西 北 工 业 大 学

硕 士 学 位 论 文

（学位研究生）

题目：基于 VxWorks 的
3D 图形组件的设计

作　　者：叶章文

学科专业：控制科学与工程

指导教师：慕德俊

2007 年　3　月

摘 要

嵌入式操作系统的引入改变了嵌入式系统落后的开发方式，加快了嵌入式系统的开发速度，提高了代码的可重用性与可扩展性。随着多媒体信息技术、互连网、消费类电子产品的发展，嵌入式操作系统由于其占用内存少、可裁减、稳定性好的特点正得到越来越广泛的应用。随着嵌入式设备如移动电话开始使用具有音频和视频内容的大量多媒体应用，对高图形质量的多媒体应用的需求很大，这就需要更高质量的3维绘制功能。在未来几年中，这将成为取得竞争优势的重要技术。要迎接这个挑战，就需要研究如何将3D图形学算法应用到嵌入式系统中。

本文分析了3D图形库的结构层次，介绍了3D图形库函数的功能及实现各个函数所需了解的图形学原理及算法,然后参照OpenGL的处理流程,在VxWorks操作系统上现有二维图形开发组件WindML的基础上设计实现3D效果所须的组件，包括绘制基本几何图元（顶点、直线、多边形）的函数，裁剪函数、矩阵转换函数、颜色、光照和纹理函数等。在设计过程中，分析测试各种3D算法，针对嵌入式系统对存储空间和运行空间的严格要求及其高可靠性、可移植性和可配置性等特点，采用了一些简单、高效的3D算法，以减小对存储空间和运行空间的依赖。所设计的3D算法具有良好的可移植性，代码稳定可靠，接口易用，所有函数均采用C语言编写,。最后在VxWorks操作系统上测试了所设计的3D组件。

关键字： 三维图形，VxWorks，嵌入式，WindML

Abstract

With the import of the Embedded system, the behindhand development method of Embedded system is improved, the development speed is accelerated and expansibility is enhanced. Along with the development of the multi-media information technique, network and consumed electronics product, due to the low memory require, cutter ability and stability, Embedded system operation system is extensively used. Embedded equipment such as mobile telephone start to use the large quantity multi-media of the audio contents and video contents. Multi-media applications require the high image quality, which require high quality 3D draw function. Te meet this challenge, we should study how to make the application of 3D graphics algorithms with Embedded system.

This thesis analyzed the structure level of the 3D graphics library, introduced the 3D graphics library function's function and the theory and algorithms needed to implement each function. Then refer to the transact flow of OpenGL, realized the 3D effect required module based on the VxWorks operation system's in existence 2D graphics development module WindML, which include all the functions to draw the basic geometry drawing(dot, line, polygon), cut out functions, matrix transform functions, color, illumination, texture, bitmap functions and so on. On the design process, we analyzed how to test the 3D algorithms. Be dead against Embedded system's requirement with storage and run memory, high reliability, portable and configure ability, we adopt some simple and high efficiency 3D algorithms to reduce storage and run memory. The 3D algorithms designed use C to develop, have a good portable, code is stable and reliable, and the it's interface is easy to use, Finally the thesis use the designed module to compile and run 3D test program on the VxWorks operation system, and test the 3D effect of the module.

Keyword: 3D graphics, VxWorks, Embedded system, WindML

西北工业大学
学位论文知识产权声明书

本人完全了解学校有关保护知识产权的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属于西北工业大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版。本人允许论文被查阅和借阅。学校可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。同时本人保证，毕业后结合学位论文研究课题再撰写的文章一律注明作者单位为西北工业大学。

保密论文待解密后适用本声明。

学位论文作者签名：叶章文　　指导教师签名：蔡绍俊

2007 年 3 月 13 日　　2007 年 3 月 15 日

西北工业大学
学位论文原创性声明

秉承学校严谨的学风和优良的科学道德，本人郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容和致谢的地方外，本论文不包含任何其他个人或集体已经公开发表或撰写过的研究成果，不包含本人或他人已申请学位或其它用途使用过的成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

本人学位论文与资料若有不实，愿意承担一切相关的法律责任。

学位论文作者签名：叶章文

2007 年 3 月 13 日

第1章 绪论

1.1 课题研究背景

由于迅速发展的 Internet 和廉价的微处理器的出现，嵌入式系统在我们的日常生活里将形成一个更大的应用领域。消费电子、交通运输、电信服务/网络工业都表现出对这个市场的关注，嵌入式操作系统将继续保持迅速的增长。从嵌入式操作系统未来的技术演变趋势来看，嵌入式操作系统在通讯，汽车，医疗，安全方面有比较有广泛的应用。同时在消费类的电子产品中，嵌入式操作系统也显示了较强的增长力。移动终端设备 PDA，手机等移动终端设备的快速增长，大大促进了嵌入式操作系统的发展。信息产业部等有关部门对我国嵌入技术的开发和应用给予了一贯的支持，无论从芯片研发、嵌入式操作系统、嵌入式数据库到应用系统研发乃至推广应用，在政策导向、标准制定、电子生产发展基金立项，倍增计划款项目和贴息等，对嵌入技术及其应用，都给予了力所能及的支持。

嵌入式软件和硬件都有良好的发展前景，特别是在图形显示相关的游戏等领域。虽然与现在的游戏平台或者个人电脑相比，嵌入式设备可能不是一个出色的游戏设备，也不具备高质量的 3D 画面效果，但是它具备了几乎所有其他设备都没有的优势：便携性。因为由于生活和工作的需要，比如大部分手机用户都会把手机随时带在身边。这就使手机用户能够随时随地使用手机设备进行游戏。只能在嵌入式设备上显示二维图形已经越来越不能满足人们对于图形显示真实感的要求了。以手机为例，随着移动技术的快速发展，一些应用开发平台如 J2ME，BREW，SYMBIAN 的成熟，手机游戏变得越来越流行。手机游戏逐渐从简单的小游戏，向大规模高冲击的视频游戏发展，嵌入式平台上也推出了嵌入式 OpenGL 标准 OpenGL ES，同时很多高档手机上出现了 ATI 加速卡，可以预见在不久的将来，3D 游戏在手机上将非常流行，而要开发 3D 游戏必须先构建 3D 图形系统，而要在速度慢，内存小，显示分辨率低下的嵌入式设备上实时显示三维图形，我们需要对已有的真实感图形学算法开展工程化，实用化的研究，并将研究成果应用于嵌入式设备中，使其可以实现高度真实感的模型，渲染强烈的感染效果，进行逼真的模拟，本课题就是在这个背景下开始研究的。

1.2 3D 计算机图形学

利用计算机绘制真实感图形在很多领域都很重要。在计算机动画片制作、城市规划、汽车制造业、建筑业、分子结构的研究、影视广告、医学、气象学、地质学、考古学等领域都有广泛应用。计算机图形的生成已由过去的近似模拟发展到今天越来越逼真的地步，有的已达到以假乱真的水平，并被广泛应用到军事、航空、航天、医学、地质勘探、文化娱乐和艺术造型等各个领域。随着计算机软、硬件突飞猛进的发展，计算机图形学在各个行业的应用也得到迅速普及和深入。真实感图形绘制是计算机图形学的一个重要组成部分，它综合利用数学、物理学、计算机科学和其它科学知识在计算机图形设备上生成象彩色照片那样的具有真实感的图形。一般说来，用计算机在图形设备上生成真实感图形必须完成以下四个步骤：一是用建模，即用一定的数学方法建立所需三维场景的几何描述，场景的几何描述直接影响图形的复杂性和图形绘制的计算耗费；二是将三维几何模型经过一定变换转为二维平面透视投影图；三是确定场景中所有可见面，运用隐藏面消隐算法将视域外或被遮挡住的不可见面消去；四是计算场景中可见面的颜色，即根据基于光学物理的光照模型计算可见面投射到观察者眼中的光亮度大小和颜色分量，并将它转换成适合图形设备的颜色值，从而确定投影画面上每一象素的颜色，最终生成图形。科学可视化、计算机动画和虚拟现实已经成为近年来计算机图形学的三大热门话题，而这三大热门话题的技术核心均为三维图形。当前，三维图形已在军事、航空、航天、医学、地质勘探、文化娱乐和艺术造型等方面有着十分广泛的应用。同时随着三维图形加速硬件性能的不断提高，以及图形硬件加速器的广泛使用，基于嵌入式设备的三维图形应用系统发展迅速。

1.3 嵌入式系统简介

1.3.1 发展与历史

嵌入式系统的定义

按照历史性、本质性、普遍性要求，嵌入式系统应定义为：“嵌入到对象体系中的专用计算机系统”。“嵌入性”、“专用性”与“计算机系统”是嵌入式系统的三个基本要素。对象体系则是指嵌入式系统所嵌入的宿主系统。

嵌入式系统的特点

嵌入式系统的特点与定义不同，它是由定义中的三个基本要素衍生出来的。不同的嵌入式系统其特点会有所差异。与“嵌入性”相关的特点：由于是嵌入到对象系统中，必须满足对象系统的环境要求，如物理环境（小型）、电气环境（可靠）、成本（价廉）等要求。与“专用性”相关的特点：软、硬件的裁剪性；满足对象要求的最小软、硬件配置等。与“计算机系统”相关的特点：嵌入式系统必须是能满足对象系统控制要求的计算机系统。与上两个特点相呼应，这样的计算机必须配置有与对象系统相适应的接口电路。嵌入式系统与对象系统密切相关，其主要技术发展方向是满足嵌入式应用要求，不断扩展对象系统要求的外围电路（如 ADC、DAC、PWM、日历时钟、电源监测、程序运行监测电路等），形成满足对象系统要求的应用系统。因此，嵌入式系统作为一个专用计算机系统，要不断向计算机应用系统发展。因此，可以把定义中的专用计算机系统引伸成满足对象系统要求的计算机应用系统。

目前国内外主要的几个嵌入式操作系统有：美国风河公司的 VxWorks，微软的 Windows CE，3Com 公司的 Palm OS 和嵌入式 Linux。这些嵌入式操作系统各有各的特点：

1) VxWorks

VxWorks 是美国 Wind River System 公司（WRS）推出的一个实时操作系统。VxWorks 是专门为嵌入式而定制的，实时性非常好，其系统本身的开销很小，进程调度、进程间通信、中断处理等系统公用程序精练而有效。VxWorks 的内核及一些系统模块可以根据需要进行定制，内核最小仅 8kB，且不失其实时、多任务的系统特征。随着近年来 VxWorks 操作系统开发环境的完善，提供了更加友善的开发界面和更加强大的模拟环境，并且改善了图形产品开发中存在的不足，使 VxWorks 成为嵌入式系统中比较成熟和完善的产品。

2) Windows CE

Microsoft Windows CE 是从 Windows 95 发展而来，提供给开发人员一个熟悉的开发环境，但是在内核结构的设计中并未考虑适应系统的高度可裁减性的要求，需要较大存储空间，应用程序也比较庞大，且在实时性方面较 VxWorks 略逊一筹。

3) 嵌入式 Linux

嵌入式 Linux 具有开放的源代码的优点，但是它的开放代码有很多都没有经

过一个严格的测试，直接使用开放代码的 BSP（board surpport packet）可以会带来不稳定的问题。它和 Windows CE 一样都是从桌面操作系统演变而成，不像 VxWorks 是专门为嵌入式而定制的，程序执行效率也没有 VxWorks 的高。

4）Palm OS

Palm OS 是由 3Com 公司开发的一种嵌入式操作系统，该系统基本上只应用于 PDA。

1.3.2 VxWorks 操作系统简介

实时多任务操作系统是能在确定的时间内执行其功能，并对外部的异步事件作出响应的计算机系统。多任务环境允许一个实时应用作为一系列独立任务来运行，各任务有各自的线程和系统资源。VxWorks 系统提供多处理器间和任务间高效的信号灯、消息队列、管道、网络透明的套接字。实时系统的另一关键特性是硬件中断处理。为了获得最快速可靠的中断响应，VxWorks 系统的中断服务程序 ISR 有自己的上下文。

VxWorks 实时操作系统由 400 多个相对独立的、短小精炼的目标模块组成，用户可根据需要选择适当模块来裁剪和配置系统，这有效地保证了系统的安全性和可靠性。系统的链接器可按应用的需要自动链接一些目标模块。这样，通过目标模块之间的按需组合，可得到许多满足功能需求的应用。

VxWorks 操作系统的基本构成模块包括以下部分：

1）、高效的实时内核 Wind

VxWorks 实时内核（Wind）主要包括基于优先级的任务调度、任务同步和通信、中断处理、定时器和内存管理。

2）、兼容实时系统标准 POSIX

VxWorks 提供接口来支持实时系统标准 P.1003.1b.

3）、I/O 系统

VxWorks 提供快速灵活的与 ANSI-C 相兼容的 I/O 系统，包括 UNIX 的缓冲 I/O 和实时系统标准 POSIX 的异步 I/O。

4）本机文件系统

5）网络特性

VxWorks 网络能与许多运行其它协议的网络进行通信，如 TCP/IP、4.3BSD、

NFS、UDP、SNMP、FTP 等。VxWorks 可通过网络允许任务存取文件到其它系统中，并对任务进行远程调用。

6)虚拟内存（可选单元 VxVMI)

VxVMI 主要用于对指定内存区的保护，如内存块只读等，加强了系统的健壮性。

7)共享内存（可选单元 VxMP)

VxMP 主要用于多处理器上运行的任务之间的共享信号量、消息队列、内存块的管理。

8)驻留目标工具

Tornado 集成环境中，开发工具工作于主机侧。驻留目标外壳、模块加载和卸载、符号表都可进行配置。

9)Wind 基类

VxWorks 系统提供对 C++的支持，并构造了系统基类函数。

10)工具库

VxWorks 系统向用户提供丰富的系统调用，包括中断处理、定时器、消息注册、内存分配、字符串转换、线性和环形缓冲区管理，以及标准 ANSI-C 程序库。

11)性能优化

VxWorks 系统通过运行定时器来记录任务对 CPU 的利用率，从而进行有效地调整，合理安排任务的运行，给定适宜的任务属性。

12)目标代理

目标代理可使用户远程调试应用程序。

13)板级支持包

板级支持包提供硬件的初始化、中断建立、定时器、内存映象等。

14)VxWorks 仿真器（VxSim)

可选产品 VxWorks 仿真器，能模拟 VxWorks 目标机的运行，用于应用系统的分析。

VxWorks 操作系统以其良好的持续发展能力、高性能的内核以及友好的用户开发环境，在嵌入式实时操作系统领域占据一席之地。它以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中，如卫星通讯、军事演习、弹道制导、飞机导航等。

1.4 研究内容

本文将从理论分析和工程应用的角度出发，系统地对3D图形组件在VxWorks中应用进行研究设计。VxWorks提供了一个图形开发组件WindML，提供了对运行在嵌入式系统上的多媒体应用程序的支持。WindML提供了独立于多种操作系统的基本图形、视频和音频技术，以及用来开发可定制的标准化设备驱动程序的框架。并且WindML提供了一系列工具用来处理来自输入设备的设备和过程事件。

WindML的主要功能有二维图形API，事件服务，区域和窗口管理，多媒体，资源管理。其中，二维图形API是最常用的部分，包括基本画图操作（画线、矩形、椭图、多边形、点），选择字体输出文体，位图，光标管理，批量画图操作，图形上下文，色彩管理，双缓冲。然而，WindML并不提供3DAPI，因此设计基于VxWorks的3D图形组件，使VxWorks具有3维绘制功能，从而使得采用VxWorks的设备具备高质量的3D图形效果，就很有必要。

嵌入式系统一般对功能、可靠性、成本、体积、功耗等有严格要求，嵌入式系统的存储空间相当有限，并具有实时性、高可靠性、可移植性和可配置性等特点。因此，针对嵌入式系统设计3D图形组件，在设计和实现过程中要充分考虑嵌入式系统的特点，解决如下几个关键问题：

（1）减小对存储空间和运行时间的依赖。

3D图形显示具有非常复杂的运算流程，要消耗大量的内存，如何降低3D运算在存储空间和运行时间上的消耗，保障嵌入式环境的实时性，是本文要解决的核心问题之一。

（2）良好的可裁减性及可移植性。

（3）高可靠性，接口的易用性。

本文首先结合计算机图形学相关知识，系统分析研究3D图形处理流程，从整体上了解3D算法的设计原理。然后，参照OpenGL的设计思想，并针对VxWorks系统在实时性、可裁减性、开放性、易用性等方面的要求采用合适的算法，设计实现一套3D图形库，包括绘制基本几何图元（顶点、直线、多边形）的函数，裁剪函数、矩阵转换函数、颜色、光照和纹理函数、位图函数等。

1.5 本文组织结构

本文在第1章主要介绍了嵌入式系统及3D图形学的基本概念，以及国内外的研究情况等。第2章主要对基本图形算法进行分析，对其中的关键技术做了初步的研究。第3章主要是研究消隐、光照模型和纹理映射等，这是增强3维图形真实感的关键技术。第4章简要介绍了OpenGL的相关知识，由于本文所设计3D图形组件将参照OpenGL，所以需要对OpenGL函数库进行深入的研究分析。第5章总的说来都属于图形的实现环节，这一章具体实现VxWorks下的3D图形组件，并给出了一些简单的3D效果图。第6章是对全文的总结和未来工作的展望。

第2章 基本图形学算法基础

2.1 引 言

光栅显示器是一个画点设备，可以看作是一个点阵单元发生器，并可控制每个点阵单元的亮度。它实际上是一个象素矩阵，是一些具有一种或多种颜色和灰度象素的集合。对一个具体的光栅显示器来说，象素个数是有限的，象素的颜色和灰度等级也是有限的，象素是有大小的，所以光栅图形只是近似的实际图形。如何使光栅图形最完美地逼近实际图形，便是光栅图形学要研究的内容。确定最佳逼近图形的象素集合，并用指定的颜色和灰度设置象素的过程称为图形的扫描转换或光栅化。对于一维图形，在不考虑线宽时，用一个象素宽的直线或曲线来显示图形。对于二维图形，需要确定区域对应的象素集，然后将各个象素设置成指定的颜色和灰度，即区域填充。任何图形光栅化后，显示在屏幕上的一个窗口里，超出窗口的部分将不予显示，因此我们要确定一个图形的哪些部分在窗口内，必须显示；哪些部分落在窗口之外，不予显示，这需要对图形进行裁剪。裁剪通常要优先进行，从而可以对图形不可见部分不必进行扫描转换。对于三维图形，由于显示器的屏幕是二维的，因而必须二维图形来表示三维图形，这就需要进行一些图形变换。

2.2 直线绘制算法

用计算机绘制三维图形时，首先要将三维图形投影到二维平面上，而绘制二维图形时要用到大量的直线段，因此绘制直线算法的效率将直接影响到整个 3D 程序算法的质量和效率。

数学上的直线是没有宽度、由无数个点构成的集合，而光栅显示器是一个点阵发生器，不可能完全地绘制出直线。光栅显示器也不能直接从单元阵列中的一个可编地址的象素画一条直线到另一个可编地址的象素，显然只有在画水平、垂直及正方形对角线时，象素点集在直线路径上位置才是准确的。所以，光栅显示器只能近似地显示直线。当我们对直线进行光栅化时，需要在显示器有限个象素中，确定最佳逼近该直线的一组象素。

由于在一个图形中，可能包含成千上万条直线，所以要求绘制算法应尽可能

快。一个象素宽的直线绘制的有三个常用算法：数值微分法（DDA）和 Bresenham 算法，中值算法。其中，Bresenham 算法效率高速度快，是目前公认的最好的光栅线段生成算法，是计算机图形学领域使用最广泛的直线扫描转换算法。它提供了一个只使用整数运算的经典算法。它跟据前一个已计算的坐标(x_i,y_i)进行增量运算以得到下一个坐标，不必进行取整操作。算法原理如下：假定直线斜率 $k=\frac{d_y}{d_x}$，$k\in[0,1]$，对于其它斜率的情况。根据主轴对称的方式作适当调整就能处理。设直线方程为 y=kx+b。假设列坐标象素已经确定为 x_i，其行坐标为 y_i。那么下一个象素的列坐标为 x_{i+1}，而行坐标要么为 y_i，要么递增 1 为 y_{i+1}。是否增 1 取决于判定变量 e 的值。误差 e 的初值 $e_0=-0.5$，x 坐标每增加 1，e 的值相应递增直线的斜率值 k，即 e=e+k。一旦 $e\geq 0.5$，就把它减去 0.5，这样保证 e 在 -0.5, 0.5 之间。当 $e\geq 0$ 时，直线与垂线 $x=x_{i+1}$ 交点最接近于当前象素 (x_i,y_i) 的右上方象素 (x_{i+1},y_{i+1})；而当 e<0 时，更接近于右方象素 (x_{i+1},y_i)。这样，我们在画直线时只需用简单的加法而不必使用费时的乘法，大大提高了效率，有利于硬件实现。

2.3 区域填充算法

在光栅扫描显示器和点阵输出的设备中表示一个区域，往往需要填充一定的灰度或不同的色彩。只要算法设计合理，就可以是图形画面逼真，更能形象且具有真实感的在显示设备上显示 3 维图形。

区域填充的基础是判断点在区域内的方法，首先建立一个函数 inside(D,x,y)：

$$\text{inside(D,x,y)}=\begin{cases}true & \text{当}(x,y)\in D\\ false & \text{当}(x,y)\notin D\end{cases} \qquad \text{公式(2-1)}$$

其中 D 表示所要填充的区域，函数 inside(D,x,y) 用来判断点 (x,y) 是否在区域 D 内。

2.3.1 扫描线多边形填充算法

一种常用的填充算法是按扫描线顺序，分别计算扫描线与多边形的相交区间，即求出每条扫描线和边界的交点，并使这些交点按 x 值按大小顺序，则扫描

线被分成若干个区间。由于连贯线，每段线段只能整段在区域内或在区域外，因此不用逐点进行区域内外的判断，这就是利用x连贯性的扫描线算法。不过x连贯性的扫描线算法不适用于所有的多边形形状，需要加以改进，一个比较有效的改进是把边分成左边和右边以及水平边，为此规定多边形的顶点按逆时针方向排列，多边形的内孔则按顺时针排列。对于同一条边P_iP_{i+1}：若$y_i < y_{i+1}$，则称为右边；若$y_i = y_{i+1}$，则称P_iP_{i+1}为水平边。扫描线与边的交点分别为左、右交点，用L,R标记，至于水平边两个端点分别用L,R标记。若交点是边的顶点，则重复左、右标记。若两个标记相同，则合并为一个标记，即当作一个交点看待；若来两个标记不同，则视为两个交点。这样就符合了顶点处的交点的计数法则，改进的x连贯性的算法为：

（1）按照边的顺序计算其与扫描线的交点，建立所有交点的表。

（2）将统一条扫描线的交点均成对出现，每一对交点中，一个是L型的，一个是R型的。如果是L型的交点的x坐标小于R型x坐标，则称此为正常对，如图2—1种P_1P_2、P_3P_4、P_5P_6、BP_0都为正常对。

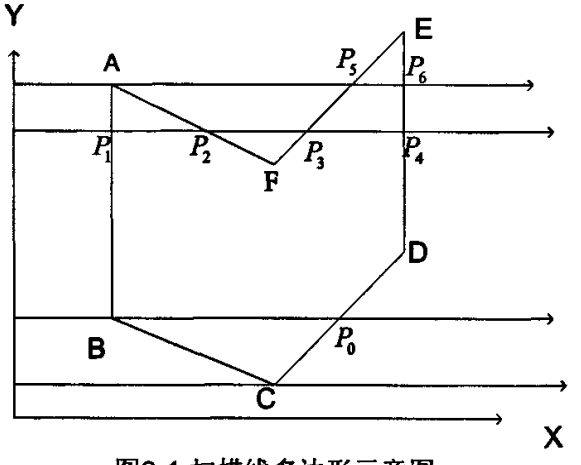

图2-1 扫描线多边形示意图

多边形填充首先要求出扫描线与边界线的交点，利用y连贯性可以简单有效的解决此问题。当一条扫描线$y = y_i$与多边形的某一条边有交点时，其相邻扫描线$y = y_{i+1}$一般也与该边相交，而且交点之间存在关联计算。设$y = y_i$扫描线与某边AB的交点为(x_i, y_i)，则$y = y_{i+1}$扫描线与AB的交点(x_{i+1}, y_{i+1})，其中

$$y_{i+1} = y_i + 1, x_{i+1} = x_i + 1/m \quad (\text{m 为该边线的斜率}) \qquad (\text{公式 2—2})$$

利用边的这种相关性，不必求出边线与各条扫描线的全部交点，只需以边线为单位，对每条扫描线建立一个记录 ET 和 AET，其内容包括：$y_{max}, x, \Delta x, next$，其中 y_{max}：边的上端点 y 坐标；x：在 ET 表中表示边的下端点的 x 坐标，在 AET 边的活化值表，则表示边与扫描线交点的 x 坐标，Δx：边斜率倒数，$next$ 指向下一条边的指针。

算法中采用较灵活的数据结构，一边有边的分类表 ET 和边的活化值表两部分组成。

边的分类 ET 是按边下端点的纵坐标 y 对非水平边进行分类的指针数组，下端点的纵坐标 y 的值等于 i 的边，归入第 i 类。有多少条扫描线，就设多少类。同一类中，各边按 x 值递增的顺序排列成行。

边的分类表 ET 可以这样建立，先按下端点的 x 坐标对所有边分类，在按递增的顺序将同一组中边排列成行。

边的活化连表 AEL 由与当前扫描线相交的所有多边形的边组成，它记录了多边形边沿扫描线交点序列，并根据递推关系式（2－2）不断地刷新交点序列。这样，当建立了边的分类表 ET 后，扫描线算法可按下列步骤进行。

步骤 1：（y 初始化）取扫描线纵坐标 y 的初始值为 ET 中非空元素的最小序号。

步骤 2：（AEL 初始化）将边的活化链表 AEL 设置为空。

步骤 3：按从上到下的顺序对纵坐标值 y 扫描线执行下列步骤，直到边的分类表 ET 和边的活化链表都变为空为止。

（1） 如边分类表 ET 中的第 y 类元素非空，则将属于该类的所有边从 ET 中取出，并插入边的活化链表 AEL 中，AEL 中的各边按照 x 值递增方向排序。

（2） 若相对于当前的扫描线，边的活化链表 AEL 非空，则将 AEL 中的边的两两依次配对，即第 1，2 边为一对，第 3，4 变为一对，依此类推。每一对边与当前扫描线的交点所构成的区段为与多边形内，依次对这些区段上的点按多边形属性着色。

（3） 将边的活化链表 AEL 中满足 $y = y_{max}$ 的边删去

（4） 将边的活化链表 AEL 剩下的每一条边 x 域累加 Δx，即 $x := x + \Delta x$。

（5） 将当前的扫描线的纵坐标值 y 累加 1，即 $y := y + 1$

上述多边形扫描线算法的数据结构和程序设计比逐点算法要复杂的多，但

由于利用x连贯性和y的连贯性，从而避免的了反复求交点的大量运算，因此，其算法效率比逐点判断法要高很多。

2.3.2 边填充算法

边填充算法也称为正负相消法。该填充算法的基本思想是：对于每一条扫描线和每条多边形的交点(x_s,y_s)，都将该扫描线上交点右方的所有像素取补，并对多边形的每条边按一定顺序做出处理。

取矩形$R(x_1 \le x \le x_2, y_1 \le y \le y_2) \supset D$区域，定义布尔数组$\mathrm{MASK}(x_1 \ldots x_2, y_1 \ldots y_2)$，并置初始值$\mathrm{MASK}(x,y) := False, (x,y) \in R$。

对于区域中每一条边与扫描线的每一个交点(x_s,y_s)，当$x \ge x_s$时，令

$$\mathrm{MASK}(x,y_s) = \overline{MASK(x,y_s)} \qquad \text{(公式 2—3)}$$

当沿D的边界经历一周后，只要$(x,y) \in D$，则$MASK(x,y) = true$;否则$MASK(x,y) = False$。然而，对$MASK(x,y)$为$true$，各点调用函数，完成对区域D内各点进行填充。

该算法虽简单易行，但对于复杂图形，每一像素可能被访问多次，输入、输出的量比有序边表算法大得多。

为了减少边填充算法访问像素的次数，可引入栅栏，即一条与扫描线垂直的直线，通常使其过多变形顶点，将多边形分成左右两部分。栅栏填充算法的填充的基本思想：对于每条扫描线与多边性的交点，将交点与栅栏之间的扫描线上的像素取补，也就是说，若交点位于栅栏左边，则将交点之右，栅栏之左的所有像素取补；若交点位于栅栏右边，则将栅栏之右，交点之左的像素取补。

栅栏填充算法仍有一些像素被重复访问。可以用边标志算法进一步改进栅栏算法，使得算法对每一个像素访问一次，边标志算法分为两个步骤：第一步对多边形的每条边进行直线扫描转换，即对多边形边界所经过的像素打上标志；第二步填充，即对每条与多边形相交的扫描线依次从左到右顺序，逐个访问该扫描线的像素，使用一个布尔量inside来指示当前点的状态，inside的初始值为False。每当当前访问的像素为打上边标志的点，就将inside取反，对未打标志的像素，inside不变，对inside作必要的操作后，若inside为真，是把该像素量按多边形填充。

2.4 裁剪

在使用计算机处理图形信息时，计算机内部存储的图形往往比较大，而屏幕显示的只是图的一部分。因此需要确定图形中哪些部分落在显示区之内，哪些落在显示区之外，以便只显示落在显示区内的那部分图形。这个选择过程称为裁剪。最简单的裁剪方法是把各种图形扫描转换为点之后，再判断各点是否在窗内。但那样太费时，一般不可取。这是因为有些图形组成部分全部在窗口外，可以完全排除，不必进行扫描转换。所以一般采用先裁剪再扫描转换的方法。

2.4.1 多边形裁剪

在画线显示中，几乎所有形式的图形都可以分成若干个直线段，然后用直线段剪取算法分别进行剪取，但是对一个由直线组成的多边形来说，若用这种方法分别来剪取各直线，则剪取后的图形往往不是封闭的多边形。多边形作为完整的图形区域（如连续色彩的图形区域），封闭的多边形裁剪后仍需封闭，需求裁剪后得到的图形仍然是封闭的多边形，即在剪取中还须加上行之有效的窗口边界线段，使图形封闭起来，这种方式的剪取成为多边形的剪取。

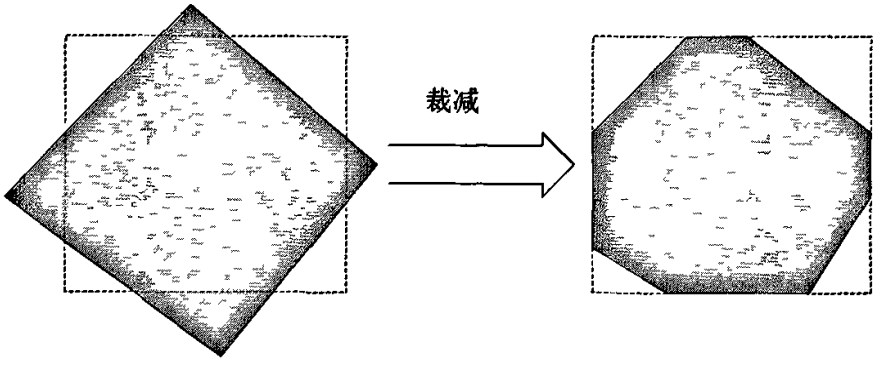

图2－2 多边形裁减

多边形的剪取比直线段的剪取要复杂，在多边形的剪取中遇到两个问题：其一是一个完整的封闭多边形，剪取后一般不再是封闭的，需要用窗边界的适当的部分来封闭它，即如何决定用来封闭图形的窗口边界部分线段；其二是经矩形窗口的四个角点是否和别的点连成线段，什么情况下角点可以不要。下面介绍逐边

裁剪法。

逐边裁剪法（见图 2—3）是 1974 年 Sutherland 和 Hodgman 提出的，其裁剪过程是每次用窗口的一条边界对要裁剪的多边形进行裁剪，然后将裁剪的多边形接着窗口的第二个边界进行剪取，依次进行，直到接四个边界裁剪完毕，最后便可得到裁剪后的多边形。

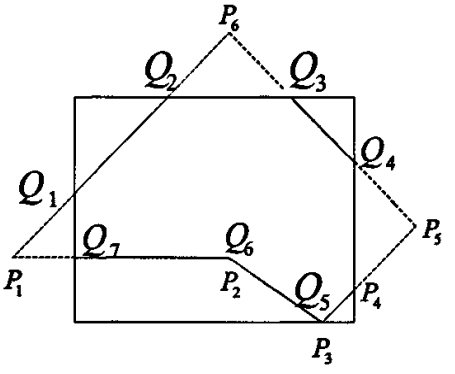

图2—3逐边裁减法

设多边形的顶点为 $P_i(i=1,2,3...,n)$，其中 n 条边界线为 $P_1P_2,P_2P_3,...P_nP_1$，多边形在剪取后应得到一个或几个新的多边形，其顶点为 $Q_1,Q_2,...Q_m$，多边形裁剪算法就是求得这些新的顶点。

（1）首先依照窗口的一个边界 e，对多边形的顶点 $P_i(i=1,2,3...,n)$ 进行检查，若顶点在边界 e 之内，保留该顶点作为剪裁后新的顶点，反之，则予以舍弃。

（2）同时检查 P_i 是否与前一顶点 P_{i-1} 处于窗口边界线 e 的同侧位置，若处于两侧，则应计算 P_iP_{i-1} 与边界线 e 的交点，并将交点保留下来，作为裁剪后的新顶点输出，而当检查最末一个顶点 P_n 时除了进行上述检查外，还须对多边形的封闭边 P_nP_1 进行检查，若封闭边 P_nP_1 与窗口边界线 e 相交，则应计算出这一交点，一并保留。

这样通过以上检查并保留出各点 $Q_1,Q_2,...Q_m$，在这些点之间依次连成折线，即为边界 e 剪裁后得到的多边形。

重复上述方法，在相对下一个边界线进行剪裁，一直到四个边界线剪裁结束。

2.4.2 三维裁剪

三维图形要在计算机屏幕上显示出来，必须经过投影变换才能在平面上显示

出来。通常，我们是对已经做过投影变换的二维图形相对于窗口或视区进行裁剪。为了避免对根本不在投影窗口内的物体边做投影变换，减少计算量，直接对三维窗口进行裁剪更有其优点。

采用投影的方法不同，三维窗口的取法也不同。对于正投影，三维窗口为投影方向的无限柱体，也可以截取其中的一段作为三维窗口，可取长方体作为三维窗口，窗口区域的点满足

$$x_{min} \le x \le x_{max}$$
$$y_{min} \le y \le y_{max}$$
$$z_{min} \le z \le z_{max}$$

对这个三维窗口进行剪裁，把可见部分投影到投影平面上去，得到平面窗口内的图形，最后进行三维窗口到视区变换，并在视区内显示和绘图。

下面将二维剪裁的编码算法推广到三维平行投影的三维裁剪算法中。

三维窗口的六个面的方程

上表面方程：$y = y_{max}$；

下表面方程：$y = y_{min}$；

前表面方程：$z = z_{max}$；

后表面方程：$z = z_{min}$；

左表面方程：$x = x_{max}$；

右表面方程：$x = x_{min}$；

利用六位二进制编码对空间任意一点$P(x,y,z)$进行编码如下，最左边的一位是第一位。

第一位为1，表示P在观察体D的上方，即$y > y_{max}$；

第二位为1，表示P在观察体D的下方，即$y < y_{max}$；

第三位为1，表示P在观察体D的右侧，即$x > x_{max}$；

第四位为1，表示P在观察体D的左侧，即$x < x_{max}$；

第五位为1，表示P在观察体D的前方，即$z > z_{max}$；

第六位为1，表示P在观察体D的后方，即$z < z_{max}$。

如不满足上述条件，则相应位置是“0”。

设空间两点P_1, P_2。

(1)先对二端点P_1,P_2按上述规则赋予编码值c_1,c_2。

(2)如果$c_1=c_2=0$，则二端点P_1,P_2均处于观察体内，则P_1P_2可见，保留；如果$c_1 \Lambda c_2 \neq 0$，二端点代码逻辑乘不为零，表示二端点有一相同位“1”，则二端点在裁剪体边界面同侧位置，则P_1P_2整个不可见，应予舍弃。

（3）若c_1,c_2不属于(2)中的两种情况，则须计算出直线与边界面的所有交点，并将线段分段后继续(2)中的两步骤，逐段舍弃位于裁剪体之外的线段，直到余下的线段符合(2)中2种简单情况为止。

2.5 坐标变换

坐标变换是本章的重点内容，它包括计算机图形学中最基本的三维变换，即几何变换、投影变换、裁剪变换、视口变换，坐标变换是真正走进三维世界的基础。

2.5.1 坐标系

1 世界坐标系

我们需要使用两个模型来描绘所要显示的对象，一个是物体模型，另一个是显示设备上的物体的图像。物体模型是存储在计算机中的物体的模型，它是建立在物体所存在的实际空间的。世界坐标系也叫实际坐标系，它是用户处理图形所采用的坐标系。

2 设备坐标系

设备坐标系也被称为屏幕坐标系，它是与图形设备相关的坐标系。

在真实世界里，所有的物体都是三维的。但是，这些三维物体在计算机世界中却必须以二维平面物体的形式表现出来。

实际上，从三维空间到二维平面，就如同用相机拍照一样，通常都要经历以下几个步骤：

第一步：指定视点，将相机对准三维景物（视点变换）。

第二步：将三维物体放在适当的位置（几何变换）。

第三步：选择相机镜头并调焦，使三维物体投影在二维胶片上（投影变换）。

第四步：决定二维像片的大小（视口变换）。

这样，一个三维空间里的物体就可以用相应的二维平面物体表示了，也就能在二维的电脑屏幕上正确显示了。

视点变换是在视点坐标系中进行的。视点坐标系于一般的物体所在的世界坐标系不同，它遵循左手法则，即左手大拇指指向 Z 正轴，与之垂直的四个手指指向 X 正轴，四指弯曲 90 度的方向是 Y 正轴。而世界坐标系遵循右手法则的。

2.5.2 几何变换

在三维坐标系里需要附加的行和列来完成计算工作，这意味着我们在 3D 中有 4 个水平参数和 4 个垂直参数，一共 16 个，所以我们使用二维矩阵 4 乘 4 矩阵。

三维空间中的点 $p(x,y,z)$ 可用齐次坐标 $p(x,y,z,\omega)$ 表示，通过将矢量 (x,y,z,ω) 乘以变换矩阵 T 而得到新的矢量 (x',y',z',ω')

$$\begin{pmatrix} x' \\ y' \\ z' \\ \omega' \end{pmatrix} = T \begin{pmatrix} x \\ y \\ z \\ \omega \end{pmatrix} \qquad \text{公式(2-4)}$$

新矢量将指向变换后的点 $p(x',y',z')$。下面是基本三维变换矩阵：

把点 $p(x,y,z)$ 沿 x、y、z 三方向分别平移 t_x 、 t_y 、 t_z 的变换矩阵为

$$\begin{vmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

把点 $p(x,y,z)$ 沿 x、y、z 三方向分别缩放 s_x 、 s_y 、 s_z 的变换矩阵为

$$\begin{vmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

把点 $p(x,y,z)$ 绕 X 轴旋转角 θ 的变换矩阵为

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

绕 Y 轴旋转角 θ 的变换矩阵为

$$\begin{vmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

绕 Z 轴旋转角 θ 的变换矩阵为

$$\begin{vmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ -\sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

另外，我们可以通过四元数方便的得到点 $p(x,y,z)$ 绕任意轴旋转的变换矩阵（见附录）。

2.5.3 投影变换

投影变换是一种很关键的图形变换，目的就是定义一个视景体，使得视景体外多余的部分裁剪掉，最终图像只是视景体内的有关部分。可以分为两种基本类型，一种是正射投影（图 2－4），另一种是透视投影（图 2－5）。它们的区别在于投影中心与投影平面之间的关系不同。如果投影平面之间的距离是有限的，那么投影是透视投影；随着投影中心越来越远，穿过任何特定物体的投影线越来越接近平行。在定义透视投影的时候，我们给定它的投影中心；对于正射投影，我们给出它的投影方向。正射投影通常用在建筑蓝图绘制和计算机辅助设计等方面，这些行业要求投影后的物体尺寸及相互间的角度不变，以便施工或制造时物体比例大小正确。透视投影符合人们心理习惯，即离视点近的物体大，离视点远的物体小，远到极点即为消失，成为灭点。它的视景体类似于一个顶部和底部都被切除掉的棱椎，也就是棱台。这个投影通常用于动画、视觉仿真以及其它许多具有真实性反映的方面。

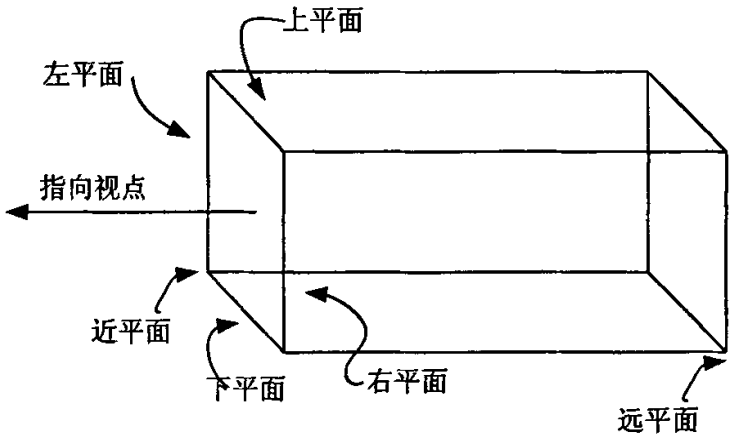

图2—4 正射投影视景图

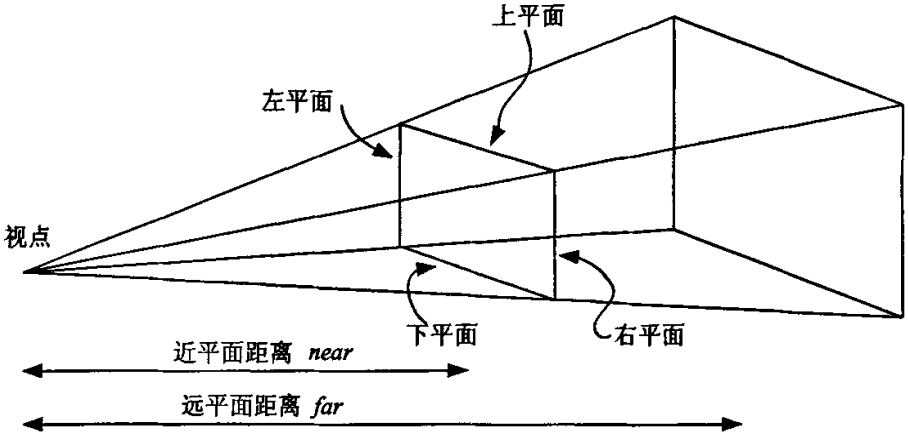

图2—5 透视投影视景图

类似于几何变换的思想，我们可以得到投影变换的变换矩阵。

假定视点坐标为(0,0,0)，用n,f分别表示近、远平面距离，用t,b,r,l上、下、左、右修剪平面坐标。则透视投影变换的变换矩阵为

$$\begin{vmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-L} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

正射投影变换的变换矩阵为

$$\begin{vmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{vmatrix}$$

2.5.4 视口变换

在计算机图形学中，它的定义是将经过几何变换、投影变换和裁剪变换后的物体显示于屏幕窗口内指定的区域内，这个区域通常为矩形，称为视口。视口变换就是将视景体内投影的物体显示在二维的视口平面上，最终确定投影后的结果。在实际应用中，视口的长宽比率总是等于视景体裁剪面的长宽比率。如果两个比率不相等，那么投影后的图像显示于视口内时会发生变形。

2.6 小结

计算机对数字化的显示物体作了加工处理后，要在图形显示器上显示，这就要在图形显示器屏幕上定义一个二维直角坐标系，这个坐标系称为屏幕坐标系。这个坐标系坐标轴的方向通常取成平行于屏幕的边缘，坐标原点取在左下角，长度单位常取成一个象素的长度，大小可以是整型数。

为了使显示的3维图形能以合适的位置、大小和方向显示出来，必须要通过投影。投影的方法有两种，即正射投影和透视投影。有时为了突出图形的一部分，只把图形的某一部分显示出来，这时可以定义一个三维视景体（Viewing Volume）。正射投影时一般是一个长方体的视景体，透视投影时一般是一个棱台似的视景体。只有视景体内的物体能被投影在显示平面上，其他部分则不能。在屏幕窗口内可以定义一个矩形，称为视口（Viewport），视景体投影后的图形就在视口内显示。为了适应物理设备坐标和视口所在坐标的差别，还要作一适应物理坐标的变换。这个坐标系称为物理设备坐标系。

三维物体经过几何、投影、视口变换后就得到三维物体的二维表示，这样就可以用二维图形来近似的显示三维图形了。

综合上面所论述的内容，我们已经可以得出一个最基本的3D处理流程，如

图2—6所示

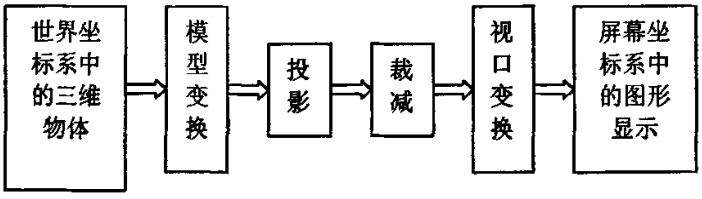

图2—6 三维图形显示流程

第3章 真实感图形算法研究

用计算机生成三维物体的真实图形，是计算机图形学研究的重要内容。真实图形在仿真模拟、几何造型、广告影视、指挥控制和科学计算的可视化等许多领域都有广泛应用。在上一章本文讨论了图形学的最基础的算法，仅仅采用这些算法并不能很好的描述三维物体，我们可以通过消隐、光照模型、纹理映射等手段来加强3维图形显示的真实感。

3.1 消隐

空间的一个三维物体输出到二维显示设备时，为了使计算机形成的二维图形能真实的反映这一情况，就必须在绘制时消除被遮挡的不可见的线或面，习惯上称作消除隐藏线和隐藏面，或简称为消隐，否则对同一图形可能会产生二义性，如图3—1表示一个长方体的输出过程。

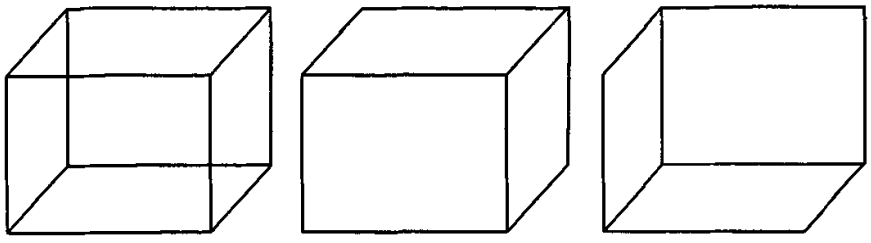

图3—1 未消隐的图产生二义性

3.1.1 消隐分类

1. 按消隐对象分类

(a) 线消隐

消除物体上不可见的边。

(b) 面消隐

消除物体上不可见的面。

2. 根据消隐算法实现所在的坐标系或空间的不同，将消隐算法分为两类：

(a) 物体对象空间的消隐算法：该算法是在所研究的物体对象所定义的世界

坐标系的三维空间进行的，着眼于物体间的几何关系来确定线和面的可见性，采用高精度计算，把图形放大而不影响其精确性。其缺点是随着物体的增多，计算量迅速增大。此算法适用于精密的工程应用领域。

(b)图像空间的消隐算法是在显示图形的二维屏幕坐标系中实现，着眼于视区中每一像素的可见性，计算精度和计算量取决于像素的多少。这种算法比较粗糙，而且所得到的画面在放大后往往不令人满意。其优点是该算法计算量较小，算法效率比较高，这是因为在光栅扫描过程中可以充分利用画面的连贯性。在物体空间中预先计算面的可见性优先级，再在图像空间中生成消隐图。

3.1.2 消除隐藏线

在线框显示模型中，用边界线表示有界平面，用边界线及若干参数曲线表示参数曲面，所以待显示的所有实体均为线。但线不可能对线有遮挡关系，只有面或体才有可能对线形成遮挡。故消隐算法要求造型系统中有面的信息，最好有体的信息。

为运算方便，一般通过平移、旋转、透视等各种坐标变换，将视点变换到Z轴的正无穷大处，视线方向变为Z轴的负方向。变换后，坐标Z值反映了相应点到视点的距离，可以作为判断遮挡的依据。另外，对视锥以外的物体应先行过滤，以减少不必要的运算。

线消隐中，最基本的运算为：判断面对线的遮挡关系。体也要分解为面，再判断面与线的遮挡关系。在遮挡判断中，要反复地进行线线、线面之间的求交运算。平面对直线段的遮挡判断算法：

（1）若线段的两端点及视点在给定平面的同侧，线段不被给定平面遮挡，转7

（2）若线段的投影与平面投影的区间无交，线段不被给定平面遮挡，转7

（3）求直线与相应无穷平面的交点，若无交点，转 4。否则，交点在直线内部或外部。若交点在线段内部，交点将线段分成两段，与试点同侧的一段不被遮挡，另一段在视点异侧，转4再判；若交点在线段外部，转4。

（4）求所剩线段的投影于平面边界投影的所有交点，并根据交点在原直线参数方程中参数值求出深度Z值。若无交点，转5。

（5）以上所求得的各交点将线段的投影分成若干段，求出第一段中点。

（6）若第一段中点在平面的投影内，则相应的段被遮挡，否则不被遮挡，其他段的遮挡关系可依次交替取值进行判断。

（7）结束

3.1.3 消除隐藏面

在生成真实感图形时需要判别出从某以特定观察位置所能看到的场景中的内容，人们根据不同的应用场合开发出了相当数目的算法，从而有效的判别可见物体。这些算法通常称为隐藏面消除算法，也称为可见面判别算法。其中，一些算法的内存开销较大，有些算法的处理时间较长，一些算法则针对某些特定的物体类型。虽然各种可见面判别算法的基本思想各不相同，但它们大都采用了排序和连贯性方法以提高效率。将场景中的物体表面根据它们与观察平面的具体方位进行排序，可以加深深度比较，而连贯性方法则充分利用场景的规则性特征。一条扫描线可能包含相同像素强度的区段，并且相邻扫描线之间的图案变化很小。动画中的各个帧之间仅在运动物体的相邻区域那有差异，而通常可以建立起场景中的物体和场景表面之间的稳定关系。

三维场景中常见的消除隐藏面判别算法

1）深度排序算法

深度排序算法的原理是：把物体的各个面片按深度递减方向排序，排序结果存在一张深度优先级表中。然后按照从表头到表尾的顺序逐个绘制各个面片。由于后显示的图形取代先显示的画面，而后显示的图形所代表的面离视点更近，所以由远及近的绘制各面，就相当于消除隐藏面。该算法也称为画家算法，因为油画作家作画时通常先画底色，然后在一层层往上画。

2）Z缓冲算法

深度排序算法中，深度排序计算量大，而且排序后，还需再检查相邻的面，以确保在深度优先级表中前者在前，后者在后。若遇到多边形相交，或多边形循环重叠的情形，还必须分割多边形。为了避免这些复杂的运算，人们发明了Z缓冲区(Z-Buffer)算法。在这个算法里，不仅需要有帧缓存区来存放每个象素的亮度值，还需要一个Z缓冲区来存放每个象素的深度值。缓冲器中每个单元的值是对应象素点所反映对象的z坐标值。Z缓冲区中每个单元的初值取成z的极小值，帧缓冲器每个单元的初值可放对应背景颜色的值。图形消隐的过程就是给帧

缓冲器和Z缓冲器中相应单元填值的过程。在把显示对象的每个面上每一点的属性值填入帧缓冲器相应单元前，要把这点的z坐标值和z缓冲器中相应单元的值进行比较。只有前者大于后者时才改变帧缓冲区的那一单元的值，同时z缓冲器中相应单元的值也要改成这点的z坐标值。如果这点的z坐标值小于z缓冲器中的值，则说明对应象素已经显示了对象上一个点的属性，该点要比考虑的点更接近观察点。对显示对象的每个面上的每个点都做了上述处理后，便可得到消除了隐藏面的图。

3）扫描线Z缓冲算法

Z缓冲算法非常简单，对多边形表面也不用排序，有利于硬件实现。但是此算法要占用大量的内存空间，如利用相关性提高点与多边形的包含性测试和深度计算的速度，就得到扫描线Z缓冲算法，该算法可以解决Z缓冲算法内存消耗过大的问题。

3.2 简单光照模型

场景中如果没有光线，那我们所产生的虚拟世界只能是漆黑一片。在自然界，当光从光源发出后，在到达用户的眼睛之前，已经经过了成千上万物体的反射。每经过一次反射，一部分光被表面吸收，一部分被散射到各个方向，而剩余的则被反射到其它表面或用户的眼睛里。这个过程一直持续，直到光衰减为零或到达用户眼中，要让我们的场景看起来更加真实，光线是必不可少的一环。

当光照射到一个物体表面上时，会出现三种情形。首先，光可以通过物体表面向空间反射，产生反射光。其次，对于透明体，光可以穿透该物体并从另一端射出，产生透射光。最后，部分光将被物体表面吸收而转换成热。在上述三部分光中，仅仅是透射光和反射光能够进入人眼产生视觉效果。这里介绍的简单光照模型只考虑被照明物体表面的反射光影响，假定物体表面光滑不透明且由理想材料构成，环境假设为由白光照明。

一般来说，反射光可以分成三个分量，即环境反射、漫反射和镜面反射。环境反射分量假定入射光均匀地从周围环境入射至景物表面并等量地向各个方向反射出去，通常物体表面还会受到从周围环境来的反射光的照射，这些光常统称为环境光；漫反射分量表示特定光源在景物表面的反射光中那些向空间各方向均匀反射出去的光，这些光常称为漫射光；镜面反射光为朝一定方向的反射光，如

一个点光源照射一个金属球时会在球面上形成一块特别亮的区域，呈现所谓“高光”，它是光源在金属球面上产生的镜面反射光。对于较光滑物体，其镜面反射光的高光区域小而亮；相反，粗糙表面的镜面反射光呈发散状态，其高光区域大而不亮。

环境光是指光源间接对物体的影响，是在物体和环境之间多次反射，最终达到平衡时的一种光。我们近似地认为同一环境下的环境光，其光强分布是均匀的，它在任何一个方向上的分布都相同。

3.2.1 材质

材质描述在三维场景中的多边形如何反射光或如何发光。本质上，材质是一组属性集合，指定对象如何反射环境光和漫反射光、材质和镜面反射高光看起来是什么样、多边形发光时看起来是什么样，即控制光线如何照亮表面。

材质属性包括：

1)镜面反射属性：镜面反射系数 ρ_s 用于控制镜面反射总量。镜面反射指数 f 通过控制镜面反射光分布的窄带控制了表面光泽度。

2)漫反射属性：漫反射系数 ρ_d 漫反射光的相对强度。

3)环境反射属性：环境反射系数决定了 ρ_a 从表面反射的环境光强。

4)发射属性：表面的发射属性决定了在没有任何入射光的情况下从表面发出的光。从表面发出的光不作为照亮其他表面的光源；相反，它仅仅影响到观察者看到的颜色。

3.2.2 光源类型

这里介绍三种常见光源类型：

点光源——场景中的点光源具有颜色和位置，但没有确定的方向。点光源向各个方向发出的光相等。

平行光——平行光只有颜色和方向，没有位置。平行光发出平行的光，这意味着所有平行光产生的光在场景中以相同的方向传播。可以认为平行光是位于无限远处的光源。

聚光灯——聚光灯具有颜色、位置和发出光的方向。聚光灯发出的光由一个

比较亮的内圆锥和一个较大的外圆锥组成，光强由内而外逐渐减小。

3.2.3 Phong 光照模型

Phong 光照模型可以产生很好的明暗变化和照明；同时又易于软件或硬件高效实现。它几乎是实时系统中最常用的。Phong 光照模型 Phong 提出的（Phong，1975）。

Phong 光照模型的核心是一个光线如何从表面反射的模型。在 Phong 光照模型中，所有的光源都模拟为点光源。同时，光线被模拟为包含三个分离的颜色分量（红、绿、蓝）。即假定所有的光线都有单一红色分量、单一绿色分量和单一蓝色分量组成。根据叠加原理，可以对每个光源和三种颜色分量的每一个单独计算光强。

Phong 光照模型属于“局部”光照模型，仅仅考虑光源直接照射到表面然后直接反射到视点的效果，并不考虑二次反射，在这种情况下光到达视点前可能经过了几个表面的反射，这也造成该模型不能正确处理光投射生成的阴影。

Phong 反射模型中对反射基本设置如图 3—2 所示，表面的一个特定点被一个点光源照到，并从某个视点观察。表面的方向由垂直于该表面的单位法线向量 n 决定。光源方向由从表面指向光源的单位向量 l 决定，视点方向由从表面指向视点的单位向量 v 决定。这三个向量，加上光源和表面材料属性被用于 Phong 模型中，来计算到达眼睛的光强。

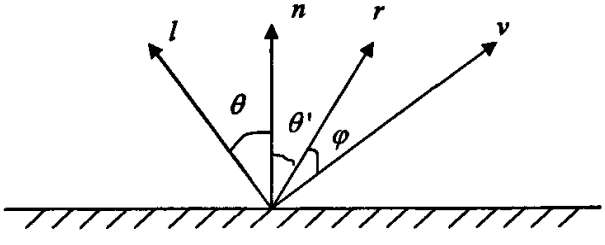

图3—2 Phong光照模型 l 为点光源的单位方向向量，n 为表面法向向量，v 为视点所在的单位方向向量

假设点光源发出的光线光强为 I^m。Phong 模型提供了计算从表面反射而到达眼睛的光强的方法。光强指垂直于光线方向单位面积通过的能量。

Phong 模型中有两种反射即漫反射和镜面反射，在图 3.1 中，θ 指点光源方向向量 l 与表面法向向量 n 的夹角，即光线从点光源到达的入射角，r 为理想镜

面反射时的反射的单位方向向量，r与n的夹角θ'为理想镜面反射时的反射角，与θ相等，φ是r与n的夹角。

在考虑单一颜色来自单一光源的光线时，由漫反射和镜面反射引起的光强计算公式如下：

漫反射的光强计算公式为 $I_d = \rho_d I_d^{in} \cos(\theta) = \rho_d I_d^{in} (l \cdot n)$ 公式（3—1）

其中I_d^{in}指漫反射光入色光强，I_d是视点方向的漫反射光强，ρ_d为常数，被称为表面漫反射系数，是材质的一个属性。按照该式进行漫反射的表面被称为Lambertian，大多数的非光泽表面都非常接近Lambertian。Lambertian的特征是：如果表面的一大块平坦区域被均匀照射，从各个观察角度看到的表面亮度和颜色是一致的。

镜面反射的光强计算公式为 $I_s = \rho_s I_s^{in} \cos(\varphi)^f = \rho_s I_s^{in} (v \cdot r)^f$ 公式（3—2）

其中ρ_s为常数，称为镜面反射系数，I_s^{in}是从光源来的镜面光光柱。

另外，在Phong模型中，我们用一个常数来模拟环境光的发射光强，用式子表示为：$I_a = \rho_a I_a^{in}$。其中：I_a^{in}为环境光的光强，ρ_a为物体对环境光的反射系数

根据叠加原理，不同类型的反射和发射可以通过简单相加结合起来，而且多个光源的结果也可以通过分开考虑每个光源的光照，然后相加来同样确定。

3.2.4 Gouraud 渲染

“渲染”是指使用插值在物体表面上创建一个亮度和颜色平滑变化的图案。没有渲染，几何模型中的每一个多边形都会被绘制为同一颜色，这样绘出的图形也明显是由多边形组成的。对于多边形，通过计算多边形顶点上的光照和颜色，然后使用插值来设置多边形内部像素的光照和颜色，可以得到很好的渲染效果。

Gouraud渲染是由Gouraud于1971年提出的，它先计算物体表面多边形各顶点的光强，然后用双线性插值，求出多边形内部区域中各点的光强。

它的基本算法描述如下：

a)计算多边形顶点的平均法向向量。

b)用Phong光照明模型计算顶点的平均光强。

c)插值计算离散边上的各点光强。

d)插值计算多边形内域中各点的光强。

下面我们分别介绍算法中的每一个步骤。

顶点法向量计算:我们用与顶点相邻的所有多边形的法向量的平均值近似作为该顶点的近似法向量。假设顶点 A 相邻的多边形有 k 个，法向分别为 $N_1,N_2,\cdots N_k$，我们取顶点 A 的法向量为 $N_a=\frac{1}{k}(N_1+N_2+\cdots N_k)$，在一般情况下，用相邻多边形的平均法向作为顶点的法向，与该多边形物体近似的曲面的切平面比较接近。顶点平均光强计算：在求出顶点 A 的法向 N_a 后，我们可以用 Phong 光照模型计算在顶点处的光亮度。

光强插值:用多边形顶点的光强进行双线性插值，可以求出多边形上各点和内部点的光强。算法首先由顶点的光强插值计算各边的光强，然后由各边的光强插值计算出多边形内部点的光强。

3.3 纹理概述

纹理是对象的“皮肤”。纹理贴图能使我们的场景更加详细和真实。近几年来，纹理的使用使得计算机三维图象具有了更好的真实感。例如，我们可以创建一些具有木头和石头图案的对象，也可以在盒子外面贴一个标签或者贴一张真实的图片。如果不采用纹理，则物体表面要么非常光滑、简单。要么就必须采用很小的多边形来绘制以便清楚地表面的细微特征。

从根本上说，纹理是物体表面的细小结构，它可以是光滑表面的花纹、图案，也可以是颜色纹理，这时的纹理一般都是二维图象纹理，当然也有三维纹理。我们将在下面的小节中分别介绍它们的映射方法；纹理还可以是粗糙的表面(如桔子表面的皱纹)，它们被称为几何纹理，是基于物体表面的微观几何形状的表面纹理，一种最常用的几何纹理就是对物体表面的法向进行微小的扰动来表现物体表面的细节。

3.3.1 纹理坐标

在计算机中，纹理实际上是一个二维数组，它的元素是一些颜色值。单个的颜色值被称为纹理元素(texture elements)或纹理像素(texel)。每一个纹理像素在纹理中都有一个唯一的地址。这个地址可以被认为是一个列(column)和行(row)的值，它们分别由 u 和 v 来表示。

纹理坐标位于纹理空间中。也就是说，它们和纹理中的(0,0)位置相对应。当我们将一个纹理应用于一个图元时，它的纹理像素地址必须要映射到对象坐标系中。然后再被平移到屏幕坐标系或像素位置上。由 u，v 坐标在纹理上所定的元素成为 texel，注意 v 坐标方向是朝下的。每个 texel 在纹理中有唯一的地址，可以认为这个地址是行和列的编号，它们分别被标记为 u 和 v。纹理坐标位于纹理空间中，也就是说，它们相对于纹理中的位置(0,0)点。当把纹理贴到三维空间中图元的表面时，纹理的 texel 必须先被映射到对象坐标系，然后再变换到屏幕坐标系，或像素的位置。这个过程称为纹理映射。

3.3.2 纹理映射

纹理映射是把我们得到的纹理映射到三维物体的表面的技术。

对于纹理映射，我们需要考虑以下三个问题：

考察简单光照明模型，我们需要了解，当物体上的什么属性被改变，就可产生纹理的效果。我们可以改变的物体属性有：漫反射系（改变物体的颜色），或者，物体表面的法向量。通过这些变化，我们就可以得到纹理的效果。在真实感图形学中，通常可以用如下的两种方法来定义纹理：

图象纹理：将二维纹理图案映射到三维物体表面，绘制物体表面上一点时，采用相应的纹理图案中相应点的颜色值。

函数纹理：用数学函数定义简单的二维纹理图案，如方格地毯。或用数学函数定义随机高度场，生成表面粗糙纹理即几何纹理。

定义了纹理以后，我们还要处理如何对纹理进行映射的问题。对于二维图象纹理，就是如何建立纹理与三维物体之间的对应关系；而对于几何纹理，就是如何扰动法向量。理论上，定义在此空间上的任何函数可以作为纹理函数，而在实际上，往往采用一些特殊的函数，来模拟生活中常见的纹理。

二维纹理域的映射：纹理映射技术中，最常见的纹理是二维纹理。映射将这种纹理变换到三维物体的表面，形成最终的图象。这个过程必须用纹理给图元在二维屏幕上对应的每个像素产生一个颜色。即对于每个像素，必须从纹理获得一个颜色值，这个过程被称为纹理过滤。在执行纹理过滤操作时，正在使用的纹理一般会被放大或缩小，换句话说，就是纹理被贴到比它大或比它小的图元上。纹理放大会使多个像素映射到一个 texel，得到的图像可能会有马赛克。纹理缩小

会使一个像素映射到多个 texel 得到的图像可能会模糊不清或有锯齿。要解决这些问题，必须在计算像素的颜色时对 texel 的颜色进行一些混合操作。

三维纹理域的映射：前面介绍的二维纹理域映射对于提高图形的真实感有很大的作用，但是，由于纹理域是二维的，图形场景物体一般是三维的，这样在纹理映射的时候是一种非线性映射，在曲率变化很大的曲面区域就会产生纹理变形，极大的降低了图象的真实感，而且对于二维纹理映射，对于一些非正规拓扑表面，纹理连续性不能保证。假如在三维物体空间中，物体中每一个点(x, y, z)均有一个纹理值 t(x, y, z)，其值由纹理函数 t(x, y, z)唯一确定，那么对于物体上的空间点，就可以映射到一个纹理空间上了，而且是三维的纹理函数，这是三维纹理提出来的基本思想。三维纹理映射的纹理空间定义在三维空间上，与物体空间是同维的，在纹理映射的时候，只需把场景中的物体变换到纹理空间的局部坐标系中去即可。

3.3.3 反走样

纹理映射过程中的一个常见问题就是走样，走样出现在当屏幕像素和纹理像素之间没有一一对应关系的时候。我们假定多边形内部的每一个象素的纹理坐标都是从多边形的顶点的纹理坐标插值而来，屏幕像素的纹理坐标经过取整而采用纹理中的最近的像素，该纹理像素的颜色被显示在屏幕像素的对应位置，即每个屏幕像素的颜色从单个纹理像素的颜色获得。

当纹理图案的分辨率小于对应的屏幕图像的分辨率时，单个纹理像素将会对应一块屏幕像素。这使的每一个纹理像素在屏幕变成一块区域。这样的结果是纹理被放大了，而每一个像素变成了一大块。

当屏幕图像的分辨率接近或小于纹理图案的分辨率时，这会导致一些视觉上很难接受的效果，如前后帧之间扰动。这是由于每一个像素只取一个纹理像素引起的，这意味着纹理图案只有一部分被选取并显示到屏幕上。这会导致帧间扰动、斑点、粒状等其他问题。

有几种方法可以用来解决纹理映射过程中出现的走样问题，其中，MIP 映射是实际中采用最广泛的方法。

MIP 映射又称作细节层次（LOD）方法，是一种预先计算低分辨率纹理的方法。该方法预先计算一组低分辨率的图案，并且始终显示分辨率和当前显示图像

分辨率最为接近的纹理图案。MIP 映射试图通过这种方法来避免当纹理图案分辨率大于屏幕图案分辨率时产生的一些问题。这样，当采用了纹理映射的物体离视点比较远的时候，可以使用低分辨率的 MIP 纹理；而离视点较近时，使用高分辨率的纹理。MIP 映射方法可以消除很多走样问题，包括绝大多数跟频闪和跳跃有关的问题。另外，使用 MIP 映射方法还可以极大的提高内存使用效率。在任何给定的场景里，通常只有相对很少的几幅纹理需要从近处观察，绝大多数纹理都是从远处观察的。观察的距离越远，需要的纹理图案的分辨率就越低，因此只有低分辨率的纹理图案需要存储在缓存中，这样可以有效的使用内存。

第4章 OpenGL研究

本文所设计的3D组件是参照OpenGL模型进行的，所有的函数原型都借鉴了OpenGL的函数原型，函数的组织结构在很大程度上也借鉴了OpenGL的结构，因此有必要分析OpenGL的函数结构，功能。

4.1 OpenGL简介

OpenGL现由业界著名的OpenGL体系结构评审委员会(ARB)控制。该委员会包括英特尔(Intel)、IBM、微软(Microsoft)、DEC、康柏(Compag)、SGI、Intergraph、Evans和Suther—land等九个成员，主要负责评审OpenGL的功能扩展和制定相关的技术规范。

作为3维绘制规范，OpenGL图形API实际上已经是个人计算机和工作站的世界范围内的标准。由于其开放性和高度可充用性，OpenGL已成为高性能图形和交互式视景处理的工业标准。比较著名的产品如动画制作软件3DMAX，MayaSoftImage，VR软件，CAM软件等都是以OpenGL为基础的。

OpenGL是一种立即模式的3DAPl，在实时3D处理中性能突出，它可以省略将图形预先储存于数据结构的步骤，直接提取图形的像素，按照编程人员的指令进行着色渲染，大大节省了3D图形的处理时间。

OpenGL不是工具包，也不提供有关描述三维对象造型的高层命令；它不是面向对象，也不执行窗口任务或获取用户输入命令。它与硬件、窗口和操作系统是相互独立的。OpenGL具有立即方式与灵活的应用格式；它是一个程序系统并具有显示列表的功能，特别是它对网络的透明性给专业设计人员带来了极大的方便。因而OpenGL被计算机工业界看作当前最先进的三维图形APl。它提供很强的绘制二维和三维图形能力，包括基本图元、造型、着色、光照、景深、阴影、混合、动画、明暗处理、隐面消除、反走样、纹理映射、图像处理等绘制功能。另外，OpenGL利用显示表（displaylists）概念引入了PHIGS中的层次结构概念。它不需要包含复杂的预定义对象，设计者只需调用OpenGL的几个简单几何图元（如点、线和多边形），即可建立所要求的模型，因而深得许多专业人员喜爱。在OpenGL的API顶部还设有实用程序库（例如，面向对象的3D图形工具包

open inventor)，支持绘制二次曲线和曲面、nurbs曲线和曲面及若干其它高级图元。OpenGL实质上是一种多平台、高性能的三维图形软件开发系统。

OpenGL是一个硬件图形发生器的软件界面。就是说，它的应用编程界面是图形硬件的软件界面。因此，它实际上是一种与硬件无关的编程界面。它由一系列OpenGL命令组成。程序员利用OpenGL命令可以创建动态的三维彩色图形的交互式程序，并能控制计算机图形技术来产生真实感图形，或利用富有想象力的方法产生虚构的艺术创作。OpenGL不仅可以处理单幅的图形，而且可用于实时的三维仿真领域，其对环境和实体的渲染达到了高度逼真的视觉效果，从而显示出强大的生命力。基于OpenGL开发的大量三维图形应用软件系统，已广泛应用于科学计算可视化、实体造型、CAD、CAM、仿真、图像处理、地理信息系统、地震、石油、数据可视化、计算机动画和影视特殊效果处理、虚拟现实、三维游戏等领域。

4.2 OpenGL的基本功能

（1）绘制物体。OpenGL提供了丰富的基本图元的绘制命令，从而可以方便地绘制物体。在OpenGL中，将点、线、多边形等称为图元。

（2）变换。OpenGL提供了一系列基本的变换，如取景变换、模型变换、投影变换和视区变换等操作。

（3）光照处理。一个可见的物体是处于一定的环境中，必然会产生光照的效果，绘制真实感的图形必须作光照处理。

（4）着色。客观世界的都有颜色，在计算机上模拟的物体也应该有颜色。OpenGL提供两种着色模式，一种是RGBA模式，另一种是颜色索引模式。

（5）反走样。图形显示在光栅显示器上，所绘制的图形在边缘处会出现锯齿，看起来不够光滑、圆整，这种情况称为走样。OpenGL可以对走样进行调整，以更真实地模拟客观世界的物体，对走样进行调整的过程称为反走样。

（6）融合。为了使三维图形更具有真实感，经常需要处理透明或半透明的物体，这就需要用到融合技术。

（7）雾化。如同自然界中存在的烟雾对物体的影响一样，OpenGL提供了雾化的功能来更真实地模拟物体。

（8）纹理映射。物体的表面表现出组成物体的材质，这种性质称作纹理。

纹理使物体的显示更趋于真实。

（9）动画。OpenGL提供了双缓冲技术实现动画功能。

4.3 OpenGL的函数名及数据类型

OpenGL 的库函数命名方式很有规律，了解这种规律后阅读和编写程序都比较容易方便。

首先，每个库函数有前缀gl、glu、glx或aux，表示此函数分属于基本库、实用库、X窗口扩充库或辅助库，其后的函数名头字母大写，后缀是参数类型的简写，取i、f等。

OpenGL的数据类型定义可以与其它语言一致，但在ANSI C下使用以下定义的数据类型。

表4-1 命令前缀和参数数据类型

前缀	数据类型	相应C语言类型	OpenGL类型
S	16-bit integer	Short	GLshort
b	8-bit integer	signed char	GLbyte
i	32-bit integer	long	GLint,GLsizei
f	32-bit floating-point	float	GLfloat,GLclampf
d	64-bit floating-point	double	GLdouble,GLclampd
ub	8-bit unsigned integer	unsigned char	GLubyte,GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned long	GLuint,GLenum,GLbitfield

4.4 OpenGL API简介

在 OpenGL 中有 115 个核心函数，这些函数是最基本的，它们可以在任何OpenGL的工作平台上应用。这些函数用于建立各种各样的形体，产生光照效果，进行反走样以及进行纹理映射，进行投影变换等等。由于这些核心函数有许多种形式并能够接受不同类型的参数，实际上这些函数可以派生出300多个函数。

OpenGL的实用函数是比OpenGL核心函数更高一层的函数，这些函数是通过调用核心函数来起作用的。这些函数提供了十分简单的用法，从而减轻了开发者的编程负担。OpenGL 的实用函数包括纹理映射、坐标变换、多边形分化、绘制一些如椭球、圆柱、茶壶等简单多边形实体等。这部分函数象核心函数一样在任

何OpenGL平台都可以应用。

本文所设计的VxWorks底下的3D组件主要参照了OpenGL中的核心函数，下面介绍一下OpenGL中与本文目前所设计的3D图形库相关的核心函数API。

(1)绘制图元API

表4-2 绘制几何图元及物体

函数名	功能
glVertex	指定顶点
glVertexPointer	定义顶点数据数组
glArrayElement	定义绘制顶点的数组元素
glBegin, glEnd	限定图元开始绘制与结束绘制
glPointSize	指定光栅化点的直径
glEdgeFlag, glEdgeFlagv	指定边界标记
glLineWidth	指定线宽
glLineStipple	指定点画线
glPolygonMode	定义多边形光栅化模式
glFrontFace	定义正面多边形和反面多边形
glDrawElements	从数组数据绘制图元
glPolygonStipple	设置多边形点
glRect	绘制矩形

(2)坐标转换API

表4-3 坐标转换

函数名	功能
glTranslate	用平移矩阵乘以当前矩阵
glScale	用缩放矩阵乘以当前矩阵
glRotate	用旋转矩阵乘以当前矩阵
glFrustum	用透视矩阵乘以当前矩阵
glViewPort	设置视口
glOrtho	用正视矩阵乘以当前矩阵
glClipPlane	指定切割几何体的平面

(3)矩阵堆栈操作API

表4-4矩阵堆栈操作

函数名	功能
glLoadNames	载入堆栈
glLoadMatrix	用任意矩阵替换当前矩阵
glMatrixMode	指定当前矩阵是哪种矩阵
glMultMatrix	用任意矩阵相乘以当前矩阵
glPushMatrix,	压入当前矩阵堆栈
glPopMatrix	弹出当前矩阵堆栈
gllnitNames	初始化堆栈

(4)颜色管理API

表4-5颜色管理

函数名	功能
glColor	设置当前颜色
glColorPointer	定义颜色数组
glIndex	设置当前颜色索引
glShadeModel	选择平面明暗模式或光滑明暗模式
glIndexPointer	定义颜色索引数组
glColorTable	指定调色板

(5)光照和材质API

表4-6光照和材质

函数名	功能
glNormal	设置当前的法向量
glNormalPointer	定义法向量数组
glLight	设置光源
glLightModel	设置光照模型参数
glColorMaterial	使材质颜色跟踪当前颜色
glMaterial	为光照模型指定材质参数

(6)纹理映射API

表4-7 纹理映射

函数名	功能
glTexCoord	设置当前纹理坐标
glTexImage2D	指定二维纹理映射
glTexParameter	设置纹理参数
glTexEnv	设置纹理环境参数
glTexGen	定义纹理坐标数组
glDeleteTextures	删除命名的纹理
glBlendFunc	指定像素融合

(7)帧缓存操作API

表4-8 帧缓存操作

函数名	功能
glClear	将缓存清除为指定的值
glClearAccum	清除累加缓存
glClearColor	清除颜色缓存
glClearDepth	清除深度缓存
glClearIndex	清除颜色索引缓存的值
glClearStencil	清除指定模板
glDrawBuffer	指定绘制的颜色缓存
glIndexMask	控制颜色缓存中单个索引位的写操作
glColorMask	激活或关闭颜色缓存的写操作
glDepthMask	激活或关闭深度缓存的写操作
glStencilMask	控制模板平面中单个位的写操作
glAlphaFunc	指定Alpha检验函数
glDepthFunc	指定深度比较中使用的数值
glStencilFunc	设置模板检验函数

4.5 OpenGL ES 简介

OpenGL ES 是为嵌入式系统而开发的 3D 图形绘制编程接口，这是针对嵌入式系统所制定的 3D 绘图 API，它能够使 3D 绘图与游戏在不同的移动设备或是嵌入式系统上方便地应用。

OpenGL ES 是在 OpenGL 上发展出来的，它并不是一个新的技术，只是 OpenGL 的一个子集。OpenGL ES 是 OpenGL Embeddedded Subset 的缩写。OpenGL ES 能够运用 OpenGL 的资源，产生良好的 3D 效果。OpenGL ES 是个与硬件无关的软件接口，可以在不同的平台如 Windows95、Windows NT、Unix、Linux、MacOS、OS/2 之间进行移植。因此，支持 OpenGL ES 的软件具有很好的移植性，可以获得非常广泛的应用。由于 OpenGL ES 是 3D 图形的底层图形库，没有提供几何实体图元，不能直接用以描述场景。但是，通过一些转换程序，可以很方便地将 AutoCAD、3DS 等 3D 图形设计软件制作的 DFX 和 3DS 模型文件转换成 OpenGL ES 的顶点数组。在 OpenGL ES 中，只能使用几种几何图元（点、直线和多边形）来构建所需要的模型。

OpenGL ES 是继承了 OpenGL 良好的扩展性、跨平台特性、灵活直观的操作性，同时还针对嵌入式设备量身定制了许多新特性，主要包括：

(1)考虑到移动平台新层面的一些问题，其中最重要的是内存应用和功耗问题。这使得该 API 需要尽可能小的占用磁盘和内存空间，同时数据总量的交换也必须保持最小化，以此来保证功耗越低越好。

(2)OpenGL ES 还照顾到全系列的移动设备，能够让软件渲染模式的特效在尽可能多的硬件上实现。这同时也方便了开发者，这样他们可以开发软件渲染引擎，当技术可行后无缝转移到相应的渲染特性。

OpenGL ES 由许多协议(profile)所组成。每一个协议通常是 OpenGL 的子集加上额外的 OpenGL ES 延伸指令集。每一个协议都能够与 OpenGL 兼容，因此在最后定案前，都可以根据市场需要做修改。每一个协议都有其专有的表头(header file)与连结/执行库(link/runtime library)来相对应命令(command)与标志(token)，只需要管理一个超集(superset)表头，就可以在执行各个标志与命令前，做好有效的管理。应用软件可以据此知道 OpenGL 的版本，以做出相对应的响应。

现在的规格是由二个不同的协议所组成：一般协议(Common Profile)与紧急安全协议(Safety Critical Profile)。

一般协议(Common Profile)是为了消费者娱乐与各种硬件平台所制定，如PDA、桌上游戏盒或其他游戏平台等。它与各种不同的平台具有最大的兼容性。其内容包括：完整 3D 绘图功能、极佳的游戏平台、移动手机平台。安全协议(Safety Critical Profile)是为了增加消费者与特殊应用产品之间的可靠度(reliability)与可证明度(certifiability)。

OpenGL ES 能够加入延伸指令集以增加新的功能。OpenGL ES 也提供了一份延伸指令集供厂商选择是否使用。这份标准的延伸指令集是确保协议的一致性。当然，厂商也可以自行加入自己的延伸指令集。但自己的延伸指令集必需符合协议的各种规范并且和协议完全兼容。

OpenGL ES 也包含了平台接口层的规范，称为 EGL。这层接口和平台间是独立的，厂商可以选择是否将他放入自己的产品里。整合平台当然也包含了一致性的测试。厂商也可以定义他自己的平台接口层。

第5章 3D图形组件的设计与实现

软件开发需要设计出完善的解决方案并论证方案的可行性，然后进行合理的任务模块分配，熟练掌握开发环境和工具，这样才能顺利的完成工程的开发，保证软件产品质量。

本章在前面对课题背景、发展趋势、3D 图形学进行介绍及研究的基础上，论述了课题研究和开发的内容、任务模块的分配，介绍了开发平台和所用工具，阐述了 WindML 的体系结构，在此基础上开发了一套 3D 图形组件。

5.1 开发平台及工具

课题使用的开发平台是：Windows XP

课题使用的主要开发工具有：Tornado2.2/VxWorks5.5，WindML 3.0

5.1.1 VxWorks 集成开发环境——Tornado

所谓嵌入式系统实际上就是用户自己设计开发的计算机系统，这个系统可以简单到只有主板，CPU 和存储器，当然这样的计算机系统我们不可能在此之上直接开发软件，因此需要借助一台通用计算机来辅助开发。这台通用计算机可以是 PC 或工作站，我们称辅助我们软件开发的通用计算机为宿主机(Host)，用户自己开发的计算机系统为目标机(Target)。宿主机上要有一个集成开发环境(IDE)来辅助我们的软件开发，这就是嵌入式系统的开发环境。这种集成开发环境包括交叉编译器(Cross Compiler)和交叉调试器(Cross Debugger)，所谓交叉编译器就是在宿主机上编译生成可以在目标机上运行的代码 IMAGE，交叉调试器就是通过宿主机和目标机之间的某种连接方式(如以太网或者串口线)实现前后台调试。安装在宿主机上的 VxWorks 的集成开发环境被称为 Tornado。在 Tornado 安装的时候，集成开发环境和 VxWorks 的原材料（一些 obj 文件）都安装到宿主机上，进行开发时使用 Tornado 中包含的工程管理软件将用户自己的代码与 VxWorks 的核心有效的组合起来，按照目标机硬件环境的需要裁剪配置操作系统内核，编译生成的在目标机上运行的 IMAGE 内就包含操作系统，这个 IMAGE 就被称为 VxWorks。

Tornado 集成开发环境支持多种宿主机环境包括：Sun OS，Solaris OS，

Win98，Win2000，WinNT 等。Tornado 使开发者可以动态链接、加载目标模块到目标机系统上。Tornado 还集成了许多方便程序员开发和调试的工具，如：图形化工程配置工具(Project Facility/Configuration)，使用该工具可以对 VxWorks 操作系统及其组件进行自动地配置。集成仿真器(Integrated Simulator)VxSim，提供与真实目标机一致的调试和仿真运行环境，VxSim 仿真器作为核心工具包含在各个软件包中，因而开发者可以在没有 BSP、操作系统配置、目标机硬件的情况下，使用 Tornado 迅速开始开发工作。C/C++编译环境(C/C++ Compilation Environment)，Tornado 提供交叉编译器、iostreams 类库和一些工具来支持 C 语言和 C++语言，交叉编译器进行了许多优化，允许开发者能够迅速产生高效，而简洁的代码。命令行执行工具 WindSh，WindSh 是 Tornado 所独有的功能强大的命令行解释器，可以直接解释执行 C 语句表达式、调用目标机上的 C 函数、访问系统符号表中登记的变量等等。

5.1.2 WindML 体系结构

要进行基于 VxWorks 的 3D 图形组件的开发，WindML 做为基础环节，是很重要的，所以有必要首先研究 WindML 自身结构及其和 VxWorks 系统之间的体系结构。WindML 由两个部分构成——软件开发包（SDK）和硬件开发包（DDK）构成。SDK 定义了应用程序代码和较底层硬件驱动层之间的应用程序接口，提供图形，设备管理，事件服务，字体和内存管理等方面 API 集合，使应用程序可以独立于底层硬件来开发。DDK 是位于 SDK 层和硬件之间的媒介层，直接面对应用程序的目标硬件设备，包括监视器，鼠标等等。DDK 层为硬件提供了一套完整的驱动（包括图形驱动，字体驱动，输入驱动等）和 API 集。

WindML 的设计理念是基于分层思想的模块化设计，它提供一套与硬件无关的逻辑 API（SDK 层）给应用层调用，而其底层（DDK 层）提供不同硬件构架的驱动，使程序具有好的移植性和可扩展性。WindML 各部分与系统各层次间的关系如图 5－1 所示：

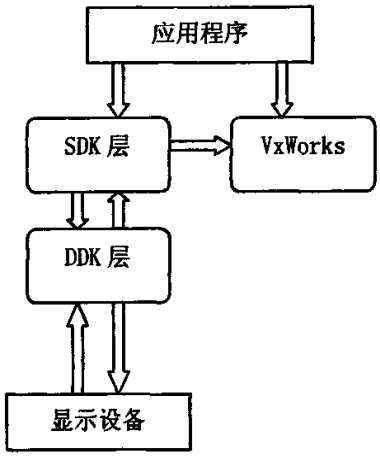

图5—1 WindML各部分与系统各层次间的关系

本文所要设计的3D图形组件将在SDK层的基础上开发，通过DDK层对底层硬件支持，本文所设计的3D图形组件将独立于硬件。

5.2 设计过程

随着图形技术和硬件水平的不断发展，3D图形渲染功能不断被增强和扩展，但基本的渲染原理至今尚未改变，如图5—2所示。本文就是根据这一流程来确定所需编写的接口函数并予以实现的。

要描绘三维场景，首先要对三维场景建模，即将实际场景中的三维物体转化为可被计算机接受的物体的几何模型数据，再将这些数据转换成3D API可直接接受的基本图元的形式，如点、线、多边形，另外对图像数据如纹理图像进行预处理，这种处理包括图像格式的转换、图像质量的改善等。

渲染流水线中的顶点是用来描述一系列包含有空间位置及其他有关属性的空间中的点。在渲染流水线中一个图元被描述为顶点的有序集合。顶点可以组成不同的图元，如基本图元点、线、三角形。在几何学中，顶点定义了多边形中边的交点，由空间坐标来表示。而渲染流水线中的顶点是构成如线，多边形之类的基本图元的基本元素，一个这样的顶点由空间中该点的坐标以及其他信息如法向量、纹理坐标、颜色等组成。

在对三维场景进行渲染前，需要设置相关的场景参数值。这些参数一般包括视点位置和视线方向，光源性质（镜射光、慢射光和环境光）、光源方位（距离和方向）、明暗处理方式（平滑处理或平面处理）、纹理映射方式等。

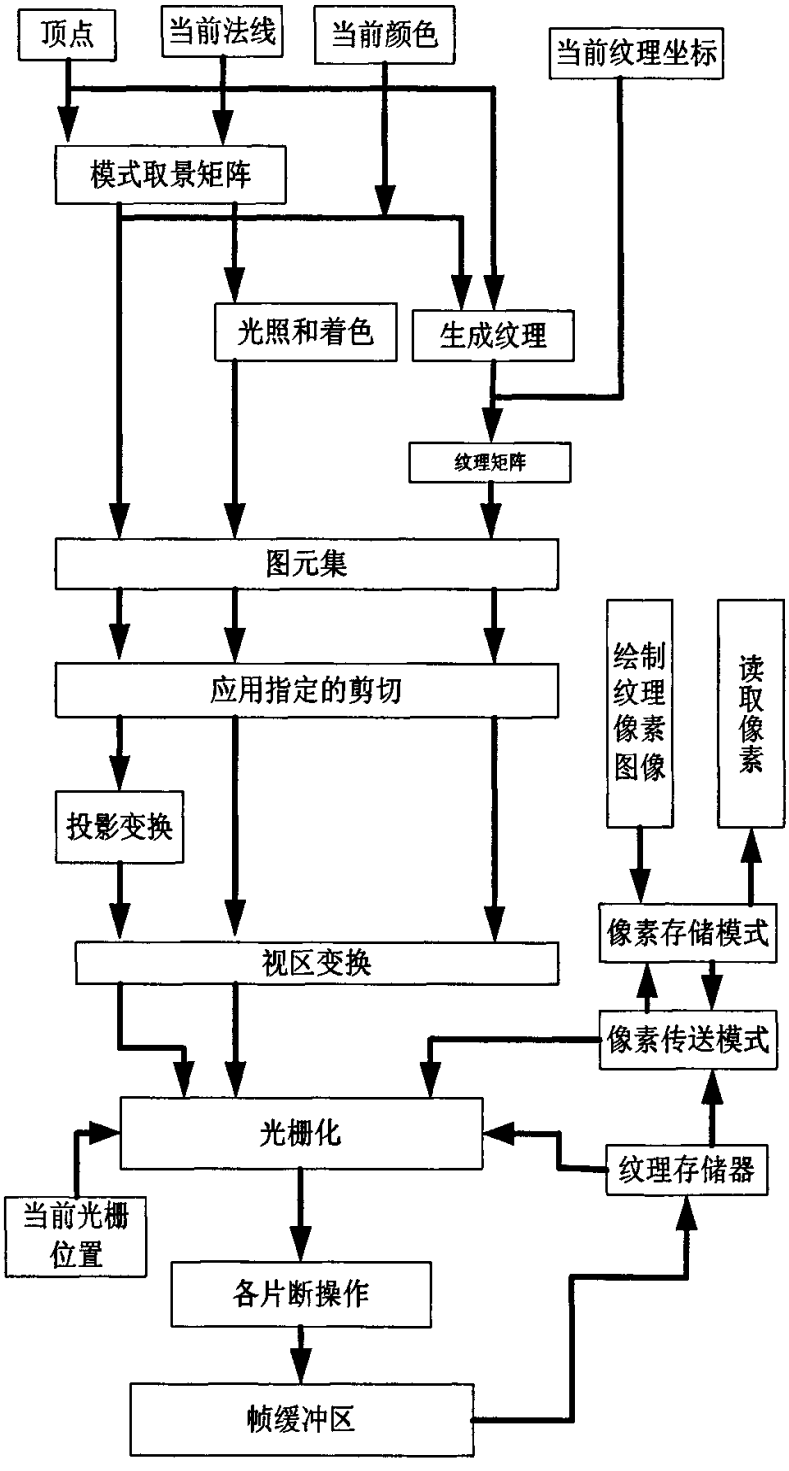

图5－2 3D图形处理流程图

由于嵌入式硬件设备方面的诸多限制和性能差异，在3D图形组件的开发过程中必须要综合考虑到多方面因素的平衡。下面列出一些关键的性能参数，它们都直接影响着嵌入式3D图形的效果。

1、屏幕大小

嵌入式设备一般使用的是都是小型屏幕。屏幕大小各不相同，从80*30到320*240，无法传统显示器相比，虽然屏幕分辨率持续提高，并且彩屏越来越普及，但是屏幕尺寸还是一直很小。分辨率的高低直接导致了3D图形显示的效果。

2、有限的色彩数量、明亮度

目前嵌入式设备能达到的色彩数量也是3D显示发挥的一个重要瓶劲。以手机为例，从过去几年的黑白屏幕手机，到如今的256色、4096色、以及真彩色，虽然现在出售的支持Java的手机的大部分都是彩屏手机，但在这些手机中12bit彩色非常流行。液晶屏幕由于其独特的发光原理，并不能达到传统显示器的亮度，同时也会导致很多色彩丰富的图案才强光下不能显示出原有的效果。

3、内存空间大小及实时性的限制

VxWorks是一种实时操作系统，对实时性的要求非常高，而且内存空间相当有限。因此嵌入式设备本身就对3D图形显示有所限制。在这样的限制条件下设计开发高效的3D图形组件显然是非常困难的。由于硬件的限制，目前嵌入式图形系统还存在许多难以实现复杂的3D功能模块，比如功能强大但计算量巨大的光线跟踪技术。

为了保障嵌入式系统的实时性，在设计3D组件就必须尽量减少软件所做的工作。这样必须对在PC机上使用的传统算法进行改进，由于嵌入式设备上的显示分辨率很低，屏幕也不大，在这种情况下，没有必要追求像在PC机上那样的高准确度和高清晰度，在一定范围内，我们可以通过改变算法来以一定的误差换取实时显示的速度。

本文在第2章和第3章中分析了绘制3D图形时各个环节的效率较高的图形算法，这些算法大都已被证明是目前已知效率最高的算法，因此在比较了3D图形学各种技术的相关原理的算法后，本文选取了这些对内存空间要求相对比较低，运行速度较快的算法。在光照处理模块，本文设计了Phone简单光照模型。在实际编程过程中，对于WindML中已有的2维图形函数进行了总结，尽量利用WindML中已有的函数避免重复开发。因此本文所设计的3D组件也可以说是

WindML扩展的3D功能插件，参照OpenGL标准，本文实现的函数有绘制基本几何图元（顶点、直线、多边形）的函数，裁剪函数、矩阵转换函数、颜色、光照和纹理函数、位图函数、光栅化函数等。相对于OpenGL而言，本文设计的函数库对3D功能进行了大量的裁减，剔除了算法实现比较复杂而又不是很必要的3DAPI，对必需的3DAPI在算法上也进行了简化。到目前为止，本文设计了的3D图形库只能提供最小化的3D图形显示功能，它包含了实现3D绘制功能所需的最简化的图形函数，但它没有窗口函数，也没有从键盘和鼠标读取事件的函数。不过WindML提供了一些基本的窗口管理函数和事件驱动函数，本文对3D函数库的测试工作都是WindML的平台下进行的。

下面简要罗列一下本文已经实现的比较重要的3D图形接口函数，其中，这些接口函数的命名方式参照OpenGL函数库的命名方式，函数功能也对应于相应的OpenGL函数。这些函数全部由C语言编写而成。利用这些接口函数，我们可以实现一些简单的3D图形绘制功能。这些函数的作用可以参见本文第4章第4节。

1) 图元及顶点数组

glVertex glPolygonStipple glColorPointer

glNormalPointer glVertexPointer glTexCoordPointer

glDrawArrays glEnableClientState glDisableClientState

glDrawElements glClientActiveTexture

2)几何及投影变换

glRotate glScale glTranslate

glFrustum glOrtho

3)视口变换：

glDepthRange glViewport

4)矩阵操作

glLoadMatrix glLoadIdentity glMultMatrix

glMatrixMode glPushMatrix glPopMatrix

5)着色与光照

设置当前颜色、法向量：

glColor glNormal

指定光源、材料或光照模式参数值：

glLight glMaterial

指定正面多边形：

glFrontFace

选择一种阴影模式：

glShadeModel

6)剪切

指定一个剪切平面：glClipPlane

返回剪切平面系数：glGetClipPlane

7)纹理

控制如何将一个纹理应用于片断：

glTexParameter glTexEnv

设置当前纹理坐标：

glMultiTexCoord

指定一个一维或二维的纹理图像或纹理子图：

glTexImage2D glTexSubImage2D

复制一个或部分纹理：

glCopyTexImage2D glCopyTexSubImage2D

建立一个指定的纹理并优化纹理：

glBindTexture glDeleteTextures glGenTextures

8)雾化

设置雾参数：

glFog

这些接口函数基本上都是绘制3D图形相应功能所必需的函数，它们共同组成了一个小型的3D图形函数库，可以被用来开发比较简单的3D应用。

5.3 算法测试

本文在Tornado自带的仿真环境下对这些API函数进行了测试，测试环境为Tornado2.2/Vxworks5.5，WindML3.0。测试的步骤如下：

1)将在3维空间中的图元在显示设备上显示；

2)对3维空间中的物体进行旋转、平移、比例变换；

3)对3维空间中的物体进行着色处理；

4)对3维空间中的物体进行剪切处理；

5)对3维空间中的物体进行光照处理；

6)对3维空间中的物体进行纹理映射；

7）综合测试。

经过大量的实验验证，本文所设计的3D图形组件已能够在VxWorks成功运行。下面给出了两个测试结果，图5—3描绘了一个圆锥体，图5—4描绘了一个3D立方图。

图5—3 圆锥体

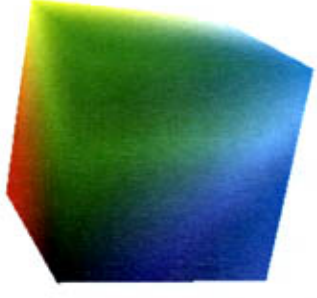

图5—4 一个多彩的立方体

第6章 总结与展望

随着多媒体信息技术、互连网、消费类电子产品的发展，嵌入式操作系统由于其占用内存少、可裁减、稳定性好的特点正得到越来越广泛的应用。嵌入式设备如移动电话开始使用具有音频和视频内容的大量多媒体应用，对高图形质量的多媒体应用的需求很大，这就需要更高质量的3维绘制功能。出于顺应技术发展潮流、适应市场选择、提高产品竞争力等方面的考虑，本文在掌握了3维图形绘制的原理和工作流程的基础上，针对嵌入式设备占用内存少、可裁减、可移植、稳定性好的特点，在嵌入式操作系统VxWorks现有图形开发组件WindML所具有的二维图形绘制功能的基础上扩展3D功能，设计实现了一套3D图形库，使其能够被用来在VxWorks中绘制3D图形，具有较强的市场价值和实用意义。

本文比较全面地讨论了3D图形学的基本概念、发展历史及应用，根据其在嵌入式环境下的应用情况和发展过程，科学地分析了该技术的发展趋势。对3维图形模型进行了详细的分析，对3维图形绘制中所涉及的图元、投影、消隐、裁减、填充、光照、纹理映射等进行了细致剖析，考虑到嵌入式操作系统实时性强，占用内存小，目前对3D图形绘制功能的要求也相对简单，本文在比较分析各种3D图形学算法的基础上，对3D图形功能进行了大量的裁减，剔除了一些不必要的3D功能，选取了一些简单高效的算法并针对嵌入式操作系统VxWorks进行改良，减小其对存储空间和运行时间的依赖，使之更多的考虑到嵌入式设备的实际情况。所设计的3D函数库在原型上借鉴OpenGL的函数原型，保证所有入口函数的自描述性，提高了其应用性。

嵌入式3D开发是个新兴的概念，并且正处于发展的高潮期。在未来几年内嵌入式设备对3D图形绘制的需求将大大提升，大量的研究将会集中于如何对各个相关领域已有的研究成果加以改进以使它们更符合3维图形显示。由于嵌入式设备对实时性、可靠性要求严格，本文所设计3D算法仍需要改进以加强显示的实时性。另外由于我个人能力有限，虽然做了很多工作，但由于3D图形系统本身是一个庞大的系统，需要不断的改进和维护，因此嵌入式3D图形设计问题还有许多问题有待解决。考虑到系统的效率问题，下一步应该重点解决的仍是内存受限和加强实时性的问题。为了更好的提高嵌入式系统的效率，仍需要改进一些

结构和算法来解决和内存之间的矛盾。其次是要实现更多的 3D 功能和开发 3D 扩展库，尽可能方便用户的操作和使用。

参考文献

[1]李颖等编著，OpenGL函数与范例解析手册，北京-国防工业出版社，2002.1
[2]Donald Hearn， M. Pauline Baker著，Computer graphics with OpenGL ，Beijing ：Publishing House of Electronics Industry， 2004
[3](美)Mason Woo，Jackie Neider， Tom Davis， Dave Shreiner著，OpenGL编程权威指南，北京-中国电力出版社，2001.8
[4](美) Richard S， Wright， Michael Sweet著，OpenGL超级宝典（第二版），北京-人民邮电出版社，2001.6
[5](美) Donald Hearn， M. Pauline Baker著.计算机图形学(第三版).北京-电子工业出版社，2005.6
[6](美)思詹著.交互式计算机图形学：基于OpenGL的自顶向下方法，北京-清华大学出版社，2006.02
[7]孔祥营，柏桂枝编著，嵌入式实时操作系统VxWorks及其开发环境Tornado ，中国电力出版社，2002.1
[8]罗国庆等编著，VxWorks与嵌入式软件开发，机械工业出版社，2003.9
[9]王学龙编著，嵌入式VxWorks系统开发与应用，人民邮电出版社，2003.10
[10](美)Wind River著，Tornado用户指南，清华大学出版社，2004.10
[11](美)Wind River著，VxWorks程序员指南，清华大学出版社，2003.8
[12]周启平,张杨,吴琼编著,VxWorks开发指南与Tornado实用手册,中国电力出版社,2004.7
[13]周启平，张杨编著，VxWorks程序员速查手册，机械工业出版社，2005.2
[14](美)罗杰斯(Rogers，D.F.)著，计算机图形学的算法基础，科学出版社，1987.11
[15]
[16](美)Donald Hearn，M.Pauline Baker著.计算机图形学:C语言版，清华大学出版社，1998.2
[17](英) Alan Watt著 ，3D计算机图形学(原书第3版)，机械工业出版社，2005.7
[18]WIND MEDIA LIBRARY Programmer’s Guide - SDK，3.0.2.Wind River Systems,Inc.2003
[19]WIND MEDIA LIBRARY API Reference，3.0.2.Wind River Systems，Inc，2003
[20]WIND MEDIA LIBRARY Programmer’s Guide - DDK， 3.0.2. Wind River Systems， Inc. 2003
[21]蔡华，卞新高，史中权，丁坤，基于VxWorks的WindML图形界面开发方法，工业控制计算机，2005，(8)
[22]梁勇，孟桥，刘铁英，嵌入式实时操作系统VxWorks中的图形界面设计，电测与仪表，2005,(9)
[23]The OpenGL graphics System:A Specification(Version 2.0 - October 22， 2004)，Kurt Akeley，Mark Segal，Silicon Graphics，Inc
[24]OpenGL Reference Manual，OpenGL ARB(Addison-Wesley，1992)
[25]OpenGL Programming Guide by Neider，Davis，and Woo(Addison-Wesley，1993)
[26]Paul Martz 著，OpenGL 2.0 精髓，人民邮电出版社，2006.7
[27](美) Randi J. Rost 著， OpenGL 着色语言人民邮电出版社 2006.10

[28]和平鸽工作室编著，OpenGL 高级编程与可视化系统开发 高级编程篇，中国水利水电出版社，2006.1
[29] Donald Hearn， M. Pauline Baker著，Computer graphics with OpenGL，Publishing House of Electronics Industry， 2004
[30]徐惠民主编，基于 VxWorks 的嵌入式系统及实验，北京邮电大学出版社，2006.9
[31]李春雨主编，计算机图形学理论与实践，北京航空航天大学出版社，2004
[32]孙家广， 杨长贵编著，计算机图形学，清华大学出版社，1995.5
[33]任爱华编著，计算机图形学，北京航空航天大学出版社，2005.12
[34](美) Philip J. Schneider， David H. Eberly 著，计算机图形学几何工具算法详解，电子工业出版社 2005.1
[35]陆国栋编著，工程计算机图形学，科学出版社，2004.8
[36](日)中嶋正之主编，三维计算机图形学，科学出版社，2004.3
[37]李忠民著，ARM 嵌入式 VxWorks 实践教程，北京航空航天大学出版社，2006.3
[38]肖世杰，嵌入式图形系统若干关键技术研究， 华中科技大学硕士学位论文，2004
[39]John Kessenich 著，The OpenGL ES Shading Language.The Khronos Group Inc. 2005
[40]Jon Leechzh 著，EGL Version 1.2. Khronos Native Platform Graphics Interface， 2005.7
[41]高翔，嵌入式三维图形引擎的设计与实现，电子科技大学硕士学位论文，2005
[42]谭建荣著， CAD 方法与技术，科学出版社，2005.8
[42]周勇,唐泽圣,Sorting-Cube 有效的多边形深度排序算法,中国计算机学会全国第八届 CAD/Graphics’94 学术年会论文专刊

论文发表情况

叶章文，慕德俊，《基于投影特征的商标图像检索方法》，科学技术与工程，已录用，拟于2007年6月发表

致 谢

首先感谢我尊敬的导师慕德俊教授，感谢慕老师给我提供一个独立自主进行科研任务的宝贵机会，在我理论学习、开题和论文的每一个环节中，无不得到了他悉心的教诲和热情的帮助。在导师的影响下，我自己也可以做到扎扎实实地学习理论知识，认认真真地进行实验。我愿借此机会向导师表示深深的谢意。

感谢郭达伟老师的帮助和指导，郭老师朴实严谨的治学态度给我留下了深刻的印象。

感谢教研室各位老师给我提供各方面的帮助。

感谢教研室的吕海波、张仲敏、曾群等等同学的无私帮助，大家对待我像一家人一样，在学习上和生活上都为我提供了很大帮助，在此向大家表示感谢。

最后要感谢的是我的父母，他们默默的、无私的支持我完成学业，感谢他们对我的爱，同时感谢姐姐对我生活上的关心。

我愿在未来的工作过程中，以更加丰厚的成果来答谢曾经关心、帮助和支持过我的所有领导、老师、同学、同事和朋友。

附录

四元数（Quaternions）为数学家 Hamilton 于 1843 年所创的三维的复数。其形式为 $w+xi+yi+zk$，其中 i、j、k 的关系如下：

$i^2=j^2=k^2=-1$

$i*j=k=-j*i$

$j*k=i=-k*j$

$k*i=j=-i*k$

假设有两个四元数：

$q_1=w_1+x_1i+y_1j+z_1k$，$q_2=w_2+x_2i+y_2j+z_2k$

四元数的加法定义如下：

$q_1+q_2=(w_1+w_2)+(x_1+x_2)i+(y_1+y_2)j+(z_1+z_2)k$

四元数的乘法定义如下：

$q_1*q_2=(w_1*w_2-x_1*x_2-y_1*y_2-z_1*z_2)+(w_1*x_2+x_1*w_2+y_1*z_2-z_1*y_2)i+$

$(w_1*y_2-x_1*z_2+y_1*w_2+z_1*x_1)j+(w_1*z_2+x_1*y_2-y_1*x_2+z_1*w_2)k$

由于 $q=w+xi+yj+zk$ 中可以分为纯量 w 与向量 $xi+yj+zk$，所以为了方便表示，将 q 表示为（S, V），其中 S 表示 w，V 表示向量 $xi+yj+zk$，所以四元数乘法又可以表示为：

$q_1*q_2=(S_1+V_1)*(S_2+V_2)=S_1*S_2-V_1\cdot V_2+V_1XV_2+S_1*V_2+S_2*V_1$

其中 $V_1\cdot V_2$ 表示向量内积，V_1XV_2 表示向量外积。

定义四元数 $q=w+xi+yj+zk$ 的 *norm* 为：

$N(q)=|q|=x^2+y^2+z^2+w^2$

满足 $N(q)=1$ 的四元数集合，称之为单位四元数。

定义四元数 $q=w+xi+yj+zk$ 共轭（Conjugate）为：

$q^*=[S-V]$

定义四元数的倒数为：

$1/q=q^*/N(q)$

假设有一任意旋转轴的向量 $A(X_a,Y_a,Z_a)$ 与一旋转角度 θ，可以定义一个四

元数：$q = w + xi + yj + zk$

$x = s * X_a$

$y = s * X_b$

$z = s * X_c$

$w = \cos(\theta/2)$

$s = \sin(\theta/2)$

使用四元数来表示的好处是可以简单的取出旋转轴与旋转角度。

假设有一向量 $p(X,Y,Z)$ 对着一单位四元数 q 做旋转，则将 P 视为纯量的四元数 $Xi + Yj + Zk$，则向量的旋转经验证如下：

$Rot(p) = qpq^*$

四元数具有纯量与向量，为了计算方便，将之以矩阵的方式来表现四元数的乘法，假设将四元数表示如下：

$q = [w,x,y,z] = [S,V]$

两个四元数相乘 $q'' = q * q'$ 的矩阵表示法如下所示：

$$\begin{bmatrix} x'' \\ y'' \\ z'' \\ w'' \end{bmatrix} = \begin{bmatrix} w & -z & y & x \\ z & w & -x & y \\ -y & x & w & z \\ -x & -y & -z & w \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

若令 $q = [S,V] = [\cos\theta, u * \sin\theta]$，其中 u 为单位向量，而令 $q' = [S',V']$ 为一四元数，则经过验证，可以得出 $q * q' * q^{(-1)}$ 会使得 q' 绕着 u 轴旋转 2θ。

由四元数的矩阵乘法与四元数的旋转，可以验证出上面的旋转公式可以使用以下的矩阵乘法来达成：

$$\begin{bmatrix} x'' \\ y'' \\ z'' \\ w'' \end{bmatrix} = \begin{bmatrix} 1-2(y^2+z^2) & 2(x*y-w*z) & 2(w*y+x*z) & 0 \\ 2(x*y+w*z) & 1-2(x^2+z^2) & 2(y*z-w*x) & 0 \\ 2(x*z-w*y) & 2(y*z+w*x) & 1-2(x^2+y^2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

如果要让向量 (x',y',z') 对某个单位向量轴 $u(x,y,z)$ 旋转角度 2θ，则 $w = \cos\theta$，代入以上的矩阵乘法，即可得旋转后的 (x'',y'',z'')。

基于VxWorks的3D图形组件的设计

作者：叶章文
学位授予单位：西北工业大学

参考文献(43条)

1. 李颖 OpenGL函数与范例解析手册 2002

2. Donald Hearn. M Pauline Baker Computer graphics with OpenGL 2004

3. Mason Woo. Jackie Neider. Tom Davis. Dave Shreiner OpenGL编程权威指南 2001

4. Richard S Wright. Michael Sweet OpenGL超级宝典 2001

5. Donald Hearn. M Paul ine Baker. 蔡士杰. 宋继强. 蔡敏 计算机图形学 2005

6. 思詹 交互式计算机图形学:基于=OpenGL的自顶向下方法 2006

7. 孔祥营. 柏桂枝 嵌入式实时操作系统VxWorks及其开发环境Tornado 2002

8. 罗国庆 VxWorks与嵌入式软件开发 2003

9. 王学龙 嵌入式VxWorks系统开发与应用 2003

10. Wind River Tornado用户指南 2004

11. Wind River. 王金刚. 高伟. 苏琪 VxWorks程序员指南 2003

12. 周启平. 张杨. 吴琼 Vxworks开发指南与Tornado实用手册 2004

13. 周启平. 张杨 VxWorks程序员速查手册 2005

14. 罗杰斯 计算机图形学的算法基础 1987

15. [15]

16. Donald Hearn. M Pauline Baker 计算机图形学:C语言版 1998

17. Alan Watt. 包宏 3D计算机图形学 2005

18. WIND MEDIA LIBRARY Programmer's Guide-SDK,3.0.2 2003

19. WIND MEDIA LIBRARY API Reference,3.0.2 2003

20. WIND MEDIA LIBRARY Programmer's Guide-DDK,3.0.2 2003

21. 蔡华. 卞新高. 史中权. 丁坤 基于Vxworks的windML图形界面开发方法 2005(08)

22. 梁勇. 孟桥. 刘铁英 嵌入式实时操作系统Vxwbrks中的图形界面设计 2005(09)

23. Kurt Akeley The OpenGL graphics System:A Specification 2004

24. OpenGL Reference Manual,OpenGL ARB 1992

25. OpenGL Programming Guide by Neider,Davis,and Woo 1993

26. Paui Martz OpenGL 2.0精髓 2006

27. Randi J Rost OpenGL着色语言 2006

28. 和平鸽工作室 OpenGL高级编程与可视化系统开发高级编程篇 2006

29. Donald Hearn. M Pauline Baker Computer graphics with OpenGL 2004

30. 徐惠民 基于VxWorks的嵌入式系统及实验 2006

31. 李春雨 计算机图形学理论与实践 2004

32. 孙家广. 杨长贵 计算机图形学 1995

33. 任爱华 计算机图形学 2005

34. Philip J Schneider. David n Eberly. 周长发 计算机图形学几何工具算法详解 2005

35. 陆国栋 工程计算机图形学 2004

36. 中嶋正之 三维计算机图形学 2004

37. 李忠民. 杨刚. 顾亦然. 刘尚军 ARM嵌入式VxWorks实践教程 2006

38. 肖世杰 嵌入式图形系统若干关键技术研究[学位论文]硕士 2004

39. John Kessenich The OpenGL ES Shading Language 2005

40. Jon Leechzh EGL Version 1.2.Khronos Native Platform Graphics Interface 2005

41. 高翔 嵌入式三维图形引擎的设计与实现[学位论文]硕士 2005

42. 谭建荣.陆国栋.张树有 CAD方法与技术 2005

43. 周勇.唐泽圣 Sorting-Cube有效的多边形深度排序算法

本文读者也读过(10条)

1. 王国夫.孙尧.苏柏泉 WindML事件驱动机制分析[期刊论文]-计算机工程2003,29(10)

2. 魏银英 基于VxWorks的成像声纳显控软件技术研究[学位论文]2008

3. 钟悠 基于VxWorks的嵌入式通信接口设备的设计与实现[学位论文]2007

4. 周进.ZHOU Jin WindML下汉字字体引擎的实现[期刊论文]-上海船舶运输科学研究所学报2006,29(1)

5. 刘巍.丁尔刚.LIU Wei.DING Er-gang 嵌入式雷达终端软件的实时性设计[期刊论文]-测控技术2007,26(6)

6. 张翀.赵刚.江勇.ZHANG Chong.ZHAO Gang.JIANG Yong 基于VxWorks操作系统的三维图形驱动开发[期刊论文]-四川理工学院学报（自然科学版）2009,22(2)

7. 董磊.周金明.杨磊 VxWorks汉字显示解决方案[期刊论文]-单片机与嵌入式系统应用2002(1)

8. 刘璟.谢拴勤 嵌入式VxWorksgh 下的图形用户界面开发工具Zinc[期刊论文]-电子产品世界2003(21)

9. 李海亮.石鹏程 VxWorks的WindML图形界面程序的框架分析[期刊论文]-工业控制计算机2007,20(1)

10. 曲宏松.张叶.曹立华.耿爱辉.陈涛.Qu,Hongsong.Zhang,Ye.Cao,Lihua.Geng,Aihui.Chen,Tao VxWorks下多页图形界面的设计[期刊论文]-微计算机信息2006,22(29)