落笔洞传说的游戏设计与实现

王　明
三亚学院教务处　海南　三亚　572022

摘　要：落笔峰又名印岭，高不过110米虽有些耸秀，但没有山的宏伟，说它是是座丘可形状却有些山的独特，在一马平川的大陆架上，此丘，远远望去，却象古王朝一枚硕大无比的玉玺，置放在案头。印岭，由其形而得名。落笔峰是由石灰岩组成的，其形自然有些奇妙，尤如桂林一带的山。即为峰，便有垂崖如削、峭壁如劈、石猴嬉耍之怪状；即为峰，便有古树虬石、百花攀崖、百鸟欢唱、白云缭绕之美景。“落笔生辉”，便成了古崖州八景之一。本游戏是一款在Unity平台下使用C#语言进行开发，并以“落笔洞传说”为背景的游戏。

关键词：横板过关；刷怪；Unity；Visual物理引擎

海南拥有许多古城古迹、历史名人故居、仿古复古建筑等，非常有利于打造遗迹旅游文化。比如，崖州古城、东坡书院、五公祠、海瑞故居、丘浚故居、宋庆龄故居、骑楼老街、南洋建筑等，能够成就独特的遗迹旅游文化。文化与互联网的融合发展成为一种必然趋势。一方面，互联网是重要的文化传播手段，特色文化作为地区名片，借助互联网进行文化展示、加速传播，从而积极扩大旅游目的地的影响、提升综合旅游形象，在客观效果上是一种特定的旅游目的地营销；另一方面，古迹文化是互联网大数据的一个重要热点，旅游目的地借助互联网可以整合地区旅游精品，扩大智慧旅游知识服务范围。此时，海南不同特色文化能充分彰显旅游活动最本质的内容，自然成为各类游客通过互联网最迫切了解与关注的内容。

海南与大陆隔海相望。导致大陆文化对海南的影响一直不明显，海南人民有着具有明显低于特点的文化思想意识，在民族文化的不断交融中变化着直到现在，这片土地所形成的文化中充满了历史与时代印记的神话传说，比如：大力神传说、鹿回头的传说、五指山的传说、落笔洞传说等等。随着工业化和城市化进程的步伐加快，社会生产与生活方式的改变，人们正在淡化传统的文化和遗产，海南亦是如此。由此，海南文化的宣传显得尤为重要。游戏作为工业化过程中完全符合现代艺术、符合当代年轻人审美的产物，将它和传统的海南传说和文化融合在一起，使得海南的文化能在年轻人之间传播，很有积极意义。

1 系统相关技术概述

1.1 开发平台介绍

Unity3D为开发者提供了大量的内置组件，比如最基本的Transform、Animation等；与物理引擎相关的Collider等；渲染组件MeshRender等。

Visual Studio：该平台主要以编写逻辑脚本为主要任务，编写的脚本主要有两类：一类是继承自Mono Behaviour，该脚本一般会挂在到游戏对象上实现游戏逻辑和交互；另一类是普通脚本，主要工作是作为工具类、管理对象类、以及程序框架的核心模块类对游戏模块进行模块化的管理。除此之外开发者能够通过它自带一套完善的Debug工具快速的定位到程序出现错误的地方予以修改。降低了开发者出错的频率。

1.2 开发相关准备

1.2.1 策划

策划人员需要针对游戏的系统、数值、剧情、战斗、核心玩法等不同模块进行详述，策划案直接决定了游戏的品质核心玩法，决定了玩家对游戏的好感度，策划是非常重要的部分。

1.2.2 程序开发

技术人员根据对应的策划案，完成UI、战斗、音乐音效、代码管理工具等不同模块的编写。

1.2.3 美术

原画师根据策划案，完成必要对应风格素材的制作，然后场景搭建师根据需求实现场景的搭建后，特效师会在已有的素材基础上做出对应风格的特效动画，然后由技术人员添加至游戏内部逻辑。

根据本游戏的需求首先设计一套合适的UI框架，尽可能的将游戏UI管理简化。

2 游戏系统设计与实现

2.1 游戏设计

2.1.1 玩法

主要场景为落笔洞的2D关卡，让玩家进入场景的第一时间就能知道场景的描述主题。场景较大程度还原了

落笔洞的周围环境，使玩家身临其境。

玩家拥有前/后/跳跃/二段跳等操作。且当玩家处于跳跃上升期和下降期时，能通过某种判断实现上跃和下落动画的过渡播放。玩家在触碰到怪物时，会触发相应的受伤动画，并且要产生短暂的不可受伤（无敌）状态。主界面有5个按钮及一个滑块。分别为：开始、加载、退出游戏以及静音、调节分辨率和一个调节BGM大小的滑块。进入游戏的UI界面主要分为两个部分：血量和得分。用来分别记录玩家的当前血量以及得分情况。场景内主要有樱桃和钻石两种物品，当玩家满血时，吃樱桃会加分，缺血时会加血，钻石则加分。场景中主要有两种怪物，会进行基本的行走行为，玩家与怪物碰上时，玩家会相应扣血。

2.2 游戏逻辑实现

2.2.1 游戏框架

单例类：为了确保一些管理器，如：UI管理器、音频管理器；一些特殊的类如：实体类的信息在整个游戏过程中对象的唯一性，于是将项目中的所有Model类和管理器均实现成单例模式。

BaseUI类：该类是所有弹窗或者面板的基类，该类通过重写MonoBehaviour类中的Awake方法，实现所有继承BaseUI类的派生类会自动添加CanvasGroup组件，而不需要手动添加。

CustomButton类：该类继承自官方默认按钮类Button，主要重写了鼠标点击以及鼠标移入方法，两个方法分别实现自IPointerClickHandler、IPointerEnterHandler接口，主要用来接受触发点击、移入触发事件。

2.2.2 人物移动操作

人物采用的RigidBody2D物理组件实现物理效果。通过Player Controller自定义组件以及CapsuleCollider设置人物的相关操作。移动用设置RigidBody组件中的velocity数值：rigid.velocity = new Vector2（moveX * moveSpeed，speedY）；跳跃rigid.velocity = new Vector2（rigid.velocity.x，jumpForce）；二段跳时，需要判断是否着地，通过Collider2D中的IsTouchingLayers方法判断玩家是否接触到“地面层”。没接触到则再次按跳跃键可实现二段跳。无敌状态的实现：当触碰到怪物时，会在Update方法内不停减少预设时间，直到满足时间小于0且变为可受伤状态。

2.2.3 怪物逻辑

怪物主要行走逻辑为：向左向右往返行走，主要通过预设单方向行走时间：float _oneDirectionMoveTime = 2f；通过每次朝当方向走记录时间_recordTime，当_recordTime > = _oneDirectionMoveTime或者_recordTime < _oneDirectionMoveTime时，改变布尔值_isRight，通过它去判断移动以及怪物精灵片的朝向。

2.2.4 游戏的存档、读档、删档

游戏主要保存了三个方面的数据：游戏内人物的位置、宝石以及樱桃在场景中的位置、怪物的生成位置。数据的读取和写入封装在一个工具类DataUtility中。

2.2.5 音频管理器

游戏中存在一个声音的监听和两个音源：音效音源和背景音乐音源。这两个类挂载的物体被设置为不销毁对象。切换场景时，两个AudioSource不会被销毁继续执行。两个脚本主要是对事件消息的处理，游戏中所有的按钮都会有移入和点击音效，因此每次都会调用事件处理方法。

关于声音的数据保存，由于数据类型较简单和数据量较小，因此使用Unity自带的PlayerPrefs类保存，系统使用键值对的方式 < dataName, Value > 保存数据，在尝试得到值时需要提前通过HasKey方法判断键是否存在。玩家在设置面板中每一次改变音效或者音乐的值时，都会触发AudioSource中onValueChanged事件回调，会自动将最新数据保存到PlayerPrefs中，在重进游戏是，Init方法会第一时间读取该值，并赋值。

2.2.6 过度场景

游戏在每次切换场景时，都会进入一段过度场景，场景会显示目标场景的加载进度，如果完成会自动切换目标场景。主要逻辑是调用SceneManager类中的LoadSceneAsync异步加载方法。进入加载场景后，会判断当前活动场景是否和预期一致，一致则开启异步加载协程方法。并一直等到异步操作完成。在次期间Update方法会持续更新场景的加载进度。进入Loading场景后，挂在该场景的脚本会读取DataUtility类中的SceneName获取数据，得到目标场景的值。

2.2.7 关卡加载

在每个关卡场景中都会挂载一个Load脚本，该脚本主要是第一时间初始化数据，和记录相关数据。每次在加载场景是有四种情况：1.有存档，加载场景存档场景一致；2.无存档，是否在第一场景；3.有存档，场景与存档不一致；4.无存档，不在第一场景。加载每个场景都会执行LoadDataUtility脚本。

2.2.8 飘文本

飘文本主要用来对于游戏中的频繁通知和反馈信息。由于飘文本的生存周期很短，并且每个飘文本的产生都是独立的。这两点导致飘文本的生成和UI面板的生成和隐藏并不能共用同一逻辑，因此需要额外写一个针

对于飘文本的管理器（FloatTextManager）。该类也是单例类，但是和UIManager最大的区别在于，飘文本管理器产生的对象不会被保存在字典中长时间保存以便后面使用。该类只存在有两个队列，一个用来表示存储当前显示在画布上的文本，还有会在显示中的飘文本有三个时，会将即时产生的文本缓存在该队列中，等待显示文本不足三个时，使其自动出列补充显示。飘文本生存时间和最大显示文本数均为常量。显示飘文本只有一个接口ShowFT。在创建出来后，文本对象会即时启动计时器，生存时间达到LITE_TIME后自动销毁。

3 游戏分析

3.1 脚本

当点击开始按钮后，帧数会骤降，这里除了系统本身的性能消耗外，还有脚本LoadDataUtility的初始化耗时，接近0.2秒。主要原因分析：每次加载场景时，都会自动初始化一次当前场景的数据。所有的数据均保存在本地文件中，这时会产生三个操作：读取文件；反序列化；玩家、怪物、交互物品的实例化。其中IO操作和反序列化过程均比较耗费时间。

3.2 内存消耗

在加载完目标场景前后，场景中的游戏对象和类对象明显增加，主要因为加载了多个怪物及交互物体对象；加载完场景后一段时间内并没有出现未知的游戏对象增多，可以大致判定游戏并不存在严重的内存泄露问题；加载完场景前后可以看到GC对内存进行了一次回收，主要是对上一个场景的游戏对象的回收。

3.3 UI

在加载完成前后，能够看到ShowCanvas下的子物体明显增加。因为UIManager管理原因：所有加载的显示的物体都会被创建在ShowCanvas下。而HideCanvas由于暂时并没有隐藏的UI项，所以其子物体为空。

4 结论

经过游戏的测试，对存在的不足进行了修改，主要的修改部分：游戏面板的重新规划（引用变量方式的统一、面板命名的统一）；游戏代码结构的优化，主要包括：通过游戏明确游戏需求，对应的实现游戏的功能模块，如：事件中心、管理器的使用（声音/音效管理器、UI面板管理器、飘文本管理器）、实现各种工具类用以提高开发效率和统一实现相近功能时的实现方式。目前，整个游戏形成了一个小的框架，有着一个完整的功能逻辑，拓展方便。

参考文献

[1]宣雨松.Unity 3D游戏开发（第2版）.人民邮电出版社,2018.9

[2]吴亚峰,索伊娜.Unity 5.X 3D游戏开发技术详解与典型案例.人民邮电出版社,2016.2