分类号 TP393 密级 公开
UDC 004.7 编号 10299Z1308012

江苏大學

硕士学位论文

基于 Unity3D 的体感游戏系统的研究

The Research on Motion Sensing Game Based on Unity3D

指导教师 陈健美 作者姓名 阚宇

申请学位级别 硕士 专业名称 计算机技术

论文提交日期 2016.4 论文答辩日期 2016.6

学位授予单位和日期 江苏大学 2016 年 6 月

答辩委员会主席 宋余庆

评 阅 人

独 创 性 声 明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已注明引用的内容以外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果，也不包含为获得江苏大学或其他教育机构的学位或证书而使用过的材料。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

年　月　日

学位论文版权使用授权书

江苏大学、中国科学技术信息研究所、国家图书馆、中国学术期刊（光盘版）电子杂志社有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致，允许论文被查阅和借阅，同时授权中国科学技术信息研究所将本论文编入《中国学位论文全文数据库》并向社会提供查询，授权中国学术期刊（光盘版）电子杂志社将本论文编入《中国优秀博硕士学位论文全文数据库》并向社会提供查询。论文的公布（包括刊登）授权江苏大学研究生处办理。

本学位论文属于不保密 □ 。

学位论文作者签名：　　指导教师签名：

年　月　日　　年　月　日

摘 要

近年来，各大公司都加大了在体感设备的投入，比如微软、索尼、HTC、Facebook等公司纷纷推出自己的体感设备。而将体感设备与游戏产业结合在一起的研究一直都在进行着。游戏产业也是现在的一个热门产业，尤其是随着这些体感设备的发布，使用最自然的方法来进行人机交互的体感游戏，也越来越受到玩家的期盼。体感游戏可以让玩家更体验到一种身临其境的感觉，并且能同时体验游戏的趣味性和培养自己身体的灵活性。但是目前为止还没有出现一款真正流行起来的体感游戏，多数开发者还在处于技术摸索中。

为了解决以上问题，本文用 Unity3D 游戏引擎进行游戏主体的开发，通过将 Kinect 作为输入设备进行一款体感游戏的设计和制作。通过对 Kinect 技术的研究和梳理，结合 Unity3d 引擎设计了游戏的主要框架结构，设计了不同游戏模块的具体实现；并使用有限状态机和优化后的 A*寻路算法为核心的算法设计了敌人的 AI；最后将 Kinect 接入 Unity3D 客户游戏中去。

论文研究的主要内容有：

1.从之前游戏开发的经验入手，对游戏的玩法和主题进行策划，设计了总体的开发流程。通过有限状态机和 A*寻路算法来设计高智商的怪物从而增加游戏的趣味性。

2.设计并修改 KinectSDK 的类库，将 Kinect 自带的方法根据系统的需求进行重新封装，为游戏客户端的调用提供接口。

3.完成基于 Unity3d 平台的游戏具体设计和制作，设计并完成资源的动态加载、粒子系统、图形界面的设计、脚本系统的设计等各个模块，并将各个模块整合起来完成游戏的客户端。

关键字：Kinect；Unity3D；体感游戏；A*寻路算法；VR 设备

ABSTRACT

In recent years, major companies have increased the input in the motion sensing equipment, such as Microsoft, SONY, HTC, Facebook and other companies have launched their own sense of the device. And the research on the combination of the equipment and the game industry has been carried out. Game industry is now a hot industry, especially with the release of these devices, the use of the most natural way to conduct a sense of human-computer interaction game, but also more and more players are looking forward to. Motion sensing game allows players to experience a more immersive feeling, and can experience the fun of the game and cultivate their own physical flexibility. But so far there has not been a real popular sense of the game, the majority of developers are still in the technical exploration.

In order to solve the above problems, the development of the Unity3D game engine game subject, by using Kinect as an input device for a sense of design and production of the game. Through the Kinect technology research and combing, combined with unity3d engine design the structure of the main framework of the game, the realization of different game module is designed; and using finite state machine and optimization of A * search algorithm as the core algorithm design the enemy AI; the Kinect access unity3d game client.

The main contents of this paper are:

1 Based on the previous game development experience, the game play and theme planning, design the overall development process. Through the finite state machine and A* routing algorithm to design a high IQ of the monster to increase the fun of the game.

2 Design and modify the KinectSDK class library, Kinect comes with the method based on the needs of the system to re package, to provide interface for the game client.

3. Complete based on unity3d platform specific game design and production, design and resources to complete the dynamic loading, particle systems, the graphical interface design, script system design of each module, and each module together to complete the game client.

Key words: Kinect, Unity3D, Motion Sensing Game, A* routing algorithm,VR equipment

目 录

第一章 绪 论

本文主要是研究并实现一款针对于Kinect的体感游戏。随着近几年多款体感设备的发布，体感游戏已经成为一个热门话题。本文主要是设计了一个基于Unity3D和Kinect技术的体感游戏系统，提出了一个体感游戏系统的解决方法和设计思路。本章主要是介绍本文的研究背景，当前国内外体感游戏和体感设备的研究现状和本人这篇论文的主要工作和本文的组织结构。

1.1 课题研究背景与意义

1.1.1 研究背景

游戏产业即是一种新兴型产业也是一种高科技型产业，是我国的文化产业的一部分。游戏产业虽然属于一种休闲产业，但是游戏产业涵盖了互联网、计算机、软件、电子电信等各种高新产业部门。游戏产业存在着无比巨大的影响力，影响着与之相关联产业的方方面面。

体感游戏，顾名思义，是一种人通过身体感觉来操控的游戏。与之前的单纯使用鼠标键盘或者手柄作为操作方法的游戏不同，体感游戏是通过玩家的身体不同动作和声音等自然行为作为输入来操控角色的新型电子游戏[1]。体感游戏作为电子游戏的新分类，虽然之前游戏都是采用了键盘鼠标或手柄作为输入设备，但是现在随着科技的进步和玩家对于游戏体验感需求的增加，更多的游戏公司针对各种不同的体验进行了特异化的输入设备和街机游戏的开发。近些年越来越多的玩家使用 PS、XBOX 和 PC 作为游戏平台，这三大主机平台的飞速发展使得体感游戏具备了最基本的硬件基础和发展方向，从而成为当今游戏界的一个主流发展方向。就像开发过《文明》系列游戏的著名游戏设计师席德梅尔说过的那样，将来电子游戏的趋势是由拥有虚拟现实技术的体感游戏来决定，与游戏有趣的互动才是未来电子游戏的发展方向，因为这类游戏十分吸引玩家[2]。体感游戏的设备发展进程如图 1.1 所示。

图 1.1 体感游戏设备的发展历程

1.1.2 研究的意义

随着目前的网络技术和 VR 技术等高新技术的飞速发展，并且越来越多的应用在生活中得到使用[3]，使得之前一些基于桌面操作和手机的一些普通操作的应用逐渐向一个更为巨大的空间进行拓展， 从而能够更好地进行操作，所以现在使用体感行为进行操控设备不仅仅停留在研究的热点这个阶段了，更重要的是已经成为现在各种新型应用所急切需要的。就目前来说，在家庭、娱乐、公共场所的展示和舞台的设计中已经开始使用了越来越多的体感设备。与传统的鼠标键盘等操作方法相比，这种使用人体的动作和行为的操控方式使人们获得了更加强烈的真实感从而有一种“身临其境”的感觉。所以对于将来的游戏开发来说，怎样设计出一个脱离老式的鼠标键盘操作并且能同时使游戏的娱乐性和调动玩家的积极性兼备的游戏已经成为了未来游戏设计的标杆。虽然近年来越来越多的健身运动游戏的不断出现，但是完全脱离鼠标键盘操作的老式游戏模式的游戏却十分稀少，设计和实现这种游戏仍然是一个不小的挑战。

我国现在正在大力投入和发展文化事业，游戏产业是文化产业的重要组成部分，尤其是现在人们对于新型的体感交互游戏有着巨大的需求，所以体感游戏拥有非常巨大的市场前景。本文以最新的 Unity3D 和 Kinect 的技术作为基础，通过编写和封装中间件来使用 Kinect 作为输入设备来获取玩家的行动，计算出玩家的基本参数，将这些计算过后的数据作为操作设备用于本文设计的游戏系统中。Kinect 是一种新

型体感设备，能非常详细的记录人体的各种数据[4]。Kinect 设备的摄像设备主要由红外摄像头和普通的彩色摄像头组成，红外摄像设备的作用是采集红外数据并将采集来的数据进行分析和计算，计算结束后会产生深度数据[5]。而 Kinect 的摄像头则采集普通的平面图像数据，平面图像和深度数据综合计算后能计算出采集人的具体信息。本文中不仅是以设计和开发一款体感游戏作为目的，并且同时以虚拟展示、体感操作等应用作为知识储备，并将这项技术拓展到其他的体感设备中。

1.2 国内外发展现状

现如今，科技在飞速的进步和发展，尤其是在人机交互领域，越来越多的大型公司将精力聚焦在这上面。现在这种人机交互领域正在进入一个飞速发展的时代，索尼的 PSVR、微软的 Kinect、任天堂的 Wii、HTC 的 vive 等体感设备如雨后春笋般出现。人们也越来越希望这种通过最自然的交互方式的设备出现在现实的生活中，而体感技术则是一种集合了各种技术为一体的高新技术，如：计算机图形学、移动交互、虚拟现实等技术。从现代设备的出现以来，与机器的交互方式已经变得越来越简单和方便，鼠标和键盘的出现曾经引起了一场输入设备的技术革命，但是现在这种以自然的方式进行体感交互的设备的出现不仅满足了人们长久以来的期盼，更有可能的是，它甚至会引起另外一次关于输入设备的革命。任何事物的出现都不是一蹴而就的，为了实现这种使用体感技术的操作设备，无数的开发者和设计师为此付出了辛勤的努力，他们也完成了许多的成果，以下就是当时引起过轰动的体感设备：

1.跳舞毯

跳舞毯是当时最早出现的一种拜托传统输入设备的一种尝试。玩家一边根据听到的音乐，另一边根据屏幕中会不断出现的指示箭头，然后通过用脚来踩踏跳舞毯上的箭头区域进行互动，玩家通过踩踏与屏幕中出现相同的箭头从而达到了根据节拍让身体舞动来完成跳舞和游戏。跳舞毯当时风靡一时，因为它巧妙的将娱乐和健身结合在一起，通过跳舞作为载体，抛弃了传统的鼠标键盘输入，而是通过使用一个拥有传感器的毯子作为输入设备。跳舞毯这种创新性的改革设备一出现再加上这种娱乐和健身为一体的操作方式，当时立刻就让全球的人们喜欢上了这种游戏。跳舞毯也是体感游戏大获成功的一种创新性实验。跳舞毯如图 1.2 所示。在现在看来当时的跳舞毯虽然取得了巨大的成功，但是还是将玩家限制在跳舞毯上面，还不全

是彻底摆脱了输入设备。但是这第一款的体感游戏也为之后的体感游戏开发提供了参考。

图 1.2 跳舞毯

2.Wii

Wii 是 2006 年任天堂公司推出的一款体感游戏设备[6]。如图 1.3 所示。虽然体感设备在 Wii 之前已经出现了许多，体感操作的概念虽然在以往的游戏中已经出现过，但是 Wii 还是取得了巨大的成功。Wii 成功的原因是因为它的设计理念，Wii 加入身体操作的最主要目的，不是为了增加一种玩法，而是为了简化游戏操作。所以 Wii 的手柄按键十分的少，而是加入了对一般人来说学习简直无成本的身体操作，这才是 Wii 能成功的真正因素。Wii 成功帮助拓展了远超过去传统游戏市场规模的轻度用户群体，第一次令游戏产业意识到蓝海市场的庞大，极大地改变了从业者对传统游戏市场的认识，催生出了以轻度用户为目标的新游戏类型。

图 1.3 Wii

3.HTC Vive

2015 年 HTC 公司在巴萨罗那发布了与 Valve 一同研发的头戴式 VR 设备 Vive。HTC Vive 由三个主要组件组成：一个头戴式显示器、两个单手持控制器和一个追踪玩家的定位器。HTC Vive 一经发布就赢得了众多玩家的眼球，因为该产品与其他的体感设备相比有很多优势，比如使用该设备玩家很少有晕眩感，而且试玩 HTC Vive 的游戏十分精致且有游戏性。产生这些优点的原因主要在于 HTC Vive 拥有相当高端的摄像头采集设备，这款摄像头的帧率达到 90 帧每秒，并为每个眼球都提供了一个 1200 x 1080 的屏幕可以让玩家消除图像的颗粒感，并且采用了相当准确的算法来有效去除抖动，而且该体感设备与包括 Valve 在内的 12 家世界范围内著名游戏厂商合作[7]。

图 1.4 Htc Vive

4.Kinect

Kinect 一开始是在 09 年在美国 E3 展会上公布的，一开始 Kinect 是作为微软的 XBOX360 的一种外设[8]。Kinect 的出现彻底使人们向往这种体感交互方式的自然交互理念，从而对之前使用鼠标键盘等操作的游戏发起了强力的挑战。Kinect 不仅仅是一个简单的摄像机，同时它导入了即时动态捕捉、影像辨识、麦克风输入、语音辨识、社群互动等功能。可以使玩家使用体感这种自然的交互方式进行游戏，与其他玩家的交流等。

综上可知，体感技术现在已经成为现在研究的一大热点，利用体感技术来增强游戏的体验感[9]。现在国内外对此的研究也越来越多，虽然国内目前还没有成熟的体感游戏，但是已经有很多的游戏公司和私人开发者开始了体感设备进行游戏的开发了。

1.4 论文主要工作

本文研究的主要目的是设计和完成一个体感游戏系统。该系统使用了目前非常火热的 Unity3D 和 Kinect 技术，设计并完成了一个使用 Unity3D 引擎的体感游戏系统。并且对 A*寻路算法进行了介绍和优化，用于本系统怪物的自动寻路中，取得了不错的效果。本系统主要以完成一个体感游戏系统为主要目的，该系统拥有十分优秀的效果。用于为以后体感游戏的开发提供积累并加深自己对体感游戏的理解。

1.5 论文结构安排

本文主要研究并实现了一个体感游戏的设计与实现，并将其中使用的主要技术和如何实现进行介绍。本文主要分为 6 个章节，每个章节的组织结构如下：

第一章 绪论部分，主要是介绍当前体感游戏的研究现状以及本体感游戏系统研究的目标和意义。然后将国内外目前体感游戏设备的发展进行了介绍，并将本人的工作成果展示出来，然后列出组织结构。

第二章 介绍本体感游戏系统使用的相关技术，首先先介绍游戏引擎 Unity3D，介绍了 Unity 的一些基本的框架。然后介绍了微软的 Kinect 体感输入设备。

第三章 该体感游戏的系统设计。首先设计了系统的总体设计，并设计好游戏的基本玩法和玩家的操作方法；然后使用有限状态机来控制玩家和怪物的状态切换；最后介绍 A*寻路算法，通过实现对 A*寻路算法使用合适的启发函数等方法进行优

化，设计成最合适于本系统的算法后加入系统中。

第四章 Kinect 作为输入设备的封装和实现，主要包括如何安装和使用 Kinect SDK 与设计和完成 Kinect 与游戏系统整合的模块。本章设计了两个主要的类用于将 Kinect SDK 中的主要方法封装。并在 Visual Studio 中封装成 dll 文件给 Unity3D 客户端程序调用。

第五章 完成基于 Unity3d 平台的游戏具体设计和制作，设计并完成资源的动态加载、粒子系统、图形界面的设计、脚本系统的设计等各个模块，并将各个模块整合起来完成游戏的客户端。同时展示了使用 Kinect 作为输入设备后的游戏效果。

第六章 总结与展望。

第二章 体感游戏系统的相关技术

体感游戏在将来将成为最为主流的游戏种类。本系统则使用了目前非常热门的 Unity3D 作为游戏引擎并结合微软的 Kinect 技术，将两种热门技术搭配在一起使用的轻型体感游戏系统。

2.1 Unity3D 相关技术

Unity3D 是由 Unity Technologies 公司研发的一个可以轻松让开发者创建 3d 游戏、基本建筑、3d 动画模型等各种内容技术相结合并能同时发布到多平台的专业游戏引擎[10-12]。Unity3D 既可以运行在 Windows 操作系统下也可以运行在 Mac OX X 下，可以同时将游戏发布到各种主流平台比如：Windows、Mac、Wii、iPhone、Windows phone 8 和 Android。也可以直接使用 Unity 自带的插件将游戏直接发布为可以同时支持 Mac 和 Windows 支持的网页游戏[13]。

2.1.1 主流游戏引擎介绍

游戏引擎是已经被做好的可以按自己要求编辑游戏系统或者一些应用程序的交互式核心组件[14]。游戏引擎为游戏的开发人员提供了各种工具来编写开发游戏，主要是为了能够让游戏开发者更简单和迅速的完成一款游戏而不是全部重新开始。最开始开发游戏引擎的目的是为了减少和避免重复性的开发，从本质上来说游戏引擎就是游戏的主题框架和为这些游戏提供的核心代码。游戏引擎提供的这些核心代码和框架又同时适用于各种其他的的游戏中，所以这些核心代码和框架单独拿出来封装成游戏引擎用于各种不同的项目中来节省开发者的时间。所以，游戏开发者使用游戏引擎进行开发可以节省大量的时间和减少前期工作量。

下面是目前热门的游戏引擎：

Cocos2d：Cocos2d-x 是一个比较轻量形的 2d 游戏引擎，它的特点是开源和跨平台，这也是该游戏引擎最大的优点[15]。它的跨平台性使它支持目前所有的主流平台：Android、IOS、Linux、Windows Phone 等平台。对于开发者来说，Cocos2d 的社区支持对于开发者来说更是有着非常大的帮助。目前，来自于微软、谷歌、三星、英特尔等跨国公司的精英游戏开发者为 Cocos2d 贡献了大量的优秀代码。Cocos2d 也具有非常广泛的应用，国内外顶尖的游戏公司和普通的小型公司都开始使用该引

擎作为他们的移动平台的游戏引擎，已经有大量的网页游戏通过该引擎移植到移动平台上，如：忘仙 OL、神仙道等。

UDK(虚幻引擎开发工具)是虚幻 3 引擎的免费版本[16]。目前，有许多使用虚幻 3 引擎开发的大型游戏，比如 EA 公司的境之边缘、卡普空公司的鬼泣、失落星球 3、2k 公司的生化奇兵等等。UDK 和虚幻引擎一样都拥有最顶级的颜色、灯光、阴影、渲染质量和帧率。并且拥有大量第三方的插件来辅助开发者进行更简单的开发，比如 Face FX 可以帮助面部动画的制作，speedtree 可以用于生成各式各样的树木的生成，Substance 可以帮忙制作纹理等等。开发者可以免费下载和使用 UDK 引擎进行游戏的开发，但是如果开发者要用于商业时则先要付款 99 美元，并且在开发者在使用 UDK 引擎的收入超过 5000 美元的收入要支付 25%的盈利部分给 UDK 引擎公司。

Unity3D 是一个可以轻松让开发者创建 3d 游戏、基本建筑、3d 动画模型等各种内容并能同时发布多平台的专业游戏引擎。Unity3D 是一种可以使用图型化环境开发的游戏引擎。Unity3D 最开始是一群德国的游戏开发者因为爱好研发出来的，这群开发者主要是开发和研发在 MAC 平台上的游戏引擎，所以当时并不支持 Windows 平台。是开发者发现其存在巨大的前景，于是开始将 Unity3D 移植到其他的主流平台上，所以现在 Unity3D 不仅仅支持大多数的主流平台，也可以直接使用 Unity 自带的插件将游戏直接发布为可以同时支持 Mac 和 Windows 平台支持的网页游戏[17]。Unity3D 目前的开发比较简单易懂，支持 JavaScript 和 C#脚本，脚本语言非常大众，而且可以使用可视化的开发工具进行开发，并且随着 Unity3D 的发展，在引擎中可以支持模型的修改，所以越来越多的开发者选择 Unity3D 作为开发引擎进行游戏开发。

根据以上对三种主流游戏引擎的介绍，可以得出下面的结论：

1.Cocos2d-x 是一个比较轻量级的引擎，可以跨平台，但是以开发 2D 游戏为主。Cocos2d 最大的好处就是开源，使用 lua 脚本可以直接开发，语法简单，是开发小型 2D 游戏最好的引擎[18-19]。

2.Unity3D 主要是针对 3D 游戏的，最大的好处在于跨平台，有很多插件可以选择，并且支持 JavaScript 和 C#脚本，脚本语言非常大众，容易上手[20]。还有 Unity3D 的开发成本比较便宜，甚至不用付费都可以完成一次商业级开发，所以个人开发者和许多游戏公司更偏爱 Unity3D 作为开发工具。

3. UDK 的引擎十分强大，渲染效果是这三个引擎中最出色的，但是 UDK 是一

个比较巨大的游戏引擎，一般都用于开发大型的 3D 游戏，比如：鬼泣、死侍、失落的星球 3、生化奇兵：无限等大型游戏。但是过高的开发成本使得一般大型公司和大型团队会使用 UDK 开发 3A 级大作。

2.1.2 Unity3D 引擎的框架组成

Unity3D 引擎主要是由以下的基本框架构架而成：

1. 基本几何：由于 Unity3D 是一款 3D 的游戏引擎，所以开发 3D 游戏首先就要基本对立体空间内各个点的几何进行计算，通过涉及到的数学知识可以精确的完成游戏中玩家和物体的移动位置的计算和分析。还需要对物体的物理引擎进行模拟，比如物体的重力，摩擦力等。通过基本几何的计算就能完成以上这些计算，节省了开发者大量的时间和工作。

2. 资源导入：Unity3D 中的美术资源一般是由专业的创建模型的软件做出的，比如 maya，3dmax 等[21]。这些外部的建模工具一般将做好的模型使用 Unity3D 定义的 fbx 格式进行导入。使用 fbx 格式的优点是如果该模型带有动画、声音，那么它会将动画和声音一并导入带 Unity3D，从而不需要做多余的操作[22-23]。

3. 脚本解析：如果在 Unity3D 引擎中需要对游戏的逻辑进行处理，则需要使用脚本来控制。同时使用游戏脚本将脚本绑定在物体上可以控制物体的移动和物体的触发条件等等。脚本解析也就是通过执行绑定在物体上的脚本，从而使物体获得脚本定义的属性并执行脚本定义的操作[24]。

4. 网络通信：由于在制作网络游戏时不可避免的要使用到网络，所以 Unity3D 支持联网功能，支持数据通过基本的协议进行传输。大量的 Unity3D 引擎制作的网络游戏的出现，不仅证明了 Unity3D 的网络通信模块的成功也为以后的拓展提供了大量的参考。

5. 辅助工具：Unity3D 也拥有大量第三方的插件。Unity3D 的 asset store 为开发者提供了大量的第三方插件和辅助开发工具[25]。开发者根据自己的游戏需要从 asset store 找寻适合自己的插件可以极大的减少开发的复杂性，大大的缩短开发周期，节约成本。

6. 音乐音效：由于音效和音乐不仅仅是游戏中也是生活中重要的组成部分，所以 Unity3D 引擎为了游戏音效的处理提供了大量的接口。

7. 核心处理模块：Unity3D 为以下几种的核心模块进行了特别的处理，这样可以大幅的提高了游戏的趣味性。

(1)对象处理模块：在 Unity3D 游戏引擎规定了最基本的游戏单元是对象，开发者根据设计不同的对象才能组成开发者自己想要的游戏场景。

(2) 事件处理模块：Unity3D 通过事件处理模块来改变物体的属性，通过不同的脚本，Unity3D 就可以实时的处理物体的属性变化带来的后果。

(3) 摄像机模块：在 Unity3D 玩家可以看到的东西是通过摄像机采集到的部分。开发者可以设计多个摄像机，并通过不停的切换摄像机可以让玩家达到切换视角的目的。这样多视角的切换会让玩家有种身临其境的感觉。

(4) 渲染模块：一个游戏品质的好坏，除了游戏的玩法外很大程度上取决于游戏的画面。但是游戏的画面一般都是通过渲染模块进行复杂的运算出来的，这样会消耗巨大的资源。所以在开发游戏的时候，一般会将一些材质和光影隐藏起来，在需要的时候在通过渲染模块计算得出最终的画面。

2.1.4 Unity3D 引擎的开发结构

Unity3D 引擎主要是由应用层、游戏场景、游戏物体和游戏脚本这四个层次共同工作完成游戏的开发的[26]。

应用层：应用层是 Unity3D 将单独的应用程序整合到游戏引擎中的。每个单独的模块都负责 Unity3D 开发中的一部分，使用这些整合过后的模块可以节省大量开发者重新开发的时间。Unity3D 游戏引擎将天空盒模块、发布游戏模块、物理引擎模块、渲染模块等多个单独的模块封装之后提供给开发者使用。开发者也可以根据自己游戏的需求将应用层重新封装成适合自己游戏的模块。

游戏场景：游戏的发生地点就在场景之中，开发者通过设计不同的场景和场景的改变来让玩家进行游戏的交互。设计和制作一个精致的场景可以显著的提高游戏的品质，让玩家在与游戏的交互中获得更多的真实感。在游戏开发的时候，开发者一般会先设计好游戏的场景，然后可以将制作精良的场景复用，这样既可以节省开发者开发场景的时间也可以使游戏的品质得到保证。

游戏物体（Game Object）：游戏物体是 Unity3D 游戏引擎中最基本的组件，在 Unity3D 中所有的东西都是 Game Object，开发者通过对 Game Object 添加不同的

材质和模型来创建玩家角色和场景中的树木、河流等。开发者对游戏逻辑的控制也是通过将控制逻辑的脚本挂载在相应的 Game Object 上来让游戏可以正常的运行。所以开发者选择和设计一个合适的 Game Object 可以大大提升游戏的开发效率和提高游戏的流畅度。

游戏脚本：游戏脚本是控制游戏流程的重要组成部分。上面在介绍游戏物体时提到了游戏脚本组件需要挂载在游戏物体上。然后开发者通过编写游戏脚本来控制所挂载的游戏物体的属性。正是通过游戏脚本的控制，从而游戏物体的物理行为、状态的变化、按设计的进行运动等等效果才可以展示出来。

开发者通过合理的使用以上四个模块来完成一款游戏的开发，既能节省了开发者大量的时间和精力，同时也简化了游戏开发的流程。

2.1.4 Unity3D 引擎的特点

Unity3D 有如下几个主要特点[26]：

1.跨平台

由于 Unity3D 支持大多数的主流游戏平台，拥有统一的编辑器，所以只要使用 Unity3D 进行开发，便可以直接发布在你想要的平台上，不需要重复开放，大大减少了开发的复杂度。

2.高度优化的图形渲染管道

Unity3D 使用 DirectX 和 OpenGL 图形引擎，对 DirectX 和 OpenGL 拥有高度优化的图形渲染管道。并整合了高质量的粒子系统，易用高效的着色器、镜头特效等功能。

3.内置物理引擎

Unity3D 使用了 PhysX 的物理引擎，Physx 是目前使用最为广泛的物理引擎，被很多游戏大作所采用，开发者可以通过物理引擎高效、逼真地模拟刚体碰撞、车辆驾驶、布料、重力等物理效果，使游戏画面更加真实而生动。

4.兼容多种外部资源

Unity3D 可以支持多种文件格式的导入，可以支持绝大多数的主流建模软件。比如：3dsMax、maya 等主流建模软件导出的模型可以直接导入 Unity3D 中使用，而且使用 fbx 文件还可以将动画和声音一并导入。

5.支持多语言并且可以可视化操作

Unity3D 不仅仅使用 C#、javascript、boo 三种语言作为脚本语言，而且可以进行可视化操作，这三种基本脚本语言十分大众，比较容易上手，并且 Unity3D 自带的编辑器 Mono 不仅仅可以实现平时的开发工作，还可以直接在编辑器内对游戏的源码进行调试，从而节省了大量的时间。Unity3D 还支持玩家使用自己熟悉的开发工具进行开发，比如 Visual Studio 等集成开发环境进行开发[27]。

2.2 Kinect 相关技术

Kinect 一开始是在 09 年在美国 E3 展会上公布的，一开始 Kinect 是作为微软的 XBOX360 的一种外设。Kinect 的出现彻底使得人们对于这种体感交互方式的自然交互理念油然而生，从而对之前使用鼠标键盘等操作的游戏发起了强力的挑战。Kinect 不仅仅实一个简单的摄像机，同时它导入了即时动态捕捉、影像辨识、麦克风输入、语音辨识、社群互动等功能[28]。可以使玩家使用体感这种自然的交互方式进行游戏，并且可以与其他玩家进行交流等。

2.2.1 Kinect 的硬件组成

Kinect for Windows 主要由红外线发射器、RGB 摄像头、红外接收器、倾角控制马达、麦克风阵列以及加速计组成[29]。而整套环境则是包含了传感器、各种 Kinect 驱动（主要包括：麦克风阵列驱动，音频和视频驱动）、音频和视频组件（辅助 Kinect 捕捉骨骼移动以及深度图像）、DirectX Media Object （DMO）组件，以及另一个十分重要的辅助设备，搭载 Win7 或者以上操作系统的计算机。硬件结构如图 2.1 所示。

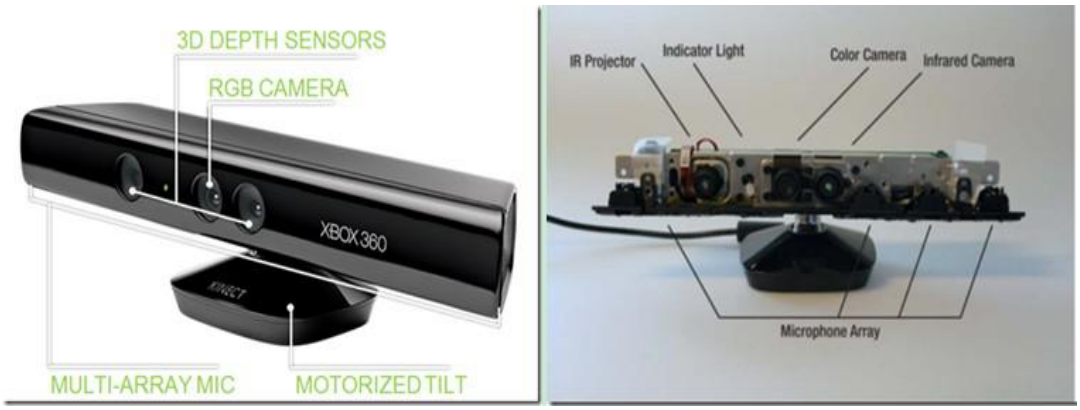

图 2.1 kinect 的硬件结构

Kinect 的主要硬件的作用如下[30]：

1.麦克风阵列：Kinect 的麦克风阵列不仅仅可以找到声源的位置，而且还能自动去除噪音。

2.红外投影机：由于发出的光源激光通过毛玻璃产生随机性很大的斑点图样，所以也叫做激光散斑，而这些图样在各个深度都是不一样的，通过观测打在物体表面的图样，就能计算出物体所在的位置。

3.红外摄像头：对之前的光谱进行计算，从而得出玩家和场景的深度数据。

4.仰角控制马达：可以让开发者使用程序控制的马达，经过控制后使 Kinect 有一个最好的视野。

5.USB 线缆：支持 USB 2.0 接口，主要是用来将 Kinect 采集的数据进行传输。由于 Kinect 的数据量比较大，所以要使用单独的电源。

6.彩色摄像头：主要是用来采集彩色图像数据。

2.2.2 Kinect for windows SDK 简介

微软在 2011 年对外发布了 Kinect for Windows SDK Beta[31]。这个版本的 SDK 使得当时众多开发者投入到 Kinect 的研究和开发中。但是这个测试版本的 SDK 当时只是给开发人员进行学习和测试的，开发者们使用这个 SDK 进行研究不能用于商业发布。一直到 2012 年，微软才发布了商业版本的 SDK 这个版本的 SDK 提供了许多新的功能和特性，并且到了这个版本才允许开发者将 Kinect 程序作为商业应用来发布和发卖。

由于 Kinect 是微软设计发布的，所以目前 Kinect for Windows SDK 需要 win7 以及以上版本的系统，使用 Visual Studio 10。需要使用 C++或者 C#作为开发语言进行开发。

Kinect for Windows SDK 完成了以下的功能[32]：

骨骼追踪：Kinect 将人体的骨骼绘制成 20 个关键的骨骼点，并且 Kinect 还可以进行人脸的识别。

深度摄像头：Kinect 使用一种叫做 Light Coding 的“光编码”技术，使用深度摄像头来采集数据[33]。采集到的深度数据是一种特殊的图像，这种图像每个像素点不是像素值，是 Kinect 通过计算后得到的的距离 Kinect 的相对距离。由于 Kinect 使用的使红外摄像头，所以即使在黑天没有光照的情况下也不会影响 Kinect 采集

Kinect 的深度数据。

音频处理：使用微软语音识别的 API，使用 4 个麦克风采集生源，同时过滤背景噪声，定位生源的位置。

2.3.3 Kinect 的技术架构

Kinect 为开发者封装好了十分高端和繁杂的类库、开发工具和各种新功能的 Demo 来帮助开发者学习和使用 Kinect。开发者可以使用这些 SDK 提供的方法，使用 Kinect 捕捉玩家的各种信息来进行开发工作[34]。Kinect 传感器通过该类库与应用程序的交互方式如图 2.2 所示。

图 2.2 应用程序与软硬件之间的交互

Kinect 设备首先会通过摄像头和麦克风等硬件设备采集到原始数据，然后 Kinect 使用自然用户界面技术（NUI）类库对硬件采集到的深度图像数据、彩色图像数据和音频数据进行计算，从而将玩家的具体信息包括玩家的骨骼数据、声音位置等数据提取出来。开发者则通过应用调用 NUI 类库中的接口来获取 Kinect 采集到的数据，然后将数据确定以后进行相应操作。NUI 是自然用户界面技术，是与普通的用户图形界面相对的，N 代表自然。NUI 的主旨思想是希望玩家摆脱普通交互方式的种种限制而使用自然的方法进行交互。简单的来说 NUI 是不使用鼠标键盘而使用人的肢体语言来进行交互。Kinect 的 NUI API 则是 Kinect SDK 中最重要的部分，NUI API 定义了如何使用摄像头和麦克风等硬件设备抓取玩家的具体信息，并且管理 Kinect 的硬件设备。NUI API 主要负责以下功能：

1.初始化 Kinect 设备并启用数据流传输；

2.通过彩色摄像头和红外摄像头来捕捉彩色图像数据和深度图像数据；

3.将被 Kinect SDK 处理之后的彩色和深度图像数据进行传输；

4.通过彩色图像和深度数据来进行骨骼追踪；

5.通过麦克风组件来获取音频数据。

2.3.4 Kinect 的工作原理

Kinect 从设计之初的目的就不仅仅是作为一个普通的传感器和摄像头，而是作为一个高端的体感输入设备。Kinect 主要是采集三种主要数据：深度数据流、彩色视频流、原始音频数据，分别对应骨骼跟踪、身份识别、语音识别等三种功能。

1.深度数据

深度数据是 Kinect 的精髓。Kinect 打开后会自动捕捉视野范围内的人形物体，由于 Kinect 使用的是红外摄像头进行捕捉数据，所以在没有光照的情况下也可以正常使用。Kinect 使用一种叫做 Light Coding 的“光编码”技术，使用深度摄像技术来采集数据。采集到的深度数据是一种特殊的图像，这种图像每个像素点不是像素值，而是 Kinect 通过计算后的距离 Kinect 的相对距离。深度数据最终是通过计算红外摄像头和红外投影仪的数据得出的。Kinect 的红外摄像机和其他的普通摄像机一样也是有视野限制的[35]。Kinect 摄像机的视野范围如图 2.3 所示。

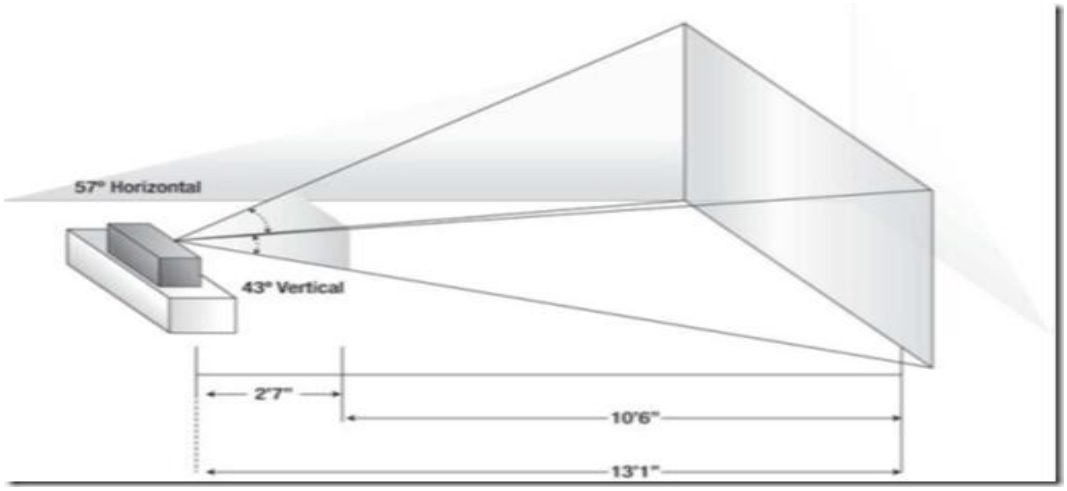

图 2.3 Kinect 的数据采集范围

Kinect 的摄像机的采集范围是 Kinect 前面的一个三角锥形。由于玩家离摄像机的距离不相同，距离 Kinect 远的玩家会采集到更大面积的范围。这就会导致采集出来的图像数据的高和宽与现实中的高和宽不完全相等。但是采集出来的数据的每个像素的深度值与真实玩家的距离是相匹配的。我们在存储 Kinect 的每帧的深度数据时，我们是使用 16 位字节来存储每个像素的[36]，Kinect 规定每 16 位中的 13 位用来

存放该像素点的深度值，而使用其中的三位来存放索引属性，深度数据的存储结构如图 2.4 所示。

图 2.4 Kinect 深度数据存储示意图

我们可以比较简单的得到一个像素点的距离，这个距离需要通过一些简单的位操作转换后才能直接使用。上文介绍可以知道，深度数值占了 16 位中的 13 位。如图 2.4 所示，深度数据存储在 3 到 15 位中，所以为了得到真正的深度数据要使用位操作，为了将玩家的索引值 Index 移除，我们需要向右移 3 位。

2.骨骼数据

微软开发 Kinect 的时候，使用的达芬奇的《维特鲁威人》来作为一个标准的人体模型[37]。Kinect 使用了 20 个关键的骨骼来组成一个人体的基本骨架结构的，具体如图 2.5 可以看到。在 Kinect 在它的视野范围内捕捉到人体的时候，Kinect 会自动把捕捉到的人体匹配成标准模型，然后用 20 个关键节点替代人体骨架形状的 20 个骨骼点，然后使用三维坐标（x，y，z）来标识出这些骨骼点的实际位置。经过这样转化之后，Kinect 才能根据骨骼点变化的位置来计算和判断出玩家的行为，然后通过行为数据来进行人机交互，实现体感操控。

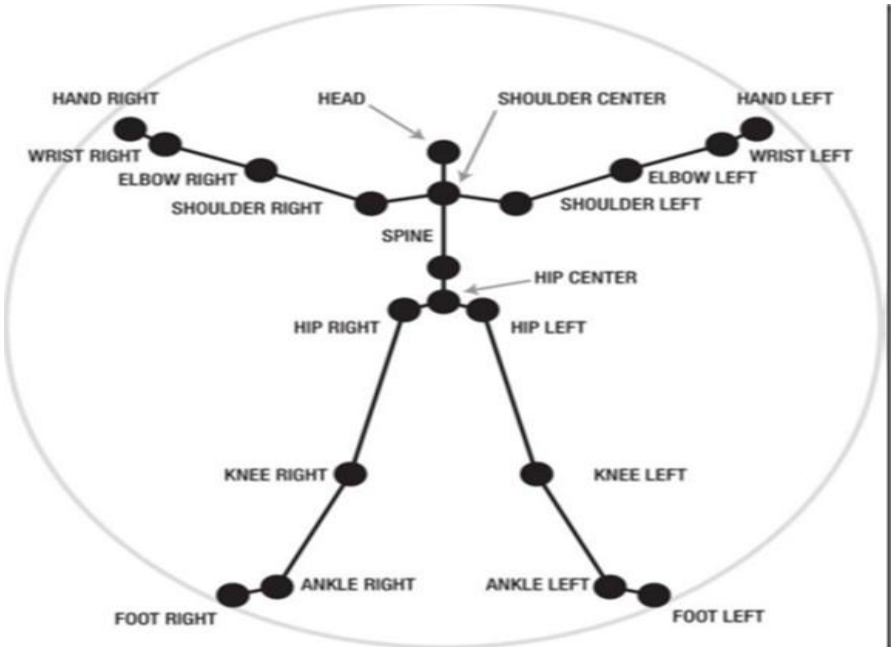

图 2.5 Kinect 默认的人类骨骼点

玩家表示骨骼数据的坐标点（x,y,z）的单位是米而与深度数据的空间坐标并不一样。但是坐标轴（x,y,z）与深度数据的坐标轴是一致的。Kinect 作为坐标系的中心原点，坐标系遵循右手螺旋法则，水平坐标 z 轴的方向是 Kinect 正对的方向，而 y 轴的方向则是垂直的指向 Kinect 的上方，最后 x 轴的方向向左延伸[38]，如图 2.6 所示。

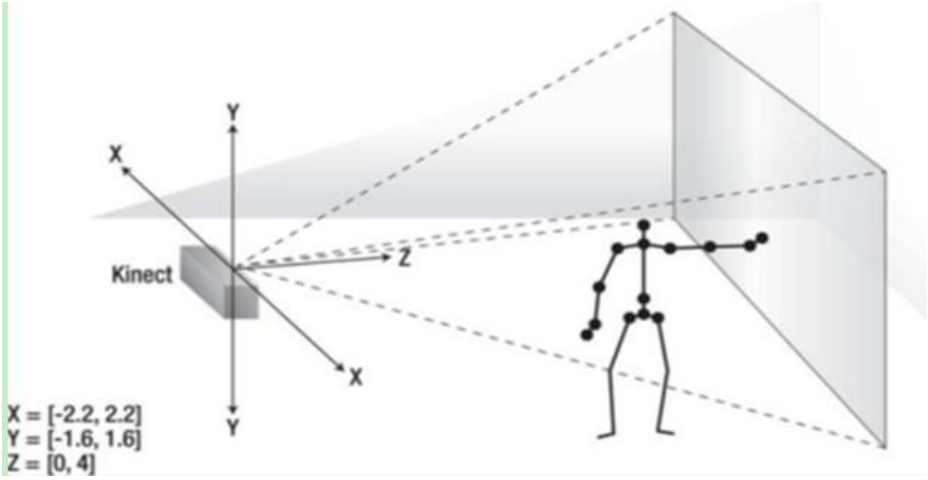

图 2.6 Kinect 的坐标系

Kinect 放置的位置会影响生成的图像。当 Kinect 放置在倾斜的平面上的时候，Kinect 在竖直方向会自动进行调整，从而获得最好的视野来进行捕捉玩家的数据。当 Kinect 不是放置在水平的桌面时，Kinect 的 y 轴也不是垂直于水平方向的所以在最后 Kinect 采集到的图像中，虽然采集到的不是笔直站立的玩家，但还是能够获取到正确的玩家信息。

Kinect可以追踪多个玩家的信息，Kinect最多可以追踪到6个人，但是却只能追踪到2个骨骼信息[38]，也就是只有两个玩家的骨骼可以被细化为20个关键点。这也就是Kinect的骨骼追踪的两种模式：主动模式和被动模式。主动模式就是Kinect主动会去分析计算Kinect的帧数据从而得到玩家的骨骼数据。而被动模式是指在最多还可以支持的6个人中的另外4个人无法获取完整的骨骼数据，仅仅有这四个玩家的位置信息。简单的说，就是对于在Kinect的视野范围内的6个玩家，Kinect能够将所有人的位置告诉你，但是只有处于主动模式的两个人才能获取全部的20个骨骼点信息，剩下的4个人是处于被动模式的玩家，Kinect只能告知他们的位置信息。

Kinect 捕捉玩家的时候有两个模式：站立模式和半身模式[39]。站立模式就是可

以正常捕捉玩家 20 个骨骼点的模式，而半身模式则是只能捕捉玩家的上半身的 10 个骨骼点。如图 2.7 所示。

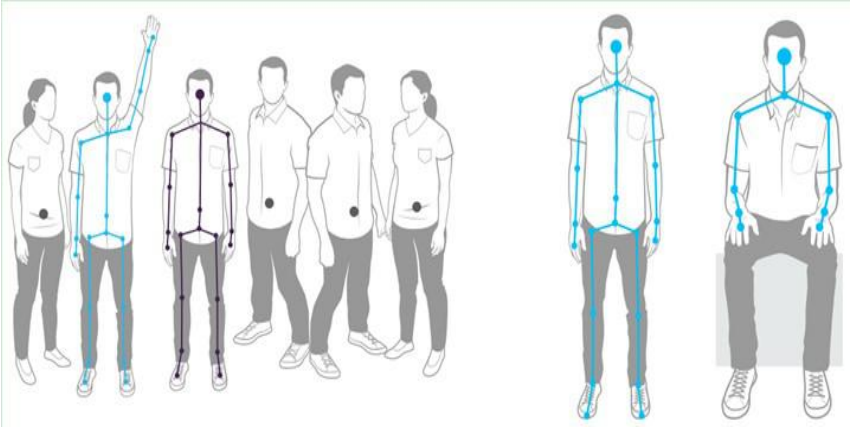

图 2.7 Kinect 能同时提供骨骼点

Kinect 所能获取的骨骼信息主要包含以下的信息：

1.相关骨骼的跟踪状态：既是上文所介绍的主动模式和被动模式，主动模式是全骨骼信息，被动模式是位置信息。

2.骨骼 ID：分配给玩家，用来区分骨骼是哪个玩家的[40]。

3.用户的位置。

2.3 本章小结

本章首先对系统实现中所涉及的各种技术进行了简单的介绍和分析，通过介绍目前主流的游戏引擎然后最终选择使用 Unity3D 作为游戏引擎，然后对 Unity3D 游戏引擎进行详细的介绍，之后再介绍和学习 Kinect 的框架、特点及相关的开发内容，为对这个体感游戏系统的知识储备做好了准备，并扫清完成系统时的技术难点。

第三章 游戏系统的设计

3.1 游戏设计的总体要求

本系统根据需求分成的模块如下图 3.1 所示。

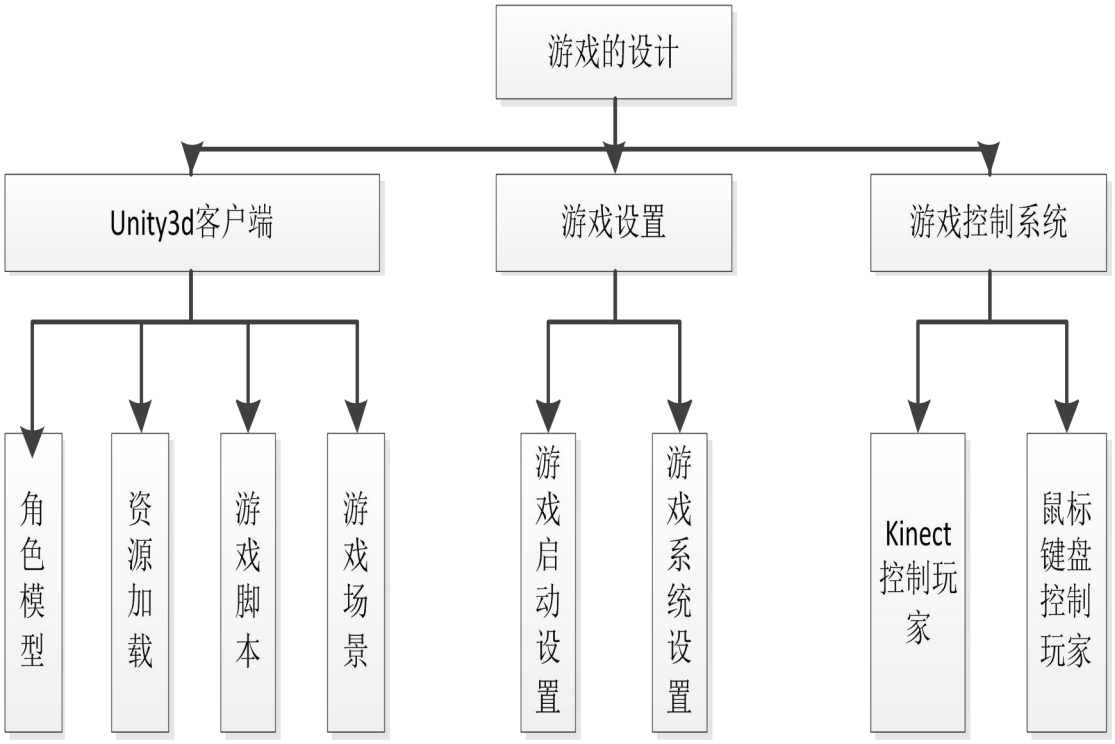

图 3.1 游戏系统模块

游戏场景：由 maya 和 3dmax 等 3d 创建模型工具设计和完成的用于游戏主要界面的场景[41]，导出后导入 Unity 中，本游戏系统采用的场景是一种比较卡通风格的场景。

游戏脚本：主要是用于游戏主程序的逻辑编程,AI 的处理[42]，通过执行绑定在物体上的脚本，从而使物体获得脚本定义的属性并执行脚本定义的操作。。

角色模型：Unity3D 中的角色资源一般是由专业的创建模型的软件做出的，然后得到后可以直接导入到 Unity3D 中。

资源加载：Unity 为开发者能够进行动态加载资源提供了封装好的类库。能够使贴图、音效、模型等资源更方便的动态加载。

游戏启动设置：游戏启动后初始化各个单独的模块进入游戏。

游戏系统设置：用作游戏设置，比如是否开启光影效果等。

Kinect 角色控制：使用 Kinect 实现对 Player 和 NPC 的控制。

普通的控制：在玩家累了的情况下可以使用鼠标键盘操作。

本文结合项目需求，介绍基于 Unity 的体感游戏系统作为游戏背景，由于本游戏是属于休闲类游戏所以使用 3D 卡通类设计风格，不仅可以使玩家的身体得到锻炼，还能使玩家的逻辑得到增强，如图 3.2 展示了游戏的功能框架图。

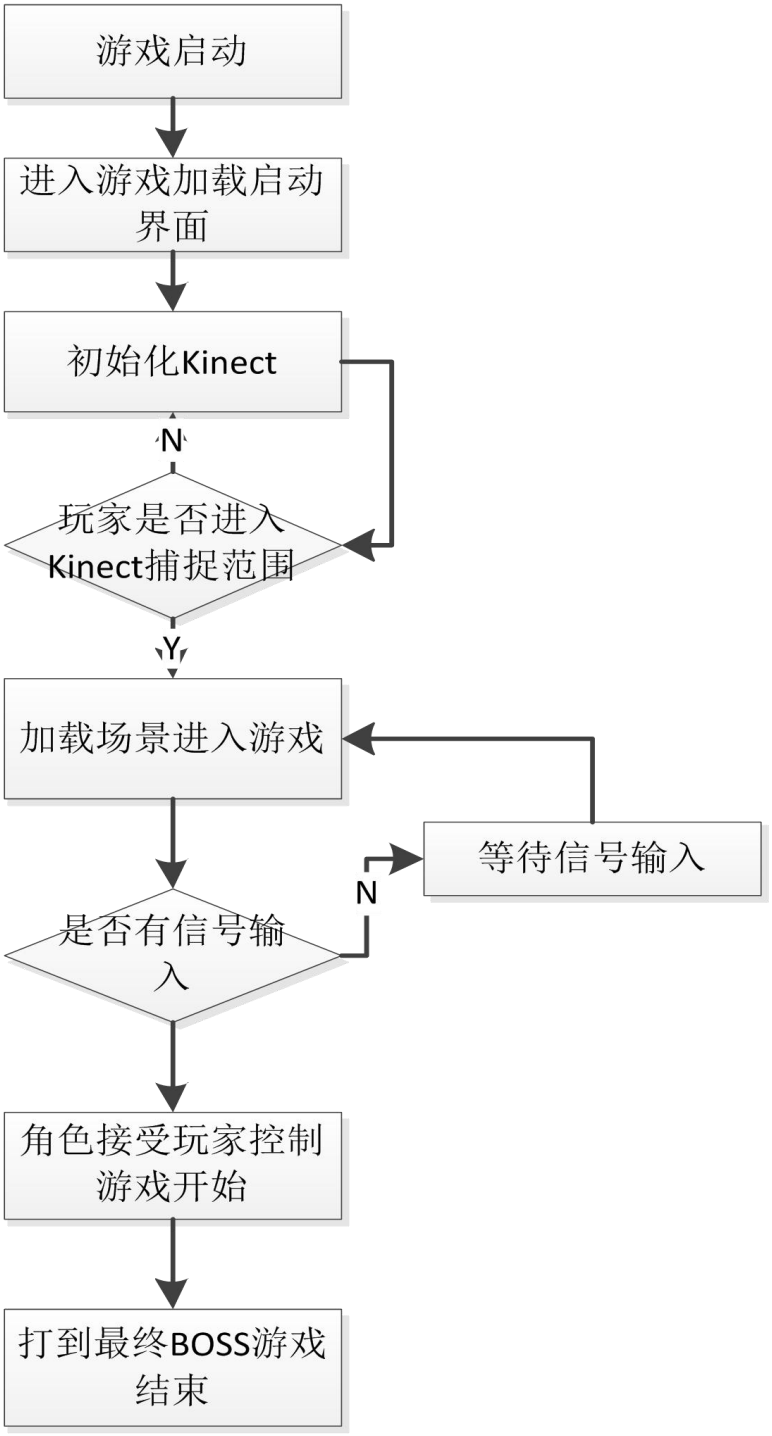

图 3.2 游戏功能框架图

3.2 游戏玩家的功能设计

结合体感游戏的设定，设计主角玩家和怪物的基本功能[43]，角色分别有站立、行走、跑步、攻击、收起武器、休息等状态动画。其中怪物的动画不由玩家操控，而是使用脚本设计的 AI 来控制，而玩家的状态则是由玩家自己操控的。玩家在没有操作的情况下保持站立的姿态，可以随时切换是否使用 Kinect 或者普通的操作方式，使儿童和玩家在劳累的时候也可以进行休息，从而达到劳逸结合的效果。图 3.3 描述了角色的功能。

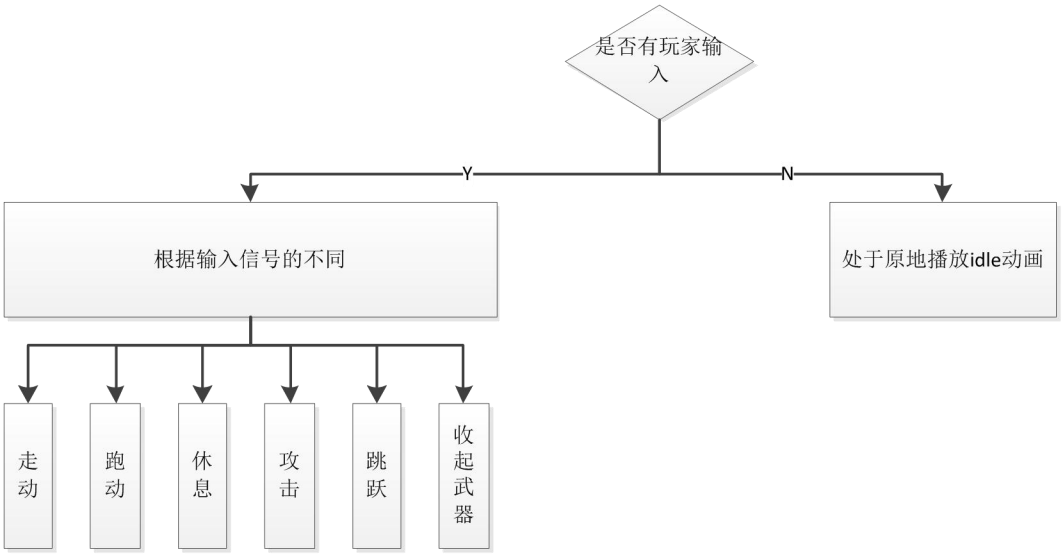

图 3.2 游戏玩家功能描述

3.3 有限状态机的设计

之前游戏的的状态切换一般都是使用判断语句 if...else...来控制，但是这样做却容易带来状态切换的代码繁琐和不易于维护等坏处。所以本系统不使用 if...else...来切换状态而是打算使用有限状态机来完成怪物的状态切换。有限状态机 FSM 是一个数学模型也是一种算法思想[44]，表示为有限个的状态和在初始状态后通过不同的输入数据后经过转换函数在有限个不同的状态间切换的数学模型[45-46]。

本篇论文首先是根据需要将玩家和怪物的状态设计出来，之后还要将根据输入之后需要改变的状态设计好也设计好，设计出来的状态和切换后的状态如表 3.2 所示，玩家和怪物的状态基本类似，都由一些基本状态组成，但是怪物切换状态时的触发条件与玩家的不同。使用有限状态机后，我们只需要将 AI 根据不同的输入切换

条件来使输出状态进行切换即可。使用有限状态机后，角色在切换状态是能体现出更好的灵活性,不仅仅帮助解决了如果切换条件比较复杂时状态会十分混乱的问题，而且还能有效的避免掉游戏程序的冗余。实践证实有限状态机和 if...else...比较起来具有十分巨大的优势，也更能简化开发者的工作。

当前状态	输入切换条件	输出状态
	玩家	
idle	玩家操控行走	Walk
Walk	玩家控制攻击	Attack
Walk	玩家控制跳跃	Jump
Walk	玩家控制休息	Rest
Rest	玩家操控行走	Walk
Rest	玩家控制攻击	Attack
	怪物AI	
Patorl	玩家进入怪物10米之内	Run
Run	与玩家距离1米	Attack
Attack	系统发出玩家死亡消息	Patorl
Attack	系统发出怪物死亡消息	Dead
Attack	怪物血量只有10%	Escape
Escape	与玩家距离10米之外	Patorl

表 3.3 玩家和怪物的状态分析

3.4 自动寻路的 A*算法

寻路算法的目的是为了寻找在地图中的起始节点和终点的一条最优的路径。这里最优路径指的并不一定是两点之间的最短的距离也有可能是消耗的最少时间或者代价最小的一条路径[47]。对于现在的游戏来说越来越多的开发者选择 A*寻路算法作为 AI 和玩家自动寻路时候的主要算法[48]。

3.4.1 A*寻路算法

现在的游戏在开发寻路模块时，为了获取两点之间的最短路径和最优解时大多使用 A*寻路算法。A*寻路算法是一种启发式的寻路算法，比起其他的自动寻路算法具有启发操作的 A*算法在性能和时间复杂度上具有较大的优势[49]。

目前的寻路算法基本都是基于深度优先（BFS）或者基于广度优先（DFS）这两种的。基于深度优先（BFS）的算法思想主要是按顺序的根据现在的路径的分支

进行寻找，优先寻找分支的路径点，直到该路径的所有分支点都没有目标点，才会切换到下个路径点重新寻找，直到找到目标点为止。基于广度优先（DFS）的算法思想主要是根据层次来寻找目标点，优先寻找该路径点同级的路径点，如果没有的情况下再寻找分支点直到找到目标点为止。根据以上两种寻路算法的流程我们能够得知，虽然这两种算法最终都能找到两点之间的最短路径的最优解，但是这两种算法具有大量的迭代次数，时间复杂度很高，并且会造成内存资源的巨大消耗。所以以上两种算法不适合很大的空间，因为如果空间很大则有很多的路径和分支，使得分支点变得不可预测，这样就会消耗很多的时间和内存资源。

A*算法是为了解决基于深度优先（BFS）和基于广度优先（DFS）算法中的内存资源消耗大和时间复杂度高这些问题而提出的最短路径的搜索算法。在启发式 A*寻路算法的搜索过程中，A*算法的启发函数至关重要，启发式 A*寻路算法的好坏很大程度上取决于启发函数的选取，合适的启发函数能获得更大的启发信息，所以能够有更好的路径选择和更好的时间复杂度。但是如果启发信息过多的话也会影响时间复杂度，所以如何选择一个适中的启发函数使得搜索准确性和时间复杂度都适合是启发函数的一个重要选择条件。

3.4.2 A*寻路算法的算法流程

主循环：

首先将起始结点放入开启列表，关闭列表置空，算法开始时：

1.如果开启列表不为空，从表头取一个结点，如果为空，那么算法失败。

2.判断该节点是否为目标解。如果不是则继续，直到找到为止。

3.将该节点的所有后继结点展开，就是从该节点可以直接关联的结点（子结点），如果不在关闭列表中，就将它们放入开启列表，并把当前节点放入关闭列表，同时计算每一个后继结点的估价值 f，将开启列表排序，最小的放在表头，重复算法，回到 1[50]。

A*算法具体流程如图 3.4 所示。

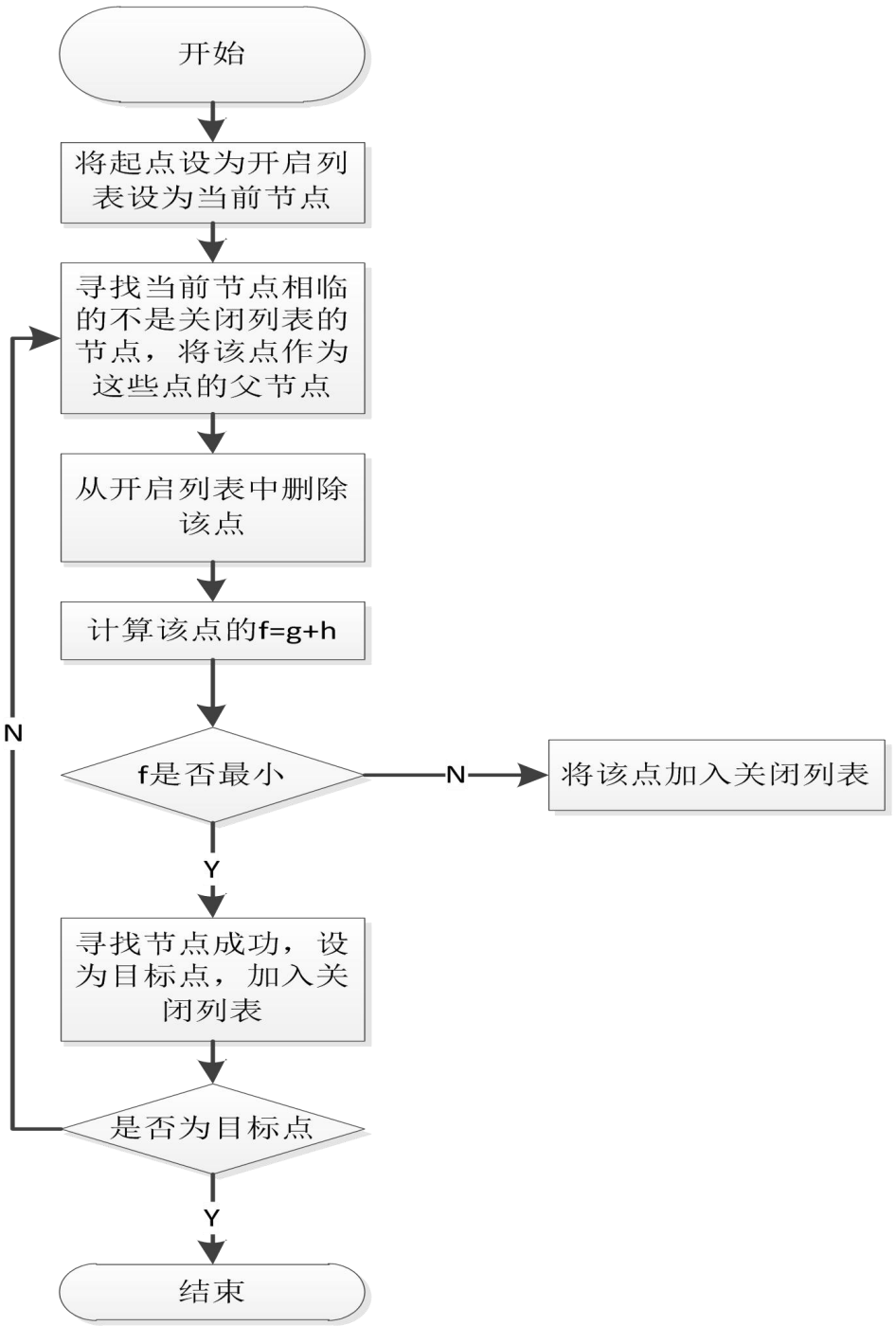

图 3.4 A*算法的流程图

3.4.3 启发函数

启发函数是一个用于计算现在的节点和下个节点之间的距离的函数，是 A*算法中非常重要的函数。目前 A*算法主要有以下三种启发函数，三种启发函数虽然都是计算距离的，但对 A*算法会产生不一样的影响。

1.曼哈顿启发函数。如图 3.5(a)，该启发函数是最基本的一种启发函数，曼哈顿

启发函数就是取出当前节点和目标节点的行数和列数的差，也就是意味着不会直接走对角线。例如：a(1,1)b(2,3)直线的代价为 1，则最后得到的结果是 3。

2.三角启发函数。如图 3.5(b)，三角启发函数计算后得到的值是两点之间最短的距离，也就是意味着取得是两点间的直线距离。

3.对角启发函数。如图 3.5(c)，对角启发函数则是一种比较准确的函数，它是先走对角线，再走直线。首先根据当前节点和目标节点的行数和列数差，先走对角线，之后根据两点间位置再走一条直线[51]。

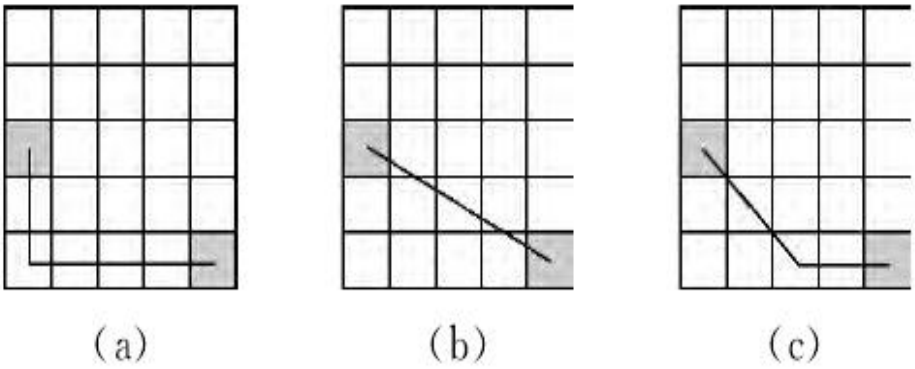

图 3.5 3 种不同的启发函数的效果

3.4.4 A*寻路算法的优化和性能分析

如何管理 A*寻路算法中的开启列表是优化 A*寻路算法的一个重要方面[52]。首先 A*寻路算法对开启列表进行排序操作，然后才会对该节点进行运算，最后将周围节点加入开启列表中。由于每次计算都要添加新的节点进入开启列表进行排序，所以这样就会导致在寻路的过程中单次排序的时间和多次进行排序都会消耗大量的时间，在计算最短路径时会造成玩家的停滞感从而影响游戏体验感。并且如果地图过于巨大，则会消耗巨大的时间。

针对于这个问题，我们可以从不同的地方进行优化处理：

1.排序算法

我们可以使用数据结构中经典的排序算法来进行排序，比如冒泡排序，归并排序，选择排序等等。而在动作脚本（Action Script）中为数组的排序定义了自己的排序方法 sort。AS3 中的 sort 方法是根据数组中元素的 Unicode 值进行排序的，并且还给我们提供了一个用于自己排序的参数。经过多次实验之后发现，使用 AS3 中的

sort 方法能最快速的得出排序结果。

2.启发函数的选择

除了排序算法的优化外，如前文所说的，选择合适的启发函数能够大大优化 A*寻路算法。最基本的启发函数有上文所说的三种，他们之间最主要的区别在于他们制造出的开启列表的长度不相同。与其他的两种启发函数比较起来虽然对角启发函数在每个阶段进行计算时会消耗更多的内存，但是对角启发函数最大的优点是会产生比另外两种启发函数小得多的开启列表。尤其是寻路的地图比较大时，开启列表会成倍的增加，使用对角启发函数会显著减少开启列表长度，从而能节省更多的时间。所以本系统拟采用对角启发函数。

在相同的游戏地图上，使用相同的起点和终点，使用三种不同的启发函数进行 A*寻路算法，然后对三种启发函数进行分析。采用 7.8 和 3.9 作为每步计算的间隔时间，计算出来的时间耗时如图 3.6 所示。

ms

计算的时间间隔	A*（曼哈顿）	A*（三角）	A*（对角）
7.813	502.343	658.594	978.906
3.906	251.172	329.297	489.453

图 3.6 不同的启发函数的耗时

3.多层面的 A*算法

可以将地图划分成几个层面。同样的地图既可以划分为 10*10 的网格地图也可以作为 1000*1000 的网格地图。划分成不同的网格在寻路事会出现不同的结果，划分的比较粗略的网格比如 10*10 时，寻路速度快，但是不精准，划分的很细致的网格地图寻路时很精准，但是时间消耗大。为了既精准又快速的找寻路径，我们使用多层面的方法。首先在较粗略的网格地图中找到一条路径，然后将路径点导入到精细的网格地图上，然后再执行寻路算法，这样就不会产生巨大的开启列表。

3.5 本章小结

本章是主要是体感游戏系统的基本设计，包括玩家和怪物的状态变化，基本功

能和整个系统的流程设计等，并详细介绍和分析了 A*寻路算法，对该算法进行一些优化，为体感游戏的完成做好准备。

第四章 Kinect 体感技术的的研究和接入系统

在本体感游戏系统内中，有一个需要重点实现的模块。Kinect 的输入模块，这一章将对Kinect 如何采集数据和与Unity3D 结合起来进行游戏的控制输入的模块进行设计和实现。

4.1 Kinect 设备的工作原理

如第二章介绍的那样，Kinect 内置体感侦测装置 Prime Sensor 和感测晶片使用 3D 侦测技术。如图 4.1 所示，Kinect 的芯片发出激光穿过扩散片，通过毛玻璃产生随机性很大的斑点图样，所以也叫做激光散斑，而这些图样在各个深度都是不一样的，通过观测打在物体表面的图样，就能计算出物体所在的位置。

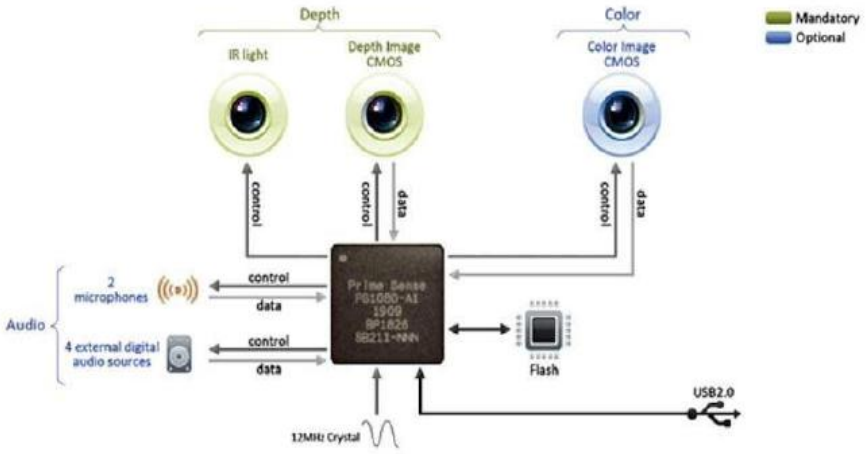

图 4.1 prime sensor 工作原理

Kinect 摄影机的硬件更新速度为 30 帧/秒，所以采集时会有一丝的延迟，没有使用更高速率的摄影机主要是因为计算模块的限制。比较于这三十分之一秒的延迟，开发者应当把减少延迟的任务放在如何能更快速的从原始数据中提取出玩家信息这一算法上。因为 Kinect 采集到原始数据后还需要经过大量的运算之后才能采集出开发者需要的信息，以及开发者在设计游戏时的优化也更能显著的减少时间。所以本文在设计系统时需要对游戏进行优化，尽量减少数据，并对 Kinect 的数据进行预处理来达到优化的目的。

4.2 Kinect 环境配置

Kinect 的系统配置：

32-bit (x86) or 64-bit (x64) processors

Dual-core, 2.66-GHz or faster processor

USB 2.0 bus dedicated to the Kinect

2 GB of RAM

Graphics card that supports DirectX 9.0c

A Microsoft Kinect for Windows Sensor

所需软件：

Windows 7/Windows 8

Visual Studio 2010/2012

.NET Framework 4.0

Kinect SDK for Windows

安装 Kinect SDK 步骤：

第一步：移除 Kinect 设备，关闭 Visual Studio。

第二步:在微软官网上下载并安装 Kinect for windows SDK 和 KinectDeveloper Toolkit 文件。其中，Kinect for windows SDK 提供 Kinect 设备驱动和设备访问，KinectDwveloper Toolkit 用于程序开发。

第三步：确定 Internet 联机正常并接上 Kinect 外接电源和计算机 USB 接口，等待下载必要的驱动程序后出现如图 4.2 的情况后代表安装成功，kinect 设备可以使用。

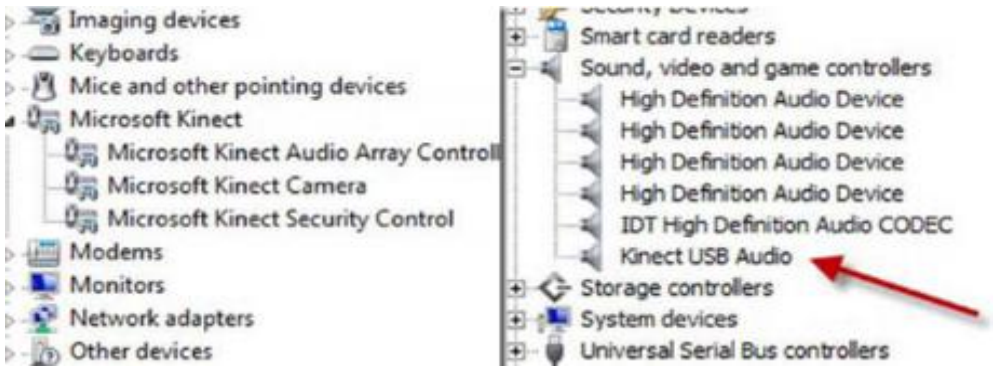

图 4.2 Kinect 安装成功的显示

4.3 Kinect 采集骨骼数据

骨骼追踪技术是 Kinect 的核心技术。骨骼追踪首先是需要将玩家的模型从深度图像数据中提取出来同时去除噪声。Kinect 首先会检查 Kinect 附近区域的像素，然后按照每个像素点去处理这些像素点,将玩家的信息从这个深度数据中扫描出来。使用包括边缘检测、噪声阈值处理、对人体目标特征点的分类等计算机图形学技术按照特征值的不同将玩家从背景中区分开来。Kinect 不仅仅需要将玩家从深度图像中提取出来，还需要提取出玩家每个关键骨骼点的信息才能生成接下来要进一步识别人体的关节点，系统根据“骨骼跟踪”的 20 个关节点来生成一个骨骼点的骨架结构然后分别对玩家所有的骨骼点进行实时的追踪，提供这些点的完整信息。Kinect for Windows SDK 为骨骼点提供了一种全新的数据类型：Joint 类型。此类型包含三个属性：

1.JointType：Kinect 定义的骨骼数据类型。它枚举了全部的 20 个骨骼点的特定名称。“HAND_LEFT”，“HEAD”等等。

2.Position：SkeletonPoint 类型表示骨骼点的位置信息。SkeletonPoint 存储了骨骼点的坐标（x，y，z）信息。

3.TrackingState：骨骼的追踪状态是一种枚举类型，有三种状态。分别代表已追踪到、未追踪和不确定三个状态，用 Tracked，NoTracked 和 Inferred 表示。

骨骼数据来自 SkeletonStream。访问骨骼数据有事件和 “拉”两种方式。在本系统中我采用基于事件的方式，因为这种方式简单，代码量少，并且是一种很普通基本的方法。获取骨骼数据由以下四个步骤：

1.搜寻一台可用的 Kinect Sensor 传感器。

2.启用骨骼数据流。

3.使用事件模式采集骨骼数据。

4.开始捕获数据。

4.4 Kinect 数据处理方法

Kinect 传感器通过上述方法采集到玩家的行为数据后，通过 Kinect 的 20 个关键点的骨骼的三位坐标信息进行识别玩家的行为类型。

首先定义参考行为类型模版，包括双手自然下落、右手向前伸出、双手向前伸

出、向左平伸左手、向右平伸右手、下蹲、向前弯腰、双手向上举起这八个主要行为。 如表 4.3 所示， 给这些行为定义识别命令，以便系统能识别出这些行为。

行为	代表命令
双手自然下垂	收起武器原地站立
双手向前伸出	攻击
右手向前伸出	拿出武器
向左平伸左手	向左转
向右平伸右手	向右转
下蹲	休息
向前弯腰	前进
双手向上举起	跳跃

表 4.3 行为对照表

4.5 Kinect 连入 Unity3D 客户端的实现

为了实现以上行为数据的采集，并提供给游戏的客户端来调用，所以在此我定义了两个主要的类文件来完成这个目标[53-55]。其中自定义的 KinectWrapper 类主要用于通过 Kinect 来获取骨骼数据；UserManager 类的主要目的是将获取到的骨骼数据进行处理，变成事先预定义的 8 种主要行为。

4.5.1 KinectWrapper 类的设计

KinectWrapper 类是主要实现了 Kinect 采集玩家的行为数据，所以需要使用 Kinect 自带 SDK 中的方法。KinectSDK 中的 Kinect10.dll 文件定义好了常规开发的应用程序接口（API），我们需要重新封装的方法主要有初始化 Kinect 和获取 Kinect 的三大数据的方法。除此之外，类似于调整 Kinect 马达的仰角和启用音频组件的方法也一起进行封装，从而可以用于将来的拓展。我们在使用这个类之前需要定义一些 Kinect 定义过的枚举类型，如上文介绍的 TrackingState、JointType、错误信息等。这样在 Kinect 传入相应的参数时，可以直接使用，不用再重复定义从而简化了主程序的代码。通过封装以上的接口后，我们就可以获取到 Kinect 的数据了。

4.5.2 UserManager 类的设计

我们虽然使用 KinectWrapper 类获取到了 Kinect 的数据，但是我们为了获取玩家的行为种类还需要设计 UserManager 类。该类需要将定义了使用 Kinect SDK 进行

手势交互开发的 API 的 Kinect Interaction170_32.dll 进行封装。该 dll 接口提供了类似于交互手信息、交互帧信息、用户信息等数据类型。获取到的 Kinect 数据是安装帧来获取的，而交互帧数据中是已经包含了用户信息的数据，而用户信息中又提供关于交互手的信息。该类定义了一些对于采集玩家数据所定义的一些重要的数据类型：

1. 手部信息类：该类主要定义了关于玩家的一个手的信息，包括这只手是否是玩家现在与系统交互的手，是左手还是右手以及现在手是按压还是握拳等状态，还有手臂的长度等；

2.交互帧类：交互帧类主要是包含的一个指针指向用户信息类；

3.目标控件类：这个类主要是存储空间的相关信息；

4. 交互流类：这个类主要的目的是存储交互帧信息并且管理这个类用于交互流；

5.用户信息类：这个类主要包含标识是哪个骨骼的用户唯一 ID 和一个指向手部信息的指针。

我们之后还要对获取到的这些数据进行预处理来满足优化游戏的目地。所以我定义了 UserManager 类来将获取到的数据进行处理。在 UserManager 类中我先定义了双手自然下落、右手向前伸出、双手向前伸出、向左平伸左手、向右平伸右手、下蹲、向前弯腰、双手向上举起这八个主要行为的枚举类型。通过计算骨骼数据的坐标点信息来判断捕捉到的玩家的骨骼数据进而可以分析出此时玩家的行为，然后将玩家的行行为信息提供给 Unity3D 客户端作为输入数据。

4.5.3 Unity3D 调用 Kinect 数据的实现

为了将 Kinect 与 Unity3D 客户端进行整合，我在之前封装了两个类来获取 Kinect 采集到的数据并且提供给 Unity3D 客户端使用。首先我们先封装 KinectWrapper 类，我们要使用 Kinect10.dll 中提供的初始化 Kinect 设备和获取数据的方法。在 KinectWrapper 类中我们使用 DllImport 将 Kinect10.dll 进行封装，经过导入后，我们就能将 Kinect10.dll 的方法定义为公有的静态方法，这样我们就可以进行调用了[56]。具体代码如下所示：

```
//使用动态链接库文件
[DllImport Attribute(@"Kinect10.dll", Entry Point = "Nui Initialize")]
```

//将 Kinect10.dll 的方法定义为公有的静态方法

public static extern int Nui Initialize(Nui Initialize Flags dw Flags);

同时我们还设计了 UserManger 类用来将获取的玩家数据转换成人类行为的特定数据。这样我们就可以直接使用 Unity3D 获取计算后的行为数据，减少了重复计算。我们首先在 Visual Studio 2010 中，建立一个类库的项目。然后将所需要的逻辑代码编写完成后就可以生成对应的 dll 文件。

我们编写封装之后的类库需要以下几个步骤：

1.首先我们定义一个类去实现交互流接口，实现这个类的目的只是为了将 IInteraction Client 接口中的 GetInteractionInfoAtLocatio 方法进行重写，使这个方法返回一个交互信息数据，这个信息主要是进行初始化；

2.将 Kinect 实例化，创建一个 Kinect 对象_kinect，使用 KinectSDK 中的方法对 Kinect 设备进行初始化；

3.我们的系统主要使用骨骼数据和深度数据，所以我们需要初始化 Kinect 设备的任务就是启用骨骼和深度数据流。由于在交互流的处理过程中需要深度流和骨骼流的数据，所以需要对深度数据和骨骼数据注册事件处理函数，之后在处理函数中开始添加深度数据和骨骼数据的处理方法；

4.由于 Kinect 采集的数据都是按帧传输的，所以我们处理的是每帧采集到的图像，我们要先定义好两个数组 DepthPixels 和 SkeletonPixels 它们分被存放深度数据和骨骼数据。然后将采集到的深度帧数据和骨骼帧数据自带的 CopyDepthImagePixelDataTo 和 CopySkeletonDataTo 函数把数据存储到数组中去，这样我们所需要的数据就存放在数组中了。当我们想获取我们的数据时，我们只需要使用一个循环，循环处理骨骼数组取出每个骨骼点。接着我们找到我们想要的骨骼点，将骨骼点的信息进行计算即可得出玩家的行为数据。我们首先可以使用 Is Tracked 属性可来判断我们的玩家是否被追踪，然后通过计算相关关节点的三位坐标来得出玩家的行为。

5.将判断行为的方法设为静态方法，封装到 dll 中间提供给 Unity3D 客户端代码来调用。

将以上的步骤完成后，对项目编译就会生成 dll 文件。我们也是通过 Unity3D 中使用 DllImport 命令来导入 dll 文件，然后对 dll 文件重新定义为公用的静态方法，在 Unity3D 中直接使用。

4.6 本章小结

本章主要完成了与 Kinect 相关的体感输入模块的完成，包括如何安装 Kinect SDK 与设计和完成 Kinect 与游戏系统整合的模块。并将这些模块封装成 dll 类库在游戏系统中直接使用。

第五章 Unity3D 游戏客户端的实现

本章主要是实现 Unity3D 客户端。为了简化游戏的开发，将整个系统分为以下几个主要模块，本章主要对如何设计和实现这些模块进行了具体的描述和说明。

5.1 游戏系统图形界面的实现

游戏的系统图形界面是游戏开发的一个难点也是游戏所必需的模块。一般游戏都使用事件驱动的思想去实现界面。事件驱动的思想如图 5.1 所示。

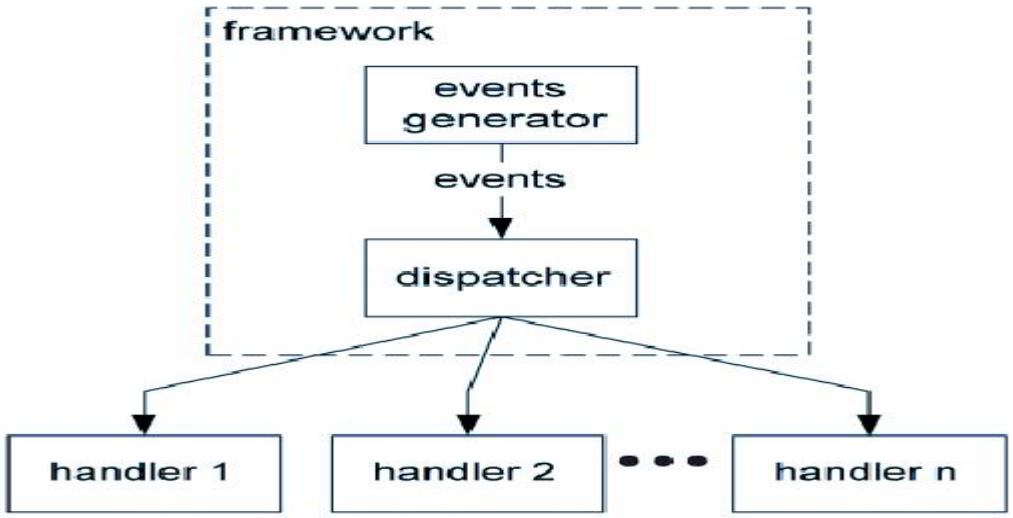

图 5.1 事件驱动模型

事件驱动的主要思想如下：首先需要事件产生器，事件产生器的作用就是生成各种事件，比如在玩家使用滑块时都会产生相应的事件。之后会把产生的事件通知到事件发布器，然后会判断事件的类型发送给不同的事件接收器。事件接收器将事件接受后进行处理相关事件。

5.2 游戏玩家和怪物的资源加载实现

Unity3D 中的美术资源一般是由专业的创建模型的软件做出的，比如 maya，3dmax 等。这些外部的建模工具一般将做好的模型使用 Unity3D 定义的 fbx 格式进行导入。使用 fbx 格式的优点是如果该模型带有动画、声音会一并导入 Unity3D 而不需要做多余的操作。

Unity3D 中最为基本的单位是 GameObject。任何的其他模型都会被声明为

GameObject。在 Unity3D 中一般会把经常使用到的资源作成 prefeb，prefeb 可以理解为是一个游戏对象及其组件的集合，目的是使游戏对象及资源能够被重复使用。相同的对象可以通过一个预设体来创建，此过程可理解为实例化。具有如下优势：

1.只要制作一次 prefeb，prefeb 中包含的特效、声音、动画等资源都可以简单的复用到各个其他场景。

2.场景中所有 prefeb 都是对 Project 列表 prefeb 的克隆。

3.如果想修改 prefeb 模型时，只需要修改一次其他的 prefeb 都会根据修改后的要求而改变，这样可以节省大量的工作。如图 5.2 所示，本系统将怪物做成 prefeb 之后大量克隆后可以直接使用。

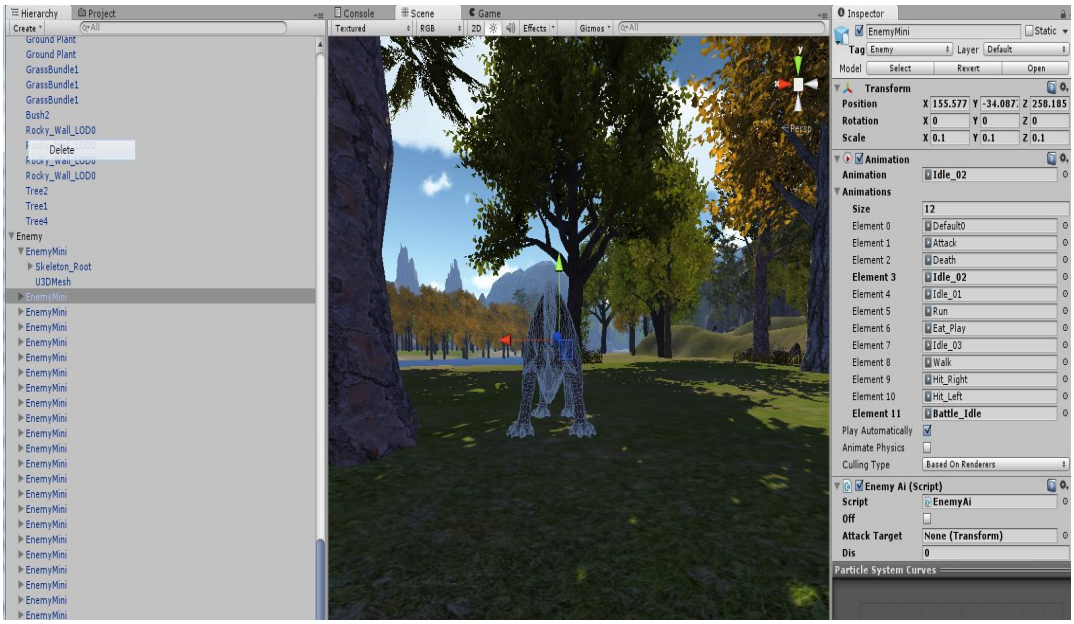

图 5.2 怪物资源的加载

本系统使用了大量资源，为了节省损耗，故采用动态加载的方式进行资源的加载。使用动态资源加载首先是将各种资源包括模型、声音、动画等先存储到本地的硬盘中，在进入游戏后进行加载。Unity3D 为动态加载主要设计了 3 种方法来实现：

1.Resources.Load()

建立一个 Resources 文件夹，然后将所有需要加载的资源都复制到该文件夹下，之后使用 load 方法来加载资源，那么 Unity3D 就会自动将所需要引用的资源加载进游戏的包里。

2.Assetbunde 形式

Assetbunde 是将所有需要打包的资源放在网络服务器中，然后通过网络请求来下载打包好的资源（一般是使用 WWW 请求）。最后将下载后的 Assetbunde 加载到

游戏中即可。

3.AssetDatebase.loadasset()

最后这种方法只能用于 Unity3D 的编辑界面，所以是给调试人员使用的。

对比以上三种方法，第一种方法比较适合单个资源的加载，而最后的一个方法只是用于 Unity3D 的编辑界面，所以是给调试人员使用的。所以本系统采用的是第二章加载资源的方法进行动态加载。

Assetbunde 方法是将所有的游戏资源和对象使用二进制格式的文件封装到 Assetbunde 中，然后将 Assetbunde 放置于网络服务器中，通过 WWW 协议下载到本机中直接 Instantiate 后游戏物体就能实例化到游戏中了。Assetbunde 首先通过网络下载到 bundle，然后使用加载方法将资源从内存中加载到游戏的场景里，经常用于实时绘制创建游戏地图和玩家装备的切换等。Assetbunde 资源打包和加载流程如图 5.3 所示。

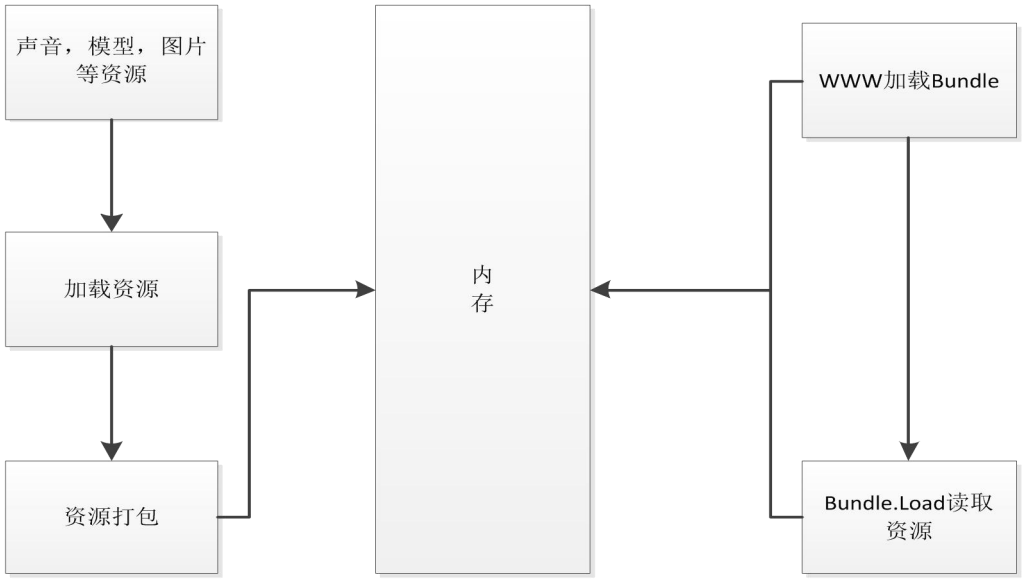

图 5.3 Assetbunde 资源打包和加载流程

5.3 游戏静态场景的实现

由于本体感游戏面向的对象主要是小孩子并将故事背景设计在山中，所以在游戏界面和场景的风格设计上偏向清秀，建立真实的山谷场景，山清水秀，让玩家完全融入到游戏的故事情景中去[57]。

5.3.1 游戏场景的地形创建

为了配合游戏的故事背景，我们创建一个山清水秀的小山谷的地形。而地形中

的水面资源设置为可碰撞不可穿越的刚体。并放置大量山峰来设置地行的边界，防止玩家走出地形。地形效果如图 5.4 所示。

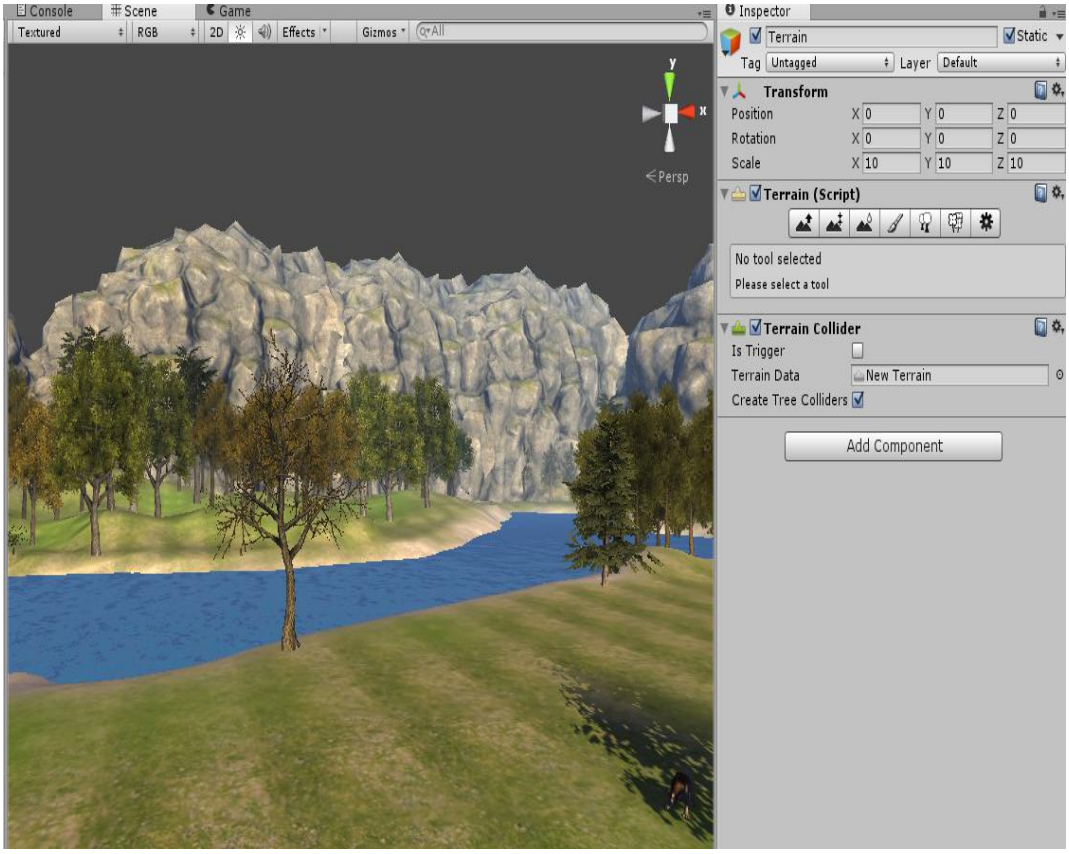

图 5.4 地形效果图

5.3.2 游戏天空盒的设置

其实 Unity3D 实现天空盒的本质就是使用一个特别的 Shader 材质一直在所有场景的外面，而且通过这种材质来模拟出天空的效果。Unity 添加天空盒有两种方式：.在当前相机上添加 skybox 或者在当前场景上添加 skybox。上面的两种方式的结果是一样的，第一种方式是在现实世界中有多个摄像机的情况下，切换摄像机需要看不同的天空，从而使用这种方式比较便利。第二种方法是针对游戏场景的，简单地讲，就是在同一个游戏场景中，无论使用哪个摄像机对象，天空盒都保持不变，并且该方式指定天空盒可以在 Scene 视图中直接显示。由于我们的游戏系统是第一视角的，只需要一个摄像机，所以我们采用第二种方式添加天空盒。添加好后的天空盒如图 5.5 所示。

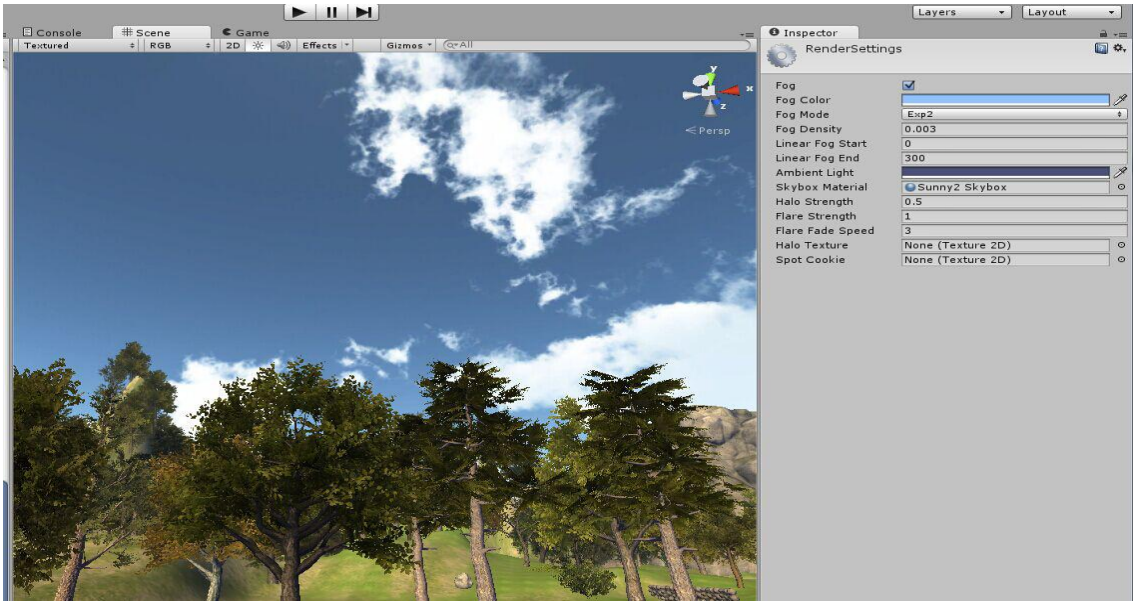

图 5.5 添加天空盒之后效果图

5.4 游戏动画系统的实现

Mecanim 动画系统是 Unity3D 从 4.0 版本以后新添加的动画管理系统。Mecanim 使用的是有限状态机机制来设计动画的。如图 5.6 所示，在 Unity3D 的编辑界面，可以使用可视化界面进行动画的控制，我们首先建立一个动画控制界面，建立之后我们便可以编辑角色的各种状态，然后设置切换条件来切换不同的状态。

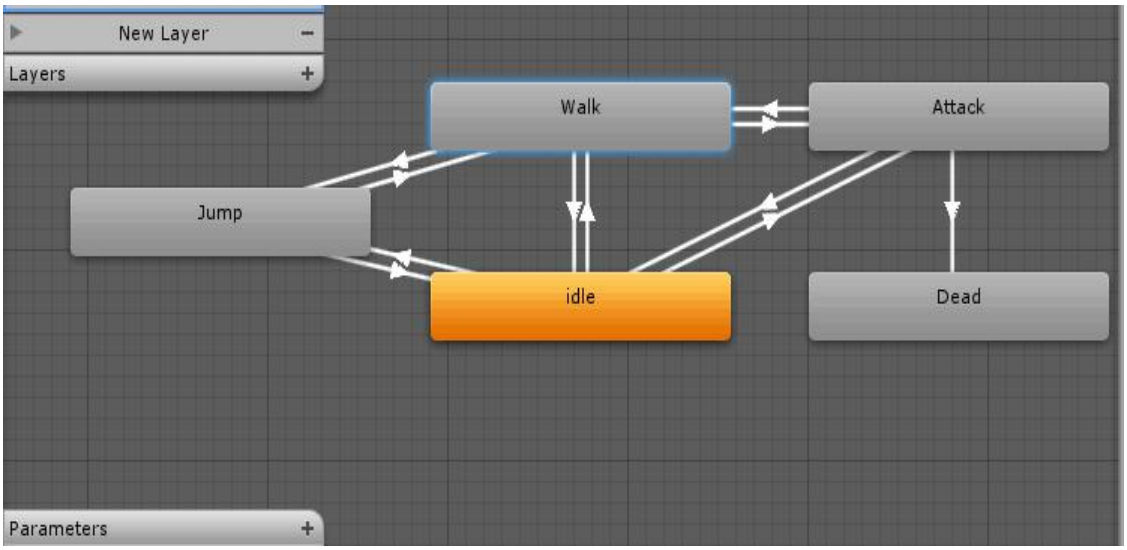

图 5.6 mecanim 状态机

一般游戏的美术人员会使用专业的 3D 建模工具来创建角色模型。而创建有动画的角色模型时就会编辑好所有的角色动画，比如走路、死亡、攻击等[58]。将带有角色动画的模型导入 Unity3D 中的时候需要特别注意的是要确保每个动作和动画的初始帧要保持一致[59]。Unity3D 的另一个好处就是提供了可视化的角色模型操作界面，

如图 5.7 所示，这样就可以使得程序开发者也可以实现类似于简单的移动动画效果。

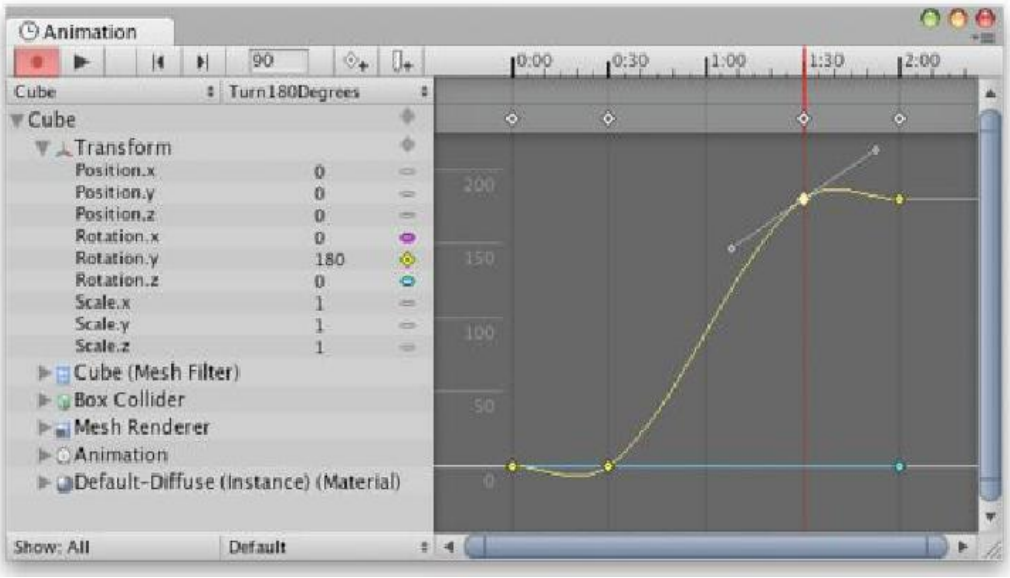

图 5.7 设置简单的动画剪辑

5.5 游戏音效的实现

Unity3D 为了处理游戏的音效为开发者设计了音频源、音频片段和音频监听器三个组件。音频源用于发出音频片段而音频监听器用于接收音频片段。在游戏中的每个场景只能设置一个监听器，用来获取场景中的音频源后再经过麦克风等外部设备播放后让玩家听到音效。在 Unity3D 中每次新建一个场景，Unity3D 会自动生成一个音频监听器，然后将该监听器挂载在摄像头上，这样玩家在使用主摄像头时就能接收来自音频源的声音片段了，但是一个游戏场景中可以有多个音频源，这样就能在游戏里模拟出同时在四处都发出声音的场景了。Unity3D 支持单声道、立体声和多声道音频资源，还支持包括.aif、.wav、.mp3 和.ogg 等多种音频格式的音频[60-61]。Unity 处理立体声的原理是在音源的位置放置音频源，从而播放声音，然后通过距离音频源的距离和位置的不同进行计算后将音频片段进行音量的衰减播放来，从而让玩家认为其实立体声。

5.6 游戏系统中特效的实现

粒子系统在 Unity3D 中是一个经常用来处理特效的系统，比如在使用火、烟雾等效果的时候，一般都采用粒子系统[62]。粒子系统可以十分简单的制作游戏中的特效，一般的粒子系统都会包含粒子发射器、粒子渲染器和粒子动画修改器这三个组

成部分。粒子发射器的主要目的是产生游戏中的粒子特效，而粒子渲染器则是将粒子发射器中发出的粒子进行染色，最后的粒子动画修改器则会计算出粒子的运动轨迹和粒子在运动的时候的变化情况，从而粒子就会自动进行移动和产生变化。有些时候，比如使用粒子模拟爆炸之后的火花，给粒子系统添加一个具有碰撞属性的组件，这样粒子就可以与其他的组件产生物理碰撞。也可以使用粒子系统发送连续不断的图片，开发者只需要将动画材质设置为自己想发送的图片就可以了。在本游戏系统中打算让粒子跟随着玩家，这样就会让玩家有一种很仙灵的感觉。先创立了一个粒子发射器，然后随着玩家的移动而移动，只需要将粒子物体进行实例化就会产生粒子了，最后设置一个时间将产生的粒子删除，这样既可以达成我们的需求也大大节省了内存空间。如图 5.6 所示。

图 5.6 粒子跟随系统

5.7 游戏系统的展示

本文主要实现了相比于普通游戏的不同的体感游戏系统，本系统主要使用了 Kinect 作为输入设备来替代普通的鼠标键盘进行操作。本人将大量的时间用于探索 Kinect 和如何具体得实现游戏系统，另外由于本人在游戏策划方面不太擅长，所以本游戏并没有太多的复杂玩法，主要以简单易操作为主；同时考虑到本游戏的受众面和个人的喜好，所以采用一种比较卡通的模型进行游戏。下面是具体的玩法介绍：

1.玩家进入游戏后从一个小山谷开始，玩家通过 Kinect 或者鼠标键盘控制角色在小山谷中进行游戏，如图 5.7 所示。

图 5.7 玩家控制角色进行游戏

2. 玩家在冒险过程中消灭怪物，如图 5.8 所示。

图 5.7 玩家攻击怪物

3. 当玩家消灭最终怪物或者玩家被怪物消灭，游戏结束。如图 5.8 所示。

图 5.8 玩家被怪物击败，游戏结束

5.8 本章小结

本章主要将 Unity3D 客户端分为图形界面、动态资源加载、静态场景的建立、动画、音效和特效等模块的实现方法。同时与第四章封装好的 Kinect 类库文件结合后进行简单的功能测试，系统整体运行稳定，各模块功能正常，达到了体感游戏系统设计的预期效果。

第六章 总结与展望

6.1 总结

目前来说游戏产业是一个热门产业，尤其是随着高新技术的发展，体感游戏作为其中的一个分类由于使用最自然的方法来进行人机交互，更能使玩家更能体验到一种身临其境的感觉，在既能进行游戏娱乐的同时又能将身体，大脑，行为同时获得锻炼的方式越来越受到欢迎。现今市面上已经推出了越来越多的体感设备，包括 htc、索尼、微软等大型公司也纷纷加入到这一领域的开发和研究之中。本文拟使用微软的 Kinect 进行体感游戏的设计和研究，也为将来能推进到其他体感设备的项目中做了充足的知识储备。

通过本次系统的开发和总结，本文不仅仅让我学会了如何去策划一场游戏的玩法，并将微软的 Kinect 作为外部的输入设备进行操作。将 Unity3D 与 Kinect 结合起来进行开发，设计并完成了一个体感 rpg 游戏。并且还有解决程序的效率问题，整体的美术设计，游戏的 UI 设计等各种关于游戏品质的问题。但是本文核心主要完成的还是关于如何将 Kinect 作为输入设备接入游戏的部分。本文封装了一个 dll 用于将 KinectSDK 的方法重写成适用于本系统的方法，并且最终完成了这样一款能够达到商业要求的体感游戏系统。现在本人做出了如下总结：

1.从以往玩游戏的经验和实际的开发中入手，从开始到结束对游戏的玩法和主题进行策划，设计了总体的开发流程，使得在实际的开发中节约了大量的时间。

2.完成基于 Unity3D 平台的游戏具体设计和制作。

3.研究了各种寻路算法学习并对 A*寻路算法进行改进，从而能够找到最适合于本游戏系统的算法。

4.研究并找到了了如何修改 Kinect 类库将 Kinect 作为输入设备结合到游戏中的方法。

5.实现了一款简单的 rpg 游戏，完成了包括资源加载、游戏动画和 UI 等基本模块部分。

6.2 展望

本系统因为使用的技术比较新，并没有太多的与本系统非常贴切的文献进行参

考，同时需要花费许多的时间进行查找文献和前期的准备工作，所以尽管完成了试运行，仍然存在不少的欠缺和需要不断完善的地方，主要体现在以下几个方面：

1.本游戏系统只是实现了一个单机的游戏系统，由于时间和知识储备的原因还未完成游戏系统的联网，希望能在未来的时间内进行联网模块的设计和完成。

2.由于场景比较大，所以只是完成了一个基本的静态场景的设计，在将来的开发完善中还可以设计更多的场景和模型，而能够完成更为详细的设计。

3.本次设计主要实现一个单人的游戏系统，而 Kinect 还可以采集另外一人的骨骼信息，在改进之后希望可以完成双人游戏的设计和实现。

4.本体感游戏系统目前主要由于软硬件等原因目前只设置了 Kinect 作为输入设备进行体感游戏，而最近一段时间出现了许多类似于 VR 头盔的虚拟现实设备，下一步可以进行更多体感设备的适配。

参考文献

[1] Tommer Leyvand,Caset Meekhof,Yi-Chen Wei,Jian Sun,BainingGuo.Kinect Identity:Technology and Experience[J].Computer,2011,4(44):94-96.

[2] Emest Adams,Andrew Rollings.游戏设计基础[M].北京：机械工业出版社，2008

[3] Sherman W R, Craig A B. Understanding virtual reality[J]. Journal of Documentation, 2013, 59(4):483-486.

[4] Izadi S, Kim D, Hilliges O, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera[C]// ACM Symposium on User Interface Software and Technology, Santa Barbara, Ca, Usa, October. 2011:559-568.

[5] Goswami G, Vatsa M, Singh R. Face Recognition with RGB-D Images Using Kinect[M]// Face Recognition Across the Imaging Spectrum. Springer International Publishing, 2016.

[6] Lee J C. Hacking the Nintendo Wii Remote[J]. IEEE Pervasive Computing, 2008, 7(3):39-45.

[7] 田东. 虚拟现实,从未如此真实 HTC Vive 开发者峰会体验手记[J]. 微型计算机, 2016(2).

[8] 战荫伟, 张昊. 基于 Kinect 传感器的人体行为分析算法[J]. 传感器与微系统, 2015, 34(1):142-144.

[9] 胡鸿.以用户为中心的交互设计[A].Proceedings of the 2011 International Conference on Industrial Design & the 16th China Industrial Design Annual Meeting[C],2011 [1].

[10] Xie J. Research on key technologies base Unity3D game engine[C]// International Conference on Computer Science & Education. 2012:695-699.

[11] Shantz M. Designing a PC Game Engine[J]. IEEE Computer Graphics & Applications, 1998, 18(1):46-53.

[12] Messaoudi F, Simon G, Ksentini A. Dissecting games engines: The case of Unity3D[C]// ACM/IEEE Netgames : the International Workshop on Network and Systems Support for Games. 2015.

[13] 倪乐波, 戚鹏, 遇丽娜,等. Unity3D 产品虚拟展示技术的研究与应用[J]. 数字技术与应用, 2010(9):54-55.

[14] 夏旺盛, 黄心渊. 3D 游戏引擎构架概述[J]. 现代计算机月刊, 2003(6):74-76.

[15] Engelbert R. Cocos2d-x by example beginner's guide[J]. Packt Publishing, 2013.

[16] 王涛. 基于 UDK 的数字校园虚拟现实引擎关键技术的研究[J]. 软件工程师, 2014(4):23-24.

[17] 吴志达. 一个基于 Unity3D 游戏引擎的体感游戏研究与实现[D]. 中山大学, 2012.

[18] 霍常伟. 基于 Cocos2d-x 引擎的移动游戏 UI 系统设计及应用[D]. 北京交通大学, 2012.

[19] 马超, 马佳琳. 基于 Lua 动态脚本语言 Cocos2d-x 应用开发的技术研究[J]. 软件工程师, 2015, 18(8):29-31.

[20] 张金钊. Unity3D 游戏开发与设计案例教程[M]. 清华大学出版社, 2015.

[21] 陈慧芳, 董继先, 王荣. 3D Max 的发展及功能分析[J]. 电影评介, 2013(5):79-80.

[22] 李春燕, 刘少华. 浅析几种三维模型格式导入 Unity3D 的途径[J]. 中国新技术新产品, 2016(5).

[23] Yao W, Liu Y, Li F, et al. Research on Marine Incinerator Operational Training System based on 3dsmax and Unity3D[C]// 2015 International Conference on Network and Information Systems for Computers (ICNISC). IEEE Computer Society, 2015:488-491.

[24] 刘孟全. Unity3D 虚拟现实平台重放功能的设计与实现[C]// 广西计算机学会 2012 年学术年会论文集. 2012.

[25] 王亚萍. 浅析 Unity3D 创建环境地形[J]. 计算机光盘软件与应用, 2012(19):175-175.

[26] 王星捷, 李春花. 基于 Unity3D 平台的三维虚拟城市研究与应用[J]. 计算机技术与发展, 2013(4):241-244.

[27] 尤海宁. 基于 Unity3D 引擎的三维游戏人物动作处理技术研究[D]. 南京师范大学, 2010.

[28] Kourosh K, Sander Oude E. Accuracy and resolution of Kinect depth data for indoor mapping applications.[J]. Sensors, 2012, 12(2):1437-54.

[29] Newcombe R A, Izadi S, Hilliges O, et al. KinectFusion: Real-time dense surface mapping and tracking[C]// Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on. IEEE, 2011:127-136.

[30] Izadi S, Kim D, Hilliges O, et al. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera[C]// ACM Symposium on User Interface

Software and Technology, Santa Barbara, Ca, Usa, October. 2011:559-568.

[31] Zhang Z. Microsoft Kinect Sensor and Its Effect[J]. IEEE Multimedia, 2012, 19(2):4-10.

[32] 张毅, 张烁, 罗元,等. 基于Kinect深度图像信息的手势轨迹识别及应用[J]. 计算机应用研究, 2012, 29(9):3547-3550.

[33] Webb J, Ashley J. Beginning Kinect programming with the Microsoft Kinect SDK[M]// Beginning Kinect Programming with the Microsoft Kinect SDK. Apress, 2012:223-225.

[34] Smisek J, Jancosek M, Pajdla T. 3D with Kinect[C]// IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November. 2011:1154-1160.

[35] 黄康泉, 陈壁金, 郑博,等. Kinect 在视频会议系统中的应用[C]// 中国教育和科研计算机网 cernet 学术年会. 2011.

[36] 况鹰, 李桂清, 姚艺,等. 基于 Kinect 运动捕获的三维虚拟试衣[C]// 中国计算机图形学大会. 2012.

[37] Wei Z Q, Wu J A, Wang X. Research on Applied Technology in Human Action Recognition Based on Skeleton Information[J]. Advanced Materials Research, 2013, 859:498-502.

[38] 郭连朋, 陈向宁, 刘彬. Kinect 传感器的彩色和深度相机标定[J]. 中国图象图形学报, 2014, 19(11):1584-1590.

[39] 王瑶, 项鹏, 孟春阳,等. 新一代Kinect传感器关键技术综述[J]. 黑龙江科技信息, 2015(24):84-86.

[40] 刘中晖. 基于 KINECT 传感器的图像特征点描述子算法研究[D]. 西安电子科技大学, 2014.

[41] 魏婷, 李艺. 国内外教育游戏设计研究综述[C]// 全国教育技术学博士生论坛. 2008:67-70.

[42] 荣钦科技. 游戏设计概论[M]. 北京科海电子出版社, 2003.

[43] Saad W, Han Z, Debbah M, et al. Coalitional Game Theory for Communication Networks: A Tutorial[J]. IEEE Signal Processing Magazine, 2009, 26(5):77 - 97.

[44] 徐小良, 汪乐宇, 周泓. 有限状态机的一种实现框架[J]. 工程设计学报, 2003, 10(5):251-255.

[45] 王巍, 高德远. 有限状态机设计策略[J]. 计算机工程与应用, 1999(7):54-55.

[46]黎文导, 卢瑜. 有限状态机(FSM)的实现[J]. 青海师范大学学报:自然科学版, 2001(4):18-21.

[47]李子强，宋余庆，陈健美，等．基于 Silverlight 网页游戏的寻径优化算法[J]．计算机工程与应用，2013,49(5):59-63

[48]Stamford J, Khuman A S, Carter J, et al. Pathfinding in partially explored games environments: The application of the A＊ Algorithm with occupancy grids in Unity3D[C]// The Workshop on Computational Intelligence. IEEE, 2014:1-6.

[49]蔡方方, 杨士颖, 张小凤,等. 双层 A*算法在游戏寻路方面的研究[J]. 微型电脑应用, 2010, 26(1):26-28.

[50]周小镜. 基于改进 A*算法的游戏地图寻径的研究[D]. 西南大学, 2011.

[51]王肖, 徐友春, 章永进,等. An Improved A*Algorithm in the Shortest Path Searching Based on GIS 一种基于 GIS 最短路径搜索的 A*改进算法[J]. 计算机系统应用, 2008, 17(5):28-31.

[52]樊莉, 孙继银, 王勇. 人工智能中的 A*算法应用及编程[J]. 微机发展, 2003, 13(5):33-35.

[53]明日科技. VISUAL C#开发技术大全(附光盘)[M]. 人民邮电出版社, 2007.

[54]Draheim D, Lutteroth C, Weber G. Generative programming for C#[J]. Acm Sigplan Notices, 2005, 40(8):29-33.

[55](美)考夫曼, 张哲峰. ASP.NET 数据库入门经典——C#编程篇[M]. 清华大学出版社, 2003.

[56]王晟. Visual C#.NET 数据库开发经典案例解析(附光盘两张)[M]. 清华大学出版社, 2005.

[57]王丹力, 华庆一, 戴国忠. 以用户为中心的场景设计方法研究[J]. 计算机学报, 2005, 28(6):1043-1047.

[58]李汉文. 3D 虚拟衣服动画系统关键技术的研究与实现[D]. 兰州大学, 2012.

[59]李亚昆, 张应中, 罗晓芳. 三维实体造型动画系统的设计及实现[J]. 计算机工程与设计, 2004, 25(8):1397-1399.

[60]魏婷, 郑豪. 游戏音效在物理引擎中的运用[J]. 福建电脑, 2008(1):168-168.

[61]黄明元, 廖静. 论网络游戏中不可或缺的元素——声音[C]// 中国科协年会. 2009.

[62] 张芹, 吴慧中, 谢隽毅,等. 基于粒子系统的火焰模型及其生成方法研究[J]. 计

算机辅助设计与图形学学报, 2001, 13(1):78-82.

致 谢

回想起刚刚开始研究生学习生涯之初的自己，再看看现在自己的成长，这三年的日子让我无法忘记。三年的日子里充满了欢笑和快乐。在江苏大学的这段时间里我不仅收获了丰富的知识，提高了个人阅历，更结识了许多乐于助人的老师和同学。在这个即将毕业的时候，我内心波澜涌动，眼前一直浮现出经帮助过支持过我的老师、同学的身影，感谢你们这三年对我的照顾。

首先需要感谢的是我的导师陈健美教授，是她对我的论文研究方向进行了指引，并在整个研究生期间对我进行了耐心的教导。从本次论文的选题开始，陈老师就对我细心的教导，并多次在研究会议中对我论文进行查漏补缺，对本文同时提出了许多的不足和修改建议。从陈老师认真的对待学习的态度和丰富的专业经验让我充分感受到了一个学者严谨的教学和科研态度，也为我们创造了自由、明主、开放的学术氛围，大家一起互相学习、取长补短。在生活中，陈老师也给了我无微不至的关怀。感谢老师的尊尊教诲，这些教诲在我以后的人生道路上也会指引我朝着梦想前行。

在此，还要感谢实验室的同学们。在学习中，他们对我学习中的研究工作提供了很大的帮助，在生活中，我们互帮互助，一起度过了难忘的研究生时光，留下了太多的美好的回忆。

然后还要感谢我的父母和家人，是你们含辛茹苦的将我抚养长大，一直以来，你们都是我的精神依靠，让我能完全没有后顾之忧，在人生道路上激励和鼓舞着我前进。

最后感谢所有审稿和答辩的专家评委们的悉心指导和审查。

在学期间取得的学术成果

获得软件著作权：

本人第一作者.基于 Unity 和 Kinect 的体感游戏系统 V1.0.计算机软件著作权，登记号：2015SR228254，取得时间：2015 年 11 月 20 日.