西南科技大学

Southwest university of science and technology

本科毕业设计（论文）

裸眼 3D 技术
在沉浸式体感游戏中的应用研究

学院名称	计算机科学与技术学院
专业名称	软件工程
学生姓名	周和繁
学号	20121241
指导教师	吴亚东（教授）

二〇一六年六月

裸眼 3D 技术在沉浸式体感游戏中的应用研究

摘要：随着裸眼3D 显示技术的逐渐成熟，结合自然体感和裸眼3D 视觉的沉浸式交互展示系统成为应用热点。本文详细介绍了柱状透镜式裸眼3D 显示技术的原理以及多视点裸眼3D 的实现方案，针对体感交互设计了动作识别算法。设计实现了一款可局域网联机的裸眼3D 体感飞行设计游戏系统。通过捕捉识别人体手臂展开左右倾斜，身体前后倾斜等肢体动作，实现了对飞船的控制。游戏中系统结果表明，结合体感交互的裸眼3D 展示游戏增强了沉浸感，而且显著增强游戏吸引力。

关键词：裸眼3D；体感游戏；Kinect；Unity；局域网联机

The Design and Application of Immersing Game Based On Auto Stereoscopic 3d and Gesture Interaction

Abstract： With Autostereoscopic 3D display technology rapidly developing, the immersive interactive display system which combined with natural somatosensory and Autostereoscopic 3D vision has become a hot spot. In this paper, the principle of 3D display technology and the implementation of multi-view Autostereoscopic 3D was detailed introduced, in addition, a action recognition algorithm is designed for body interaction in system. Finally, a multi-player LAN Autostereoscopic 3D Somatosensory flight game was developed. By capturing the human body action to recognize tilt of body, human body can control of the spacecraft by posture. The result shows that the combination of the Autostereoscopic 3D interactive display game enhanced Somatosensory, but also enhance the attractiveness of the game significantly

Key words: Stereoscopic 3d；Motion-somatosensory game；Kinect；Unity；Online game

目 录

第 1 章 绪论

1.1 概述

随着科技的不断发展，人类对影音体验的需求不断扩张，从1939年推出的黑白电视机，到1954年推出 RCA 彩色电视机。从早期的默片电影，到1952年第一部3D 长片。到了2009年规模空前技术最先进的3D 电影《阿凡达》上映，《阿凡达》中震撼的3D 场景特效让全世界观众享受了一次3D 立体视觉盛宴，而3D 立体显示技术也因此被推上了技术热潮之巅。3D 显示技术其实在70年前就已经诞生了，从诞生到现在，经过长达70多年的研究，3D 显示技术不断突破技术难点，直至今日，目前市面上已经有了多种立体显示设备，分别是：头盔显示器、3D 立体眼镜、裸眼立体显示器以及手执式观测器[1]。

同时游戏行业也不断去陈推新，如今电子游戏已经发展成了文化产业中的支柱型产业[2]。虽然游戏内容日新月异，但是游戏的输入方式却依然无法脱离鼠标键盘等简单的交互方式，玩家首先需要把想要的操作转化为鼠标键盘对应的操作才能与游戏交互，这样增加了游戏的操作难度，同时也让游戏与玩家产生了隔阂[3]。为了让玩家获得更自然的游戏体验，世界著名科技公司 Microsoft 退出了 Kinect 体感设备，微软的 Kinect 不需要使用任何控制器，它依靠相机捕捉三维空间中玩家的运动[4]，让玩家与游戏有了更自然的交互方式。

Unity 游戏引擎也在短短几年内不断崛起[5]。以其强大的跨平台优势特性迅速拥有了大量的用户。同时 Unity 相较其他游戏引擎而言，物理系统、粒子系统、动画系统、及输入控制几大系统的高度集成化和大量可扩展、便于管理维护的插件也更加方便用户开发[6]，尤其是 Unity5.0之后加入了 Enligten 的光照系统，弥补了 Unity 光影表现的短板[7]。

综上所述，裸眼3D 技术和体感游戏将成为未来几年内值得研究的一个方向。同时强大的 Unity 引擎，降低了游戏开发的门槛，使用 Unity 能便捷的将裸眼3D 技术运用到体感游戏中。

1.2 课题背景及意义

裸眼3D，以其生动的表现力，优美高雅的环境感染力以及强大的视觉冲击力感

染了观者[8]。它是图像制作领域的一场革命，更是未来世界的主流发展趋势。因此，裸眼 3D 的探索与研究对于未来文化创意产业的发展具有重大意义。

而裸眼3D 技术在沉浸式体感游戏中的应用研究则是将先进数字视听技术与全新交互方式进行结合的一种尝试，将裸眼 3D 影像配合体感互动体验方式，力求打破常规交互方式的局限，为玩家提供一种更为新颖、有趣、真实的游戏体验。本文主要探究裸眼 3D 与体感交互结合的游戏设计方法，力求在给游戏的形式上注入新的活力，给玩家更加真实的操控体验与视觉冲击。

1.3 国内外研究现状

本来体感游戏的开发就需要了解 Kinect 的工作原理，熟悉 SDK 提供的 API 接口，如何通过用户的骨骼点位置数据识别用户的一次输入手势，还需要较强的3D 数学能力和算法能力。裸眼3D 的实现需要根据裸眼屏幕的硬件特性实现专门的裸眼图片合成算法，用合图算法对每一帧获取的图像处理才能得到能在裸眼屏幕上展示的图像[9]，因此研究裸眼3D 在体感游戏有较高的门槛。

开发裸眼3D 体感游戏比开发普通游戏有更多的限制。在游戏的场景美术设计方面，为了达到更好的裸眼3D 效果，场景内的模型需要菱角分明。由于裸眼算法的约束，需要用八个摄像机同时渲染，在渲染压力上是普通游戏的八倍，这就需要游戏场景上设计上扬长避短，重点突出裸眼的效果。

Kinect 的人体骨骼跟踪需要大量的计算，提供给裸眼屏幕显示的图片的合成也需要在每一帧中进行大量的计算，Unity 游戏引擎作为一款跨平台的引擎，其引擎跨平台的技术是建立在.net 技术基础上的，而.net 本身是在虚拟机中运行，所以使用 Unity 开发的游戏在效率方面本来就不占优势，从三者结合来看，裸眼3D 体感游戏对游戏的优化工作要求较高。

虽然裸眼3D 体感游戏的开发比普通的游戏开发有着更多更苛刻的限制，但是还是有人在该方向进行了研究，北京工业大学的刘立强提出了基于体感交互的裸眼3D 互动展示技术与实现[10]。而在裸眼3D 显示方面，国内外的研究成果就很丰富了，其主要产品击中在液晶显示器、电视机、手机、游戏机等几个方面。裸眼3D 立体显示主要可分为两种技术，光屏障式和柱状透镜[11][12][13]。国外各企业推出的裸眼3D 详细产品信息见表1-1。

表1-1 裸眼 3D 国外企业研究总结表

企业名称	研究方向	研究内容	发布产品	
			年份	产品
夏普	裸眼3D 手机、裸眼显示器	光屏障式	2010	裸眼3D 手机 SH8158U
			2011	裸眼3D 手机 SH8298U
			2013	与飞利浦合作 85 寸 8K 显示器 Fractional Lenticular Lens
LG	裸眼显示器	分光式显示	2011	25 寸显示器 Flatron DX2000
			2012	25寸高清显示器 D2500N-PN
东芝	裸眼显示器、3D 笔记本	柱状透镜	2011	Qosmio F750 笔记本
			2013	21英寸4K 医用裸眼3D 显示器
飞利浦	裸眼显示器	柱状透视	2013	与夏普合作85寸8K 显示器 Fractional Lenticular Lens
			2013	4K*2K 的全高清液晶面板 PFL8830系列
HTC	裸眼3D 手机	立体显示	2011	HTC EVO-3D
三星	裸眼显示器		2014	55寸 Glassless 3D UHD TV
索尼	裸眼显示器、3D 摄像机	光屏障式摄像机	2011	24 寸裸眼显示器
			2012	3D 摄像机 TD20E
任天堂	游戏掌机	光屏障式	2011	任天堂3DS
松下	裸眼显示器	光屏障式	2012	103寸等离子显示器
			2012	145寸8K Super Hi Vision TV
富士	3D 摄像机	摄像机	2009	富士 FinePix REAL 3D W1
			2010	富士 3D W3

在国内的裸眼3D 显示方面的研究情况与国外类似，国内企业如长虹也推出了自己的裸眼3D 显示设备。但是在内容制作方面，国内研究主要在视频显示上，能取得较好的用户体验的只有广告展示。国内裸眼3D 研究的具体情况见表1-2。

表1-2 裸眼 3D 国内企业研究总结表

企业名称	研究方向	研究内容	开发产品
长虹	裸眼显示器	光屏障式	21.5英寸、48英寸、55英寸、82英寸裸眼3D 显示器
			21.5英寸、48英寸、55英寸、82英寸4K 裸眼3D 显示器
重庆卓美华视光	立体显示	柱状透镜	裸眼3D 笔记本

电有限公司	游戏互动		21.5英寸、46寸英寸、82英寸裸眼3D 广告机
广州市朗辰电子科技有限公司	立体显示 软件开发 影像制作	柱状透镜 光屏障式	58、86英寸4K 超高清显示器
			24、32、46、55、65、82英寸1080高清显示器
			对媒体广告系统
浙江天禄光电有限公司	立体显示2D 内容制作3D 内容制作	光屏障式	裸眼立体3D 四核手 N3D-D500
			65寸教学触摸一体机
			裸眼立体平板电脑
			裸眼立体拼屏3*3
			40寸裸眼立体显示器

1.4 本论文结构安排

本论文的结构安排如下：

第1章 绪论：主要分析了裸眼体感游戏的主要研究内容、研究涉及到的技术方面以及当前主要采用的开发手段，该课题的项目背景以及研究意义，国内外在该课题上取得的研究成果。

第2章 裸眼3D 的技术基础：主要介绍了人眼立体视觉形成的原理以及基于该原理的裸眼3D 显示技术，并详细介绍了柱状透镜式裸眼3D 的原理，最后还介绍了多视点裸眼3D 技术。

第3章 体感游戏的技术基础：主要介绍了游戏开发中的组织分工、开发体感游戏过程中的技术要点，游戏美术制作的基本流程以及游戏动画的主要制作方法，游戏策划的使命与职责。最后还介绍了体感游戏接入 Kinect SDK 需要做的准备工作。

第4章 裸眼3D 体感游戏系统方案设计：主要介绍了系统的总体框架，并具体阐述了游戏美术设计；飞船控制系统；裸眼模块实现；网络通信系统；Kinect 的数据获取和工作识别。

第5章 系统实现与结果分析：主要展现了系统中各个功能的实现效果，对实现的结果进行了分析与总结，验证了该系统的可靠性、稳定性以及性能的消耗情况，总结了该系统存在的不足之处与可以改进的方面。

第 2 章 裸眼 3D 体感游戏系统需求分析

本系统将把裸眼 3D 技术应用到一款 3D 体感游戏中，游戏内容主要是局域网联机的飞行射击。游戏采用个人竞技模式，即其他所有的玩家都是自己的敌人，在游戏中玩家要通过肢体动作操作自己的飞船避过障碍物和其他玩家的激光炮，还要找到机会发射激光炮击落其他玩家的飞机。

2.1 项目目标

系统应实现的功能有：

（1） 实现裸眼 3D 显示，玩家在不佩戴任何特殊设备的情况下，能看到明显的立体效果。

（2） 实现体感交互，玩家通过自然人体姿势与游戏交互。

（3） 实现局域网联机功能，玩家能通过局域网一起游戏。

2.2 游戏规则

玩家可在同一局域网下加入其它玩家建立的房间，也可以自己建立房间等待其它玩家的加入，每个房间最多允许 4 个人加入，当房间内的所有玩家都已经准备好之后游戏开始。在游戏中，其它玩家都是自己的敌人，玩家的目的控制飞船射击以击落其它玩家的飞船。每个玩家的飞船初始有 6 滴血，每被击中一次减少一滴血，当血量为 0 时飞船被击落，飞船被击落 3S 后飞船重生。当有玩家总共击落 3 艘飞船后，游戏结束。

2.3 涉众分析

本系统的本质是一款游戏，所以其涉众只有玩家和开发人员和玩家两类，具体的涉众分析见表 2-1。

表2-1 系统涉众分析

编号	涉众名称	对系统的期望
1	玩家	希望看到裸眼3D 游戏效果，游戏有较高的可玩性，能通过肢体与游戏交互。
2	开发人员	能够满足玩家对系统的需求，实现稳定、流畅的游戏。

2.4 游戏非功能性需求

软件的非功能性需求是允许软件以满足用户的业务需求部分以外的要求。非功能性需求的是一个很容易被忽视的模块，但它对一个软件的开发和维护是非常重要的。游戏的本质也是软件，系统非功能性需求应包括系统性能，可靠性，可维护性，健壮性等方面的问题。本游戏的非功能性需求定义如下：

（1）易用性：界面美观，交互姿势符合自然人体动作。

（2）可扩展性：游戏代码结构设计合理，方便扩展加入更多玩法设计。

（3）可靠性：游戏运行稳定，努力不发生故障。

（4）健壮性：游戏在发生故障时应有合理的处理和容错机制。

2.5 游戏功能性需求分析

本游戏的主要用例描述如下，用例图如图 2-1 所示。

（1） 设置显示模式：玩家设置游戏是以裸眼 3D 显示模式还是普通模式运行游戏。

（2） 设置控制模式：玩家设置游戏的操作方式是体感操作或是键盘鼠标操作。

（3） 背景音乐设置：玩家设置是否打开背景音乐。

（4） 控制飞船飞行：玩家操作飞船左右转向，加速，以及爬升和俯冲。

（5） 控制飞船射击：玩家操作飞船瞄准射击其它飞船。

（6） 查看游戏结算：游戏结束后，玩家查看击杀排行榜。

（7） 建立房间：玩家建立房间等待其它玩家加入。

（8） 加入房间：玩家选择房间加入。

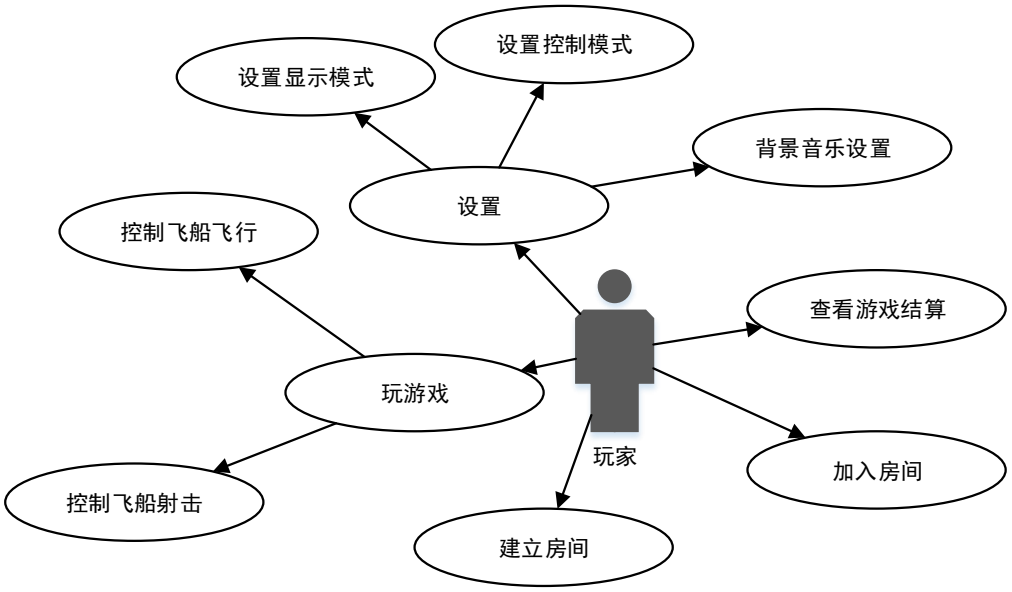


图 2-1 游戏用例图

2.6 业务流程

游戏的业务流程如图 2-2 所示，玩家点击游戏 exe 程序，运行游戏，游戏打开后首先是主选项界面，玩家可旋转加入其它玩家建的游戏房间也可以自己创建游戏房间等待加入，当房间内所有玩家都准备后，跳转到游戏界面，玩家开始游戏，游戏结束后可选择回到主选项界面重新开始游戏也可以退出游戏。

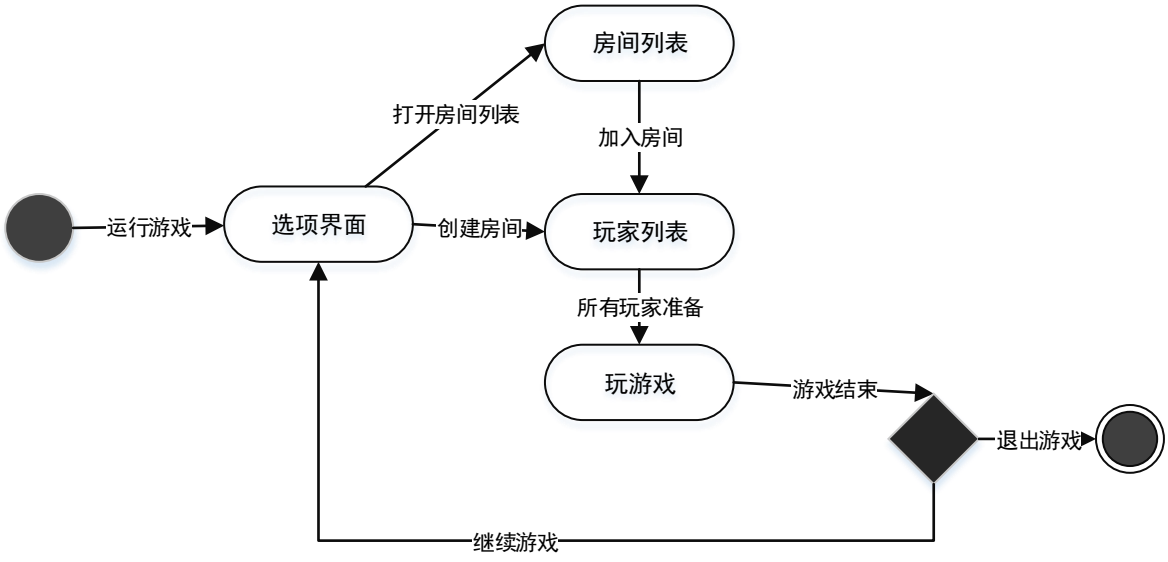


图 2-2 业务流程图

2.7 系统功能设计

本游戏主要分为 5 个模块，如图 2-3 所示，分为裸眼模块、网络模块、体感交互模块、界面模块、游戏逻辑模块。裸眼模块主要处理关于裸眼图像的获取和最终裸眼图像的合成，同时还要包括为游戏提供切换显示模式的功能，网络模块主要处理处理网络相关逻辑。体感交互模块主要处理体感姿势的识别，获取到玩家的操作意图提交到游戏逻辑中，并且需要实现空气鼠标模拟，为方便界面事件，需要兼容 UGUI 的事件系统。界面模块主要处理游戏中各种界面的界面响应和界面刷新，游戏逻辑模块主要根据体感模块传入的玩家操作控制飞船的运动，以及射击，更重要的该模块还要负责游戏的伤害计算。

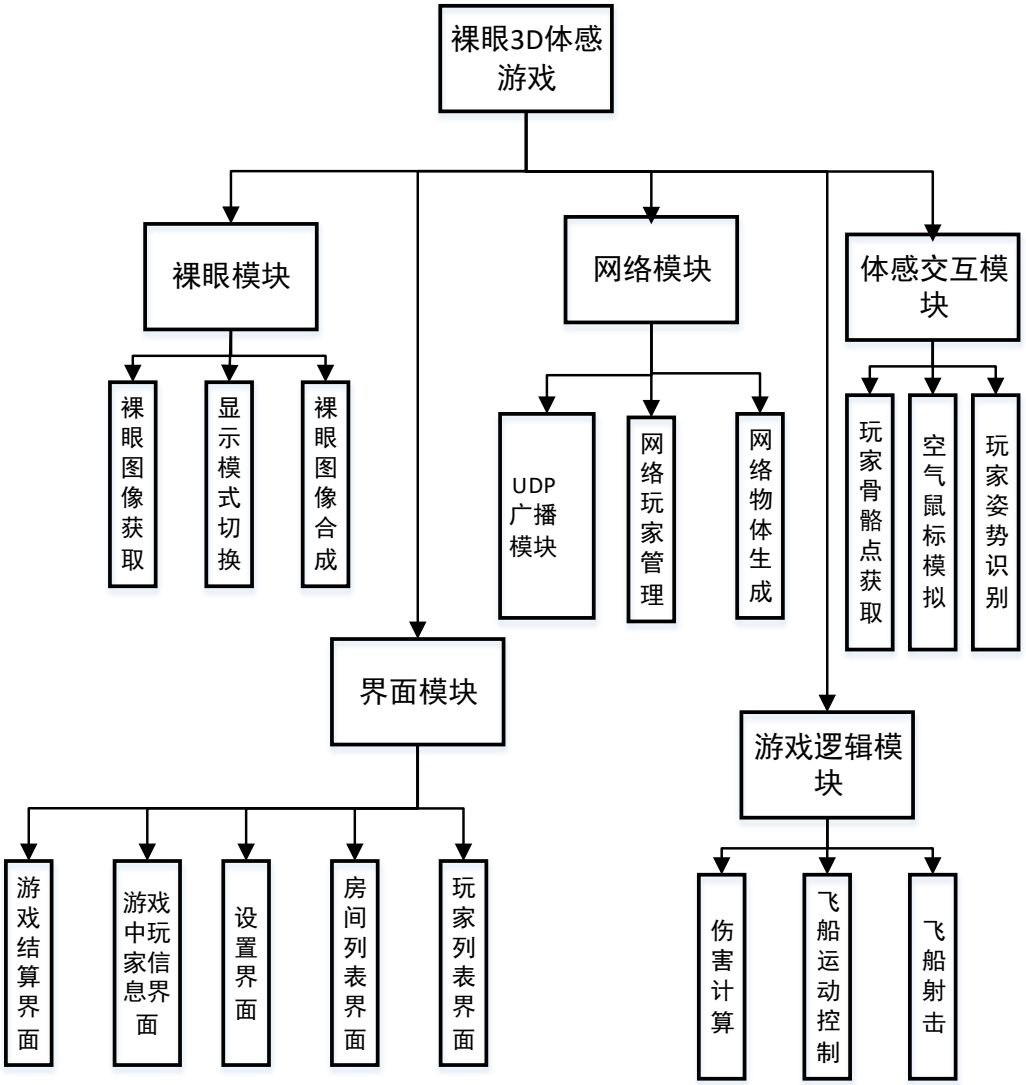


图 2-3 系统功能模块图

2.8 分析类图

经过上一节的系统功能分析，得出如图 2-4 和图 2-5 的一共 19 个类，由于游戏采用的是 Unity 开发，而 Unity 采用的组件式结构，组件式结构提倡类之间采用组件结合而不是继承，这样可以降低类之间的耦合性，提高类之间的灵活性，因此本游戏中的类大多只有关联关系，并没有太多继承关系，下面将分别介绍着些类。

界面管理基类：由于界面类主要处理界面的按钮逻辑，公共方法只抽出了关闭界面和打开界面两个方法。

设置界面管理类：主要包括对显示模式按钮和控制模式按钮两个按钮的响应操作。

玩家列表界面控制类：该类主要的功能是显示房间类的玩家信息。

房间信息类：为一个房间的实体类，包括房间名，房主，并且提供了加入该房间的方法。

房间列表界面管理类：主要负责房间信息的显示和刷新，并且还要处理加入房间按钮的响应。

UDP 广播类：工具类，封装 UDP 广播相关操作。

房间管理类：提供广播的相关函数，并且通过广播消息维护局域网中的房间列表，当房间列表变化时驱动房间列表界面刷新。

游戏信息界面管理类：监听玩家信息改变事件，当玩家信息改变时，刷新玩家的信息，主要包括玩家血量，玩家击杀数。

摄像机控制类：加到游戏主摄像机上的组件，主要功能为控制摄像机跟随玩家的飞船。

飞船运动控制类：通过玩家控制类驱动，当玩家运动状态改变，控制飞船物体运动和发射子弹。

玩家控制类：保存玩家的各种信息，包括姓名，血量，攻击力，击杀数，死亡数，玩家状态，飞船的类型。监听输入管理类的输入事件并驱动飞船控制类来控制飞船物体运动，当接收到射击事件后，实例化子弹。该类还提供进入房间和准备开始游戏的方法。

子弹控制类：子弹物体上的组件，当子弹被实例化后，控制子弹飞行，并且当碰撞到其他物体后将碰撞到的物体和子弹的所有者发送给网络游戏管理类进行伤害判断。

Kinect 控制类：封装 Kinect 操作，从 KinectSDK 获取到人体关节点数据，为姿态识别类提供 Kinect 的操作方法。

裸眼显示类：封装裸眼显示的操作，控制裸眼相机的排列，并且负责合成裸眼图像，为设置界面提供切换显示模式方法。

键盘输入类：监听键盘输入，将有意义的键盘输入转化为操作事件提供给输入管理类。

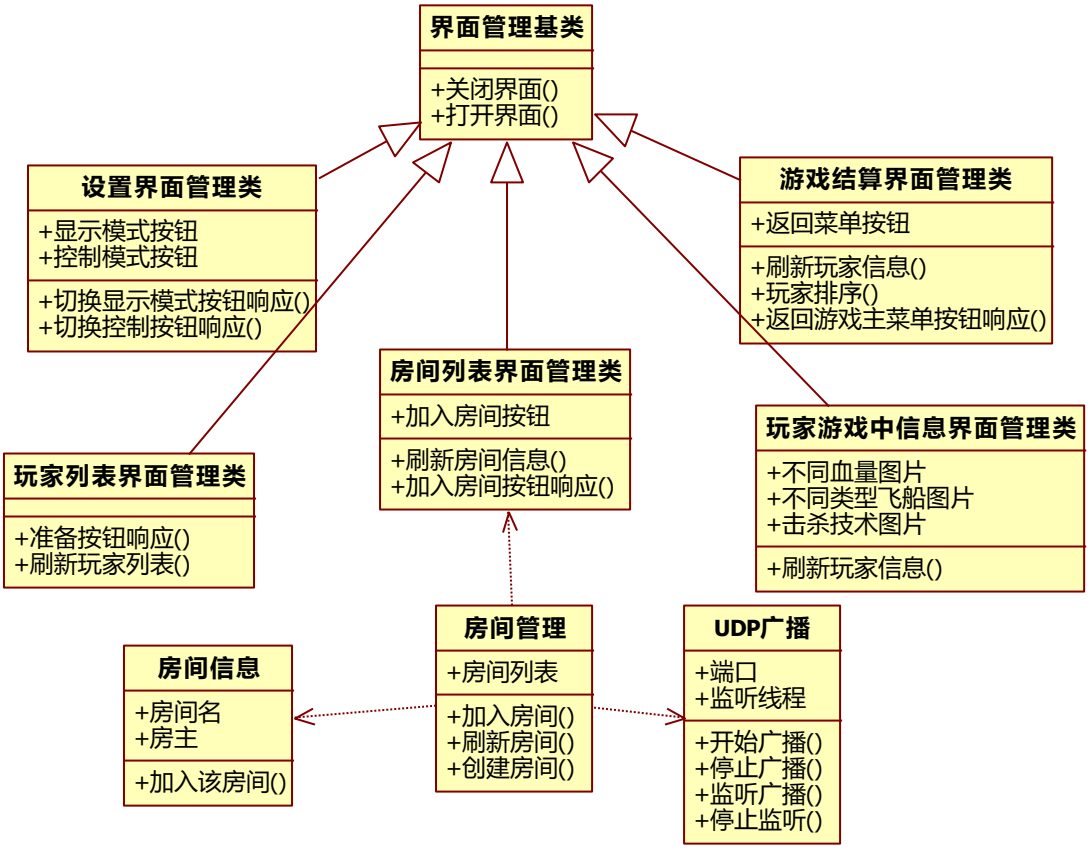


图 2-4 类间的关系 1

人体姿势识别类：通过人体骨骼数据识别人体动作，将有意义的动作映射为玩家操作时间提供给输入管理类。

输入管理类：提供玩家的输入接口给游戏逻辑，同时提供切换输入方式的方法。

网络游戏管理：管理整个游戏的运行，采用单例模式设计，负责网络联接的管理，网络物体的创建和同步，同时还要维护一个当前房间中的玩家列表。

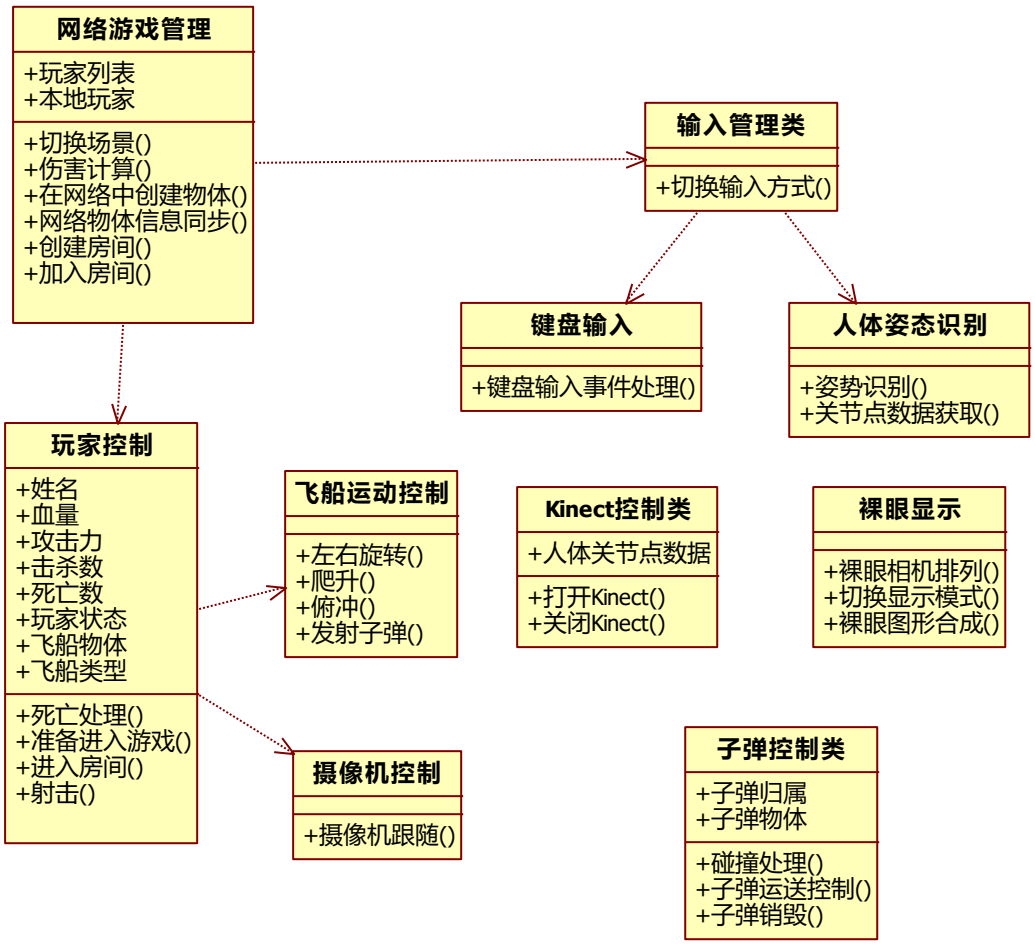


图 2-5 类间的关系 2

2.9 本章小结

为有效地避免游戏架构失误，保证游戏的质量，本章首先描述了游戏的规则，然后进行了非功能和功能需求分析，提出了本系统必须实现的功能点和非功能要求，接着根据游戏规则提出了游戏的业务流程，然后对游戏进行用例分析，通过用例图分析出了游戏的功能模块，最后根据功能模块分析出了游戏的类图。本章对系统的需求进行了详细的分析，为之后的系统设计和开发做好了准备。

第 3 章 裸眼 3D 的技术基础

人眼看到立体影像的主要原理是双目视差是形成立体视觉，即要看到立体图像必须使左右眼分别看到有一定差别的图像。通过特定的装置将输入的两路图像分别转换为左右眼图像，两个眼睛各自看到物体具有不同的空间信息，我们的大脑使用这些信息来判断物体的距离，产生一个错觉，感觉两张图片从屏幕上被拔出来，并通过延长线投射在屏幕的前面。在两个眼睛的光线交汇的地方，就看到一个立体的图像。

裸眼立体显示技术就是利用人眼的立体成像原理，通过使用光学器件在显示屏幕上改变双眼视图的走向，使人的双眼能分别看到对应的立体视图。因为具有不固定的图像通道及图像分割装置，一般称这个实现立体显示的方法称之为空分法。裸眼 3D 显示都运用了上节所说的双眼视差位移的原理来产生立体视觉，实现技术主要有光屏障技术、柱状透镜技术和指向光源技术三种[17]。柱状透镜式裸眼 3D 技术因其亮度不受影响而得到广泛研究，下面将会详细介绍柱状透镜式裸眼 3D 成像原理。

3.1 柱状透镜式裸眼 3D 原理

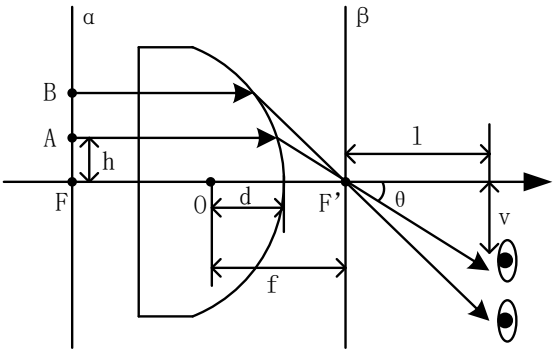


图 3-1 平凸透镜光路图

柱状透镜 3D 技术的原理是在液晶显示屏的前面加上一层柱状透镜，该透镜称为柱状光栅，使液晶屏的像平面位于透镜的焦平面上，透镜下面有多个像素点组成，这些像素点称为子像素，每个子像素经过透镜折射都会在空间固定方向形成影像，人眼处于空间不同位置观看屏幕就会看到不同子像素的影像。柱状透镜使用平凸透镜作为光栅材料，其光学特性可以描述为图 3-1 所示。

图 3-1 中透镜光轴为 x 轴方向，F 和 F' 为第一、第二焦点，O 为透镜光心，α 为

第一焦平面，β为第二焦平面，透镜焦距为 f，光心与透镜顶点的距离为 d。设像素点 A 与光轴距离为 h，经透镜后折射光线进入人眼，人眼与透镜第二焦平面距离为 l，与主光轴距离为 v，设光线经透镜出射点在光轴上的投影距离顶点为δ，根据几何关系有公式 3.1、3.2。

$$\delta = r - \sqrt{r^2 - h^2} \tag{3.1}$$

$$\tan\theta = \frac{h}{f - d + \delta} = \frac{\mathrm{v}}{l} \tag{3.2}$$

由此推导出公式 3-3。

$$\frac{v}{l} = \frac{h}{f - d + r - \sqrt{r^2 - h^2}} \tag{3.3}$$

公式 3.3 中 f,d,r 都是常数，v 与 l 的比值只与像素点与主光轴的距离 h 有关，即(l,v)的坐标关系由 h 决定，随着 h 的变化，像素点光线的传播路线也在发生改变。位于柱透镜焦平面的子像素点坐标发生变化时，其经过透镜折射在空间的投射光路也会改变。

3.2 多视点裸眼 3D 技术

多条平行放置的柱透镜平面被固定在屏幕像素前方，每条柱透镜后面有多个子像素发光点，这些像素点在空间形成固定的可视区域。从多个角度拍摄的场景画面经算法融合后输出立体图像显示在柱透镜后方，透镜的分光作用将多幅图像还原到空间不同位置给用户观看。这种技术被称为多视点技术[18]。多视点示意图如图 3-3 所示。

在图 3-2 中，以 4 视点为例，对于柱状透镜光栅焦平面上的像素点，其发出的光线经过透镜后在空间形成固定位置的可视点，只要人的左右眼睛分别接收到两个不同视点的图像就可以产生裸眼效果。

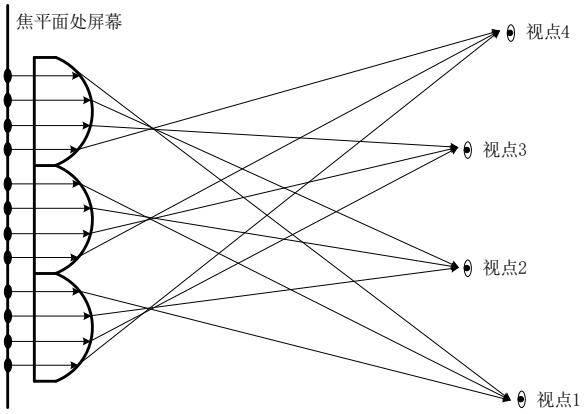


图 3-2 柱透镜光栅成像

对人眼立体感的产生贡献最大的是水平视差[19]，为了得到较好的 3D 效果，往往采用多幅具有水平视差的图像合成图像，而在竖直方向上不做多视点处理。多视点技术提供了自由的观看位置与角度，同时因为裸眼视差都是在水平方向，所以人眼实际接收到的图像分辨率会在水平方向严重降低而竖直方向没有变化，导致分辨率失调。缓解分辨率失调的问题可以将柱状透镜倾斜放置，降低的分辨率被分摊到水平和竖直方向，从而人眼接收到的分辨率不至于太低。

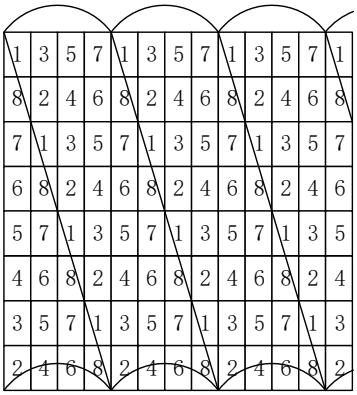


图 3-3 8 视点裸眼透镜摆放与像素索引排列

如图 3-3 所示的 8 视点裸眼屏幕为例，将柱状透镜倾斜放置，与水平面成一定夹角，该夹角由像素点物理宽高决定，屏幕像素的排列也交错放置，在透镜倾斜方向依次排列 1 到 8 视点对应的像素，每个透镜宽度区域内包含了完整的 8 个视点的像素点。经过倾斜放置，可将降低的分辨率分摊到水平和竖直方向。存在一种坏的情况，当人左眼看到视点 8 的图像，右眼看到视点 2 的图像，根据 8 视点排列原理，此时画面左右颠倒，同时模拟两眼的相机间距最大，好比于眼睛瞳距过大，人眼很难聚焦因而无法获得立体图像，长时间观看会给观看者造成严重眩晕感。

3.3 本章小结

本章首先概述了基于视差位移原理的裸眼 3D 技术，并单独介绍了柱状透镜式裸眼 3D 原理。然后，通过柱状透镜式展开介绍了多视点裸眼 3D 技术。本章介绍的裸眼 3D 原理仅仅停留在理论表层，并未做深入研究，是因为本论文是讨论裸眼 3D 技术的应用研究，深入研究原理已经超出研究范畴，本章只是为后面在 Unity 中实现裸眼效果提供理论支持。具体如何在游戏中运用裸眼 3D 将在后续章节中介绍。

第 4 章 Unity 中开发体感游戏的技术基础

体感游戏同普通的电子游戏最大的不同在于，普通游戏采用鼠标、键盘、手柄或者模拟摇杆作为玩家的输入，而体感游戏通过图形分析技术识别出玩家全身的关节点，有了关节点数据进而能分析肢体动作或手势，体感游戏便是通过玩家做出不同的动作或手势控制游戏，体感游戏能给玩家带来新奇有趣的互动体验，同时玩家在游戏过程中需要不断的做出不同的肢体动作，也能达到体育锻炼的目的。本系统采用的体感硬件设备是目前效果最好的微软 Kinect For Windows 传感器[20]。游戏引擎采用 Unity，采用 Unity 主要是因为 Unity 拥用简洁的开发结构，用户能很快的搭建出游戏原型，并且 Unity 也有众多的用户，有着丰富的开发文档和活跃的开发论坛。体感游戏的总体结构如图 4-1 所示。

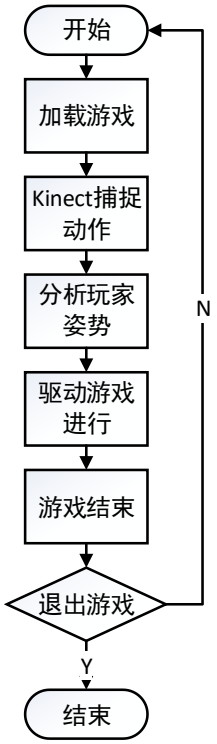


图 4-1 体感游戏框架结构

体感游戏相比于普通游戏主要差别在于，体感游戏的输入方式为玩家的姿势或者手势，在体感游戏的玩法设计上需要考虑如何设计出更自然的人机交互方式，在实现过程中需要将玩家输入提取出单独的模块以方便切换输入方式。除去体感操作之外，体感游戏的开发流程同普通流程没有区别。一款普通的游戏开发流程如图 4-2 所示，游戏开发的需求都是由策划提出，策划会分别像程序开发人员和美工设计人员提出需

求，美工接受到策划的需求后开始画出游戏场景设计的原画，完成原画后美术将原画反馈给策划，策划不满意则提出修改意见美术重新设计原画。策划满意则开始下一步美术制作，主要包括模型制作，UI 图标设计。程序接到策划提出的需求后首先实现游戏的底层系统如：网络系统、热更新系统、游戏文件加载模块等等。当美术开始制作模型后，程序也开始具体的游戏模块设计如：角色控制模块、UI 模块、人工智能模块等等。最后美术模型制作完毕将模型导入到游戏中，完成整个游戏。

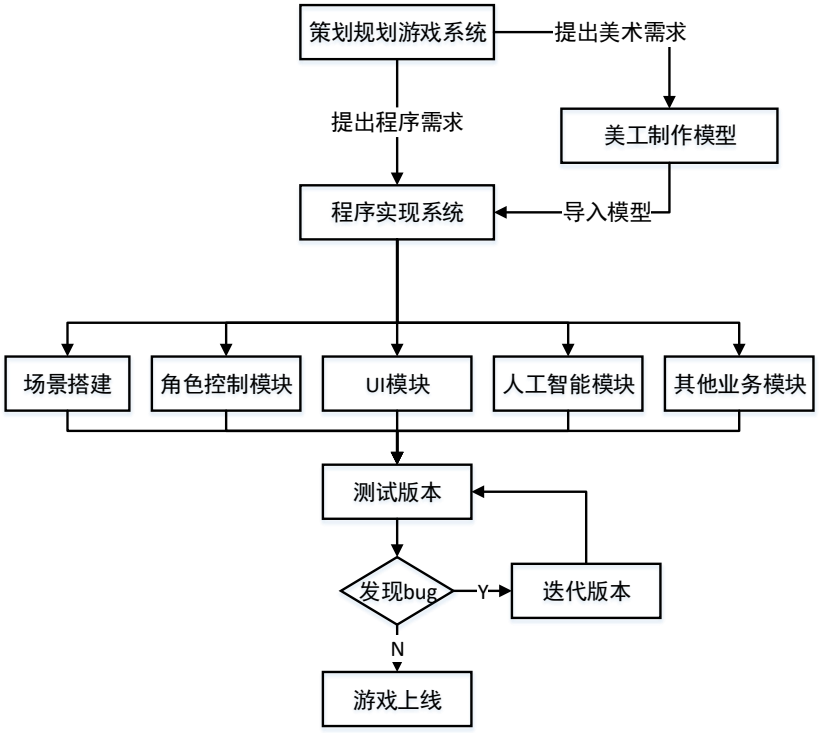


图 4-2 3D 游戏的基本开发流程

4.1 游戏开发的组织分工

从上一节介绍的游戏开发流程可以以看到，游戏的画面依赖于大量的美术资源，游戏可玩性依赖于关键性的玩法设计和数值系统，这也是其不同于其他软件开发最大的不同。由于游戏开发涉及到不同专业领域的技术知识，因此往往一个能相互协调的团队显得异常重要。一个专业完整的游戏团队，主要由制作人、构成[21]，对应的职责如下：

（1）制作人：全面掌握了从研发到运营整个体系知识的游戏项目的总负责人，把控游戏开发的进度以及游戏产品的质量把控；

（2）程序员：市面上的游戏通过是否联网可简单分为网络游戏与单击游戏，网

络游戏有客户端和服务器端，客户端同服务器的开发在技术方向上有很大的区别，很少有人能同时精通这两个方向，所以开发单击游戏只需要客户端程序员即可，而网络游戏则还需要开发服务器的程序员；

（3）美工：美工也是游戏开发的核心人员，在一些游戏团队中美工甚至比程序更重要，一款成功的游戏必定拥有一流的美术设计，美术主要负责游戏中玩家模型的制作，游戏场景的设计以及 UI 图标的设计；

（4）运营人员：游戏运营主要职责是处理玩家关系，负责游戏的推广，玩家反馈的整理，当游戏出现有损玩家利益的 BUG 时需要第一时间发出补偿弥补玩家损失，运营人员还要与策划人员合作推出游戏活动，增加游戏人气；

（5）策划人员：策划是一个款游戏的灵魂人物，策划又可细分为数值策划、系统策划、文案策划、剧情策划。数值策划负责设计游戏中的数值关系如升级所需经验值、伤害值等等，系统策划负责游戏中的系统设计如背包系统，剧情策划同文案策划工作职责并没有明确的区分都负责游戏中有关文字的部分；

（6）测试人员：测试人员最主要的职责是保证游戏质量，主要工作是找出游戏中的 BUG，反馈给程序人员；

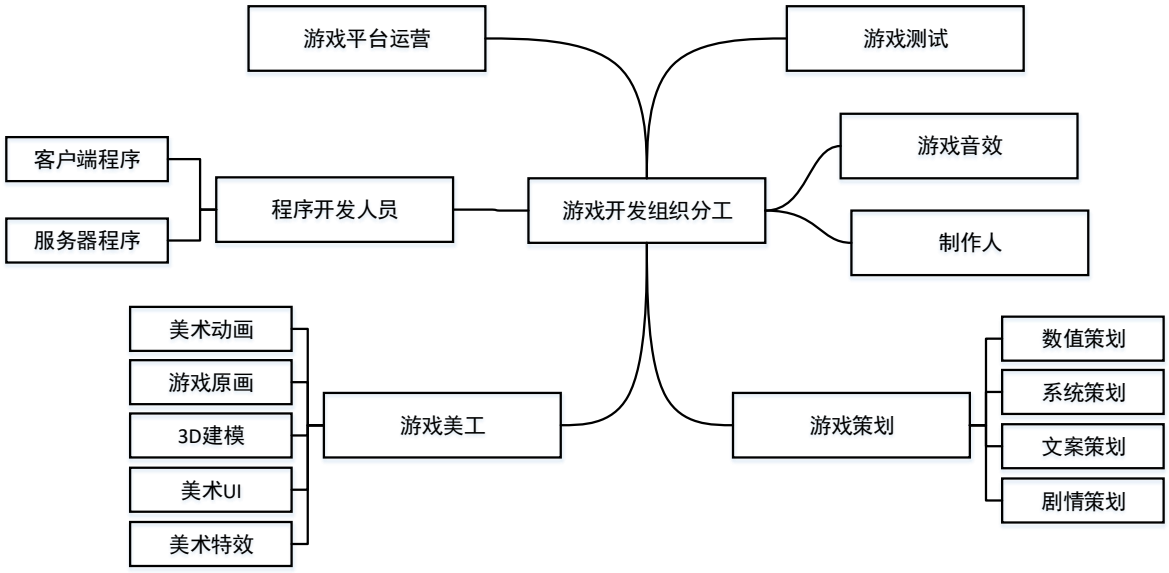


图 4-3 3D 游戏开发的详细组织分工

以上简要概括了游戏开发过程中各个角色的主要职能，一般来讲，一个游戏往往可能包含多个复杂的系统。详细来说，会有更多的角色需要参与到游戏开发当中，图 4-3 为游戏开发的详细组织分工。

4.2 体感游戏开发的技术要点

体感游戏的开发即要考虑 Kinect 的运行环境，也需要考虑 Unity 的运行环境，幸运的是 Unity 采用跨平台的.NET 运行环境 Mono 作为运行环境，可以通过动态链接库的方式封装好 Kinect 接口提供给 Unity 使用。在介绍体感技术前，先介绍 Unity 相关技术。

4.2.1 Unity 开发中组件的编写

Unity 支持三种语言编写的脚本，分别为 JavaScript、C#和 Boo。Unity 中的 JavaScript 与在浏览器中运行的 JavaScript 只是在语法上类似，实际上是 Unity 的开发公司专为 Unity 设计的语言，其和 Boo 都是最终被编译到能在.net 上运行的中间语言，而 C#有微软的技术支持，有着全面高质量的基础类库，因此在 Unity 中使用最普遍的是 C#语言。

Unity 提供给用户的开发结构为组件模式，组件模式推荐以组件的方式处理类之间的关系而不是继承，这种模式能降低类与类之间的耦合性。在 Unity 中最基本的物体是 GameObject，在游戏中各种各样的物体是都是通过在 GameObject 上挂载组件的方式实现，一个物体可以挂载多个组件。如果挂载物理组件那么这个物体就有能进行物理模拟，挂载上碰撞盒那么这个物体就能处理碰撞。正是这种方式提高了程序的灵活性，让用户更容易接受，这也是 Unity 入门容易的原因。

在 Unity 中，只要一个类继承了 MonoBehaviour 类，这个类即可称为组件，一个组件可以完成一种特殊的功能，也可以是一个单独的工具类。MonoBehaviour 提供了一系列的方法，按照 Unity 调用的时机排序分别为：Awake()，Start()，Update()，LateUpdate()，FixedUpdate()，OnTriggerEnter()，OnDisable()。Awake 函数为组件所挂载物体初始化时调用，一般在该函数中执行初始化工作。Start 函数在所有组件的 Awake 执行完毕后在整个游戏的第一帧执行前调用。Update 函数是一个非常重要的函数，该函数在游戏运行的每一帧都会调用，主要做一些逻辑更新操作，由于该函数调用频率非常高，所以在写该函数时需要严格控制代码执行效率。LateUpdate 在所有组件的 Update 函数被调用后调用，其最典型的应用场景为摄像机跟随逻辑实现。FixedUpdate 函数会根据游戏运行效率决定是否调用，一般在该函数中处理计算量较大的逻辑，如物理模拟。OnTiggerEnter 函数载物体挂载了物理组件和碰撞器组件的前提下，当物体与其他碰撞器碰撞时被调用。当物体被销毁时，该物体上挂载的所有

组件的 OnDisable 函数被调用。在 Unity 中创建脚本会默认实现 Start 函数和 Update 函数，如果没有用 Update 函数一定要删除掉，不然空函数也会被每帧调用，影响游戏性能。

4.2.2 位移和旋转

在 3D 游戏当中，位移与旋转是最基本、核心的要求之一。在 Unity 中直接或间接与位移和旋转变换相关联的组件主要有：Transform 组件、RigidBody 组件、NavMeshAgent 组件、CharacterController 组件等等。各大组件见有功能重合部分，但又有细微的区别，如果不能够理清他们的关系，很难很好的控制物体的运动。表 4-1 列出了位移组件间的区别。

表4-1 Unity 中位移相关组件

组件	函数/属性	描述
Transform 组件	Translate 函数	向某方向移动物体多少距离或者相对某物体移动
	Position 属性	在世界空间坐标 Transform 的位置
RigidBody 组件	Velocity 属性	刚体的速度向量
	AddForce 函数	添加一个力到刚体。作为结果刚体将开始移动。
	MovePosition 函数	移动刚体到指定点
NavMeshAgent 组件	SetDestination 函数	设置自动寻路的目标点
CharacterController 组件	Move 函数	一个更加复杂的运动函数，每次都绝对运动
	SimpleMove 函数	以一定的速度移动角色

旋转在 3D 中是比较复杂的，一般描述旋转有三种方式：旋转矩阵、四元数和欧拉角[23]。其中四元数与旋转矩阵相比其只由 4 个数组成，而旋转矩阵需要 9 个数，四元数占用内存小于旋转矩阵，并且四元数不会有欧拉角的万向锁问题。除以上几点之外，四元数更容易插值，更容易做规范化和正交化。但四元数也有其缺点，四元数是从矩阵变换而来的，直接操作四元数抽象性较高。在 Unity 中，提供了非常简洁的 API 操作旋转，屏蔽了底层的复杂数学，四元数在 Unity 中的操作全部被封装在 Quaternion 类中，表 4-2 列举了 Quaternion 类中几个常用的 API。

表4-2 Unity 中旋转相关函数

函数	描述
LookRotation（Vector3 forward, Vector3 upwards）	创建一个旋转，沿着 forward（z 轴）并且头部沿着 upwards（y 轴）的约束注视。也就是建立一个旋转，使 z 轴朝向 y 轴朝向 up。

Angle（Quaternion a, Quaternion b）	返回传入的两个旋转之间的角度。
Euler(float x, float y, float z)	返回一个旋转角度，绕 z 轴旋转 z 度，绕 x 轴旋转 x 度，绕 y 轴旋转 y 度
Slerp(Quaternion a, Quaternion b, float t)	球形插值，通过 t 值 from 向 to 之间插值。
FromToRotation(Vector3 fromDirection, Vector3 toDirection)	从 fromDirection 到 toDirection 创建一个旋转。
Quaternion.identity	返回恒等式旋转（只读）。这个四元数对于“无旋转”：这个物体完全对齐于世界或父轴。
Quaternion.operator *	由另一个四元数来旋转一个旋转角度，或由一个旋转角度来旋转一个向量

4.2.3 Unity 中的协程

C#提供了对多线程的完整支持，但 Unity 引擎是主循环结构，逻辑更新和画面更新的时间点要求有确定性，如果在逻辑更新和画面更新中引入多线程，就需要处理线程间数据同步问题，而处理这种问题及时是经验丰富的程序员也很容易出错，为此 Unity 提供了一种“伪多线程”的处理方式，这种方式被称为协程。

从程序结构的角度来讲，协程是一个有限状态机，说到协程，首先需要提到另一样东西，那就是子例程，子例程一般可以指函数，函数是没有状态，函数返回之后，函数的所有局部变量会被释放内存，但是在协程中可以在一个函数里多次返回，局部变量被当作状态保存在协程函数中，直到最后一次返回，协程的状态才被清除。简单来说，协程就是：一段顺序的代码，标明哪里需要暂停，然后在下一帧或者一段时间后，系统会继续执行这段代码[24]。本质上，协程和多线程根本不是一个概念，但是可以通过协程的使用，来实现宏观上程序中数据的同步。

协程的返回值必须返回迭代器 IEnumerator，而协程的调用则需要用 StartCoroutine 函数调用。协程有几种典型的使用方式：

1．可通过 WaitForSeconds 类达到延时效果，如 yield return new WaitForSeconds (5)，Unity 调用到这句代码后立即返回，在 5 秒后 Unity 继续调用返回语句下面的语句。

2． 返回 NULL 达到并行运行的目的，当 Unity 运行到语句 yield return null 立即返回，在游戏的下一帧的 Update 调用后继续调用返回语句后的语句。通过将 yield return null 语句放入 while(true)循环内即可达到一种与“伪并行”调用的效果。

协程在 Unity 中是一种很便捷的编程方式，在实现延时问题，异步调用问题上有着得天独厚的优势。

4.2.4 组件模式下对象间通信

在游戏开发中，除了一些如运动之类必须要每帧调用不断更新的逻辑外，为了提高程序执行效率，很多逻辑都是通过对象间相互调用驱动的。在 Unity 的组件模式下游戏对象间通信本质上是挂载在对象上的组件间相互通信，总的来说有三种通信方式：事件订阅、发布方式；GameObject 上的 SendMessage()方法实现函数调用；直接将函数和属性定义为公用，关心该属性的物体通过 FindGameObject()方法找到该物体，通过 GameObject 上的 GetComponent()方法获取组件对象，然后在每一帧中刷新组件的属性，监听组件属性的改变。这三种方式各有各的优势和应用场景，为了集中叙述界面刷新方式，事件订阅、发布方式将在 5.2.3 节中详细叙述。这里主要详细阐述另外两种方式：

1．直接调用公开属性和方法

直接调用方式适合简单的应用场景，Unity 提供了可视化的拖拽操作为组件公开属性赋值，组件的属性界面上所有公共属性都有一个编辑槽，可将想调用的组件所依附的物体直接拖放到编辑槽上，使用这种方式就可省去在代码调用 FindGameObject 方法和 GetComponent 方法。这种方式虽然便捷但是在复杂的逻辑中面对组件间复杂的关系往往会显得混乱不堪，而且将属性和方法公开也违反了面向对象的封装原则，不可滥用这种方法。

2．SendMessage 方法

SendMessage 是 MonoBehaviour 类的父类 Component 类提供的方法，所以所有继承了 MonoBehaviour 的类也有了 SendMessage 方法。SendMessage 接受多个参数，第一个参数是字符串，为调用函数的名字，后面所有的参数依次为调用该函数的参数。通过这个方法只用知道函数名就可调用组件上的该方法，不论是公有方法还是私有方法。这种方式可避免上一种方法破坏封装性的缺点，但是面对复杂的组件间关系还是显得力不从心。

4.3 游戏美术及动画的主要制作方法

4.3.1 游戏美术的基本制作流程

前面在阐述游戏团队的组成时就阐述过美术资源对于游戏的重要性，因此在游戏中占有重要的美术资源在制作过程中也有一套固定的流程。美术制作流程中的很多步骤都是和程序工作有关联的，所以程序也需要大概了解美术的工作流程，但是由于作者并不是美工，术业有专攻，这里只是稍微介绍下最基本的流程。如图 4-5 所示，首先由策划提出美术需求，通过简单的示意图和文字描述整个游戏世界应该长什么样，风格是写实还是卡通，游戏故事背景是怎样。原画美术根据策划描述开始制作场景和角色原画设计，设计完成后再有策划确认，策划觉得体现出了自己所描述的场景后，美工开始全面制作 3D 模型、模型贴图、UI 图标，做完模型并贴上贴图后还要让模型动起来，即动画得制作，动画也由策划向动画师提出动画需求，动画师制作完毕策划审核，反复迭代直到达到策划所需要的效果为止。加了动画的模型需要导入到 3D 游戏引擎中，在实际的程序中展示效果，在这一步中美术资源需要和程序协调，在程序效率和美术表现上有时只能满足一个，这是就需要策划来取舍，所以美术资源还有可能会调整。以上流程贯穿整个游戏开发周期，美术资源的制作也需要反复迭代，在商业游戏开发中，甚至游戏开发到一半美术资源推翻重做这种极端情况也有出现。

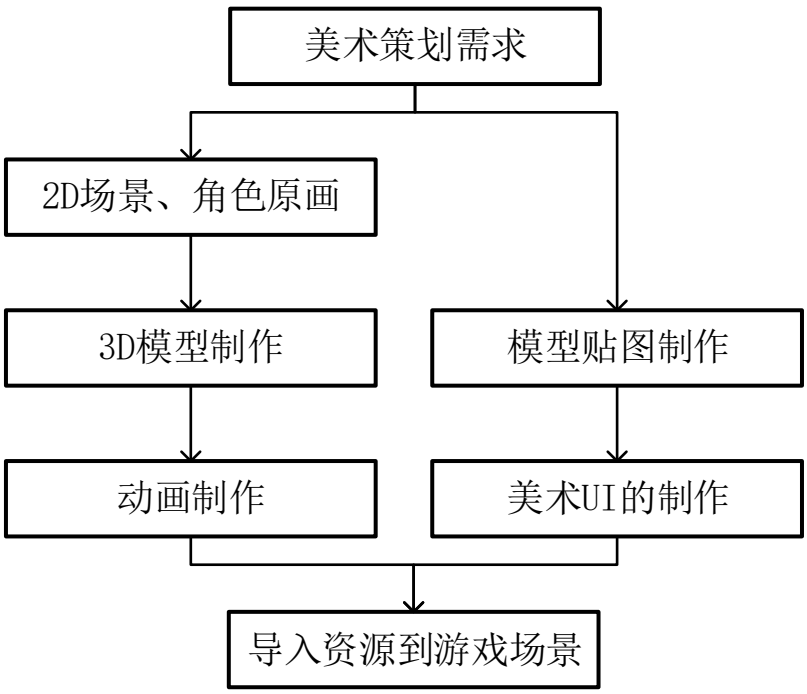


图 4-5 游戏美术的基本制作流程

4.3.2 Unity 中动画的几种实现方式

在游戏中动画按照维度可分为 2D 动画、3D 动画。2D 动画主要用在 2D 游戏中或者 3D 游戏的界面上，实现方式主要是采用逐帧动画。3D 动画主要用在 3D 模型上，主要实现方式有蒙皮骨骼动画和插值动画。对于动画这种基础模块，Unity 中提供了完善支持，基本只用简单的几部就可以实现动画效果。下面将分别详细介绍这三种动画实现原理以及在如何在 Unity 中使用：

（1）逐帧动画

逐帧动画是将一个连续的动作细分为一帧帧图片，将这些图片以一定的效率播放出来就会给人眼睛造成连续动画的错觉[26]。动画图片可一张张分别画也可由 3D 模型录制出来。在 Unity 中先导入由美工制作好的动画图片，剩下的工作就是切换动画每一帧图片。切换图片可以在代码中直接替换显示图片，更简单的方法是利用 Unity 提供的动画制作器和 Animation 组件。如图 4-6 为使用 Unity 动画制作器制作爆炸效果的界面，可直观的看出整个动画由八张图组成，在界面顶部的一个个菱形节点即是切换图片的时间节点。将一个制作好的动画片段拖入到 Animation 动画组件中即可通过动画组件控制动画片段的播放和暂停已经更多的细节操作。

虽然逐帧动画制作简单，在 Unity 中使用起来也很方便，但是要想获得更平滑细腻的动画效果，就需要投入更多的美术成本，制作帧数更高的动画。同时对于程序来说，帧动画会占用相当一部分内存。

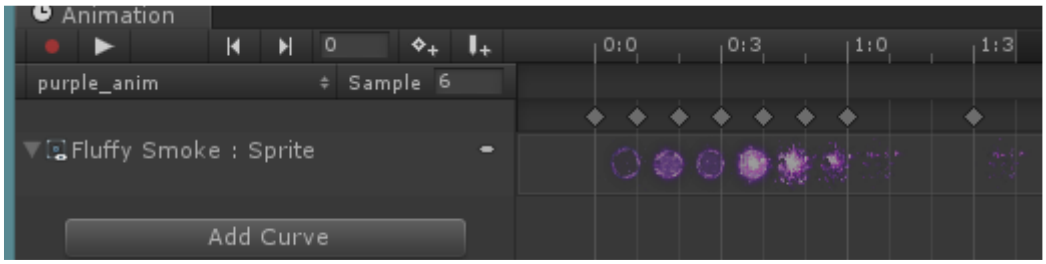


图 4-6 逐帧爆炸效果动画

（2） 蒙皮骨骼动画

蒙皮骨骼动画是通过数学的方法模拟一套骨骼，而模型表面的点则根据骨骼点的位置移动[27]。该方法多用于人体和动物的动画，因为人体和动物身体内有骨骼，使用蒙皮骨骼动画能完美的模拟人体或动物的动作。而如果将该方法放在其他物体上，往往效果并不理想。骨骼动画得制作可由动画师根据经验一帧一帧的调整骨骼点的位

置，也可采用更先进的动作捕捉技术用真人完成动作，通过骨骼点跟踪设备记录骨骼的位置，再将动作数据导入到动画制作软件中加以修饰以得到更好的动画效果。

在 Unity 中有 Avatar 概念，每个拥有骨骼的模型导入 Unity 后，Mecanim 组件会自动为该模型生成一个 Avatar 文件，只要两个模型的骨骼结构相同，则两个模型通过使用同一个 Avatar 组件可实现蒙皮骨骼动画的共用。播放蒙皮骨骼动画需要使用 Unity 中的 Animator Controller 组件，将该组件挂载到模型的根节点上即可通过该组件控制物体的动画了。

（3） 插值动画

插值动画本质上是通过算法给定两个值，计算得到中间的值。在 Unity 中可通过动画制作工具可视化的编辑动画曲线生成插值动画，但是使用起来麻烦需要先制作动画片段再通过 Animation 组件控制播放。为此有了一些插值动画插件的出现，比较著名的有 ITween 插件，ITween 是以 dll 的方式提供的，因此只能在代码中使用，通过 ITween 提供的接口可以很方便的创建如移动、颜色变化、音量变化、旋转等插值动画。

4.4 使用 Kinect 前的准备

在 Unity 中使用 Kinect 前需要做以下准备：

1. Kinect 是硬件设备，所以首先要到微软官网下载 Kinect 驱动包，安装完成后将 Kinect 连接到电脑上，在 Kinect 驱动安装目录下，找到 SDK Browser 程序，运行程序选择 Kinect Studio 运行，若能显示 Kinect 捕捉到的彩色图像证明驱动安装成功并且 Kinect 设备运行正常。

2. 微软为提供 Unity 提供的 Kinect 开发插件包中包含了 Kinect 的全部功能，完全导入的话游戏打包后会占用几十兆的空间，并且一些如三维重建模块需要消耗巨大的计算量。因此导入开发包后需要根据项目需求删掉不需要的部分。

3. 在游戏开发过程中需要不断的运行游戏测试效果，而游戏采用体感控制时运行游戏会很繁琐，因此在项目架构设计时需要将输入控制单独出来，方便在测试时切换为普通输入，提高测试效率。

4.5 本章小结

本章首先简单介绍了完整游戏团队的成员组成与分工，明确了游戏开发过程中几个重要角色的职责。然后分 4 小节详细介绍了 Unity 开发体感游戏的技术基础。由于游戏开发涉及到大量美术，在工作中不可避免要与美术协调工作，因此又简单介绍了游戏美术制作的基本流程和几种实现动画的方式，最后介绍了在 Unity 中使用 Kinect 前的准备工作。本章主要对使用到的几点关键技术做介绍，更多的细节设计及编码工作将在下一章中体现。

第 5 章 裸眼 3D 体感游戏系统设计

本系统实现了一款局域网联机 3D 射击体感游戏，并使用裸眼 3D 算法将游戏画面合成能在裸眼屏幕显示的图像。因为游戏地图的容量限制，同时考虑到可玩性，最多可四个人通过局域网联机游戏，玩家通过特定的肢体动作操作飞船 360 度飞行，通过手势控制飞船发射激光炮击杀敌人玩法采取个人竞技模式，其他人都是你的敌人。当某一玩家击杀超过 3 人后，游戏结束。肢体动作的识别有一定的延迟，并且也会有一定的误差，所以游戏总体来讲比较考验用户的反应能力和操作能力，因此游戏设计飞船碰到障碍物后没有惩罚，以此降低游戏难度。

在本文第二章中已经对游戏的需求进行了详细了分析，本章将详细介绍游戏各个模块的实现细节。

5.1 游戏整体结构设计

本游戏的结构整体分为：硬件接口层、网络层和应用层三层，如图 5-1，为游戏整体结构设计图。

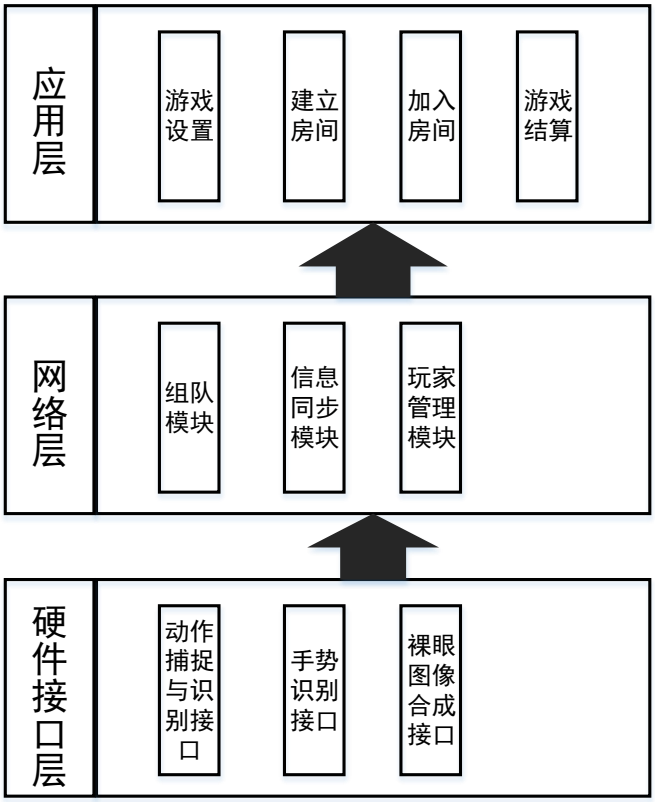


图 5-1 游戏整体结构设计图

在第二章中已经简要描述了游戏的运行流程，在本节将详细阐述游戏的运行流程，如下为游戏详细运行流程：

（1） 运行游戏，游戏首先初始化 Kinect 模块，根据是否初始化成功选择游戏操作方式，根据游戏设置选择是否打开裸眼模式。然后进入游戏主菜单；

（2） 用户选择创建房间或者加入房间，进入房间后，选择进入准备状态，当房间内所有人进入准备状态后，进入游戏场景；

（3） 进入游戏场景后，飞船开始自动飞行，用户通过偏转身体方式实现对飞船的方向控制，通过握拳攻击其他飞船。当被其他玩家的激光炮击中后会减少血量，血量小于等于零则判定为被击杀；

（4） 开始游戏后也可以进入游戏菜单界面，可以选择关闭背景音乐或者退出游戏，由于是网络游戏所以进入菜单界面游戏并不会暂停；

（5） 直到有人击杀数达到 3 人，游戏结束，展示结算列表。

以下小节将从游戏美术设计、裸眼模块实现、网络通信系统实现、体感控制实现几个模块分别阐述系统的设计过程，图 5-2 为该系统的总体流程图。

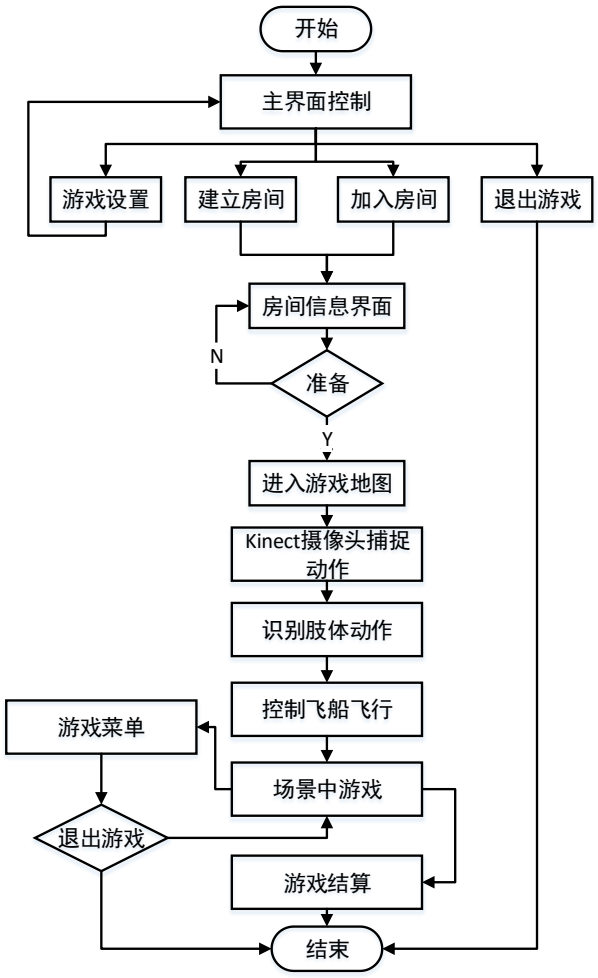


图 5-2 系统总体流程

5.2 游戏美术设计

游戏美术的重要性在 4.1 节中已经简单讨论过了，玩家打开游戏第一眼看到的就是游戏的画面，如果游戏没有一个能抓住玩家眼睛的画面表现，玩家根本没有心情探究游戏的玩法。现在很多商业游戏对美术的投入也是越来越巨大，这也能从侧面表明游戏美术设计的重要性。

本节将从游戏场景设计、游戏特效设计和界面设计三个方面来阐述游戏中的美术设计。

5.2.1 游戏场景设计

为了支撑游戏玩法，避免因为游戏场景太大，无法找到玩家的位置，如图 5-3 所示，本游戏的场景是一个比较简单的室内场景，整个场景已经在建模软件中组装完成，只用导入 Unity 中即可。将模型导入 Unity 中也有几点需要注意的地方：首先模型格

式最好为 fbx 格式，因为 Unity 对 fbx 格式的模型支持最完善，如果模型格式不是 fbx，常用的建模软件如 3DMax、Maya 都支持将其他格式的模型导出为 fbx；其次还要注意模型贴图格式，Unity 支持的格式为 dds、png、tga、jpg。如果贴图采用的是 Unity 不支持的格式，在导入模型前要先把格式转为 Unity 支持的格式，否则会造成模型贴图丢失，再重新转换格式后需要手动为模型指认贴图，将会是一件很繁琐的事情；

为了优化性能，将整个静止的场景都标记为静态（static），在 Unity 中静态物体在脚本加载的时候就开始渲染，不随着 update 每帧更新，并且在渲染时，Unity 会通过静态批处理（Static Batching），将标记为静态的物体组成 Batch，在一次调用 GPU 渲染中批量处理多个物体，提高渲染效率。

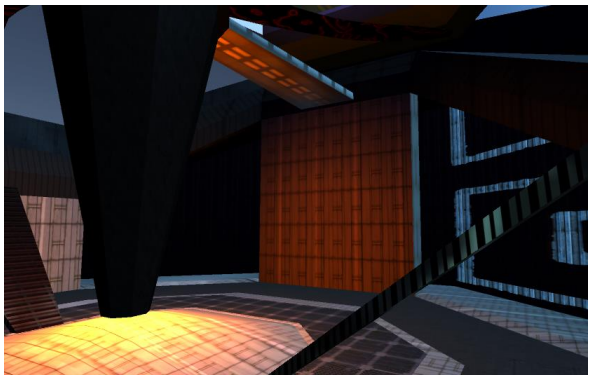


图 5-3 游戏主场景

5.2.2 游戏特效设计

在 3D 游戏中，游戏中出现的道具、场景、人物等等都是由网格构成，在 2D 游戏中则都是由图片构成，然而在游戏中，有一些物体无法用网格或者图片来构成，比如烟雾、云、火焰和魔法，这些物体一般被称为特效。

特性的实现需要用粒子系统来实现，粒子系统由大量的细小粒子构成，每个粒子拥有简单的纹理贴图，当每个粒子安照一定的轨迹移动，所有粒子构成的整体效果就是绚丽的特效。如图 5-4 中为飞船引擎喷出的火焰。

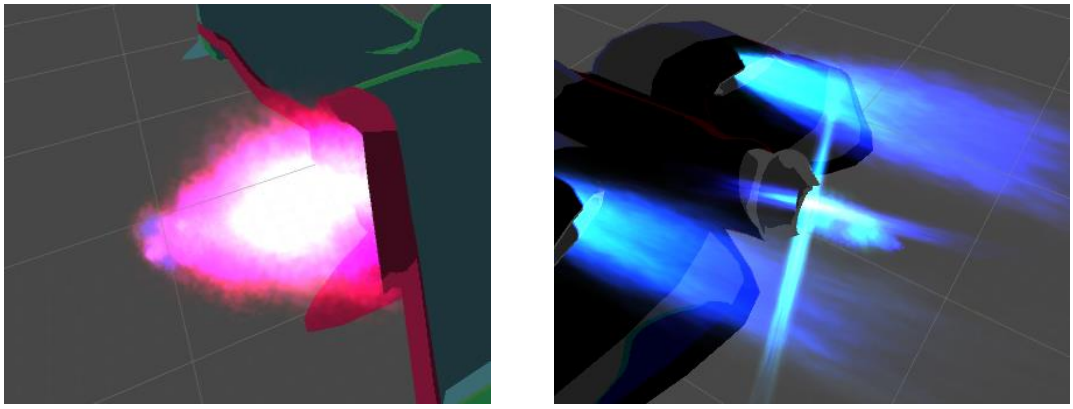


图 5-4 飞船引擎喷出火焰效果

5.2.3 界面设计

Unity 中制作 UI 可使用插件 NGUI 也可以使用 Unity 内置的 UGUI 系统， UGUI 是 Unity 官方根据 NGUI 设计的，虽然设计上 UGUI 参照了 NGUI 但是 UGUI 由于是官方出品，其能得到 Unity 更底层的支持，也相对于 NGUI 更方便编辑，因此本系统采用 UGUI 作为界面设计工具。

本系统一共有 6 个界面，分别是设置界面、开始菜单、房间信息界面、玩家信息界面、运行界面、结算界面。界面中的重要信息和界面跳转逻辑见图 5-5UI 设计逻辑图。

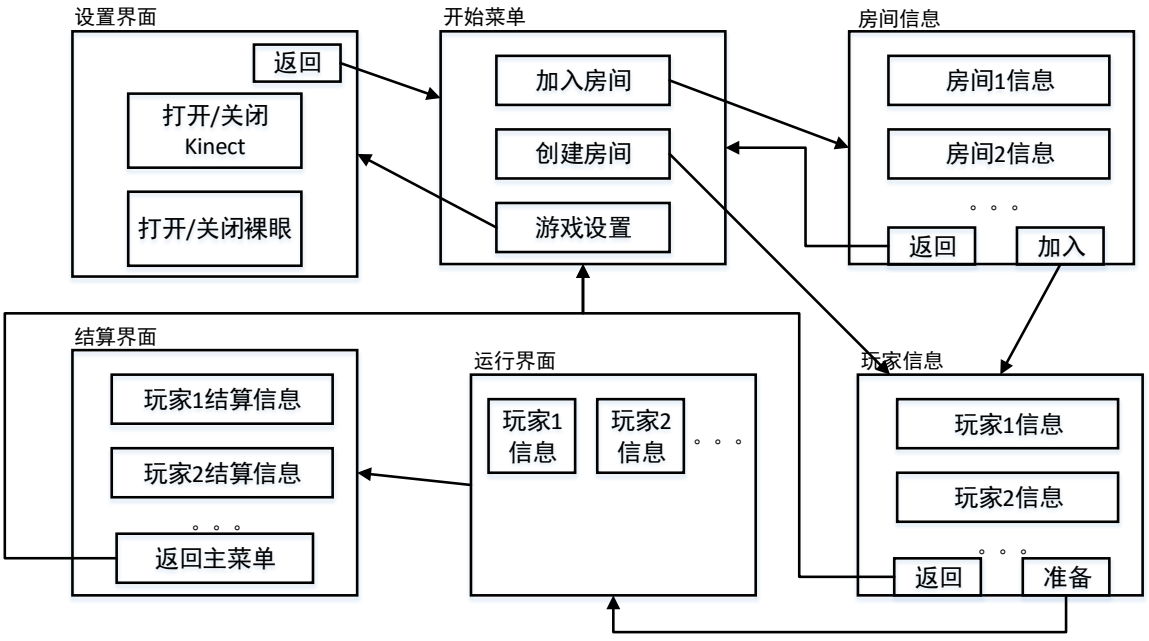


图 5-5 UI 设计逻辑图

游戏界面主要处理的逻辑可以分为两种：一种是用户操作如点击、拖拽的响应；另一种是当界面显示的信息发生变化，界面需要刷新显示。而在 Unity 中常用于界面逻辑的两种比较好的方法分别是事件订阅与消息响应，下面将分别介绍这两种方法：

（1）事件订阅与消息响应

由于游戏的界面中的逻辑很大一部分不需要每帧更新，例如判断鼠标是否按下这种情况，如果关心鼠标按下的界面元素都在每一帧中去判断鼠标是否按下将造成极大的性能浪费，正确的做法是以事件驱动，提取出单独的鼠标事件模块，模块负责监听鼠标是否按下，当鼠标按下那一帧调用关心鼠标按下的元素对应的处理函数。

在 C#中，可以通过委托(delegate)和事件(event)很便捷的事件订阅与响应系统，委托类似于函数指针，定义了响应函数的返回值类型和参数个数及类型。事件是对委托的封装，通过事件能方便的通过操作符“+=”订阅事件，“-=”操作符取消订阅。

2. 实现事件的响应函数。

（2） 协程实现 UI 刷新

对于界面上需要频繁刷新的情况，在 4.2.3 节介绍的协程正适合用来处理这种情况，如此不必将界面逻辑杂糅到 Update 函数中，能有效避免 Update 函数的臃肿杂乱问题。如图 5-6 为用于 UI 刷新的协程函数逻辑实现。

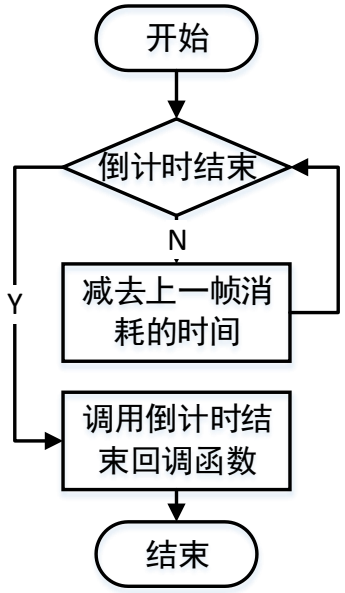


图 5-6 UI 刷新的协程函数

5.3 裸眼模块实现

此前在第二章已经对裸眼 3D 的原理进行了简单的原理描述，本节将实际实现裸眼 3D 合图算法。本系统使用的硬件为长虹的 48 英寸 4K 裸眼屏幕，具体参数见表 5-1。裸眼模块是将游戏画面合成裸眼屏幕能展示的图像，其中涉及到着色器（Shader）的编写，是整个系统的技术难点，其直接影响着最后在裸眼屏幕上的显示效果。

表5-1 长虹4K 裸眼屏幕参数

屏幕尺寸	48英寸	裸眼3D 技术	柱镜光栅
视点个数	8视点	屏幕比例	16:9
亮度	350cd/m²	对比度	1200:1
分辨率	1920*1080	3D 最佳可视角度	120°
3D 最佳观看距离	3-6m	规格	1124mm*663mm*73mm

5.3.1 八视点图像获取

系统使用的屏幕只支持 8 视点，所以游戏的画面获取采用了由 8 个摄像机的摄像机组，8 个子摄像机分别获取 8 个视点的所看到的图像。要达到更好的裸眼效果，需要让摄像机尽可能的模拟人两个眼睛间的关系，即 8 个子摄像机的相对位置和旋转，关键在于三点：子摄像机排列方式，子摄像机间隔，摄像机组焦距。子摄像机排列排列方式采用弧形排列，如图 5-7，使每个相机到焦点的距离相同，这样能有效避免两个相机间的画面断层问题。

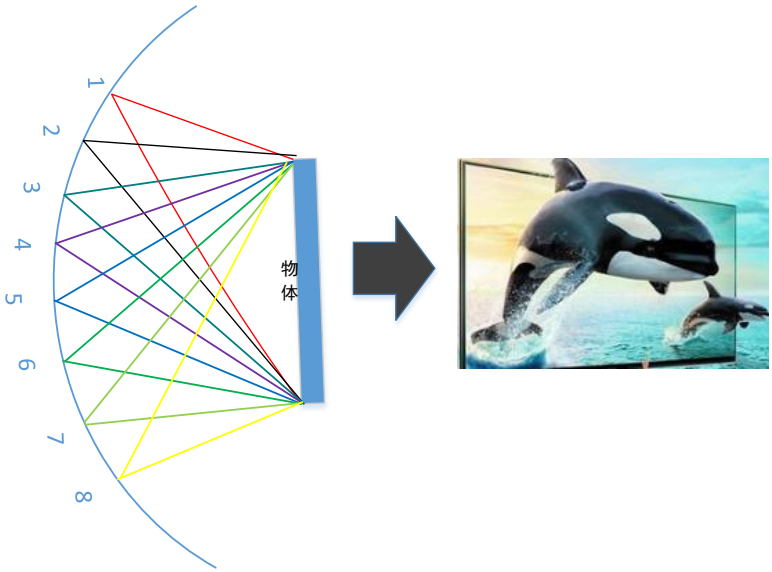


图5-7 相机排列示意图

摄像机的聚焦很好解决，Unity 的 Transform 组件提供了 LookAt 方法，可改变物体的旋转使之朝向传入的点的位置。为了方便子摄像机的排列，将子摄像机全部放入摄像机组物体下，如此，只用考虑其在父物体坐标系下的位置即可，同时在操作摄像机组做普通相机的跟随效果时，也不用考虑其 8 个子摄像机的位置。子摄像机间隔排列通过给定的焦距以及相机间隔确定。如图 5-8，为子摄像机排列方式的俯视图，1-8 为 8 个摄像机排列的位置，横轴为 Z 轴，纵轴为 X 轴，由于子摄像机都处于同一水平面上，所以可将所有子摄像机的 Y 值都设为 0，使其简化为二维。如公式 5.1，H 为焦距，delta 为子摄像机间弧长。可先求得两个摄像机间的夹角 θ，然后代入摄像机的序号 i，则可求得摄像机和 Z 轴的夹角 θ_i，将夹角 θ_i 代入式 5.1 中的最后两个子公式，即求出子摄像机的 X 坐标 X_i 和 Z 坐标 Z_i。如此就得到了 8 个子相机在父物体坐标系下的坐标，摄像机的排列问题就解决了。

$$\begin{cases} \theta = delta / H \\ \theta_i = 4.5 \times \theta - i \times \theta \\ x_i = \sin\theta_i \times H \\ z_i = \cos\theta_i \times H \end{cases} \tag{5.1}$$

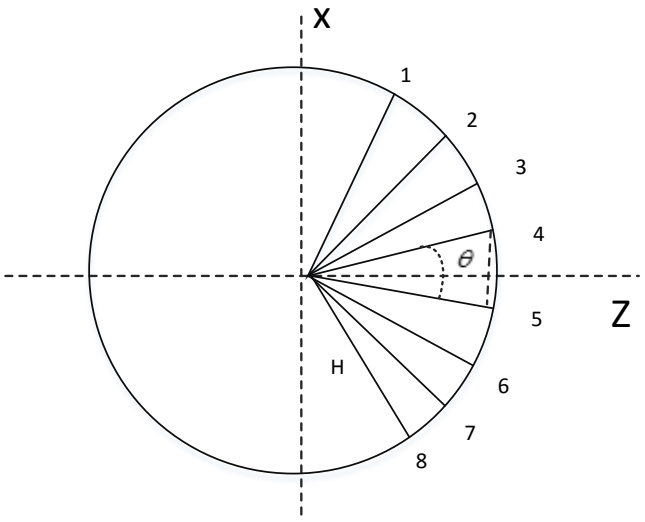


图5-8 相机排列俯视图

5.3.2 裸眼图像合成

通过上一节的相机组，系统获得了 8 张依次排列的画面，而裸眼屏幕和普通屏幕一样，只接受一张图显示，所以需要根据屏幕的特性，开发特定的算法，由于涉及到屏幕的详细硬件设计，属于商业机密，因此对原算法的优化难度相当大，其算法裸眼

呈现效果理想，所以系统只是优化了算法的一些效率问题，并没有改变核心算法。长虹提供的算法涉及到了屏幕上硬件的适配，算法细节不详细讨论，只介绍一下算法的简单流程图。如图 5-9，算法以 GPU 编程 Shader 实现，通过 Unity 传入的 UV 坐标获取到像素的实际坐标，由于裸眼屏幕像素的排列为倾斜，所以需要将 X 减去 cot*Y 的偏移，然后将坐标余 8 求得当前像素应该从哪张图上取。如图 5-10(a)为普通相机看到的方块，5-10(b)为通过 8 视点合成算法合成的图片。

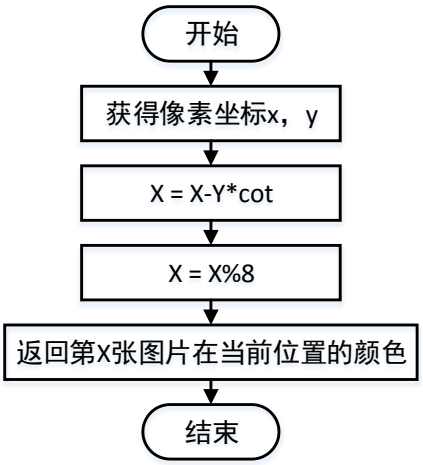
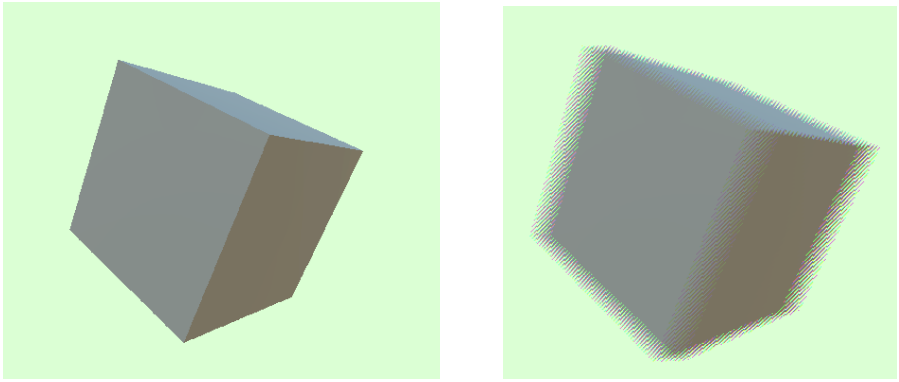


图5-9 裸眼图像合成算法简要流程图



（a） （b）

图5-10 裸眼图像合成算法效果图

5.4 网络通信系统

Unity 在 5.1 中推出了新的网络引擎模块 UNET，通过 UNET 不用单独开发服务器，可以很方便的创建，UNET 提供了一套网络开发 HLAPI（高级 API），通过 HLAPI 开发网络游戏无需关心底层网络细节。HLAPI 主要提供了以下功能：

（1） 消息处理程序

（2） 通用高性能序列化

（3） 分布式状态管理

（4） 状态同步

（5） 网络组件

5.4.1 UNET 网络模块

使用 UNET 开发的网络游戏有一个服务器和多个客户端。当没有专用的服务器时，客户端之一扮演服务器的角色，称之为此客户端 host（本地主机）。在 host 上，服务器和客户端在同一进程中，在 host 上的客户端 LocalClient（本地客户端），而其他客户端被称为 RemoteClients（远程客户端），其结构如图 5-11。LocalClient 与服务器通信通过直接的函数调用和消息队列，因为它是同一进程中。它实际上与服务器共享一个场景。RemoteClients 通过定期的网络连接与服务器进行通信。

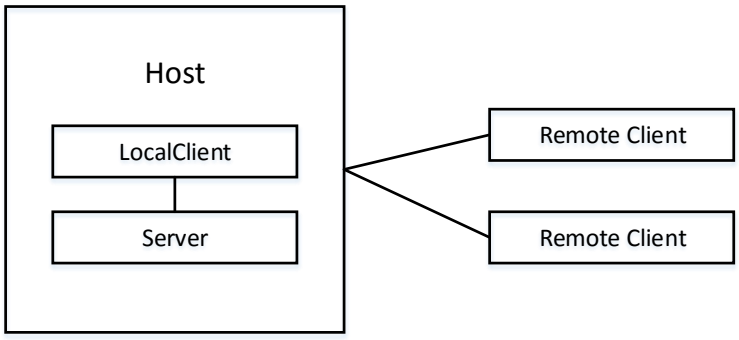


图5-11 UNET 服务端客户端结构

在单机游戏中，创建新的游戏物体直接在本地使用 GameObject.Instantiate 函数即可，但开发联机游戏，只能在服务器上创建一个游戏物体，然后游戏将被分布式状态管理模块同步到客户端。在网络系统中，玩家对象都是特殊的，每个人的 player 对象命令都会发送到该对象。一个人不能对另一个人的 player 对象调用命令。所以有“my player”对象的概念。当为一个客户端添加一个 player 时，这个 player 对象就成为该玩家客户端上的“local player” 对象。如图 5-12 显示两个客户端和他们 local players。属于本地客户端的 player 对象 被设置了 isLocalPlayer 标志。这标志可以用于控制输入的处理以及相机的跟随。除了 isLocalPlayer 标志，player 对象还有“local authority”标志。 改标志为 True 表示拥有客户端上的 player 对象的管理权，最常用到的是玩家

输入控制运动。对于非玩家对象如敌人，没有相关联的客户端，其控制权就在服务器上。NetworkBehaviour 上的属性 "isAuthority" 表示本地客户端是否有对象的控制权。

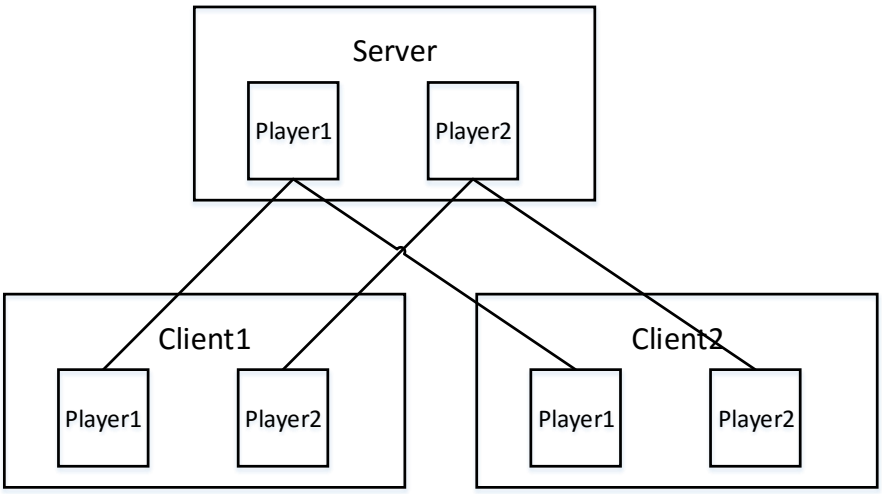


图5-12 UNET 物体同步

Unity 提供了两种方式使客户端和服务器直接交互，RPC（Remote Procedure Call 远程过程调用）方式，PRC 在服务器调用，在客户端执行。Commands（命令调用）方式在客户端调用，在服务器执行。联机游戏的总体流程如图 5-13 所示，首先客户端连入到服务器，服务器生成物体，服务器将物体同步到客户端，服务器和客户端通过 CMD 和 RPC 交互，客户端断开连接，服务器销毁游戏物体。

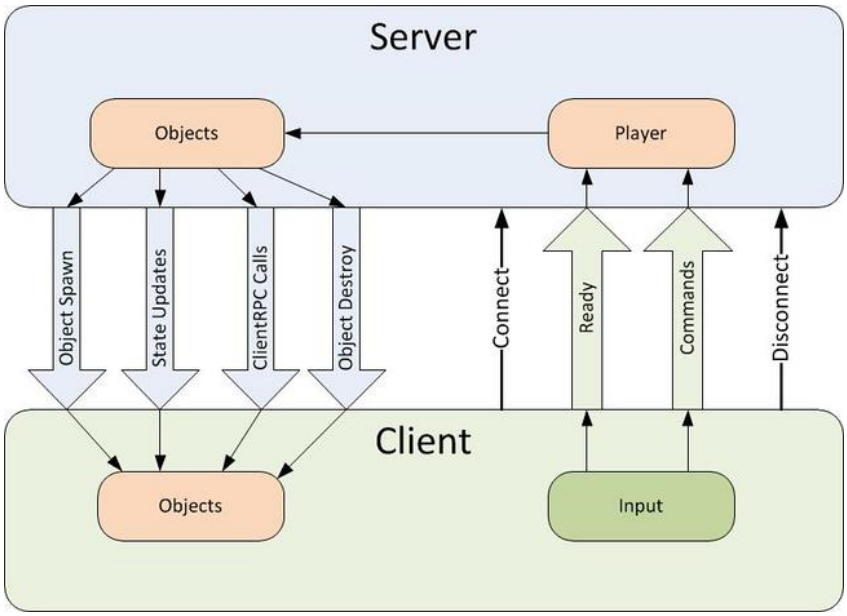


图5-13 客户端服务器交互

5.4.2 局域网广播

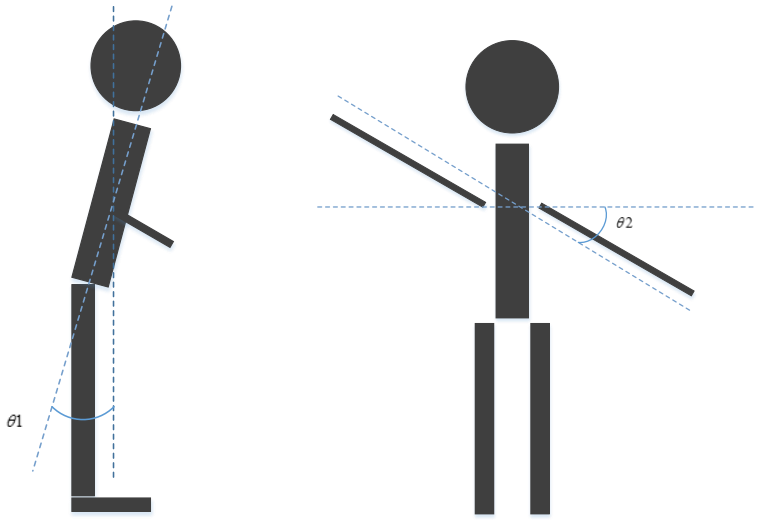
有了 UNET，可以很便捷的处理物体的状态同步，UNET 还有完善的消息机制和序列化机制。客户端和服务器的通讯也封装的很简洁。UNET 提供的局域网连接方式是客户端通过传入服务器的 IP 地址和端口，和服务器建立 TCP 连接，但是在局域网内，玩家可能并不知道服务器的 IP 和端口，针对这种问题，系统实现了基于 UDP 的局域网内广播系统，当玩家建立服务器后，在开始游戏前都会向整个局域网内广播服务器信息，开始游戏后停止发送信息。而当客户端打开服务器列表界面后，将会开启一个单独的线程监听广播端口，每收到一个信息就尝试解析服务器信息，解析成功则将信息加入到服务器链表中，当关闭服务器列表界面后，关闭线程并清空服务器链表。由于在 Unity 中并不支持在子线程中操作游戏物体，所以服务器列表界面管理类需要每帧不断遍历服务器链表并刷新到服务器列表界面上。

5.5 体感控制实现

体感控制模块是本系统的重难点模块之一，其识别速度与准确率直接关系到游戏的用户体验。在做了 4.4 节的开发前准备后，由于本系统只用到了骨骼点数据，只用调用微软提供的 KinectV2 开发包提供的获取骨骼位置函数就可以获得相应骨骼的位置。有了骨骼位置信息后，即可根据设计的体感姿势开发特定的识别算法。本节将从体感姿势设计和姿势识别两个部分阐述体感控制的实现过程。

5.5.1 体感姿势设计

如图 5-14(a)玩家通过身体前倾或者后仰控制飞船的俯仰角。如图 5-14(b)玩家通过展开双手与水平面保持一定的角度来控制飞船的左右旋转。左右任意一只手握拳控制飞船射击。



（a）　　（b）

图 5-14 控制飞船方向姿势

5.5.2 姿势识别

由于本系统的控制动作设计并不复杂，如上节说描述，主要包含五个动作：双手展开左/右倾斜、身体前/后倾斜、任意一只手握拳。其中握拳在 Kinect 开发包中已经提供，不用重新开发握拳算法，下面将介绍前 4 个动作的识别方式。

（1）双手展开左/右倾斜

首先需要判断双手是否展开，由于每两个人臂展都不相同，不能简单通过判断双手的空间距离是否大于一个固定的阈值判断，考虑到每个人的臂展和上半身的长度是有一定的比例，因此系统通过计算玩家上半身的长度再乘以一个比例作为阈值，双手空间距离改阈值则判断为双手展开。在双手展开状态下通过双手坐标即能计算出双手与水平面的夹角。如图 5-16 为倾斜角度的识别流程图，MIN 为最小识别角度，MAX 为最大识别角度，θ 为双手展开同水平面的夹角，V 为最终得出的左右倾斜度。为避免误操作，当θ 小于最小识别角度 MIN 时判断为无效输入，当角度大于最大识别角度 MAX 时，将角度设置为最大角度，最后通过最后一步中的公式将角度归一到 0 到 1 之间，最后得到的 V 就是一个 0 到 1 之间的数。

（2）身体前/后倾

同双手的倾斜算法相似，将双肩到臀部的向量同 Y 轴所成的夹角代入到图 5-15 的描述的算法中即可同样得到一个 0-1 之间的数。需要注意的是判断身体前/后倾时识别范围需要调整。

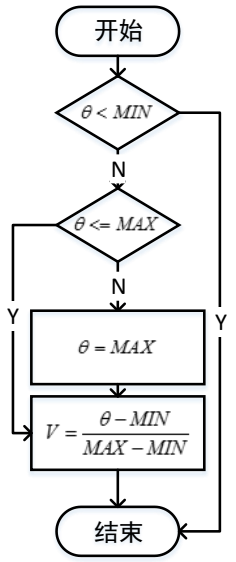


图5-15 倾斜角度计算流程图

5.6 本章小结

本章主要详细介绍了裸眼 3D 体感游戏的系统设计，首先介绍了游戏的整体架构，然后分场景、特效、界面三个方面阐述了系统中的美术设计，接着从裸眼图像获取和裸眼图像合成两个方面阐述了裸眼模块的实现，然后详细介绍了 Unity 的 UNET 网络系统，并且阐述了局域网广播的实现方式。最后分体感姿势设计和姿势识别两部分阐述了体感控制的实现过程。

第 6 章 裸眼 3D 体感游戏系统实现与结果分析

用来开发和测试本游戏的图形工作站详细配置与开发环境如下表所示。

表6-1 实验平台参数

序号	属性名称	参数
1	处理器	Inter(R) Core(TM) i7-4790M
2	CPU 主频	3.6GHz
3	显卡	NVIDIA GeForce GTX 970
4	内存	8GB
5	操作系统	Windows10（64位）
6	显示器分辨率	1920*1080
7	IDE	MonoDevelop5.9.6
8	Debug 工具	MonoDevelop5.9.6
9	3D 游戏引擎	Unity5.3
10	界面制作工具	UGUI
11	图片处理工具	Photoshop CS6
12	模型处理工具	3DMax 2014
13	开发语言	C#
14	体感摄像头	Kinect for Windows2.0
15	游戏帧数率	35FPS

本章从系统的几大模块来验证系统的实现情况，并给出了游戏运行过程中的截图。在本章最后对系统的实现情况做了分析和总结。

6.1 裸眼 3D 体感游戏系统实现

本系统的流程已经在 5.1 节中详细介绍了，本节就从整个游戏流程来验证游戏的完成情况。

系统在 Unity 上开发完成后，通过 Unity 直接打包出可执行文件。由于游戏支持体感和键盘两种输入方式，同样在画面输出方面，游戏也支持裸眼屏幕输出和普通屏幕输出两种方式，因此在运行游戏前用户可根据需要选择是否将 Kinect 和裸眼屏幕连接到电脑。启动可执行文件后，游戏默认全屏运行。打开游戏后，首先进入游戏的主菜单界面。如图 6-1 为游戏的开始界面。

图6-1 游戏主菜单界面

主菜单界面主要包括 4 个按钮，分别是：CREATE（创建服务器），JOIN（加入服务器），SETTING（设置），QUIT（退出）。当系统检测到 Kinect 时会自动启用 Kinect 控制，用户可通过左手或右手控制界面上的手掌标志，控制手掌悬浮在按钮上保持 3S 即完成了按钮的点击操作，如没有 Kinect 可使用鼠标直接点击。点击设置按钮跳转到如图 6-2 的游戏设置界面，可以根据是否插入 Kinect 和裸眼屏幕选择是否打开 Kinect 控制和是否打开裸眼显示模式。注意在此处如果没有插入 Kinect 设备，无法选择打开 Kinect 控制。设置信息会保存，下次打开游戏会以上一次的设置结果运行游戏。

图6-2 游戏设置界面

点击创建服务器，会进入玩家列表界面，如图 6-2 界面显示连入改服务器的玩家列表。在主界面菜单点击加入服务器按钮，页面会跳转到服务器列表界面，不断接受

局域网中的服务器广播消息如图 6-3 为局域网中的服务器列表，选中某一服务器点击 JOIN 跳转到图 6-2 的玩家列表，当服务器中的所有玩家都选则准备后，进入游戏场景。

图6-2 玩家列表

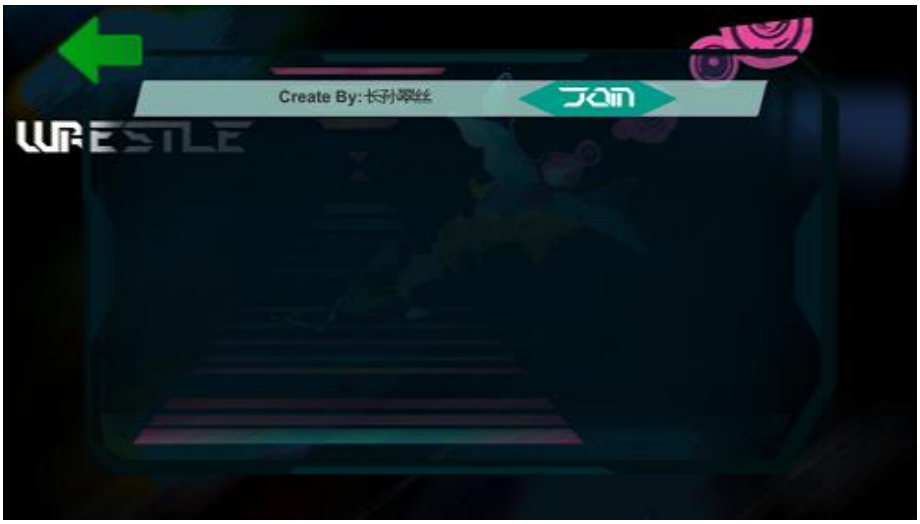


图6-3 服务器列表

在游戏主场景中，UI 界面主要显示玩家信息，包括玩家名、血量、击杀数和飞船外观，为了避免玩家间分不清敌我，如图 6-4 系统提供了四种飞船外形，玩家通过手势或者鼠标操作飞船发出激光炮，飞船每被激光炮打中一次减少一滴血，当血量为 0 时，玩家飞船损毁。如图 6-5 每当玩家击落一艘飞船，将会在玩家信息下加一个击杀图标。当玩家被击杀后，会显示如图 6-6 的倒计时画面，倒计时结束后玩家重生。当某一玩家击落 3 艘飞船后游戏结束，显示如图 6-7 的结算界面。结算信息包括击杀数，

死亡数，自己的信息将蓝色高亮显示，点击退出图标游戏退出到主界面。在游戏过程通过键盘或者手势可以打开系统菜单，有关闭/打开声音选项和退回到主菜单选项。

（a）　　（b）　　(c)　　(d)

图6-4 四种飞船外形

图6-5 游戏进行中画面

图6-6 重生倒计时画面

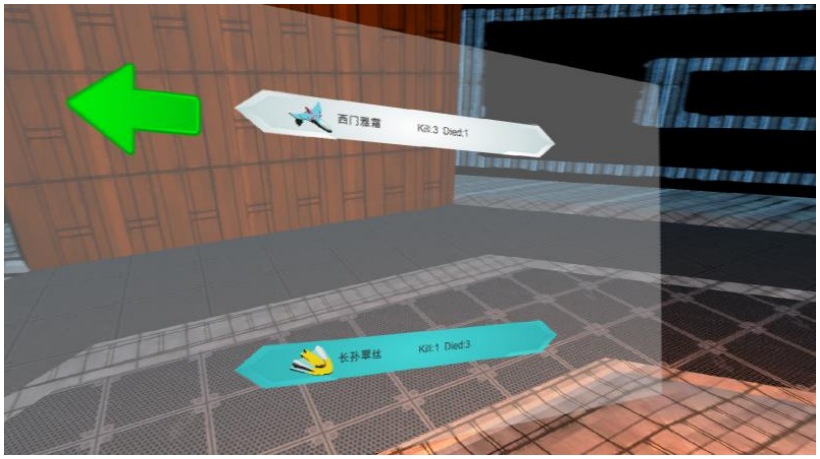


图6-7 结算界面

6.2 结果分析与总结

在上一节中，详细展示了游戏运行中的各个界面，验证了游戏的完成情况，在本节中将在电脑上实际运行，从游戏性能出发，对游戏进行分析与总结。如表 6-2，是根据 20 次游戏运行的 CPU 和内存占用采样统计。

表 6-2 系统性能数据

序号	CPU 占用(%)	内存占用(Mb)	序号	CPU 占用(%)	内存占用(Mb)
1	17	126	11	16	187.1
2	13	126	12	11	193.1
3	28	138.5	13	18	125.2
4	30	158.3	14	24	137.2
5	27	157.9	15	25	130.6
6	26	156.8	16	30	130.2
7	23	167.3	17	20	158.6
8	22	173.4	18	14	143.2
9	19	177.3	19	16	183.6
10	13	185.7	20	23	178.5

通过 20 次运行测试发现，在游戏启动阶段，游戏占用内存并不高，当游戏进入到游戏场景时，游戏会载入大量的模型和纹理资源，导致内存使用增加 30M 左右，除此之外游戏在运行过程中，内存占用稳定。在测试过程中并没有感觉到明显卡顿，游戏帧率可保持在 60FPS 左右。

通过分析发现，游戏有较高的稳定性，在负担 Kinect 与裸眼图像合成的情况下依然能达到 60FPS，游戏拥有不错的运行效率，但是游戏还是存在有以下问题：

1. 游戏内存优化还有提升空间。游戏工程中资源冗余量较高，一些无用的资源也没有剔除，造成游戏安装包达到惊人的 400M。

2. 没有实现单人玩法，只有对战玩法，玩法上有所欠缺；

3. 裸眼效果不理想，缺乏硬件知识，无法改进裸眼算法；

4. 对 C#语言和 Unity 引擎的掌握程度还不够熟练，在一些高级用法上显得力不从心。 整个游戏的代码结构设计也并不满意，缺乏游戏结构设计经验。

6.3 本章小结

本章通过游戏各个界面的截图详细展示了裸眼 3D 体感游戏的实际运行效果，安装游戏流程，依次展示了游戏开始菜单界面、设置界面、房间中玩家信息列表界面、局域网中服务器信息界面、游戏运行中玩家信息界面和游戏结束后的玩家游戏结算界面。最后通过测试游戏运行中对 CPU 和内存的占用分析了游戏的稳定性和执行效率，并对游戏的完成情况做了总结。

结论

随着计算机软硬件的不断发展，虚拟现实与人机交互行业不断的变革。裸眼 3D 技术不断发展，裸眼 3D 技术成为了影像行业最前沿的高新技术之一，它改变了传统平面图像带给人们的视觉疲惫感，以其生动的表现力，优美高雅的环境感染力以及强大的视觉冲击力感染了大众。

裸眼 3D 带来了更真实的沉浸式视觉体验，体感交互带来了一种全新的交互方式，为人们的娱乐方式提供了一种全新的理念。本系统将两者有机的结合起来，探索了裸眼 3D 游戏的可行性，实现了一款飞行射击裸眼 3D 体感游戏，玩家通过自然的人体姿势控制飞船运动和涉及，裸眼 3D 立体效果明显，为游戏玩家带来了沉浸式的游戏体验，体感丰富的肢体动作又使得用户能够充分体验游戏的互动性，并且体感的输入控制让玩家正好可以在裸眼 3D 最佳观看区域同游戏交互，弥补了裸眼 3D 效果必须在离屏幕一定距离的位置观看的缺点，两者结合起来相得益彰。

本文首先研究了裸眼 3D 和体感输入两个技术要点，然后根据游戏涉及到的游戏开发知识进行了较为深入的讨论和分析，对于游戏设计中主要包含的场景设计、游戏美术设计、角色控制部分、Kinect 接入 Unity 的方法、裸眼图片合成的方法进行了较为细致的讨论，并举例阐述了当前游戏美术中重要的设计制作方法。网络游戏并不是在单击游戏上简单的修改即可完成，需要处理很多在单击中根本不会遇到的问题，在网络模块的设计中，在阅读完 Unity 提供的网络模块官方文档后，由于网络模块推出时间不久，文档并不全面，根本不知道如何下手。在开发过程中遇到很多问题，只能去阅读源代码。在磕磕碰碰中，终究功夫不负有心人，完成了整个游戏的设计。

总的来说，将裸眼 3D 展示技术应用于体感游戏，能够给玩家带来新奇的游戏体验。同时在国内还没有游戏厂商尝试推出裸眼 3D 游戏，裸眼 3D 技术和体感技术也在不断的发展，裸眼显示效果和体感识别效果会不断提高。因此裸眼 3D 体感游戏拥有广阔的市场前景。

致谢

时光荏苒之间，回首来看，四年匆匆而过，就要告别母校西科大了，四年来学到了很多，大一泡图书馆学会了如何自己学习，管理个人时间，大二开始泡实验室，不断寻找自己的方向，寻找过程中不断补充基础知识。在大二下学期，开始了游戏开发学习，专业技能不断提升，更重要的是通过不断的遇到问题解决问题，学习到了解决问题的方法，感谢西科大教会了我这么多。

感谢一路上给予指引的老师们。其中首先感谢是指导老师吴亚东老师和张晓蓉老师。吴老师和张老师在专业上严谨认真、敬业负责的态度一度让我感动，让自己不断向他们看齐，从他们身上学到了太多太多。

感谢将作者领进虚拟现实实验室的陈永辉老师，使得我从大二开始就有机会接触 Unity 游戏开发，还要感谢虚拟现实实验室，给了这么好一个平台，让作者学以致用。感谢实验室的师兄师姐们，和大家在一起学习，奋斗真的很开心。

最后，最为感激的还是父母。四年的外地读书已经让他们为我操心不少，即将离开校园，为了更好的机遇去了离家更远的北京也让父母更加牵挂。这几年每次回家停留的日子都不多 ，但是幸好有一个幸福完美的家庭，有身体尚且健康的父母。感激他们一直以来的支持和鼓励。暂时走向远方是为了风雨兼程的努力之后、明天一个更强大更独立的人回报父母对我的养育之爱，为了我爱和爱我的你们，作者会努力让自己越来越强大，让你们越来越骄傲。

参考文献

[1] LAWTON.3D displays without glasses: coming to a screen near you[J]. computer,2011,44(1):17-19

[2] 夏勇峰. Kinect 与人机交互的未来[J]. 商业价值, 2011(2):60-63.

[3] XunBo Yu. Autostereoscopic three-dimensional display with high dense views and the narrow structure pitch [J]. Chinese optics letters ,2014(6):30-33.

[4] 雷雯. 基于体感交互中间件的人体互动框架的设计与实现[D].北京：北京交通大学，2012.1.

[5] 胡颖群. 基于 Kinect 体感识别技术的研究与实现[J]. 甘肃科技纵横,2013(7):18-20.

[6] 王树斌. 浅析 Unity3d 开发游戏流程及常用技术[J]. 电脑知识与技术,2012(2):351-352.

[7] 吴志达. 一个基于 Unity3d 游戏引擎的体感游戏研究与实现[D]. 广州：中山大学，2012.

[8] 郭冠军. 裸眼3D 视频转换技术研究[D]. 电子科技大学, 2013.

[9] Jianli Luo. GPU-Based Multi-view Rendering for Spatial-Multiplex Autostereoscopic displays[C]. IEEE International Conference on Computer Science and Information Technology.2010:28-32

[10] 刘立强. 基于体感交互的裸眼3D 互动展示设计与实现[D]. 北京工业大学, 2014.

[11] 刘东泽. 基于柱镜光栅的仿真立体图像生成方法[J]. 印刷杂志,2012(11)：23-25.

[12] 侯春萍,许国,沈丽丽.基于狭缝光栅的自由立体显示器视区模型与计算仿真[J]. 天津大学学报,2012,45(8):677-681.

[13] 姚剑敏,辛琦,郭太良. 自由立体显示器中锯齿状交错狭缝光栅的设计[J]. 光子学报,2012,41(10):1176-1179.

[14] LIU R. Study on Binocular Stereo Imaging Based on Computer Stereo Vision[D]. Dissertation of Chongqing University,2007

[15] WANG S X. Research on the Realization Method of Nakedness-Eye Stereoscopic Display Based on Single Chip DMD[J]. Chinese Journal of Electron Devices, 2008(1):16-18.

[16] GUO H,QING K,MAO M,etl. Real-Time Tiled Multi-projector Autostereoscopic Display Algorithm for Dual-view 3D Video Files [J]. Journal of Computer-Aided Design & Computer Graphics,2015,27(9): 1734-1742.

[17] ZHOU L,TAO Y,WANG Q,etl. Design of Lenticular Lens in Autostereoscopic Display[J]. Acta Photonica Sinica,2009,38(1):30-33

[18] SHI D,KANG X,LI S,etl. Multi-Views 3D Display Technology[J]. Electronic Science & Technology,2014,01(03):276-282

[19] Huang J, Wang Y. 30-view projection 3D display[J]. Proceedings of SPIE - The International Society for Optical Engineering, 2015, 9385:93850B-93850B-7.

[20] 王宏安, 戴国忠. 自然人机交互技术[J]. 中国图象图形学报, 15(7):980-983.

[21] 黄石. 论游戏设计的基本原则[J].装饰, 2007, (6):34-36.

[22] 王树军.三维游戏引擎中物理引擎关键技术的研究[D].天津：天津大学，2007.

[23] 张帆. Unity3D 游戏开发基础[M]. 杭州:浙江工商大学出版社，2013：34-35.

[24] 宣雨松. Unity3D 游戏开发[M]. 成都:四川电子音像出版中心，2012：15-18.

[25] 刘炜伟，张引，叶修梓. 3D 游戏引擎渲染内核架构及其技术[J]. 计算机应用研究，2006，（8）：45-48.

[26] 徐宇峰，刘秀珍，王革. 3D游戏引擎构架及游戏动画渲染技术[J]. 多媒体技术及其应用，2008，（7）：1324-1328.

[27] 王超.3D 游戏引擎场景渲染技术的研究与实现[D].武汉：武汉理工大学，2006.