密级：公开

基于分布式架构的五子棋微信小游戏的设计与实现

**Design and Implementation of Gobang WeChat Mini Game Based on Distributed Architecture**

|  |  |
| --- | --- |
| **学 院：** | 软件学院 |
| **学 号：** | 151203414 |
| **专业班级：** | 软件工程1504班 |
| **学生姓名：** | 刘昕禹 |
| **指导教师：** | 邵中 |

2019年 6月

摘 要

网络游戏是互联网经济的重要支柱产业，随着基础通信网络与移动智能设备性能的不断提高，面向智能手机的网络游戏已成为该产业新的产品发展方向。微信小游戏平台为广大Web开发者提供了一种轻应用开发模式，并成为今后网络游戏开发一种重要的技术选型。

以五子棋为游戏题材，综合运用Web前端开发技术、人机对弈算法、分布式服务器架构等技术方法，开发了一款以微信小游戏形式发布的五子棋游戏软件，实践验证了一种基于分布式体系结构的对弈类网络游戏设计方案。游戏客户端自主设计了基于HTML5 Canvas元素的轻量级动画绘制及交互渲染引擎，该引擎以抽象接口定义游戏场景中图形元素的基础行为，实现对场景元素的对象化封装，集成了关于场景重绘、元素拾取、屏幕尺寸自适应等基础功能。人机对弈的AI算法利用极大极小值搜索形成博弈树，并通过Alpha-Beta剪枝、启发式搜索、棋面评分缓存等技术手段优化落子选择的计算性能。服务器端采用分布式部署结构，由1台应用服务器统一处理游戏基本业务请求和大并发任务的负载分配，由多台游戏服务器构建服务集群，处理游戏过程中的通信控制、胜负判定、断线恢复等任务，设计并采用了一种服务器集群动态扩展的技术方案和工作机制，可实现对服务器数量的弹性调整，兼顾了游戏运营成本和大并发任务的处理性能。

该五子棋游戏基于微信用户身份登录，提供了人-机、人-人两种对弈模式，实现了好友对战、对等棋力对手的随机匹配、落子提示、悔棋、断线恢复等功能，可在游戏过程中与对方棋手进行聊天交流。该游戏已通过微信小游戏平台正式发布，累计用户访问数达96余万。实际应用效果表明游戏功能完整，运行稳定，所采用的AI算法与同类游戏软件和专业训练软件对战胜率分别达到93%和43%。

**关键词：**微信小游戏；网络游戏；五子棋AI；分布式架构

**Abstract**

Online game is an important pillar industry of the Internet economy. With the continuous improvement of the performance of basic communication networks and mobile intelligent devices, online games for smart phones have become the new product development direction of the industry. WeChat small game platform provides a light application development model for the majority of Web developers, and has become an important technology selection for online game development in the future.

With the game theme of Gobang, using the Web front-end development technology, human-machine game algorithm, distributed server architecture and other technical methods, a game of Gobang game released in the form of WeChat game was designed, and a distributed system was verified. The practice verifies a game-based network game design scheme based on distributed architecture. The game client has designed a lightweight animation rendering and interactive rendering engine based on HTML5 Canvas element. The engine defines the basic behavior of graphic elements in the game scene with an abstract interface, realizes object-oriented encapsulation of scene elements, and integrates the scene. Basic functions such as drawing, element picking, and screen size adaptation. The AI algorithm of man-machine game uses the maximal minimum value search to form the game tree, and optimizes the computational performance of the drop selection through Alpha-Beta pruning, heuristic search, and chess face scoring buffer. The server side adopts a distributed deployment structure, and one application server uniformly processes the load distribution of the game basic service request and the large concurrent task, and the service cluster is constructed by multiple game servers, and the communication control, the winning and losing judgment, and the disconnection in the process of the game are processed. Recovery and other tasks, designed and adopted a technical solution and working mechanism of server cluster dynamic expansion, can achieve flexible adjustment of the number of servers, taking into account the game operating costs and processing performance of large concurrent tasks.

The Gobang game is based on the WeChat user identity and provides two game modes: human-machine and person-to-person. It realizes the functions of friend match, random match of opponents, opponents, repentance, disconnection recovery, etc. In the process, chat with the opponent player. The game has been officially released through the WeChat mini-game platform. As of June 1, 19, the cumulative number of user visits reached more than 96 million. The actual application results show that the game function is complete and the operation is stable. The AI ​​algorithm used in the non-professional man-machine battle wins 93%, and the professional Gobang win rate reaches 43%.

**Keywords:** WeChat Mini games; Online games; Gobang AI; Distributed architecture

目 录

[摘 要 I](#_Toc11679200)

[Abstract II](#_Toc11679201)

[第 1 章 引 言 1](#_Toc11679202)

[1.1 课题的提出 1](#_Toc11679203)

[1.2 研究现状 1](#_Toc11679204)

[1.2.1 HTML5Canvas元素动画生成与交互研究 2](#_Toc11679205)

[1.2.2 在线游戏负载优化研究 3](#_Toc11679206)

[1.2.3 五子棋AI算法研究 4](#_Toc11679207)

[1.3 研究内容及主要工作 4](#_Toc11679208)

[1.4 开发环境与实验条件 5](#_Toc11679209)

[第 2 章 相关理论及技术分析 6](#_Toc11679210)

[2.1 Java平台支持 6](#_Toc11679211)

[2.1.1 网络通信支持 6](#_Toc11679212)

[2.1.2 多线程技术 7](#_Toc11679213)

[2.2 HTML5特性 7](#_Toc11679214)

[2.2.1 Canvas元素 7](#_Toc11679215)

[2.2.2 Web Socket 8](#_Toc11679216)

[2.3 五子棋AI算法 8](#_Toc11679217)

[2.3.1 基本术语 8](#_Toc11679218)

[2.3.2 棋面评估 9](#_Toc11679219)

[2.3.3 极大极小值搜索算法 9](#_Toc11679220)

[2.3.4 Alpha-Beta剪枝算法 10](#_Toc11679221)

[2.3.5 启发式搜索 10](#_Toc11679222)

[第 3 章 系统分析与设计 11](#_Toc11679223)

[3.1 需求分析 11](#_Toc11679224)

[3.1.1 功能定义 11](#_Toc11679225)

[3.1.2 游戏流程分析 12](#_Toc11679226)

[3.2 系统设计 13](#_Toc11679227)

[3.2.1 总体结构设计 14](#_Toc11679228)

[3.2.2 系统工作机制 16](#_Toc11679229)

[3.3 数据库设计 17](#_Toc11679230)

[第 4 章 五子棋AI算法设计 23](#_Toc11679231)

[4.1 算法描述与程序实现 23](#_Toc11679232)

[4.1.1 棋面评估 23](#_Toc11679233)

[4.1.2 极大极小值搜索 25](#_Toc11679234)

[4.1.3 Alpha-Beta剪枝优化 26](#_Toc11679235)

[4.1.4 启发式搜索 29](#_Toc11679236)

[4.1.5 评估缓存优化 29](#_Toc11679237)

[4.2 实际测试 31](#_Toc11679238)

[4.2.1 落子计算性能测试 31](#_Toc11679239)

[4.2.2 AI棋力测试 31](#_Toc11679240)

[第 5 章 游戏系统的详细设计与实现 32](#_Toc11679241)

[5.1 游戏客户端 32](#_Toc11679242)

[5.1.1 HTML5Canvas游戏引擎 32](#_Toc11679243)

[5.1.2 客户端游戏逻辑处理模块 35](#_Toc11679244)

[5.1.3 断线场景恢复 37](#_Toc11679245)

[5.2 社区服务 38](#_Toc11679246)

[5.3 游戏服务器设计 40](#_Toc11679247)

[5.3.1 开机认证 40](#_Toc11679248)

[5.3.2 建立Web Socket连接 41](#_Toc11679249)

[5.3.3 数据上报 42](#_Toc11679250)

[5.4 实际应用情况 43](#_Toc11679251)

[第 6 章 结 论 44](#_Toc11679252)

[参考文献 45](#_Toc11679253)

[在学研究成果 46](#_Toc11679254)

[致 谢 47](#_Toc11679255)

1. 引 言

## 课题的提出

近年来电子游戏产业高速发展，并随着移动设备的普及和硬件性能的提升，游戏玩家持续增多，网络游戏已经成为人们日常生活一种重要的休闲娱乐方式。具不完全统计，2018年我国网络游戏市场销售收入已经达到1948亿元，其中游戏类规模保持领先，约138万款，可以看出我国的网络游戏行业是一个蓬勃发展的行业，越来越多的人也投入到游戏开发的事业中。网络游戏设计与普通应用相比具备一定差异，其设计要求较高，同时根据游戏素材和需求不同，还将涉猎其他领域的知识。

2014年，万维网联盟宣布了HTML5标准，主流浏览器都在逐步完善并支持该标准，该标准提出了Canvas元素的绘图JavaScript接口，使得网页上可以制作复杂的动画内容，但是在使用细节上，各大厂家的浏览器还是存在细微的差别。2017年12月28日，微信平台发布了专门为游戏设计的生态产品——微信小游戏，该平台的发布首要解决了移动设备对HTML5标准支持的统一性，同时该平台发布的游戏不需要如同原生APP进行下载安装才可以使用，具备较好的便捷性。微信小游戏是基于HTML5标准衍生而来，可以实现在Android平台与iOS平台几乎相近的运行，极大降低开发者的开发成本。截止2018年末阿拉丁数据统计，微信小游戏市场规模达到60亿元，日活跃用户超过1亿，已经成为被大众广泛接受的新生态。

本课题完成一款以五子棋为题材的微信小游戏应用，实现在微信小游戏平台下标准的游戏产品设计。该应用支持单人游戏，玩家将与一个实力较强的AI进行对弈，同时也支持双人联机游戏，可以与自己的微信好友或是陌生人进行在线对抗。作品同时还配套了完善的社区功能，为用户提供相应的赛后交流服务。在应用架构方面，服务器后台采用分布式架构设计，支持游戏服务器数量的的弹性控制，以适应不同时期的用户需求，降低运营成本。以此满足一款网络游戏的基本运营要求，顺应当前以轻应用为潮流的移动应用开发潮流，具有一定的实用价值。

## 研究现状

网络游戏在近几十年中的迅速发展，已经吸引大量相关研究人员对网络游戏相关技术进行研究，同时五子棋AI题材也作为一种典型的AI算法，也有较多的同类研究文献，结合相关的应用领域、开发目标及技术选型，对相似性研究工作进行了调研整理。当前国内外研究现状主要集中于如下三个方面。

### HTML5Canvas元素动画生成与交互研究

在HTML5标准提出前，在Web中实现交互式动画是比较困难的，通常需要借助诸如Flash插件完成，相比于Flash，HTML5动画技术有通用的网络标准、跨平台、响应式网页设计、即时更新的Web客户端等优点[1]。

在Web APP中2D动画渲染主要基于Canvas的内置API实现[2],该方向的主要研究问题在于动画数据的信息结构、生成方法和渲染处理技术[3,4]。赵剑给出了一种HTML5标准的动画引擎设计方案，该引擎可解析自定义的动画文件结构，整合应用Canvas和SVG API进行播放出来，同时设计了动画模式适配算法以改进渲染性能[5]。王妍设计了动画自动生成系统的实验原型[6],并重点解决了在动画生成过程中的动作规划和路线规划问题。

此外，贾赛赛给出了一种HTML5的游戏引擎的开发方案，图1-1给出对应的示意图。

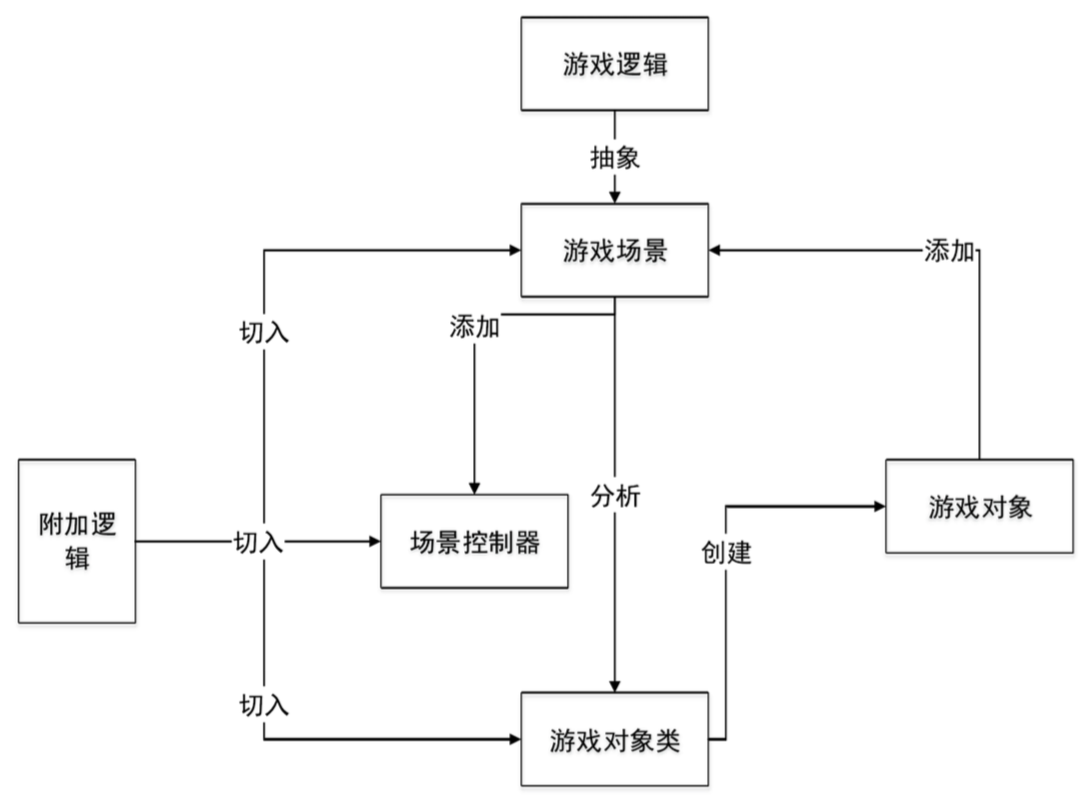


图1-1 HTML5游戏引擎开发方案

### 在线游戏负载优化研究

联机游戏对网络通讯需求要求较高，提供良好的服务性能，保证网络游戏的实时性、顺畅性是考验一款网络游戏的关键环节。而目前多数游戏均采用TCP/IP协议实现客户机与服务器的实时通信，为此王鹏等人提出一种以网关服务器、消息路由服务器、登录服务器、游戏服务器、数据库服务器五个子系统构建的游戏服务器集群基础架构[7-9]，其中网关服务器负责保持与客户端之间的TCP长连接，承担客户端与服务器组之间的消息中转功能，云淼给出一种服务器架构，其结构示意图如图1-2。

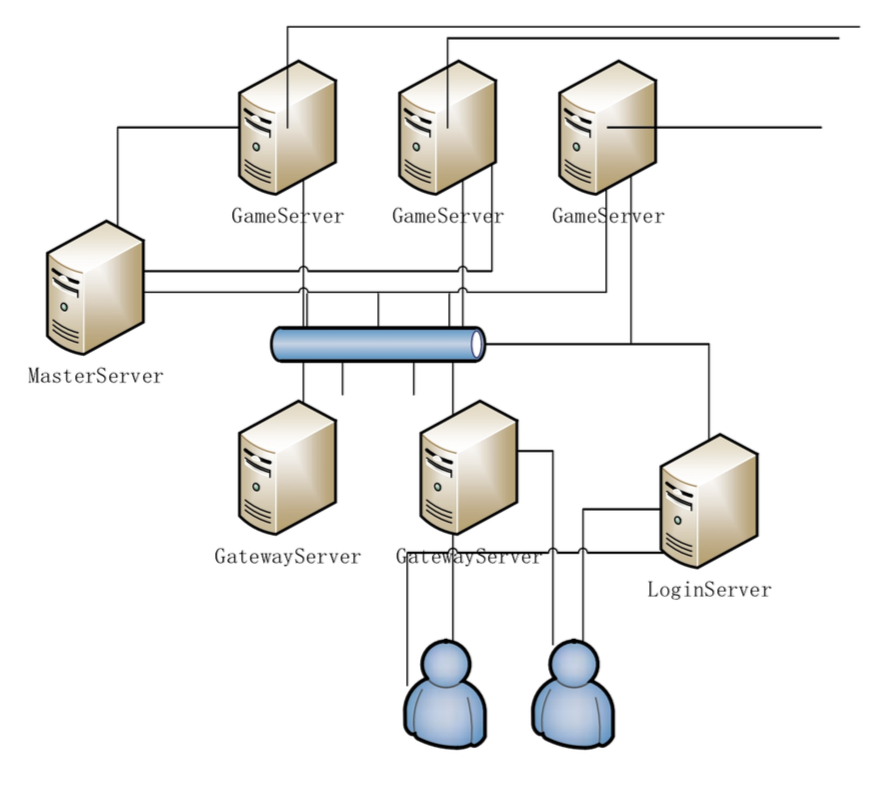


图1-2 服务器架构图

存在网管服务器的好处有：客户端程序直接与网管服务器通信，无需知道具体的游戏服务器内部架构；网关将内外网隔离，一定程度上保障了内网服务器的安全；使用多个网关分散负载，即使其中某个网管服务器宕机，在客户端做一些处理可切换网管，由新的网关中转此前的业务数据，实现无缝切换等能力。

### 五子棋AI算法研究

五子棋AI算法有许多种，目前普遍使用贪心算法与博弈树搜索算法[10]。

（1）贪心算法

贪心算法的主要思想是在当前局面中，找出主观认为最好的落点，然后落子。这种贪心算法的实现，最重要的是设计一种评估函数，该函数可以评估一个点位的优劣情况，刘瑞给出了一种可能的实现[11]。

对于棋盘中的每个位置，考虑未来一步棋子放到该位置后，它与周围的棋子连成什么样的棋形。总结的说对于棋盘中的一个位置(x,y)，考虑横、竖、斜和反斜四个方向，(x,y)最多属于20个相连5个棋子的成员，使之产生1个、2个、3个直至5个己方颜色棋子相连情况，赋予0，10，30，100，500，10000不同的得分，全盘扫描空子点位后选择得分最高点位作为AI的落子点。

该方法思想直观，实现简单，但该算法有一定局限性，只考虑一步的利益，而实战中人类往往会向后联想多步，因此棋力相对较弱。

（2）博弈树搜索

下棋双方在棋盘轮流下棋，局面就会从一种状态转移到另一种状态，把每次下棋的局面作为一个节点，将新生成的子状态与生成前的状态用相互连接，继续递归，直到下棋结束，将生成一棵树，为博弈树[12]。

对于一场对局而言，从一个局面生成一棵完整的博弈树这是不现实的，由于完整博弈树非常庞大，目前的计算机无法以足够快以及足够大的空间进行运算与存储。姜勇提到完整的博弈树之所以庞大存在于两个因素：一个是每个局面有很多可行走法，节点度数过大；二是下棋的步数太多，也就意味数的深度太深。针对此类问题，目前通用的方案是首先从众多可供选择的行动方案中选出对自己有益的选择，因此对于五子棋而言，目前提出的基于极大极小树搜索（Min-Max Tree Search）实现五子棋最优解[13]。张聪品总结了在极大极小树搜索算法中利用启发式搜索、Alpha-Beta剪枝的优化方案[14]，利用Alpha-Beta剪枝技术，把生成后继和倒推值估计结合起来，即时剪掉一些无用的分枝，以此提高搜索效率，并给出基于择序、小窗口搜索、信息重用等方案的改进思路[15,16]。

## 研究内容及主要工作

课题的主要目标是在微信小程序平台设计实现一套完整的五子棋网络游戏作品，可以投入实际运营。对于游戏客户端而言，实现游戏画面的渲染、用户交互，允许玩家进行人机对弈，提供一种棋力较强的AI进行对抗，同时允许用户与微信好友、陌生人进行匹配对弈，游戏中体验良好，系统可靠性较高；此外还将提供游戏社区服务，用户可以获得游戏相关资料，查阅自己曾经对弈历史记录、游戏复盘，与游戏内好友进行站内私信；对于游戏服务器而言，具备良好的弹性增减特性，以保证不同时期的用户需求，降低运营成本，满足市场需求。

游戏客户端以微信小游戏平台为载体，使用HTML5标准Javascript语言完成客户端设计，对于信息服务使用PHP脚本语言，游戏服务器使用Java语言完成设计。旨在带领读者了解小游戏的开发基本原理，和总结游戏实战应用时遇到问题的解决方案。

## 开发环境与实验条件

游戏客户端使用微信开发平台，开发语言采用JavaScript；应用服务器使用PHP语言开发；游戏逻辑服务器使用Java语言开发。实验环境服务器采用Windows Server2008，客户端环境采用iPhone8以及OPPOR12。

1. 相关理论及技术分析

设计的游戏客户端、游戏服务器、应用服务器均采用了不同的开发语言与架构，其中包括Java平台支持、HTML5标准中提出的相关特性、五子棋AI的相关理论的使用。下面对涉及到的原理进行较为详细的介绍。

## Java平台支持

Java是一种优秀的后台语言，提供了丰富的网络通信支持与多线程处理能力，同时借助JVM允许服务器软件在多种服务器操作系统中正常运行。

### 网络通信支持

TCP/IP是一种安全可靠的通信协议，由IP协议与TCP协议组成，IP协议使Internet成为一种允许连接不同类型计算机和不同操作系统的网络，TCP提供了可靠并且无差错的通信服务。TCP/IP协议广泛应用于网络应用、游戏应用中，Java平台对TCP/IP协议也提供了丰富的支持，在javax.net包中提供了对Socket相关操作类。

1. SSLServerSocket类

SSLServerSocket类是用来接收外界客户端访问的请求入口，该类将会对系统对应端口产生监听，以此获得客户机请求时带来的套接字（Socket）操作句柄，产生一段虚拟链路。

1. SSLSocket类

SSLSocket在服务端将来自于SSLServerSocket的产生，基于该类提供的输入输出流接口，实现与客户机的通信服务。

1. SSLContent类

SSL全称安全套接层（Secure Sockets Layer），是一种位于传输层的加密协议，采用非对称加密算法，由公钥私钥配套使用，私钥存储于服务器端且不对外公开，在确保私钥保密的情况下，可有效保证数据离开客户机抵达服务器前数据不被截取或窃听。SSLContent类将简化这一复杂的过程，正确配置后使得SSLSocket类与普通Socket类通信并无差别。

### 多线程技术

Java在本研究中承担GS（GameServer）角色，GS将负责游戏复杂的业务工作，面对来自外界的客户机的访问，GS无法避免涉及到对多线程技术的使用。Java平台同样的对多线程技术提供了丰富的支持。

1. 线程池

线程的创建与消亡将会对服务器造成额外的处理量，而对于GS而言，用户的登入登出本就是一件十分频繁的事情，因此需要利用线程池有效维护已经创建的线程，减少服务器不必要的开销，简单的说线程池是对线程的一种集合管理，对于已经创建且已经完成当前任务的线程并不急于回收，而是作为一种空闲状态暂时留存于内存中，当有需要的任务需要处理时，再次激活予以处理。Java中ExecutorService类提供线程的维护能力。

1. 线程安全

多线程的引用自然带来并发访问公共资源时的安全问题，例如GS中的当前在线用户集合、游戏房间等公共享用资源。Java的java.util包中给出的所有集合类，自身均不具备线程安全的读写能力，因此需要对这些资源进行人为的线程安全控制，通常在Java中解决线程安全问题可以使用以下三种方案：

a. 同步代码块。使用synchronized(this){}代码块，将产生并发问题的函数处理包裹住即可。

b. 同步方法。在产生并发问题的函数声明部位添加synchronized关键词。

c. Lock锁机制，通过创建Lock对象，采用lock()加锁，unlock()解锁，来保护特定的代码快。

## HTML5特性

万维网的核心语言、标准通用标记语言下的一个应用超文本标记语言（HTML）的第五次重大修改，本次修改为Web开发者带来了福音，原本Web自身不能实现的功能例如Web中的全双工通讯，甚至此前需要借助其他插件完成的事情，如网页交互动画早些年需要使用Flash完成，在本次标准均给出了Web原生的实现标准，各大浏览器厂家也都在积极支持这一标准。

### Canvas元素

HTML5标准定义了Canvas元素在JavaScript中丰富的绘图接口，希望以此替代早些年的内置在浏览器中Flash插件。

Canvas元素自身是一张矩形区域的画布，在JavaScript中提供了其对应的一些2D绘图接口，允许开发者在这样的一张画布中利用这些接口完成图像的任意绘制。Canvas中底层图像显示是以像素为单位，因此Canvas本身是一种位图绘制面板，所有的绘制操作均是进行逐像素渲染。此外，本次标准中还赋予了一些基本的操作方法，例如绘制路径、矩形、圆形、字符、添加图像以及矩阵变换得到的图像旋转、缩放等接口。

与传统语言如Java中Canvas相比，HTML5中Canvas有所优化，该元素已经优化了逐帧渲染导致的画面闪烁问题，这一问题在传统Canvas中需要通过双缓冲技术完成帧与帧之间的过渡，因此对Web开发者而言降低了使用难度。

### Web Socket

HTML5标准中提出了基于浏览器实现长连接通信的支持，给出Web Socket接口的相关定义，打破了此前浏览器通信中仅可以使用HTTP协议的单向拉取弊端。Web Socket是一种在单个TCP连接上进行全双工通信的协议，使得浏览器与服务器之间的数据交换变得简单，允许服务器端主动向浏览器发起数据推送。在Web SocketAPI中，只需要完成一次握手即可创建持久性连接，继而进行双方的通信。

## 五子棋AI算法

五子棋AI算法是一种较为典型的算法题目，以最简单的五子连珠为目的，使得计算机能够与人完成游戏的对弈。在五子棋AI领域上，目前有多种算法模型可供使用，一些常见的五子棋AI算法包括贪心法、字典表法、博弈算法等。

### 基本术语

为了更方便表达五子棋相关算法，简单对一些术语进行解释。

1. 天元：棋盘最中间的点。
2. 阳线：棋盘中可见的横纵直线。
3. 阴线：棋盘中无实线连接的隐形斜线。
4. 连：在棋盘中阳线和阴线任意一条线上形成2个或2个以上同色棋子连续排列。
5. 五连：在棋盘中形成5个同色棋子的“连”。
6. 活四：在棋盘中形成4个同色棋子的“连”，且两端延长线上各有一个无子的交叉点与此4子紧密相连。
7. 冲四：除“活四”外，再下一子便可形成五连。
8. 活三：在棋盘中形成3个同色棋子的“连”，且两端延长线上各有一个无子的交叉点与此3子紧密相连。
9. 眠三：除“活三”外，再下一子便可形成冲四。

### 棋面评估

无论采用贪心法还是博弈树法，对于五子棋某一局面均需要一种对双方有利程度的量化表示，以此表达当前局面对黑棋有利或是白棋，因此需要对当前局面进行评分，评分规则根据当前局面中所存在的一些特定棋子类型进行累加记分，若以黑棋为参考，黑棋所达成的所有棋形类型总和为正数，同理，白棋总和记为负数，黑白棋总值则代表当前局面的一个评分，评分的正负代表了对黑棋或是对白棋有利，评分的绝对值大小则代表了有利程度。

### 极大极小值搜索算法

在五子棋AI中，较有棋力的AI算法通常选用博弈算法中的极大极小值搜索算法，其核心思想为：

（1）对弈的双方以黑白棋区分，每一次局面对黑棋有利，则局面估值越大，以此设为Max，则白棋为Min。

（2）在递归推演过程中，对于黑棋执棋时总是选择对自己有利的选择，因此该层为Max层，同理，白棋为Min层，以此递归，则形成一颗博弈树。

接下来，将通过一张示意图表达极大极小值搜索算法，如图2-1。

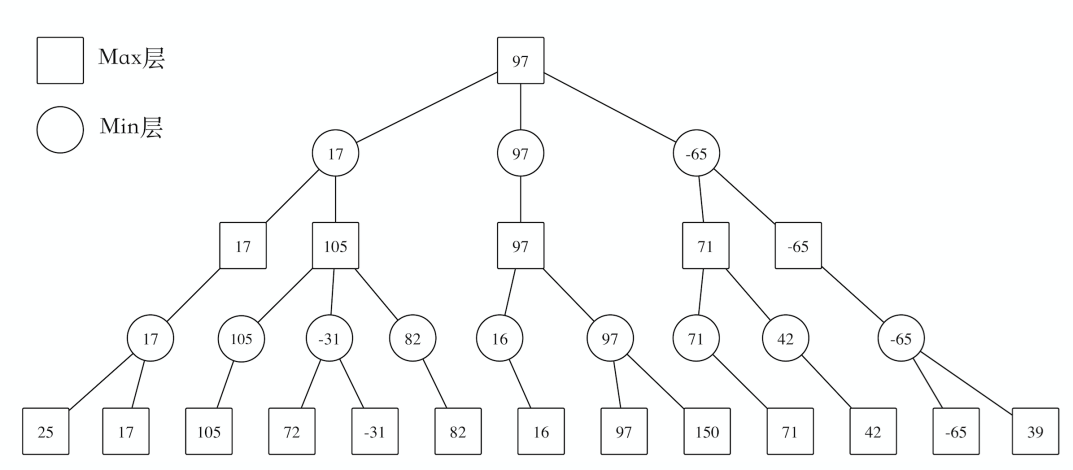


图2-1 极大极小值搜索算法示意图

### Alpha-Beta剪枝算法

使用极大极小值搜索算法，可以从某一局面衍生出一颗完整的博弈树，单从博弈树而言，意味着从任意局面开始，计算机已经得到了最终的结果，但这实际上是不合理的，首先这颗博弈树十分巨大，普通的计算机是难以在短时间内完全生成。在整个博弈树中我们发现到，当Max层中此前的推演已经获得一个较大值（或Min层的较小值），则在其他节点分支Max层向下推演时没有必要对比该值还小（或Min层中还大）的值继续递归，从对局理解角度，己方在执棋时如果下一手明显对对手产生优势，自然就不会继续落子。以此可以减掉大部分的分枝，其剪枝示意图如图2-2所示。

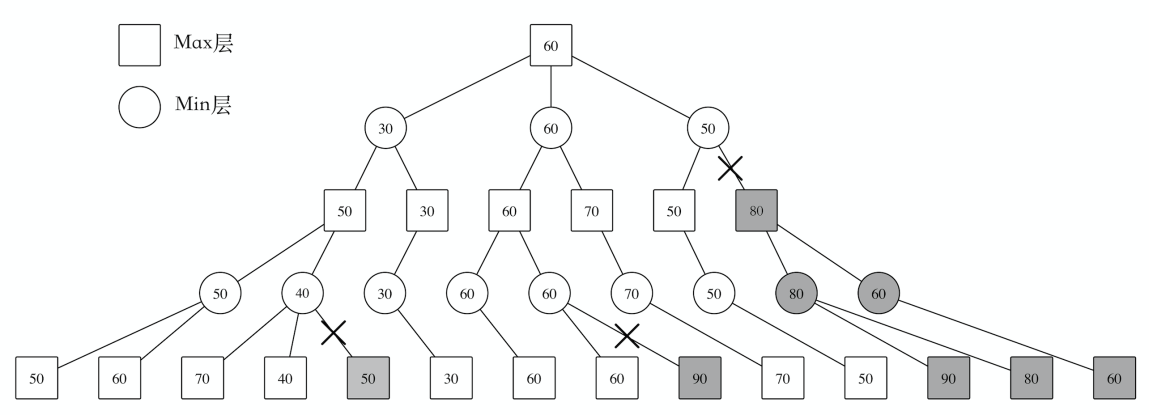


图2-2 Alpha-Beta剪枝优化示意图

### 启发式搜索

对于任意一层搜索而言，所有空子点都可以是未来推演的一条分支，但并不是所有的空子点都需要带入博弈树中，例如某一点位与其它所有棋子的点位间隔6个甚至更多的空点距离，意味着直至推演到未来一方取胜前，该子都没能参与到进攻或是防守的布局中，如同让对手一子，这很显然是不合理的。同时，基于2.3.4节所给出的Alpha-Beta剪枝算法，剪枝效率也完全取决于能否尽快搜索到最优解，因此每层代入的落子点应进行一次优先级扫描。这一搜索与棋面评分很类似，通常认为更具备威胁力的点位未来更有可能被选为AI给出的落点位置，点位优先的顺序：可形成连五、活四、冲四、活三、眠三、活二、其它，其它一般认为围绕当前已经落子情况，向外延长两个子棋的所有其它空子点集合。

1. 系统分析与设计

在进行游戏开发之前，必须要对游戏架构进行合理规划。本章对需求分析、系统的整体架构及所涉及的数据库设计进行详细介绍。

## 需求分析

课题的设计目标是开发一款可以实际运营的五子棋微信小游戏，基于该要求图3-1进行了简单的结构整理。

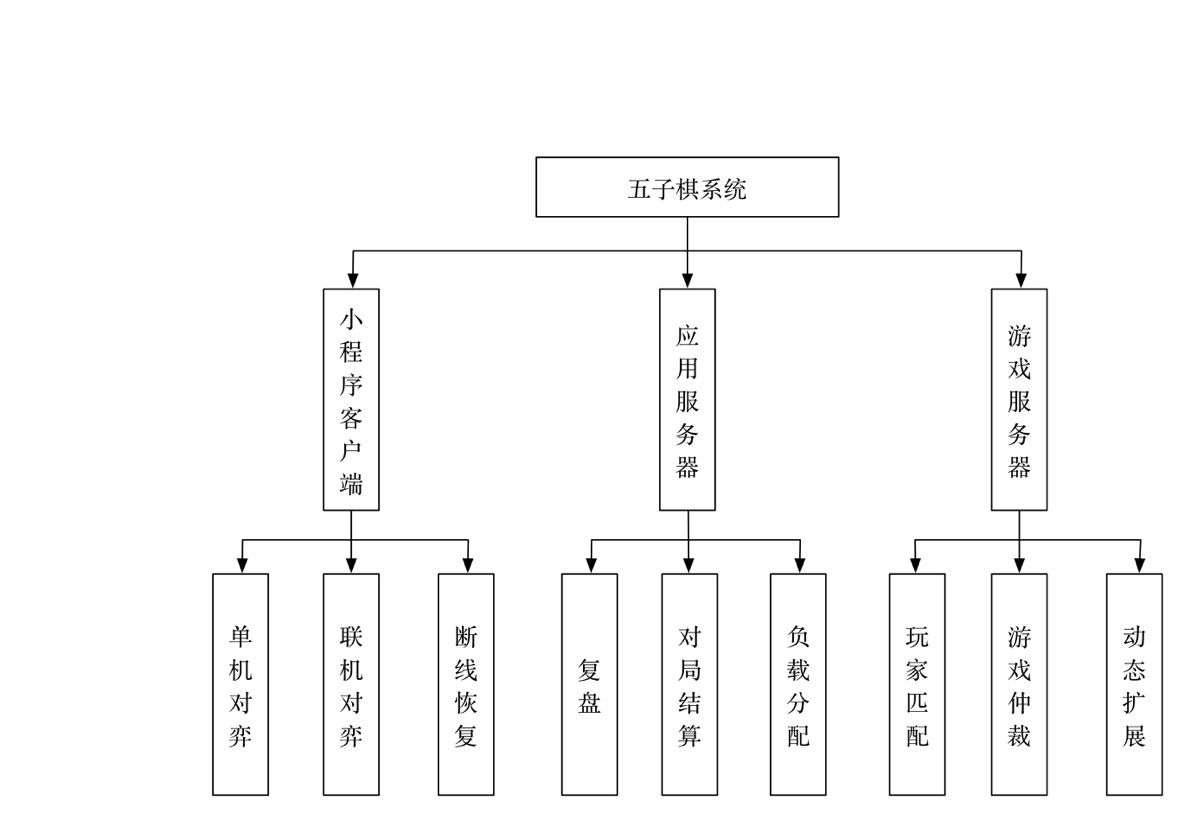


图3-1 主要功能模块

### 功能定义

(1)客户端小程序

客户端小程序需要完成游戏场景的渲染、与用户交互，提供清晰明了的操作界面，同时五子棋智能AI算法由客户端提供，在联机对弈中客户端应处理好意外断线重连后的游戏场景恢复能力。

(2)游戏规则

为满足大众娱乐需求，简化软件设计难度，课题中五子棋游戏规则采用15×15尺寸棋盘，并采用五子棋无禁手规则，在横、竖、左斜、右斜四个方向，优先形成相同颜色棋子的五子连珠，则持该颜色棋子方取得获胜。游戏对弈过程中允许悔棋，联机对弈中允许申请议和与宣告认输，且在一局游戏结束后双方互换棋子颜色可重新开局。

(3)五子棋智能AI

五子棋AI算法是单人模式下的核心组件，承担与人类玩家对弈时的执棋策略的分析与计算，根据实际的使用需求，该AI应具备以下4点基本需求。

a) 支持15×15尺寸五子棋棋盘规格；

b) 支持先后手切换能力；

c) 最长响应时间应不超过3秒，对于多数情况下应低于2秒响应时间；

d) 允许悔棋等操作。

为了丰富作品的功能，在单人模式下，AI同时也应具备对危险局面的落点提醒功能。

(4)应用服务器

应用服务器用于记录对弈结束后的对局结算信息，为用户提供复盘能力，且游戏服务器的开机认证统一由应用服务器完成记录，用户在使用客户端连接服务器时，应用服务器应提供当前已经完成开机的游戏服务状态列表以供连接。

(5)游戏服务器

服务器作为联机模式下重要的组成部分，应具备多用户同时在线并发处理能力；面相用户提供相对安全可靠的基于微信身份验证的登录机制；提供在游戏开始前玩家之间的匹配能力；为保证游戏的公平性，应在游戏内起到对弈中的操作权限控制以及游戏结果仲裁控制；在面对大量用户的使用中，还应提供一种可动态的扩展能力。

### 游戏流程分析

根据人们使用棋牌类游戏习惯，图3-2给出五子棋小游戏的使用流程图，用户访问该应用时，选择所需要的游戏模式，系统将根据需求分配对手，确定对手后将开始游戏。游戏中双方开始轮流执黑棋先行，每次落子都将进行胜负检查，决出胜者则对弈结束，期间用户可以进行悔棋操作，将连退两步，游戏结束后将重新开始。

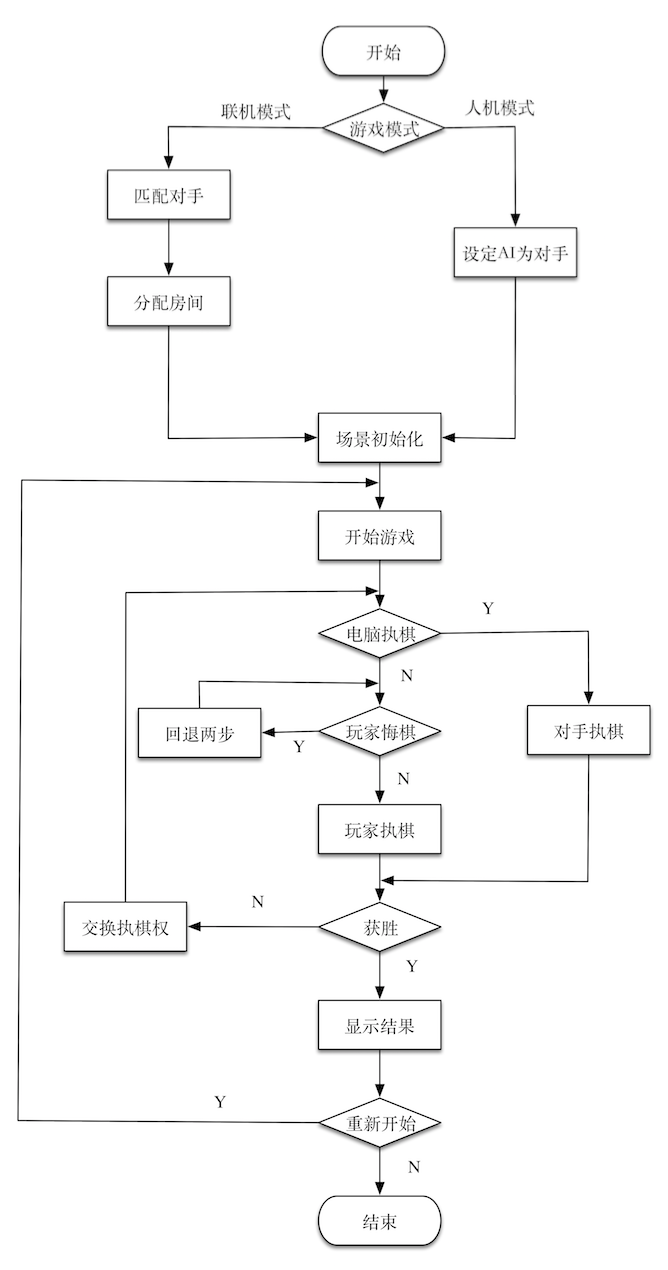


图3-2游戏对弈流程图

## 系统设计

课题涉及到客户端、游戏服务器、应用服务器三种角色的设计，本节将对其进行逐一介绍。

### 总体结构设计

客户端、游戏服务器、应用服务器的实际物理部署关系如图3-3所示。其中应用服务器使用PHP语言开发，配合MySQL数据库进行信息关系化存储，游戏服务器使用Java语言平台开发，社区客户端与游戏客户端分别使用微信平台下的微信小程序及小游戏技术开发。应用服务器作为数据中心，记录用户信息、游戏信息以及提供社区服务，游戏服务器承担联网模式下的实时对弈服务，游戏客户端完成游戏场景的渲染及用户交互。

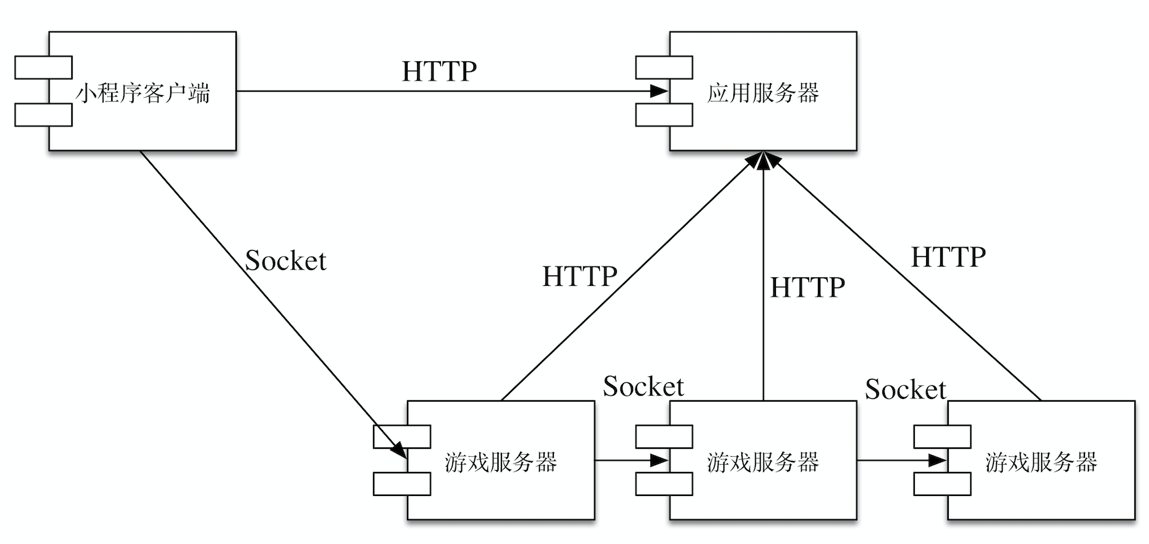


图3-3物理部署关系构件图

(1)小程序客户端结构设计

客户端由渲染引擎类(Render)、游戏驱动类(Gobang)、网络操作类(Net)、智能AI类(AI)构成，其主要构成如图3-4所示。

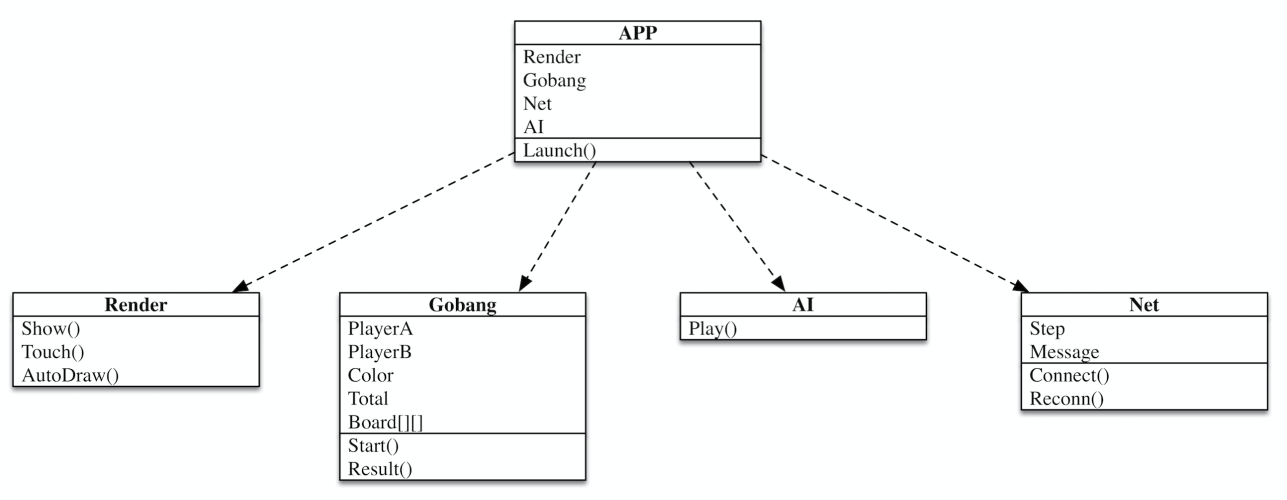


图3-4小程序客户端主要类类图

(2)应用服务器结构设计

应用服务器采用MVC框架结构设计，由负载均衡模型(SLB)、游戏信息模型(Record)、私信聊天模型(Chat)构成，其结构设计如图3-5所示。

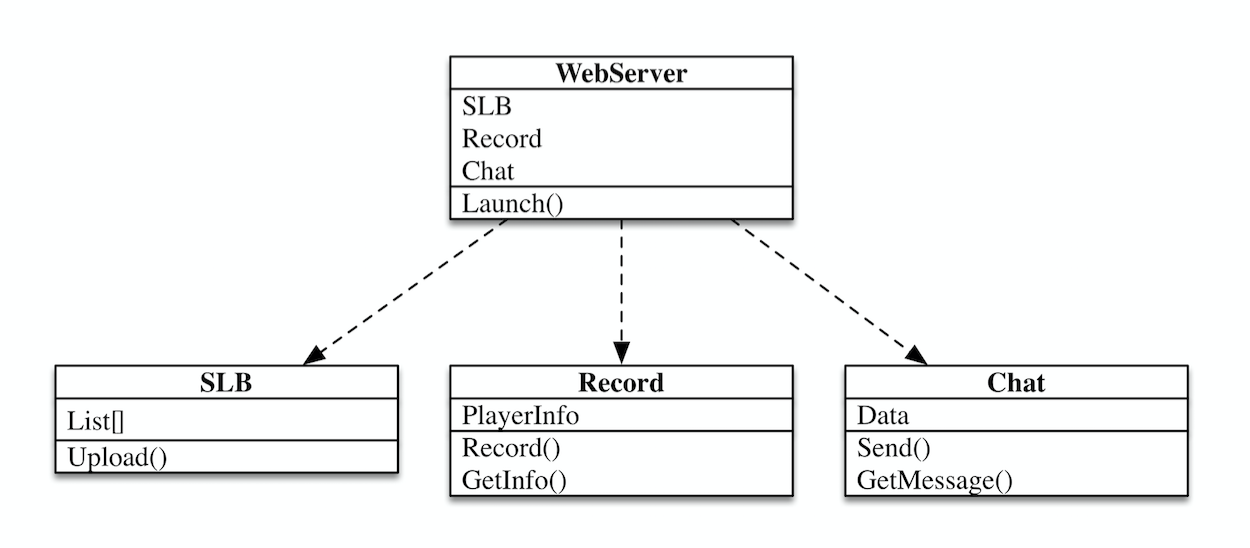


图3-5应用服务器主要模型类图

(3)服务器集群结构设计

游戏服务器由房间类(Room)、匹配模块(Match)、拓扑通信(Topology)、玩家类(Player)构成，其结构设计如图3-6所示。

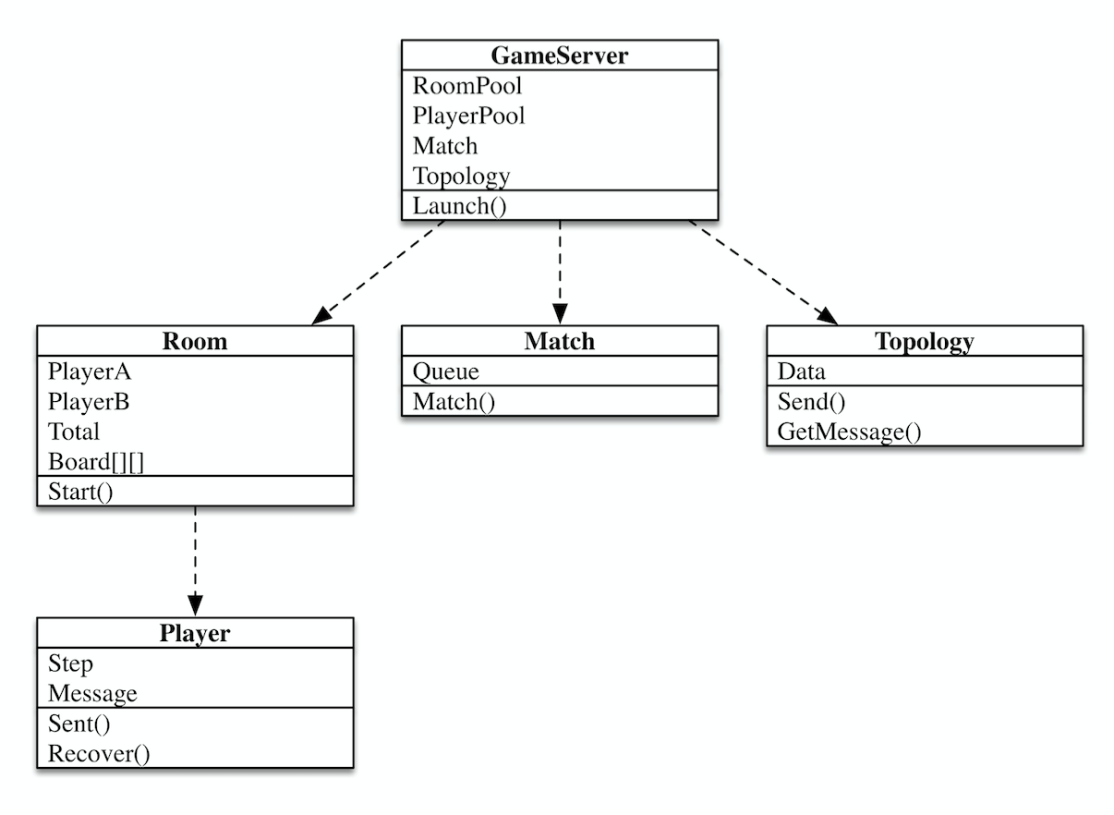


图3-6游戏服务器主要类类图

### 系统工作机制

(1)服务器动态扩展机制

图3-7给出了服务器动态扩展的顺序图，游戏服务器周期性向应用服务器SLB模块递交状态报告，新增服务器时，需要向应用服务器请求开机许可，并获得当前已经完成开机的其他游戏服务器的拓扑表，许可成功后完成开机并与其他主机建立通信，等待客户端的访问。客户端在需要访问游戏服务器时，先向应用服务器请求可用资源列表，根据资源可用情况进行游戏服务器的访问。

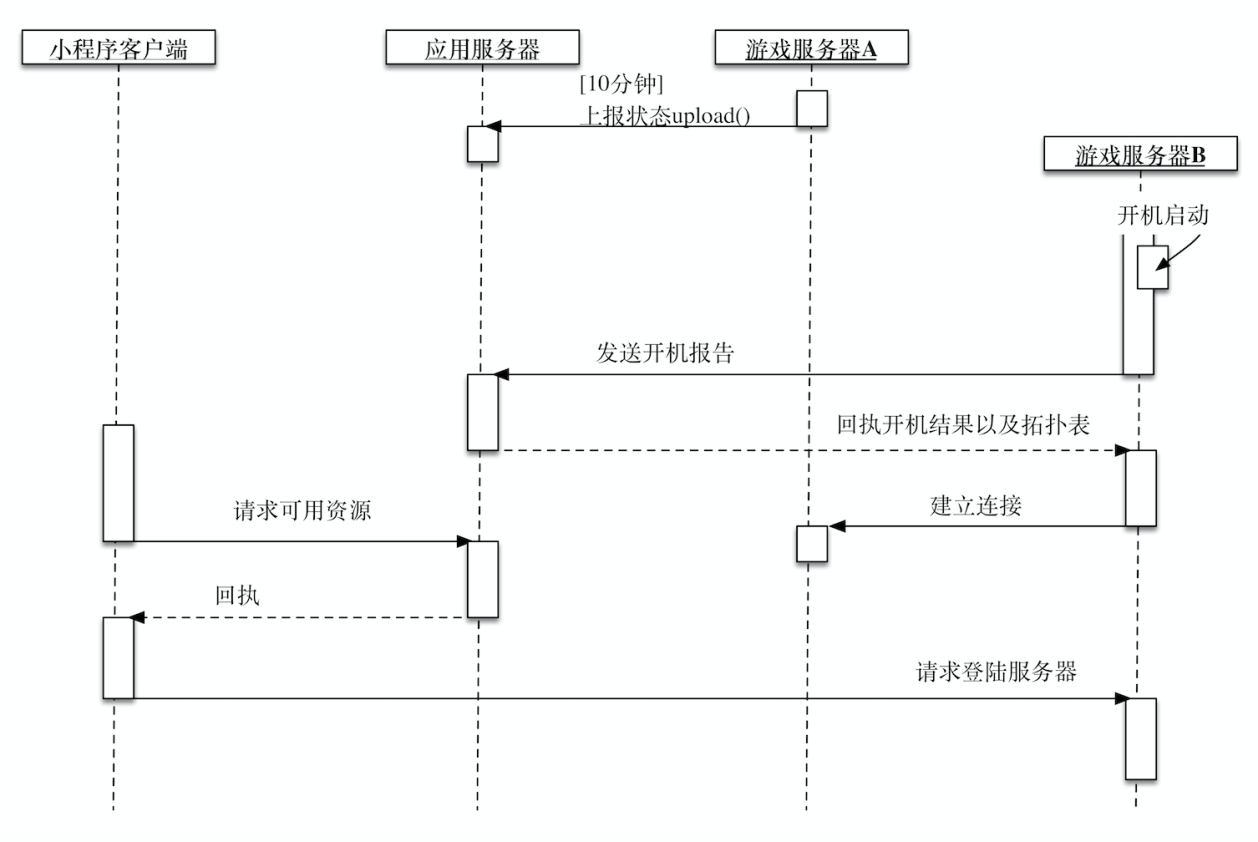


图3-7游戏服务器动态扩展顺序图

(2)鉴权机制

客户端获得服务器资源后立即访问游戏服务器，游戏服务器需要对来访者进行身份鉴定，身份信息统一由应用服务器维护。为了避免用户微信身份信息的泄露，鉴权过程中没有直接使用用户注册微信时的SessionID，而是由用户首先向应用服务器申请具有短时效的临时Token。访问游戏服务器时递交Token，游戏服务器将向应用服务器询问Token的有效性，从而确定是否保留连接，图3-8给出了这一过程的顺序图表示。

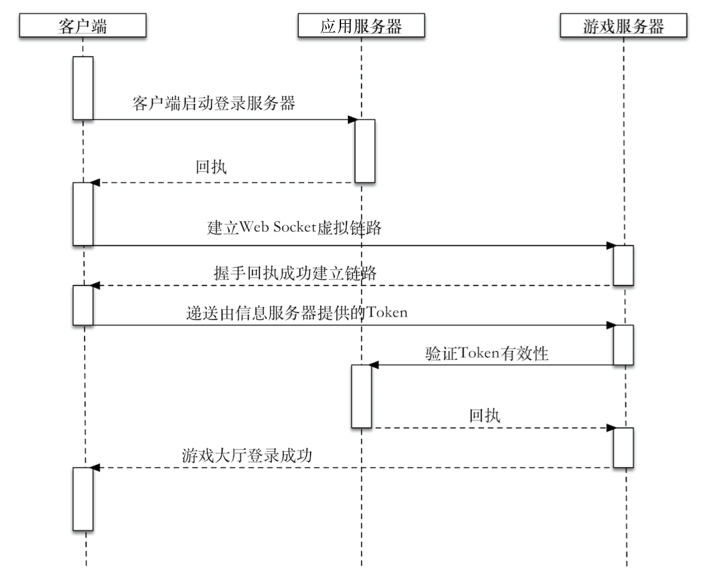


图3-8鉴权过程顺序图

## 数据库设计

在本系统中，数据库为信息系统提供数据存储服务，主要包括用户游戏数据、推荐文章、对局数据、用户好友关系等。下面将结合实际业务对各个表进行说明。

1. 管理员帐号表（admin）

用于记录管理员登录所需信6息，具体设计如表3-1所示。

表3-1管理员帐号表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 用户ID | 主键，自增 |
| 2 | user | varchar (16) | 帐号 |  |
| 3 | password | varchar (32) | 密码 | 使用MD5加密 |

1. Banner表（banner）

用于管理员设置首页宣传面板，Banner内容主要由展示图片构成，点击后产生对应的事件，可以打开特定APP，或者访问特点网页。

表3-2 Banner表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | BannerID | 主键，自增 |
| 2 | src | varchar (512) | 图片地址 |  |
| 3 | method | varchar (16) | 打开类型 |  |
| 4 | value | varchar (512) | 值 |  |

1. 热点文章表（information）

在配套的用于信息展示的小程序首页，将为用户提供五子棋相关教学文章，文章内容有该表控制，具体包括文章标题、简介、点击后对应URL地址、发布日期等字段。

表3-3 热点文章表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 文章ID | 主键，自增 |
| 2 | title | varchar (128) | 文章标题 |  |
| 3 | introduce | varchar (1024) | 简介 |  |
| 4 | url | varchar (512) | 目的URL |  |
| 5 | release\_time | int (11) | 发布时间戳 |  |

1. 私信黑名单表（char\_ban）

因为系统包含用户之间互发消息功能，为了避免用户遭受骚扰，该表将建立私信黑名单关系。

表3-4 私信黑名单表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 记录ID | 主键，自增 |
| 2 | playerA | int (11) | 用户ID |  |
| 3 | playerB | int (11) | 被限制用户ID |  |
| 5 | time | int (11) | 设置时间戳 |  |

1. 私信消息表（chat\_message）

该数据库表用于记录用户互发具体消息，包括消息内容、发送时间、用户ID、已读未读标识等字段。

表3-5 私信消息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 消息ID | 主键，自增 |
| 2 | content | int (11) | 消息内容 |  |
| 3 | time | int (11) | 消息时间 |  |
| 5 | playerA | int (11) | 发送者ID |  |
| 6 | playerB | int (11) | 接收者ID |  |
| 7 | is\_read | int (4) | 是否已读 |  |

1. 消息关系表（chat\_relation）

私信功能不支持匿名发送，只有双方互相关注后才可以发送消息，该表用于建立用户双方关系。

表3-6 消息关系表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 消息ID | 主键，自增 |
| 2 | playerA | int (11) | 用户ID |  |
| 3 | playerB | int (11) | 被关注者ID |  |
| 4 | time | int (11) | 建立时间 |  |

1. 信息服务下的用户表（clsc\_player）

信息服务拥有独立的用户表，用于记录用户在微信侧的授权信息，并使得用户可以在下次访问app时自动登录。

表3-7信息服务下的用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 记录ID | 主键，自增 |
| 2 | openID | varchar (64) | 微信OPENID |  |
| 3 | name | varchar (64) | 昵称 |  |
| 4 | sex | int (4) | 性别 |  |
| 5 | is\_authorized | int (4) | 是否授权 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | 续表3-7 |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 6 | imgURL | varchar (512) | 头像 |  |
| 7 | registerTime | int (11) | 注册时间 |  |
| 8 | sessionKey | varchar (64) | 会话安全密钥 |  |
| 9 | sessionID | varchar (64) | 会话ID |  |
| 10 | city | varchar (64) | 城市 |  |
| 11 | country | varchar (64) | 国家 |  |
| 12 | province | varchar (64) | 省 |  |

1. 历史对局表（history）

玩家在一次对局后，由游戏服务器将对局信息进行上报，被应用服务器成功接受并被记录，其中对局的详细信息将被该表记录，主要用于记录一场对局的胜负信息、复盘信息等。

表3-8 历史对局表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 消息ID | 主键，自增 |
| 2 | playerA | int (11) | 玩家AID |  |
| 3 | playerB | int (11) | 玩家BID |  |
| 4 | result | int (4) | 对局结果 | 1黑2白3平 |
| 5 | start\_time | int (11) | 对局时间戳 |  |
| 6 | time | int (11) | 持续时间 | 秒 |
| 7 | data | varchar (1024) | 棋盘数据 |  |
| 8 | black | int (4) | A执棋颜色 |  |
| 10 | type | int (4) | 对局类型 |  |

1. 玩家表（player）

游戏的玩家信息表，记录玩家在微信侧的授权信息，并使得用户可以在下次访问游戏时自动登录，同时记录玩家一些对局信息，例如：总场次、胜场次、平局场次等信息。

表3-9 玩家表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 消息ID | 主键，自增 |
| 2 | playerA | int (11) | 玩家AID |  |
| 3 | playerB | int (11) | 玩家BID |  |
| 4 | result | int (4) | 对局结果 | 1黑2白3平 |
| 5 | start\_time | int (11) | 对局时间戳 |  |
| 6 | time | int (11) | 持续时间 | 秒 |
| 7 | data | varchar (1024) | 棋盘数据 |  |
| 8 | black | int (4) | A执棋颜色 |  |
| 9 | total | int (11) | 总手数 |  |
| 10 | type | int (4) | 对局类型 |  |

(10)授权关系表（relation）

信息服务与游戏服务中玩家的表均为独立管理，这取决于微信openID的机制导致（同一用户不同app下ID不同），因此该表将用于建立两款应用建立同一用户的关系。

表3-10 授权关系表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 记录ID | 主键，自增 |
| 2 | player\_id | int (11) | 用户ID |  |
| 3 | classic\_player\_id | int (11) | 信息系统表ID |  |

(11)积分表（score）

为自己的游戏系统设计了一种游戏内货币——积分，该表用于存储用户所得积分情况。

表3-11 积分表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 记录ID | 主键，自增 |
| 2 | pro\_id | int (11) | 用户ID |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | 续表3-11 |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 3 | score | int (11) | 信息系统表ID |  |
| 4 | time | int (11) | 最后领取时间 |  |
| 5 | today\_score | int (11) | 今日额外分数 |  |

(12)签到表（sign）

签到作为目前作品中获得积分的一种途径，签到记录又该表完成。

表3-12 签到表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 数据库字段 | 类型长度 | 字段含义 | 其他说明 |
| 1 | id | int (11) | 记录ID | 主键，自增 |
| 2 | pro\_id | int (11) | 用户ID |  |
| 3 | data | varchar (7) | 签到数据 |  |
| 4 | last\_signtime | int (11) | 最后签到时间 |  |

1. 五子棋AI算法设计

本章集中讨论五子棋对弈中所涉及的AI算法，主要包括棋面评估、极大极小值搜索、Alpha-Beta剪枝优化、启发式搜索、评估缓存等。

## 算法描述与程序实现

为了简化问题，以15×15标准五子棋棋盘作为研究对象，且无禁手规则。

### 棋面评估

对于任意一种局面而言，AI需要一种对当前局面优劣情况的量化表示，将连续5个位置所组成的状态记为式(4-1)所示的五元组：

(4-1)

其中，表示棋盘中的一个落子点，。满足约束条件为：所有点位于同一条直线上，图4-1给出五元组S的四个方向覆盖区域。

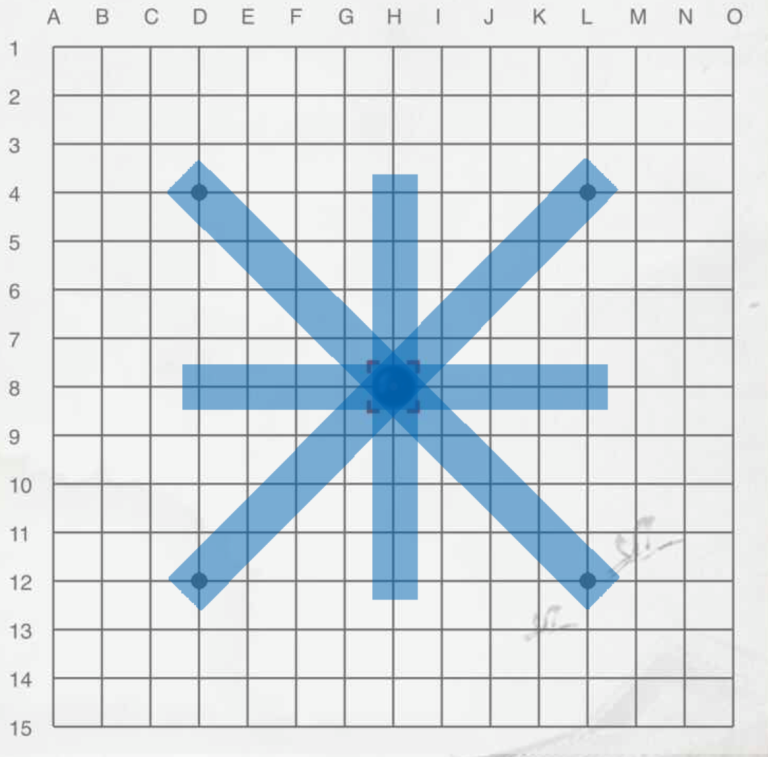


图4-1 五元组*S*的4个方向

对于任意的和*S*，如果*S*，则*P*可以是*S*中任意位置。由于*S*有4个方向，对于每个方向，根据*P*的位置由1变化到5，可构成5种可能。故包含点的五元组至多可能有4×5=20种。若*P*位于棋盘边缘，因为后续生长受到抑制，则会低于20种可能。

将所有包含*P*的*S*记为集合，则

(4-2)

根据五子棋比赛规则，对于任意，根据下列规则对赋予合适的权重：

(1)为五连状态， ;

(2)为活四状态， ;

(3)为活三、冲四状态， ;

(4)为活二、眠三状态， ;

(5)为活一、眠二状态， ;

(6)为其他状态， ;

则对于棋盘一个可落子的位置，按式(4-3)计算该点与部分落子点达到的最好状态的评估值*F*()：

*F*()*=*  (4-3)

将某一对弈局面*ϖ*中的全部黑子点位记为集合*α*，全部白子点位记为集合*β*。则计算黑棋局面的评估函数*Hα*为：

*Hα* = (4-4)

同理，计算白棋局面的评估函数*Hβ*为：

*Hβ* = (4-5)

令*f*(*ϖ*)作为局面的总评分，对黑棋得分记为正数，白棋为负数，则评估总分计算如式(4-5)：

*f*(*ϖ*)*= Hα-Hβ*  (4-6)

整个局面某一颜色棋子的估值为各点估值总和，下列代码给出对应的程序实现方法。

var evl = function(board,color){

var Score = 0;

for(var x in board){

for(var y in board[x]){

if(board[x][y] != color)

continue;

var Score = 0;

Score += Cal100000(x,y,board);//有值则返回对应值，无值则返回False

if(!Score)

Score += Cal10000(x,y,board);

if(!Score)

Score += Cal1000(x,y,board);

if(!Score)

Score += Cal100(x,y,board);

if(!Score)

Score += Cal10(x,y,board);

}}

return Score;

}

### 极大极小值搜索

一个棋面在计算AI执棋颜色时，所得分为正数，计算人执棋颜色得分为负数，则当前棋面得分应为AI得分+人得分。于是对于一层遍历而言，将得到一个总得分值，该值可用于得出当前局面更有益于AI亦或人类， AI层更希望获得高得分，人类层更希望获得低得分，根据不同层所期望的值有大有小，则该基本博弈算法极大极小由此而来，关键代码（伪代码）如下所示：

var maxmin = function(board,deep){

var points = getPoints(board,deep); //启发式搜索

var best = 0;

for(var i in points){

var pot = points[i];

board[pot[0]][pot[1]] = R.A; //尝试落子

var v = min(board,deep - 1);

if(v >= best){

best = v;

bestPoints.push(pot);}

board[pot[0]][pot[1]] = 0; //将落子归位

}

return best;

}

var min = function(board,deep){

var valueA = evl(board,R.A);

var valueB = evl(board,R.B); //分别计算双方棋子局面评分

var v = valueA - valueB; //差值作为当前局面评分

var points = getPoints(board,deep); //启发式搜索

var best = 0;

for(var i in points){

var pot = points[i];

board[pot[0]][pot[1]] = R.A; //尝试落子

var v = max(board,deep - 1);

if(v <= best){

best = v;

bestPoints.push(pot);}

board[pot[0]][pot[1]] = 0; //将落子归位

}

return best;

}

其中best指搜索当前搜索中所得到的最优值（在Min层中指最小值，Max层中指最大值），变量v指当前点位代入后所得分值，在Max层中代码中仅best与v比较值和Min层不同，因此代码将不进行给出。

### Alpha-Beta剪枝优化

设推演层数为N，推演子节点数为M，则极大极小值未经优化时的时间复杂的为T()=O(MN)，当每局约50手，推演4层时，所构建的博弈树将包含504=6,250,000个字节点，目前硬件处理及存储设备无法满足该需求，不适用于游戏使用。

当从一种棋面延伸到一整个博弈树，通过极大极小算法每一层将可以获得对于AI或者人类具有价值的下一种棋面，但考虑一种问题，若尝试完成一次落子推演后，获得新的一种棋面，该棋面将会给出一种得分，对于AI而言，如果在先前推演中已经获得一个较高的得分局面，则在下一次推演时无需再对比该局面还低得分局面进行推演，对于人类层而言亦是如此，因此可以得到如此结论：

在Max层中，如果发现下一个节点低于此前推演的一个节点的得分，则跳过该节点；在Min层中，如果发现下一个节点高于此前推演的一个节点的得分，则跳过该节点。图4-2给出剪枝流程图。

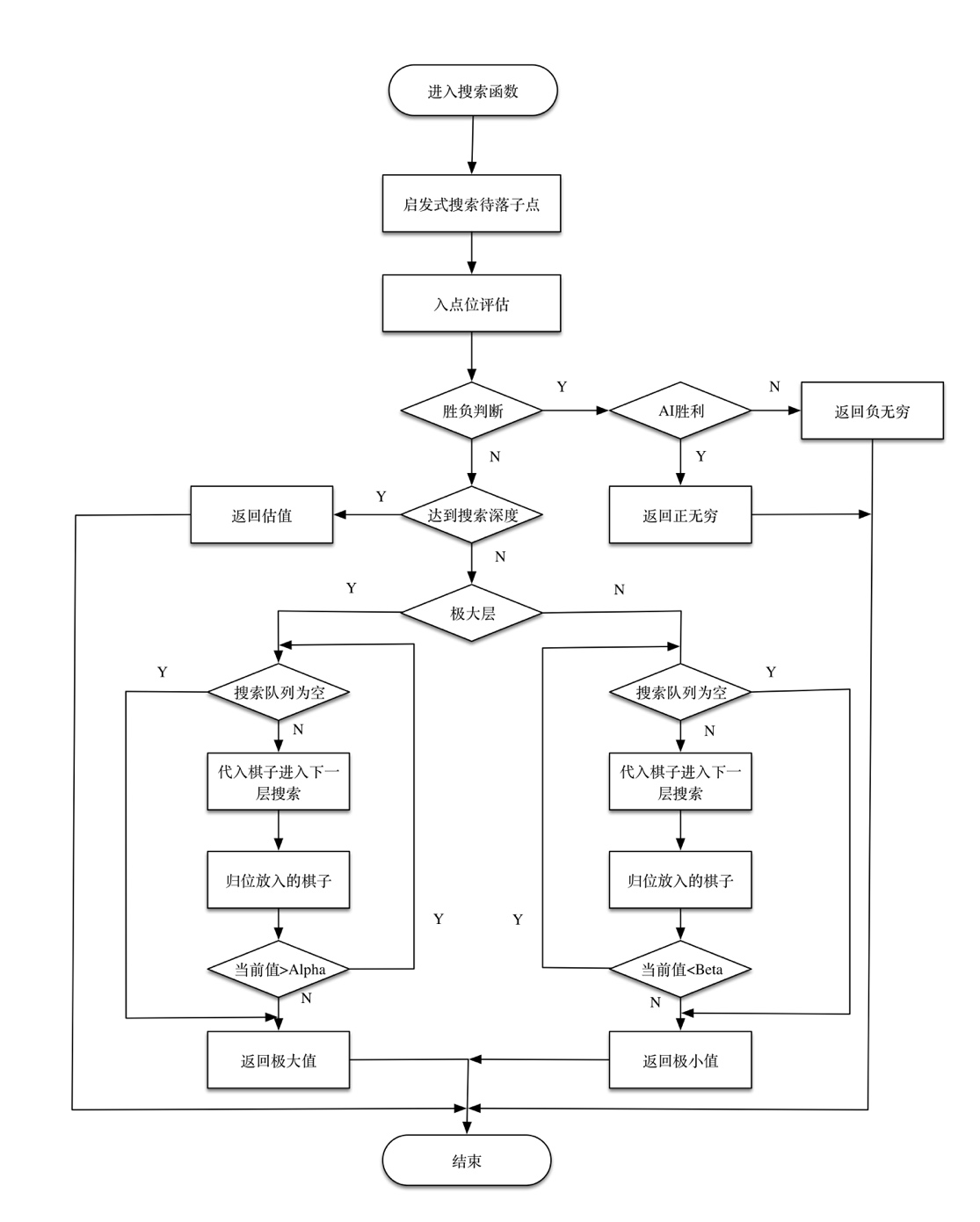


图4-2 剪枝算法流程图

代码实现上，将在极大极小递归函数中新增变量值，用以剪枝，新增代码如下:

var min = function(board,deep, alpha, beta){

var valueA = evl(board,R.A);

var valueB = evl(board,R.B); //分别计算双方棋子局面评分

var v = valueA - valueB; //差值作为当前局面评分

if(deep <= 0 || win(board)) {

return v;

}

//继续搜索

var points = getPoints(board,deep); //启发式搜索

var best = 0;

for(var i in points){

var pot = points[i];

board[pot[0]][pot[1]] = R.A; //尝试落子

var v = max(board,deep - 1);

if(v <= best){

best = v;

bestPoints.push(pot);

}

if(v < beta) { //AB 剪枝

ABcut ++;

break;

}

board[pot[0]][pot[1]] = 0; //将落子归位

}

return best;

}

在计算极大值的过程中，该方法中以alpha为剪枝标准，将beta变量替换为alpha，其它计算过程与min过程一致，不再陈述。

### 启发式搜索

对于系数M存在明显需优化地方，例如当面对一个棋面时，所需要推演的落子节点并不需要遍历当前棋盘所有的空子点，因此在获得推演节点集合时，需要一种搜索策略，给出具备有较大概率使得能够较快访问到极大极小所需要的极值点，结合“得分”计算规则，先对当前局面进行简单搜索，对于可以成五、活四、双三这类必杀局面的点位排序较前，并对形成其他局面根据分数做出简单排序，为了防止遗落，最终仍对所有落子点，四周延展两个棋子位置的落子点做最后的点位候选，点位所产生的棋形的评分定义为：

module.exports = {

ONE: 10, //活一

TWO: 100, //活二

THREE: 1000, //活三

FOUR: 100000, //活四

FIVE: 1000000, //成五

BLOCKED\_ONE: 1, //眠一

BLOCKED\_TWO: 10, //眠二

BLOCKED\_THREE: 100, //眠三

BLOCKED\_FOUR: 10000 //眠四

}

搜索分为三步骤：

（1）扫描全局空点位，其四周延长线2个点位内存在已落子点。

（2）对步骤（1）所给出集合依次代入当前布局，在新局面中根据评分定义给出该点位得分。

（3）根据得分重新排序，得分高者优先。

### 评估缓存优化

由于在推演的过程中，一次推演的某一分支将会作为下一次推演分支的重要组成部分，为了减少未来的计算量，在先前推演的过程中，将一些局面得分进行缓存，以备之后的推演使用加快搜索。这里将考虑如何使用一种低代价的表达方式，可以代表某一种局面。

由于五子棋棋盘大小游戏，因此可以对某一颜色棋子单独提取，按照0、1序列逐行填入位中，五子棋标准棋盘大小为152=225，则可以使用最大256位长度序列存储该局面，值得注意的是该序列无需从棋盘左上角第一个点位开始到棋盘右下角结束进行排列，仅需要按照一定的扫描顺序从棋盘中第一个有子点位开始到该扫描顺序最后一个棋子点位结束即可，因为局面评估关心的是黑白棋子之间的相对位置，最终可以得到一个数值，该数值即可表达一种局面，如此存放，此后在计算局面得分时仅需要检索此前的缓存记录，可以提高计算效率。其点位扫描代码为：

var search = function(board){

var str = ""; //棋盘序列

var record = false; //开始记录标识，扫描到第一个有子点位时记为True

for(var x in board){

for(var y in board){

if(!record){

//未开始

if(board[x][y] == 0)

continue;

else

record = true;

}

if(record){

if(board[x][y] == c){

str += '1';

}else{

str += '0';

}

}

}

}

return str;

}

## 实际测试

### 落子计算性能测试

经过对算法优化，目前已经可以保证单步骤AI思考时间低于3秒，且绝大多数响应时间低于1秒。图4-3给出在iPhone8设备中一次单人对弈中AI思考时间折线图，思考时长均不超过1.4秒。

图4-3 对弈AI思考时间折线图

### AI棋力测试

使用本作品AI与目前同类小游戏作品中给出的AI进行棋力对抗，得出以下测试数据，如表4-1，基本满足普通玩家的使用需求。

表4-1 棋力测试表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 应用名称 | 胜局 | 总次数 | 胜率 |
| 1 | 五子棋Pro(iOS) | 43 | 100 | 43% |
| 2 | 欢乐五子棋腾讯版 | 93 | 100 | 93% |
| 3 | 人机五子棋 | 100 | 100 | 100% |

1. 游戏系统的详细设计与实现

本章详细介绍了系统的主要功能模块的设计与实现，将分别对游戏客户端、应用服务器、游戏服务器三个角色中用到的关键技术进行具体描述。

## 游戏客户端

游戏客户端是在微信小程序平台发布，由近似HTML、CSS、Javascript的微信改装语言构成，其中主要的功能模块包括HTML5Canvas游戏引擎，五子棋AI算法，封装Web Socket通信模块，使其适合游戏使用。

### HTML5Canvas游戏引擎

游戏需要更加流畅和自由的画面渲染要求，相比于HTML中DOM树结构的元素构成而言，无论是渲染效率还是影像的自由都有较大的限制，因此在游戏中使用可以自由绘制Canvas元素用于对游戏画面输出。但是Canvas绘图HTML5标准中仅给出了基本绘图API，对点线面的生成，位图的引用等。由于对于DOM事件仅可以获取到Canvas元素的点击事件，因此相比于DOM树结构又缺少了简单的事件机制。综上所知，如果直接使用Canvas底层绘图API直接设计较为复杂的游戏，在程序设计上将不是一个合理的结构，本课题设计一种设计模式，借鉴了Windows可视化开发的编程习惯，如同管理Windows窗口程序一样管理自己的游戏内容，更加方便的获取场景事件。

1. 场景管理器

游戏所有场景的渲染将统一由场景管理器控制，场景管理器中给出活跃窗口概念，活跃窗口指用户当前可视窗口画面，管理器获得具体窗体实体后将会自动产生渲染，对用户点击事件进行自动拾取并反馈，下面将单独拿出几处关键部位具体说明。

1. 自动渲染。

管理器在获得窗口实体后，将根据用户当前设备性能进行反复重绘，以保证画面在不影响其他业务性能的情况下得到动画画面。重绘关键代码如下：

AutoDraw ( Status = true ) {

if ( Status ) {

this.AutoDraw( false );

this.\_AutoDrawStatus = true;

this.\_AutoDraw();

} else {

this.\_AutoDrawStatus = false;

if ( this.\_AutoDrawID != null ){

cancelAnimationFrame( this.\_AutoDrawID );

this.\_AutoDrawID = null;

}

}

}

\_AutoDraw(){

this.\_AutoDrawID = requestAnimationFrame (

function() {

this.Draw();

if ( this.\_AutoDrawStatus ) {

this.\_AutoDraw();

}

}.bind( this )

);

}

其中requestAnimationFrame()将产生一次当前性能可靠的间隔回调，当设备足够流畅时，FPS（画面每秒传输帧数）为60，若当前设备性能较低时，在用户侧将会感到画面明显的卡顿，减少绘制频率为其他业务逻辑提供运算资源。

2）元素拾取

Canvas与传统DOM事件有所不同，DOM事件不会返回Canvas画布内部元素的回调事件，因此需要根据DOM事件返回的Canvas区域坐标，自行计算出用户所点位置所被拾取的元素位置，因此无论什么元素，在继承Element.js时需要给出元素的有效范围计算公式，具体重写代码例如：

Over(x,y){

if( x >= this.\_Left &&

x <= this.\_Left + this.\_Width &&

y >= this.\_Top &&

y <= this.\_Top + this.\_Height){

return true;

}

}

场景管理器在渲染页面时将自动调用该重写方法，根据坐标运算给出拾取结果。

3）比例坐标

由于目前市面上移动设备总类繁多，屏幕尺寸各不相同，因此在游戏场景设计中，不能使用像素单位作为坐标定位，但是建议在设计时，使用最具备大众使用的设备尺寸，开发者通常以iPhone6设备尺寸（750×1334）作为参考。在设计中为整体渲染注入缩放参数，只需要在应用程序初始化时，根据用户当前设备与参考尺寸做好兼容，即可以适应多数用户设备，本作品兼容及设置相关代码如下所示：

constructor( Canvas ){ //初始化上屏Canvas

this.\_Canvas = Canvas;

this.\_ctx = Canvas.getContext( "2d" );

this.\_Width = Canvas.width;

this.\_Heigth = Canvas.height;

this.\_Pages = new Array();

this.\_CurrentPage = null;

//倍率

this.\_Scale = 1;

this.\_AutoDrawStatus = false;

this.\_AutoDrawID = null;

this.AutoDraw( true ); //默认应为 true

//拾取

this.\_Touch( true );

}

由于Canvas本身分辨率较低，若直接渲染对于图像将产生很严重的锯齿感，由于微信小游戏的视窗一定是自动拉伸Canvas使其充满用户屏幕，所以可以通过强制的放大Canvas画布属性，提供用户屏幕单位区域的Canvas有效像素点以增加图像的输出质量，通常的将画布放大2~3倍则足够清晰，过大将会增大系统对图像的处理负担和游戏客户端本身的体积。具体设置代码：

Init(){

let System = wx.getSystemInfoSync(); //放大屏幕分辨率消除图像锯齿

console.log( System.screenWidth / 375 \* 3 / 2 );

Global.Scale = System.screenWidth / 375 \* 3 / 2;

Global.CanvasWidth = System.screenWidth \* 3 / Global.Scale;

Global.CanvasHeight = System.screenHeight \* 3 / Global.Scale;

Global.ScreenWidth = System.screenWidth;

Global.ScreenHeight = System.screenHeight;

this.\_Animation.Scale( System.screenWidth / 375 \* 3 / 2 );

……

}

### 客户端游戏逻辑处理模块

本作品支持单机及联机对弈，因此游戏在本地上需要有五子棋相关的逻辑处理，而实际上，即使是一款纯联机游戏，为了提高游戏的执行效率，减少一定的服务器压力，在用户设备上仍然要存在游戏逻辑上的必要处理，这如同用户注册时虽然服务器脚本会检查账号密码的有效性，但对于浏览器而言，提交前会进行简单校验，避免不必要的网络请求。本节将简单介绍五子棋游戏在用户设备上的核心处理类。

1. 基本结构及胜利检查

五子棋是一个种入门的游戏逻辑题材，标准五子棋棋盘大小为15×15，因此建立一个15×15二维数组描述棋面较为适合。分别通过0，1，2三个数值代表，空、黑、白格子占用情况，因此判断一局的胜利仅需要对该二维数组进行完全遍历，遍历到非0坐标时，以此为起点，分别向右上、右、右下、下四个方向延长寻找是否存在连续与起点相同数值4个内容，若存在，则该颜色执棋者胜利，本部分较为基础，则不给出具体代码，胜利检查效果如图5-1。

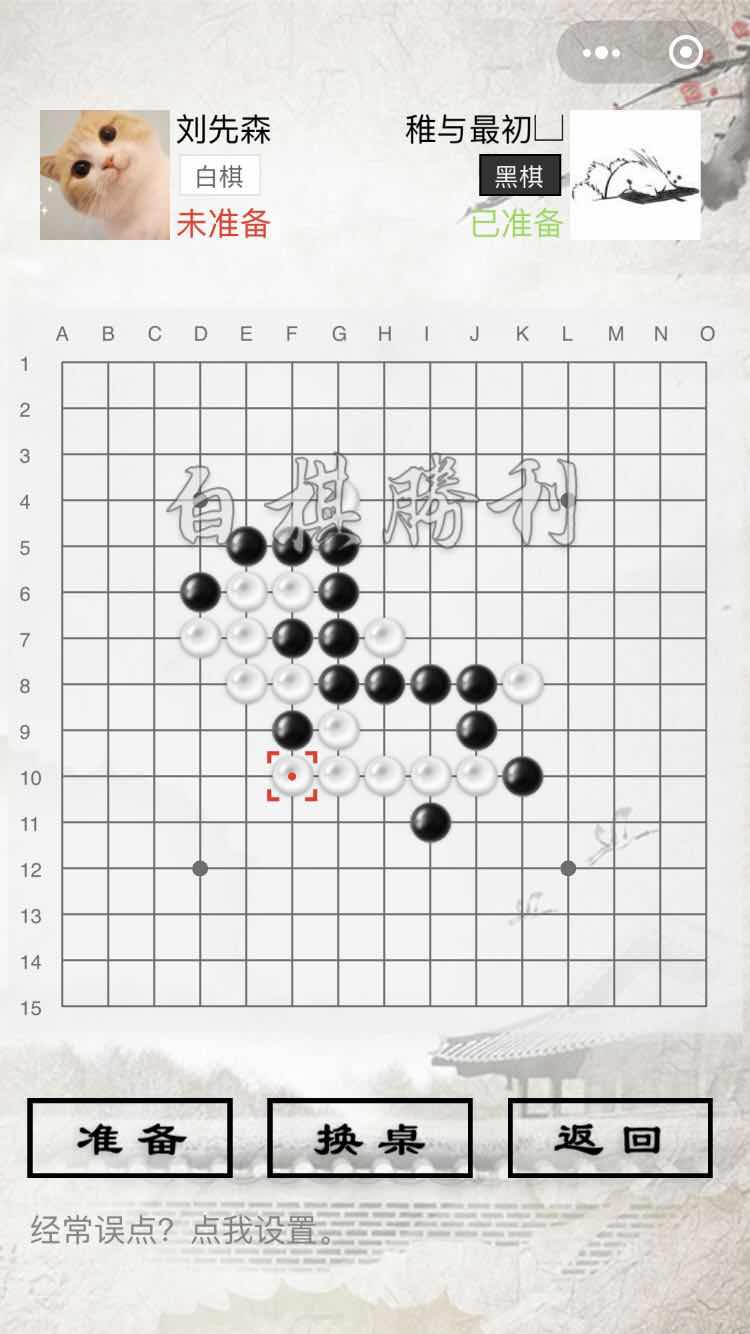


图5-1 胜利检查效果

1. 操作权限控制

玩家在实际对弈过程中，有多种可操作内容，例如执棋、悔棋、议和，而有些情况下，有些操作是被限制使用，或者无法使用。比如对方执棋期间，我方无法落子，第一步是不允许悔棋的等等，如图5-2。

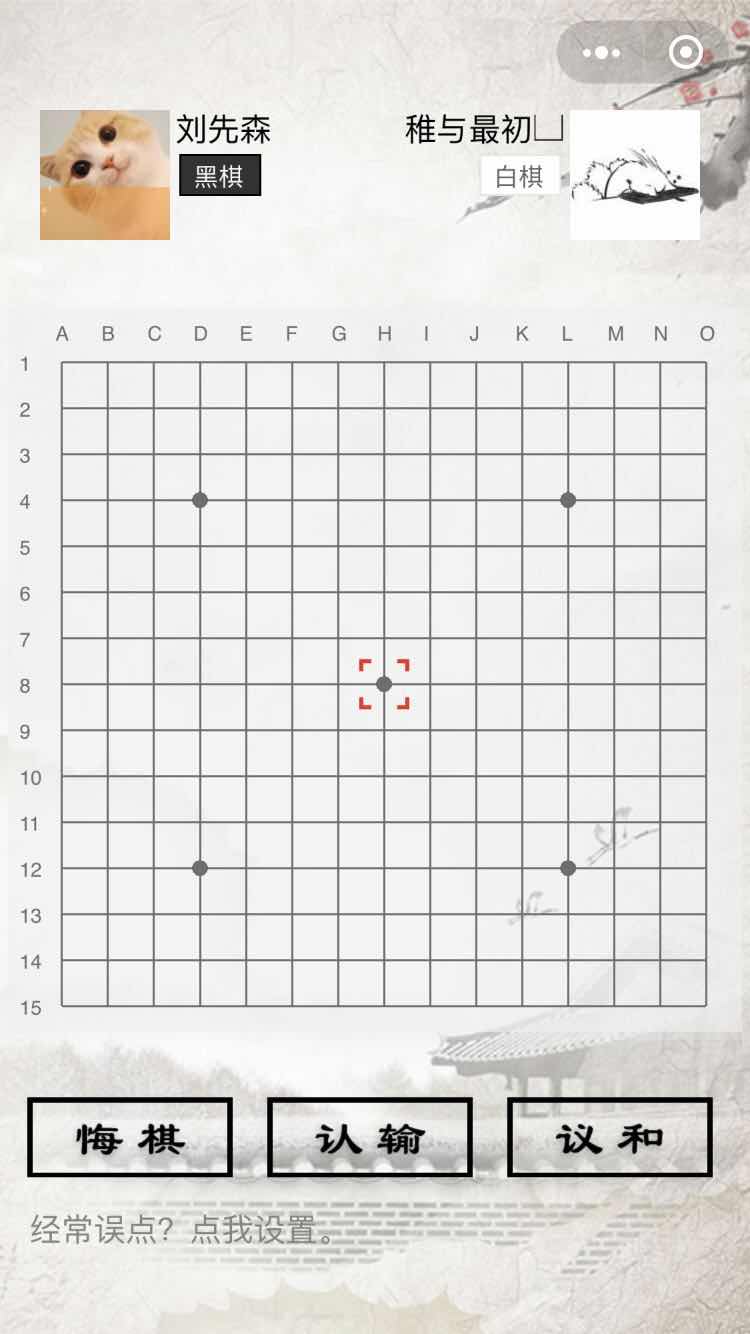


图5-2 操作权限控制

因此，本地的核心处理类除了判断一场游戏的胜负外，更重要的是当用户在申请做一些操作时，检查当前状态是否有必要向服务器发出申请，使得服务器接收到的动作指令通常为有效指令。

### 断线场景恢复

HTML5提供的Web Socket提供的是通讯协议，不适用于游戏模块的直接调用，同时游戏对于网络请求的要求通常会变的比较高，好的游戏设计需要处理如果游戏因为信号、网络变化等因素突然离线，在用户设备的网络恢复中，本地软件应做好游戏场景的恢复措施。游戏场景恢复存在两种情况，第一种情况为游戏网络的短暂变化，即游戏客户端并未关闭，这种情况时，仅需要做好消息队列的确认回执，网络再次重连接时，向服务器拉取本地最后一次处理好的消息ID，服务器以此补充即可。另一种情况为游戏客户端彻底关闭，消息队列无记录，此时将获得服务器留存在的本局中全部消息，客户端根据消息列表逐一执行，快速跟进至最新进度以完成场景的恢复。

对于所有消息的进入存放入MessageList中，每个消息均存在全局消息ID，其结构可以是：

class Message {

this.ID = ''; //全局ID

this.Data=""; //消息原Data

this.Timestamp=""; //抵达时间戳

this.finished=false; //处理标识

}

对断线事件做监听处理：。

this.WebSocket = wx.connectSocket({

url: Config.MatchServerIP,

});

this.WebSocket.onClose(this.\_OnErr.bind(this));

this.WebSocket.onError(this.\_OnErr.bind(this));

如果断线将立即尝试循环重连，通过循环标识用以用户主动放弃重连：

\_OnErr(){

//因为JavaScript是单线程程序，不能使用While作为循环条件，而使用异步递归方式

var conn = function(){

//进行重连

this.connect({

success:function(){

//成功，开始推送本地已经成功处理的最后一条消息全局ID

},

fail:function(){

if(this.continue){ //重试标识

conn();

}}});}}

## 社区服务

游戏配套发布了信息小程序，该小程序将通过用户授权的方式共享小游戏的部分数据，其具体功能如下几点。

(1) 热门文章

为了建设更佳全面的游戏社区，信息系统客户端首页为用户提供了由系统管理编辑并发布的五子棋相关教学文章，使得用户在访问时将获得更加完整的产品体验，效果如图5-3所示。



图5-3 五子棋首页

1. 历史对弈与复盘

每一场游戏对局结束后，游戏服务器将会对一场游戏所产生的数据进行上报，由Web服务器获得游戏数据并进行相关的记录，该记录包括用户的个人游戏数据变化、历史对弈的局面复盘，可供用户进行学习交流，历史记录如图5-4所示。



图5-4 历史对弈

单击对应记录可进入复盘模式，如图5-5所示，拖动下方进度条可对局面进行复盘。

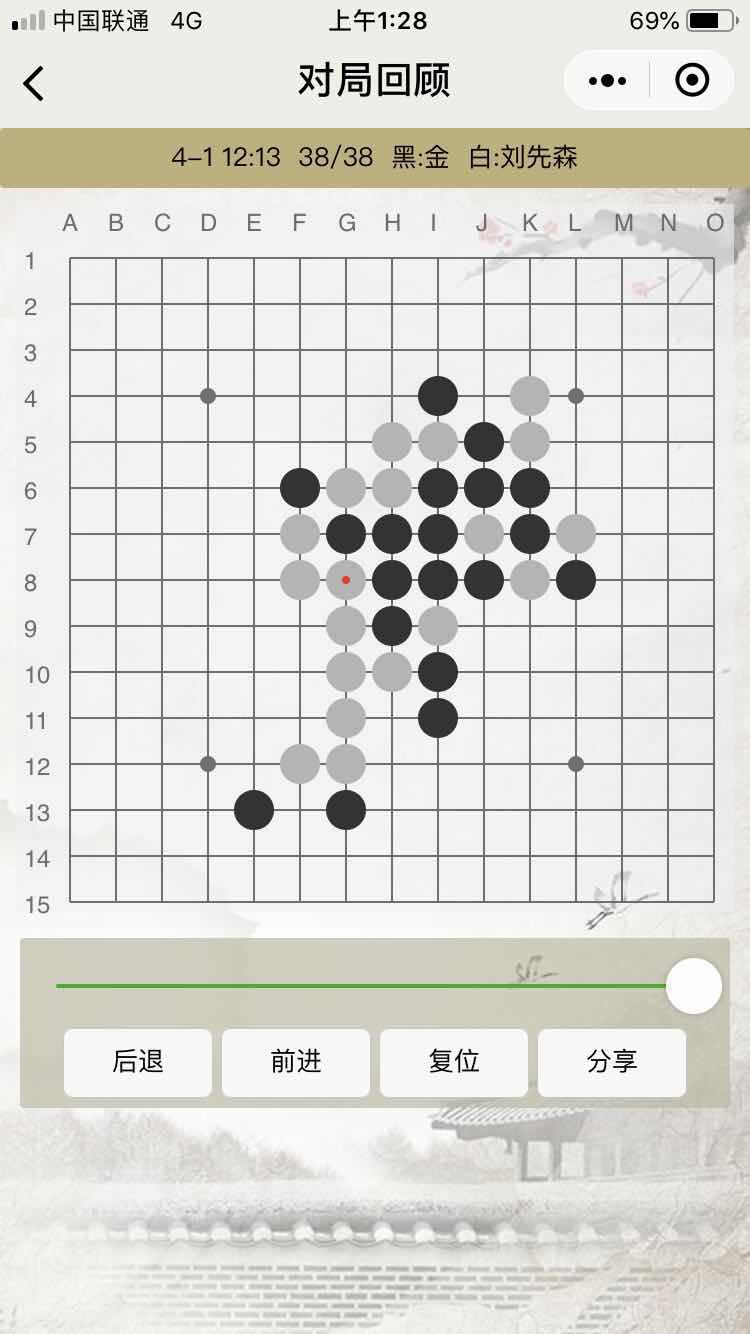


图5-5 复盘模式

## 游戏服务器设计

游戏服务器主要承担玩家对弈过程中游戏的实时交互，因此该服务器在物理结构上对于内存与处理器有较高的要求，同时具备以下几点功能。

### 开机认证

游戏服务器只提供游戏过程中的实时用户交互行为。因此，该服务器具备了更加灵活的部署方案，可以随着不同时期用户的需求对服务器数量的增加与减少来满足业务需求，同时可以降低运营成本。

在任意一台可允许公网访问的服务器上安装JVM后运行对战服务软件，将给出如图5-6所示启动配置。



图5-6 配置游戏服务器

线上游戏需要使用SSL安全证书，并默认占用443端口，在WebServerAPI配置给出信息系统API地址，用于完成系统的自我认证，认证提交关键代码：

public class GAME {

private PlayerManager playerManager;

private Match matchServer;

private Game gameServer;

public boolean Start(){

try{

playerManager = new PlayerManager();

matchServer = new Match();

gameServer = new Game();

//配置适配器

playerManager.SetAdapterMatch(matchServer.GetAdapterMatch());

matchServer.SetAdapterGameServer(gameServer.GetAdapterGameServer());

//开启接待

playerManager.Start();

}catch(Exception e){

e.printStackTrace();

return false;

}

return true;

}

... //省略其他非关键代码

}

### 建立Web Socket连接

成功开启服务后，将持续等待用户的链接，用户采用Web Socket连接方式，这与Socket请求稍有不同，在普通Socket连接中，客户机仅需要成功发送连接请求，通常而言自动完成连接的建立，而Web Socket连接中，需要完成一次双方的握手，服务器需要对客户机完成一次特殊的回应，该过程具体实现将由三步骤完成：

（1）获得请求头并解析出SecWebSocketKey。

使用正则表达式获得请求头中SecWebSocketKey后的随机密钥，其Java下关键代码为：

Pattern p = Pattern.compile("^(Sec-WebSocket-Key:).+",

Pattern.CASE\_INSENSITIVE | Pattern.MULTILINE);

Matcher m = p.matcher(req);

if(m.find()){

String foundstring = m.group();

return foundstring.split(":")[1].trim();

}

（2）计算出回执签名。

对解析出的SecWebSocketKey拼接GUID，再对其进行SHA-1加密，所得密文为回执签名，签名过程：

private String getSecWebSocketAccept(String key) throws Exception{

String guid = "258EAFA5-E914-47DA-95CA-C5AB0DC85B11";

key += guid;

MessageDigest md = MessageDigest.getInstance("SHA-1");

md.update(key.getBytes("utf-8"),0,key.length());

byte[] sha1Hash = md.digest();

Encoder base = Base64.getEncoder();

key = base.encodeToString(sha1Hash);

return base.encodeToString(sha1Hash);

}

（3）拼接到请求消息中Sec-WebSocket-Accept字段中回执给客户机，完成回执。

将计算出的回执签名拼接在通用相应头格式中，标准的响应头内容如下：

String response = "HTTP/1.1 101 Switching Protocols\r\nUpgrade:websocket\r\n Connection:Upgrade\r\nSec-WebSocket-Accept:"+getSecWebSocketAccept(secKey)+"\r\n\r\n";

### 数据上报

一局游戏的结束，游戏服务器需要将本局产生的一些信息上报至应用服务器，由应用服务器完成用户数据的变更及留存，具体包括本次对局胜负带来的用户游戏成绩的变动，本次对局的复盘数据的留存等，将所有需要上报的数据按照JSON格式进行格式化，进行数据上报，其关键代码：

public void UploadGameInfo(int Aid,int Bid,int AColor,int Result,int total,long time,String data,int type){

try{

JSONObject json = new JSONObject();

json.put("playerA", Aid); //A的id

json.put("playerB", Bid); //B的id

json.put("black", AColor); //A的执棋颜色 0A黑 1A白

json.put("result", Result); //获胜结果 0 黑胜利 1 白胜利 2 平

json.put("total", total); //执棋数量

json.put("time", time); //持续时间

json.put("data", data); //棋盘数据

json.put("type", type); //对弈类型

String JSON = json.toString();

System.out.println(JSON);

String Param = "settle="+JSON;

ThreadPool.GetPool().submit(new UploadThread(Param));

}catch(Exception e){}

}

## 实际应用情况

游戏目前已经顺利在微信小游戏平台运营，微信给出的用户数据截止2019年6月1日本作品运行的10个月中累计为96.5万用户提供游戏服务，如图5-7所示。



图5-7 微信小程序平台给出的应用使用数据

1. 结 论

设计并实现了基于分布式架构的五子棋网络游戏软件。游戏客户端基于微信小游戏平台发布，提供了单人对弈及联机对弈两种模式。单人对弈模式下的AI棋力满足非专业五子棋选手的对弈需求，联机模式通过自动选择胜率相近的对手实现用户之间的公平对弈。软件同时配套了社区服务，为玩家提供获取学习资料以及相互交流的平台。游戏后台使用PHP语言开发了信息服务系统，使用Java语言开发游戏服务器功能。游戏服务器采用分布式集群架构，提供了游戏服务器弹性增减及动态负载分配的工作机制，可在保证游戏服务稳定运行的同时有效控制运营成本，达到课题预期设计目标。

本文主要设计工作取得如下结论：

(1) 设计并实现了一种基于HTML5 Canvas的场景交互及画面渲染引擎，提供了关于场景重绘、元素拾取、动画渲染的内置功能和API接口，可以快速处理微信小游戏前端的交互动作和场景渲染，降低开发工作量。

(2) 实践验证了一种分布式架构的后台服务器部署方案，提供了服务器集群动态扩展和负载分配的工作机制，可在多数网络小游戏中作为典型模式使用。

(3) 所采用的五子棋AI算法在棋力与落子计算时间方面取得了较好的平衡，与同类游戏软件和五子棋专业训练软件的对战胜率分别达到93%和43%。

课题后续的重点研究内容包括：

(1) 游戏引擎在元素拾取时，可以引入更为复杂的异型区域识别算法，同时在引擎的渲染效率上仍可进一步优化。对于超出窗口视野的元素，可以跳过绘制或采取局部渲染处理。

(2) 目前采用的基于博弈树的五子棋AI算法无法保证绝对胜利，且一旦出现败局则存在一定概率以相同的方式再次被击败。可通过引入机器学习的技术方法，使AI通过对弈局数的增加通过自我学习逐步提高棋力，避免相同错误的再次发生。

参考文献

1. 赵剑. HTML5动画引擎技术的研究与实现[D]. 北京:北京邮电大学, 2014.
2. 张卫国. 基于HTML5的2D动画的设计与实现[D]. 广州:中山大学, 2014.
3. Roberto Y, Eduardo M, Sergio I.A System for Selecting Camera Views in Live Broadcasting of Sport Events Using a Dynamic 3D Model[J]. Multimed Tools and Applications, 2015, 74(4): 4059–4090.
4. Wentao L. Applied Information Technology in Graphics Algorithm Implementation Based on the Web Canvas[J]. Advanced Materials Research, 2014, 908:543-546.
5. 贾赛赛. 基于HTML5的游戏引擎研究与设计[D]. 绵阳:西南科技大学, 2014.
6. 王妍.动画自动生成系统中运动规划及其路径规划的研究与实现[D].北京:北京工业大学,2009.
7. 许登元,王鹏.分布式实时游戏服务器集群架构设计[J].微电子学与计算机,2014,31(7):105-112
8. 云淼. 实时网络游戏中关键技术的研究[D]. 北京: 北京工业大学.
9. 唐伟桥. 分布式网页游戏服务器端架构技术研究与实现[D]. 成都:西南交通大学.
10. 郑培铭,何丽. 基于计算机博弈的五子棋AI设计[J].电脑知识与技术,2016,12(33):80-82.
11. 刘瑞. 五子棋人工智能算法设计与实现[D]. 广州: 华南理工大学.
12. 姜勇. 五子棋人机对战系统设计[D]. 成都: 电子科技大学.
13. 李红, 吴粉侠, 刘小豫. 博弈树搜索算法研究[J]. 长春工程学院学报, 2007, 8(2):59-62.
14. 张聪品,刘春红,徐久成.博弈树启发式搜索的α-β剪枝技术研究[J].计算机工程与应用,2008,44(16):54-55.
15. Plamenka B, Milena L. Efficiency of Parallel Minimax Algorithm for Game Tree Search[J]. International Conference on Computer Systems and Technologies,2007,2(7):1-6.
16. Takahisa I, Tomoyuki K. Enhancements in Monte Carlo Tree Search Algorithms for Biased Game Trees[C]. IEEE Conference on Computational Intelligence and Games:Taiwan, 2015:43-50.

在学研究成果

1. 基于微信小游戏的五子棋Pro软件(V1.0)，软件著作权登记(登记号：2018SR448030)，第一著作权人 。
2. 基于分布式集群架构的在线五子棋微信小游戏，2018年中国大学生计算机设计大赛一等奖 ，项目负责人。

# 致 谢

感谢邵中老师在毕业设计阶段对我的细心帮助和指导，以负责的态度带我从最基础的分析论文结构开始做起，建立基本概念，再到对问题进行全面调研，展开深入了解，使我不会在设计的过程中迷失方向。这次项目经历使我技术上有了一定的提高，对微信小程序的设计有了更深的理解。这也是我第一个正式发布的项目，为我今后的创业打下了基础。

同时感谢其他各位老师对我的毕业设计提出的建议，感谢我的同学们在毕业设计过程中提供了技术上的帮助。感谢母校的领导和老师们在大学期间给予我的知识和教导，令我为今后的挑战做好了准备。感谢舍友和同学们，谢谢你们与我相伴，共同走过这美好的大学生活。