分类号： 学校代号：11845

UDC： 密级： 学 号：211140308

**广东工业大学硕士学位论文**

（工程硕士专业学位）

基于Delta3D三维虚拟实验室的虚拟仪器

的设计与实现

张翔霄

指导教师姓名： 刘东峰教授

学科（专业）或领域名称： 电子与通信工程

学生所属学院名称： 信息工程学院

论文答辩日期： 2017年5月

A Dissertation Submitted to Guangdong University of Technology for the Degree of Master

（Master of Engineering）

Design and Implementation of Virtual Instrument

Based on Delta3D Virtual Laboratory

Master Candidate: zhangxiangxiao

Supervisor: LiuDongFeng

May 2017

School of Information Engineering

Guangdong University of Technology

Guangzhou, Guangdong, P.R.China, 510006

# 摘 要

随着[虚拟实验](http://baike.baidu.com/view/1668896.htm)技术的不断成熟，虚拟实验室在教育领域的应用价值也越来越高，它一方面可以辅助学校的科研工作，另一方面在实验教学中也具有交互性，可展示性，使学生易理解，具有趣味性，易维护等优点。虚拟实验室相对于传统的实验室，最大的特点在于它是与真实的实验环境相分离的，与此同时又给实验操作员一个身临其境的实验平台，操作员可以在实验室中随心所欲的创建各种实验，同时又不会给生产环境带来风险。因此，建立一套广泛应用的虚拟实验室已被很多高校列入计划当中。其中，虚拟仪器及虚拟仪表的设计，作为执行实验的部件，用于展现一个完整的实验。而界面作为人机交互的门面，操作员通过界面上的文字描述及仪器列表可以轻松地选择仪器进行组装实验，从而达到教学的目的。

本文采用游戏开发引擎Delta3D搭建整个虚拟实验系统，其中QT作为一个开源的GUI库与Delta3D的API进行整合，使用QT来进行与仪器相关的二级界面的开发，利用QT的信号-槽机制进行界面功能按键的实现，同时使用3DS MAX模型制作软件来进行三维虚拟仪器的设计，并使用Delta3D的扩展库对仪表盘编码实现。本文设计了一套中学电子课程方面的虚拟仪器，包括电流表，电压表，小灯泡，开关，电源，导线，万用表，电阻电容等部件。

最后，本文利用开发完成的仪器编程实现了一个中学虚拟实验并进行演示，利用本虚拟实验系统，可以进行扩展，只需要根据需求进行相关的模型设计即可编码实现。本平台具有很强的教育意义及使用价值，可以应用到中学的物理课程之中，不仅可以展现电子实验，还可以展现光学实验，力学实验等。

作为课堂教学的补充，本系统提供给学生更多的是对物理实验的认知感，普及基础教育并且激发学生的学习兴趣。同时也解决了许多实际问题：一些中小学实验经费不足，实验设备质量不高、数量不足等问题。学生通过虚拟实验教学，更容易接受课堂中的知识点，并且可以得到迅速的反馈。

关键词：虚拟仪器；Delta3D；QT；

# Abstract

With the continuous mature of virtual experiment technology, the value of virtual laboratory in the field of education is getting higher and higher. On the one hand ,it can assist the work in school, on the other hand, it has some advantages in teaching such as interactivity, displayability, easy to understand, interesting, easy to maintain, etc. Different from the traditional laboratory, the biggest feature virtual laboratory has is that its design and implementation is isolated from the production environment, but it still gives the operator an immersive experimental platform, operator can create experiments at will without risking the production environment. Therefore, many colleges and universities plan to establish virtual laboratories. The virtual instrument, as the implementation of experimental component, has been used to show a complete experiment. GUI, as the façade in human-computer interaction, can be used to do experiments by selecting instrument icon, and can be used to show some experimental description. Through this way, we can achieve the purpose of teaching.

This article uses the game development engine Delta3D to build the whole virtual experiment system. QT, as an open source GUI library, is integrated into this system with Delta3D's API, and used to develop instrument-related secondary interface. The signal-slot mechanism can help to implement interface buttons. At the same time, this article uses 3ds MAX to carry out three-dimensional virtual instrument design. This article designs a set of instrumental models for high school students in their electronic classes, including ammeter, voltmeter, small bulb, switch, power supply, wires and other components. Students can use these in their classes.

Finally, this article implements one high school virtual experiment and displays it, the virtual experimental system can be extended, only need to design models, in this way, we can realize experiments we want. This platform has a high educational value, and can be applied in the high school physics courses, it not only can show electronic experiments, but also can show optical experiments, and other experiments.As a supplement to classroom teaching, this system gives students more awareness of physical experiments, popularizes basic education and stimulates students' interest in learning. At the same time, it also solves a lot of practical problems, such as the shortage of experimental funds in primary and secondary schools, the poor quality of experimental equipment, the lack of quantity and so on.

Through the virtual experiment teaching, students are more receptive to the knowledge points in the classroom, and can get quick feedback.

**Keywords**: virtual instrument; Delta3D; QT;

# 目录

[摘 要 I](#_Toc479786717)

[Abstract II](#_Toc479786718)

[目录 IV](#_Toc479786719)

[Contents VI](#_Toc479786720)

[第一章 绪论 1](#_Toc479786721)

[1.1 课题研究背景、目的及意义 1](#_Toc479786722)

[1.1.1课题的研究背景 1](#_Toc479786723)

[1.1.2课题的研究目的 1](#_Toc479786724)

[1.1.3课题研究的意义 2](#_Toc479786725)

[1.2虚拟实验室的发展现状 2](#_Toc479786726)

[1.2.1国外研究发展现状 2](#_Toc479786727)

[1.2.2国内研究发展现状 3](#_Toc479786728)

[1.3论文的主要内容与结构安排 4](#_Toc479786729)

[1.4本章小结 5](#_Toc479786730)

[第二章 实验界面的设计及结果展示 6](#_Toc479786731)

[2.1界面的总体设计 6](#_Toc479786732)

[2.1.1界面设计的原则 6](#_Toc479786733)

[2.1.2教育类实验室界面分析 7](#_Toc479786734)

[2.2 GUI程序开发框架-QT 9](#_Toc479786735)

[2.2.1 QT概述 9](#_Toc479786736)

[2.2.2 QT的信号与槽机制 12](#_Toc479786737)

[2.3 界面实现的步骤 12](#_Toc479786738)

[2.3.1 界面平台的搭建 12](#_Toc479786739)

[2.3.2界面模块的实现 14](#_Toc479786740)

[2.4 结果展示 19](#_Toc479786741)

[第三章 三维虚拟仪器的设计 23](#_Toc479786742)

[3.1建模工具的选择 23](#_Toc479786743)

[3.1.1三维建模工具3DS MAX介绍 23](#_Toc479786744)

[3.1.2 主流建模软件的异同 24](#_Toc479786745)

[3.2模型设计的基本准则 25](#_Toc479786746)

[3.3模型的结果展示 26](#_Toc479786747)

[3.4利用dtGauge制作仪表盘 33](#_Toc479786748)

[3.4.1 dtGauge概述 33](#_Toc479786749)

[3.4.2仪表盘的编程实现 33](#_Toc479786750)

[3.5 dtGauge的优势与不足 38](#_Toc479786751)

[第四章 三维虚拟实验 40](#_Toc479786752)

[4.1 Delta3D游戏引擎的使用 40](#_Toc479786753)

[4.1.1 Delta3D概述 40](#_Toc479786754)

[4.1.2 Delta3D的实用工具 42](#_Toc479786755)

[4.1.3 Delta3D的编译 43](#_Toc479786756)

[4.2 Delta3D中的游戏角色 44](#_Toc479786757)

[4.2.1游戏角色的实现 44](#_Toc479786758)

[4.2.2角色实现的关键问题 50](#_Toc479786759)

[4.3 虚拟实验演示 51](#_Toc479786760)

[总结与展望 55](#_Toc479786761)

[参考文献 56](#_Toc479786762)

[学位论文独创性声明 59](#_Toc479786763)

[学位论文版权使用授权声明 59](#_Toc479786764)

[致谢 60](#_Toc479786765)

# Contents

[Abstract II](#_Toc479786718)

[Contents VI](#_Toc479786720)

[Chapter I Introduction 1](#_Toc479786721)

[1.1 Research background, purpose and significance 1](#_Toc479786722)

[1.1.1 Research background 1](#_Toc479786723)

[1.1.2 The purpose of the research 1](#_Toc479786724)

[1.1.3 Significance of research 2](#_Toc479786725)

[1.2 The development of Virtual Laboratory 2](#_Toc479786726)

[1.2.1 Foreign research and development 2](#_Toc479786727)

[1.2.2 Domestic research and development 3](#_Toc479786728)

[1.3 The main content and structure of the paper 4](#_Toc479786729)

[1.4 Summary 5](#_Toc479786730)

[Chapter II Design and results of the experimental interface 6](#_Toc479786731)

[2.1 The overall design of the interface 6](#_Toc479786732)

[2.1.1 Principles of interface design 6](#_Toc479786733)

[2.1.2 Interface analysis of Educational Laboratory 7](#_Toc479786734)

[2.2 GUI program development framework-QT 9](#_Toc479786735)

[2.2.1 QT overview 9](#_Toc479786736)

[2.2.2 Signal and slot mechanism of QT 12](#_Toc479786737)

[2.3 Interface implementation steps 12](#_Toc479786738)

[2.3.1 Interface platform construction 12](#_Toc479786739)

[2.3.2 Implementation of interface modules 14](#_Toc479786740)

[2.4 Results show 19](#_Toc479786741)

[Chapter III 3D virtual instrument design 23](#_Toc479786742)

[3.1 Selection of modeling tools 23](#_Toc479786743)

[3.1.1 3DS MAX introduction 23](#_Toc479786744)

[3.1.2 Comparison of mainstream modeling software 24](#_Toc479786745)

[3.2 Basic principles of model design 25](#_Toc479786746)

[3.3 Model display 26](#_Toc479786747)

[3.4 Making instrument panel with dtGauge 33](#_Toc479786748)

[3.4.1 dtGauge overview 33](#_Toc479786749)

[3.4.2 Programming of instrument panel 33](#_Toc479786750)

[3.5 Advantages and disadvantages of dtGauge 38](#_Toc479786751)

[Chapter IV 3D virtual experiment 40](#_Toc479786752)

[4.1 The use of Delta3D game engine 40](#_Toc479786753)

[4.1.1 Delta3D overview 40](#_Toc479786754)

[4.1.2 Delta3D utilities 42](#_Toc479786755)

[4.1.3 Delta3D compile 43](#_Toc479786756)

[4.2 Game characters in Delta3D 44](#_Toc479786757)

[4.2.1 Game characters realization 44](#_Toc479786758)

[4.2.2 Key issues in character realization 50](#_Toc479786759)

[4.3 Virtual experiment show 51](#_Toc479786760)

[Summary and outlook 55](#_Toc479786761)

[References 56](#_Toc479786762)

[Dissertation original statement 59](#_Toc479786763)

[The dissertation use copyright authorization statement 59](#_Toc479786764)

[Acknowledgement 60](#_Toc479786765)

# 第一章 绪论

## 1.1 课题研究背景、目的及意义

### 1.1.1课题的研究背景

随着计算机技术的飞速发展，虚拟实验室的建设作为教育发展的方式之一，逐渐成为国内外远程教育和实验教学领域的热门话题。成熟的虚拟实验室平台，需要的技术是多方面的，包括界面的美化与设计，三维模型的构建，以及平台代码的开发与功能的实现。首先，人与机器交互通信是通过界面来完成的，界面功能的实现不仅是软件设计和艺术美感相结合的产物，而且融入了心理学、社会学等学科的成果。而三维模型的构建，改变了以往平面的2D实验室的现状，由各种工具所构建的三维模型，可以用在虚拟现实技术所创建的虚拟实验场景中，用户在虚拟场景中完成预定实验所产生的结果与在真实的场景中完成相同实验所产生的结果相比，两者是基本等效的，前者甚至还要优于后者[1]。在实物实验和理论研究之外，虚拟实验用来作为两者的补充，它解决了许多现实存在的问题，比如实验设备质量不高、数量不足等问题，同时还具有成本低，效率高，沉浸性，创新型等特点。

### 1.1.2课题的研究目的

本课题主要是在原有实验平台的基础之上进行升级改造，并利用升级之后的实验平台设计了一个三维虚拟实验，该系统结合了教育性和游戏性等特性，并且具有直观的展示效果，寓教于乐，激发学生在课堂中的学习兴趣，并提高学生的探究精神，无论是老师还是学生，都可以轻松地操作实验。在程序编程方面，本文使用QT在原有平台的基础之上扩展了一个二级界面，并进行界面的开发及按钮功能的实现。在模型方面，使用3DS MAX模型制作软件设计了一套三维模型，并编写Delta3D的游戏角色及游戏角色代理，将模型与游戏角色代理绑定在一起作为Delta3D的动态角色层(DAL)，Delta3D就可以将其载入三维世界中并将其与界面按钮整合在一起。

最后，本文在上述功能已经实现的基础之上，设计了一个简单的中学虚拟实验并进行仪器功能的实现，此实验平台可以进行横向扩展。

### 1.1.3课题研究的意义

本文以游戏探究型学习模式为基础，在游戏软件的基础上，建立了一套实用有趣的三维虚拟实验，克服了传统实验室的弊端，具有很高的趣味性及灵活性，因此，三维虚拟实验室将在许多实际场景中得到广泛的使用，虚拟实验室的创建将使实验教学发生革命性的变化[2]。本文利用Delta3D游戏仿真引擎所开发的虚拟实验平台，与普通平面虚拟实验室最大的区别是它提供给实验操作人员一个身临其境的仿真环境，具有很强的视觉效应和沉浸感。而与其它的三维虚拟实验平台相比，本系统具有的最大特点就是开源性，其源代码是可见的，Delta3D将一些开源软件合为一体并不断地经过世界各地的组织和个人的升级改造和功能补充，与售价高昂的商业游戏引擎相比具有很大的优势[3]。本文运用QT设计界面的功能按钮，并和设计完成的游戏角色绑定起来，通过这些开源的、跨平台的工具进行代码开发，使得本虚拟实验室具有很强的扩展性和可靠性。

## 1.2虚拟实验室的发展现状

当前，网络中存在大量的软件项目和编程学习资源，这些资源有些由个人提供，而有些则是由开源的组织所提供，它们为虚拟实验项目的创建提供了必要的物质基础。而在计算机虚拟实验室蓬勃发展的十几年期间，各国均有科研团队投身虚拟实验室的研究和设计[4]。总之，虚拟实验借助了计算机CPU的强大运算性能，在进行实验演示的同时学生们可以加深对相关知识点的理解，然后就会激发他们动手实践的兴趣[5]。虚拟实验室带给我们崭新的教育思维和广阔的应用领域。

### 1.2.1国外研究发展现状

目前，在发达国家中，虚拟实验室的建设已经有了显著的进展，很多理工类高校以计算机为依托平台，模拟一系列的复杂操作，然后进行研究和总结，且已经有了很大的成效[6]。

例如，美国俄勒冈大学物理系主办的VLAB物理实验网站，该网站中设计了几十种虚拟物理实验，包括热学、天体物理、电磁学等。该系统是B/S架构的应用程序，用户打开浏览器连接到该实验室网站并进行实验演示时，需要先从服务器端下

载运行程序相关的插件。

由德州休斯顿大学和约翰逊太空中心联合建立的VETL实验室，该实验室致力于将虚拟现实技术应用在科学、教育、工程和数据可视化等领域[7]。主要成果有：（1）开发了一套强大的软件工具，非专业的设计师可以利用这些工具来创建一个多感官的、连接到一个特定硬件上的三维环境。（2）为分散在世界各地的军事人员开发了一套虚拟现实训练系统和培训系统。（3）通过研究虚拟现实在教育方面的运用，来确定虚拟现实在教育系统方面的设计原则，该实验室的主要目标是创建一个高度互动的智能虚拟环境[8]。

印地安那州立大学研发的MBL化学实验室，化学专业的学生可以在该实验室中做如下工作：（1）用该实验系统操作、演示虚拟实验，并收集、分析实验结果。（2）在网络上发布相关的实验材料和用户说明手册，如文档、图片、PPT、视频等[9]。

### 1.2.2国内研究发展现状

目前，国内相关的组织和个人也看到了虚拟实验的广阔前景，并在其中投入了大量的人力和财力。虚拟实验尤其是在医学、化学、电子信息等教学领域中，起到了举足轻重的作用。

中国农业大学开发的土壤—作物虚拟实验室系统，该实验室系统通过对虚拟作物的信息采集，可以对作物生长与土壤的化学元素含量、水分之间的相互作用进行定量化分析研究，并以3D数字图形化技术展现实验结果。该实验在某种程度上可以替换实物实验，并可与实物实验进行相互比较。在一些需要昂贵实验器材以及对时间、环境非常敏感的化学实验中，该土壤—作物实验系统具有非常明显的优势[10]。

北京航空航天大学是国内最早对虚拟现实技术进行分析研究的大学之一，首先它们研究了虚拟场景中物体物理特性的表示与处理，在图形接口部分开发硬件驱动并对相关的算法进行实现。之后提供相关的开发平台，在该平台上实现了用于飞行

员训练的虚拟现实系统并实现了分布式的网络设计，之后将该系统与相关单位进行

远程连接[11]。

浙江大学计算机学院开发出了一套校园建筑场景实时漫游系统，该系统不仅可以安装在PC上，还可以转换为App安装在手机中。同时，系统功能中也引入了三

维重力感应，当用户移动手机时相应的场景也会随着手机的位移发生改变[12]。除了上述所示的机构之外，复旦大学、上海交通大学等院校也正在进行虚拟实验室的研发工作[13]。

## 1.3论文的主要内容与结构安排

本文主要研究的是三维虚拟实验室中虚拟仪器的设计及其功能的实现。首先开发出一个二级界面并实现界面上的仪器按钮，通过点击按钮实现对仪器的操作，达到人机交互的目的。之后，使用建模工具来设计仪器模型并且使每个仪器模型都与Delta3D中的游戏角色一一对应，利用Delta3D中的扩展类库来设计仪器上仪表的内容。最后，对开发出来的仪器模型进行组装，使其实现一个电学物理实验并对其功能进行测试。内容包括了GUI的设计，三维仪器及仪表盘的制作，以及实验内容的实现。本文设计与实现的三维虚拟实验具有较好的可移植性、易用性、兼容性。

论文总共四章内容，结构安排如下：

第一章，绪论。本章从总体上介绍三维虚拟实验室的相关概念及其在国内外的发展状况并阐述了虚拟实验将是未来实验教学的发展方向，从而引出了本文的主题。

第二章，实验界面的设计及结果展示。本章先概括分析了教育类界面的设计理念并介绍了界面的开发工具QT。之后，详细阐述了虚拟实验室中使用QT进行界面开发的步骤，包括：游戏场景选择模块、虚拟仪器选择模块等功能的实现。

第三章，三维虚拟仪器的设计。本章先对三维模型制作软件3DS MAX进行介绍，并使用此工具设计了一套与中学物理实验相关的三维仪器。之后探讨了利用Delta3D中的dtGauge扩展库对三维仪器中仪表盘的编码实现，并与主流的仪表设计软件进行比较，分析它的优点与不足。

第四章，三维虚拟实验。本章先对Delta3D仿真引擎及其游戏角色的概念进行介绍。利用C++语言面向对象的特性设计一套游戏角色类并与之前创建的仪器模型进行关联，并详细阐述了游戏角色实现过程中的关键性问题。之后，将开发出来的虚拟仪器进行组装并在系统平台下实现一个虚拟电学实验。

## 1.4本章小结

本章对全文的研究内容和总体结构进行了概述，首先论述了虚拟实验室在教育领域的背景和意义，概括了虚拟实验室在国内外的科研情况，国内外高校开发的虚拟实验室及其特点，然后论述了虚拟实验在虚拟教学中的重要作用，之后引出本文的主题：基于Delta3D虚拟实验室的虚拟仪器的设计与实现。本章重点在于交待全文整体思路，引出本文的主题。

# 第二章 实验界面的设计及结果展示

## 2.1界面的总体设计

### 2.1.1界面设计的原则

在人机交互的发展过程中，各个团队都以用户为中心作为人机交互设计的思想基础，用户需要通过一个层面即我们所说的界面，来发送操作指令控制软件的行为，用户界面是软件项目分析、设计的一个重要组成。界面的构建是美学、心理学、逻辑学等不同学科参与其中的复杂过程，作为与应用程序的内核沟通交流的唯一途径，提供方便高效的服务是它首先要考虑的问题。然而，随着产品的推广，后期维护团队应该广泛收集用户反馈意见，对收集的信息与意见针对性地调整软件的功能与界面，不断提升产品的市场竞争力。一般来说，除了以用户为核心，界面设计还应当遵守如下一些原则：

（1）遵循一致，确立标准。控件的显示风格，信息提示用语，界面的颜色和布局，这些都要遵守一定的标准，从而在整体上显示出和谐一致的风格。

（2）一个窗口一个话题。每创建一个窗口，都需要为它建立一个唯一的话题，这样当用户点击打开该窗口时对内容便可以一目了然，使用起来也更加简单方便，同时设计人员也容易进行升级维护。如果我们把多个话题放在一个窗口中，首先就会造成歧义，其次窗口的布局也会显得凌乱不堪。

（3）交互时避免突兀，转换自然。人机交互时界面的转换是紧密相连的，所以在设计时要考虑界面下一步呈现的内容是否合适，尽量避免不合时宜的内容，因为这样会让用户不知所措。

（4）适当地组合信息元素，减轻使用者的认知负担。人脑对于事物的处理和记忆是有局限性的，界面的设计的好坏直接影响人类大脑处理信息的能力和反应，界面的操作主要是靠人类的短暂的记忆，设计的界面要让用户多的去浏览信息而不是记忆信息，所以界面的设计的要合理，减少用户在进行界面操作时必须记忆的东西。

（5）对产品进行详细的调查并了解用户需求，同时考虑使用环境和时机。需要

对目标用户的喜好进行分析，邀请用户进行产品的测试并收集使用反馈，系统界面才可以得到用户的欢迎。

### 2.1.2教育类实验室界面分析

教育游戏界面是面对用户的软件界面并且会直接对学生的学习成绩产生影响。一个优秀的软件界面，不仅要以界面设计的基本原则作为理论指导，还要了解其他软件的界面设计[14]，通过对同类教育软件的GUI进行分析比较，从而取长补短，下面选择了几款具有代表性的虚拟实验室软件，分析它们的界面系统。

（1）机械虚拟仿真教学平台。该虚拟仿真实验教学管理平台是由北京润尼尔网络科技有限公司开发，该平台的目标是实现学校、地区的多种实验教学资源的整合及教学资源的管理。该平台系统采用Unity3D游戏仿真引擎开发，该系统具有操作简单，良好的交互性能等特点，适用于国内的各大高校的在校学生及一线的技术工人。

虚拟仿真实验教学管理平台中机械原理虚拟实验室界面如图2-1所示：

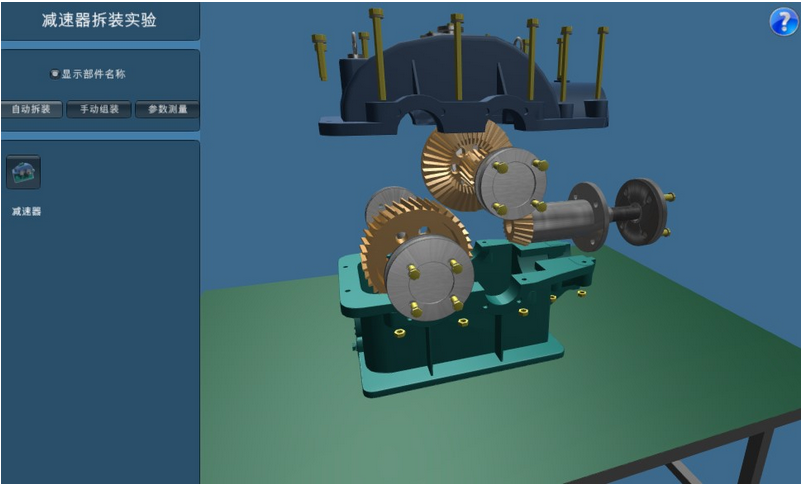


图2-1机械原理虚拟实验室

Fig.2-1 Mechanical Principle Virtual Laboratory

界面中的整个窗口由一个三维空间所占据，该三维空间同时也是用户进行实验、展示实验的“实验区”。右上角是帮助选项，如果用户在操作上有不明确的地方，可

以点此查看。左侧是虚拟实验的“工具栏”，在工具栏中，存放了构成机械的部件，

学生可以手动进行装配，也可以让它们实现自动装配，“工具栏”的顶部列出了实验的项目，并且工具栏可以手动隐藏。该软件界面系统非常直观，一目了然，用户便于直接操作。缺点则是机械中的零件由系统自动指定，不能进行自主设计的实验，学生更多的是了解流程和步骤，没有很强的参与感。

（2）仿真物理实验室。该物理实验室是一个进行电学物理实验的虚拟实验室，它提供了一些平面的电子实验元器件，包括电源、开关、电阻、电流表等，当点击相应的电子元器件时界面右方还会显示该元器件的设置说明，用户可以在工具箱中选择导线根据标准电路图来将这些电子元器件连接起来，这个过程可以类比到真实环境中的实验操作，该实验平台可以用来辅助教师的物理教学工作。

仿真物理实验室的界面如图2-2所示。

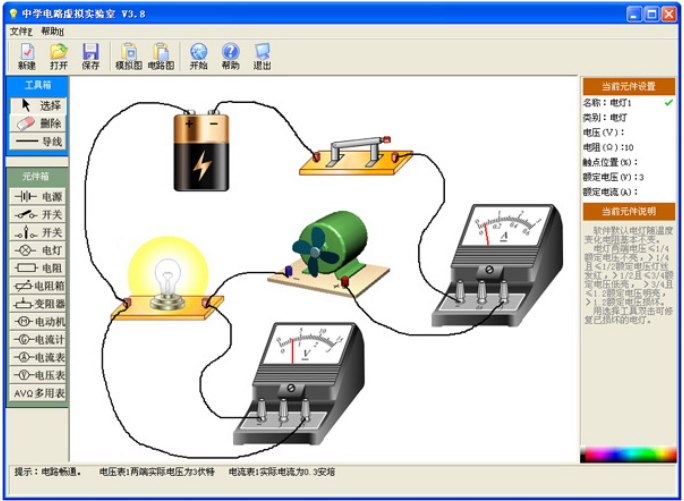


图2-2仿真物理实验室

Fig.2-2 The simulation virtual physics laboratory

此界面包含了丰富的GUI组件，它们分别为：“菜单栏”，“工具栏”，“工具箱”，“元件箱”，“实验区”，“功能说明栏”等。其中，“标题栏”位于窗口的最顶部，显示当前应用程序名、文件名等。“菜单栏”提供了应用程序的通用基本功能和操作命令，这里只有两个功能按键，分别是“文件”，“帮助”，点击之后，它们会分别产生一个二级菜单，代表了程序之中的所有功能命令。此“工具栏”功能区分别有以下

几种命令：“新建”代表在实验区中，重新建立一个实验场景。“打开”点击之后会

出现一个文件系统对话框，找到本应用程序需要的实验文件将其打开。“保存”点击之后会出现一个文件系统对话框，可以将本应用程序所产生的实验文件保存到相应的文件夹中。“模拟图”点击之后，会将实验区中的内容以图片的形式进行保存，程序所支持的图片包括bmp、jpg、gif等格式。“电路图”点击之后，程序会智能判断在实验区中用户所连接的元器件实验并根据实验自动生成标准格式的电路图。“帮助”用来查看应用程序的使用说明。“工具箱”包括了电子实验的主要操作工具，比如选择、移动、删除、导线等。“元件箱”这是物理实验室最重要的模块之一，元件箱中包含了中学物理电路实验中常用的实验元器件。“实验区”是系统完成元件放置和操作显示的界面，在这里可以观察到实验效果。“功能说明栏”对已在实验区的被选中元件的操作方法以及该元器件重要的参数、属性进行简要说明。仿真物理实验室用来对物理实验中的电学部分进行实验模拟，界面简单明了，与学生的交互性很强，缺点在于界面过于传统，缺少美感，具有很浓厚的办公软件风格[15]。

## 2.2 GUI程序开发框架-QT

### 2.2.1 QT概述

Qt是一个基于C++的GUI开发框架，它使用C++面向对象的特性并使用特殊的代码进行扩展，它不但可以为PC上的通用软件设计界面，还可以开发部署嵌入式应用[16]。Qt使用简易直观，具有很好的跨平台特性，速度快效率高且兼具艺术级的视觉展示效果。Qt于1994年开发完成，到目前为止有超过70家的企业使用Qt来开发自己的桌面应用、嵌入式移动软件，使用Qt开发的一些著名的软件如下所示：

Autodesk Maya，顶级的三维动画软件，它用于创建交互式的3D应用程序，包括视频游戏、动画电影、虚拟现实应用等[17]。

Google地球，Google地球是一个表示虚拟地图、显示地理信息的应用程序，它最初被称为EarthViewer 3D由Keyhole公司所创建[18]。

WPS Office：金山软件公司推出的可以媲美微软Office的办公软件，体积小巧功能强大[19]。

Qt在功能上强大而完备，并且具有串口驱动，可以方便地设计工控程序和终端

程序。Qt的各功能模块如图所示：



图2-3 QT基本模块

Fig.2-3 QT basic module

（1）QtCore，Qt的核心模块是QtCore，其它所有的Qt模块都依赖于这个核心模块，它提供了Qt的核心功能，包括：一个称为信号和槽的对象通信机制，对象属性的可查询、可修改，分层的和可查询的对象树组织，跨类库的动态类型转换[19]。

（2）QtGUI，QtGUI是除了核心类库之外最重要的类库，它提供了窗口系统的集成，事件处理，OpenGL和OpenGL ES集成，二维图形，基本成像，字体和文本。

（3）QtWidgets，QtWidgets模块提供了一组UI元素来创建经典的用户界面。而Widgets是创建界面的基本元素，它可以显示数据和状态信息，接收用户输入为组合在一起的其它widgets提供一个容器。它提供了渲染的基本功能并可以处理用户输入事件，Qt提供的所有UI元素要么是QWidget的子类，要么和QWidget的子

类有关联关系，通过继承QWidget类并重写虚函数来创建自定义的窗口，它们之间的关系如下图所示：

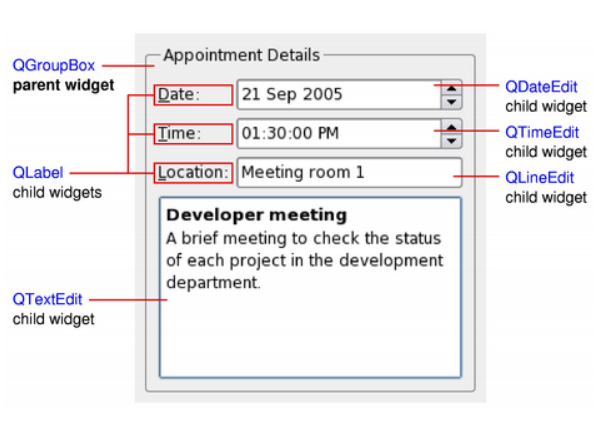


图2-4 QWidget关系图

Fig.2-4 QWidget class diagram

从图中可以看出，背景窗口是一个QWidget对象，Qt由他派生出了许多常用的子类，这些子类当中也存在父子继承关系。QGroupBox包围了所有的控件，是其它控件的父类，它提供了一个包围框，一个框顶标题，一个键盘快捷方式，在内部显示多种不同控件的功能。QLabel用来显示文字描述和图像，起到一个标签的功能。QLineEdit是一个单行的文本编辑器，用户可以输入编辑单行的无格式文本，与之相对的是QTextEdit控件，它可以用来编辑多行文本。QDateEdit和QTimeEdit用来对时间进行操作。

（4）QtMultimedia，Multimedia是Qt中的一个核心模块并提供了一组QML类型和C++类来处理媒体内容，它还提供了一套必要的API访问相机和无线电功能，包括Qt Audio引擎提供环绕立体声和内容管理[20]。

（5）QtNetwork，QtNetwork为应用程序提供了一组API来使用TCP/IP协议。网络请求、cookies、通过HTTP发送数据都有对应的C++类来处理，该模块同时也支持OpenSSL工具箱。

（6）QtQML，Qt QML提供了一个框架，在这个框架中可以使用QML语言开发应用程序和类库，它定义并实现了语言引擎的基础并提供了一个API使开发者利用自定义类型和JavaScript、C++来扩展QML语言。

（7）QtQuick，Qt Quick提供了QML的自定义类型用来创建用户界面，并提供

一个可视化的画布用来创建并驱动可视化组件[21]。

### 2.2.2 QT的信号与槽机制

信号和槽是Qt中引入的一个语言结构用于在对象之间交换信息，它在避免产生冗余代码的同时，更好的实现了观察者模式。信号和槽的概念即是GUI的窗口组件可以发送包含消息的信号，该信号可以被其它组件和控制器接收，接收信号的函数就称之为槽。它和C/C++中回调函数的概念类似，但是信号和槽机制确保了回调函数中参数类型的正确性并且很好地适配了图形用户界面的设计。同样，在非GUI环境中也可以使用信号和槽，例如：异步输入输出（套接字、管道、串行设备），事件通知，对象实例方法关联超时事件等。信号和槽很容易使用并且不需要写注册和调用代码，因为Qt的元对象编译器（MOC）会自动地产生所需的代码。

信号：就像声明一个成员函数一样，只需要给出信号名和参数列表即可。

signals:

void testsignal(const QString& parameter);

从上面可以看出，signals是信号声明的关键字。

槽：一个普通的成员函数用来接收信号，当信号触发时可以被调用。

public slots:

void testslot(const QString& parameter);

从上面的定义可以看出，slots是Qt中声明槽函数的关键字，而槽函数的形参数量是可以根据需求进行变化的，这里并不是只有一个。信号与槽的连接与触发：信号和槽的关联通过connect（）函数来实现，当发射者发出一个信号时，接收者获取到信号的信息，之后槽函数作为响应函数就会被调用。

## 2.3 界面实现的步骤

### 2.3.1 界面平台的搭建

在界面的实现过程中需要使用Delta3D游戏仿真引擎中的dtQt模块，这个模块负责处理Qt与Delta3D的集成，有关Delta3D的详细内容将在后面的章节中进行阐

述，本节中将只介绍dtQt的使用。首先使用dtQt需要相关的源代码文件及相关的库文件dtQt.lib，在VS2008开发工具的平台中需要使用界面开发工具Qt来实现三维虚拟实验室的界面系统，所以需要将Qt嵌入Delta3D虚拟环境中。首先进入Qt的下载页面下载Qt源码包，解压之后放在指定的文件目录下，打开Visual Studio中的命令提示符并进入Qt的文件目录运行configure，之后使用nmake进行编译安装，安装完成之后设置环境变量。新建用户变量QT\_ROOT，设置其值为Qt的安装路径，再修改系统变量，在Path环境变量中添加%QT\_ROOT%\bin。

Delta3D的dtQt模块用来处理Delta3D和Qt之间的整合，整合的通用开发模板如下所示：

（1）dtQt::QtGuiWindowSystemWrapper::EnableQtGUIWrapper();

QtGuiWindowSystemWrapper类的静态方法EnableQtGUIWrapper()的作用为将Delta3D的窗口作为中心控件加入到Qt的界面当中，等于是说当程序启动时，得到的是一个Qt窗口，而Qt窗口的中心部分则是Delta3D的窗口。

（2）dtCore::RefPtr<App> app = new App();

App类是自定义的继承自QObject和dtABC::Application的类，因为该类继承自dtABC::Application，所以它代表了Delta3D的应用程序本身并且包含了应用程序所需的基本组件。又因为它继承了QObject，所以该类具有了Qt中对象的基本功能，包括消息的处理以及信号与槽的功能，而RefPtr<App>代表了一个指向App对象的智能指针。

（3）dtCore::System::GetInstance().Start();

当Delta3D应用程序创建完成之后，上面的代码就是利用单例模式，得到系统中已经创建完成的应用程序，使之开始运行。

（4）dtQt::DeltaStepper stepper; stepper.Start();

这两个步骤则是用来开启Delta3D应用程序窗口的正常处理功能，包括消息事件的处理，以及窗口三维视图的刷新功能。它可以执行一个完整的系统框架和DeltaWin的处理步骤，如果想要控制整个系统的帧处理，这是非常有用的。

（5）qapp.exec();

qapp是QApplication的对象，当运行这行代码时触发了Qt中的一个槽函数，从而开始了整个Qt的事件循环系统。通过这种方式，Delta3D和Qt就绑定在了同一个窗口界面中，实现了整合。

### 2.3.2界面模块的实现

上节中，介绍了Delta3D的dtQt库使用的相关细节，本节将介绍基于系统平台下二级界面的创建以及界面的布局管理。首先，系统的主界面如图所示：



图2-5 系统平台主界面

Fig.2-5 Main interface of system platform

系统的主界面由六大模块组成，分成三块布局，分别为左部、中部、右部，每一块布局中都有上下两部分。七大模块包括：用户实验模块，这一模块主要用来显示实验人员的信息和已经添加的实验，主要是由图标和文字所组成的，对应于界面中的左上角，可以引导用户快速定位。工具模块，在这一模块中有三个选项，分别是聊天室、计算器、绘图，对应于界面的左下角。动画模块，在界面中间的下部，

主要用来动态演示实验图片的，其中设置了一个定时器可以指定间隔时间，这样每隔一段时间界面图片就会进行切换。而鼠标放上去时也会自动切换到对应图片。实验室模块，这一模块是本章所关注的重点，模块中的每一个按钮点击之后，都可以打开一个二级界面，在这个二级界面中可以进行虚拟实验设计。按钮则根据具体的

实验类别进行分类，其中包括：电学实验室、电学、光学等实验，当然也可以根据用户的需求进行扩展。实验记录及消息公告模块，这两个模块主要用于对用户的操作进行记录并显示通知，实验记录中对用户的实验操作进行了记录。

界面对象中几个重要的成员变量如下所示：

（1）dtCore::RefPtr<GameApp> \_app;

成员变量\_app是一个指向GameApp对象的智能指针，智能指针是一种抽象的数据类型，它除了普通指针的功能之外还具有其它的功能，比如自动内存管理和边界检查。而GameApp是继承自dtABC::Application的类，它代表了Delta3D的应用程序对象。

（2）QWidget\* \_centralWidget、\_leftWidget、\_rightWidget;

这三个成员代表了界面所使用的窗口，同时也可以看出页面的总体布局，界面首先有一个中央窗口，中央窗口中又分成两部分，左侧和右侧各有一个窗口。

（3）CommonGaugesWidget\* \_commonGaugeWidget；

CommonGaugesWidget代表了界面上的通用仪器模块，通用仪器模块是和特殊仪器模块进行区分的，通用仪器模块中的实验仪器，是所有虚拟实验中都需要的仪器，比如导线，而特殊仪器模块中的实验仪器，是做某一项实验时需要用到的实验仪器。CommonGaugesWidget类的声明如下所示：

class CommonGaugesWidget : public QWidget

{

……

//该类的构造函数

CommonGaugesWidget(BaseTopicAppWidget\* app,QWidget\* parent=NULL);

//和界面进行关联

BaseTopicAppWidget\* \_app;

//通用仪器模块中所使用的仪器，用来展示各种最基本的实验用具，比如直

//尺，各种线缆，量杯，天平

GaugeGUI\* \_scaleButton;

……

//通用仪器模块中的可扩展部分，可以添加自定义的仪器类型，将自定义仪

//器加入到该集合中

\_gauge\_list\_type \_gauges;

};

（4）CableEditorWidget\* \_cableWidget;

CableEditorWidget是一个继承了QWidget的类，该窗口类提供用于线缆编辑的功能，它可以根据用户的需求动态地生成线缆。

（5）std::vector<GaugeGUI\*> \_gaugeGUIs;

这个集合代表了界面中的特殊仪器模块中仪器的父类，\_gaugeGUIs是指向GaugeGUI指针的集合，而特殊仪器模块中的虚拟仪器都必须有一个相同的父类，即GaugeGUI。GaugeGUI类的声明如下所示：

class GaugeGUI: public QObject, public dtCore::Loadable

{

……

void setPos(QGridLayout\* layout,int x,int y,int xSpan,int ySpan);

//返回该类中包含的角色类型

dtCore::ActorType\* getActorType(){return \_gauge->actorType.get();}

//该类所关联的仪器

dtCore::RefPtr<Gauge> \_gauge;

//和界面进行关联

BaseTopicAppWidget\* \_app;

};

系统中用来显示一个仪表的GUI，点击该GUI调用一个仪表到系统中。目前用一个QPushButton表示，每一个GaugeGUI都是添加到一个TopicAppWidget上的。在用户选择一个子课题之前，点击这些按钮会收到提示：请选择适当的课题。

（6）std::vector<LearningProblemGUI\*> \_problemGUIs

这个代表了该BaseTopicGUI实验项目下的学习问题，该成员是一个集合对象，对象中的成员是LearningProblemGUI的指针，LearningProblemGUI中定义了一个按

钮，点击之后即可触发一个实验。这些按钮需要定义在界面中的左侧，每一个问题实际上都是该实验项目下的一个子问题，因此，点击按钮之后会为Delta3D创建一个游戏场景（map），之前所创建的游戏场景会被替换掉。当创建界面对象时，会调用对象的构造函数：

BaseTopicAppWidget::BaseTopicAppWidget(BaseTopicGUI\* topic):

QMainWindow(),\_app(new GameApp()),\_centralWidget(new QWidget(this)),

\_topic(topic),\_commonGaugeWidget(NULL)

{

……

//设置窗口上的图标按钮

setupLeftGUI();

setupRightGUI();

//设置系统中的消息处理，信号与槽函数进行绑定

setupConnects();

//设置中央窗口

setCentralWidget(\_centralWidget);

}

当构造函数处理完成之后，界面的主体模块就会被创建出来，与此同时，注意到BaseTopicAppWidget类的成员变量\_app为GameApp类的对象，所以当GampApp创建时也会调用它的构造函数，GameApp的构造函数声明为：

GameApp::GameApp():dtABC::Application(),\_gameManager(NULL)

,\_curMap(NULL)

{

\_rootScene=GetScene()->GetOSGNode()->asGroup();

}

因为GameApp继承了dtABC::Application，所以在调用构造函数时，会先调用父类的构造函数，同时它的\_rootScene变量会获取场景中的OSG节点。本界面中共分为六大模块，界面中分为两大部分，左半部分有四个模块，右半部分有两个模块，其中，界面左上角为虚拟实验室说明模块，用于说明当前实验的内容，功能，达到什么样的效果。实验说明模块由QtoolButton和QtextEdit组成。实验说明模块下面

则是特殊仪器模块，前文中已经提到，特殊仪器模块是由实验中的虚拟仪器构成的，这里只是添加了仪器按钮，如果要正常使用仪器，还需要与游戏角色进行绑定，这是下章中提到的内容。特殊仪器模块所使用的类为ElectricalGaugesWidget类，该类的声明如下：

class ElectricalGaugesWidget : public QWidget

{

……

ElectricalGaugesWidget(BaseTopicAppWidget\* app,QWidget\* parent=NULL);

SingleAmmeterGUI\* \_singleAmmeter;

}

其中的成员变量SingleAmmeterGUI、SinglePowerGUI等，就是所用的虚拟仪器类，它们都有相同的父类，即GaugeGUI。

游戏场景选择模块。该模块主要是用来选择游戏场景的，在delta3D中，游戏场景是使用Map来表示的，Map是一个特定关卡或地图的基础，它有一个名字，并保持它创建的所有代理。创建地图的相关代码如下所示：

void LearningProblemGUI::createNewMap()

{

std::string contextDir=dtCore::Project::GetInstance().GetContext();

//如果指定名称的地图文件存在，就加载该文件

\_app->loadMap(\_problem->name);

}

通用仪器选择模块。该模块正如之前所述，通用仪器模块中的实验仪器，是所有虚拟实验中都需要的仪器，通用仪器模块是由CommonGaugesWidget类来表示的。

三维场景渲染模块。在该模块中，显示了Delta3D的应用程序窗口，它可以用来进行游戏场景的演示，Delta3D应用程序由GameApp来控制，相关的代码 如下所示：

void BaseTopicAppWidget::initializeGL(QLayout\* layout,QWidget\* widget)

{

dtQt::OSGGraphicsWindowQt\* osgGraphWindow =

dynamic\_cast<dtQt::OSGGraphicsWindowQt\*>(\_app->GetWindow()->GetOsgVie

werGraphicsWindow());

……

dtCore::System::GetInstance().Start();

stepper.Start();

}

实验记录模块。在这一模块中，记录了实验的操作记录，可以对实验记录进行保存。同时，也可以显示操作实验仪器的注意事项。

## 2.4 结果展示

根据上节介绍的三维虚拟实验室界面设计及其实现，本节展示了三维虚拟物理实验室的界面设计。本文设计的界面系统功能完善，各个模块窗口之间的功能相对独立。本小结将采用图文形式，按照用户的操作顺序来展示界面系统。



图2-6 主界面显示模块

Fig.2-6 Main interface display module

系统主界面是系统中已经存在的界面，在这个显示模块中，需要添加自己设计的二级界面，并且把它做成界面点击按钮的形状，这样当我们点击按钮时，就会弹出二级界面的窗口。

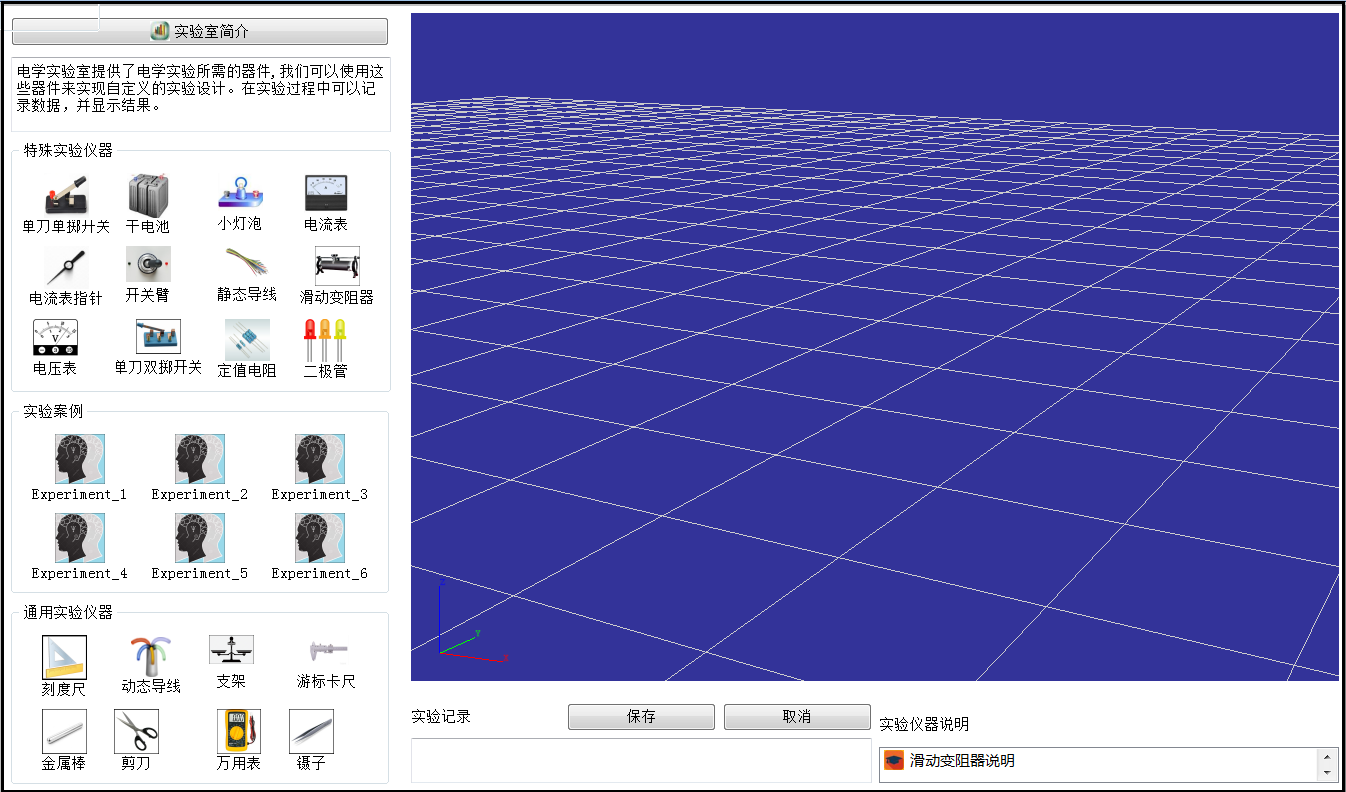


图2-7 电学实验界面

Fig.2-7 Electrical experiment interface

如图所示，当点击主界面按钮时，系统就会弹出此电学实验的二级界面，此二级界面中分为了5大模块，分别为特殊实验仪器、实验案例、通用实验仪器、三维渲染窗口及实验记录与说明。



图2-8 特殊实验仪器

Fig.2-8 Special experimental instrument

如图所示为特殊实验仪器模块。在该模块中，定义了进行电学实验时所使用的仪器，包括开关、干电池、小灯泡、导线等，与此同时，我们也可以根据自己的需求进行动态地添加和删除，这里的仪器都是后文中进行虚拟物理实验时所使用的。

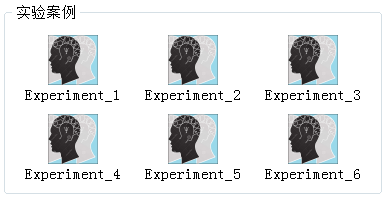


图2-9 实验案例

Fig.2-9 Experimental case

如图所示为实验案例模块。在该模块中，包含了六个实验案例，其中的实验案例其实就是Delta3D中的Map，Map是一个特定关卡或地图场景，它有一个名字，并保持它创建的所有代理。可以在这六个场景中进行任意想要做的实验。



图2-10 通用实验仪器

Fig.2-10 General experimental instrument

如图所示为通用实验仪器模块。通用实验模块中的实验仪器，大部分是进行物理实验时都可以使用到的实验仪器，因为本文所关注的是电学物理实验，所以这里给出了几个样例，更多的实验仪器可以根据用户的需求进行手动添加。

界面的右上角则是三维渲染模块。它实质上是Delta3D的应用程序窗口，用来展示三维场景，它被嵌入到了Qt的应用程序界面中，在界面中点击仪器按钮，都会触发一个槽信号，在槽函数中引入了Delta3D的代码，用于将游戏角色加入游戏场景中。

界面右下角则是实验记录以及实验仪器说明。在实验记录中，用户可以记录下本次实验的学习心得，之后点击保存按钮，会弹出一个文档对话框，这样就可以将学习记录以任意的格式记录到计算机的磁盘中。三维渲染及实验记录模块如下图所示：

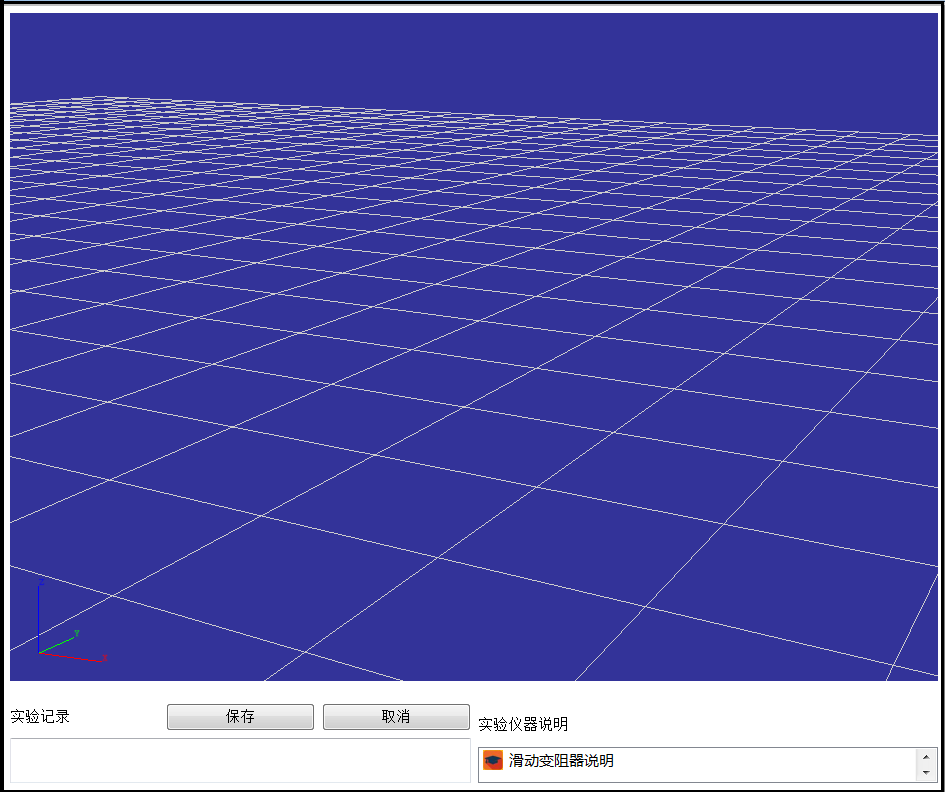


图2-11 三维渲染及实验记录

Fig.2-11 3D rendering and experimental recording

# 第三章 三维虚拟仪器的设计

## 3.1建模工具的选择

### 3.1.1三维建模工具3DS MAX介绍

3DS MAX是由Autodesk Media and Entertainment公司设计开发出的一款专业计算机3D图形处理软件，它用来制作三维动画、模型、游戏及图像，也是目前全球拥有最多使用者的三维制作软件。它具有三维建模能力和强大的插件构建能力并且在游戏开发者、电视广告工作室、建筑设计工作室中被广泛使用。对于它的建模和动画工具而言，最新版本的3DS MAX还提供了着色器（比如环境光遮蔽和次表面散射等），动态模拟，粒子系统，辐射度，标准地图创建，渲染，全局光照等功能[22]。由于3DS MAX拥有很强大的功能而且简单易学，所以在个人PC及虚拟现实中被广泛地应用。3DS MAX的建模技术有如下几种方式：

（1）旋转建模方法。对于中心轴对称的物体，例如一些水果、柱子、各种瓷器等，都会用到这种建模技术，这类物体的造型相对容易，可以先绘制出对称的截面，然后在修改器中加入旋转参数，这样就可以看到三维物体了。

（2）loft放样建模方法。在3DS MAX中有大量的标准几何体用于建模，使用这些几何体建模方便快捷、易学易用，只需要改变几个参数并通过旋转、缩放等功能就可以把它们堆砌起来，组成简洁美观的外形，这对于初学者来说是最好的建模方式。

（3）ProBoolean建模方法。ProBoolean是一种类似于布尔操作的建模方法，它将大量功能添加到传统的3DS MAX布尔对象中，可以自动的将布尔结果分割成一个可修改的截面。因此ProBoolean建模方法适用于创建对精度或细节处理要求较高的3D模型，比如工业制造等领域。

（4）复制、堆积建模方法。如果遇到不规则的堆积状物体，例如石块、云朵、树叶等，常使用复制、堆积建模方法。这需要对单个的物体做详细的设计，之后经过复制、旋转、缩放、位移等将这些物体形成不规则的排列。

（5）依附建模方法。当遇到空间变形或物体扭曲的情况时，需要采用依附建模

的方法。例如水纹、随着水波上下摆动的水草，这些物体不是独立存在的，它们的存在形式与它们所依附的物体是相关的，对这些物体进行建模需要根据已经创建好的模型为基础，再结合一定的变形或空间扭曲才产生的。

### 3.1.2 主流建模软件的异同

在影视、动画、游戏、建筑设计等行业中，3D建模无疑是数字生产的一个关键组成部分，下面列出了除3DS MAX以外的几款主流的3D建模软件以及它们的优劣比较。

（1）Maya



图3-1 Maya主界面

Fig.3-1 Maya main interface

Autodesk Maya通常缩写为Maya，是一款3D计算机图形软件，它最初由Alias Systems Corporation开发，用于创建交互式的3D应用程序包括视频游戏，动画电影，电视剧和一些高级视觉效果。Maya是高端3D软件而3DS MAX是中端软件，当遇

到一些高级需求时（角色动画，运动学模拟）远不如Maya强大，Maya从建模到动画、速度都非常出色[23]。3DS MAX属于普及型三维软件而Maya主要是为了影视应用而研发。

（2）SketchUp

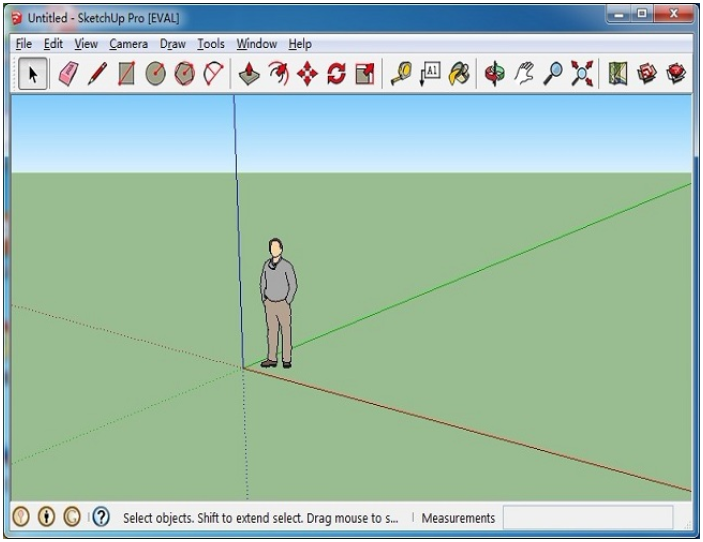


图3-2 SketchUp主界面

Fig.3-2 SketchUp main interface

SketchUp使用简便，建模效率高，适用于室内设计、建筑景观、土木工程等三维设计，它有一个免费的版本和一个付费的版本，对于制图基础不高的人也能很好地掌握。SketchUp和3DS MAX都是三维建模软件，都能完成从导入图纸到建模出图的一系列过程，无论是做单帧的图片还是动画都是可以的，不过SketchUp需要使用独立的渲染器来进行渲染，它可以外接亚特兰蒂斯或者内置的Vray渲染器来渲染特定效果的图片[24]。

（3）Blender

Blender是一款免费的软件，和3DS MAX相比它最大的特点就是开源并且功能简洁，有了Blender后，用户不用花大价钱也可以制作自己喜爱的模型了[25]。

## 3.2模型设计的基本准则

在生产环境和企业开发过程中，相对于传统的二维平面设计方式，使用三维数字化软件来设计产品具有非常大的优势。首先，有了物体的三维可视化模型就可以产生任意的视图，且视图之间可以保持正常的投影关系，这就为生产工程图带来了方便。此外，三维模型可以很自然的生成透视图和轴侧图，这在二维的平面设计中是无法实现的。最后，三维模型的设计是人工绘图的延伸，过度自然且学习简单。但是，在模型设计的过程中，还需要遵守如下的一些规范，这样可以使建模更加高效且易于后期的维护。

（1）确定模型单位及坐标。这是我们在建模初期首先要考虑的问题，在没有特殊要求的情况下，可以将模型的单位坐标设为米。在模型分工之前需要根据一个CAD底图来确定自己分工区域的模型位置，设计人员最好将该底图导入到3DS MAX的(0,0,0)位置，以便初始模型在零点附近。

（2）删除模型中多余的面。当我们使用环绕、拖动等模式依然观察不到模型中的一些特殊的面时，就可以将它们删除。这样做可以提高模型的载入速度和贴图的利用率，同时降低整个场景的面数，提高交互场景的运行速度。

（3）贴图和材质的要求。3DS MAX中不是所有的材质都被三维仿真软件所支持，在保证模型通用性的前提下，可以使用默认的标准材质球。而三维仿真软件目前只支持Bitmap贴图类型，只支持漫反射和自发光贴图通道，同时在贴图的大小和尺寸上也需要根据实际情况做出相应的调整。

（4）按照标准来命名模型。在英文系统中不可以使用中文命名的模型，这就要求英文格式的模型名称，并且必须按规范命名。例如对于建筑模型的命名需要使用如下格式：区域名\_jz\_编号。

（5）模型需要分类制作。当在近景使用模型时，就需要制作精细的模型，除了很细的栏杆用贴图外其余所有部分均用模型制作。而在中景中使用时，需要中等精度的模型，例如柱子、墙面，它们的装饰要尽量节省面数。在远景使用时，就需要使用一些渲染贴图来代替模型。

## 3.3模型的结果展示

首先从官网上下载3DS MAX进行安装，本文所使用的3DS MAX版本为3DS MAX 2014，安装完成之后打开，3DS MAX主界面如图所示：

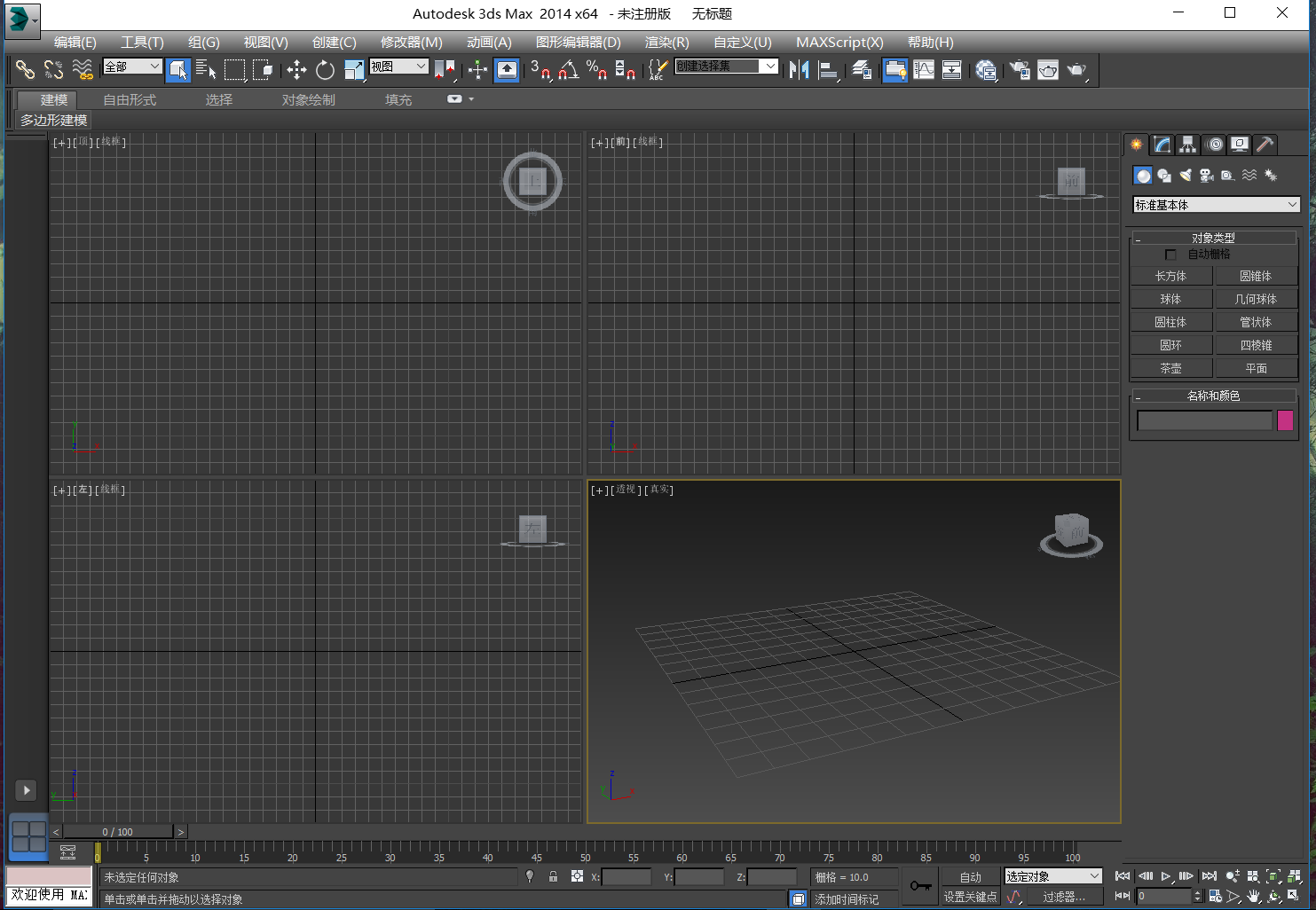


图3-3 3DS MAX主界面

Fig.3-3 3DS MAX main interface

启动并进入3DS MAX中文版系统后，即可看到如图所示的初始界面。该界面中包括以下几个主要部分：菜单栏、主工具栏、视图区、命令面板、视图控制区、动画控制区等[26]。界面中顶部即标题栏，它用于管理文件和查找信息。标题栏下方是菜单栏，命令面板、工具栏中的按钮功能都可以在菜单栏中找到对应的命令，当用户使用应用操作三维模型时可以直接在命令面板、工具栏中找到对应的按键，这比直接在菜单栏中操作方便很多。菜单栏中的一些功能按钮会带有三角形的下标，这表示会有多余的功能选项。在3DS MAX菜单栏的下方有一栏工具按钮称为主工具栏，如图所示：



图3-4 3DS MAX工具栏

Fig.3-4 3DS MAX toolbar

当首次打开应用时，界面的正中央以四个视图分隔的窗口就是应用的视图区，三维模型的操作，比如建模，贴图，赋予材质，光源设置等都是在这里完成的，并且在这里也可以显示操作完成之后的效果。四个视图分别显示了模型的正视图，侧视图，俯视图和透视图，并且每个视图都可以选择不同的显示效果，例如可以只显示模型的网格或者显示模型的贴图加上环境光[27]。命令面板位于界面的最右则，它

包含了操作对象模型的大部分功能并且可以对模型的参数进行修改，它是三维建模的核心工作区也是使用最为频繁的部分之一，创建任何物体或场景主要通过命令面板进行控制操作，命令面板中包括六个部分分别为创建、修改、层次、运动、显示、工具。视图控制区，用来控制模型的显示状态包括平移，旋转，缩放等操作，可以将相机调整到用户需要的位置。动画控制区，位于界面的下方用来控制动画的播放，可以对动画播放的速度及进度进行调整[28]。

通过对软件界面的简单介绍，可以对3DS MAX的功能有一个大致的了解，更多的功能将在模型展示阶段进行介绍，本文设计了一套用于虚拟物理实验的模型，现展示如下：

（1）干电池模型。使用3DS MAX设计的干电池模型如下图所示：

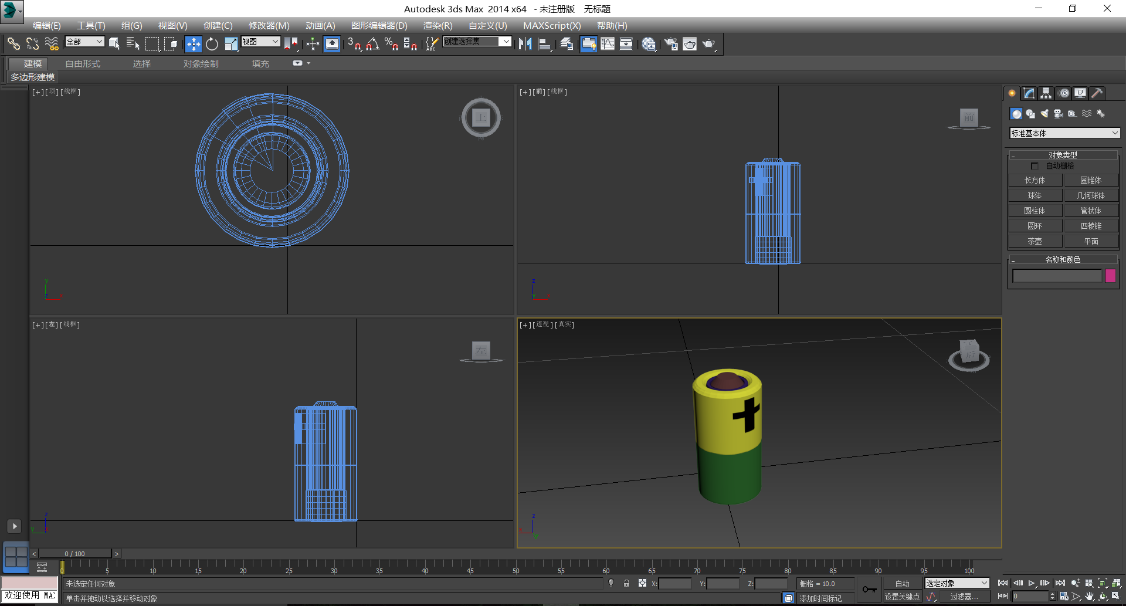


图3-5 电池模型

Fig.3-5 Battery model

干电池的设计所使用的基本模型是一个圆柱体，把圆柱体的顶部和底部做成类似于电池正负极的形状，圆柱体的侧面分成两部分选择材质上色。干电池是中学电学实验中最常使用的电源，它提供了稳定的电压。

（2）电流表模型。使用3DS MAX设计的电流表模型如下图所示：

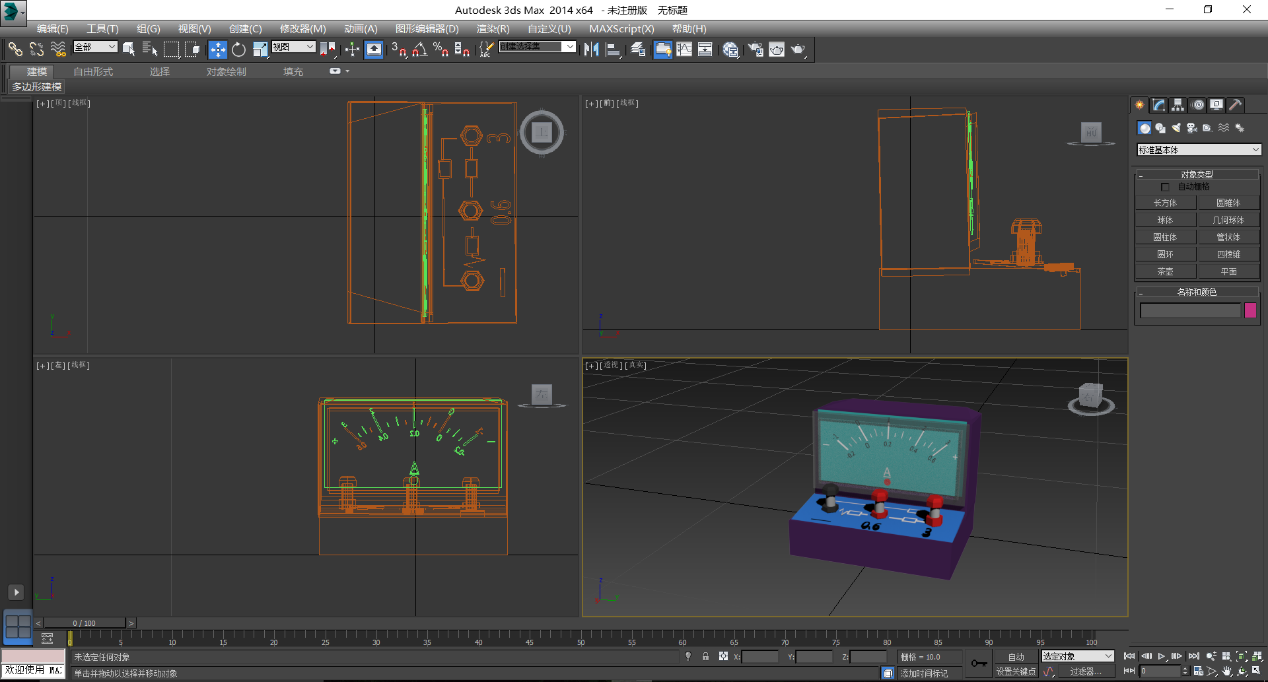


图3-6 电流表模型

Fig.3-6 Ammeter model

电流表和电压表所使用的基本模型是相似的，这里只给出了电流表的模型图。它总体上使用了两个不规则长方体和三个螺母，仪表盘的透明外壳需要一个带边框的内嵌面板并设置面板的透明参数使它看起来具有玻璃的视觉效果。仪表盘和面板的设计需要使用小贴片来绘制。在面板上需要嵌入三个螺母，两个红色的螺母分别代表0.6A、3A的电流输入，黑色的螺母代表了电流的输出。

（3）小灯泡模型。使用3DS MAX设计的用于电学实验的小灯泡模型如下图所示：

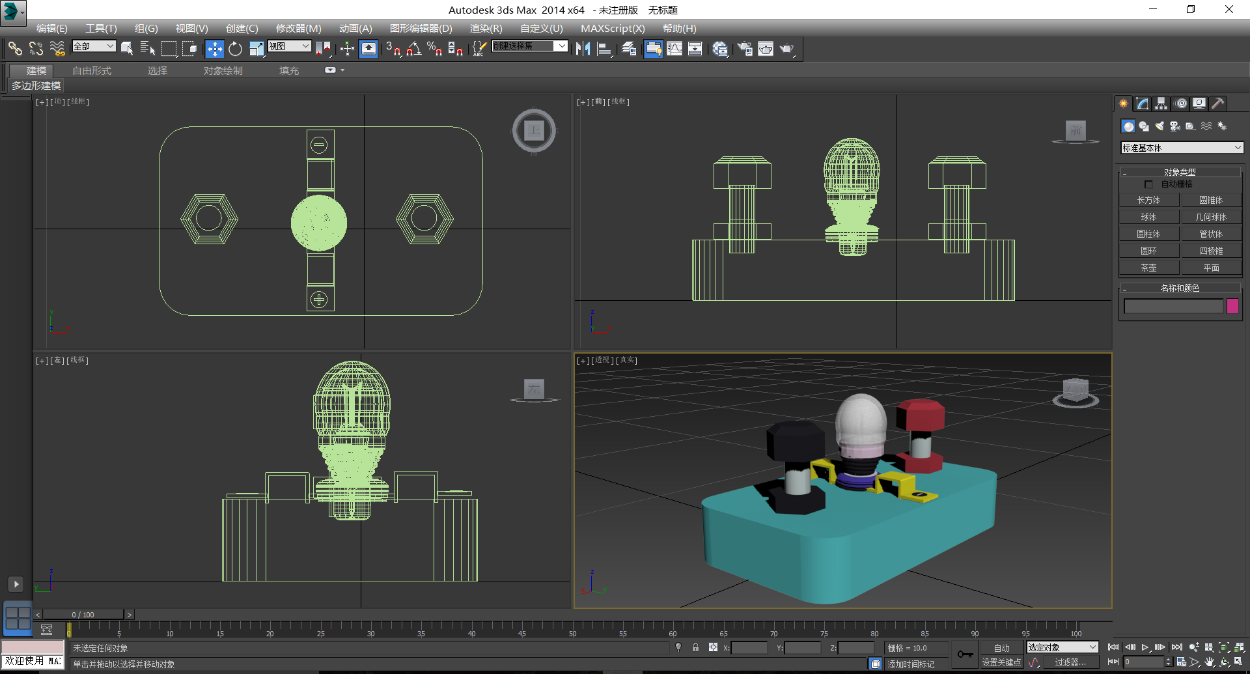


图3-7 小灯泡模型

Fig.3-7 Light bulb model

小灯泡的建模包括了长方体底座，两个螺母，中间的铁片及上部的小灯泡及内

部的电阻丝。其中重点是小灯泡的设计，首先灯泡的外壳是一个半椭圆形中空的球体并且将它的材质设置为白色透明，在球体的内部需要五根透明的玻璃柱，在玻璃柱的顶部设计螺旋线代表了电阻丝。在长方体的底座上有两个螺母用来连接导线。在灯泡的底部设计了结合部分并且用黄色的模型代表铁片。

（4）单刀单掷开关模型。因为开关有两种状态，打开、闭合，所以在设计单刀开关时就需要分别设计它的底座和手臂，开关闭合时，显示如下模型：

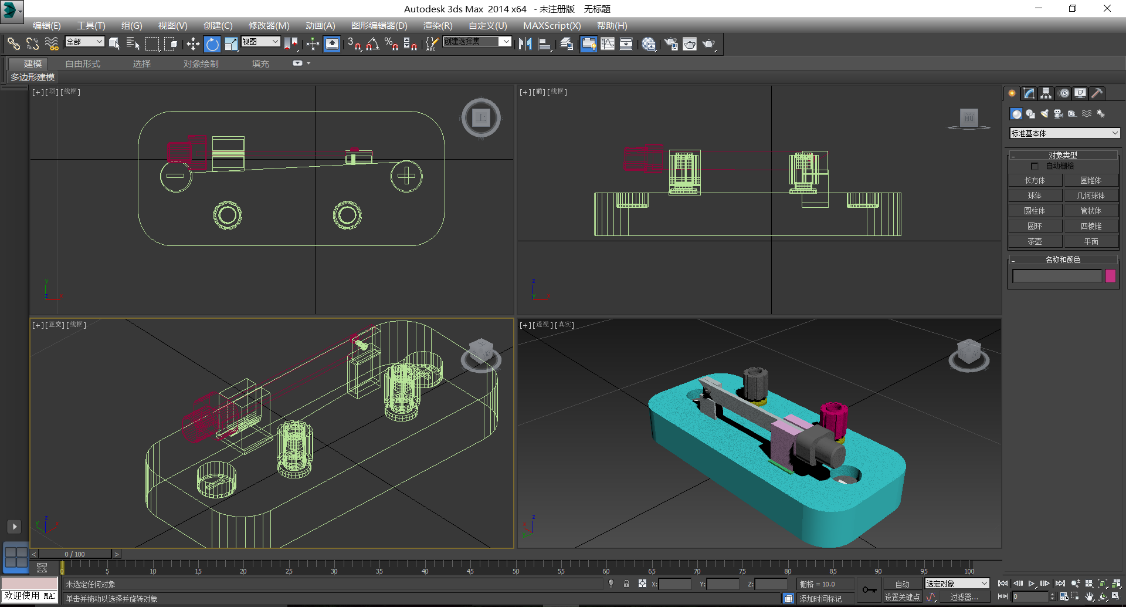


图3-8 开关闭合模型

Fig.3-8 Switch close model

开关打开时，显示如下模型：

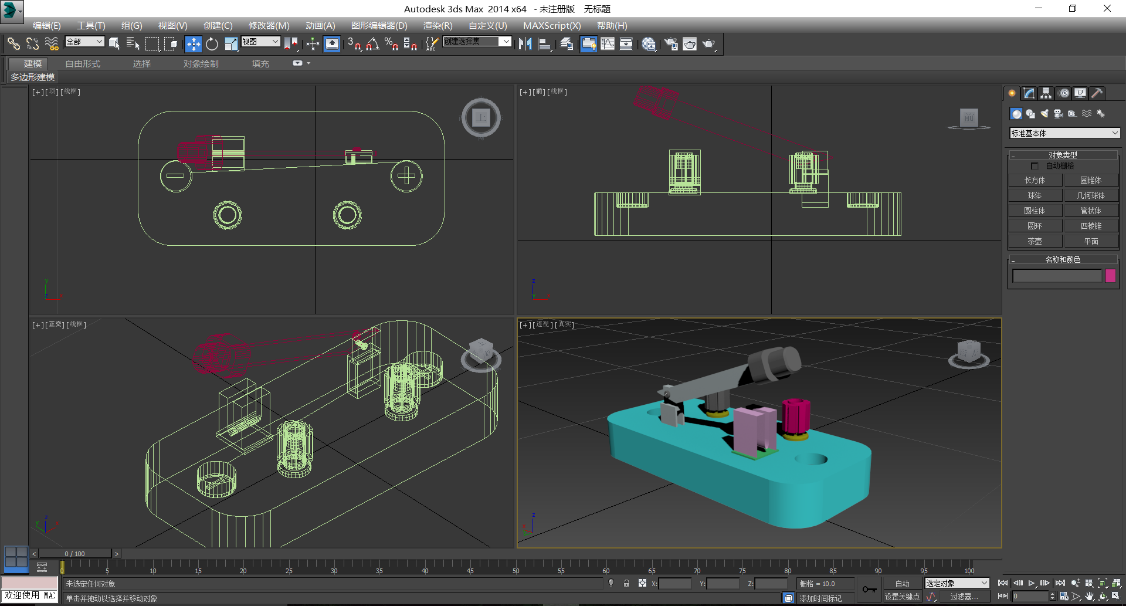


图3-9 开关打开模型

Fig.3-9 Switch open model

开关有两种模式：打开、闭合，在设计模型时要分两步来进行，首先设计开关的底座，接着设计它的手臂。这两部分并不相连在一起，导入模型时在游戏场景中可以预先计算它们坐标使它们在视觉效果上是连在一起的。这样以后就可以使用代码对手臂进行旋转操作。

（5）导线模型。使用3DS MAX设计的导线模型如下图所示：

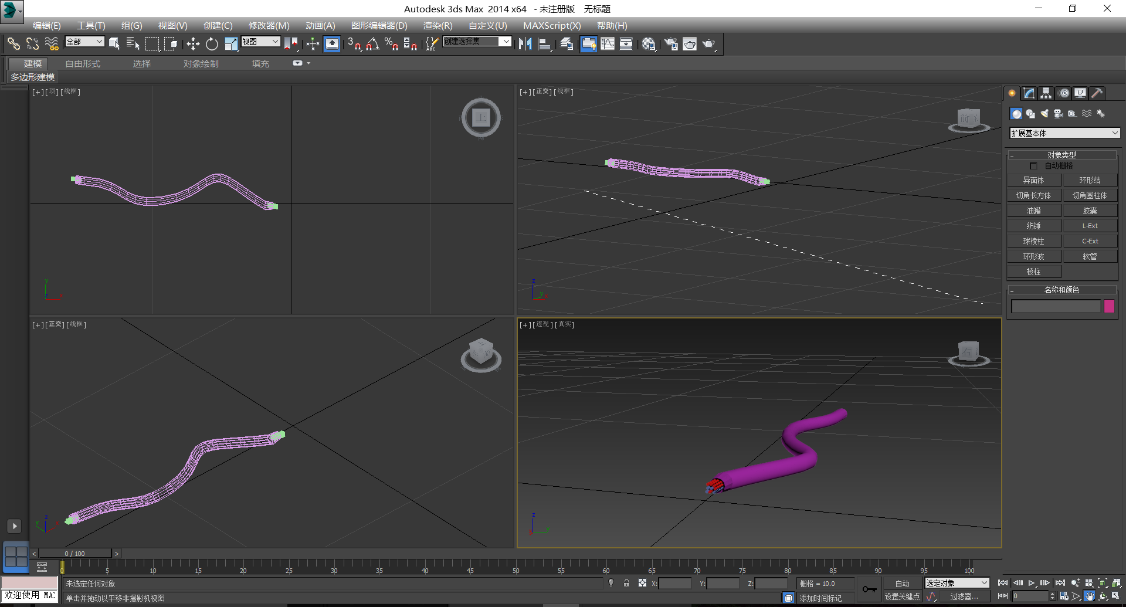


图3-10 导线模型

Fig.3-10 Wire model

导线的设计比较简单，将一个圆柱体进行拉伸弯曲处理即可得到一条导线模型，之后在两端做成中空并插入若干不同颜色的细线。

（6）万用表模型。使用SketchUp草图大师设计的万用表模型如下图所示：

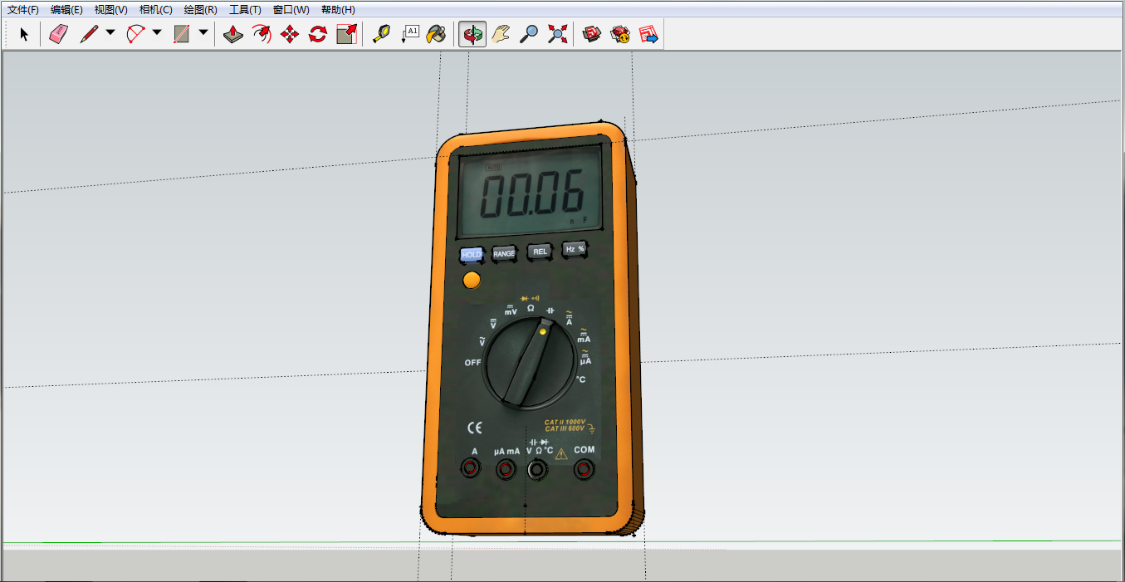


图3-11 万用表模型

Fig.3-11 Multimeter model

万用表模型使用的是草图大师建模软件，它的模型主体是一个长方体，在长方体的外表面按照实物模型进行构建，包括凸起和下凹的按钮，旋转的横梁等。之后在外表面进行贴图处理，这里需要注意的是贴图时坐标的计算及模型细节方面的处理。

（7）电容电阻模型。使用SketchUp草图大师设计的电容电阻模型如图所示：

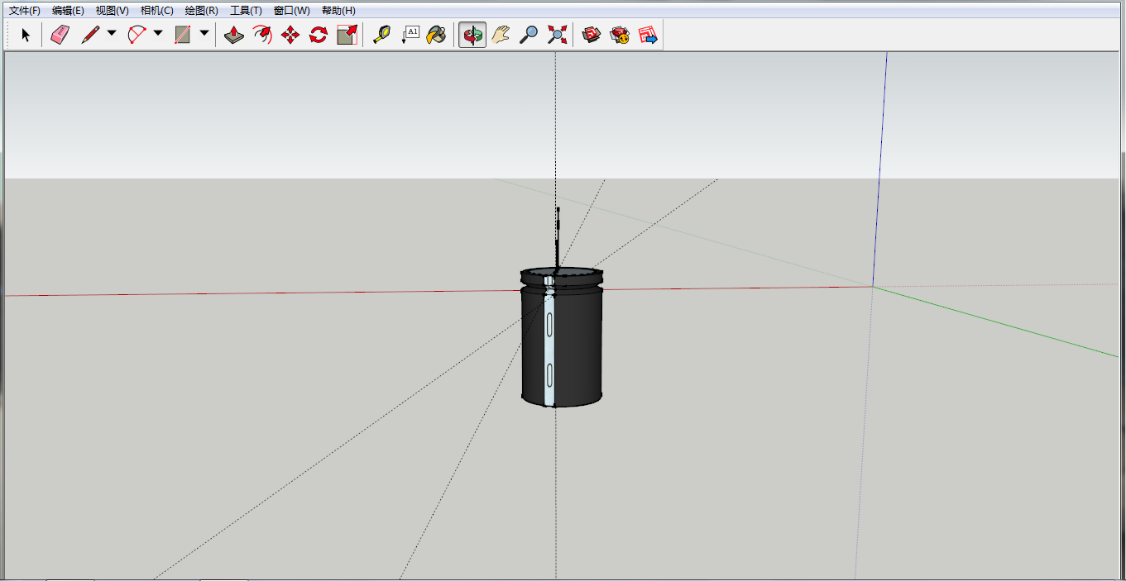


图3-12 电容模型

Fig.3-12 Capacitance model

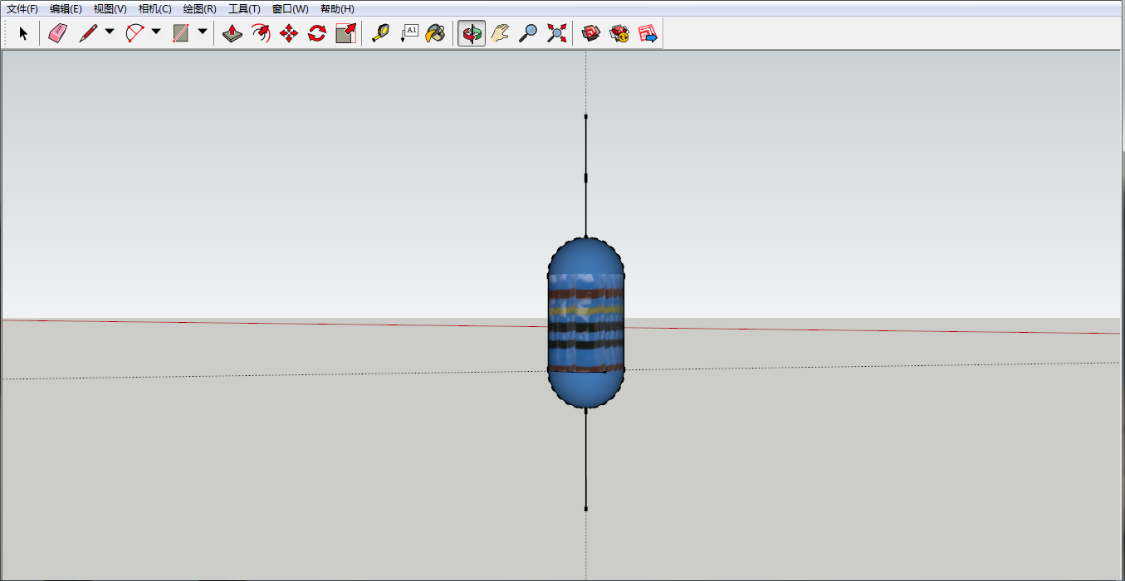


图3-13 电阻模型

Fig.3-13 Resistance model

电容电阻模型的设计都是以圆柱体为基础修改而来，这里需要注意的是圆柱体光滑部分的处理及外部贴图的实现。当需要若干个电容电阻时，只需要改变它外部的模型贴图即可。

以上就是本文所建立的部分电学实验模型。在三维建模时，需要注意如下问题：在开始三维建模前首先要确认模型的大小，可以通过正视图的格数来确定比例，然后设置格子的长度就可以了。在虚拟项目制作过程中，为便于项目管理及制作可以把项目的工作路径进行统一。建模完成后模型要处于世界坐标的中心，而且模型的底部要与X轴和Z轴所构成的平面紧贴，角色模型要在Y轴的正方向，并且要保证角色正面朝向Z轴，同时，建模完成后要及时删除历史以及清理垃圾节点，当拿到设计完成的角色文件时要先检查是否有问题，如果发现有错误要及时沟通。

## 3.4 利用dtGauge制作仪表盘

### 3.4.1 dtGauge概述

dtGauge是Delta3D的扩展类库，它是在osgGauge基础之上进行源代码的重新修改、封装得到的，所以需要首先了解osgGauge，osgGauge是osg的相关扩展，它是一个2D仪表盘渲染库。dtGauge在它的基础上演变而来，可以渲染三维模型上的仪表盘，这样，我们可以用代码的形式来开发仪器模型中的仪表盘而不是直接贴图或者使用小贴片，使用这种方式可以简单方便的开发特殊仪器的仪表盘，使仪器看上去更加真实且在开发阶段就可以将仪表指针与仪器绑定在一起，减少后期开发的复杂性[29]。

首先得到dtGauge源代码，之后对其编译得到dtGauge.dll和dtGauge.lib，将dtGauge.dll和dtGauge.lib分别拷贝到项目指定的路径中，加上相关的源文件即可方便开发需要的仪表盘。

### 3.4.2仪表盘的编程实现

使用dtGauge类库实现虚拟环境中的仪表盘所需的代码非常简单，为了将仪表贴到三维仪器对应的平面上需要做如下的工作：

（1）计算仪表盘贴图相对于仪器的位置坐标。三维仪器对应的平面上定位仪表盘每一层的四个坐标，该步骤可以在三维建模时由建模人员记录下局部坐标系下的坐标值，也可以在STAGE中用坐标取值的方法进行确定。如果在STAGE中三维仪

器不处于原点，则应该计算出坐标相对原点的位移矢量。得到贴图的位置信息后，将它写入相关的XML文件中由程序负责读取，也可以直接硬编码到成员变量中。如果采用第一种方式，仪表相关的XML文件样例如下所示：

<?xml version="1.0" encoding="utf-8"?>

<Gauges>

<Gauge x=" " y=" " name=" ">

<Layers>

<Layer width=" " height=" " textureFilename=" ">

……

<Operations>

<Operation type=" " property=" " offsetX=" " offsetY=" ">

……

</Operation>

</Operations>

</Layer>

</Layers>

</Gauge>

……

<Gauges>

根据标签名可以很容易看出其所对应的功能，<Gauge></Gauge>标签中的子标签<Layers></Layers>代表仪表贴图的层，在每一个层中可以有对应的操作，相关的数据被封装在< Operations ></ Operations >标签之中。之后，使用XML解析工具将数据提取出来，保存到dtGauge::GaugeHandler类的数据成员中：

dtGauge::GaugeHandler handler;

dtUtil::XercesParser parser;

bool result = parser.Parse("testGauge.xml", handler);

当然也可以直接硬编码将数据写入dtGauge::GaugeHandler类中，这个过程是在\_createGaugeData()函数中实现的。

（2）将仪表层和三维模型合并成一个整体，\_createGaugeWithGaugeData()或\_createGaugeWithXML()函数中完成该过程的实现，第一个函数是直接将位置坐标中

的数据放入相关的对象中。这两个函数所实现的工作，都是将得到的仪表数据添加到GaugeDrawable中，GaugeDrawable是Delta3D的dtCore::DeltaDrawable的派生类，其核心数据成员是osg的群节点osg::ref\_ptr<osg::Group> mRootNode。

（3）对需要的图层添加相应的位移操作。仪器中的指针是需要动态变化的，要完成这项功能就需要用到dtGauge中的OperationData类，该类用于描述施加到某个仪表层的操作。OperationData类的声明如下所示：

struct OperationData

{

OperationData():mType(Operation::NONE),mTranslationOffsetX(0.f),

mTranslationOffsetY(0.f),mInputScale(1.f){ }

//需要确定操作的类型

Operation::GaugeLayerOperationType mType;

//设定属性参数

float mTranslationOffsetX;

float mTranslationOffsetY;

……

……

};

OperationData所做的操作包含了旋转仪表层、上下左右平移仪表层、上下左右滚动仪表层、旋转仪表层。OperationData类提供了一个偏移量offset，用于针对不是围绕中心旋转的情况，或者用于平移过程的平移量。调用SetOffsetTranslation()函数调节该量。通过SetInputScale()函数可以设定层操作的输入量程值（input scale value），对实现钟表类Gauge的连续旋转非常有用。

本文所设计的仪表基类为GaugeActor，其它具体的仪表都要由该类派生，该类的声明如下所示：

class GAUGE\_EXPORTS GaugeActor : public dtActors::GameMeshActor

{

……

void addGauge( BaseGauge &gauge );

//将创建的仪表盘添加到角色当中

virtual void \_createGaugeWithXML();

virtual void \_createGaugeWithGaugeData();

virtual GaugesData \_createGaugeData();

virtual void \_update(float dt);

……

GaugeMaster \_gaugeMaster;

};

该类继承了GameMeshActor基类是一个游戏角色，有关游戏角色及游戏角色代理的概念将在下章实验开发阶段进行介绍，这里将只关注创建仪表的过程。该GaugeActor类用于创建三维仪器，每个这样的仪器均有一个仪表盘。创建具体的仪器仪表就可以对GaugeActor进行派生，重写相关函数即可。

在GaugeActor中有一个GaugeMaster对象 \_gaugeMaster，该对象是三维仪器中仪表盘创建的基础和出发点。\_createGaugeWithGaugeData()函数在仪器角色添加到系统时被调用，一般重载仪器角色的虚函数AddedToScene（dtCore::Scene\* scene）并在该函数中调用\_createGaugeWithGaugeData()，该函数定义如下所示：

void GaugeActor::\_createGaugeWithGaugeData()

{

\_gaugeMaster.CreateGauges(\_createGaugeData());

GetMatrixNode()->addChild(\_gaugeMaster.getGaugeNode());

}

在派生类中无需重新定义\_createGaugeWithGaugeData()函数，从这里可以看出每个仪器具体的仪表盘定义在\_createGaugeData()中，所以每个GaugeActor的派生类都要定义自己的\_createGaugeData()函数以实现具体的仪表盘，并将该仪表盘添加至三维仪器中。当对GaugeActor进行派生创建具体的仪器时，就可以对继承下来的虚函数进行重写，例如本文中创建的电流表仪表盘就可以对\_createGaugeData()函数进行重写，相关代码如下所示：

GaugesData AmmeterActor::\_createGaugeData()

{

//里面有一个向量,封装了所有的GaugeData

GaugesData gd;

//单个仪表盘相关的数据，只针对电流表

GaugeData clockData;

clockData.mName = "Ammeter";

clockData.mX =0.0f;

clockData.mY =0.0f;

//仪表层相关的数据

LayerData baseLayerData;

//对baseLayerData中的数据成员进行赋值，它代表了底层仪表盘

……

//这个是指针图层

LayerData needleLayerData;

//对needleLayerData中的数据成员进行赋值，它代表了指针图层

//Operation操作用来旋转指针，我们添加了一个偏移量这样在旋转

//时就不会围绕指针的中心旋转

OperationData needleRateOp;

needleRateOp.mType = Operation::ROTATION;

needleRateOp.mTranslationOffsetY = needleLayerData.mHeight/2.f;

//指针图层中添加操作

needleLayerData.mOperations.push\_back(needleRateOp);

//仪表盘上依次添加图层

clockData.mLayers.push\_back(baseLayerData);

clockData.mLayers.push\_back(needleLayerData);

……

}

当仪器加入到场景中时就可以将该函数放入AddedToScene()中，相关的仪表界面即可显示，如下是使用dtGauge开发出的仪表盘：



图3-14 使用dtGauge创建仪表盘

Fig.3-14 Create instrument panel with dtGauge

上图中可以看出，在Delta3D的场景中创建出了一个虚拟的仪表盘，在模型上方它由两个图层所构成，第一个图层是底部的圆形仪表盘，第二个图层是在仪表盘上部的指针图层。它们的坐标事前已经计算过了，所以在模型导入时图层与图层之间，图层与模型之间就可以非常紧密的贴合在一起。

## 3.5 dtGauge的优势与不足

本文所构建虚拟仪器的步骤为：首先使用三维建模软件设计模型的主体部分，之后使用Delta3D的扩展类库dtGauge来创建虚拟模型的仪表显示部分，当然在创建仪表盘时还有另外一种思路，即在3DS MAX中对模型的仪表盘进行贴图处理，仪表的指针当作另外一个模型制作，之后在进行实验开发时计算两部分的坐标使它们贴合在一起。

使用Delta3D扩展库制作的虚拟仪器与当前市场上所流行的虚拟仪器开发软件比如LabVIEW是不同的。首先Delta3D和LabVIEW是两种不同的平台，Delta3D是基于C++的一个游戏引擎而LabVIEW是由国家仪器公司开发的一款系统设计平

台并提供了一个可视化的编程环境[30]。以LabVIEW为代表的软件工具，主要是用于解决实际生产环境中的一些高精度测量，比如说测绘领域中的一些问题或者是电学示波器的测量分析。而本文使用dtGauge类库设计的虚拟仪表，主要是用于教学辅助的，它可以设计一些比较简单的仪表如电流表、时钟的钟摆、汽车上的时速盘等，但是当面对一些像示波器这样比较复杂的仪表时，它的开发过程就会十分复杂而且不容易维护。而使用LabVIEW就会非常的简单，它的图形化编程只要根据需求进行鼠标拖拽即可自动完成。

概括起来，使用dtGauge有如下几个优势：

（1）免费和开源。dtGauge继承自OSG，而OSG是一个开源的三维图形渲染引擎，所以开发者可以修改底层的代码来满足不同的需求而且也可以开发各种风格的仪表。开源即意味着免费，相比LabVIEW昂贵的授权费用来说，使用dtGauge来开发一些功能简单的仪表也是一种很好的解决方案。

（2）可扩展性。dtGauge是Delta3D的扩展库，它底层使用OSG渲染引擎，开发者可以对它进行二次开发，并且它也可以在所有Delta3D和OSG的项目中使用。

（3）跨平台性。可以运行dtGauge的平台需要具备OpenGL的支持能力，以及C++的编译环境，满足要求的系统平台包括Solaris、Windows、Mac OSX、Sony Playstation等。

而dtGauge的不足也是很明显的，正像上文提到的，相对于LabVIEW的图形化编程而言，开发者需要熟练掌握OSG和C++编程，这无疑增加了开发的难度。

# 第四章 三维虚拟实验

## 4.1 Delta3D游戏引擎的使用

### 4.1.1 Delta3D概述

Delta3D是一个开源的游戏仿真引擎，可以用于计算机仿真、虚拟现实等项目的开发。Delta3D是基于模块化设计的，它集成了许多非常有名的开源项目如OSG、ODE、CAL3D、OpenAL、Qt、CEGUI等[31]。Delta3D的底层使用的是 OpenGL图形渲染标准，它的目标主要是为应用可视化提供一套方便简洁的API库。Delta3D有三个主要的子系统，如下图所示：

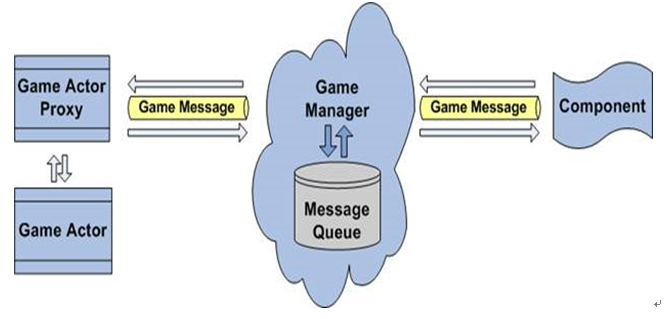


图4-1 Delta3D三个子系统

Fig.4-1 Delta3D three subsystem

游戏管理器（Game Manager）是Delta3D的核心子系统，它负责游戏角色和消息的传送。第二个子系统是游戏角色代理（Game Actor Proxy）和游戏角色（Game Actor），游戏角色代理是游戏角色的包装，游戏管理器可以通过角色代理来操作游戏角色。游戏角色及其代理是存在于虚拟世界的实体，它们被创建出来并在游戏管理器中进行工作，受到游戏管理器的管理。第三个子系统是组件（components），组件则是用来接收、处理、发送消息。可以创建一个简单的组件用来监听应用程序的输入，比如单机鼠标按键产生的消息就会传入组件当中，同时也可以创建复杂组件用来处理游戏角色的消息传递，组件也是在游戏管理器中接受游戏管理器的管理。由此可见，游戏角色和组件都存在于游戏管理器中受到它的管理。游戏管理器的功

能如图所示：

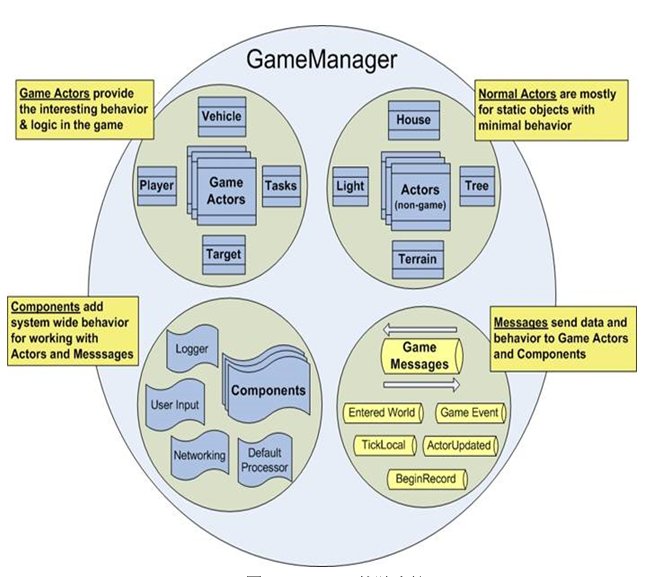


图4-2 游戏管理器的功能

Fig.4-2 Game Manager function

游戏管理器在游戏运行时做了三件事情：第一件，游戏消息管理。在游戏管理器中，角色或组件是通过消息来进行交互的，系统内部各模块的状态变化也是通过消息来交互的，游戏管理器就像是消息中转站一样接收、路由消息。第二件，游戏管理器维护其中存在的所有角色，包括游戏角色及游戏角色代理。游戏管理器会加载地图场景中创建的角色并通知角色处理消息，确保角色在进入、离开场景时执行正确的操作。第三件，游戏管理器支持游戏组件（GM组件）。组件是一个高层次的对象，它可以对消息的处理进行封装。

Delta3D的类库中主要包含如下功能模块：

（1）dtCore：包含了引擎的通用功能，包括：键盘鼠标的输入，行动模型，烟、火花、爆炸等微粒系统作用，摄像头控制，多窗口支持，照明设备等[32]。

（2）dtABC：dtABC库是Delta3D提供的一个应用基类库，包含了一系列开发3D应用程序的高端基类。该库提供了应用程序模板、天气相关的接口、开发向导等功能模块[33]。

（3）dtAnim：用于支持骨骼网格动画，该类库对一个角色进行封装，以便于角

色进行基本的转向。它有两个高端类分别为CharacterWrapper和AnimationHelper。

（4）dtTerrain：Delta3D的dtTerrain类库用来支持应用程序中与地形相关的行为。它起源于两个非常专业的地形开发库：SOARX和GENETICS，将这两个开发库合并成了Delta3D的一部分，即dtTerrain。

（5）dtDAL：提出了一个泛型模板的概念，允许编辑器将角色当成泛型对象来与角色打交道，泛型模板的核心是每个角色使用它的名称、ID和属性来让游戏管理器泛型对待[34]。

（6）dtActor：该库中的类从dtCore和dtABC库中的类派生而来，包括GameMeshActor和GameMeshActorProxy。

### 4.1.2 Delta3D的实用工具

Delta3D除了完成功能的基本模块外还提供了许多其它的工具，比如：STAGE，这是Delta3D的游戏场景设计器，如下图所示：



图4-3 游戏场景设计器

Fig.4-3 Game scene designer

STAGE的全称为simulation, training, and game editor，主要是做游戏场景的布置和角色操作。角色用类来表示并提供行为和数据，比如一辆汽车[35]。该编辑器支持Delta3D应用的方方面面，它主要包括以下方面的内容：地图编辑、项目资源和库管理，可视化创建和操纵角色，属性编辑和角色链接，分屏视口-透视图、顶视、侧视、

正视图，网格、光照和纹理模式，可重构的用户界面，角色搜索和全局角色管理，纹理、网格、声音和粒子浏览器[36]。

可视化粒子编辑器（ParticleEditor）可以用来可视化开发游戏中的粒子系统，为编辑粒子特效提供一个方便的操作，其操作界面如图所示[37]：

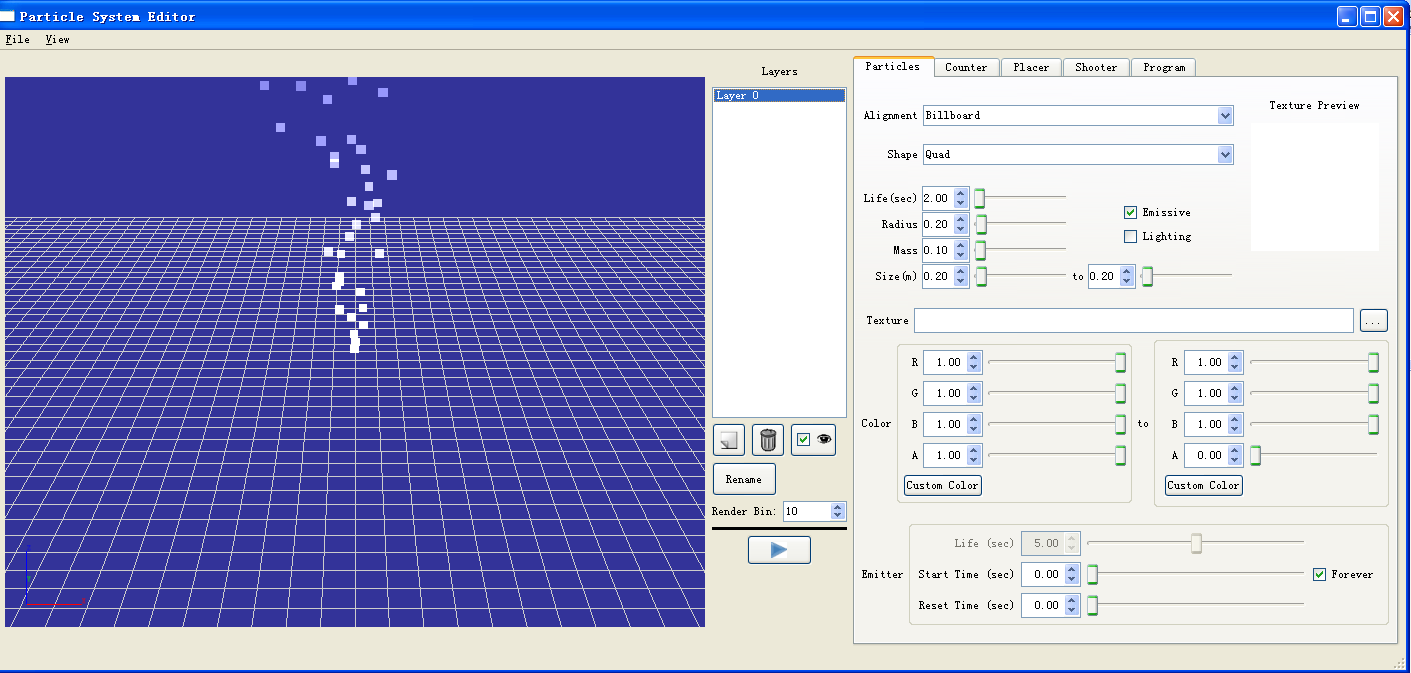


图4-4 粒子编辑器

Fig.4-4 Particle Editor

此外还有3D模型浏览器，BSP编译器，HLA阅读器等实用工具，可以为Delta3D提供更加方便的操作。

### 4.1.3 Delta3D的编译

在项目中如果想要正常使用Delta3D，需要用到它的头文件（.h），源文件（.cpp），以及库文件（包括静态库文件、动态链接库文件）。头文件和源文件可以通过设置环境变量来指定路径，而库文件则需要手动进行编译来产生。首先，从Delta3D的官方网站下载Delta3D的源代码包，例如：delta3d\_REL-2.7.1，将Delta3D拷贝到指定的目录，例如：D:\delta3D。之后，在环境变量中添加如下的内容：

（1）DELTA\_ROOT D:\delta3D\delta3d\_REL\_2.7.1

（2）DELTA\_LIB %DELTA\_ROOT%\data

（3）DELTA\_INC %DELTA\_ROOT%\lib;%DELTA\_ROOT%\ext\lib

……

……

同时在PATH中，需要根据需求添加相应的环境变量。因为Delta3D中是使用cmake来构建项目工程，所以需要先确保电脑中有cmake应用。打开Delta3D的安装目录，将安装目录中的cmakelist.txt文件拖入cmake中并根据自己的需要进行设置，即可生成DELTA3D.sln文件，双击该文件用VS2008将其打卡，随后对其进行编译，生成的库文件如图所示：



图4-5 Delta3D的库文件

Fig.4-5 Delta3D library file

当生成Delta3D的库文件之后，我们就可以使用它来开发相关的应用了。

## 4.2 Delta3D中的游戏角色

### 4.2.1游戏角色的实现

在本节中，介绍了Delta3D中的一个游戏案例：坦克大战游戏，并利用其中的角色进行场景展示。而本文自定义的一些游戏角色，比如电流表、电压表、小灯泡、干电池等，都可以类比到坦克游戏角色中。Delta3D中可以创建自定义的角色，并将其导入STAGE然后在dtCore::Scene中进行展现。要完成此功能需要如下的几个步骤：首先对自定义角色进行编程处理，定义相关的类来表示自定义角色。然后，开发自定义的角色代理类，用这个代理类在STAGE中代表角色，之后开发一个动态链接库将相应的角色输出到这个库中。最后，利用GameStart或dtABC::Application来运行程序。

根据上述的游戏角色编程步骤，首先创建自定义游戏角色TankActor及游戏角色代理TankActorProxy。游戏角色的类声明如下：

class TUTORIAL\_TANK\_EXPORT TankActor : public GameMeshActor

{

public:

//TankActor的构造函数。并给该角色命名为“HoverTank”，从代码中可

//以看到，角色的构造函数是以游戏角色代理dtGame::GameActorProxy

//的引用作为参数的，所有的游戏角色都必须有这样的构造函数

TankActor(dtGame::GameActorProxy& proxy);

//重载基类的相应函数，只要该角色对应的角色代理注册TICK\_LOCAL

//消息，则系统会自动调用该成员函数。该函数与相关变量构成一个

//Invokable（在这个游戏角色相对应的游戏角色代理中的BuildInvokable

//（）函数中创建），当本地角色接收到一个tick时调用该函数。参数

//tickMessage表示包含tick相关信息的消息。注意到Tick.GetDeltaSimTime()

//函数返回最近的一次tick算起的模拟时间的变化值

virtual void OnTickLocal(const dtGame::TickMessage& tickMessage);

//当远程游戏对象接收到tick消息时，该函数就会自动调用

//TickMessage代表了游戏两帧之间的改变

virtual void OnTickRemote(const dtGame::TickMessage& tickMessage);

//泛型消息处理函数也称为Invokable，基类成员函数（来自

//dtGame::GameActor类）的重载函数，缺省下游戏角色代理GameActor

//Proxy调用该函数，该函数的参数是dtGame::Message的一个引用。在

//Delta3D中除了在dtGame::Actor中定义的这个Invokable，在dtGame::

//GameActorProxy中还定义了另外两个Invokable，即上面OnTickLocal()

//和OnTickRemote()函数

virtual void ProcessMessage(const dtGame::Message& message);

//坦克速度属性的getter和setter方法

void SetVelocity(float velocity);

float GetVelocity() const { return mVelocity; }

//坦克转速属性的getter和setter方法

void SetTurnRate(float rate);

float GetTurnRate() const { return mTurnRate; }

//当一个游戏事件消息传送到游戏管理器中时，这个函数就会被调用

void HandleGameEvent(const dtGame::Message& msg);

//该函数是一个虚函数，当角色进入游戏世界时（角色添加到GM时）

//调用该函数，在该函数中对游戏角色执行各种动作。游戏角色的属性

//和Invokable创建之后，在GM中的AddActor()函数中调用该函数。在

//这个函数中可以进行声音的播放，游戏事件的激发。也可以在这个函

//数中对在BuildInvokables（）中创建的Invokable函数进行注册，进行

//Invokable注册时要保持与其创建时一致的名称

virtual void OnEnteredWorld();

protected:

virtual ~TankActor() {}

private:

void ComputeVelocityAndTurn(float deltaSimTime);

void MoveTheTank(float deltaSimTime);

……

//游戏角色中的一些属性

dtCore::RefPtr<dtCore::ParticleSystem> mDust;

dtCore::RefPtr<dtCore::ParticleSystem> mCannonShot;

……

};

上述代码创建的是游戏角色TankActor，该类继承dtActors::GameMeshActor类，GameMeshActor类是一个具有静态网格的游戏对象的基类，在游戏管理器中可以管理从GameMeshActor派生出来的类，派生类可以建立简单的具有静态网格的游戏对象。在类声明中有一个TUTORIAL\_TANK\_EXPORT宏定义，该宏定义代表了\_\_declspec(dllexport)，在32位的编译器版本中，可以使用该宏定义代表的关键字从DLL导出数据、函数、类或类成员函数，这个宏定义在export.h头文件中，当我们将这个头文件包含进来，就可以转换为DLL输出。而与\_\_declspec(dllexport)相对应的是\_\_declspec(dllexport)，它用于将DLL动态链接库中的数据导入到应用程序中。

游戏角色代理类的声明如下所示：

class TUTORIAL\_TANK\_EXPORT TankActorProxy : public

dtActors::GameMeshActorProxy

{

public:

//创建游戏角色对应的属性窗口，在该窗口中可以对属性直接进行修改

virtual void BuildPropertyMap();

protected:

virtual ~TankActorProxy() {}

//创建游戏角色代理对应的游戏角色

virtual void CreateActor();

//当游戏角色代理加入到游戏管理器中时，该函数就会被调用，游戏角

//色中有一个相同的函数也会被调用，可以把代码加入到其中任意一个

//函数中

virtual void OnEnteredWorld();

};

游戏代理的类声明和游戏角色是相似的，它的类声明中也有一个宏定义，将类导出为DDL动态链接库，并且游戏代理类也继承了GameMeshActorProxy游戏代理基类。当定义完成游戏角色及角色代理类后，还需要对代理进行册，这样STAGE和GM才能找到自定义的新角色类型，角色注册类ActorsRegistry如下所示：

class TUTORIAL\_TANK\_EXPORT ActorsRegistry : public

dtCore::ActorPluginRegistry

{

public:

static dtCore::RefPtr<dtCore::ActorType> TANK\_ACTOR\_TYPE;

static dtCore::RefPtr<dtCore::ActorType> KILLABLE\_ACTOR\_TYPE;

ActorsRegistry();

virtual void RegisterActorTypes();

};

该类也在声明时使用了TUTORIAL\_TANK\_EXPORT宏，表示导出为DDL动

态链接库。同时，它继承了dtCore::ActorPluginRegistry，开发者就可以建立自己的角色注册器，ActorPluginRegistry作为一个对象工厂，知道如何通过角色类型来建立角色代理。它的静态成员TANK\_ACTOR\_TYPE会被赋值为相对应的ActorType对象，在RegisterActorTypes()函数中，会将代理和ActorType绑定起来：

void ActorsRegistry::RegisterActorTypes()

{

mActorFactory->RegisterType<TankActorProxy>(TANK\_ACTOR\_TYPE.get(

));

mActorFactory->RegisterType<KillableTargetActorProxy>(KILLABLE\_

ACTOR\_TYPE.get());

}

在将代理注册之后，角色和角色代理才算开发完成，本文中所开发的虚拟实验仪器类，与上面所述的TankActor类的开发是相似的，只是它的消息处理函数、属性是有所不同的。例如小灯泡，可以将其类声明为如下所示：

class GAUGE\_EXPORTS BulbActor:public dtActors::GameMeshActor

{

public:

BulbActor(dtGame::GameActorProxy& proxy);

void loadFile(const std::string& dir)

{

\_resourceDir=dir;

dtActors::GameMeshActor::SetMesh(\_resourceDir+"/bulb.3ds");

}

virtual void OnTickLocal(const dtGame::TickMessage& tickMessage);

virtual void ProcessMessage(const dtGame::Message& message);

virtual void OnEnteredWorld();

protected:

virtual ~BulbActor(){}

private:

std::string \_resourceDir;

……

};

如上所示，该类中的loadFile()方法将BulbActor类和模型bulb.3ds绑定在了一起，其中的bulb.3ds就是之前文中所述的三维模型。当在应用的主程序中加入这个动态链接库时，该BulbActor类就会被加入进游戏管理器中，此时函数OnEnteredWorld()就会被调用，而ProcessMessage(const dtGame::Message& message)函数当游戏角色接收到指定的消息时就会被调用。OnTickLocal(const dtGame::TickMessage& tickMessage)函数在游戏进行的每一帧都会被触发。

Delta3D有两种方式来创建应用程序，第一种是直接利用Delta3D提供的dtABC::Application类来创建，另一种是利用Delta3D提供的GameStart.exe和相应的入口程序来创建。下面我们介绍利用GameStart.exe来加载定义的游戏角色进行显示，GM（Game Manager）架构提供了一个独立可执行文件GameStart.exe和相应的应用程序入口，用来直接启动应用程序而不用将该应用程序编译为可执行文件。GameStart运行之后会寻找库中的GameEntryPoint类，GameEntryPoint类声明如下：

class TUTORIAL\_TANK\_EXPORT MyGameEntryPoint: public

dtGame::GameEntryPoint

{

public:

MyGameEntryPoint() {}

virtual ~MyGameEntryPoint();

//应用程序初始化时会调用该函数，命令行中的输入参数会传入到

//函数的参数中，这里直接使用基类实现即可，这个函数可以进行游戏

//早期的配置工作，比如命令行的解析，游戏窗口的配置，数据文件的

//路径设置等，但不要将有关GM的工作在此处进行

virtual void Initialize(GameApplication& app, int argc, char\*\*argv);

//OnStartup()函数，该函数在游戏循环开始之前调用，创建游戏管理器

//CreateGameManager()，组件的创建、地图的导入、角色的创建等工作

//所以该函数是一个非常重要的函数，程序的主要工作都在这个函数中

//完成，该函数是其父类的纯虚函数的重载函数，在GameApplication

//类的Config()函数中动态地调用，即通过GameStart程序的主函数中

//自动调用

virtual void OnStartup(dtGame::GameApplication& app);

private:

std::string mMapName;

……

……

};

这个类即应用程序的入口点，GameStart.exe程序可以像这样来加载：GameStart.exe TutorialGameActors，这个类中创建了游戏管理器和消息通信所需的组件。

### 4.2.2角色实现的关键问题

首先可以看到，所有的角色代理类都派生自dtActors::GameMeshActorProxy该基类中有一个成员函数：virtual void BuildInvokables();该函数会产生三个角色类中的消息函数，这些消息函数会将游戏角色带入游戏管理器的消息循环系统中去，游戏角色实现的关键性问题，即是要搞清楚这三个消息函数之间的相似和不同。消息函数声明如下：

virtual void OnTickLocal(const dtGame::TickMessage& tickMessage);

virtual void OnTickRemote(const dtGame::TickMessage& tickMessage);

virtual void ProcessMessage(const dtGame::Message& message);

其中，OnTickLocal()和OnTickRemote()功能是相同的，只不过针对的游戏对象是不同的，OnTickLocal()针对的是本地游戏对象，而OnTickRemote()针对的是远程的游戏对象。在游戏中每进行一帧它们都需要被调用一次，同时如果想要正常使用，必须在OnEnteredWorld()函数中注册消息，这样当游戏角色进入游戏管理器（GM）中时，GM才会对角色发送消息。OnEnteredWorld()函数中的代码如下所示：

RegisterForMessages(dtGame::MessageType::TICK\_LOCAL,dtGame::GameAct

orProxy::TICK\_LOCAL\_INVOKABLE);

RegisterForMessages(dtGame::MessageType::TICK\_REMOTE,dtGame::GameA

ctorProxy::TICK\_REMOTE\_INVOKABLE);

ProcessMessage()函数是通用的消息处理函数，它可以接受任意的Message对象，而在Delta3D的消息模块中，dtGame::Message是所有消息的基类，同样，如果想要使游戏角色正常接收到消息，必须要在OnEnteredWorld()函数中注册消息，这里可以对接收的消息进行选择。这样当游戏角色进入游戏管理器（GM）中时，GM才会对角色发送消息。

当制作完成游戏角色和游戏角色代理，得到它的DLL动态链接库时，还需要手动将动态链接库加入游戏管理器中，类似的代码如下所示：

\_gameManager->LoadActorRegistry("GaugeDesign");

## 4.3 虚拟实验演示

根据上节所述，本节创建了一系列游戏角色例如：BulbActor、PipeActor、AmmeterActor、PowerActor、SwitchActor等。并且将它们与二级界面的按钮进行绑定，当用户点击二级界面中相应的功能按钮后，在三维窗口中将显示虚拟仪器。本节根据显示的三维仪器进行了手动的组装，用来展现虚拟电学实验，使这套实验平台真正具有功能实现。本虚拟实验系统具有灵活性、扩展性等特征。传统的教学实验平台，教学内容一般是固定的，学生需要根据教学说明的指引来一步步进行，这种类似于教学短片的实验并不具有灵活性。而本实验平台，用户可以根据自己的需要在游戏场景中任意拖动、组装实验，这样就具有非常大的灵活性。而只要具有相对应的游戏角色及模型，就可以将其绑定在用户界面上，所以系统的后期扩展也并非难事。

串并联电路实验是中学物理实验中一个非常重要的知识点，它探究了串联电路和并联电路中各点电流之间的关系。本次实验中需要的实验器材如下：干电池、小灯泡、开关、电流表、电压表、导线。设计实验，首先将两只灯泡串联起来，组成一个简单的串联电路，在这个串联电路中，选取一个关键点用电流表测出这点的电流，而每个小灯泡都用电压表测量它的电压。实验组装如下图所示：

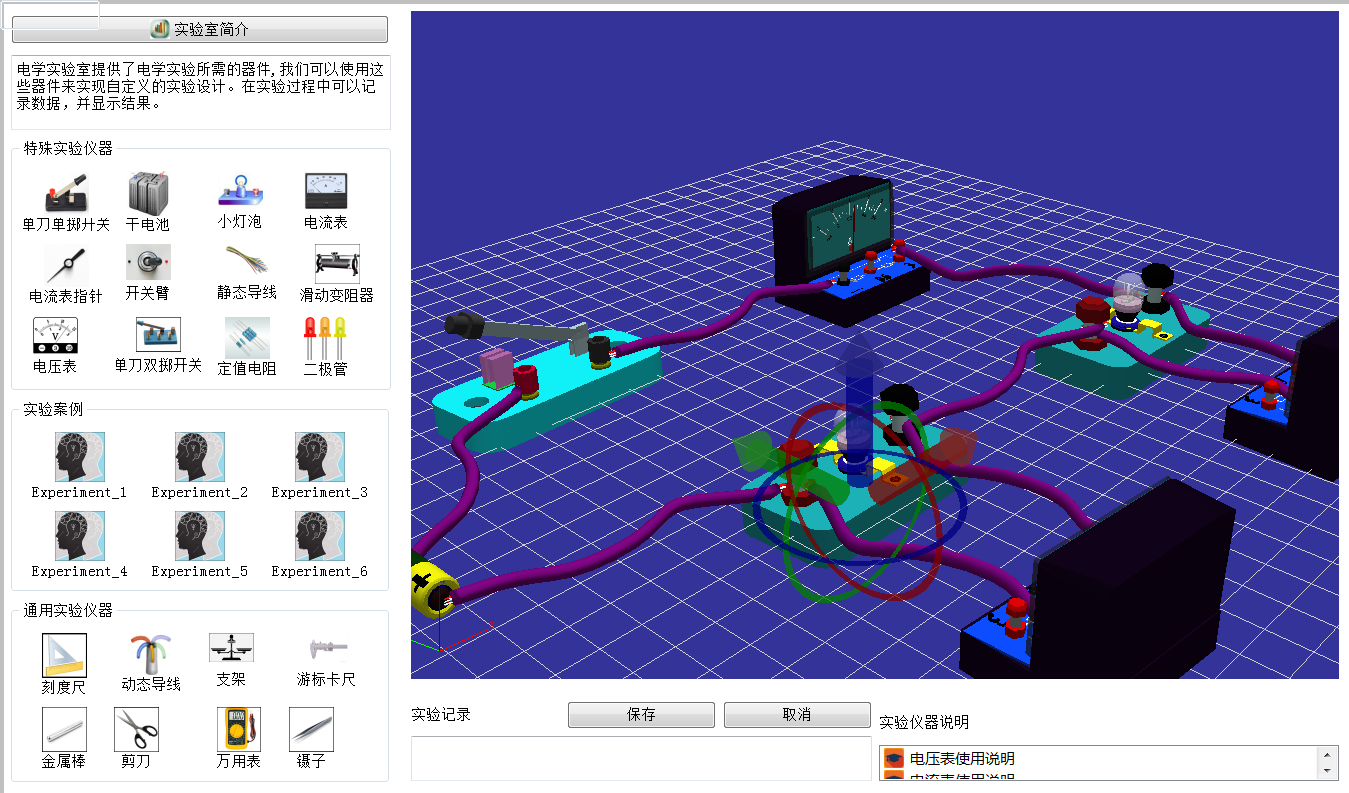


图4-6 串联电路准备

Fig.4-6 Series circuit preparation

从图中可以看出串联电路的实验准备阶段，在本平台中所使用的虚拟仪器为干电池、小灯泡、电流表、电压表、开关、导线，其中的场景对应于实验案例中的一个实验。导线连接确认无误，当开关闭合时，会产生和正常物理实验一样的效果：

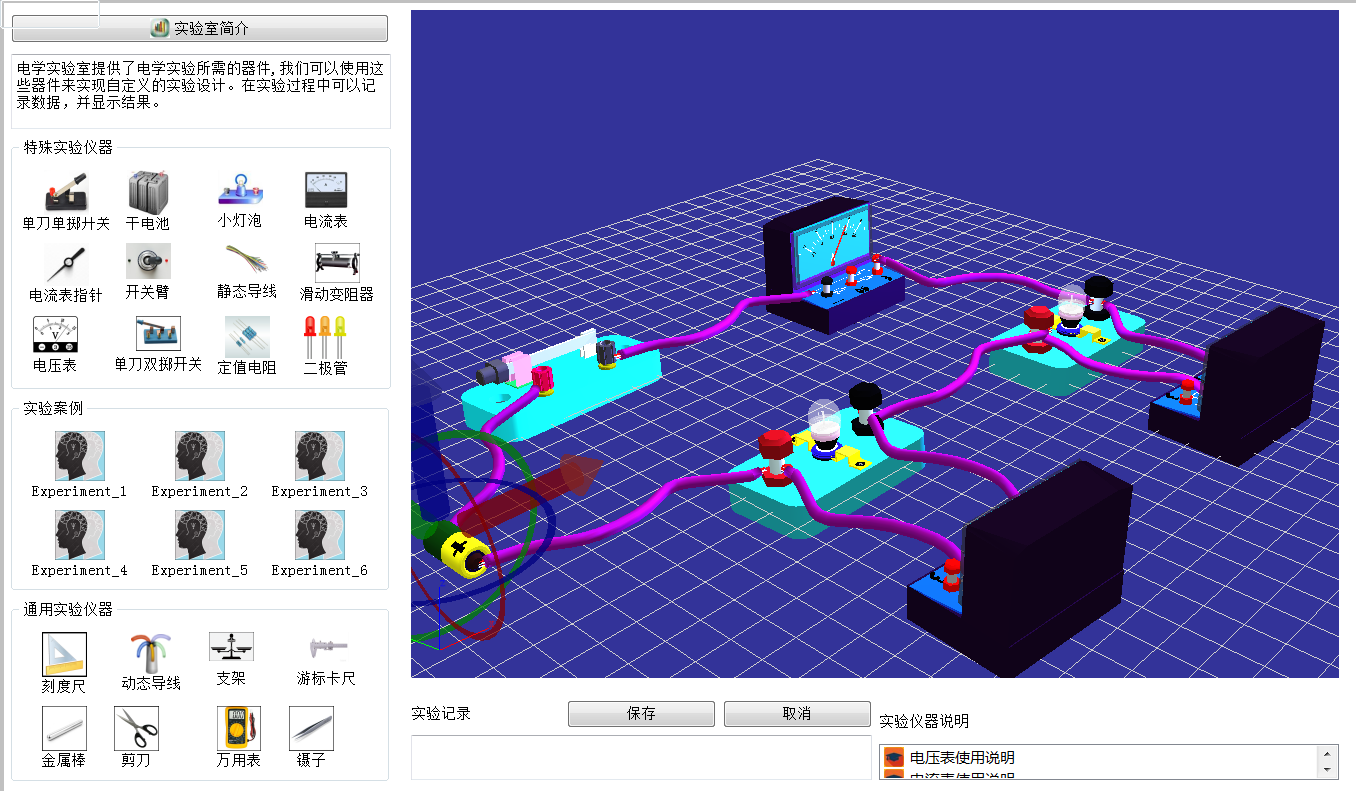


图4-7 串联电路实验

Fig.4-7 Series circuit experiment

当开关闭合时，可以使小灯泡发光，而电流表指针会进行偏转，从图中可以看到这个效果。在并联电路设计时，需要使用如下的虚拟仪器：干电池、小灯泡、开关、导线、电流表。将两只灯泡并联起来，组成一个简单的并联电路，在这个并联

电路中，使用电流表分别测量两个支路的电流，与串联电路中的电流表做比较。并联电路的实验组装如下图所示：

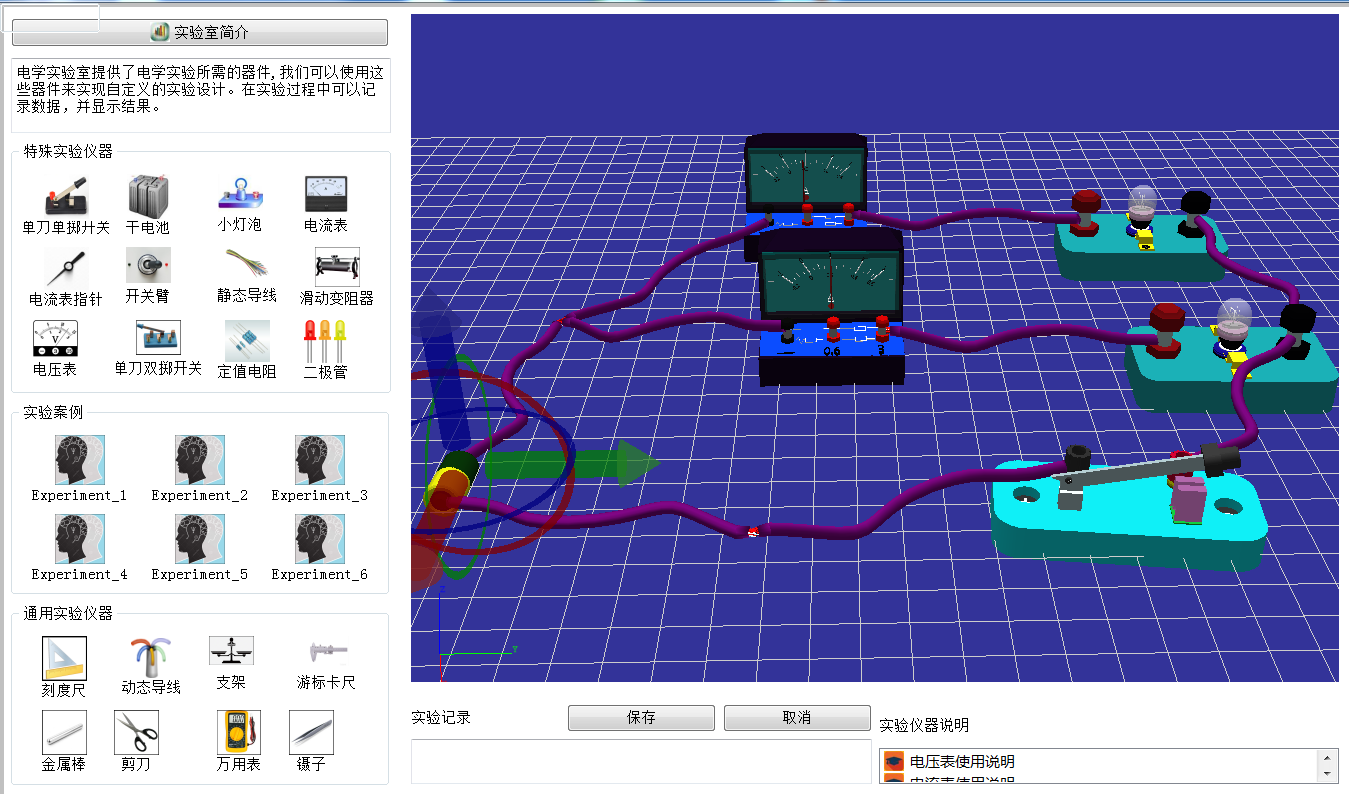


图4-8 并联电路准备

Fig.4-8 Parallel circuit preparation

从上图中可以看出，从干电池中引出了两条并联的导线，分别连接电流表和小灯泡，之后由开关来控制主干线路的联通。当导线连接无误后，闭合开关会产生实际实验时的效果：

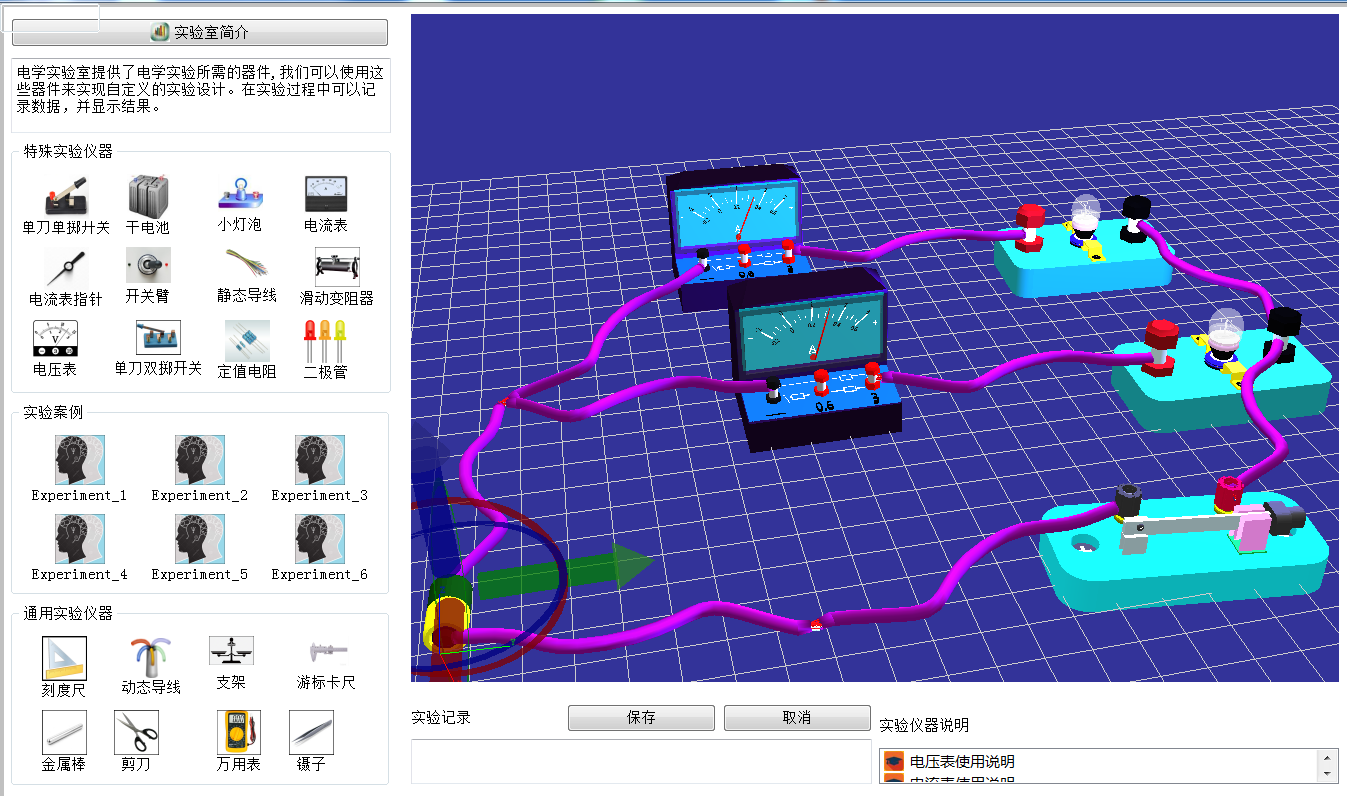


图4-9 并联电路实验

Fig.4-9 Parallel circuit experiment

当开关闭合时，可以使小灯泡发光，而电流表指针会进行偏转，从图中可以看

到这个效果。本实验系统在中小学生中进行了简单的展示，根据收到的反馈来看，优点和缺点是并存的，首先系统具有的优点是功能齐全，界面简易，通过演示三维的物理实验可以更好的引领学生进入边学边玩的状态，将枯燥的物理实验融入到有趣的游戏当中。而系统存在的不足则是：第一，系统优化不够全面，应用程序反应过慢。第二，实验仪器需要手动进行拼装，在这个过程中需要耗费大量的时间。

# 总结与展望

本文基于Delta3D游戏引擎开发了三维虚拟物理实验室平台，利用QT界面开发工具开发了实验平台的功能界面。该界面基于模块化的设计理念，使设计出来的各个模块之间的功能相互独立，该界面具有一定的通用性、科学性。之后利用Delta3D游戏角色的概念对该界面的功能模块进行了实现，使用三维建模软件3DS MAX设计了一套与之相关的模型，并将模型与游戏角色进行了绑定。在本文的最后，利用设计好的虚拟仪器完成了一个物理电学实验的演示。

本文研究的重点为虚拟实验室中虚拟仪器的设计与实现。其中，界面是人机交互的通道，而本文所设计的虚拟仪器则需要在界面当中体现出来。文章首先分析了界面设计的基本原则和设计理念，在科学与规范的理论指导下，深入分析了三维物理实验室作为一个教育类软件，其界面设计的科学理论，之后，本文选择了几个代表性的界面系统进行了简要的介绍。

在界面系统的实现阶段，本文重点分析了Delta3D游戏开发引擎与界面开发工具Qt及其辅助工具。同时对界面系统开发的各个GUI模块实现做了详细阐述，不仅如此，本文还对界面开发过程中遇到的问题进行了深入的分析，在界面中各模块功能实现的同时，增强了自身分析问题的能力，编程开发的能力。之后，对三维物理实验室中的二级界面和各个模块进行了文字描述和图片展示。

在文章的最后，介绍了3DS MAX三维建模软件及其在虚拟现实系统中的应用。由于3DS MAX具有强大的三维建模功能，而且简单易学，价格低廉，所以在虚拟实现中被广泛使用，本文也介绍了几款当前主流的建模软件并比较了它们之间的异同。之后，本文设计了几款电学实验中使用到的模型，用图片进行了展示。而在Delta3D的游戏角色概念中，使用了C++面向对象的特性来设计角色的实现，详细阐述了游戏对象设计过程中需要注意的问题，以及它的回调函数的功能。当游戏角色完成之后，就可以利用它们来设计虚拟实验了，本文设计了一个常用的中学电学实验用来展示它的功能。

三维虚拟实验室具有很强的教学研究意义，具有广泛的使用价值。本文设计的软件系统使学生在枯燥的课堂当中找到学习的乐趣，体现出了“寓教于乐”的原则，同时也激发了学生的创造性，主动设计实验，加深对知识的掌握。

# 参考文献

[1] 胡珊，于光，周明秀. 基于3D和VIRTOOLS技术的物理虚拟实验室架构设计[J]. 计算机工程与设计，2008，29(1): 206-209.

[2] 李伟. 基于VRML工程制图虚拟实验室研究与实现[D]. 浙江工业大学，2008.

[3] 江峰. 3D游戏引擎研究与实现[D]. 浙江大学, 2005.

[4] A Ferrero, V Piuri. A simulation tool for virtual laboratory experiments in a WWW environment[J]. IEEE Transactions on Instrumentation & Measurement, 1999, 48

(3): 741-746.

[5] 胡丽娜. 浅谈高中物理教学中多媒体技术的运用[J]. 新课程研究：基础教育, 2010(3): 174-175.

[6] 夏岩. 略论基于JAVA的网络实验室的设计[J]. 中国科技博览, 2010(9): 105-105.

[7] D Raineri. Virtual laboratories enhance traditional undergraduate biology laboratories[J]. Biochemistry & Molecular Biology Education, 2001, 29(4): 160-162.

[8] B Stiller，C Class，M Waldvogel. A flexible middleware for multimedia communication: design, implementation, and experience[J]. IEEE Journal on Selected Areas in Communications, 1999, 17(9): 1580-1598.

[9] S Kangbo，H Jungdong. Establishment of sesame cultivation practices as succeeding cropping system of watermelon in the greenhouse condition[J]. Korean Journal of Crop Science, 2010, 55(3): 195-199.

[10] 檀鹏，张树杨. 虚拟现实开发平台系统的研究及实现[J]. 江苏师范大学学报(自然科学版), 2005, 23(2): 57-59.

[11] 成迟薏，潘志庚. 递进网格的一种快速生成算法[J]. 中国图象图形学报, 1998, 3(11): 946-951.

[12] 黄慕雄. 高等教学型虚拟实验室建设的现状与建议[J]. 电化教育研究, 2005, 9: 77 -80.

[13] 童怡青. 未来手机的用户界面设计发展[J]. 通信世界, 2005(13): 35-35.

[14] 胡润波. 基于第三方信息的移动商务信任评价方法研究[D]. 大连理工大学,

2012.

[15] P. J. Mosterman, M. A. M. Dorlandt. Virtual engineering laboratories: design and experiments[J], J. Eng.Educ . , 1994, 83: 279–285.

[16] M Quek, D Boland. Simulating the feel of brain-computer interfaces for design, development and social interaction[J]. International Conference on Human Factors in Computing Systems, 2011: 25-28.

[17] CY Wong. Mobile User Interface for Seniors: an Impact of Ageing Population on Mobile Design[J]. Design Principles & Practices, 2010, 4(4): 231-248.

[18] 尹治宏，丁鹏. 一个引入用户参与的人机界面模型的研究[J]. 计算机工程，2003，29(8): 84-86.

[19] 张翱旻. 人机界面设计研究[D]. 上海交通大学，2007.

[20] 陈超，鱼江海. 降低复杂算法对人机界面实时性影响的探讨[J]. 指挥信息系统与技术, 2010, 01(5): 55-58.

[21] Kan Qiao, Jinyuan Jia. An extensible and lightweight framework of game GUI[J],

Proceedings of the 10th International Conferenceon Virtual Reality Continuum and Its Applications in Industry, 2011, Dec: 601 - 604.

[22] 缪琦. 多域单点登录系统的设计与实现[D]. 北京邮电大学, 2007.

[23] 郝欣. 以用户为中心的DCS操作员站人机界面设计[J]. 东北电力技术, 2005, 26(5): 47-49.

[24] YM Cheng，SH Kuo. The Construction of an Online Competitive Game-Based Learning System for Junior High School Students[J]. Turkish Online Journal of Educational Technology, 2012, 11(2): 214-227.

[25] Q Miao，F Zhu. A Game-Engine-Based Platform for Modeling and Computing Artificial Transportation Systems[J]. IEEE Transactions on Intelligent Transportation Systems, 2011, 12(2): 343-353.

[26] P Mcdowell，R Darken. Delta3D: A Complete Open Source Game and Simulation Engine for Building Military Training Systems[J]. Journal of Defense Modeling & Simulation, 2006, 3(3): 143-154.

[27] 唐毅. 3D游戏场景编辑器的设计与实现[D]. 西南交通大学, 2010.

[28] Jasmin Blanchette. C++ GUI Programming with Qt 4[M]. 电子工业出版社，2013.

[29] 张准. MAYA在三维动画方面的优势[J]. 艺术科技, 2013, 26(3): 16-16.

[30] 孙英泽，胡婧. Google Earth应用于基建项目管理系统的研究与实现[J]. 农业网络信息, 2012(9): 12-16.

[31] S. Diwakar, K. Achuthan, P. Nedungadi. Biotechnology virtual labs-integrating wet-lab techniques and theoretical learning for enhanced learning at universities[J],

International Conference on Data Engineering, 2010: 10-14.

[32] 杨潇. 基于Qt的富客户端软件设计与实现[D]. 电子科技大学, 2009.

[33] Z Xu-Feng, JJ Zhu. 3ds MAX Modeling and its Application in Virtual Reality[J]. Computer Simulation, 2004, 21(4): 94-97.

[34] K Abdelmalek, J Yang. Towards a new generation of virtual humans[J]. International Journal of Human Factors Modelling & Simulation, 2006, 1(1): 2-39.

[35] L Zhou, W Guo. SketchUp aided dynamic landscape design—A case of Helv Fuan shopping street landscape design of Hudai[J]. IEEE International Conference on Computer-aided Industrial Design & Conceptual Design, 2010, 2: 1067-1070.

[36] L Flavell. Beginning Blender: Open Source 3D Modeling, Animation, and Game Design[M]. Apress, 2010.

[37] 孙倩. 基于3DS MAX的三维建模及其在Virtools环境中的应用[J]. 中国科技信息, 2008(12): 94-95.

# 学位论文独创性声明

本人郑重声明：所提交的学位论文是在导师的指导下展开的研究工作所取得的研究成果。据我了解，本文除了特别加以标注以及致谢外，不涉及其他人已经发表过或撰写过的劳动成果。对本研究所做的任何贡献的同志们均已在本文中作了特别说明，并表达了谢意。本人依法享有和承担由此论文所产生的权利和责任。

论文作者签名： 日期：

# 学位论文版权使用授权声明

本学位论文作者完全了解学校有关保存、使用学位论文的规定，同意授权广东工业大学保留并向国家有关部门或机构送交该论文的印刷本和电子版本，允许该论文被查阅和借阅。同意授权广东工业大学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、扫描或数字化等其他复制手段保存和汇编本学位论文。保密论文在解密后遵守此规定。

论文作者签名： 日期：

指导教师签名： 日期：

# 致谢

时光荏苒，我的研究生生活即将结束，回顾这几年的生活、学习，其中充满了辛酸和困惑，更有收获和成长。在这三年来，感谢陪伴我一起度过美好时光的每一位老师和同学，正是他们的帮助，我才能解决疑问，直到学业完成。

本人的学位论文是在我的导师刘东峰教授的关怀和指导下进行并完成的，衷心感谢刘老师对我的悉心关怀。自进入实验室的第一天起，我就被刘东峰老师严谨的治学态度所感染，从这以后，我就跟随刘老师进行课题学习。从课题的选择，技术的实现，一直到最后论文的完成，刘老师都始终给予耐心的引导和支持，让我的思维不断发散，逻辑不断变得明确。

同时也感谢实验室的何家峰老师和程昱老师，他们也对我的学习、生活起到了很大的帮助，在实验室学习的这三年中，两位老师也给了我很多的意见和指导，从而让我快速的进入角色，找到课题的突破口。

感谢实验室中的师弟们，两年来我们朝夕相处，相互关心。每一次对技术的交流都开阔了我们的视野。

最后，感谢我的父母在我的学习生涯中所给予的支持和包容，感谢帮助和关心我的所有的同学和老师，在此表示诚挚的谢意!

张翔霄

2017年4月9日