

分类号：\_\_\_\_\_  
论文编号：202001032103

密 级：\_\_\_\_\_

贵 州 大 学  
2020 届硕士研究生学位论文

# 基于 Django 的斗地主游戏平台设计 与实现

学科专业：软件工程  
研究方向：不区分研究方向  
导 师：李 丹  
研 究 生：刘 满 义

中国 ▪ 贵州 ▪ 贵阳

2020 年 6 月

# 目 录

|                       |    |
|-----------------------|----|
| 摘 要.....              | I  |
| Abstract.....         | II |
| 第一章 绪论.....           | 1  |
| 1.1 选题背景与意义.....      | 1  |
| 1.2 国内外研究现状.....      | 2  |
| 1.3 本文研究内容.....       | 3  |
| 1.4 论文组织结构.....       | 4  |
| 1.5 本章小结.....         | 4  |
| 第二章 相关理论和技术.....      | 5  |
| 2.1 贪婪算法.....         | 5  |
| 2.2 MCTS.....         | 6  |
| 2.3 UML.....          | 6  |
| 2.4 Python.....       | 7  |
| 2.5 Django.....       | 7  |
| 2.6 MySQL.....        | 8  |
| 2.7 本章小结.....         | 8  |
| 第三章 系统需求分析.....       | 9  |
| 3.1 功能需求.....         | 10 |
| 3.1.1 系统设置.....       | 10 |
| 3.1.2 游戏空间.....       | 11 |
| 3.1.3 算法对抗.....       | 13 |
| 3.1.4 战绩查询.....       | 15 |
| 3.1.5 信息查询.....       | 16 |
| 3.1.6 个人中心.....       | 17 |
| 3.2 非功能需求.....        | 19 |
| 3.3 本章小结.....         | 19 |
| 第四章 系统设计.....         | 20 |
| 4.1 系统总体设计.....       | 20 |
| 4.1.1 系统体系结构设计.....   | 20 |
| 4.1.2 系统架构设计.....     | 21 |
| 4.1.3 系统功能架构设计.....   | 22 |
| 4.2 系统详细设计.....       | 23 |
| 4.2.1 功能模块设计.....     | 24 |
| 4.2.1.1 系统设置.....     | 24 |
| 4.2.1.2 游戏空间.....     | 25 |
| 4.2.1.3 算法对抗.....     | 32 |
| 4.2.1.4 战绩查询.....     | 33 |
| 4.2.1.5 信息查询.....     | 35 |
| 4.2.1.6 个人中心.....     | 36 |
| 4.2.2 数据库设计.....      | 38 |
| 4.2.2.1 数据库实体类设计..... | 38 |
| 4.2.2.2 数据库物理设计.....  | 40 |

|                         |    |
|-------------------------|----|
| 4.3 智能算法设计 .....        | 47 |
| 4.3.1 贪婪选择出牌算法设计 .....  | 47 |
| 4.3.2 权重选择出牌算法设计 .....  | 49 |
| 4.3.3 MCTS 算法应用设计 ..... | 50 |
| 4.4 本章小结 .....          | 51 |
| 第五章 系统实现 .....          | 52 |
| 5.1 关键功能实现 .....        | 52 |
| 5.1.1 贪婪选择出牌算法实现 .....  | 52 |
| 5.1.2 权重选择出牌算法实现 .....  | 53 |
| 5.1.3 算法排序实现 .....      | 55 |
| 5.1.4 第一视角实现 .....      | 56 |
| 5.1.5 实时通讯实现 .....      | 57 |
| 5.2 系统功能实现 .....        | 58 |
| 5.2.1 用户注册与登录 .....     | 58 |
| 5.2.2 用户管理 .....        | 59 |
| 5.2.3 单机模式 .....        | 60 |
| 5.2.4 在线模式 .....        | 61 |
| 5.2.5 好友模式 .....        | 63 |
| 5.2.6 锦标赛模式 .....       | 64 |
| 5.2.7 残局模式 .....        | 66 |
| 5.2.8 算法对抗 .....        | 66 |
| 5.2.9 战绩查询 .....        | 69 |
| 5.3 本章小结 .....          | 70 |
| 第六章 系统测试 .....          | 71 |
| 6.1 测试系统部署及测试环境 .....   | 71 |
| 6.2 测试用例 .....          | 71 |
| 6.2.1 算法测试 .....        | 71 |
| 6.2.2 功能测试 .....        | 73 |
| 6.3 压力测试 .....          | 77 |
| 6.4 本章小结 .....          | 79 |
| 第七章 总结与展望 .....         | 80 |
| 7.1 总结 .....            | 80 |
| 7.2 展望 .....            | 81 |
| 致谢 .....                | 82 |
| 参考文献 .....              | 83 |
| 图版 .....                | 87 |
| 表版 .....                | 89 |

# 基于 Django 的斗地主游戏平台设计与实现

## 摘 要

随着人工智能技术（特别是机器学习技术）在大规模棋牌类游戏中取得的重要进展，例如谷歌的 AlphaGo 和 CMU 的 Libratus 分别在围棋和一对一无限注德州扑克游戏上取得了超越当前人类职业玩家的能力，“斗地主”等棋牌类游戏也引起了人工智能研究人员的广泛兴趣，研发开放可共享的棋牌类博弈平台成为该类研究的重要内容。

本文研究开放“斗地主”博弈平台，为“斗地主”博弈提供研究的数据集，同时，也为“斗地主”算法提供开放的对抗试验场景。具体工作如下：

1. 基于 UML 的设计模式进行平台设计，为游戏数据的收集设计了单机模式、在线模式、好友模式等模式的“斗地主”博弈，为数据收集奠定了坚实的基础，同时制定数据收集和存储的标准来保证所收集数据的质量；为“斗地主”算法测试设计了智能算法对抗试验场景，研究人员只需设置试验参数和选择算法即可开始算法的对抗试验，提高研究人员的工作效率。

2. 基于当前手牌的局部最优出牌选择设计并实现了贪婪选择出牌算法；制定手牌权重计算规则进行手牌权重值计算，在手牌权重值的基础上设计和实现了权重选择出牌算法；对基于蒙特卡洛树搜索实现的“斗地主”算法进行应用方面的设计和实现，为本系统提供了用于对抗试验场景的基础算法。在此基础上设计了三个算法的智能度排序试验，根据算法在试验中的胜率进行智能度排序，从高到低为蒙特卡洛树搜索算法、权重选择出牌算法、贪婪选择出牌算法。

3. 基于 B/S 架构，使用 Django 框架采用 Python 语言实现了该平台。其中，数据存储采用 MySQL，游戏中的实时在线通讯采用 Django Channels，前端页面框架采用 Layui。

**关键词：**贪婪选择出牌算法；权重选择出牌算法；WEB 系统；Django；Django Channels

# **Design and implementation of django-based Doudizhu game platform**

## **Abstract**

As artificial intelligence technology (especially the machine learning technology) in card games made important progress, such as Google AlphaGo and CMU Libratus respectively on the Go and one to one No-limit Texas Hold 'em made beyond the ability of professional players. The card game "Doudizhu" has aroused wide interest in artificial intelligence researchers, the shareable card game platform become an important part of the research.

This paper studies the realization of an open "Doudizhu" game platform, provides a research dataset for the "Doudizhu" game, and also provides an open algorithm confrontation experiment scenario for the "Doudizhu" algorithm. The specific work is as follows:

1. This paper base on UML design the game platform, and designs the game of fighting the "Doudizhu" in stand-alone mode, online mode, friend mode and other modes for collecting game data sets, which supplys a solid foundation for data collection. At the same time, it sets standards for data collection and storage to ensure the quality of data. An intelligent algorithm confrontation experiment scenario is designed for the experiment of "Doudizhu" algorithm. Researchers need to set experiment parameters and choose the algorithm to start the confrontation experiment between algorithms, so as to improve the working efficiency of researchers.

2. This paper base on the local optimal selection of current cards, the greedy selection algorithm is designed and implemented. The card weight value calculation rule is made to calculate the card weight value. On the basis of the card weight, the weight selection is designed and implemented. The application design and implementation of the "Doudizhu" algorithm based on Monte Carlo tree search. Those provide the basic algorithms for the algorithm confrontation experiment scenario. On

the basis of those, this paper designs three algorithms' intelligence ranking experiment, which sort the intelligence ranking according to the winning rate of the algorithm in the experiment. From high to low, the algorithms are MCTS algorithm, weight selection algorithm of play the card and greedy selection algorithm of play the card.

3. The game platform architecture is Browser/Server, Web framework is Django and implementation language is Python. Among them, MySQL is used for data storage, Django Channels is used for real-time online communication during the game, and Layui is used for web page UI framework.

**Key words:** Greedy selection algorithm of play the card; Weight selection algorithm of play the card; The WEB system; Django; Django Channels

# 第一章 绪论

## 1.1 选题背景与意义

人工智能作为一门新的技术科学，在智能游戏领域内，得到了愈加广泛的重视，使得智能棋牌类游戏的研究人员也越来越多，棋牌类游戏包括斗地主、象棋、围棋、麻将等。对于其中的斗地主来说，目前智能斗地主算法的研究人员数量在不断的增加，但在智能算法的研究过程中，基本上都面临着试验数据匮乏和算法对抗试验繁琐的问题，极少有研究人员或游戏平台会共享斗地主游戏的数据集，算法的对抗试验通常需要对以往论文中的智能算法进行复现，这不仅使研究人员需要浪费大量的时间对其他算法进行复现而且不能保证复现的算法与论文中的原算法完全一致。

本文通过对各种模式的斗地主游戏数据进行收集，对收集的游戏数据进行清洗，按照一定的标准对数据进行选取和转换，构建标准的斗地主游戏数据集，根据不同智能算法的特定需求再对数据集进行相应处理，得到该智能算法研究所需的试验数据集。通过资源共享的方式为其他进行智能斗地主算法的研究人员提供试验数据集，同时还为玩家提供游戏数据的查询权限，以便玩家对以往游戏对局数据和出牌数据进行查询。

本文通过对算法对抗试验过程进行分析，根据算法试验的需求进行智能斗地主算法对抗试验的开放平台设计，并对试验过程中的数据进行收集，以便研究人员可根据对抗试验结果进行分析。通过开放平台的方式为研究人员提供快速搭建自身算法与其他算法之间进行对抗试验的环境，以此来节省研究人员对其他算法进行复现的时间，使研究人员可以更加专注的进行自身算法的研究同时也很好的避免复现算法不一致的问题。

综上所述，本文的研究价值主要是收集平台各种模式的游戏数据并按照一定的标准生成游戏数据集，为智能算法提供试验数据集；提供方便、快捷搭建算法对抗试验环境的平台，为研究人员提供算法对抗试验场景和收集试验数据的便利，提高研究人员的工作效率。

## 1.2 国内外研究现状

斗地主游戏平台的研究在我国由来已久,1998 年联众游戏上线,正式开启我国线上棋牌游戏市场,其作为国内最早成立的网络游戏公司,曾一度成为世界上最大的网络游戏娱乐网站。2003 年 QQ 游戏正式上线,是首款自主研发的游戏产品,也是目前国内最大的休闲游戏社区平台。2007 年 JJ 比赛正式上线,是国内首个靠组织棋牌游戏比赛来吸引用户的赛事游戏平台<sup>[1]</sup>。还有许多小公司也推出众多具有地方特色的麻将游戏、扑克游戏、棋类游戏<sup>[1]</sup>。这些游戏公司也在人工智能技术发展的影响下进行智能斗地主算法的研究,为玩家提供托管和出牌提示服务,但其使用的智能算法尚未进行公开,这意味着新算法无法跟其已有的智能算法进行对抗,对智能算法的研究和进步是极其不利的。同时此类系统对玩家游戏数据查询的权限进行限制,也不共享任何的游戏数据集资源,从而导致了智能斗地主算法研究中试验数据集极其匮乏。

除众多游戏公司之外还有许多研究人员也在进行斗地主游戏的研究,诸如 2018 年赵娟,弋改珍<sup>[2]</sup>、2017 年饶子建<sup>[3]</sup>、2016 年张焕甫<sup>[4]</sup>和周进森<sup>[5]</sup>、2015 年李竹林<sup>[6]</sup>、2008 年曹庆皇,杨晓琴<sup>[7]</sup>都对斗地主游戏系统进行设计和实现,诸如 2016 翟飞<sup>[8]</sup>对棋牌游戏规则进行研究,2011 年吕宝生,钟宝荣<sup>[9]</sup>棋牌游戏中的麻将进行研究。他们在研究中都实现了基本的斗地主游戏功能,但其实现的斗地主游戏模式单一;有些研究则只是实现单机版的斗地主游戏;有些研究中未对智能斗地主算法进行研究无法达到智能斗地主的高度;游戏研究中的设计不符合玩家的操作习惯,使得设计不够人性化,不能提供更好的玩家体验。总之,他们的研究仅限于斗地主游戏的实现,不能够为其他研究人员提供任何试验数据集支持,使智能算法研究中试验数据集匮乏的现状更加严峻。

网络游戏发展的基础是互联网技术和实时通讯技术,互联网解决网络游戏中的跨地域问题,实时通讯技术解决网络游戏中实时在线通讯问题。WebSocket 是 B/S 架构实时通讯应用中最关键的协议,该协议在浏览器和服务器之间构建一条全双工的通信信道,双方随时可以互发消息,且客户端实现规范基于 JavaScript,因此在提供通信功能的同时也提供页面元素操作<sup>[10][11][12]</sup>。诸如 2019 年卢振<sup>[13]</sup>、2018 年包文祥<sup>[14]</sup>、2016 年高聪<sup>[12]</sup>对 WebSocket 协议进行研究并在不同的领域进行应用;诸如 2017 年丁伟<sup>[15]</sup>、2016 潘正辉<sup>[16]</sup>、2014 年王鹏<sup>[17]</sup>对网络游戏中的



关键技术、服务器架构方面进行研究,提供网络游戏研究技术和架构方面的参考。他们的研究证明 **WebSocket** 在实时通讯中应用的可行性,为本文收集各种模式的游戏数据奠定技术基础也验证了本文系统的可行性。

综上所述,目前的现状是大部分游戏公司和研究人员都仅仅是为玩家提供斗地主游戏平台,未向玩家提供游戏数据的查询权限和向外共享任何的游戏数据集资源,更是从来没有任何平台为研究人员提供智能算法对抗试验场景。

### 1.3 本文研究内容

本文的主要内容是设计开放的智能算法对抗试验平台、研究智能出牌算法和设计斗地主游戏数据集收集平台。本文设计并实现的系统主要功能是满足算法之间进行对抗试验和在线斗地主游戏并收集数据,游戏过程中通过智能出牌算法为玩家提供出牌的提示,同时还提供斗地主游戏的数据查询和统计。具体研究内容如下:

1、对抗试验平台设计。根据对智能斗地主算法的分析进行开放的智能算法对抗试验平台设计,通过参数设置和算法选择的方式为研究人员提供便利的对抗试验场景,为研究人员试验过程提供便利。

2、智能出牌算法研究。根据斗地主出牌规则及出牌类型进行智能出牌算法的研究,首先查阅相关的文献和资料,对其使用的方法或技术进行分析;然后与斗地主进行结合,设计出斗地主的出牌算法;最后对设计的出牌算法进行验证和完善,编写能对外调用的接口。

2、斗地主游戏设计。首先设计传统模式的斗地主游戏,明确斗地主的出牌规则、积分规则、出牌类型及其大小关系的规则制定和设计;然后在传统模式的基础上进行玩法的创新,设计出多种更加吸引玩家的玩法;最后综合多种模式的斗地主游戏进行整体设计和完善。通过设置多种模式的斗地主游戏为游戏数据的收集奠定基础。

3、系统构建的相关技术。根据对系统的分析,系统采用 **UML** 的设计模式进行设计;对基于 **Python** 的 **Web** 系统开发框架进行研究,特别对 **Django** 框架进行详细的学习;对数据库方面相关的知识和技术进行研究分析,针对 **MySQL** 的特性、操作、功能等方面进行多方面的学习;对服务器部署和维护方面的知识及

其应用技能进行深入的学习，对 Nginx 在 Linux 服务器上部署和维护方面的知识和应用进行深入学习。

## 1.4 论文组织结构

本文的组织结构如下：

第一章，绪论。介绍本文的选题背景，国内网络游戏平台的发展及相应的技术研究现状以及本文的主要研究内容：智能算法研究、开放平台设计与实现。

第二章，相关理论和技术。主要介绍本文开发所需技术或理论支撑依据，包括贪婪算法、MCTS、UML、Python、Django、MySQL 等。

第三章，系统需求分析。从功能需求和非功能需求两方面对系统的需求进行分析，得出功能需求分析出的用例图且给出对应用例图的详细说明，同时给出系统应满足的性能、界面、安全等方面的非功能性需求。

第四章，系统设计。对系统的总体设计和详细设计进行阐述，完成智能算法设计和系统的系统结构、功能结构的设计，和功能模块设计，完成功能对应的活动图及相应的数据库设计。

第五章，系统实现。首先介绍系统关键功能的实现，包括智能算法、第一视角、实时通讯功能；然后介绍系统功能的实现和功能界面的展示。

第六章，系统测试。先介绍测试系统部署的服务器硬件环境及软件环境，测试系统的版本及使用的浏览器种类，然后介绍系统测试所编写的测试用例及其测试结果。

第七章，总结与展望。对本文的研究工作总结，对本文研究的成果进行概要性的描述，然后对本文的不完善或者可扩展的部分进行展望。

## 1.5 本章小结

本章首先介绍本文的选题背景与意义和国内的研究现状，说明本文选题的意义，通过对现状的分析说明本文技术的可行性；然后介绍本文主要的研究内容，明确本文的研究主要围绕智能斗地主系统的建设而进行研究；最后对本文的组织结构进行说明，本文分七章对本文所研究的内容进行详情的阐述。

## 第二章 相关理论和技术

在目前的斗地主系统研究中,开发语言有 C/C++、Java、C#等,实现的系统有 B/S 架构的 Web 应用、C/S 架构的桌面应用和移动端的 APP 应用。考虑到本系统是一个开放性的平台,因此采用 B/S 架构结合 MySQL 数据实现 Web 平台,采用 UML 设计模式进行平台设计。在算法对抗试验中需进行智能算法的动态调用,因此开发语言采用具有动态加载模块特性的 Python,Python 不仅可以动态调用 Python 实现的智能算法还能调用 C 语言编译后的 .so 文件和 TensorFlow 的算法模型;结合系统架构和开发语言确定系统框架为 Django,并且 Django Channels 能完美解决在线斗地主游戏的实时通讯问题;本文还基于贪婪算法思想进行智能斗地主算法的研究,对 MCTS 智能斗地主算法进行应用方面的研究。

### 2.1 贪婪算法

贪婪算法是一种只关注当前情形下的局部最优解,而不从全局情形来考虑最优解的算法。这样即使贪婪算法不能得到问题最终的全局最优解,也能得到问题的局部最优解,从而得到近似的全局最优解,因为可以通过对一系列局部最优进行贪婪选择,最终得到整体最优解,所以对于选择范围十分庞大的最优化问题来说,贪婪算法是一种最直接、最实用、最快速的算法设计思想。具体来说,通常求最优化问题的全局最优解是从贪婪选择开始的,通过贪婪选择后,将一个规模较大的优化问题简化为一个规模较小的子问题,然后循环进行贪婪选择过程,最终求得一个整体近似的最优解<sup>[18]</sup>。

贪婪算法在解决问题的实际过程中,一般包括三个关键点:第一个是设置初始条件,初始条件通常是根据所需解决的实践问题产生的。第二个是如何解决局部最优化问题,最优化问题是通过设计一个优化函数来解决,优化函数的质量直接决定着整个贪婪算法的质量,它是贪婪算法中最关键的部分。第三点是结束条件的判断,一个是贪婪算法整体的结束条件,该条件要根据实际问题的要求来确定;另一个是进行局部最优选择的约束条件,该条件决定优化函数是否能得到局部最优解<sup>[18]</sup>。

## 2.2 MCTS

蒙特卡洛树搜索（MCTS）是将蒙特卡洛方法与博弈树搜索相结合形成的一种搜索方法。该方法保证在降低问题规模的同时能保持所求近似解的最优性，其中蒙特卡洛方法是利用经验平均值来代替随机变量的期望。博弈树搜索是一般博弈问题常用的方法，其主要思想是当前玩家通过对手以往的动作进行对手下一步可能采取动作的预测，进行对手可能采取动作后的博弈局面分析，进而通过回溯方式得出当前玩家应该采取的最优动作，但其受到博弈树深度的限制，在博弈树深度过大的实际问题中无法使用<sup>[19]</sup>。

MCTS 算法通常先初始化搜索树，然后对搜索树循环进行选择、扩展、模拟和反馈过程<sup>[51]</sup>，直到满足最大抽样次数或达到设置的执行时间；最后通过对每个节点的估值比较，从中选择一个决策作为本次 MCTS 算法的最佳决策<sup>[19]</sup>。

## 2.3 UML

统一建模语言（UML）是一种为面向对象系统的产品进行说明、可视化和编制文档的一种标准语言，是非专利的第三代建模和规约语言<sup>[21]</sup>。UML 是面向对象设计的建模工具，通过标准图符构成图形来描述模型，成为软件建模的通用标准语言，并且在其他领域也得到广泛应用。软件过程规定软件开发的阶段、步骤和工作，使用 UML 来描述需求模型、逻辑模型、设计模型和实现模型等。

UML 作为建模语言，是对一个软件系统建立模型，主要包括功能模型、对象模型、动态模型。它由以下几部分构成：基本语言的构成成分包括要素、关系、图；语义规则，语言的语法和语义规则；公共机制，规范说明和语言扩展。它包括四类基本构成要素：结构语言的静态构成要素，有 7 种不同种类的图；行为，语言的动态组成要素，表示对象的变化和状态；分组，对模型中对象分组进行组织；注释，对模型中对象标注和解释<sup>[21]</sup>。

用例图主要用于需求建模，描述角色以及角色与用例之间的关系。类图是描述系统中的类以及类之间关系的静态视图。活动图描述用例要求所要进行的活动以及活动间的约束关系，有利于识别并行活动。序列图是用来显示参与者如何以

一系列顺序的步骤与系统对象交互的模型。一个结构良好的模型应该在语义上是前后一致的，并且与所有的相关模型协调一致<sup>[22]</sup>。

UML 目前支持工具有：Rational Rose、Together Control、Poseidon、Visio（MS office 企业版套件）、Visual Paradigm for UML、Archware、Enterprise Architect、UModel。本文采用的建模工具是 Visio 2013。

## 2.4 Python

Python 是一种面向对象、动态解释性、交互式的编程语言。目前，Python 日趋成熟，几乎可以在所有的操作系统上使用，也可以在 Java，.NET 开发平台上使用。Python 具有易学习、易阅读、易维护、广泛的标准库、互动模式、可移植、可扩展、支持多种数据库接口、GUI 编程、可嵌入等诸多特点<sup>[23]</sup>。

Python 吸收了 Perl 和 Tcl 脚本语言的特点，Python 既具备 Tcl 的扩展性，也具备 Perl 的文本解析和匹配能力，遵循 GPL（GNU General Public License）协议。Python 支持多种编程范式，包括命令式编程、面向对象程序设计、函数式编程等。Python 还具备垃圾回收机制，采用引用计数方式自动管理内存，即当某个对象在其作用域内不再被其他对象引用时，Python 就会自动清除该对象，有效地避免发生内存泄漏。它经常作为脚本语言用于处理系统管理任务和 Web 应用开发<sup>[23]</sup>。

由于 Python 语言的动态解释性，所以程序解释执行的速度要比编译型语言偏慢。在某些对时间效率要求较高的情况下，可以采用 JIT 技术（常用的是 Pysco）或者使用 C/C++ 语言完成部分功能。可扩充性是 Python 语言的特色，某些模块（module）可采用 C/C++ 语言编写，向外提供 API，这样避免 Python 语言严谨的语法羁绊，而将程序开发的注意力转向在所要实现的任务上。Python 开发通常只用一种方法来完成一个任务<sup>[24]</sup>。

## 2.5 Django

Django 是用 Python 编写的 Web 应用框架，其代码是在实际应用环境中积累起来的，遵循 MVC 模式的开源开发框架，实际上采用 MVT 的设计模式，即模型 Model、视图 View、模版 Template。其主要目标是简便、快速的开发数据库

驱动的网站，十分注重模型和模版的重用性和 DRY 法则。框架的开发者始终致力于为开发人员节约开发时间，设计出更加简单且易维护的程序，同时又能保证程序运行的效率<sup>[21][23][24][25][26]</sup>。

Django 的开发极具应用针对性，具备很多功能模块，主要包括对象关系映射器（ORM），将数据模型和特定的数据库连接起来，为开发者提供简捷的数据库 API；基于正则表达式的 URL 分发器，框架本身并没有对 URL 分发器进行特定限制；View 系统用于处理 HTTP 请求；独立的、轻量级的 Web 开发服务器；模版系统用于定制个性化的模版，提高系统开发的效率<sup>[27][28][29]</sup>。

## 2.6 MySQL

MySQL 是一个开放源代码且最流行的小型关系型数据库管理系统。在 WEB 应用方面，MySQL 是最好的 RDBMS 应用软件之一<sup>[30]</sup>。MySQL 由于体积小、速度快、性能高、成本低、可靠性好的特点，已经成为当今世界最流行的开源数据库，广泛应用于一些中小型企业网站。MySQL 的应用架构多样，单点模式（Single），适合小型企业的软件应用；复制模式（Replication），适合中小型企业软件应用；集群模式（Cluster），适合大型企业的软件应用<sup>[23][31]</sup>。

MySQL 采用 C/C++ 语言编写，并使用多种编译器测试，源代码有着良好的可移植性；支持 Linux，Unix，Windows 等多种操作系统；为 Python，C/C++，Java，.NET 等编程语言提供 API；支持多线程、多用户，充分利用 CPU 的资源；有一套优化的 SQL 查询算法，有效地提高了数据查询的效率；提供 TCP/IP，ODBC 和 JDBC 等多种数据库连接方式<sup>[31]</sup>。

## 2.7 本章小结

本章主要是对本文研究相关的理论基础和技术进行介绍，首先贪婪算法和 MCTS 的相关知识进行介绍；然后对本文的设计模式依据 UML 和本文系统开发所使用的语言 Python、系统框架 Django 及其本身具备的特点进行介绍，说明本系统使用该技术与其特点具有极大的相关性；最后对本文使用的数据库 MySQL 进行介绍，说明该数据库本身及其使用方面的优势。

### 第三章 系统需求分析

需求分析是系统研发过程中十分重要的阶段，是系统总体设计、详细设计和系统实现的基础。需求分析阶段需要确定系统具体包含哪些模块，各个模型需包含哪些功能，系统中需要使用的数据有哪些，以及非功能需求方面需满足的系统性能<sup>[32]</sup>。开放斗地主平台主要是满足研究人员进行智能算法对抗试验并收集试验的游戏数据、多玩家在线进行多种模式的斗地主游戏并收集各种模式的游戏数据；其次是满足管理员对系统的基础数据进行管理，玩家进行游戏对局数据和出牌数据的查询；再次是满足用户进行用户名、密码、性别等个人数据的维护和密保设置<sup>[33]</sup>。系统用户分为超级管理员、管理员和普通用户（玩家），管理员由超级管理员添加或从注册用户中指定；自主注册用户默认为普通用户，普通用户也可以由超级管理员或管理员添加。综合本系统的需求分析可得如图 3.1 的系统总体用例图。

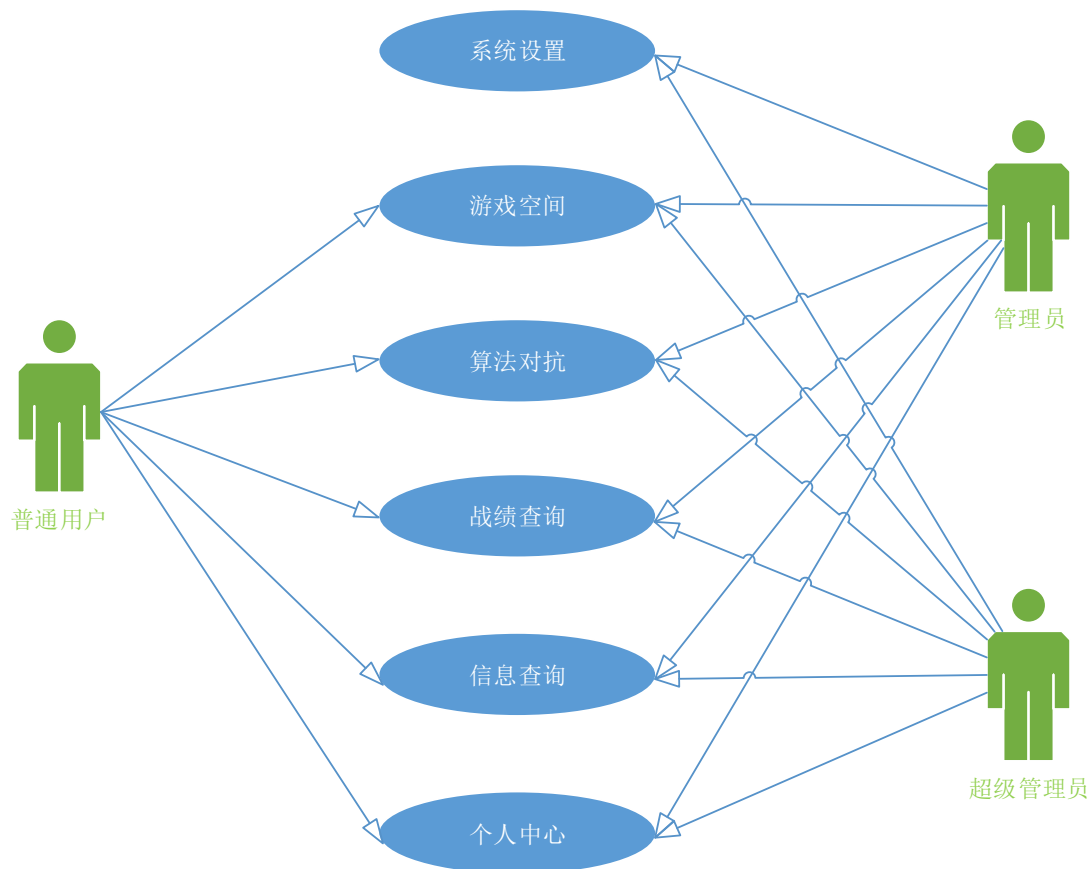


图 3.1 系统总体用例图

### 3.1 功能需求

通过对系统需求进行分析，系统功能模块可分为系统设置、游戏空间、算法对抗、战绩查询、信息查询、个人中心六大功能模块。

#### 3.1.1 系统设置

该模块主要是对系统运行所需的基础数据进行管理，管理的数据包括用户数据、菜单数据、角色数据、锦标赛数据、密保问题数据、残局数据。超级管理员默认拥有所有数据的管理权限；管理员拥有超级管理分配的数据管理权限；普通用户不具有任何数据的管理权限<sup>[34]</sup>。

本系统的权限管理采用 RBAC (Role-Based Access Control 基于角色的权限访问控制) 权限控制方式，该方式相较于传统的访问控制方式极大的简化系统权限管理，使系统角色的权限分配可根据需求动态调整。在此权限控制方式的基础上，本系统还提供用户权限的特殊设置，在继承角色权限的基础上对其进行增减，便于超级管理员或管理员对某些特殊用户权限进行动态调整。

综合以上对系统设置模块功能需求的分析，系统设置模块的用例图如图 3.2 所示：

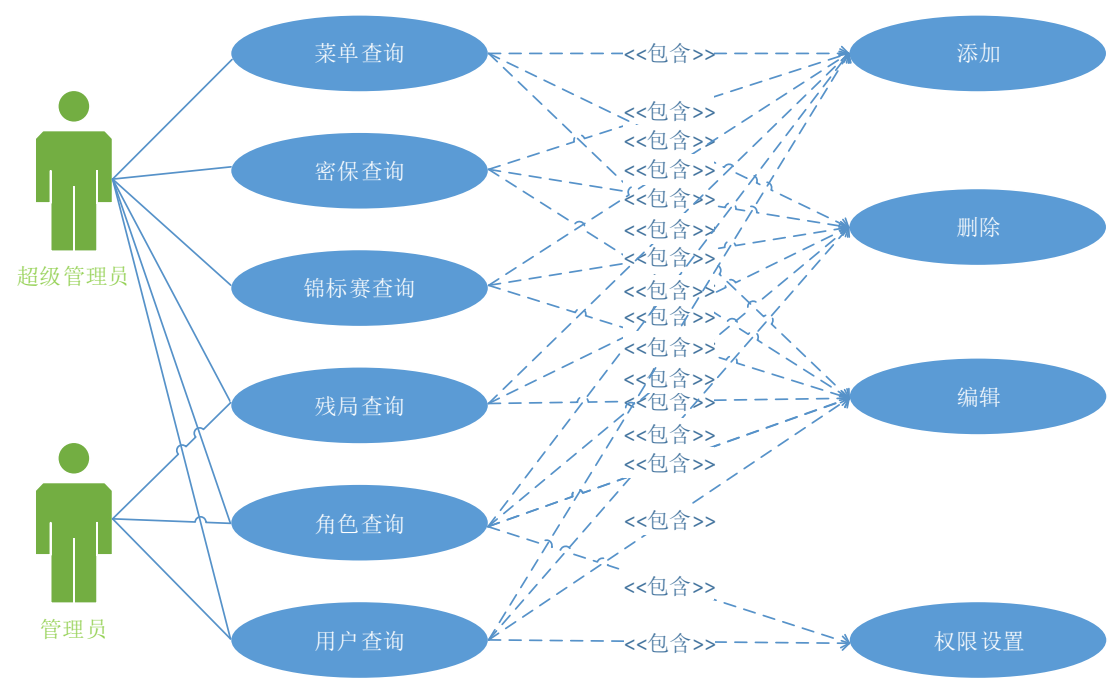


图 3.2 系统设置用例图



系统设置模块中部分用例详细描述如下所示：

表 3.1 编辑用例描述

|      |   |
|------|---|
| 用例名称 | 编辑  |
| 用例描述 | 超级管理员或管理员对系统设置数据进行编辑的过程，以残局数据为例   |
| 参与者  | 超级管理员或管理员   |
| 前置条件 | 用户已经登录系统且具有相应的管理权限  |
| 正常流程 | 1. 用户查询残局数据，系统检测是否存在残局数据。<br>2. 系统显示残局数据，用户选择一条残局数据进行编辑。<br>3. 系统显示需编辑的数据，用户进行数据编辑。<br>4. 用户提交编辑后的数据，系统进行数据规范性检查。<br>5. 系统进行提交数据保存状态检查。<br>6. 系统提示数据保存成功。 |
| 扩展流程 | 1a: 不存在残局数据。<br>1a1: 系统提示暂无残局数据。<br>4a: 提交数据规范性不符合要求。<br>4a1: 系统提示提交数据不符合规范性要求。<br>5a: 提交数据保存失败。<br>5a1: 系统提示提交数据保存失败。                                    |
| 后置条件 | 无   |

表 3.2 权限设置用例描述

|      |   |
|------|---|
| 用例名称 | 权限设置  |
| 用例描述 | 超级管理员或管理员对用户进行特殊权限设置和对角色进行角色权限的设置过程，以角色数据为例   |
| 参与者  | 超级管理员或管理员   |
| 前置条件 | 用户已经登录系统且具有相应的管理权限  |
| 正常流程 | 1. 用户查询角色数据，系统检测是否存在角色数据。<br>2. 系统显示角色数据，用户选择一个角色进行权限设置。<br>3. 系统查询当前登录用户的权限数据和所选角色的权限数据。<br>4. 用户进行角色权限的设置，提交设置后的权限数据。<br>5. 系统进行提交数据保存状态检查。<br>6. 系统提示角色权限设置成功。 |
| 扩展流程 | 1a: 不存在角色数据。<br>1a1: 系统提示暂无角色数据。<br>5a: 提交数据保存失败。<br>5a1: 系统提示角色权限设置失败。   |
| 后置条件 | 无   |

### 3.1.2 游戏空间

该模块为系统的核心模块之一，是斗地主游戏数据集的收集模块。为数据收集提供单机模式、在线模式两种基本游戏模式之外，还提供了锦标赛模式、好友

模式、残局模式三种多样化的游戏模式。在线模式游戏中为玩家提供智能出牌提示，为超级管理员提供所有在线玩家的数据查询功能，以便超级管理对目前在线玩家的游戏状态进行查询；好友模式中为玩家提供创建房间功能来进行房间的创建，房间创建成功后需对房间号进行反馈，提供加入房间的功能使玩家可通过输入房间号的方式加入游戏；在单机模式中设置不同难度的智能电脑玩家，玩家可选择与不同智能电脑玩家进行游戏；在残局模式中玩家可进行游戏关卡挑战，系统不断更新关卡数量和提升关卡的难度；锦标赛模式提供所有锦标赛的查询功能，玩家可任意选择查询的锦标赛进行报名，系统根据报名成功与否的结果为玩家作出相应的响应。所有模式的游戏过程中都需要使用一定的标识对玩家的游戏身份、出牌状态进行表示。

锦标赛淘汰是锦标赛模式游戏过程中一个关键性的过程，淘汰规则的制定应当做到公平公正。锦标赛比赛采用多轮多局制，在每轮结束时采用多种指标结合的方式进行玩家的淘汰过程，避免游戏中的运气成分对游戏公平性的影响。

综合以上对游戏空间模块功能需求的分析，游戏空间模块的用例图如图 3.3 所示：

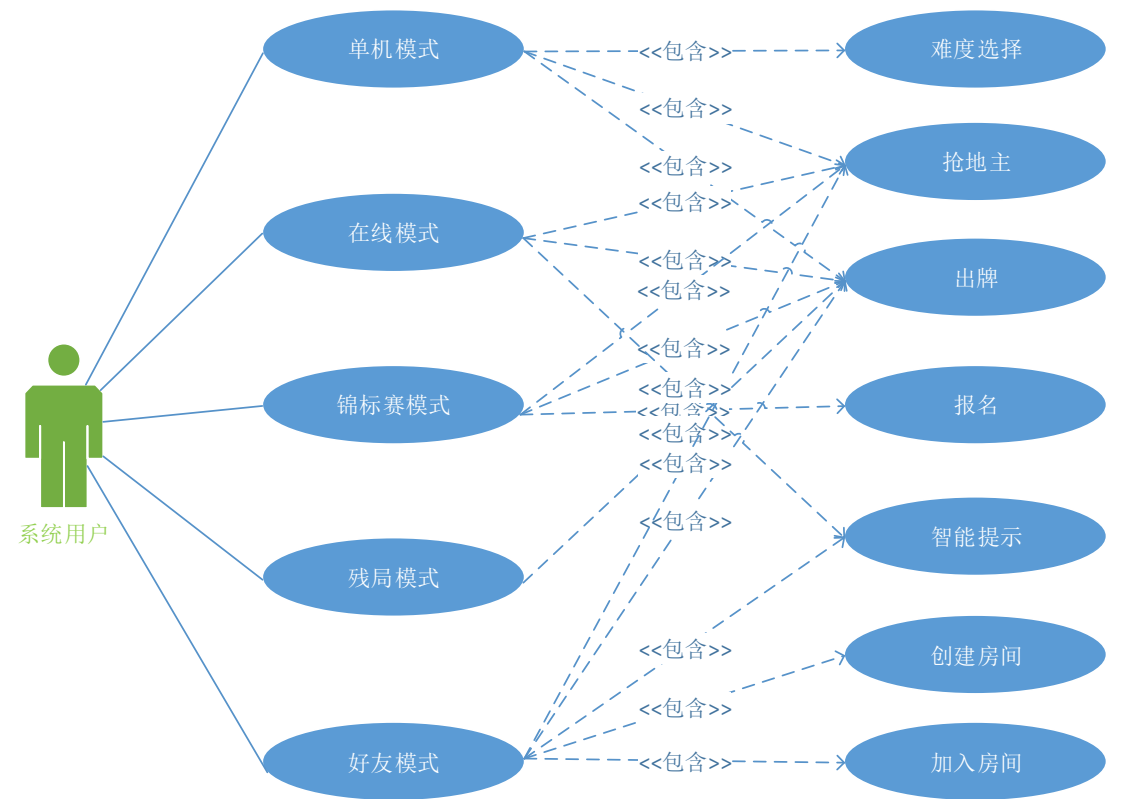


图 3.3 游戏空间用例图

游戏空间模块部分用例详细描述如下所示：

表 3.3 出牌用例描述

|      |   |
|------|---|
| 用例名称 | 出牌  |
| 用例描述 | 用户在各种模式在游戏过程中，选择手牌并打出的过程  |
| 参与者  | 系统用户  |
| 前置条件 | 用户已经登录系统且在游戏中   |
| 正常流程 | <ol style="list-style-type: none"> <li>1. 用户选择需要打出的手牌，系统判断选择的手牌张数。</li> <li>2. 系统判断出牌类型是否合法。</li> <li>3. 系统进行出牌数据保存并检查数据的保存状态。</li> <li>4. 系统更新当前游戏对局的出牌和当前用户手牌的显示。</li> </ol>      |
| 扩展流程 | <ol style="list-style-type: none"> <li>1a: 选择的手牌张数为 0。<br/>1a1: 系统提示出牌不能为空。</li> <li>2a: 出牌类型不合法。<br/>2a1: 系统提示出牌为不合法出牌类型。</li> <li>3a: 出牌数据保存失败。<br/>3a1: 系统提示出牌数据保存失败。</li> </ol> |
| 后置条件 | 系统通知下一玩家出牌  |

表 3.4 报名用例描述

|      |  |
|------|--|
| 用例名称 | 报名   |
| 用例描述 | 锦标赛模式中用户报名的过程  |
| 参与者  | 系统用户   |
| 前置条件 | 用户已经登录系统且未在锦标赛游戏中  |
| 正常流程 | <ol style="list-style-type: none"> <li>1. 用户查询锦标赛数据，系统判断是否存在锦标赛数据。</li> <li>2. 系统显示锦标赛数据，用户选择一场锦标赛进行报名。</li> <li>3. 系统检查锦标赛状态。</li> <li>4. 系统提示报名锦标赛成功。</li> </ol> |
| 扩展流程 | <ol style="list-style-type: none"> <li>1a: 不存在锦标赛数据。<br/>1a1: 系统提示暂无可报名的锦标赛。</li> <li>3a: 锦标赛的状态为进行中或者结束。<br/>3a1: 系统根据锦标赛状态进行相应的提示。</li> </ol>                    |
| 后置条件 | 系统更新用户状态为等待比赛中   |

### 3.1.3 算法对抗

该模块也是系统的核心模块之一，为研究人员提供智能算法管理、智能算法对抗试验平台和数据查询功能。智能算法管理是研究人员对上传的智能算法数据进行基本管理操作，可设置上传的算法是否与其他研究人员进行共享。智能算法对抗平台是为研究人员提供进行智能算法对抗试验的功能，使玩家可快速的搭建智能算法对抗试验的环境，提高研究人员的工作效率，研究人员在试验前可对参

数进行设置、对智能算法进行选择来满足不同试验的要求，在试验过程中系统对游戏数据进行收集。数据查询是为研究人员提供以往试验数据的查询和导出功能，以便研究人员对试验结果进行分析，开展下一步智能算法的研究工作。

综合以上对算法对抗模块功能需求的分析，算法对抗模块的用例图如图 3.4 所示：

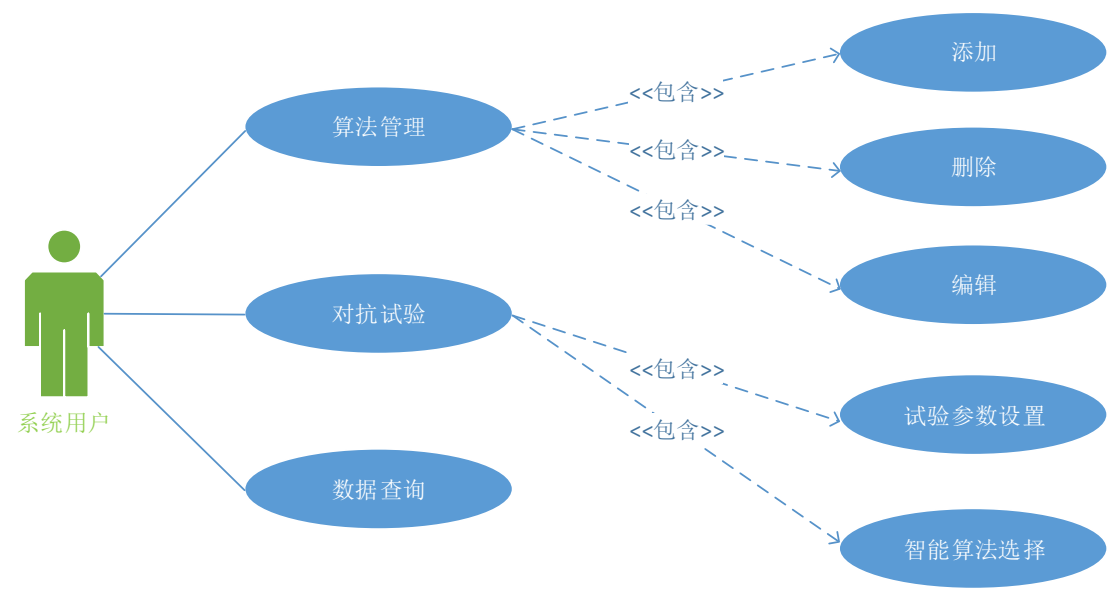


图 3.4 算法对抗用例图

战绩查询模块部分用例详细描述如下所示：

表 3.5 对抗试验用例描述

|      |   |
|------|---|
| 用例名称 | 智能算法对抗试验  |
| 用例描述 | 研究人员进行智能算法对抗试验的过程   |
| 参与者  | 系统用户  |
| 前置条件 | 用户已经登录系统且用户已上传智能算法  |
| 正常流程 | 1. 系统查询用户是否已上传智能算法。<br>2. 系统查询可供选择的智能算法，系统判断是否存在可供选择的算法。<br>3. 用户进行智能对抗试验的算法选择。<br>4. 用户进行试验参数的设置，系统验证试验参数是否合法。<br>5. 系统开始智能算法对抗试验。 |
| 扩展流程 | 1a: 用户未上传智能算法。<br>1a1: 系统提示请先上传智能算法。<br>2a: 不存在可供选择的算法。<br>2a1: 系统提示暂无可选择的智能算法。<br>4a: 试验参数不合法。<br>4a1: 系统提示试验参数设置不合法。              |
| 后置条件 | 无   |

3.1.4 战绩查询

该模块是对各种游戏模式的斗地主游戏对局数据进行展示，展示的对局数据包括本局玩家、玩家对应身份、本局赢家、叫地主分数、玩家对应的手牌、已出手牌等。在此基础之上，玩家可对每局游戏的详情出牌数据进行列表查询和回放查询，列表查询是对用户出牌数据进行列表展示，使用标识进行当前用户的区分；回放查询是对整个游戏的出牌过程进行重现，用户可以对整个游戏的回放过程进行人为控制，使用此方式让玩家观看游戏中的出牌情况及分析整体牌局形势，从中发现自我优劣势所在，提高自身的竞技水平。该模块还采用游戏模式和排名方式相结合的查询方式对当前平台玩家进行积分制排名，排名结果采用可视化方式进行展示，可选择某一玩家进行个人战绩统计查询。玩家能对积分制排名的游戏模式、排名方式进行选择，游戏模式包含所有系统提供的斗地主游戏模式，排名方式包含多种不同的积分计算方式。

综合以上对战绩查询模块功能需求的分析，战绩查询模块的用例图如图 3.5 所示：

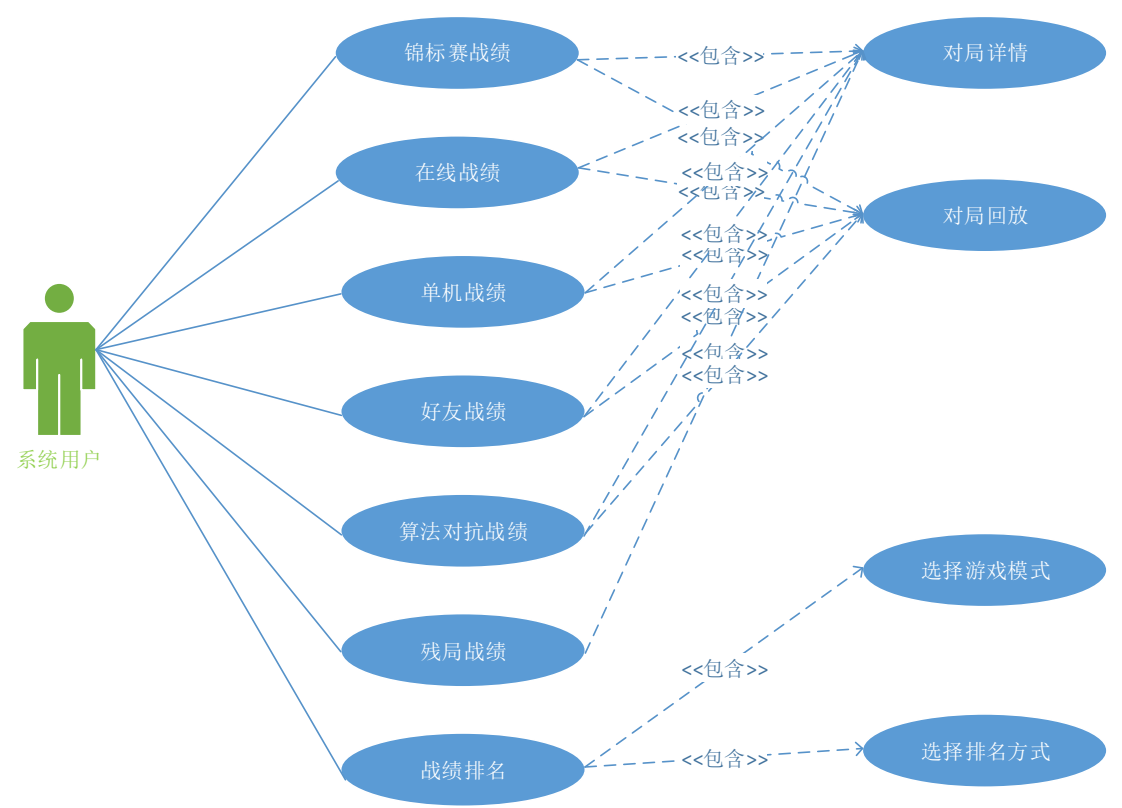


图 3.5 战绩查询用例图

战绩查询模块部分用例详细描述如下所示：

表 3.6 锦标赛战绩用例描述

|      |   |
|------|---|
| 用例名称 | 锦标赛战绩   |
| 用例描述 | 用户对锦标赛模式的游戏对局进行查询的过程  |
| 参与者  | 系统用户  |
| 前置条件 | 用户已经登录系统  |
| 正常流程 | 1. 用户查询参加过的锦标赛数据，系统判断用户是否参加过锦标赛。<br>2. 用户选择某一锦标赛，查询该锦标赛的游戏对局数据，系统判断是否有对局数据。<br>3. 系统显示对局数据。<br>4. 用户翻页进行对局数据查询。 |
| 扩展流程 | 1a: 用户未参加锦标赛。<br>1a1: 系统提示尚未参加过锦标赛。<br>2a: 不存在对局数据。<br>2a1: 系统提示暂无游戏对局数据。                                       |
| 后置条件 | 无   |

表 3.7 对局回放用例描述

|      |  |
|------|--|
| 用例名称 | 对局回放   |
| 用例描述 | 用户通过回放形式查看游戏对局详情的过程  |
| 参与者  | 系统用户   |
| 前置条件 | 用户已经登录系统   |
| 正常流程 | 1. 用户查询某一游戏模式的对局数据，系统判断是否有对局数据。<br>2. 用户选择某一游戏对局，查询该游戏对局详细出牌数据，系统判断是否有详细出牌数据。<br>3. 系统显示模拟出牌场景，开始模拟出牌。 |
| 扩展流程 | 1a: 不存在对局数据。<br>1a1: 系统提示暂无游戏对局数据。<br>2a: 不存在详细出牌数据。<br>2a1: 系统提示查询详细出牌数据失败。                           |
| 后置条件 | 无  |

### 3.1.5 信息查询

该模块包含对用户登录历史、积分数据进行查询和展示。登录历史展示数据包括用户名、登录时间、登录地址等；通过对登录历史数据中的登录时间及登录地点的分析，采用明显的颜色标注异常的登录记录。积分情况展示数据包括用户名、当局积分、总积分、地主分数、炸弹数、是否春天等；在此基础之上，还提供与战绩查询模块相同功能对每局游戏的出牌详情进行查询，用户可将当局积分与当局的出牌详情想结合进行分析，总结游戏经验、提升游戏竞技水平。

综合以上对信息查询模块功能需求的分析，信息查询模块用例图如图 3.6 所示：

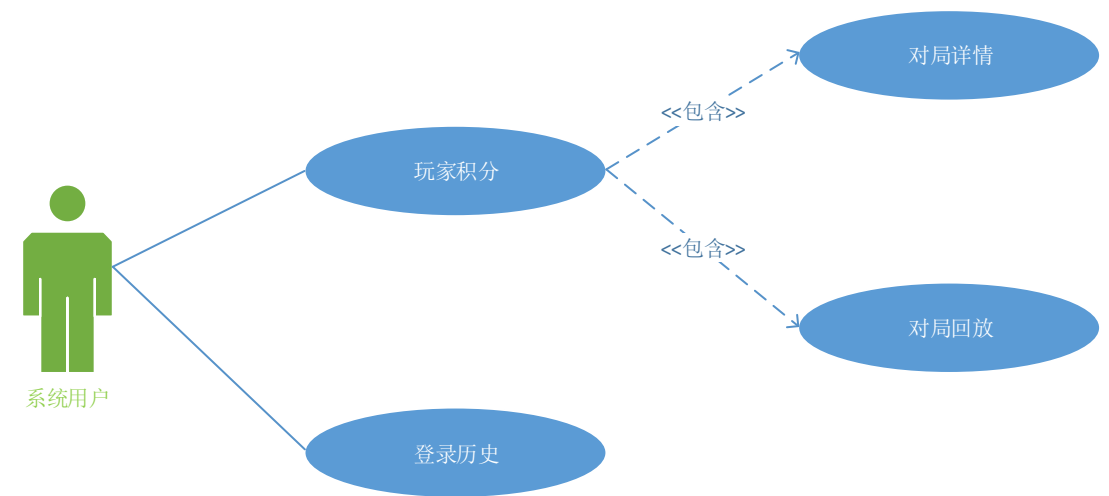


图 3.6 信息查询用例图

信息查询模块部分用例详细描述如下所示：

表 3.8 玩家积分用例描述

|      |  |
|------|--|
| 用例名称 | 玩家积分   |
| 用例描述 | 用户通过选择游戏模式对积分进行查询的过程   |
| 参与者  | 系统用户   |
| 前置条件 | 用户已经登录系统   |
| 正常流程 | 1. 用户选择游戏模式。<br>2. 系统查询该游戏模式对应的积分数据，判断是否有积分数据。<br>3. 系统显示积分数据。<br>4. 用户翻页进行积分数据查询。 |
| 扩展流程 | 2a: 不存在积分数据。<br>2a1: 系统提示暂无积分数据。   |
| 后置条件 | 无  |

3.1.6 个人中心

该模块主要是用户对个人数据进行维护，包括密码修改、头像修改、信息修改、密保设置、密码重置。密码修改、头像修改、信息修改是进行个人数据维护的基础功能。密保设置为用户根据系统提供的密保问题进行密码保护设置，是用户能进行密码重置的前提和保障。密码重置是在密码忘记且用户名未忘记的情况下，验证用户名和用户设置的密保问题答案对密码进行重置的功能<sup>[35]</sup>。

综合以上对个人中心模块功能需求的分析，个人中心模块用例图如图 3.7 所示：

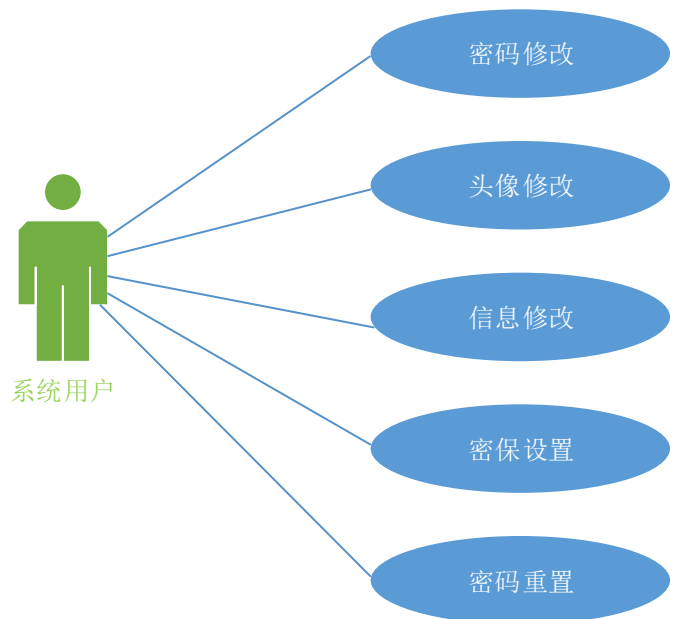


图 3.7 个人中心用例图

个人中心模块部分用例详细描述如下所示：

表 3.9 密保设置用例描述

|      |  |
|------|--|
| 用例名称 | 密保设置   |
| 用例描述 | 用户进行密保问题选择和答案设置的过程   |
| 参与者  | 系统用户   |
| 前置条件 | 用户已经登录系统   |
| 正常流程 | 1. 用户提供登录密码，系统进行身份验证。<br>2. 用户选择密保问题并设置问题对应的答案，系统判断问题及答案是否符合规范。<br>3. 用户再次进行密保问题答案的设置，系统检查两次输入的答案是否一致。<br>4. 用户确认并保存密保设置数据，系统检查数据保存状态。<br>5. 系统提示密保设置成功。             |
| 扩展流程 | 1a: 身份验证失败。<br>1a1: 系统提示密码不正确，身份验证失败。<br>2a: 用户设置的问题或者答案不符合规范。<br>2a1: 系统根据错误类型进行提示。<br>3a: 两次设置的答案不一致。<br>3a1: 系统提示两次设置的密保答案不一致。<br>4a: 数据保存失败。<br>4a1: 系统提示密保设置失败。 |
| 后置条件 | 系统更新用户当前密保设置状态为已设置   |



## 3.2 非功能需求

非功能性需求，指的是系统中保证系统安全可靠、界面需求、可维护性、可扩展性、系统性能等方面的相应需求，其在软件需求中是必不可少的部分，对软件需求的影响也是不可忽略的<sup>[36][37]</sup>。

1、安全可靠。软件必须具有高可靠性、高稳定性、高可用性，除在硬件上考虑系统本身的容错性、冗余性、兼容性等方面之外，软件自身还要充分考虑数据的容错性，并且在系统关键环节建立应急措施，以避免系统错误造成数据丢失。另外，系统安全性包括系统本身设计、防止内部或外界对整个系统的攻击及病毒的破坏、系统用户数据保密性和安全性等各方面的内容。

2、简洁明了的操作界面。由于该系统是通用性的，用户大部分不具备相关的专业知识，系统的界面设计应该做到简洁、直观，尽可能的为用户提供便捷的操作，提高人机交互的体验感，减少用户的记忆以及操作的步骤和次数。

3、可维护性。这要求系统在出现故障后必须能够在最短时间之内对系统进行故障修复，使系统恢复到正常运行状态，故障出现的次数较多及修复速度较慢都会给用户带来极差的体验，故需要尽可能保证线上系统在正常状态运行。

4、可扩展性。可扩展性是目前软件系统必不可少的设计原则之一，保证了软件系统的拓展能力。可通过软件框架、设计良好的类层次结构、良好的代码设计、标准一致的命名规范等方式实现。

5、系统性能。系统的部署必须考虑到具体的软硬件、网络环境、用户量等因素，选择适宜的部署环境，保证系统能够在正常状态下运行，满足用户对系统响应速度、并发用户、请求成功率、抗压能力等方面的要求。

## 3.3 本章小结

本章对系统进行需求分析，需求分析阶段是进行系统设计和系统实现的基础，是软件生命周期过程中最重要的阶段。首先通过需求分析明确本文的系统分为六个功能模块；然后对六个功能模块进行详细的阐述，说明各个模块所需完成的功能；最后对非功能需求进行描述，保证系统在完成功能的同时也能满足运行过程中的性能要求。

## 第四章 系统设计

### 4.1 系统总体设计

系统总体设计是在系统需求分析的基础之上，对系统做进一步的设计。在本系统中总体设计主要解决确定系统体系结构和功能架构两大问题。

#### 4.1.1 系统体系结构设计

本系统采用浏览器/服务器（Browser/Server, B/S）结构结合 Django 框架进行体系结构设计，系统体系结构如图 4.1 所示：

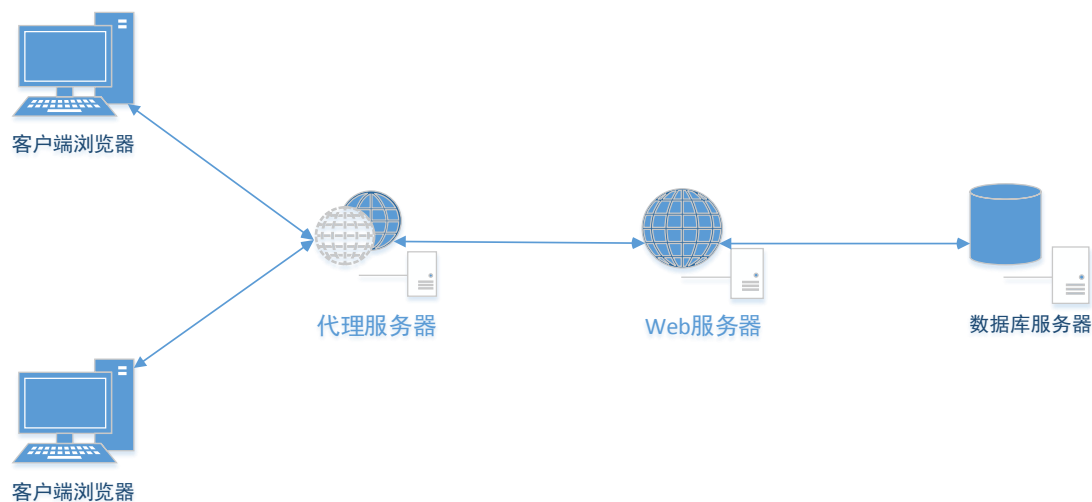


图 4.1 系统体系结构图

B/S 体系结构是利用 Web 技术，结合浏览器强大的脚本解释能力，通过浏览器实现原本需要复杂的、特定的、专业的软件才能完成的功能，节约大量的开发成本和维护成本，用户通过浏览器便可享受服务、共享性强、使用方便快捷、实用性强<sup>[38][39]</sup>。

在 B/S 体系结构中，应用程序以静态或动态网页的形式存储在 Web 服务器上。本文系统的设计如下：用户通过客户端的浏览器上输入网址进行系统访问，通过 Nginx 代理服务器的地址转化，访问使用 uwsgi 或 Daphne 部署在 Web 服务器上所对应的请求资源，Web 服务器根据客户端请求向 MySQL 数据库服务器

发送数据操作命令，数据库服务器对数据操作命令进行处理，最后将结果逐层返回到客户端并通过浏览器解析和显示<sup>[40]</sup>。

### 4.1.2 系统架构设计

根据对系统进行的需求分析，系统的架构设计应当在满足功能需求的基础上，尽可能的满足非功能需求。系统基于 Django 框架的 MVC 分层结构对系统架构进行设计，设计完成的系统架构层次结构为：表现层、控制层、业务逻辑层、数据模型层<sup>[15][41][42]</sup>。系统架构设计图如图 4.2 所示：

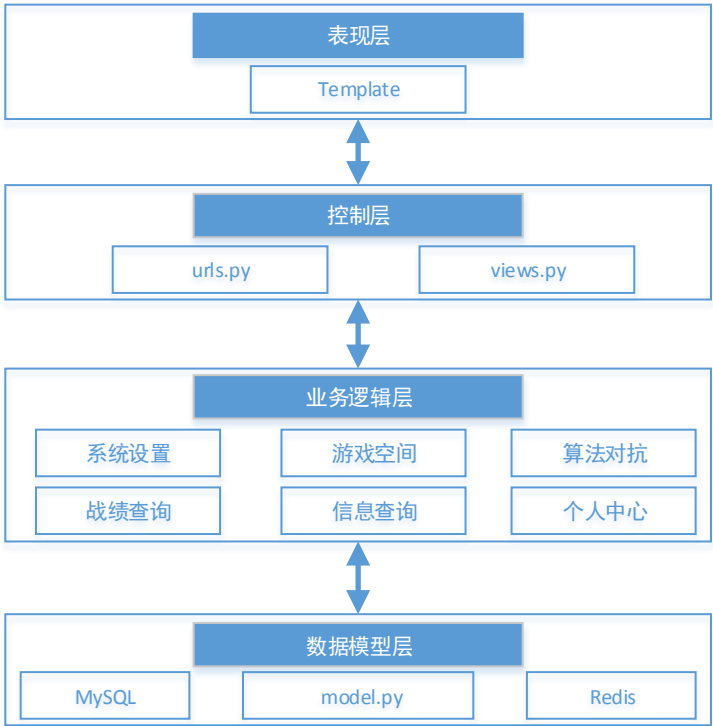


图 4.2 系统架构图

1、系统表现层。系统表现层的主要功能是为了实现系统管理数据、查询数据、数据统计排名、斗地主游戏界面的呈现，在 Django 框架中表现层使用 Template 来对页面数据进行处理和渲染<sup>[27]</sup>。在本系统中采用 Template 模版与 Layui 前端框架相结合方式实现具体页面的数据处理和渲染功能，Template 模版负责对接接口返回数据进行处理并渲染，Layui 前端框架负责对页面外观进行美化和对布局元素进行控制。

2、系统控制层。系统的控制层主要是实现将来自表现层的请求信息传递到业务逻辑层的转发功能，即系统的路由功能。在 Django 框架中路由功能是由

URLconf 模块实现，通过 Django 框架中的视图文件 `views.py` 和路由地址文件 `urls.py` 创建路由映射表提供路由映射功能，在 Django 框架中的路由是通过正则表达式进行路由匹配的，数据接口按照 REST 架构风格进行设计与开发<sup>[43]</sup>。系统采用分模块的设计方式，按照模块的方式进行路由的设计与开发，通过这样的方式达到降低模块之间的耦合度、提高模块内部之间的聚合度。

3、系统业务逻辑层。系统的业务逻辑层主要是对系统中所有的业务逻辑进行处理，系统设置模块主要处理的是对管理数据的增删查改操作；游戏空间模块主要处理的是实时通讯类的对外接口管理、智能斗地主算法的调用、对游戏数据的添加和更新；算法对抗模块主要是处理智能算法的上传、提供对抗试验的场景、收集试验的数据；战绩查询和信息查询模块主要处理的是对游戏数据查询和统计；个人中心模式主要处理的是用户对个人数据的查询和更新；其他的业务逻辑有系统注册、系统登录登出、用户身份验证等。

4、系统数据模型层。系统的数据模型层主要实现对数据库的相关操作。Django 框架中使用 `model.py` 对数据库的表结构进行设计，通过命令行操作进行数据库表的创建，Django 框架的对象关系映射（ORM）提供便于理解和方便操作的对象化数据存取方法，将对数据库的操作转化成为对类属性和方法的操作，减少了对数据库 SQL 语句的编写；实现对数据模型与数据库的解耦，屏蔽不同数据库操作的差异，系统需更换数据库时只需对数据库配置进行修改即可，无需修改任何代码，通过 ORM 的运用，提高开发的效率<sup>[28]</sup>。Redis 主要对内存中的数据进行管理。

### 4.1.3 系统功能架构设计

根据对功能需求的分析，系统功能主要包括系统设置、游戏空间、算法对抗、战绩查询、信息查询、个人中心六大功能模块。系统中的功能模块分别与功能需求分析中各模块的用例图对应，保证功能需求中的每个功能点都能在系统功能中得到体现，保证从需求到功能的设计或从功能到需求的回溯对应关系都是完整的，确保设计的功能完全满足整个系统的需求，避免按功能设计进行开发时出现功能遗漏的情况，避免这样的情况带来返工对整个工程造成拖延工期、经济损失、资源浪费等严重后果。系统功能架构设计图如图 4.3 所示：

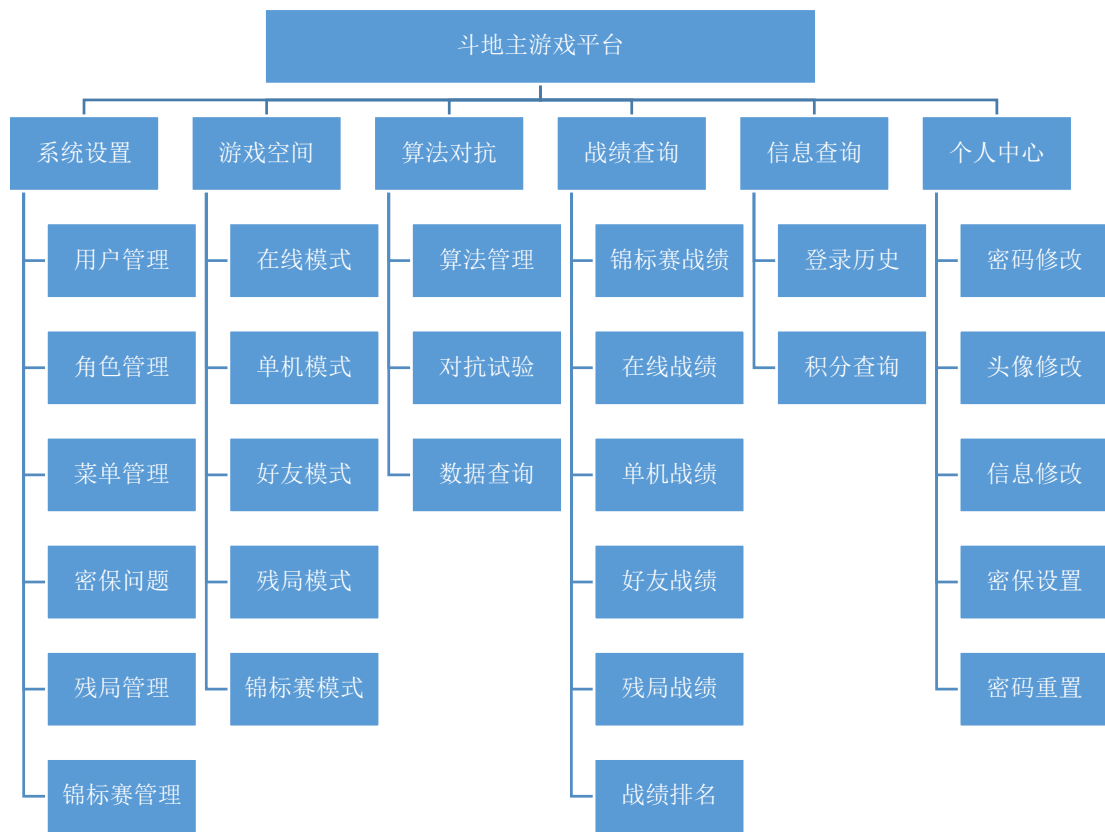


图 4.3 系统功能架构图

## 4.2 系统详细设计

系统详细设计是在总体设计的基础上对功能模块下的子功能进行详细设计，描述各个子功能所需完成的工作、其内部的活动、数据结构等。本系统的功能模块设计主要采用 UML 设计中的包图、活动图对功能的详细设计进行描述。包图的作用是对系统按照功能模块包含的子功能进行描述；活动图的作用是描述一个子功能执行过程中所完成的活动或者动作，它们按照一系列的顺序进行执行，表明它们在系统中发生的先后顺序。数据库设计分为实体类设计和物理设计两个阶段：实体类设计是根据需求分析进行实体抽象的过程，借助实体类图对实体关系进行表示；物理设计是对实体类属性的存储结构进行设计，明确实体类各属性的物理存储类型。

4.2.1 功能模块设计

4.2.1.1 系统设置

系统设置模块管理的数据可分为两大类：基础数据和游戏数据。

基础数据包括用户数据、角色数据、菜单数据、密保数据。用户管理是对系统的用户的用户名、密码、角色、性别等数据进行管理；角色管理是对系统中存在的角色的角色名称、角色菜单数据进行管理；菜单管理是对系统的菜单的名称、地址、图标、上级菜单等数据进行管理，菜单数据是系统菜单导航栏的唯一数据来源；密保问题管理是对系统密保问题设置中所提供选择的问题进行管理。

游戏数据包括残局数据、锦标赛数据。残局管理是对系统残局挑战中的残局的挑战的手牌、难度、关卡等数据进行管理；锦标赛管理是对锦标赛中的锦标赛的名称、每轮对战局数、玩家人数限制等数据进行管理。

综合以上的分析，可得系统设置模块数据管理的活动图如图 4.4 所示：

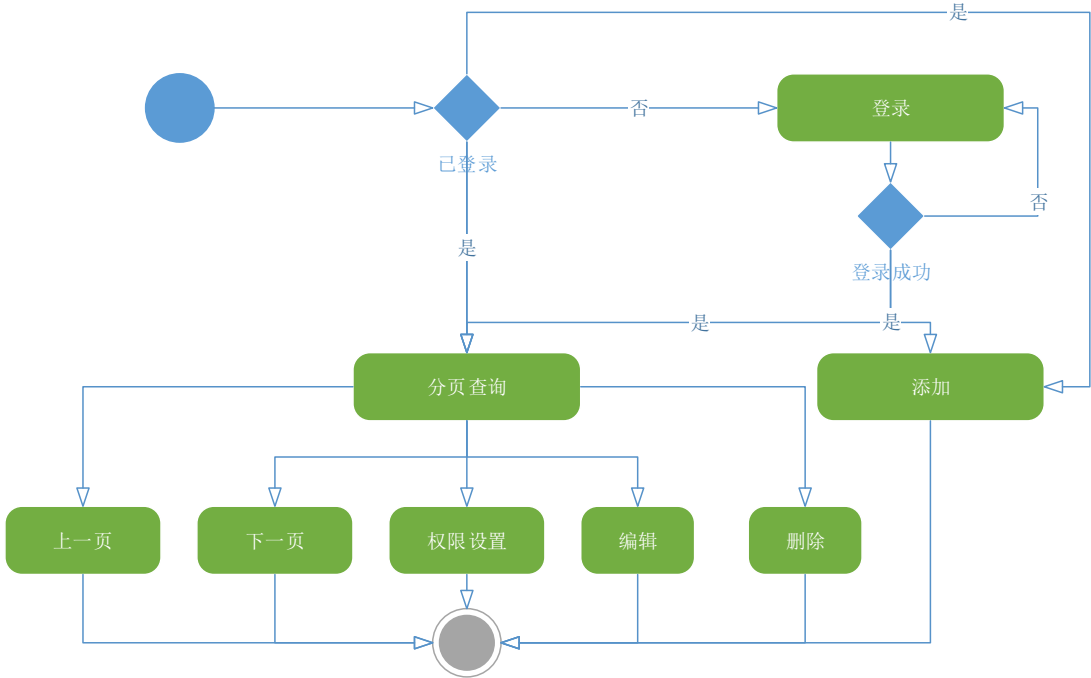


图 4.4 数据管理活动图

根据系统设置模块对数据管理功能和活动的分析，可得数据管理功能的界面设计如下：第一，数据列表界面，该界面对所管理的数据通过列表形式进行显示，

提供数据添加、编辑、删除、查看操作按钮，当数据为用户数据或角色数据时还需提供权限设置操作按钮。第二，数据添加界面，该界面能进行添加数据的输入或选择，对不合法的输入数据进行提示，提供数据添加确认操作按钮。第三，数据编辑界面，该界面对需进行编辑的数据进行显示，确保可编辑的数据能进行输入或选择，对不合法的输入数据进行提示，提供数据编辑确认操作按钮。第四，权限设置界面，该界面对用户或角色的权限数据采用树形结构进行显示，显示的权限数据可进行勾选或取消操作，提供权限设置确认操作按钮。第五，数据查看界面，该界面对查看的数据进行显示且数据处于不可编辑状态。以用户管理功能为例，其管理功能流程图如图 4.5 所示：

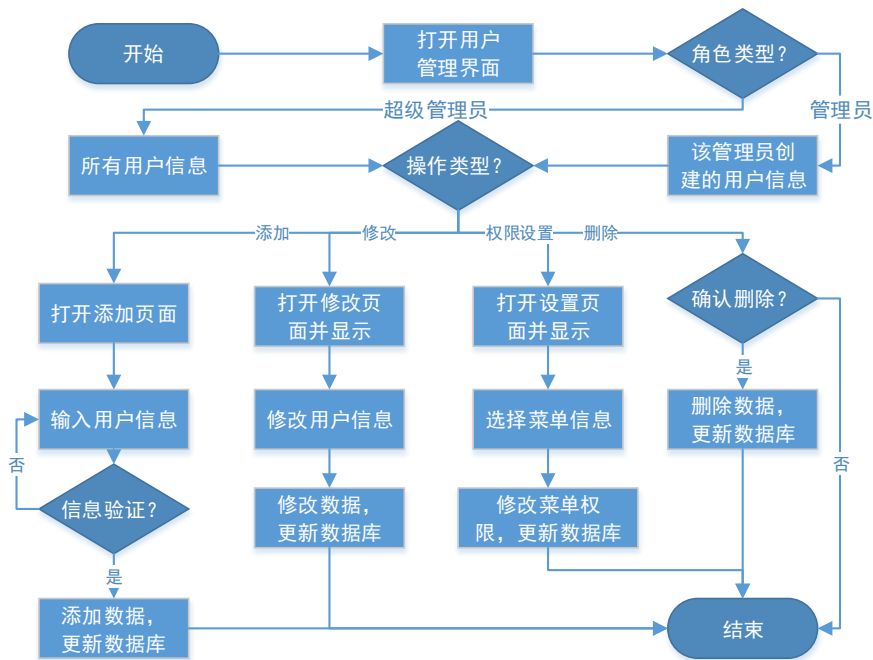


图 4.5 用户管理流程图

4.2.1.2 游戏空间

游戏空间模块的功能包括单机模式、在线模式、好友模式、残局模式、锦标赛模式五种模式的斗地主游戏。

在斗地主游戏中，扑克牌（不区分花色）对应的数字表示及其在系统中的显示符号<sup>[53]</sup>的设计如表 4.1 所示：

表 4.1 数字表示及显示符号表

|     |     |    |   |   |   |   |   |    |    |
|-----|-----|----|---|---|---|---|---|----|----|
| 扑克牌 | 3-9 | 10 | J | Q | K | A | 2 | 小王 | 大王 |
|-----|-----|----|---|---|---|---|---|----|----|

|      |     |    |    |    |    |    |    |    |    |
|------|-----|----|----|----|----|----|----|----|----|
| 数字表示 | 3-9 | 10 | 11 | 12 | 13 | 14 | 16 | 17 | 18 |
| 显示符号 | 3-9 | T  | J  | Q  | K  | A  | 2  | L  | B  |

出牌类型及其数字对应关系如表 4.2 所示

表 4.2 出牌类型及其对应表

|      |    |    |    |     |      |       |       |      |        |    |
|------|----|----|----|-----|------|-------|-------|------|--------|----|
| 出牌类型 | 不出 | 单牌 | 对子 | 三不带 | 三带一  | 三带一对  | 连子    | 炸弹   | 四带二    | 王炸 |
| 数字表示 | 0  | 1  | 11 | 111 | 1112 | 11122 | 12345 | 1111 | 111123 | 12 |

在表 4.2 中当对子出牌类型的长度为其类型长度的 3 及以上倍数时为连对；当三带出牌类型中的长度为其类型长度的 2 及以上倍数时为飞机。

斗地主游戏设计原则有：尽可能使玩家操作简单快捷、人机交互体验好、保证玩家游戏公平、游戏中的提示准确又简洁易懂、游戏中故障处理人性化、游戏的响应速度应保证在玩家可接受范围内。

斗地主游戏设计过程分为以下几个步骤：

第一，出牌类型。本系统设计的出牌类型包括不出、单牌、对子、连对、三不带、三带一、三带一对、连子、炸弹、四带二、飞机、王炸。

第二，出牌类型限制。连子必须 5 张及以上且 2 和大小王不能参与，连对必须 3 对及以上且 2 和大小王不能参与，飞机必须由两个及以上的三带一或三带一对组成且 2 不能参与。

第三，出牌的大小关系。单牌、对子、连子、连对同出牌类型比较大小时根据最小牌面的大小关系确定；三不带、三带一、三带一对同出牌类型比较大小时根据三个一样牌的牌面大小关系确定；飞机出牌类型比较大小时根据最小的三个一样牌的牌面大小关系确定；四带二出牌类型比较大小时根据四个一样牌的牌面大小关系确定；炸弹出牌类型比较大小时根据四个一样牌的牌面大小关系确定，炸弹可以大过其他非炸弹出牌类型，炸弹出牌类型中最大为王炸，除炸弹外其他不同的出牌类型不能交叉进行大小比较。

第四，发牌和抢地主规则。二者都采用随机数方式确定开始的玩家位置，发牌过程中每人先发 17 张牌，留三张地主牌，待地主确定后分配给地主玩家；抢地主采用 0、1、2、3 分制，0 分表示不抢地主，后面玩家抢地主分数必须比前面抢地主的玩家分数高，抢地主结束后分数最高的玩家为地主。

第五，出牌过程的规则。作为本轮第一个出牌的玩家，可打出任意出牌类型的牌，但是不能不出。非本轮第一个出牌的玩家时，只能打出已出扑克牌所属出牌类型的牌或者炸弹，当打出一样出牌类型的牌时，所出牌的牌面必须符合大小



关系比较的规则，否则为不合法的出牌；若不能大过已出扑克牌或不需出牌时，可选择跳过本轮出牌。

第六，游戏获胜的规则。斗地主游戏获胜的规则为玩家把手中的扑克牌打完，第一个把手牌打完的玩家为本局游戏的赢家。

单机模式是当前玩家与两个智能电脑玩家进行的斗地主游戏。用户可选择与不同难度的智能电脑玩家进行游戏，智能电脑玩家分为简单、一般、困难三个等级。在游戏过程中，发牌和抢地主开始位置采用随机数进行确定，电脑玩家抢地主也采用随机数完成；使用图片标识对玩家的身份进行标识，玩家身份包括地主、农民；使用出牌状态标识正在出牌的玩家；在游戏界面的下方实时显示已出牌记录，包括所打出的手牌、出牌玩家位置数据<sup>[5]</sup>。单机模式的活动图如图 4.6 所示：

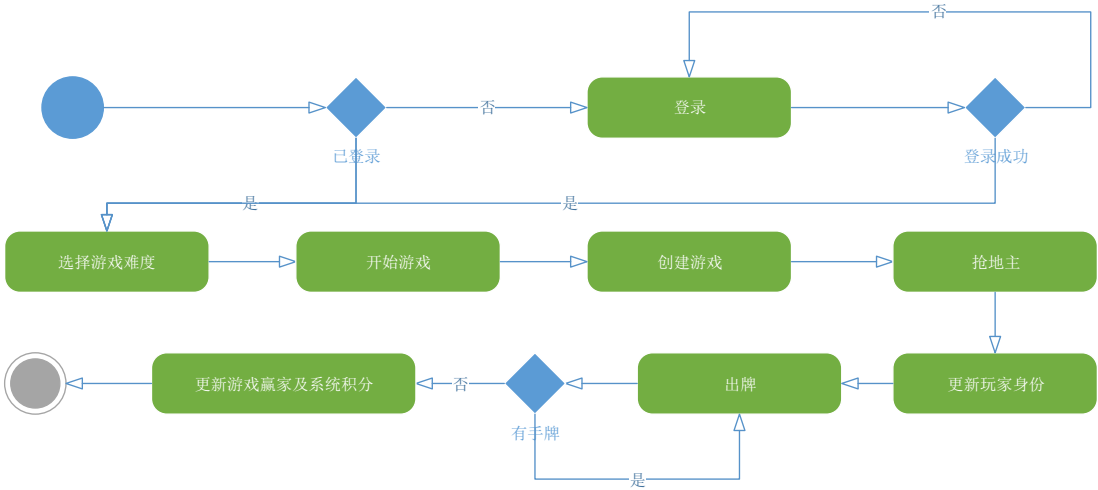


图 4.6 单机模式活动图

在线模式是玩家之间在线进行斗地主游戏，用户通过选择游戏桌的方式进入游戏，一个玩家同时只能参与一张游戏桌的斗地主游戏。当该游戏桌的玩家人数达到 3 人数时，首先对斗地主游戏对局进行创建，开始发牌过程，然后依次进行抢地主过程，最后轮流进行出牌过程，直到斗地主游戏的结束，诞生本局斗地主游戏的赢家。对游戏中各个过程的数据进行存储，当存储过程中出现错误时，系统则根据错误类型作出不同的响应，在斗地主游戏过程中为用户提供智能出牌提示功能。同单机模式一样使用标识表示玩家的出牌状态和游戏身份。若玩家身份为超级管理员时，可对在线模式游戏中的玩家数据进行查询。在线模式的活动图如图 4.7 所示：

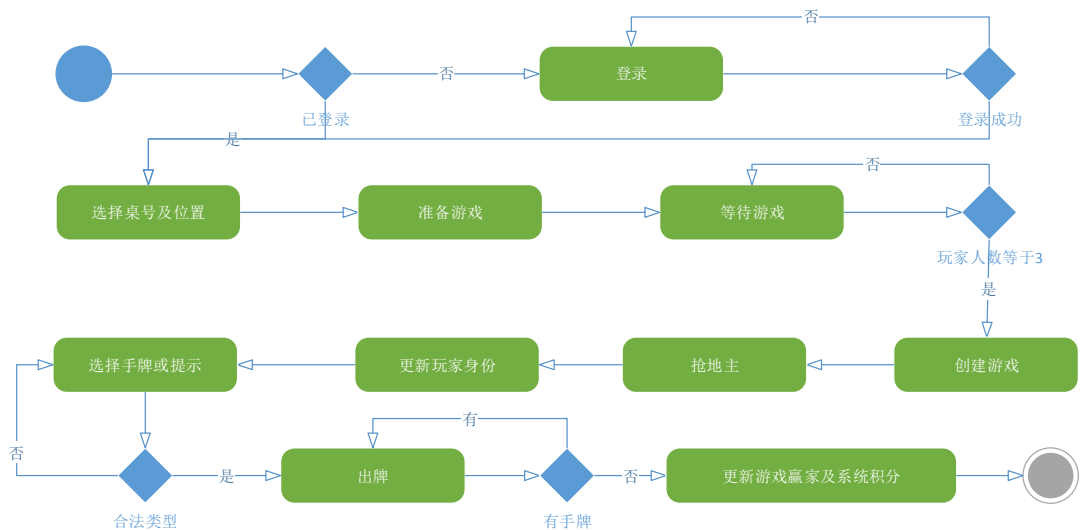


图 4.7 在线模式活动图

好友模式是玩家通过创建房间或加入房间的方式进行斗地主游戏，其限制条件为一个玩家同时只能在一个房间进行游戏。第一种，玩家通过创建房间来获取房间号，等待其他玩家加入房间。第二种，玩家通过输入房间号加入到指定的房间，若加入失败时，则系统给出用户相应的提示信息。当有新玩家加入房间时，在房间中等待的玩家会收到系统的提示；斗地主游戏的过程与在线模式游戏过程相同，同样对游戏过程中的数据进行存储、对错误作出相应的响应、使用标识表示玩家的出牌状态和游戏身份。好友模式的活动图如图 4.8 所示：

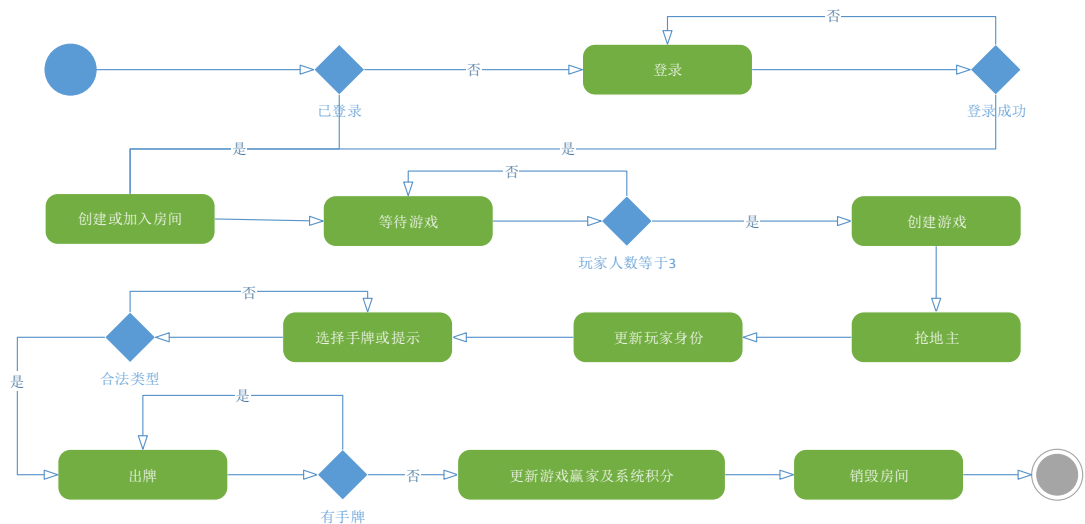


图 4.8 好友模式活动图

锦标赛模式是玩家对系统所提供的斗地主锦标赛进行报名，与其他玩家共同进行斗地主竞技。若用户报名锦标赛时，该锦标赛已经处于进行中或已结束状态，

则用户报名该锦标赛失败，系统给出用户相应的提示信息；报名锦标赛成功的玩家，自动进入等待状态，当报名玩家人数到设定的人数时便开始锦标赛，在锦标赛的游戏过程中可与其他玩家共同在线进行斗地主竞技。当玩家等待时间超过设定的等待时间时，系统则需作出相应的响应。锦标赛模式的斗地主游戏过程与在线模式游戏过程相同，同样对游戏过程的数据进行存储、使用标识对玩家的出牌状态和游戏身份进行表示，但考虑到锦标赛的公平公正性，游戏过程中不提供智能提示功能。

系统当前制定的锦标赛规则包括积分规则和淘汰规则。

在每局游戏结束之后，系统根据对局结果对用户的积分进行计算和更新，积分规则设置如下：地主获胜时地主的积分加两分、地主的胜场加一、农民的积分各减一分；农民获胜时农民的积分加一分、农民的胜场加一、地主积分减两分。在有人退出游戏导致游戏结束时，退出游戏的玩家无法再次加入该锦标赛，未退出游戏的玩家积分加一分、胜场加一。

在每轮游戏结束时，系统根据本轮玩家的积分和胜场情况进行淘汰，淘汰规则分四步：

第一步，选取本轮比赛中胜场数超过一半的玩家，验证玩家人数是否符合开始下一轮比赛的条件，符合条件则进入下一轮比赛，否则计算尚差玩家人数  $M$ 。

第二步，对剩余玩家根据积分进行排名，选取排名前  $M$  且积分大于平均积分的玩家，验证玩家人数是否符合开始下一轮比赛的条件，符合条件则进入下一轮比赛，否则计算尚差玩家人数  $N$ 。

第三步，对剩余的玩家根据积分及胜场进行综合排名，选取排名前  $N$  的玩家，验证玩家是否符合开始下一轮比赛的条件，符合条件则进入下一轮比赛，否则计算尚差玩家人数  $P$ 。

第四步，从剩余的玩家中随机选取  $P$  名玩家，进入下一轮比赛。

锦标赛模式的活动图如图 4.9 所示：

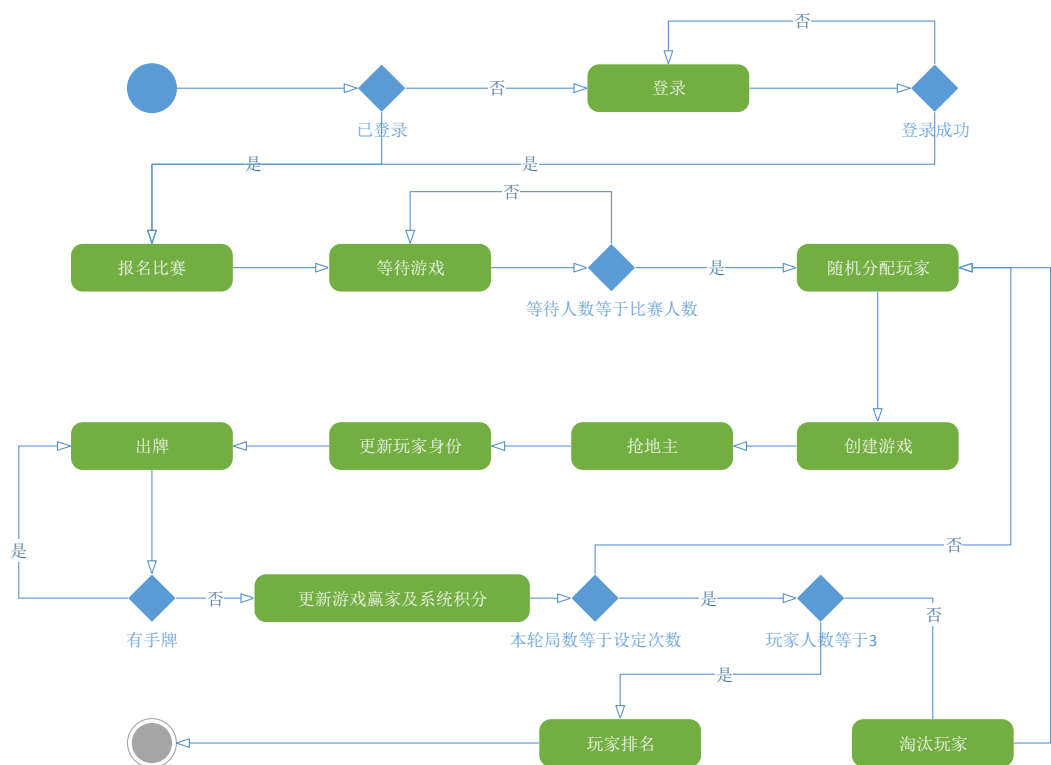


图 4.9 锦标赛模式活动图

综合以上对单机模式、在线模式、好友模式及锦标赛模式游戏中功能活动的分析,可知它们在斗地主游戏中的界面完全一致,只是在进入斗地主游戏之前的界面不同<sup>[44]</sup>。故此以在线模式为例进行界面设计的描述,该模式的界面包括游戏桌界面、游戏界面和玩家查询界面:

第一,游戏桌界面,该界面通过游戏桌的方式进行显示,每张游戏桌上显示三个进入游戏的按钮,玩家通过点击按钮打开游戏界面。

第二,游戏界面,该界面根据游戏的状态显示不同的信息,首先,玩家打开该界面时,显示开始游戏按钮,玩家点击按钮后,界面显示等待其他玩家加入中;其次,当发牌结束之后,界面显示玩家的手牌信息,抢地主过程中,显示抢地主的分数信息按钮,地主产生之后,界面更新玩家的手牌信息及地主牌信息;然后,显示出牌操作按钮,当玩家出牌之后,界面更新当前出牌信息和玩家手牌信息;最后,游戏结束时,根据玩家在本局游戏的输赢情况显示对应信息。

第三,玩家查询界面,该界面是超级管理员查询在线模式玩家数据的特定界面,在该界面使用列表的形式对玩家的用户名、游戏状态、游戏桌号、玩家的客户端 IP 等数据进行展示,并且界面的显示数据根据玩家的加入或退出、玩家游戏状态实时进行更新。

根据在线模式进行游戏的界面设计可得其操作流程图如图 4.10 所示：

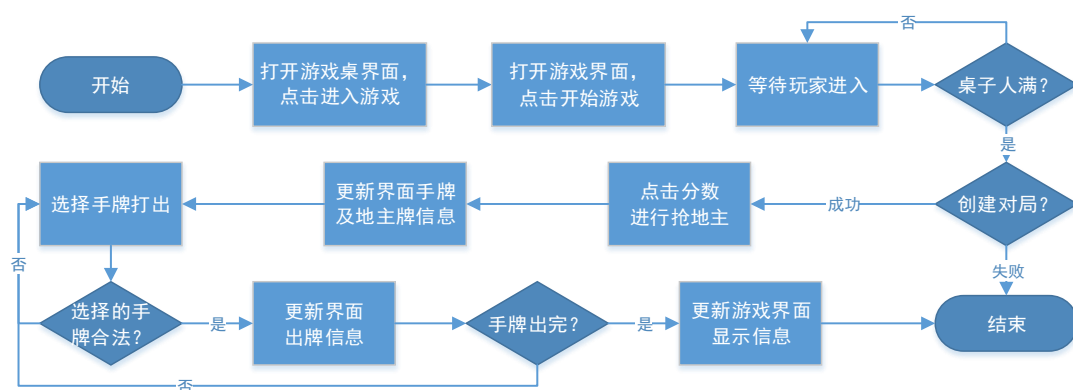


图 4.10 在线模式流程图

残局模式是用户对系统所提供的残局斗地主游戏进行挑战。该模式为玩家提供与以往斗地主游戏不同的游戏模式，使用户体验到不同玩法带来的游戏享受。残局模式的挑战难度分为简单、困难、专家三个等级，每个等级下设有不同数量的关卡，用户可根据自己的水平选择不同的难度等级进行挑战。若关卡挑战成功时，系统会根据挑战成功的出牌次数选择是否对数据存储记录进行更新，使数据库中始终存储挑战过程中出牌次数最少的数值；若关卡挑战失败时，系统自动恢复到初始的残局挑战界面。同种难度等级下，只有通过上一关卡才能进行下一关卡的挑战，不同难度等级之间不存在限制。残局模式的活动图如图 4.11 所示：

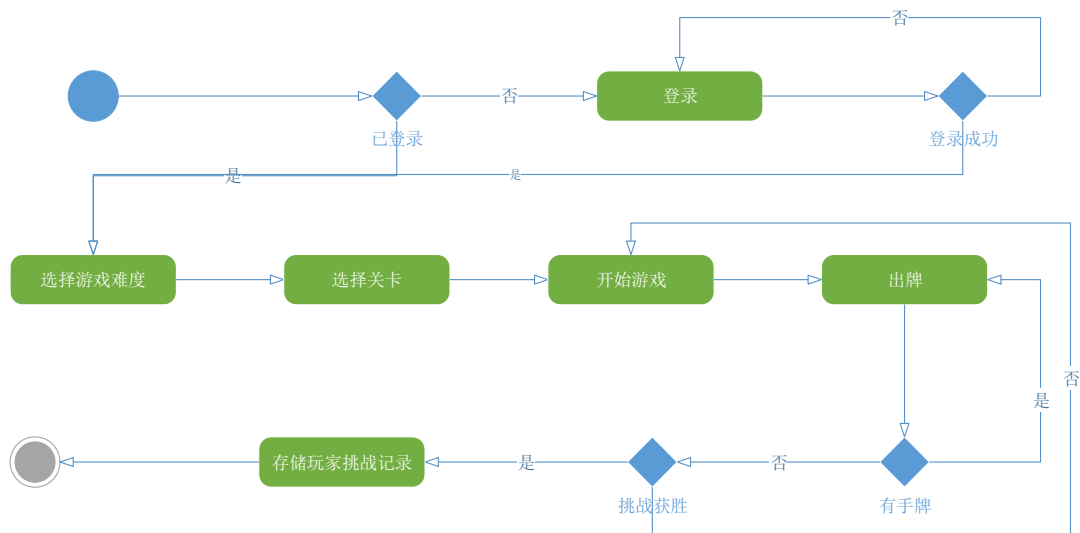


图 4.11 残局模式活动图

根据对残局模式活动的分析，可得其界面设计如下：第一，难度等级界面，该界面通过按钮的形式显示系统中残局数据对应的难度等级，玩家点击按钮进行难度等级的选择。第二，关卡选择界面，该界面对选择的难度等级下的游戏关卡进行显示，关卡分为可点击和不可点击两种状态，玩家选择可点击的关卡进行挑战。第三，游戏界面，该界面根据选择的关卡数据进行扑克牌显示，当轮到挑战玩家出牌时，显示出牌操作按钮，出牌之后，界面更新当前出牌信息和玩家的手牌信息，当挑战成功时，提示重新挑战和返回选择下一关卡进行挑战。

4.2.1.3 算法对抗

算法管理是为研究人员提供智能算法的管理功能，包括上传、修改、编辑等，上传的智能算法可以是 Python 语言编写的.py 文件、C 语言文件经过编译后的.so 文件、TensorFlow 框架生成的算法模型文件。对抗试验是为研究人员提供进行算法对抗试验的平台，其前提是研究人员已上传过智能斗地主算法，在算法对抗试验功能中，系统首先判断研究人员是否已上传过智能算法，当研究人员未上传过任何算法时，系统给出相应提示；然后研究人员选择需要进行对抗的算法，对抗算法的来源是系统提供的基础算法和其他研究人员上传并且共享的智能算法；最后设置对抗试验的局数、开始发牌位置、地主位置等试验参数，让选择的三个算法模拟三个玩家进行算法对抗试验，系统对试验数据进行收集。数据查询则是为研究人员提供试验数据的分页查询、导出、统计等功能。对抗试验功能的活动图如图 4.12 所示：

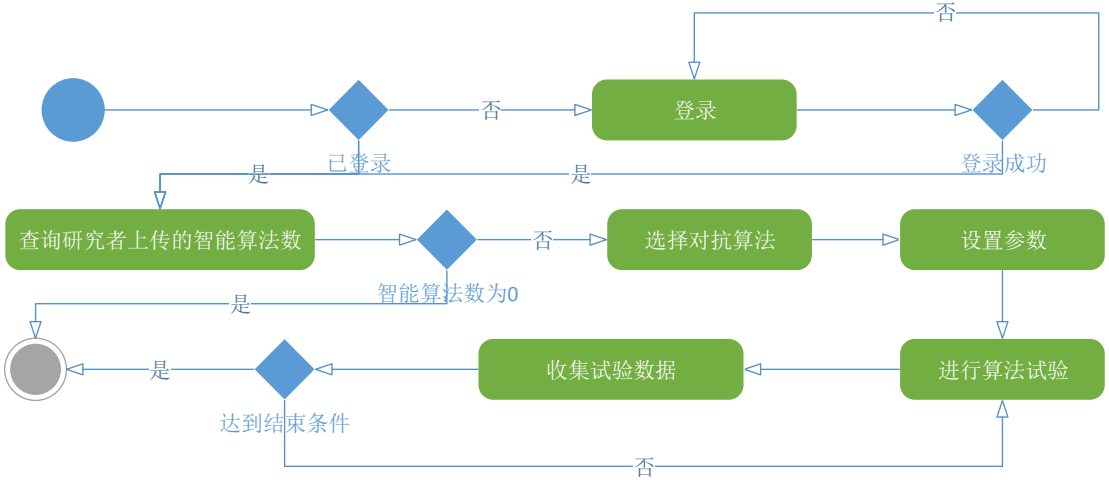


图 4.12 对抗试验活动图

4.2.1.4 战绩查询

单机战绩、在线战绩、好友战绩是提供系统用户对该模式斗地主游戏战绩数据的查询，查询结果采用列表方式对游戏玩家及其对应身份、游戏时间、赢家等数据进行展示，用户可对列表数据进行翻页查询，对某一对局进行详情和回放查询。详情查询以列表的形式展示该对局的所有出牌详情数据，使用不同颜色的文字对当前用户出牌数据进行标识；回放查询是采用真实画面对整个斗地主游戏出牌过程进行还原，使用游戏对局数据对游戏界面进行还原，该过程中系统自动循环出牌模拟整个斗地主游戏过程，使用详细出牌数据完成整个游戏的模拟出牌过程，用户可以对模拟出牌过程中进行操作，可进行暂停、继续出牌操作，在模拟出牌结束时可进行重新开始回放操作。

残局战绩是提供系统用户对挑战过的残局记录进行查询，通过列表的方式对挑战记录进行展示，展示的数据包括用户信息、难度等级、当前关卡、出牌轮数、挑战次数、是否过关等。用户可选择某一挑战记录进行挑战过程的出牌详情查询，若存在挑战多次的情况，通过选项卡方式对多次挑战记录进行展示。残局战绩查询活动图如图 4.13 所示：

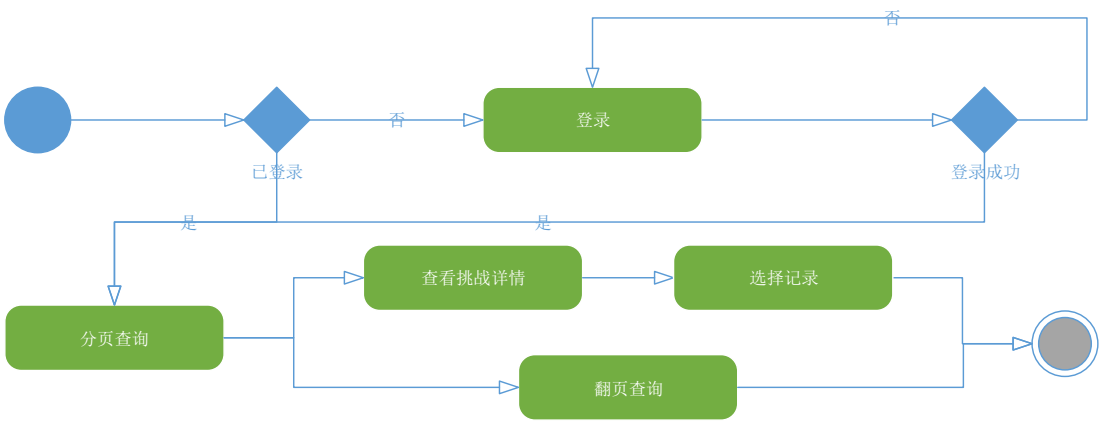


图 4.13 残局战绩查询活动图

锦标赛战绩是提供系统用户对参与过的锦标赛数据及锦标赛相对应的对局数据进行查询。首先查询用户参加过的锦标赛数据，通过列表进行锦标赛名称显示，然后用户选择某一锦标赛名称进行对应的游戏对局查询，对局数据的查询结果的展示形式、展示数据及相关操作与其他模式战绩相同。锦标赛战绩查询的活动图如图 4.14 所示：

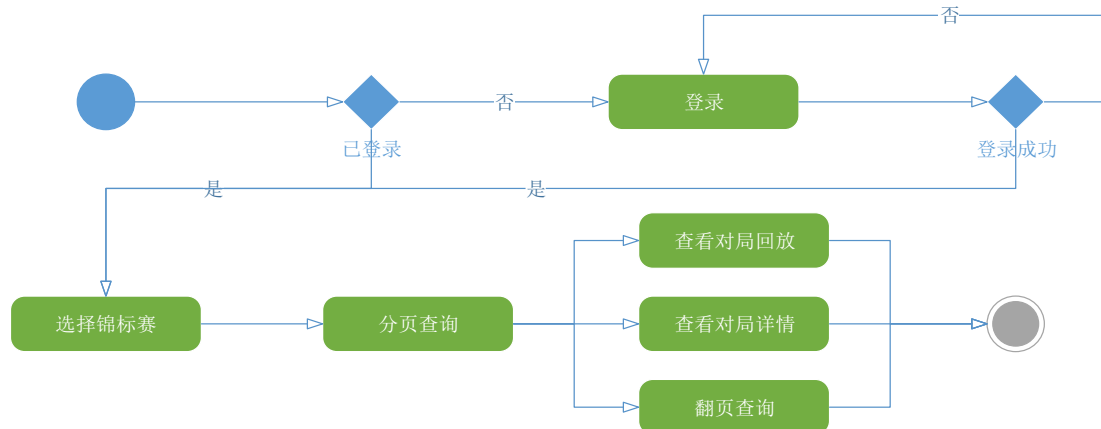


图 4.14 锦标赛战绩查询活动图

综合以上对单机战绩、在线战绩、好友战绩、锦标赛战绩、残局战绩查询功能的描述，其功能实现的界面都一致，只有锦标赛战绩查询与其他战绩查询相比多锦标赛选择操作。故此以锦标赛战绩查询为例，其界面设计如下所述：第一，战绩界面，该界面分为左右两个部分，左边显示锦标赛名称，选中的锦标赛名称使用其他颜色与为选中锦标赛名称进行区分；右边通过列表形式显示锦标赛对应的游戏对局数据，每局游戏数据提供对局回放和对局详情查看操作按钮。第二，对局详情界面，该界面显示对局详细出牌信息的列表。第三，对局回放界面，该界面根据游戏回放的状态进行内容显示，游戏开始时，界面显示三个玩家的手牌及地主牌信息；游戏过程中，界面显示暂停、继续、重新开始按钮。由上所述的锦标赛战绩界面设计，可得其界面操作流程如图 4.15 所示：

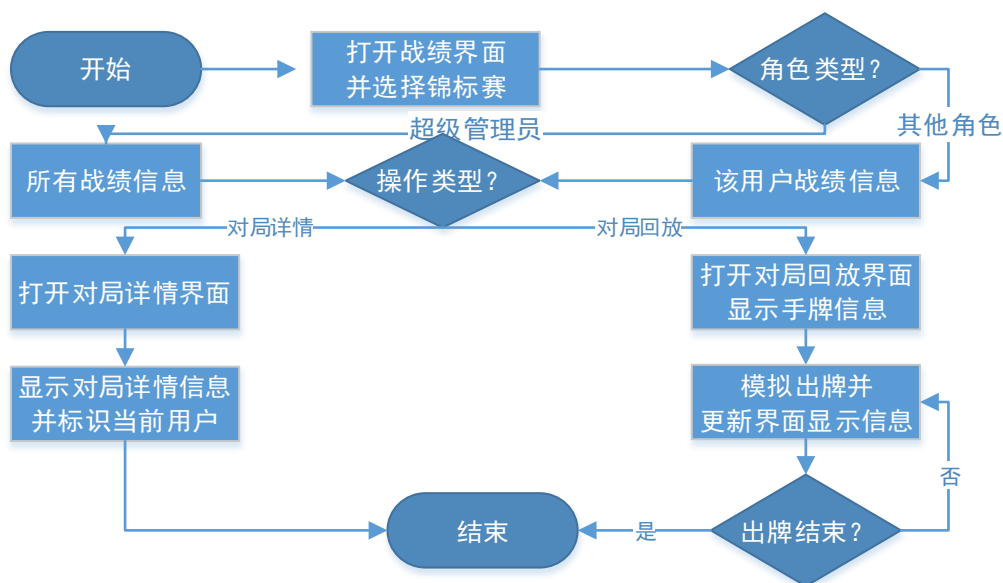


图 4.15 锦标赛战绩流程图



战绩排名是根据选择的 game 模式、排名方式两个查询条件对平台玩家进行排名统计，其中 game 模式包含所有模式、在线模式、锦标赛模式、单机模式、好友模式，排名方式包含综合排名、胜场排名、积分排名。玩家的统计条件为参与过该模式的斗地主游戏，统计的结果使用柱状图的方式对用户排名情况进行直观展示，同时对用户积分数据使用的图形化方式进行展示，可选择某一玩家查询对应的个人战绩统计，统计数据分 game 模式进行显示。

4.2.1.5 信息查询

登录历史是用户对以往登录记录的查询，查询的结果以列表形式进行展示，展示的数据包括登录地址、登录空间位置经纬度、定位方式、登录时间等，使用特殊的颜色对异常的登录记录进行标识，起到提示用户的作用。

积分查询默认是对所有模式的 game 对局积分数据进行查询，查询的结果以列表形式进行展示，展示的数据包括是否春天、炸弹个数、本局积分、总积分、game 模式等。用户可以通过查询对局积分了解目前在系统总的积分、每局获得积分情况，可同战绩查询的详情查询功能一样进行对局详情及对局回放的查询。玩家可以通过选择 game 模式的方式对某一特定的 game 模式进行 game 积分的查询，了解各个模式的积分输赢情况，可选择的 game 模式包括所有模式、在线模式、锦标赛模式、单机模式、好友模式。积分查询活动图如图 4.16 所示：

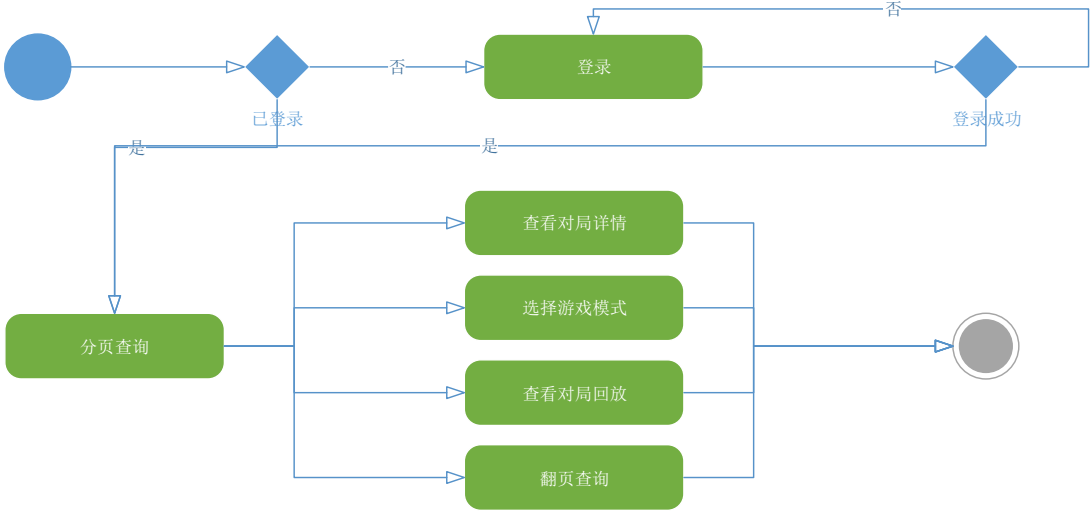


图 4.16 积分查询活动图

#### 4.2.1.6 个人中心

个人中心模块的功能包括密码修改、信息修改、头像修改、密保设置、密码重置。

密码修改是用户在成功登录系统之后对密码进行修改的功能，其修改过程的步骤为：第一步，使用当前登录密码对用户身份进行验证；第二步，设置新密码及确认密码，系统对两次设置的密码进行检验；第三步，系统进行密码更新，对当前登录用户进行注销，回到登录界面。

信息修改是用户在成功登录系统之后对个人数据进行修改的功能，在信息修改功能中对个人的数据进行完善，个人数据中必填数据包括用户名、性别，可选数据包括真实姓名、身份证号码、电话、QQ、邮箱等，必填数据不能为空，但可选数据用户可根据自己的意愿进行填写。个人数据保存成功之后，系统会根据修改后的结果对显示的数据进行更新。

头像修改是用户在成功登录系统之后对目前头像进行修改的功能，在本系统中头像修改提供两种方式：第一种，用户可以选择当前系统提供的图片，点击任意一张图片进行头像修改，点击图片后能进行选择头像的预览。第二种，用户可以上传本地的图片作为头像，图片成功上传后能进行头像的预览，并且上传的图片默认共享给其他用户作为头像修改时的可选图片。用户确认选择的图片后，系统进行图片的保存及用户数据的更新，对显示的用户头像进行更新。头像修改活动图如图 4.17 所示：

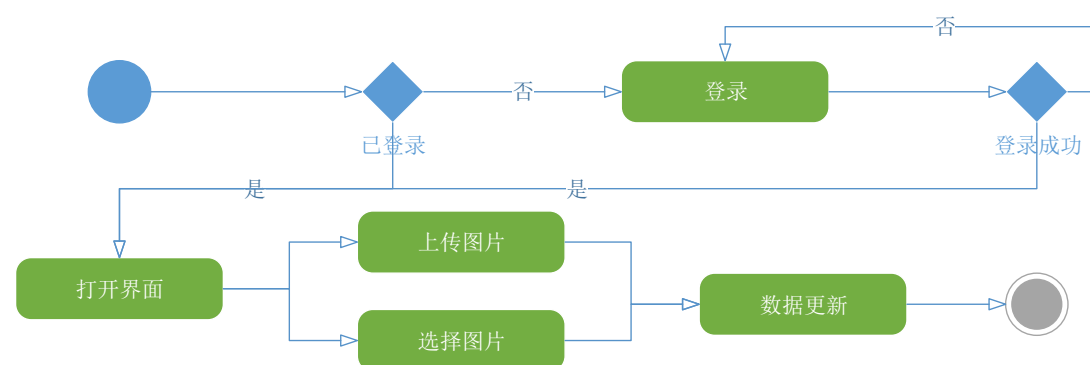


图 4.17 头像修改活动图

密保设置为用户忘记密码，对密码进行重置的前提条件。在密保设置界面对密保问题进行选择时，若界面未显示可选择的密保问题，则可能是尚未添加密保问题，可联系超级管理员进行密保问题的添加；也可能是数据获取失败，可尝试

进行刷新页面操作。密保设置分为四个步骤进行：第一步，使用当前登录用户的密码进行身份验证，验证失败则进行系统提示。第二步，选择三个不同的密保问题，并分别对相应密保问题进行答案设置，选择的三个问题不能出现重复，问题答案不能为空。第三步，对选择的三个密保问题再次输入答案，系统验证同样的密保问题两次输入的答案是否一致，不一致则进行系统提示。第四步，显示用户选择的密保问题和设置的答案，用户确认设置内容无误则更新用户密保设置数据。密保设置活动图如图 4.18 所示：

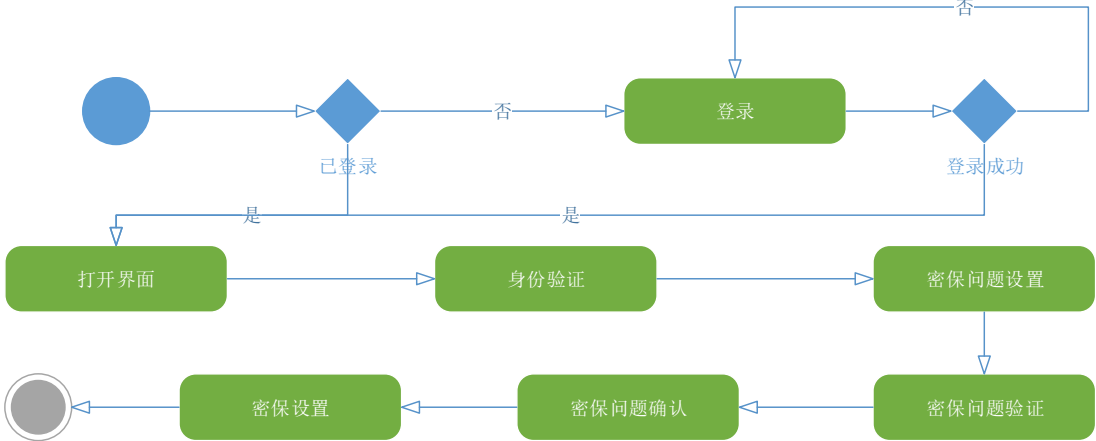


图 4.18 密保设置活动图

密码重置是用户忘记密码时，可对密码进行重置的操作。密码重置界面的操作步骤如下：首先使用用户名对身份进行验证，若未通过系统验证则系统给出提示；然后分别输入三个密保问题的答案来进行密保问题验证，若出现的密保问题不在设置的密保问题中时，则输入无代替答案，引入未设置的密保问题可提高对用户密码安全性的保护；最后设置新的密码完成密码重置，密码重置成功则系统则对设置的密保问题进行清除操作，返回到系统登录界面。密码重置活动图如图 4.19 所示：

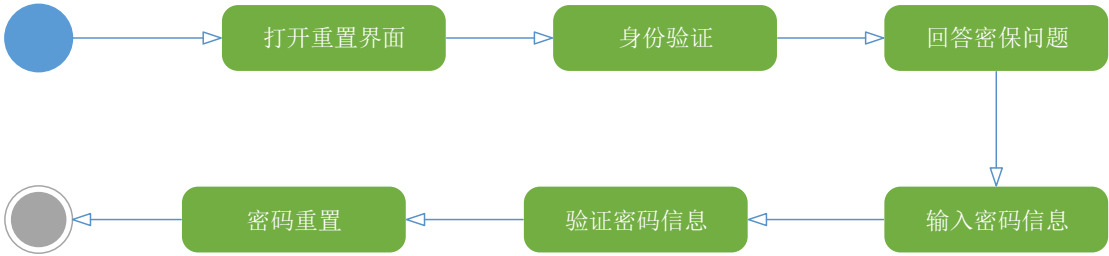


图 4.19 密码重置活动图

4. 2. 2 数据库设计

4. 2. 2. 1 数据库实体类设计

类图是用于描述系统中所包含的类以及它们之间的相互关系，是系统分析和设计阶段的重要产物，也是系统编码和测试的重要模型依据。类图通常包含实体类、控制类和边界类三种，实体类是对系统需求分析和设计阶段所需的数据实体及其关系进行表示，在本文中根据实体类数据的功能分为系统基础数据实体、游戏数据实体、算法试验实体三大类进行设计。

第一，系统基础数据实体类，根据系统的需求分析和设计，系统基础数据实体包括用户及其相关实体、角色及其相关实体、菜单实体，用户及其相关实体分别为用户实体、用户权限实体、密保问题实体、用户密保实体、登录历史实体，角色及其相关实体分别为角色实体、角色权限实体，对每个实体的实体属性和实体间的关系进行分析，可得各实体类属性（属性只列出主外键）及其之间关系如图 4.20 所示：

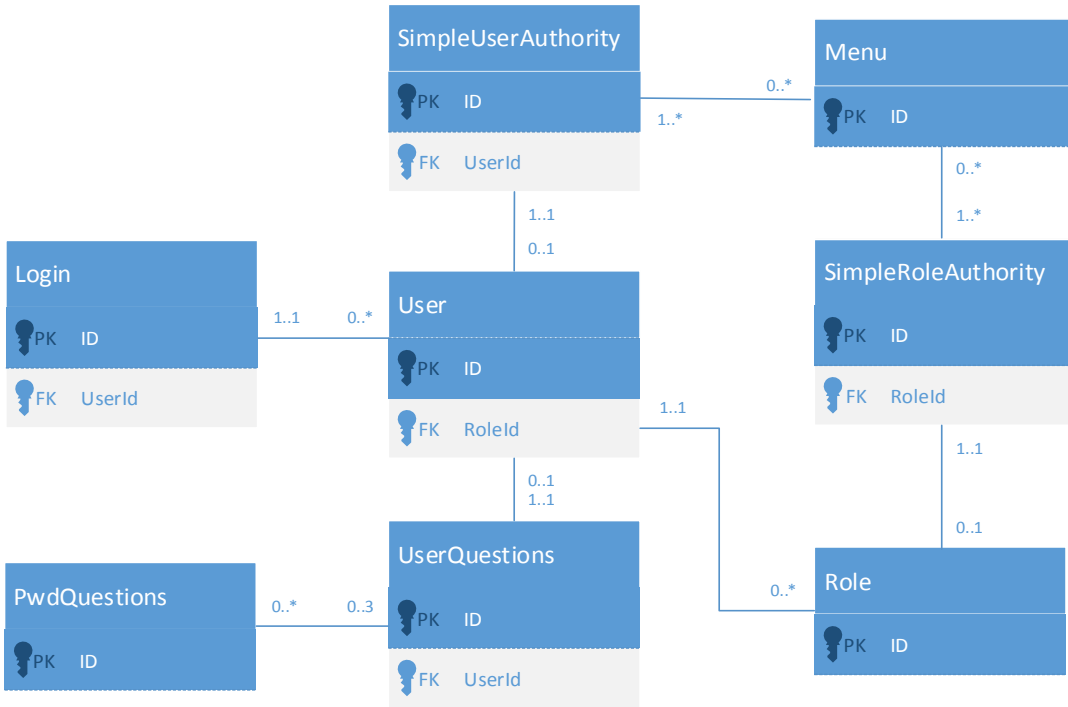


图 4.20 系统基础数据实体类图

第二，游戏数据实体类，根据系统的需求分析和设计，单机模式、在线模式、好友模式、锦标赛模式需要游戏数据实体、游戏出牌实体和游戏积分实体对游戏

过程中的数据进行管理和操作，锦标赛模式在此基础上还需锦标赛实体对锦标赛发布数据和锦标赛报名实体对锦标赛的报名数据进行管理和操作。残局模式需残局实体、残局挑战实体、挑战出牌实体对残局游戏中所需数据进行管理和操作，对各实体对应的游戏功能进行分析可得其实体属性（属性只列出主外键）和各实体类之间关系如图 4.21 所示：

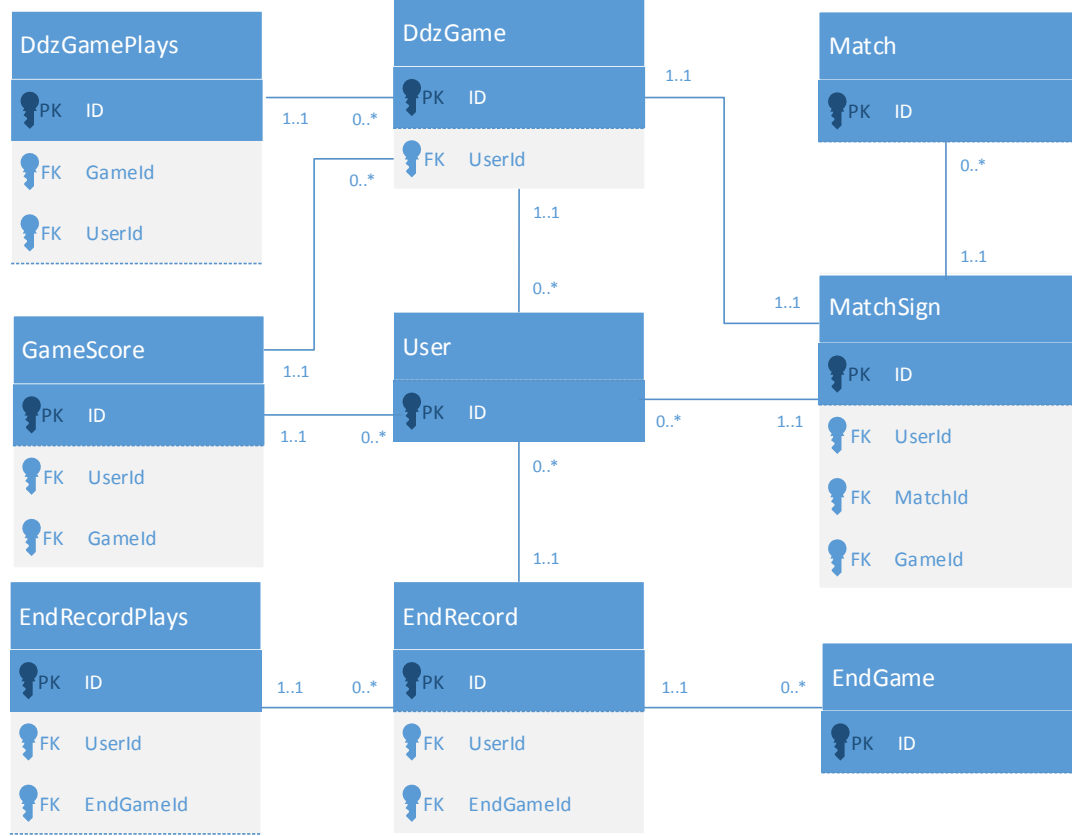


图 4.21 游戏数据实体类图

第三，算法试验实体类，根据系统的需求分析和设计，算法对抗试验中需使用算法数据实体、试验记录实体对用户上传的算法数据和试验记录进行存储，为算法试验过程中算法动态调用奠定数据基础；使用试验对局数据实体、试验对局出牌实体对试验过程中的每局试验数据和每次出牌数据进行存储，因为算法试验中的游戏过程与斗地主游戏过程一致，故采用与游戏数据实体中游戏数据实体、游戏出牌实体代替算法试验实体中的试验对局数据实体、试验对局出牌实体进行数据存储。对各实体对应的功能进行分析可得其实体属性（属性只列出主外键）和各实体类之间关系如图 4.21 所示：

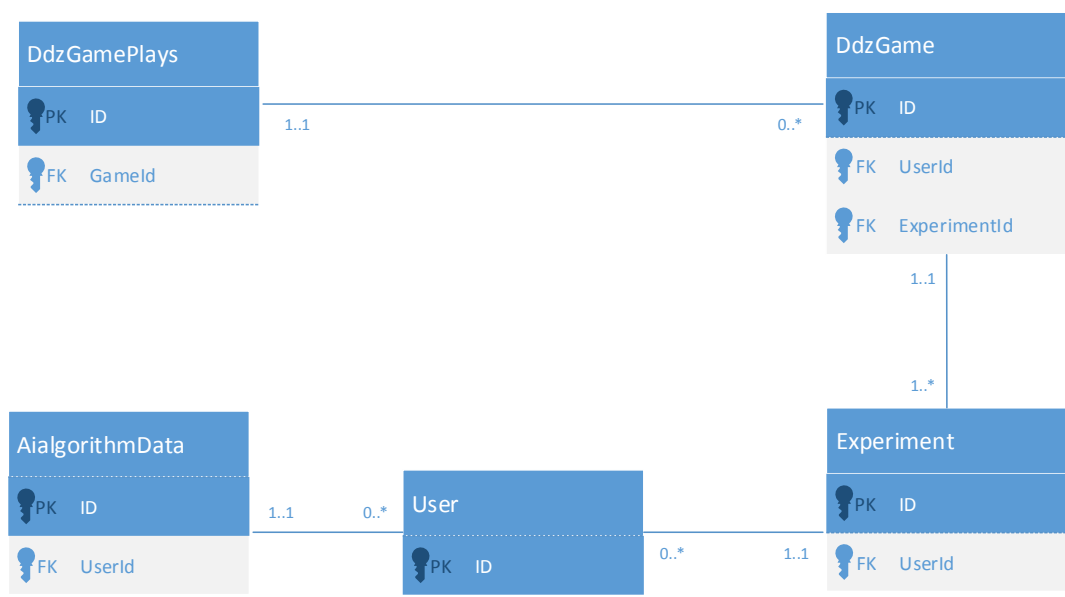


图 4.22 算法试验实体类图

#### 4.2.2.2 数据库物理设计

根据对数据库实体类的分析可知系统基础数据实体类对应的数据库表有 8 张，游戏数据实体类对应的数据库表有 8 张（除去 User），算法试验实体类对应的数据库表有 2 张（除去 User、DdzGame、DdzGamePlays），所以整个系统共需建立 18 张数据库表，下面分类型对实体属性在数据库中的存储类型进行设计并进行详细描述。

用户及其相关实体根据其实体类属性设计的数据表包括用户表、用户权限表、密保问题表、用户密保表、登录历史表。用户表是对用户的数据进行存储，必须具有的数据是用户名、密码、角色 ID、性别，可选数据有电话、QQ 号、邮箱、身份证号码、真实姓名。用户权限表是对用户的菜单权限进行存储，存储的数据有用户 ID、权限数据，一个用户最多能存储一条权限数据。密保问题表是对用户密保设置功能中的密保问题数据进行存储，存储的数据有密保问题描述。用户密保表示对用户设置的密保问题及答案数据进行存储，存储的数据有用户 ID、密保问题 ID、描述及答案，一个用户最多能存储一条密保设置数据<sup>[45]</sup>。登录历史表是对用户的登录历史数据进行存储，存储的数据有用户 ID、登录时间、登录地点、登录地点的经纬度、定位方式等。详细的表结构如下所示：

表 4.3 用户表(User)

| 字段          | 中文名称  | 类型           | 默认值 | 备注             |
|-------------|-------|--------------|-----|----------------|
| ID          | ID    | int          | 1   | 主键，自增长         |
| DisplayName | 用户名   | nvarchar(50) | 否   | 用户名具有唯一性       |
| UserPwd     | 密码    | nvarchar(50) | 否   | 加密后的密文         |
| IsDisable   | 是否禁用  | int          | 1   | 0:禁用;1:可用      |
| RoleId      | 所属角色  | int          | 否   |                |
| Sex         | 性别    | nvarchar(5)  | 否   | 性别取值：男、女、保密    |
| UserPic     | 用户头像  | nvarchar(50) | 否   | 用户头像对应的图片地址    |
| IsRegister  | 用户来源  | int          | 否   | 0:注册;1:管理员添加   |
| IsMenuIds   | 菜单来源  | int          | 0   | 0:继承角色;1:单独分配  |
| MenuIds     | 菜单 ID | nvarchar(50) | 否   | 单独分配的菜单 id 字符串 |
| Tel         | 电话    | nvarchar(15) | 否   |                |
| QQNumber    | QQ 号  | nvarchar(20) | 否   |                |
| Email       | 邮箱    | nvarchar(30) | 否   |                |
| IdCrad      | 身份证号码 | nvarchar(18) | 否   |                |
| RealName    | 真实姓名  | nvarchar(20) | 否   |                |

表 4.4 用户权限表(SimpleUserAuthority)

| 字段     | 中文名称  | 类型            | 默认值 | 备注              |
|--------|-------|---------------|-----|-----------------|
| ID     | ID    | int           | 1   | 主键，自增长          |
| UserId | 用户 id | int           | 0   | 外键              |
| MenuId | 权限数据  | nvarchar(200) | 否   | 菜单 id 必须存在于菜单表中 |

表 4.5 登录信息表(Login)

| 字段                | 中文名称  | 类型           | 默认值  | 备注     |
|-------------------|-------|--------------|------|--------|
| ID                | ID    | int          | 1    | 主键，自增长 |
| UserId            | 用户 id | int          | 0    | 外键     |
| Logintime         | 登录时间  | datetime     | 当前时间 |        |
| LoginAddress      | 登录地点  | nvarchar(50) | 否    |        |
| LoginProvince     | 登录省份  | nvarchar(50) | 否    |        |
| LoginCity         | 登录城市  | nvarchar(50) | 否    |        |
| LoginDistrict     | 登录区域  | nvarchar(50) | 否    |        |
| LoginStreet       | 登录街道  | nvarchar(50) | 否    |        |
| Logintownship     | 登录乡镇  | nvarchar(50) | 否    |        |
| LoginLat          | 定位纬度值 | nvarchar(50) | 否    |        |
| LoginLng          | 定位经度值 | nvarchar(50) | 否    |        |
| LoginLocationType | 定位方式  | nvarchar(50) | 否    |        |

表 4.6 密保问题表(PwdQuestions)

| 字段          | 中文名称 | 类型            | 默认值 | 备注     |
|-------------|------|---------------|-----|--------|
| ID          | ID   | int           | 1   | 主键，自增长 |
| DisplayName | 问题描述 | nvarchar(100) | 否   |        |

表 4.7 用户密保表(UserQuestions)

| 字段                | 中文名称     | 类型            | 默认值 | 备注     |
|-------------------|----------|---------------|-----|--------|
| ID                | ID       | int           | 1   | 主键，自增长 |
| UserId            | 用户 id    | int           | 0   | 外键     |
| QuesOnetionId     | 第一个问题 id | int           | 0   | 密保问题标识 |
| QuesOnetionText   | 第一个问题描述  | nvarchar(100) | 否   | 密保问题描述 |
| AnswerOne         | 第一个问题答案  | nvarchar(50)  | 否   | 密保问题答案 |
| QuesTwotionId     | 第二个问题 id | int           | 0   | 密保问题标识 |
| QuesTwotionText   | 第二个问题描述  | nvarchar(100) | 0   | 密保问题描述 |
| AnswerTwo         | 第二个问题答案  | nvarchar(50)  | 否   | 密保问题答案 |
| QuesThreetionId   | 第三个问题 id | int           | 0   | 密保问题标识 |
| QuesThreetionText | 第三个问题描述  | nvarchar(100) | 0   | 密保问题描述 |
| AnswerThree       | 第三个问题答案  | nvarchar(50)  | 否   | 密保问题答案 |

角色及其相关实体根据其实体类属性设计的数据表包括角色表和角色权限表，角色表对角色数据进行存储，存储的数据有角色名称、上级角色 ID，当角色为顶级角色时其上级角色 ID 为-1。角色权限表对角色的权限数据进行存储，存储的数据有角色 ID、权限数据，一个角色最多能存储一条权限数据。详细的表结构如下所示：

表 4.8 角色表(Role)

| 字段           | 中文名称    | 类型           | 默认值 | 备注         |
|--------------|---------|--------------|-----|------------|
| ID           | ID      | int          | 1   | 主键，自增长     |
| DisplayName  | 角色名称    | nvarchar(50) | 否   | 角色名称具有唯一性  |
| TreeParentId | 上级角色 id | int          | -1  | -1 表示为顶级菜单 |
| Creator      | 创建者     | int          | -1  | 创建用户的 ID   |

表 4.9 角色权限表(SimpleRoleAuthority)

| 字段      | 中文名称  | 类型            | 默认值 | 备注              |
|---------|-------|---------------|-----|-----------------|
| ID      | ID    | int           | 1   | 主键，自增长          |
| RoleId  | 角色 id | int           | 0   | 外键              |
| MenuId  | 权限数据  | nvarchar(200) | 否   | 菜单 id 必须存在于菜单表中 |
| Creator | 创建者   | int           | -1  | 创建用户的 ID        |

根据菜单实体类属性设计的表为菜单表，对菜单名称、菜单地址、菜单图标、上级菜单 ID、菜单排序、菜单深度等数据进行存储，当菜单为顶级菜单是上级菜单 ID 为-1。菜单表存储的数据是系统菜单的唯一来源，在系统初始化时菜单表中必须存在菜单管理所对应的数据，理论上来说菜单管理所对应的记录是在系统管理中是不允许删除的，需要删除时必须联系数据库管理员进行操作。详细的表结构如下所示：



表 4.10 菜单表(Menu)

| 字段           | 中文名称    | 类型            | 默认值 | 备注         |
|--------------|---------|---------------|-----|------------|
| ID           | ID      | int           | 1   | 主键, 自增长    |
| DisplayName  | 菜单名称    | nvarchar(20)  | 否   | 菜单名称具有唯一性  |
| MenuUrl      | 菜单地址    | nvarchar(100) | 否   |            |
| MenuIcon     | 菜单图标    | nvarchar(30)  | 否   |            |
| TreeParentId | 上级菜单 id | int           | -1  | -1 是表示顶级菜单 |
| DisplayOrder | 菜单排序    | int           | 1   | 菜单在界面显示的排序 |
| TreeDepth    | 菜单深度    | int           | 1   |            |

游戏实体类中的游戏数据实体、游戏出牌实体、游戏积分实体根据其属性设计的数据表为游戏数据表、游戏出牌数据表、游戏积分表, 游戏数据表是对创建斗地主游戏的数据进行存储, 存储的数据有三个玩家的用户 ID、身份、手牌、手牌 ID 以及游戏的赢家 ID、地主牌、地主牌 ID、桌号、游戏模式、地主分数、时间, 其中桌号只有在线模式中使用。游戏出牌数据表是对游戏过程中的每次出牌数据进行存储, 存储的数据有游戏 ID、用户 ID、当前出牌、出牌类型、三个玩家当前的手牌, 当前的手牌包含玩家未出手牌、玩家已出手牌、玩家本轮出牌、玩家本轮出牌前手牌、出牌类型。详细的表结构如下所示:

表 4.11 游戏出牌数据表(DdzGamePlays)

| 字段            | 中文名称       | 类型            | 默认值 | 备注               |
|---------------|------------|---------------|-----|------------------|
| ID            | ID         | int           | 1   | 主键, 自增长          |
| GameId        | 比赛 id      | int           | 0   | 外键               |
| UserId        | 玩家 id      | int           | 0   | 外键               |
| IsFirst       | 是否第一手出牌    | int           | 0   | 1:第一手出牌;0:非第一手出牌 |
| PlayCards     | 当前出牌       | nvarchar(100) | 否   |                  |
| SurplusCards  | 当前玩家未出手牌   | nvarchar(100) | 否   |                  |
| BCards        | 当前玩家出牌前的手牌 | nvarchar(100) | 否   |                  |
| PutCards      | 当前玩家已出手牌   | nvarchar(100) | 否   |                  |
| CardsType     | 出牌类型       | nvarchar(100) | 否   |                  |
| UpCards       | 上家未出手牌     | nvarchar(100) | 否   |                  |
| UpPlayCards   | 上家本轮出牌     | nvarchar(100) | 否   |                  |
| UpBCards      | 上家出牌前的手牌   | nvarchar(100) | 否   |                  |
| UpPutCards    | 上家已出手牌     | nvarchar(100) | 否   |                  |
| UpCardsType   | 上家出牌类型     | nvarchar(100) | 否   |                  |
| DownCards     | 下家未出手牌     | nvarchar(100) | 否   |                  |
| DownPlayCards | 下家本轮出牌     | nvarchar(100) | 否   |                  |
| DownBCards    | 下家出牌前的手牌   | nvarchar(100) | 否   |                  |
| DownPutCards  | 下家已出手牌     | nvarchar(100) | 否   |                  |
| DownCardsType | 下家出牌类型     | nvarchar(100) | 否   |                  |

表 4.12 游戏数据表(DdzGame)

| 字段               | 中文名称      | 类型            | 默认值  | 备注                        |
|------------------|-----------|---------------|------|---------------------------|
| ID               | ID        | int           | 1    | 主键, 自增长                   |
| MatchTime        | 游戏时间      | datetime      | 当前时间 |                           |
| MatchPlayerOne   | 一号玩家      | int           | 0    | 用户 ID                     |
| pOneIdentity     | 一号玩家身份    | int           | 0    | 1:地主;0:农民                 |
| pOneCards        | 一号玩家手牌    | nvarchar(100) | 否    |                           |
| pOneCardsids     | 一号玩家手牌 id | nvarchar(100) | 否    |                           |
| MatchPlayerTwo   | 二号玩家      | int           | 0    | 用户 ID                     |
| pTwoIdentity     | 二号玩家身份    | int           | 0    | 1:地主;0:农民                 |
| pTwoCards        | 二号玩家手牌    | nvarchar(100) | 否    |                           |
| pTwoCardsids     | 二号玩家手牌 id | nvarchar(100) | 否    |                           |
| MatchPlayerThree | 三号玩家      | int           | 0    | 用户 ID                     |
| pThreeIdentity   | 三号玩家身份    | int           | 0    | 1:地主;0:农民                 |
| pThreeCards      | 三号玩家手牌    | nvarchar(100) | 否    |                           |
| pThreeCardsids   | 三号玩家手牌 id | nvarchar(100) | 否    |                           |
| LandlordCards    | 地主牌       | nvarchar(10)  | 否    |                           |
| LandlordCardsids | 地主牌 id    | nvarchar(10)  | 否    |                           |
| MatchWinner      | 比赛赢家      | int           | 1    | 用户 ID                     |
| IsLandlord       | 是否地主      | int           | 0    | 1:地主;0:农民                 |
| TableNumber      | 桌号        | int           | 0    |                           |
| MatchType        | 游戏模式      | int           | 1    | 0:试验;1:在线;2:锦标赛;3:单机;4:好友 |
| MatchScore       | 地主分数      | int           | 0    |                           |

表 4.13 游戏积分表(GameScore)

| 字段         | 中文名称   | 类型  | 默认值 | 备注                    |
|------------|--------|-----|-----|-----------------------|
| ID         | ID     | int | 1   | 主键, 自增长               |
| UserId     | 用户 id  | int | 0   | 外键                    |
| GameId     | 对局 id  | int | 0   | 外键                    |
| IsSpring   | 是否春天   | int | 0   | 0:其他;1:春天;2:反春        |
| BombNumber | 炸弹个数   | int | 0   |                       |
| GameType   | 游戏模式   | int | 0   | 1:在线模式; 3:单机模式;4:好友模式 |
| UserScore  | 当局获得积分 | int | 0   |                       |
| AllScore   | 总共获得积分 | int | 0   |                       |

对于锦标赛来说, 还需根据锦标赛实体和锦标赛报名实体的属性设计锦标赛数据表和锦标赛报名表, 锦标赛数据表存储的数据包括锦标赛名称、锦标赛状态、每轮比赛局数、比赛人数, 每轮比赛局数的设置是锦标赛比赛过程中进行玩家淘汰进入下一轮的依据。锦标赛报名表对玩家进行锦标赛报名和比赛过程中的数据进行存储, 存储的数据有用户 ID、锦标赛 ID、游戏 ID、锦标赛积分、比赛轮数、

比赛局数、玩家状态，锦标赛比赛过程每局游戏都对应一条数据记录。详细的表结构如下所示：

表 4.14 锦标赛数据表(Match)

| 字段             | 中文名称   | 类型  | 默认值 | 备注                                      |
|----------------|--------|-----|-----|---|
| ID             | ID     | int | 1   | 主键，自增长                                  |
| MatchName      | 锦标赛名称  | int | 0   | 锦标赛名称具有唯一性                              |
| MatchStatus    | 锦标赛状态  | int | 1   | 1:比赛开启;0:比赛关闭;2:比赛正在进行                  |
| MatchNumIndex  | 每轮比赛局数 | int | 1   |   |
| MatchPlayerNum | 比赛人数   | int | 3   | 参加比赛的人数限制,最少为 3 人,目前最多为 30,且人数必须为 3 的倍数 |

表 4.15 锦标赛报名表(MatchSign)

| 字段            | 中文名称   | 类型  | 默认值 | 备注                   |
|---------------|--------|-----|-----|----------------------|
| ID            | ID     | int | 1   | 主键，自增长               |
| UserId        | 用户 id  | int | 0   | 外键                   |
| MatchId       | 锦标赛 id | int |     | 外键                   |
| GameId        | 游戏 id  | int | 0   | 外键                   |
| MatchScore    | 锦标赛积分  | int | 0   |                      |
| MatchNum      | 比赛轮数   | int | 1   | 比赛轮数                 |
| MatchNumIndex | 比赛局数   | int | 1   | 该轮的局数                |
| CurStatus     | 玩家状态   | int | 1   | 0:被淘汰;1:报名成功; 2:正在比赛 |

根据残局数据实体、残局挑战实体、挑战出牌实体的属性设计的数据表为残局数据表、残局挑战表、挑战出牌表，残局数据表存储的数据游戏残局等级、玩家的牌、挑战的牌、残局关数。残局挑战表是对用户的残局挑战记录进行存储，存储的数据有用户 ID、残局数据 ID、游戏标识、出牌轮数、挑战成功与否、挑战次数，出牌轮数存储的数据为挑战成功时的最小出牌次数。挑战出牌表对每次挑战的出牌数据进行存储，存储的数据有用户 ID、残局游戏 ID、游戏标识、是否出牌、当前出牌，当前出牌存储的是扑克牌在游戏过程中对应的数字，游戏标识设置是用于区分同一用户进行同一关卡多次挑战时的出牌数据。详细的表结构如下所示：

表 4.16 残局数据表(EndGame)

| 字段         | 中文名称 | 类型           | 默认值 | 备注     |
|------------|------|--------------|-----|--------|
| ID         | ID   | int          | 1   | 主键，自增长 |
| GameLevel  | 残局等级 | nvarchar(50) | 否   |        |
| OwnCards   | 玩家手牌 | nvarchar(50) | 否   |        |
| OtherCards | 挑战手牌 | nvarchar(50) | 否   |        |
| PassNum    | 残局关数 | int          | 1   |        |

表 4.17 残局挑战表(EndRecord)

| 字段         | 中文名称    | 类型           | 默认值 | 备注         |
|------------|---------|--------------|-----|------------|
| ID         | ID      | int          | 1   | 主键, 自增长    |
| UserId     | 用户 id   | int          | 0   | 外键         |
| EndGameId  | 残局数据 id | int          | 0   | 外键         |
| GameFlag   | 游戏标识    | nvarchar(50) | 否   | 本局游戏时间戳标识  |
| UserPlays  | 出牌轮数    | int          | 0   | 成功时出牌次数最小值 |
| UserWin    | 挑战成功与否  | int          | 0   | 1:成功;0:失败  |
| UserNumber | 挑战次数    | int          | 0   |            |

表 4.18 挑战出牌表(EndRecordPlays)

| 字段        | 中文名称    | 类型            | 默认值 | 备注        |
|-----------|---------|---------------|-----|-----------|
| ID        | ID      | int           | 1   | 主键, 自增长   |
| UserId    | 用户 id   | int           | 0   | 外键        |
| EndGameId | 残局数据 id | int           | 0   | 外键        |
| GameFlag  | 游戏标识    | nvarchar(50)  | 否   | 本局游戏时间戳标识 |
| PlayFirst | 是否出牌    | int           | 0   | 1:是;0:否   |
| PlayCards | 当前出牌    | nvarchar(100) | 否   |           |

根据算法数据实体和试验记录实体属性设计的表为算法数据表和试验记录表, 算法数据表存储的数据包括算法名称、算法文件名称、算法函数名称、用户 ID、是否公开、是否基础算法等, 系统基础算法在系统管理中不允许进行删除、编辑操作, 需对其进行操作时必须联系数据库管理员进行操作; 试验记录表存储的数据包括试验名称、试验局数、开始发牌方式、开始发牌位置、地主玩家确定方式、地主玩家位置等, 当开始发牌方式为指定时, 开始发牌位置存储的值为有效值, 当地主玩家确定方式为指定时, 地主玩家位置存储的值为有效值。详细的表结构如下所示:

表 4.19 算法数据表(AialgorithmData)

| 字段              | 中文名称   | 类型           | 默认值 | 备注                 |
|-----------------|--------|--------------|-----|--------------------|
| ID              | ID     | int          | 1   | 主键, 自增长            |
| UserId          | 用户 id  | int          | 0   | 外键                 |
| AialgorithmName | 算法名称   | nvarchar(50) | 否   |                    |
| FileName        | 文件名称   | nvarchar(50) | 否   | 算法文件的对应名称          |
| FunctionName    | 函数名称   | nvarchar(50) | 0   | 调用的函数名称            |
| FileType        | 算法文件类型 | nvarchar(10) | py  |                    |
| IsShare         | 是否公开   | int          | 1   | 1:公开;0:不公开         |
| IsBase          | 是否基础算法 | int          | 0   | 0:系统基础算法;1:非系统基础算法 |

表 4.20 试验记录表(AialgorithmData)

| 字段               | 中文名称     | 类型           | 默认值 | 备注               |
|------------------|----------|--------------|-----|------------------|
| ID               | ID       | int          | 1   | 主键，自增长           |
| UserId           | 用户 id    | int          | 0   | 外键               |
| ExperimentName   | 试验名称     | nvarchar(50) | 否   |                  |
| ExperimentNumber | 试验局数     | int          | 1   | 试验局数取值范围为 1-1000 |
| StartCardType    | 发牌方式     | int          | 0   | 0:随机;1:指定        |
| StartCardPos     | 开始发牌位置   | int          | 1   | 取值为 1、2、3        |
| LandlordPosType  | 地主确定方式   | int          | 0   | 0:随机;1:指定        |
| LandlordPos      | 地主玩家位置   | int          | 1   | 取值为 1、2、3        |
| OneName          | 一号位置算法名称 | nvarchar(50) | 否   |                  |
| TwoName          | 二号位置算法名称 | nvarchar(50) | 否   |                  |
| ThreeName        | 三号位置算法名称 | nvarchar(50) | 否   |                  |

## 4.3 智能算法设计

智能算法在本文中是指使用设计的算法代替玩家完成扑克牌选择。本文设计了贪婪选择和权重选择两个智能出牌算法，对 MCTS 算法进行研究，对其应用方面进行设计。

### 4.3.1 贪婪选择出牌算法设计

贪婪算法总是能做出当前情形下来看最优的选择，不从整体最优上加以考虑，其得出的是局部最优解的。但是在问题面临的选择十分广泛时，这种求解方式往往是最有效的，斗地主游戏中的出牌选择就是这样一个问题<sup>[18]</sup>。贪婪选择出牌算法就是在扑克牌拆分和选择时采用贪婪算法的思想进行设计，选择出当前情形下最优的出牌选择，确保每轮出牌尽可能对自己当前的局势有利，保证玩家的手牌在每轮出牌中都尽可能的减少，增加在相同出牌次数下游戏获胜的概率。出于对斗地主游戏中两个农民是合作关系的考虑，在算法中通过设置一定的条件来考虑两个农民之间的合作问题<sup>[46][47][48]</sup>，两个农民之间不接牌的条件设置如下：己方未能打完手牌、对方出牌超过 Q 以上、对方手牌小于等于 5 张、对方出牌类型为炸弹。贪婪选择出牌算法通过伪代码的形式描述如图 4.23 所示：

输入：1、本轮第一手出牌 first  
2、当前玩家的手牌 curPokers  
3、当前玩家的手牌身份 curIdCrad  
4、上一出牌玩家的手牌张数 upPokerNum  
5、上一出牌玩家的身份 upIdCrad  
6、上一玩家的出牌 upPlayPokers

输出：选择的出牌 choosePokers

过程：1) 初始化 i=0, types=[11122,1112,111,11,1];  
2) IF first == 1 {  
3) for(i=0;i<5;i++) {  
4) 根据 curPokers 和 types[i]获取手牌拆分结果 splitsPokers  
5) IF splitsPokers == null {  
6) choosePokers = splitsPokers  
7) 跳出 for 循环  
8) }  
9) }  
10) } else {  
11) 根据 upPlayPokers 获取出牌类型 pokerType  
12) IF curIdCrad == 0 and upIdCrad == 0 {  
13) 根据 curPokers、upPlayPokers、upPokerNum 和 pokerType 获取农民之间接牌的判断结果 isPlay  
14) IF isPlay == TRUE {  
15) 根据 curPokers 和 pokerType 获取手牌拆分结果 splitsPokers  
16) IF splitsPokers == null {  
17) choosePokers = null  
18) } else {  
19) choosePokers = splitsPokers  
20) }  
21) } else {  
22) choosePokers = null  
23) }  
24) } else {  
25) 根据 curPokers 和 pokerType 获取手牌拆分结果 splitsPokers  
26) IF splitsPokers == null {  
27) choosePokers = null  
28) } else {  
29) choosePokers = splitsPokers  
30) }  
31) }  
32) Return choosePokers

图 4.23 贪婪选择出牌算法伪代码

### 4.3.2 权重选择出牌算法设计

权重选择出牌算法是在扑克牌拆分过程中对每次选择的合法出牌进行权重值设置，最终选择权重值最小的出牌。权重值设置采用合法出牌对应的出牌类型与合法出牌中关键牌牌面值对应的权重值相加的方式，例如三带出牌类型的关键牌为三个一样的牌。权重选择出牌算法通过伪代码的形式描述如图 4.24 所示：

```
输入：1、当前玩家的手牌 curPokers
      2、上一出牌玩家的出牌 upPlayPokers
输出：选择的出牌 choosePokers
过程：1) 初始化 Types 为出牌类型及其权重值的集合，Pokers 为一副完整扑克牌使用
      数字表示的集合；
      2) 根据 Types 和 Pokers 获取所有的出牌组合 allHands
      3) IF allHands== null {
      4)     choosePokers = null
      5)     Return choosePokers
      6) }
      7) 根据 allHands 和 curPokers 获取本轮合法的出牌组合 legalHands
      8) IF legalHands == null {
      9)     choosePokers = null
      10)    Return choosePokers
      11) }
      12) 根据 legalHands 和 curPokers 获取本轮的合法出牌并计算权重值得到待选出
      牌集合 playSet
      13) IF playSet == null {
      14)     choosePokers = null
      15)     Return choosePokers
      16) }
      17) IF upPlayPokers== null {
      18)     根据 playSet 获取待选出牌集合中权重值最小的合法出牌 minPlayPoker
      19)     choosePokers = minPlayPoker
      20) } else {
      21)     根据 upPlayPokers 和 playSe 获取待选出牌集合中权重值最小的合法出牌
      minPlayPoker
      22)     IF minPlayPokert == null {
      23)         choosePokers = null
      24)     } else {
      25)         choosePokers = minPlayPoker
      26)     }
      27) }
      28) Return choosePokers
```

图 4.24 权重选择出牌算法伪代码

扑克牌牌面值与权重值的对应关系如表 4.21 所示：

表 4.21 牌面值与权重值对应表

|     |     |     |     |     |     |     |     |     |     |   |   |   |   |    |    |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|---|---|---|---|----|----|
| 牌面值 | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | J   | Q | K | A | 2 | 小王 | 大王 |
| 权重值 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 | 1 | 1 | 2 | 2  | 2  |

出牌类型对应的权重值如表 4.22 所示：

表 4.22 出牌类型权重值对应表

|      |   |    |     |      |       |       |      |        |    |
|------|---|----|-----|------|-------|-------|------|--------|----|
| 出牌类型 | 1 | 11 | 111 | 1112 | 11122 | 12345 | 1111 | 111123 | 12 |
| 权重值  | 4 | 2  | 1   | 1    | 1     | 3     | 6    | 5      | 7  |

### 4.3.3 MCTS 算法应用设计

MCTS 算法分为两个模块：较小拆分和 MCTS 搜索模块<sup>[19]</sup>。首先使用较小拆分算法对博弈树根节点进行剪枝，然后对剪枝后的博弈树进行蒙特卡洛抽样，进而提高求解效率。较小拆分算法的依据是对玩家的手牌进行拆分后，选择出牌组合比较少的拆分结果，出牌组合数小则表明该玩家在相同的出牌次数对局中获胜的几率较大<sup>[19]</sup>。MCTS 模块根据斗地主游戏的手牌隐蔽特性和发牌的随机性<sup>[52]</sup>，将蒙特卡洛树搜索选择过程分解为五个过程：手牌较小拆分过程、博弈树节点选择过程、博弈树节点抽样扩展过程、模拟抽样博弈环境游戏过程、博弈树节点反馈过程<sup>[53][54]</sup>。

MCTS 算法模型过程的描述如下：当游戏开始的时候，首先使用较小拆分模块求解出待抽样的出牌集，然后通过蒙特卡洛树搜索模块，对待模拟抽样的所有出牌进行游戏模拟，根据其模拟结果的收益，选择平均收益最大的出牌作为本次的最佳出牌<sup>[19][20][55]</sup>。

本文根据对 MCTS 算法的研究，结合其在本系统中的应用需求，对其在系统中具体应用的参数设计如下：七个输入参数分别是当前玩家下标（0 表示地主，1 表示地主下家，2 表示地主上家）、当前出牌玩家的手牌、0 号玩家已出牌、1 号玩家已出牌、2 号玩家已出牌、本轮下家出牌、本轮上家出牌；一个输出参数为当前选择的出牌，当输出参数值为 0 时，表示未选择出牌。



## 4.4 本章小结

本章是对系统设计阶段进行阐述。首先对系统总体设计进行阐述，确定系统采用 B/S 的体系结构进行设计，系统的架构可分为表现层、控制层、业务逻辑层、数据模型层，并对系统架构的每一层的详情进行描述；然后对系统详细设计的工作进行描述，对系统的功能模块中的功能采用活动图、流程图等方式进行说明，进一步完成对数据库的设计，给出相应的实体类图以及详细的数据表结构；最后本文的智能斗地主算法设计进行详情的描述，给出算法的具体伪代码。

## 第五章 系统实现

### 5.1 关键功能实现

#### 5.1.1 贪婪选择出牌算法实现

根据贪婪选择出牌算法的设计以及对其运用环境的分析,本文采用 JavaScript 语言进行算法的实现,算法实现具体的函数描述如下:

出牌类型判断函数 `getPokerType`,其输入参数为上一出牌玩家的出牌或选择的出牌,通过对输入的出牌特征进行提取,将出牌特征与出牌类型进行匹配得到其对应的出牌类型,匹配失败时则使用 `null` 值作为出牌类型。

判断两个农民之间是否接牌的函数 `getContinuePlay`,其输入参数为当前农民手牌、另一农民本轮的出牌、另一农民当前的手牌张数、另一农民本轮的出牌类型。首先判断另一农民当前的手牌张数是否小于等于 5,如果是返回 `false` 表示不接牌;其次判断另一农民本轮的出牌中主牌面值是否超过 Q,如果是返回 `false` 表示不接牌;然后判断另一农民本轮的出牌类型是否为炸弹或王炸,如果是返回 `false` 表示不接牌;最后判断当前农民手牌是否可以打完,如果是返回 `true` 表示不接牌,否则返回 `false` 表示不接牌。

根据出牌类型进行手牌拆分的函数 `getSplitPoker`,输入的参数为当前玩家的手牌、出牌类型。首先对当前玩家的手牌中各牌面值的数量进行统计得牌面值与数量关系的对象;然后根据统计结果与出牌类型进行比较,判断是否存在同样的出牌类型或大于的出牌类型,如果不存在则返回 `null` 表示拆分结果为空;最后根据出牌类型进行手牌拆分,使用 `getComparePoker` 函数比较拆分的结果是否大于当前的出牌,如果是则返回当前的拆分结果作为选择的出牌,否则返回 `null` 表示拆分结果为空。

比较两组扑克牌大小的函数 `getComparePoker`,输入参数为两组扑克牌,默认是输入的前一组扑克牌小于后一组扑克牌,根据斗地主游戏设计的出牌大小关系比较规则进行大小关系的确定,输入参数值为 1 表示后一组大于前一组,0 表示后一组等于前一组,-1 表示后一组小于前一组,非本轮第一手出牌时,在 `getSplitPoker` 函数中调用该函数对选择的出牌和上一出牌玩家的出牌。

贪婪选择出牌算法的输入参数为本轮第一手出牌 `first`，取值为 1 表示本轮第一手出牌，0 表示非本轮第一手出牌；当前玩家的手牌 `curCards` 和身份 `curId`，`curCards` 为手牌对应的数字表示数组，`curId` 取值为 1 表示地主，0 表示农民；上一出牌玩家的手牌张数 `upNum`、身份 `upId`、出牌 `cards`，`upId` 取值为 1 表示地主，0 表示农民，`cards` 为出牌对应的数字表示数组；输出为选择的出牌 `playCards`，`playCards` 为选择出牌的数字表示数组，取值为 `null` 表示未选择出牌。其具体实现流程图如图 5.1 所示：

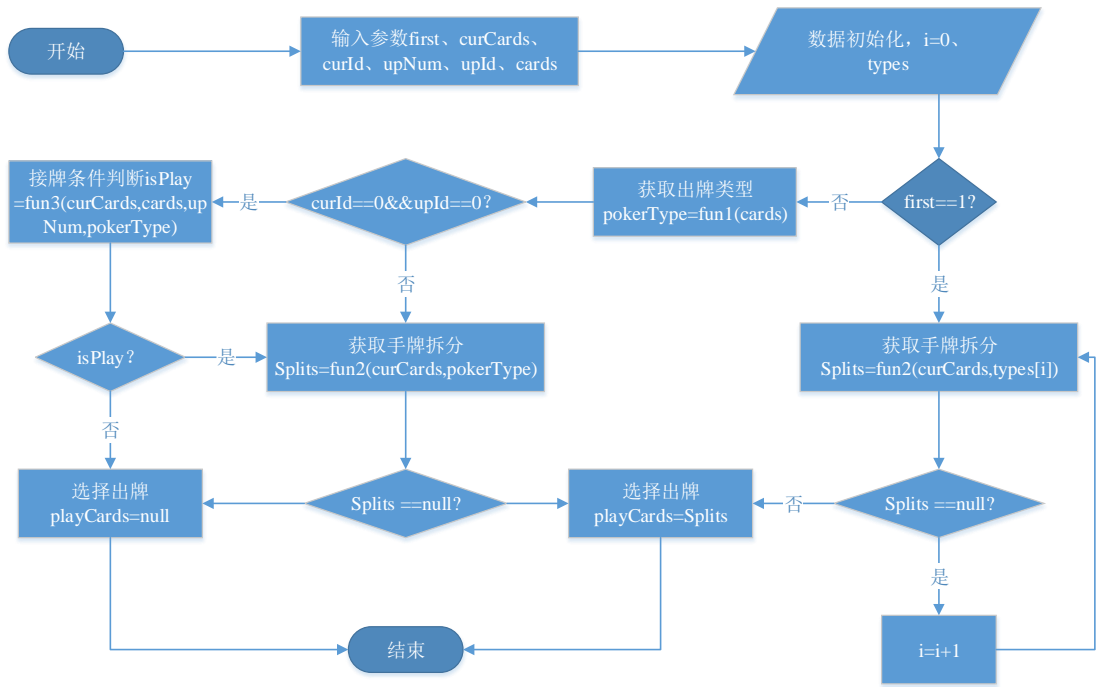


图 5.1 贪婪选择出牌算法流程图

在图 5.1 中，`fun1` 代表的函数是 `getPokerType`，`fun2` 代表的函数是 `getSplitPoker`，`fun3` 代表的函数是 `getContinuePlay`，`types` 为出牌类型对象数组。

### 5.1.2 权重选择出牌算法实现

根据权重选择出牌算法的设计采用 Python 语言进行算法的实现，算法实现具体的函数描述如下：

获取完整出牌组合的函数 `getAllHands`，输入参数为所有合法出牌类型的字典。首先统计每个牌面值出现的次数得牌面值与其对应次数的字典对象；然后遍历字典对象，根据当前牌面值出现的次数得到其对应的出牌组合（顺子类型除外）；最后遍历出牌组合的单牌类型，根据顺子类型的规则得到所有的出牌组合。

获取合法出牌组合的函数 `getLegalHands`，输入参数为当前玩家的手牌和所有出牌组合。统计当前玩家的手牌中每个值出现的次数，采用双层循环的方式获取所有的合法出牌组合，外层对所有出牌组合进行遍历，统计当前出牌组合中每个值出现的次数，内层对当前出牌组合统计结果进行遍历，比较当前玩家的手牌统计结果中和出牌组合统计结果中相同 `key` 对应的 `value` 大小，如果手牌统计结果中的值都大于出牌组合统计结果中值，则当前出牌组合则被选择为当前玩家手牌的合法出牌组合。

获取待选出牌集合的函数 `getPlaySet`，输入参数为合法出牌组合和当前玩家的手牌。采用双层循环的方式获取所有待选出牌，外层从头开始对合法出牌组合遍历，每轮都将待选的手牌初始化为当前玩家的手牌，里层对从当前出牌组合开始对合法出牌组合遍历，每轮根据当前出牌组合选择待选出牌并计算其权重值，更新待选的手牌开始下一轮选择。

选择出牌的函数 `getPlayPoker`，输入参数为待选出牌集合，通过对待选出牌集合的遍历，每轮将当前出牌的权重值与设定的最小权重值进行比较，若小于则更新设定的最小权重值为当前出牌的权重值，否则继续比较下一个，最终选择权重值最小的出牌。

根据上一出牌玩家的出牌进行选择出牌的函数 `getPlayPokerByType`，输入参数为上一出牌玩家的出牌或待选出牌集合，通过对待选出牌集合的遍历，首先每轮将当前出牌与上一出牌玩家所属类型进行比较，若所属类型相同则将其添加到可选出牌列表中，否则进一步对当前出牌所属类型进行判断，若为炸弹或王炸类型则将其添加到可选出牌列表中，然后对待选出牌列表进行是否为空的判断，如果是则返回 `None` 表示选择出牌为空，否则调用 `getPlayPoker` 函数获取选择的出牌，选择出可以大过上一出牌玩家所出牌的出牌且同时保证其权重值在待选出牌集合中最小。

权重选择出牌算法的输入参数为当前玩家的手牌 `curPokers` 和上一出牌玩家的出牌 `upPlayPokers`，`curPokers` 和 `upPlayPokers` 都是扑克牌所对应的数字表示数组，`upPlayPokers` 取值为 0 表示本轮第一手出牌，输出参数为选择的出牌 `playCards`，`playCards` 为选择出牌的数字表示数组，取值为 `None` 表示未选择出牌。其具体实现流程图如图 5.2 所示：

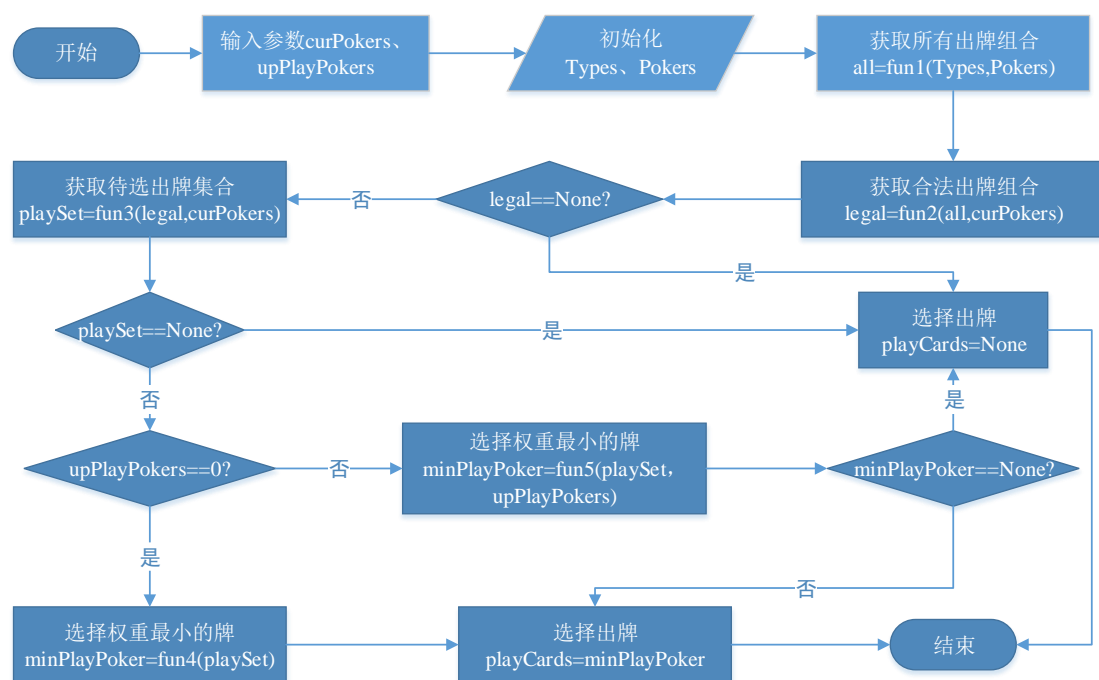


图 5.2 权重选择出牌算法流程图

在图 5.2 中, fun1 代表的函数是 getAllHands, fun2 代表的函数是 getLegalHands, fun3 代表的函数是 getPlaySet, fun4 代表的函数是 getPlayPoker, fun5 代表的函数是 getPlayPokerByType, Types 是所有出牌类型及对应的权重值的字典对象, Pokers 为一副完整扑克牌对应的数字的数组。

### 5.1.3 算法排序实现

单机模式中三个难度等级对应算法的排序依据是三个算法进行斗地主试验的获胜比例。试验的设计过程如下：让三个算法模拟三个玩家在无人为因素干扰的情况下进行斗地主游戏，发牌、抢地主的初始位置均采用随机数来确定，是否进行抢地主和抢地主分数也采用随机数确定，出牌过程调用算法对应的接口完成。在斗地主游戏过程中，发牌结束后调用游戏对局数据添加接口对游戏数据进行保存，地主确定后调用游戏对局数据更新接口对游戏数据进行更新，出牌过程中调用游戏出牌数据添加接口对出牌数据进行保存，游戏结束时调用游戏对局数据更新接口对游戏数据进行更新。通过对试验的游戏数据进行统计，试验共进行 9656 局有效的斗地主游戏，三个算法的胜场及胜率统计结果如表 5.1 所示：

表 5.1 算法排序试验结果统计表

|  | 贪婪选择出牌算法 | 权重选择出牌算法 | MCTS 算法 |
|--|----------|----------|---------|
|--|----------|----------|---------|

|        |          |      |      |      |
|--------|----------|------|------|------|
| 自己获胜   | 地主       | 1048 | 1213 | 1767 |
|        | 农民       | 564  | 2177 | 2887 |
| 队友获胜   | 贪婪选择出牌算法 |      | 564  | 564  |
|        | 权重选择出牌算法 | 2177 |      | 2177 |
|        | MCTS 算法  | 2887 | 2887 |      |
| 自己获胜比例 |          | 17%  | 35%  | 48%  |
| 总体获胜比例 |          | 69%  | 70%  | 76%  |

通过对统计的试验数据进行分析,可得出以下几点分析结果:第一,通过总体获胜比例来看,贪婪选择出牌算法和权重选择出牌算法相当,MCTS 算法相较其他两个算法略胜;第二,通过自己获胜的比例来看,MCTS 算法最好,权重选择出牌算法次之,贪婪选择出牌算法最差,表明贪婪选择出牌算法还有提升的空间。综合以上的试验结果和分析结果,三个算法的智能程度从高到低的排序为:MCTS 算法、权重选择出牌算法、贪婪选择出牌算法,即单机模式中简单难度等级对应贪婪选择出牌算法、一般难度等级对应权重选择出牌算法、困难难度等级对应 MCTS 算法。

#### 5.1.4 第一视角实现

斗地主是三个玩家同时在线进行的游戏,在服务端对三个玩家进入游戏的顺序进行记录,作为三个玩家位置确定标识和循环出牌顺序。但是在客户端按照三个玩家进入的顺序进行游戏界面的渲染,就会产生只有一个玩家是以第一视角进行游戏,其他两个玩家都是非第一视角的问题。本系统采用以下的方式实现三个玩家都以第一视角进行游戏,具体实现的步骤如下:

第一,在服务端使用 1、2、3 记录玩家进入游戏的顺序,并将玩家对应的顺序位置返回到客户端进行存储,以下使用 1 号代替顺序为 1 的玩家,使用 2 号代替顺序为 2 的玩家,使用 3 号代替顺序为 3 的玩家。在客户端界面采用逆时针排序方式规定 1 号、2 号、3 号玩家的显示位置:1 号玩家显示在界面下方,即玩家的第一视角;2 号玩家显示在界面的右边;3 号玩家显示在界面的左边。

第二,在客户端对玩家对应的顺序进行判断,若玩家为 1 号,不进行任何转换操作;若玩家为 2 号,则按照顺时针方向进行玩家位置的转换;若玩家为 3 号,

则按照逆时针方向进行玩家位置的转换。经过这样的转换操作之后，对于每个玩家来说视角都是在 1 号玩家的位置。

第三，根据玩家位置的转换结果进行游戏界面的渲染，便可保证三个玩家都以第一视角进行游戏，但是对服务端来说，记录的进入游戏顺序不产生任何影响。

### 5.1.5 实时通讯实现

实时通讯使用基于 Django 的 Channels 组件实现，在系统中根据不同的在线游戏模式设计不同的实时通讯接口，包括在线模式实时通讯接口、锦标赛模式实时通讯接口、好友模式实时通讯接口。实时通讯接口与前端界面采用 WebSocket 协议进行连接，该协议的通讯通道采用全双工的方式，保证通讯的两端可以任意的给对方发送消息。在本系统中，实时通讯接口和前端界面采用相同的数字来表示通讯时发送的消息类型。

在线模式实时通讯接口是根据在线斗地主所需功能及特点所设计，包括对斗地主过程中游戏桌初始化、加入游戏桌、发牌、抢地主、更新玩家身份、出牌、更新赢家及积分，其中发牌、抢地主、更新玩家身份、出牌、更新赢家及积分部分需对数据库进行更新或添加操作。与前端消息类型对应的数字包括 0 表示游戏桌初始化、1 表示加入游戏桌、2 表示发牌、3 表示抢地主、4 表示更新玩家身份、5 表示出牌、6 表示更新赢家及积分、7 表示没人抢地主游戏结束、8 表示有一个退出游戏、9 表示有两人退出了游戏且游戏结束、10 表示程序错误导致游戏结束、11 表示该游戏桌的玩家人数已满加入失败，12 表示超级管理员查询在线游戏的玩家数据。

锦标赛模式实时通讯接口是根据锦标赛的功能及特点所设计，包括对比赛过程中加入锦标赛、游戏分桌、发牌、抢地主、更新玩家身份、出牌、记录赢家及积分、淘汰玩家、锦标赛排名，其中游戏分桌、发牌、抢地主、更新玩家身份、出牌、淘汰玩家部分需对数据库进行更新或添加操作。与前端消息类型对应的数字包括 0 表示加入锦标赛、1 表示开始锦标赛且对玩家进行游戏分桌、2 表示发牌、3 表示抢地主、4 表示更新玩家身份、5 表示出牌、6 表示记录赢家及积分、7 表示淘汰玩家、8 表示有一个退出游戏、9 表示有两人退出了游戏且游戏结束、10 表示程序错误导致游戏结束

好友模式实时通讯接口是根据好友一起玩的游戏功能及特点所设计,包括对斗地主游戏过程中的创建房间、加入房间、发牌、抢地主、更新玩家身份、出牌、更新赢家及积分,其中发牌、抢地主、更新玩家身份、出牌、更新赢家及积分需对数据库进行更新或添加操作。与前端消息类型对应的数字包括 0 表示创建房间、1 表示加入房间、2 表示发牌、3 表示抢地主、4 表示更新玩家身份、5 表示出牌、6 表示更新赢家及积分、7 表示没人抢地主游戏结束、8 表示有一个退出游戏、9 表示有两人退出了游戏且游戏结束、10 表示程序错误导致游戏结束、11 表示创建房间失败、12 表示人数已满加入房间失败。

前端界面根据实时通讯接口返回的消息类型对界面进行不同操作元素的渲染或者作出相应的系统响应,同时将相应的操作信息通过实时通讯接口发送到服务端。

## 5.2 系统功能实现

系统实现是采用相应的编程语言、数据库技术根据系统的详细设计进行功能开发的过程<sup>[49]</sup>。本系统采用 Python 作为主要开发语言,采用 HTML5、CSS3、JavaScript 进行前端界面开发及业务逻辑处理,采用 Ajax 技术实现前端界面与接口的连接,采用 WebSocket 与 Channels 建立实时通信的双向消息发送通道,采用 MySQL 进行数据的存储。按照系统详细设计进行系统功能的开发,确保所有设计的功能都包含在所开发的系统中。

### 5.2.1 用户注册与登录

系统注册功能实现的界面如下,在该界面输入用户名、密码和性别进行用户注册。注册接口中使用密码本身作为 salt 对密码进行 hash 加密,增强密码存储的安全性;对用户名进行验证,保证用户名的唯一性。注册成功则自动返回登录界面,注册失败系统则给出相应提示。

系统登录功能实现的界面如下,在该界面输入用户名、密码和验证码进行登录验证。登录接口中使用 hash 对输入的密码进行加密与输入用户名对应的密码进行比较,登录成功则打开系统的主界面,接口添加用户的 session 值,用于



数据操作时的身份验证；登录失败系统则给出相应提示。用户登录界面如图 5.3 所示：

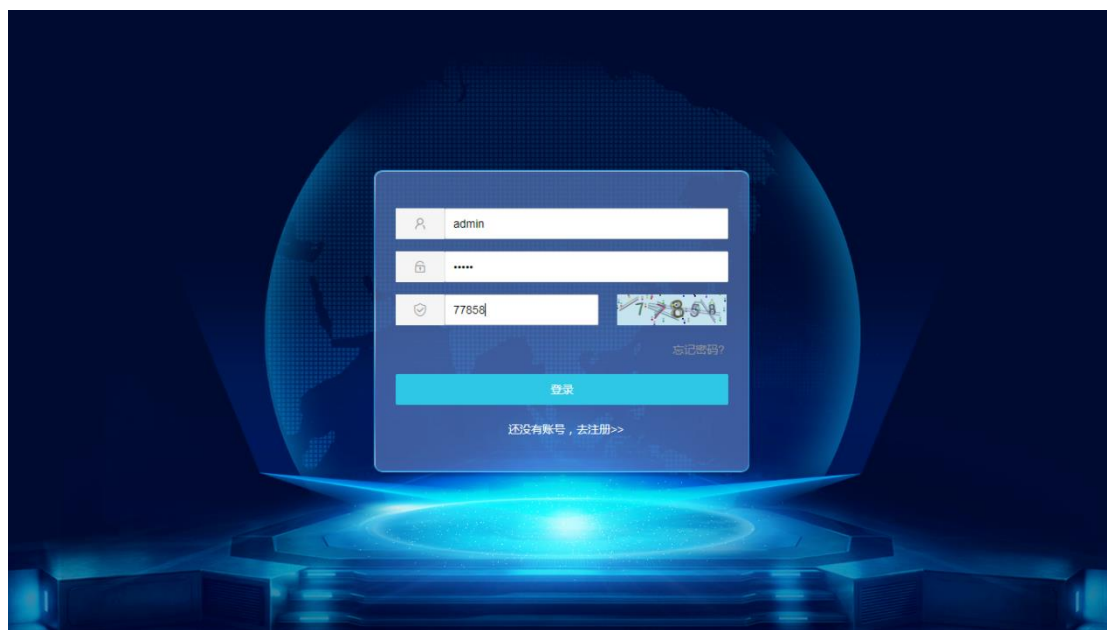


图 5.3 用户登录界面图

## 5.2.2 用户管理

用户管理功能实现的是超级管理员和管理员对用户的基本数据进行管理和菜单权限进行设置。数据管理的前提是已成功登录系统，超级管理员可对系统的所有用户数据进行管理和权限设置；管理员只可对自己创建的用户进行管理和权限设置且权限设置不能超出本身的权限范围。权限设置界面的菜单数据采用可多选的树形结构进行显示，便于管理员对功能模块及子功能之间的关系进行查看，功能模块和子功能的勾选或取消是自动进行级联的，便于管理员进行设置。用户管理功能实现界面包括数据查询、添加、修改、查看、权限设置五个界面，每个界面都有相对应的数据接口，在添加和修改接口中对用户名的唯一性进行检查。其他数据管理功能与用户管理功能的界面基本一致，仅仅是每个管理功能管理的数据不同。用户管理数据查询界面如图 5.4 所示：

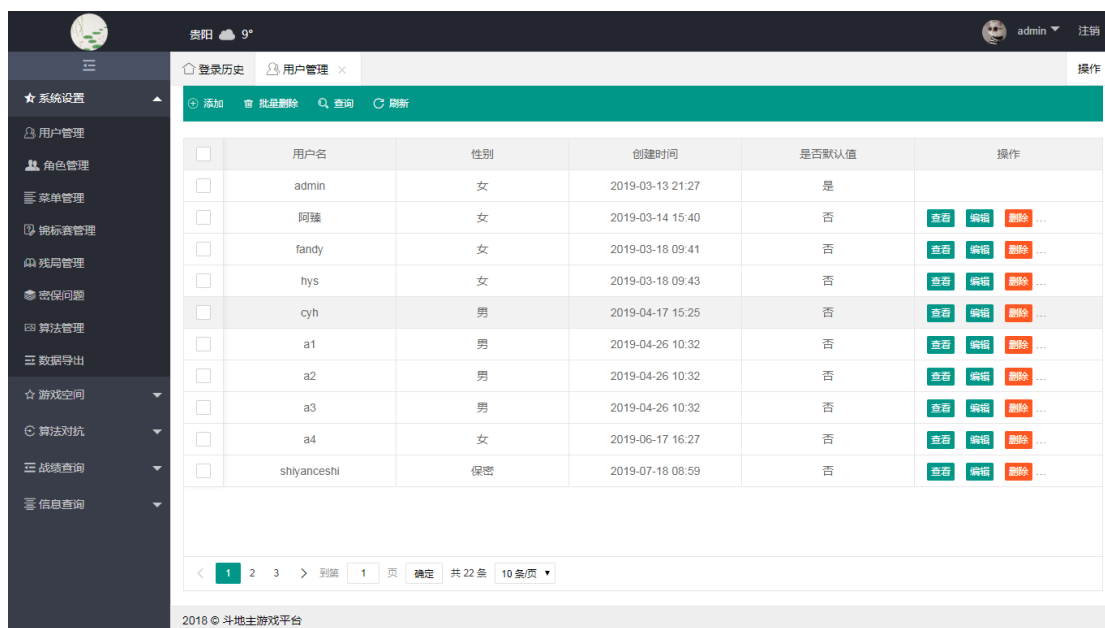


图 5.4 用户数据查询界面图

用户权限设置界面如图 5.5 所示：

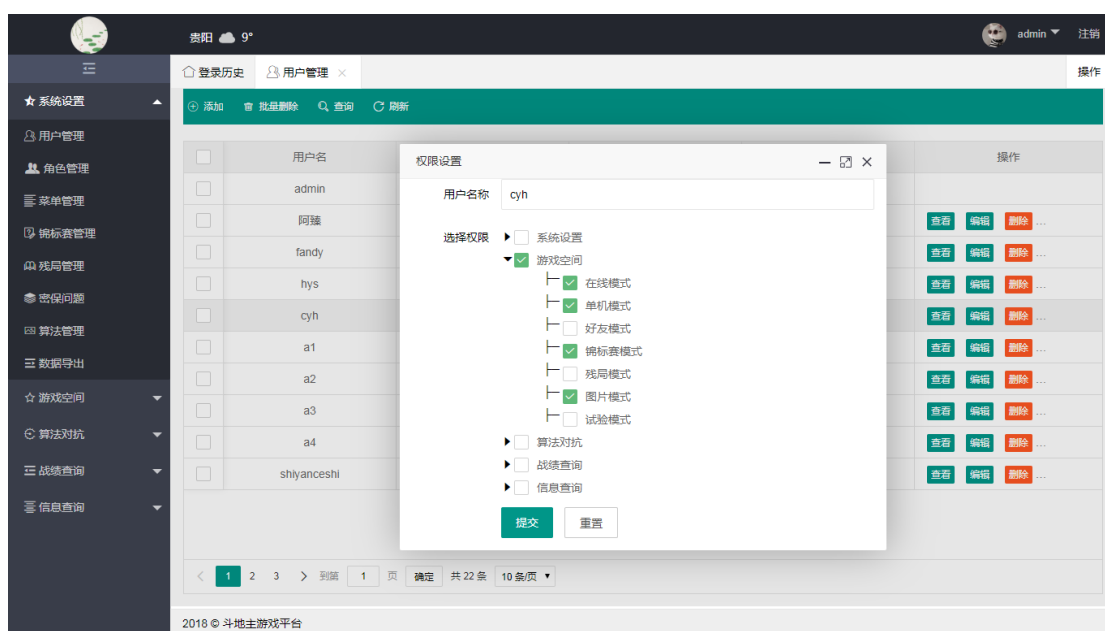


图 5.5 用户权限设置界面图

### 5.2.3 单机模式

单机模式实现的是为玩家提供无需其他用户参与的斗地主游戏模式。在界面上方显示简单、一般、困难等级的按钮供玩家进行等级选择；玩家点击开始游戏则调用创建游戏接口进行单机斗地主游戏的创建，创建成功则在界面上渲

染玩家的手牌对应图片，显示抢地主对应的分数按钮，地主产生后调用游戏身份更新接口对创建的游戏数据进行更新，在界面上显示两种不同的图片对玩家的地主或农民身份进行标识；在游戏过程中，显示出牌文字对玩家的出牌状态进行标识，在界面下方显示实时的出牌情况，调用出牌数据保存接口对每轮的出牌数据进行存储，出牌数据包含三个玩家当前游戏中的各项数据，其对应的是扑克牌图片所代表数字组成的字符串，若当前出牌的玩家是左边电脑，则数据库存储的下家数据即为当前玩家的数据，数据库存储的上家数据即为右边电脑玩家的数据，轮流出牌过程中各玩家的数据以此类推；游戏结束时调用游戏赢家更新接口对创建的游戏数据进行更新。单机模式界面图如图 5.6 所示：

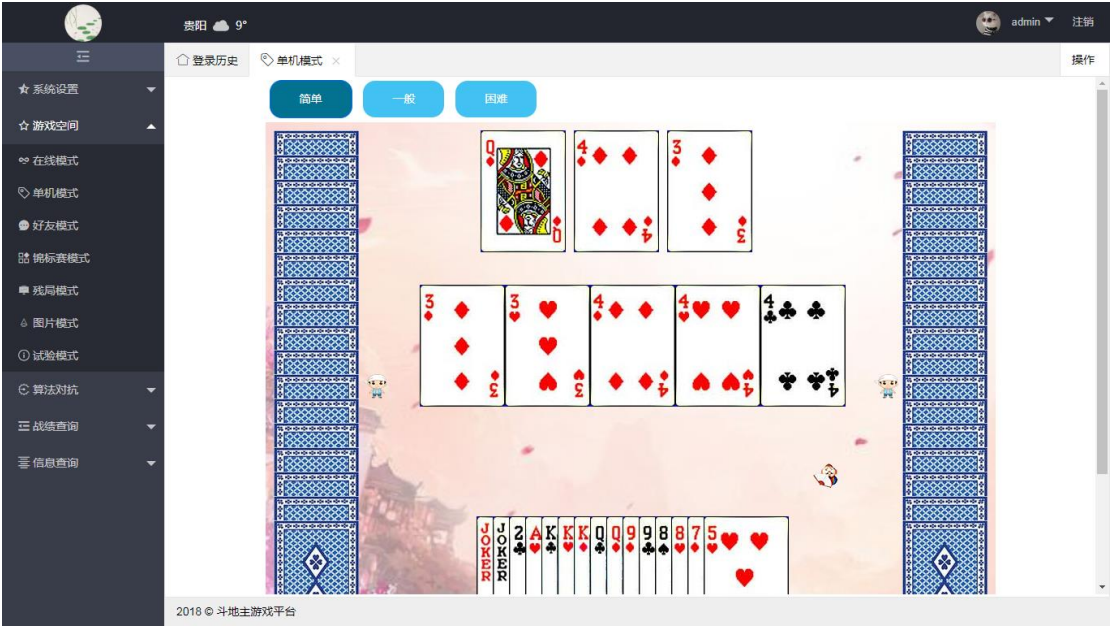


图 5.6 单机模式界面图

5.2.4 在线模式

在线模式基于在线模式实时通讯接口采用固定桌子方式实现玩家在线进行斗地主游戏。该功能的界面包括游戏桌界面和游戏界面，玩家打开游戏桌界面时则与通讯接口建立 WebSocket 连接，发送获取游戏桌状态请求并根据获取的数据在界面上进行渲染，有人的位置则显示用户头像和用户名信息，没有人的位置则显示进入游戏，玩家点击进入游戏便可打开游戏界面。在游戏界面点击开始游戏进入等待游戏的状态，当该游戏桌人数满三人时便开始游戏，游戏的

过程与单机模式一样，只是调用的数据接口不同。在游戏过程中出现一个玩家掉线的情况，采用权重选择出牌算法代替玩家进行出牌，保证本局游戏的正常进行。游戏过程中使用 MCTS 算法提供智能出牌提示功能，在界面上点击提示功能，调用 MCTS 算法接口获取出牌。游戏中抢地主界面图如图 5.7 所示：

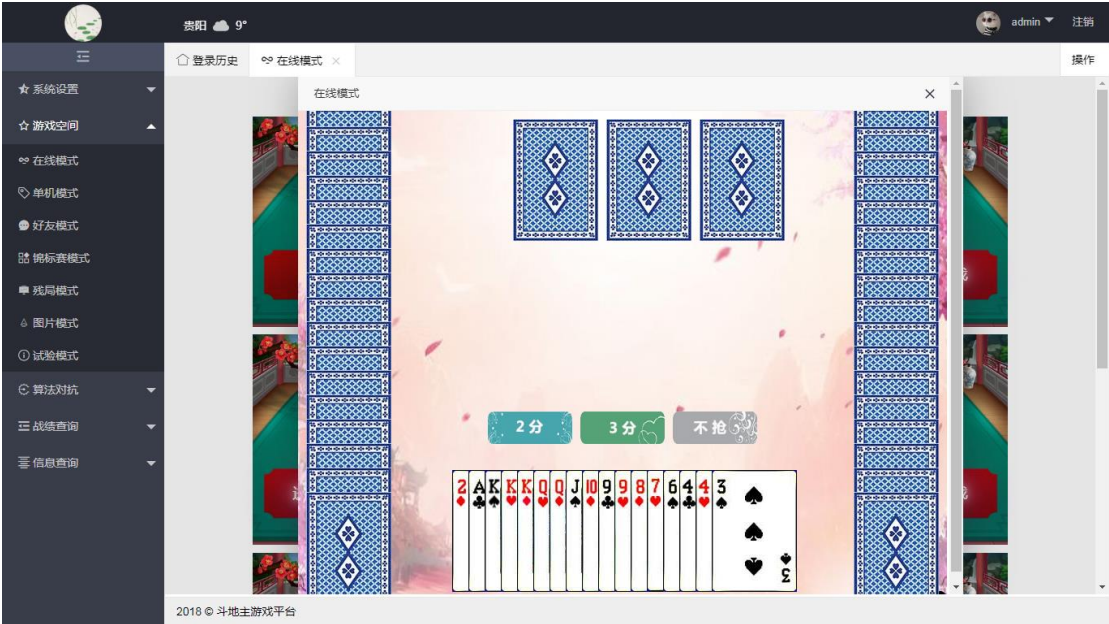


图 5.7 在线模式抢地主界面图

在线模式游戏中出牌界面图如图 5.8 所示：

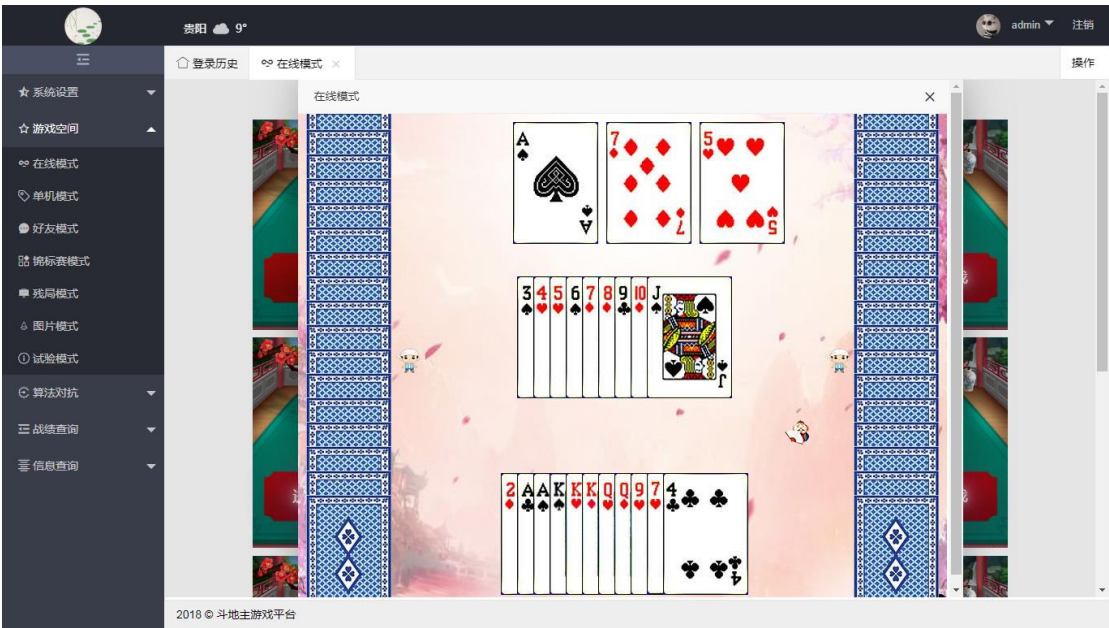
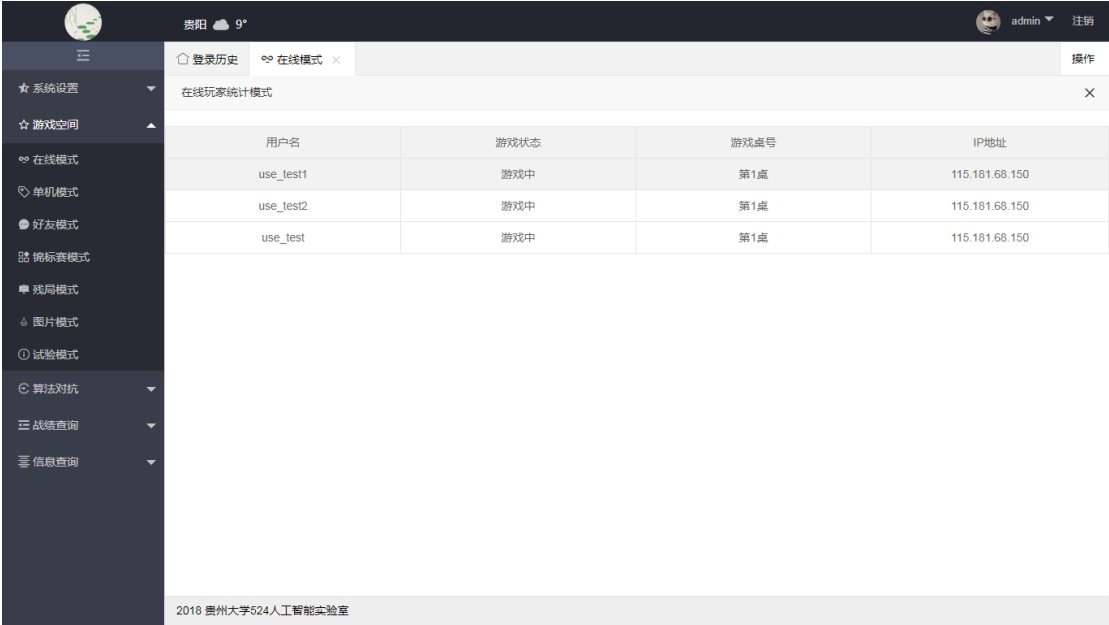


图 5.8 在线模式出牌界面图

若当前用户角色为超级管理员则可通过对在线模式游戏中的玩家数据进行查询，获取所有在线玩家的数据并以列表形式对玩家数据进行展示。在线模式超级管理查看在线玩家数据界面图如图 5.9 所示：



| 用户名       | 游戏状态 | 游戏桌号 | IP地址           |
|-----------|------|------|----------------|
| use_test1 | 游戏中  | 第1桌  | 115.181.68.150 |
| use_test2 | 游戏中  | 第1桌  | 115.181.68.150 |
| use_test  | 游戏中  | 第1桌  | 115.181.68.150 |

图 5.9 在线玩家信息界面图

5.2.5 好友模式

好友模式基于好友模式实时通讯接口采用房间方式实现玩家好友之间在线进行斗地主游戏。玩家打开好友模式界面时与通讯接口建立 WebSocket 连接，玩家可以通过两种方式进行游戏，一是玩家点击创建房间按钮向接口发送房间创建请求并将房间号显示在界面上，切换玩家状态为等待游戏中，创建失败时系统则给出玩家提示，玩家可通过刷新页面或者关闭页面重新打开进行房间创建的尝试。二是玩家输入房间号，点击加入房间按钮向接口发送加入房间请求，加入成功则在界面上显示已在房间中的玩家头像和用户名信息，切换玩家状态为等待游戏中；加入失败的可能性包括房间人数已满、房间已销毁、网络错误，当出现加入失败时系统根据错误类型给出相应的提示信息。当房间玩家人数为三人时开始游戏，游戏的过程与单机模式一样，但调用的数据接口不同。当出现玩家掉线的情况时，处理方式与在线模式一样。好友模式游戏中的出牌界面图如图 5.10 所示：



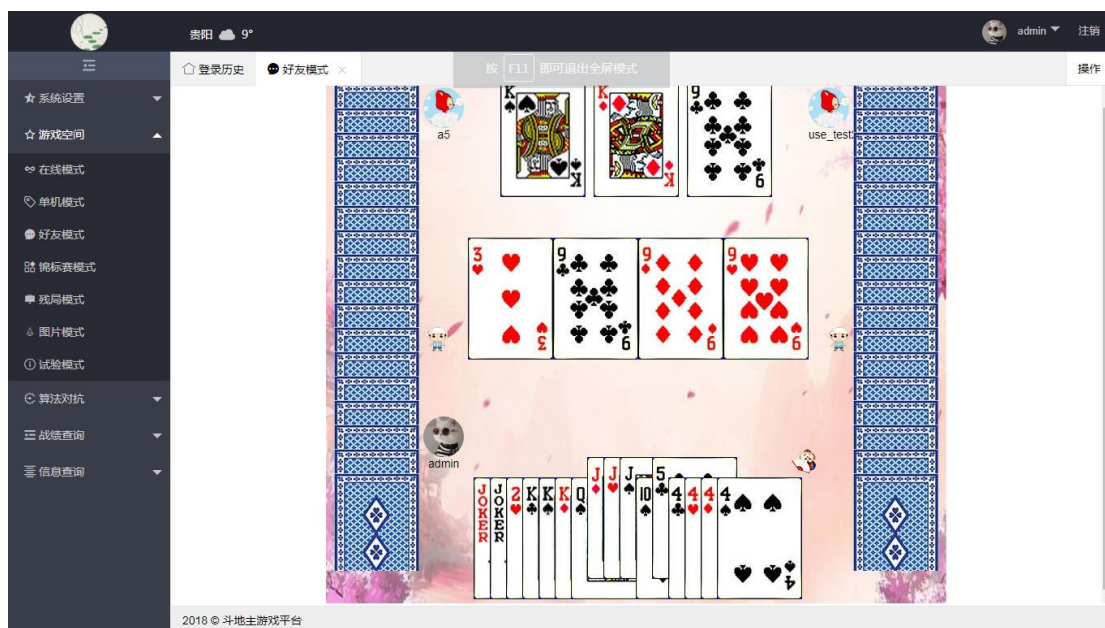


图 5.10 好友模式出牌界面图

### 5.2.6 锦标赛模式

锦标赛模式基于锦标赛实时通讯接口采用报名方式实现玩家之间进行在线斗地主游戏。该功能的界面包括锦标赛界面和游戏界面，锦标赛界面是对所有的锦标赛进行展示，对每个锦标赛的状态进行显示，显示文字为报名的锦标赛表示可进行报名。点击报名则打开游戏界面并与通讯接口建立 **WebSocket** 连接，连接建立成功则切换玩家的状态为等待比赛，调用锦标赛报名数据添加接口对用户的报名数据进行添加；当玩家人数达到锦标赛设置的人数时则开始比赛，切换玩家状态为比赛中并调用锦标赛数据更新接口对锦标赛的状态进行更新，确保在锦标赛进行过程中新的玩家无法加入到比赛中；接下来对玩家进行三人一组的随机分组，分组之后便开始斗地主锦标赛的比赛过程，游戏过程与单机模式一样，但调用的数据接口不同。每局斗地主游戏结束时按照锦标赛设计中制定的积分规则对玩家的积分和胜场进行更新，调用锦标赛数据的更新接口对相应玩家的积分和胜场数据进行更新；每轮比赛结束时对本轮玩家人数进行判断，若本轮玩家人数为三，则根据玩家最终的积分和胜场进行排名，否则按照锦标赛设计中的淘汰规则对本轮玩家进行淘汰，选取符合条件的玩家进入

下一轮的比赛，被淘汰的玩家直接剔除锦标赛，并且切换玩家的状态为比赛结束。锦标赛模式锦标赛界面如图 5.11 所示：



图 5.11 锦标赛界面图

锦标赛模式游戏中的出牌界面图如图 5.12 所示：

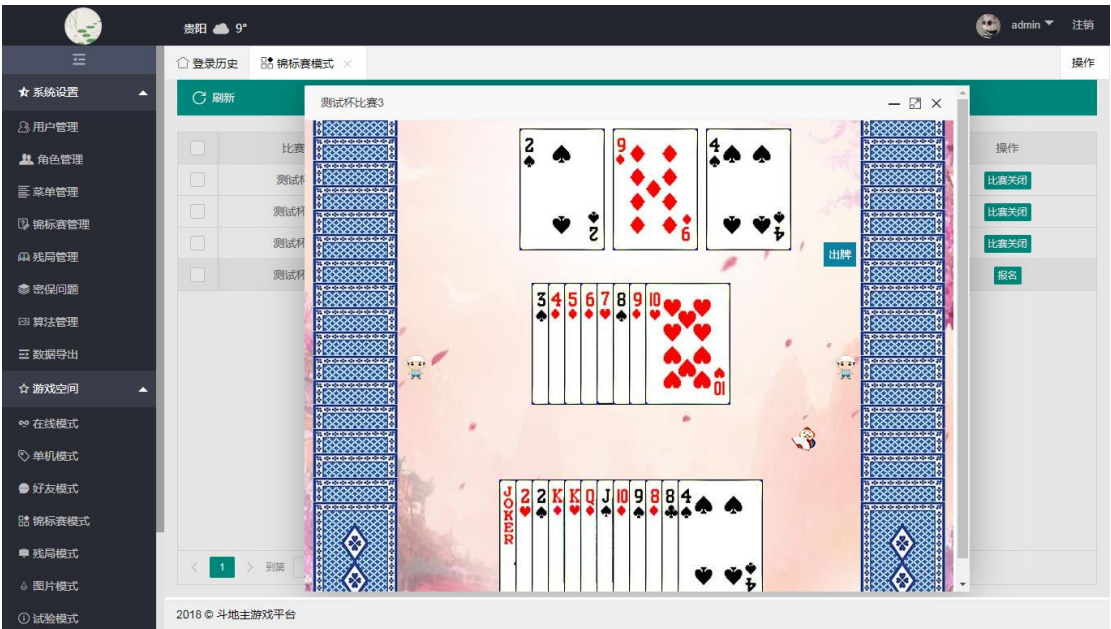


图 5.12 锦标赛模式出牌界面图

5.2.7 残局模式

残局模式实现的是玩家对残局斗地主关卡的挑战。该功能实现的界面包括难度选择界面、关卡选择界面和游戏界面。难度选择界面显示的是难度等级对应的按钮；点击难度等级按钮打开关卡选择界面，调用关卡查询接口获取该难度等级对应的关卡数据和进度查询接口获取该玩家在对应难度等级的挑战进度，根据玩家的挑战进度对关卡的状态进行控制，蓝色关卡表示可点击，灰色关卡表示不可点击；点击对应的关卡打开游戏界面，调用关卡数据查询接口获取关卡的残局斗地主数据，根据数据在界面上渲染对应的扑克牌图片，默认是玩家开始出牌，出牌时调用挑战出牌数据保存接口对出牌数据进行保存，出牌数据是扑克牌图片对应的数字所组成的字符串。当挑战结束时，系统根据挑战结果做出相应的响应。残局模式游戏关卡挑战界面图如图 5.13 所示：

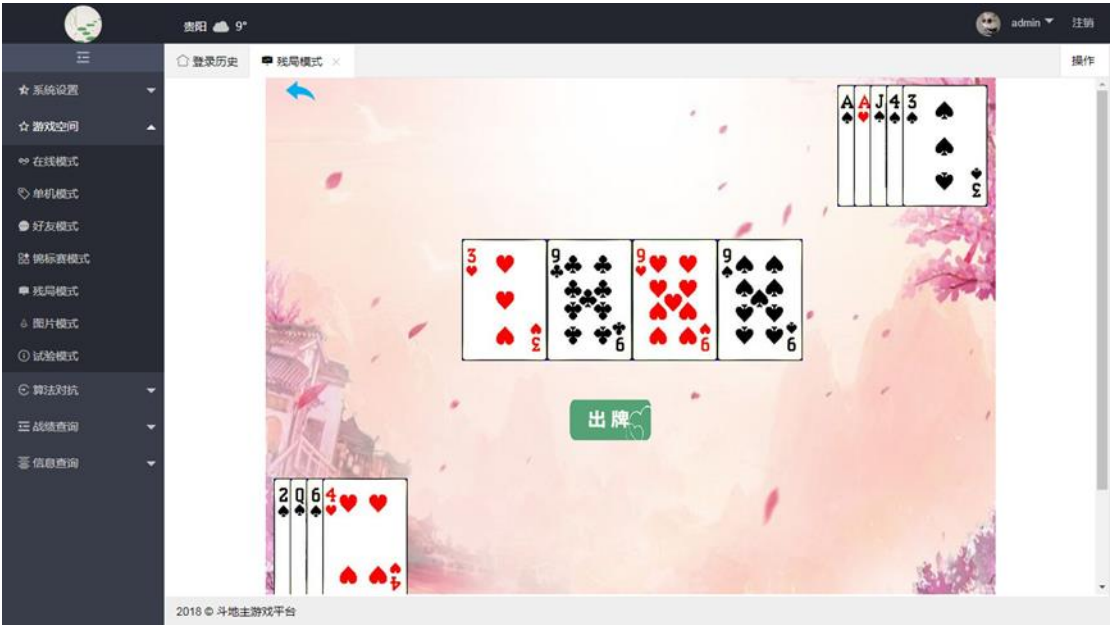


图 5.13 关卡挑战界面图

5.2.8 算法对抗

算法对抗模块包括算法管理、对抗试验和数据查询功能，算法管理是研究人员对上传的智能算法进行管理，上传的智能算法文件中必须包含出牌函数，



出牌函数所有中使用的参数和函数返回值都需遵循系统所制定的标准，参数中扑克牌的表示符号与表 4.1 中的显示符合一致，系统提供的参数如表 5.1 所示：

表 5.1 系统对外提供参数表

| 参数名                  | 参数解释                 | 参数类型   | 取值范围         | 特殊值含义                          | 备注                              |
|----------------------|----------------------|--------|--------------|--------------------------------|---------------------------------|
| current_player       | 当前玩家位置<br>(该他出牌)     | int    | 1、2、3        |                                | 1：地主，2：地主上家，3：地主下家              |
| current_player_cards | 当前玩家手牌<br>(手中剩余的牌)   | string | 长度范围<br>1-20 |                                | 顺序为按照手牌大小降序排列                   |
| no1_done_cards       | 地主每轮出的牌<br>(当前轮除外)   | array  | 最大长度<br>20   | ['-1']表示没有出过牌<br>'0'元素示 pass   | 数组中的元素也为数组，每个元素表示一轮的出牌情况        |
| no2_done_cards       | 地主下家每轮出的牌<br>(当前轮除外) | array  | 最大长度<br>20   | ['-1']表示没有出过牌<br>'0'元素示 pass   | 数组中的元素也数组，每个元素表示一轮的出牌情况         |
| no3_done_cards       | 地主上家每轮出的牌<br>(当前轮除外) | array  | 最大长度<br>20   | ['-1']表示没有出过牌<br>'0'元素示 pass   | 数组中的元素也数组，每个元素表示一轮的出牌情况         |
| current_round_cards  | 当前轮出牌                | array  |              | ['-1']表示本轮第一个出牌<br>'0'元素示 pass | 根据 current_player 值倒推本轮第一个出牌人角色 |

系统提供的智能算法文件中出牌函数实例如图 5.14 所示：

```
1) def test(**kwargs):
2)     # kwargs 为字典类型，包含系统提供的所有参数
3)     # 获取对应参数对应的值
4)     current_player=int(kwargs.get("current_player",0))
5)     current_player_cards = kwargs.get("current_player_cards","")
6)     no1_done_cards = kwargs.get("no1_done_cards",[])
7)     no2_done_cards = kwargs.get("no2_done_cards", [])
8)     no3_done_cards = kwargs.get("no3_done_cards", [])
9)     current_round_cards = kwargs.get("current_round_cards", [])
10)    #执行智能算法
11)    cardsResult=AlgorithmFunction()
11)    #解析智能算法结果获取出牌
11)    playCards =parserResult(cardsResult)
12)    #返回参数为字符串类型，返回值为",0,0'0'都可以表示不出
13)    return playCards
```

图 5.14 出牌函数实例

对抗试验实现的是研究人员通过设定一系列的参数进行智能算法对抗试验的功能。首先，用户在界面上设置试验局数、开始发牌位置、地主玩家位置三个参数，其中试验局数最大值为 1000，最小值为 1，开始发牌位置和地主玩家

位置的取值为 1、2、3；然后对算法对抗试验中的智能算法进行选择，选择的算法可以是任意组合的三个算法；最后点击开始按钮进行对抗试验，在试验过程中根据选择的智能算法动态调用对应的算法接口进行出牌，对试验过程中游戏数据进行收集并存储。游戏过程中动态调用智能算法接口完成自动出牌过程如下所述：

第一，根据当前出牌的算法名称查询该算法对应的.py 文件名和调用的具体函数名，然后通过 Ajax 请求向智能算法接口发送文件名、函数名、当前游戏中各玩家的手牌和身份以及当前游戏出牌等数据。

第二，智能算法接口接收相应的参数数据，解析出文件名和函数名数据，利用 Python 可动态加载模块的特性根据文件名加载该算法对应的模块，通过 Python 提供的 eval 函数执行动态模块下的函数名所对应函数，并返回函数执行的结果。

第三，对返回的执行结果进行转换并自动进行手牌选择，对选择的手牌进行合法类型检查，若合法则打出所选的手牌，并通知下一玩家进行出牌，系统根据当前出牌玩家所对应的算法循环进行智能算法动态调用过程，直到整个试验结束。

进行算法对抗试验的界面图如图 5.15 所示：

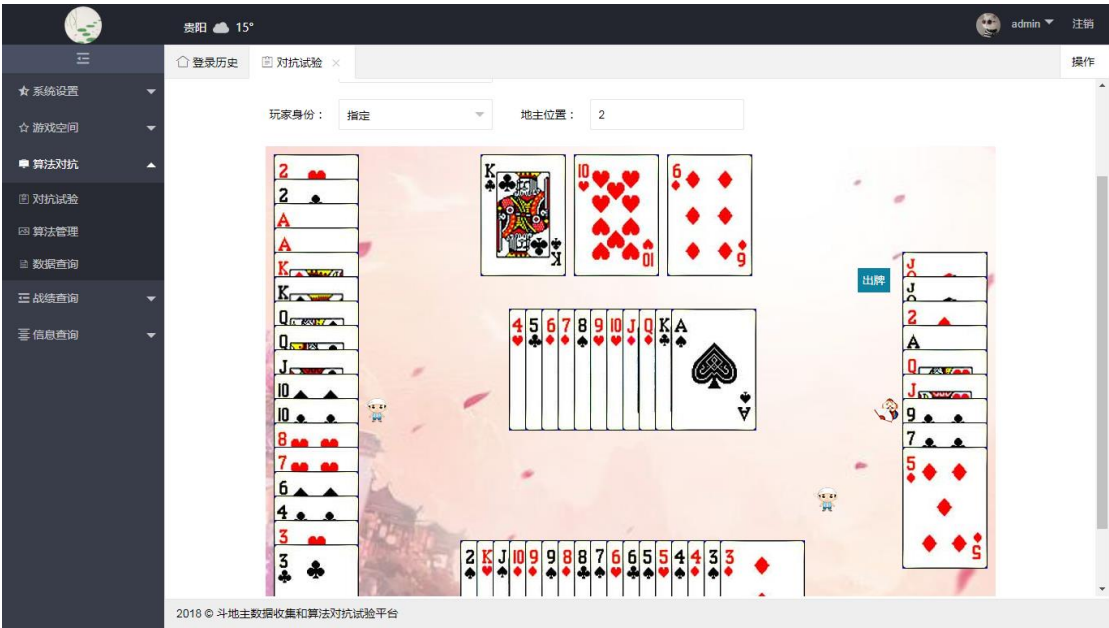


图 5.15 算法对抗试验界面图

5.2.9 战绩查询

战绩查询实现的是对单机模式、在线模式、锦标赛模式、好友模式、残局模式的游戏战绩进行查询。该功能实现的界面包括数据查询界面、对局详情界面、对局回放界面，数据查询界面是对查询的战绩信息进行显示，可对数据进行翻页查询，每条数据都有对局详情和对局返回查看按钮。点击对局详情按钮打开对局详情界面，调用出牌数据详情接口获取该局游戏的详细出牌数据，对获取的详细出牌数据进行显示，使用淡绿色对当前用户的出牌进行标识。点击对局回放按钮打开对局回放界面，调用游戏数据详情接口获取该局游戏的详情数据，根据详情数据在界面上渲染三个玩家的手牌和游戏地主牌对应的扑克牌图片、玩家身份标识图片，调用出牌数据详情接口获取该局游戏的详细出牌数据，根据详细出牌数据构造模拟游戏过程的出牌队列并开始模拟出牌过程。该界面上还有暂停、继续、重新开始按钮，模拟过程过程中，暂停按钮与继续按钮状态总是处于相反的状态，模拟过程结束时，重新开始按钮处于激活状态，暂停、继续按钮处于未激活状态，点击重新开始按钮则重新开始对局回放过程。锦标赛模式对局回放界面图如图 5.16 所示：

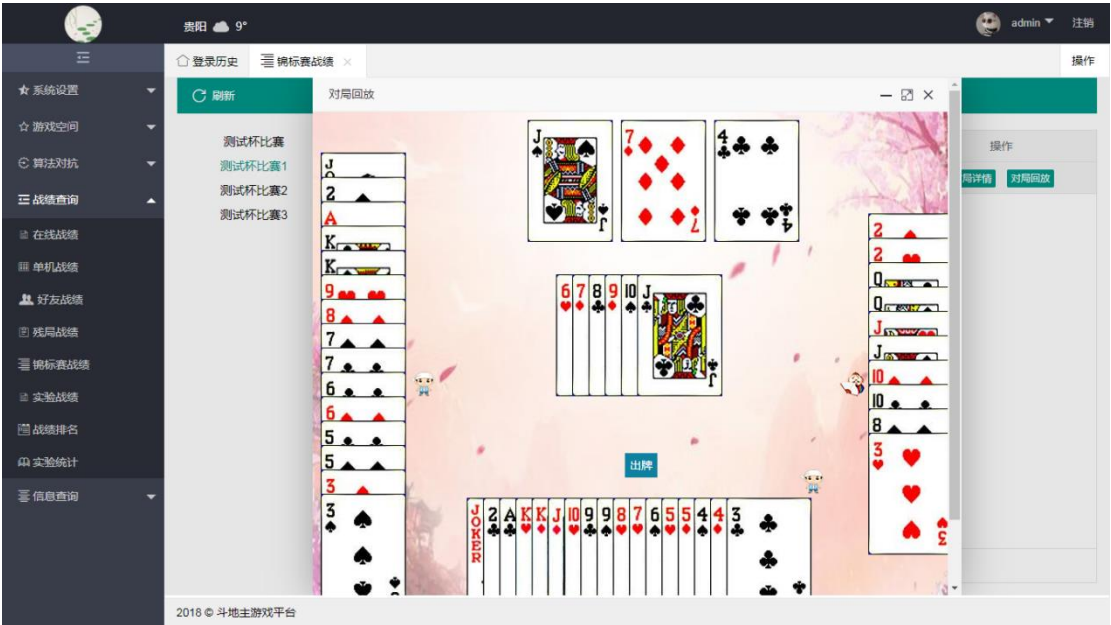


图 5.16 对局回放界面图

战绩查询还实现对用户的战绩排名功能，该功能实现的界面包括战绩排名界面、个人战绩界面，战绩排名界面是调用战绩排名查询接口根据游戏模式和

排名方式条件对参加过该游戏模式的用户进行排名，排名数据结合 Echarts 使用柱状图和图形化方式进行展示。点击图形化中的用户名打开个人战绩界面，调用个人战绩统计接口对个人战绩进行统计，按照游戏模式进行游戏场次数和胜场数进行数据统计和展示。战绩排名界面图如图 5.17 所示：

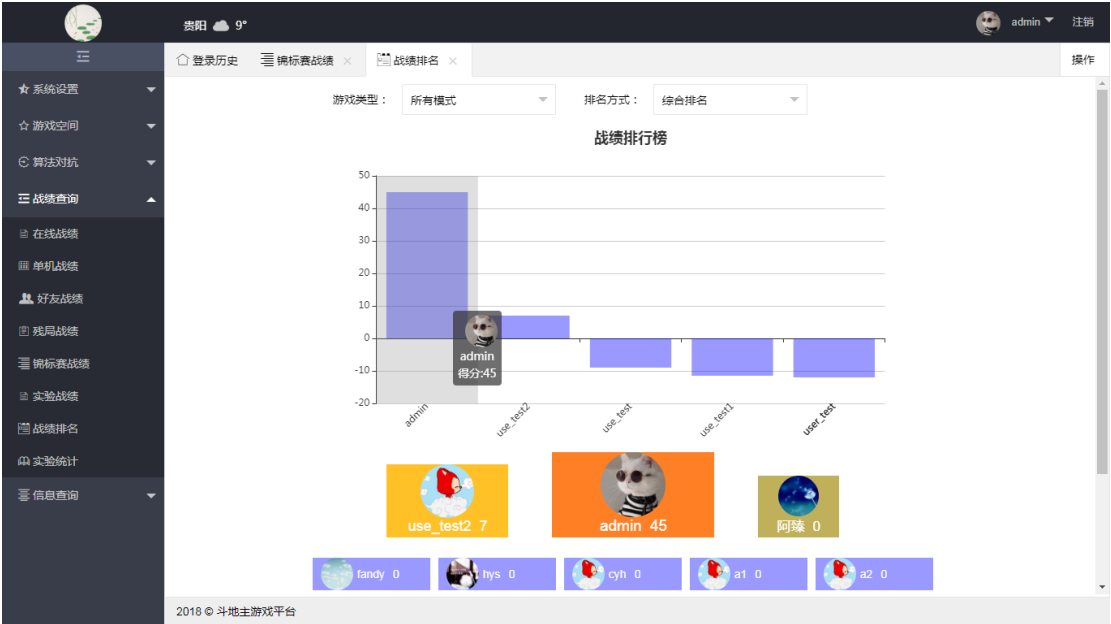


图 5.17 战绩排名界面图

### 5.3 本章小结

本章介绍本文系统实现阶段完成的工作，首先介绍系统关键功能的实现，使用 JavaScript 语言对贪婪选择出牌算法进行实现和使用 Python 语言对权重算法进行实现，在此基础之上对单机模式中三个难度等级对应的算法排序通过试验的方式进行确定，第一视角功能的实现解决三个玩家都能以第一视角进行游戏的问题，实时通讯是本文在线游戏中最关键的技术，解决在线游戏中玩家之间进行实时通讯的问题；然后介绍系统实现的成果，介绍系统实现功能具体实现的界面，对每个界面显示的内容和调用的数据接口进行描述，并使用界面截图对系统运行结果进行展示。

## 第六章 系统测试

### 6.1 测试系统部署及测试环境

本系统部署在腾讯云服务器上进行测试，服务器配置的操作系统为 Ubuntu Server 18.04.1 LTS 64 位，CPU 为 1 核，内存为 2GB，公网带宽为 1Mbps。数据库为 MySQL 5.7，内存数据为 Redis，Python 环境配置为 Python 3.6，Django 版本为 Django 2.2，Django Channels 版本为 2.1.7。Django 项目使用 uwsgi 进行部署，Django Channels 项目接口使用 Daphne 进行部署，最后使用 Nginx 对服务进行代理，统一通过通用的 Web 服务的 80 端口开放对外访问，Nginx 具有处理负载均衡的能力，可保证承受较多用户同时访问的压力<sup>[50]</sup>。

对本系统进行测试的应用地址为 <http://172.81.251.56/>，测试的计算机配置如下：操作系统为 Windows7 64 位，内存为 4GB，处理器为 Intel i5-3230M，网络环境为以太网。测试使用的浏览器有 IE（Edge）、Chrome 和 Firefox，系统设置模块中的管理功能、单机模式和残局模式游戏的测试只需单个用户登录即可完成，因此采用 Chrome 浏览器进行管理功能和单机斗地主游戏的测试；在线模式、锦标赛模式和好友模式游戏需要至少三个玩家同时在线才能进行测试，所以使用 IE（Edge）、Chrome 和 Firefox 浏览器登录三个不同用户进行各模式的斗地主游戏测试，其中测试锦标赛模式游戏之前需对锦标赛数据进行添加。

### 6.2 测试用例

#### 6.2.1 算法测试

算法测试是对本文设计并实现的贪婪选择出牌算法和权重选择出牌算法进行验证，对算法在不同输入参数下选择出牌结果进行测试，对比期望结果与执行结果是否一致进行算法可用性和正确性的检验。根据贪婪选择出牌算法的实现流程图进行测试用例的设计，确保流程图中的每条路径都能被覆盖，保证测试用例的覆盖率，同时测试还包括系统中设计的每种出牌类型，测试用例及执行结果如表 6.1 所示：

表 6.1 贪婪选择出牌算法测试用例

| 算法名称      | 贪婪选择出牌算法  |                    |      |
|-----------|---|--------------------|------|
| 测试目的      | 贪婪选择出牌算法在不同参数下的可用性和正确性  |                    |      |
| 用例编号      | 参数(未使用的参数不列出)   | 期望结果               | 执行结果 |
| Greedy-1  | first=1,curCards=3444552222                                     | playCards=44455    | 通过   |
| Greedy-2  | first=0,curCards=3444552222,cards=3334,curId=1,upId=0           | playCards=3444     | 通过   |
| Greedy-3  | first=0,curCards=3444552222,cards=3334,curId=0,upId=0,upNum=5   | playCards=null     | 通过   |
| Greedy-4  | first=0,curCards=444552222,cards=3333,curId=0,upId=0,upNum=8    | playCards=null     | 通过   |
| Greedy-5  | first=0,curCards=444552222,cards=3QQQ,curId=0,upId=0,upNum=8    | playCards=null     | 通过   |
| Greedy-6  | first=0,curCards=22,cards=QQ,curId=0,upId=0,upNum=8             | playCards=22       | 通过   |
| Greedy-7  | first=0,curCards=3345679TJ,cards=5,curId=0,upId=0,upNum=7       | playCards=6        | 通过   |
| Greedy-8  | first=0,curCards=34999TTT22,cards=33344456,curId=1,upId=0       | playCards=34999TTT | 通过   |
| Greedy-9  | first=0,curCards=356789TQQ22,cards=345678,curId=0,upId=1        | playCards=56789T   | 通过   |
| Greedy-10 | first=0,curCards=3333JJLB,cards=456789TJ,curId=0,upId=0,upNum=8 | playCards=3333     | 通过   |
| Greedy-11 | first=0,curCards=3333JJLB,cards=7777,curId=1,upId=0             | playCards=LB       | 通过   |

根据权重选择出牌算法的实现流程图进行进行测试用例的设计，确保流程图中的每条路径都能被覆盖，保证测试用例的覆盖率，同时测试还包括系统中设计的每种出牌类型，测试用例及执行结果如表 6.2 所示：

表 6.2 权重选择出牌算法测试用例

| 算法名称     | 权重选择出牌算法                             |                    |      |
|----------|--------------------------------------|--------------------|------|
| 测试目的     | 权重选择出牌算法在不同参数下的可用性和正确性               |                    |      |
| 用例编号     | 参数(未使用的参数不列出)                        | 期望结果               | 执行结果 |
| Weight-1 | curCards=3444552222,cards=0          | playCards=44455    | 通过   |
| Weight-2 | curCards=3444552222,cards=3334       | playCards=3444     | 通过   |
| Weight-3 | curCards=3444552222,cards=3JJJ       | playCards=2222     | 通过   |
| Weight-4 | curCards=34445522,cards=3JJJ         | playCards=None     | 通过   |
| Weight-5 | curCards=34567899TTJJQQ,cards=345678 | playCards=456789   | 通过   |
| Weight-6 | curCards=8JJQQ22,cards=0             | playCards=8        | 通过   |
| Weight-7 | curCards=8JJQQ22,cards=0             | playCards=8        | 通过   |
| Weight-8 | curCards=8JJQQ22,cards=34777888      | playCards=8JJJJQQ2 | 通过   |
| Weight-9 | curCards=88TTTLB,cards=AAAA          | playCards=LB       | 通过   |

通过对算法测试结果的分析，证明本文所实现算法的正确性和可用性。因斗地主中出牌类型较多，与其对应的出牌选择可能性集合十分庞大，本次测试中对

每种出牌类型对应的一个实例进行测试，但是无法穷尽测试所有实例的可能性，因此尽管所有的测试结果都达到预期的结果，只能表明算法在测试的实例中都是正确且可以的，但是还是无法保证使用过程中遇到特殊的出牌可能会导致算法出现错误，导致返回的结果存在错误。

### 6.2.2 功能测试

系统功能测试是为了验证系统所完成的功能是否满足系统的需求和设计。依据系统划分的功能模块，对主要的功能模块下子功能中的功能点进行测试，测试的范围一般是系统详细设计的所有功能，在测试环境部署的系统上进行逐一测试，但在该系统中存在子功能的功能点重复的情况，故在测试过程中只选择其中一个子功能作为代表进行测试，并对其测试用例进行详细描述。

系统的登录注册功能编写的测试用例及执行结果如表 6.3 所示：

表 6.3 系统登录注册测试用例

|             |             |   |   |                             |      |
|-------------|-------------|---|---|-----------------------------|------|
| 功能名称        | 系统登录注册功能    |   |   |                             |      |
| 测试目的        | 测试用户注册、登录功能 |   |   |                             |      |
| 预置条件        | 无           |   |   |                             |      |
| 用例编号        | 目的          | 操作步骤  | 输入数据  | 期望结果                        | 执行结果 |
| LoginTest-1 | 测试用户注册功能    | 1.打开登录界面；<br>2.点击去注册；<br>3.输入注册信息；<br>4.点击注册。 | 用户名:UserTest；<br>密码:qwe123456。                | 1.系统提示用户注册成功；<br>2.返回登录界面。  | 通过   |
| LoginTest-2 | 测试用户登录功能    | 1.打开登录界面；<br>2.输入登录信息；<br>3.点击登录。             | 用户名:UserTest；<br>密码:qwe123456；<br>验证码:28535。  | 1.系统提示用户登录成功；<br>2.打开系统主界面。 | 通过   |
| LoginTest-3 | 测试用户登录功能    | 1.打开登录界面；<br>2.输入登录信息；<br>3.点击登录。             | 用户名:UserTest1；<br>密码:qwe123456；<br>验证码:28535。 | 1.系统提示用户名不存在。               | 通过   |
| LoginTest-4 | 测试用户登录功能    | 1.打开登录界面；<br>2.输入登录信息；<br>3.点击登录。             | 用户名:UserTest；<br>密码:qwe12345；<br>验证码:28535。   | 1.系统提示用户密码错误。               | 通过   |

系统设置模块主要是对数据进行基本的管理，其基础的管理功能一致，只有用户管理和角色管理中还包括对用户或角色进行权限设置的功能，故此以用户管理子功能作为代表功能进行测试，测试的功能包括数据的增删查改和权限设置，其测试用例及执行结果如表 6.4 所示：

表 6.4 用户管理测试用例

|            |                               |  |  |  |      |
|------------|-------------------------------|--|--|--|------|
| 功能名称       | 用户管理功能                        |  |  |  |      |
| 测试目的       | 测试用户数据分页查询、添加、修改、删除、详情及权限设置功能 |  |  |  |      |
| 预置条件       | 以管理员身份登录系统                    |  |  |  |      |
| 用例编号       | 目的                            | 操作步骤   | 输入数据   | 期望结果   | 执行结果 |
| UserTest-1 | 测试用户数据分页查询功能                  | 1.点击系统设置下的用户管理。                                | 无  | 1.打开用户管理界面；<br>2.正确显示用户信息；<br>3.可进行翻页查询操作。   | 通过   |
| UserTest-2 | 测试用户数据添加功能                    | 1.点击用户管理界面的添加；<br>2.输入对应的用户信息；<br>3.点击确定。      | 用户名：User1；<br>密码：qwe123；<br>确认密码：qwe123；<br>角色：默认角色。 | 1.页面信息可输入或选择；<br>2.用户数据正确添加。                 | 通过   |
| UserTest-3 | 测试用户数据修改功能                    | 1.点击用户管理界面的修改；<br>2.对需要的用户信息进行修改；<br>3.点击确定。   | 用户名：User1；<br>角色：默认角色；<br>电话号码：15761633212。          | 1.页面可修改信息能输入或选择；<br>2.用户数据正确修改。              | 通过   |
| UserTest-4 | 测试用户数据删除功能                    | 1.点击用户管理界面的删除；<br>2.点击确定。                      | 无  | 1.弹出数据删除确认框；<br>2.用户数据正确删除；<br>3.用户管理界面数据刷新。 | 通过   |
| UserTest-5 | 测试用户数据查看功能                    | 1.点击用户管理界面的查看。                                 | 无  | 1.页面正确打开；<br>2.显示查看的用户数据。                    | 通过   |
| UserTest-6 | 测试用户权限设置功能                    | 1.点击用户管理界面的权限设置；<br>2.勾选或取消对应的菜单权限；<br>3.点击确定。 | 无  | 1.菜单权限可勾选或取消；<br>2.用户权限正确设置。                 | 通过   |

说明：其他数据管理功能与用户管理功能类似，故不再一一进行描述。

战绩查询模块以锦标赛战绩查询子功能为例进行测试，因该功能在其他战绩查询功能的基础上还包括锦标赛的选择功能，同时对战绩排名功能进行测试，战绩查询测试用例及执行结果如表 6.5 所示：



表 6.5 战绩查询测试用例

|             |                            |                                    |                            |  |      |
|-------------|----------------------------|------------------------------------|----------------------------|--|------|
| 功能名称        | 战绩查询功能                     |                                    |                            |  |      |
| 测试目的        | 测试锦标赛战绩、战绩排名的查询功能及游戏详情查看功能 |                                    |                            |  |      |
| 预置条件        | 系统用户成功登录系统                 |                                    |                            |  |      |
| 用例编号        | 目的                         | 操作步骤                               | 输入数据                       | 期望结果   | 执行结果 |
| QueryTest-1 | 测试锦标赛战绩查询功能                | 1.点击战绩查询下的锦标赛战绩;<br>2.点击锦标赛名称进行选择。 | 无                          | 1.打开锦标赛战绩界面并显示锦标赛名称;<br>2.锦标赛名称可进行点击选择;<br>3.选择锦标赛,界面会刷新显示对应该锦标赛的对局信息;<br>4.对局信息的对局详情和对局回放按钮可点击。 | 通过   |
| QueryTest-2 | 测试对局详情列表查看功能               | 1.在战绩查询界面,点击对局详情。                  | 无                          | 1.打开对局详情界面并详细对应的信息;<br>2.使用淡绿色的文字标识当前用户的出牌信息。  | 通过   |
| QueryTest-3 | 测试对局回放功能                   | 1.在战绩查询界面,点击对局回放。                  | 无                          | 1.打开对局回放界面并显示操作按钮,操作按钮可点击;<br>2.正确渲染玩家的手牌和地主牌;<br>3.按照游戏中的出牌顺序进行循环模拟出牌。                          | 通过   |
| QueryTest-4 | 测试战绩排名功能                   | 1.点击战绩查询下的战绩排名。                    | 游戏类型: 所有模式;<br>排名方式: 综合排名。 | 1.打开排名界面;<br>2.正确渲染数据;<br>3.游戏类型、排名方式可进行选择,点击查询后界面会刷新显示结果。                                       | 通过   |

说明: 其他模式战绩查询与锦标赛战绩查询功能类似, 故不再一一进行描述。

游戏空间模块是系统的核心模块, 故对该模块下的所有模式斗地主游戏中的功能点进行测试, 针对所有模式的斗地主游戏中不同的功能点都进行测试用例的编写。所有模式的测试用例及执行结果如下所示:

表 6.6 单机模式测试用例

|               |                             |   |      |  |      |
|---------------|-----------------------------|---|------|--|------|
| 功能名称          | 单机模式斗地主功能                   |   |      |  |      |
| 测试目的          | 测试单机模式下难度等级选择、开始游戏、抢地主、出牌功能 |   |      |  |      |
| 预置条件          | 系统用户成功登录系统                  |   |      |  |      |
| 用例编号          | 目的                          | 操作步骤                                    | 输入数据 | 期望结果   | 执行结果 |
| OfflineTest-1 | 测试单机模式下的难度选择功能              | 1.点击游戏空间下的单机模式;<br>2.点击难度等级进行选择。        | 无    | 1.打开单机模式游戏界面;<br>2.难度等级可进行选择。                      | 通过   |
| OfflineTest-2 | 测试单机模式下的抢地主功能               | 1.在游戏界面可点击对应的分数按钮进行抢地主。                 | 无    | 1.游戏界面显示对应的分数按钮;<br>2.分数按钮可点击。                     | 通过   |
| OfflineTest-3 | 测试单机模式下的出牌功能                | 1.在游戏模式界面点击手牌进行牌选择;<br>2.点击不出或出牌进行出牌选择。 | 无    | 1.游戏界面显示出牌选择按钮;<br>2.手牌可以选择或取消选择;<br>3.出牌失败给出错误提示。 | 通过   |

表 6.7 在线模式测试用例

|              |                                   |                              |      |                                    |      |
|--------------|-----------------------------------|------------------------------|------|------------------------------------|------|
| 功能名称         | 在线模式斗地主功能                         |                              |      |                                    |      |
| 测试目的         | 测试在线模式斗地主的进入游戏、开始游戏、抢地主、出牌、结束提示功能 |                              |      |                                    |      |
| 预置条件         | 系统用户成功登录系统                        |                              |      |                                    |      |
| 用例编号         | 目的                                | 操作步骤                         | 输入数据 | 期望结果                               | 执行结果 |
| OnlineTest-1 | 测试在线模式进入游戏功能                      | 1.点击游戏空间下的在线模式;<br>2.点击进入游戏。 | 无    | 1.打开在线模式游戏桌面界面;<br>2.点击进入游戏打开游戏界面。 | 通过   |
| OnlineTest-2 | 测试在线模式开始游戏功能                      | 1.在游戏界面点击开始。                 | 无    | 1.游戏界面刷新,显示等待界面。                   | 通过   |
| OnlineTest-3 | 测试在线模式结束提示功能                      | 1.点击继续游戏或者离开休息。              | 无    | 1.继续游戏进入等待界面;<br>2.离开休息则回到游戏桌面界面。  | 通过   |

说明: 该模式下的抢地主、出牌功能与单机模式的功能一样, 故不再重复阐述。

通过对系统功能测试用例所执行过程的记录及执行结果的分析, 本文所完成的系统还存在少量的错误或不足, 存在的问题大部分在各种模式的斗地主游戏中, 由于在实现过程无法完全考虑到游戏过程中发生的情况而造成, 但在进行修改和

完善后，再次测试证明所有的功能已经完全符合系统设计的功能要求，满足预期的需求。

### 6.3 压力测试

软件压力测试是一种基本的质量保证行为，它是每个重要软件测试工作的一部分。本文在系统部署的服务器上安装压力测试软件 webbench 进行测试，测试的参数设置如表 6.9 所示：

表 6.9 参数设置表

| -t (运行测试 URL 的时间/秒) | -c (并发数) |
|---------------------|----------|
| 60                  | 100      |
| 60                  | 500      |
| 60                  | 1000     |
| 60                  | 2000     |
| 60                  | 3000     |
| 60                  | 4000     |
| 60                  | 5000     |

并发数为 100、500、1000 的压力测试结果如图 6.1 所示：

```
root@VM-0-10-ubuntu:~# webbench -t 60 -c 100 http://172.81.251.56/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Benchmarking: GET http://172.81.251.56/
100 clients, running 60 sec.

Speed=737 pages/min, 107572 bytes/sec.
Requests: 737 susceed, 0 failed.
root@VM-0-10-ubuntu:~# webbench -t 60 -c 500 http://172.81.251.56/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Benchmarking: GET http://172.81.251.56/
500 clients, running 60 sec.

Speed=662 pages/min, 107264 bytes/sec.
Requests: 662 susceed, 0 failed.
root@VM-0-10-ubuntu:~# webbench -t 60 -c 1000 http://172.81.251.56/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Benchmarking: GET http://172.81.251.56/
1000 clients, running 60 sec.

Speed=616 pages/min, 105967 bytes/sec.
Requests: 616 susceed, 0 failed.
```

图 6.1 并发数 100、500、1000 压力测试结果图

并发数为 2000、3000、4000、5000 的压力测试结果如图 6.2 所示：

```
root@VM-0-10-ubuntu:~# webbench -t 60 -c 2000 http://172.81.251.56/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Benchmarking: GET http://172.81.251.56/
2000 clients, running 60 sec.

Speed=619 pages/min, 105683 bytes/sec.
Requests: 609 succeed, 10 failed.
root@VM-0-10-ubuntu:~# webbench -t 60 -c 3000 http://172.81.251.56/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Benchmarking: GET http://172.81.251.56/
3000 clients, running 60 sec.

Speed=614 pages/min, 104377 bytes/sec.
Requests: 596 succeed, 18 failed.
root@VM-0-10-ubuntu:~# webbench -t 60 -c 4000 http://172.81.251.56/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Benchmarking: GET http://172.81.251.56/
4000 clients, running 60 sec.

Speed=616 pages/min, 104656 bytes/sec.
Requests: 610 succeed, 6 failed.
root@VM-0-10-ubuntu:~# webbench -t 60 -c 5000 http://172.81.251.56/
Webbench - Simple Web Benchmark 1.5
Copyright (c) Radim Kolar 1997-2004, GPL Open Source Software.

Benchmarking: GET http://172.81.251.56/
5000 clients, running 60 sec.
problems forking worker no. 4665
fork failed.: Resource temporarily unavailable
root@VM-0-10-ubuntu:~#
```

图 6.2 并发数 2000、3000、4000、5000 压力测试结果图

在图 6.1 和 6.2 中，测试输出的结果形式为 Speed=x pages/min, y bytes/sec，其中 x 的值表示每分钟响应的请求数；y 表示每秒传输的数据量。

通过对测试结果的分析，当最大并发数为 5000 时，系统无法承受此时的并发数量，可得本系统在当前部署环境下的最大并发数在 4000-5000 之间，经过再次测试系统的最大并发数为 4500；页面请求最大的失败率为 3%，表明系统在稳定性方面的性能是可接受的；每秒传输的数据都在 100kB 以上，完全可以满足本系统的数据传输要求；在并发数大于 1000 时每分钟响应的页面数量大约为 600 个，也就是说每个页面的相应时间平均为 100ms，可能是受到部署系统网络带宽的限制。系统承受的压力和服务器的 CPU、内存及带宽相关，在 CPU 和内存都

增加的情况下，系统的最大并发数会增加；在网络带宽增加的情况下，每个页面的响应时间也会降低，系统的整体性能也会得到相应的提升。

## 6.4 本章小结

本章主要是介绍测试系统的部署情况，包括部署的操作系统、服务器配置、软件环境等，测试该系统所使用的操作系统及浏览器情况；然后介绍算法测试和功能测试的测试用例，详细描述用例进行测试时的操作步骤并将测试系统所得的结果进行记录，对算法和系统存在的错误进行完善和修复；最后对系统进行压力测试，根据测试结果分析当前部署环境的下系统所能承受的最大压力。

## 第七章 总结与展望

### 7.1 总结

本文通过对智能算法对抗试验平台的设计、智能算法的研究、斗地主游戏的设计与开发，完成了开放斗地主平台的搭建。本平台使用 UML 对各个阶段进行分析和设计，在系统的需求分析阶段建立系统的用例图，在总体设计阶段建立系统的架构和功能模块结构图等，在详细设计阶段建立系统模块的包图、活动图等，使用实体类对系统的数据库设计进行描述，并给出对应的实体属性及其物理存储类型。

对本文的主要研究成果为以下几点：

第一，为智能斗地主算法研究人员提供算法对抗试验平台，算法研究人员在平台中上传算法来与其他智能算法进行对抗试验，可对试验的数据进行查询、统计和导出，以便对试验结果进行分析。

第二，设计并实现了贪婪选择出牌算法和权重选择出牌算法，并对 MCTS 算法的应用进行设计和实现，并且对算法的正确性进行测试。在此基础之上，进行算法对比试验的设计，根据试验结果对三个算法的智能程度进行排序，完成单机模式中的难度等级与算法的对应功能。

第三，设计并实现了开放斗地主游戏平台，在这个过程中使用 UML 中的对系统的各个阶段进行设计；采用 Django 框架作为 WEB 系统的开发框架，Python 作为系统的开发语言进行系统的开发；在通讯功能实现方面选用 Django Channels 作为关键技术；使用 MySQL 结合 Django 数据模型进行数据库的建立，对系统中的数据进行结构化存储；使用高德地图、Echarts 进行数据多样化的展示；对平台进行功能测试和压力测试。

第四，构建数据集，对系统中各种模式的游戏数据进行存储，按照一定的标准对无效数据进行剔除及有效数据的选择和转换，根据智能算法研究人员的需求进行试验数据集的生成，为智能算法的研究提供试验数据集获取的便利。

## 7.2 展望

本文在对三种智能算法研究的基础之上设计并实现了五种模式的斗地主游戏，构建了基于 WEB 的斗地主游戏平台。但是还存在很多方面的不足：

第一，智能算法动态调用接口。在智能算法对抗试验平台目前支持动态调用的算法是使用 Python 实现的算法，上传的文件为.py 文件，在试验中通过对选择的算法进行判断来调用相应的接口。对于使用 TensorFlow、PyTorch、C 等进行智能斗地主算法的研究人员来说是极其不方便的，应该支持多种算法的动态调用，为更多的研究人员提供智能算法对抗试验的便利。

第二，算法智能方面。算法目前能满足斗地主游戏中的出牌提示功能，但是在设计的过程中缺少与其他算法的对比，无法估计其智能程度，但从试验的结果来看，本文的贪婪选择出牌算法和权重选择出牌算法在智能程度方面还需进一步的提高，使其在斗地主游戏中具有更好的表现。

第三，玩家交流方面。应该在考虑满足游戏需求的同时也要满足社交需求，玩家们可以互相把对方加为好友或者群友，可满足玩家们在进行游戏的同时也可进行斗地主技术的交流与分享。

第四，界面设计方面。在界面设计方面还要进一步加强设计，目前各个模式斗地主游戏的界面统一，尚未突显出该游戏模式的特点。

第五，模式新颖方面。需要寻求更多新颖的斗地主游戏，给玩家们带来不一样的斗地主游戏体验，让人们斗地主游戏有一个新的认识。

以上所述的不足之处，将在以后的研究过程中进行进一步的考虑，使本文研究成果更加完善，使整个系统变得更加智能化、人性化和便捷化。

## 致谢

在我三年的研究生的生活、学习中，我得到了父母、导师、师兄师姐、同学、室友、朋友等很多人的帮助，我的内心对这些人充满了感激之情，在此对他们表示衷心的感谢和祝福。

首先我要感谢我的父母，您们在这三年中给予我生活上很大的帮助和精神上的鼓励，是我在这三年学习中不可缺少的支撑力量。特别是我父亲教会了我很多做人的道理，做事的方法，让我在遇到困难和挫折时，始终都能找到解决它们的方法。

然后我要感谢我的导师李丹老师、王以松老师及给我授课的老师，我的导师和王以松老师在我的学业和论文完成过程中给我极大的帮助，帮助我制定合理的学习规划和解决学习过程中的疑难问题，在论文方面多次对我的论文进行指导修改，使我的论文能够按时完成。

其次我要感谢我的师兄师姐、同学、室友，特别是于小明博士师兄，为我的毕业设计提供诸多思路和想法，使我的设计能够保质保量的完成，同时还认真负责的帮我修改论文，指出我论文中存在的不足。还有冯仁艳博士师姐帮助我进行开题报告的修改，使我能够顺利的进行开题，也给本论文的完成奠定了良好的基础。也同样感谢其他师兄师姐们给出帮助，与他们的交流过程是我受益匪浅，从中给我完成设计和论文带来很多的启发。

最后我要感谢我实习单位的领导们及认识的同事们，他们在我的实习工作中给予我极大的帮助和包容。感谢领导们对我生活和学习的关心、工作的包容，感谢同事们对我的实习工作的帮助，耐心的教我熟悉工作流程，熟悉工作的开展步骤，一步一步的带我进行工作，保证我在团队中不会拖后腿。使我在实习工作中积累大量的工作经验，增强团结合作意识，提高自身的能力。

没有这么多的人对我的关爱和帮助，我是无法完成研究生期间的学业和论文的，所以我在这里再一次对我的父母、导师、师兄师姐等帮助过我的人表示衷心的感谢和祝福。



## 参考文献

- [1] 中国地方特色棋牌游戏行业白皮书[C].艾瑞咨询系列研究报告(2017 年第 2 期):上海艾瑞市场咨询有限公司,2017:88-137.
- [2] 赵娟,弋改珍.斗地主游戏的设计与实现[J].现代信息科技,2018,2(11):81-82+85.
- [3] 饶子建.手机游戏斗地主的设计与客户端实现[D].厦门大学,2017.
- [4] 张焕甫.“互乐棋牌”游戏大厅及“斗地主”子系统的设计与实现[D].北京交通大学,2016.
- [5] 周进森.基于 Android 平台的单机版斗地主设计与实现[J].福建电脑,2016,32(11):130+129.
- [6] 李竹林.基于 Android 系统的斗地主游戏的设计与实现[J].河南科学,2015,33(02):200-203.
- [7] 曹庆皇,杨晓琴.用 J2ME 开发手机网络游戏——斗地主[J].电脑编程技巧与维护,2008(09):56-69.
- [8] 翟飞.基于规则引擎的纸牌类游戏开发的应用研究[D].厦门大学,2016.
- [9] 吕宝生,钟宝荣.网络棋牌之鞍山麻将[J].电脑知识与技术,2011,704:878-879.
- [10] 包文祥.基于 WebSocket 协议的实时网页通信的研究与实现[D].江苏科技大学,2018.
- [11] 刘书国.面向中小学教育的在线实时答疑系统的设计与实现[D].上海交通大学,2018.
- [12] 高聪.信息安全技术在 WebSocket 实时通信的应用研究[D].华北电力大学,2016.
- [13] 卢振.基于 WebSocket 协议的分布式云推送平台研究与实现[D].大连理工大学,2019.
- [14] 包文祥.基于 WebSocket 协议的实时网页通信的研究与实现[D].江苏科技大学,2018.
- [15] 丁伟.基于安卓的棋牌游戏系统的关键技术研究实现[D].上海交通大学,2017.
- [16] 潘正辉.基于 Android 系统手机游戏的设计与开发研究[J].电子世界,2016(13):128+130.

- [17]王鹏.网络游戏服务器算法及架构设计[D].重庆交通大学,2014.
- [18]孙魁伟.基于贪婪算法的自动排课系统设计与实现[D].大连理工大学,2013.
- [19]彭啟文,王以松,于小民,等.基于手牌拆分的“斗地主”蒙特卡洛树搜索[J].南京师大学报(自然科学版),2019,42(03):107-114.
- [20]李翔,姜晓红,陈英芝,等.基于手牌预测的多人无限注德州扑克博弈方法[J].计算机学报,2018,41(01):47-64.
- [21]刘元伟.基于 Django 的翻译协作和共享平台的设计与实现[D].北京交通大学,2019.
- [22]郑建国.基于 UML 的贵州公路集团账务处理系统的分析与设计[D].云南大学,2012.
- [23]牛宁.基于 Django 的智慧园区平台系统设计与实现[D].电子科技大学,2018.
- [24]白相辰.基于 Django 框架的 Web 在线教育平台的设计与实现[D].北京交通大学,2019.
- [25]孟亚茹.农业云平台下基于环境的蔬菜生长周期预测模型的研究与应用[D].贵州大学,2019.
- [26]方鹏.基于 Django 的海量媒体数据分析平台的设计与实现[D].广西大学,2017.
- [27]齐伟.跟老齐学 Python: Django 实战(第2版)[M].北京:电子工业出版社,2019.
- [28]安东尼奥·米勒.Django 项目实例精解(第2版)[M].李伟.北京:清华大学出版社,2019.
- [29]黄永详.玩转 Django 2.0 [M].北京:清华大学出版社,2018.
- [30]于娇.证券公司反洗钱监控系统的设计与实现[D].厦门大学,2018.
- [31]Baron Schwartz,Peter Zaitsev,Vadim Tkachenko.高性能 MySQL(第3版)[M].宁海元,周振兴,彭立勋等.北京:电子工业出版社,2013.
- [32]刘书国.面向中小学教育的在线实时答疑系统的设计与实现[D].上海交通大学,2018.
- [33]刘桐.网络游戏用户需求研究[D].北京邮电大学,2019.
- [34]廖长江.基于 J2EE 的新疆建设局档案管理系统的设计与实现[D].厦门大学,2017.

- [35]王万银.基于 J2EE 的电力用户用电信息采集系统的设计与实现[D].电子科技大学,2017.
- [36]陈婷婷.基于某企业的人力资源管理系统的设计与实现[D].天津大学,2017.
- [37]张德武.基于工作流的政府办公自动化管理系统的设计与实现[D].电子科技大学,2013.
- [38]李鹏飞.基于 Web 技术的校园论坛设计与实现[D].内蒙古科技大学,2019.
- [39]姚智超.基于 B/S 模式的真空设备远程监控系统的开发[D].兰州理工大学,2017.
- [40]梁浩.基于服务器推送技术的在线交流平台的研究与实现[D].燕山大学,2015.
- [41]苗勃,吴力夫,王羽,施娟.网络棋牌游戏平台通用架构与关键技术[J].广播电视信息,2010,03:39-43.
- [42]易仁伟.基于 WebSocket 的实时 Web 应用的研究[D].武汉理工大学,2013.
- [43]王舒洋.基于 REST 架构的企业安全平台日志报表子系统的设计与实现[D].西安电子科技大学,2019.
- [44]王凯,李军,刘迪.跨平台网络扑克游戏设计与实现[J].科技信息,2013(04):316-317.
- [45]陈渊博.学位论文格式检测系统设计与实现[D].大连理工大学,2017.
- [46]胡开亮.基于状态抽象和残局解算的二人非限制性德州扑克策略的研究[D].哈尔滨工业大学,2017.
- [47]李昌.基于 Q 学习算法的非完备信息机器博弈的研究[D].哈尔滨工业大学,2015.
- [48]庄琦赋.基于 Android 移动终端的人机博弈游戏的研究[D].山东大学,2013.
- [49]周钱.基于云平台的客户关系管理系统的设计与实现[D].湖南大学,2015.
- [50]葛钰.面向云服务的 web 服务器负载均衡设计与实现[D].西南交通大学,2019.
- [51]Chaslot G, Bakkes E, Szita I. Monte-Carlo Tree Search: A New Framework for Game AI [C]. In Proceedings of AIIDE-08, 2008, 4(2):216-217.
- [52]Jonathan Rubin, Ian Watson. Computer poker: A review [J]. Artificial Intelligence, 2010, 175(5):958-987.
- [53]Qiqi Jiang, Kuangzheng Li, Boyao Du, et al. DeltaDou: Expert-level Doudizhu AI through Self-play[C]. IJCAI 2019: 1265-1271.

- [54]Brown N, Sandholm T.Superhuman AI for heads-up no-limit poker: Libratus beats top professionals [J]. Science, 2018, 359(6374):418-424.
- [55]Heinrich J, Silver D. Smooth UCT search in computer poker[C]// International Conference on Artificial Intelligence. IJCAI 2015:554-560.

## 图版

|                         |    |
|-------------------------|----|
| 图 3.1 系统总体用例图.....      | 9  |
| 图 3.2 系统设置用例图.....      | 10 |
| 图 3.3 游戏空间用例图.....      | 12 |
| 图 3.4 算法对抗用例图.....      | 14 |
| 图 3.5 战绩查询用例图.....      | 15 |
| 图 3.6 信息查询用例图.....      | 17 |
| 图 3.7 个人中心用例图.....      | 18 |
| 图 4.1 系统体系结构图.....      | 20 |
| 图 4.2 系统架构图.....        | 21 |
| 图 4.3 系统功能架构图.....      | 23 |
| 图 4.4 数据管理活动图.....      | 24 |
| 图 4.5 用户管理流程图.....      | 25 |
| 图 4.6 单机模式活动图.....      | 27 |
| 图 4.7 在线模式活动图.....      | 28 |
| 图 4.8 好友模式活动图.....      | 28 |
| 图 4.9 锦标赛模式活动图.....     | 30 |
| 图 4.10 在线模式流程图.....     | 31 |
| 图 4.11 残局模式活动图.....     | 31 |
| 图 4.12 对抗试验活动图.....     | 32 |
| 图 4.13 残局战绩查询活动图.....   | 33 |
| 图 4.14 锦标赛战绩查询活动图.....  | 34 |
| 图 4.15 锦标赛战绩流程图.....    | 34 |
| 图 4.16 积分查询活动图.....     | 35 |
| 图 4.17 头像修改活动图.....     | 36 |
| 图 4.18 密保设置活动图.....     | 37 |
| 图 4.19 密码重置活动图.....     | 37 |
| 图 4.20 系统基础数据实体类图.....  | 38 |
| 图 4.21 游戏数据实体类图.....    | 39 |
| 图 4.22 算法试验实体类图.....    | 40 |
| 图 4.23 贪婪选择出牌算法伪代码..... | 48 |
| 图 4.24 权重选择出牌算法伪代码..... | 49 |
| 图 5.1 贪婪选择出牌算法流程图.....  | 53 |
| 图 5.2 权重选择出牌算法流程图.....  | 55 |
| 图 5.3 用户登录界面图.....      | 59 |
| 图 5.4 用户数据查询界面图.....    | 60 |
| 图 5.5 用户权限设置界面图.....    | 60 |
| 图 5.6 单机模式界面图.....      | 61 |
| 图 5.7 在线模式抢地主界面图.....   | 62 |
| 图 5.8 在线模式出牌界面图.....    | 62 |
| 图 5.9 在线玩家信息界面图.....    | 63 |
| 图 5.10 好友模式出牌界面图.....   | 64 |

|  |    |
|--|----|
| 图 5.11 锦标赛界面图.....                         | 65 |
| 图 5.12 锦标赛模式出牌界面图.....                     | 65 |
| 图 5.13 关卡挑战界面图.....                        | 66 |
| 图 5.14 出牌函数实例.....                         | 67 |
| 图 5.15 算法对抗试验界面图.....                      | 68 |
| 图 5.16 对局回放界面图.....                        | 69 |
| 图 5.17 战绩排名界面图.....                        | 70 |
| 图 6.1 并发数 100、500、1000 压力测试结果图.....        | 77 |
| 图 6.2 并发数 2000、3000、4000、5000 压力测试结果图..... | 78 |

## 表版

|  |    |
|--|----|
| 表 3.1 编辑用例描述.....                      | 11 |
| 表 3.2 权限设置用例描述.....                    | 11 |
| 表 3.3 出牌用例描述.....                      | 13 |
| 表 3.4 报名用例描述.....                      | 13 |
| 表 3.5 对抗试验用例描述.....                    | 14 |
| 表 3.6 锦标赛战绩用例描述.....                   | 16 |
| 表 3.7 对局回放用例描述.....                    | 16 |
| 表 3.8 玩家积分用例描述.....                    | 17 |
| 表 3.9 密保设置用例描述.....                    | 18 |
| 表 4.1 数字表示及显示符号表.....                  | 25 |
| 表 4.2 出牌类型及其对应表.....                   | 26 |
| 表 4.3 用户表(User).....                   | 41 |
| 表 4.4 用户权限表(SimpleUserAuthority) ..... | 41 |
| 表 4.5 登录信息表(Login).....                | 41 |
| 表 4.6 密保问题表(PwdQuestions) .....        | 41 |
| 表 4.7 用户密保表(UserQuestions) .....       | 42 |
| 表 4.8 角色表(Role).....                   | 42 |
| 表 4.9 角色权限表(SimpleRoleAuthority).....  | 42 |
| 表 4.10 菜单表(Menu).....                  | 43 |
| 表 4.11 游戏出牌数据表(DdzGamePlays) .....     | 43 |
| 表 4.12 游戏数据表(DdzGame) .....            | 44 |
| 表 4.13 游戏积分表(GameScore).....           | 44 |
| 表 4.14 锦标赛数据表(Match).....              | 45 |
| 表 4.15 锦标赛报名表(MatchSign) .....         | 45 |
| 表 4.16 残局数据表(EndGame) .....            | 45 |
| 表 4.17 残局挑战表(EndRecord).....           | 46 |
| 表 4.18 挑战出牌表(EndRecordPlays).....      | 46 |
| 表 4.19 算法数据表(AialgorithmData).....     | 46 |
| 表 4.20 试验记录表(AialgorithmData).....     | 47 |
| 表 4.21 牌面值与权重值对应表.....                 | 50 |
| 表 4.22 出牌类型权重值对应表.....                 | 50 |
| 表 5.1 系统对外提供参数表.....                   | 67 |
| 表 6.1 贪婪选择出牌算法测试用例 .....               | 72 |
| 表 6.2 权重选择出牌算法测试用例.....                | 72 |
| 表 6.3 系统登录注册测试用例.....                  | 73 |
| 表 6.4 用户管理测试用例.....                    | 74 |
| 表 6.5 战绩查询测试用例.....                    | 75 |
| 表 6.6 单机模式测试用例.....                    | 76 |
| 表 6.7 在线模式测试用例.....                    | 76 |
| 表 6.9 参数设置表.....                       | 77 |

