**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**

**Institute of food science and engineering Marine and food sciences 183117044 Xiao Longhao**

**Instructor Yu Juan Associate professor**

**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**

# 1 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。

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。

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，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，画出本项目用例图及系统类图，判断项目实现风险及难度。

第四章游戏设计：对本项目中各个功能模块进行分析设计，画出本项目主流程的活动图和时序图，并对项目所需数据库进行建模，画出E-R图与信息表。

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。

# 2 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，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。

### 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。

# 3 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。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，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。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：

1. 游戏客户端稳定运行。玩家可以正常运行并操作游戏客户端，游戏客户端需支持在不同配置的PC环境下保持游戏系统的稳定性。实现游戏数据缓存至本地的功能，避免出现数据丢失的问题，使玩家可以正常进行游戏，从而不破坏游戏体验。
2. 商店功能。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。商店模块用例图如图3-1所示。



图3-1 商店模块用例图

1. 电脑AI，AI会根据本回合双方的状态和属性值判断下一回合的操作，使玩家在战斗过程不会感到枯燥，提升游戏体验，电脑AI的用例图如图3-2所示。



图3-2 电脑AI用例图

（4） 完整正常的游戏流程。玩家进入游戏后，显示主界面UI,点击主界面UI的关卡挑战按钮可选择关卡进行挑战。战斗过程中，玩家每回合可操控精灵使用技能或道具，在友方精灵未全部阵亡的情况下击败敌方全部精灵则关卡胜利，并发放游戏结算奖励，每一关都有特定的关卡目标，游戏结算奖励将会根据关卡目标的完成数量而增多。若关卡失败则不会发放奖励。在精灵背包UI中，玩家可对精灵进行培养升级，提高精灵属性，从而提高游戏性。完整的游戏系统类图如图3-3所示。



图3-3 游戏系统类图

（5）精灵养成功能。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。

## 3.5 非系统功能需求分析

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阶段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。另外，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。

## 3.6 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：游戏客户端稳定运行，商店功能，电脑AI，完整正常的游戏流程，精灵养成功能等。游戏需要实现的非系统需求有：主流程各个阶段代码块独立，游戏UI风格确定。

# 4 游戏设计

## 4.1 游戏主菜单设计

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

用户签名

用户昵称

用户头像

用户货币信息

底部功能栏

背景场景

右侧功能列表

## 4.2 游戏商店设计

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。目前商店道具信息表如表4-1所示。

表4-1 商店道具信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否为空 | 是否主键 | 说明 |
| ShopId | varchar | 10 | not null | 是 | 商品Id |
| Item | varchar | 10 | not null |  | 物品Id 对应所购买的物品 |
| Num | int |  | not null |  | 购买数量 |
| BuyValue | int |  | not null |  | 购买价格 |

道具价格等属性可以通过商店道具信息表进行配置，商店信息表中的内容在游戏中将通过导表工具转换成为Json格式的数据以便游戏解析调用，道具信息表也会根据字段生成相应配置表类，有关代码内容如下：

export class ShopConfig {

    /\*\* 商品ID \*/

    Id: string;

    /\*\* 物品ID \*/

    ItemId: string;

    /\*\* 单次购买数量 \*/

    Num: number;

    /\*\* 购买价格 \*/

    BuyValue: number;

    /\*\* 商品排序 \*/

    Sort: number;

}

{

    "1001": {

        "Id": "1001", //商品ID

        "ItemId": "1001", //物品ID

        "Num": 1, //单次购买数量

        "BuyValue": 100, //购买价格

        "Sort": 1 //商品排序

    }

}

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。商品购买流程图如图4-2所示。



图4-2 商品购买流程图

## 4.3 游戏单位设计

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，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。游戏单位数据模型代码如下：

/\*\* 精灵类 \*/

export class PetData {

    /\*\* 精灵ID \*/

    id: string = null;

    /\*\* 精灵属性ID \*/

    attribute: string = null;

    /\*\* 经验类型 \*/

    expType: string = null;

    /\*\* 已获取经验 \*/

    exp: number = 0;

    /\*\* 精灵等级 \*/

    level: number = 1;

    /\*\* 个体值 \*/

    talentValue: number = 0;

    /\*\* 战斗属性值 \*/

    battleValue: BattleValue = new BattleValue();

    /\*\* 学习力 \*/

    learningValue: LearningValue = new LearningValue();

    /\*\* 性格 \*/

    character: EnumCharacter = null;

    /\*\* 特性 \*/

    features: EnumFeatures = null;

    /\*\* 抗性 \*/

    resistance: Resistance = new Resistance();

    /\*\* 刻印 \*/

    signet: Array<Signet> = new Array<Signet>();

    /\*\* 技能 \*/

    skills: Map<SkillConfig, number> = new Map<SkillConfig, number>();

}

/\*\* 战斗属性值 \*/

export class BattleValue {

    atk: number = 0;

    sp\_atk: number = 0;

    def: number = 0;

    sp\_def: number = 0;

    spd: number = 0;

    max\_hp: number = 0;

    hp: number = 0;

}

/\*\* 学习力 \*/

export class LearningValue {

    atk: number = 0;

    sp\_atk: number = 0;

    def: number = 0;

    sp\_def: number = 0;

    spd: number = 0;

    hp: number = 0;

    constructor() { }

}

/\*\* 抗性类 \*/

export class Resistance {

    damageResist: DamageResist = new DamageResist();

    abnormalResist: AbnormalResist = new AbnormalResist();

}

/\*\* 伤害抗性类 \*/

export class DamageResist {

    /\*\* 暴击伤害抵抗 \*/

    criticalResist: number = 0;

    /\*\* 固定伤害抵抗 \*/

    fixedResist: number = 0;

    /\*\* 百分比伤害 \*/

    percentage: number = 0;

    constructor() { }

}

/\*\* 刻印 \*/

export class Signet {

    atk: number = 0;

    sp\_atk: number = 0;

    def: number = 0;

    sp\_def: number = 0;

    spd: number = 0;

    hp: number = 0;

    constructor() { }

}

## 4.4 游戏玩法设计

本课题设计的游戏的基本玩法为每回合操控游戏单位使用技能或道具进行对战，在友方精灵未全部阵亡的情况下击败敌方全部精灵则关卡胜利，并发放游戏结算奖励，每一关都有特定的关卡目标，游戏结算奖励将会根据关卡目标的完成数量而增多，若关卡失败则不会发放奖励。游戏主流程包含以下设定与功能：

（1）技能功能，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。从而提升了游戏的多样性与复杂性，有关技能信息表如表4-2所示。

表4-2 技能信息表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 是否为空 | 是否主键 | 说明 |
| SkillId | varchar | 10 | not null | 是 | 技能ID |
| SkillName | varchar | 20 | not null |  | 技能名称 |
| Type | int |  | not null |  | 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 | not null |  | 属性Id，关联属性克制表 |
| Power | int |  | not null |  | 技能威力大小 |
| PP | int |  | not null |  | 技能使用次数 |
| Priority | int |  | not null |  | 技能先制等级，默认为0 |
| HitRate | int |  | not null |  | 技能命中率 |
| CritRate | int |  | not null |  | 技能暴击概率 |
| Desc | varchar | 50 |  |  | 技能描述 |
| Effect | varchar | 50 | not null |  | 技能效果，关联效果信息表Id |

1. 对战系统，本课题设计的游戏对战机制为回合制对战，回合开始时，玩家拥有30秒的时间进行操作，操作过程中，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，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。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。战斗系统活动图如图4-3所示。



图4-3 战斗系统活动图

1. 属性克制功能，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。详细枚举类型属性定义代码如下：

/\*\* 属性 \*/

export enum EnumAttribute {

    /\*\* 水系 \*/

    water = "1001",

    /\*\* 火系 \*/

    fire = "1002",

    /\*\* 草系 \*/

    grass = "1003",

    /\*\* 飞行系 \*/

    flying = "1004",

    /\*\* 地面系 \*/

    ground = "1005",

    /\*\* 机械系 \*/

    mechanics = "1006",

    /\*\* 电系 \*/

    thunder = "1007",

    /\*\* 冰系 \*/

    ice = "1008",

    /\*\* 暗影系 \*/

    dark = "1009",

    /\*\* 光系 \*/

    light = "1010",

    /\*\* 超能系 \*/

    psychic = "1011",

    /\*\* 战斗系 \*/

    fight = "1012",

    /\*\* 普通系 \*/

    normal = "1013",

    /\*\* 圣灵系 \*/

    saint = "1014",

    /\*\* 神灵系 \*/

    deity = "1015",

    /\*\* 混沌系 \*/

    chaos = "1016",

}

1. 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及说明如表4-3所示：

表4-3 异常状态效果表

|  |  |  |
| --- | --- | --- |
| Id | Name | Desc |
| 1001 | 烧伤 | 1-2回合内，进入重伤效果（体力恢复效果减半），每回合受到最大体力值1/8的真实伤害。 |
| 1002 | 冻伤 | 1-2回合内，攻击技能先制-1，每回合受到最大体力值1/8的真实伤害。 |
| 1003 | 中毒 | 每回合受到最大体力值1/16的真实伤害。 |
| 1004 | 麻痹 | 1-3回合内无法行动。 |
| 1005 | 疲惫 | 1-3回合内无法行动。 |
| 1006 | 害怕 | 1-3回合内无法行动。 |
| 1007 | 睡眠 | 1-3回合内无法行动。 |
| 1008 | 灼热 | 1-3回合内无法行动，状态结束后进入1回合烧伤效果。 |

续：表4-3

|  |  |  |
| --- | --- | --- |
| 1009 | 冰封 | 1-3回合内无法行动，状态结束后进入1回合冻伤效果。 |
| 1010 | 感染 | 1-3回合内无法行动，状态结束后进入中毒效果。 |

## 4.5 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功能，游戏单位设定等并详细介绍了本课题设计的游戏玩法设计如战斗系统、属性克制关系及异常状态效果等，从而完善了游戏的完整性，是游戏性更为丰富。

# 5 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：

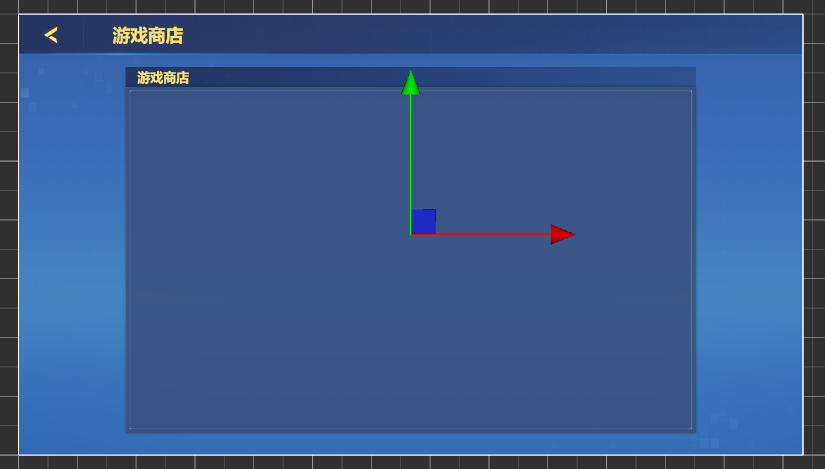


图5-2 商店预制体ShopUI



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！");

ResPathEnum.PropIcon.resPath, item.Name);

        this.ui.ico\_prop.getComponent(Sprite).spriteFrame = frame;

    }

            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;

        //判断武平类型

        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

## 5.3 游戏单位实现

            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;

    }