武汉理工大学

硕士学位论文

基于JAVA3D虚拟现实场景的改进LOD算法研究

姓名：葛顺

申请学位级别：硕士

专业：通信与信息系统

指导教师：吕锋

20050501

摘 要

三维虚拟现实技术是多学科的交叉研究领域，是当前最热门的研究方向之一，正以其直观性、交互性等优良特性被广泛应用于建模与仿真、科学计算可视化、设计与规划、教育与训练、遥作与遥现、医学、艺术与娱乐等诸多方面，在科技发展和国民经济建设等方面发挥着越来越不可替代的作用。

本课题是图像信息处理与智能控制教育部重点实验室开放研究基金课题“数字城市的信息获取与可视化关键技术研究”的三维虚拟现实构建与应用子课题。本文的主要工作和内容如下：

1、相比当前主流的开发虚拟现实的工具诸如 VRML 等，本课题采用了全新的基于 Java 的 Java3D 三维图形软件包作为开发工具，该开发工具继承了其来自 Java 的纯粹的面向对象和网络应用的优良特性，使得构建的三维场景不仅效果逼真，而且功能齐全。本文在介绍了 Java3D 的建模方法等问题之后，从类和数据组织的总体的层次上介绍了构建一个完整场景的全部过程，从这个过程中，我们可以看到 Java3D 在构建虚拟现实技术中的优势和未来在网络应用中的巨大潜力，以此佐证了选择该图形包作为开发工具的新颖性和正确性。

2、另外，本文的工作重心是在分析了传统的 LOD 算法的不足的基础上对该算法提出了改进的思路，将原来的为不同显示细节建立多级 LOD 模型的思路，改而为将多级显示细节放在一个 LOD 模型的不同条件下触发显示，这样就在不改变分层显示的总体思路下，减少了建立多级 LOD 模型所产生的冗余数据，使得改进后的 LOD 算法不仅在运行效率和效果上达到和传统算法一致的程度，而且在数据量上比传统算法有相当程度的缩减，减少了数据负担，在网络应用中降低了传输数据的带宽需求，使得应用改进后 LOD 算法的三维虚拟现实技术更适合于网络的应用，增加了虚拟现实技术的应用范围。

3、最后，本文在对传统 LOD 算法和改进后 LOD 算法在虚拟现实技术中应用的对比的基础上得出改进 LOD 算法在三维虚拟现实场景构建中的意义所在，说明改进后的 LOD 算法的确可以达到预期的目标，降低程序的代码量和网络传输的数据量，并指出了课题的未来研究方向和应用前景。

关键词：虚拟现实　Java3D　LOD　三维场景

ABSTRACT

Three-dimension virtual reality technology is the cross research field among many subjects. It's one of the most famous research spots nowadays. Because of its advantages such as intuition and interaction, three-dimension virtual reality is applied in modeling and emulating, scientific caculation, visual designing and planning, distance education and distance control, medical science, art and amusement and so on. It has been having more and more unreplacable function in development in science and technology and national economic construction.

This subject, the modeling and application of three-dimension virtual reality, is the sub task of "the study of key technology of digital information obtaining in the digital city ", which is the picture information processing and intellectual control Ministry of Education key laboratory open research foundation subject. Major work and content of this paper are as follows:

1、Compared with the popular developing tools nowadays such as VRML, this subject use Java3D to be the developing tool, which bases on Java and is a new three-dimension image tool kit. Java3D inherits Java so many advantages such as OOP and fits for network application, which makes the graph scene made by Java3D has compeletly functions as well as vividly scene. After introduces Java3D and other relative things, the article introduces the whole process of modeling a integrate graph scene from the level of arranging classes and data, through which we confirm that Java3D has the advantage to be applied in the virtual reality technology and has the potential to be applied in the Internet in the future. This also proves that it is novel and correct for us to choose Java3D to be the developing tool.

2、At the same time, the important work of the article is to analyse the conventional LOD(Level Of Detail) algorithm and point out its flaws, then put forward the improving method, insteading of the conventional algorithm of modeling many LOD models to render different details, the improving algorithm try to conbine these different details to the integrate model and render them by different stimules,

this improving can help us to achieve the goal of reducing the redundant data made by many models while hold the conventional thinking of rendering with different levels, hoping not only the efficiency and effect can reach the level as the conventional algorithm can, but also compress the data which can let it more fit for applying in the Internet. It widens the area in which the virtual reality may use.

3、At last, based on comparing the improving LOD method and the conventional LOD algorithm used in the virtual reality modeling, the article makes the conclusion that the improving method has more advantages than the conventional algorithm, the improving LOD algorithm can reality achieve the goal to reduce the code and data to be transport on the Internet. Then the article points out forward work of this subject and application foreground.

KeyWords: virtual reality there-dimension graph scene Java3D LOD

第1章 绪 论

1.1 本课题的选题意义

虚拟现实技术（Virtual Reality），又称灵境技术，是从90年代起一项为科学界和工程界所关注的技术。它的兴起，为人机交互界面的发展开创了新的研究领域；为智能工程的应用提供了新的界面工具；为各类工程的大规模的数据可视化提供了新的描述方法。这种技术的特点在于，计算机产生一种人为虚拟的环境，这种虚拟的环境是通过计算机图形构成的三度空间，或是把其它现实环境编制到计算机中去产生逼真的“虚拟环境”，从而使得用户在视觉上产生一种沉浸于虚拟环境的感觉。这种技术的应用，改进了人们利用计算机进行多工程数据处理的方式，尤其在需要对大量抽象数据进行处理时；同时，它在许多不同领域的应用，可以带来巨大的经济效益。虚拟现实技术以三个I，即Immersion沉浸感，Interaction交互性，Imagination思维构想性，作为虚拟现实技术最本质的特点，并融合了其它先进技术。在国际互联网发展迅猛的今天，被广泛的应用于建模与仿真、科学计算可视化、设计与规划、教育与训练、遥作与遥现、医学、艺术与娱乐等多个方面。

那么在最简化的情况下，虚拟现实变成一个只有可视界面，让观者从视觉的角度感受到一个虚拟的世界，这也是在电子技术、计算机技术、网络通信技术飞速发展的今天应用最为广泛的虚拟现实了。虚拟现实技术甚至可以与无线通信技术相结合，为移动用户提供可视化的应用。

Java3D是SUN公司最近几年发布的专为三维场景构建的软件包。它继承了Java语言的优良特性，为虚拟现实技术的发展开辟了新的路径。它的跨平台性，面向对象性与当前网络技术相结合，更为虚拟现实技术做出了巨大的推动作用。本文以Java3D为开发工具来研究虚拟现实场景技术，可以直接应用于当前各个领域，为经济发展作出贡献。

层次细节技术（即LOD技术）在传统的建模语言VRML中已经有应用。实践证明，该项技术可以大幅提高图像显示的速度和降低图像生成的计算量。SUN把这项技术引入Java3D中。这项技术的实质是以冗余数据作为代价来实现场景

构建时的层次显示以提高效率。本文提出了在使用 Java3D 创建几何形体和实物时对形体和实物的细节使用条件触发来决定是否显示的方法来实现层次细节技术，这样不仅达到了显示的高效性也避免了冗余数据。该方法在大型场景的构建中效果尤为明显。

1.2 本文相关研究现状

目前开发虚拟现实三维场景的工具主要有 VRML、Direct3D、OpenGL 等。VRML 全称是 Virtual Reality Model Language，是一种虚拟现实建模语言。它主要用于嵌入到 HTML 网页中，以简洁的语法构建满足应用要求的三维场景。现在在网页中应用相当广泛。但是 VRML 也有它的不足，VRML 作为一种嵌入式的建模语言，只能满足一些一般应用，对于物体碰撞等高层应用，VRML 就无能为力了。

Microsoft Direct3D 是交互媒体技术的实时三维图形组件，它是 Microsoft 公司推出的三维图形编程 API，它主要应用于三维游戏的编程，虽然用 Direct3D 做出的场景能达到非常好的效果，但是目前相关的学习资料难于获得，并且由于它一般需要 VC 等编程工具进行编程，需要编程人员具有较高的 C++等高级语言的编程功底，因而难以普及。

另一方面，作为一种事实上的标准，由于 OpenGL 的跨平台特性，许多人利用 OpenGL 编写三维应用程序，不过对于一个非计算专业的人员来说，利用 OpenGL 编写出复杂的三维应用程序是比较困难的，且不说 C（C++）语言的掌握需要花费大量时间精力，当我们需要处理复杂问题的时候，我们不得不自己完成大量非常繁琐的工作。当然，对于编程高手来说，OpenGL 是他们发挥才能的非常好的工具。显然，这就限制了它的普及和应用，使得它难以在市场极其广阔的虚拟现实中得到广泛应用。

但是这样的问题在 SUN 公司推出 Java3D 开发包之后就有了改观。Java3D 是建立在 Java2（Java1.2）基础之上的，Java 语言的简单性使 Java3D 的推广有了可能。OpenGL 和 Java3D 之间的比较可以看成汇编语言与 C 语言之间的比较，一个是低级的，一个是高级的（也许这样比较不太恰当）。Java3D 给我们编写三维应用程序提供了一个非常完善的 API，它可以帮助我们：

生成简单或复杂的形体（也可以直接调用现有的三维形体）；使形体具有颜

色、透明效果、贴图；可以在三维环境中生成灯光、移动灯光；可以具有行为（Behavior）的处理判断能力（键盘、鼠标、定时等）；可以生成雾、背景、声音等；可以使形体变形、移动、生成三维动画；可以编写非常复杂的应用程序，用于各种领域如虚拟现实。

相比前面列出的当前几种虚拟现实主流开发工具而言，Java3D 比 VRML 功能强大，比 Direct3D 和 OpenGL 更易于为大众所接受，并且能在当前大多数平台上流畅的运行，学习并将 Java3D 应用于虚拟现实将会是更大的推动虚拟现实技术的前进和市场化，深入的改变人们的社会生活的方方面面，更好的促进国民经济的发展和人民的生活。

层次细节技术（LOD 技术）其实在当前的主流三维虚拟现实场景的建立和生成语言例如 VRML 中已有较成熟的应用。大量的应用已经证明该项技术能够在物体距离视点较远时避免显示其细节而提高显示效率，同时降低系统的运算量。在 Java3D 中，LOD 技术同样将有广泛和深入的应用以提高系统的性能。

1.3 本文主要研究内容

本课题是图像信息处理与智能控制教育部重点实验室开放研究基金课题“数字城市的信息获取与可视化关键技术研究”的子课题，主要研究基于 Java3D 的虚拟现实实现技术。深入研究虚拟现实中用户与场景的研究和虚拟现实在跨平台操作，并实现该系统的可扩充性，涉及到 Java3D 的场景组成结构，三维实体的创建和调用，实现动画和交互方式，以及 Java3D 扩展包自动下载技术等，利用虚拟现实技术实现虚拟三维环境的漫游浏览，使人们能够在一个虚拟的环境内部中漫游。

通过本文的研究和论述，作者希望用 Java3D 开发出用于虚拟现实场景的各种基本形体类，并用 Java 和 Java3D 编写相应的程序调用上面这些基本形体类生成各种基本形体对象来再现虚拟场景，最后实现 Java3D 虚拟场景的 Internet 应用。

本文的篇章结构为：

第一章为绪论，简绍了本论文的选题意义以背景，主要研究内容等。

第二章为虚拟现实（VR）技术概述，大体介绍虚拟现实技术的起源、发展以及对社会发展，人民生活的重要意义，指出本文研究虚拟现实的实际意义。

第三章论述图像的虚拟现实系统，即所谓的桌面虚拟现实系统。并对比当前主流的几种开发图像虚拟现实系统工具，在比较了各自的特点后论述本文以Java3D 为开发工具的原因和优势所在。然后从 Java 入手，介绍了致力于开发三维可视化代码的 Java 开发包 Java3D。并认为 Java3D 的出现将会改变以往可视化开发工具复杂难以普及应用的现状，而促进虚拟现实技术的应用。

第四章是从已经应用的 LOD（Level of Detail）算法的分析入手，针对传统 LOD 算法的缺陷提出对 LOD 算法的改进思路。通过改进前后算法差别的对比，得出结论：改进后的算法不仅能够满足图像生成过程中的分级显示的高效性，并大大减少数据量，能够更好的应用于大型虚拟现实场景的构建和网络传输应用。

第五章是用 Java3D 语言开发虚拟可视化场景的具体过程。选择了适当的开发硬件和软件环境，确定了系统开发的整体构架，对各种几何体，数据等类和对象分类组织存储，通过纹理样本的获取和表面粘贴，使得三维场景更为逼真，同时指出可以用更改和增加环境背景信息的方法提高场景的效果。

第六章是使用改进过的 LOD 算法用以优化代码的运行。分析用 LOD 算法改进后的优化效果和意义。最后是本文的结论和下一步须要完成的工作，并对用 Java3D 实现虚拟现实和改进 LOD 算法的前景展望。

第 2 章　虚拟现实技术概述

2.1 虚拟现实技术简介

虚拟现实是指通过人的多种感知通道(视觉、听觉、触觉、嗅觉、味觉等)进行实时模拟与交互的高级人机交互系统。虚拟现实（Virtual Reality）又可称为人工现实（Artificial Reality）、虚拟环境（Virtual Environment）等。许多学者认为，虚拟现实具有沉浸（Immersion）、交互（Interaction）及想象（Imagination）等所谓的 3I 特性。虚拟现实技术在进入九十年代以后有了飞速的发展，在多媒体、科学计算可视化、计算机视觉等领域处处可见它的身影，它被广泛用于各种教育、各种模拟训练、游戏娱乐、大型军事演习、电影特技处理、建筑等行业中。在一个典型的虚拟现实系统主要由三部分组成：用来生成虚拟环境的计算机、一个或多个计算机操作人员、用来与虚拟环境交互的人机界面，它将是 21 世纪广泛应用的一种新技术。

1995 年，在德国斯图加特召开的虚拟现实国际会议（Virtual Reality World Conference）上，Stephan Ellis 给虚拟现实下了这样一个定义：

虚拟现实是一个当用户佩戴上适当的装置后观察到的综合的、交互式的、虚幻的环境，它能提供模仿物理环境的感官信息的协调表达[1-4]。

虚拟现实是近年来十分活跃的技术研究领域，是一系列高新技术的汇集，这些技术包括计算机图形学、多媒体技术、人工智能、人机接口技术、传感器技术以及高度并行的实时计算技术，还包括人的行为学研究等多项关键技术。虚拟现实是多媒体技术发展的更高境界，是这些技术的更高层次的集成和渗透。它能给用户以更逼真的体验，它为人们探索宏观世界和微观世界以及由于种种原因不便于直接观察事物的运动变化规律提供了极大的便利川。由于它的诱人前景，一经问世就立即受到了人们的高度重视。有关人士认为，80 年代是个人计算机的年代，90 年代是多媒体计算机的年代，21 世纪初将是虚拟现实技术的时代。为了把握虚拟现实这一新技术，美、英、日等国政府及大公司已不惜投入巨额资金进行该领域的研究与开发工作，并在许多应用领域显示出良好的应用。

从本质上说，虚拟现实就是一种先进的计算机用户接口，它通过给用户同时提供诸如视、听、触觉等各种直观而又自然的实时感知交互手段，最大限度地方便用户的操作，从而减轻用户的负担、提高整个系统的工作效率（2）。根据虚拟现实所应用的对象的不同，虚拟现实的作用可以表现为不同的形式，例如将某种概念设计或构思成可视化和可操作化;实现逼真的现场效果;达到任意复杂环境的廉价模拟训练目的等。

虚拟现实的定义可以归纳如下：虚拟现实是利用计算机生成一种模拟环境(如飞机驾驶舱、操作现场等)，通过多种传感设备使用户“投入”到该环境中，实现用户与该环境直接进行自然交互的技术。这里所谓模拟环境就是用计算机生成的具有表面色彩的立体图形，即是通过视、听、触觉等作用于用户，使之产生身临其境感觉的交互式视景仿真，它可以是某一特定现实世界的真实体现，也可以是纯粹构想的世界。传感设备包括立体头盔(Head Mounted Display, HMD)、数据手套(Data Glove)、数据衣(Data Suit)等穿戴于用户身上的装置和设置于现实环境中的传感装置(不直接戴在身上)。自然交互是指用日常使用的方式对环境内的物体进行操作(如用手拿东西、行走等)并得到实时立体反馈。

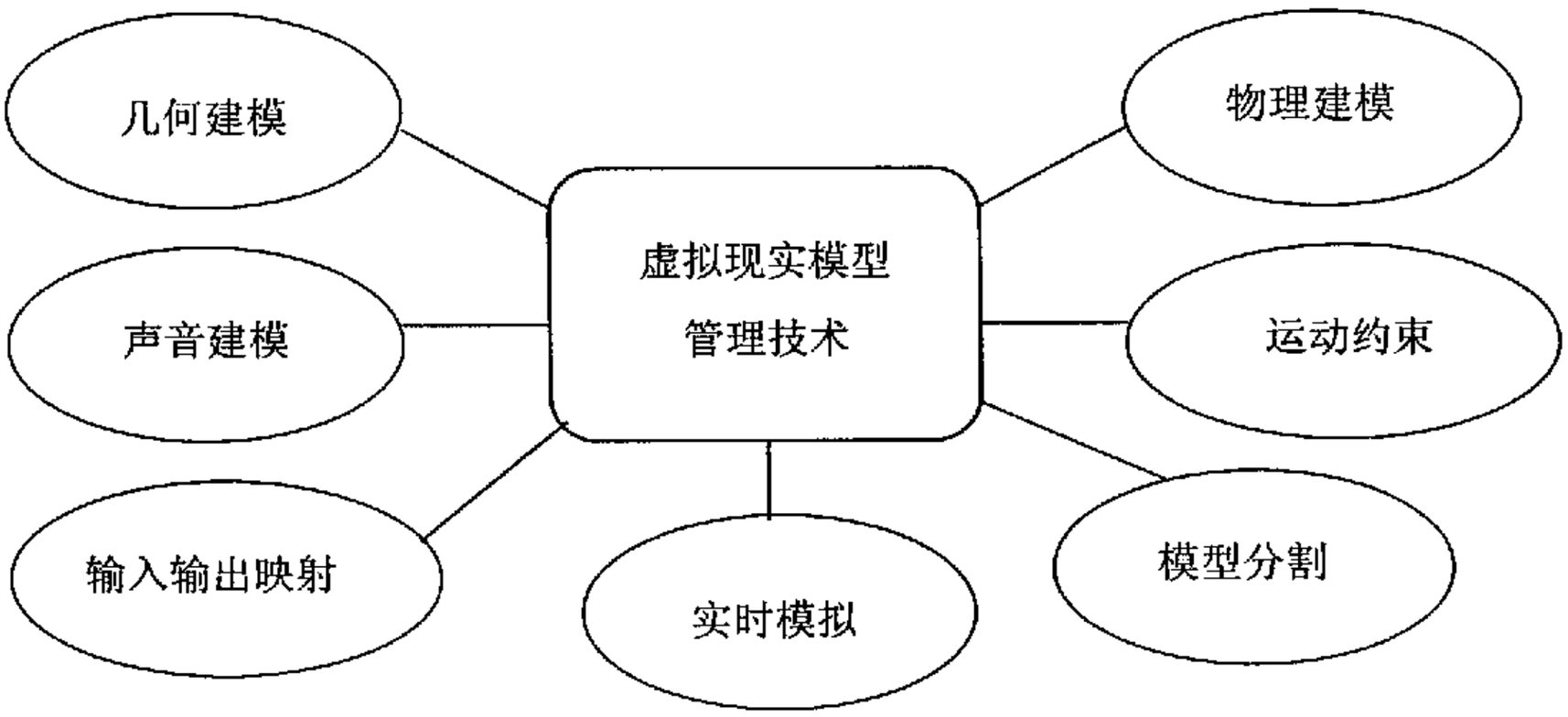


图 1－1　虚拟现实模型图

虚拟现实技术是一种逼真地模拟人在自然环境中视觉、听觉、运动等行为的人机界面技术。其基本特征可以用一个如图 1－2 的三角行来表示：第一个特征是沉浸，让参与者有身临其境的真实感觉;第二个特征是交互，虚拟现实的交互特性主要是通过使用虚拟交互接口设备实现人类自然技能对虚拟环境对象的

交互考察与操作；第三个特征是构想，从图形角度来说，传统的计算机图形学强调了三维场景在屏幕上的二维显示，虚拟现实技术则强调三维图形的立体显示。虚拟现实技术的根本目的是：不仅能够在多维信息空间仿真建模，而且能够帮助人们获取知识和形成新的概念。

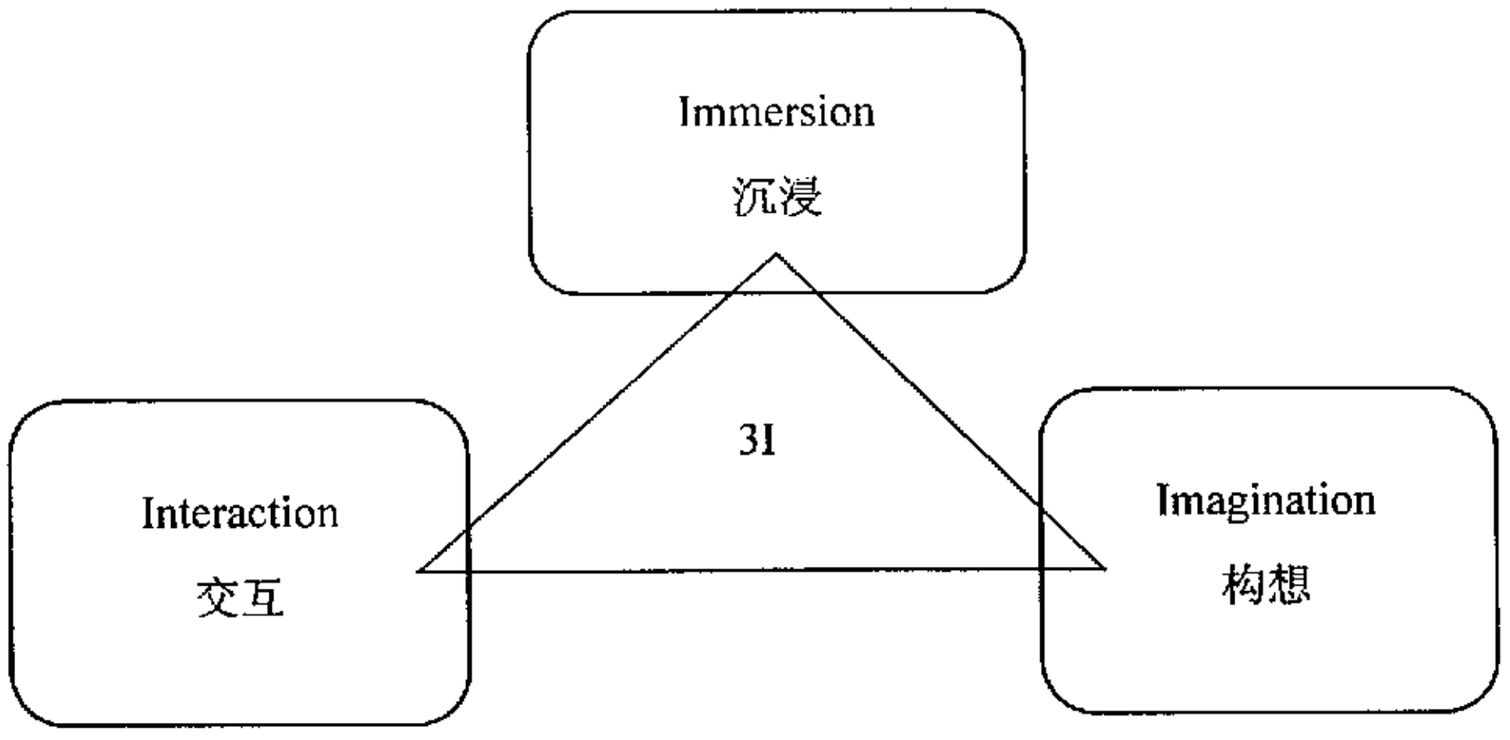


图 1－2　虚拟现实技术的基本特征

一个完整的虚拟现实系统可以分解为视觉、听觉、触觉等子系统。从组成角度来分析，虚拟现实系统包括主机系统、场景显示系统、虚拟现实接口设备。

Mark Green 教授给出了一个如图 1－3 所示的简明的虚拟现实应用系统模型。其中，计算包括所有应用中非图形的计算；几何建模包括一个计算中的数据的高级图形表示，或者说包括科学计算可视化；观察指用户查看应用数据，这是要用真实感图形表示的内容，这个内容的主要目的是向用户提供程序帮助；行动体(Active Agents)是指以同样方式仿真用户与系统中的对象交互。

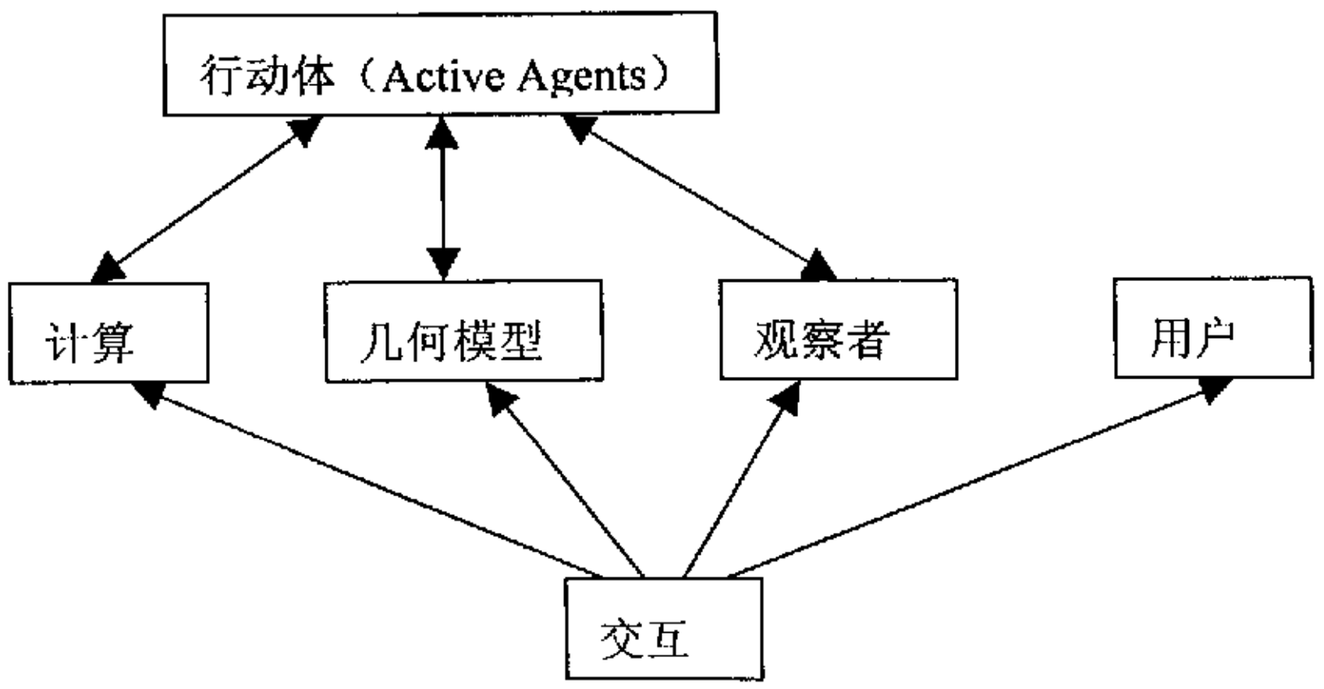


图 1－3　虚拟现实系统模型

根据参与者参与虚拟现实的不同形式以及沉浸的程度不同，参与者在虚拟环境中的活动可分为三个层次：桌面级的虚拟现实、投入的虚拟现实、增强现实性的虚拟现实。桌面级的虚拟现实利用个人计算机和低级工作站进行仿真，计算机的屏幕用来作为用户观察虚拟世界的一个窗口，各种外部设备一般用来驾驭虚拟环境。它包括基于静态图像的虚拟现实技术、VRML(虚拟现实造型语言)、桌面 CAD 系统等。投入的虚拟现实系统利用头盔式显示器或其它设备，把参与者的视觉、听觉和其它感觉封闭起来，并利用位置跟踪器、数据手套、其它手控输入设备等使参与者产生一种身在虚拟环境中、全心投入和沉浸其中的感觉。常见的投入式系统有:基于头盔式显示器的系统、投影式虚拟现实系统、远程存在系统等。增强现实性的虚拟现实用来增强参与者对真实环境的感受，也就是增强现实中无法感知或不方便感知的感受。这种类型的虚拟现实典型的实例是战机飞行员的平视显示器，它可以将仪表读数和武器瞄准数据投射到安装在飞行员面前的穿透式屏幕上，它可以使飞行员不必低头读座舱中仪表的数据，从而可集中精力盯着敌人的飞机和导航偏差。

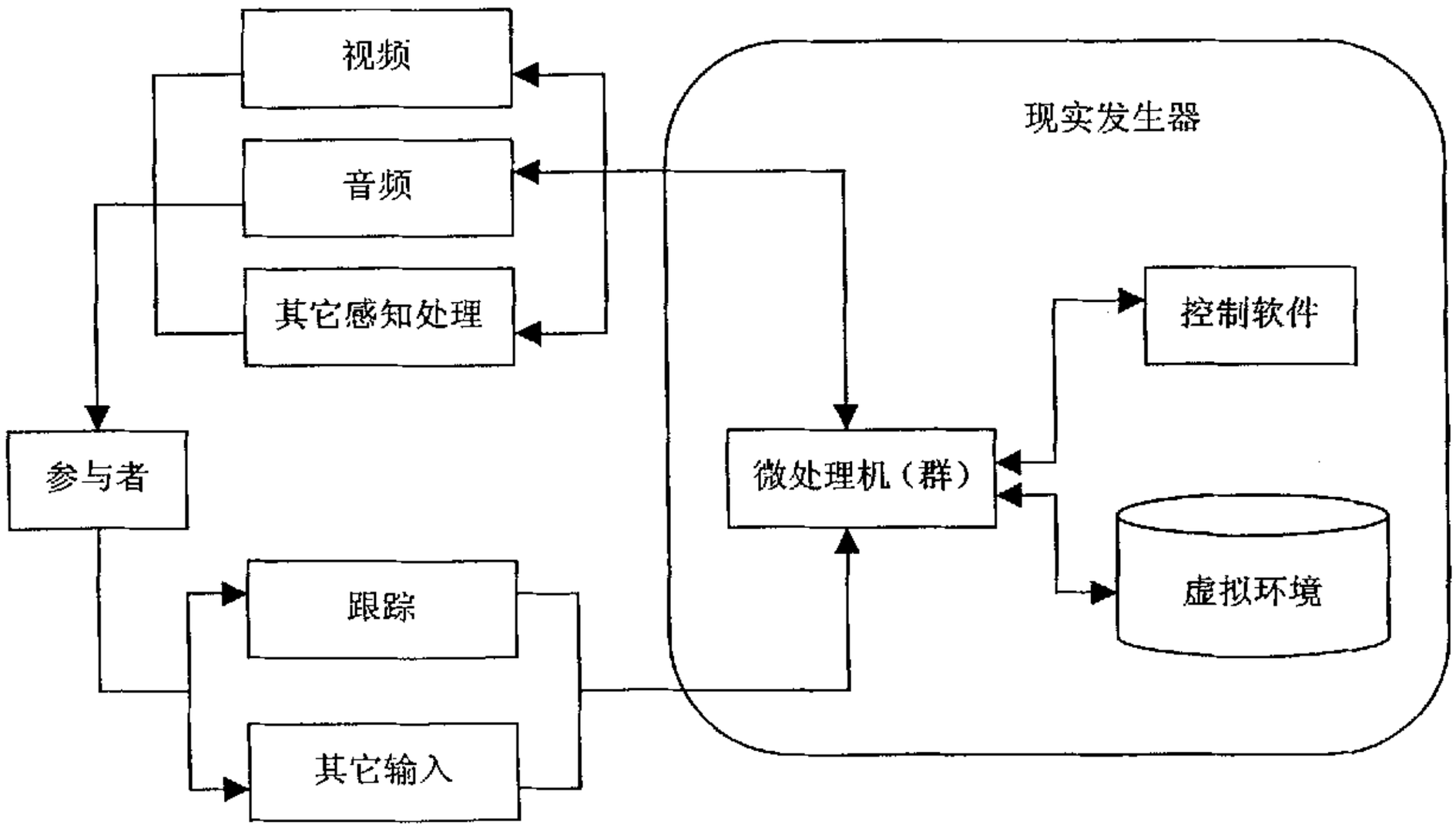


图 1—4　虚拟现实系统结构

虚拟现实系统可分为三大类：桌面虚拟现实系统、沉浸式虚拟系统和分布式虚拟系统。桌面虚拟由于采用标准的 CRT 显示器和立体显示技术，其分辨率

较高，价格较便宜。在使用时，桌面虚拟现实系统设定一个虚拟观察者的位置。桌面虚拟现实系统通常用于工程 CAD、建筑设计以及某些医疗应用。沉浸式虚拟现实系统利用头盔显示器把用户的视觉、听觉和其它感觉封闭起来，产生一种身在虚拟环境中的错觉。分布式虚拟现实系统则是在沉浸式虚拟现实系统的基础上将不同的用户通过网络联结在一起，共享同一个虚拟空间，使用户达到一个更高的境界[5-8]。

2.2 虚拟现实技术的主要研究内容

虚拟现实的研究内容粗略地分为虚拟现实技术本身及其应用两大类。由于本文属于虚拟现实技术的应用领域，这里将重点介绍应用领域的研究内容。

虚拟现实技术研究的主要内容包括:人与场景的融合技术、物体对象的仿真技术、三维地形建模技术、图形图象的实时生成与合成技术、虚拟现实接口在智能计算机系统中的应用、多维信息的表示、实时处理及并发处理的 OOP 技术、高性能的计算机图形处理硬件研究及分布式虚拟环境，基于网络环境的虚拟现实研究。目前相对成熟的感知建模和显示技术仅仅局限于视觉、听觉和触(力)觉感知模型，相对应的显示装置为立体显示器、空间声播放系统和触(力)觉反馈装置。虚拟现实系统中人机自然交互的研究内容仍然很不成熟，至今可用做人机自然交互操作的输入硬件还十分有限，操作不便，实时性差，定位困难，精度较低。

虚拟现实技术的应用领域十分广泛，我们可以把众多的应用归结为两大类：一类是“真实世界仿真”，另一类是“抽象概念建模”，也可称为“可视化”。真实世界仿真类应用的实例有大规模战争的战略和战术演练、飞机训练、航天飞机的风洞试验仿真、核爆模拟以及医学手术模拟等等。这类应用是针对那些采用实物仿真有困难，或受条件限制难以实现的场合。随着虚拟现实技术的成熟和普及，此类应用面将进一步扩大，并发挥越来越重要的作用。抽象概念建模应用的实例有综合环境模型的建立与评估、自然灾害的预测与评估、气象预报、石油勘探数据分析与预测、新型药物分子结构合成、虚拟原型设计与制造、远程教育、文物保护等。此类应用大多与科学计算可视化技术的应用相结合，充分发挥虚拟现实技术提供的沉浸感和自然人机交互技术的特点，为科学家提供直观的、更易理解的模型，便于分析和综合。本文属于虚拟现实技术在远程教

育方面的应用研究。虚拟现实技术在教育领域的应用，学生可以通过虚拟现实技术学习解剖或探索星系，尤其是那些与健康和安全有关的培训项目，纽约一家公司利用虚拟现实技术要求受训者步行穿过虚拟工厂，并了解公害状况，这种设身处地的体验远比读一本手册或听一堂课强的多。今后，学生们可以通过虚拟世界学习到他们想学的知识：化学专业的学生不必冒着爆炸的危险却可以做试验;天文学专业的学生可以在虚拟星系中遨游，以掌握它们的性质；历史专业的学生可以观看不同的历史事件，甚至可以参与历史人物的行动，英语专业的学生可以看莎士比亚戏剧，如同这些剧目首次上演一样，他们还可以进入书中与书中人物进行交流。随着虚拟现实技术的发展，在远程教育方面将得到越来越广泛的应用。

2.3 虚拟现实技术的发展历程

虚拟现实的发展史建立在相关技术的发展基础之上的。20 世纪，由于把人的想象力与电子学结合在一起，为虚拟现实的飞速发展建立了牢固的基础。换言之，电子学支持了电话、视频技术以及计算机的发展，今天，计算机又把这些技术组合在虚拟现实的环境中。现在，市场上已经出现了许多虚拟现实终端产品，既有以个人计算机为平台的低档系统，也有并行与分布式结构为基础的高性能系统。LINK 飞行模拟器是虚拟现实的先驱之一。1929 年 Edwin Link 设计了一种 LINK 飞行模拟器是虚拟现实的先驱之一。1929 年 Edwin Link 设计了一种竞赛乘坐器，它使得乘客有一种乘坐飞机飞行的感觉。这种乘坐器后来发展成为飞行模拟器，作为飞行员的训练设备。

60 年代初期，由 Morton Heiling 研制的 Sensorama 摩托车模拟器是虚拟现实技术的另一先驱，使参与者象坐着一台摩托车在街道上行驶，感觉到耳边的风声，马路边的电线杆在后退，座位在摇动，甚至闻到从食品店散发出来的阵阵诱人的香味。

80 年代以后，随着技术的进步，陆续地研制出较实用的头盔显示器、能提供六个自由度的数据手套、立体声耳机及相应的计算机硬件系统，为虚拟现实的研究奠定了良好的硬件基础。美国国防部高级项目计划局(Defense Advanced Research Projects Agency DARPA)成功开发了 Simulation Networking(模拟器网络计划)，简称 SIMNLT，它最初是企图将分散在不同地点的地面车辆(坦克、装甲

车)仿真器用计算机网络联系起来，形成一个整体战场环境，进行各种复杂任务的训练和作战演习。到 1990 年，SIMNET 被发展成为 DTS (Distributed Interactive Simulation)技术，扩展到包括陆、海、空各种武器平台的综合环境，并实现了体系对抗仿真。

但是，即使对虚拟现实技术的发展历史有清晰的认识，想要准确展望虚拟现实未来若干年的前景仍是不容易的。不过，虚拟现实的发展方向是很明确的：更快、更高质量的图形;更便宜、设计得更完善的头盔式显示器和其它输入、输出设备；更快的计算机处理能力。这些方面的改进将会影响到虚拟现实在实际应用中的各个方面，包括虚拟境界内的视觉、听觉质量，而且还会影响到虚拟现实在各个新领域内的广泛应用。虚拟现实将越来越广泛地进入日常生活。

2.4 虚拟现实国内外研究现状

虚拟现实技术汇集了计算机图形学、多媒体技术、人工智能、人机接口技术、传感器技术、高度并行的实时计算技术和人的行为学研究等多项关键技术。它给用户以逼真的体验，为人们探索宏观世界和微观世界中由于种种原因不变于直接观察的事物的运动变化规律，提供了极大的便利。由于它的诱人前景，一经问世就立即受到了人们的高度重视。

1)国外虚拟现实技术的研究现状

美国的研究状况:美国是虚拟现实技术的发源地。目前美国在该领域的基础研究主要集中在感知、用户界面、后台软件和硬件四个方面。

美国宇航局(NASA)的 Ames 实验室完善了 HMD，并将 VPL 的数据手套工程化，使其成为可用性教高的产品。NASA 研究的重点放在对空间站操纵的实时仿真上。

北卡罗来纳大学(UNC)主要研究:分子建模、航空驾驶、外科手术仿真、建筑仿真等。在显示技术上，UNC 开发了一个帮助用户在复杂视景中建立实时动态显示的并行处理系统，叫做像素飞机(Pixel Planes) 。

麻省理工学院(MIT)建立了一个虚拟环境下的对象运动跟踪动态系统。另外，MIT 还在进行“路径计划”与“运动计划”等研究。

SR 工研究中心建立了“视觉感知计划”，研究现有 UR 技术的进一步发展。它还利用遥感技术进行外科手术仿真的研究。

SOFTIMAGE 公司的专家们提出了渗透将有助于扩大虚拟现实的美学感，这是虚拟现实未来的一个发展方向。

欧洲的研究现状:欧共体(CEC)认为虚拟现实是一门新兴技术，已经组织了多次评价虚拟现实的专题活动。

德国 Damastadt 的 Fraunhofer 计算机图形学研究所开发出一种名为“虚拟设计”的虚拟现实组合工具，可使得图像伴随声音实时显示。

德国国家数学与计算机研究中心(GMD)专门成立了一个部门，研究科学视算与虚拟现实技术。研究的课题有虚拟现实表演，冲突检测，装订在箱子中的物体的移动，高速变换以及运动控制。

英国的 ARRL 有限公司关于远地呈现的研究试验主要包括虚拟现实技术重构问题。

荷兰的虚拟现实研究主要是研究一般性的硬件/软件结构问题、人员因素问题，以及在工业和培训中的应用。

日本的研究状况:在当前实用虚拟现实技术的研究与开发中日本是居于领先位置的国家之一，主要致力于建立大规模 UR 知识库的研究。另外在虚拟现实游戏方面也做了很多研究工作。

东京技术学院精密和智能实验室研究了一个用于建立三维模型的人性化界面，称为 SPIDAR 的系统。

NEC 公司的计算机和通信分部中的系统实验室开发了一种虚拟现实系统，它能让操作者使用“代用手”去处理三维 CAD 中的形体模型。该系统通过 UPL 公司的数据手套把对模型的处理与操作者手的运动联系起来。

东京大学的高级科学研究中心将他们的研究重点放在远程控制方面，最近的研究项目是主从系统。

富士通实验室有限公司正在研究的一个项目是虚拟生物与虚拟现实环境的相互作用。

2)国内虚拟现实技术的研究现状

我国的虚拟现实技术和一些发达国家相比，还有一定的差距，但已引起政府有关部门和科学家们的高度重视。根据我国的国情，制定了开展虚拟现实技术的研究，例如:九五规划、国家自然科学基金会、国家高技术研究发展计划等都把虚拟现实列入了研究项目。

北京航空航天大学计算系是国内最早进行虚拟现实研究、最有权威的单位

之一，他们着重研究了虚拟环境中物体物理特性的表示与处理;在虚拟现实中的视觉接口方面开发出了部分硬件;实现了分布式虚拟环境网络设计。

浙江大学 CAD&CG 国家重点实验室开发出了一套桌面型虚拟建筑环境实时漫游系统。另外，他们还研制出了一种新的虚拟环境快速漫游算法和一种递进网格的快速生成算法。

哈工大计算机系已经成功地虚拟出了人的高级行为中特定人脸图像的合成，表情的合成和唇动的合成并正在研究人说话时头势和手势动作。

清华计算机和技术系对虚拟现实临场感的方面进行了研究。

西安交通大学信息工程研究所对虚拟现实中的关键技术—立体显示技术进行了研究。

北方工业大学 CAD 研究中心完成了体视动画的自动生成部分算法与合成软件处理，完成了虚拟现实图像处理与演示系统的多媒体平台及相关的音频资料库，制作了一些相关的体视动画光盘。

西北工业大学 CAD&CAM 研究中心，上海交通大学图像处理及模式识别研究所，长沙国防科技大学计算机研究所，华东船舶工业学院计算机系，安徽大学电子工程与信息科学系也在 UR 方面进行了一些研究工作和尝试。

第 3 章　桌面虚拟系统及开发工具的选择

3.1 桌面虚拟现实简介

虚拟现实是一项高投入的技术,一般而言,一套基于图形工作站和 UNIX 操作系统的虚拟现实软、硬件系统,往往需要 400～500 万元人民币的资金。如何降低虚拟现实技术的投资,拓展应用领域,扩大用户范围,是摆在我们面前急需解决的课题。

桌面虚拟现实技术主要是利用个人计算机进行仿真,使用者通过计算机屏幕观察虚拟世界,利用外部设备辅助操纵虚拟世界中存在的各种物体,常见的外部设备有鼠标、跟踪球等。目前,我国基于 PC 机的虚拟现实开发技术已取得了可喜成绩,为扩大在各个领域的应用奠定了较好的基础。桌面虚拟现实技术除了具有虚拟现实技术的一般特点外,相对于图形工作站虚拟现实系统,其特点在于:

（1）投资少。随着计算机技术,尤其是 PC 机的软、硬件技术水平的大幅度提高,计算机的运算速度愈来愈快,计算机图形功能愈来愈强。当前 Inter P4 2.2G 和 AMD 2100+ CPU 的运算速度都能达到数 10 亿次,TI4600 图形卡数据传输率达到 3.2GB/s,为在微机上实现虚拟现实技术打下了坚实的基础。如此功能强大的 PC 机,其售价才 1 万多元人民币,而 Windows 操作系统下的开发工具和虚拟现实系系统软件的性价比相当高,基于 PC 机的虚拟现实软、硬件系统只需投入十几万到几十万元。

（2）易使用。用户易于掌握与使用是基于 PC 机虚拟现实系统的最大特点。众所周知,大多数人都能熟练使用微机和 Windows 操作系统,具有 Windows 操作系统风格的虚拟现实系统,对于用户来说,易于学习与掌握。

（3）适应性强。不同行业的用户,具有不同的应用领域,也就有着不同的需求和应用目的。PC 机虚拟现实系统的设计与开发是基于面向对象的程序编程技术（OOP）,针对不同的需求,完全可以将虚拟现实系统与为特定目的设计的软件组件（Component）有机地结合在一起,从而使虚拟现实系统适应来自各个方面的需求。

（4）应用面广。PC 机虚拟现实系统投资较少,使用户不但在心理上,而且在

实际承受能力上都有比较可靠的保证。再加上系统的易于使用,以及针对行业的需求,在虚拟现实系统中可增加特殊的系统功能这些特点,从事虚拟现实系统的推广应用有着广阔的前景。

虚拟现实最关键的技术难点在于要在短时间内完成大数据量运算。系统中,一般要求能做到25帧的快速漫游。也就是说,要在1/25s内,根据视点和视角,计算视场范围大小,再根据视场范围,提取视场内“可见的”物体,进行三维、光照、纹理贴图、坐标转换等等一系列计算。在虚拟现实环境中漫游时,展现在我们面前的景色,都是利用计算机实时计算出来的。要达到象看电视般漫游虚拟环境,实际上就是要在40ms时间内计算出一帧图像,这40ms时间内要做的工作包括:

(1) 根据观察者所处的视点位置、高度、水平和垂直视角,计算视场范围;

(2) 在DEM数据中抽取出视场高程数据;

(3) 提取视场中所有要素的数据,如在城镇中的建筑、街道、路灯、树木、花草,等等;

(4) 所有物体均由多个三角面来表现,在光照计算前,首先计算各个顶点的法向量;

(5) 提取物体的纹理,纹理贴图;

(6) 进行光照计算;

(7) 坐标的平移、旋转;

(8) 坐标系的转换,即世界坐标系到视坐标系的转换,视坐标系到屏幕坐标系的转换;

(9) 碰撞分析、热区响应及传感器消息响应等。

由上面的论述,可现在计算机中模拟再现三维场景其工作量是相当大的,工作也是相当复杂的。如果这些都要程序员来完成的话,将耗费程序员大量的时间和努力,使程序员难以将工作集中在场景的建立上来,提高编程的效率。

好在现在的三维场景建模工具已经把这些工作完全承担下来,使程序员从这些繁杂的低效的工作中解脱出来,能集中精力到三维虚拟现实场景建立上来,而不必担心底层的工作。

3.2 开发工具的比较和选择

3.2.1 当前主流开发工具的比较

当前应用颇为广泛和成熟的开发桌面三维虚拟现实系统的工具主要VRML、Flash、Viewpoint、Cult3D、QTVR以及Java3D等技术。它们都可以不同程度的在网络上实现三维可视化，它们也有各自的特点。这些中的Java3D技术、VRML 技术、Viewpoint 技术和 Cult3D 技术主要是基于几何模型建模（Geometry based Modeling），它对硬件的计算能力和图形加速性能都有很高的要求。

基于图像建模的技术是把写实图像绘制成圆筒或者球形，有从中心点环视的方式，也有一种方式是从各个不同的位置拍摄照片，再进行综合。因为使用的是照片，所以画面非常漂亮。目前包括有 livePicture Plug-in 和 IPIX Plug-in。这些都可以提供基本的 Zoom-in/out 功能和 360 度旋转功能，使得使用者可以环顾四周。

QTVR 在前面的章节已经有过介绍，它是一种基于静态图象的技术，能够实现对一个物体或空间进行 360 度全景观察。现在有许多公司的网页上都出现了这样的全景图(Panorama)，浏览者为这种自由旋转展示的空间兴奋不已，但是一旦仔细观察，就会发现这只不过是图片另一种显示方式而已，它是将四幅或六幅图片贴在一个以视点为中心的巨大的立方体的四个或六个画面。它的优势主要在于照片级的图象质量、较小的体积和不需要任何插件就可以展示虚拟的物体，缺点是交互性不强。

VRML 是虚拟现实造型语言(Virtual Reality Modeling Language)的简称.这种技术的目的主要是为了在网页中实现三维动画效果以及基于三维对象的用户交互它是一种 ASCII 的描述语言，其来源为 Open Inventer，是 SGI 公司为其本身需求而开发出的 3D 图形描述语言。和 HTML 一样，VRML 也是可由浏览器解释的描述语言，只不过 VRML 不是描述成一个 Page 的格式，而是描述成 3D 环境和目标的布局。VRML 浏览器的主要功能是读入 VRML 代码文件，并把它解释成图形映象。和其他三维图形软件相比较，VRML 三维图形的特点是在运行时才进行着色，而普通的三维图形软件是在制作时进行着色的.虽然，使用诸如 3D Studio Max 之类的软件可以制作出效果极为丰富的三维效果，但是将这种三

维效果导出为文件之后通常是体积庞大，显然用这种方式在网页中实现三维动画是很不现实的。而 VRML 有效地解决了这个问题，其原理是在用户端提供一些基本的三维图形库，并且在网页运行时实时进行上色。这使得在网络上传输的数据量大大减少。而且，可以在形体上贴图、增加灯光效果、建立用户事件响应等。

VRML 通过节点进行三维描述，节点又由域和事件构成:域定义节点的属性;事件定义用户与场景之间的交互，使虚拟世界具有动感。在节点间创建通道(Route)，通过发送一个事件使一个节点控制另一个节点。对每一个节点的描述就构成了 wrl 文件。客户端安装插件后浏览器就可以浏览扩展名为 wrl 的 VRML 文件，并实时生成虚拟场景。国际上支持 VRML 的插件有多种多样，如:SGI 公司提供的 CosmoPlayer(这是用得最多的一种 VRML 浏览器)、微软公司 IE 浏览器自带的 VRML2. 0、日本 SONY 提供的 CommunityPlaceVRML2.0 等等。

VRML 文件是虚拟空间的文本性描述可采用文本编辑器生成，以.wrl 为扩展名，也可以由能够生成 VRML 三维空间的工具(如 3DS , AutoCAD ,Rhino 等)可视化地生成。用 VRML 构造网上三维空间具有以下四种优势:

（1）简单、方便、易用，利用 VRML 语言可以很快地开发出具有一定水准的网上虚拟现实系统。

（2）可以非常方便地生成三维几何形体.VRML 不但提供基本造型，还可通过高级造型方法(包括挤压空间造型、海拔栅格造型及点线面造型等)，创建复杂的三绅模型或者由其它可视化工具生成后输入。

（3）VRML 是一种跨平台的面向对象的网络语言 VRML 文件可以直接嵌入到 HTML 文件中去，用户只要具有 Web 和 VRML 浏览器就可以浏览 VRML 内容。

（4）VRML 利用 Script 节点可以通过 Java 或者 JavaScript 语言编写的程序脚本来扩展其功能.

VRML 语言构造虚拟场景的缺点是:画面不够生动逼真;VRML 语言功能目前还不是很强，与 Java 和 JavaScript 语言交互较难掌握等。国际著名 GIS 软件 ESRI 的 ArcVievr 3D Model 就是采用 VRML 来实现网络的三维浏览功能(Huang Bo and LinHui, 1999)。但是，由于 VRML 的一些严重的先天性缺陷:要求的数据文件太大、缺乏数据库接口、效率很低、不支持三维编辑等，目前没有形体之间的碰撞检查功能，其在 GIS 领域特别是数码城市的进一步应用受到限制。

Cult3D 是瑞典 Cycore 公司推出的一种应用于主流操作系统和应用程序的交互三维渲染软件，它主要用于在网页上建立互动的 3D 实体，具有一个跨平台的 3D 引擎。由于使用 Cult3D 构造的三维物体的文件尺寸非常小(一般在 lOK 到 400K 之间)，Cult3D 渲染速度也确实很快，并且渲染效果非常出色，所以成为网络上三维产品展示的最佳方案。使用 Cult3D 技术，用户可以在线浏览、观察可交互的三维产品模型，同时 Cult3D 文件可以应用于网页、Office 文档、Acrobat 文档等。用户通过鼠标单击，即可以翻转、缩放和平移 Cult3D 模型，从任何角度观察它，单击 Cult3D 对象中设置的交互区域可以开启或者关闭模型的部件或者播放音乐、语音解说等。

特别 Real time 3D particle 生成或者使用 Environment Mapping 技术产生的反射效果，已经达到了可以挑战 Web3D 的表现极限的水平，能够提供当今技术可以达到的最完美的画面。因为这一优点，Cult3D 非常适合博物馆站点或者宣传产品的电子商务网站，同时，它也支持,Java，也具有更多样化的扩展性，也就适用于游戏或者动画制作的开发。

Cult3D 使用的是 Java 技术，为其跨平台的支持所有的主流浏览器产品打下了良好的基础。除了 Windows 用户外，像 Linux, Macintosh, Solaris, BeOS 用户也能够享受到 Cult3D 技术。已经有很多著名的公司在网站中使用了 Cult3D 的产品，如:Ericsson, Nikon, Nokia, Toyota, Yamaha 等。

此外，其他的技术例如 Pulse3D, Sev, 3DML 等，它们因为都是使用开发公司自身的技术，所以需要 200-800KB 的各自不同的 Plue-in。

Pulse Entertainment 公司的 Pulse3D 通过只能在 Cult3D 中使用的 ReflectionMapping 可以实现完全的反射效果，通过 HTTP-Streamed 方式的 Audio 以及动画制作文件，可以实现丰富多彩的三维世界。它可以提供各种开发制作工具中的自身手写器编辑功能，以及只支持在专业三维图形工具中使用的 Inverse Kinematics 功能，该样就可以实现充满幻想的各种动作了。

Sev （Superscape e-Visualizer）是英国 Superscape 公司以 Web3D 标准开发任的一项技术，目前还处在一个初级阶段。此外还有 3DML，作为互联网上描绘名为 Spot 的三维空间的技术，它是与 HTML 类似的一种动画制作语言。不过，虽然它与 HTML 有很多类似的部分，但实际上，与 HTML 相比，它是配合 XTML 规格开发出来的。在 HTML 中，包括其 EMBED 的形态，HTML 和 DHTML 都可以灵活使用 Java 和 XML。

Java3D 综合了上述优点，它除具有 VRML 具有的可交互性、支持多媒体和节省网络带宽外，还具有简化三维应用程序的开发、更强的交互性。Java3D 是用于开发三维图形的 API，他从高层次为开发者提供对三维实体的创建、操纵和着色，使开发工作变得较为简单。同时，Java3D 的低级 API 是依赖于现有的三维图形系统的，这种体系结构既可以使其开发的程序“到处运行”，又使其充分利用系统的三维特性。用 Java3D 开发的基于 Web 的小程序可以与 Internet 很好的集成，在浏览器上观赏或对虚拟现实场景进行交互式操作时不需要下载相关的插件。鉴于 Java3D 在网络三维方面优异的表现，我们选用 Java3D 作为实现网络三维的工具。

3.2.2 开发工具选择小结

从上面列举的多种当前广泛应用的，具有代表性的开发桌面虚拟现实三维可视系统的工具可以看出，QTVR 只是简单的全景图的拼接，虽然可以以较小的数据量来达到较好的可视效果，但是它的交互只能在全景图的内部，其可视范围受到了很大的局限，且不具有高层次的交互性。Direct3D 开发工具适于对底层进行开发，可以实现很好的三维效果，但是由于是 MicroSoft 公司开发的工具，所以针对 Windows 系统工作良好，这从另一方面限制了该工具的跨平台性。这在当前网络日益普及，平台种类众多的今天显然不太适合网络三维可视化桌面虚拟现实系统的开发。Cult3D 本身也使用的是 Java 技术，它也具有很好的跨平台性和交互性，但是显然不如 Java3D 本身对 Java 体系的兼容性好。VRML 语言虽然在网页的嵌入中有良好的发挥，当前也应用颇为广泛，但是严格讲来 VRML 只是一种图形格式，用纯粹的 VRML 开发工具开发出来的产品难以达到希望的完全的功能。比如 VRML 就没有碰撞检测的功能。换句话讲，所有的 VRML 程序都可以通过 Java3D 改写实现，但是复杂一些的 Java3D 程序仅用 VRML 本身就无能为力了。OpenGL 是底层的图形开发库，具有优良的特性，但是程序编写复杂，难以掌握。如果能够有一种工具能以 OpenGL 在底层工作，以体现 OpenGL 本身的优良性能特性，而又将对该工具的调用对程序开发者透明化，而只需要用相对简单的语法就可以实现代码目标，这将是我们所希望的。Java3D 就是如此。Java3D 本身以 OpenGL 为底层图形实现工具，却将相关细节屏蔽起来，对 Java3D 程序员而言，甚至对普通程序开发者而言，用 Java3D 开发所需的三维可视化桌面虚拟现实系统只需要掌握相对简单的多的语言规范，并且能够方便的

与当前应用相当广泛的 Java 语言体系相兼容，甚至调用 EJB 模块，就可以实现效果优良的，功能复杂的，兼容性强的，扩展性好的，跨平台的可视化代码。现在已经有大批的公司宣布使用 Java3D 图形技术，如 Teneo 计算公司、WebScope 公司、在线数字设计工作室 Improv Technologies 公司、Infobyte 公司采用 Java3D 技术对埃及 Nefertari 女王的古代坟墓进行虚拟游览等等。Internet Explorer、Netscape 等网络浏览器也完全支持 Java3D 图形显示技术。甚至现在已经有 Nokia、SonyEricsson 等手机厂商生产出支持 Java3D 图形的手机产品，并且预装了《三界传说》等三维无线社区游戏，这不仅证实 Java3D 可以在有线及无线三维图形领域的成熟应用，也展示了 Java3D 技术的魅力和广泛的应用前景。

也正是基于上面所论述的 Java3D 的技术优势，本论文选取 Java3D 为工具开发适合网络应用的简单桌面三维虚拟现实系统 。

3.3 Java3D 语言概述

3.3.1 Java 简述

十年前也许还没有多少人知道 Java，但是时至今日，如果还有谁不知道 Java 的话，也许他就会被认为是 IT 业或软件业不称职的人士了。这十年来，是 Java 经历了从诞生到发展，到无所不能，到席卷全球，到在网络上风起云涌的十年。业内人士普遍认为“Java 的诞生是八十年代以来计算机界的一件大事”，甚至全球首富，微软总裁比尔·盖茨在悄悄地观察了一段时间后，都不无感慨地说：“Java 是长时间以来最卓越的程序设计语言”，他还将微软整个软件开发的战略从 PC 单机时代向着以网络为中心的计算时代转移，而购买 Java 则是实施他的重大战略决策的具体行动。为什么 Java 会赢得专业开发者如此热烈的拥护呢？

首先、Java 很像 C++，很容易为有经验的程序员掌握使用，但它比 C++有很大的改进。Java 更容易编写可移植、可重用、无错误的程序；

其次，Java 的交叉平台（cross-platform）的兼容性是它迅速获得成功的主要因素。Java 编译器（windows，MacOS 及 Unix 平台上都有）把 Java 源代码转换成字节代码的类文件。这个类文件与其它语言编译器生成的可执行二进制文件完全对应，但不像一般的本机（native）二进制文件，Java 的字节代码不是针对特定的微处理器体系结构，它的“本机”体系结构为以软件形式存在的 Java VM（Java 虚拟机）。这样，Java 类文件对拥有 Java 运行环境的任何硬件平台来说都

是可移植的。Java 运行环境包括：Java VM，Java 标准类库、用于安全的 Java 字节代码验证器及字节代码解释器。这种解释器运行 VM 上的类文件，无需程序员重写源代码，甚至无需对源代码重编译。Java 的“编写一次，可到处运行”的广泛性，推动了不少公司开发大量的可在任何机器上运行的 Java 开发工具。Ignite Technologies 公司开发的“可视化 GUI 构造器”工具就是一个典型的例子。这样，同一个项目可用同样的工具，但可在任何平台上运行，这为应用项目的可移植性，开发工具的可移植性，以及程序员的可移植性创造了条件。有人预言：Java 将是网络上的“世界语”，今后所有的用其它语言编写的软件统统都会用 Java 语言来改写。

3.3.2 面向对象的 Java

和其它编程语言相比，Java 有以下四大特性：

- 面向对象

Java 语言的第一个特点就是它是一套面向对象的编程语言。如果以比较严谨的态度来看面向对象的特性的话，那么一套真正的面向对象的工具至少应该包括以下四个特点：

（1）封装性（Encapsulation），必须有模块化的性质以及信息隐藏的能力。

（2）多态性（Polymorphism），不同的对象对同一种信息，可以按照对象本身的性质加以回应。

（3）继承性（Inheritance），可以定义一套对象之间的层次关系，下层的对象继承了上层对象的特性，藉此可以实现程序代码的重复使用，并且有效地组织整个程序。

（4）动态联编（Dynamic binding）一个对象一旦生成以后，要使用这个对象只需要简单地把信息传递给它，不再需要去参考对象当初设计时的规格。只有在程序执行时，才会真正锁定需要的对象，这样的方式可以使程序设计具有最大的灵活性。

事实上，如果用以上四个特点去衡量现有的一些编程语言或工具，有很多都不能算是完全的面向对象。如 Visual Basic 缺乏数据的封装性，而 C++并没有办法做到动态联编。只有 Java 在这四点上都可以做的很好，也就是说，只有 Java 是真正的、纯粹的面向对象的编程工具。

- 跨平台

如果要让我们的程序可以在各种不同的机器及操作系统上执行，那么首先在编写程序源代码时，必须让自己的程序不使用编程语言中由编译器或平台本身定义的功能，譬如说在 C 语言中的 int 整数类型并没有实际定义它的长度是二个字节或四个字节，必须由在那种机器或操作系统下执行才能决定。因此，编程语言严格的定义是确保程序可以在各种平台上工作的第一步。在 Java 的语言定义中，我们看不到任何取决于工作平台或编译器的功能或特性。

另外一个最大的问题是，各个机器都有不同的低级汇编语言，在这个机器上编译好的机器码，肯定不能直接拿到其它不同种的机器上使用，即使在同一种机器的机器码，由于程序的执行过程和操作系统信息相关，只要是不同的操作系统，程序编译好的机器码往往也有很大的差异，而不能互用，而必须重做一次编译工作。为了解决这一难题，Java 的策略是采取半编译、半解释的方式，定义出 Java 自己的虚拟机以及这套虚拟机上所使用的机器码—Java Bytecode,从而解决了 Java 与操作平台的无关性。

Java 程序执行的流程如图 1 所示。一个编写好的 Java 程序源代码，先通过 Java 编译器编译，产生出 Java 虚拟机的代码 Bytecode，再经过 Java 解释器将 Bytecode 转换成实际使用的机器和操作系统上的机器码去执行。

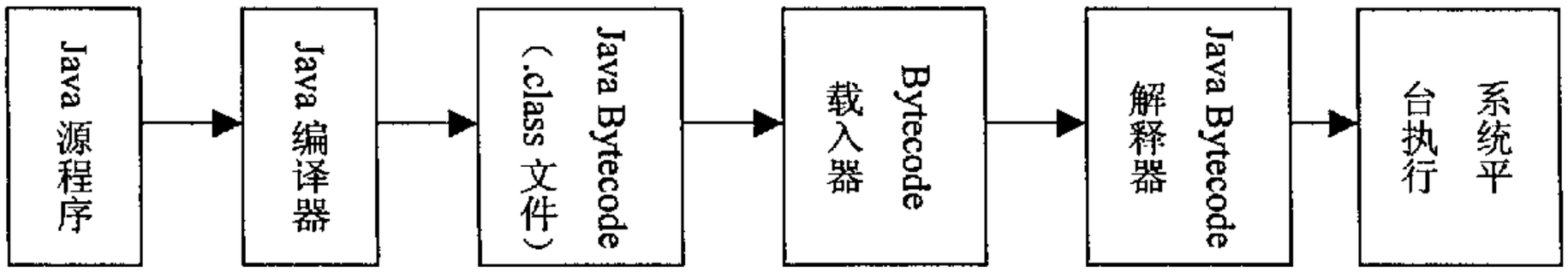


图 3—1　Java 程序执行的流程

如此一来，只要实际使用的操作平台上有 Java 解释器，这个操作平台就可以执行各式各样的 Java 程序。而对于程序开发者而言，不需要担心将来程序在执行时，所使用的操作系统和机器是什么，因为只要完成一次编译工作，做出一份 Bytecode，各种机器的 Java 解释器都可以执行它。

- 多线程

随着个人计算机采用的微处理器的飞速发展，个人计算机上的操作系统也纷纷采用多任务和分时设计，将一早期只有大型计算机才具有的系统特性，带给了个人计算机。一般在多任务或分时操作系统中，都有所谓进程（Process）

概念。简单地说，一个进程就是一个执行中的程序，而每一个进程都有自己独立的一块内存空间和一组系统资源。譬如说，每一个在 Windows 95 中正在执行的程序，都可以视作是一个进程。

所谓线程（Thread），其实也是一个执行的程序，但是与进程不同的是，多个线程是共享一块内存空间和一组系统资源，而线程本身的数据通常只有微处理器的寄存器数据和一个供程序执行时使用的堆栈。所以系统在产生一个线程或者在各个线程之间进行切换时，负担要比进程小得多，也正因为如此，线程被称为是轻负荷进程。

Java 语言提供了多线程的功能。在 Java 基本函数库中定义了 Thread 这个基本类，内置了组方法，使程序设计者编写多线程程序时，只要继承这个类，就可以利用原编写好的方法，生成一个新的线程、执行一个线程、终止一个线程的任务。或者查看执行状态。

由于目前网络上传输文件的速度受到相当大的限制，如果在 Java 程序中适当运用线程的功能，可以让使用程序的人等待的时间缩短。譬如说，在播放一个动画文件时，以多线程的方式同时传送每一幅图片，或者在网络获取数据时，用另外一个线程放音乐给用户听。

- 安全机制

因为 Java 面向对象的封装特性和严格的类的继承特性也使得病毒代码在被运行之前被虚拟机（JVM）装载时检查出来从而避免了机器被病毒的破坏。另一方面， Java 语言不是像 C 语言一样直接运行在机器的硬件平台上，所以就比 C 语言有更好的安全性。换句话说，当 C 语言的程序中有恶意代码时，它直接运行在硬件平台上就有可能直接将硬件物理特性破坏掉，而 Java 语言运行在 Java 虚拟机上，即使病毒代码侥幸没有被类检查机制检查出来而进入运行期，最坏的情况就是 JVM 系统被破坏，只要重新安装 JVM 就可以继续运行 Java 程序，而避免了机器物理硬件的损伤。

由上面的论述可以看到， Java 语言是一种纯粹的面向对象的语言，其代码只能以封装成类或对象的形式存在、编译和运行。运行在 Java 虚拟机 JVM 上的已经被编译成二进制字节码的.class 文件的运行机制显然和直接运行在系统物理硬件上的 C 系的语言的运行机制不一样，而是有一个加载各种类和包的过程。事实上，Java 语言被编译成字节码后在 JVM 上运行的过程如下图如示：

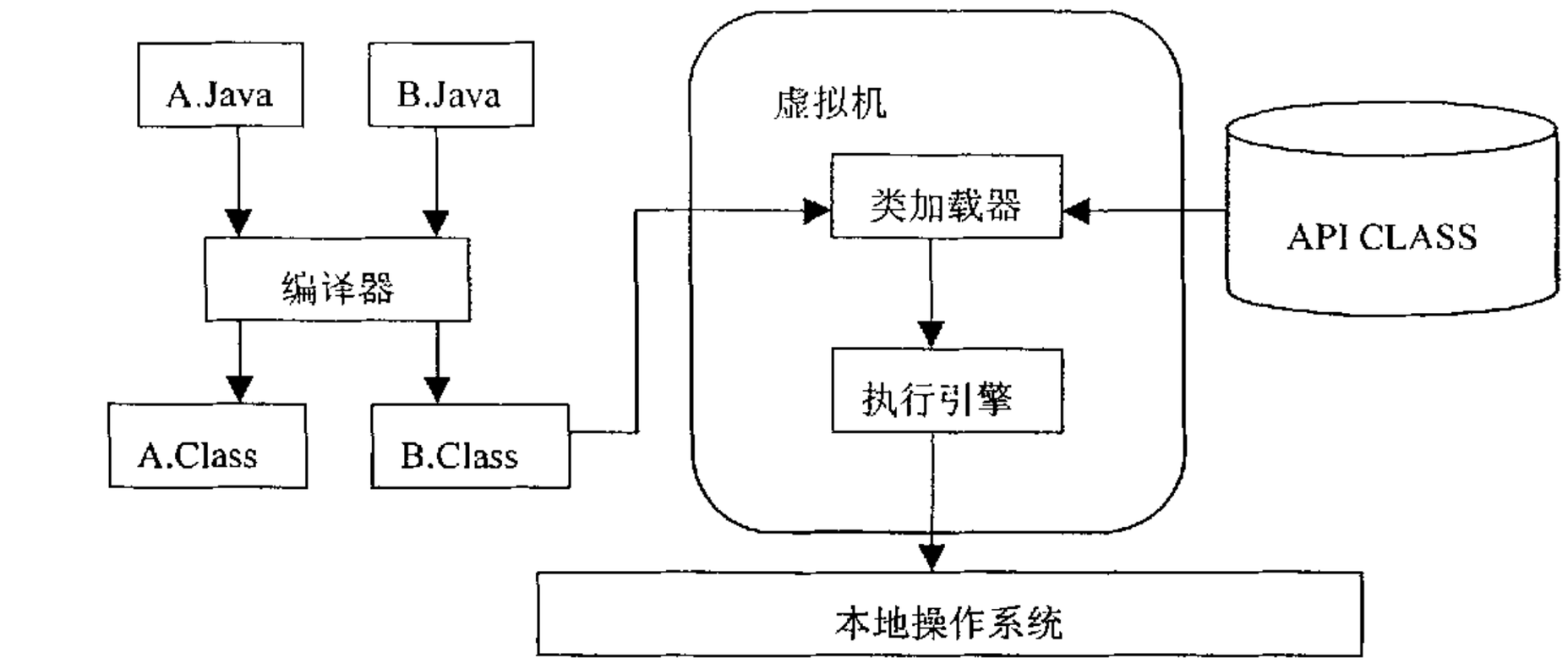


图 3—2 Java 代码编译运行过程

由上图可知，运行 Java 程序必须在操作系统上安装 Java 运行环境，也就是 Java 虚拟机 JVM。JVM 中最重要的两个部分就是类加载器和执行引擎。类加载器的功能是加载编译程序员编写的 Java 代码所生成的.class 字节文件和相应需要调用的 API 类。这些类都是已经被封装的类和对象。类加载器将这些类或对象载入 JVM 后通过执行引擎运行，由执行引擎来处理和操作系统的沟通工作。

3.4 Java3D 技术

在 Java 语言体系发展的如火如荼的当今，也正是三维图形应用甚至是网络应用日益广泛的时候。虽然用 Java 语言本身也可以编写三维场景，但是始终没有专业编程语言那样方便，如果想再现逼真的场景效果就更难以简单的实现。

当然，人们也提出了很多单机或网络上解决三维可视化代码编写的方案，有一些当下正在广泛应用当中，如将现在与网页结合很好的虚拟现实建模语言 VRML 和 Java 结合起来编写适于网络应用的三维可视化软件等等。显然程序员在编写这样的三维应用时，为了使二种编程语言能很好的结合，也不得不牺牲各自的优势，使得应用无法体现出编程语言的优良特性。人们也从没有放弃过对更好的方法的研究和尝试。

在这样的情况下，SUN 公司于 1997 年推出新开发的专门应用于三维图形应用领域的 Java3D 开发包。新出现的虚拟现实建模语言 Java3D 综合了传统建模

语言诸如 OpenGL、VRML、Direct3d 等的优点，它是以 Java1.2 的一个标准扩展，是 Java 语言中一个用于开发三维图形的 API。它从高层次为开发者提供对三维实体的创建、操纵和着色，使开发工作变得较为简单。同时，Java3D 的低级 API 是依赖于现有的三维图形系统的，如 Direct3d、OpenGL 等。Java3D 的这种体系结构既可以使其开发的程序“到处运行”，又使其能充分的利用系统的三维特性。用 Java3D 开发的基于 Web 的小程序可以与 Internet 很好的集成，在浏览器上观赏或者对虚拟现实场景进行交互式操作时不需要下载相关的插件，因为 Java3D 在虚拟现实建模中的应用有很大的便利和通用性。

Java3D是Java1.2的一个标准扩展，它从高层次为开发者提供对三维实体的创建、操纵和着色，使开发工作变得较为简单Java3D的低级API依赖于现有三维图形系统，如Direct3D、OpenGL等，它为我们编写三维应用程序提供了一个非常完善的API，其功能主要有[9]：

（1）生成简单或复杂的形体（也可以直接调用现有的三维形体）；

（2）使形体具有颜色、透明效果、贴图；

（3）在三维环境中生成灯光、移动灯光；

（4）具有行为的处理判断能力（键盘、鼠标、定时等）；

（5）可以生成雾、背景、声音等；

（6）使形体变形、移动、生成三维动画；

（7）编写非常复杂的应用程序，用于各种领域如虚拟现实系统。

三维图形技术的鼻祖是 SGI 公司推出的 OpenGL 三维图形库，Java3D 则是在 OpenGL 基础上发展起来的，因而 Java3D 的数据结构也和 OpenGL 一样，采用的是场景图的数据结构；但是 Java3D 的场景图根据 Java 语言编程的特点，增加了一些新的内容，更易于实时处理及特殊的三维效果的显示，更加方便最新的三维图形加速技术的应用。Java3D 的场景图是 DAG，具有方向性的不对称图形。Java3D 的场景图是由 Java3D 运行环境直接转换成具有三维显示效果的显示内存数据，从而在计算机上显示出三维效果。显示内存中的数据不断接收 Java3D 运行产生的最新结果，直接显示出来，从而产生三维动画效果。

Java3D 的场景图中有很多线和线的交汇点。交汇点称为节点，不管什么节点，都是 Java3D 类的实例，即面向对象编程方式所生成的对象。线则表示各对象之间的关系。编写 Java3D 应用程序与画一棵大树很类似：先用准备好的笔画出各个交汇点；然后用各种颜色的彩笔画出所需要的树干和树枝。画交汇点就

是定义好所需要的对象，如显示的是什么形体，有什么颜色，形体按照什么样的运动方式运动；画树干和树枝就是给出各对象这间的关系，例如谁是父节点，谁是子节点。Java3D 的场景图中，最底层的节点是 Virtual Universe。每一个场景图只能有一个 Virtual Universe，好似一棵大树的树根一样。

Java3D 程序只要正确地定义了三维图形的具体观察位置及观察参数，建立了一个带有三维及其属性的一个场景图，就可以在计算机屏幕上显示程序的运行效果了。

3.4.1 Java3D 的场景图数据结构

Java3D 的数据结构采用 DAG（Directed-Acyclic Graph）式的场景图（Scene Graph）,即具有方向性的不对称图形。图中线和线的交汇点称节点（Node），这

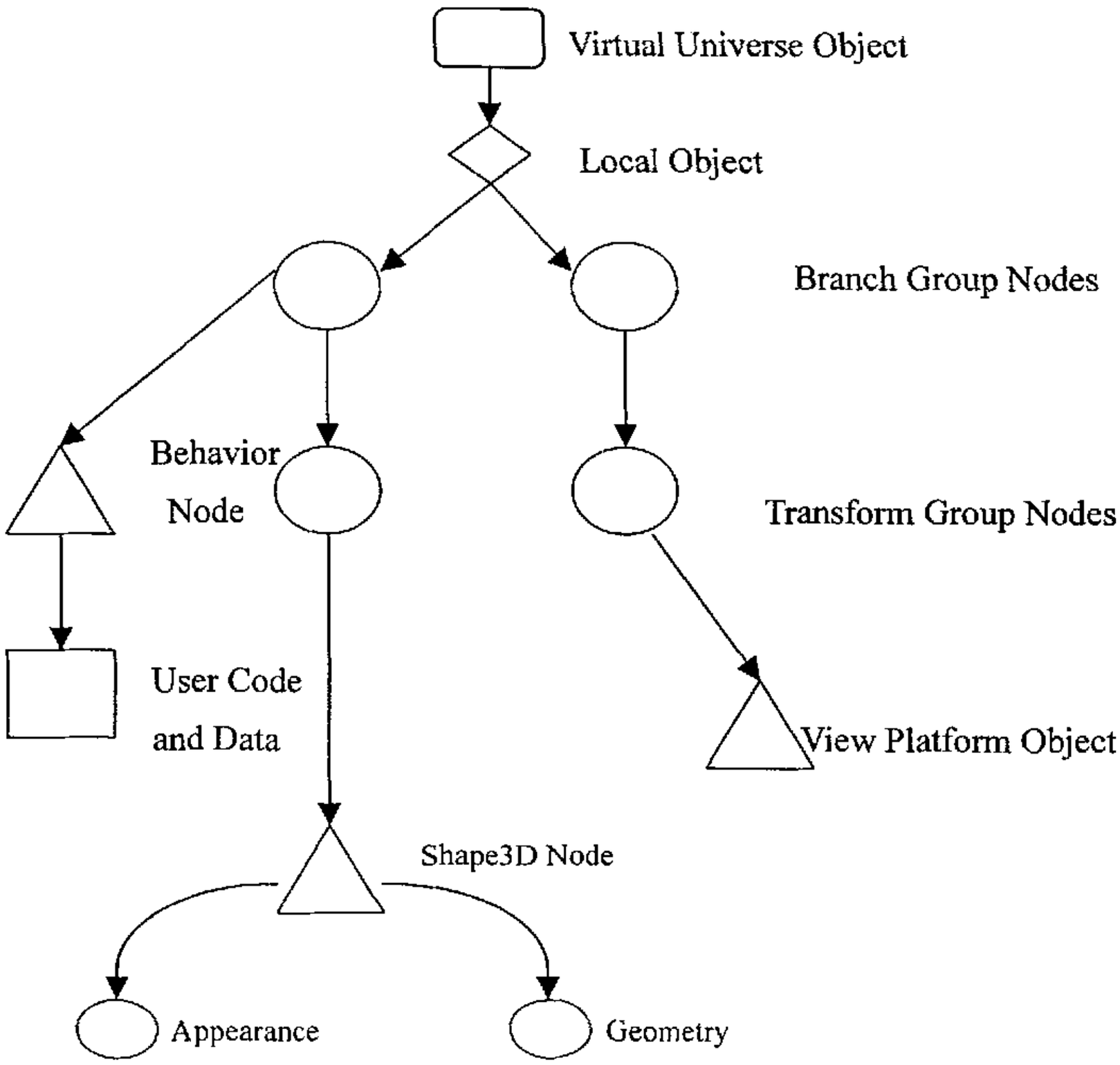


图 3－3　Java3D 场景结构图

些节点都是 Java3D 类的实例；线（Arc）表示实例之间的关系。最底层的节点是 Virtual Universe，每个场景图只能有一个 Virtual Universe，Virtual Universe 的上面是 Local 节点，每个程序可以有一个或多个 Local，但同时只能有一个 Local 处于显示状态，绝大数程序只有一个 Local，复杂的 Java3D 应用程序可以建立多个 Local，由程序控制当前的 Local 是哪一个。第一个 Local 上面拥有一个到多个 BranchGroup 节点。一般来说，要想建立三维应用环境，必须定义所需要的形体，给出形体的外观及几何信息，再把它们摆放在 BranchGroup 节点上面。摆放的具体位置由节点 TransformGroup 节点定义。另外，摆放好三维形体后，还需要设定具体的观察位置 View Platform 及观察参数和视角。完成了这些工作，就算完成了三维场景图的建立。如下图给出了 Java3D 应用程序的场景图。

3.4.2 Java3D API 中的类以及类的关系

Java3D 核心包包括 javax.media.j3d 和 javax.vecmath，其结构层次如下面图所示。其中，javax.media.j3d 提供了 100 多个类及接口，是 Java3D 的核心部分；javax.vecmath 则包括了一些矩阵和数组运算的类。Java3D 还提供了一个重要的有助于快速编程的应用类型的包，即 com.sun.j3d.utils 包，Utility 不是 Java3D 编译环境的核心组成部分，可以不用它，但使用它会大大提高程序的编写效率。一些基本的形体如立方体，圆柱体等，可以由 Utility 方便的生成；对复杂形体的生成，则需对基本形体进行一系列的几何坐标变换来实现。

Javax.media.j3d

Virtual Universe

Local

View

Physicalbody

PhysicalEnviroment

Screen3D

Canvus3D

SceneGraphObject

Node

Group

Leaf

NodeComponent

Transform3D

Javax.vecmath

Matrix Classes

Java3D 所提供的类，根据其作用主要有两种类型：Node 和 NodeComponent.

（1）Node 类 含有 Group 及 Leaf 两个子类。Group 类用于将形体等按一定的方式组合在一起。Leaf 类如 Light、Sound、Background、Shape3d、Appearance、Texture 及其属性等内容，还有 ViewPlatform、Sensor、Behavior、Morph、Link 等，类似于 VRML 语言的相应节点，是 Java3D 场景图的重要组成部分。

（2）NodeComponenet 类 用于表示 Node 的属性，它不是 Java3D 场景图的组成部分，而是被场景图所引用，用来修饰某些 Leaf 对象，如某个颜色可以被多个形体引用。Bounds 及其子类、Transform3D 并不是 NodeComponenet 的子类，但它们同样作用于 Leaf 对象，因而也是 NodeComponent 类型的对象。

3.4.3 Java3D 中形体的生成

对于 Java3D 编程，三维形体是最重要的处理对象。可以有三种方式生成 Java3D 所需要的三维形体，一是利用 Java3D 提供的用于编写点、线、面对象，二是使用 Utility 里面的可用来生成几个基本形体的 geometry classes，三是通过调用外部其它格式的三维图形文件，生成复杂的三维形体对象。

Java3D API 中，点的生成有两种方式：PointArray 和 IndexedPointArray。前者用于生成给定数目的点，后者可以在多个顶点中选择部分或全部顶点，并以给定的属性显示出来。

Java3D API 中，线的生成有四种方式：LineArray、LineStripArray、IndexedLineArray 和 IndexedLineStripArray。第一种方法按给定顶点顺序连成线，第二种按照顶点数组中的顺序按顺序连成折线段，第三种方法在给定点组中挑选点生成线段，最后一种方法在给定点中选取点生成折线段。

Java3D API 中还有很多生成三角形面和四边形面的方法。这些方法种类繁多，但主要思路是按照给定的顶点数组或者索引数组中的点生成面。

Java3D 的核心类中没有基本形体类，所以没有办法生成经常使用的一些基本形体。可以通过复杂的编程生成这些基本形体。另外，Java3D 的外部类 Utility 中的 geometry classes 中的几个基本形体 Box、Cone、Sphere 和 Cylinder 等可以

帮助生成不复杂的基本形体。

3.4.4 形体的组合和组的生成

Java3D 应用程序中，一般都拥有多个三维形体，需要对其进行组合，这样才能对指定的形体进行几何变换，对指定的形体进行操作，生成所需要的交互式三维应用程序或三维动画。组节点一般是 Group 或者其子孙类。

Group 对象只能有一个父类，对于 BranchGroup 而言是 Locale，而其它的 Group 类型的对象则可以将其它 Group 类型的对象作为其父类。Group 类型的对象可以加载有任意多个 Children。

Group 的构造方法为 Group（）。它有以下这些 Flags：ALLOW_CHILDREN_READ；ALLOW_CHILDREN_WRITE 和 ALLOW_CHILDREN_EXTEND。这些 Flags 通过 setCapability 方法的设置，能够允许 Group 对象在程序运行时增加、减少、插入所包含的内容。

3.4.5 Java3D 的视模型

Java3D 视模型是通过将虚拟环境和物质环境完全独立的方式来实现上述功能的，且该视模型可将虚拟环境中视平台的位置、方向和大小与 Java3D 绘制的与视平台位置、方向相一致的虚拟场景相区分。一般应用程序控制视平台的位置和方向，而绘制着色系统则依据终端用户的物质环境以及用户在物质环境中的位置和方向来确定显示场景[11][12]。

Java3D 视模型由虚拟环境和物质环境两部分组成，其虚拟环境由 View Platform 对象来表示，它是虚拟对象存在的空间；而物质环境则由 View 对象以及和它相关的对象来表示。在这其中，View 对象和它的相关对象就描述了用户所处的显示的操纵的输入设备环境。虽然视模型将虚拟环境和物质环境相互独立，但可通过一一对应关系来建立两种世界之间相互通信的桥梁，这样将使得终端用户的行为影响虚拟环境中的对象，同时虚拟环境中的对象行为也会影响终端用户的视点。

Java3D 可通过几个对象来定义视模型参数。这些对象包括 View 对象及其相关对象、PhysicalBody 对象、Canvas3D 对象、Physical Environment 对象、Screen3D 对象。视模型相关的对象其作用如下：

ViewPlatform 用来标志场景图中视点位置的节点。其父节点则指明了视平台在虚拟环境中的位置、方向和大小。若是通过修改与 Transform Group 节点相关

的 Transform3D 对象，就可以在虚拟场景中随意移动视平台。这样就可以方便的在虚拟环境中漫游。

View 用于指定需要处理场景图的信息。

Canvas3D 定义了 Java3D 绘制图象的窗口，它提供了 Canvas3D 在 Screen3D 对象中的大小、形状和位置信息。

Screen3D 用于描述显示屏幕的物理属性。

Physical Body 用于封装那些与物质体相关的参数(如左、右眼的位置等)。

Physical Environment 用于封装那些与物质体环境相关的参数(如，用于头状物体或头盔式跟踪器的校验信息)。

所以，视模型就可以通过这样的程序框架来构建：

1. 创建一个 TransformGroup 来控制 platform

```
TransformGroup viewGroup = new TransformGroup();
ViewGroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRI
TE);
```

2. 加入一个 ViewPlatform

```
ViewPlatform myPlatform = new ViewPlatform();
myPlatform.setActivationRadius(1000.0f);
myPlatform.setViewAttachPolicy(View.NOMINAL_HEAD);
viewGroup.addChild(myPlatform);
```

3. 将它们加入到 BranchGroup 的 View 分支

```
BranchGroup viewBranch = new BranchGroup();
viewBranch.addChild(viewGroup);
myLocale.addbranchGraph(viewBranch);
```

4. 创建一个具有默认配置的 Canvas3D(自动建立了一个 Screen3D)

```
Canvas3D myCanvas = new Canvas3D(null);
```

5. 建立一个视对象并且给他赋予 Canvas3D

```
View myView = new View();
myView.setCanvas3D(myCanvas);
```

6. 在视对象上附加上视平台

```
myView.attachViewPlatform(myPlatform);
```

7. 使用默认的 physical body，physical environment

这样就可以建立一组视平台和视点，此后桌面虚拟现实系统工作时就可以自动渲染可见物体，并且在其它相关参数的辅助下接受相应输入设备的输入信息实现三维浏览漫游。

第 4 章　LOD 算法的分析和改进

4.1 LOD 算法的引出

就目前计算机图形学水平而言，只要有足够的计算时间，就能生成准确的像照片一样的计算机图像。但虚拟现实系统要求的是实时图形生成，由于时间的限制，使我们不得不降低虚拟环境的几何复杂度和图像质量，或采用其它技术（如纹理映射）来提高虚拟环境的逼真程度。

所谓实时显示，是指当用户的视点变化时，图形显示速度必须跟上视点的改变速度，否则就会产生迟滞现象，要消除迟滞现象，计算机每秒钟必须生成 10 帧到 20 帧图像（而现在的视频技术实际要求是每秒 25～30 帧图像），当场景很简单时，例如仅有几百个多边形，要实现实时显示并不困难，但是，为了得到逼真的显示效果，场景中往往有上万个多边形，有时多达几百万个多边形。此外，系统往往还要对场景进行光照处理、反混淆处理及纹理处理等等，这就对实时显示提出了很高的要求。就图形学发展而言，起关键作用的无疑是图形硬件加速器的发展。高性能的图形工作站和高度并行的图形处理硬件与软件体系结构是实现图形实时生成的一个重要途径。然而应用模型的复杂程度往往超过当前图形工作站的实时处理能力，考虑到虚拟现实系统对场景复杂度几乎无限制的要求，在虚拟现实系统高质量图形的实时生成要求下，如何从软件着手，减少图形画面的复杂度，已成为虚拟现实系统中图形生成的主要目标。

要提高图形显示速度，一个实践证明非常有效的方法是降低场景的复杂度，即降低图形系统需处理的多边形数目。目前，比较常用的方法有这么几种：

预测计算：根据各种物体运动的速度和加速度，用预测或者外推法在下一帧画面绘制之前估算出该物体的下一步出现的位置即预测输入设备的输入，从面减少由输入设备所带来的延迟。

脱机运算：由于虚拟现实系统是一个多任务的模拟系统，所以可以尽可能的将一些可预先计算好的结果事先计算好并存储到相应的结构中。

场景分块：将一个复杂的场景分成若干个子场景，各子场景之间几乎不可见或完全不可见。这样在显示一个子场景的时候就不必做繁杂的运算去处理不

必要的数据。但是这种方法在封闭的空间内使用有效，对开放空间则难以使用。

细节层次模型：即使采用了场景分块技术及可见消隐技术，有时用户能“看见”的场景仍很复杂，为此细节层次(Level of Details，简称 LOD)模型方法应运而生。所谓 LOD 模型方法，即为每个物体建立多个相似的模型，不同模型对物体的细节描述不同，对物体细节的描述越精确，模型也越复杂。根据物体在屏幕上所占区域大小及用户视点等因素，为各物体选择不同的 LOD 模型，从而减少需要显示的多边形数目。这是一种很有效的方法，然而，这种方法对场景模型的描述及维护提出了较高的要求。

当然还有一些其它方法。但是前面几种方法仅适用于某些特殊的情况，而 LOD 模型方法则具有普适性。现在，LOD 模型的自动生成和绘制技术已成为一个很有前途的研究方向，受到了全世界范围内相关研究人员的高度重视。

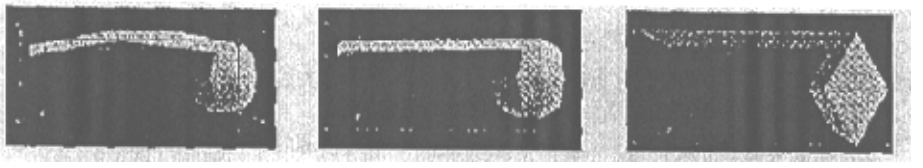
4.2 LOD 技术的传统思路和不足

当画面中含有大量可见面而需要考虑对场景的简化时，如果许多可见面在屏幕上的投影小于一个像素，就可以合并这些可见面而不损失画面的视觉效果。细节层次 LOD 技术就是顺应这一要求而发展起来的一种快速绘制技术。LOD 技术在不影响画面视觉效果的前提条件下，通过逐次简化景物的表面细节来减少场景的几何复杂性，从而提高绘制算法的效率。该技术通常对每一原始多面体模型建立几个不同逼近精度的几何模型。与原模型相比，每个模型均保留了一定层次的细节。当从近处观察物体时，采用精细模型，而当从远处观察物体时则采用较为粗糙的模型。同时，在两个相邻层次模型之间形成光滑的视觉过渡，即几何形状过渡，以避免视点连续地变化时两个不同层次的模型间产生明显的跳跃。这样，计算机在生成场景时，根据该物体所在位置与视点间的远近关系不同，分别使用不同精细程度的模型，避免了不必要的计算，既能节约时间又不会降低场景的逼真度，使计算的效率大大提高。LOD 的研究主要集中在建立不同层次细节的模型和建立相邻层次多边形网络之间的形状过渡两个方面。

综上所述，LOD 方法的基本思想是：对场景中的不同物体或物体的不同部分，采用不同的细节描述方法，在绘制时，如果一个物体离视点比较远，或者这个物体比较小，就可以用较粗的 LOD 模型绘制。反之，如果一个物体离视点比较近，或者物体比较大，就必须用较精细的 LOD 模型来绘制。同样，如果场

景中有运动的物体，也可以采用类似的方法，对处于运动速度快和处于运动中的物体，采用较粗的 LOD,而对于静止的物体，采用较细的 LOD。

例如对一个门把手建立不同精度的模型（如下图所示）。每个图下所标识的是相应建模的面的个数。由图可以看出：最右边是只有 7 个面的门把手模型，相当简单，这样的模型在显示时显然是不用经过大量运算的，也就降低了运算复杂度；在相应用的精度要求下，用到了中间的模型，该模型有 27 个面，这样的门把手已经比只有 7 个面的模型清楚很多，已经能够大概分出这是一个门把手；最高精度的就是最左边那个门把手模型，这个模型共有 52 个面，这样的门把手当然最精细，但是该模型对系统的运算要求也是最高的[10]。



（a）52　　（b）27　　（c）7个

图 4—1 三个不同精度的门把手模型

于是根据应用，可以对该门把手建立三个级别的 LOD 模型（高、中、低），即当视点和门把手距离 5 米以上时，渲染调用低级别的 LOD 模型，只有门把手轮廓，其它细节都不可见，大大降低了系统运算量；当视点和门把手距离 2～5 米时，调用中级别的 LOD 模型，这时门把手上的部分细节可见，可以进一步的满足视觉要求；而当视点和门把手距离在 2 米以内时，才调用运算量最大的高级别 LOD 模型，此时门把手上的各个棱、角等细节均清晰可见，可以达到近距离观察的要求。

上例是一个简单的 LOD 实例。该方法对一个模型建立了三个级别的 LOD 模型，以 2 米和 5 米作为切换的分界线。这样就有一个图像跳变的问题。所以解决这个问题的方法就是对模型建立更多的 LOD 模型，使得每个模型之间的差别缩小，从而减小分界线处的图像的跳变感。下图显示了视点与可视物体距离和相应被调用的 LOD 模型之间的关系。

由该图可知，当视点和物体之间的距离小于 L1 时，调用 LOD1 模型，在 L1 和 L2 之间时调用 LOD2 模型，以此类推。

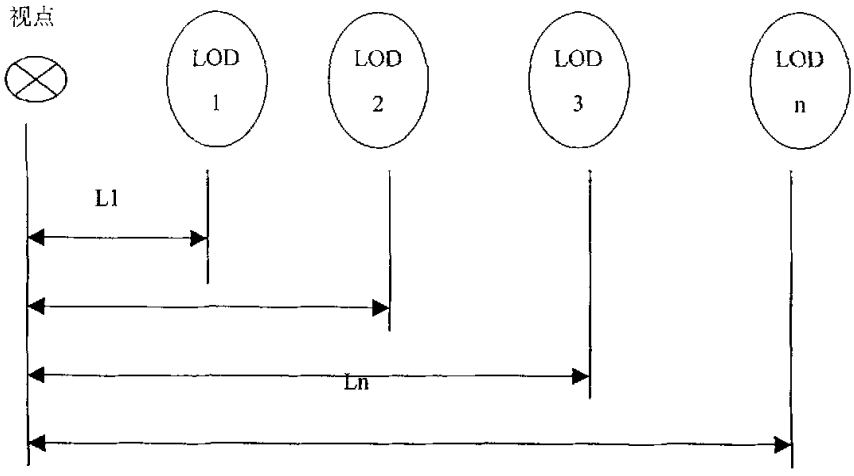


图 4—2 LOD 层次细节模型

回到本文的内容上来。本文是以 Java3D 为工具开发三维虚拟现实场景。由第三章的介绍可知，此时的物体和场景等都是以编程的方式来生成。如果延用前面的传统的 LOD 思路，对一个物体按照视距的不同建立多个 LOD 模型，那么将会凭空多出许多模型，也就多出了许多模型数据。尤其在大型场景构建中，因为此时场景中本来已经数量巨大的物体模型都因为要满足视距变化和实时运算而建立多级模型，使得 LOD 模型数量更为巨大。这样不仅使得代码量增大，而且在网络传输中为了传输这些额外的物体多级模型将会占用不必要的带宽。

4.3 Java3D 中对传统 LOD 算法的改进

由前一小节的论述可知，传统 LOD 算法的缺陷在于为了满足实时要求而对可视物体建立的多级 LOD 模型使数据量大为增加，从而不仅加大数据量而且该数据占用大量带宽使得降低网络应用率。

我们发现，相邻两级 LOD 模型代码中大部分数据是相同的，只是每近一级的 LOD 模型比远一级的模型多了一部分细节数据代码，再近一级的则又多一部分细节数据代码。因此多出的那部分数据代码其实是一样的。换言之，除了最远处是完全抽象成为几何形体这外，在视点从能够分辨出物体轮廓到逐渐逼近物体模型的过程中，LOD 代码是在相应前一级 LOD 代码基础上逐渐增加的。那么很自然的我们就会想到可不可以将那些完全相同的数据代码只保留一次，而

不必多次重复，只是每次视点靠近一级就多显示一些模型细节即可。如此只需建立一个模型，而只要在该模型中将相关细节分级显示即可，这样的思路就避免了对同一物体重复建模而带来的代码冗余问题。并且延用这样的思路，LOD模型过少而导致的图像跳变和LOD模型过多而导致的数据量过大之间的矛盾也就相就解决了。因为我们大可以在一个LOD模型内部将显示层次分的细一些，使得每次 LOD 切换时变化少一些从而减少图像跳变，而这些代码在一个 LOD 模型内只需写一次，也不用担心数据冗余。

这样的思路在 Java3D 中完全可以实现。前面一章介绍的 Java3D 的语法中指出，Java3D 虽然在编写代码时就将模型封装成类而不可变动，但是为了满足交互性，Java3D 也专门设定了 CAPABILITY 命令，该命令中相应参数的设定可以允许被显示的物体在渲染过程中根据用户的输入或其它控制实现相应的变换和运动。这些运动包括大小、形状、颜色、透明度、位置等等。当然我们也可以以这样的方式对已经编译封装的模型进行细节调用和显示。这样就达到了我们前面所设想的不增加冗余数据代码的前提下分层次显示物体模型的预期。

具体来说，以一栋房子为例。从视距可以分辨房子的轮廓到近处可以看到房子的细节的过程假设分为三个 LOD 级别：①看到房子的大体轮廓，能区别房顶和侧面的墙②在前面的基础上分辨出大门、窗户、阳台等③继续靠近，可以看到合金窗户框、窗玻璃颜色、墙体材料等。如下图所示。

这是典型的 LOD 算法的应用。用传统的 LOD 算法，要对这栋房子建立三个 LOD 层次级别的模型，下图最右边为级别最低的，中图次之，左边的图是级别最高的，各个细节也是最清楚的。很明显，右图中的数据只是中图的一个真子集，中图中的数据又是左图的真子集。这样建立三个级别的 LOD 模型的结果是在视点不靠近这栋房子的时候降低了系统的运算量，但是，为了建立这样的三个模型，中图的信息被记录了两次，右图的数据被记录了三次。

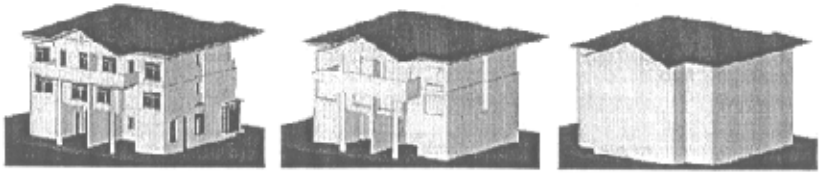


图 4－3　房屋的层次细节模型

可以想象，这样建立三个 LOD 级别的图像还是有跳变的，若想减少这种跳变感，就需建立更多的级别模型，然而建立的级别模型越多，数据重复记录的情况就越严重。当一个大型的场景中有多数物体需要用到 LOD 层次模型技术的时候，这个问题变得越来越严重。如果这个虚拟现实场景是被用于网络上，那么每个物体的 LOD 层次模型在网络上来回传输占用的带宽也就相应越来越大。

用改进的 LOD 算法这个问题就能得到很好的解决。我们只需要建立上图中最左边的那个模型，也就是最清晰的模型（每个 LOD 模型组中必有一个最清晰的模型），只不过把左图和中图两个模型中的信息数据的差值设置成最高显示渲染级别，中图和右图两个模型中的信息数据的差值设置成次高的显示渲染级别，而右图的信息数据被设置成中要这栋房子在视点可视范围内就永远可见，就在系统的显示负载上达到与传统 LOD 算法一致的运算量，而信息数据的存储却比传统 LOD 模型少许多。并且改进的 LOD 算法有很好的可扩展性，程序员可以为了减少图像跳变而把显示信息分成更多的显示级别，而不用担心数据量的增大。因为每多分一级显示级别，无非是在整个数据的基础上多设置几个标识位而已。

4.4 传统 LOD 算法和改进 LOD 算法对比

以三个分界点，四级 LOD 模型为例，传统的 LOD 算法结构是这样的：

主程序：

```
BranchGroup objRoot = new BranchGroup();
BoundingSphere bounds = new BoundingShere();
TransformGroup objMove = new TransformGroup();
Switch targetswitch = new Switch();
TargetSwitch.setCapability(Switch.ALLOW_SWITCH_WRITE);
targetSwitch.addChild(model_1);
targetSwitch.addChild(model_2);
targetSwitch.addChild(model_3);
targetSwitch.addChild(model_4);
float[] distances = {5.0f,10.0f,20.0f};
DistanceLOD dLOD = new DistanceLOD(distances,new Point3f());
```

```
dLOD.addSwitch(targetSwitch);
dLOD.setSchedulingBounds(bounds);
objRoot.addChild(objMove);
objMove.addChild(dLOD);
objMove.addChild(targetSwitch);
return objRoot;
```

LOD 分级模型建立的子程序有四个：

```
public class model_1{
                    //第一级模型
                    }
public class model_2{
                    //第二级模型
                    }
public class model_3{
                    //第三级模型
                    }
public class model_4{
                    //第四级模型
                    }
```

而上面的四级模型中有相当多的数据是完全冗余重复的。

改进后的 LOD 算法结构是这样的：

```
BranchGroup objRoot = new BranchGroup();
BoundingSphere bounds = new BoundingSphere();
TransformGroup LODgroup = new TransfromGroup();
LODgroup.setCapability(Group.ALLOW_CHILDREN_WRITE);
LODgroup.setCapability(Group.ALLOW_CHILDREN_EXTEND);
LODgroup.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
model_1{};   //
differentpart_1-2{}; //LOD 模型 1 和 2 之间的差别部分
differentpart_2-3{}; //LOD 模型 2 和 3 之间的差别部分
differentpart_3-4{}; //LOD 模型 3 和 4(即完整细节)之间的差别部分
```

```
LODgroup.addChild(model_1);
If(boundary1<distance< boundary2){
                                        LODgroup.addChild(differentpart_1-2);
                                        }
If(boundary2<distance< boundary3){
                                        LODgroup.addChild(differentpart_2-3);
                                        }
    If(boundary3<distance){
                                LODgroup.addChild(differentpart_3-4);
                                }
    objRoot.addChild(LODgroup);
    return objRoot;
```

由上面的改进前后的 LOD 算法结构，结合前面说明的 Java3D 的运行模式可以看出，改进后的算法结构和传统的算法结构相比，其数据量有相当幅度的减少，这种缩减在 LOD 级别分的更细层次更多时表现更为突出。同时，改进后的算法可以达到和传统的算法一样的显示效果和系统运算量。所以这样改进之后的 LOD 算法不仅降低了代码的数据量，而且使得这样的场景更适合网络的传输和应用，扩大了以 Java3D 为工具开发的虚拟现实场景在互联网中的应用范围。

第 5 章　用 Java3D 开发桌面虚拟现实系统

5.1 技术路线

系统的设计目标是实现桌面三维虚拟现实场景的网络运行。用户终端可以通过普通的浏览器（当然必须安装了相关的 Java 运行环境）就可以漫游这个桌面三维虚拟现实场景，使三维可视化显示不再停留在单机版，而是能够通过互联网来实现。同时，用户终端还可以根据自己的需要与三维虚拟现实场景进行交互，通过简单的操作在该三维虚拟现实场景中任意浏览漫游。

本文前面的章节介绍了当前桌面三维虚拟现实场景的情况和各种开发工具，最终选择 Java3D 为开发工具，其目的就是可以利用继承了 Java 优良特性的 Java3D 的对向对象和跨平台等特性，并且能够完美的和普通网络浏览器相结合。Java3D 本身的开发三维场景的专业性，LOD 等特点也正好适合于网络应用。

5.2 系统环境和构架

5.2.1 开发环境

- 硬件环境：

处理器 Intel x86 体系构架处理器 最低 300M 建议用更高主频的 CPU

内存 最低 128M　256M 以上最佳

显卡 NVIDIA GeForce4 MX 440 或以上

显示器 分辨率 1024*768　　刷新率 85Hz

- 软件环境

操作系统 Windows NT 4.0 SP3 或更新

服务器 Apache Tomcat 5.0 Server

Java 运行环境 Java 2 Runtime Environment Standard Edition v1.4.1_02

Java 开发包 j2sdk-1_4_1_02-windows-i586

OpenGL 包 OpenGL v1.1

Java3D 运行环境 Java3D-1_3_1-windows-i586-opengl-rt

Java3D 开发包 Java3D-1_3_1-windows-i586-opengl-sdk

浏览器 Internet Explore 6.0

开发工具 Eclipse Release 2.1.3

5.2.2 系统的构架

本系统采用三层 B/S 体系结构，由客户端，Web 服务器和数据库构成，如下图所示，各层的功能如下：

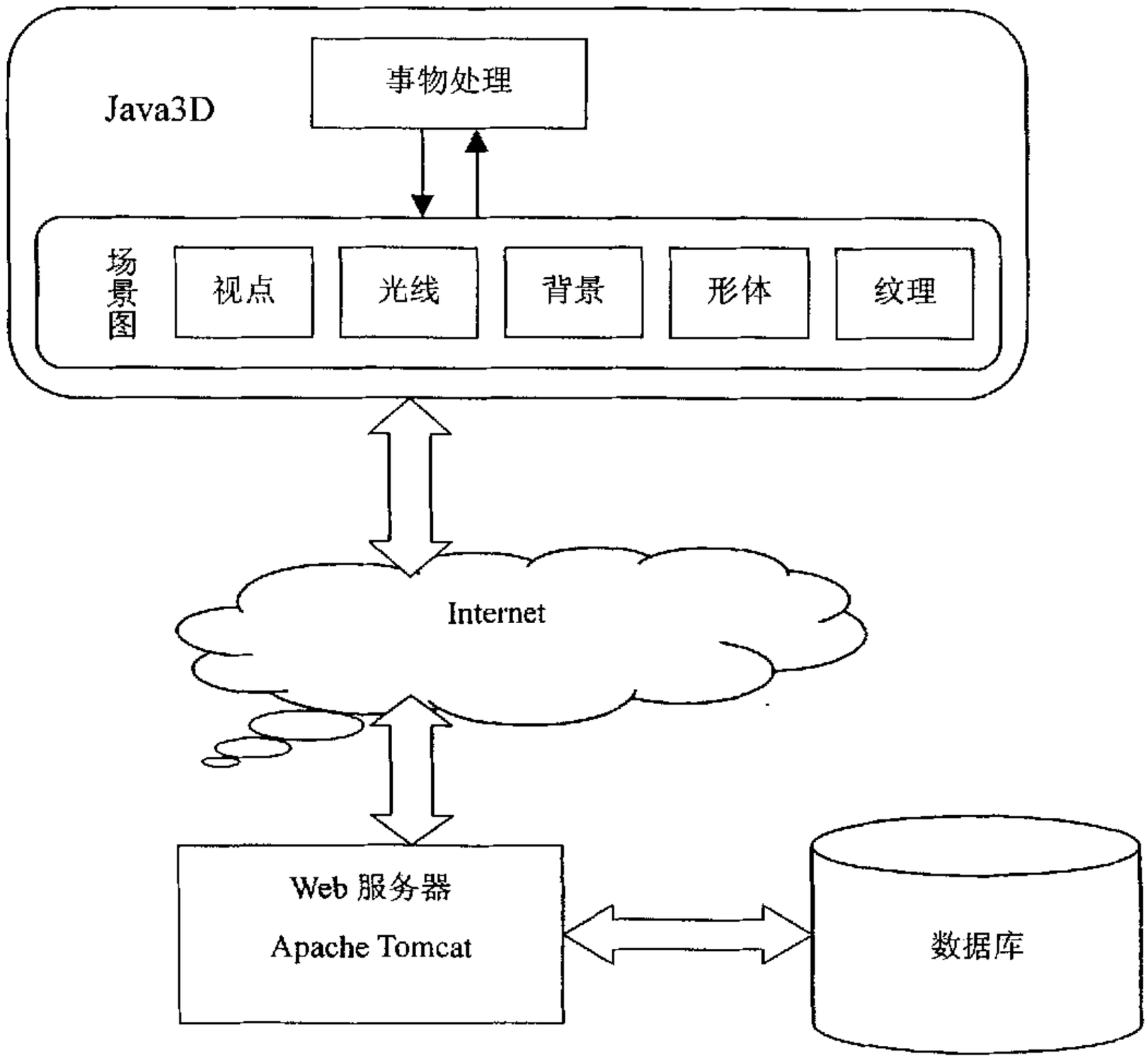


图 5－1 系统框架结构图

（1）客户端浏览层。完成桌面三维场景图的下载，处理显示以及和用户的交互工作。基于 Java3D 的客户端采用在网页中使用的 Applet 技术，通过 Java3D 提供的功能强大的 API 来实现场景的三维建模，交互漫游等操作，从而可以在

浏览器中进行浏览。本系统采用胖客户端瘦服务器模式，即获取数据后的场景构建和交互工作都由客户端完成，这样可以让系统有更大的容量提供更多的用户使用。

（2）应用服务器层。处理用户的请求，向数据服务器请求相应的数据，并进行数据处理、转换，传回给客户端相应的数据或查询结果。

（3）数据层。存储管理系统所需的类、包，以及注册用户数据等信息。提供应用服务器相应的数据请求，返回给服务器供应用服务器使用。

5.3 系统实现

下面给出一个 C/S 模式的虚拟现实系统构架图。这个构架图由客户端的浏览器和服务器端两个部分构成。服务器端是符合高层次体系结构的模块，并且该模块也已经注册到系统中并能够给客户端提供数据服务。服务器端初始化时就是作为只向客户端提供数据服务的而不是干预客户端处理代码的“忠实的”服务器，这样减轻了服务器的负担，服务器只需要将请求的数据传给相应的客户端就完成这个客户端的请求工作，从而加大了整个系统的负载能力。

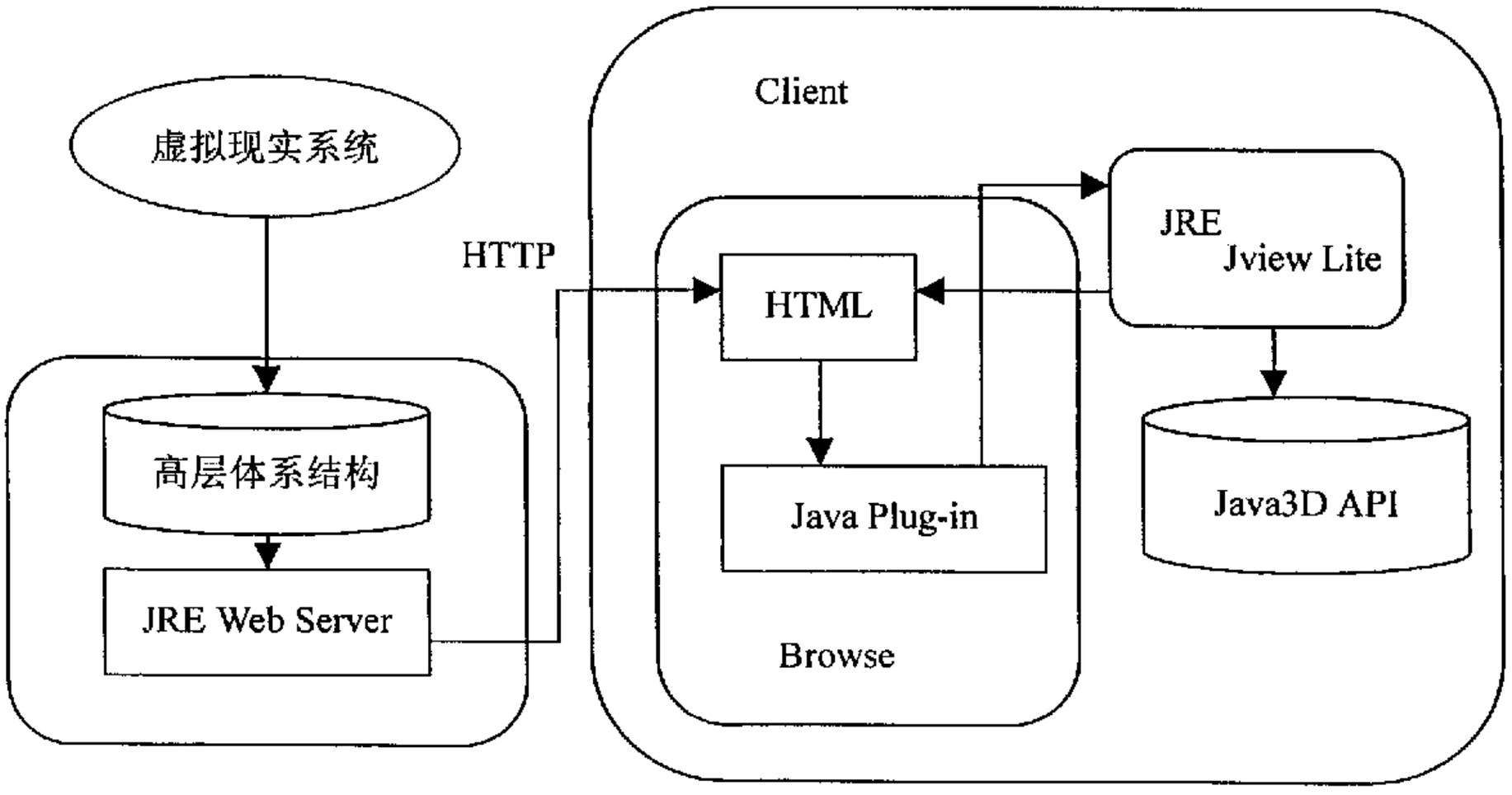


图 5—2　桌面虚拟现实系统框架图

上图就是 Java3D 在 Web 上工作时远程调用时的工作框图。图中 Jview—Lite

是嵌入 IE 浏览器中的 Applet 工具的一个方法。有了 Java Plug-in 和 Java3D API 工具集，Java3D 工作环境的三维渲染程序就可以启动，Applet 运行时可以让观察者从任意希望的角度和距离去观察虚拟现实场景。并且进一点看，现在的浏览器还可以让观察者同时打开多个浏览窗口，每个窗口中开启自己的 Applet 进程对同一个虚拟场景进行浏览，因为现在的客户端被允许观看场景中的所有的信息。

一旦客户端的浏览器发出请求，浏览进程就被启动起来，通过一个 HTTP、连接开始对仿真场景进行浏览。服务器提供一个链接用于传递网页的代码以及被嵌入到网页中去的 Applet 程序。客户端浏览器从被初始化开始就开始一个高效的 3D 仿真三维虚拟现实场景的渲染过程。在用户进行浏览的过程中，这条从客户端到服务器的 HTTP 连接保持活动状态，以便可以随时向客户端提供请求的响应。这整个活动一旦开启，服务器就不断的将请求所需的数据包括物全数量，位置，形体，向量等信息传给客户端，客户端就不断的基于这些数据去渲染这个三维仿真场景图。

当客户端的浏览器被关闭时，它将发出一条信息给服务器端，告知服务器这条传送给特定客户端的请求和响应数据的链接可以停止使用了；或者是服务器端已经通过其它方式确信客户端浏览器不再需要新的数据时，服务器的这条特定链接将被关闭，而开始建立其它的在等待队列中的客户端数据请求。

5.3.1 对象的分类

根据 Java3D API 本身的功能特性和开发的数据要求，可以将所有的对象分为下面这样几类：

1、 实体类：所有的建筑物单元、街道、树、桥等具体的都可以用实体类来表示。并且每个实体类都有相应的标识号，节点号等参数以便可以唯一的标识该物体方便场景代码对其的调用和环境对物体的生成和渲染。

2、 数据类：这里的数据类指的是各个实体的表面属性包括材质、纹理等相关数据，以及光线，物体光反射特性等数据。

3、 场景类：因为有一些物体需要在某一个场景中联合显示或者被某一个光照等物所影响，所以需要将这些相关实体和属性等结在一个组节点下，这样就引出了场景类 BranchGroup 类。

4、 操作类：为了使场景能和用户进行方便的交互，必须定义一系列相关的

操作类，当用户进行这些操作的时候场景图就会做出相应的反应，达到交互的效果。这其中当然包括鼠标的运动 MouseRotate、MouseZoom 等。

将场景中的所有信息以这样的方式分类是为了方便类的组织和调用，相应数据可以重复使用以减少冗余代码，达到少码高效的效果。经过了这样的分类，就可以下一步的工作，来设计三维场景了。

5.3.2 三维场景模块的生成

本论文中的三维场景实现是用 Java3D API 为工具。在用 Java3D 来构造场景大致要经历这么几个步骤：几何数据的预处理，参数的设置，模型的构造，投影变换，视口变换到场景的建立。

(1)数据预处理　数据预处理主要是将建模后得到的场景中物体的几何模型数据转换成一 Java3D API 可接受的基本图元的形式，如点、线、面。

(2)参数设置　在对三维场景进行渲染之前，需要设置相关的场景参数值。这些参数一般包括光源性质(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)等。

(3)模型构造　模型构造就是对整个场景模型的集合中间过程。几何重建过程根据 Java3D API 中的有关命令函数将模型数据“装配”成场景模型。

(4)投影变换　在从 Java3D API 中应用的是透视投影变换。

(5)视口变换　视口是指计算机屏幕中的矩形绘图区域。视口变换的目的就是将三维空间坐标影射为计算机屏幕上的二维屏幕坐标。

由前面的 Java3D 介绍可知，在用 Java3D 编写的场景代码中，几何形体有三种生成方式，其一是用坐标系中的点，线，面结合生成各种形体。这个方法的优点是不论形体简单复杂都可以实现，但是每个物体都以这样的方法来生成显然是大费周章。一般不提倡使用这个方法来生成形体。第二种方法是以 Java3D 实用扩展包 Utility 中的基本形体，包括 Box、Sphere、Cylinder、Cone 等为基础构建一般形体。这个方法对一般物体简单易行，只要生成相应的简单形体，通过平移和旋转等方法将生成的简单形体放置到空间中相应位置，然后将若干个简单形体组合成 Group 节点就可以生成一些相对较复杂的形体。在一般场景构建中，这样的方法是很值得借鉴的。第三种方法是对相当复杂的形体的生成。这样的形体用前两种方法都将使工作量大大增加，且难以得到逼真的效果。此时我们采取直接读取其它格式的图形调用的方法。如 3D MAX，VRML 格式生

成的复杂形体。这些工具可以相对以简单的多的方法生成逼真的，复杂的形体。而 Java3D 可以直接读取诸如 3D MAX 格式，VRML 格式或者 AutoCAD 格式等的对象，并将其转化为 Java3D 本身的存储格式的形体。所以，对于特别复杂的形体我们采取第三种方法来生成，其它一般物体我们采取第二种方法生成，极少数不适合这两种方法的形体才用第一种方法来生成。

有了这样的思路，我们就可以采用适当的方法来生成各种各样的形体。例如我们用 Utility 中的简单几何形体来组合生成场景中的街道、路面、电线杆、电线、水泥墙、石柱、人行栏杆、天桥等。例如下面的代码段即为生成一个石柱的代码。

```
        private SharedGroup buildSharedColumn( )
    {
        Appearance columnApp = new Appearance( );

        Material columnMat = new Material( );
        columnMat.setAmbientColor( 0.6f, 0.6f, 0.6f );
        columnMat.setDiffuseColor( 1.0f, 1.0f, 1.0f );
        columnMat.setSpecularColor( 0.0f, 0.0f, 0.0f );
        columnApp.setMaterial( columnMat );

        TextureAttributes columnTexAtt = new TextureAttributes( );
        columnTexAtt.setTextureMode( TextureAttributes.MODULATE );

columnTexAtt.setPerspectiveCorrectionMode( TextureAttributes.NICEST);
        columnApp.setTextureAttributes( columnTexAtt );

        if ( columnTex != null )
            columnApp.setTexture( columnTex );

        GothicColumn columnShape = new GothicColumn(
            ColumnHeight,   // height
            ColumnRadius,   // radius
```

```
            GothicColumn.BUILD_TOP,   // flags
            columnApp );     // appearance

    // BEGIN EXAMPLE TOPIC
        // Build a shared group to hold the column shape
        SharedGroup column = new SharedGroup( );
        column.addChild( columnShape );
    // END EXAMPLE TOPIC

        return column;
    }
```

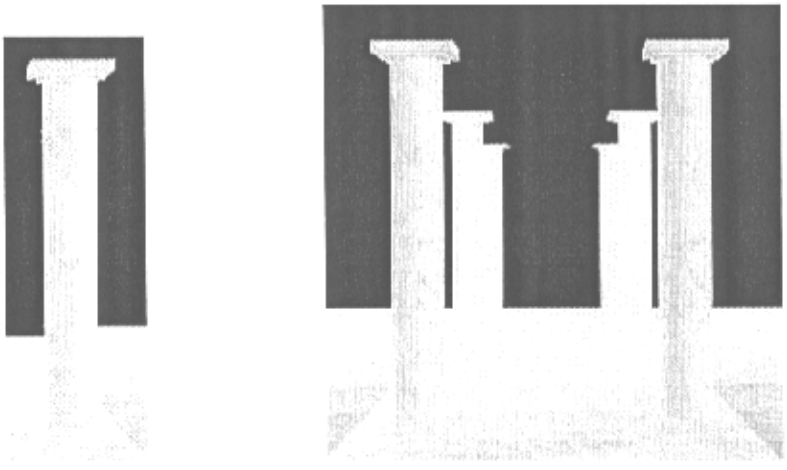


图 5—3　石柱模型图

上面右图就是生成的单个石柱的图像。设置石柱成两排，并沿着石头路两边等距摆放，所得的效果图如上面的右图所示。

这样的石柱显得单调而又不逼真。既然为石柱，我们选择大理石的纹理图片贴在石柱的表面，使得石柱看起来效果更好。

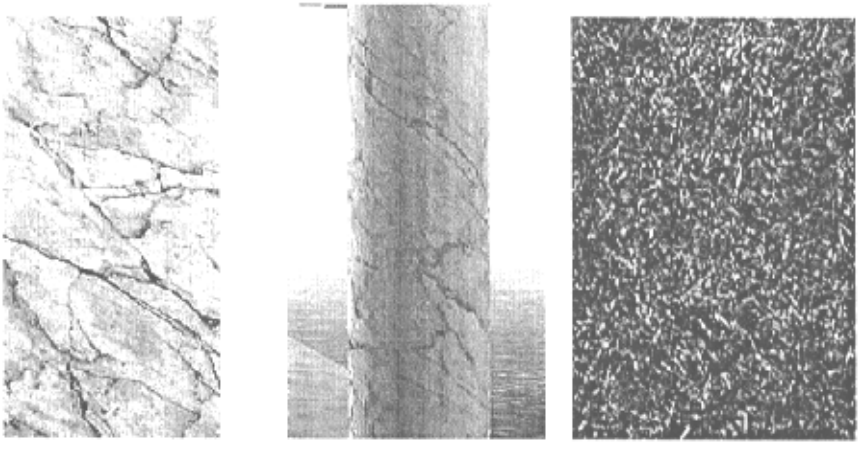


图 5—4　石柱和草地的材质图

上面的右图就是大理石纹理样本。将这个样本贴在石柱上，就得到上面中间的石柱图片。这样的石柱看起来就相当逼真了。

然后我们将石头路两边的地面设置成草地。方法同上，只要获取草地的材质纹理图片，将它平铺在石头路两边的空白地面上就可以生成逼真的草地了。草地纹理图片如上面右图所示。石头路两旁铺上草地，沿着石头路两测放置上石柱（石头路和石头地面都贴上大理石纹理图片效果）之后的效果图如下图所示。

这样，一个基本的场景就建立起来了。该场景中有一条大理石的路面，沿着路的两侧有两行等距矗立的大理石石柱，路面两边是绿色的草地。

图 5－5　贴上纹理的场景图

5.3.3 视点及视平面的设定

Java 3D 可通过几个对象来定义视模型参数。这些对象包括 View 对象及其相关对象、Physical Body 对象、Canvas3D 对象、Physical Environment 对象、Screen3D 对象。Java3D 给这些对象提供了默认参数，从而确定了初始的观察分支，使编程工作的主要精力集中在内容分支上。但在用户观察三维场景时，观察内容和观察方位应该是可以改变的。视平台的父节点确定了视平台在虚拟环境中的位置和方向。若通过修改与 Transform Group 节点相关的 Transform3D 对象，就可以在虚拟场景中随意移动视平台。本系统开发了改变视点和在场景中漫游的功能用户可在前面所说的二维平面图上设置初始视点、视线，系统根据此参数，改变初始的观察位置。比如：

```
public TD(){
    setLayout(new BorderLayout());
    GraphicsConfiguration                    config
```

```
                    SimpleUniverse.getPreferredConfiguration();
    Canvas3D c = new Canvas3D(config);
    add("Center", c);

        ViewPlatform                viewPlatform;
        Viewer viewer = new Viewer(c);
        Vector3d viewpoint = new Vector3d(0.0,0.0,4.0);

        Transform3D t = new Transform3D();
        t.set(viewpoint);
        ViewingPlatform v = new ViewingPlatform();

    v.getViewPlatformTransform().setTransform(t);

      BranchGroup scene = createSceneGraph(c);
      SimpleUniverse u = new SimpleUniverse(v, viewer);
      u.getViewingPlatform();
      u.addBranchGraph(scene);
        }
```

在默认的情况下，视点被放在距离可视物体 2.41 米处。这样就可以将一个视平台和一个视点联系起来，并将其放置在空间的相应位置上。于是在代码开始运行时，Java3D 运行环境就在视点可见的范围内工作，渲染相应的物体。若要实现漫游效果，也就是要视点可以人为的操控而运动，就要设置键盘监听事件。com.sun.java.util.behaviors.keyboard 包用于监听键盘事件的产生，KeyNavigatorBehavior 类实现了采用键盘控制视点的移动。所以要先用语句 import com.sun.java.util.behavoirs.keyboard.*;导入这个包，然后在代码编写中用 getViewingPlatform().getViewPlatformTransform() 方法获得观察者所处的 TransformGroup 对象，然后将其添加到 KeyNavigatorBehavior 对象中去，设置它的作用范围，最后将将它添加到场景中接收输入就可以实现键盘控制场景漫游。同样的道理，导入鼠标监听事件就可以实现由鼠标控制的场景漫游。

5.3.4 交互的设计

交互性能对于三维虚拟现实环境来说是至关重要的，Java3D 的交互式应用主要有三类：一类是利用 Java 的事件处理模型，一类是利用 Java3D 提供的 Behavior 包中的 Utility 对象，一类是定义自己的 Behavior 对象。

Java 的事件处理采用的是一种“事件派遣模型”，在应用中，首先向事件源注册监听器，若事件源处发出了一个事件，针对已就那个事件向事件源注册的所有监听器对象，事件源就会向它们全体发出一个通知。监听器对象属于一个类的实例，该类实现了监听器接口，监听器对象必须实现相应的监听器接口方法，实现对事件的处理。比如用 implements 说明该类实现的监听器接口

```
public class classmane extends Applet implements WindowLinstener
ActionListener,
ItemListener{}
注册一个监听器，如一个菜单条作为事件源，
Menu m = new Menu(String s);
m.addActionListener(this);
然后实现监听器接口的方法
public void actionPerformed (ActionEvent e){
//触发事件的说明
}
```

第二种方法是用 Behavior 类实现。Behavior 类是 Java3D 中定义的一个抽象类，它定义了所有 Behavior 对象的公共属性。它的运行机制在于它可以接收特定的激励（stimuli），如键盘按键，鼠标移动等，然后对这一激励做出响应，如改变观察者的视点，虚拟物体的位置，外观等。创建一个 Behavior 对象实现交互有这样三个要素①编写一个构造函数，在该函数中保存控制对象的引用②重载 initilize()方法，指定第一次唤醒条件③重载 processStimulus 方法，根据不同的唤醒条件触发不同的动作，重新设置唤醒条件。

5.3.5 背景效果的生成

为了使得构建的场景更逼真，更具有沉浸感，交互性和思维构想性的特性，可以对整个场景增加背景的设置。背景环境包括以下三个方面：

- 背景图像

背景图像是指在场景中加上背景，比如天空，沙漠，海洋等，这样是为了让桌面虚拟现实场景更加逼真，仿佛就是在现实中取的景一样，更加增添了场景的沉浸性。

- 背景声音

背景声音可以有两种方式，一种是全局背景声音，也就是说在场景开始渲染时音效就开始工作，在整个浏览过程中声音一直沿续比如游戏里的背景音乐。另一种是局部背景声音，是在某个局部子场景中生效，或者在某个事件被触发时生效，例如点击鼠标时，或者从一个场景进入另一个场景中的时候。

支持声音文件的包位于 javax.media.j3d.Sound 下面的 javax. media. j3d. BackgroundSound 和 javax.media.j3d.PointSound 包中。Java3D 可以直接调用支持的典型声音文件格式有 AIF、AU 和 WAV 等。调用声音背景时也可以使用 Alpha 参数设置声音播放次数，也可以设置为无限循环播放。

- 背景效果

Java3D 还可以支持背景效果。比如光，雾等。例如前面创建的大理石路，石柱，草地场景，给它加上雾气背景后的场景效果图如下所示。调整相应的参数就可以实现不同的雾气效果，例如调整雾的浓和淡。其中第一个图是薄雾的效果图，第二个图是浓雾的效果图。

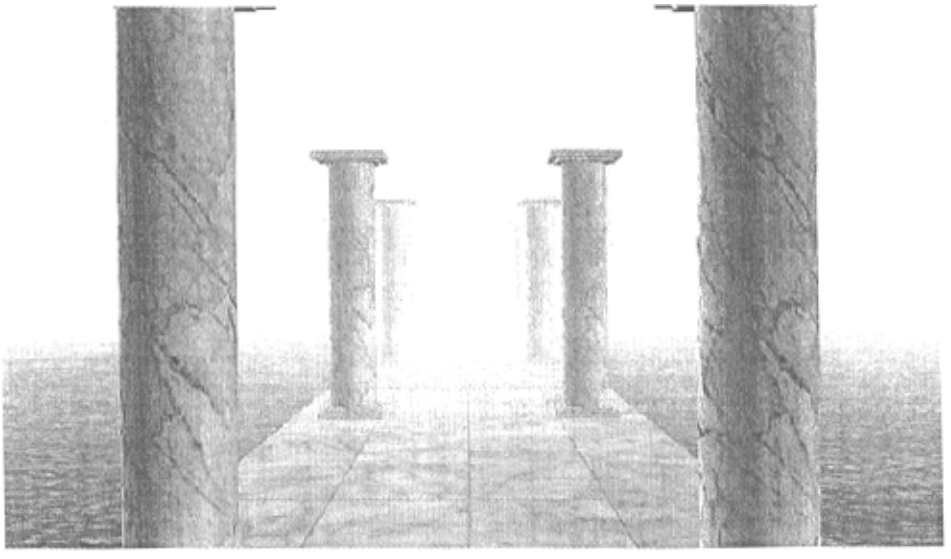


图 5－6　有雾的效果的场景图

第 6 章　改进 LOD 思想的应用与展望

6.1 用改进的 LOD 思想优化代码

下面将改进的 LOD 算法思想应用在前面的例子中的一个最简单的物体上，来比较分别应用传统 LOD 算法思想和改进后的 LOD 算法思想的代码的差别。

前面例子中是了一个简单的场景，即一条两边均是草地的大理石路上沿着路的两侧等距矗立着大理石的石柱。这个场景中最典型的应用 LOD 算法的地方就是石柱。在最简化的情况下，我们为大理石柱建立两个 LOD 细节模型。一个是距视点较远时，只看到白色的圆柱体和顶上的方体；另一个是细节模型，即近距离观看时可以看到其上的纹理，光照，反射情况等。

传统思想 LOD 模型及调用代码如下：

```
BranchGroup objRoot = new BranchGroup();
BoundingSphere bounds = new BoundingShere();
TransformGroup objMove = new TransformGroup();
Switch targetswitch = new Switch();
TargetSwitch.setCapability(Switch.ALLOW_SWITCH_WRITE);
targetSwitch.addChild(model_1);
targetSwitch.addChild(model_2);
float distances = 10.0f;
DistanceLOD dLOD = new DistanceLOD(distances,new Point3f());
dLOD.addSwitch(targetSwitch);
dLOD.setSchedulingBounds(bounds);
objRoot.addChild(objMove);
objMove.addChild(dLOD);
objMove.addChild(targetSwitch);
return objRoot;
```

下面是第一个简单模型的建立：

```
public class model_1 extends Applet{
```

```
Appearance app = new Appearance();
Material material = new Material();
material.setEmissiveColor(new Color3f(1.0f,1.0f,1.0f));
app.setMaterial(material);

Cylinder …… //建立石柱的圆柱部分

    Box …… //建立石柱顶端的方形部分

    … addChild … //将各部分组合成组节点
                                    }
```

这是细节模型的建立：

```
public class model_2 extends Applet{
        Appearance app = new Appearance();
        Material material = new Material();
        material.setAmbientColor( 0.6f, 0.6f, 0.6f );
        material.setDiffuseColor( 1.0f, 1.0f, 1.0f );
        material.setSpecularColor( 0.0f, 0.0f, 0.0f );
        app.setMaterial(material);

        app.setTexture( shizhuTex );
        TextureAttributes shizhuTexAtt = new TextureAttributes( );
        shizhuTexAtt.setTextureMode( TextureAttributes.MODULATE );
        shizhuTexAtt.setPerspectiveCorrectionMode( TextureAttributes.NICEST);
        texLoader = new TextureLoader( "granite07rev.jpg");
        shizhuTex = texLoader.getTexture( );
        shizhuTex.setBoundaryModeS( Texture.WRAP );
        shizhuTex.setBoundaryModeT( Texture.WRAP );
        shizhuTex.setMinFilter( Texture.NICEST );
        shizhuTex.setMagFilter( Texture.NICEST );
```

```
shizhuTex.setMipMapMode( Texture.BASE_LEVEL );
shizhuTex.setEnable( true );
app.setTextureAttributes( shizhuTexAtt );

Cylinder ..... //建立石柱的圆柱部分

Box ..... //建立石柱顶端的方形部分
... addChild ... //将各部分组合成组节点
                                    }
```

改进后的 LOD 模型调用代码为：

```
BranchGroup objRoot = new BranchGroup();
BoundingSphere bounds = new BoundingShere();
public class model extends Applet{

 if(distance>10.0f){
        Appearance app = new Appearance();
      Material material = new Material();
      material.setEmissiveColor(new Color3f(1.0f,1.0f,1.0f));
      app.setMaterial(material);
                    }
else{
        Appearance app = new Appearance();
      Material material = new Material();
      material.setAmbientColor( 0.6f, 0.6f, 0.6f );
      material.setDiffuseColor( 1.0f, 1.0f, 1.0f );
      material.setSpecularColor( 0.0f, 0.0f, 0.0f );
      app.setMaterial(material);

      app.setTexture( shizhuTex );
```

```
        TextureAttributes shizhuTexAtt = new TextureAttributes( );
        shizhuTexAtt.setTextureMode( TextureAttributes.MODULATE );
      shizhuTexAtt.setPerspectiveCorrectionMode( TextureAttributes.NICEST);
      texLoader = new TextureLoader( "granite07rev.jpg");
       shizhuTex = texLoader.getTexture( );
       shizhuTex.setBoundaryModeS( Texture.WRAP );
       shizhuTex.setBoundaryModeT( Texture.WRAP );
       shizhuTex.setMinFilter( Texture.NICEST );
       shizhuTex.setMagFilter( Texture.NICEST );
       shizhuTex.setMipMapMode( Texture.BASE_LEVEL );
       shizhuTex.setEnable( true );
          app.setTextureAttributes( shizhuTexAtt );
    }
          Cylinder …… //建立石柱的圆柱部分

          Box …… //建立石柱顶端的方形部分

          … addChild … //将各部分组合成组节点
    }
    objRoot.addChild(model);
    return objRoot;
```

由前面的代码对比可以看出，LOD 算法改进后代码行数缩减了。编译后生成的字节码文件也就相应缩小了一本来要生成二个 LOD 模型，现在却只需要生成一个模型，只不过其中一部分信息有的时候不显示而已，而渲染的可视化效果当然也就没有下降。

上面只是就一个简单的石柱为例，并且只建立了二级 LOD 模型，已经可以看到改进算法的效果。如果在一个大型场景中，有很多物体需要建立 LOD 模型，并且为了减少图像在界线之间的跳变感，需要建立多级 LOD 模型，这时那些许多本来需要建立的冗余的，之间差别并不大的模型就都被省略掉了，这种情况下改进的 LOD 算法就显得更加有效。

6.2 结论和展望

随着可视化的发展和 Internet 互联网络的日益普及，在包括建模与仿真、科学计算可视化、设计与规划、教育与训练、遥作与遥现、医学、艺术与娱乐等的众多方面，虚拟现实技术正在得到广泛应用。对虚拟现实技术的研究，尤其是对桌面虚拟现实系统的研究已经成为未来一个重要的研究方向。

本论文在简单介绍了虚拟现实系统的相关背景之后，对比了当前主流的虚拟现实系统开发工具包括 Java3D、VRML、OpenGL 等，由于具有纯粹的面向对象特性、可交互性、支持多媒体、可与 Java 体系无缝结合、适于网络应用等优点，最后选择以 Java3D 图形包为开发工具，阐述了以 Java3D 开发桌面虚拟现实系统的方法和步骤。

对于为了提高系统性能而使用 LOD 分级建模的方法导致的模型过多，使得占用了多余的网络带宽的问题，本文提出了改进的 LOD 算法。改进的思路是将传统 LOD 思想中和创建多级 LOD 模型的思路合而成只需要一个模型，将原本需要多个模型来显示的细节全都写在这一个模型中，用条件触发的方式来显示，也就是说在相应条件满足的情况下，比如视距在某个范围内或者其它事件，将触发一部分细节信息被加入到渲染对象的组节点中去渲染显示。该思路使得本来需要生成多个级别层次模型的工作现在只需要生成一个模型，将许多冗余重复的数据摒弃掉，压缩了数据量，却并没有降低图像质量，相反，在网络应用中减少了传输的冗余数据，节省了网络带宽。而且这样的优势在大场景的建模中体现的更加明显，因为大场景的建模将会有许多物体要分许多级模型，冗余的数据就相应更多。

由于时间紧迫，本文只是就一个新的思路作了一般的探究，籍此希望能找到让三维虚拟现实的代码更简洁，运行效率更高，并相应压缩网络传输数据量的方法，使三维虚拟现实技术能更好的在网络上得到广泛应用。本文对 LOD 的优化算法进行了初步探究，该思路在理论上是可以达到预期效果的，但是其网络应用还需在进一步的实验和研究。另外，该思路和另外一些当前正在研究的几何图形 LOD 处理算法是在不同的局部对图像进行处理的，如果能让这两种处理结合起来或者相互借鉴，将会对三维景像有更全面的处理效果，更加优化其网络运行。另外，对于建立几何模型的第三种方法，即直接读取调用其它相关

软件生成的物体模型的方法，在建立 LOD 多级模型的时候理论上也可以延用本文的思路，将生成 LOD 的多级模型每级中只有相应增加的部分，而不必撰写相同冗余的部分。这样理论上也可以达到降低数据量的效果，但本文未对此思路作详细探讨，在今后的研究工作当中此亦不失为一种研究方向。

综上所述，本文研究的改进 LOD 算法模型可以在不降低虚拟现实图像质量的前提下减少程序的代码量和数据量，使得虚拟现实技术可以更好的应用于网络中，扩大了虚拟现实技术的应用范围，深化了虚拟现实技术的应用前景。相信对改进的 LOD 算法研究的深入和广泛的应用将是未来的趋势，也将为科学研究、人民生活以及国家经济做出更在的贡献！

参考文献

[1] 万华根.基于虚拟现实的 CAD 方法研究[博士论文].杭州:浙江大学，1999.1-20.

[2] 王立峰.基于几何和图象高真实感图形绘制方法研究〔博士论文〕.杭州:浙江大学，1999.2-70.

[3] SAWADA, KAZUYA. Virtual reality technology and its industrial applications. Control Eng Pract, 1999, 7 (11):1 381-1 391.

[4] 殷国富，陈田，戈鹏，等.基于虚拟现实技术的卫生瓷产品参数化 CAD 系统的研究.计算机辅助设计与图形学学报，2002 14 (2):168-171.

[5] 马小虎.虚拟现实中多细节层次模型的研究〔博士论文〕.杭州:浙江大学，1999.1-63.

[6] 赵小林，张俊，刘剑青.网页制作技术一虚拟现实篇.北京:国防工业出版社，2002.1-10.

[7] Brown Judith R,Van dam Andy,Earn Shaw ,et a1..Human entered computing,online communities and virtual environments. IEEE Comput Graphics Appl,1999,19(6):70-74.

[8] 覃征，刘晓艳，王利荣，等.电子商务案例分析.西安:西安交通大学出版社，2001.

[9] Java 3D Tutorial， http://java.su.com/product/java.media/java 3D.

[10] 瞿炜娜 基于虚拟现实技术的虚拟实验室的研究和实现 2003.2.20

[11] 淮永建，郝重阳.面向 VR 应用系统的 Java3DAPI.中国图象图形学报.2000,5(12):10441048.

[12] The Java3D White Paper http://java.sun.com/product/java3d

[13] 陈静勇 周来水 基于 Java3D 的虚拟现实建模方法 计算机应用研究 2002 年第五期

[14] 刘锦德 敬万钧 关于虚拟现实----核心概念与工作定义 计算机应用 1997年5月 第 17 卷第 3 期

[15] 王汝传 张登银 辛晨昀 虚拟现实中的 3D 图形建模方法的研究 计算机辅助工程 2000 年 12 月第 4 期

[16] 淮永建 郝重阳 面向 VR 应用系统的 Java3D API 中国图象图形学报 2000 年 12 月第 5 卷第 12 期

[17] 龚建成 张佑生 基于 JAVA 的 3D 图形开发技术 安徽工程科技学院学报 2003 年 3 月第 18 卷第 1 期

[18] 张大坤 薛忠明 王光兴 基于Java3D的不规则形体三维造型及真实感处理 计算机工程 2003 年 2 月第 29 卷第 2 期

[19] 张杰 梁志明 基于 JAVA3D 技术的参数化三维图形设计方法汕头大学学报 2003 年 8 月第 18 卷第 3 期

[20] 何丹 穆振东 江顺亮 Java3D 可视化技术及其应用 计算机与现代化 2003 年第 8 期

[21] 罗宁 傅秀芬 Java3D-----Java 语言的三维图形解决方案 现代计算机 2001 年 5 月总

第 115 期

[22] 李志均　傅秀芬 Java3D API 与 Java3D 编程技术　现代计算机　1999 年 4 月总第 73 期

[23] Bruce Eckel 著　侯捷译 JAVA 编程思想（第二版）　机械工业出版社

[24] 徐洪珍 周书民 汤彬　VRML 和 JAVA 的交互及其应用　计算机与现代化 2003 年 11 期总第 99 期

[25] 王详书　汪厚祥　张正霞　WEB3D 及其动态交互技术舰船电子工程　2000 年第 4 期

[26] 王大江　沈旭昆　数字航空博物馆三维场景的构建太原理工大学学报　2002 年 11 月第 33 卷第 6 期

[27] 刘鹏 李三立 都志辉网络环境下高性能计算的可视化　小型微型计算机系统　2002 年第 10 期

[28] 余小燕基于 Java3D 的交互式三维动画编程技术　工矿自动化　2003 年 12 月第 6 期

[29] 石教英 虚拟环境(VE)技术及其应用 全国第一届虚拟环境研讨会论文集，杭州，1994.

[30] 张杰 JAVA3D 交互式三维图形编程 北京:人民邮电出版社，1999

[31] 黄心渊 虚拟现实技术与应用 北京:科学出版社，1999.

[32] 王朔 可用于虚拟建筑环境的 Web3D 技术初探 2004.11

[33] Mueller-Spahn, F.; Hofmann, M.; Roessler, A.; Bullinger, A.H. ，3D virtual reality (VR) systems, cognition and creativity， European Psychiatry， Vol: 13, Supplement 4, 1998

[34] Huang, Bo; Lin, Hui，A Java/CGI approach to developing a geographic virtual reality toolkit on the Internet， Computers and Geosciences，Vol: 28, Issue: 1, February, 2002

[35] Emmen, Ad; Mulder, Martijn; Barosan, Ion，Supporting 3D and VR applications in a metacomputing environment， Future Generation Computer Systems， Vol: 14, Issue: 3-4, August, 1998

[36] JAVA 3D API SPECIFICATION, JavaSoft, A SUN Microsystems, inc, business, version 1.0, August 1, 1997

[37] Baerten Herbert Van Reeth Frank，Using VRML and JAVA to visualize 3D algorithms in computer graphics education, Computer Networks and ISDN Systems，Vol: 30, Issue: 20-21, November 12, 1998

[38] Nikishkov, G.P. ，Generating contours on FEM/BEM higher-order surfaces using Java 3D textures，Advances in Engineering Software，Vol: 34, Issue: 8, August, 2003

[39] Wang, Lihui; Sams, Ryan; Verner, Marcel; Xi, Fengfeng，Integrating Java 3D model and sensor data for remote monitoring and control， Robotics and Computer-Integrated Manufacturing， Vol: 19, Issue: 1-2, February - April, 2003

[40] Chad F. Salisbury Steven D. Farr Jason A. Moore. Web-based simulation visualization using java3d，1999 Winter Simulation Conference (WSC' 99)，vo1.2, Phoenix, AZ, USA, Decembers-8, 1999.

[41] Lihui Wang Brian Wong Weiming Shen Sherman Lang. A web-based collaborative workspace using java 3d，The 6th International Conference on Computer Supported Cooperative Work inDesign (CSCWD 2001)，Ontario, Canada, July 1214, 2001.

[42] Huang, Bo， Web-based dynamic and interactive environmental visualization， Computers, Environment and Urban Systems， Vol: 27, Issue: 6, November, 2003

[43] 潘志庚 马小虎 石教英 虚拟现实中多细节层次模型自动生成技术综述 中国图象图形学报 1998

[44] 陈彦云 高度复杂场景真实感绘制技术的研究 中国科学院软件研究所 2000

[45] It Just Works Technology for Online 3D Professionals http://www.eyematic.com/products_shout3d.html

[46] Henrry A. Sowizral(Organizer) David R. Nadeau University of Califormia at San Diego An Introduction to Programming AR and VR Applications in Java3D

[47] Yiming Zhao, Lijun Guo, Xialli Wang, Zhigeng Pan A W3D Virtual Shopping Mall That Has the Intelligent Virtual Purchasing Guider and Cooperative Purchasing Functionalities Department of Computer Science & Technology, NingBo University, China State Key Laboratory of CAD&CG, Zhejiang University Hangzhou, China

[48] Fabio R. Miranda, Carlos S.dos Santos, Joao E. Kogler JR Graphic Simulation for Virtual Prototyping with Java3D Faculdade SENAC-Rua Tito CEP 05051-000 Sao Paulo, SP, Brasil

[49] 罗亚波.基于 INTERNET 的图象与建模相结合的虚拟现实关键技术研究 博士学位一论文，武汉理工大学，2002.3

[50] 杨键，耿卫东，潘云鹤.基于图像的虚拟景观漫游.计算机辅助设计与图形学报 2001.3

[51] J su, Z.Y Zhauang, S. D. Lee, J.R. Wu and Mouhyoung Photo VR: An Imaged-Based Panoramic View Environment walk-throuhg. System proceedings of IEEE International conference on consumer Electronic (ICCE'97) 1997

在学研究成果

1、吕锋，葛顺，the Study of Image Modeling in the Digital City Based on Distributed Computation，DCABES2004 国际会议论文集收录，2004 年 9 月

致　谢

本文是在我的导师吕锋教授的悉心指导下完成的。从论文的选题、研究思路和确定、论文的撰写到修改的整个过程中，吕老师都倾注了大量的心血和精力。在三年的硕士研究生学习和生活期间，吕老师对我的悉心教诲和无微不至的关怀，使我不仅学到了专业知识和专业技能，更学到了严谨的工作作风和科研工作上一丝不苟、实事求是的科学态度。吕老师认真求实的作风和积极向上的人生态度永远是我的典范和楷模，在此我谨向吕锋教授表示最衷心的感谢和崇高的敬意！

在我三年的硕士研究生学生和生活中，武汉理工大学信息工程学院的领导，老师也给了我很大的帮助，在此表示深深的谢意！还要感谢同窗三年的同学们，尤其是实验室里的周亮、郑继伟、章懿、郭颖丽、易晓峰、张家明、曾华荣等师兄弟姐妹，是这个和谐的，友爱的大集体不断的鼓励和帮助给予我科研和工作的激情和动力。

同时，我还要对我的父母致以最衷心的感谢，是他们的爱和无私的帮助鼓励我顺利地完成学业！

最后，还要衷心感谢各位答辩委员在百忙中抽出时间来审查和听取答辩，并提出宝贵的意见，谢谢！