南阳理工学院

本科生毕业设计(论文)

学院(系)： 软件学院

专 业： 软件工程

学 生： 宗权利

指导教师： 惠征

完成日期 2020 年 05 月

南阳理工学院本科生毕业设计（论文）

基于Unity3D的马里奥冒险游戏的设计与实现

Design and implementation of Mario adventure game

based on unity3d

总计：毕业设计(论文) 29页

表格：11个

图片：13个

南阳理工学院本科毕业设计(论文)

基于Unity3D的马里奥冒险游戏的

设计与实现

Design and implementation of Mario adventure game

based on unity3d

学 院(系)： 软件学院

专 业： 软件工程

学生姓名： 宗权利

学 号： 1615925241

指导教师(职称)： 惠征 讲师

评阅教师： 段云涛

完成日期： 2020年05月05日

南阳理工学院

Nanyang Institute of Technology

基于Unity3D的马里奥冒险游戏的

设计与实现

软件工程宗权利

1. 随着科技的不断扩大和生活质量的不断提高，游戏种类也逐渐的丰富了起来，人们对游戏的需求也逐渐的扩大。在如今出现的游戏开发引擎中，Unity3D是一款比较成熟的游戏开发引擎，此款引擎主要用于3D游戏的开发与制作。Unity3D引擎具有比较完善的UI系统、物理系统和声音系统等.本次课题也是利用Unity3D引擎做出的一款3D游戏。本课题主要研究的是利用Unity3D引擎开发一款3D版本的马里奥冒险游戏。通过EasyTouch组件实现玩家的操作杆，实现玩家的摇杆操作。在游戏中，人物的运动效果通过Mecanim动画系统实现，利用状态机实现切换。游戏界面使用的UI是通过UGUI组件使用的。游戏中数据的读取利用的是json的解析实现。本次课题不仅仅有学习和研究的意义，同时也可以给游戏开发行业带来一种老游戏2D转3D的思路，从而带来更多有趣的游戏。
2. 游戏开发；Unity3D；UI；物理系统；EasyTouch

Design and implementation of Mario adventure game

based on unity3d

Software Engineering Major Zong Quanli

1. With the continuous expansion of science and technology and the continuous improvement of quality of life, the types of games are gradually enriched, and people's demand for games is gradually expanding. In today's game development engine, unity3d is a relatively mature game development engine, which is mainly used for the development and production of 3D games. Unity3d engine has a relatively complete UI system, physical system and sound system. This project is also a 3D game made by using unity3d engine. The main research of this project is to develop a 3D version of Mario adventure game with unity3d engine. Through Easytouch components to achieve the player's joystick, to achieve the player's rocker operation. In the game, the motion effect of the characters is realized by mecanim animation system, and the state machine is used to switch. The UI used in the game interface is used through ugui components. The reading of data in the game uses the parsing implementation of JSON. This project not only has the significance of learning and research, but also can bring a kind of thinking of old game 2D to 3D to the game development industry, so as to bring more interesting games.
2. Game development; Unity3d; UI; physical system; Easytouch

目 录

1 绪论 1

1.1 课题研究背景与意义 1

1.2 国内外现状分析 2

1.3 课题主要内容 2

1.4 论文主题结构 3

1.5本章小结 3

2 相关技术 3

2.1 Unity 3

2.2 C#语言 4

2.3 QFramework 框架 4

2.4 响应式编程 5

2.5 UGUI 5

2.6 DOTween 5

3 需求描述与分析 6

3.1 游戏需求分析 6

3.2 游戏用例分析 6

3.3 非功能性分析 7

3.3.1 稳定性与安全性 7

3.3.2 兼容性 7

3.4 可行性分析 7

3.4.1 技术可行性性分析 7

3.4.2 操作性分析 8

3.4.3 经济性分析 8

3.5 本章小结 8

4 游戏设计 9

4.1 游戏资源设计 9

4.2 游戏面板设计 9

4.3 功能设计 10

4.3.1 进入游戏 10

4.3.2 选择角色 10

4.3.3 选择关卡 10

4.3.4 游戏关卡 10

4.3.5 退出游戏 11

4.3.6 返回大厅 11

4.4 游戏数据设计 11

4.4.1 角色数据 11

4.4.2 关卡数据 11

4.5 本章小结 12

5 游戏实现 12

5.1 主界面功能实现 12

5.1.1 角色选择功能实现 12

5.1.2 关卡选择功能实现 16

5.2 游戏场景功能实现 18

5.2.1 数据管理系统实现 18

5.2.2 角色控制系统 19

5.2.3 场景生成系统实现 20

5.2.4 金币管理系统实现 21

5.2.5 事件系统 22

6 游戏测试 23

6.1 游戏测试概述 23

6.2 游戏测试内容 23

6.3 游戏测试用例 23

6.3.1 游戏主界面模块测试用例 23

6.3.2 游戏局内模块测试 25

6.4 游戏测试总结 26

结束语 27

参考文献 28

致谢 29

# 绪论

3D版本的马里奥游戏是在原本经典的2D马里奥游戏的基础上进行优化和创新的一款3D的游戏。此款游戏利用的是Unity3D引擎开发的，之所以选择此款游戏的开发，主要是让大家能在如今可以发达的背景下能够继续体验以前老版本马里奥游戏的乐趣。如今的游戏虽然是五花八门的，但是却又是千篇一律的，不管是玩法还是剧情，都有很多相似点，此款游戏不仅仅采用的2D马里奥游戏的模式，同时也在3D模式下有一定的创新，不仅仅让玩家能回顾以前的马里奥游戏，同时也能体验现在3D游戏的风格和新的玩法。从研究方面上讲，对游戏行业来说，如果这次研发效果良好，可以让更多古老的2D游戏以3D的方式重现。对个人来说，更能深入研究游戏开发个Unity3D技术。

## 课题研究背景与意义

3D游戏的设计与开发将是现在以及未来的流行趋势[1]。3D版本的马里奥游戏是新游戏开发的另一种创新，在3D游戏技术还未出现的时候，全球就出现了大量的2D的游戏，虽然当时并没有3D技术，也没有发达的游戏引擎，但但是也出现大量有趣的游戏，当时游戏并没有像现在这样游戏拥有复杂的玩法，都是一些简单的操作，比如贪吃蛇、坦克大战和俄罗斯方块等一些古老游戏，虽然操作简单，但是游戏中依旧有着奇幻的乐趣。在这些古老的2D游戏中，充满了前辈们的智慧。如今的游戏行业越来越发达，新游戏也层出不穷，但是，新奇玩法或者新奇模式的游戏并不多，大部分都是换皮游戏，因为游戏行业的突飞猛进，我们很难再像以前一样新类型游戏层出不穷了，但是我们可以去深究古老游戏在当前行业前景下的新奇开发。

马里奥游戏是一款在古老2D游戏中比较突出的游戏，类似的马里奥、魂斗罗和冒险岛等游戏，虽然模式都类似，但是拥有大量的玩家。我们可以利用当前发达的3D游戏开发技术，让这些曾经的老玩家再次回温一下古老的马里奥游戏，和普通的马里奥游戏不一样的是，这次不仅仅是2D从3D的转变，同时增加了3D的元素，和3D的玩法，任务不再是单一的左右和上下移动，障碍和敌人都是都时3D元素了。这样玩家更能体验3D的乐趣，同时，3D物体都是从2D元素3D化转变而来。同时，我也增加了选择关卡和切换角色等一些当前游戏比较流行的元素。

次游戏主要是解决当前游戏模式停滞不前，同时市场上也出现了大量的换皮游戏，而大部分主机游戏都是还是停留在外国的游戏机上面，而PC端大部分都是移植或者模仿的，都没有游戏的灵魂。所以，在游戏紧缺的背景下，出现一款3D版本的马里奥游戏，肯定会深得一些马里奥老玩家的喜爱。

Unity3D是当前游戏开发行业中普遍使用的3D游戏开发引擎，在如今存在的游戏引擎中，Unity3D引擎是一款相对成熟的游戏开发引擎。这款游戏引擎是由丹麦的Unity公司开发的一款游戏开发工具，此款游戏开发引擎最大的特点就是可以跨平台发布，并且具有出现色的物理、网络、着色器、UI、粒子系统等。Unity3D引擎和其他游戏引擎比较，历史悠久，拥有大量的用户同时也有大量的用户在维护此款引擎，此款引擎拥有大量的第三方插件库，让新用户更加容易的快速入门。

通过利用Unity3D引擎去开发这款3D版本的马里奥游戏，不仅能快速的开发出来，同时也能是游戏更加的出色，Unity3D引擎的出色是游戏开发者有目共睹的，我相信此款引擎一定能使此款游戏做到更好。

## 国内外现状分析

第一款马里奥游戏是FC《超级马里奥兄弟》，是在80年代出现的，而3D游戏则是90年代初才开始出现，但是当时3D技术并不是很发达。虽然期间出现了大量以马里奥为主题的游戏，但是在流行的所有游戏中并没有3D版本的游戏。

随着3D技术的逐渐发展，并且游戏机也逐渐强大了起来，逐渐也出现了很多款3D版本的马里奥游戏，其中比较突出的便是任天堂的马里奥系列游戏，任天堂是日本的一家公司，马里奥系列游戏在日本还是比较发达的，3D系列的也不少，但是大部分都是游戏机上面的游戏，任天堂的也都是Switch游戏机上的游戏，国内的马里奥游戏就没有出名的，所以如果现在国内出现一款比较好的3D版本马里奥游戏还是很好的。

在游戏开发行业中，还是国外游戏比国内发达的，虽然现在有很多游戏开发引擎，但是比较好的游戏都是那几款出名的游戏开发引擎开发的，而其他游戏开发引擎大部分都是开发小游戏使用的。至今流行的大型游戏中，大部分都是采用Unity3D引擎进行开发的，Unity3D引擎是进几年才在中国流行的，但是在国外早已流行了起来。

以前的游戏开发出来后，因为设备的限制，开发出来游戏后，想再其他设备上玩的时候还需要有专业的人士进行移植，但是Unity3D引擎完美的解决了这个问题，使用Unity3D引擎开发后，可以向当前比较流行的设备进行打包，只需要一次性开发，后面只要打不通的包就可以了，这样就不需要再考虑针对某个单独的设备进行开发了。我们应该多向国外游戏开发者们学习，不仅仅学习他们的游戏引擎，同时他们的游戏开发思路也是值得我们研究的。

我们开发的游戏不应该再局限于国内的市场，我们应该多学习外国的游戏开发思路，这样我们就可以扩宽外国的游戏市场。

## 1.3 课题主要内容

主页功能:玩家进入主界面后会显示当前人物角色，此角色是玩家上次登录的时候玩家选择的角色，如果是第一次玩，则是默认的第一个角色，玩家可以在主界面左边菜单选择角色，同时在右边选择需要进入的关卡。

游戏内功能:在游戏内玩家可以通过界面按钮操作人物进行移动，场景内会存在障碍、迷宫、敌人等，玩家需要通过操作躲避这些障碍进入最终的城堡来完成关卡。

## 1.4 论文主题结构

论文针对该系统的具体的内容，研究的过程，功能性，非功能性，需求分析等一系列设计，共分为以下六个部分：

第一部分主要介绍当前研究的课题内容，课题背景、课题现状以及组架构

第二部分主要介绍当前课题需要用到的相关技术

第三部分主要介绍当前课题的需求分析和功能模块

第四部分主要介绍了当前课题的整个游戏架构的设计以及缓存数据的设计

第五部分主要介绍对整个游戏的设计模型的总结以及功能的分析。介绍了游戏的操作逻辑、数据储存与加载的设计以及界面UI的层级框架设计。

第六部分主要描述游戏的功能测试，这一部分主要是对游戏所设计出来的功能进行测试，主要包含两大部分，一部分是对游戏主页面的UI、人物角色和关卡选择的测试；还有是对游戏内角色玩法的测试。分别从测试概念、测试目标、测试方法和测试内容四个方面进行分析测试。

## 1.5本章小结

本章主要介绍的是该课题研究的背景和意义，以及对国内外在该课题党项的现状和未来趋势进行了深入的剖析。同时也介绍了该课题所研究的主要内容和框架，并且也研究了该课题的可实施性以及未来的发展趋势。对整个论文主题的结构进行了介绍。

# 相关技术

## Unity

Unity是一款比较成熟的游戏开发引擎，也是当今全球比较流行的游戏开发引擎。游戏引擎一直被喻为是游戏的发动机[2]。Unity引擎是由丹麦的一家名叫Unity的公司发的用来游戏开发的工具，它诞生于丹麦的阿姆斯特丹。此款引擎正式发布与2015年，当时只是为了开发游戏方便，就在Mac端作为了扩展工具进行发布，因为此款引擎的优秀，没过多久就出现了windows的版本了。

一次开发即可轻松部署到Windows、Mac、iOS、Android、Wii、PS3等平台，告别以往高难度的、耗时的跨平台游戏开发，使快速的、高质量的游戏开发成为可能[3]。其中，最突出的一点就是跨平台发布，不仅可以让大型公司更方便的发布于不同的平台，同时也是独立游戏这的福音，独立游戏开发者只需要学习一套游戏开发的技术就可以在大部分平台发布自己的游戏。程序员的三大浪漫是编译原理、操作系统和图形学[4]。Unity引擎是一款出色的3D游戏引擎，虽然也可以开发2D游戏，但是该引擎最出色的依旧是在3D方面，其中的物理引擎尤为出色，能直接有3D的视觉效果。同时在美术方面，提供了柔和阴影与烘焙等光影渲染系统和强大的可视化编辑器[5]。其中内置大量的Shader完全可以满足开发者，开发者可以自己写Shader脚本来实现更精彩的3D世界，使开发者更容易的开发3D游戏。

在游戏脚本方面，Unity支持了三种脚本，分别是JavaScript、C#和Boo，其中的JavaScript和Boo很少被开发者使用，开发者使用最多的还是C#，因此，引擎在C#脚本方面提供了大量的扩展，使开发正更容易进行开发。

在文档方面，Unity官方提供了大量的官方资源和案例，使开发者可以快速的入门和开发，同时在网络上也出现了大量的教程和API详解，因此Unity快速成为了一款轻松入门、快速开发的引擎。

## C#语言

C#语言是由C/C++衍生出来的一种面向对象的解释性语言，C#是一种现代的、通用的、面向对象的语言。所有这些编程技术还有一个缺点是它们主要针对桌面程序而不是Internet进行开发[6]，C#语言是有微软(Microsoft)开发的。C#语言拥有面向对象、面向组件、结构化语言等特点，同时它也是.Net框架的一部分。

C#语言虽然是由C/C++衍生出来的，但是它却像Java一样有许多强大的编程功能，所以很多程序员还是十分喜欢这款语言的。C#语言有索引器、组件版本、属性和事件、委托和事件管理、简单的多线程、自动垃圾回收等重要特性。多线程的优点：可以同时完成多个任务[7]。

用C#语言开发出来的程序的源代码并不是编译成二进制代码，而是编译成一种中间代码，接着就是通过.NETFramework🡪CLR(通用语言运行库)🡪执行。C#语言并不难，也非常适用于学习基本编程技术[8]。

## QFramework 框架

QFramework框架的作者是曾经知名游戏公司的主程凉鞋，此款框架是在2015年开始进行开发的，框架刚出现的时候并没有特别出名，但是通过一系列的宣传和使用者的推荐，此款框架逐渐火了起来。

QFramework发布至今已经到2020版本，QFramework的版本命名是根据优化年份命名的，QFramework框架主要包含六大模块：Core、ResKit、UIKit、ScriptCollection、

ShaderLib、EditorToolKit。QFramework框架最突出的特点就是包含了大量的模块，上到UI面板管理，资源管理、下到Shader、Lua等细节之处的优化。

QFramework是参照GameFramework、DBFramework等当前比较火的框架进行综合优化的。QFramework框架的底层是MV\*的结构。设计模式（Design Patterns）是各种软件设计技巧的呈现方式，善用它们，更能表现出游戏设计工程师优雅的一面[9]。不仅优化了内存消耗，而且使用户更加容易的去使用此款框架，很清晰的能梳理其中结构。不仅此款框架有这种结构，用户在使用此款框架的绑定结构去开发游戏的使用，能很简单的形成MVP结构，研究比较深的玩家能在MV\*结构中灵活运用。

## 响应式编程

响应式编程并不是一种工具，也不是一种技术，而是一种编程的思想方式，而本课题研究的项目在部分模块中采用的就是这种思想实现的。响应式编程和面向对象编程有异曲同工之妙。响应式编程是把所有的东西都当做数据流，然后数据流之间通过相互观察联系了起来。

响应式编程的核心点在于设计模式中的观察者模式。如果说，数学是思维的体操，那设计模式，就是面向对象编程思维的体操[10]。在以往的面向对象的编程中，是通过对象之间包含或者被包含的关系进行组合，或者是通过事件进行不同对象之间的联系。而响应式则是通过一种序列将观察者和被观察者联系起来。

本课题实现这种思想所采用的技术是UniRx,这是Unity扩展的一种库，是根据微软所提供的Rx.Net所扩展出来的，完全展现了响应式编程的思想。用户使用UniRx能在Unity上很轻松的就实现响应式编程，从而增加代码的简洁和可读性。

## UGUI

UIGI是Unity引擎提供的一种UI控件，负责Unity开发的项目中UI的管理。Unity常用的UI控件分别是UGUI和NGUI，因为UGUI控件是Unity引擎开发公司的团队进行开发的，所以相对于UGUI拥有更好的兼容性。

UGUI包含大量常用的UI控件元素，可以使用户能够更加方便的开发游戏中的UI元素。UGUI在2014年优化后，UGUI的性能有了显著的提升。UGUI在开发的时候不需要再像NGUI一样，后期需要重新调整图集，在使用UGUI的时候，系统会自动整理图集，同时也提供了接口，方便人为的进行控制。同时UGUI的Mesh的合并是在C++里面进行的，C++的性能是比较顶尖的。UGUI也拥有多线程的支持。但是UIGU也有界面打开比较慢、UI动画会卡顿等缺点。

## DOTween

DOTween是Unity引擎游戏开发常用的一款快速高效并且类型比较安全的面向对象的补间动画引擎。DOTween性能非常的快，同时也很高效。这是因为DOTween中的一切都是被缓存并且重复使用的，这样就避免了一些无用的GC的分配。

DOTween的时间是计算方式是非常精确的，API的旨意也在于提高效率、直观并且易用。DOTween还有一个比较强大的特性就是游戏中的一切皆可以动画，程序中的一些数值或者其他属性都可以利用DOTween做成一些动画，DOTween还可以为字符串设置一些动画并且支持富文本。

DOTween在控制动画的时候可以控制动画的播放、倒带、结束、暂停、重启等一些完全控制的操作。DOTween在播放动画的时候可以随时更改补间动画的开始或者结束的时间，也可控制持续的时间。DOTween可以将补间动画合并的序列中，也可以分组，所以它可以创建一些复杂的动画。DOTween可以在补间动画的过程中使用协程进行完成、杀死、启动等操作。

DOTween是一个非常强大的库，这个库基本上满足的各种动画的需求，同时在性能上也是比较优秀的。同时它也对Unity中一些原生的API进行看扩展，所以非常适合Unity开发者使用。

# 需求描述与分析

## 游戏需求分析

本课题所研究的是一款利用Unity3D引擎制作所实现的3D版本的马里奥游戏，该

课题的脚本所使用的C#脚本语言，游戏中的元素都是模仿2D版本的马里奥游戏中的一些元素，这些元素都进行了3D化，使整个游戏有了3D的效果。游戏的玩法同时也模拟了2D版本马里奥游戏中的玩法——逐个关卡进行冒险的形式。

游戏关卡内，玩家通过界面左边的摇杆来操作游戏角色的前后左右的移动，玩家可以通过点击界面的右边的按钮来操作角色进行跳跃。玩家通过两个操作杆实现了3D世界的角色操作。

游戏主界面，玩家可以通过主界面左边的面板进行角色的选择，然后玩家可以通过主界面右边的面板来进行关卡的选择。

## 游戏用例分析

该游戏主要是玩家独立一个人进行游戏，因此该游戏所面对的对象只有玩家个人。玩家的用例图如图3-1所示。

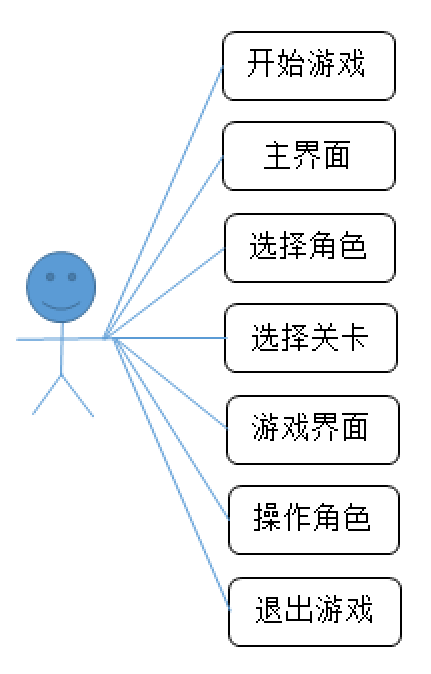


图 3-1 玩家需求用例图

玩家通过安装客户端并点击客户端进入游戏，首先进入的是游戏的主界面，在主界面中，玩家可以选择退出游戏从而结束游戏。玩家还可以通过点击触发角色选择面板，在面板里进行角色的切换，玩家还可以通过点击触发关卡选择面板，然后玩家通过选择关卡可以进入游戏界面。玩家在游戏界面可以通过摇杆和按钮的操作来控制人物通过关卡，从而通关次关卡。

## 非功能性分析

### 稳定性与安全性

游戏的稳定性与安全性是评价一款游戏的好与坏的重要因素。因为游戏对玩家的操作开放，玩家可以不按照游戏设计的流程进行通关，从而破坏游戏的公平性，因此开发者在开发游戏的时候要考虑一些非正常的操作，在设计游戏的时候，需要考虑其他操作的可能性，从而进行阻拦，防止玩家进行非正常操作。游戏数据的加密也是非常重要的一步，文件未加密不仅会导致游戏被抄袭，还有可能出现大量游戏外挂从而破坏游戏，Unity引擎不仅能自动对文件进行加密，也能是开发者很容易开放操作，从而发现游戏的漏洞进行修补。

### 兼容性

游戏所发布的平台越多，游戏所吸引的玩家数量也就越多。如图3-4所示，Unity的多平台发布特性就显得尤为重要，开发者不需要考虑开发的游戏能否在其他的平台进行兼容，开发者只需要发布不同平台的安装包，用户只需要根据自己的平台去下载对应平台的安装包即可。玩家遍可以在自己拥有的平台进行兼容。

## 可行性分析

### 技术可行性性分析

本游戏是利用Unity3D游戏开发引擎进行开发的，因为Unity3D引擎自带3D系统、物理系统和UI系统，所以开发者只需要使用对应的C#脚本便可以进行开发，Unity3D提供了许多C#语言的接口，开发者不需要考虑引擎与脚本的交互，通过对应的接口便可以控制引擎中的元素。游戏中的UI元素则是采用Unity3D引擎自带的UGUI库，因为是引擎公司自己提供的库，因此同样有大量的接口提供。游戏中的模型则是可以从网络平台中找到，因为Unity3D引擎拥有大量的用户，所以在网络中也有大量的游戏模型进行提供，因此很容易便可以找到自己需要的模型或者UI。

游戏中的缓存数据则是采用Unity3D引擎自己提供的PlayerPrefs进行数据的保存于读取，比如游戏的关卡和选择的角色，均可用PlayerPrefs进行保存。游戏中的一些配置数据则是可以采用json进行保存，开发者可以先进行配表设置，然后通过表格转json的方式进行转换，同时C#语言中也提供了Json读取的API。游戏中的关内数据或者人物模型的数据便可采用这种方式进行配置。

游戏中的UI面板的控制则可以采用QFramework框架所提供的绑定的方式进行处理，开发者只需要开发过程中进行配置，便可通过QFramework框架中提供的API进行面板的控制。游戏中的相互通信则是可以根据C#的委托写一套事件系统，也可以使用UniRx进项MVC结构控制，从而达到对象之间的通信。

Unity的一键打包可以轻松的打包成安装包，然后玩家可以通过安装打出来的安装包直接安装到对应的平台，便可以直接打开进行游戏。

### 操作性分析

该游戏主要是在移动端进行发布，玩家安装安装包后可以直接在手机上打开，玩家只需要通过手的点击便可进行游戏中的操作。

此游戏界面比较简单，玩家一目了然，很容易知道每个按钮的作用。游戏的摇杆或者按钮都是模仿其他3D游戏的，游戏中也只有两个按钮，所以玩家省去了很多复杂的操作，因为摇杆的强大，在游戏界面内，玩家通过手指的滑动进行游戏内的摇杆操作，配合按钮的跳跃便可以很容易的操作角色。

玩家不需要其他设备便可以操作3D角色，玩家的视角也是第三人物视角，玩家可以很容易观察到角色的正面视野，玩家的视觉很舒服，玩家还可以很容易观察到周围的场景，操作角色的时候也不影响环境的观察。

### 经济性分析

Unity3D引擎存在免费版本，只要不进行发布就可以免费试用，所以在游戏研究方向是可以随便使用的。Unity3D游戏引擎之所以获得成功的一大原因，就是得益于对无力承担游戏引擎高额许可费用的独立开发者的支持[11]。游戏开发中大量的库都是官方或者使用者自己发布的免费的库，同时游戏所使用的QFramework框架也是一款开源的免费框架。在此次研究中，除了一些3D和2D资源方面需要少量金钱，大部分都是免费提供的。同时打包的安装包只要不是发布，玩家可以自己安装进行游戏，因此只需要消耗少量的流量即可。

## 本章小结

本章主要是对游戏进行需求分析，首先对游戏进行了简单的介绍，介绍了游戏的操作界面和主界面，接着对游戏进行了用例分析，分析游戏主要的功能的用例。接着进行了非功能性分析，通过稳定性与安全性、兼容性两个方面进行了分析。最后通过技术可行性、操作性和经济性三个方面对游戏的可行性进行了分析。

# 游戏设计

## 游戏资源设计

每款游戏均需要游戏管理系统协助[12]。游戏资源设计是游戏开发最初的设计，首先要对整个游戏中所用到的游戏资源进行一个规划，这样才能更容易的规划资源的用途。如图4-1所示，对游戏所用到的资源归类到对应的文件夹，首先规划资源的大类别，2D、3D、脚本、插件和库都要一一分类、然后再在每个大类中进行小类的细分，这样我们在接下来设计游戏的时候，可以根据资源进行游戏的整个架构设计，游戏的资源决定游戏的整个玩法。我们先把2D资源规划出来，从而决定了游戏中的界面的结构趋势，然后通过游戏的3D资源决定了游戏中玩家的操作。接着就是脚本和库的管理，从而决定了游戏的框架。

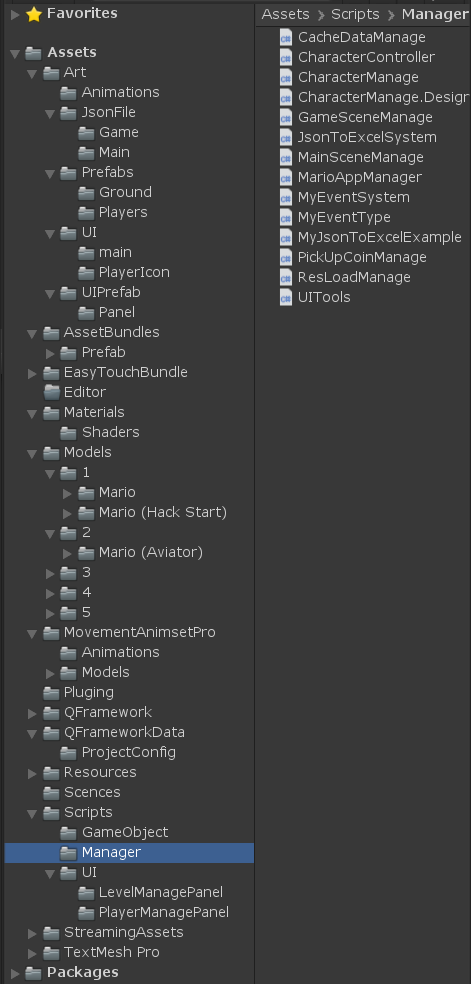


图 4-1 游戏文件组织设计图

## 游戏面板设计

游戏中界面的结构就是如图4-2所示，面板的管理比较简单，因为是3D游戏，主要突出的是3D，所以面板比较少，游戏中只需要一个面板的管理对象，分别对角色选择面板、关卡选择面板和游戏操作面板管理，然后游戏中只需要通过面板管理对象进行三个面板的调用和切换，通过MVC的结构对三个面板的数据和界面分离。面板管理只需要对面板数据处理，就可以控制面板。

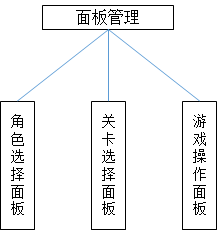


图 4-2 游戏面板框架设计图

## 功能设计

程序设计能力是软件工程专业学生最基本的专业能力[13]。游戏的乐趣取决于游戏中的功能，有的的功能越丰富，玩家能体会到的乐趣也就越多。

### 进入游戏

运行安装的客户端经过的是Unity3D引擎自带的一个进入游戏的动画，然后会进入游戏的主界面，这中间的流程只要使用Unity进行打包即可。

### 选择角色

进入主界面后，主界面的左边会有一个按钮，点击按钮就会滑动出来一个选择角色的面板，第一次进入游戏的人是默认的一个初始的角色，玩家可以通过点击角色选择面板中的角色头像来实现主界面中3D角色的切换，同时切换的角色数据也会保留在缓存中，下次玩家再进入游戏后，主界面显示的角色就是上次玩家最后一次选择的角色。角色选择面板中的UI和对应的人物数据都是保留在json的配置文件中的。同时开发者可以通过配置表格生成对应的json文件来实现角色选择的数据，配置对应的id和头像图片名字与模型名字即可。然后通过数据读取来实现。

### 选择关卡

进入主界面后，主界面的右边有一个按钮，点击按钮会弹出一个选择关卡的面板，在面板中，已通过的关卡或者即将要通过的最新关卡显示的是蓝色，其他的关卡则是灰色。只有蓝色的关卡可以点击并且进入关卡，灰色的则是点击无效。同时玩家通过的最新关卡是可以保留在缓存中，这样下次玩家再打开游戏的时候就会保留以前通过的关卡数，就可以玩自己未通关的关卡了。

### 游戏关卡

玩家通过选择关卡进入关卡后，展现在玩家眼前的就是另一个场景了，此场景就游戏场景，在此场景中，玩家页面会显示两个UI，一个是摇杆一个是按钮，玩家通过摇杆操作玩家的左右前后移动，通过按钮点击控制玩家的跳跃。玩家页面还会显示一些3D元素和玩家选择的角色，玩家控制角色通过障碍到最后的城堡就是通过，通关后会返回主界面，玩家就会解锁最新的关卡。

### 退出游戏

玩家在游戏主界面可以点击移动端的返回按钮，就会弹出是否退出游戏的面板，玩家选择是就会退出游戏，选择否就会关掉面板，返回到游戏的主界面。

### 返回大厅

玩家有游戏场景中，点击移动端的返回按钮就会弹出是否返回大厅的面板，玩家选择是就会返回主界面，点击否就会返回游戏场景，玩家可以继续游戏。

## 游戏数据设计

游戏中的元素均可以配置成数据，配置成数据后，开发者可以通关修改数据来实现游戏中的表现，开发者不把数据写死就可以很方便的控制自己游戏中的数据。但并不是所有的元素都配置成数据就很好，当数据过多或者过于复杂的时候，数据元素就会很容易导致游戏出错，而且开发者也不容易出现在配置数据。所以游戏数据的设计也是非常重要的。

### 角色数据

游戏中的角色数据就如表4-1所示，表格中的playerid代表的是角色的唯一id；playericonname代表的是角色头像对应的图片的名字；playerprefabname代表的是角色对应的模型所生成的prafab文件的名字。此表格会通过工具转换成json配置文件，游戏中通过读取json文件生成角色选择面板和角色选择中的出现的模型。开发者只需要在此数据增加数据，游戏中不需要再更改其他脚本就可以直接出现新增的角色和角色的模型。

表4-1角色信息表

|  |  |  |
| --- | --- | --- |
| playerid | playericoname | playerprefabname |
| 2 | PlayerIcon\_2 | Player\_2 |
| 3 | PlayerIcon\_3 | Player\_3 |
| 4 | PlayerIcon\_4 | Player\_4 |
| 5 | PlayerIcon\_5 | Player\_5 |

### 关卡数据

游戏中的关卡数据如表4-2所示，itemid代表的是游戏关卡的唯一id，level代表对应的关卡数，scencename代表的并不是关卡场景，代表的是关卡内数据对应的配置文件名字。关卡内的场景是无数个小场景拼接起来的，如表4-3所示，id代表的是依次排列的顺序，通过此表格可以配置对应关卡的场景，达到场景的数据控制。

表4-2关卡数据

|  |  |  |
| --- | --- | --- |
| itemid | level | scenename |
| 1 | 1 | GameScene\_1 |
| 2 | 2 | GameScene\_2 |
| 3 | 3 | GameScene\_3 |
| 4 | 4 | GameScene\_4 |
| 5 | 5 | GameScene\_5 |
| 6 | 6 | GameScene\_6 |
| 7 | 7 | GameScene\_7 |
| 8 | 8 | GameScene\_8 |
| 9 | 9 | GameScene\_9 |

表4-3关卡内数据

|  |  |
| --- | --- |
| id | prefabname |
| 1 | ground\_start |
| 2 | ground\_1 |
| 3 | ground\_2 |
| 4 | ground\_end |

## 本章小结

本章主要分析游戏中的资源、面板、功能和数据的设计，介绍了游戏中的重要元素和元素的设计理念，和设计的方式，让开发者可以快速的了解整个游戏的各个模块的框架和配置信息，从而使自己的脚本更加简洁并且开发更加的高效。

# 游戏实现

## 主界面功能实现

### 角色选择功能实现

游戏中角色选择界面如图5-1所示，玩家通过点击左边面板中的角色头像选择切换角色，中间的模型会根据选择的角色切换对应的模型。



图5-1角色选择效果图

游戏中脚本的结构如图5-2所示，PlayerData脚本为游戏中角色数据类，用来保存从json数据中读取出的角色信息。

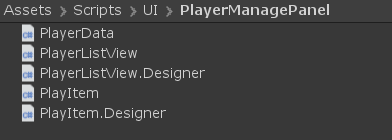


图5-2 角色选择脚本结构

代码如下所示：

namespace QFramework.MFO

{

/// playerItem数据

public class PlayerData

{

[SerializeField] public int playerid;

[SerializeField] public string playericoname;

[SerializeField] public string playerprefabname;

}

/// 从json读取所有的playerItem到链表里

public class PlayerDataList

{

public ReactiveCollection<PlayerData> mPlayerDataList = new ReactiveCollection<PlayerData>();

/// 从json文件读取到链表里

public void InitLoadJson()

{

}

}}

PlayerListView脚本与PlayerListView.Designer脚本是通过QFramework框架中的ViewController脚本进行绑定生成出来的脚本。如图5-3所示，通过此脚本还可以实现Prefab的生成，从而达到快捷高效。

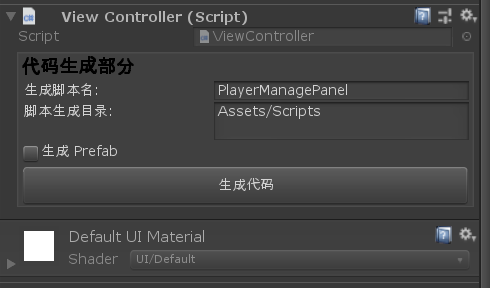


图5-3 QFramework框架绑定图

QFramework中的ViewController脚本实现代码结构如下：

public class ViewController : MonoBehaviour

{

[HideInInspector] public string ScriptName;

[HideInInspector]

public string ScriptsFolder = "Assets/Scripts";

[HideInInspector]

public bool GeneratePrefab = false;

[HideInInspector]

public string PrefabFolder = "Assets/Prefabs";

}

QFramework中绑定脚本所生成的效果代码如下所示：

public partial class PlayerListView : UIElement

{

private string playerName;

//通过data数据和playitem物体绑定

public Dictionary<PlayerData,PlayItem> mPlayItemsView = new Dictionary<PlayerData, PlayItem>();

/// 初始化playeritem显示的数据

/// </summary>

public void InitPlayerItem(PlayItem mPlayItem, PlayerDataList mModel)

{

}

public void OnShow()

{

mPlayItemsView.ForEach(data =>

{

});

}

/// 增加一条显示的playeritem

public void AddPlayItemView(PlayItem mPlayItem, PlayerData mData) => mPlayItem.Instantiate() //实例一个对象

.Show();

//选中的item发生的改变的时候触发的事件

public void SelectPlayer(PlayerData date)

{

}

}

框架生成的.Designer脚本只是类的一个扩展，用来框架识别和标注的，代码如下：

public partial class PlayerListView

{

public void Clear()

{

}

public override string ComponentName

{

get { return "PlayerListView";}

}

}

游戏中角色选择是根据MVC框架写的，QFremework框架的绑定正是规定了这种结构，角色选择面板也就需要面板中的组件作为View的控制，QFramework同样也做了约束，面板中的组件可以采用Bind脚本进行绑定。如图5-4所示：

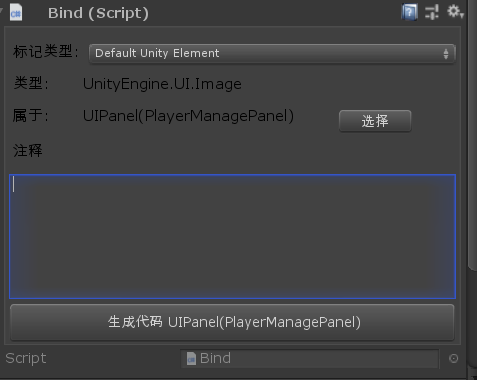


图5-4 QFramework组件绑定图

角色选择脚本的结构中，PlayItem和PlayItem.Designer就是对应的组件的控制脚本，在组件绑定后，重新生成面板的绑定的时候就会自动绑定面板所有子物体的组件，角色选择数据的组件脚本如下所示：

public partial class PlayItem : UIElement

{

private PlayerData mData;

private Image iconBg;

public Toggle mToggle;

public ReactiveProperty<PlayerData> selectPlayer = new ReactiveProperty<PlayerData>();

//需要注入uitools

[Inject] public IUITools mUiTools { get; set; }

private void Awake()

{

mData = new PlayerData();

iconBg = GetComponent<Image>();

mToggle = GetComponent<Toggle>();

}

protected override void OnBeforeDestroy()

{

}

/// 初始化item的数据并且初始化显示

public void OnInitData(PlayerData data)

{

}}

当组件下面同时有组件进行绑定的时候，组件的脚本就会同样绑定其子物体的组件信息，这样就可以很好的控制事件的传递，再配合上UniRx库，就可以以简洁的代码实现MVC结构。

### 关卡选择功能实现

游戏中关卡选择的效果图如图5-5所示。



图5-5 关卡选择效果图

关卡选择的脚本结构如图5-6所示。同角色选择脚本一样，是利用QFramework框架的绑定脚本生成的。



图5-6 关卡选择脚本结构

LevelData与LevelItem的结构与角色选择的Data和Item类似的，因为面板和组件类似，所以View都相似，所以在Mode和Control上有控制的区别，在Item脚本中，控制场景切换的代码如下所示：

public void OnInitData(LevelData mleveldata)

{

mLevelData = mleveldata;

LevelText.text = mleveldata.level.ToString();

//注册点击关卡按钮的事件

GetComponent<Button>().onClick.AddListener(() =>

{

//Debug.LogError("切换场景");

UIMgr.CloseAllPanel();

CacheDataManage.Instance.SetIntData(DataType.playLevel,mleveldata.level);

SceneManager.LoadSceneAsync("Game");

});

}

关卡的面板控制脚本如下所示：

public partial class LevelListView : UIElement

{

private int maxLevel;

//通过data数据和levelitem物体绑定

public Dictionary<LevelData,LevelItem> mLevelItems = new Dictionary<LevelData, LevelItem>();

//需要注入uitools

[Inject] public IUITools mUiTools { get; set; }

private void Awake()

{

//注入UITools

MarioAppManager.Container.Inject(this);

}

/// 初始化levelitem显示所有数据

public void InitLevelItem(LevelItem mLevelItem, LevelDataList mLevelDataList)

{

}

public void AddLevelItem(LevelItem mLevelItem,LevelData mLevelData)

{

}

}

## 游戏场景功能实现

游戏中管理和控制的脚本结构如图5-7所示：

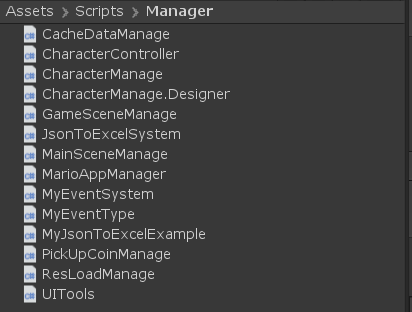


图5-7 游戏管理脚本结构

### 数据管理系统实现

游戏中的角色关卡都是采用PlayerPrefs进行数据的读取和写入的，虽然Unity对PlayerPrefs封装的很好，但是并没有针对游戏中的数据进项管理。因为不同的游戏中有不同的数据，因此需要自己制作适应自己游戏的数据管理系统。游戏中调用和写入数据的地方都是通过自己封装的管理系统进行实现。数据管理系统需要先把游戏中的数据进行枚举，然后数据系统提供接口根据不同的枚举进行数据的获取与写入。数据管理系统枚举代码如下所示：

public enum DataType

{

//玩家选择的角色

playerName,

//玩家通过的最高等级

levelMax,

//玩家正在玩的关卡

playLevel

}

数据管理系统的代码实现如下所示：

public class CacheDataManage : Singleton<CacheDataManage>

{

public void RescoveyCacheData()

public void SetIntData(DataType type,int data)

public int GetIntData(DataType type)

public void SetStringData(DataType type, string data)

public string GetStringData(DataType type)

private CacheDataManage()

}

### 角色控制系统

游戏中角色的移动和跳跃的控制采用的是Unity自带的动画系统，Unity提供了接口但是并没有提供控制系统，游戏中的控制系统都是需要自己创建的。游戏中的控制UI采用的是EasyTouch插件，如图5-8所示：

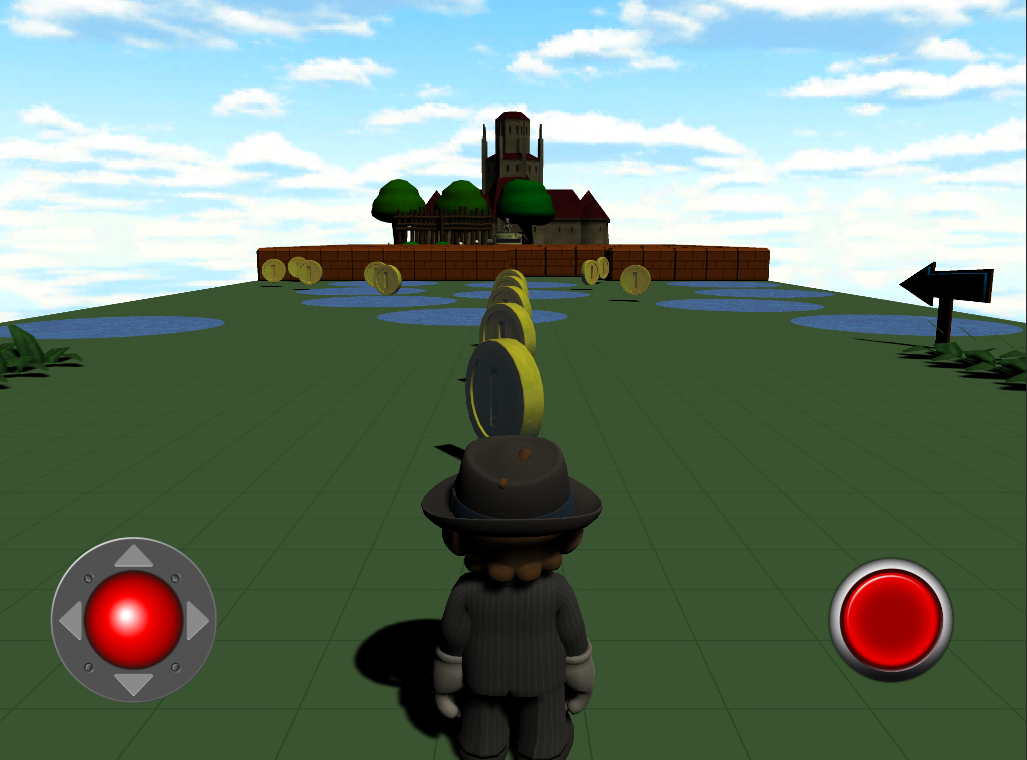


图5-8游戏操作界面

游戏中操作的控制是通过监听摇杆的旋转来控制人物的转向，然后通过摇杆拉开的距离控制人物的移动，游戏中角色的跳跃是通过监听EasyTouch的Button按下来控制角色动画的切换。游戏中的动画控制器如图5-9所示。

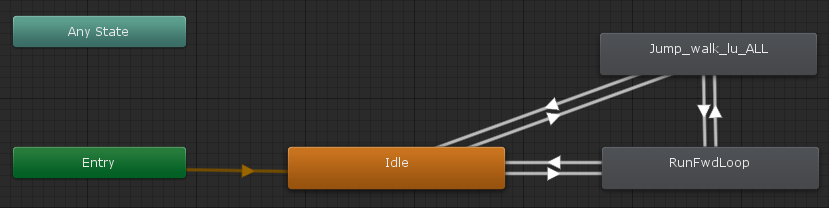


图5-9角色控制器

角色控制系统的代码如下所示：

void Start()

{

moveAnim = GetComponent<Animator>();

//Debug.LogError(moveAnim);

myJoystick = GameObject.Find("EasyTouchControlsCanvas/TestJoystick")

.GetComponent<ETCJoystick>();

JumpButton = GameObject.Find("EasyTouchControlsCanvas/JumpButton")

.GetComponent<ETCButton>();

//监听控制器值的改变

myJoystick.onMove.AddListener(PlayerMove);

myJoystick.onMoveEnd.AddListener(PlayerStop);

myJoystick.onMoveStart.AddListener(PlayerStart);

JumpButton.onPressed.AddListener(PlayerJumpStart);

JumpButton.onUp.AddListener(PlayerJumpStop);

gameObject.OnTriggerEnterAsObservable()

}

角色控制器动画的切换代码如下所示：

/// 角色开始运动

private void PlayerStart()

{

moveAnim.SetBool("Run", true);

}

/// 角色停止移动

private void PlayerStop()

{

moveAnim.SetBool("Run", false);

}

/// 角色开始运动

private void PlayerJumpStart()

{

moveAnim.SetBool("Jump", true);

}

/// 角色停止移动

private void PlayerJumpStop()

{

moveAnim.SetBool("Jump", false);

}

/// 角色移动

private void PlayerMove(Vector2 arg0)

{

}

### 场景生成系统实现

游戏中的场景是采用动态控制的方式生成的，不是关卡设置好的，开发者可以根据测试调整关卡的场景。场景是利用多个小场景根据顺序生成的，场景生成的配置文件是读取Json文件，Json文件的配置是利用Excel转换成Json的。游戏中的场景Json配置文件如下所示：

[ {

"id": 1,

"prefabname": "ground\_start"

},

{

"id": 2,

"prefabname": "ground\_1"

},

{

"id": 3,

"prefabname": "ground\_2"

},

{

"id": 4,

"prefabname": "ground\_3"

},

{

"id": 5,

"prefabname": "ground\_end"

}

]}

场景生成系统是根据Json文件读取到数据中，然后依次读取数据中的prefabname属性加载出来对应的小场景组件，然后通过id计算位置放在对应的场景中。场景生成系统的代码如下所示：

/// <summary>

/// 初始化加载局内场景的地面

/// </summary>

/// <param name="level"></param>

public void LoadSceneGround(int level)

{

//清空数据，方便数据初始化

groundDateList.Clear();

hasGroundCount = 0;

//读取需要加载的场景的记录数据json

var jsonText = ResLoadManage.Instance.mResLoader

.LoadSync<TextAsset>("gamescene\_" + level).text;

groundDateList = jsonText.FromJson<List<GameGroundData>>();

//遍历初始化加载读取出来的数据场景

groundDateList.ForEach(item =>

{

});

}

### 金币管理系统实现

游戏中的金币是比较多的，一次性生成所有的金币会造成资源的消耗，所以游戏中的金币的生成采用的是对象池进行管理，有单独的一套金币生成的系统。游戏中的金币动画是利用Unity制作的，如图5-10所示

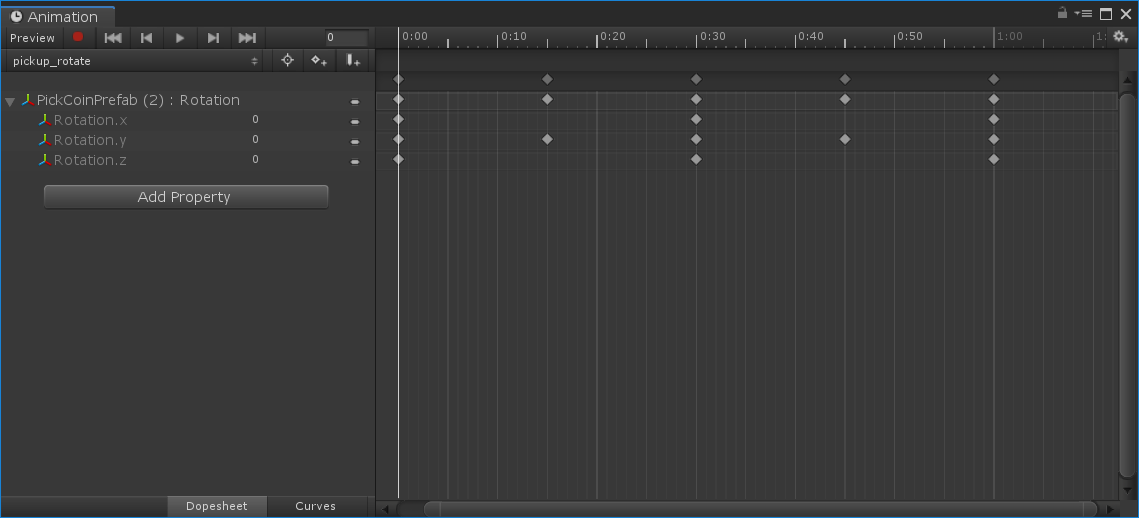


图5-10金币动画

游戏中金币对象池是采用先生成一部分金币，玩家吃金币利用的是碰撞检测，当场内金币较少的时候会利用被吃掉的金币重新生成新的位置。如果一开始金币并没有吃掉，会根据Prefab生成新的金币，如果玩家金币一直不吃，就会同时存在大量的金币。碰撞检测最基本的是模拟现实环境中物体在遇到障碍时发生的本能反应[14]。金币生成系统代码如下所示：

public class PickUpCoinManage : MonoSingleton<PickUpCoinManage>

{

public ResLoader mResLoader;

public Sequence mSequence;

public List<Transform> mAllocTransforms;

public Queue<Vector3> mCoinPosList;

public override void OnSingletonInit()

{

mResLoader = ResLoader.Allocate();

mSequence = DOTween.Sequence();

mAllocTransforms = new List<Transform>();

mCoinPosList = new Queue<Vector3>();

...

}

public void InitPickCoin(int level)

{

}

public void RecyCoin(Transform tF)

{

}

public void AddCoin(Transform coinTf)

{

}

}

金币在场景中出现的位置也是根据配置的Json文件生成的。配置的Json文件如下所示：

[

{

"x": 4,

"y": 0,

"z": 0

},

{

"x": 8,

"y": 0,

"z": 0

},

{

"x": 12,

"y": 0,

"z": 0

},

...

]

### 事件系统

游戏中对象之间的通信是必不可少的，事件系统是游戏中常见的系统。Unity游戏中通常使用的消息事件是直接只用委托实现的[15]。本游戏中的事件系统采用的是UniRx和委托进行处理的，UniRx负责监听对象，利用委托和枚举保存对应的事件，通过UniRx监听调用对应的委托形成事件系统。事件系统代码如下所示：

/// <summary>

/// 接口 只负责存储在字典中

/// </summary>

interface IRegisterations

{

}

/// 多个注册

class Registerations : IRegisterations

{

/// 委托本身就可以一对多注册

public Subject<object[]> Subject = new Subject<object[]>();

}

public class MyEventSystem

{

/// 存储已经注册的事件

private static Dictionary<MyEventType, IRegisterations> mTypeEventDict

= new Dictionary<MyEventType, IRegisterations>();

/// 注册事件

public static Subject<object[]> GetEvent(MyEventType type)

{

}

/// 发送事件

public static void Send(MyEventType type, params object[] param)

{

}

}

# 游戏测试

## 游戏测试概述

在游戏开发的流程中，测试是必不可少的一部分，因为游戏需要对外发布，使用的人群众多，并不是所有的玩家都会按照开发者设计的流程进行操作。当游戏的功能开发结束后，开发者不仅需要测试是否能运行，还要考虑非常规操作存在的漏洞。如果发布的游戏存在漏洞，有可能会导致财产的损失，因此，游戏开发结束之后，还需要进行大量测试。

## 游戏测试内容

通过游戏测试找到游戏中存在的漏洞或者逻辑错误，避免出现游戏错误或者进入死循环。测试游戏进入的情况，通过不同的逻辑进入游戏，确保游戏可以正常进入，不会出现游戏无法打开。打开游戏角色面板，测试非正常选择和非正常点击，确保玩家在选择角色的时候不会出现数据错误或者其他漏洞情况。打开关卡选择面板，非正常方式点击关卡，预防玩家非正常方式点击关卡出现数据错误或者进关卡错误。进入关卡测试测试操作玩家进行非常正闯关，确保玩家操作的角色不会进入死循环的地方或者出现角色卡死在地方无法通关。测试退出游戏，确保玩家退出游戏不会出现无法退出的情况，并且确保玩家数据保存，重新进入游戏不会出现错误。

## 游戏测试用例

### 游戏主界面模块测试用例

点击客户端进入游戏，主界面正常显示测试用例如表6-1所示。

表6-1 进入测试用例表

| 功能描述 | 进入游戏显示游戏主界面功能和角色 | | |
| --- | --- | --- | --- |
| 所属模块 | 进入游戏功能模块 | | |
| 用例目的 | 检验不同客户端进入游戏后，游戏主界面功能显示正常，并且玩家角色也正常出现，玩家的初始数据或者缓存数据都显示都正常，玩家可以正常进行游戏。 | | |
| 前提条件 | 用户安装游戏的客户端 | | |
| 用例ID | 输入/动作 | 期望结果 | 实际情况 |
| MFO-ZJM-001 | 安装并点击游戏客户端 | 进入游戏并且主界面显示正常 | 正常进入游戏，主界面显示正常 |

点击角色面板的角色头像，切换角色测试用例如表6-2所示。

表6-2角色切换测试用例表

| 功能描述 | 切换角色模型正常切换 | | |
| --- | --- | --- | --- |
| 所属模块 | 角色选择功能模块 | | |
| 用例目的 | 检验玩家进入游戏后是否可以正常切换角色 | | |
| 前提条件 | 玩家进入游戏并打开角色切换面板 | | |
| 用例ID | 输入/动作 | 期望结果 | 实际情况 |
| MFO-ZJM-002 | 点击角色选择面板上的角色头像 | 主界面角色模型切换成自己选择的角色模型 | 主界面模型成功显示自己选择角色模型 |

关卡选择面板选择关卡测试用例如表6-3所示。

表6-3 关卡选择测试用例表

| 功能描述 | 玩家能选择关卡并进入对应关卡或者点击未解锁关卡无效果 | | |
| --- | --- | --- | --- |
| 所属模块 | 关卡选择功能模块 | | |
| 用例目的 | 检验玩家是否可以正常进入关卡进行游戏 | | |
| 前提条件 | 玩家正常打开关卡选择面板 | | |
| 用例ID | 输入/动作 | 期望结果 | 实际情况 |
| MFO-ZJM-003 | 玩家点击关卡选择面板的可点击关卡按钮 | 进入游戏场景 | 正常进入对应的关卡场景 |

退出游戏测试用例如表6-4所示。

表6-4 关卡选择测试用例表

| 功能描述 | 玩家正常退出游戏并且关闭客户端 | | |
| --- | --- | --- | --- |
| 所属模块 | 退出游戏模块 | | |
| 用例目的 | 检验玩家是否可以正常退出游戏 | | |
| 前提条件 | 玩家已经打开客户端并在主界面 | | |
| 用例ID | 输入/动作 | 期望结果 | 实际情况 |
| MFO-ZJM-004 | 玩家点击设备对应的返回按钮 | 退出游戏并关闭客户端 | 正常退出游戏并关闭客户端 |

### 游戏局内模块测试

玩家操作角色的摇杆和按钮测试用例如表6-5所示。

表6-5 角色操作测试用例表

| 功能描述 | 通过游戏关卡界面摇杆和按钮操作游戏角色 | | |
| --- | --- | --- | --- |
| 所属模块 | 角色操作功能模块 | | |
| 用例目的 | 检验玩家是否可以通过关卡界面的摇杆和按钮操作角色 | | |
| 前提条件 | 玩家进入关卡界面 | | |
| 用例ID | 输入/动作 | 期望结果 | 实际情况 |
| MFO-JN-001 | 玩家滑动摇杆和点击操作按钮 | 正常操作人物角色移动和跳跃 | 角色正常移动和跳跃 |

金币管理功能测试用例如表6-6所示。

表6-6 金币管理测试用例表

| 功能描述 | 金币按照对象池规则出现并带有动画，消失也有动画 | | |
| --- | --- | --- | --- |
| 所属模块 | 金币管理系统 | | |
| 用例目的 | 检验金币是否正常出现和消失 | | |
| 前提条件 | 玩家进入关卡界面 | | |
| 用例ID | 输入/动作 | 期望结果 | 实际情况 |
| MFO-JN-002 | 操作玩家碰撞金币 | 金币正常消失并且其他地方正常出现带动画的金币 | 金币可以正常消失并且带有动画并且新出现的金币也有动画 |

游戏通关功能用测试例如表6-7所示。

表6-7 游戏通关测试用例表

| 功能描述 | 局内游戏通关后返回主界面 | | |
| --- | --- | --- | --- |
| 所属模块 | 游戏通关功能 | | |
| 用例目的 | 检验玩家是否可以正常通关 | | |
| 前提条件 | 玩家进入关卡界面 | | |
| 用例ID | 输入/动作 | 期望结果 | 实际情况 |
| MFO-JN-003 | 玩家操作角色通过障碍到大城堡 | 到达城堡后返回主界面 | 正常返回主界面 |

角色死亡测试用例如表6-8所示。

表6-8 角色死亡测试用例表

| 功能描述 | 角色无妨通过障碍死亡时返回初始位置 | | |
| --- | --- | --- | --- |
| 所属模块 | 局内角色管理功能 | | |
| 用例目的 | 检验关卡内角色死亡是否恢复原位置并重新进行游戏 | | |
| 前提条件 | 玩家进入关卡界面 | | |
| 用例ID | 输入/动作 | 期望结果 | 实际情况 |
| MFO-JN-004 | 操作角色触碰到可以死亡的障碍 | 玩家回到关卡初始角色位置 | 正常回到关卡角色初始位置 |

## 游戏测试总结

黑盒测试法是游戏测试中常用的方法。所谓黑盒测试法就是测试人员在不知道游戏的开发逻辑的情况下，根据游戏的说明介绍进行测试。测试人员在测试的时候从玩家角度进行游戏测试，把玩家可能会进行的额全部操作尽可能的全部测试一遍。通过此次黑盒的测试，准确发现游戏存在的的逻辑漏洞并进行修复。确保了游戏在发布后不会出现大量漏洞造成财产损失。经过一系列的测试知道的测试的流程，并且了解到了测试的重要性，知道一款游戏如果不经过测试会出现大量的额漏洞导致游戏无法正常的运行。

结束语

通过这次3D版本的玛丽奥游戏开发，学到了大量Unity的知识，了解的Unity中自带的UGUI插件、粒子系统、动画状态机等知识。通过这个项目，了解了游戏的基本框架，游戏中使用的QFramework框架是一款综合性的框架，包含了大量游戏框架知识。制作游戏不再是盲目的按照逻辑一步一步开发，而是通过先分析游戏，然后在写框架，接着才是正常的逻辑开发。如果一步一步快发，虽然可以开发出来功能，但是很难和其他模块进行交互，并且很难对功能进行扩展。如果先创建好基本的模块框架再进行开发，就会很容易扩展和交互。

游戏开发过程中最重要的一点就是MVC结构。这次游戏开发深刻的了解了MVC架构，知道了数据和视图进行分离，不仅能很容易的控制页面的显示，也将数据进行了剥离。数据剥离的有点就是使整个游戏是数据进行驱动的，当逻辑出现修改，只需要修改数据，并不需要处理页面的东西即可。当页面有修改的东西，同样不需要修改数据。游戏中的金币、场景和角色等数据就是采用这种方式。当有角色需要新增的时候，只需要把数据和资源添加进去，并不需要对代码进行修改，就会自动增加新的角色。大大增加了游戏的扩展。

这次游戏采用的还有一个UniRx库。通过使用这个库让我的思想有了大大的提高。UniRx库的使用让我更加清晰了对象的概念，了解了数据的概念。不管是对象还是数据，都不是游戏逻辑，游戏逻辑是对数据进行处理的，所以写逻辑的时候并不能考虑有哪些数据存在，只需要了解自己需要处理的数据对象即可，这种方法就实现了数据修改，逻辑不变也新增游戏的数据。

本次游戏的缺点也有很多，并没有声音处理系统。声音处理系统也是一款游戏必要的地方，游戏的声音往往能吸引住玩家，和美术是一样的重要。游戏中的数据也比较少，比如金币规则，积分规则和关卡内敌人AI等，这些都还没有实现，所以游戏还是有很大的提升空间。

总之，通过本次游戏开发，不仅认识到了自己不足的地方，同时也学习到了很多新的知识，比如游戏框架、游戏测试等。清晰的了解了游戏开发的流程。我相信通过我的不断学习，将来一定能将这款游戏扩展的更加强大，逻辑更加复杂，还原度更加高。

参考文献

1. 刘多多.基于OpenGL的3D游戏设计[D].吉林大学,2016.
2. 张策. 基于Unity3D的三维游戏场景设计与实践[D].合肥工业大学,2016.
3. 宣雨松.Unity3D游戏开发[M].人民邮电出版社,2012.6.
4. 冯乐乐.Unity Shader入门精要[M].人民邮电出版社,2016.5.
5. 熊耀.基于Unity3D粒子系统的三维影视特效开发研究[J]．软件导刊,2012。
6. 索利斯.C#图解教程[M].人民邮电出版社,2009.
7. 张典华，陈一民，李磊．基于Unity3D的多平台三维空战游戏的开发[J]．计算机技术与发展,2014.
8. Karli Watson，Christian Nagel. C#入门经典[M].清华大学出版社,2006.5.
9. 蔡升达.设计模式与游戏完美开发[M].清华大学出版社,2017.1.
10. 程杰.大话设计模式[M].清华大学出版社,2007.12.
11. 陈嘉栋.Unity3D脚本编程[M].电子工业出版社,2016.9.
12. 张灿.游戏管理系统的设计与实现[D].北京交通大学,2016.
13. 徐婉珍，李强，魏菊霞.在游戏开发实践中培养程序设计能力[J].计算机教育,2017.
14. 朱玉.基于Unity3D漫游系统的设计与研究[J].信息安全与技术,2014。
15. 姜雪伟. Unity3D实战核心技术详解[M].电子工业出版社,2017.1.

致谢

毕业设计到此已经完成了，随之即将结束的也是我的大学生活，我的学业生涯也有可能结束，我在此非常感谢我在南阳理工学院的大学四年的任课老师，是他们造就了现在的我，是他们教会了知识和学习方法，让我在计算机这个行业中能走的更远。

我还要感谢那些参考文献的作者和网上资料的那些无私奉献者，是他们的文献让我在Unity上面有更大的进步，也是在他们的帮助下，我完成了这次的毕业设计，在此，我忠心的感谢他们。

本次毕业设计过程中，我最要感谢的是我的毕业设计导师惠征老师，在毕业设计刚开始的时候，是在老师的意见下定了这次毕业设计的选题，在我完成毕业设计任务书时，也是老师给我提供了宝贵的意见和建议。老师每次都能一针见血的指出我的缺点和不足，我相信，如果没有老师的指导，这次毕业设计我是不可能完成的。在毕业设计论文的编写和修改过程中，也是老师认认真真的分析我的论文，指出论文的不足和错误，让我的能力有了很大的提升，受益匪浅。再次我再次感谢指导老师对我的帮助。

过去的时间已经流逝，但在过去时间里，老师对我的帮助我回永远记在心中，大学的同学、任课老师、辅导员以及大学的点点滴滴都给我留下了美好的回忆，母校，再见了，希望下次见到你的时候你会变得更加出名、更加强大！