阳 光 学 院

本科毕业设计(论文)

题　　目：基于Unity引擎的3D坦克大战游戏开发

院(系)别：人工智能学院

专　　业：软件工程

年　　级：2018级

学　　号：182730107

姓　　名：陈宇轩

指导教师：林世平（签名）

2022 年　　5 月　　10 日

基于 Unity 引擎的 3D 坦克大战游戏开发

摘要

随着智能终端的发展和普及，使用智能设备的用户群体越来越广大，大众对游戏产品的需求也日益提高。当前游戏产业从最初小众产业也发展为现象级产业。以往的智能设备用户主要为成年人，现在网络设备的使用门槛越来越低，游戏市场也慢慢扩展为全年龄向。相比于已经高度成熟化的成人游戏市场，面向青少年市场的游戏需求长期被忽略。开发一款面向 8 到 14 岁青少年的游戏产品变得十分有必要。本课题设计并实现了一款基于 Unity 引擎的 3D 坦克大战游戏 。

根据当前市场的主流情况以及所掌握的专业知识技能，课题使用 Unity 游戏引擎作为主要开发编辑器，Unity-3D 作为游戏框架，游戏内置逻辑实现通过 C#语言编写，使用 UGUI 进行 UI 界面的布局和搭建，游戏模型和美术资源则部分采用 Asset Store 商城中开源资源，部分使用 blender 建模软件进行搭建。游戏项目版本控制器通过 Git 进行托管和控制。毕业设计主要工作包含以下方面：

在游戏需求和设计上对其进行分析，将课题需求分解为角色，物理碰撞，游戏流程控制，控制角色操作中，使用 MovePosition 来修改角色刚体位置实现角色的水平移动和旋转。使用滑块组件作为角色血量，冷却 CD 等信息的显示组件。在物理碰撞检测中通过调用 OverlapSphere 获取碰撞体数组来实现获取碰撞体，使用 OnTriggerEnter 方法来实现物理碰撞更新，防止过度频繁更新导致性能损耗。实现对游戏性能更好的优化。为实现更逼真的游戏效果，游戏中还使用了 AddExplosionForce 用来模拟炮弹爆炸的冲击波推力。在游戏流程控制中，由于协程的特性很适合进行游戏流程的控制和管理。因此在游戏总流程中使用协程控制游戏一步一步执行。游戏内暂停倒计时也采用协程进行编写，相较于传统的逐帧更新，同样优化了游戏性能。

3D 坦克大战《TANK BOOM ！》游戏拥有两个可操控角色，两个地图场景可供游玩。游戏操作易上手，规则简单，针对于双人博弈，界面简洁有设计。玩法设计中有炮弹冷却，发射蓄力，炮弹冲击波等体系。丰富了游戏内容，提高策略性和博弈性。通过系统测试后，课题中的功能和需求在游戏中得到了很好的体现和完成。

关键词：　坦克大战　3D 游戏　双人博弈

目 录

1 绪论

1.1 引言

为体现游戏以人的核心思想，达到玩家益智益脑，放松身心，增进感情的目的，课题使用 Unity 引擎打造一款双人对战类博弈射击游戏，通过玩家之间的对游戏角色的操控和策略性的博弈来实现游戏的趣味性，使玩家享受游戏博弈的乐趣，放松身心[1]。

当前双人博弈游戏仍是相对冷门的产品，游戏产品市场上主要被各种棋牌类游戏和拟人类游戏所占据。《TANK BOOM ！》作为一款双人对战的坦克射击游戏，没有棋牌类的赌博元素，也避免了使用人形角色的带来的血腥暴力等问题。使用坦克作为游戏角色进行对战使其更加符合我国国家相关法律规定。坦克作为从第一次世界大战就被使用的经典武器，其悠久的历史和数不胜数的影视作品，使其被广泛大众所熟知。也作为游戏元素出现在无数的游戏当中[1]。采用坦克作为游戏玩家操作角色会更加贴合广大玩家群众的习惯和认识。

在多人游戏中，玩家渴望的是获得与人博弈的乐趣。《TANK BOOM ！》作为一款双人游戏，在保证公平的前提下，创新性的加入了其他游戏玩法，使得游戏更加具有创意和策略性[2]。在玩家操作坦克射击时存在炮弹发射冷却时间，只有冷却时间结束才能发射下一发炮弹，使得玩家在每次发射后都面临一段时间的无法再次开火的空窗期，在没有把握的情况下是否开火成了玩家相互博弈与思考的问题之一。

1.2 研究内容和实现功能

开始着手研究和开发该项目之前，首先进行了类似课题的研究和其他学习资料的查寻，借阅了一些与 Unity、游戏设计相关的书籍来进行学习。最终定下游戏内容和游戏基本设计方案[3]。

玩家可进行操作和控制的功能如下：坦克的移动和方向的旋转，发射炮弹，记录炮弹发射 CD，根据蓄力情况基于炮弹速度。查看自身血量和发射冷却时间，游戏内暂停与恢复，地图的选择等[4]；

游戏内游戏管理脚本实现的功能如下：游戏内状态的顺序控制，显示场景和地图，自动生成玩家坦克，记录玩家操作信息并反馈。比赛分数的统计，玩家对局信息的文本显示。不同地图的改变和显示，如夜晚地图将显示路灯，坦克车灯，及坦克的轮廓光。

项目开发期间也使用到 Git 作为游戏版本控制管理器，用来恢复错误分支，保证游戏版本正确性，规避由未知 BUG 引起的返工等一系列问题[5]。

1.3 论文结构

本文详细研究和分析了基于 Unity3D 引擎的坦克大战游戏开发设计与实现的过程。整篇论文的结构安排如下：

第一部分：介绍课题研究的意义及目的。

第二部分：对 Unity、C#、blender、Git 等基本工具和框架。

第三部分：对游戏设计理念和需求分析。

第四部分：介绍项目整体的游戏和玩法的设计。

第五部分：实现游戏功能模块的关键代码和实机界面的展示。

第六部分：系统的测试。

2 相关开发技术

2.1 UNITY(游戏引擎)简介

Unity3D 游戏引擎是一个十分全面的游戏引擎，界面简洁、功能强大，可以轻松创建 2D 游戏、3D 游戏、3D 可视化建筑和实时的三维动画等内容。所见即所得的方式极大的方便了场景设计人员。此外，Unity3D 引擎的跨平台特性也是一大亮点[1]。可以更好的开发 PC 端，Android 端，IOS 端。Unity 引擎有自带的 UI 系统框架 UGUI 来实现界面布局等交互功能。和物理模拟系统和基于触摸屏、作为一套完成的游戏开发的方案，拥有可视化编辑和其他基础的框架，使得游戏开发者能够轻松实现游戏构想[5]。出色的代码编译极大的提升了游戏性能[6]。

同时 Unity 也拥有丰富完整的生态，拥有自己的技术文档和官方培训课程，全球的开发者可以共同在 Unity 的技术论坛中畅所欲言，交流个人开发心得，为其他开发者提供创作灵感[7]。

2.2 C# 简介

C#语言是一种十分成熟的编程语言，是微软公司发布的一种面向对象的编程语言[8]。C#拥有常见的接口，继承等关系[8]。C#可以用简介优秀的代码开发程序。在 Unity 引擎中，绝大数开发者使用 C#作为开发语言。

2.3 Blender 简介

Blender 是一款免费开源 3D 图形图像设计软件， 内置的实时 3D 建模游戏引擎，使得可以进行一些物体的 3D 建模并用于游戏创作[9]。

2.4 Git 简介

是一个开源的分布式版本控制系统，可以有效、高速地处理从很小到非常大的项目版本管理[10]。作为一种代码托管技术，很多代码托管平台也是基于 Git 来实现的。Git 做到例如代码的版本控制，分支管理，版本合并等功能。基于其 Git 强大的功能，开源的特性，以及协同合作，相互成就的软件开源设计理念[11]。也因此诞生了如今世界最大的程序源码交流网站—GitHub。

3 游戏设计理念和需求分析

本章节主要描述了游戏设计理念和游戏设计思想，详细介绍游戏规则和玩家与游戏的交互内容，以及游戏应该实现的基本系统。3D 坦克大战—TANK BOOM ！是一款基于 PC 平台，目标为使青少年玩家与好朋友进行坦克对战，增加友情，轻松益智。利用游戏丰富的设计和规则使得游戏充满新鲜和刺激，非拟人化的角色使得玩家可以放开手脚进行博弈[13]。

3.1 游戏模式与设计

游戏有一个开始界面使得玩家进入游戏。拥有不同地图的选择，丰富游戏内容。坦克对局进行后自动生成，炮弹的发射和碰撞判定。对局胜利和积分系统，游戏中断功能等。

下面对游戏内需求功能进行详细的介绍以及设计方案：

3.1.1 基本游玩模式

（1） 进入游戏开始页面，点击开始按钮进入游戏地图选择页面，点击退出游戏则关闭程序。在选择地图中选择地图后进入游戏对局。

（2） 进入对局后，首先显示对局信息，然后系统自动在出生点生成双方玩家操纵角色。短暂等待后玩家获得角色操控权，通过键盘输入指令使坦克进行前后移动。

（3） 坦克生成后有固定属性，如血量，蓄力条，冷却条等属性。并且会在对局中数值和 UI 也会动态发生改变。

（4） 玩家将通过键盘控制坦克发射炮弹对敌方造成伤害，造成的伤害导致敌对玩家血量清零后获得本场游戏的胜利。

（5） 游戏进行中可以随时中断暂停，在暂停界面可以进行继续游戏，游戏重新开始，退回主页面等操作。继续游戏后经过等待时间后返回游戏。

（6） 当有一方玩家血量降为 0 后，判定该玩家失败，回合结束。获胜一方获得得分，失败玩家不获得积分。同时统计玩家双方得分情况并在屏幕中央显示。

（7） 当有一方玩家获胜回合达到规定回合数后，则比赛结束，显示出最终比赛信息和最终获胜玩家。同时提供重新开始选项和退回主页选项。

3.1.2 游戏模式设计

（1） 关于游戏开始页面，首先进入游戏显示一个欢迎页面和背景，背景播放游戏相关视频或图片，达到吸引玩家游玩的目的。在页面内可进行游戏地图的选择，游戏设计两款地图，使用同一地图资源，分别为白天地图和黑夜地图。通过调节地图光照来进行区分。达到同样的地图和资源，不同的体验。

（2） 关于坦克移动方面，与传统的 2D 坦克大战相比，3D 坦克大战引入了空间概念，但是处于对游戏难度的考虑和现实的实际效果，以及拟采用相机视角为 2.5D 正交视角。坦克只进行 X，Y 轴方向上的移动，因为高度的改变在正交镜头下不明显，且对战斗中玩家相互判断敌方位置和炮弹落点不友好，故并不进行对坦克的高度的改变。

（3） 关于坦克属性设计，则包含血量，设定基础血量为 100。收到伤害时则扣除血量，血量为 0 时坦克死亡。同时引入了蓄力条设计，当蓄力时间越久，坦克的射程也越远，蓄力的量则有滑块 UI 直接反馈给玩家。坦克还有炮弹冷却设计，每当坦克发射一枚炮弹后，则会进入冷却期，期间无法再次发射炮弹，当冷却时间过后则可以再次发射，冷却时间同样将以滑块 UI 反馈给玩家。

（4） 关于炮弹发射与碰撞反馈，由于引入了创新的 3D 模式，相比于老款的 2D 坦克大战，炮弹发射处理为直线，3D 坦克大战考虑到游戏难度以及面向群体，决定将发射出去的炮弹为抛物线，这样会带来一下好处也会规避以下问题：

1） 不采用直线设计更贴近于现实战场，符合 3D 世界逻辑。现实世界中打出的炮弹也多为抛物线，使得玩家更有身临其境的感觉。

2） 由于场景中存在一些建筑物，建筑物有高有低，如果使用直线碰撞则大概率优先撞击场景模型而非敌对玩家，造成游戏体验下滑，使用抛物线弹道则在炮弹飞行中途达到最高点，规避掉低矮的建筑同时保持炮弹一定高度，防止炮弹提前发生碰撞。

3) 由于游戏引用了蓄力系统，当长时间按住开火键时则会给予炮弹更大的力 ，如果采用直线发射则蓄力系统只能改变炮弹速度，远近对炮弹没有意义，为保留此重要玩法，最终选择抛物线弹道炮弹作为设计目标。抛物线弹道在蓄力后可以使其飞行距离相对较远但又合理。

小结：

使用抛物线炮弹弹道有以上好处，同时也带来一个问题，即抛物线炮弹如何判断命中玩家，在代码实现中，通过给炮弹设定了一个爆炸范围，使得命中坦克造成伤害的难度大大降低，不再需要刚好发生碰撞，只需要在碰撞范围中即可根据碰撞中心和坦克的距离造成比例性伤害。

（5） 关于出生点的设计，在地图中合理位置布置出生点，回合开始时则获得出生点的位置，旋转等信息，并克隆生成的坦克上面。

（6） 关于暂停页面的设计，也有诸多以下考虑，暂停应该可以在游戏运行中随时介入，随时中断游戏进程，随时暂停游戏状态，并且在暂停结束后存在计时器来进行游戏暂停结束的缓冲，使得游戏双方都有反应时间，提高公平竞争性。

（7） 关于结算页面的设计，应该在对局最后结束时统计积分，当达到游戏要求时则产生胜利者，最终显示到大屏幕上，同时包含再来一局选项以及重新开始选项。

3.2 可行性分析

项目采用的是 Unity 引擎作为游戏开发，Unity 本身集成了美术资源编辑，UI 界面搭建，以及具体功能和脚本的挂载。操作简单直观，同时集成多种自带编辑器，方便用户不使用代码，直接通过编辑器进行编辑和操作，脚本的参数也拥有可视化 UI 设计，方便用户快速对脚本参数进行调节，使得项目更加直观，便于理解。当前国内外 3D 游戏被虚幻引擎和 Unity 引擎所占据，期中轻量化小型游戏项目中 Unity 更是占据绝对的统治地位，成为行业标准，截至目前诸多大型公司的大型项目如腾讯旗下的王者荣耀，米哈游旗下的原神，均采用 Unity 引擎开发并平稳运行数年。大量使用 Uinty 引擎开发的游戏项目在游戏市场上经营。技术十分成熟稳定，同时也拥有活跃度较高的开发者社区进行技术和资源的分享。

4 项目模块设计

4.1 设计目标

目标为界面设计符合直觉，且在任何条件下允许玩家进行中断操作。游戏流程环节有逻辑感，即使玩家第一次使用也可以立刻上手没有障碍。移动操作方式和命中判断也符合美术设计和场景设计，符合直觉。实现手感舒适，简单易上的目标。

4.2 设计规范

在开发之前，以质量高、界面交互符合逻辑、更改性强，可扩展性作为开发的目标。在这开发的目标下，设计一套定制的游戏设计原则是非常有必要的。因此，在此次毕设研究开发中，规定了如下几条设计原则：

（1）每个界面都可以直接返回游戏主页，即任何时候都可以中断当前游戏。

（2）公平性原则，即双方玩家所用于的资源，属性极可能一致，保证游戏公平性。

（3）界面友好原则。有一个美观的界面，和符合逻辑的 UI 交互体验。

（4）游戏具有良好的扩展性。

4.3 系统功能设计

此项目将主体功能分为四个模块，分别为游戏开始模块、游戏中管理和控制模块、坦克控制模块、炮弹发射判断模块。

4.3.1 基本游玩模式

（1）进入游戏后，来到游戏主页，玩家可以选择开始游戏，进入地图选择界面。地图选择确定后进入游戏场景加载，游戏根据玩家选择的地图加载相应的场景资源以及开关一部分地图脚本如车灯，路灯，场景光开关控制。游戏开始流程图如图 4-1 所示：

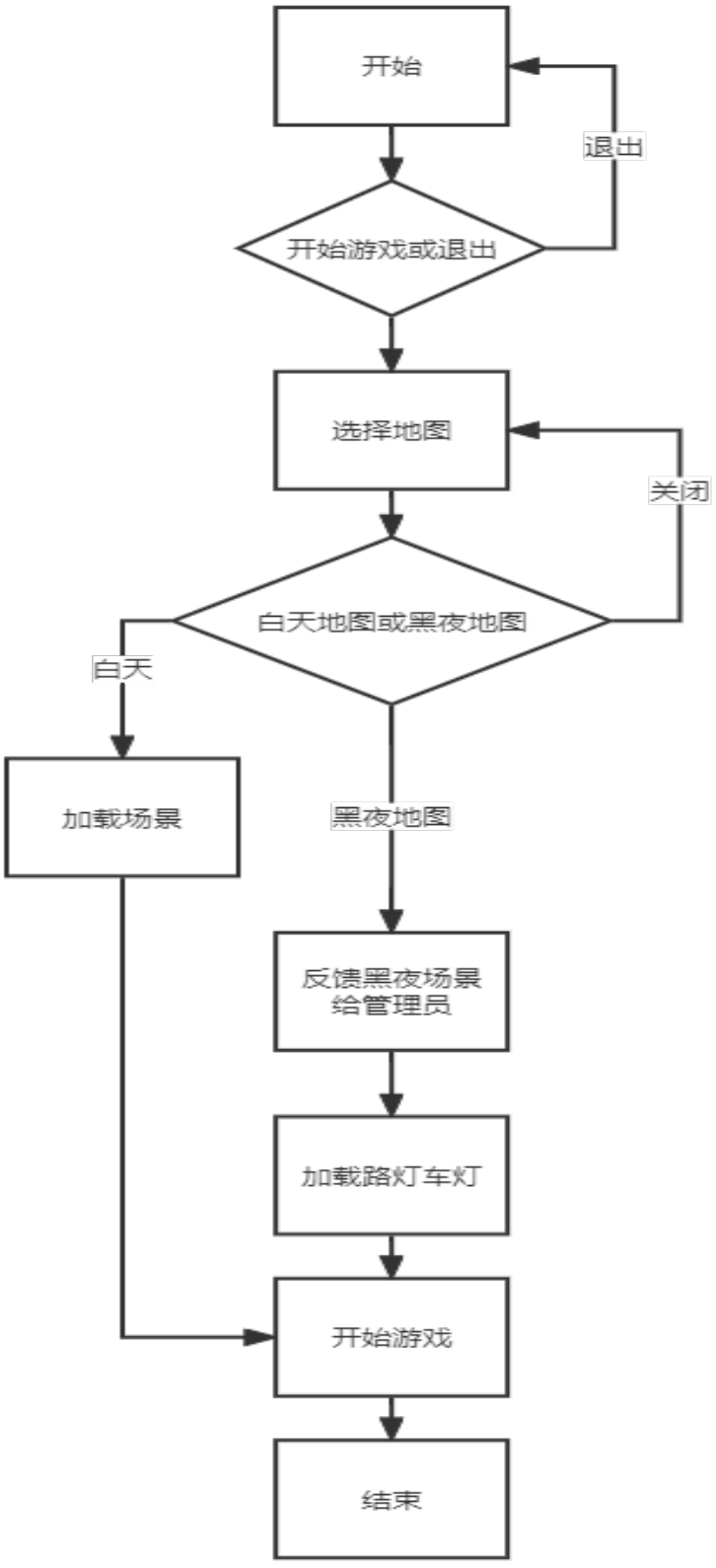

图 4-1 开始游戏功能流程图

4.3.2 游戏管理和控制模块

玩家进入对局后首先显示对局信息，玩家信息，比分信息等。在等待时间过后玩家获得坦克操控权，开始正式进入对局。对局中玩家可以随时中断和暂停。在暂停过程中玩家可以选择重新开始或者返回主页，也可以选择继续游戏。选择继续游戏则在等待时间过后继续进行对局。等待时间期间玩家无法进行一系列操作。对局结束后产生回合赢家，显示回合信息并给回合获胜玩家积分加一。若有积分达到比赛胜利条件者则显示比赛结算界面。管理和控制模块流程图如图 4-2 所示：

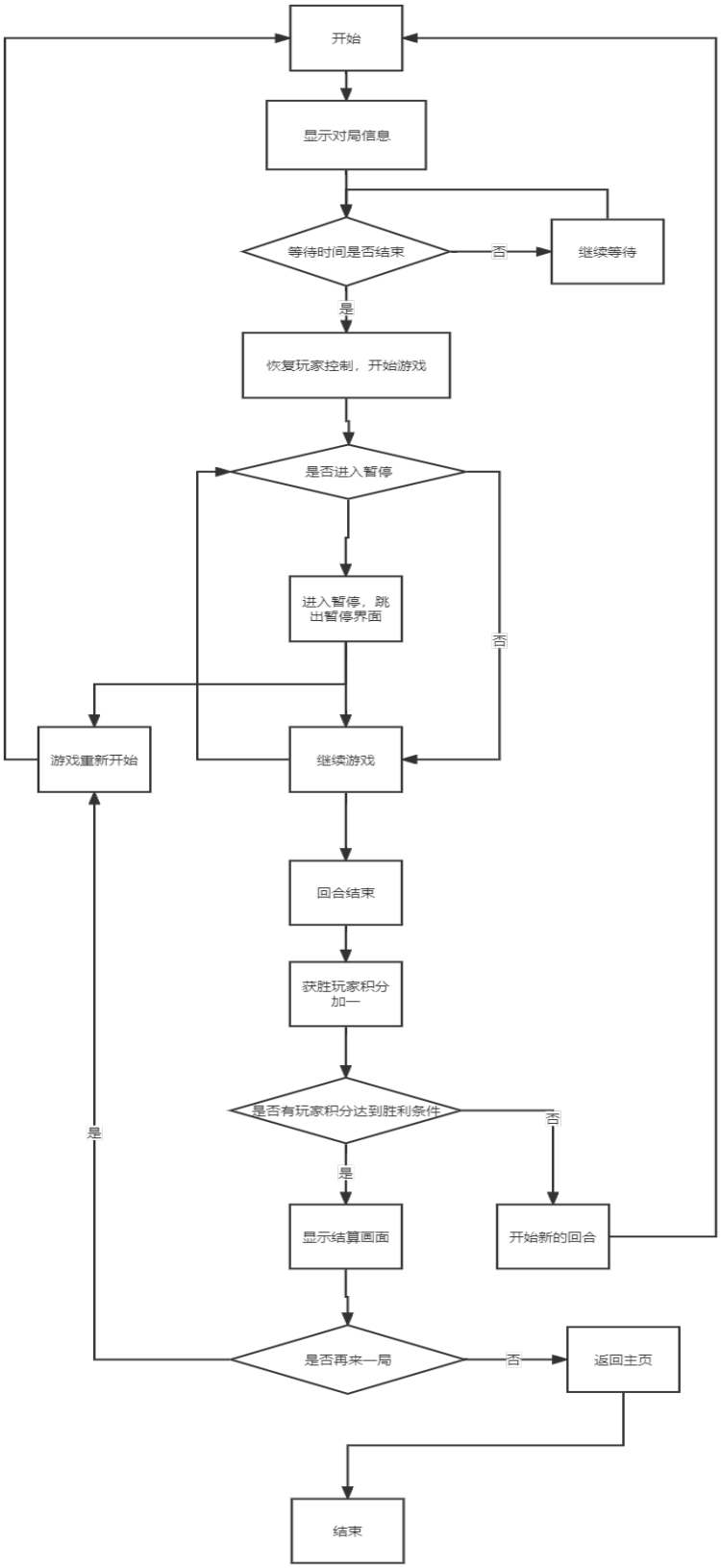

图 4-2 管理和控制流程图

4.3.3 坦克控制模块

开始游戏后查看坦克操纵状态，当前可操纵时则获取玩家操纵信息进行坦克操纵。当玩家按下开火键时，首先查看炮弹冷却状态，若有冷却则进行炮弹发射，同时冷却清空，炮弹进入无冷却状态。当炮弹再次拥有冷却时则可以再次发射炮弹。坦克控制模块流程图如图 4-3 所示：

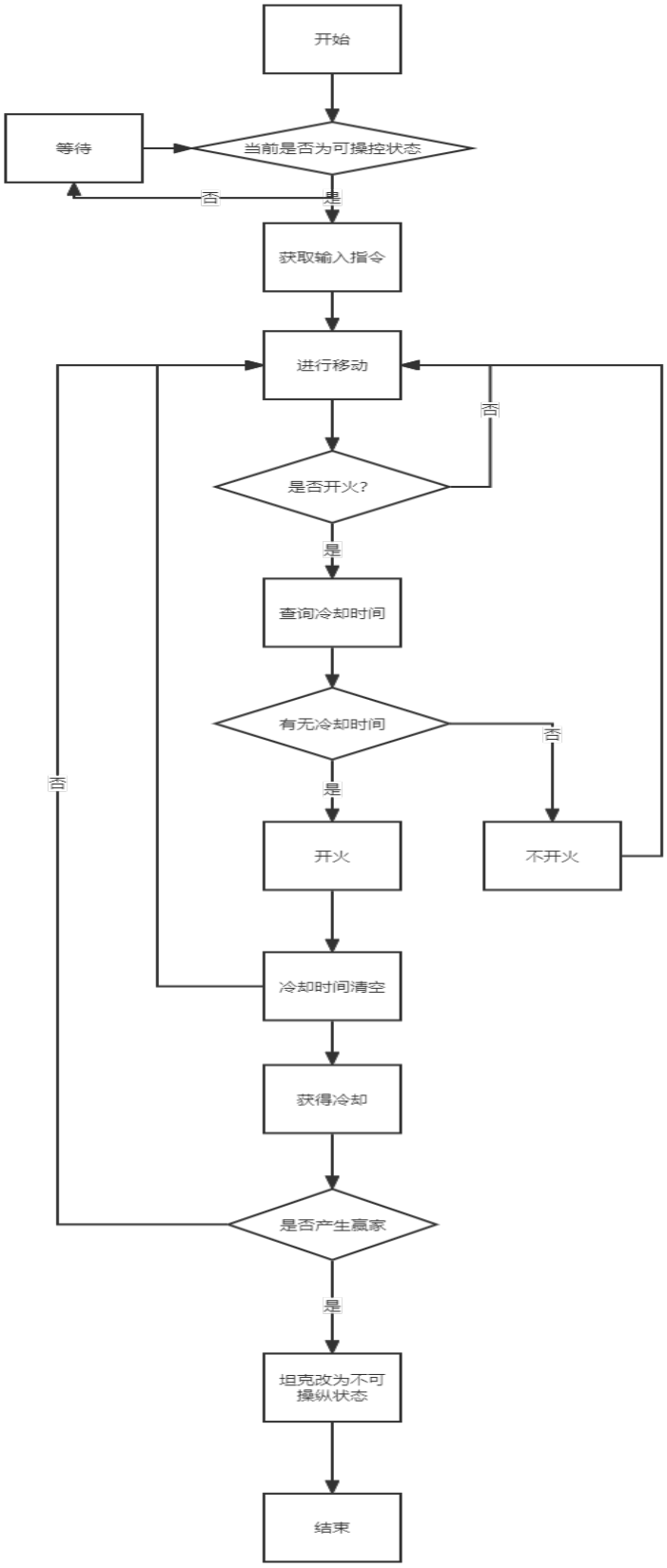
开始
等待
当前是否为可操控状态
否
是
获取输入指令
进行移动
否
是否开火?
是
查询冷却时间
有无冷却时间
否
是
否
开火
不开火
冷却时间清空
获得冷却
是否产生赢家
是
坦克改为不可操纵状态
结束

图 4-3 坦克控制流程图

4.3.4 炮弹发射和判断模块

按下开火键后，进行炮弹发射判断。炮弹发射后，获取炮弹伤害范围，检索炮弹伤害范围内是否有物体（Box Collider）[15]。若无物体则继续下一个物体判断，若有在玩家层（m_TankMask）的物体来判断是否有为碰撞体，若有则判断是否存在刚体（Rigidbody），获取玩家刚体后根据爆炸中心给予该刚体爆炸推力，同时根基爆炸中心和玩家的距离比例给予玩家伤害。然后播放粒子效果和音效，最后销毁炮弹。炮弹发射流程图如图 4-4 所示：

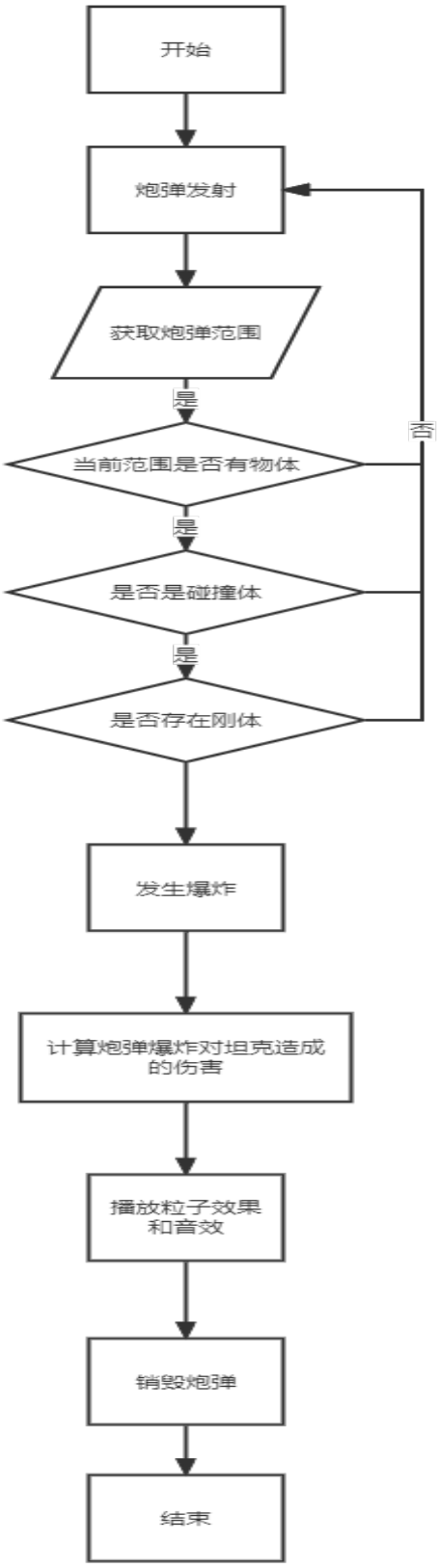
开始
炮弹发射
获取炮弹范围
是
当前范围是否有物体
否
是
是否是碰撞体
是
是否存在刚体
发生爆炸
计算炮弹爆炸对坦克造成的伤害
播放粒子效果和音效
销毁炮弹
结束

图 4-4 炮弹发射流程图

4.3.5 游戏暂停模块

点击 Esc 进入暂停，首先判断当前是否可以暂停。在游戏回合结束结束和开始期间无法进入暂停状态，仅当游戏正在运行中才允许进入暂停。进入暂停后进行可以进行选择，有返回游戏，重新开始，退回主页选项。再次按下 Esc 后则回到游戏，进入数字倒计时，当数字倒计时结束后恢复玩家控制，游戏继续。游戏暂停流程图如图 4-5 所示：

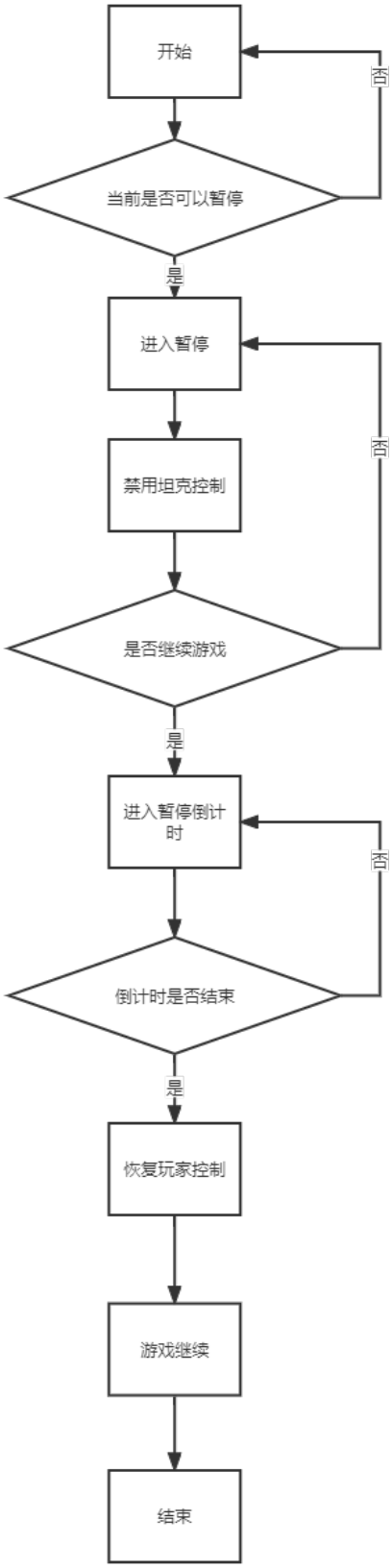
开始
否
当前是否可以暂停
是
进入暂停
否
禁用坦克控制
是否继续游戏
是
进入暂停倒计时
否
倒计时是否结束
是
恢复玩家控制
游戏继续
结束

图 4-5 暂停模块流程图

4.3.6 游戏结算模块

回合结束后则进入游戏结算，显示当前玩家积分信息，当有玩家积分达到获得比赛胜利条件后，则显示结算信息。结算信息内有按钮选项，为重新开始和退回主页。重新开始则重新加载当前场景，退回主页则回到游戏开始页面。游戏结算模块流程图如图 4-6 所示：

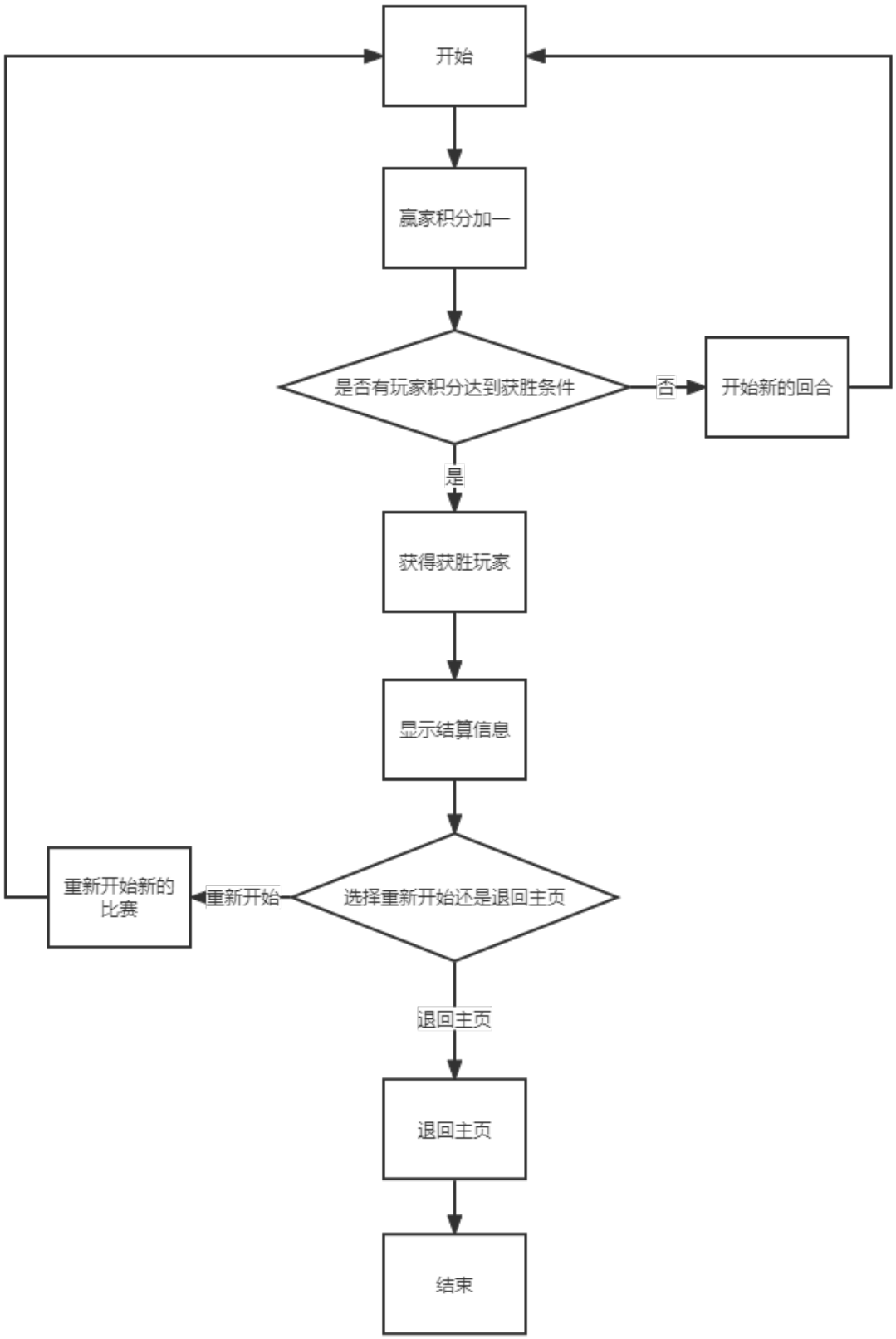

图 4-6 游戏结算模块流程图

5 系统实现

本章着重讲解该游戏项目中核心功能具体实现方式，代码细节。将上述项目需求和设计进行代码层面实现。

5.1 项目开发环境

（1）Windows 11 操作系统；
（2）Unity 编辑器；开发版本 2019.4；
（3）Visual Studio 2022；
（4）Blender 模型编辑器；
（5）Git 版本管理器：

5.2 开发工具

游戏项目代码方面采用 visual Studio 进行开发，由于对微软平台良好的支持性，导致成为开发运行于 PC 端的首选开发工具，同时 Unity 的主要脚本语言为 C#，使其在 VS 中也可以得到良好的运行。

5.3 运行环境

此项目需要在 Windows 系统 win10 及以上版本运行工程源文件

5.4 主要功能模块实现

项目主要实现模块如下

（1） 游戏开始界面：实现背景播放视频动画，有两个按钮，开始游戏和退出。点击开始游戏后则进入地图选择界面。

（2） 在地图选择界面选择好地图则开始游戏。地图模块拥有地形，建筑物，光照等基本元素。游戏管理员控制坦克在生成点生成，场景管理员获得当前地图编号来控制路灯开关。执行协程程序，游戏进入回合信息播放。期间禁用玩家操作，等待时间过后对局正式开始，恢复玩家操作。

（3） 坦克控制模块执行功能如下：获取玩家操作信息，进行坦克移动和发射炮弹。获取当前场景信息判断是白天还是黑夜，若是黑夜则开启车灯。

（4） 坦克 UI 模块则负责将坦克信息更新到坦克的 UI 滑块组件如血条，冷却 CD 条，蓄力值条。同时负责扣除坦克生命值并反馈给游戏管理员。

（5） 炮弹模块负责计算炮弹造成的伤害和反馈力，同时控制播放粒子特效和声音。

（6） 相机控制模块主要实现功能如下：始终保持双方坦克处于对称居中与游戏画面，同时当坦克移动导致中心点改变后平滑移动到新的中线点。当坦克过近时则保持最小距离使其画面不过小，同时设置边距使坦克不过分贴近相机画面边缘。

（7） 坦克管理员主要负责坦克预制体的上色和区分不同玩家的对坦克的控制，以及坦克脚本的启用和关闭。

（8） 游戏管理员模块主要负责实例化预制体，如坦克，地图，炮弹等并且联合坦克管理模块，坦克控制模块，相机控制模块进行统一的参数传递。同时执行协程程序负责游戏的开始，运行，结束。以及期间游戏文本信息的显示，控制游戏中断暂停。

（9） 场景管理员模块主要负责一部分 UI 通用方法的编写，如显示和隐藏，场景的加载，场景编号的获取，关闭页面，游戏地图加载，按钮方法等。

（10） 光照管理员主要负责黑夜模式下光照的控制，实现黑天白天的循环。

（11） 界面管理员主要继承场景管理员中方法，实现界面功能按钮的实现。

5.5 脚本模块实现

游戏内脚本主要实现功能如下：

5.5.1 坦克控制类

（1） Tank Movement （坦克移动脚本），控制坦克移动，引擎播放，粒子特效播放。在坦克激活初期采用 isKinematic 控制物体是否受物理影响，通过 Input.GetAxis 来获取玩家输入信息。其中改变坦克位置使用 MovePosition 将要改变位置的 Rigidbody（刚体）向 position（坐标）移动。MovePosition 可以记录力对刚体的影响，在后续炮弹爆炸后需要造成冲击力使得坦克移动，相对于 Translate 直接改变物体位置则过于突兀，后面还要对炮弹冲击波进行转换后才能改变坦克位置。故使用更改刚体位置方式来实现位置改变。并且在坦克的运动方向的改变上，由于角度要求值为四元数，于键盘输入值浮点型不符合，需强制转换为四元数，来改变旋转角度。坦克移动核心代码如图 5-1 所示：

```
//移动
1 个引用
private void Move()
{
    //移动的坐标值为    运动方向（前后方向or左右）*正负（前or后）*（速度*时间）即每秒运动speed距离
    Vector3 movemet = transform.forward * m_MoveValue * m_Speed * Time.deltaTime;
    m_Rigidbody.MovePosition(m_Rigidbody.position + movemet);//MovePosition   https://docs.unity.cn/
}
//旋转
1 个引用
private void Turn()
{
    float turn = m_TureValue * m_TurnSpeed * Time.deltaTime;
    Quaternion turnRoation = Quaternion.Euler(0f, turn, 0f);//强制将角度转换为四元数，（角度不支持浮点类型）
    m_Rigidbody.MoveRotation(m_Rigidbody.rotation * turnRoation);
}
```

图 5-1 坦克控制脚本

（2） Tank Health（坦克健康脚本）主要负责存储坦克的血量信息，更新血量信息，死亡后粒子特效的播放以及血量条 UI 的更新和显示。其中血量的更新使用 Unity 的滑块组件，通过更改滑块的值使得血量显示发生改变，引用 Lerp 线性插值使得血量在满血时设定值绿色，当血量逐渐扣除时颜色由绿色渐变为红色，过度更加平滑。坦克血条 UI 改变核心代码如图 5-2 所示：

```
// 将当前声明的值赋值给UI滑块。
m_TankHealthSlider.value = m_TankCurrentHealth;
//根据当前血量和满血的百分比，在选定的颜色之间插入条的颜色。
//Lerp线性插值  https://docs.unity.cn/cn/2019.4/ScriptReference/Color.Lerp.html

m_TankHealthFillImage.color =
    Color.Lerp(m_ZeroHealthColor, m_FullHealthColor, m_TankCurrentHealth / m_TankStarHealth);
```

图 5-2 坦克血条 UI 改变核心代码

（3） Tank Fire（坦克开火脚本）主要负责炮弹发射，蓄力状态下的 UI 控制，蓄力声音控制，炮弹冷却时间及其 UI 控制，炮弹爆炸粒子特效。发射炮弹通过在炮口期望发射炮弹处预制了一个 gameObject（对象），在实例化炮弹前引用其对象坐标给炮弹，同时获得炮弹刚体并根据 m_UpFireButtonValue 给与炮弹的刚

体一个力使其发射。同时清空发射炮弹 CD。坦克发射炮弹核心代码如图 5-3 所示：

```
2 个引用
private void Fire()
{
    // 使得状态改为发射了。
    m_FireShoot = true;
    // 创建一个子弹的实例，并将炮口的位置和旋转赋值给新创建的实例，并引用它的刚体。
    Rigidbody ShellInstance =
        Instantiate(m_Shell, m_FireTransform.position, m_FireTransform.rotation);

    // 设置炮弹的速度方向为坦克的前进方向。
    //velocity https://docs.unity.cn/cn/2019.4/ScriptReference/Rigidbody-velocity.html
    //  forward   https://docs.unity.cn/cn/2019.4/ScriptReference/Vector3-forward.html
    ShellInstance.velocity = m_UpFireButtonValue * m_FireTransform.forward;

    // 改变剪辑射击剪辑并播放它。
    m_ShootAudio.clip = m_FireClip;
    m_ShootAudio.Play();

    //距离上次发射时间清0
    m_FireTime = 0;

    // 重置发射按钮。这是一种预防措施，以防丢失按钮事件。
    m_UpFireButtonValue = m_MinFire;
}
```

图 5-3 坦克发射炮弹核心代码

判断当前蓄力状态以及炮弹冷却 CD 核心代码如图 5-4 所示：

```
Unity 消息|0 个引用
private void Update()
{
    // 滑块应该有最小发射力的默认值。
    m_Aim.value = m_MinFire;

    //距离上次发射小于发射CD时，距离上次发射时间值叠加
    if(m_FireTime < m_ReFireTime)
    {
        m_FireTime += Time.deltaTime;
    }
    else
    {
        m_FireTime = m_ReFireTime;
        //Debug.Log("Fire");
    }
    UpdateFireTime();
```

```
// 开火按钮刚刚开始被按下
if (Input.GetButtonDown(m_FireButtonName))
{
    //重置发射状态和发射力量。
    m_FireShoot = false;
    m_UpFireButtonValue = m_MinFire;
    // 播放充能声音
    m_ShootAudio.clip = m_ChargingClip;
    m_ShootAudio.Play();
}
//按住了射击键，而炮弹还没有发射
else if (Input.GetButton(m_FireButtonName) && !m_FireShoot)
{
    // 增加发射力并更新滑块。
    if (m_UpFireButtonValue≤m_MaxFire)
    {
        m_UpFireButtonValue += m_ChargeSpeed * Time.deltaTime;
    }
    else
    {
        m_UpFireButtonValue = m_MaxFire;
    }
    m_Aim.value = m_UpFireButtonValue;
}
// 发射按钮被释放，炮弹还没有发射,使其发射
else if (Input.GetButtonUp(m_FireButtonName) && !m_FireShoot)
{

    if (m_FireTime == m_ReFireTime)
    {
        Fire();
    }
}
// 如果超过了最大力，而炮弹还没有发射,且松开按键
else if (m_UpFireButtonValue ≥
    m_MaxFire && !m_FireShoot && !Input.GetButtonUp(m_FireButtonName))
{

    m_UpFireButtonValue = m_MaxFire;
    //距离上次发射时间==CD，可以发射
    if (m_FireTime == m_ReFireTime)
    {
        Fire();
    }
```

图 5-4 坦克发射炮弹核心代码

（4） shell Explosion (炮弹爆炸控制脚本)当炮弹爆炸时则执行，在检测的更新方法中选择 OnTriggerEnter，与常规的 UpDate 不同的是，该方法并非随帧更新，它的调用次数相对较少，和 FixedUpdate 相似，不过会直接返回和它发生碰撞的碰撞体。主要运行原理是当一个碰撞器进入到炮弹的 Caspule Collider(胶囊触发器)中时，则调用 OnTriggerEnter 方法，并返回该碰撞体。使用 OverlapSphere 获得炮弹所遇到的所有碰撞体，通过筛选层为只扫描 tank，使其爆炸冲击力只对 tank 生效。检测碰撞体核心代码如图 5-5 所示：

```
//碰撞事件一旦发生，则调用OnTriggerEnter方法，比FixedUpdate执行次数小 （如果另一个碰撞器进入到子弹触发器中，则调用）
Unity 消息|0 个引用
private void OnTriggerEnter(Collider other)       // Collider  所有碰撞体的基类
{
    // 收集子弹碰撞范围内的所有碰撞体，生成数组      OverlapSphere   https://docs.unity.cn/cn/2019.4/ScriptReference/Pl
    Collider[] colliders = Physics.OverlapSphere(transform.position, m_ShellExplosionRadius, m_TankMask);
                                                 // 爆炸中心              爆炸半径                在那些层查询(plays)

    for(int i = 0; i < colliders.Length; i++)
    {
        // 获取他们的刚体。
        Rigidbody targetRigidbody = colliders[i].GetComponent<Rigidbody>();
        // 如果没有刚体，继续下一个物体。
        if (!targetRigidbody)

            continue;

        // 添加爆炸的推力。 API addExplosionForce  向模拟爆炸效果的刚体施加力
        // https://docs.unity.cn/cn/2019.4/ScriptReference/Rigidbody.AddExplosionForce.html
        targetRigidbody.AddExplosionForce(m_MaxDamage*5, transform.position, m_ShellExplosionRadius);
                                          //中心伤害            位置                范围

        // 找到与刚体相关的TankHealth脚本。
        TankHealth TankHealth = targetRigidbody.GetComponent<TankHealth>();

        // 根据目标与炮弹的距离计算出目标应该承受的伤害。
        float amount = MakeDamage(targetRigidbody.position);

        //将伤害给坦克
        TankHealth.TankDamage(amount);
```

图 5-5 检测碰撞体核心代码

该脚本还负责计算爆炸造成伤害，由于伤害为球形伤害范围，距离爆炸中心即球心则伤害最大，需要一个方法来计算实际伤害，原理如下，当炮弹爆炸时，传入坦克坐标，获取坦克坐标和爆炸中心的距离，通过 Vector.magnitude 将坐标相减的坐标向量的向量长度。计算出坦克和子弹爆炸中心相距(爆炸半径)的

比例。和因此计算伤害核心代码如图 5-6 所示：

```
private float MakeDamage (Vector3 targetPosition)
{
    //定义一个坐标为 目标坐标-子弹爆炸点坐标              即目标和子弹爆炸点的差
    Vector3 ToTarget = targetPosition  - transform.position;

    // 计算炮弹到目标的距离。 (即向量长度) 爆炸距离
    float ExplosionDistance = ToTarget.magnitude;      // magnitude 向量长度

    // 计算目标和子弹爆炸中心相距(爆炸半径)的比例。
    float i = (m_ShellExplosionRadius - ExplosionDistance) / m_ShellExplosionRadius;

    // 根据最大可能伤害的比例计算伤害。
    float damage = i * m_MaxDamage;

    // 确保最小伤害总是0。
    damage = Mathf.Max(0f, damage);

    return damage;
```

图 5-6 计算伤害核心代码

5.5.2 相机管理类

（1） Camera Manager（相机管理脚本）主要负责相机跟随等控制，实现原理为获取玩家 x，y 轴上的中心点，当双方玩家位置发生移动后，不断更新当前中心点，使得相机画面中心始终等于 x，y 轴上的中线点，同时使用 SmoothDamp 使其平滑移动。通过此方法当游戏中仅剩一个坦克时同时会实现将获胜坦克放大居中等效果。相机移动核心代码如图 5-7 所示：

```
private void Move()
{
    //获取两个玩家的中心点
    FindAveragePosition();
    //移动到目标位置
    //当前位置  目标到达点  参考变量 (ref表示将要回到那个变量)  所用时间 Vector3.SmoothDamp  https://docs.unity.cn/cn/2019.4/Script
    transform.position = Vector3.SmoothDamp(transform.position, m_CameraTargetPosition, ref m_CameraMove, m_DampTime);
    //                                      当前位置              目标位置          平滑移动 (float)   多少时间完成
}
```

图 5-7 相机移动核心代码

在保证了相机始终处于双方玩家中心点时，当双方坦克组成的四边形发生改变时，则需要改变相机的 orthographicSize(正切角度（投影）大小)，使得画面放大和缩小，当双方坦克组成的四边形变大时，则 orthographicSize 变大以显示全部，同时也设定余量使坦克不完全贴近相机画面边缘，符合观感。控制相机正切大小核心代码如图 5-8 所示：

```
//获得一个相机的期望大小并返回 (size)
2 个引用
private float FindRequiredSize()
{
    //找到相机在空间中的移动位置            InverseTransformPoint  将 position 从世界空间变换到本地空间。
    Vector3 m_TargetPosition = transform.InverseTransformPoint(m_CameraTargetPosition);
    //从0开始相机大小的计算
    float size = 0;
    //遍历所有玩家
    for (int i = 0; i < m_Targets.Length; i++)
    {
        // 在相机的局部空间中查找目标的位置
        Vector3 playerPos = transform.InverseTransformPoint(m_Targets[i].position);
        // 从相机的局部空间的期望位置到目标的位置的差。
        Vector3 desiredPosToTarget = playerPos - m_TargetPosition;
        // 从当前的尺寸中选择最大的和坦克“向上”或“向下”距离相机。
        size = Mathf.Max(size, Mathf.Abs(desiredPosToTarget.y));
        // 从当前的尺寸和计算的尺寸中选择最大的，基于坦克是在相机的左边还是右边。
        size = Mathf.Max(size, Mathf.Abs(desiredPosToTarget.x) / m_Camera.aspect);   //m_Camera.aspect   宽高比（宽度除以高度）
    }

    //给出边缘的空白区域
    size += m_Blank;
    //相机最近不会小于的尺寸
    size = Mathf.Max(size, m_MinSize);
    return size;
}
```

图 5-8 控制相机正切角度核心代码

5.5.3 灯光管理

（1） Light Manager（灯光管理脚本）在游戏黑夜地图中，引入了白天与黑夜交替的概念。具体实现为当时间流逝时，调节 Light 组件中的 intensity（环境光）大小，来更改光照的强度，从而模拟日夜交替的效果，环境光控制核心代码如图 5-9 所示

```
1 个引用
void SunUp()
{
    Sun.GetComponent<Light>().intensity = SunValue;

    if (SunValue < 1 && ! SunLight)
    {
        SunValue += Time.deltaTime * SunMoveSpeed;
        if (SunValue ⩾1)
        {
            SunLight = true;
        }
    }
    if(SunLight)
    {
        SunValue -= Time.deltaTime * SunMoveSpeed;
        if (SunValue ⩽ 0)
        {
            SunLight = false;
        }
    }

}
```

图 5-9 环境光控制核心代码

5.5.4 坦克管理

（1） Tank Manager（坦克管理员）主要负责区分不同玩家序号，坦克上色等功能并且在游戏开始后初始化双方玩家坦克，使用 enable 启用和停用玩家坦克所挂载的脚本，初始化坦克核心代码如图 5-10 所示：

```
public void Setup()
{
    // 获取对组件的引用。
    m_Movement = m_Instance.GetComponent<TankMovement>();
    m_Fire = m_Instance.GetComponent<TankFire>();
    // 设置玩家号码，使其在脚本中保持一致。
    m_Movement.m_Playernum = m_PlayerNumber;
    m_Fire.m_Playernum = m_PlayerNumber;
    // 使用html富文本创建一个字符串
    //使得玩家代号颜色为玩家颜色
    m_ColoredPlayerText = "<color=#" + ColorUtility.ToHtmlStringRGB(m_PlayerColor) + ">玩家 " + m_PlayerNumber + "</color>";
    // 找到坦克的网格渲染组件。        renderers 为实例化后的tank的子对象的所有渲染网格
    MeshRenderer[] renderers = m_Instance.GetComponentsInChildren<MeshRenderer>();
    // 浏览所有的子模型网格Mesh并附上颜色
    for (int i = 0; i < renderers.Length; i++)
    {
        //将他们的材质颜色设置为该玩家特有的颜色。
        renderers[i].material.color = m_PlayerColor;
```

图 5-10 初始化坦克核心代码

5.5.5 游戏管理

（1） Game Manager （游戏管理员）负责串联所有脚本，维持游戏正常运行。通过 Ienumerator （协程）程序使得游戏按 RoundStarting — RoundPlaying — RoundEnding 顺序。Ienumerator（协程）程序代码如图 5-11 所示：

```
private IEnumerator GameLoop()
{
    //游戏的初始化
    yield return StartCoroutine(RoundStarting());

    //游戏运行中
    yield return StartCoroutine(RoundPlaying());

    // 回合结束的判定
    yield return StartCoroutine(RoundEnding());
```

图 5-11 Ienumerator（协程）程序代码

（2） 在 RoundStarting 中实例化并初始化 tank，禁用坦克操作，并显示回合信息，当实例化完成后则运行 RoundPlaying。RoundStarting 代码如图 5-12 所示：

```
1 个引用
private IEnumerator RoundStarting()
{
    // 这个功能是用来打开所有的坦克和重置他们的位置和属性。
    ResetAllTanks();
    //禁止玩家操作和启用tank功能
    DisableTankControl();

    //重置摄像机位置，初始化
    m_CameraManager.ResectCamera();

    // 设定回合数，每次回合开始回合数++
    m_RoundNUm++;
    m_Text.text = "回合 " + m_RoundNUm;
    //Debug.Log("star");

    // 等待指定的时间长度，执行下一个协程
    yield return m_StartWait;
}
```

图 5-12 RoundStaring 代码

（3） 在 RoundPlaying 中恢复坦克操作，并激活暂停判断程序.当按下暂停时弹出暂停界面同时禁用坦克操作，暂停恢复后则进入暂停倒计时，倒计时结束重新激活坦克操作。当场上坦克仅剩一个即回合存在胜利者时则运行 RoundEnding。RoundPlaying 代码如图 5-13 所示：

```
//游戏进行中
1 个引用
private IEnumerator RoundPlaying()
{

    // 游戏正式开始，让玩家控制坦克。
    EnableTankControl();
    //是否在回合结束计分状态 (false)     主要用于防止在计分状态暂停游戏出现bug
    IsEnding = false;

    // 清除屏幕上的文本。
    m_Text.text = string.Empty;
    //Debug.Log("playing");
    // 直到没有坦克
    while (!IsOnlyOneTank())  //只剩一个的时候执行接下来的协程
    {
        // 下一帧返回。
        yield return null;
    }
}
```

图 5-13 RoundPlaying 代码

（4） 在 RoundEnding 中重新禁用坦克操作，并进行积分统计，并展示玩家积分信息，最后重新交给游戏循环.若当前没有产生比赛赢家，则回到 RoundStarting，RoundEnding 代码如图 5-14 所示：

```
1 个引用
private IEnumerator RoundEnding()
{
    //防止结算中暂停    结算界面正在播放
    IsEnding = true;
    // 阻止玩家对坦克的操作。
    DisableTankControl();

    //清除前一回合的赢家
    m_RoundWinner = null;

    // 现在回合结束了，看看是否有赢家。
    m_RoundWinner = GetRoundWinner();

    // 如果有赢家，增加他们的分数。
    if (m_RoundWinner != null)
        m_RoundWinner.m_WinTime++;

    //现在胜者的分数增加了，看看是否有人攒够积分
    m_GameWinner = GetGameWinner();

    // 显示回合信息
    string message = GameMessage();
    m_Text.text = message;
    // 等待指定的时间长度，直到将控制权交还给游戏循环。
    yield return m_EndWait;
}
```

图 5-14 RoundEnding 代码

（5） 其中暂停功能代码如图 5-15 所示：

```
//控制游戏暂停
1 个引用
private void IsRoundSuspend()
{
    //esc被按下，游戏暂停
    if (Input.GetKeyUp(KeyCode.Escape)&& !IsEnding)
    {
        if (!Suspending)
        {
            //暂停坦克控制
            DisableTankControl();
            //SuspendText.text = string.Empty;
            m_SpendPage.SetActive(true);
            Suspending = true;
        }
        else if (Suspending)  //暂停中,准备结束暂停
        {
            m_SpendPage.SetActive(false);
            m_SpendTime.SetActive(true);
            DisableTankControl();

            //延时三秒实现--恢复坦克运动
            Invoke("SuspendEnd", 3f);
        }
    }
}
//结束暂停状态    (在 IsRoundSuspend 中延时三秒调用)
0 个引用
private void SuspendEnd()
{
    //恢复坦克运动
    EnableTankControl();
    //暂停状态改为false
    Suspending = false;
}

//是否在暂停中  (弃用，有BUG)
0 个引用
public bool IsSuspend()
{
    return Suspending;
}
```

图 5-15 暂停功能代码

（6） 暂停倒计时同样采用协程完成，暂停倒计时核心代码如图 5-16 所示：

```
IEnumerator CountDown()
{
    yield return StartCoroutine(CountDown1());
    if (!(Number == 0))
    {
        StartCoroutine(CountDown());
    }
    else
    {
        NumberText.text = string.Empty;
        Number = FirstNumber;
        Hide();
    }
}

1 个引用
IEnumerator CountDown1()
{
    Number--;
    if (Number == 0)
    {
        NumberText.text = "开始! ";
    }
    else
    {
        NumberText.text = Number.ToString();
    }

    yield return WaitTime;
}
```

图 5-16 暂停倒计时核心代码

5.5.6 场景以及 UI 管理

（1） ScenesManager（场景管理脚本）负责集成页面和 UI 管理方法，后续 UI 界面方法的调用采用继承 ScenesManager 来使用方法。其中主要方法有获取场景序号以及按钮方法，如图 5-17，5-18 所示

```
3 个引用
public int SceneNumber()
{
    //判断一下当前所在场景 (day or night)
    Scene a = SceneManager.GetActiveScene();
    // API  bulidIndex
    int m_SceneNumber = a.buildIndex;
    return m_SceneNumber;
}
```

图 5-17 获取场景序号代码

```
0 个引用
public void GameStar()
{
    MapChange.SetActive(true);
}

0 个引用
public void GameExit()
{
    Application.Quit();
}

0 个引用
public void DayMap()
{
    SceneManager.LoadScene(1);
}

0 个引用
public void NightMap()
{
    SceneManager.LoadScene(2);
}

0 个引用
public void Close()
{
    MapChange.SetActive(false);
}
```

图 5-18 按钮方法

（2） UI 管理脚本主要负责个界面的显示和按钮功能的实现，通过继承 SceneMangaer 脚本来调用界面通用方法，再挂载给各个界面给界面所含的按钮来进行功能实现。所有 UI 脚本如图 5-19 所示：

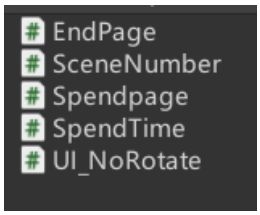

图 5-19 所有 UI 脚本

5.6 游戏流程展示

（1） 游戏开始界面

为游戏进入的启动画面，可以开始游戏选择地图和退出游戏。使用 UGUI 中 button 组件来搭建按钮模块供客户选择，背景视频则采用 video play 组件并设置循环播放模式实现进入游戏后开始视频播放。游戏开始界面如图 5-20 所示：

图 5-20 游戏开始界面

（2） 地图切换模块

显示正确地图则由用户选择来加载不同场景实现，通过调用 unity 中自带库 SceneManagement 的 buidIndex 获得当前场景编号用来告知当前显示脚本，以此来控制车灯开关。地图模块界面如图 5-21 所示：

图 5-21 地图模块界面

（3） 进入白天地图，显示回合信息如图 5-22 所示：

图 5-22 地图模块界面

（4） 游戏正式开始，坦克上方绿色方形长条为血量显示，下方环形条为炮弹 CD 显示，坦克前方为炮弹蓄力显示。如图 5-23 所示：

图 5-23 坦克 UI

（5） 游戏中点击 Esc 进入暂停，暂停界面如图 5-24 所示：

图 5-24 暂停界面

（6） 再次点击 Esc 取消暂停，进入暂停倒计时，暂停倒计时如图 5-25 所示

图 5-25 暂停倒计时

（7） 当一方玩家操控坦克死亡后，出现回合结算界面，回合结算界面如图 5-26 所示：

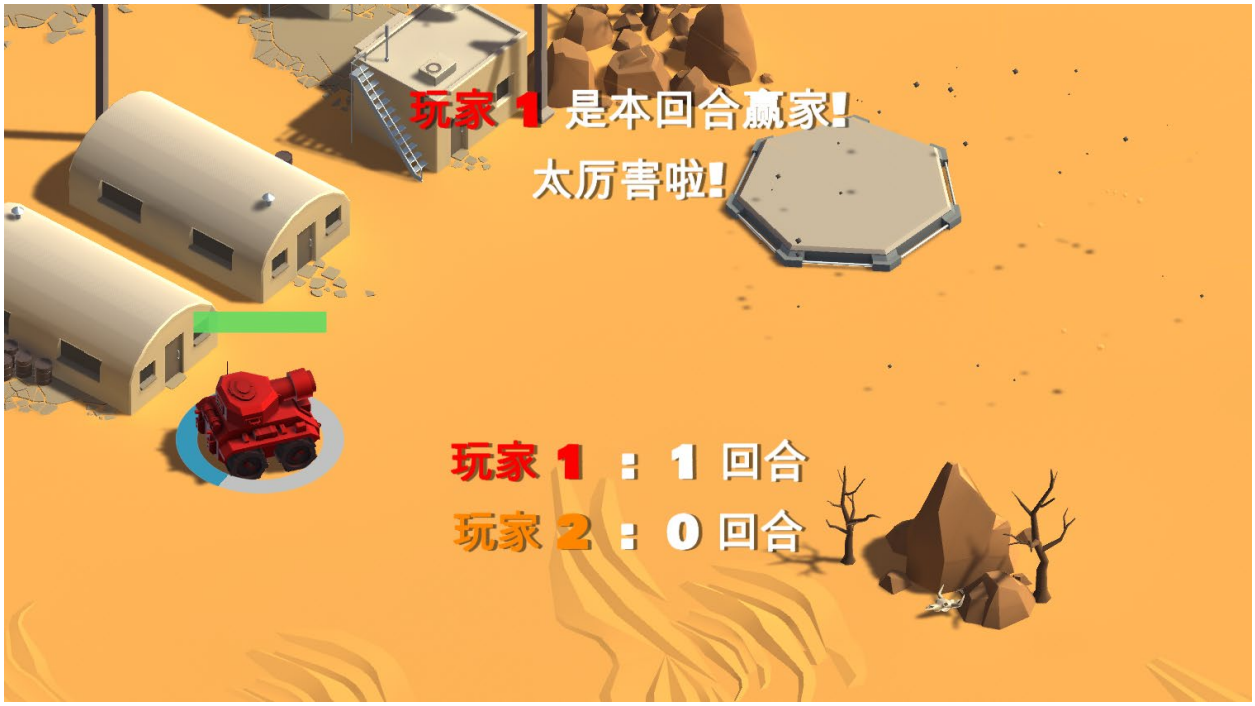

图 5-26 回合结算界面

（8） 当比赛一方获得足够积分获得比赛胜利时，出现比赛结束界面，如图 5-27 所示：

图 5-27 比赛结束界面

（9） 黑夜地图则加载黑夜场景，以及打开坦克车灯，顶光。其他同上，如图 5-28 所示：

图 5-28 回合结算界面

6 系统测试

项目测试是对项目成果的检验，测试在整个毕业设计中也占据这重要地位，只有通过测试才能发现隐藏的问题。

此次毕业设计的功能用例测试表如下表 6-1 所示：

表 6-1 用例测试表

系统名称	基于 Unity 引擎的 3D 坦克大战游戏开发		
用户类型	用户		
测试方法	通过大致的操作流程进行测试		
测试日期	2022-4-28		
Precondition	硬件配置	Window11 操作系统	
	软件配置	Unity 引擎	
	测试配置	PC	
操作过程			
No.	Input	Expected process	Test result
1	双击 TANK_BOOM!.exe 进入程序	成功进入游戏	ok
2	开始界面点击开始进行地图选择	跳转到地图	ok
3	进入地图后对坦克进行操作，发射炮弹	成功实现操作	ok
4	点击 Esc 进入暂停界面	成功暂停游戏	ok
5	再次点击 Esc 恢复游戏	跳转到暂停倒计时，之后恢复游戏运行	ok
6	击败敌方坦克	成功显示回合结算信息	ok
7	获得比赛胜利	成功显示比赛结算信息	ok
8	比赛结算信息点击重新开始	成功重新加载游戏	ok
9	选择黑夜地图	成功加载黑夜地图，正确显示路灯，车灯，轮廓灯。	ok

测试结果表明，游戏的功能已基本实现，游戏项目实现的功能已基本没有 bug，本次测试成功。

结论

本课题“基于 Unity 引擎的 3D 坦克大战游戏开发”设计并实现了一个运行在 PC 端的双人游戏。经过测试，此游戏能流畅的在 PC 端上运行。游戏流程遵循游戏设计。测试用户可以在游戏内容中获得乐趣，放松心情。以下是本课题的主要工作内容：

（1） 对 3D 双人坦克大战进行了需求及功能的分析，分析了系统的开发方法，完成了游戏玩法设计设计。

（2） 根据对游戏的分析和体系结构设计，使用 C#语言实现游戏各个模块和脚本逻辑处理。

（3） 通过 Unity 编辑器将资源和脚本进行整合，最终完成游戏项目。

（4） 对整个游戏项目进行测试，结果表明，游戏可以完成游戏设计内容并且没有出现 bug。

本次课题的创新之处在于双人游戏的扩展以及炮弹冷却 CD，蓄力系统，炮弹冲击推力的模式的添加。使得游戏拥有良好的竞技性和观赏效果。不过由于自身水平的限制以及时间的问题，项目目前还不够完善。比如在技术许可的条件下，还可以增加技能系统，继续增加游戏中变数，丰富其竞技性。还可以加入 AI 敌人，使得游玩模式可以切换为双人模式，单人模式，和双人对抗人机模式等，继续完善并提高游戏的可玩性，趣味性。

参考文献

[1] 郭东方. 基于Unity3D坦克战争游戏的设计与实现[D].河北科技大学,2018.

[2] 赵强.基于Unity3d的坦克大战游戏中的小技巧[J].数码世界,2018(10):58-59.

[3] 徐连霞.基于Unity3D的飞机大战游戏设计[J].数码世界,2019(09):64.

[4] 陈阳.基于Unity3D的游戏开发[J].电子技术与软件工程,2020(09):36-37.

[5] 郭欣桐.基于Unity的飞船大战游戏设计与开发[J].科技资讯,2021,19(11):67-69.DOI:10.16661/j.cnki.1672-3791.2103-5042-8166.

[6] 赵若津. 基于Unity的游戏功能模块设计与实现[D]. 山东大学,2020.DOI:10.27272/d.cnki.gshdu.2020.001620.

[7] 何小慧,廖晓芳,关韵婷.海豚保卫战 2D 游戏在 Unity3D 下的开发[J].电子技术与软件工程,2022(01):61-64.

[8] 唐迪. 基于Unity3D引擎的第一人称射击游戏设计与实现[D].电子科技大学,2021.DOI:10.27005/d.cnki.gdzku.2021.002547.

[9] 路宜松. 基于Unity引擎的2D角色扮演游戏的设计与实现[D].沈阳理工大学,2021.DOI:10.27323/d.cnki.gsgyc.2021.000043.

[10] 胡静,胡欣宇.基于Unity3D引擎的游戏设计与开发[J].电子元器件与信息技术,2021,5(02):138-140+154.DOI:10.19772/j.cnki.2096-4455.2021.2.064.

[11] 于万国,胡宗森,隋丽娜,迟剑,蔡永华,傅冬颖.新工科下软件工程专业实践案例构建研究——以基于Cocos2d-x引擎的跨平台游戏开发为例[J].计算机技术与发展,2021,31(02):191-196.

[12] 吴晓雪,何南,缪新颖,王魏.基于Cocos2d-x引擎的移动游戏设计与应用[J].现代电子技术,2018,41(24):106-109+113.DOI:10.16652/j.issn.1004-373x.2018.24.026.

[13] 刘梦颖,马宏琳.基于Unity的冒险游戏设计与实现[J].河南科技,2021,40(17):30-32.

[14] 胡静,胡欣宇.基于Unity3D引擎的游戏设计与开发[J].电子元器件与信息技术,2021,5(02):138-140+154.DOI:10.19772/j.cnki.2096-4455.2021.2.064.

[15] Kang Sung Yun,Park Ki Hong. Design and Implementation of Sleigh Game Content based on Physical Force using Unity3D Game Engine[J]. Journal of Digital Contents Society,2019,20(12).

致谢

二十载求学路将尽，行文至此，思绪万千，求学之路始于家乡，而今终于福州，一路行之如饮水，冷暖自知。

落其实者思其树，学其成时念吾师。课题开展前遇到过诸多困难，经历过诸多痛苦。期间较晚的去重新更改毕设题目和研究方向。有过迷茫，有过彷徨。幸得恩师林世平的肯定和耐心指导，对学生我更改毕设的事情鼎力支持。也感谢灵达老师和学校领导对我更改毕业设计的肯定。使得我更加相信自己，坚定自己的信念。无以为报，揖礼还授。今蒙受诸位老师之恩，方能做此文。经师易遇，人师难遇，微微寸心难报之。

年年遇人，人遇去去，遇遇至散散。忆同行之挚友，轻重自在心头。学生时代最后的时光，感谢有你们相伴。愿有岁月来日再聚首，故人相聚也自有方。

而今远行，故园仍在桃李灼灼春风里，余自当剥削而日参省乎已，力求知明而行无过，以报恩师家长栽培之情。

文毕，且祝诸君平安喜乐，万事顺意。