泉州师范学院
QUANZHOU NORMAL UNIVERSITY

毕业论文（设计）

题目 基于 CocosCreator 引擎的回合制对战游戏设计

学　　院	软件学院	专　业	软件工程（软件开发方向）
学生姓名	肖龙昊	学　号	18065589627
指导教师	于娟	职　称	讲　师
完成日期	2022 年 4 月 30 日		

教务处制

基于 CocosCreator 引擎的回合制对战游戏设计

软件学院　软件工程专业 183117044 肖龙昊

指导教师　于娟　　讲师

摘要

随着国内游戏市场快速发展，游戏也逐渐成为人们生活中重要的社交娱乐事物之一。其中，回合制游戏在游戏历史上有着显著的地位。无论是历史悠久的棋牌类游戏，或是如今全球盛行的自走棋类游戏，都不能缺少回合制的游戏机制。另外，随着《三国杀》、《炉石传说》等国内外知名卡牌策略游戏的逐渐完善，使回合制游戏在国内游戏市场蒸蒸日上，受到了众多玩家的追捧。同时，在游戏开发方面，CocosCreator 引擎作为目前主流游戏开发引擎之一，具有多种强大和便捷的接口和功能，使游戏开发的效率大大提高。本文选题结合目前回合制游戏如火如荼的现实背景和开发技术理论，基于全球游戏市场较为热门的几款回合制对战游戏，对游戏机制和玩法进行了创新；并对如何使用 CocosCreator 引擎进行开发展开了思考，对代码设计与实现中可能会发生的情况进行分析，最终完成一款机制较为完善且多样化的 PC 端回合制对战游戏。

关键词：PC 端；CocosCreator 引擎；回合制游戏

Turn-based battle game design based on CocosCreator engine

College of Software　Software Engineering（Software Development）　183117044　Xiao Longhao
Instructor Yu Juan　Lecturer

Abstract

With the rapid development of the domestic game market, games have gradually become one of the important social entertainment things in people's life. Turn-based games have a prominent place in the history of games. Turn-based game mechanics are indispensable for both the long-established board games and the worldwide popular self-propelled board games. In addition, with the gradual improvement of well-known turn-based strategy games at home and abroad, such as SGS and Hearthstone, turn-based games are flourishing in the domestic game market and are popular among many players. Meanwhile, in terms of game development, CocosCreator engine, as one of the current mainstream game development engines, has a variety of powerful and convenient interfaces and functions, which greatly improve the efficiency of game development. In this paper, based on several popular turn-based battle games in the global game market, the game mechanism and gameplay are innovated in combination with the current reality background and development technology theory of turn-based games. And how to use CocosCreator engine for development, the code design and implementation of the possible situation is analyzed, and finally complete a relatively perfect and diversified PC turn-based battle game.

Key words：PC; CocosCreator engine; Turn-based games

目录

第一章 绪论

1.1 研究的意义和目的

在当今电子设备实力愈发强大的同时，电子游戏的兼容性也随之提高，从而吸引了大量新玩家涌入游戏市场，电子游戏也由最初的娱乐消遣工具成为当下最受年轻人欢迎的社交工具之一。得益于电子游戏丰富的类型与有趣的玩法，对不同游戏特点进行抽象、组合、创新，往往会衍生出新型游戏模式，造就新一批的现象级游戏进入游戏市场，这也使得电子游戏能够在互联网历史上经久不衰、蓬勃发展。

游戏产业作为社会前景优越的产业之一，同时带动了如外设周边、游戏直播、电竞赛事等产业发展，近年来，游戏市场规模水涨船高，2021 年，中国游戏市场实际销售收入 2965.13 亿元，其中自主研发游戏占 2558.19 亿元，代理国外游戏已不再是增加国内游戏市场收益的主要途径，而是顺应国家政策进行自主研发，同时也为国内游戏研发行业提供了有效的人才支持与技术支持[1]。

电竞赛事，是指能够通过竞技的方式进行的电子游戏比赛，目前我国的电子竞技行业发展较为成熟，其紧张刺激的比赛节奏，极具观赏性的比赛画面收获了大批观众与支持者，结合国内游戏玩家对竞技类型游戏的喜爱，电竞赛事无疑成为了国内年轻人关注度最高的比赛项目之一。而电子竞技类型的游戏也广受中国玩家所热爱，目前国内电子竞技用户规模已达到将近 5 亿名，其中用户主要以中青年为主，且主要集中在一二线城市。随着 2021 年电竞入亚和多家中国游戏战队在世界赛夺冠的背景下，电子竞技也因此得到了社会的广泛认可[2]。

目前电子竞技游戏在中国游戏市场中拥有良好的前景，而电子竞技游戏类型主要分为 FPS（第一人称射击游戏）类、MOBA（动作即时战略游戏）类、TPS（第三人称射击游戏）类、RPG（角色扮演游戏）类以及 RTS（回合制战略游戏），近几年来，回合制战略游戏在电子竞技游戏领域逐渐站稳脚跟，因其操作简单，玩法及风格多变，考验玩家决策力和大局观的特点，广受全年龄段玩家的喜爱。本课题将通过研究国内外回合制战略游戏特点，开发并实现回合制游戏主流程玩法，拓宽回合制游戏的玩法与操作。

1.2 国内外研究现状

目前国内回合制游戏数量繁多，尽管游戏主题五花八门，但大部分游戏玩法单一枯燥、游戏性极其匮乏。不难看出，国内大部分所谓极具创新的回合制战略游戏，其实都是在旧游戏上进行“换皮”重置后再随意投入市场。《仙剑奇侠传》、《新绝代双骄》、《轩辕剑》等经典回合制游戏也逐步退出历史舞台，国内自主研发的新型回合制游戏已寥寥无几。

相比国内，国外回合制游戏市场则是百花齐放，大放异彩，由比利时游戏工作室 Larian Studios 开发的战旗策略类游戏《神界原罪 2》因其独特的回合制策略模式以及出色的战斗系统收获了全球游戏市场的青睐，在 Steam 全球销售量中，国内玩家占比高达 25%。作为经典回合制游戏《宝可梦》系列的诞生地日本，在回合制游戏研发上独具创新，游戏玩法丰富多样，具有较高的游戏性。由知名游戏发行商任天堂发布的回合制 RPG 游戏《八方旅人》在国内售价高达 400 元，尽管当时的国内市场不少 3A 大作最高售价也不高于 298 元，《八方旅人》令人望而却步的价格依旧无法抵挡玩家对回合制游戏的热爱，截至目前，《八方旅人》全球销量已 5250 万，更是广受各大玩家好评。

回合制游戏之所以能在全球游戏市场经久不衰，是因为回合制游戏的可塑性是极其可观的。谈及回合制游戏，如果没有切身去体验过多款回合制游戏，大部分玩家对回合制游戏的印象则是认为回合制游戏呆板、枯燥、不入流，这也是国内多款“换皮”回合制游戏给玩家产生的深远的固定思维。回合制游戏的上限，与对回合中各个阶段的操作成正比，每回合中影响战局的因素越多且越不稳定，其可操作性和可玩性也越高，而这也是回合制游戏大作中共有的特点，本课题设计实现的游戏将对以往的经典回合制游戏进行总结与创新，使玩家在不影响操作难度的同时可以对回合制游戏中的因素做出分析与利用[3]，并在战斗系统上进行创新，增加多种游戏状态与道具能够和游戏战斗产生联动，在提高游戏性的同时丰富了游戏的玩法。

1.3 主要工作

首先确定游戏主流程功能，结合主流程功能分析出所需要的系统功能，根据项目技术需求对部分细节功能进行调整。整理各个功能模块所需要的资源素材，完成对各个功能的 UI 制作，在保证功能完整的同时对 UI 界面进行美化。确定项目需要使用到的插件和工具，对插件的部分功能进行优化，以方便项目开发。对项目架构进行规划，对游戏各个功能模块进行划分，实现模块独立开发。最后对项目的数据模型进行补充完善，确定项目数据存储方式，并结合数据模型完成项目主流程开发。

1.4 论文结构

第一章绪论：结合当下时代背景交代该课题研究的意义与目的，对国内外课题相关案例进行总结与分析，提出本课题的创新性并阐述课题主要工作。

第二章相关技术概述：对本项目开发环境以及 CocosCreator 引擎的相关知识进行详细介绍，展示项目中使用插件等工具。

第三章游戏需求分析：对本项目进行综合性的可行性分析，进而对游戏各个功能模块进行确认，画出本项目用例图，用例规约和系统类图加以描述，判断项目实现风险及难度。并对本项目所需数据库进行建模，列出主要数据库表格。

第四章游戏设计：对本项目中各个功能模块进行分析设计，确认各个功能的流程，画出本项目主流程的流程图和时序图进行阐述。

第五章游戏实现：对本项目中所需要制作的 UI 及功能进行实现，展示项目逻辑代码与逻辑思路。

第六章测试结果和分析：对项目进行测试，逐步列出测试期望及测试结果，并对测试结果进行分析与总结。

第七章总结与展望：对本项目所完成功能进行总结，并对项目可优化点进行分析，阐述对项目未来的展望。

第二章 相关技术概述

2.1 运行环境

本次课题项目将会使用 CocosCreator 引擎进行构建发布，使项目能够在 Windows 平台完成运行与测试。项目使用 Sourcetree 完成项目的版本控制，保证开发的完整性和安全性。在开发工具上，本项目使用 Visual Studio Code 工具进行开发，开发语言为 TypeScript。通过 Photoshop 完成游戏 UI 素材的裁剪，并使用 TexturePackerGUI 对游戏 UI 图集的进行打包已释放资源，最后在 CocosCreator 引擎进行 UI 排版与事件处理。游戏相关数据则使用导表工具 excel-template 进行表格与 Json 的数据转换，从而实现数据处理。运行环境详见表 2-1.

表 2-1 运行环境

目标平台	Windows 操作系统
游戏引擎	CocosCreator3.4.2
版本控制	Sourcetree
数据存储格式	Json
开发语言	TypeScript
相关工具	Visual Studio Code
相关插件	Excel-template、simple-code

2.2 开发工具及相关技术

本次课题设计的游戏是由 Visual Studio Code 开发环境、CocosCreator 引擎、json 数据文件的结合使用下进行开发，游戏 UI 及动画由 CocosCreator 进行绘制，游戏算法由 TypeScript 语言进行开发，通过导表工具获取游戏数据。

2.2.1 CocosCreator 引擎

本次课题设计的游戏由 CocosCreator 引擎进行开发，引擎版本为目前最新版本 3.4.2，其项目结构包括 asset（资源目录）、build（构建目录）、library（导入的资源目录）、local（日志文件目录）、profiles（编辑器配置）、temp（临时文件目录）、packag.json（项目配置），其核心结构为资源目录，项目的源码、UI 素材、数据模型等资源都存储在资源目录中。asset 目录中的文件会在 CocosCreator 编辑器中的资源管理器窗口显示。

CocosCreator 是一款轻量、高效、免费开源的跨平台游戏引擎，其具有强大的编辑器功能[4]，可以使开发者能够实时预览与调试游戏，同时也对设计师友好，能够允许设计师深入参与到开发游戏的进程中。CocosCreator 还可以完美支持 Vulkan、Metal、WebGL、OpenGL ES 等负载均衡的多线程渲染器，还拥有基于 FrameGraph 的定制渲染管线和来自华为 CGKit 的移动端延迟渲染管线，使引擎性能得以大幅提升。且 CocosCreator 不但可以开发 2D 或 3D 等目前市场主流游戏，而且在 HMI、XR 等领域都拥有一套较为成熟的解决方案，引擎中内建了 Spine、DragonBones、TiledMap、Box2D 和 Texture Packer 等 2D 开发中间件的支持，能够有效降低开发难度，提高游戏开发的效率。Cocos Creator 深度支持各大主流平台，游戏可以快速发布到 Web、iOS、Android、HarmonyOS、Web、Windows、Mac，以及各个小游戏平台，让用户最大化游戏产品的可见度和成功概率。在 Web 和小游戏平台上提供了纯 JavaScript 开发的引擎运行时，以获得更好的性能和更小的包体。在其它原生平台上则使用 C++ 实现

底层框架，提供更高的运行效率[5]。

2.2.2 Simple-code 插件

simple-code，又称快闪-代码编辑器，是针对 CocosCreator 开发的一款能够提升游戏开发效率的插件，目前 simple-code 支持 CocosCreator3.0 以上的版本，其主要功能为可以在 CocosCreator 编辑器上通过点击节点来编辑绑定在该节点上的脚本代码，从而达到方便快捷修改代码块的效果。除此之外，simple-code 可以实现一键导出预制体节点脚本、一键将预制体导入至脚本、自动同步代码 import/require 引用路径等强大功能，从而满足游戏开发的基本需求。

2.2.3 TexturePackerGUI

TexturePackerGUI 是一款适用于 Unity、Cocos 等游戏引擎的图集打包工具，其拥有高效的打图集算法，能够在极短的时间将多张图片压缩成固定大小的图集，并移除图片中的透明部分，使图片更加紧密，能最大程度的节省项目资源，释放项目空间。将相关游戏图片资源打包成为图集有利于项目的管理与资源的调用。

2.3 本章小结

本章逐步介绍了本次课题所开发游戏的运行环境、开发工具及将会使用到的插件和工具。并详细说明了开发该游戏所使用的 CocosCreator 引擎在游戏开发上的优势。通过插件和工具的介绍，为后续游戏开发提供了参考价值。

第三章 游戏需求分析

3.1 系统需求概述

本游戏设定为2D视角回合制策略战斗游戏，目前可支持游戏平台暂定为PC端。在一场游戏战斗中，玩家可分别控制其背包中的精灵进行战斗，背包最多存在6只精灵，且在战斗前玩家可在背包和待定背包中最多12只精灵中进行选择精灵出战，战胜对手全部精灵即可获得胜利。

游戏主流程为回合制对战系统，对战系统具备五种状态：游戏开始、回合开始、回合结束、游戏结束，等待。在各个状态中分别执行与监听相应事件，完成战斗系统中战斗数值计算与战斗数据更新，以达到回合制战斗反馈的效果[6]。同时为了使游戏难度增加，因此会为电脑玩家设定一套智能出招系统，使电脑玩家能够通过场上局势做出尽量正确的判断，并通过给电脑玩家附加一定增益效果，提高游戏难度，从而使玩家能够沉浸式体验游戏。

游戏功能除主流程功能外，还具有商店系统、道具系统、精灵养成系统等相关功能，每个系统都具备完整的UI，能够清晰直观的让用户通过UI获取相关功能的信息。每个功能都拥有一套独立的数据模板，且都能够缓存到游戏中，以实现重新进行游戏后保留数据。

另外因为该游戏为单机游戏，因此目前游戏暂无登陆系统，而是能够让用户做到打开游戏即可游玩，不需要进行联网或登录等操作。

3.2 游戏背景

游戏开始在一艘宇宙飞船上，22世纪，地球资源匮乏，玩家为了能够给祖国带回无尽能源，开始了英勇的星际探险，在探险途中，玩家会遇到强大的敌人和可靠的外星伙伴，而作为战斗指挥家的玩家将会通过指挥外星伙伴进行战斗从而打败邪恶的敌人。而玩家在星际的探索也会从星球扩大到星系，所遇到的敌人能力也会愈发强大，但好在玩家的外星伙伴也能通过一定方式强化并突破自己，为了完成祖国的光荣使命，玩家与他的外星伙伴将会联手铲除宇宙中的恶势力。

3.3 系统可行性分析

可行性分析主要是对本课题所需要实现的内容进行评估，判断及研究[7]，从而确定项目能否顺利完成开发，并提供能够综合分析系统可行性的方法。一般来说，评估系统可行性的方式有三种，分别为技术可行性分析、经济可行性分析以及操作可行性分析，下面将对以上三种方式进行列举说明。

3.3.1 技术可行性

本次课题设计的游戏基于CocosCreator引擎开发，CocosCreator引擎本身具有属性检查、资源管理等大部分游戏开发基本功能，对开发者的要求则是能够使用TypeScript语言对游戏的动效表现，逻辑功能进行开发与优化，在开发前安装的相关插件与工具能够一定程度上减轻开发压力，提升开发效率。目前在技术上有两大难点，第一点是技术上较为考验开发者的代码能力与逻辑能力。另外，由于回合制游戏处理数据时数据量较大，则需要保证主流程逻辑稳定的同时保证战斗系统流畅性。因此该项目技术上第二难点为对游戏主流程的逻辑优化。

3.3.2 经济可行性

首先，本次项目所使用的CocosCreator引擎以及开发工具Visual Studio Code均为免费软件，TexturePackerGUI也拥有免费版可以满足图集打包的需求，尽管免费版算法与性能没有付费版高级，但基本能满足项目需求。excel-template则是论坛用户发布的开源工具，能够免费使用。另外本项目为单机游戏，无网络功能，从而节省了搭建服务器的成本。而项目中唯一付费使用的为simple-code插件，考虑到在往后的项目均可以使用该插件进行开发，因此可不计入开发成本。

其次，本次项目设计的游戏面向PC平台，且对设备的硬件需求极低，因此不需要额外支出来满足项目的开发需求。项目的设计、开发、测试及维护都可以由开发者独立完成，项目开发的经济风险对于

本次项目来说基本可以忽略。

最后，目前全球游戏市场的繁荣发展，无数游戏开发者都可以通过自主研发游戏获利[8]。在本次课题设计的游戏完成开发后可在游戏平台进行发布，如果得到一定的关注度，也能获得可观的收益。又因为本次课题设计的架构特点为能够将各个游戏模块独立使用，因此极大程度的减小了类似游戏的开发成本，并可以根据当下游戏市场潮流对游戏模块进行优化与丰富，从而提高游戏的收益。

3.3.3 操作可行性

和大部分回合制游戏的特点相同，本游戏的操作难度极低，玩家只需要通过鼠标点击 UI，即可完成游戏的操作，其游戏流程简单，提示明显，全程线性操作，玩家不会对提示产生任何误导性操作。且游戏画面风格简洁清爽，未使用到粒子特效和序列帧等内存占用较大的资源，因此基本不会对设备硬件有过高的要求。本游戏主要考验玩家的思维能力与应对能力，因此该游戏具有一定的学习成本，需要玩家对战斗进行一定的分析才能获得游戏的胜利。但游戏难度也不会过于苛刻，本游戏在难度设置上也会尽量满足玩家的游戏体验，让玩家在获得游戏胜利的同时也能够产生成就感，从而提高用户粘性与游戏沉浸感。

3.4 系统功能需求分析

本课题设计的游戏为单机游戏，游戏客户端应具备以下基本功能：

3.4.1 商店功能模块

模块说明：玩家可以在商店购买游戏道具，游戏道具可以在对战时使用，点击道具图标即可查看道具的详细信息。游戏道具也具有对应货币价格，需要玩家消耗同等货币才能完成购买操作，若货币不足则购买失败，游戏货币可以在游戏对战中获取。

参与者：玩家，系统

功能描述：玩家获取游戏道具及消耗游戏货币的方式。

商店功能用例图如图 3-1 所示。

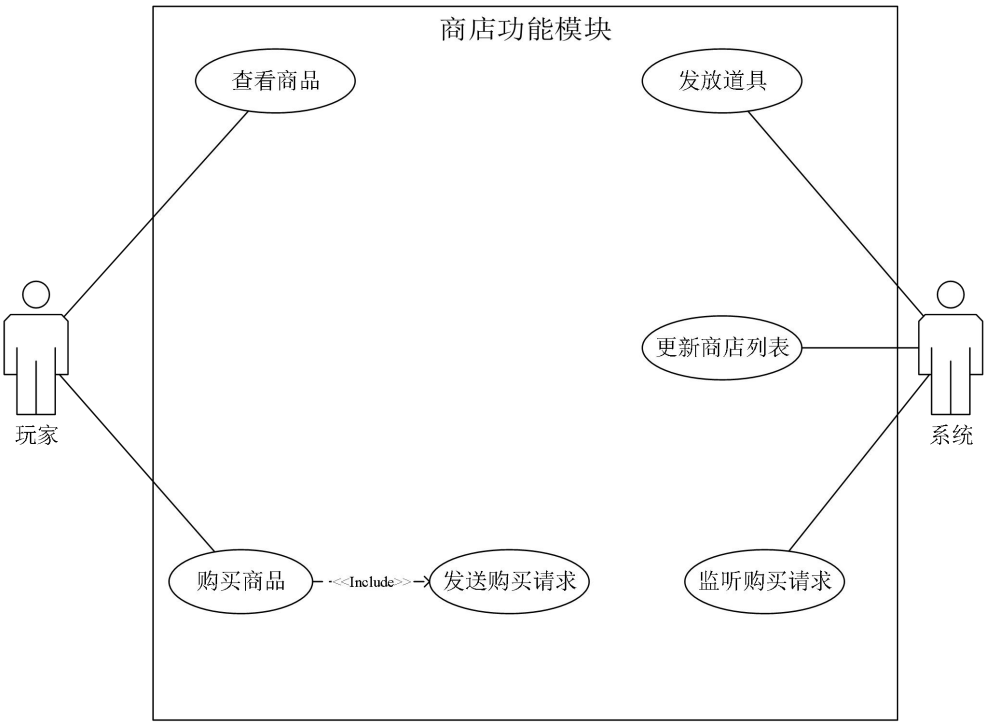

图 3-1 商店功能用例图

表 3-1 商店功能用例规约

商店功能用例规约	
用例编号	01
用例名称	商店功能用例
用例描述	本用例规定了系统用户购买商品的各种操作
执行者	系统用户及系统
包含	1.查看商品 2.购买商品
前置条件	用户金币余额大于商品价值
后置条件	成功购买商品后，商品会保存至用户背包数据缓存中
基本流程	1.用户打开商店页面 2.用户选择商品进行购买 3.购买成功，用户扣除相应金币，商品添加至用户背包。

3.4.2 精灵战斗功能模块

模块说明：玩家在游戏中的每个回合中可以进行一次操作，分别为使用技能、使用道具切换精灵，使用技能用例可分为使用攻击技能和使用属性技能。使用道具用例可分为使用 HP 道具和使用 PP 道具。玩家可对战局进行判断并自行操作。

参与者：玩家

功能描述：玩家根据目前局势，在当前回合做出操作，并通知游戏系统进行数据处理，返回对战结果。

精灵战斗功能用例图如图 3-2 所示。

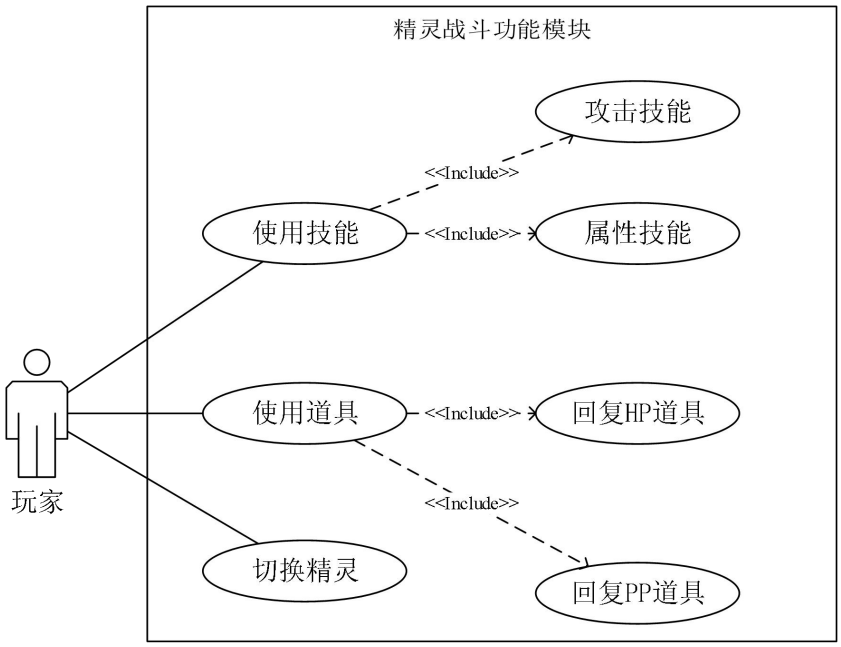

图 3-2 精灵战斗功能用例图

表 3-2 精灵战斗功能用例规约

精灵战斗功能用例规约	
用例编号	02
用例名称	精灵战斗功能用例
用例描述	本用例规定了玩家在游戏中每回合可以进行的操作
执行者	玩家
包含	1.使用技能 2.使用道具 3.切换精灵
前置条件	游戏尚未结束且玩家未对本回合进行操作
后置条件	分析处理操作数据并更新表现
基本流程	1.游戏回合开始，玩家开始操作 2.游戏系统对本回合操作结果进行数据处理，返回对战结果 3.游戏界面根据本回合数据更新场景表现

3.4.3 精灵养成功能模块

模块说明：玩家能够对已拥有的游戏单位进行培养，从而提升游戏单位的属性值，属性值的提高有助于玩家使用游戏单位挑战更困难的关卡。目前精灵单位所开放的可培养属性为等级、学习力、性格、特性等，且可供玩家选择能力提升方向。这种培养方式能够丰富游戏玩法，使玩家对游戏能够产生更多的投入与思考。

参与者：玩家

功能描述：玩家对已拥有的游戏单位使用相应养成道具，提升游戏单位能力属性。

精灵养成功能用例图如图 3-3 所示。

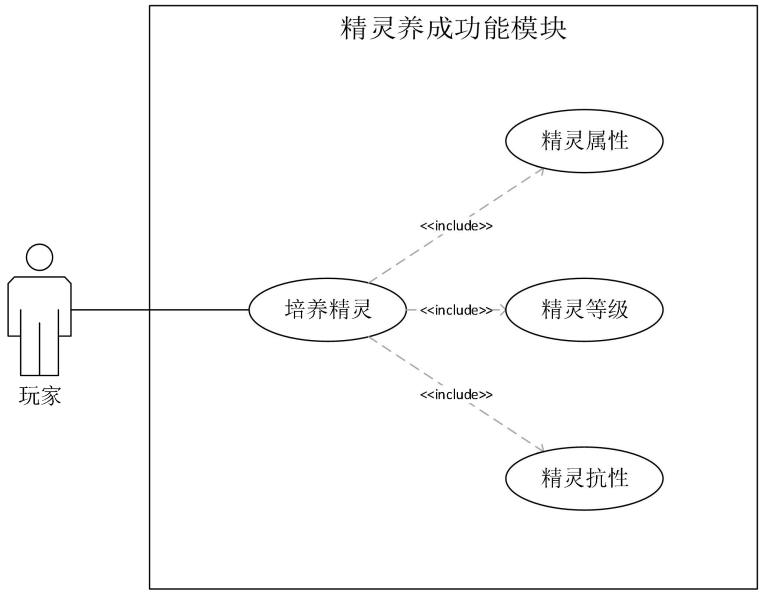

图 3-3 精灵养成功能用例图

表 3-3 精灵养成功能用例规约

精灵养成功能用例规约	
用例编号	03
用例名称	精灵养成功能
用例描述	本用例规定了玩家在游戏中可对游戏单位进行的培养方式
执行者	玩家
包含	1.提升精灵属性 2.提升精灵等级 3.提升精灵抗性
前置条件	玩家拥有相应培养道具
后置条件	消耗玩家道具，提升游戏单位属性并保存数据
基本流程	1.玩家对游戏单位使用培养道具 2.游戏单位提升相应能力，玩家道具消耗 3.系统保存玩家游戏单位数据

3.5 系统类图

由于 CocosCreator3.3.2 引擎使用了新版 API[9]，为了提高开发效率，本课题根据新版 API 开发新框架，游戏中基本 UI 类继承 BaseUI，BaseUI 中封装了展示 UI，关闭 UI，监听事件等方法供各个 UI 调用，BaseUI 继承所有 UI 的基类 Component 类，Component 中封装了 UI 的生命周期方法。UI 类主要是做表现层逻辑，若需要处理业务层逻辑，会新建相应 Manager 类进行业务处理，Manager 类处理完毕数据后，会将数据返回到 UI 类更新表现，再将业务数据保存到数据层进行数据存储。完整的游戏系统类图如图 3-4 所示。

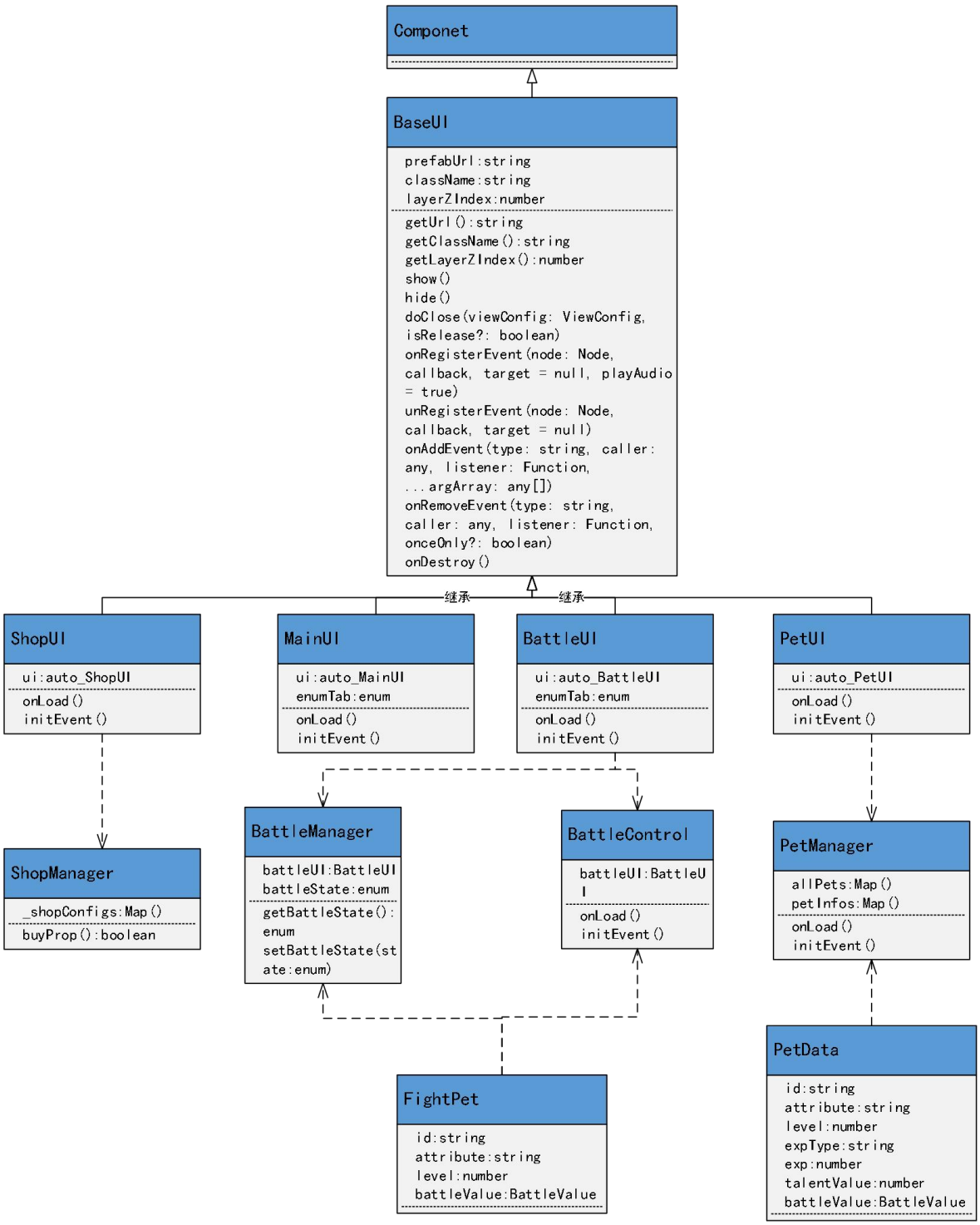

图 3-4 游戏系统类图

3.5 非系统功能需求分析

3.5.1 游戏 UI 设计风格需求

本次课题设计的游戏背景关键词为宇宙，飞船，冒险，因此游戏 UI 应该选择贴近游戏背景的科技感 UI，同时需要注重画面简洁明亮，让用户能够对游戏页面所展示的功能一目了然。

3.5.2 游戏脚本逻辑规范需求

考虑到单机游戏的可扩展性和灵活性，本课题设计的游戏除了满足基本功能需求之外，还需要对游戏主流程各个状态所涉及的代码块进行区分，以此实现状态独立的目的。状态独立有助于开发者后续对

代码与游戏流程进行维护与优化，从而高效实现游戏功能的扩展。

3.5.3 游戏完整性需求

因为目前该课题设计的游戏全程由个人研发，在限定时间内完成项目并保证主流程完整性的同时也需要省略或删除一些比较花费时间的流程，如游戏动画特效表现，游戏动画特效主要是由龙骨动画及粒子特效等文件构成，但是由于该特效制作难度大，学习成本高，因此将本次课题此方案作为后续游戏优化的任务中进行安排。

3.6 本章小结

本章通过对本课题设计的游戏的各个功能模块的需求进行详细介绍与描述，进一步明确了游戏的风格与特点。在本次可行性分析中，游戏需要实现的系统需求有：商店功能模块需求，精灵对战功能模块需求，精灵养成功能模块需求等。游戏需要实现的非系统需求有：游戏 UI 设计风格需求，游戏脚本逻辑规范需求，游戏完整性需求。

第四章 游戏设计

4.1 游戏主菜单布局设计

游戏主菜单是每个游戏进行主流程时必须操作的节点，也是能够让玩家产生深刻印象的 UI 部分，因此对于游戏主菜单的设计往往是游戏中优先级最高的流程。游戏主菜单在设计过程中需要注意的事项有两点：一是需要注重 UI 界面的布局与空间感，在保证 UI 不显得臃肿的同时需要使游戏功能尽可能展示给玩家，让玩家有了解功能的想法。

游戏主菜单 UI 布局分为顶部，底部及右侧 UI，顶部 UI 主要展示用户头像，昵称，签名及货币等信息。底部展示游戏周边功能信息，如精灵、个人、图鉴等信息，点击底部信息栏目按钮即可跳转至相应 UI 页面。右侧 UI 主要展示游戏主流程功能，如对战、商城等功能。每个功能都拥有一套独立 UI 可供玩家预览。游戏主菜单界面布局如图 4-1 所示。

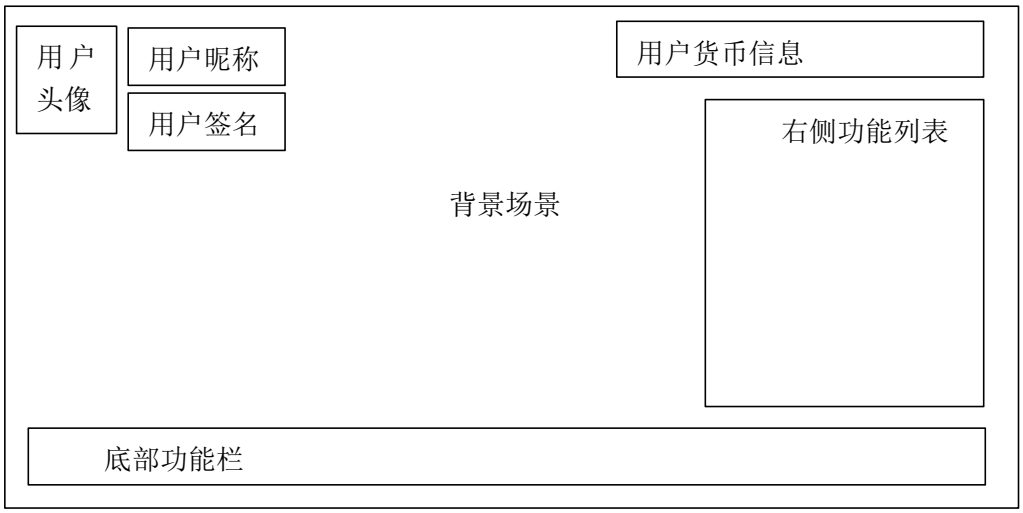

图 4-1 游戏主菜单界面布局图

4.2 游戏详细设计

本小节中将会对上述的各个游戏功能模块进行介绍，并用流程图与时序图辅助帮助阐述游戏功能模块的设计。

4.2.1 商店功能模块设计

玩家可以在商店购买游戏道具，游戏道具可以在对战时使用，玩家进入商城，系统将会检索商城表中所有道具信息并展示在商城页面的滚动视图中。游戏道具也具有对应货币价格，若玩家选择购买道具，系统将会检测玩家信息中的游戏货币信息，若货币不足则购买失败，游戏货币可以在游戏对战中获取。商品购买流程图如图 4-2 所示。

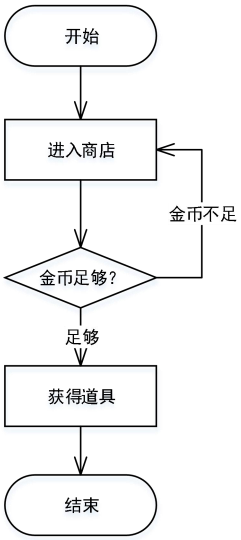

图 4-2 商品购买流程图

该模块时序图如图 4-3 所示。

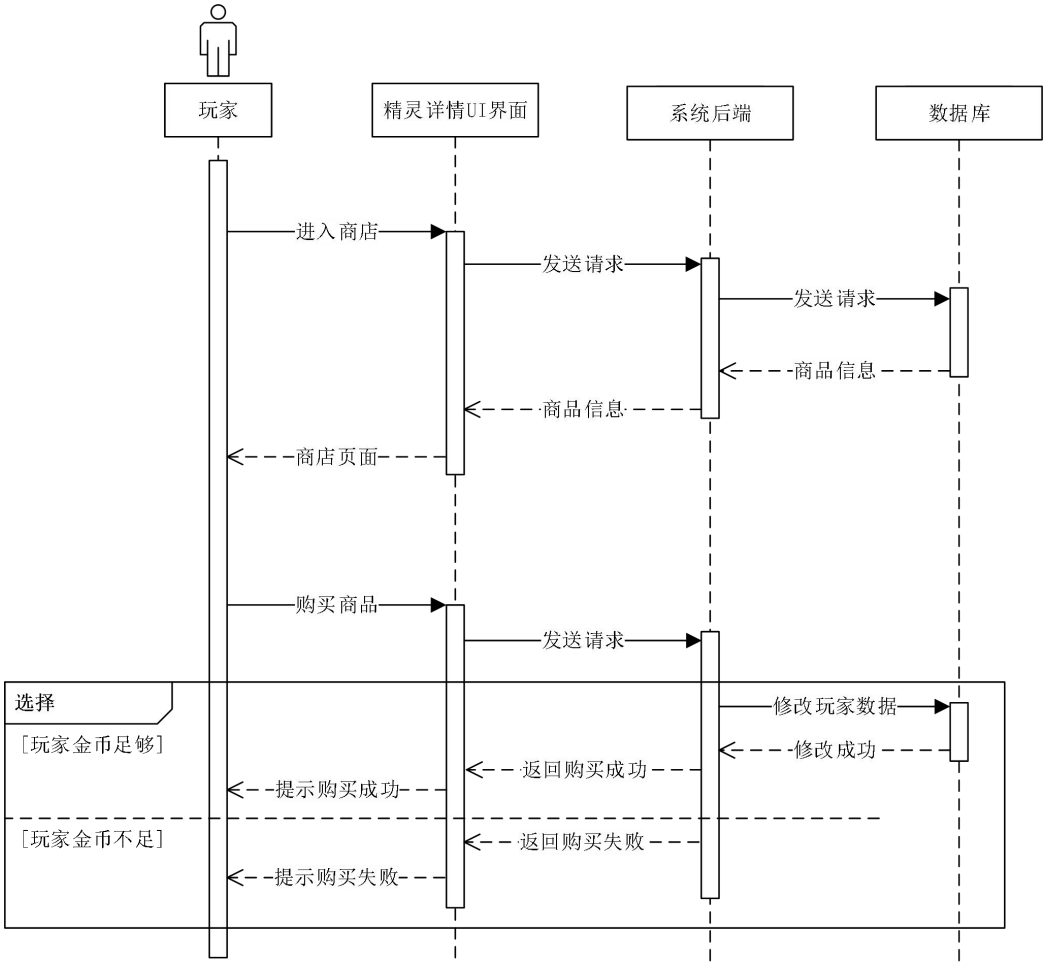

图 4-3 商店功能模块时序图

4.2.2 精灵战斗功能模块设计

本课题设计的游戏对战机制为回合制对战，回合开始时，玩家可以选择在本回合使用技能或道具。若在倒计时内玩家未作出操作，则系统将强制玩家使用随机一项技能。同时对本回合战斗数据进行处理[12]，判断双方优先级，双方的优先级由各自游戏单位的速度能力值与本回合双方的技能先制等级决定，

当双方技能限制等级相同时，系统将对比精灵单位的速度能力值判断优先级，并根据双方优先级依次完成伤害计算[13]。当游戏单位体力低于或等于 0 时，则游戏单位下场，在本局游戏中不能再次上场，玩家需选择其他游戏单位上场继续作战，当任意一方游戏单位全部下场时，则另一方胜利，游戏结束。精灵战斗流程图如图 4-4 所示。

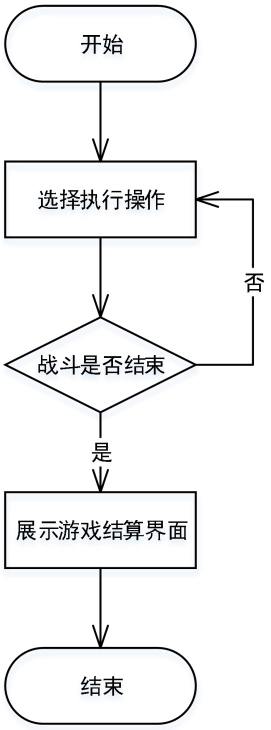

图 4-4 精灵战斗流程图

该模块时序图如图 4-5 所示。

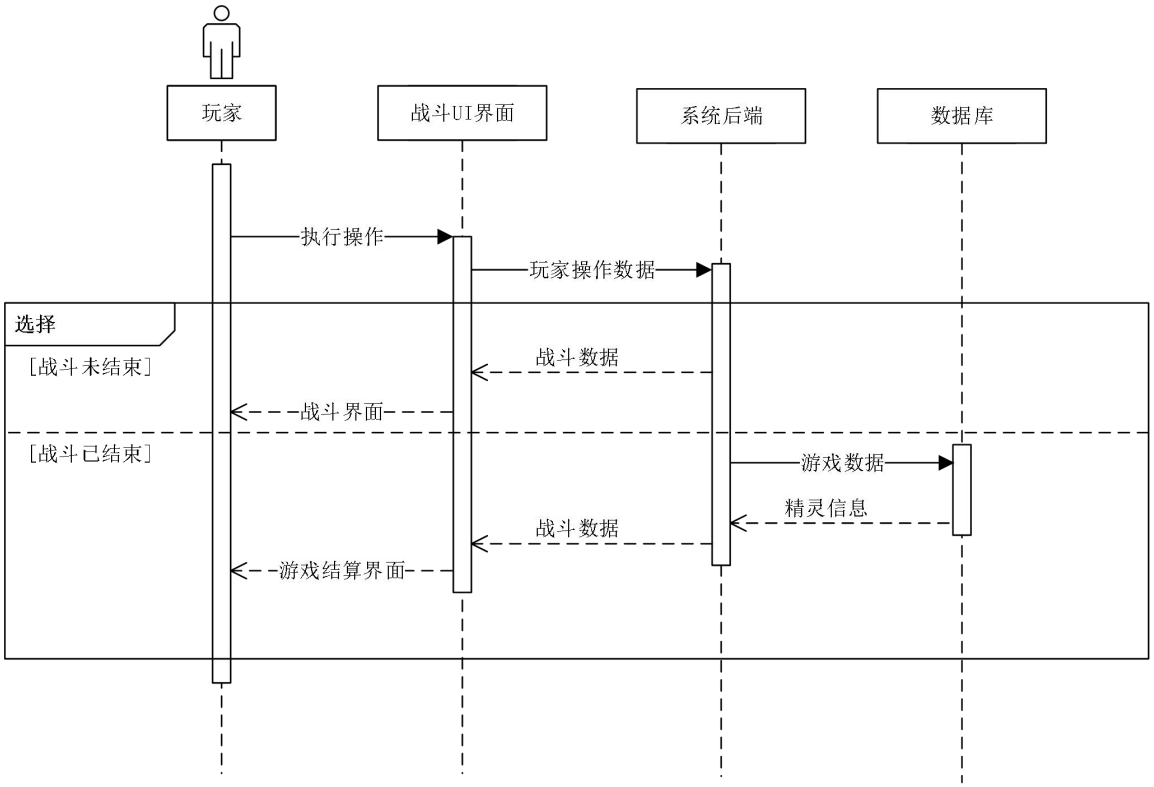

图 4-5 精灵战斗功能模块时序图

4.2.3 精灵养成功能模块设计

在精灵详情 UI 中，玩家可使用培养道具对精灵进行培养升级，提高精灵属性，从而提高游戏性。目前精灵单位所开放的可培养属性为等级、学习力、性格、特性等，且可供玩家选择能力提升方向。每一项培养属性都有其上限值，防止玩家过度培养，增加游戏可拓展性。精灵养成流程图如图 4-6 所示。

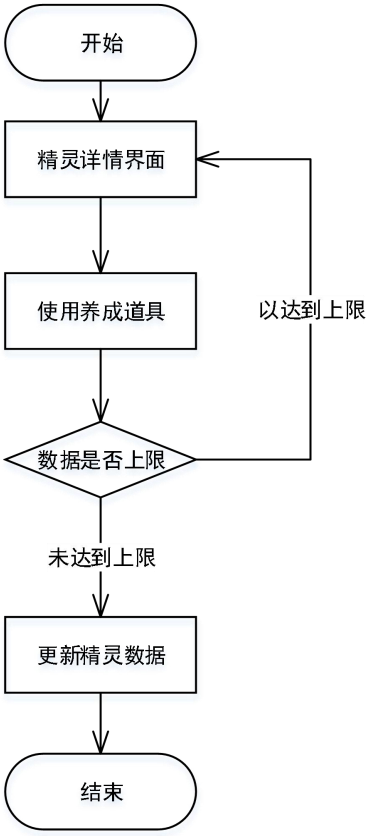

图 4-6 精灵养成流程图

该模块时序图如图 4-7 所示。

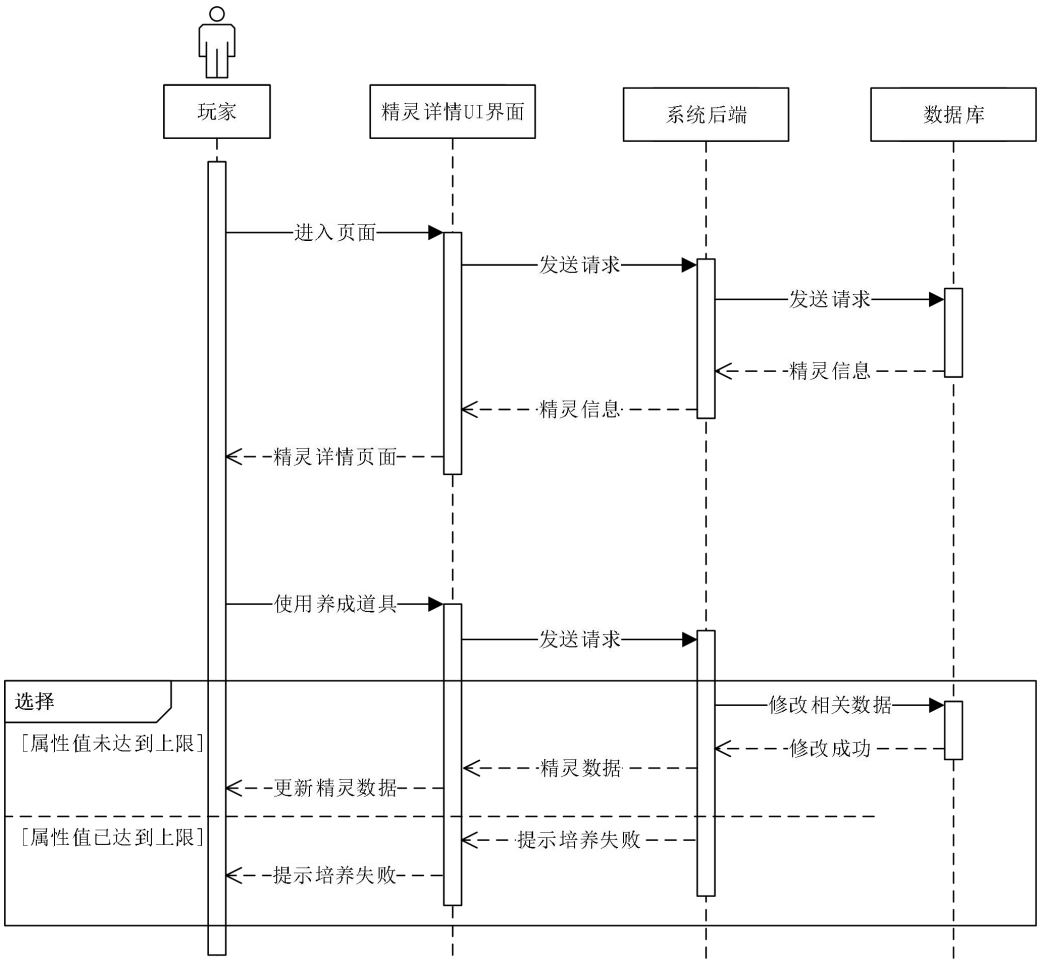

图 4-7 精灵养成功能模块时序图

4.3 游戏数据模型设计

本游戏数据存储方式采用本地存储的方式，将 Json 格式的游戏数据保存至本地存储中。游戏数据模型字段与属性通过插件 excel-temple 导表工具转换为相应的 Json 格式文件并保存在项目中，玩家修改数据后，系统将会对 Json 文件进行相应的增上改查操作。

4.3.1 游戏数据库物理结构设计

游戏商店具有购买游戏道具的功能，玩家可以使用游戏货币在商城中购买所需要的道具，商店道具主要为战斗系统服务，玩家可在战斗中使用道具，可以提高游戏的可玩性。目前商店道具设计的数据库表格如表 4-1 所示。

商店道具信息表(ShopConfig)：

表 4-1 商店道具信息表

字段名	数据类型	长度	为空	主（外键）	说明
ShopId	varchar	10	N	Y	商品 Id
Item	varchar	10	N	Y	物品 Id 对应所购买的物品

续表 4-1 商店道具信息表

字段名	数据类型	长度	为空	主（外键）	说明
Num	smallint		N	N	购买数量
BuyValue	smallint		N	N	购买价格

在游戏中，技能作为精灵单位的核心操作，是决定游戏胜利的关键。每个游戏单位都拥有 5 个技能，结合精灵单位能力值考虑，技能分为物攻技能、特攻技能及属性技能三种类型。每个技能都具备威力，使用次数，属性克制等属性，用于决定技能的下限效果。而技能强度上限由技能效果决定，技能效果能直接改变游戏单位的属性值，为游戏单位附加增益效果和减益效果，通过与游戏单位的特质配合往往可以产生影响战局的能力[10]。从而提升了游戏内容的多样性与复杂性，有关技能信息表如表 4-2 所示。

技能信息表(SkillConfig)：

表 4-2 技能信息表

字段名	数据类型	长度	为空	主（外键）	说明
SkillId	varchar	10	N	Y	技能 ID
SkillName	varchar	20	N	N	技能名称
Type	smallint		N	N	技能类型，1 为物攻技能，2 为特攻技能，3 为属性技能
Attribute	varchar	10	N	Y	属性 Id，关联属性克制表
Power	smallint		N	N	技能威力大小
PP	smallint		N	N	技能使用次数
Priority	smallint		Y	N	技能先制等级，默认为 0
HitRate	smallint		N	N	技能命中率
CritRate	smallint		N	N	技能暴击概率
Desc	longtext		Y	N	技能描述
Effect	varchar	50	Y	N	技能效果，关联效果信息表 Id

异常状态。为了使游戏战局丰富多变，因此需要在游戏流程中加入异常状态的设定使战斗具有一定风险性。在游戏设定上，异常状态作为回合类效果附加在游戏单位上，需要经过一定回合才可完全解除异常状态效果，目前游戏配置的异常状态信息如表 4-3 所示：

异常状态信息表(AbnormalConfig)：

表 4-3 异常状态信息表

字段名	数据类型	长度	为空	主（外键）	说明
Id	varchar	10	N	Y	异常状态 Id
Icon	varchar	20	N	Y	图标 Id，关联图片资源表 Id
Name	varchar	20	N	N	异常状态名称
Type	smallinti	2	N	N	1：非控制类异常状态 2：控制类异常状态
Desc	longtext		N	N	异常状态效果描述

每个游戏单位及其拥有的技能都拥有与之对应的属性，本课题设计的游戏一共具有 16 种属性，不同属性之间拥有一定的克制关系，克制关系分为克制、普通、微弱、无效。克制关系决定游戏单位造成与受到的伤害大小。目前游戏配置的属性克制信息如表 4-4 所示：

属性克制表(AttributeConfig)：

表 4-4 属性克制表

字段名	数据类型	长度	为空	主（外键）	说明
Id	varchar	10	N	Y	属性克制 Id
Icon	varchar	20	N	Y	图标 Id，关联图片资源表 Id
Name	varchar	20	N	N	属性克制名称
Attribute	float		N	N	对其他属性的克制倍数

除此之外，游戏单位还具有种族值等其他决定游戏单位强度的字段，游戏单位是游戏对战系统中的基础，在本课题设计的游戏中，游戏单位的能力值由同样的字段决定[11]，但是每个单位的字段属性各不相同，因此玩家对与不同的游戏单位都有不同的培养方法，从而增强了游戏性。游戏单位信息表具体信息如表 4-5 所示：

游戏单位信息表(PetConfig)：

表 4-5 游戏单位信息表

字段名	数据类型	长度	为空	主（外键）	说明
Id	varchar	10	N	Y	游戏单位 Id
PetAvatar	varchar	10	N	Y	精灵头像，对应图片表 Id
PetPic	varchar	10	N	Y	精灵信息，对应图片表 Id
Attribute	varchar	10	N	Y	精灵属性,对应属性表 Id
Name	varchar	20	N	N	精灵名称
Atk	smallint		N	N	精灵攻击种族值

续表 4-5 游戏单位信息表

字段名	数据类型	长度	为空	主（外键）	说明
spAtk	smallint		N	N	精灵特攻种族值
Def	smallint		N	N	精灵防御种族值
spDef	smallint		N	N	精灵特防种族值
Spd	smallint		N	N	精灵速度种族值
Hp	smallint		N	N	精灵体力种族值
Skills	longtext		N	N	精灵技能，由多个技能 Id 拼接为字符串

各个数据模型创建完毕后，需要根据模型信息，来确定各个数据模型之间的关系，数据库实体关系图如图 4-8 所示。

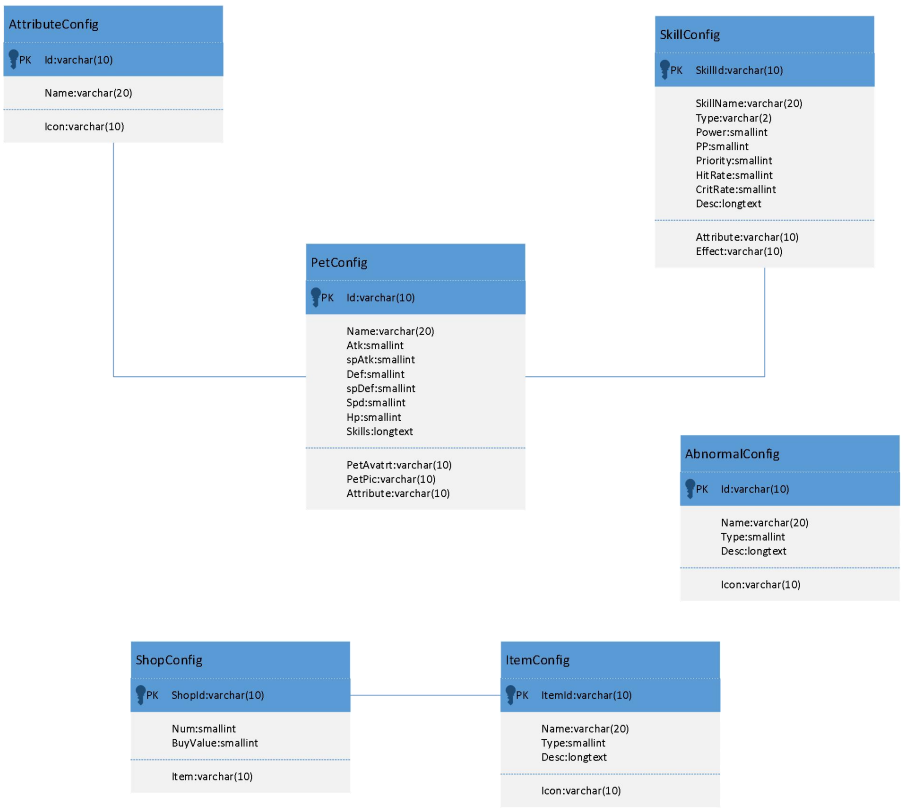

图 4-8 数据库实体关系图

4.4 本章小结

本章对本课题的游戏项目进行深入设计，确定了游戏主菜单的 UI 布局和游戏主要功能设定如商店等功能设计，并使用流程图及时序图加以描述，详细介绍了本课题设计的游戏流程如战斗系统、精灵养成功能模块等，从而完善了游戏的完整性，是游戏性更为丰富。

第五章 游戏实现

5.1 游戏主菜单界面的实现

游戏主菜单是每个游戏进行主流程时必须操作的节点，也是能够让玩家产生深刻印象的 UI 部分。根据第三章游戏需求分析中的非系统功能分析及第四章游戏设定中的游戏主菜单设计所确认的 UI 风格与主菜单 UI 布局。本课题设计的游戏将由 CocosCreator 编辑器进行 UI 的布局与排版，最终 UI 效果如图 5-1 所示。

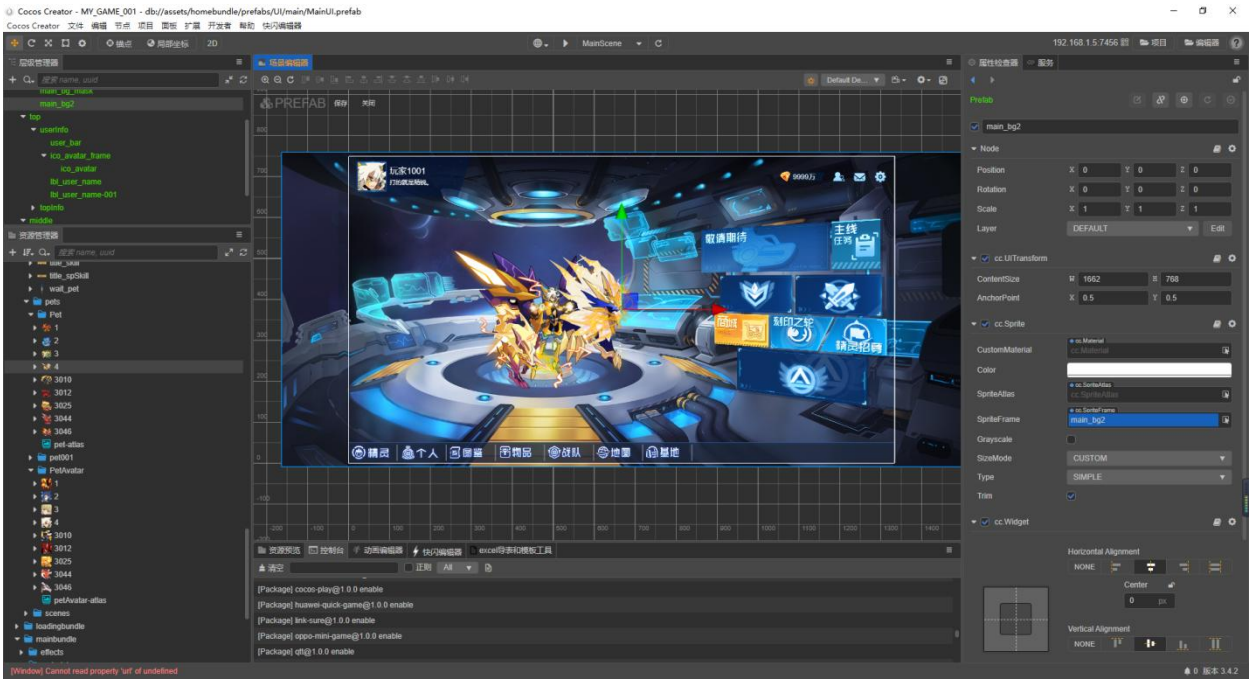

图 5-1 CocosCreator 编辑器界面

表 5-1 介绍了 CocosCreator 编辑器中常见基础组件

表 5-1 CocosCreator 编辑器常见基础组件表

组件名称	组件介绍
cc.UITransfrom	UI 变换组件，用于控制节点的大小及锚点
cc.Widget	控制边距组件，用于约束节点相对父节点的边界距离
cc.Sprite	精灵组件，用于显示项目资源中的图片
cc.Label	标签组件，用于显示文字
cc.Button	按钮组件，用于提供点击事件
cc.Layout	布局组件，用于横向或竖向均匀排列节点
cc.ScrollView	滚动视图组件，用于提供滑动事件

完成主菜单 UI 界面的布局后，需要为主菜单的按钮添加事件，以商店功能为例，当用户点击商店

按钮时，将会触发主菜单打开商店 UI 的事件，该事件代码如下：

```
this.onRegisterEvent(this.ui.btn_mall, () => {        //点击节点 btn_mall,即商店按钮
engine.uiManager.openUIAsync(UIConfigs.shopUI); //打开商店 UI
}, this)
```

其中 openUIAsync()方法为打开 UI 的通用方法，可应用于游戏内所有已配置 UI 的开启。

5.2 游戏商店的实现

游戏商店作为能够让玩家购买道具的 UI，需要清晰直观的展示出商店中存在的购买道具信息，因此游戏商店将由 ShopUI 预制体及 ShopItem 预制体组成，ShopUI 预制体作为商店的容器，用于展示商品 ShopItem 预制体，ShopItem 可直观显示的信息为商品名称，商品图标及商品价格，具体 UI 如图 5-2、图 5-3 所示：

图 5-2 商店预制体 ShopUI

图 5-3 商品预制体 ShopItem

游戏商店打开后，将会读取商店配置表，并将配置表信息转换成数组类型的数据，通过遍历的方式将信息保存值 ShopItem 中，商店生成商品实现代码如下：

```
    if (this.ui.content.children.length == 0) {       //若商店内容为空
            let shops = ConfigReader.readShopConfig();  //读取商店信息表
            let arr: Array<ShopConfig> = [];              //初始化数组用于存储商品信息
            for (let key in shops) {
                arr.push(shops[key]);                     //将商品信息存入数组中
            }
            arr.sort((a: ShopConfig, b: ShopConfig) => {     //对数组进行排序
                return a.Sort - b.Sort;
            })
            this.ui.scv_prop.getComponent(ScrollViewContent).setData(arr);  //将数组传递至
ScrollViewContent 组件用于遍历生成 ShopItem 预制体
        }
```

商品初始化信息代码如下：

```
    setData(data: ShopConfig) {
        let item: ItemConfig = ConfigReader.readItemConfig(data.ItemId);
        //商品 Id
        this.shopId = data.Id;
        //商品描述
        this.shopDesc = item.Desc
        //商品名称
        this.ui.lbl_name.getComponent(Label).string = item.Name;
        //商品价格
        this.ui.lbl_num.getComponent(Label).string = String(data.BuyValue);
        //商品图标
        let frame = engine.resLoader.getAtlasByTag(ResPathEnum.PropIcon.bundle,
ResPathEnum.PropIcon.resPath, item.Name);
        this.ui.ico_prop.getComponent(Sprite).spriteFrame = frame;
    }
```

商品购买实现代码如下：

```
    this.onRegisterEvent(this.ui.btn_buy, () => {
            HomeManager.shopManager.buyItem(this.shopId)
        }, this)

    /**
     * 购买物品
     */
    buyItem(id: string) {
        let shopConfig: ShopConfig = this._shopConfigs[id];
        //金币购买
        if (!PlayerManager.getInstance().addGold(shopConfig.BuyValue * -1)) {
            UIHelp.showTip("金币数量不足！");
```

```
                return false;
            }
            //添加商品至物品栏
            if (HomeManager.propManager.addProp(shopConfig.ItemId, shopConfig.Num)) {
                UIHelp.showTip("商品添加成功");
                return true
            }
            else {
                return false
            }
        }
        /**
          * 增加金币
          * @param num  金币数量
          */
        addGold(num: number) {
            //获取玩家信息
            let playerInfo = GameDataManager.getInstance().getGameData().playerInfo;
            //玩家金币不够时则返回 false
            if (playerInfo.gold + num < 0) return false;
            playerInfo.gold += num;
            playerInfo.save();
            engine.listenerManager.trigger(ListenerType.RefreshTopInfo);
            return true;
        }
        /**
         * 增加道具
         * @param propId
         */
        addProp(propId: string, num: number) {
            //物品信息
            let itemConfig: ItemConfig = ConfigReader.readItemConfig(propId);
            //背包列表
            let propList: Map<number, PropInfo> =
GameDataManager.getInstance().getGameData().propList;

            //判断物品类型
            switch (itemConfig.Type) {
                //PP 道具
                case EnumAllProp.PP:
                //HP 道具
                case EnumAllProp.HP:
                //性格道具
                case EnumAllProp.Character:
                //特性道具
```

```
            case EnumAllProp.Feature:
                //背包中已有该道具，则数量+1
                if (propList[propId]) {
                    propList[propId].count += num;
                    break;
                }
                //背包中无该道具，则添加道具
                propList[propId] = new PropInfo();
                propList[propId].id = propId;
                propList[propId].count += num;
                break;
        }
        //存储至本地缓存
        engine.storage.setLocalItem(LocalKeys.LOCAL_PROPLIST, propList);
        return true;
    }
```

5.3 游戏单位实现

游戏单位是战斗系统中的玩家操控的基本单位，在本课题设计的游戏中，游戏单位将会作为独立的预制体，预制体中封装了战斗单位的信息及战斗方法。游戏单位预制体结构如图 5-4 所示：

图 5-4 游戏单位预制体结构图

游戏单位具有一个子节点 pet_sprite，其功能是用来展示游戏单位图片，将 pet_sprite 锚点设为(0.5, 0)，则可实现固定底部的表现。游戏单位在游戏中的效果如图 5-5 所示。

图 5-5 游戏单位战斗预览图

本课题设计的游戏会在游戏单位预制体上挂载脚本，用于封装游戏单位的属性及功能[14]，精灵单位属性及初始化方法代码如下。

```
/** 精灵基本信息 */
private petInfo: FightPet = null;
/** 战斗专属 Id */
private fightId: number = 0;
/** 精灵增益效果 */
private pet_buffs: Map<EnumBuff, number> = null;
/** 精灵减益效果 */
private pet_deBuffs: Map<EnumDeBuff, number> = null;
/** 精灵异常状态 */
private pet_abnormal: Map<EnumAbnormal, number> = null;
/** 先制等级 */
private pet_priority: number = 0;
/** 造成伤害倍率 */
private multi_damage: number = 1;
/** 受到伤害倍率 */
private multi_hurt: number = 1;
/** 初始化精灵状态，切换上场时调用 */
initPet() {
    // 初始化精灵 Buff
    this.pet_buffs = new Map<EnumBuff, number>();
    // 初始化精灵 DeBuff
    this.pet_deBuffs = new Map<EnumDeBuff, number>();
    // 初始化精灵异常状态
    this.pet_abnormal = new Map<EnumAbnormal, number>();
    // 初始化精灵先制等级
    this.pet_priority = 0;
    // 初始化精灵造成伤害倍率
    this.multi_damage = 1;
    // 初始化精灵受到伤害倍率
    this.multi_hurt = 1
}
```

5.4 游戏养成系统实现

玩家获取到的游戏单位能够在游戏背包中查看，在游戏背包内，玩家看查看游戏单位的详细数据，并对游戏单位的能力进行培养，游戏背包 UI 如图 5-6 所示。

图 5-6 游戏背包 UI 图

游戏背包中包含治疗与详情功能，点击治疗能够恢复精灵状态。点击详情时，游戏背包会传递当前选中精灵 Id 至精灵详情页，精灵详情页会通过精灵 Id 获取精灵信息并展示精灵详情页。精灵详情 UI 如图 5-7 所示。

图 5-7 游戏详情 UI 图

游戏详情页分为信息、技能、抗性、道具等四个模块，信息页用于查看精灵基础属性，技能页用于查看精灵技能，抗性页用于查看精灵抗性，道具页用于查看玩家可使用道具，UI 初始化代码如下所示。

```
        initPetTab() {
            //精灵信息初始化
            //等级
            this.ui.lbl_level.getComponent(Label).string = String(this.petInfo.level);
            //学习力
            this.ui.lbl_learn.getComponent(Label).string = String(this.petInfo.learningValue.all);
            //天赋
            this.ui.lbl_talent.getComponent(Label).string = String(this.petInfo.talentValue);
            //性格
            this.ui.lbl_character.getComponent(Label).string =
ConfigReader.readCharacterConfig(this.petInfo.character.valueOf()).Name;
            //特性
            this.ui.lbl_feature.getComponent(Label).string =
ConfigReader.readFeatureConfig(this.petInfo.features.valueOf()).Name;
            //体力上限
            this.ui.lbl_maxHp.getComponent(Label).string = "0";

            //精灵属性
            //攻击
            this.ui.lbl_atk.getComponent(Label).string = String(this.petInfo.battleValue.atk);
            this.ui.lbl_learn_atk.getComponent(Label).string = String(this.petInfo.learningValue.atk);
            //特攻
            this.ui.lbl_spAtk.getComponent(Label).string = String(this.petInfo.battleValue.sp_atk);
            this.ui.lbl_learn_spAtk.getComponent(Label).string =
String(this.petInfo.learningValue.sp_atk);
            //防御
            this.ui.lbl_def.getComponent(Label).string = String(this.petInfo.battleValue.def);
            this.ui.lbl_learn_def.getComponent(Label).string = String(this.petInfo.learningValue.def);
            //特防
            this.ui.lbl_spDef.getComponent(Label).string = String(this.petInfo.battleValue.sp_def);
            this.ui.lbl_learn_spDef.getComponent(Label).string =
String(this.petInfo.learningValue.sp_def);
            //速度
            this.ui.lbl_spd.getComponent(Label).string = String(this.petInfo.battleValue.spd);
            this.ui.lbl_learn_spd.getComponent(Label).string =
String(this.petInfo.learningValue.spd);
            //体力
            this.ui.lbl_hp.getComponent(Label).string = String(this.petInfo.battleValue.hp);
            this.ui.lbl_learn_hp.getComponent(Label).string = String(this.petInfo.learningValue.hp);
        }
```

目前拥有的精灵养成方式为：恢复精灵体力、提升精灵等级，提升提升精灵学习力，提升精灵个体，提升精灵抗性等，详细实现代码如下。

```
/**
 * 刷新战斗属性
 * @param id 精灵 Id
 */
refreshBattleValue(pet: PetData) { }

/**
 * 增加学习力
 * @param id 精灵 Id
 * @param type 能力
 * @param num 点数
 */
addLearning(pet: PetData, type: EnumLearnType, num) { }

/**
 * 恢复体力
 * @param id 精灵 Id
 * @param hp 体力值
 * 判断是否满体力
 */
recoverHp(pet: PetData, hp: number) { }

/**
 * 提升等级
 * @param id 精灵 Id
 * @param level 等级值
 * 判断是否满级
 */
upgradeLevel(pet: PetData, level: number) { }

/**
 * 设置等级
 * @param id 精灵 Id
 * @param level 等级值
 */
setLevel(pet: PetData, level: number) { }

/**
 * 提升抗性
 * @param id 精灵 Id
 * @param type 抗性类
 * @param abId 异常抗性 Id
 * 判断异常抗性是否存在
 * 判断抗性值是否满
 */
```

```
upgradeResist(pet: PetData, type: EnumResistType, abId?: string) { }

/**
 * 设置性格
 * @param pet
 */
setCharacter(pet: PetData, cid?: string) { }

/**
 * 设置特性
 * @param pet
 */
setFeature(pet: PetData, cid?: string) { }

/**
 * 重置抗性
 */
resetResist() { }

/**
 * 刷新抗性
 */
refreshResist() { }
```

5.5 战斗系统实现

在本游戏中，玩家可通过挑战关卡进行战斗，进入战斗时，战斗系统会初始化玩家游戏单位信息，将游戏单位信息转化为战斗可使用的数据，并实例化相应预制体进行数据封装。在战斗中，战斗系统会将本回合玩家与敌方进行的操作进行数据处理，最后将数据发送至表现层，通知表现层做出相应表现[15]。战斗界面 UI 如图 5-8 所示。

图 5-8 战斗界面 UI

战斗系统核心代码如下。

```
showPet(player: EnumPlayer, fightId?: string) {
        let pet_node: Node = null;
        if (player == EnumPlayer.Own) {     //判断目前操作身份(玩家  or AI)
            if (!fightId) {       //如果不是主动切换精灵，则选择任意一只可上场精灵切换
                for (let i in this.own_pets) {
                    let tmp_PetNode = this.own_pets[i].getComponent(PetUI).getPetInfo();
                    if (tmp_PetNode.battleValue.hp > 0) {    //若精灵体力大于 0
                        pet_node = tmp_PetNode;
                        break;
                    }
                }
            } else {
                pet_node = this.own_pets[fightId];
            }

            if (pet_node) {
                this.own_pet_now = pet_node;
                //发送表现层修改通知
                engine.listenerManager.trigger(GameListenerType.RefreshBattlePetUI, player)
            }
        } else if (player == EnumPlayer.Enemy) {       //AI 执行相同逻辑
            if (!fightId) {
                for (let i in this.enemy_pets) {
                    let tmp_PetNode = this.enemy_pets[i].getComponent(PetUI).getPetInfo();
                    if (tmp_PetNode.battleValue.hp > 0) {
                        pet_node = tmp_PetNode;
                        break;
                    }
                }
            } else {
                pet_node = this.enemy_pets[fightId];
            }
            if (pet_node) {
                this.enemy_pet_now = pet_node;
                //发送表现层修改通知
                engine.listenerManager.trigger(GameListenerType.RefreshBattlePetUI, player)
            }
        }
    }
```

5.6 本章小结

本章通过展示 UI 界面及代码的方式详细说明了各个功能的结构及逻辑实现思路，使用游戏主流程需求得以实现。

第六章 测试结果和分析

6.1 游戏运行环境测试

本课题设计的游戏主要环境配置分为硬件配置和软件配置。

硬件开发环境如表 6-1 所示：

表 6-1 硬件开发环境表

硬件类型	硬件名称
处理器	AMD Ryzen 5 3600 6-Core Processor(3600 MHz)
内存	16.00 GB (2400 MHz)
显卡	AMD Radeon RX 6600 XT
硬盘	KIOXIA-EXCERIA SSD500 GB

软件开发环境如表 6-2 所示：

表 6-2 软件开发环境表

软件类型	硬件名称
开发工具	Microsoft Windows 10 专业版 (64 位)
开发引擎	CocosCreator3.4.2
DIRECTX 支持	12

游戏完成编译后，将会用于在不同运行环境下测试，本课题设计的游戏在三种不同的运行环境下运行的测试结果如表 6-3 所示：

表 6-3 运行环境测试表

系统环境	硬件	测试结果
Windows7	Intel(R)i5-4460 处理器，4GB 内存，显示适配器：GT630，512GB 硬盘	正常运行
Windows8	Intel(R)i7-8750H 处理器，8GB 内存，显示适配器：GTX1050Ti，256GB 固态硬盘	正常运行
Windows10	AMD Ryzen 5 3600 处理器，16GB 内存，显示适配器：RX 6600 XT，512GB 固态硬盘	正常运行

6.2 游戏基本功能测试

功能测试会对项目的研发结果进行验收，以此判断项目是否能够发布。在本小节，将会对项目进行用例测试，并根据测试结果判断功能是否符合预期标准，游戏用例测试表如表 6-4 所示：

表 6-4 游戏用例测试表

测试模块	测试功能	操作步骤	预期结果	测试结果
客户端	启动客户端	在不同运行环境下启动游戏客户端	能够正常打开客户端	游戏客户端能够在不同的运行环境下正常启动游戏客户端，符合预期
	退出客户端	在不同运行环境下退出游戏客户端	正常退出客户端且用户数据能够保存	退出后用户数据能够保存，符合预期
游戏主菜单	主菜单按钮功能	点击主菜单存在的按钮	能够进入目前已开放的功能界面	能够进入主菜单目前已开放的功能界面，符合预期
游戏主菜单	主菜单表现效果	购买道具后退回主菜单	能够正常显示用户信息且主菜单金币数量变化且数据正常	能够正常显示用户信息且主菜单金币数量变化且数据正常，符合预期
游戏商店	购买道具功能	购买道具	购买时会根据情况做出不同处理方式。金币不够时会提示“金币数量不足”，道具购买成功后会提示“购买成功”。且购买成功后道具能够正常发放	购买时，金币不够时会提示“金币数量不足”，道具购买成功后会提示“购买成功”。且购买成功后道具能够正常发放，符合预期
精灵背包	精灵背包按钮功能	点击精灵背包存在的按钮	能够进入目前已开放的功能界面	能够进入精灵背包目前已开放的功能界面，符合预期
精灵背包	精灵背包表现效果	进入精灵背包页面	能够正常展示精灵背包中的精灵信息	能够正常展示精灵背包中的精灵信息，符合预期
精灵背包	精灵治疗功能	点击精灵治疗按钮	恢复当前背包中所有精灵的状态	恢复当前背包中所有精灵的状态，且表现实时刷新，符合预期
精灵详情	精灵详情表现效果	进入精灵详情页面	能够正常展示精灵详情中的精灵详情信息	能够正常展示精灵详情中的精灵详情信息，符合预期
战斗系统	战斗流程	选择关卡进入战斗，体验战斗流程	战斗流程及表现正常，战斗数据能够正常处理。	战斗数据能够正常处理并实时刷新战斗表现，符合预期
战斗系统	技能功能	战斗时点击技能按钮	游戏单位能够正常使用技能，且技能效果正确	游戏表现正常，技能效果能与战斗数据对应，符合预期
战斗系统	道具功能	战斗时点击道具选项，并选择道具使用	游戏单位能够正常使用道具，且道具效果正确	游戏表现正常，道具效果能与战斗数据对应，符合预期

6.3 游戏性能测试

本次游戏性能测试针对本课题设计的游戏的执行效率进行测试[15]，从而对游戏性能进行评估。由于本游戏界面设计简洁，并没有引用占用内存较大的资源文件，通过预加载资源的方式，能够达到表现流畅的效果。并且通过相关算法的优化，显著提高了战斗时数据处理效率。经过在不同环境下对游戏进行性能测试，游戏帧率都能稳定保持 60 帧左右，使玩家拥有到流畅的游戏体验。因此本游戏通过游戏性能测试。

6.4 本章小结

本章通过在不同运行环境下运行本课题设计的游戏，设计了较为全面的测试用例对游戏的功能，对游戏的功能及性能进行详细的测试。经测试，游戏符合预期设计标准，对部分系统功能还可以做进一步优化。

第七章 总结与展望

7.1 研发工作总结

在国内游戏市场日益壮大的背景下，自主研发成为了国内游戏厂商的迫切需求。本文通过阐述国产回合制游戏特点，对比了国内外回合制游戏的差距，指出国内回合制游戏弊端并加以研究，最终确定了本课题所设计的回合制游戏研发方向并加以实现。

本文通过使用目前游戏市场主流引擎之一 CocosCreator，介绍了游戏开发的基本流程，完成了一款具有较为完整流程的回合制游戏的设计与研发。由于 CocosCreator3.3.2 引擎使用了新版 API，为了提高开发效率，本课题根据新版 API 开发新框架，并结合 Simple-code 插件等插件及工具完成游戏功能模块的开发。通过此次游戏研发的实践，显著提升了我的代码与逻辑能力，也让我对游戏开发行业充满憧憬，更加坚定了我想要从事游戏开发相关工作的想法。同时，本次项目的完成，也能使更多玩家认识到回合制游戏的魅力，从而改变国内玩家对回合制游戏的传统看法。

7.2 未来工作展望

本文虽然完成了回合制游戏的基本主流程，但由于本人开发能力有限，并且没有足够的时间深入研究游戏研发的相关知识，本游戏目前还存在着许多不足与可优化的地方，例如：通过远程服务器接收并处理玩家信息完成远程玩家对战；游戏可增加剧情模块，提升玩家游戏体验；可增加多人游戏模式，使多名玩家能够同时作战等。因此还需要在后续的的开发维护工作中作进一步完善。

参考文献

[1]杜渐. 我国网络游戏产业研究[D]. 对外经济贸易大学.

[2]黄大林, 黄晓灵, 蒋波. 电竞体育产业市场潜力分析及其建议[J]. 当代体育科技, 2016, 6(5):4.

[3]郝琳琳. 回合制网络游戏中的信息传递[J]. 福建质量管理, 2019, 000(014):284.

[4]闵磊. 基于 Cocos Creator 引擎的游戏开发技术研究[J]. 2020.

[5]张景焱, 马春江. 利用 Cocos Creator 进行游戏开发的分析[J]. 信息与电脑, 2019(6):2.

[6]严珮婷. 回合制游戏中数据处理方法,装置以及电子终端:, CN110772786A[P]. 2020.

[7]蒲冬梅. 软件项目可行性分析评审的要点[J]. 电子技术与软件工程, 2017, 000(024):54-55.

[8]王龙, 李韬伟, 杨振发. 游戏引擎研究与分析[J]. 软件导刊, 2018, 17(2):3.

[9]Zhang J， Ma C， Computer D O. Analysis of Game Development with Cocos Creator. 2019.

[10]王啸天. 互动式动漫游戏数据库模型研究[J]. 数字通信世界, 2019, No.180(12):275-275.

[11]Kavanagh W J， Miller A， Norman G，et al. Balancing Turn-Based Games with Chained Strategy Generation[J]. IEEE Transactions on Games, 2019, PP(99):1-1.

[12]黄叶, 吕唐杰, 范长杰,等. 一种游戏AI的策略决策模型训练方法和装置:, CN111330279A[P]. 2020.

[13]鲍方. 模拟经营游戏中智能决策系统的研究与实现[D]. 杭州电子科技大学.

[14]刘璐. 游戏 AI 行为逻辑控制方法和系统:, CN111111202A[P]. 2020.

[15]姜亮. 计算机应用系统性能测试技术及应用研究[J]. 信息与电脑, 2018(7):3.

致谢

我首先要感谢我指导老师，她耐心的指导和独到的见解给予我莫大的帮助。她孜孜不倦的钻研和勤奋进取的精神令我终生难忘，感谢我的导师在我进行毕业设计的时候帮助我解决了许多项目实现上的困难，使我能够顺利完成毕业设计的选题、设计与实现。

同时，我也要感谢泉州师范学院软件学院软件工程（软件开发方向）的授课老师以及所有的同学们，我们在大学四年中能够互相学习，了解软件开发相关专业知识，使我有能力去面对未来的挑战，希望在未来的时光里，我们都能够不负青春，茁壮成长。

最后感谢我的家人，感谢他们一直以来对我的关心，培养我成长，教会我做人。

衷心感谢在百忙之中评阅论文和参加答辩的各位教师、教授！