分类号：**中图分类号** UDC：D10621-408-(选择)流水号-0

密　级：选择编号：**2014052063**

成都信息工程大学

学位论文

基于Unity的二维游戏“W”的设计与实现

|  |  |
| --- | --- |
| 论文作者姓名： | 王昊辰 |
| 申请学位专业： | 计算机科学与技术 |
| 申请学位类别： | 工学学士 |
| 指导教师姓名（职称）： | 杨昊（选择） |
| 论文提交日期： | **2018年5月10日** |

基于Unity的二维游戏“W”的设计与实现

**摘　要**

随着科学技术的发展，近些年来游戏产业蓬勃发展，甚至在近些年呈现喷井趋势，其中过关类飞行射击游戏一直是经久不衰的游戏经典。本文比较全面地描述了基于Unity3D游戏引擎的2D游戏的开发与实现，利用Unity易上手，跨平台，组件丰富等特性，开发一个轻量级的PC端小游戏。本游戏主要功能包括：不同主角的选择，主角的行为控制，敌人的生成和AI行为，道具系统，计时计分系统等功能，玩家需要通过键盘控制主角的行为通过关卡完成游戏。该游戏的设计采用了闯关模式，游戏分为多个关卡，难度依次增加，设置了几种不同的怪物种类，使得游戏的可玩性更高，同时各种不同的游戏道具和技能的使用，也使得玩家在游戏中可以获得更多乐趣。

**关键词：**Unity3D;射击游戏;2D;C#

**The design and development of 2D game “W” based on UNITY3D**

**Abstract**

With the development of science and technology, the game industry has flourished in recent years, even in recent years, showing the trend of jet wells. Among them, passing flight shooting games have always been an enduring game classic. This article describes the development and implementation of 2D games based on the Unity3D game engine, and uses Unity's features of easy-to-use, cross-platform and rich components to develop a lightweight PC-based mini-game. The main functions of the game include: selection of different protagonists, behavior control of protagonists, generation of enemies and AI behavior, prop system, and timing scoring system. Players need to control the protagonist's behavior through the keyboard to complete the game through the level. The design of the game adopts a mode of crossing points, the game is divided into multiple levels, the difficulty is increased in turn, several different types of monsters are set, making the game more playable, and the use of various game props and skills. It also allows players to have more fun in the game.

**Key words:** Unity3D; Shooting Game; 2D; C#

**目 录**

论文总页数[[1]](#footnote-0)：28页

[1 引言 1](#_Toc20185)

[1.1 游戏设计背景 1](#_Toc8646)

[1.2 国内外研究现状 2](#_Toc13408)

[1.2.1 国内研究现状 2](#_Toc11807)

[1.2.2 国外研究现状 3](#_Toc28971)

[1.3 游戏设计目标 3](#_Toc2136)

[1.4 主流开发引擎介绍 4](#_Toc3664)

[1.4.1 Cocos2D-x 4](#_Toc5276)

[1.4.2 Cry Engine3 4](#_Toc2317)

[1.4.3 UDK 4](#_Toc26087)

[1.4.4 Unity3D 4](#_Toc28738)

[1.5 开发语言介绍 5](#_Toc16423)

[2 需求分析及项目方案 5](#_Toc20102)

[2.1 需求分析 5](#_Toc2160)

[2.1.1 概要设计 5](#_Toc19983)

[2.1.2 功能需求 6](#_Toc31688)

[2.2 项目方案 6](#_Toc11472)

[2.2.1 项目特色 6](#_Toc2385)

[2.2.2 游戏存档技术简介 6](#_Toc2670)

[2.2.3 关卡异步加载技术简介 8](#_Toc21222)

[2.2.4 帧动画技术简介 9](#_Toc9985)

[2.2.5 协同程序技术简介 9](#_Toc12123)

[2.2.6 开发工具和运行环境 10](#_Toc8379)

[3 项目设计及实现 11](#_Toc9218)

[3.1 系统整体框图 11](#_Toc28651)

[3.2 功能模块的设计与实现 11](#_Toc14120)

[3.2.1 玩家控制模块的设计与实现 11](#_Toc7159)

[3.2.2 AI模块的设计与实现 14](#_Toc19365)

[3.2.3 关卡加载模块的设计与实现 23](#_Toc14656)

[3.2.4 掉落系统模块的设计与实现 23](#_Toc30127)

[3.2.5 角色选择模块的实现 24](#_Toc32527)

[4 特殊问题及解决方案 25](#_Toc31638)

[4.1 UI布局问题 25](#_Toc32250)

[4.1.1 问题描述 25](#_Toc2325)

[4.1.2 解决方案 25](#_Toc609)

[4.1.3 结果 26](#_Toc4689)

[4.2 脚本引用丢失问题 26](#_Toc19449)

[4.2.1 问题描述 26](#_Toc3015)

[4.2.2 解决方案 26](#_Toc19068)

[4.2.3 结果 27](#_Toc1717)

[4.3 进度条过渡生硬问题 27](#_Toc20718)

[4.3.1 问题描述 27](#_Toc32197)

[4.3.2 解决方案 27](#_Toc31783)

[4.3.3 结果 27](#_Toc15357)

[结 束 语 27](#_Toc11897)

[参考文献 29](#_Toc22593)

[致　　谢 31](#_Toc20902)

[声　　明 32](#_Toc20977)

# 引言

## 游戏设计背景

近些年来，由于我国市场经济的告诉发展，从而使得文化产业的发展更加的突出。尤其是以互联网络游戏以及移动客户端游戏等为代表的相关产业，呈现出了高速发展的趋势，各方面的研发团队不断壮大，各方对其的自资金投入也不断提高，行业规模不断增大，增长速度十分的突出。[1]网络游戏已经发展成为我国文化展业的支柱，同时也是推动相关新型文化发展的关键动力。互联网络的蓬勃发展以及在人们日常生活中的循序普及让相关联的电子游戏成为了一个大规模的产业，发展成为一个富有活力的新型经济，同时也将促进其科技新媒体和相关制造业的持续发展，成为促进语气有关产业的新的经济增长[2]。

因此越来越多的游戏引擎也被开发了出来。 Unity3D引擎集合了许多以往大获成功的游戏引擎的技术，同时又在跨平台发布游戏的简便性上做出了长足的进步， 使得开发过程中可以省去很多的底层编码环节，能把工作的中心更多地放在游戏的具体表现上，从而大大地缩短了开发周期， 且降低了开发成本[3]，并且通过 Unity3 D可以与时俱进地开发出 AR， VR游戏。 Jiahui Du等人[4]就通过Unity3D对AR技术进行了调查与研究。 Unity3D引擎为开发者在开发游戏的过程中提供了基础的框架和一些共有的特性，作为开发游戏的基础，使得该引擎在相关游戏的开发领域收到了强烈的欢迎， 同时最近提供了对2 D游戏开发的支持[5]，增加了生成2 D动画的 Animation和 Animator控制器。 因此选择Unity3D作为所设计游戏的开发引擎。

关于目前移动端游戏日渐火爆而客户端游戏日益衰落的说法，根据艾瑞咨询由图 1-1 艾瑞数据可得知，真相并不尽如此：Q3中国网络游戏产业规模高达441亿元，环比增长5.9%，同比增长19%，根据数据显示端游产业的市场占比在循序回升。而事实上，主机客户端游戏与移动客户端游戏并不是天然的的竞争关系，相反的两者在许多方面都是具有互补功能。手机游戏不仅不会抢走其他客户端的玩家，反而实在接受端游流失的玩家，并在吸引更多新的游戏玩家方面起到了积极的良性作用。现在许多端游的热门IP都推出了其在手游段的版本，正是由于两者之间是相互促进，相互提升的关系。在我看来，不论是端游还是移动客户端，仅仅是游戏在不同终端的展现，而决定一款游戏能否吸引更多的玩家，使得更多的玩家可以在游戏中有所收获，还是要在于游戏本身是否具有可玩性以及创新性的玩法。

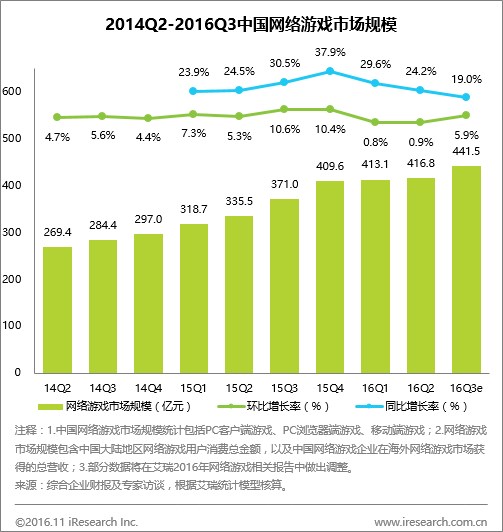


图 1-1 艾瑞数据

## 国内外研究现状

### 国内研究现状

近些年来，随之电子信息技术和三维数字技术的急速发展，越来越多的三维游戏逐步成为了市场的主流游戏，然而二维游戏并没有收到很大程度的影响，相反在移动游戏领域、网页游戏领域等仍然收到了玩家的热烈欢迎。根据相关的行业报道显示：尽管三维游戏数量在市场上占据主流地位，但最受玩家关注任然是二维游戏，占据了近六成的关注度，领先与三维游戏的关注度[6]。而在国内，由于三维游戏的研发技术还不是那么成熟，再加上市场投入的资金有限，使得国内三维游戏的发展比较缓慢。相反的是由于二维游戏开发的整体技术难度较低，使得需要投入资金较少，在市场上仍具有广阔的发展空间。同时二维游戏表现能力强，形式众多，玩法多种多样，也深受玩家群体的喜爱。尽管如今 AR， VR技术盛行，2 D游戏却始终受到大众的喜欢，比如前段时间的 steam上的热门游戏《 Don’ t Starve Togethor》《机械迷城》， 还有近期火爆的像素游戏《星露谷物语》，这些游戏都是 PC端的2 D小游戏。 以《 Don’ t Starve Togethor》来说，玩法就是玩家控制角色在一个虚拟的野外场景中生存， 使用场景中的道具使角色存活下来，游戏以天数来计时，这个游戏具有很高的游戏粘性，游戏中会出现各种不定因素，让玩家更想要继续玩下去。 一旦角色死去便重新开始游戏，这个方式也有可能会让玩了很久的玩家产生一种放弃的心态。 还有《纪念碑谷》这个移动端的闯关游戏，画风精美，玩法新奇，巧妙地运用了玩家的视觉差， 在开发人们的大脑的同时，给人们以视觉上的享受。

### 国外研究现状

Jinghai Jin等人[7]，设计开发出了一款基于机接口的口袋PC游戏，重点研究人机交互方面。FanKang[8]开发了一款交互性的游戏，旨在研究用户对游戏引擎交互性的体验。在国外，Seo J S等人[9]对于2D射击游戏进行了提升游戏体验的研究和实验。胡虚怀等人[10]结合一个实际的开发项目给出基于移动设备的2D游戏的设计方法。

闯关类的冒险游戏一直都深受玩家的喜爱，从80，90年代任天堂出品的红白机风靡全球起，《超级玛丽》，《魂斗罗》等游戏就一直被奉为经典之作。此外根据苹果软件的数据先手，经典的二维游戏如《口袋妖怪》、《魂斗罗》等，在手机版本上架首日就获得超过百万的下载量，而手机超级马丽华，在美国下载量达到350玩，全球下载量更是超过了千万。 根据游戏的火爆程度来看，这种横版闯关游戏在国内外仍旧存在一定的市场和许多热衷的玩家， 但是传统的游戏移植到移动平台之后，获得的游戏体验反馈却不是很好，很多人表示在移动平台， 像是手机这样一个小的屏幕很难操作游戏，游戏的体验感大不如前。

## 游戏设计目标

本文主要是基于Unity3D引擎来设计与开发一款PC端的飞行射击小游戏。 游戏的主要内容是玩家用键盘控制主角的移动，攻击等动作，躲避敌人的子弹并且杀敌敌人等，最终在生命值大于0的情况下，消灭敌人，完成通关。 三关游戏的内容各不相同，难度逐渐增加。 主要的需要完成的几个内容： （1）游戏的总体规划（2）碰撞检测（3）分数与生命值的计算（5）动画的实现

通过对于这款飞行射击小游戏的设计与开发，能够初步掌握游戏的发展前景，并且能够独立去学习与游戏引擎相关的知识。 更多地能够熟悉游戏创作的流程，从游戏策划，美术设计等等到了解玩家的心理需求的问题。 同时用C#语言来写脚本，能够使自己对C#语言的使用更加地熟练。 总体来说，锻炼了个人独立解决困难、制作游戏的能力。

## 主流开发引擎介绍

目前的游戏引擎大致可分成三类：商业引擎，开源引擎与专用内部引擎，还有一些开发者自己设计游戏开发引擎。其中商业引擎占了很大比例，比如Unreal3,CryEngine3,Unity3D等。Unity3D是因其开发成本低，以及支持多平台等优势而受到众多开发者的青睐。开源引擎则是由业余或者专业的开发者制作后再在网上免费发布的，其源代码是公开的。比如cocos2D-x等。专业内部引擎是一些公司自己开发的游戏引擎。如顽皮狗公司所开发的《古惑狼（Crash Bandicoot）》系列游戏引擎[11]。黎忠文等人[12]在研究Android手机游戏引擎基本原理和相关人工智能技术的基础上，设计实现了一个Android手机游戏引擎。

### Cocos2D-x

cocos2D-x是一个开源的，跨平台的，轻量级的游戏引擎。它的一大优点是开源跨平台游戏引擎，支持各种主流的移动平台。尤其是在开发2D游戏方面，cocos2D-x受到很多开发者的喜爱，但这也正是cocos2D-x的不足之处，因为其只能开发2D游戏，所以面对目前日益火爆的3D游戏开发，就稍显吃力。第二个不足之处是cocos2D-x只是一个开发库，如果想要用它来开发游戏，必须找到与其相关的编辑器，并为其集成第三方的物理引擎后才能很好地进行开发，没有集成的开发环境方便。

### Cry Engine3

CE(Cry engine 3)游戏引擎是由德国CRYTEK公司出品的一款和最新技术DirectX11相对应的游戏引擎。基本上是以Cry Engine 2为基础开发的。Cry Engine 3游戏引擎提供了精美的游戏画面，在画面渲染过程中使用了光影技术，包括实时动态全局光站技术，视野范围雾化技术，延迟光照技术。

### UDK

UDK(the Unreal Development Kit虚幻引擎开发工具包)是Epic公司在2020年宣布对外发布著名引擎虚幻动作第三代（Unreal Engine 3虚幻引擎3）引擎的免费版本。

商业方面，游戏市场上众多的大型游戏都是使用虚幻引擎3开发的，如镜之边缘，荣誉勋章空降兵，生化奇兵等。同样，在美术开发方面，基于虚幻引擎的UDK同样具有顶级的渲染质量,灯光，阴影，颜色，场景面数和帧率。不过，UDK仍有不足之处，只能支持pc端和ios平台的开发。

### Unity3D

Unity3D是由Unity Technologies公司发行的游戏开发引擎，是一个跨平台的综合型游戏开发工具。创新的可视化创作环境，简单易懂的使用方法，和大量成熟的官方教程，以及高效的资源管理，成熟的语言开发等，为unity3D吸引了大批开发者和游戏爱好者。[13]

Unity3D是一个专业全面的游戏引擎。通过Unity3D友好简洁的开发环境，我们可以轻松地完成许多困难的程序工作。如Unity3D为我们提供了许多基本的控制脚本：第一人称视角控制脚本、第三人称视角控制脚本等。图形处理方面，Unity3D不但拥有对DirectX和OpenGL高度优化的图形渲染管道，并且同时支持3DSMAX和Maya等主要的美术资源制作工具导出的游戏模型文件以及相应的模型动画，能让开发团队便捷的保留以前的美术资源积累[14]。也有很多游戏厂商应用Unity3D开发游戏，美国的Playernet公司到现在为止已经推出比较受欢迎的军事网游 《激战海陆空》 [15]。上官大堰等人[16]设计并实现一款基于Android移动平台Unity3D双人联机对弈象棋游戏。朱玉等人[17]提出了一种基于Unity3D的漫游系统的设计模式,并在此基础上讨论了在漫游系统中用到的U3D引擎的碰撞检测、自动寻路、基于NGUI的UI制作以及粒子系统等关键技术。

## 开发语言介绍

在Unity3D中，脚本可以使用JavaScript，C#,以及Boo。考虑到大部分商业游戏开发都是使用C#作为脚本语言，因此选择C#作为游戏的设计语言。[18]C#是微软公司推出的一种简单，安全，稳定的面对对象的编程语言，它在继承了C和C++强大功能的同时去掉了一些他们的复杂性。C#的语言体系被设计成一种“简单，现代，通用”，以及面向对象的程序设计语言[19]。

C#能够提供对于以下软件工程支持的要素的支持：强类型检查，数组维度检查，未初始化的变量引用检测，自动垃圾收集。由于C#的这些特性使得它成为嵌入式系统或者独立系统编程语言的又一个选择。不论是繁杂的大型系统，还是简单的小系统都可以利用C#进行程序编写[20]。

正是由于C#面对对象的卓越设计，使它成为构建各类组件的理想之选，无论是高级的商业对象还是系统级的应用程序。

# 需求分析及项目方案

## 需求分析

### 概要设计

游戏开发主要基于引擎和开发工具来进行系统实现，使用图形引擎和声效引擎，物理引擎提供对画面和场景音乐音效的支持。利用脚本编辑器工具来完成游戏的逻辑实现。使用C#语言完成游戏对象的脚本编辑，从而实现对游戏对象的数据以及显示逻辑。

本文中基于Unity3D游戏引擎设计制作的2D飞行射击类游戏，主要内容是玩家用键盘控制主角移动，射击等动作，躲避子弹或者杀死怪物等，最终在生命值大于0的情况下，到达下一关/目的地。三关游戏的难度逐渐增加。

### 功能需求

1)主角控制：玩家通过键盘控制主角移动和攻击，来躲避怪物，或者消灭怪物。

2）敌人控制：每个关卡都有不同的怪物，不同的怪物具有不同的攻击模式，如可以发射不同类型的子弹对主角进行攻击。

3）道具系统：主角可以拾取敌人掉落不同种类的道具，从而获得相对应的功能。

4）选人系统：玩家可以在开始界面选择不同的主角进行游戏。

5）UI系统： 游戏应该有相对应的计时计分系统，以及射击时的音效和背景音乐等功能。

## 项目方案

### 项目特色

1) 良好的平台移植性：本项目采用Unity3D作为开发引擎，Unity3D的C#基础脚本模块是通过Mono来实现的，Unity通过Mono将C#脚本代码编译成CLI，然后Mono运行时利用JIT或者AOT将CLI编译成目标平台的原生代码实现跨平台。

2) 丰富的AI模式：游戏中共设计了8类行为模式、属性各异的小怪敌人，并包含一类共有8种不同攻击模式的Boss，AI模式内容丰富，可玩性较强。

3) 反馈机制：掉落系统与战机的大招技能释放，通过对战斗属性的类型与数值修改，加上各类视觉特效，给予玩家丰富的游戏反馈，提升战斗体验。

4) 角色选择：游戏开始前，可对主角机与人物进行选择。

### 游戏存档技术简介

Unity里提供了一种本地化存储的方式，能够简单的存一些比较小的数据。

可以理解为持久化储存，还可以理解为游戏存档，存档中记录了玩家之前进行游戏中保存过的游戏数据。

这些都是以数据的形式存在PlayerPrefs中的。

1. 概念：PlayerPrefs是Unity内置的一个静态类，可以用于存储一些简单的数据类型：int ,string ,float。可以用它来做成计分板这类需要可持续化存储的小地方。
2. PlayerPrefs
   1. 分为五类：SetXXX , GetXXX , DeleteXXX, HasKey , Save.
   2. SetXXX : 是你要储存的数据类型，这里必须是一组键值对，第一个参数是Key,第二个参数是对应的Value。
   3. GetXXX：这里是获取数据的函数，只有一个参数Key,用来获取对应Key的Value.
   4. HasKey: 这个可以用来做判断。比如有两组数据一组有XX这个Key,另一组没有XX这个Key,这个时候就可以对这两组数据进行差异化操作。
   5. DeleteXX:用于删除数据。
   6. Save:用于在突发退出程序时，保存数据以备恢复时使用，但是会导致程序间断所以不建议调用。
3. PlayerPrefs数据的存储位置：

在Windows独占模式下，PlayerPrefs被存储在注册表的 HKCU\Software\[company name]\[product name]键下，这里company和product名是在Project Setting中设置的。

* 1. 首先，HKCU就是Windows中的注册表，首先通过Win+R 进入运行界面，输入regedit进入注册表如图 2-1：

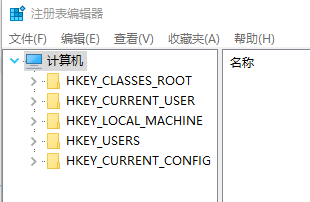


图 2-1注册表

* 1. 第二个就是HKCU，此时，如果直接从HKCU中找到Software，再找Company是找不到的，需要从以下路径找到： Unity/UnityEditor/DefaultCompany/donghua
  2. 存储的数据就在这里。这里company和product名可以在Unity中找到如图 2-2图 2-3：

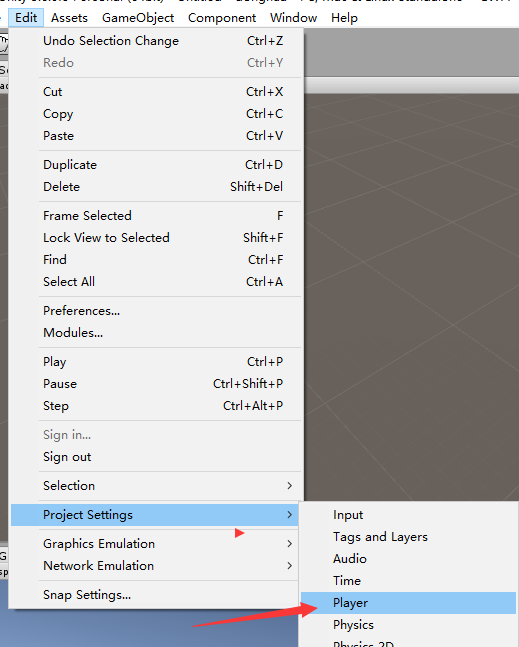


图 2-2

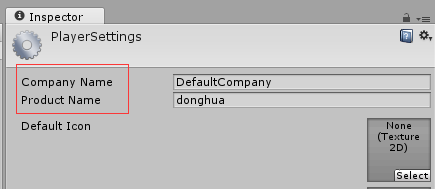


图 2-3

### 关卡异步加载技术简介

1. 背景：

游戏中经常需要进行场景的加载，而有时候有些场景因为需要加载大量的资源，导致在加载场景时感觉很卡。这时候需要使用异步场景加载来解决这样的问题。

1. 异步加载场景API：

**Func**:

public static AsyncOperation LoadSceneAsync(sceneName, mode)

public static AsyncOperation LoadSceneAsync(sceneIndex, mode)

**Parameters**:

sceneName: Name or path of the Scene to load.

sceneBuildIndex: Index of the Scene in the Build Settings to load.

Mode: If LoadSceneMode.Single then all current Scenes will be unloaded before loading.

* 1. 描述：

在后台异步加载场景。给定的场景名称既可以是完整的场景路径，也可以是构建设置窗口中显示的路径，也可以是场景名称。 如果只给出场景名称，则会加载匹配列表中的第一个场景。 如果您有多个具有相同名称但路径不同的场景，则应在构建设置中使用完整的场景路径。

* 1. 应用：

通常游戏的主场景包含的资源较多，这会导致加载场景的时间较长。为了避免这个问题，可以首先加载Loading场景，然后再通过Loading场景来加载主场景。因为Loading场景包含的资源较少，所以加载速度快。在加载主场景的时候一般会在Loading界面中显示一个进度条来告知玩家当前加载的进度。在Unity中可以通过调用SceneManager.LoadSceneAsync函数来异步加载游戏场景，通过查询AsyncOperation.progress的值来得到场景加载的进度。

### 帧动画技术简介

1. 概念：通过在关键帧设置该帧对应的精灵图片，将多帧连接起来，则形成由图片序列组成的2D帧动画；
2. 应用：导入所需动画的精灵图片序列，通过Unity自带的动画编辑器，选取关键帧，并在该帧从导入的动画图片序列中选取并设置欲表现的单张精灵图片，完成对该帧的编辑。对每一帧做相同步骤的操作，再播放此动画剪辑，则形成由精灵图片序列组成的2D帧动画。

### 协同程序技术简介

1. 概念：Coroutine在Unity3D中叫做协程或协同程序，和多线程类似，也就是说开启协同程序就是开启一个线程。但是在任意指定时刻只有一个协程执行，其他协程挂起。
2. Coroutine的相关函数：
   1. StartCoroutine：启动一个协程。
   2. StopCoroutine：终止一个协程。
   3. StopAllCoroutine：终止所有协程。
   4. WaitForSeconds：等待几秒。
   5. WaitForFixedUpdate：等到下一次FixedUpdate调用时执行。

使用MonoBehaviour.StartCoroutine方法即可开启一个协同程序，也就是说该方法必须在 MonoBehaviour 或继承于MonoBehaviour的类中调用。

1. Coroutine in Unity:

在Unity3D中，StartCoroutine(string methodName),

StartCoroutine(IEnumerator routine)都可以开启一个线程。

区别在于使用字符串作为参数可以开启线程并在线程结束前终止线程，相反使用IEnumerator 作为参数只能等待线程的结束而不能随时终止(除非使用StopAllCoroutines()方法)；

另外使用字符串作为参数时，开启线程时最多只能传递 一个参数，并且性能消耗会更大一点，而使用IEnumerator 作为参数则没有这个限制。

在Unity3D中，使用StopCoroutine(string methodName)来终止一个协同程序，使用StopAllCoroutines()来终止所有可以终止的协同程序，但这两个方法都只能终止该 MonoBehaviour中的协同程序。

### 开发工具和运行环境

1. 系统开发工具和开发环境如表2‑1所示。

表2‑1 开发工具表

|  |  |
| --- | --- |
| **开发工具名称** | **软件用途** |
| **Unity5.6.0f** | 开发平台 |
| **Windows 10 Pro** | 运行环境 |
| **应用宝** | 移动端版本打包工具 |
| **XMind** | 系统设计，创建流程图等 |
| **Adobe Photoshop CS6** | 美术素材制作与编辑 |

# 项目设计及实现

## 系统整体框图

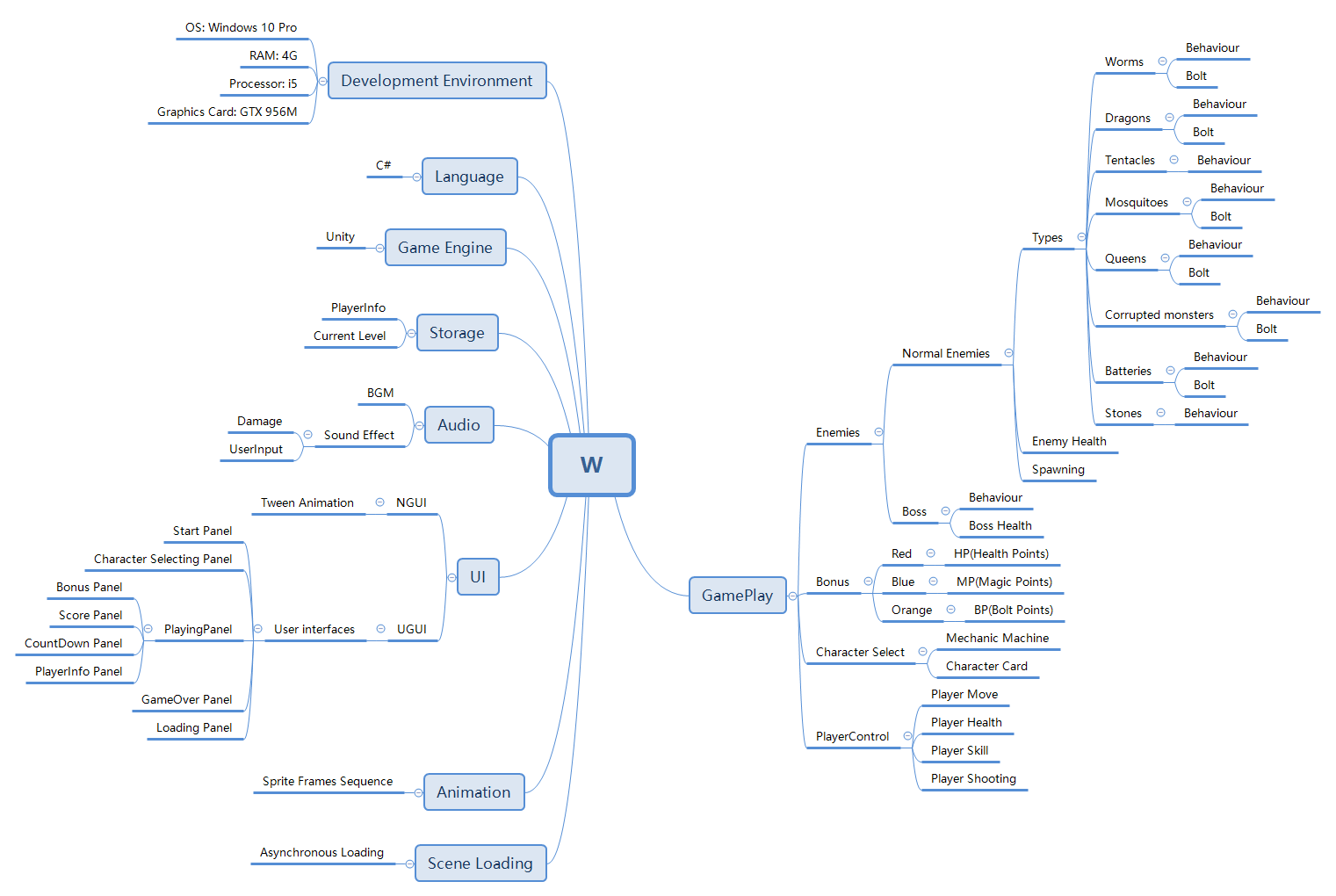


图 3-1

## 功能模块的设计与实现

### 玩家控制模块的设计与实现

1. 运动
   1. 键盘控制移动：通过获取键盘输入中的水平轴(Horizontal Axis)与垂直轴(Vertical Axis)的值，确定玩家运动向量的方向，根据实际需求可在外部调整其数值大小。确认移动速度的方向与大小后，将其赋值给刚体(Rigidbody 2D)组件中的速度(velocity)属性，实现运动功能；

float h = Input.GetAxis("Horizontal");

float v = Input.GetAxis("Vertical");

Vector2 move = new Vector2(h, v);

GetComponent<Rigidbody2D>().velocity = move \* speed;

* 1. 位置限制：

确定游戏物理边界，欲将主角位置限制在该边界内：每当主角位置发生移动后，通过调用Mathf.Clamp()方法，将主角的位置的数值大小控制在指定的限定范围内，防止玩家位置越过游戏主界面之外。

GetComponent<Rigidbody2D>().position = new Vector2

(

Mathf.Clamp(GetComponent<Rigidbody2D>().position.x, boundary.xMin, boundary.xMax),

Mathf.Clamp(GetComponent<Rigidbody2D>().position.y, boundary.yMin, boundary.yMax)

);

1. 射击：

调用Input.GetKey(KeyCode key)方法获取指定按键输入，作为射击键，并执行以下射击逻辑：

1. 开启发射口：根据当前发射状态级别，开启指定个数的发射口，并在各已被激活的发射口处实例化子弹物体；

void ShootSpawnsControl(int \_level) // 发射口控制

{

// \_level: 当前发射状态级别

for (int i = 0; i <= \_level; i++)

{

foreach (Transform singleSpawn in shotSpawnsDic[i])

{

// 实例化子弹物体

Instantiate(bolt\_Ordinary, singleSpawn.position, singleSpawn.rotation);

}

}

}

1. 播放射击音效：通过访问AudioSource组件，设置指定音效剪辑(AudioClip)，再调用Play方法，对音效进行播放；

GetComponent<AudioSource>().clip = audioClips[0]; // 指定射击音效

GetComponent<AudioSource>().Play(); // 播放射击音效

1. 设置射击间隔：为防止按键频率过高导致短时间内生成大量子弹，需要为两次连续的射击操作之间设置一定的时间间隔，类似技能冷却：当一次射击完成后，设置允许再次进行射击的时间间隔(即：从现在起，需要再经历多长时间才能进行下一次射击)，通过Time.time获取自游戏开始以来至当前帧所经历的时间，即当前时间刻度。将下一次射击的时间刻度设置为Time.time+fireRate(射击频率)，只有时间到达该刻度，才允许再次进行射击；

if (Input.GetKey(KeyCode.Z) && Time.time > nextFire) // 普通射击

{

nextFire = Time.time + fireRate;

}

1. 大招：

调用Input.GetKey(KeyCode key)方法获取指定按键输入，作为释放大招键，并执行以下逻辑：

1. 开启大招：开启一个协程，激活大招物体：粒子系统或帧动画，使其保持一段时间的激活状态后，再关闭使其不可见；

IEnumerator CastUltimateSkill() // 开启大招

{

ultimateSkill.SetActive(true);

isUnmatched = true;

yield return new WaitForSeconds(ultimateDuration);

ultimateSkill.SetActive(false);

isUnmatched = false;

}

1. 播放音效：通过访问AudioSource组件，设置指定音效剪辑(AudioClip)，再调用Play方法，对音效进行播放；
2. 设置冷却时间：开启大招后，通过获取可自定义的、允许再次进行大招释放的时间间隔(即：从现在起，需要再经历多长时间才能进行下一次射击)设置允许再次开启大招的时间刻度值，通过Time.time获取自游戏开始以来至当前帧所经历的时间，即当前时间刻度。将下一次允许开启大招的时间刻度设置为Time.time+cooldown(冷却时间)，只有时间到达该刻度，才允许再次释放大招；
3. 生命值损耗：

通过碰撞检测，对敌人或子弹与玩家自身是否进行碰撞进行逐帧监测，以判定健康值是否损耗：

1. 生命值损耗：通过调用OnTriggerEnter2D(Collider2D collision)，若coolision所代表的物体是敌人或子弹(可通过tag或layer进行判断)，则减血；否则无特殊响应。

void OnTriggerEnter2D(Collider2D other)

{

if (other.tag == "EnemyBolt" || other.tag == "EnemyAndBolt")

{

// do the damage logic

}

}

1. 游戏结束判断：若减血后，玩家当前生命值已小于等于0，则表示玩家所在生命点数用尽，调用游戏结束方法，进入游戏结束界面并提示进行返回操作；否则生命值尚存，调用减血方法，当前生命值减1。

if(livesCurCount > 0 && Time.time > nextDamage && !isUnmatched) {

nextDamage = Time.time + reDamagePeriod;

Dead();

} else if (livesCurCount <= 0) {

if (GameOver\_PlayerDead != null) {

GameOver\_PlayerDead();

}

}

void Dead()

{

GetComponent<AudioSource>().clip = audioClips[2];

GetComponent<AudioSource>().Play();

GetComponent<Animator>().SetTrigger("isDead"); // 重生动画

Instantiate(deadExplosion, transform.position, Quaternion.identity);

RedSqureMinus();

}

### AI模块的设计与实现

1. 设计：
   1. 小怪：
      * 1. 飞虫：
           1. 数量：3只/波
           2. 行为模式：3只飞虫依次飞入游戏主屏幕，开始以一随机旋转量发射1波包含4个飞行方向各不相同的子弹，每颗子弹伤害值为1；飞虫持续停留一段时间，直至受到伤害并死亡，再触发下一波敌人的生成。
           3. 图例：

图 3-2

* + - 1. 飞龙
         1. 数量：6只/波
         2. 行为模式：6只飞龙3只一组，两组飞龙分居两侧，每只飞龙隔随机时间发射一波子弹，每波子弹数量为6颗，生成机制如下：每波子弹发射前基于一个随机的旋转量，再从0-360范围内，以60的整数倍为其绕z轴的旋转量，朝本地x轴向外发射。
         3. 图例：

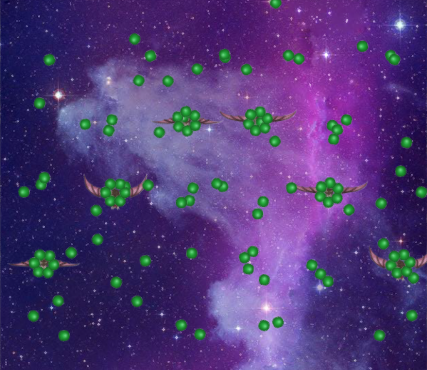


图 3-3

* + - 1. 触手：
         1. 数量：4只/波
         2. 行为模式：4只触手按顺时针方向，依次从左上角至左下角4个角落向主角当前所在位置戳去，每只触手在顶端部分到达目标位置后即收回，并触发下一只触手的行为。
         3. 图例：

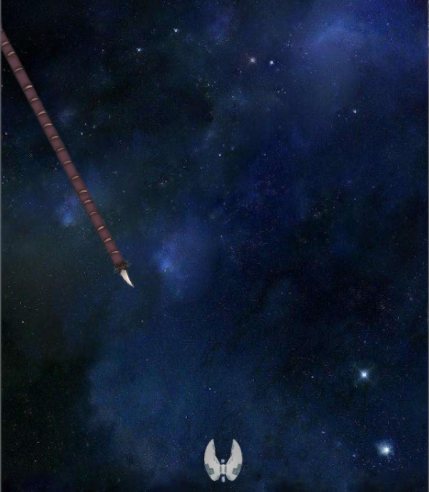


图 3-4



图 3-5

* + - 1. 自爆蚊：
         1. 数量：10只/波
         2. 行为模式：每波自爆蚊分两组，以相对于垂直方向顺时针旋转30度的直线作为运动轨迹，从靠近屏幕左上角与右下角两角落处作为生成点，保持直线运动；在自爆蚊受到攻击时，将以当前主角位置为目标，发射一串由4个子弹组成的子弹波。
         3. 图例：



图 3-6

* + - 1. 女王：
         1. 数量：2只/波
         2. 行为模式：2只女王怪分居屏幕左右，每只女王外围围绕着两颗发射球，一边不断绕女王作圆周运动，一边沿两条中心点不断旋转的曲线轨迹发射子弹。
         3. 图例：

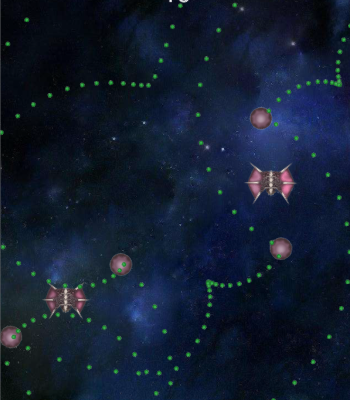


图 3-7

* + - 1. 腐化者：
         1. 数量：8只/波
         2. 行为模式：每波腐化者共8只，2只一组共分4组。4组腐化者分别依次从屏幕4个角落进入游戏主屏幕，每只腐化者以自身位置为起点，以当前主角位置为击打目标，发射孢子子弹。
         3. 图例：

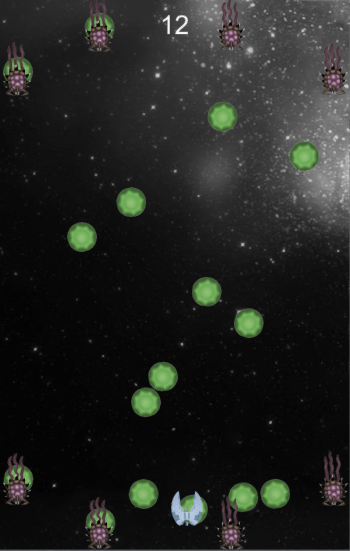


图 3-8

* + - 1. 孢子炮台：
         1. 数量：3只/波
         2. 行为模式：每只孢子炮台每隔一定时间间隔，从中心发射一波呈散射状的自爆蚊，自爆蚊属性参见上文；同时，从中心发射一波数量为4，旋转量随机的子弹，每颗子弹均以子弹生成点为中心，持续绕z轴进行逆时针旋转。
         3. 图例：

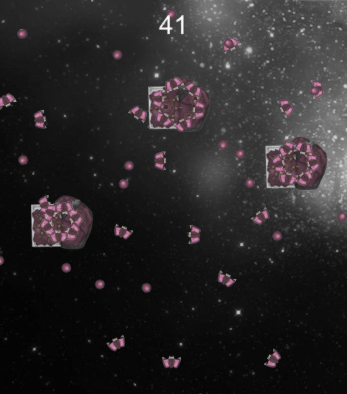


图 3-9

* + - 1. 陨石：
         1. 数量：10个/波
         2. 行为模式：每波陨石从屏幕正上方依次生成，已知屏幕宽度为w，高度为h，则每颗陨石随机的生成位置范围为从(-w/2, 2/h)至(w/2, 2/h)。陨石生成后给一个初始的旋转量并加上重力，模拟其物理运动。
         3. 图例：

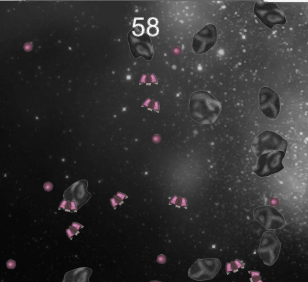


图 3-10

* 1. Boss：
     + 1. 行为模式：Boss共有5条血，分别对应5种行为模式：自爆蚊静态攻击模式、女王+水平扫射模式、触手攻击模式、自爆蚊智能攻击模式、多女王+水平扫射模式
       2. 图例：

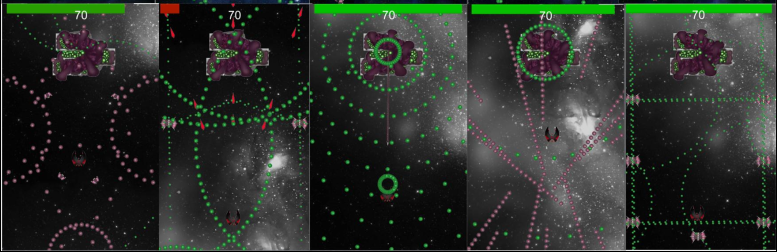


图 3-11

1. 实现：以触手怪为例，简述AI实现方法：
   1. 行为模式：4只触手按顺时针方向，依次从左上角至左下角4个角落向主角当前所在位置戳去，每只触手在顶端部分到达目标位置后即收回，并触发下一只触手的行为。
   2. AI逻辑：
      1. 开启协程，控制其行为模式的切换：
         1. 戳向主角；
         2. 挂起并停留一段时间；
         3. 从主角处缩回
         4. 关键代码如下：

IEnumerator HitPlayer()

{

yield return new WaitForSeconds(hitDelay);

// Go

TweenScale.Begin(gameObject, goDuration, endScale);

// Stay

yield return new WaitForSeconds(hitStay);

// Back

TweenScale.Begin(gameObject, backDuration, startScale);

yield return new WaitForSeconds(destroyDelay);

// Destroy

Destroy(gameObject);

}

* + 1. 逐帧判断，根据当前行为模式做出响应：
       1. 若当前模式为戳向主角，则：
          1. 调整朝向，使得触手顶端指向主角；
          2. 朝目标位置戳去；
       2. 否则：从主角处缩回
       3. 关键代码如下：

void Update () {

if(ifCanHit && !ifCanBack)

{

LookAtPlayer(); // 面向主角

TweenPosition.Begin(gameObject, hitDuration, playerBeHitPos); // 朝目标位置戳去

}

if(ifCanBack && !ifCanHit)

{

TweenPosition.Begin(gameObject, backDuration, startPosVec3); // 从主角处缩回

}

}

void LookAtPlayer()

{

if(Time.time > nextHit)

{

nextHit = Time.time + hitRate;

transform.LookAt(player); // 调整朝向，面向主角

transform.Rotate(new Vector3(90f, 0f, 0f)); // 绕 x 轴顺时针（自前向后正视）旋转 90 度

transform.Rotate(new Vector3(0f, 90f, 0f)); // 绕本地 y 轴顺时针（自右向左侧视）旋转 90 度

playerBeHitPos = player.position;

}

}

### 关卡加载模块的设计与实现

1. 设计
   1. 异步加载：不影响当前游戏场景的前提下加载新场景。通常异步加载的方式分为两种：第一种是异步加载新游戏场景，当新场景加载完成后进入新场景并且销毁之前的场景。第二种:同样异步加载新场景，新场景加载完毕后，保留旧场景的游戏对象并且进入新场景。本项目使用第一种方式进行场景的异步加载。
   2. 加载界面：通常游戏的主场景包含的资源较多，这会导致加载场景的时间较长。为了避免这个问题，可以首先加载Loading场景，然后再通过Loading场景来加载主场景。因为Loading场景包含的资源较少，所以加载速度快。在加载主场景的时候一般会在Loading界面中显示一个进度条来告知玩家当前加载的进度。
2. 实现：
   1. 异步加载：

在Unity中可以通过调用SceneManager.LoadSceneAsync函数来异步加载游戏场景；

* 1. 进度条：通过查询AsyncOperation.progress的值来得到场景加载的进度。通过Slider组件实现加载界面。

### 掉落系统模块的设计与实现

1. 设计
   1. 掉落类型：
      1. HP：Health Points, 拾起可增加健康值；
      2. MP: Magic Points, 拾起可增加技能释放次数；
      3. BP: Bolt Points, 拾起可增加发射弹口，获得发射增益效果。
   2. 触发条件：

在小怪被主角击杀后，根据小怪类型配置掉落类型的组合与其掉落概率进行触发配置。

1. 实现：
   1. 掉落类的实现：创建掉落类，包含两个字段：掉落类型与掉落概率，每种小怪中加入三种掉落类对象，依次代表三种掉落物品，配置其对应的类型与概率，关键代码如下：

// 掉落物品类

[System.Serializable]

public class DropOut

{

public GameObject dropOutObj; // 掉落类型

public float dropOutRate; // 掉落概率

}

public DropOut redDropOut; // HP

public DropOut blueDropOut; // MP

public DropOut yellowDropOut; // BP

* 1. 掉落触发条件的实现：在EnemyHealth.cs(怪物生命值控制脚本)中，在怪物被击杀后加入对掉落系统逻辑触发的判定：怪物被击杀后，分别通过随机数计算对三种掉落物品的掉落概率，若结果在概率范围内，则生成该掉落物品，代码如下：

// 敌人被击杀

protected virtual void EnemyDestroy()

{

GetComponent<SpriteRenderer>().enabled = false;

GetComponent<Collider2D>().isTrigger = false;

GameObject.Find("UI").GetComponent<ScoreManager>().AddScore(this.scoreIncrement);

DropOut(); // 触发掉落条件

}

private void DropOut()

{

DropOutBonus(redDropOut); // HP

DropOutBonus(blueDropOut); // MP

DropOutBonus(yellowDropOut); // BP

}

### 角色选择模块的实现

1. 设计：
   1. 主角类型：

分两类，每类均包含两种信息：

1. 战斗机甲
2. 人物卡牌
   1. 角色选择：在进入游戏前先进行角色选择，通过用户点击输入，获取玩家选择的角色信息，将其存储起来，进入游戏后载入相应的角色信息。
3. 实现：
   1. 玩家信息类PlayerInfo：
      1. 单例：必须将该类作为单例进行使用
      2. 关键字段：人物角色id、战斗机甲id
      3. 接口：
         1. 对关键字段的setter接口
         2. 对关键字段的getter接口
   2. 进入游戏后的角色类型设置PlayerInfoReadAndSet：
      1. 进入游戏后，调用PlayerInfo单例的getter接口，获取玩家所选则的主角信息；
      2. 根据得到的角色id，对相应UI元素设置对应的精灵图片，完成外观设置；
      3. 根据得到的角色id，对玩家游戏数值进行相应设置，完成数据设置；
   3. 用户输入操作：
      1. 选择操作：实现两个按钮：上一个、下一个，两个按钮的监听事件对应着对id号的切换；
      2. 确认操作：确认选择后，调用PlayerInfo单例的setter接口，设置其人物角色id与战斗机甲id；
   4. 选择完毕、进入游戏：激活PlayerInfoReadAndSet，使其获取玩家所选则的主角信息并进行相关设置。

# 特殊问题及解决方案

## UI布局问题

### 问题描述

若将游戏分辨率切换至另一不同于初始设置的分辨率，则UI布局杂乱或不可见。

### 解决方案

1. 自适应：游戏中的分辨率自适应主要包括两部分：一是在不同尺寸的屏幕下，整体缩放比例的计算方式；二是在不同比例（宽高比）的屏幕下，UI控件所处的位置关系，也称为布局。
2. UGUI缩放机制：
   * 1. Constant Pixel Size：固定像素尺寸，即按素材的“固定像素”渲染；
        1. Scale Factor：缩放比例，在素材原尺寸上的缩放比例，默认值是1；
        2. Reference Pixels Per Unit：每个unity单位对应的像素数；
     2. Scale With Screen Size：根据屏幕尺寸缩放；
        1. Reference Resolution：标准分辨率，这是我们提供给美术做图的标准分辨率，所有的UI素材都应该按这个分辨率去做；
        2. Screen Match Mode：Match Width Or Height 以宽高权重匹配；
        3. Match：宽高所占权重，默认值是0，相当于以“标准分辨率的宽”和“实际屏幕的宽”的比例作为缩放比例。同理，如果值是1，相当于以“标准分辨率的高”和“实际屏幕的高”的比例作为缩放比例。如果值是0.5，则相当于宽和高的比例权重相等，最终的缩放比=宽缩放比\*宽权重+高缩放比\*高权重；
        4. Reference Pixels Per Unit：每个unity单位对应的像素数；
     3. Constant Physical Size：固定物理尺寸；
        1. Physical Unit：物理单位，包括点，英寸，厘米，毫米等；
        2. Fallback Screen DPI：对应物理单位的像素密度；
        3. Default Sprite DPI：默认精灵的像素密度；
        4. Reference Pixels Per Unit：每个unity单位对应的像素数
3. 使用方法：
   1. Canvas Scaler 选择 Scale With Screen Size；
   2. Screen Match Mode 选择 Match Width Or Height，比例设为1，即只和高度进行适配，如图 4-1；

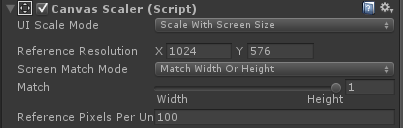


图 4-1

### 结果

在不同分辨率下，UI元素可根据实际情况进行自适应：

1. 保持相对位置不变；
2. 保持宽高比例不变。

## 脚本引用丢失问题

### 问题描述

对脚本资源进行修改后，部分游戏物体上的某些组件提示“Nothing Selected”，脚本引用发生丢失。导致子弹爆炸效果未能正常销毁。

### 解决方案

将版本回退至对脚本资源进行修改前的正常版本，Unity将自动导入先前丢失的资源与引用。

### 结果

引用被重新定位，组件恢复正常，爆炸效果贴图被正常销毁。

## 进度条过渡生硬问题

### 问题描述

通过LoadSceneAysnc进行异步加载时，进度条的数值更新不连续的，效果生硬不自然。

### 解决方案

每一次更新进度条的时候插入过渡数值。当获得AsyncOperation.progress的值后，不立即更新进度条的数值，而是每一帧在原有的数值上加1，这样就会产生数字不停滚动的动画效果，关键代码如下：

while (operation.progress < 0.9f) {

destProgress = operation.progress \* 100f;

while (displayProgress < destProgress) {

// 对于当前进度数值，每一帧在实际的基础上+1

loadingBar.transform.Find("LoadingProgText").GetComponent<Text>().text = ++displayProgress + "%";

loadingBar.value = displayProgress / 100f; // 设置进度条数值

yield return new WaitForEndOfFrame();

}

}

### 结果

经过插值操作之后，进度条数值每一帧都在实际加载进度的范围内进行更新，过渡自然，实现了柔和的动画效果。

结 束 语

通过本次毕业设计，让我受益匪浅，这篇论文也算是对这段时间对研发本项目的一个小结。通过本次设计我了解到了制作一款游戏一个相对完整的制作流程，从游戏开始前期的策划，对关卡的设计，对不同敌人的行为模式的规划，再到具体制作游戏对每一种敌人的设计，对游戏的具体把握，都让我对一款游戏的开发与设计有了更深入的了解。最终我完成了一款功能相对齐全的游戏，游戏的主要功能模块包括：主角的移动和攻击，游戏开始界面可以选择不同的主角进行游戏，设计多种多样的敌人和攻击模式，设计了道具系统，有种多种道具，主角可以通过拾取道具获得增加生命值，增加技能点数，增加攻击子弹数等功能。总体来说，本款游戏，具有一定的可玩性，也具有一定的创新型。

在制作本游戏的初期，由于之前缺乏开发一个完整游戏的经验，我从狭隘、不全面的角度去考虑问题，使得自己在开发过程中遇到许多的问题。后来在指导老师和同学的帮助下，我学会了用更加全面的角度去考虑问题，在这些过程中，虽然有时会让我很煎熬，但是回报有合适很丰富的，我学到的知识以及看待问题的方式都大大提升了我的专业素质和工作能力。由于自身的能力有限，所做出的游戏仍然有许多地方需要完善处理，今后对于这款游戏会添加更有新意的道具玩法和更多关卡，并使游戏本身具有更友好的交互系统，音效音乐的多样化，视图的精美化，使得玩家在视听觉上同时得到更加愉悦的感受。

参考文献

1. 杜磊. 电子游戏 文化产业的重要一环[J]. 计算机与网络, 2016, 42(1): 1-1.
2. 许君婵. 我国网络游戏产业的现状和发展趋势[J]. 电子技术与软件工程, 2017(1): 23-23.
3. 路朝龙. Unity3D游戏开发从入门到精通[M]. 北京:中国铁道出版社, 2013: 102-103.
4. Jiahui Du, Pengkai Xu,Tingyu Hou. Research of Augmented Reality based on Unity3D[J]. ICMMCCE. 2015,12: 326-336.
5. Hongye Zhong, Jitian Xiao. Apply Technology Acceptance Model with Big Data Analytics and Unity Game Engine[J]. IEEE. 2015, 9: 147-154.
6. 宋玮.中国网络游戏产业发展研究[D]. 北京:首都经济贸易大学, 2012: 23-24.
7. Jinghai Yin, Jianfeng Hu.Design and Implement of Pocket PC Game Based on Brain-Computer Interface [J]. IEEE. 2011, 9: 1917-1932.
8. FangKang. Development of interactive Game Based on User experience Game Engine[J]. MEITA. 2016, 3: 152-157.
9. Seo J S, Kim N G. Empirical Analysis of the Feeling of Shooting in 2D Shooting Games[J]. Synthetic Communications, 2010, 41(16): 2359-2373.
10. 胡虚怀. 移动设备中基于MIDP 2.0的2D游戏的开发[J]. 计算机工程与应用, 2005, 41(13): 104-107.
11. 程明智, 王一夫. Unity游戏开发技术[M]. 北京:国防工业出版社, 2012: 45-48.
12. 黎忠文,覃志东, 王全宇等. 基于Android平台手机游戏引擎的设计与实现[J]. 计算机工程与设计, 2014, 35(1): 119-124.
13. 李志鹏. 基于Unity3D引擎的空中战机游戏设计与实现[D]. 北京:首都经济贸易大学, 2016: 21-22.
14. Ling He, Xiao-qiang Hu, Rui Su. The Development of Educational Games Based on the Combination of Leap Motion and Unity 3D[J]. IIMM . 2015, 11: 201-203.
15. 李志鹏. 基于Unity3D引擎的空中战机游戏设计与实现[D]. 北京:首都经济贸易大学, 2016: 11-12.
16. 上官大堰, 黄心渊, 孙望舒等. 基于Android平台的Unity3D网络功能的开发[J]. 计算机应用与软件, 2015(8): 72-75.
17. 朱玉. 基于Unity3D漫游系统的设计与研究[J]. 信息安全与技术, 2014(12): 78-81.
18. 林深华, 范志尚, 蒋建兵等. 基于Android平台Unity3D游戏设计与实现[J]. 企业科技与发展, 2013, 10: 40-42.
19. 刘仙勇. 从iPhone谈苹果公司营销策略的创新与应用[M]. 北京:机械工业出版社, 2012: 432-436.
20. 陈风. 手机游戏盈利模式待探索[J]. IT时代周刊, 2010, 7: 41-42.

致　　谢

本文实在杨昊老师的悉心指导下完成的，他在该项目的开发以及论文完成上给予了我许多的关心和帮助，同时还要感谢魏敏、文武、林岗老师在游戏开发给予我的指导，我的进步和成长离不开老师们的帮助!

同样感谢各位同窗好友，感谢大家陪我度过四年的美好大学生活

最后向在百忙之中评审本文的各位专家、老师表示衷心的感谢！

**作者简介：**

|  |  |  |  |
| --- | --- | --- | --- |
| 姓 名： | 王昊辰 | 性别： | **男** |
| 出生年月： | 1995.10.13 | 民族： | **汉族** |
| E-mail： | whc1995@vip.qq.com |  |  |

声　　明

本论文的工作是2017年10月至2018年5月在成都信息工程大学计算机学院完成的。文中除了特别加以标注地方外，不包含他人已经发表或撰写过的研究成果，也不包含为获得成都信息工程大学或其他教学机构的学位或证书而使用过的材料。

关于学位论文使用权和研究成果知识产权的说明：

本人完全了解成都信息工程大学有关保管使用学位论文的规定，其中包括：

（1）学校有权保管并向有关部门递交学位论文的原件与复印件。

（2）学校可以采用影印、缩印或其他复制方式保存学位论文。

（3）学校可以学术交流为目的复制、赠送和交换学位论文。

（4）学校可允许学位论文被查阅或借阅。

（5）学校可以公布学位论文的全部或部分内容（保密学位论文在解密后遵守此规定）。

除非另有科研合同和其他法律文书的制约，本论文的科研成果属于成都信息工程大学。

特此声明！

作者签名：

2018年 5 月 9

1. [↑](#footnote-ref-0)