻辑关系,它们共同特点是有一个投影面和一个投影中心或投影方向。分类是根据投影中心到投影面的距离、投影线方向与投影面的夹角、投影面与坐标轴的夹角来进行的。下面将详细介绍各种投影的定义、特性和数学计算方法。

图 7-2　投影的分类

三、透视投影

如图 7-3,设投影中心在坐标原点,投影面与 Z 轴垂直,在 $z=d$ 的位置上。点 $P(x,y,z)$ 在投影面上的投影点为 $P(x_p,y_p,d)$。现欲计算 x_p,y_p 的值。

根据相似三角形对应边成比例的关系,由图 7-3(b)和(c)得:

$$\frac{x_p}{d}=\frac{x}{z},\quad \frac{y_p}{d}=\frac{y}{z}$$

于是,

$$x_p=\frac{x}{z/d}$$
$$y_p=\frac{y}{z/d} \tag{7-1}$$

设透视变换矩阵为 T_{per},则用齐次坐标技术可将上述关系表示为:

$$[X_p,Y_p,Z_p,H]=(x,y,z,1)T_{per}$$
$$=(x,y,z,z/d) \tag{7-2}$$

这里 $H=z/d$。于是

图 7-3　透视投影

$$x_p = X_p / H = \frac{x_p}{z/d}$$

$$y_p = Y_p / H = \frac{y_p}{z/d}$$

$$z_p = Z_p / H = d$$

与(7-1)式一致。

由(7-2)式的最后一个等式，可写出

$$T_{per} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/d \\ 0 & 0 & 0 & 0 \end{matrix} \qquad (7\text{-}3)$$

下面讨论透视投影的特性。由(7-1)式可以看出，距离 d 仅仅是作用于 x_p 和 y_p 的比例因子，而以 z 除却说明了远处的物体比近处的物体的投影要小，即物体透视投影的大小与物体到投影中心的距离成反比，这就是所谓的透视缩小效应。这种效应所产生的视觉效果十分类似于照相系统和人的视觉系统。图 7-4 中，等长的两直线段，都平行于投影面，但离投影中心近的线段，其透视投影大，而离投影中心远的线段，其透视投影小。与下面将要详细介绍的平行投影相比，透视投影的深度感更强，看上去更加真实，但透视投影不能真实地反映物体的精确尺寸和形状。

对于透视投影，一束平行于投影面的平行线的投影可保持平行，而不平行于投影面的平行线的投影会汇聚到一个点，这个点称为灭点(图 7-5)。灭点可以看作是无限远处的一点在投影面上的投影。

透视投影的灭点有无限多个，不同方向的平行线在投影面上就能形成不同的灭点。坐标轴方向的平行线在投影面上形成的灭点又称作主灭点。因为有 X、Y 和 Z 三个坐标轴，

图 7-4　平行于投影面的等长的两线段,离投影中心近的线段的透视投影大

图 7-5　平行于投影面的平行线 AB 和 CD 的透视投影仍保持平行,不平行于投影面的平行线 AC 和 BD 的透视投影汇聚于一个灭点

所以主灭点最多有三个。当某个坐标轴与投影面平行时,则该坐标轴方向的平行线在投影面上的投影仍保持平行,不形成灭点。

透视投影是按主灭点的个数来分类的,即按投影面与坐标轴的夹角来分类的。可分为一点透视、两点透视和三点透视。一点透视有一个主灭点,即投影面与一个坐标轴正交,与另外两个坐标轴平行。图 7-6 显示了一个方块的两种不同的一点透视。(7-1) 式给出的计

图 7-6　一个方块的一点透视投影

算透视投影的公式仅适合于这种一点透视。两点透视有两个主灭点，即投影面与两个坐标轴相交，与另一个坐标轴平行。图 7-7 显示了一个方块的两点透视。三点透视有三个主灭点，即投影面与三个坐标轴都相交。三点透视用得不多，主要原因是它难于构造。

图 7-7　一个方块的两点透视投影图

四、平行投影

平行投影不具有透视缩小性，能精确地反映物体的实际尺寸。平行线的平行投影仍是平行线。平行投影可根据投影方向与投影面的夹角分成两类：正投影和斜投影。当投影方向与投影面的夹角为 90 时，得到的投影为正投影，否则为斜投影（图 7-8）。

图 7-8　平行投影

1. 正投影

正投影根据投影面与坐标轴的夹角又可分成两类：三视图和正轴测，当投影面与某一坐标轴垂直时，得到的投影为三视图，这时投影方向与这个坐标轴的方向一致。否则，得到的投影为正轴测（图 7-9）。

三视图有主视图、侧视图和俯视图三种，投影面分别与 X 轴、Y 轴和 Z 轴垂直。图 7-10 显示了一个小房子的三视图。

三视图的投影变换矩阵分别为：

图 7-9　正投影　　图 7-10　一个小房子及其三视图

主视图

$$T_{front}=\begin{bmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

侧视图

$$T_{side}=\begin{bmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}$$

俯视图

$$T_{top}=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&1\end{bmatrix}$$

三视图常用于工程制图,因为在三视图上可以测量距离和角度。但一种三视图上只有物体一个面的投影,所以单独从某一个方向的三视图上是很难想象出物体的三维形状的,只有将主、侧、俯三个视图放在一起,才有可能综合出物体的空间形状。

正轴测有等轴测、正二测和正三测三种。当投影面与三个坐标轴之间的夹角都相等时为等轴测;当投影面与两个坐标轴之间的夹角相等时为正二测;当投影面与三个坐标轴

之间的夹角都不相等时为正三测(图 7-11)。

下面我们推导正轴测的投影变换矩阵。如图 7-12(a),投影面分别与三个坐标轴交于 A、B 和 C,投影方向与投影面垂直。首先把物体及投影面绕 Y 轴顺时针旋转 φ角,得图 7-12(b);再绕 X 轴逆时针旋转 θ角,得图 7-12(c)。

图 7-11　正轴测及一个方块的正轴测投影图

图 7-12　正轴测投影变换矩阵计算示意图

$$\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi\sin\theta & \sin\phi\cos\theta & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ -\sin\phi & -\cos\phi\sin\theta & \cos\phi\cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

最后,在 Z 轴方向上作正投影,就可得到正轴测的投影变换矩阵:

$$T_{axono}=\begin{bmatrix}\cos\phi & -\sin\phi\cos\theta & 0 & 0\\ 0 & \cos\theta & 0 & 0\\ -\sin\phi & -\cos\phi\sin\theta & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{7-4}$$

等轴测的条件是投影面与三个轴方向的夹角都相等,于是在图 7-12(a)上,有

$$OA=OB=OC$$

$$\phi=45^\circ,\quad \sin\phi=\cos\phi=\frac{\sqrt{2}}{2}$$

$$BD=\sqrt{OD^2+OB^2}$$

$$=\frac{\sqrt{6}}{2}OB$$

$$\sin\theta=\frac{OD}{BD}=\frac{\frac{\sqrt{2}}{2}OA}{\frac{\sqrt{6}}{2}OB}=\frac{\sqrt{3}}{3}$$

$$\cos\theta=\frac{OB}{BD}=\frac{\sqrt{6}}{3}$$

代入(7-4)式得等轴测的投影变换矩阵

$$T_{iso}=\begin{bmatrix}\frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} & 0 & 0\\ 0 & \frac{\sqrt{6}}{3} & 0 & 0\\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

正二测的条件是投影与某两个坐标轴之间的夹角相等。例如设投影面与 X 和 Z 轴之间的夹角相等,则在图 7-12(a)上有

$$OA=OC$$

$$\phi=45^\circ\quad \cos\phi=\sin\phi=\frac{\sqrt{2}}{2}$$

代入(7-4)式得正二测的投影变换矩阵为

$$T_{dim}=\begin{bmatrix}\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\sin\theta & 0 & 0\\ 0 & \cos\theta & 0 & 0\\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2}\sin\theta & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

正三测的投影变换矩阵就是(7-4)式。

由于正轴测的投影面不与任何坐标轴垂直所以正轴测能同时反映物体的多个面,具有一定的立体效果,而且平行线仍保持平行,沿三个主轴方向都可测量距离。但值得注意的是等轴测在三个主轴方向的距离因子相等,而正二测只在两个主轴方向的距离因子相等,正三测在三个主轴方向的距离因子都不相等。

2. 斜投影

斜投影是第二类的平行投影,与正投影的区别在于投影方向与投影面不垂直。斜投影将正投影的三视图和正轴测的特性结合起来,既能像三视图那样在主平面上进行距离和角度测量,又能像正轴测那样同时反映物体的多个面,具有立体效果。通常选择投影面垂直于某个主轴,这样,对平行于投影面的物体表面可进行距离和角度的测量,而对物体的其它面,可沿这根主轴测量距离。

常用的两种斜投影是斜等测和斜二测。当投影方向与投影面成 45°角时,得到的是斜等测(图 7-13(a)),这时,和投影面垂直的任何直线段,其投影的长度不变,即图 7-13(a)中,AB= A′B′。当投影方向与投影面成 arctg(2)的角度时,得到的是斜二测(图 7-3(b)),这时,和投影面垂直的任何直线,其投影的长度为原来的一半,即图 7-13(b)中,AB= 2A′B′。

图 7-13　斜投影

下面我们给出斜投影的投影变换矩阵。如图 7-14,设投影面为 XY 平面,投影方向与投影面的夹角为 α,投影线和 Z 轴所组成的平面与 XZ 面的两面角为 β。点 P′(x_p', y_p', 0)是点 P(0, 0, z)在投影面上的斜投影。于是

$$k = z\operatorname{ctg}\alpha$$

$$x_p' = k\cos\beta$$

$$y_p' = k\sin\beta$$

对于空间任意一点 Q(x, y, z),在投影面上的斜投影的坐标可从图 7-15 直接得出:

$$x_p = x_p' + x = z\operatorname{ctg}\alpha\cos\beta + x$$

$$y_p = y_p' + y = z\operatorname{ctg}\alpha\sin\beta + y$$

于是,斜投影的投影变换矩阵为

$$T_{obl}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ \operatorname{ctg}\alpha\cos\beta & \operatorname{ctg}\alpha\sin\beta & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

对于斜等测，$\alpha=45^{\circ}$，$\operatorname{ctg}\alpha=1$。对于斜二测，$\alpha=\operatorname{arctg}(2)$，$\operatorname{ctg}\alpha=\frac{1}{2}$。

图 7-14　P(0, 0, z)的斜投影

图 7-15　空间任意一点的斜投影

图 7-16 表示的是一个单位立方体在 XY 平面上的几种斜投影。那些倾斜线是与 XY

图 7-16　一个方块的斜投影

平面垂直的立方体棱边的投影，它们与水平轴 X 的夹角就是图 7-14 中的两面角 β。β一般取 45 和±30°。

第二节　观察空间的定义及转换

一、观察空间

在显示物体的过程中，我们不仅要像上一节介绍的那样定义一个投影平面和一个投影中心或投影方向，将三维物体投影到二维投影面上，还要在投影之前对这个物体进行裁剪，把物体上我们不关心的部分去掉，留下我们感兴趣的那部分，并将这部分投影到投影面上显示出来。这就需要我们在用户坐标系中指定一个观察空间，将这个观察空间以外的物体裁剪掉，只对落在这个空间内的物体作投影变换并予以显示。

1. 观察坐标系

图 7-17　在用户坐标系 OXYZ 中，用点 VRP 和向量 VPN 定义观察平面

如图 7-17 所示，在用户坐标系 OXYZ 中，我们用一个点和一个向量定义一个平面。这个平面叫做观察平面，就是我们前面提到的投影面。这个点叫做观察参考点，记作 VRP。这个向量叫做观察平面法向，记作 VPN。

再如图 7-18 所示，定义三维观察坐标系 VRC。令 VRP 为观察坐标系的原点，VPN 为观察坐标系的一个坐标轴，记为 n 轴。观察坐标系的第二个坐标轴叫做 v 轴，是观察正向 VUP 在观察平面上的正投影。观察正向 VUP 是定义在用户坐标系中的一个向量，用来确定观察坐标系的 v 轴。确定了 n 和 v 两个坐标轴之后，确定第三个坐标轴 u 就容易了，只要保证 u 与 n 和 v 正交且 u、v 和 n 构成三维左手坐标系就可以了。坐标轴 u 和 v 总是在观察平面上。这样用一个点 VRP 和两个向量 VPN、VUP 就可以在用户坐标系中定义一个观察坐标系。

2. 观察空间

(1) 我们先在观察平面上指定一个观察窗口。观察窗口的作用与二维图形变换中窗口的作用相似。窗口内的部分将被映射到视图区，而窗口外的部分将不被显示。如图 7-19 所示，在观察平面上用 (u_{min}, v_{min}) 和 (u_{max}, v_{max}) 定义观察窗口的左下角点和右上角点。图中，CW 为窗口的中心。从图上我们可以看到，相对于观察参考点 VRP，窗口没有必要是对称的。VRP 可以在窗口内、窗口外，也可以在窗口的边框上。

(2) 再看透视投影的投影中心和平行投影的投影方向，它们是由投影参考点 PRP 和投影类型的两个参数来确定的。当投影类型为透视投影时，投影中心 COP 为投影参考点 PRP。当投影类型为平行投影时，投影方向 DOP 从投影参考点 PRP 指向窗口中心 CW (图 7-20)。

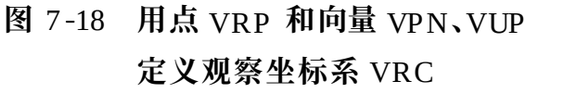

图 7-18　用点 VRP 和向量 VPN、VUP 定义观察坐标系 VRC

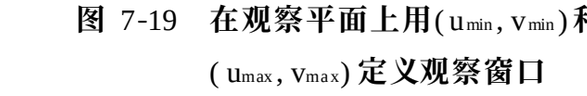

图 7-19　在观察平面上用(u_{min}, v_{min})和(u_{max}, v_{max})定义观察窗口

图 7-20　观察空间的定义

投影参考点 PRP 的坐标是在观察坐标系中定义的,而不是在用户坐标系中定义的。这样,在用户坐标系中改变定义观察坐标系的参数 VRP、VPN 和 VUP 不会影响投影参考点 PRP 相对于观察参考点 VRP 的位置。

对于透视投影,观察空间是顶点在观察参考点 PRP、其棱边穿过观察窗口四个角点、没有底面的四棱锥(图 7-20(a))。而对于平行投影,观察空间是一个四边平行于投影方向 DOP、两端没有底面的长型方管(图 7-20(b))。

(3) 大多数场合,我们希望观察空间是有限的。图 7-21 显示了如何用前截面和后截面将无限的观察空间截成有限的观察空间。前后截面都平行于观察平面,即前后截面的法向就是观察平面的法向,前后截面的位置分别由前截面距离 F 和后截面距离 B 确定。F 和 B 分别是从观察参考点 VRP 沿观察平面法向 VPN 到前截面的距离和后截面的距离。当截面在观察参考点的正向时,此距离为正;在观察参考点的负向时此距离为负。图 7-21 中,F 为负值,B 为正值。用观察平面法向 VPN 和前后截面的距离 F 和 B 可以唯一地确定有限观察空间的前截面和后截面。对于透视投影,前截面必须在投影中心 PRP 和后截面之间。对于平行投影,前截面必须在负无穷远和后截面之间。有限观察空间又叫裁剪空间,因为物体是相对于这个空间进行裁剪的。

3. 观察空间的参数

表 7.1 列出了定义观察空间的参数、名称、对应的英文名称、参数类型、默认值及其作用。这些参数可以分成两类。VRP、VPN 和 VUP 属于第一类,它用于确定观察坐标系,从

图 7-21　裁剪空间

而可以将用户坐标系中的点转换到观察坐标系中,即将(x, y, z)变换成(u, v, n)。另一类参数用来在观察坐标系中定义裁剪空间,由 PRP、u_{min}、u_{max}、v_{min}, v_{max}和 F、B 组成。在 PHIGS 图形标准中,这两类参数由两个 4× 4 的矩阵来表示。表 7-1 中的默认值显示在图 7-22(a)中,相应的裁剪空间显示在图 7-22(b)和(c)中。

表 7-1　观察空间的参数

参数	名　称	英　文　名　称	类　型	默认值	作　　用
VRP	观察参考点	view reference point	用户坐标系中的三维点	(0, 0, 0)	确定观察坐标系原点
VPN	观察平面法向	view- plane normal	用户坐标系中的向量	(0, 0, - 1)	确定观察平面法向
VUP	观察正向	view up vector	用户坐标系中的向量	(0, 1, 0)	确定观察平面上 v 轴的方向
F	前截面距离	front distance	实数	- 1	确定前截面的位置
B	后截面距离	back distance	实数	0	确定后截面的位置
pt	投影类型	projection type			定义投影是平行投影还是透视投影
PRP	投影参考点	projection reference point	观察坐标系中的三维点	(0. 5, 0. 5, - 0. 5)	确定投影中心或投影方向
u_{min}, u_{max} v_{min}, v_{max}	观察窗口	view window	观察坐标系中的二维点	0, 1 0, 1	在观察平面上定义观察窗口

二、例子

为了进一步加深对观察空间各参数的理解,这一小节将以图 7-23 所示的小房子为

图 7-22

例,专门讨论如何定义空间参数来生成各种投影图。对于下面所讨论的每一个投影图,我们将给出 VRP、VPN、VUP、PRP、窗口和投影类型这些相应的参数,并用(WC)和(VRC)表明这些参数是用户坐标系的参数,还是观察坐标系的参数。

1. 透视投影

先看图 7-24(a)所示的房子的一点透视图。为了得到这个透视图,我们将投影中心设在(8, 6, 84)处,(这里,8 是房子水平面中心的 X 坐标,6 差不多是站在 XZ 平面上人眼的高度,84 是任意取的 Z 值)。窗口很大,能保证房子落在观察空间中。其它参数还是表 7-1 中列出的默认值。这样,整个定义观察空间的参数值如下:

VRP(WC)	(0, 0, 0)
VPN(WC)	(0, 0, -1)
VUP(WC)	(0, 1, 0)
PRP(VRC)	(8, 6, -84)
窗口(VRC)	(-50, 50, -50, 50)
投影类型	透视投影

图 7-23 小房子

图 7-24

图 7-24(a)中的图形的确是房子的透视投影,但太小,而且不在视图区中心。由于房子在 X、Y 向的范围都是 0 到 16,所以如果使观察平面与房子的正面重合,将窗口的 u 和 v 都

改为- 1 到 17, 就可以得到如图 7-24(b)所示的投影更正、能充满视图区的图形。

把 VRP 放在房子的前左下角(0, 0, 54)处, 就可以使观察平面与房子的正面重合。投影中心 PRP 仍保持不变, 由于它是在观察坐标系中定义的, 所以它的坐标这时应是(8, 6, - 30)。图 7-25 显示了这一新的安排, 对应的参数如下:

VRP(WC)	(0, 0, 54)
VPN(WC)	(0, 0, - 1)
VUP(WC)	(0, 1, 0)
PRP(VRC)	(8, 6, - 30)
窗口(VRC)	(- 1, 17, - 1, 17)
投影类型	透视投影

图 7-25 图 7-24(b)的观察环境

接下来, 让我们生成一个图 7-26 所示的房子的二点透视图。首先我们将投影中心从 Z 轴向房子的右边和上边移一点, 定在(36, 25, 74)的地方, 这样可以看到房子的顶部和侧面。然后将房子的一角(16, 0, 54)选作观察参考点 VRP。观察平面仍是房子的正面, 即 Z = 54 的平面, 相应地, 投影中心在观察坐标系中的坐标变为(20, 25, - 20)。窗口的范围在 u 方向定为(- 20, 20), 在 v 方向定为(- 5, 35)。即:

VRP(WC)	(16, 0, 54)
VPN(WC)	(0, 0, - 1)
VUP(WC)	(0, 1, 0)
PRP(VRC)	(20, 25, - 20)
窗口(VRC)	(- 20, 20, - 5, 35)
投影类型	透视投影

用上面的参数得到的是图 7-27 所示的透视图。这个视图与图 7-26 所示的透视图很相似, 但它仍是一点透视, 因为观察平面只与一个坐标轴相交。很显然, 要旋转观察平面, 使其与 X、Z 两个坐标轴都相交才能得到房子的两点透视图。图 7-28 显示了这一情况, 这里, 观察平面的法向 VPN 为(- 1, 0, - 1)。这时, 如果投影中心仍定在(36, 25, 74)处, 则其在新的观察坐标系下的坐标为(0, 25, - 20 $\sqrt{2}$)。

图 7-26 小房子的两点透视

图 7-27 VPN 平行于 Z 轴时小房子的透视图

图 7-29 显示了如何计算投影中心在新的观察坐标系下的坐标,以及如何确定窗口在 u 方向的范围。

图 7-28 对应图 7-26 的观察坐标系

图 7-29 房子的俯视图,用来计算投影中心和确定窗口的大小

生成图 7-26 所示的二点透视图的观察参数如下:

VRP(WC)	(16, 0, 54)
VPN(WC)	(-1, 0, -1)
VUP(WC)	(0, 1, 0)
PRP(WC)	$(0, 25, -20\ \overline{2})$
窗口	(-20, 20, -5, 35)
投影类型	透视投影

图 7-30 显示了 VUP 改变后的效果。图 7-30 与图 7-26 很相似,但房子有点右倾。在前面的例子中,观察坐标系的 v 轴始终与用户坐标系的 Y 轴保持平行,即 VUP=(0, 1, 0),所以窗口的侧边与房子的侧边保持平行。图 7-30 的观察参数与图 7-26 的观察参数完全相同,只是观察正向 VUP 从 Y 轴旋转开了 10°。

2. 平行投影

图 7-31 是房子正面的正平行投影,投影方向与 Z 轴平行。由于投影方向是由投影参

图 7-30 VUP 旋转后房子的二点透视图

图 7-31 房子的正平行投影

考点 PRP 和窗口中 CW 确定的,所以当窗口为(-1,17,-1,17),其它的参数为默认值时,投影参考点应为(8,8,-100)。图 7-32 显示了这一观察环境,对应的观察参数如下:

VRP(WC)	(0,0,0)
VPN(WC)	(0,0,-1)
VUP(WC)	(0,1,0)
PRP(WC)	(8,8,-100)
窗口(VRC)	(-1,17,-1,17)
投影类型	平行投影

图 7-33 是房子的等轴测图,投影方向为(-1,-1,-1),观察参数如下:

图 7-32 图 7-31 的观察环境

图 7-33 房子的等轴测图

VRP(WC)	(8,8,42)
VPN(WC)	(-1,-1,-1)
VUP(WC)	(0,1,0)
PRP(WC)	(0,0,-10)
窗口(VRC)	(-20,20,-20,20)
投影类型	平行投影

图 7-34 是房子的斜等测投影图,观察参数为:

VRP(WC)	(8,8,54)	房子正面的中心
VPN(WC)	(0,0,-1)	Z 轴
VUP(WC)	(0,1,0)	Y 轴
PRP(WC)	($\cos\beta$, $\sin\beta$, -1)	
窗口(VRC)	(-15,15,-15,15)	
投影类型	平行投影	

由于窗口中心 CW 是(0,0,0),所以投影方向为

$$CW - PRP = -PRP = (-\cos\beta, -\sin\beta, 1)。$$

3. 有限观察空间

在上述所有的例子中,都假定观察空间是无限的,没有考虑前后截面这两个参数。图 7-35 显示的房子没有后墙,由下列观察参数确定:

VRP(WC)	(0, 0, 54)	房子正面左下角
VPN(WC)	(0, 0, - 1)	Z 轴
VUP(WC)	(0, 1, 0)	Y 轴
PRP(VRC)	(8, 6, - 30)	
窗口	(- 1, 17, - 1, 17)	
投影类型	透视投影	
F	- 1	在 z= 54+ 1= 55 处
B	+ 23	在 z= 54- 23= 31 处

图 7-34　房子的斜等测投影图

图 7-35　后截面在 z= 31 时房子的透视图

这些参数与图 7-24(b)的参数相同,只是加上了前截面和后截面而已。后截面将房子的后墙给截掉了。

三、空间转换

空间转换包括从用户坐标系到观察坐标系、从裁剪空间到规范化投影空间、从规范化投影空间到规范化图象空间这三个转换。在用户坐标系下定义的物体,经过空间转换转换到规范化图象空间之后,可以直接作正平行投影,投影到观察平面,再映射到视图区就可以显示出来。

1. 用户坐标系到观察坐标系的转换

前面,我们介绍了在用户坐标系 OXYZ 中如何用参数 VRP、VPN 和 VUP 唯一地定义一个观察坐标系 VRC。这里,我们记观察参考点 VRP 的坐标为(x_R、y_R、z_R),观察平面法向 VPN 为(x_n、y_n、z_n),观察正向 VUP 为(x_a、y_a、z_a),取观察平面法向 VPN 上的单位向量 n 和观察正向 VUP 上的单位向量 a,则

$$n = \frac{VPN}{|VPN|} = \frac{x_n}{\sqrt{x_n^2 + y_n^2 + z_n^2}}i + \frac{y_n}{\sqrt{x_n^2 + y_n^2 + z_n^2}}j + \frac{z_n}{\sqrt{x_n^2 + y_n^2 + z_n^2}}k \quad (7\text{-}5)$$

$$a = \frac{VUP}{|VUP|} = \frac{x_a}{\sqrt{x_a^2 + y_a^2 + z_a^2}}i + \frac{y_a}{\sqrt{x_a^2 + y_a^2 + z_a^2}}j + \frac{z_a}{\sqrt{x_a^2 + y_a^2 + z_a^2}}k \quad (7\text{-}6)$$

n 和 a 分别记为(n_x, n_y, n_z)和(a_x, a_y, a_z)。于是观察坐标系的 u 轴在用户坐标系下的单位向量为:

$$u = n \times a$$

$$= (n_y a_z - n_z a_y)\mathbf{i} + (n_z a_x - n_x a_z)\mathbf{j} + (n_x a_y - n_y a_x)\mathbf{k} \quad (7\text{-}7)$$

记作(u_x, u_y, u_z)。v 轴在用户坐标系下的单位向量为:

$$v = u \times n$$

$$= (u_y n_z - u_z n_y)\mathbf{i} + (u_z n_x - u_x n_z)\mathbf{j} + (u_x n_y - u_y n_x)\mathbf{k} \quad (7\text{-}8)$$

记作(v_x, v_y, v_z)。

将用户坐标系转换成观察坐标系分以下两步进行。首先,将用户坐标系 OXYZ 平移,使原点 O 与观察坐标系 VRC 的原点 VRP 重合,形成 O′X′Y′Z′ 坐标系(图 7-36)。平移矩阵为:

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_R & -y_R & -z_R & 1 \end{bmatrix} \quad (7\text{-}9)$$

图 7-36　坐标系原点 O 平移到 VRP

这时,如果(x, y, z)是原坐标系下的一个点的坐标,则这个点在新坐标系下的坐标(x', y', z')为:

$$(x', y', z', 1) = (x, y, z, 1)T_1$$

$$= (x - x_R, y - y_R, z - z_R, 1)$$

接下来,将 O′X′Y′Z′ 坐标系转换成观察坐标系 VRC。设 O′X′Y′Z′ 坐标系下有一位置矢量 p(x, y, z),该矢量在 VRC 坐标系下的坐标为(u, v, n),这是 p 在 u、v 和 n 轴上的投影,用 p 与单位矢量 u、v 和 n 的点积可表示这种投影关系,即:

$$u = p \cdot u = xu_x + yu_y + zu_z,$$

$$v = p \cdot v = xv_x + yv_y + zv_z,$$

$$n = p \cdot n = xn_x + yn_y + zn_z。$$

也就是,

$$(u, v, n, 1) = (x, y, z, 1)T_2$$

这里,

$$T_2 = \begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7\text{-}10)$$

是将 O′X′Y′Z′ 转换为 VRC 的转换矩阵。

综合以上两步,从用户坐标系到观察坐标系的转换矩阵可表示为:

$$T_{wv} = T_1 T_2 \quad (7\text{-}11)$$

经过 T_{wv}的转换,用户坐标系下定义的物体将被转换到观察坐标系。例如,对于图 7-26 所示的房子的二点透视,其观察坐标系参数为:

VRP　　(16, 0, 54)

VPN　　(-1, 0, -1)

VUP (0, 1, 0)

即 $(x_R, y_R, z_R) = (16, 0, 54)$,

$(x_n, y_n, z_n) = (-1, 0, -1)$,

$(x_a, y_a, z_a) = (0, 1, 0)$。

于是,由式(7-5)得, $n = (n_x, n_y, n_z) = -\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}$,

由式(7-7), $u = (u_x, u_y, u_z)$

$$= \frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}},$$

由式(7-8), $v = (v_x, v_y, v_z) = (0, 1, 0)$,

由式(7-9), $T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -16 & 0 & -54 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

由式(7-10), $T_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

由式(7-11), $T_{wv} = T_1 T_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \\ 19\sqrt{2} & 0 & 35\sqrt{2} & 1 \end{bmatrix}$

T_{wv}就是我们所求的从用户坐标系到观察坐标系的转换矩阵。用这个转换矩阵,我们可以将用户坐标系中的各点转换到观察坐标系中。例如,投影参考点 PRP,在用户坐标系下的齐次坐标为(36, 25, 74, 1),则在观察坐标系下的齐次坐标可由下式求出:

$$(36, 25, 74, 1) T_{wv} = (0, 25, -20\sqrt{2}, 1)。$$

小房子的角点在观察坐标系下的齐次坐标如下:

$$(0, 10, 54, 1) T_{wv} = (-8\sqrt{2}, 10, 8\sqrt{2}, 1),$$

$$(16, 0, 54, 1) T_{wv} = (0, 0, 0, 1),$$

$$(8, 16, 30, 1) T_{wv} = (8\sqrt{2}, 16, 16\sqrt{2}, 1),$$

$$(16, 10, 30, 1) T_{wv} = (12\sqrt{2}, 10, 12\sqrt{2}, 1),$$

$$(16, 0, 30, 1) T_{wv} = (12\sqrt{2}, 0, 12\sqrt{2}, 1)。$$

2. 裁剪空间到规范化投影空间的转换

在观察坐标系中,透视投影的裁剪空间如图 7-21(a)所示,平行投影的裁剪空间如图

7-21(b)所示。其中,窗口为(u_{min}, u_{max}, v_{min}, v_{max}),前截面距离为F,后截面距离为B,投影参考点PRP的坐标为(u_p, v_p, n_p)。裁剪空间到规范化投影空间的转换是将这个裁剪空间转换成如图7-37所示的规范化投影空间。

图 7-37　规范化投影空间

裁剪空间到规范化投影空间的转换由以下三步组成。

第一步,将观察坐标系VRC的原点由VRP平移到投影参考点PRP(比较图7-21和图7-38)。平移矩阵为:

$$T_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -u_p & -v_p & -n_p & 1 \end{bmatrix} \tag{7-12}$$

图 7-38　坐标系原点由VRP移到PRP

经过平移,窗口左下角的坐标变为($u_{min}-u_p$, $v_{min}-v_p$, $-n_p$),右上角的坐标变为($u_{max}-u_p$, $v_{max}-v_p$, n_p)。

第二步,对坐标系进行错切变换,使投影参考点和窗口中心 CW 的连线移到 n 轴,即变成图 7-39 所示的坐标系。

图 7-39　坐标系错切,使 CW 在 n 轴上

在图 7-38 中,窗口中心 CW 的坐标为:

$$\begin{aligned} u_{cw} &= \frac{1}{2}(u_{max} + u_{min}) - u_p, \\ v_{cw} &= \frac{1}{2}(v_{max} + v_{min}) - v_p, \\ n_{cw} &= -n_p。 \end{aligned} \tag{7-13}$$

比较图 7-38 和图 7-39 可以看出,错切是在 u 和 v 方向进行的,且都是关于 n 方向的错切。沿 u 方向的错切分量为 $-\frac{u_{cw}}{n_{cw}}$,沿 v 方向的错切分量为 $-\frac{v_{cw}}{n_{cw}}$。于是,变换矩阵为:

$$T_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{u_{cw}}{n_{cw}} & -\frac{v_{cw}}{n_{cw}} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7-14}$$

第三步,在 n 轴方向进行比例变换,将后截面变成 n= 1 的平面,在 u 和 v 轴方向也进行比较变换,使后截面介于 $-1 \leq u, v \leq 1$ 的范围内,即变成图 7-40 所示的空间。

在这一步,平行投影和透视投影应分别对待。对于平行投影,先将前截面平移到 uv 平面上,再进行比例变换。两个变换矩阵为:

$$T_{5par} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -(F - n_p) & 1 \end{bmatrix} \tag{7-15}$$

图 7-40　经比例变换后的观察空间

$$T_{6par}=\begin{bmatrix}\dfrac{2}{u_{max}-u_{min}} & 0 & 0 & 0\\ 0 & \dfrac{2}{v_{max}-v_{min}} & 0 & 0\\ 0 & 0 & \dfrac{1}{B-F} & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{7-16}$$

对于透视投影,比例变换矩阵为:

$$T_{5per}=\begin{bmatrix}\dfrac{2}{u_{max}-u_{min}}\cdot\dfrac{-n_p}{B-n_p} & 0 & 0 & 0\\ 0 & \dfrac{2}{v_{max}-v_{min}}\cdot\dfrac{-n_p}{B-n_p} & 0 & 0\\ 0 & 0 & \dfrac{1}{B-n_p} & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{7-17}$$

综合上述三步,对于平行投影,从裁剪空间到规范化投影空间的转换矩阵为:

$$T_{vcper}=T_3T_4T_{5par}T_{6par} \tag{7-18}$$

对于透视投影,从裁剪空间到规范化投影空间的转换矩阵为:

$$T_{vcper}=T_3T_4T_{5per} \tag{7-19}$$

3. 从规范化投影空间到图象空间的转换

所谓规范化图象空间,是如图 7-37(b)所示的正四棱柱,满足$-1\le u,v\le 1,0\le n\le 1$。

由于平行投影的规范化投影空间与规范化图象空间是一样的,所以我们只需要求出透视投影的规范化投影空间(图 7-37(a))到规范化图象空间的转换矩阵。

将图 7-37(a)所示的四棱台变成图 7-37(b)所示的四棱柱,直观地看,这个变换中必包括透视变换,且灭点在 $n>0$ 的方向上(图 7-41)。因为只有这样,才能使四棱台中 4 条汇

聚于原点的棱边变成棱柱中 4 条平行于 n 轴的边。设此变换矩阵为:

$$T_{6per} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & m & n \\ 0 & 0 & s & t \end{matrix} \tag{7-20}$$

图 7-41　向规范化图象空间转换时灭点在 n 轴上

则空间中的任意一点 p(x, y, z)经过 T_{6per}作用后将变为 p′(x′, y′, z′), 即

$$(X', Y', Z', H) = (x, y, z, 1)T_{6per}$$

于是,

$$x' = X'/H = \frac{x}{nz + t}$$

$$y' = Y'/H = \frac{y}{nz + t} \tag{7-21}$$

$$z' = Z'/H = \frac{mz + s}{nz + t}$$

设 f 是规范化投影空间中前截面到坐标系原点 PRP 之间的距离, 于是由式(7-12)和(7-17)得:

$$f = \frac{F - n_p}{B - n_p}$$

前截面的角点(x, y, z) = (±f, ±f, f)将被变为(x′, y′, z′) = (±1, ±1, 0), 代入式(7-21), 得:

$$\frac{\pm f}{nf + t} = \pm 1$$

$$\frac{mf + s}{nf + t} = 0 \tag{7-22}$$

后截面的角点(x, y, z) = (±1, ±1, 1)将被变为(x′, y′, z′) = (±1, ±1, 1), 代入式(7-21), 得:

$$\frac{\pm 1}{n + t} = \pm 1$$

$$\frac{m + s}{n + t} = 1 \tag{7-23}$$

联合式(7-22)和式(7-23),可解得:

$$m = \frac{1}{1-f}$$

$$n = 1$$

$$s = -\frac{f}{1-f}$$

$$t = 0$$

这样,从规范化的投影空间到规范化图象空间的变换矩阵为:

$$T_{6per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{1-f} & 1 \\ 0 & 0 & -\frac{f}{1-f} & 0 \end{bmatrix} \tag{7-24}$$

第三节　三维裁剪

为了去掉物体中不需要显示的部分,在任意裁剪空间变换到规范化投影空间之后,我们对物体要进行三维裁剪。对于透视投影,裁剪是相对于如图 7-37(a)所示的规范化投影空间进行的,这是一个正四棱台。对于平行投影,裁剪是相对于如图 7-37(b)所示的规范化投影空间进行的,这是一个正四棱柱。二维图形的裁剪一节中介绍的编码裁剪法和中点分割裁剪法都可以推广到三维裁剪。除此而外,还有其它很多三维裁剪方法。本节将介绍三维编码裁剪法。

三维编码裁剪法是将空间直线的端点,相对于裁剪空间的 6 个面,转换为 6 位二进制代码,然后采用简单的逻辑判断,舍弃裁剪空间以外的部分,接受裁剪空间内的部分,必要时与裁面求交,分段处理。

对于透视投影,其规范化投影空间是正四棱台,6 个面的方程为:

$$\begin{aligned} y &= z, \quad y = -z \\ x &= z, \quad x = -z \\ z &= 1, \quad z = f \end{aligned} \tag{7-25}$$

空间中任一线段的端点,相对于这 6 个面都可以赋予一个 6 位的二进制代码。设最左边的位是第一位,则这个二进制代码各位的含义如下:

第一位为 1,表示端点在裁剪空间之上,即 $y > z$;

第二位为 1,表示端点在裁剪空间之下,即 $y < -z$;

第三位为 1,表示端点在裁剪空间之右,即 $x > z$;

第四位为 1,表示端点在裁剪空间之左,即 $z < -z$;

第五位为 1,表示端点在裁剪空间之后,即 $z > 1$;

第六位为 1,表示端点在裁剪空间之前,即 $z < f$;

对于平行投影,其规范化投影空间是正四棱柱,6 个面的方程为:

$$y = 1, \quad y = -1$$
$$x = 1, \quad x = -1 \qquad (7\text{-}26)$$
$$z = 1, \quad z = 0$$

相对于这 6 个面,空间线段端点的位置可用二进制代码表示,其含义是:

第一位为 1,表示端点在裁剪空间的上方,即 $y > 1$;

第二位为 1,表示端点在裁剪空间的下方,即 $y < -1$;

第三位为 1,表示端点在裁剪空间的右方,即 $x > 1$;

第四位为 1,表示端点在裁剪空间的左方,即 $x < -1$;

第五位为 1,表示端点在裁剪空间的后边,即 $z > 1$;

第六位为 1,表示端点在裁剪空间的前边,即 $z < 0$。

跟二维编码裁剪法一样,如果线段两端点的 6 位二进制编码都为 0,则该线段一定完全落在裁剪空间之内,应予以接受。若两端点的编码按位求逻辑与后为非零,则该线段一定完全落在裁剪空间之外,可以舍弃;否则,该线段一定与裁剪空间的某个面相交。对于这种相交的情况,需要计算线段与裁剪空间 6 个面的交点,用求出的交点对线段进行再分割。然后再求出交点的 6 位二进制编码,重复上述判断,直至所有的直线段均被接受或者舍弃为止。

空间线段与裁剪面的求交运算,可以按下述方法进行。

设线段 L 的两端点分别为 $P_1(x_1, y_1, z_1)$ 和 $P_2(x_2, y_2, z_2)$,空间中任一平面 Q 的方程为:

$$ax + by + cz + d = 0$$

则 L 与 Q 的交点(x, y, z)应满足:

$$\begin{aligned} x &= x_1 + pt \\ y &= y_1 + qt \\ z &= z_1 + rt \\ ax &+ by + cz + d = 0 \end{aligned} \qquad (7\text{-}27)$$

其中

$$p = x_2 - x_1, \quad q = y_2 - y_1, \quad r = z_2 - z_1$$

由式(7-27)可解得:

$$t = -\frac{ax_1 + by_1 + cz_1 + d}{ap + bq + cr}$$

上式中,若 $ap + bq + cr = 0$,则说明直线 L 在平面 Q 上或与平面 Q 平行;若 $t \notin [0, 1]$,则说明交点不在线段上,所以交点无效;若 $t \in [0, 1]$,则将 t 代入式(7-27)的前三个等式,就可求出线段 L 与平面 Q 的交点。

对于透视投影,平面 Q 是式(7-25)中的任何一个,对于平行投影,平面 Q 是式(7-26)中的任何一个。这些都是简单的平面方程,在计算 t 时可以减少很多乘法。另外,由于这些平面方程很简单,在生成线段端点的 6 位二进制编码时,只用 6 个简单的比较就可以了。这就是为什么三维裁剪相对于规范化投影空间进行,而不是相对于任意裁剪空间进行的原因。

习　　题

1. 已知投影面为 XOY 面，投影中心在 Z 轴的正向、Z = d 的位置上，求透视投影变换矩阵。
2. 已知单位立方体，一个顶点为原点 O，与之相对的顶点为 A(1, 1, 1)，若设 OA 为投影方向，投影面与投影方向垂直，问这种投影是何种投影？请写出投影变换矩阵。
3. 投影面为 XOY 面，将(x, y, z)点投影到(lcos α lsin α 0)的斜投影矩阵是什么？
4. 写出各投影的特征。
5. 编一个输出流水线程序，将用户坐标系下的点转换到图象空间。程序的输入参数为表 7.1 中的观察空间参数。
6. 将规范化投影空间转换为图象空间的意义何在？
7. 将二维中点分割裁剪法推广到三维裁剪。请编出三维中点分割裁剪算法的程序。

第八章　消隐及真实感图形生成技术

有三种表现三维图形的方式。一种是线框图,图上的线条为形体的棱边。另一种是消隐图,图上只保留了形体上看得见的部分,看不见或被遮挡掉的部分就不画出来或用虚线表示。还有一种是真实感图形,能够表现形体的光照效果。下面是各种图形的输出过程。

本章将介绍消隐技术及真实感图形生成技术。

第一节　消隐技术

一、基本概念

在用户坐标系下定义的物体,经过前面介绍的空间转换、裁剪,可以投影到观察平面,再变换或映射到视图区并显示出来。像这样显示出的图形是物体的线框图。图 8-1(a)所示的图形是一个立方体的线框图。

图 8-1　一个立方体的线框图和消隐图

线框图通过物体的棱边和轮廓线表示这个物体。但我们很难确定,图 8-1(a)所示的立方体是代表(b)中的立方体呢,还是代表(c)中的立方体?可见,在显示三维物体时,我们不仅要决定物体上每条棱边在视图区中的位置,而且要决定哪些棱边是可见的,必须显示,哪些棱边被遮挡了,不可见,必须抹掉或用虚线表示。找出并消除物体中不可见的部分,称为消隐。经过消隐得到的图形称为消隐图。图 8-1(b)和(c)都是立方体的消隐图。

消隐的对象是三维物体。三维物体在计算机中的表示方法很多,最简单的表示方法是用其表面上的平面多边形表示。如果物体的表面是曲面,则可以用多个平面多边形作近似表示。多边形又可以用其边界上的边来表示,边又可进一步用其两个端点来表示。这样由点到边、由边到面、由面到体,形成三维物体的体面边点四表结构(参见三维物体的表示一节)。根据消隐对象的不同,消隐算法可分为两类。若消除的是物体上不可见的线段,即棱边,称为线消隐;若消除的是物体上不可见的面,称为面消隐。

消隐不仅与消隐对象有关,还与观察物体的方式,即投影方式有关。图 8-1(b)和(c)都是一个立方体的消隐图,其区别就在于观察这个立方体的观察点不同。随着观察点、观察方向、或投影面的改变,物体上某些可见的部分将会变成不可见,某些不可见的部分又会变成可见。如图 8-2,当视点在 V_1 时,物体上 A、B 两面可见,C、D 两面不可见。而把视点从 V_1 移到 V_2 后,A 面由可见变成不可见,C 面由不可见变为可见。所以在本节我们讨论的几个消隐算法中,都假定物体经过空间转换,转换到了如图 7-37(b)所示的规格化投影空间,或者进一步转换到了屏幕坐标空间。消隐就是在规范化投影空间或屏幕坐标空间中进行的。根据消隐空间的不同,消隐算法又可分为两类:

图 8-2 观察方式的改变影响面的可见性

(1) 物体空间的消隐算法。

物体空间是指物体所在的空间,即规范化投影空间。这类算法是将物体表面上的 k 个多边形中的每一个面与其余的 k- 1 个面进行比较,精确地求出物体上每条棱边或每个面的遮挡关系。显然,这类算法的计算量正比于 k^2。

(2) 图象空间的消隐算法。

图象空间是相对于物体空间而言的,在消隐算法中,图象空间就是上述的屏幕坐标空间(注意与第七章中图象空间的区别)。这类算法对屏幕的每一象素进行判断,以决定物体上哪个多边形在该象素点上是可见的。若屏幕上有 m× n 个象素点,物体表面上有 k 个多边形,则该类消隐算法的计算量将正比于 mnk。

虽然消隐算法的种类繁多,但它们必然都涉及排序和相关性这两个基本原则。

排序是为了确定消隐对象之间的遮挡关系。通常在 X、Y、Z 三个方向上都要进行排序。一般说来,先对哪个坐标排序不影响消隐算法的效率,但大多数消隐算法都是先在 Z 方向排序,确定体、面、边、点相对于观察点的距离,因为一个物体离观察点越远,越有可能被离观察点近的物体所遮挡(图 8-3(a))。但图 8-3(b)表明,并不是所有离观察点远的对

象都会被离观察点近的对象所遮挡。所以，在 Z 方向排序之后，还要在 X 方向和 Y 方向进一步排序，以确定这种遮挡关系。消隐算法的效率在很大程度上取决于排序的效率。

图 8-3　物体间的遮挡关系

在消隐算法中利用相关性是提高排序效率的一种重要手段。所谓相关性，是指所考察的物体或视图区内的图象局部保持不变的一种性质。对于相关性，我们并不陌生，在前面介绍多边形区域填充的扫描转换算法时，我们就用到了边的相关性和扫描线相关性、可以说，相关性利用得越充分、越巧妙，消隐算法的效率也就越高。

二、深度缓存算法

深度缓存算法(Z- buffer 算法)是一种典型的、也是最简单的图象空间面消隐算法。它需要一个深度缓存数组 ZB，数组的大小与屏幕上象素点的个数相同，也与显示器的帧缓存 FB 的单元个数相同，彼此一一对应。

如图 8-4，在屏幕坐标系中，Z 轴的方向为观察方向。过屏幕上任一象素点(i，j)作平行于 Z 轴的射线 R，与物体表面上的多边形相交于 p_1 点和 p_2 点。我们称 p_1 和 p_2 为多边形平面上对应象素(i，j)的点，p_1 和 p_2 的 Z 值称为该点的深度值。

图 8-4　射线 R 与物体相交于 p_1、p_2 点

深度缓存消隐算法比较 p_1 和 p_2 的 Z 值,将最小的 Z 值存入深度缓存数组 ZB,最小 Z 值所对应的多边形的颜色存入显示器的帧缓存 FB。帧缓存 FB 为每一个象素点保存应显示的颜色。由于屏幕上每一象素点所对应的多边形平面上点的深度值都要存入 ZB 数组,所以,若屏幕上象素点的个数为 $m\times n$,则 ZB 数组的大小也应为 $m\times n$。

设有 N 个多边形,则算法的基本步骤如下:

1. 初始化 ZB 和 FB,使

   ZB(i,j)= 机器最大值,

   FB(i,j)= 背景色。

   其中,i= 1,2,…,m, j= 1,2,…,n。

2. FOR j= 1,n, /* 第 j 根扫描线* /

   FOR i= 1,m, /* 第 j 根扫描线上第 i 个象素点* /

   FOR k= 1,N, /* 多边形 P_k* /

   2.1 令 Z_{ij}= 机器最大值;

   2.2 判断点(i,j)是否落在多边形 P_k 在 XOY 面上的投影多边形内;

   2.3 若(i,j)在 P_k 的投影多边形内,则计算多边形 P_k 在点(i,j)处的深度值 $Z_{i,j}$;

   2.4 比较 Z_{ij} 与 ZB(i,j)的大小,若 $Z_{ij}<$ ZB(i,j),则令

   ZB(i,j)= Z_{ij},

   FB(i,j)= 多边形 P_k 的颜色。

算法结束后,显示器帧缓存 FB 中存放的就是消隐后的图象。由于显示器的扫描系统实时地扫描帧缓存 FB,并将其内容实时地显示出来,所以在深度缓存消隐程序运行过程中,可以从显示器的屏幕上看到消隐的全过程。程序运行结束后,从屏幕上可以看到物体的消隐图。

实现上述算法还要解决如下两个问题:

1. 判断点(i,j)是否落在多边形 P_k 在 XOY 平面上的投影多边形 P_k 内;
2. 计算多边形 P_k 在点(i,j)处的深度值 Z_{ij}。

判断点(i,j)是否在多边形 P_k 内的问题,实际就是多边形对点的包含性检查的问题。这在多边形扫描转换一节中已经解决,可采用射线法或弧长法。

计算多边形 P_k 在(i,j)处的深度值,方法如下:

设多边形 P_k 的平面方程为:

$$ax + by + cz + d = 0$$

若 $c\neq 0$,则把(i,j)代入面方程,就可得深度值 Z_{ij},

$$Z_{ij} = -\frac{ai + bj + d}{c} \qquad (8\text{-}1)$$

若 c= 0,则说明多边形 P_k 的法向与 Z 轴垂直,P_k 在 XOY 的投影为一条直线,在算法中可以不考虑这种多边形。

深度缓存消隐算法的最大优点就在于简单。它在 X、Y、Z 方向上都没有进行任何排序,也没有利用任何相关性。算法复杂性正比于 $m\times n\times N$。在屏幕大小,即 $m\times n$ 一定的

情况下,算法的计算量只与多边形个数 N 成正比。

另一个优点是算法便于硬件实现。实际上,在当今生产的大多数高档图形工作站上,除了帧缓存外,还带有用于消隐的深度缓存,从而克服了深度缓存算法占用大量存储单元的缺点。

在深度缓存算法中,屏幕上哪个象素点的颜色先计算,哪个后计算,其先后顺序是无关紧要的,不影响消隐结果。正因为如此,所以它不仅无需预先排序,从而省去各个方向的排序时间,更重要的是它可以并行化。最直接的并行化方法就是将屏幕均匀地划分成多个更小的屏幕。对于这些小屏幕,用多个处理机同时进行消隐。

三、扫描线算法

这里介绍 Z- buffer 扫描线算法,它是直接从深度缓存算法中导出的。在深度缓存算法中,没有用到任何相关性。事实上,在算法的两个关键部分,即判断点(i, j)是否在投影多边形 P_k 内,和计算多边形 P_k 在点(i, j)处的深度值时,都可以利用边的相关性和点的相关性。这一节,我们先讨论如何利用相关性来提高判断点(i, j)是否在多边形内和计算多边形 P_k 在点(i, j)处的深度值的速度,然后导出扫描线消隐算法。

1. 计算投影多边形 P_k 的边 L 与扫描线 y = j 的交点 x_j。

如图 8-5,设 P_k 的边 L 的方程为:

$$px + qy + r = 0$$

则在 y= j 这根扫描线上,边 L 与扫描线的交点的 X 坐标为:

$$x_j = -\frac{qj + r}{p}$$

这里,p ≠0。若 p= 0,则说边 L 与水平轴平行。

图 8-5 边的相关性示意图

在 y= j+ 1 这根扫描上,交点的 X 坐标为:

$$x_{j+1} = \frac{q(j+1) + r}{p} = -\frac{qj + r}{p} - \frac{q}{p}$$

$$= x_j - \frac{q}{p} = x_j - \Delta x \qquad (8\text{-}2)$$

式中 $\Delta x = \frac{q}{p}$ 为边 L 在相邻两扫描之间 X 坐标之差。上式表明,将边与前一条扫描线的交点 x_j 简单地做一次减法,就可以得到与下一条扫描线的交点 x_{j+1}。

2. 计算多边形在其边界上的深度值

设 L 为多边形 P_k 的一条边,与扫描线 j 的交点为 x_j,则多边形 P_k 在交点(x_j, j)处的深度为:

$$Z_j = -\frac{ax_j + bj + d}{c}$$

式中,a、b、c、d 为多边形 P_k 的面方程系数,且 c≠ 0。若 c= 0,则说明多边形 P_k 面与屏幕 XOY 面相垂直。

当扫描线向上移动时，即 $y=j+1$ 时，边 L 与扫描线的交点坐标为 $(x_{j+1}, j+1)$，多边形 P_k 在此交点处的深度为

$$
\begin{aligned}
Z_{j+1} &= -\frac{ax_{j+1}+b(j+1)+d}{c} \\
&= -\frac{a\left(x_j-\frac{q}{p}\right)+b(j+1)+d}{c} \\
&= -\frac{ax_j+bj+d}{c}+\frac{a}{c}\times\frac{q}{p}-\frac{b}{c} \\
&= Z_j+\Delta Z_x\times\Delta x-\Delta Z_y
\end{aligned}
\tag{8-3}
$$

式中，ΔZ_x 为多边形 P_k 在 x 方向的增值，ΔZ_y 为多边形 P_k 在 y 方向的增值。上式表明，根据多边形在边与扫描线 j 的交点处的深度值 Z_j，用一次乘法运算和两次加减运算，就可以得到多边形在该边与下一条扫描线 $j+1$ 的交点处的深度值 Z_{j+1}。

3. 计算多边形 P_k 在点 (i, j) 处的深度

如图 8-6，(i, j, Z_{ij}) 为多边形 P_k 内的一点，其中，Z_{ij} 为多边形 P_k 在点 (i, j) 处的深度，则，

$$
Z_{ij}=-\frac{ai+bj+d}{c},\quad c\neq 0
$$

而在点 $(i+1, j)$ 处，多边形 P_k 的深度为：

$$
\begin{aligned}
Z_{i+1,j} &= -\frac{a(i+1)+bj+d}{c} \\
&= -\frac{ai+bj+d}{c}-\frac{a}{c} \\
&= Z_{ij}-\Delta Z_x
\end{aligned}
\tag{8-4}
$$

图 8-6　扫描线相关性示意图

上式表明，在求出了多边形 P_k 在 (i, j) 处的深度值 Z_{ij} 之后，用一个减法就可以得到这个多边形在同一条扫描线 j 上相邻后续点 $(i+1, j)$ 处的深度值 $Z_{i+1,j}$。

4. 判断点 (i, j) 是否在投影多边形 P_k 内

在扫描线 $y=j$ 上，计算 P_k 的每条边与扫描线的交点。如图 8-7 所示，扫描线 j 与多边形有交点 p_1、p_2、p_3 和 p_4，而且，凡是 $p_1<x<p_2$、$p_3<x<p_4$ 的点，都在 P_k 内的区间段，其端点所在的两条边称之为边对。图中，点 p_1、p_2 所在的边 e_1 和 e_2 组成一个边对，p_3 和 p_4 所在的边 e_3 和 e_4 组成一个边对。e_1 为边对 e_1e_2 中的左侧边，e_2 为边对 e_1e_2 中的右侧边。

而在扫描线 $y=j+1$ 上，边对 e_1e_2、e_3e_4 与扫描线交点之间的区间段，仍在多边形 P_k' 内。所以，对于多边形上的任何一条边 e，只要它不是水平的，总能找到与之组成边对的边，从边 e 的最低点（即 y 值较小的端点）开始，到最高点（即 y 值较大的端点）为止，每一条扫描线与边对相交形成的区间段都在多边形内。如图 8-8，阴影部分表示了边 e_6 先与 e_5 组成的边对、然后又跟 e_2 组成的边对，与扫描线相交后在多边形内形成的区间段。

5. 数据结构

为了有效地利用上述相关性，我们在扫描线消隐算法中，建立如下数据结构：

(1) 多边形 Y 桶

图 8-7　多边形与扫描线相交形成边对

图 8-8　边对之间的区域在多边形内

图 8-10 是图 8-9 的多边形 Y 桶。桶的长度与扫描线的数目一样多。根据多边形顶点中最小的 Y 坐标,确定多边形在多边形 Y 桶中的位置。对于桶中的多边形,要记录其番号和多边形顶点中最大的 Y 坐标,根据多边形番号可以从定义多边形的数据表中取出多边形平面的平面方程系数、多边形的边、多边形的顶点坐标、多边形的颜色等属性。

图 8-9　消隐对象

(2) 有效多边形表 APT

APT 中记录与当前扫描线相交的各个多边形的信息,包括多边形番号、多边形顶点中最大的 Y 坐标,指向边 Y 桶的指针。图 8-11 表示了扫描线为 y= 10 时图 8-9 的有效多边形表。

(3) 边 Y 桶

有效多边形表中的每一个多边形,都有一个边 Y 桶。图 8-12 是图 8-9 中多边形 B 的边 Y 桶。边 Y 桶的长度等于与多边形相交的扫描线数。例如多边形 B 的最大 Y 坐标值为 12,最小 Y 坐标值为 4,则与其相交的扫描线数为 12- 4+ 1= 9。根据边的两端点中较小的 y 值与多边形顶点中最大的 y 值之差,确定边在边 Y 桶中的位置。例如,边 e_1 的 y 值较小的端点的 Y 坐标为 4,多边形 B 的最大 y 值为 12,则边 e_1 在边 Y 桶中的位置为 12- 4= 8。

图 8-10　多边形 Y 桶

图 8-11　有效多边形表

图 8-12　边 Y 桶

对于桶中的边,要记录其端点中较大的 y 值、式(8-2)中的 Δx、以及 y 值较小的那个端点的 X 坐标和 Z 坐标。

(4) 有效边形 AET

AET 中,记录多边形边界与当前扫描线相交的各个边对的信息。包括:

x_l: 左侧边与扫描线的交点的 X 坐标,

Δx_l: 左侧边按式(8-2)求出的 Δx 值,

y_{lmax}: 左侧边两端点中最大的 y 值,

x_r: 右侧边与扫描线的交点的 X 坐标,

Δx_r: 右侧边按式(8-2)求出的 Δx 值,

y_{rmax}: 右侧边两端点中最大的 y 值,

Z_l: 左侧边与扫描线的交点处多边形的深度,

ΔZ_x: 对应式(8-3),边对所在的多边形的 ΔZ_x,

ΔZ_y: 对应式(8-3),边对所在的多边形的 ΔZ_y;

IP: 边对所在的多边形的番号。

(5) 深度缓存 ZB

ZB 用来存放一个扫描行中各象素点的深度值,所以其大小为一个扫描行的长度,即屏幕水平方向的象素点的个数。

(6) 颜色缓存 CB

颜色缓存 CB 与深度缓存 ZB 一一对应,其大小也是一个扫描行的长度,存放一行中各象素点的颜色。在深度缓存消隐算法中,为了节省空间,我们只好用显示器的帧缓存 FB 代替了颜色缓存 CB。这里,我们专门开辟了一个颜色缓存 CB。

6. 扫描线算法

一个完整的扫描线消隐算法可描述如下:

(1) 对于每一个多边形,根据多边形顶点的最小的 y 值,将多边形置入多边形 Y 桶。

(2) 有效多边形表 APT 初始化为空; 有效边表初始化为空。

(3) 对于每一条扫描线 j, j 从最小值 1 开始,做以下工作:

① 颜色缓存 CB 置成背景色。

② 深度缓存 ZB 置成机器无穷大。

③ 检查多边形 Y 桶中对应扫描线 j 的链中是否有新的多边形。如有,则将新多边形加到有效多边形表 APT 中。

④ 对于新添到有效多边形表中的多边形,生成相应的边 Y 桶。

⑤ 对于有效多边形表 APT 中的每一个多边形,若其边 Y 桶中对应扫描线 j 有新的边,则将新边配对加到有效边表 AET 中。

⑥ 对于有效边表 AET 中的第一个边对,做以下工作:

■ Z_{ij} 初值置为 $Z_l + \Delta Z_x$;

■ 对于每一满足 $x_l \leq i \leq x_r$ 的象素(i, j), i 从最小值 x_l 开始,做以下工作:

(i) $Z_{ij} = Z_{ij} - \Delta Z_x$;

(ii) 将 Z_{ij} 与 ZB(i) 比较。若 $Z_{ij} <$ ZB(i), 则

ZB(i) = Z_{ij};

CB(i) = 边所在的多边形的颜色。

⑦ 将 CB 的内容写入显示器帧缓存的第 j 行。

⑧ 检查有效多边形表 APT 中的多边形，若多边形顶点的最大 Y 坐标已等于 j，则从 APT 中删去该多边形，释放相应的边 Y 桶的存储空间，并从有效边表 AET 中删去属于该多边形的所有边对。

⑨ 检查有效边表 AET 中每一个边对，做以下工作：

■ 若 y_l 或 y_r 已等于 j，则从边对中删去左侧边或右侧边。

■ 若左侧边和右侧边都从边对中被删去，则从有效边表 AET 中删去这个边对；若边对中只有一条边被删掉，则从该边所属的多边形的边 Y 桶中找到与剩下的那条边配对的边，组成新的边对。

■ 计算新的截交点：

$$x_r = x_r - \Delta x_r$$

$$x_l = x_l - \Delta x_l$$

■ 根据边对所在的多边形，计算多边形在左侧边上的深度：

$$Z_l = Z_l + \Delta Z_x \times \Delta x_l - \Delta Z_y$$

7. 多边形的边配对算法

在上述扫描线算法的第⑤步和第⑨步，都要对多边形的边进行配对，形成边对加到有效边表中。如图 8-13 所示的多边形，对于边 e_1，有以下三种情况，需要进行边配对的工作：

(1) 扫描线首次与边 e_1 相交(图中，对应扫描线 j_1)；

(2) 与 e_1 配对的边 e_2 退出扫描线(图中，对应扫描线 j_2)；

(3) 有新的可以与 e_1 配对的边进入扫描线(图中对应扫描线 j_3)。

对多边形的边进行配对的算法可描述如下：

(1) 对于扫描线 j，取出边 Y 桶中所有要加到有效边表中的边的端点 X 坐标。

(2) 取出有效边表中属于该多边形的各边对的 x_l，x_r。

(3) 对由(1)和(2)两步得到的 x 值，按从小到大的次序排序。

(4) 在上述 x 序列中，从小到大，两两配对，并将相应的边对加到有效边表中。

在上述算法中，对图 8-14 所示的顶点 A 和顶点 B 要特别注意。如果 A 点和 B 点在 x 序列中都按一次计算，则在扫描线 j_2 上，x 序列为：

$x_1 x_3 x_4 x_B x_5$

x_1 和 x_3 配对，x_4 和 x_B 配对后，x_5 就无法配对。如果 A 点和 B 点在 x 序列中都按两次计算，则在扫描线 j_1 上，x 序列为：

$x_1 x_A x_A x_4 x_5$

x_1 和 x_A 配对，x_A 和 x_4 配对，x_5 无法配对，而且 $x_A x_4$ 的配对是错误的。所以在 x 序列中，A 点只能算一次，B 点必须算两次。

由于在扫描线算法中，我们先删去了与 e_1 配对的边 e_2，再对 e_1 重新配对，所以在 x 序列中，A 点只作为 e_3 的端点计算一次，所得的序列为：

图 8-13　边配对示意图

图 8-14　对特殊顶点的处理

$X_1X_AX_4X_5$

配对后得，X_1X_A 和 X_4X_5 两对，即 e_1e_3 和 e_4e_5 两个边对。

而在 B 点，由于 B 点既属于 e_6，又属于 e_7，所以在 x 序列中，B 点被计算成两次，得到的 x 序列为：

$X_1X_3X_4X_BX_BX_5$

配对后得 X_1X_3、X_4X_B、X_BX_5 三对，于是可得 e_1e_3、e_4e_7、e_6e_5 三个边对。

将 e_4 和 e_7 配对、e_6 和 e_5 配对，而不是将 e_4 和 e_6 配对、e_7 和 e_5 配对，这只要比较一下 x_4、Δx_7、Δx_6、x_5 就可以了。因 $\Delta x_7 < \Delta x_6$，$x_4 < x_5$ 所以 x_4 与 x_7 配对，x_6 与 x_5 配对。

四、多边形区域排序算法

这种算法是将线消隐和面消隐结合起来的一种物体空间的消隐算法。算法的基本思想是：在规格化图象空间中，将多边形按深度 Z 值自小至大排序，用前面的可见多边形去切割其后面的多边形，使得最终每一个多边形要么是完全可见的，要么是完全不可见的。这个算法包括两个主要部分：一是通用的多边形裁剪算法，二是消隐算法。现分叙如下：

1. 2D 多边形裁剪算法

由于整个消隐算法是在规格化图象空间中进行的，所以投影方向与 Z 轴一致。空间上的两点，如果它们在 XOY 投影面上的投影重合，即它们的 X 坐标和 Y 坐标分别相等，则可直接比较它们的 Z 坐标，确定它们的前后遮挡关系。所以必须在投影面上，对各多边形的投影多边形作二维裁剪运算，找出重叠部分。

我们先用前面介绍的韦勒-阿瑟顿的双边裁剪算法。其中：作为裁剪样板的多边形称为裁剪多边形 P_c；作为被裁剪对象的多边形称为主多边形 P_s。算法要求多边形中的边是有方向的。例如，在图 8-15 中，多边形 P 由 9 条有向边围成，其中，e_1、e_2、e_3、e_4 和 e_5 组成多边形的外环，e_6、e_7、e_8 和 e_9 组成多边形的内环。沿边的方向前进，多边形的内域总是在前进方向的左侧。这样，多边形的内环方向与外环的方向总是相反。

当用 P_c 去裁 P_s 时，P_c 将 P_s 分为内部多边形 P_{in} 和外部多边形 P_{out}。如图 8-16 所示，

P_c 为多边形 A 在 XOY 面上的投影多边形，P_s 为多边形 B 在 XOY 面上的投影多边形。P_c 将 P_s 分为 B_1 和 B_2 两部分，其中 B_1 为 P_{in}，B_2 为 P_{out}。

图 8-15　多边形 P 由有向边围成

图 8-16　裁剪多边形 P_c 将主多边形 P_s 裁为内部多边形 B_1 和外部多边形 B_2

2. 消隐算法

消隐算法将内部多边形 P_{in} 的各顶点的 X 坐标和 Y 坐标代入多边形 A 和 B 的平面方程中，求出 A 和 B 重叠部分的各顶点的 Z 值，据此比较其深度，确定是否要选 P_{in} 为新的 P_c，去裁剪其它的多边形。设有 n 个多边形，则多边形区域排序算法可描述如下：

(1) 初步的深度排序：搜索各多边形 P_k($k=1, 2, \cdots, n$) 各顶点的 Z 的最小值，并按其大小将多边形作初步排序。

(2) 以当前深度最小(即具有最小 Z 值，也即离视点最近) 的多边形作为裁剪多边形 P_c。

(3) 用 P_c 对多边形序列中那些深度更大的主多边形 P_s 进行裁剪，产生内部多边形 P_{in} 和外部多边形 P_{out}。

(4) 比较 P_c 与内部多边形 P_{in} 的深度，检查 P_c 是否是离视点较近的多边形。如果不是，则选择 P_{in} 为新的裁剪多边形，重新回到(3)。

(5) 在多边形序列中，选择下一个深度最小的多边形作为裁剪多边形，从(3) 开始重复做，直至所有的多边形都处理完为止。

(6) 在输出的多边形中，除去所有内部多边形外，其余的多边形均是可见多边形。

五、列表优先算法

这种算法是介于物体空间消隐和图象空间消隐的一种算法，它在物体空间中预先计算物体上各面可见性的优先级，然后再在图象空间中产生消隐图。

计算物体上各面可见性的优先级，实际上是按多边形离视点的远近进行排序，建立一张优先级表。算法约定距视点近的优先级高，距视点远的优先级低。生成图象时，优先级低的多边形先画，优先级高的多边形后画。这样，后画的多边形就会将先画的多边形遮挡

住,从而达到消隐的效果。

算法的关键在于在深度方向上对多边形进行排序,建立优先级表。但是,像图 8-17 所示的三个多边形,它们互相遮挡,我们就无法指出它们的优先级,无论先画谁或后画谁,都不会产生正确的消隐图。对于这样互相遮挡的多边形,我们必须用一个平面将其切开,分成可以建立优先级的两部分。例如,用垂直于 XOY 面的平面将多边形 A 切成两个小多边形 A_1 和 A_2,将多边形 B 切成 B_1 和 B_2(图 8-18)。这样,在分割面 P 的正向,多边形 A_1 遮挡 B_1,所以 A_1 的优先级大小于 B_1 的优先级。在 P 的负向,多边形 B_2 遮挡 C,C 遮挡 A_2,所以 B_2 的优先级大于 C,C 的优先级又大于 A_2。

图 8-17　多边形互相遮挡

图 8-18　用平面将多边形分割成可以建立优先级的小多边形

下面我们提出面集的概念。一个面集是若干个面的集合,它可以包含几个物体,也可以是一个物体的一部分。图 8-17 中,A、B、C 三个多边形组成一个面集。面集可以用平面进一步分割成子面集。图 8-18 中,A_1、B_1 组成的面集和 A_2、B_2、C 组成的面集就是图 8-17 所示面集的两个子面集。如果两个面集可以用一个平面将它们分开的话,则说这个面集是线性可分的,这个平面称为分离面。图 8-18 中的平面 P 就是一个分离面。

列表优先消隐算法是美国通用电气公司的舒麦克尔(Schumacker)基于上述面集的概念提出的。算法分为三步:

1. 在每个面集的内部计算面的优先级;
2. 计算各面集的优先级;
3. 实时产生消隐图。

算法假定各个多边形都是凸多边形。如果多边形不是凸多边形,则应预先将多边形分割成凸多边形。如图 8-19,多边形 ABCDEF 是一个凹多边形,在 CF 处将多边形分割开,就可得到两个凸多边形 ABCF 和 CDEF。

算法还假定消隐对象被预先划分成 K 个面集。如图 8-20 所示,先用平面①和平面②将小房子划分成三个面集 A、B 和 C,再调用列表优先算法对小房子进行消隐。

图 8-19　将凹多边形分割成凸多边形　　图 8-20　面集 A、B、C 及分离面①、②

划分面集的原则通常有两个：一是将相互分离的物体划分到不同的面集。如图 8-21 所示，O_1、O_2、O_3 是三个相互分离的物体，它们可以用平面①和②分开。这样，我们可以得到三个面集 O_1、O_2 和 O_3。

另一个原则是尽量使用面集中不出现图 8-17 所示的互相遮挡的面。否则无法计算面集中的面的优先级。通常，算法的调用者很难预先知道一个面集中是否有相互遮挡的多边形，所以在算法中计算面优先级这一步，必须具有判断这种互相遮挡的功能。

下面我们分别介绍算法的各步。

1. 面优先级的计算

如图 8-22 所示，V 为视点，a 为观察方向，n 为平面 P 的外法向。若 $n \cdot a < 0$，则说面 P 对于视点 V 为正向面（图 8-22(a)）；若 $n \cdot a > 0$，则说面 P 对于视点 V 为反向面（图 8-22(b)）。

图 8-21　用平面①和②划分分离的物体　　图 8-22　正向面和反向面

如果从任何视点看过去，在删去了所有反向面之后，面 A 有可能遮挡面 B（一部分或全部），而面 B 不可能遮挡面 A，则面 A 的优先级高于面 B 的优先级，记为 A > B。如图

8-23, 箭头表示面的外法线方向。从 A 和 C 的正面看过去, A 有可能遮挡 C; 从其它方向看过去, A 和 C 要么互不遮挡, 要么都被作为反向面删掉, 那么, 可以记为 A> C, 或 C< A。从 A 的正面看, E 面可作为反向面被删掉; 从 E 的正面看, A 面可作为反向面被删掉; 而从其它的方向看过去, A 和 E 要么互不遮挡, 要么均被作为反向面删掉, 那么我们说 A 和 E 的优先级是相等的, 可记为 A= E。同样, C= E。

图 8-23　多边形各面的外法线方向

值得注意的是, A= E, C= E 和 A> C 这三者之间并不矛盾。A> C 是说 A 必须在 C 之后显示才能得到正确的消隐图, 而 A= E 和 C= E 之间, 无论是先显示 C 还是先显示 E, 都能得到正确的消隐图。所以, 按 CAE、或 CEA、或 ECA 的顺序显示, 都是正确的。

计算面优先级的方法如下:

(1) 从多边形 A 的某顶点 i 向多边形 B 的某顶点 j 引一矢量 a_ib_j;

(2) 求矢量 a_ib_j 与多边形 A 的内法向 N_A 的点积 $a_ib_j \cdot N_A$;

(3) 求矢量 a_ib_j 与多边形 B 的内法向 N_B 的点积 $a_ib_j \cdot N_B$;

(4) 对所有 i 和 j, 重复步骤(1)～(3)。

在图 8-24 中, a_1b_1、a_1b_2、a_1b_3、a_1b_4 分别为多边形 A 的 1 号顶点向多边形 B 的各顶点所引的矢量, 正号和负号表示这些矢量和多边形法向点积后的符号。用这些点积的符号可以表明所测试面的优先级。如果用数偶(sign1, sign2) 表示 $a_ib_j \cdot N_A$ 的符号和 $a_ib_j \cdot N_B$ 的符号, 则

正数偶(+ , +)和中性数偶(+ , -)、(- 、+) 表示 A> B。

图 8-24　面优先级计算示意图

注意多边形的法向为内法向

负数偶(-，-)和中性数偶(+，-)、(-，+)表示 A< B。

中性数偶(+，-)、(-，+)和点积为零表示 A= B。

在计算多边形的优先级时，若同时出现正数偶和负数偶，即 A> B，同时 B> A，或者出现 A> B、B> C、C> A 这种能形成闭环的优先级，则说明原来预先划分的面集中，有多边形彼此遮挡，必须进一步划分，直至面集中不再出现这样的冲突为止。

对面集中所有的多边形两两进行比较，都排出优先级高低之后，可以产生一个优先级表。如图 8-25 所示，A、B 为小立方体的两个前侧面，C、D 为小立方体的两个后侧面，E 为小立方体的顶面，F、G 为大立方体的两个前侧面，H、J 为大立方体的两个后侧面，I 为大立方体的底面，K 为大立方体的顶面。各面的优先级序列为 KABCDEF GH JI，其中，K 的优先级最低，I 的优先级最高。

从上面的面优先级的计算中可以看出，一个面集中的面优先级只与多边形的空间位置和几何形状有关，与视点无关。无论视点移到哪里，面的优先级都固定不变。同一张面优先级表可用来产生多帧视点不同的画面。因而这种方法特别适用于飞行模拟，因为在飞行模拟中，环境不变，而视点却在不停地变化。

2. 面集优先级的计算

如图 8-20 所示，分离面①和②将物体分成三个面集 A、B、C。面集 A 位于分离面①的正向，面集 B 和 C 位于分离面①的反向，且其中 B 位于分离面②的正向，C 位于②的反向。在计算面集优先级之前，我们可以建立如图 8-26 所示的二叉树，表明面集和分离面之间的位置关系。

图 8-25　多边形各面的优先级　　图 8-26　对应图 8-20 的二叉树

假定视点 V_1 位于分离面①的正向和分离面②的正向，则沿分离面①的正向搜索，得到 A，沿①的负向和②的正向搜索，得到面集 B，最后在②的负向上得到面集 C。于是，对于视点 V_1，面集的优先级顺序为 A> B> C。

面集的优先级与视点有关。若视点移到 V_2，在分离面①的负向和②的负向，则沿①的负向和②的负向搜索二叉树，首先得到的是面集 C，其次是 B，最后是 A。故对于视点 V_2，面集的优先级顺序为 C> B> A。

面集的优先级决定了显示顺序。如果面集 A 的优先级大于面集 B 的优先级，那么在显示时，应先显示优先级较低的面集 B 中的多边形，再显示优先级较高的面集 A 中的多

边表,使面集 A 自然覆盖面集 B。

3. 消隐图的生成

生成消隐图是在图象空间中进行的。算法如下:

(1) 对于给定的视点、去掉面优先级表中所有反向面。

(2) 对于每一个面集,从面集优先级低的面集开始,做以下工作:

(3) 对于面集中的每一个多边形,从面优先级的多边形开始,做以下工作:

(4) 将多边形投影到 XOY 平面上;

(5) 对于每一条扫描线,计算投影多边形各边与扫描线的交点。如图 8-27 所示,扫描线 j 与投影多边形各边的交点为 l_1、l_2、l_3、r_1、r_2、r_3。这一步可利用前面介绍的扫描线消隐算法中用到的扫描线相关性。

图 8-27　计算扫描线与凸多边形的交线段

(6) 由于多边形是凸多边形,所以每个多边形在每条扫描线上最多只有一段。根据多边形顶点的 y 最大值和 y 最小值,将多边形的边分成左侧的边和右侧的边。图 8-27 中,L_1、L_2、L_3 为左侧边,R_1、R_2、R_3 为右侧边。

(7) 在左侧边与扫描线的交点中求出交点最靠右的那条边 L_{max}(即交点 X 坐标最大的一条边),在右侧边与扫描线的交点中,求出交点最靠左的那条边 R_{max}(即交点的 X 坐标最小的一条边)。

(8) 若 $L_{max} < R_{max}$,则多边形与扫描线相交,可以按多边形的颜色显示交点 L_{max} 到 R_{max} 一段。

第二节　真实感图形生成技术

一、引言

在上一节介绍的图象空间消隐算法中,我们总是以离象素最近的多边形的颜色作为

象素的颜色。在那里，我们把注意力集中在消隐算法上。至于多边形的颜色是如何得到的，我们没有做任何解释。简单的办法有两个。一是当象素点为多边形的边界时，用前景色；当象素点在多边形的内部时，用背景色。这样得到的图形是消隐了的线框图。另一个办法是预先给每一个多边形赋予一种颜色。这样得到的消隐图就像一幅彩色填充图，多边形之间及其内部没有均匀的颜色过渡。

尽管如此，用这种简单的方法处理多边形的颜色，得到的消隐图较之线框图来说，有质的飞跃。因为消隐图能反映物体间的相互遮挡关系，能增强图形的深度感和层次感，且能排除视觉上的二异性。但还是不尽人意。人们总希望计算机生成的图形能跟人们平常肉眼所看到的五彩斑斓的景象相媲美，也就是要求所生成的图形具有以下特点：

① 能反映物体表面颜色和亮度的细微变化；

② 能表现物体表面的质感；

③ 能通过光照下的物体阴影，极大地改善场景的深度感和层次感，充分体现物体间的相互遮挡关系；

④ 能模拟透明物体的透明效果和镜面物体的镜象效果。

总之，要求生成具有真实感的图形。

那么，决定一个物体外观的因素有哪些呢？首先当然是物体本身的几何形状。自然界中物体的形状是很复杂的，有些可以表示成多面体，有些可以表示成曲面体，而有些根本无法用简单的数学函数来表示(如云、水、雾、火等)。

其次是物体表面的特性。这包括材料的粗糙度、感光度、表面颜色和纹理等。对于透明体，还要包括物体的透光性。纸和布的不同在于它们是不同类型的材料，而同样是布，又可通过布的质地、颜色和花纹来区分。

第三是照射物体的光源。从光源发出的光有亮有暗、光的颜色有深有浅，我们可以用光的波长(即颜色)和光的强度(即亮度)来描述。光源还有点光源、线光源、面光源和体源之分。

第四是物体与光源的相对位置。

最后是物体周围的环境。它们通过对光的反射和折射，形成环境光，在物体表面上产生一定的照度。它们还会在物体上形成阴影。

生成真实感图形技术的关键在于充分考察上述影响物体外观的因素，建立合适的光照模型，并通过显示算法将物体在显示器上显示出来。

对于光栅扫描图形显示器，屏幕上显示的一幅图形是由一个一个象素点组成的一个整体。每个象素点又是红(R)、绿(G)、蓝(B)三个基色的混合表现。对于通常24位的彩色显示器，其RGB三基色的取值范围都是0到255。

图8-28 RGB色彩空间

图8-28是RGB终端色采模型。图中，R、G、B三个坐标轴分别表示红、绿、蓝三个基色，方块表示RGB颜色空间。这样，任何一种颜色都可以表示成RGB三基色的组合，在光栅扫描显

示器上显示出来。例如,若 B= 0,R= G,则得到的颜色为黄色。

图 8-28 不仅可以表示象素的颜色,还可以表示象素点的亮度。例如,整个坐标轴 R 都表示红色,但从 0 到 1,红色的亮度在不断增强。

可见,显示真实感图形,实际上就是计算显示屏上各象素点的 RGB 的取值。下面,我们将从简单的光反射模型出发,逐步深入地介绍真实感图形生成技术。

二、简单光反射模型

如图 8-29 所示,当光照射到一个物体的表面上时,物体对光会产生反射、透射和散射作用,物体内部还会吸收一部分光。这可用如下等式表示:

$$\text{入射光} = \text{反射光} + \text{透射光} + \text{散射光} + \text{吸收光} \tag{8-5}$$

图 8-29 光照射物体

用简单光反射模型模拟光,照射到物体表面时,产生了光的反射效果,它假定光源是点光源,物体是非透明体,于是在式(8-5)中,透射光和散射光将近似于零,可以忽略不计。

反射光由漫反射光、环境光和镜面反射光三部分组成。下面我们分别加以介绍。

1. 漫反射光

一个粗糙的、无光泽的表面表现为漫反射。当光线照射到这样的表面上时,光线沿各个方向都作相同的反射,所以从任何角度去看这种表面都有相同的亮度。漫反射的特点是:光源来自一个方向,反射光均匀地射向各个方向。

漫反射模型可以用兰伯特(Lambert)余弦定律描述。如图 8-30 所示,设物体表面在 P 点的法向为 N,从 P 点指向光源的向量为 L,两者的夹角为 θ,于是,点 P 处的漫反射光的强度为:

$$I_d = I_p \times K_d \times \cos\theta \quad 0 \leq \theta \leq \frac{\pi}{2} \tag{8-6}$$

这里,I_p 为入射光的强度,即点光源的光强;K_d 为漫反射系数,是 0～1 之间的一个常数,取决于物体的材料属性和入射光的波长;θ为光线的入射角,即 L 与 N 之间的夹角。

若 L 和 N 都已规格化为单位向量,则上式可以写成

$$I_d = I_p \times K_d \times (L \times N) \tag{8-7}$$

式(8-7)表明,漫反射光的强度 I_d 与反射角无关,也即与视点的位置无关,视点移到任何地方,接受到的漫反射光强都相等。但漫反射光的强度与入射角 θ有关。图 8-31 中,面①

平行于光线，θ= 90°；面②面向光源，θ= 0°；面③的 θ小于 90°；所以，光线照不到面①，面②比面③更亮。

图 8-30　漫反射

图 8-31　三个漫反射面

如果用 RGB 三个分量表示入射光的光强 I_p，即

$$I_p = (I_{pr}, I_{pg}, I_{pb})$$

则式(8-7)就分解为

$$\begin{aligned} I_{dr} &= I_{pr} \cdot K_{dr} \cdot (L \cdot N) \\ I_{dg} &= I_{pg} \cdot K_{dg} \cdot (L \cdot N) \\ I_{db} &= I_{pb} \cdot K_{db} \cdot (L \cdot N) \end{aligned} \tag{8-8}$$

其中，K_{dr}、K_{dg}、K_{db}分别为 RGB 三基色的漫反射系数，若入射光为白光，即 $I_{pr} = I_{pg} = I_{pb}$，则物体的颜色就由漫反射系数确定。所以通过设定不同的 K_{dr}、K_{dg}、K_{db}，可以调节物体表面所呈现的颜色。

上式中，I_{dr}、I_{dg}、I_{db}为漫反射光强 I_d 的 RGB 三个分量，反射了物体表面在 P 点的颜色和亮度。

如果有多个点光源，则漫反射光照模型为：

$$I_d = K_k \cdot \sum_n I_{p,n}(L_n \cdot N) \tag{8-9}$$

2. 环境光

在图 8-31 中，面①平行于光线，因而是不可见。但事实上，在现实生活中，它却是可见的。这是为什么呢？是环境光使它可见。

环境光是光在物体和周围环境(如墙)之间多次反射的结果，它不是直接来自光源，而是来自周围的环境对光的反射。环境光的特点是：照射在物体上的光来自周围各个方向，又均匀地向各个方向反射，如图 8-32 所示。

物体上某点 P 所接收到的环境光来自它周围其它点发出的发射光，而其它点所发出

的反射光又可能受到 P 点所发出的反射光的影响。在辐射度方法中我们将精确地模拟这种效果。这里我们近似认为,同一环境下的环境光是恒定不变的,对任何物体的表面都相等。设 I_a 为环境光的强度,于是,P 点对环境光的反射强度为

$$I_e = I_a \cdot K_a \tag{8-10}$$

其中,K_a 为物体表面对环境光的反射系数,I_e 为环境光反射强度。

3. 镜面反射光

镜面反射的特点是:光源来自一个方向,反射光集中在反射方向。

一个理想的光泽表面,如同一面镜子,仅仅在反射角等于入射角时,光线才会被反射。如图 8-33 所示,只有在反射方向上(即 $\alpha = 0$),观察者才能看到从镜面反射出来的光线,而在其它方向都看不到反射光。对于这种光滑的反射面,镜面反射的光强要比漫反射的光强和环境光的光强高出很多倍。这时,如果观察者正好处在 P 点的镜面反射的方向上,就会看到 P 点比周围亮得多,有一个高光点。

图 8-32　环境光的反射

图 8-33　镜面反射

对于那些非理想的反射面,譬如苹果的表面,镜面反射光的强度就会随 α角的增加而急剧地减少,但观察者还是可以在 α很小时接收到这种改变了方向的一部分镜面反射光。

镜面反射模型又称冯(Phong)模型,可表示为:

$$I_s = I_p \cdot K_s \cdot \cos^n \alpha \tag{8-11}$$

其中,I_s 为镜面反射光在观察方向上的光强,I_p 为点光源的强度,K_s 为镜面反射系数,α为视点方向 V 与镜面反射方向 R 之间的夹角,n 是与物体表面光滑度有关的一个常数,一般取为 1～2000,表面越光滑,n 越大。

若 R 与 V 已规格化为单位向量,则式(8-11)可改写为:

$$I_s = I_p \cdot K_s \cdot (R \cdot V)^n \tag{8-12}$$

镜面反射产生的高光点只反映光源的颜色,因此,在白光的照射下,高光点是白色。K_s 与物体的颜色无关,物体的颜色通过设置漫反射系数 K_d 来控制。

综上所述,从视点观察到物体上任一点 P 处的颜色和亮度 I 应为漫反射光强 I_d,环境光反射光强 I_e 及镜面反射光强 I_s 的总和,即

$$I = I_d + I_e + I_s \tag{8-13}$$

将式(8-7)、式(8-10)和式(8-12)代入上式,则得:

$$I = I_p \times K_d \times (L \times N) + I_a \times K_a + I_p \times K_s \times (R \times V)^n \qquad (8\text{-}14)$$

4. 几何上的考虑

式(8-14)的计算量是很大的,尤其是计算镜面反射方向R。为减少计算工作量,我们要进行一些假设和近似。例如,假设光源在无穷远,视点也在无穷远。这样,向量L和V就是一个常向量。接下来,用N·H近似R·V。这里H是向量L和V的平分向量,即

$$H = (L + V)/2 \qquad (8\text{-}15)$$

如图7-75所示。β为H和N之间的夹角,α为R和V之间的夹角,则有

$$\theta + \beta = \frac{1}{2}(2\theta + \alpha)$$

图 8-34 用H·N近似R·V

于是,

$$\beta = \alpha/2$$

H和N之间的夹角虽然是V和R之间的夹角的一半,但必要时,可通过调整指数n对镜面反射光的高光域进行补偿。图8-35显示了n=1时用(H·N)和用(R·V)所生成的高光域之间的区别。(H·N)所对应的高光域较大。

图 8-35 (H·N)和(R·V)对应的高光域

用(H·N)代替式(7-41)中的R·V,可得到:

$$I = I_p \times K_d \times (L \times N) + I_a \times K_a + I_p \times K_s \times (H \times N)^n \qquad (8\text{-}16)$$

这里,H应规格化为单位向量。

由于光源和视点都设在无穷远,所以,在用上式计算物体上任何一点的亮度时,L和V都保持不变,因而H是一个常向量。就是说只要计算一次H的值就可以。这就是用(H·N)近似(R·V)的好处。

5. 色彩

在简单光照模型中,镜面反射产生的高光点只能模拟光源的颜色,物体的颜色是通过

设置漫反射系数 K_d 的 RGB 三个分量的值来实现的。将式(8-16)改写成 RGB 三个分量的形式,就是:

$$\begin{aligned} I_r &= I_{ar} \cdot K_{ar} + I_{pr}[K_{dr}(L \cdot N) + K_s(H \cdot N)^n] \\ I_g &= I_{ag} \cdot K_{ag} + I_{pg}[K_{dg}(L \cdot N) + K_s(H \cdot N)^n] \\ I_b &= I_{ab} \cdot K_{ab} + I_{pb}[K_{db}(L \cdot N) + K_s(H \cdot N)^n] \end{aligned} \tag{8-17}$$

这里要注意的是,镜面反射一项对三个等式都是一样。将上式合写成一个等式,就是:

$$\begin{aligned} I(r,g,b) = {} & I_a(r,g,b)K_a(r,g,b) \\ & + I_p(r,g,b)[K_d(r,g,b)(L \cdot N) + K_s(H \cdot N)^n] \end{aligned} \tag{8-18}$$

6. 在深度缓存消隐算法中应用简单光照模型

在深度缓存消隐算法中,对于显示屏幕上的任何一点(i,j),计算多边形 P_k 在这一点的深度 Z_{ij}。当 Z_{ij} 小于深度缓存 $ZB(i,j)$ 的值时,就要计算多边形 P_k 在(i,j,Z_{ij})点的颜色。作为例子,我们这里用简单光照模型(式(8-18))计算多边形 P_k 在(i,j,Z_{ij})点的颜色。具体做法是,先设定 $I_a(r,g,b)$, $K_a(r,g,b)$, $K_d(r,g,b)$、K_s 和 n 的值。再假定光源在无穷远处,光源的强度为 $I_p(r,g,b)$,光线的方向为 L。并假定观察方向始终是 Z 轴方向,即 V = (0,0,1)。于是:

$$H = \frac{(L + V)/2}{|(L + V)/2|}$$

仍为单位向量。最后,对于多边形 P_k,设其平面方程为:

$$ax + by + cz + d = 0$$

则其法向量 N= (a, b, c)。将上述各参数代入式(8-18)就可求出多边形 P_k 在点(i,j,Z_{ij})处的亮度(I_r, I_g, I_b)。

算法可描述如下:

(1) 由 L 和 V 计算 H。

(2) 初始化 ZB 和 FB。

(3) FOR j= 1, n　　/* 第 j 根扫描线* /

　　FOR i= 1, n　　/* 第 j 根扫描线上第 i 个象素点* /

　　FOR k= 1, N　　/* 多边形 P_k* /

① 令 Z_{ij}= 机器最大值;

② 判断点(i,j)是否落在投影多边形 P_k 内;

③ 若(i,j)在 P_k 内,则计算多边形 P_k 在点(i,j)处的深度值 Z_{ij};

④ 比较 Z_{ij} 和 $ZB(i,j)$ 的大小;

⑤ 若$Z_{ij} < ZB(i,j)$,则令

　　$ZB(i,j) = Z_{ij}$;

　　用式(7-45)计算(i,j)点的颜色 I_{ij};

　　$FB(i,j) = I_{ij}$。

由于光源和视点都在无穷远,所以 L 和 V 恒定不变。因此对于同一多边形域内的每一个点,由于其法向都等于多边形的法向,所以 L · N 和 H · N 也相等,因而光强 I 也相

等。这样,在同一个多边形域内,其颜色和亮度是不变的,用上述算法生成的图形仍然像一幅填充图。即使是两个不同的多边形,只要它们的法向相同,其表面颜色和亮度就无法区分。只有那些法向各异的多边形,我们才能分出它们的亮度,但这在多边形和多边形的邻接处,又会产生所谓的马赫带效应,即肉眼感觉到的亮度变化比实际的亮度变化要大。

而在计算机图形学中,多边形常常被用来近似地表示曲面,这样就更需要使多边形之间的色彩和亮度平滑过渡,使由若干块平面多边形近似的曲面呈现匀称的色彩。下一小节介绍的增量式光反射模型将解决这一问题。

三、多边形表示的明暗处理

有两种增量式光反射模型,可以解决多边形之间亮度和色彩的光滑过渡问题。

1. 双线性亮度插值法

这种方法又称作哥罗德(Gouraud)明暗法,主要针对简单光反射模型中的漫反射项。假设光源在无穷远处,那么,对同一多边形上的点,$(L \cdot N)$为一恒定的值。为保证多边形之间的光滑过渡,算法先计算出物体上各个顶点的亮度,然后用双线性插值,求出多边形内域中每一点的亮度值。

顶点的亮度还是用式(8-18)计算,但这里的 N 是顶点的法向量,是由与顶点相邻的所有多边形的法向量求平均值得到的。如图 8-36 所示,A 点的法向量 N_a 等于相邻 4 个多边形的法向量的平均值,即

$$N_a = \frac{1}{4}(N_1 + N_2 + N_3 + N_4)$$

这是双线性插值法的一个重要特征,即顶点的法向量与曲面本身的法向量较接近。

计算了所有顶点的亮度之后,利用双线性插值,可进一步求出多边形内部各点的亮度。这一插值过程可以与扫描线消隐算法结合起来,先由顶点的亮度插值求出各边的亮度值,再由各边的亮度值插值求出多边形域内各点的亮度值。如图 8-37 所示,沿着扫描线

图 8-36　顶点的法向量等于相邻多边形的法向量平均值

图 8-37　亮度插值法图示

y_s,由顶点(x_1, y_1)和(x_2, y_2)的亮度值 I_1、I_2 计算左侧的边的亮度 I_a,由顶点(x_1, y_1)和(x_4, y_4)的亮度值 I_1、I_4 计算右侧的边的亮度值 I_b。再由左右两侧边的亮度值 I_a、I_b,计算多边形内在扫描线 y_s 上各点的亮度值 I_s。插值公式为:

$$I_a = \frac{1}{y_1 - y_2}[I_1(y_s - y_2) + I_2(y_1 - y_s)]$$

$$I_b = \frac{1}{y_1 - y_4}[I_1(y_s - y_4) + I_4(y_1 - y_s)] \tag{8-19}$$

$$I_s = \frac{1}{x_b - x_a}[I_a(x_b - x_s) + I_b(x_s - x_a)]$$

上式可以改写成增量的形式。当 y_s 由 j 变为 j + 1 时,

$$I_{a,j+1} = I_{a,j} + \Delta I_a$$

$$I_{b,j+1} = I_{b,j} + \Delta I_b \tag{8-20}$$

其中,

$$\Delta I_a = \frac{1}{y_1 - y_2}(I_1 - I_2)$$

$$\Delta I_b = \frac{1}{y_1 - y_4}(I_1 - I_4)$$

当 x_s 由 i 增为 i + 1 时,

$$I_{s,i+1} = I_{s,i} + \Delta I_s \tag{8-21}$$

其中,

$$\Delta I_s = \frac{1}{x_b - x_a}(I_b - I_a)$$

用双线性亮度插值法计算物体表面亮度的算法可描述如下:

(1) 计算多边形法向;

(2) 计算顶点的法向;

(3) 计算顶点的亮度;

(4) 在扫描线消隐算法中,当扫描线由第 j 行增为 j + 1 行时,用式(8-20)计算边的亮度;当象素点由 i 变为 i + 1 时,用式(8-21)计算象素点的亮度。

双线性亮度插值法解决了两多边形之间亮度的不连续过渡以及多边形域内亮度单一的问题,而且计算速度较之简单光反射模型有明显的提高。用这种方法得到的图形比用简单光反射模型得到的图形,其真实感要强得多。但马赫带效应还没有完全消除。另外,这种方法只考虑了漫反射,而对镜面反射,其效果就不太理想,主要表现在高光域的形状不规整,高光域只能在顶点周围形成,不能在多边形域内形成。再有就是这种方法会出现下述错误现象。如图 8-38 所示,用与顶点相邻的多边形的法向平均作为顶点的法向。虽然多边形的法向不相同,但顶点的法向都完全相同,这样各个顶点的亮度,以致整个面的亮度都一样了。这样一来,原来是一个波浪状的曲面,经亮度插值法计算之后却变成了平面的样子被显示出来。这虽然是一个问题,但只要用足够多的多边形去近似

图 8-38　波浪状曲面的顶点法向

曲面,这种现象就不会出现了。

2. 双线性法向量插值法

这种方法是由冯(Phong)提出的,这不仅克服了亮度插值法中一些不足之处,而且成功地把镜面反射综合在这一方法中。即使在顶点的法向量方向不产生高光,用这种方法也能在多边形域内生成一个高光域。综合起来,这种方法有如下特点:

(1) 仍保留双线性插值,对多边形边上各点和内域各点,还是用增量式。

(2) 对顶点的法向量进行插值,而不是对顶点的亮度进行插值。顶点的法向量仍是相邻多边形的法向量平均值。

(3) 对每一个象素,由插值得到的法向量计算各自的亮度。

(4) 仍假定光源和视点都在无穷远处,这样,每点的亮度只是这点法向量的函数。

法向量插值算法的第一步与亮度插值算法一样,计算各顶点的法向。然后,对于每一条扫描线,通过插值计算在扫描线与边的交点处的法向 N_a、N_b(图 8-39),再用 N_a 和 N_b 插值计算多边形域内在扫描线上各点的法向 N_s。N_s 与实际曲面上的切平面法向量接近,这是法向量插值的特征。图 8-40 表明了这一情况。

图 8-39 法向量插值法图示

图 8-40 插值的法向量与曲面的切平面法向接近

采用图 8-37 和图 8-39 上的符号,我们有:

$$
\begin{aligned}
N_b &= \frac{1}{y_1 - y_2}[N_1(y_s - y_2) + N_4(y_1 - y_s)] \\
N_b &= \frac{1}{y_1 - y_4}[N_1(y_s - y_4) + N_4(y_1 - y_s)] \\
N_s &= \frac{1}{x_b - x_a}[N_a(x_b - x_s) + N_b(x_s - x_a)]
\end{aligned}
\tag{8-22}
$$

跟亮度插值公式一样,上式也可以写成增量的形式。当 y_s 由 j 增为 j + 1 时,

$$
\begin{aligned}
N_{a,j+1} &= N_{a,j} + \Delta N_a \\
N_{b,j+1} &= N_{b,j} + \Delta N_b
\end{aligned}
\tag{8-23}
$$

其中,

$$
\Delta N_a = \frac{1}{y_1 - y_2}(N_1 - N_2)
$$

$$\Delta N_b = \frac{1}{y_1 - y_4}(N_1 - N_4)$$

当 x_s 由 i 增为 i+1 时,

$$N_{s,i+1} = N_{s,i} + \Delta N_s \tag{8-24}$$

其中,

$$\Delta N_s = \frac{1}{x_b - x_a}(N_b - N_a)$$

3. 两种方法的比较

亮度插值法是显示漫反射曲面的一种有效的方法,它虽然也能模拟镜面反射,但高光的形状取决于多边形的相对位置。亮度插值法的优点在于它的计算量小,只要求计算各顶点的亮度,然后用双线性插值就可以求出其它各象素点的亮度值。

法向量插值性可以生成高光,但计算量较大,既要计算各顶点的法向量,插值计算曲面上各点的法向量,还要用光照模型计算各象素点的亮度值。有多种加速法向量插值的方法,这里就不再赘述了。

四、整体光照模型与光线跟踪算法

1. 反射和折射的几何光学原理

对于透明或半透明物体,在光线与物体表面相交时,一般会产生反射和折射。经折射后的光线将穿过物体的另一表面射出,形成透射光。因此,式(8-5)中透射光一项就不能忽略。光线从物体的内部穿出物体表面时,还会产生折射(图 8-41)。如果我们站在反射光的方向上,我们将会看到反射光,如果我们站在折射光线的方向上,我们就能看到透过物体的透射光。

反射光和折射光的方向可以用几何光学原理来确定。如图 8-42 所示,若用 I 表示入射光方向上的单位矢量,N 表示面的单位法向量,则根据反射定律,反射光与入射光在同一个平面上,与法向所形成的夹角相等,且在法向的两侧。于是,如果 R 表示反射方向上的单位矢量,则 R 可表示成

$$R = I + 2N\cos\theta \tag{8-25}$$

根据折射定律,设 η 是法向所指空间的介质的折射率,η 是物体的折射率,那么,入射角 θ 和折射角 θ 有如下关系:

$$\frac{\sin\theta}{\sin\theta} = \frac{\eta}{\eta} \tag{8-26}$$

折射方向和入射方向仍与法向 N 在同一个平面上。若用 T 表示折射方向上的单位矢量,则有:

$$T = \frac{1}{\eta}I - \left(\cos\theta - \frac{1}{\eta}\cos\theta\right)N \tag{8-27}$$

其中,

$$\eta = \frac{\eta}{\eta}$$

$\cos\theta$ 由式(8-26)解得:

$$\cos\theta = \sqrt{1 - \frac{1}{\eta^2}(1 - \cos^2\theta)}$$

图 8-41　光的反射和折射

图 8-42　反射方向和折射方向的计算示图

当光线从一个高密介质射向一个低密介质时，折射光线有可能与介质的分界面平行(图 8-43(a))，这时的 θ_t 被称作临界角。如果 $\theta_i > \theta_t$，则折射完全变成了内反射(图 8-43(b))。

图 8-43　内部反射

2. 光透射模型

在简单光反射模型(式(8-16))上，再加上透射光一项，就可以模拟透明体在光源照射下的透明效果。透射光是隔着透明体观察到的，视点和光源应在透明体的两侧(图 8-44)。

一个理想的透明介质，只有在折线方向才能看到透过来的光线，而在其它方向都看不到透射光。而对于那些半透明的物体，观察者在透射方向的附近可以接受到一部分透射光，但透射光的强度随 α的增大而急剧减小。这里 α为观察方向 V 和透射方向 T 之间的夹

角(图 8-45)。

图 8-44　隔着透明体可观察到透射光　　图 8-45　透射

用下式可以模拟透射光的这种变化:

$$I_t = K_t \cdot I_p \cdot (\cos\theta)^n$$
$$= K_t \cdot I_p \cdot (V \cdot T)^n 。 \quad (8\text{-}28)$$

这里, I_t 映为在观察方向接受到的透射光的强度, K_t 为透射系数, I_p 为光源的光强, n 为一常数, 反映透光域的大小。T 由式(8-27)给出。

式(8-28)就是光透射模型。为减少计算量, 上式可近似写成

$$I_t = K_t \cdot I_p \cdot (H' \cdot N)^n \quad (8\text{-}29)$$

这里, H′ 的含义与简单光反射模型中的 H 的含义类似, 可表示成光线 L 和视线 V 的函数。

$$H' = \frac{-L-(\eta_t/\eta_i)V}{(\eta_t/\eta_i)-1} = \frac{-L-\eta V}{\eta-1} \quad (8\text{-}30)$$

(注意, 为避免上式过分复杂, H′ 没有单位化)。

在简单光反射模型中, 加上 I_t 一项, 就可以得到透明体的光照模型

$$I = I_a K_a + I_p[K_d(N \cdot L) + K_s(N \cdot H)^{n_s} + K_t(N \cdot H')^{n_t}] \quad (8\text{-}31)$$

值得提醒的是, 对于一个给定的光源, 上式中后两项, 只有一项起作用。当视点与光源在物体的同一侧时, 镜面反射项 $K_s(N \cdot H)^{n_s}$ 一项有值, 而当视点与光源在物体的异侧时, 透射项 $K_t(N \cdot H')^{n_t}$ 一项有值。漫反射项 $K_d(N \cdot L)$ 仍然可以用来模拟透射光的漫透射效果。

3. 整体光照模型

照射到物体上的光线, 不仅有从光源直接射来的, 也有经过其它物体的反射或折射来的。如图 8-46 所示, 在视点所看到的 A 点的亮度, 由两部分组成。一部分是从光源直接照射到 A 点, 经漫反射和镜面反射到达视点的光强。另一部分是面②上的 B 点对光源的反射光, 经 A 点的反射到达视点的光强。前者称为直接光照, 用前面介绍的简单光反射模型

或透射模型来模拟。后者称为整体光照,用整体光照模型来模拟。

模拟直接光照的模型在这里又称作局部光照模型,与整体光照模型相对。局部光照模型为

$$I_{local} = I_a ¡¤K_a + I_p[K_d ¡¤(N ¡¤L) + K_s ¡¤(N ¡¤H)^{n_s} + K_t ¡¤(N ¡¤H')^{n_t}] \quad (8\text{-}32)$$

由环境光、漫反射光、镜面反射光和透射光四部分组成。这里要注意的是,整体光照模型要表现物体间的光反射,这种反射主要是指物体间的镜面反射,应与局部光照模型中的环境光区分开来。环境光是来自周围物体的漫反射,在各个方向都是均匀的,且是一个常数。

整体光照模型可表示如下:

$$I_{global} = K_R I_R + K_T I_T \quad (8\text{-}32)'$$

这里,如图 8-47 所示,I_{global}为整体光照对 P 点的光贡献;I_R 为其它物体从视线的反射方向 R 反射或折射来的光强,K_R 为 A 点的反射系数;I_T 为其它物体从视线的折射方向 T 反射或折射来的光强,K_T 为 A 点的折射系数。K_R 和 K_T 为常数,与物体的材料属性有关。整体光照模型反映了反射方向和折射方向的光对物体表面上各点光强的贡献。用整体光照模型可以表现透明体的透明效果,以及镜面物体的镜面映像效果。

图 8-46　A 点的亮度由直接光照和整体光照形成

图 8-47　整体光照模型图示

把局部光照模型和整体光照模型结合起来,就可以得到物体表面上点的光强,即

$$I = I_{local} + I_{global} \quad (8\text{-}33)$$

4. 光线跟踪算法

光线跟踪算法采用整体和局部光照模型,模拟光的反射和透射,表现镜面物体的镜面映像、透明体的透明效果、以及物体间的阴影。视点可以在无穷远,也可以在空间中的任意一个地方。光源也是这样,而且可以有多个点光源。

算法的基本思想可如图 8-48 所示。在物体空间中,从视点逆着光线的方向向视平面上的象素点作射线 V,以确定这一点的亮度。射线 V 将与物体 O_1 相交于 A 点,用局部光照模型计算 A 点在光源直接照射下的颜色和亮度。由于物体 O_1 的表面对光有镜面反射,同时又是一个透明体,因此在交点 A 处,还要加上物体间的反射光或透射光对 A 点的贡献。用整体光照模型计算这一贡献。于是视平面上象素点 P 的亮度,也就是从视点看到的 A 点的亮度由下面三部分组成:

(1) 光源直接照射及环境光所产生的局部光强。

(2) 反射方向②来的光对 A 点的贡献。

(3) 透射方向③来的光对 A 点的光强。

图 8-48　光线跟踪算法的基本思想图示

于是,应用式(8-33),得 A 点的光强

$$I_A = I_{localA} + K_{RA}I_{RA} + K_{TA}I_{TA} \tag{8-34}$$

为了得到 I_{RA}和 I_{TA},我们要在反射方向(光线②)和折射方向(光线③)继续逆着光线跟踪下去。在反射方向,光线与另一物体 O_2 相交于 B 点。同样,B 点的光亮度与是由上述三部分组成,即

$$I_B = I_{localB} + K_{RB}I_{RB} + K_{TB}I_{TB}$$

物体 O_2 为非透明体,所以 $I_{TB}=0$。I_{RB}为反射方向(光线⑤)来的光对 B 点的贡献,它正好是光源的光强 I_p。

在 A 点的折射方向,光线与同一物体上的另一表面相交于 C 点,并从 C 点穿出。C 点的光亮度由局部光强、内部反射光强和折射光强三部分组成,即

$$I_C = I_{localC} + K_{RC}I_{RC} + K_{TC}I_{TC}。$$

这里,I_{RC}为内反射方向(光线⑥)来的光对 C 点的贡献,继续沿光线⑥跟踪下去,可求出 I_{RC}。I_{TC}为折射方向(光线④)来的光对 C 点的贡献,光线④与背景相交,因而 I_{TC}= 背景色。

最后,用 $I_{RA}=I_B$,$I_{TA}=I_C$ 代入式(8-34),可得到 A 点的光强。

跟踪光线的结束条件,除了上述的光线与光源相交和光线与背景相交这两个条件之外,还有一个条件,那就是被跟踪的光线对首交点处的光强贡献趋近于 0。

下面是光线跟踪算法的 C 语言描述。这是一个递归算法。代入 TraceRay 的有三个参数,分别是起点 start,跟踪的方向 direction 和已跟踪的深度 depth,返回的是光线 direction 的颜色。

```
Color TraceRay(start, direction, depth)
Vector start, direction;
int depth;
{
    if (depth> MAX- DEPTH) color= 黑色;
    else{
```

```
        光线与物体求交,找出离 start 最近的交点;
        if(无交点) color= 背景色;
        else{
          local_color= 用局部光照模型计算出的交点处的光强;
          用式(8-25)计算反射方向;
          Reflected_color= TraceRay(交点,反射方向,depth+ 1);
          用式(8-27)计算折射方向;
          Transmitted_color= TraceRay(交点,折射方向,depth+ 1);
          color= combine(local_color,Reflected_color,KR,Transmitted_color,KT);
          }
      }
      return(color);
  }
```

对于屏幕上的每一个象素点,都可以从视点穿过这个象素点发出这样的一条射线,调用 TraceRay 就可求出这个象素点的颜色。简单的方法是使射线穿过象素点的中心(图 8-49)。但这意味着用一根无限细的线对环境中的物体进行采样,自然会产生走样。不过,这样做的好处在于射线之间是相互独立的,适合于并行处理。

图 8-49　射线穿过象素的中心

5. 光线跟踪与反走样

由于象素点和象素点之间是离散的,因此用象素点阵组合出的图形,与真实景物之间必然存在一定的误差。比如,直线或曲线往往呈现锯齿状,细小物体在图上显示不出来等。这种现象就是图形的走样。产生走样现象的主要原因及反走样技术详见第九章。其中最简单的反走样方法就是增加采样点,或选用合适的滤波器。

光线跟踪是一个典型的采样过程,因而也会产生走样现象。光线跟踪算法本身就是很费时的,要是再过分地加密采样,显然是不合实际的。

前面介绍的算法是使光线穿过每个象素的中心。如果改成使光线穿过象素的角点,走样现象还是有所克服的(图 8-50)。这样做,只是在行方向和列方向各增加了一个采样点,不会花费很多时间。

具体做法是采用象素细分技术。如果象素的某个角点的亮度与其余三个角点的亮度相去甚远,则将这个象素均匀分割成四个小方块。这样又得到了五个新的角点。也就是说,对象素进行一次分割就增加了五个采样点(如图 8-51 所示)。用光线跟踪算法求这五个新增加的采样点的亮度,然后再比较这些小方块角点的亮度。若这些点的亮度仍相差很大,则继续分割,直到大致相等为止。这个象素点最终的亮度为这些小方块的角点亮度的加权平均。加权平均是一种最简单的滤波器。

光线跟踪算法与上述反走样方法结合起来,可以归纳如下:

(1) 对每一个象素的角点计算光线跟踪的亮度。

(2) 比较象素的各个角点的亮度,确定要进一步分割成小方块的象素。

(3) 对分割后新增的角点计算光线跟踪的亮度。

图 8-50　光线穿过象素的角点

图 8-51　象素的细分与新增的采样点

(4) 重复(2)和(3),直到各角点的亮度比较接近为止。

(5) 加权平均求各象素点的亮度。

6. 阴影与光线跟踪

在光线跟踪算法中,很容易生成物体之间的阴影。在光线与物体的每一个交点 p 处,都用局部光照模型计算 I_{local},这是假定在光源与交点 p 之间没有其它物体存在。显然,如果存在这样的物体,则交点 p 就在这个物体的阴影中。从交点 p 到光源再增加一条射线,就可以确定是否有其它物体在这条射线上(图 8-52)。

如果这条射线不与任何物体相交,则 p 点受光源的直接照射,p 点不在任何物体的阴影中。如果这条射线与某个不透明的物体相交,则说明至少有一个不透明的物体在这条射线上,p 点处在这个物体的阴影中。这时,p 点处的 I_{local} 就只有环境光一项了。

图 8-52　从 p 点到光源增加一条射线,确定 p 点的阴影

如果这条射线上的物体是透明的或半透明的,则要根据这个物体的透明度,使到达 p 点的直接照射的光强变弱,然后再根据这个变弱了的光照强度计算 p 点的 I_{local}。在这种情况下,还应考虑透明体的折射。折射会使从点光源来的光线改变方向。由于折射在简单的光线跟踪算法中很难实现,所以一般都不考虑折射情况。

五、物体间的漫反射和辐射度方法

在简单光反射和透射模型中,我们认为物体间的漫反射是一个恒定的环境光。在整体光照模型中,我们虽然可以模拟光在物体间的镜面反射和透射,但没有考虑光在物体间的漫反射。因为在光线跟踪算法中,企图在各个漫反射方向跟踪漫反射光是不可能的。光线跟踪只能解决物体间的光反射问题,而物体间的漫反射问题则由本小节介绍的辐射度方法来解决。

1. 辐射度方法

辐射度方法是物体空间中的一个真实图形显示算法。与光线跟踪算法计算屏幕上各象素点的亮度不同,辐射度方法计算景物表面上各点的亮度,因此它的计算结果与视点无关。在求出了景物表面上各点的亮度之后,我们可以再根据视点的位置,调用前面介绍过的双线性亮度插值算法,求出显示屏上各象素点的亮度。

辐射度方法把景物和光源视为一个封闭的系统,并假定构成景物的各个曲面片都是理想的漫反射面,而且可以近似成一系列小的平面片。所谓辐射度,是指在单位时间内从曲面上单位面积漫反射出去的光能量,记作 B。由于曲面是理想的漫反射面,所以对于近似曲面的小平面片,在近似程度足够高的情况下,可以假设每一小平面片的漫反射各向均等,且漫反射光能量为一常值,即每一小面的辐射度 B 不变。辐射度方法确定封闭系统中各景物之间的漫反射,求出各面片的辐射度。

如图 8-53 所示,对于面片 i,微分元 dA_i 上的辐射度可根据能量守恒定律,由下式求得:

$$B_i dA_i = E_i dA_i + R_i \int_j B_j F_{ji} dA_j \tag{8-35}$$

这里,E_i 为面片 i 本身作为光源向空间均匀散射出去的光能;R_i 为面片 i 的漫反射率;积分式为曲面片 j 向微分元 dA_i 散射的光能,是曲面 j 上微分元 dA_j 的辐射度 $B_j dA_j$ 和形状因子 F_{ji} 的乘积的积分,这样的积分应包括环境中所有的曲面。

形状因子 F_{ji} 表示两个面片 A_i 与 A_j 之间相互辐射的能量交换,取决于两个面片的方向和相对距离,它的值等于面片 A_j 向面片 A_i 直接辐射的能量与面片 A_j 向空间各个方向均匀辐射的能量之比,即:

图 8-53　辐射度方法图示

$$F_{ji} = F_{A_j A_i} = \frac{1}{A_j} \iint_{A_j A_i} \frac{\cos\phi \cos\phi}{\pi^2} dA_i dA_j \tag{8-36}$$

根据积分可交换原则,有

$$F_{ji} A_j = F_{ij} A_i \tag{8-37}$$

将上式代入式(8-35),并除以 dA_i 可得面片 i 的辐射度

$$B_i = E_i + R_i \int_j B_j F_{ij} \tag{8-38}$$

在整个封闭系统中,由于曲面片被离散成了一系列小平面片,且假设每个小面片的辐射度为一常数,所以上式的积分可近似成对所有小面片的求积,即

$$B_i = E_i + R_i \sum_{j=1}^{n} B_j F_{ij} \tag{8-39}$$

对环境中的每一个面片都存在上述关系式,所以可以得到如下 n 个联立方程:

$$\begin{bmatrix} 1-R_1F_{11} & -R_1F_{12} & \cdots & -R_1F_{1n} \\ -R_2F_{21} & 1-R_2F_{22} & \cdots & -R_2F_{2n} \\ \vdots & \vdots & & \vdots \\ -R_nF_{n1} & -R_nF_{n2} & \cdots & 1-R_nF_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} \tag{8-40}$$

E_i 只在面片 i 本身为发光体的表面时才不为零,它们代表了封闭系统中光能量的来源。F_{ij} 只与景物的几何形状和位置有关,可由式(8-36)算出。由于小面片全假设是平面,所以 $F_{ii}=0$,这是因为由平面辐射出去的光不经其它面的反射不会再碰到这个面本身。R_i 是已知的漫反射率。对于 RGB 三基色,R_i 的三个分量可能不同,所以式(8-40)要分解成三组联立方程,分别求对应 R、G、B 的各面片的辐射度 B_i。

式(8-40)是一个线性方程组,不难从中解出各面片的辐射度 B_i。这实际上就等于求出了各个面片的光亮度 I_i。根据双线性亮度插值算法,用这组亮度作为初值,就可算出面片各顶点的亮度,再针对特定的视点可插值算出显示屏上各象素点的亮度值,从而得到整个景物的图象。

具体算法可描述如下:

(1) 按式(8-36)计算形状因子 F_{ij}。

(2) 按式(8-40)解方程组,求各面片的辐射度 B_i。

(3) 计算面片各顶点的亮度。

(4) 根据观察参数(视点、观察方向、视平面)将物体空间转换为图象空间。

(5) 用双线性亮度插值法计算各象素点的亮度。

从上述算法可以看到,计算辐射度与观察方式无关。这一特点使我们在计算出整个场景中各面片顶点的亮度之后,迅速地显示出以不同的观察方式观察这个场景的真实感图形。

上述算法的另一个特点就是形状因子的计算与光源 E_i 无关、与漫反射率 R_i 无关,所以即使重新调整了场景中光源的光强度、景物表面的颜色(可由漫反射率的 RGB 三个分量反映出来),也无需重新计算形状因子。

2. 形状因子的计算

我们用半立方体的方法求形状因子的值。首先用微分元 dA_i 代替 A_i 整个面片,这样可将面片到面片的形状因子 F_{ij} 的计算式近似成微分元到面片的形状子 $F_{dA_iA_j}$:

$$F_{ij} \approx F_{dA_iA_j} = \int_{A_j} \frac{\cos\phi_i \cos\phi_j}{\pi r^2} dA_j \tag{8-41}$$

下面我们考虑如何计算微分元 dA_i 到面片 A_j 的形状因子。dA_i 是面片 i 的中心点,即面片 i 相对于面片 j 的平均位置。这种近似的精确度取决于两个面片的面积与它们之间的距离比。接下来,以微分元 dA_i 为中心,在面片 i 的正上方作一个单位半球,将面片 j 投影到这个半球上,按式(8-41)算出形状因子 F_{ij}。

但是,对于一个复杂的多物体的场景,我们还要把这个半球近似成半立方体(图 8-54),因为平面投影比球面投影的计算量要小得多。半立方体以面片 i 的中心为中心,其 Z 轴与面片 i 的法向一致,其表面被均匀地划分成足够小的正方形。我们可以把这些小正方

形也视为“象素”。

场景中其它面片都投影到这个半立方体上。图 8-55 显示了面片投影到半立方体上的情景。有可能有两面会投影到同一个“象素”上,对于这样的两个面,要在投影方向上进行深度比较,较远的一个面要从这个“象素”上去掉,因为从半立方体的中心看过去,它被较近一个遮住了,看不到。这一步跟深度缓存消隐算法很相似,也要建立一个深度缓存,记录在半立方体的各个“象素”上能看到的面片。

图 8-54　面片 i 中心的半立方体被均匀划分成“象素”

图 8-55　面片 j 在半立方体上的投影

设 q 是半立方体上的一个“象素”,面片 j 的微分元 dA_j 投影到它的上面,则微分元 dA_i 到微分元 dA_j 的形状因子 $F_{dA_i dA_j}$ 可近似成微分元 dA_i 到“象素”q 的微形状因子 ΔF_q,即:

$$\Delta F_q = F_{dA_i dA_j} = \frac{\cos\phi_i \cos\phi_j}{\pi r^2} dA_j \tag{8-42}$$

将所有这样的微形状因子全加起来,就得到了微分元 dA_i 到面片 A_j 的形状因子。即

$$F_{ij} = \sum_q \Delta F_q \tag{8-43}$$

上式表明,对式(8-41)的形状因子的计算可简化为面片 j 在面片 i 的半立方体上的投影“象素”的微形状因子的求和运算。这样,对于任意两个面片,不管它们的形状和空间位置如何,只要它们在面片 i 的半立方体上的投影重叠,它们与面片 i 的形状因子都相等。

对于所有的面片都进行这样的加法运算,就可以得到式(8-40)方程组中第 i 行的 n 个形状因子。再对所有的面片,重复这种半立方体的操作,就可以求出式(8-40)中各行的形状因子。

半立方体的每一个“象素”的微形状因子都预先计算好存放在一个查寻表中。例如,对于在半立方体块顶面的“象素”q(图 8-56),

$$\Delta F_q = \frac{1}{\pi(x^2 + y^2 + 1)^2} \Delta A \tag{8-44}$$

这是因为,

$$r = \sqrt{x^2 + y^2 + 1}$$

图 8-56　微形状因子 ΔF_q 的计算图示

$$\cos\phi = \cos\varphi = \frac{1}{\sqrt{x^2 + y^2 + 1}}$$

$$\Delta F_q = \frac{\cos\phi\cos\varphi}{\pi r^2}\Delta A$$

同样,对于半立方体侧面的“象素”q,

$$\Delta F_q = \frac{z}{\pi(x^2 + y^2 + 1)^2}\Delta A \tag{8-45}$$

计算形状因子 F_{ij} 的算法可归纳如下:

(1) 按式(8-44)和式(8-45)计算微形状因子查寻表。

(2) 对于场景中的每一个面片 i,做以下工作:

(3) 以面片 i 的中心为原点,法向为 Z 轴方向建立坐标系和半立方体,并将场景中的所有的面片都变换到这个坐标系中。

(4) 对半立方体上的每一个“象素”q,用深度缓存消隐算法求各面片在其上的投影,并将在投影方向离该“象素”最近的面片号记入深度缓存 ZB。

(5) 对于每个面片 j,根据深度缓存 ZB 中记录的面片号和微形状因子查寻表,将与面片 j 有关的半立方体“象素”的 ΔF_q 加起来,形成面片 i 到面片 j 的形状因子 F_{ij}。

以上我们介绍了如何用辐射度方法求物体的漫反射光强,并结合双线性亮度插值算法提供了一种生成图形的手段。但上述的辐射度方法仅适用于由理想漫反射面所组成的场景,没有考虑镜面物体和透明体。如何将镜面反射和折射也包含到辐射度方法中,也就是如何将光线跟踪算法和辐射度方法结合起来,这里就不再深入讨论了。

六、阴影生成技术

在光线跟踪算法中,我们已介绍了如何结合光线跟踪算法,产生阴影的方法。这里我们介绍生成阴影的一般方法。

阴影是由于物体截断了光线而产生的,所以,如果光源位于物体一侧的话,阴影总是位于物体的另一侧,也就是与光源相反的一侧。从理论上来说,当观察者的位置与光源的位置重合时,是看不到阴影的。只有当观察者的位置离开了光源,才能看到阴影。

对于点光源,阴影算法与隐藏面消除算法相似。消隐算法是确定哪些面从视点看过去是可见的,而阴影算法则要确定哪些面从光源位置看过去是可见的。显然,从视点以及从光源看过去都是可见的面不会落在阴影中,只有那些从视点看过去是可见的,而从光源看过去是不可见的面,肯定落在阴影之内。

根据以上分析,生成一幅有阴影的图象,要做以下两个工作。首先是生成有关阴影区的描述;其次是决定在观察者看来,哪些面是可见的,这些可见面上的阴影形状如何、以及可见面的颜色和亮度等等。

有关阴影生成的算法如下:

(1) 将视点移到光源位置,用多边形区域排序消隐算法,将多边形分成两大类:向光多边形和背光多边形。所谓向光多边形是指那些从光源看过去是可见的多边形;而背光多边形则是指那些从光源看过去是不可见的多边形,包括被其它面遮挡了的多边形和反向面多边形。向光多边形不在阴影区内,背光多边形在阴影区内。

(2) 将视点移到原来的观察位置,对向光多边形和背光多边形进行消隐,并选用一种光照模型计算多边形的亮度,就可得到有阴影效果的图形。若选用简单光反射模型,则对于背光多边形,由于不能得到光源的直接照射,只有环境光对其亮度有贡献。

习　　题

1. 消隐的意义是什么?
2. 请在计算机上用程序语言实现深度缓存消隐算法。
3. 设计一个算法,能有效地决定一条直线是否贯穿由 x_{min}、y_{min}、z_{min} 和 x_{max}、y_{max}、z_{max} 定义的长方体区域。
4. 如何判断一点是在一个平面的正面、反面、还是在平面上?
5. 设计一个算法,确定一点是在多边形内、多边形外,还是在多边形的边界上。并估计算法复杂性。
6. 已知有 n 个顶点的多边形,如何求多边形的平面方程?
7. 如何确定一条直线与一平面平行?
8. 物体表面的颜色由哪些因素决定?
9. 写出各个光照模型之间的区别、能模拟的光照效果和不能模拟的光照效果。
10. 设计一种算法,将亮度插值法和法向量插值法结合起来,仅在有高光域的地方用法向量插值,其余部分都用亮度插值。
11. 编写一个简单光反射模型的程序,显示 20 个球。其中,K_d 和 K_s 分别取(0, 0, 1.0),(0.25, 0.75),(0.50, 0.50),(0.75, 0.25),(1.0, 0.0),n 取 5, 10, 20 和 40。并在显示屏幕上观察各个球的光照效果。

第九章　数字图象处理技术初步

数字图象指的是一组数值组成的阵列，每一个数值用以描述图象中每一个象素的特性。通常，这些数值是一个区域内的实数的整数表示。例如，从 0.0 到 1.0 这一区域的实数可用 0 到 255 的整数来表示，这些数字常常是用来表示图象中一点的灰度等级。

正如第一章中所述，尽管计算机图形学及图象处理原本属于两个不同的技术领域，但是由于所生成和处理的对象都是数字图象，因此，又有着不可分割的联系。随着科学技术的发展，二者的结合日趋紧密。当我们采用计算机图形生成技术产生一幅图象之后，要对图象加以变换或修改，一种方法是回到产生该幅图象的几何模型上去，加以变换或修改，然后再重新产生一幅图象，这当然是比较麻烦的。那么，能不能直接由一幅图象变换为另一幅图象呢？答覆是肯定的，这就是图象变换技术。在科学计算可视化这一图形学的重要应用领域中，一种重要的显示三维数据场的方法是体绘制技术。这一方法将三维空间的离散数据场直接变换为二维数字图象，其实质是一个重新采样问题，这也是数字图象处理中经常遇到的问题。计算机动画技术中的一个重要方向是将计算机生成的图形和摄象机摄入的背景合成在一起，形成逼真的画图，这就要解决图象的合成问题。而多媒体技术的发展，将使得人与计算机的界面和人与人的界面不只是字符和数字，而发展成为图象，语言和文字组成的多种媒体。所有这一切都表明，数字图象技术将与计算机图形学紧密结合在一起，日益深入到人类的工作和生活之中，因此，我们认为有必要在计算机图形学这本教材中增加一章“数字图象处理技术初步”，作为计算机图形学工作者了解数字图象处理技术的一个入门，为进一步的学习打下基础。

在本章中，我们将首先介绍数字图象处理的重要理论基础之一——采样理论。然后介绍与计算机图形学紧密结合的一些内容，如图象变换、及反走样技术等。

第一节　采样理论

一、几个基本概念

采样理论是信号处理领域的一个重要理论，因此，我们首先要介绍信号处理领域的几个基本概念。

1. 信号(Signal)

信号是表示信息的一个函数。它通常表示为时间的函数，但是，也可以表示为其它变量的函数。因此，我们可以认为数字图象是空域中的信号，是空间坐标的函数，而不是时域中的信号，也不是时间的函数。虽然数字图象是两个独立变量(X 和 Y)的二维函数，但是，为了便于讨论采样理论，我们常常只讨论一维的情况。当我们用一个极薄的片子切割二维图象时，我们就可以得到一个沿 X 轴的一维图象。它表示该图象沿一条水平线的光照强度的分布情况(见图 9-1)。

图 9-1

(a) 为一幅图象; (b) 为(a)中图象沿扫描线 α的光照强度分布

2. 连续信号(Continuous Signal)和离散信号(Discrete Signal)

在空间上一个连续的域内定义的信号称为连续信号,而在空间内一组离散点上定义的信号则称为离散信号。从第三章可知,在扫描变换之前,三维物体在观察平面上的投影可以看作是二维连续信号。在投影区域内每一个无限小的点上的数值表示该点的光照强度。与此相反,在图形系统的帧缓存中,由象素点的数值组成的矩阵是一个二维的离散信号。它的值仅仅定义在该矩阵的离散点上。扫描变换及其它有关的绘制算法就是要决定矩阵中有限的象素点上的光照强度值,以便最好的表现相应的二维连续图象。

3. 采样(Sampling)及样本(Samples)

对于一个连续信号来说,当它的参数改变时,可以引起函数值的非常迅速的变化(即具有很高的频率),从而使连续信号可以拥有丰富的变化细节。而离散信号则不然,它只能在确定的离散点上产生函数值的变化,因而只具有某一个最大的变化率。因此,当我们将一个连续信号转换为一个有限的离散点阵时,必定会引起信息的丢失。我们的目的是,使得信息的丢失尽可能的少,以至于使得该离散信息点阵可用来表示原有的连续信号,而看起来和原来的图象极其近似。

从一个连续信号中选择有限个数值的过程称为采样(Sampling),而被选择的数值则称为样本(Samples)。当取得了一系列样本之后,我们还必须由这些样本重新产生原始的连续信号,这一过程称为重构(Reconstruction)。这里,我们可以举一个例子,在图象显示器帧缓存中的象素点阵是由图形系统的显示硬件实现重构的。它将帧缓存中的光照强度转变为连续的模拟信号,施加于 CRT 的电子枪,在屏幕上显示出图象来。

二、采样理论

1. 图象信号的空域和频域表示

当我们在空间域中讨论图象信号时,它是以空间位置的函数来表示的。如果我们要在频域中讨论图象信号,又将如何呢?

根据数学中讲授的傅里叶级数的性质，一个周期函数可以用一系列具有不同幅值和相移的正弦函数之和来表示。这些正弦函数的频率是该周期函数基波频率的整数倍(如图9-2)。而一个非周期函数可以被认为是周期为无限长的一个函数，这时得到一个傅里叶级数的极限形式，原始的非周期函数不能用多个正弦函数之和来表示，而必须表示为一个连续频谱的积分，称为傅立叶积分。其基波的频率则趋近于零。

图 9-2　一个方波可表示为一系列正弦波之和

对于图象信号而言，一般具有非周期的性质，而且其空间范围是有限的。我们可以认为该图象信号在图象区域之外的值为零。这样的图象信号可以按非周期函数来处理。图9-3 中表示出图 9-1 中的一维图象信号在频域中的前 10 个成份。从图中可以看出，空间域中的每一个信号在频域中都有其相应的表示，反之亦然。

设一维图象信号以 $f(x)$ 表示，则其傅里叶积分 $F(u)$ 为：

$$F(u)=\int_{-\infty}^{\infty} f(x)[\cos 2\pi ux - i\sin 2\pi ux]\,dx \qquad (9\text{-}1)$$

式中，u 表示 Sine 和 Cosine 函数的频率，$i=\sqrt{-1}$。$F(u)$ 表示在原始信号 $f(x)$ 中频率为 u 的成份。显然，它是一个复数，可以写为 $R(u)+iI(u)$。其中，$R(u)$ 和 $I(u)$ 分别为 $F(u)$ 的实部和虚部。于是，我们可将 $F(u)$ 的幅值表示为：

$$|F(u)|=\sqrt{R^2(u)+I^2(u)}$$

其相角为：

$$\phi(u)=\tan^{-1}\frac{I(u)}{R(u)}\ 。$$

在这里，傅里叶积分将图象信号在空域中的表示 $f(x)$ 转换为相应的频域表示 $F(u)$，因此，也称为傅里叶变换。

反过来，可用傅里叶逆变换将频域中的 $F(u)$ 逆变换为空域中的 $f(x)$。即：

$$f(x)=\int_{-\infty}^{\infty} F(u)[\cos 2\pi ux + i\sin 2\pi ux]\,du \qquad (9\text{-}2)$$

一个信号的傅里叶积分 $F(u)$ 常常表示为频率的幅值与频率的关系，而忽略相角的偏

图 9-3　图 9-1(b)中的一维图象在频域中的前 10 个成份

移。图 9-4 表示出三个不同信号的空域表示及其相应的频域表示。这样的频域表示称为该信号的频谱。在这里,特别要指出的是,在空域表示中,横坐标是象素中心点的序号;而在频域表示中,横坐标是每两个象素中心点之间的周期数(与通常的频率概念——每个时间单元内的周期数——是不同的)。

当我们研究的对象不是连续的非周期信号,而是均匀分布的离散样本时,应当采用傅立叶变换的离散形式。即

$$F(u) = \sum_{0 \le x \le N-1} f(x)[\cos(2\pi ux/N) - i\sin(2\pi ux/N)]$$
$$0 \le u \le N-1 \qquad (9\text{-}3)$$

其逆变换为:

$$f(x) = \frac{1}{N} \sum_{0 \le u \le N-1} F(u)[\cos(2\pi ux/N) + i\sin(2\pi ux/N)]$$
$$0 \le x \le N-1 \qquad (9\text{-}4)$$

上二式中,N 为均匀分布的离散样本数。

2. 滤波(Filtering)

采样理论告诉我们,当从一个信号中抽取样本的频率大于该信号频谱中最高频率(f_h)的两倍时,才能从它的样本中正确的重构该信号。这一采样频率下限称为奈魁斯特频率(Nyquist Frequency)。对于这一结论,我们在这里不准备给出正式的证明,而只是给出一个说明。

假设在某一信号中,它的最高频率成份的频率为 f^h(如图 9-5 所示)。从图中可以看出,为了反映这一最高频率成份的整体形状,最低限度需要在一周内取两个采样点。即在单位时间内需要有 $2f_h$ 个采样点。事实上,要用 $2f_h$ 个采样点来反映最高频率成份的整体形状还仅仅当这些采样点恰好取在正弦波的极大值和极小值处才能成功(图 9-5(a))。如

图 9-4　三个不同信号的空域表示和频域表示

(a) 正弦波;　(b) 方波;　(c) 一幅图象

图 9-5　用奈魁斯特频率采样的例子

(a) 峰值点采样;　(b) 峰值点间采样;　(c) 过零点采样

果将这些采样点取在正弦波的其它位置,还不能正确反映这一频率成份的整体形状(图 9-5(b)),甚至会误认为它是零(图 9-5(c))。

如图 9-5 所示,如果采样频率低于奈魁斯特频率,那么所得到的结果就和在较低频率的原始信号中采样的结果是一样的。这种在信号重构中将高频信号误认为低频信号的现象称为走样(aliasing)。

图 9-6　采样频率低于奈魁斯特频率的例子

一个信号的频谱分布决定了该信号的形状。变化剧烈的波形含有丰富的高频分量。在图 9-7 中,物体的投影 A, B, C, D 具有非常清晰的边界,因而含有无限高的频率成份。这样的图形,不可能用有限的样本将它理想的表示出来。如果按图 9-7 的方式采样,物体 A, C 被采样了,而物体 B, D 则被忽略掉了。在显示器屏幕上生成的图象,沿着一条边界出现的锯齿状就是由于物体投影边界的不连续性引起的。因为在象素矩阵中,一个象素点或者落在投影区之内,或者落在投影区之外,从而造成了沿物体投影边界处的锯齿状,这也是走样问题的另一种形式。

图 9-7　物体投影边界的采样问题

那么,有什么办法可以消除上述走样现象呢?本段所讨论的滤波技术就是常用的一种方法。

如果我们能将原始图象中的高频成份滤掉,那么滤波后的图象就可以通过采样频率较低的样本加以重构。高频成分滤去得越多,所需的采样频率越低,但是,重构后的图象也会比原始图象更加模糊一些。

一个完全的低通滤波器可以将高于某一截止频率的高频分量完全滤去,而让低频分量通过。这样的低通滤波可以在频域内将信号的频谱乘以一个脉冲函数来实现(如图 9-8 所示)该脉冲函数可表示为:

$$S(u)=\begin{cases}1 & \text{当 } -k\le u\le k\\ 0 & \text{当 } u \text{ 为其它频率时}\end{cases}$$

显然,当信号频谱和此脉冲函数相乘时,该脉冲函数可将信号频谱中高于 k 的分量去掉,而低于或等于 k 的分量则保持不变。

从这一过程来看,当我们需要对一幅图象进行低通滤波时,似乎需要首先将该幅图象的空域表示转换为频域表示,然后乘以适当的脉冲函数,再将所得乘积反变换为空域表示,这当然是正确的。但是,由于信号的空域表示和频域表示之间存在着一个重要关系,使

图 9-8　频域中的低通滤波

(a) 信号频谱;　(b) 低通滤波器频谱;　(c) 信号频谱与低通滤波器频谱相乘;　(d) 滤波后的频谱

得上述过程成为不必要的。这一重要关系是: 两个傅里叶变换在频域中的乘积与它们的逆变换在空域中的卷积(Convolution)运算的结果是等价的。

两个信号 f(x)和 g(x)作卷积运算的结果是一个新的信号 h(x), 可表示为 f(x)* g(x)。则 h(x)的定义如下: 信号 h(x)每一点的数值是信号 f(x)与滤波函数 g(x)的乘积的积分, 但此时应将 g(x)移动到使其原点恰好位于该点的位置上。如果以 τ表示卷积运算中的积分的哑变量, 则卷积的定义可写为:

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) g(x - \tau) d\tau$$

这也可以理解为, h(x)上每一点的数值是信号 f(x)上围绕该点的相邻各点数值的加权平均, 相邻点的范围及加权系数由滤波函数决定。在这里, 滤波函数常被称为“卷积核”或“滤波核”。反过来, 两个傅里叶变换在频域中的卷积也与它们的逆傅里叶变换在空域中的乘

积是相等的。

根据上述重要性质,一个信号的频域表示乘以一个脉冲函数就等价于它的空域表示与该脉冲函数的逆傅里叶变换作卷积运算。频域中的脉冲函数经逆傅里叶变换后在空域中的表示称为 Sinc 函数,其定义为 $\sin(\pi x)/\pi x$,(见图 9-9)。从图中可以看出,Sinc 函数有一个不好的性质,即在离开原点任意远处,它仍然不为零。这就给卷积运算带来不便。为此,人们常选用其它的滤波函数,图 9-10 中给出常用滤波函数的空域表示和频域表示。

图 9-9　频域中的脉冲函数对应于空域中的 Sinc 函数

图 9.10　常用滤波函数的空域表示和频域表示

(a) 脉冲——Sinc　(b) 三角——Sinc^2;　(c) 高斯——高斯

图 9-11　采样及重构

(a) 原始信号；　(b) 采样信号；　(c) 采样信号准备用 Sinc 函数进行重构；

(d) 用 Sinc 函数重构后的信号；　(e) 采样信号准备用三角函数进行重构；

(f) 用三角函数重构后的信号。

3. 重构与重新采样

重构(Reconstruction)是由一个信号的样本重新构造出该原始信号的过程。样本是离散的,而原始信号是连续的,因此,这也是一个由离散信号重新产生连续信号的过程。当我们以有限的采样频率对原始信号进行采样时,所产生的离散采样信号的频谱范围是无限的。显然,重构原始信号有两条途径:一是在频域内以一个脉冲函数乘以离散信号的频谱,去掉高频成份,然后再将乘积作逆傅里叶变换从而得到重构的结果。另一种方法是在空域内将离散采样信号与 Sinc 函数直接作卷积运算得到的。因此重构过程实际上也是一个滤波过程,也可采用其它的滤波函数来实现,这时,滤波函数也称为重构核。

图 9-11 表示出一个连续信号的采样、重构过程,用了两种不同的重构核。图中不仅给出了空域表示,而且给出了相应的频域表示。

在计算机图形学和图象处理中常常需要将一幅图象变换为另一幅图象。例如,图象的几何变换、纹理映射等。这实际上是一个离散信号的重新采样过程。实现重新采样(Resampling),从理论上说应有以下几个步骤:

① 选择适当的重构核函数,对离散的采样信号进行卷积运算,重构连续的原始信号。

② 一般说来,结果信号的采样频率是已知的,由此可计算出被采样信号的奈魁斯特 Nyquist 频率极限,采用低通滤波函数去掉高于这一极限的频率成份。

③ 对滤波后的函数重新采样。

由于图象是一个二维的离散信号矩阵,因此需要在二维平面上实现以上步骤,这是相当复杂的。因此,常常根据需要简单化和合并一些步骤来实现这一过程。

第二节　图象的空间变换

由于图象的空间变换被广泛的应用于计算机图形学、计算机视觉、医学图象处理及遥感图象处理等领域中,因此,在本节中,我们将对图象的空间变换作简单介绍。

图象的空间变换指的是在某图象的全部象素点与其变换后的对应部分之间的映射关系。在一幅输入图象中,每一个象素点的坐标及其函数值是已知的,而输出图象则是由变换后的象素点组成的。这二者之间的关系可以用两种不同的形式表示,其一是以输入坐标为自变量,用其变换后的值表示输出坐标,其二则与此相反。于是,我们有

$$[x, y] = [X(u, v), Y(u, v)],$$

或

$$[u, v] = [U(x, y), V(x, y)]$$

其中,[u, v]指的是输入象素的坐标,[x, y]是其相应的输出象素的坐标。X, Y, U, V 是定义空间变换关系的映射函数。因为 X, Y 将输入象素映射到输出象素,故称为正向映射函数,而 U, V 则将输出象素映射到输入象素,故称为逆向映射函数。

一、正向变换

正向变换是由正向映射函数[X, Y]将每一个输入象素拷贝到输出图象的相应位置上。图 9-12 表示出在一维情况下正向变换的示意图。离散的输入和输出象素分别由位于整数坐标值上的一个象素链表示。每一个输入象素通过正向映射函数后得到一个新的输

出坐标值。输入象素坐标是均匀分布的整数值，而变换后的输出值则可能是不均匀分布的实数值。由于输出象素也是定义在整数值上的有限个元素，因此，如果在输入和输出之间进行点到点的映射，则有可能出现空洞或重叠。在图 9-12 中，变换后，输入象素链上的 F、G 两点都映射到输出象素链的 G 上，形成重叠，而 F 则成为一个空洞。

这个问题，可以通过四角映射法来解决。在这种方法中，将每一个象素点看成一个小的方形面片。经过映射后，它可以变换成输出图象中任意形状的四边形，这样，可以保证变换后的图象仍然是连续的。由于输入图象中的小四方形面片在变换后可以覆盖几个输出象素或落入一个输出象素中(如图 9-13)，因此需要计算输入象素对每一个输出象素的贡

图 9-12　正向变换　　图 9-13

献。当一个输出象素被多个输入象素覆盖时，要把这些输入象素对该输出象素的贡献集成起来才能得到该输出象素的强度值。在集成时，每一个输入象素对该输出象素的贡献与该输入象素的变换结果所覆盖的输出象素的面积成比例。即输出象素的强度值为 $\sum_{i=0}^{N} W_i f_i$，其中 f_i 是第 i 个输入象素的强度值，W_i 是加权值，它反映了被第 i 个输入象素的变换结果所覆盖的输出象素的面积的份额，显然，为了得到这一结果，必须进行求交计算。N 是对该输出象素有贡献的输入象素的数目。虽然四角映射法可以避免空间和重叠的产生，但需要作费时的求交运算。

二、逆向变换

与正向变换相反，逆向变换是通过映射函数 U，V 将每一个输出象素点变换到输入图象上，再将相应的输入图象象素点的数值拷贝到输出象素。图 9-14 表示出在一维情况下逆向变换的示意图。在此图中，每一个输出象素通过逆向变换映射到输入象素。与前相同，输出象素位于整数坐标值上，而经逆变换后却得到实数值。与点到点的正向变换不同，逆向变换将对每一个输出象素进行计算，从而不会留下空洞。但是在输入一侧却有可能存在输入象素不被采样的问题，如果有较多的输入象素不被采样，则会在输出图象中引起不应有的畸变。因此仍然需要采用滤波技术，来集成有关输入象素对输出象素的贡献。尽管如此，采用逆向变换后，输出图象中位于裁剪窗口之外的象素就不必计算了。当输出屏幕是串行写入而输入图象可以完全存储在内存中时，逆向变换是一个有用的方法，也是一个常

用的方法。

Feibush, Levoy 和 Cook 给出的算法就是一个采用逆向变换方法实现图象变换的很好的例子。这个算法的目的是要将一幅图象(称为原图象)变换为另一幅图象(称为目标图象)。目标图象的边界称为目标多边形,正是目标多边形中的每一个象素点的强度值是需要加以计算的。这一算法本来是用于纹理映射的,但是也可以用来实现图象变换,其基本原理是一样的。

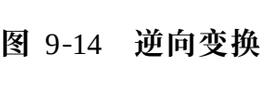

图 9-14　逆向变换

这一算法首先将目标多边形各顶点的坐标经过给定的逆向变换映射至原图象平面,得到原图象多边形。接着,选择一个适当的滤波函数,该滤波函数在中心点附近的一个小范围内具有非零值。该范围称为滤波函数的支撑范围(Support)。为该支撑范围构造一个外接矩形,然后将该外接矩形的中点移至目标多边形所在平面的每一个象素上,当此矩形与目录多边形相交时,我们就认为该象素点是在目标图象之内。这时,滤波函数的支撑范围也称为该象素点的卷积模板,该矩形也称为该象素点的包围矩形。我们再将每一个象素点的包围矩形逆变换到原图象中,一般说来,它将是一个任意四边形,再将这个任意四边形用原图象多边形予以裁剪。于是,落在裁剪后的四边形内的原图象象素点将对目标象素作出贡献,它们的强度值将由滤波器给出的权值进行加权平均后赋给目标象素,从而得出该目标象素的强度值。加权平均的具体实现方法可参考有关文献。图 9-15 给出了这一算法的示意图。

图 9-15　Feibush, Levog 和 Cook 算法示意图

第三节　反走样技术

在第一节中曾经提到,如果采样频率低于被采样信号的奈魁斯特频率时,那么就会在信号重构中将被采样信号误认为是某一低频信号,这种现象称为走样,在信号处理中又称

为混叠。常见的一种走样现象就是在光栅显示屏幕上绘制一条直线或圆弧时普遍存在的阶梯状边界。这是因为在对直线或圆弧作扫描变换时,屏幕上的离散象素点要么被置为图素的颜色,要么不作改变,从而引起阶梯状边界。

当我们在具有白背景色的屏幕上画一条斜率在 0 至 1 之间、宽度为一个象素的黑色直线时,在直线穿过的每一列象素中,算法将离该直线最近的象素置为黑色(如图 9.16(a)所示)。当直线从一列象素移至另一列象素时,如果离直线最近的象素不在同一行上,就会出现一个鲜明的阶梯,这就是走样现象。从傅里叶变换的角度看来,一个图形的轮廓线具有无限高的频率成份,而扫描变换实质上也是一个采样过程。由于屏幕上的象素点是有限的,由有限个采样点对无限高的频率进行采样,必然出现走样。消除走样的方法称为反走样(Antialiasing),本节将介绍几种主要的反走样方法:

一、提高屏幕象素点的分辨率

如图 9-16(b)所示,如果我们将屏幕象素点的分辨率在水平及垂直方向上均提高一倍,那么,同样的一条直线所穿过的象素点的行数和列数将增加一倍,因百所形成的阶梯数也将增加一倍,但是在 x,y 两个方向上阶梯的大小均只有原来的一半,从而使得图形看起来要好一些,这是因为采样频率提高了一倍的原故。但由此而引起存储空间的扩大,存储器频带的加宽,扫描变换时间也要相应的增加,因而代价是很大的。

图 9-16　(a)、(b)走样现象举例

二、不加权的面积采样法

在一般的扫描变换算法中,将一条直线看作是一个宽度为零的理想图素,而象素点在赋值时是被当作一个整体来看待的,要么被赋为图素的颜色,要么不作改变,从而引起鲜明的阶梯状边界。

但是,如果认为一条直线的宽度为非零值,情况就不同了。如图 9-17 所示,一条直线被看作是一个具有一定宽度的矩形,在屏幕上覆盖了一部分象素点。同时,我们认为屏幕上的象素点是一系列相互连接的小方格的二维矩阵,形成一个二维网格,而象素的中心点则位于网格的定义点上。在这种情况下,我们假定一条直线对一个象素点的值的贡献正比于该直线所覆盖的该象素点的面积的份额。如果一个象素点小方格完全被直线所覆盖,则在黑白显示器上该象素点应赋值为黑色,如果部分地被该直线所覆盖,则应赋值为灰色,

其值正比于被直线所覆盖的面积的份额(图 9-18)。显然,这种算法只能在具有多灰度等级的显示器上实现。其结果是使图形的边界不具有鲜明的阶梯状,而使得在黑白两色之间有一个平缓的过渡,从而使图形看起来要美观得多。

图 9-17　具有非零宽度的直线

图 9-18　象素值正比于被直线所覆盖的面积

这种方法称为不加权的面积采样法,它具有两个特点:

1) 只有当表示直线的矩形与某象素点小方格相交时,才对该象素点的值有贡献,如果不相交,是没有影响的。

2) 当表示直线的矩形与某象素点小方格相交时,对该象素值的贡献,仅与该矩形所覆盖的小方格的面积大小有关,而与所覆盖面积距象素中心点的远近无关。

三、加权的面积采样法

与不加权的面积采样法不同,在这种方法中,当直线形成的矩形与某象素点小方格相交时,对该象素点的贡献不仅与该矩形所覆盖的面积大小有关,而且与所覆盖的面积距象素中心点的远近有关;离中心点越近,贡献越大;离中心点越远,贡献越小。而且,为了避免当图素移动时在屏幕上引起闪烁,加权的面积采样法还可以使图素离象素点很近但并不相交时,就开始对该象素点的值有影响。

从采样理论的角度来看,不加权的面积采样相当于一个象素具有一个长方体形的滤波函数,在该象素的小方格范围内,滤波函数的值相等(如图 9-19)。该滤波函数决定了表

示直线的矩形对象素点贡献的权值。因此只有当直线与象素小方格相交时，该滤波函数才起作用，而对该象素值的贡献则决定于所覆盖的小方格面积乘以滤波函数的值。由于滤波函数的值在小方格范围内是常数，因此，对该象素的贡献只与覆盖面积的大小有关，而与覆盖面积的位置无关。

图 9-19　不加权面积采样法的长方体形滤波函数

加权的面积采样法则相当于一个象素具有一个非恒定值的滤波函数。例如，在象素点中心处，滤波函数的值最大，随着离象素点中心的距离的增加，滤波函数呈线性衰减，如设滤波函数在 x，y 方向上是对称的，那么它是一个圆锥形的滤波函数，该圆锥的底面(即滤波函数的支撑范围)的半径应等于单位网格的距离(图 9-20)。由于圆锥形滤波函数的支撑范围大于一个小方格，因此当直线与支撑范围相交但尚未与小方格相交时，就会对该象素点的值有贡献。而且，直线对象素点的贡献决定于所覆盖的小方格面积乘以滤波函数的值，因滤波函数在小方格范围内不是常数，因而与覆盖面积所在的位置有关。

图 9-20　加权面积采样法的圆锥形滤波函数

习　　题

1. 现有一脉冲函数 f (x)，当- 1≤x≤1 时，f (x)= 1。当 x 为其它值时 f (x)= 0。试证：f (x)的傅里叶变

换为多重 Sinc 函数。

2. 利用上述结果，证明一个三角函数的傅里叶变换为 Sinc^2。
3. 试证明：用一个三角形滤波函数重构一个信号对应于样本的线性插值。
4. 试列举三种反走样方法的基本原理。

附 录 A　上 机 实 验 题

一、实验环境

主机：微机 386 或 486。
操作系统：DOS 3.0 以上。
语言：Microsoft C7.00。
图形卡：VGA。
图形交互输入设备：鼠标器。

二、参考书目

1. C7.00 程序员手册。
2. Mouse 编程手册。

三、本实验的目的和要求

本实验由 10 道练习题组成，可构成一个小型的图形交互输入输出系统。通过实验，可以巩固所学的图形学方面的基本知识，了解和掌握构成图形交互系统的方法和技能。要求熟悉 DOS 操作系统、C 语言编程方法，以及图形学的基本原理。实验完成后，要求能够用自己编写和构成的图形交互系统设计一个工程零件，并在有实验条件的情况下用绘图仪画出来。

四、实验题

1. 将下面的程序输入计算机，存到 cg.c 文件中，并按下述方式编译联结和运行：

c> cl cg.c /Link graphics

c> cg

程序运行后，将在显示器上显示出如图 A-1 所示的窗口画面。

光标在屏幕的左上角，移动鼠标器可改变光标的位置。所提供的程序已实现了三个命令菜单：

Line：画直线段，

Clear：清绘图区，

Exit：退出图形交互系统。

程序还提供了两个状态菜单：

Color：设置作图的颜色，

Grid：在作图区打上网格辅助作图。

将光标移到 Line 菜单项，然后按一下鼠标器上的左键，就点中了 line 菜单项，系统将在参数区提示：

图 A-1　图形交互系统接口

"Select the start point of the line."

再将光标移到图形区,按一下鼠标器的左键,就定下了直线段的起点。这时候,系统又在参数区提示:

"Select the end point of the line."

将光标在图形区移动到合适的位置,按下左键就定下了终点。系统迅速地从起点到终点画一条直线。请重复上述操作,在图形区画多条直线。

将光标移到 clear 菜单项,按一下左键,点中 clear 菜单项,清除图形区的所有线段。

选 exit 菜单项,从图形系统退到 DOS。

将光标移动到 Color 状态菜单项,按一下左键,该菜单的颜色就会改变成另一种颜色。再用 Line 画线时,线的颜色就是 Color 状态菜单项的颜色。一共有 16 种颜色可供选择,连续按左键,可以选到合适的颜色。

将光标移到 Grid 状态菜单项,按一下左键,将在绘图区显示辅助网格,再按一下左键,将清除绘图区中的辅助网格。Grid 菜单,是一个开关式状态菜单。

2. 增加状态菜单项 Line- style

Line- style 设置直线段的线型。当选中 Line- style 时,应在参数区提示用户输入线型号。线型号存放在全程变量 line- num 中。

为了使在以后的画线过程中,能以 line- num 中的线型画直线,应修改 line( ) 画线例程,在其一开始增加- setlinestyle( )。

3. 增加命令菜单项 bresenham。用 bresenham 方法作直线。功能与命令菜单项 line 完全相同,只可调用图形库的- setpixel 函数。

4. 增加状态菜单项 text- font

text- font 设置字符的字体。字体号由用户在参数区键入,存放在 font- num 全程变量中,调用

- setfont( )

将这以后要显示的字符用这种字体显示出来。

增加命令菜单项 text。text 从参数区读入字符串,由鼠标器在绘图区确定字符串的起点和终点,然后用 font- num 中指定的字体显示字符串。由于画直线和显示字符都用 setcolor 设置颜色,所以在调用显示字符串的图形库函数 outgtext( ) 之前,要调用 setcolor( ) 设置字符串的颜色。

本题可调用如下图形库函数:

- setfont( ),
- setcolor( ),
- moveto( ),
- outgtext( )。

5. 增加命令菜单项 arc, ellipse, parabola, hyperbola 和 Bezier。

arc 画一条圆弧。圆弧的起点、终点和圆弧上的一点由鼠标器在绘图区指定。用逐点比较法插补圆弧。

ellipse 画一个整椭圆。椭圆的中心由鼠标器在绘图区指定。长轴、矩轴和长轴的倾角由键盘从参数区键入。用角度 DDA 法作椭园。

parabola 画一抛物线。抛物线的起点、终点和其上一点都由鼠标器确定。用参数拟合法作抛物线。

hyperbola 画一双曲线。双曲线的起点、终点及其上一点都由鼠标器指定。

Bezier 画一三次 Bezier 曲线。曲线的四个控制点由鼠标器指定。

6. 增加状态菜单项 fillmask 和命令菜单 fill。

状态菜单 fillmask 将调用 setfillmaks 函数设置填充的图案。填充图案是一个 8× 8 的点阵,可用 8 个字节表示。

fill 进行区域填充。起点用鼠标器指定。

7. 增加命令菜单 delete。

delete 从绘图区中删除一个图形元素。图形元素可以是直线、字符串、圆弧、椭圆、抛物线、双曲线。要删除的图象用鼠标器拾取。

为了完成 delete 的功能,以及第 8 题的变换功能,有必要建立一种数据结构,将在绘图区绘出的各种图素保存和记录下来。当用鼠标器拾取某个图素时,能从这个数据结构中迅速地找到这个图素。

可能的数据结构如下:

```
struct{
  int   primitive- type;
  int   color;
  int   line- style;
```

```
    int    point- number;
    int    x- left- down, y- left- down;
    int    x- right- up, y- right- up;
    float   * px;
    float   * py;
  }gsegment[ ];
```

其中 x- left- dwon、y- left- down 和 x- right- up、y- right- up 定义了一个图素的最小外接框。在拾取时,如果光标定在这个框内,则这个图素有可能就是被拾取的图素。有时会出现光标在多个这样的框内的情况,这时就通过交互的问答方式确定所选择的对象。为了便于拾取,在图素为水平直线(这时 y- left- down= y- right- up)时,应保证 y- left- down 和 y- right- up 有一定的差值;同样,当图素为一竖直的直线时,应保证 x- left- down 和 x- right- up 有一定的差值。

当图素为 line、bresenham、circle、ellipse、parabola、hyperbola 和 Bezier 时,px 指向记录这种图素 x 坐标的数组,py 指向记录 y 坐标的数组。坐标数组的长度由 point- number 给出。

当图素为 text 时,px 指向记录 text- font 的字符串,py 指向要显示的字符串。

图素的类型记录在 primitive- type 中。

请修改命令菜单项对应的各个函数,使其具有保存图素的功能。

8. 增加命令菜单项 move、rotate、scale、copy。

move 用鼠标器拾取一个图素,将这个图素移到光标指定的位置。

rotate 用鼠标器拾取一个图素,然后再用鼠标器指定旋转中心,从键盘输入旋转角度,最后将这个图素按上述参数旋转。

scale 用鼠标器拾取一个图素,并指定缩放中心,然后从键盘读入缩放倍数,最后将这个图素按上述参数缩放。

copy 用鼠标器拾取一个图素,将这个图素拷贝到光标指定的位置。

9. 增加命令菜单项 save 和 load。

save 将数据结构中保存的内容转储到磁盘。文件名从参数区键入。

load 将磁盘上的文件读入数据结构,并在绘图区中画出来。文件名从参数区键入。

10. 增加 plot 命令菜单项。

plot 将绘图区中的内容在绘图仪上绘制出来。

附 录 B　参 考 文 献

1. James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes. Computer Graphics, Principles and Practice. Addison-wesley publishing Company, 1990
2. Mortenson M. E., Geometric Modeling. John Wiley & Sons, 1985
3. George Wolberg. Digital Image Warping. IEEE Computer Society Press Monograph, 1990
4. 唐荣锡,汪嘉业,彭群生等. 计算机图形学教程. 科学出版社, 1990 年
5. Rogers D. F., Procedural Elements for Computer Graphics. McGraw-Hill Book Company, 1985
6. R. F. Sproull, W. R. Sutherland, M. K. Ullner. Device-Independent Graphics with examples from IBM Personal Computers. McGraw-Hill Book Company, 1985
7. 唐泽圣,周嘉玉. 计算机辅助设计技术基础. 中国科学技术出版社, 1988 年
8. 唐泽圣、周嘉玉等译. 交互式计算机图形学基础. 清华大学出版社, 1986 年
9. 苏步青、刘鼎元著. 计算几何. 上海科学技术出版社, 1980 年
10. 梁友栋,石教英,彭群生译. 计算机图形学的算法基础. 科学出版社, 1987 年
11. 易晓东等译. 对话式计算机图形显示原理. 科学出版社, 1984 年
12. 张素琴,蒋维杜等译. Microsoft C/C+ + 7.0 运行库参考手册. 清华大学出版社, 1993 年
13. Microsoft Windows 3.1 程序员参考大全系列丛书. 清华大学出版社, 1993 年
14. 李新友,唐泽圣,孙家广. 正则几何形体及正则集合运算. 计算机学报, 1989 年 3 月, Vol.3
15. 孙家广,陈玉健“几何造型中的数据结构”计算机学报, 1989 年 3 月, Vol.3
16. John Staudhammer, Qunsheng Peng (Eds.), Proceedings of CAD/Graphics '91. Aug. 1991, Hangzhou, China
17. Zesheng Tang (Ed.), Proceedings of CAD/Graphics '93. Aug. Beijing, China: 1993
18. 孙家广,陈玉健,黄汉文. 计算机辅助设计技术基础. 清华大学出版社, 1990 年

・309・　・310・　・311・　・312・