西南交通大学323实验室工程

西南交通大学“323实验室工程”系列教材

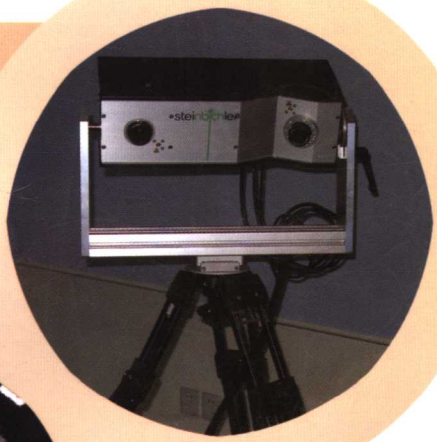
机械基础实验教学示范中心系列实验教材

虚拟现实和逆向工程技术 实验教程

主 编 王培俊 高 明

主 审 西南交通大学实验室及设备管理处

XUNI XIANSHI HE NIXIANG GONGCHENG JISHU SHIYAN JIAOCHENG



西南交通大学出版社
Http://press.swjtu.edu.cn

西南交通大学“323实验室工程”系列教材
机械基础实验教学示范中心系列实验教材

虚拟现实和逆向工程技术实验教程

主编 王培俊 高 明
主审 西南交通大学实验室及设备管理处

西南交通大学出版社
·成 都·

内 容 简 介

本书是西南交通大学“323 实验室工程”系列教材中，四川省机械基础实验教学示范中心的系列实验教材之一。全书由两部分内容组成：虚拟现实技术、逆向工程技术。

本书可作为高等学校机械工程、土木工程、材料工程等专业本科生的机械设计基础课的实验教材，也可作为本科生工程实践环节的指导教材，还可供从事机械设计教学实验的技术人员参考。

图书在版编目（CIP）数据

虚拟现实和逆向工程技术实验教程 / 王培俊，高明主编. —成都：西南交通大学出版社，2006.11
（西南交通大学“323 实验室工程”系列教材）
机械基础实验教学示范中心系列实验教材
ISBN 7-81104-428-5

Ⅰ. 虚… Ⅱ. ①王…②高… Ⅲ. 虚拟技术－应用－机械设计－高等学校－教材 Ⅳ. TH122

中国版本图书馆 CIP 数据核字（2006）第 096675 号

西南交通大学“323 实验室工程”系列教材
机械基础实验教学示范中心系列实验教材

虚拟现实和逆向工程技术实验教程

主编 王培俊 高 明

*

责任编辑 李晓辉
封面设计 何东琳设计工作室

西南交通大学出版社出版发行
（成都二环路北一段 111 号 邮政编码：610031 发行部电话：028-87600564）
http://press.swjtu.edu.cn
四川森林印务有限责任公司印刷

*

成品尺寸：185 mm×260 mm 印张：5.75
字数：139 千字 印数：1—3 000 册
2006 年 11 月第 1 版 2006 年 11 月第 1 次印刷
ISBN 7-81104-428-5
定价：8.00 元

前　言

西南交通大学机械基础实验教学示范中心自建立以来，坚持整合优势资源，实施精品化战略。经过一系列的改革与实践，逐步形成了以高素质创新人才和个性化人才培养为目标，以创新与实践能力培养为核心，以提高科学实验素质和工程实践能力为主线的新的机械基础实验教学体系，开设了能满足不同层次教学要求的实验课程，构建了以基本型实验模块、综合设计型实验模块及研究创新型实验模块为架构的机械基础实验教学平台。

针对21世纪高素质创新人才和个性化人才的培养要求，按照夯实基础、拓宽视野、及时反映现代科学技术发展与进步的原则，本书旨在把新的科学技术和现代设计方法引入本科教学，引入实验教学，为本科生开设前沿性感知实验以及个性化开放型实验。

全书分为两部分，上篇为虚拟现实技术，由王培俊编写，共六章，主要内容有虚拟现实技术的主要特征与系统组成、主要类型和关键技术、常用的软件平台及特点、虚拟现实环境下的产品设计与制造技术概述、各种输入输出设备以及实验报告等。这部分还结合自主研制开发的虚拟手机外形设计系统、虚拟售票机、水泵的虚拟装配仿真系统等多个虚拟现实系统，介绍了虚拟设计在工程中的应用。下篇为逆向工程技术，由高明编写，共五章，包括逆向工程技术概况、实验设备介绍、数据获取及处理技术、模型重建技术、实验步骤与实验报告。

本书是西南交通大学机械基础实验教学示范中心的系列实验教材之一，也是西南交通大学“323实验室工程”系列教材的组成部分。在编写过程中，得到了西南交通大学机械工程学院罗大兵老师的帮助，在此表示感谢。

由于编者水平有限，加以时间仓促，书中不妥之处，诚请批评指正。

编　者

2006年8月

目　录

上篇　虚拟现实技术

下篇　逆向工程技术

上篇

虚拟现实技术

第一章　虚拟现实技术概况

第一节　背景与目的

一、背　景

在经济全球化趋势影响下，作为国民经济发展重要支柱之一的制造业面临着严峻的竞争与挑战。其中，新产品的快速响应与开发是企业提高市场竞争能力的关键环节。自动化技术、计算机应用技术、信息技术、网络通信技术的快速发展，为实施先进的设计制造技术提供了有力的支持。

传统的产品开发过程通常是在产品制造出来以后，再对产品的外观、功能及装配性能进行分析，从而暴露出产品在造型、性能、工艺、装配可行性等方面存在的问题；然后根据存在的缺陷，对产品的设计进行修改。然而设计上的任何改动都将导致产品实物模型重建，导致原材料浪费和开发成本的增加，同时也延长了产品的开发周期。

实践证明，将信息技术应用于制造业的开发过程，是现代制造业发展的必由之路。以信息技术为依托，以协同设计、并行设计、虚拟设计、定制设计等为代表的先进设计技术，使产品的开发有了新的突破，代表了21世纪设计理论与实践的重要发展趋势。

在经济全球化趋势下，以数字化描述为基础的产品虚拟设计与制造日益重要。数字化制造在产品的设计和制造中进行两个层次的虚拟过程：一是设计阶段，用户能够参与建立数字化产品模型，实现高交互和沉浸式并行开发的虚拟环境，从而达到预先体验产品性能的目的；二是在制造系统层次上，强调对生产制造性能进行有效且近乎实时的评价。数字化制造通过虚拟运作可以实现事先风险评估、实时运筹调度和全局优化。

虚拟现实（Virtual Reality，简称VR）技术的投入性和交互性可以很好地帮助产品的设计。VR可将各种影响因素集成在设计过程中，从而减少用于验证设计概念所需的模型个数。在设计过程的各个阶段，可以用仿真系统来验证假设，减少费用和制模时间，同时满足产品的规格多样性要求。特别需要指出的是，网络化环境下开发VR系统，实现虚拟设计，是网络化设计的重要内容。根据国情，开发低成本的VR系统应占主导地位。在低成本基础上开展并推广网络化虚拟设计，使协同工作系统适应广大中小企业的承受能力，对于提高我国企业的整体竞争力具有重要的理论与现实意义。

二、目　的

本书的目的是通过虚拟现实与制造前沿性感知实验，将先进的计算机技术引入机械设计

的教学实践，拓宽学生的知识面，使之了解国内外最新的设计思想、设计理论与设计手段，从而有助于培养符合信息时代要求的、具有较强竞争能力的高级机械工程技术人才。

第二节　国际先进制造技术发展战略简介

自 20 世纪 90 年代开始，美国、日本、德国及其他一些欧洲国家相继制定了一系列关于制造技术发展的策略与规划，陆续开展了相关领域的研究与推广工作，取得了显著的成效。

1. 美　国

敏捷制造（Agile Manufacturing）战略。其目标是利用信息高速公路和动态联盟的组织形式，加速新产品的设计开发和制造。

① 1983 年，美国国家标准局提出了《虚拟制造单元》报告，计划在制造设计过程中实现以更大的柔性完成分析行程和时序安排，完善报告和监控等功能。

② 1989 年，麻省理工学院的《虚拟制造》报告，论述了“虚拟制造”在产品的概念设计和早期性能评价方面的优势。

③ 1993 年，爱荷华大学的《制造技术的虚拟环境》报告，提出了建立支持虚拟制造的环境，包括虚拟制造的评估系统、装配顺序计划、材料去除过程模拟以及离线编程等技术。

④ 1995 年，美国标准与技术研究所的《国家先进制造实验台的概念设计》计划，提出分散的、多节点的分散虚拟制造（DVM，即虚拟企业）概念，强调政府、企业和大学的联合。

⑤ 1997 年，美国标准与技术研究所的《使用 VRML 的制造系统建模》报告，则探讨了虚拟现实技术及其在网络上的应用。

目前，美国已经从虚拟制造的环境和虚拟现实技术、信息系统、仿真和控制、虚拟企业等方面进行了研究和开发，多数单元技术已经进入实验和完善阶段。

2. 日　本

智能制造系统 IMS（Intelligent Manufacturing System）研究计划。其重点是实现生产技术的标准化，开发出能使人和智能设备都不受操作和国界限制、彼此相互合作的高技术生产系统。

日本对虚拟制造技术的研究着重于应用，主要进行虚拟制造系统的建模和仿真，以及虚拟工厂的构造环境研究。

3. 德　国

生产 2000（Production 2000）战略计划。它重点利用信息和通讯技术，促使制造业的现代化，以及解决全球制造企业间协同所需的标准等问题。

4. 欧　洲

Esprit Ⅱ计划中的 Eruocoop 项目，就是开发能够支持分布式协同工作的系统。欧洲以大

学为中心纷纷开展了虚拟制造技术研究，如虚拟车间、建模与仿真工程等。

5. 亚　洲

亚洲国家自 1996 年起，每年召开一次“CSCW in Design”的国际会议，讨论设计中的协同工作问题。

第二章　虚拟现实技术的主要特征与系统组成

虚拟现实是一种可以创建和体验虚拟世界的计算机系统，或者说是一种基于可计算信息的视、听、触觉一体化的交互环境，是在图形学、电子显示、语音识别与合成、传感器等技术上发展起来的一门综合仿真技术。VR 技术具有 3 个重要特征：沉浸感（Immersion）、交互性（Interaction）与想象力（Imagination），简称“3 个 I”。VR 系统主要由软件环境、高性能计算机系统和输入/输出设备 3 部分组成，如图 2.1 所示。

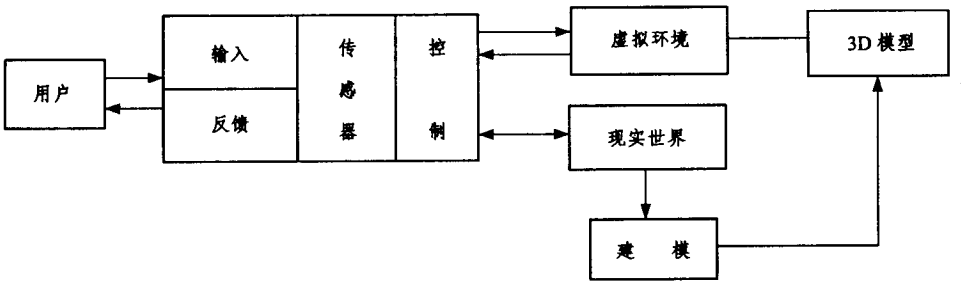


图 2.1　VR 系统的构成

第一节　软件环境

软件主要用于生成逼真的三维场景。简单的场景可采用 OpenGL、VRML 等标准直接进行开发，大而复杂的场景则最好使用专门的 VR 工具软件如 WTK、Open Inventor、dVISE、Multigen & Vega、Quest 等设计。虚拟现实系统的开发需要一定的图形库支持，这些图形库中有比较接近底层的 OpenGL，也有更高级的 WTK 和 Java 3D 等开发工具，实际使用时，应根据具体情况选用。为满足异地协同设计的需要，分布式虚拟现实技术已成为一个新的研究领域，在开发中要考虑虚拟现实系统是否能够基于网络。

一、常用三维可视化软件

常用三维可视化软件有：Cult3D®（Cycore 公司）、MultigenCreator®（Multigen- Paradigm 公司）、AtmoSphere™（Adobe Systems 公司）、WorldUp®（Sense8®公司）、MockUp®（Parametric

Technology Corporation)、3DSMax®（Autodesk 公司)、Maya®(AliaslWavefront™)、Lightwave (Newtek 公司）和 Softimage（Microsoft 公司）等。

利用 VRML、Java 3D 和 OpenGL 等开放的标准和技术，是开发低成本虚拟环境的有效途径之一。其中，VRML、Cult3D 和 Java 3D 具有鲁棒性很强的平台独立性，适用于开发基于 Web 的 VR 应用系统。本书重点介绍 VRML、OpenGL、WTK、Java 3D、Direct 3D 和 Cult3D 的特点，对 VRML 和 Java 3D 的语法进行了举例说明。

1. VRML（Virtual Reality Modeling Language）

VRML是在Internet上定义三维交互式可视化应用的技术。VRML的基本特征包括分布式、三维、交互性、多媒体集成、场景逼真性等。事实上，目前采用 VRML 技术取得成功应用的案例已经很多，例如“探路者”机器人到达火星后的信息就是利用 VRML 在因特网上即时发布的，网络用户可以以三维方式随“探路者”探索火星。

(1) VRML 概述。

VRML 用文本信息描述三维场景，通过 Internet 传输，在本地机上由 VRML 的浏览器解释生成三维场景。文本描述的信息在网络上的传输比图形文件迅速，把复杂的处理任务交给本地机，从而减轻了网络的负荷。由于浏览器是本地平台提供的，因而实现了 VR 的平台无关性。

VRML 用树状的场景图（Scene Graph）来描述三维世界。场景图不仅使得对三维世界的描述变得清晰，而且通过封装属性和建立场景图的内部消隐通道，可以很方便地实现虚拟实体的交互和动画等功能。场景图的基本元素为节点（Node），节点是 VRML 为多媒体和交互对象定义的一个对象集。单个节点描述造型、颜色、光照、视点、以及动画定时器、传感器、内插器等的定位和朝向等。

① 节点的组成：

- 节点的类型（必需）
- 一对括号（必需）
- 括号中的一定数目的描述节点属性的域（可选）和域值。

例如：

```
Sphere {
    radius 0.1
}
```

括号将节点的域信息组织在一起，在括号中的域是属于节点的。由节点及其相关域定义的造型或属性在空间中被视为一个整体。域是节点中包含的参数，事件用于参数的传递。域和事件是 VRML 节点用来定义对象性质的基本属性的。域和事件的定义包括名称、功能类型、数值类型以及缺省值。域和事件在节点中的定义没有顺序差别。从数据结构上，它们可以分成两类，其中一类只包含一个值（一个值可以是一个数、一个矢量，甚至是一幅图像、另一个点）；而另一类可以包含多个值，可以看作数组。单值的域或事件的数值类型命名，以 SF 开头，多值的域或事件以 MF 开头。

② 节点的分类（按类型）：

- 造型尺寸、外观节点：Shape、Appearance、Material
- 原始几何造型节点：Box、Cone、Cylinder、Sphere

- 造型编组节点：Group、Switch、Billboard
- 文本造型节点：Text、FrontStyle
- 造型定位、旋转、缩放节点：Transform
- 内插器节点：TimeSensor、PositionInterpolater、OrientationInterpolater、ColorInterpolator、ScalarInterpolator、CoordinateInterpolator
- 传感器节点：TouchSensor、CylinderSensor、PlaneSensor、SphereSensor、VisibilitySensor、ProximitySensor、Collision
- 点、线、面集节点：PointSet、IndexedLineSet、IndexedFaceSet、Coordinate
- 海拔节点：ElevationGrid
- 挤出节点：Extrusion
- 颜色、纹理、明暗节点：Color、ImageTexture、PixelTexture、MovieTexture、Normal
- 控制光源的节点：PointLight、DirectionalLight、SpotLight
- 背景节点：Background
- 声音节点：AudioClip、MovieTexture、Sound
- 细节层次节点：LOD
- 雾节点：Fog
- 空间信息节点：WorldInfo
- 锚点节点：Anchor
- 脚本节点：Script
- 控制视点的节点：Viewpoint、NavigationInfo
- 用于创建新节点类型的节点：PROTO、EXTERNPROTO、IS

③ VRML 中 14 个保留的关键字：

分别是：DEF、EXTERNPROTO、FALSE、IS、NULL、PROTO、ROUTE、TO、TRUE、USE、event In、event Out、exposed Field 和 field。它们不能作为自定义的域名、节点名和对象名。这些关键字的定义如下：

- DEF：给后续的节点命名，这个名字就是节点名，典型格式为：

  DEF 节点名 节点
- USE：引用 DEF 定义的节点名。典型格式为：

  USE 节点名
- TRUE：表示“真”、“1”、“是”等，用于给 SFBool 域赋值。
- FALSE：表示“假”、“0”、“否”等，用于给 SFBool 域赋值。
- NULL：表示空值，用于给 SFNode 域赋空值。
- PROTO：用于声明自定义节点的原型。典型格式为：

```
PROTO 节点名称 [
域的自定义（包括其缺省值）
事件的自定义
]{
执行体
}
```

● EXTERNPROTO：用于预解释引用的外部定义节点的原型。典型格式为：

EXTERNPROTO 节点名称 {
域的自定义（不包括其缺省值）
事件的自定义
]
外部节点的资源定位。

其中域、事件的类型和名称，必须与引用的外部节点中的定义一样。资源定位可以为 URL 或 URN 格式，当使用 "URL/URN" 或 ["URL/URN", "URL/URN", …] 的数组形式时，浏览器使用数组中第一个正确寻获的资源。

● ROUTE：构成事件通路。典型格式为：

ROUTE fromNode. fromEvent TO toNode.toEvent

其中，fromNode 为发出事件的节点的名称；fromEvent 为事件输出的名称；toNode 为接受事件的节点的名称；toEvent 为事件输入的名称

● TO：见 ROUTE 的说明。

● event In：定义事件输入，典型格式为：

event In 数值类型 事件名

● event Out：定义事件输出，典型格式为：

event Out 数值类型 事件名

● field：定义私有域，典型格式为：

field 数值类型 域名 缺省值

● exposed Field：定义公共域，典型格式为：

Exposed Field 数值类型 域名 缺省值

● IS：用于原型声明中，把自定义的域和事件与执行体中节点的域和事件等同起来。典型格式为：

执行体中节点的域或事件 IS 自定义的域或事件

④ VRML 的场景建模：

目前主要有两种方法在 VRML 中进行场景建模。对于简单的三维模型，可直接利用 VRML 内部定义的基本几何节点创建，这样得到的 VRML 文件的体积非常紧凑，常常在几十 K 左右。VRML 的基本几何节点包括：Box、Cone、Cylinder、ElevationGrid、Extrusion、IndexedFaceSet、IndexedLineSet、PointSet、Sphere 和 Text。VRML 还可以将用户自定义的节点封装、存贮在指定的库中，如 EXTERNPROTO 和 InLine 等。对于复杂的工业产品模型，仅靠 VRML 本身的建模能力是远远不够的，这些基本的节点很难或者不可能建立复杂的、具有较强真实感的三维模型，这时就需要第二种方法，利用各种 CAD 软件（如 Pro/Engineer、UGII、SolidEdge、SolidWorks、AutoCAD、CATIA 或 3DSMax、Maya 等）创建产品模型，然后导出为 VRML 格式。值得强调的是，这种从 CAD 到 VR 的过程是不可逆的，导出的 VRML 文件体积往往较大。为适应网络传输和实时渲染的要求，必须对导出的 VRML 文件进行一系列优化。除此之外，VRML 提供了两个用于动态地向三维场景增加新物体的方法：其一是用于增加简单物体的 createVrmlFromString 方法；另一个是用于增加复杂物体的 createVrmlFromURL。

⑤ 交互性的实现：

交互性是虚拟现实系统区别于传统 CAD 系统和动画的重要特征。在 VRML 中，借助传感器节点和插补器节点，可以实现各种人机交互。

传感器节点有：Anchor、Collision、CylinderSensor、PlaneSensor、ProximitySensor、SphereSensor、TimeSensor、TouchSensor 和 VisibilitySensor。插补器节点 Interpolator 主要用于创建线性的帧动画，它包括：ColorInterpolator、CoordinateInterpolator、NormalInterpolator、OrientationInterpolator、PositionInterpolator 和 ScalarInterpolator。VRML 本身动画的产生是由于变动了任何一个坐标系的位置、方向和形体比例，从而使物体按设计者所想的方式飞行、平移、旋转或按比例缩放。

VRML 的绑定指令描述如何将节点绑定在一起。VRML 绑定包括：绑定在一起的节点和在节点之间绑定的路由（ROUTE，或称为路径）。绑定两个节点之后，第一个节点通过这样的路径传送给第二个节点的信息叫做事件（Event），事件包含一个值。当一个节点接收到一个事件时，它将根据节点的特征开始动画或者其他事情。通过绑定多个节点，用户可以创建许多路由，从而使空间更加具有动感。要实现交互行为，需要给对象附带一个传感器，该传感器使用一个定点设备（例如鼠标）来感知观察者的动作（如移动、点击和拖动）。当观察者点击到一个附带有传感器的造型时，传感器就输出一个事件，这个事件就被路由到其他的节点来引起造型的移动和动画的播放。

大多数 VRML 节点都可以绑定在线路上，每个节点都有输入插座 eventIn 和输出插座 eventOut。一些节点同时具有输入、输出插座，而另外的一些节点仅有其中的一种。当链接一个路由时，eventIn 接受输入，eventOut 将事件输出。例如，Group 节点的语法如下：

```
Group{
Children          [ ]                  #exposedField    MFNode
bboxCenter        0.0  0.0  0.0        #field           SFVec3f
bboxSize          -1.0 -1.0 -1.0       #field           SFVec3f
addChildren                            #eventIn         MFNode
removeChildren                         #eventOut        MFNode
}
```

该节点中的 addChildren 是输入插座，removeChildren 就是输出插座。上述节点中的 children 域是一个 exposedField 域，它拥有两个隐含插座：一个用来设置它的值，另一个用来传送域值。隐含的输入插座总是以 set_XXX 命名，其中 XXX 是 exposedField 域的名称。与此相似，隐含的输出插座是以 XXX_changed 命名。在这里，children 域隐含的插座就是 set_children 和 children_changed。当创建路由时，使用节点的 eventIn 和 eventOut 的名称来设定链接路由的地方。对于上述节点来说，创建路由时链接到 addChildren 和 removeChildren 上。对于一个 exposed 域，可以绑定一个路由至隐含的 eventIn 和 eventOut，也可以直接绑定至 exposed 域本身。对于 Group 节点来说，即绑定至 children 域。

节点的每个输入/输出路由也有类型。例如一个 SFFloat 类型的 eventOut，当它绑定一个路由时，输出浮点数。SFFloat 类型的 eventIn，能够接收浮点值。创建路由之后，线路路由将处于睡眠状态，直到有一个事件从发送节点发送到接收节点。接收节点接收事件之后将作出反应，反应类型依赖于：

- 接收事件节点的类型
- 路由所绑定的节点输入插座
- 事件所包含的数值
- 当前节点的活动状态

在各种传感器节点中，TimeSensor 节点就像一个时钟，用来执行开始、停止或者其他控制动画的动作。随着时间的流逝，这个传感器就会产生事件来表示时间的变化。这些变化通过 TimeSensor 节点的 eventOut 路由到其他节点，从而使这些节点发生相应的变化。如果要使一个坐标系平移、旋转和按比例缩放的话，应将 TimeSensor 节点事件路由至 PositionInerpolator 和 OrientationInterpolator 节点。这些节点中就产生新的位置和旋转值，并通过他们的 eventOut 传送这些值。按顺序将这些值路由到 Tranform 节点，就可以使节点的坐标系随动画过程的发展而平移、旋转或按比例缩放。TouchSensor 是一种用来检测观察者的接触和将事件输出的传感器。这些输出描述了在何时、何地，观察者接触到了可感知的造型。CylinderSensor、PlaneSensor 和 SphereSensor 节点也可用来检测何时观察者接触到一个可感知的造型，并且提供了用来改变造型位置和方向的输出。

上面是利用 VRML 本身的各种传感器节点创建交互行为，但对于更为复杂的交互操作使用它本身所提供的传感器是远远不够的，这就需要通过 VRML 与 Java 之间的交互控制和信息传递，利用 Script 和 Proto 节点等对其进行功能上的扩展，最好是利用 Java 与 VRML 的外部编程接口界面 EAI 技术完成这些扩展工作。EAI 定义了一系列可由外部环境访问的 VRML 浏览器函数（Java 类），允许外部环境利用 VRML 的事件机制访问三维场景中指定的节点。通过 EAI 的合理设计，用户能够借助普通的二维鼠标和键盘对三维场景进行动态实时操纵，从而在外部直接操作、控制和修改 VRML 世界内部的场景。

（2）VRML 的文件结构和应用示例。

VRML 文件结构分为 4 个主要部分：文件头、原型、场景（造型和脚本）和路由。并不是所有的文件都包括这些要素，唯一不可缺少的是 VRML 文件头。文件头为：#VRML V2.0 utf8，它有 3 个含义：① 表明这个文件是一个 VRML 文件；② 文件符合 VRML 2.0 版本；③ 文件使用的是 utf8 字符集，这是多种语言中键入字符的一种标准方式。VRML 文件以扩展名.wrl 或.wrz 结尾，表示这是一个包含 VRML 空间的文件。

下面是用 VRML 实现交互控制的一个例子，运行该程序得到的渲染场景如图 2.2 所示。在这个例子中，初始场景是两个球体，一个呈红色，尺寸较小，另一个呈蓝色，尺寸较大。如果用户用鼠标点击红球，第一次点击后，蓝色球体变为圆锥体，再点击，圆锥体变为圆柱体，第三次点击后，圆柱体又变为立方体。整个过程就好像下面的红色球体是一个控制开关，控制上面的几何形体的依次变化。

```
#VRML V2.0 utf8      # VRML 的声明
WorldInfo {      # 程序说明，不影响显示效果
  title "The Use of TouchSenser"
  info ["VRML Interaction Example"]
}
Background { skyColor 0.4 0.5 0.6 }      # 设置场景的彩色背景
```

```
DEF CYCLE Script {      # 定义 Script 节点，借助 JavaScript 进行交互控制
  eventIn SFTime touchTime
  field SFInt32 number 4
  eventOut SFInt32 output
  url "javascript:
    function initialize() {      # 利用 JavaScript 的基本函数 initialize( ) 进行初始化设置
      output = 0;
    }
    function touchTime(value, time) {
      if (output == number - 1) output = 0;
      else ++output;
    }
  "
}

Transform {
  translation 0 -.5 8
  children [      # 设置两个子节点
    DEF SENSOR TouchSensor {      # 第一个子节点是接触传感器 TouchSensor
    }
    Shape {      # 第二个子节点定义半径为 0.1 的球体
      appearance Appearance {
        material Material {
          emissiveColor 1 0 0      # 定义球体为红色
        }
      }
      geometry Sphere {
        radius 0.1
      }
    }
  ]
}

DEF SWITCH Switch {      # 定义开关节点 Switch，以便在不同的几何体之间切换
  whichChoice 0      # 切换操作开始前显示初始场景
  choice [      # 定义依次切换的场景
    Shape {      # 定义切换的场景一：蓝色立方体
      appearance DEF BLUE Appearance {
        material Material {
          diffuseColor 0 0 1
```