

《Java程序设计》

课程大作业

（ 2023 / 2024学年 第二学期）

|  |  |
| --- | --- |
| 专业 |  |
| 学生学号 |  |
| 学生姓名 |  |
| 指导教师 |  |
| 完成日期 | 2024年 6月 3日 |

目 录

[第一章 绪论 1](#_Toc168343305)

[1.1 课题背景及意义 1](#_Toc168343306)

[1.2 论文章节介绍 1](#_Toc168343307)

[第二章 开发环境及其主要技术 2](#_Toc168343308)

[2.1 开发技术 2](#_Toc168343309)

[2.1.1 网络编程技术 2](#_Toc168343310)

[2.1.2 人工智能技术 2](#_Toc168343311)

[2.1.3 用户界面设计技术 2](#_Toc168343312)

[2.2 开发工具 3](#_Toc168343313)

[2.2.1 集成开发环境（IDE） 3](#_Toc168343314)

[2.2.2 Java开发工具包（JDK） 3](#_Toc168343315)

[2.2.3 用户界面组件库 3](#_Toc168343316)

[第三章 分析与总设计 4](#_Toc168343317)

[3.1 需求分析 4](#_Toc168343318)

[3.2 功能及技术需求 4](#_Toc168343319)

[3.2.1 功能需求 4](#_Toc168343320)

[3.2.2 技术需求 4](#_Toc168343321)

[3.3 总体设计 5](#_Toc168343322)

[3.3.1 事件响应流程 5](#_Toc168343323)

[3.3.2 总体设计及UML图 6](#_Toc168343324)

[第四章 详细设计 7](#_Toc168343325)

[4.1 功能模块设计 7](#_Toc168343326)

[4.1.1 模块设计 7](#_Toc168343327)

[4.1.2 类、接口设计 7](#_Toc168343328)

[4.1.3 美术资源 7](#_Toc168343329)

[4.2 功能模块图 8](#_Toc168343330)

[4.3 重要功能描述 8](#_Toc168343331)

[4.3.1 Computer AI 8](#_Toc168343332)

[4.4.2 JudgeWinner 10](#_Toc168343333)

[4.4.3 PCChessBoard 10](#_Toc168343334)

[第五章 测试与分析 12](#_Toc168343335)

[5.1 测试样例 12](#_Toc168343336)

[5.1.1 人机对战测试 12](#_Toc168343337)

[5.2.2 双人对战测试 14](#_Toc168343338)

[5.2 缺陷与改进 15](#_Toc168343339)

[5.2.1 缺陷识别 15](#_Toc168343340)

[5.2.2 改进措施 15](#_Toc168343341)

[5.2.3 持续改进 15](#_Toc168343342)

[结束语 16](#_Toc168343343)

[附 录 17](#_Toc168343344)

[1. 项目结构 17](#_Toc168343345)

[2. 项目源码 17](#_Toc168343346)

[JudgeWinner.java 17](#_Toc168343347)

[Chess.java 19](#_Toc168343348)

[Computer.java 19](#_Toc168343349)

[Coord.java 24](#_Toc168343350)

[timerThread.java 24](#_Toc168343351)

[NetTool.java 25](#_Toc168343352)

[ChessBoard.java 25](#_Toc168343353)

[MainBoard.java 27](#_Toc168343354)

[PCChessBoard.java 29](#_Toc168343355)

[PCMainBoard.java 31](#_Toc168343356)

[PPChessBoard.java 32](#_Toc168343357)

[PPMainBoard.java 34](#_Toc168343358)

[SelectMenu.java 36](#_Toc168343359)

[WinDialog.java 37](#_Toc168343360)

# 第一章 绪论

## 1.1 课题背景及意义

在当今数字时代，五子棋作为一种古老而受欢迎的棋类游戏，已经成功地融入了现代技术的怀抱。本项目采用Java语言开发了一款具有局域网联机和人机对战功能的五子棋游戏。Java不仅因其跨平台的特性而被广泛应用于游戏开发，同时其健壯的网络功能也为实现游戏的局域网联机模式提供了强大支持。

局域网联机功能允许玩家在同一网络环境中与其他玩家进行实时对弈，带来了更为直接和互动的游戏体验。而在人机对战方面，我们采用了基于评分函数（Evaluation Function）的五子棋AI算法算法来优化游戏的人工智能。这个算法通过计算每个空位置的潜在得分来决定下一步落子的位置，大大提高了AI的决策效率和质量，使得电脑对手更具挑战性，同时也保证了游戏的流畅性和响应速度。

通过结合这些先进技术，我们的五子棋游戏不仅提供了传统棋艺的乐趣，还增添了现代科技的刺激元素。无论是与朋友进行的激烈对弈，还是与经过精心设计的AI进行的策略挑战，玩家都将在这款游戏中获得独特而丰富的体验。

五子棋的源代码已经开源到Github：<https://github.com/SUC-DriverOld/gobang-java>，并且可以到Releases页面下载打包好的程序，一键运行游戏，体验五子棋的乐趣。

## 1.2 论文章节介绍

本论文主要包括以下几个章节：

第一章 绪论：本章介绍了课题的背景和意义，以及论文的主要研究内容和结构安排。通过绪论部分，读者可以了解本课题的研究动机和重要性。

第二章 开发环境及其主要技术：本章详细描述了开发过程中使用的技术和工具。包括网络编程技术、人工智能技术和用户界面设计技术等开发技术，以及集成开发环境（IDE）、Java开发工具包（JDK）和用户界面组件库等开发工具。

第三章 分析与总设计：本章进行需求分析和总体设计。首先介绍了系统的功能需求和技术需求，然后描述了系统的总体设计，包括事件响应流程和UML图等。

第四章 详细设计：本章深入探讨各个功能模块的设计，包括模块设计、类和接口设计以及美术资源的设计，并提供功能模块图和重要功能的详细描述。

第五章 测试与分析：本章介绍了系统的测试过程，包括人机对战测试和双人对战测试，并对测试结果进行分析。此外，还讨论了系统中存在的缺陷以及相应的改进措施和持续改进的计划。

结束语及附录：总结了本学期的学习以及论文的主要内容，提供了项目的结构和源码，以供参考和进一步研究。

# 第二章 开发环境及其主要技术

## 2.1 开发技术

实现五子棋游戏核心功能需要一些复杂的技术。这些技术将覆盖网络编程、人工智能算法、用户界面设计等多个方面，以确保我们能够提供一个高效、稳定且用户友好的五子棋游戏平台。

### 2.1.1 网络编程技术

局域网联机对战是本游戏的一大亮点，为此我们需要掌握和应用相关的网络编程技术。

1. Java网络编程：Java提供了丰富的网络编程库和API，例如java.net包，包含了Socket编程的基本类。通过Socket编程，我们可以实现客户端和服务器端之间的数据通信，确保玩家能够在局域网内进行实时对战。
2. 多线程技术：为了保证多名玩家同时对战的流畅性和稳定性，我们需要使用多线程技术来处理并发连接。
3. 网络优化：网络稳定性对于在线游戏至关重要。通过使用UDP协议，我们能够提供可靠的连接，同时，我们会进行网络数据包的优化和压缩，以减少网络延迟，提高游戏的响应速度和流畅性。

### 2.1.2 人工智能技术

为了实现具有挑战性和多样性的AI对战模式，我们需要使用高效的人工智能算法。

1. 评分函数算法：五子棋的AI核心在于评分函数算法。该算法评估每个可能的落子点的得分，并选择得分最高的点进行落子。评分函数会考虑多种因素，例如连珠的数量、进攻和防守的平衡等，从而提高AI的决策质量。
2. 极大极小算法（Minimax）和α-β剪枝：为了增强AI的智能性，我们将使用极大极小算法来模拟对手的可能走法，并通过α-β剪枝技术来减少搜索空间，加速决策过程。这些算法能够让AI在考虑更多步数的情况下，保持较快的响应速度。
3. 机器学习：我们还可以引入一些机器学习技术，通过分析大量对局数据，不断优化和改进AI的策略，使其能够适应不同水平的玩家。

### 2.1.3 用户界面设计技术

用户界面的友好性是提升用户体验的关键因素之一，为此我们使用了以下技术：

1. Java Swing：为了实现一个直观且美观的用户界面，我们将使用Java Swing技术。Java Swing提供了丰富的图形和控件库，能够帮助我们创建动态且交互性强的游戏界面。
2. 用户交互优化：我们会进行多次用户测试和反馈，优化用户操作流程和界面元素的布局，使游戏更易于上手，并且在视觉上具有吸引力。

通过以上技术的综合应用，我们将能够实现一个功能丰富、技术先进且用户体验优秀的五子棋游戏平台。这些技术不仅支持了核心功能的实现，还为未来的功能扩展和性能优化打下了坚实的基础。

## 2.2 开发工具

为了确保五子棋游戏的开发过程高效顺利，我们选择了一套成熟且高效的开发环境。该环境包括集成开发环境（IDE）、Java开发工具包（JDK）、以及用于构建用户界面的组件库。以下是详细的开发环境介绍：

### 2.2.1 集成开发环境（IDE）

我们选择使用IntelliJ IDEA作为主要的集成开发环境。IntelliJ IDEA是一款由JetBrains开发的Java集成开发环境，具有以下优点：

1. 智能代码辅助：IDEA提供了强大的代码补全、代码导航和重构工具，能够显著提高开发效率和代码质量。
2. 调试功能：IDEA提供了强大的调试工具，包括断点调试、变量监控、表达式求值等，帮助开发者快速定位和解决问题。
3. 版本控制集成：IDEA支持多种版本控制系统，方便代码管理。

### 2.2.2 Java开发工具包（JDK）

我们使用JDK8作为开发的基础工具包。JDK8是Java的一个稳定版本，提供了丰富的类库和工具，适合进行各种应用的开发。

### 2.2.3 用户界面组件库

在用户界面的开发方面，我们使用Swing组件库。Swing是Java标准库的一部分，提供了构建图形用户界面（GUI）的丰富组件。其主要优势包括：

1. 跨平台性：Swing是纯Java实现的GUI库，可以在所有支持Java的平台上运行，具有良好的跨平台性。
2. 丰富的组件：Swing提供了丰富的GUI组件，如按钮、文本框、表格、树等，可以满足各种应用的需求。
3. 高度可定制：Swing组件支持高度的定制，可以通过设置属性、继承组件类、重写绘制方法等方式，实现复杂的用户界面。
4. 事件驱动编程：Swing支持事件驱动编程模型，通过事件监听器处理用户的操作，使得应用程序具有良好的交互性。

# 第三章 分析与总设计

## 3.1 需求分析

首先，我们需要确定五子棋游戏的目标用户群体。这将涉及到年龄段、技能水平、游戏偏好等因素。例如，游戏是否主要面向儿童，还是要吸引成年爱好者？用户是否倾向于休闲游戏，还是寻求具有一定难度和竞争性的体验？

在深入分析用户群体的特征和偏好后，我们对五子棋游戏的核心功能有了更明确的认识。这些功能不仅体现了技术的先进性，也考虑到了用户体验的多样性。

首先，局域网联机对战是游戏的一大亮点。它允许玩家在同一网络环境下进行实时对弈，这不仅增加了游戏的互动性和竞技性，还为玩家提供了一个社交平台，让他们能够在游戏中结识志同道合的朋友。此外，考虑到网络稳定性对于在线游戏至关重要，我们将特别优化网络代码，确保连接的稳定性和流畅性，从而提供无缝的游戏体验。

对于人机对战模式，我们使用了高效的评分函数算法来优化AI的决策过程。这种算法不仅提高了AI的响应速度，还增强了其策略多样性和挑战性，能够适应各种技能水平的玩家。无论玩家是刚入门的新手还是经验丰富的老手，都能在游戏中找到适合自己的难度级别，并从中获得乐趣和成长。

我们非常重视用户界面的友好性。我们的目标是创建一个既美观又直观的界面，使得游戏易于上手，同时也兼具趣味性和专业感。界面设计会考虑到不同年龄段和不同背景的用户，确保其简洁性和美观性。

通过这些精心设计的核心功能，我们的五子棋游戏不仅提供了传统游戏的乐趣，还融入了现代技术的创新元素，使其成为不同用户群体都能享受的互动平台。

## 3.2 功能及技术需求

### 3.2.1 功能需求

根据用户群体的特征和偏好，我们可以确定游戏的核心功能。这包括但不限于：

1. 局域网联机对战：提供玩家在同一网络环境下进行实时对弈的功能。
2. 人机对战模式：使用高效的基于评分函数的五子棋AI算法，评估每个可能的落子点的得分，并选择得分最高的点进行落子。让AI成为你的棋友。
3. 用户界面友好性：确保游戏界面直观易用，便于新手和非技术用户操作。

### 3.2.2 技术需求

除了功能需求外，还需考虑实现这些功能所需的技术需求。例如：

1. 网络编程：掌握和应用Java在网络编程方面的能力，实现稳定的局域网联机。
2. 人工智能算法：应用评分算法等先进技术，提高AI的效率和对战质量。

## 3.3 总体设计

本五子棋游戏系统采用经典的MVC（Model-View-Controller）架构，将应用程序分为三个核心部分：

Model（模型）：负责游戏的逻辑处理和数据维护。

View（视图）：负责展示游戏的用户界面。

Controller（控制器）：处理用户输入，协调视图和模型。

### 3.3.1 事件响应流程

事件响应流程：玩家选择任意模式并开始游戏后，程序首先等待玩家操作，当玩家执行落子操作后，程序根据玩家落子的位置更新棋盘状态，然后判断玩家是否获胜。若是，则显示获胜信息，然后游戏结束；若不是，则AI进行操作或者对手进行操作后，玩家继续操作。直到产生获胜者为止。

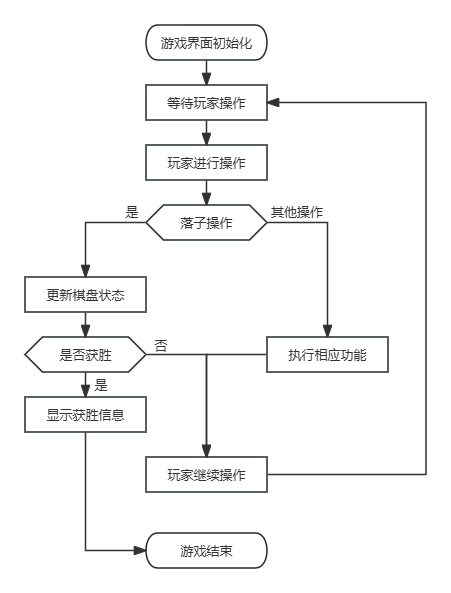


图 1五子棋事件响应流程图

### 3.3.2 总体设计及UML图

游戏主要分为初始化，设置，开始，过程，结束这五大部分组成，各部分的具体内容如下：

初始化游戏：加载视图组件，建立网络连接（如果是联机游戏）。

游戏设置：选择对战模式（人机或联机）。

游戏开始：玩家交替落子，系统通过Control模块处理游戏逻辑。

游戏过程：Model模块监测游戏状态，View模块更新用户界面。

游戏结束：宣布获胜方，提供重新开始或退出游戏的选项。

|  |  |
| --- | --- |
|  |  |
| 图 2 简化的人机对战流程图 | 图 3 局域网联机对战流程图 |

# 第四章 详细设计

## 4.1 功能模块设计

### 4.1.1 模块设计

1. Control Module（控制模块）：

负责决策获胜者，包含JudgeWinner类。

1. Net Module（网络模块）：

处理局域网内的多玩家连接、数据同步和通信协议，包含NetTool类。

1. Model Module（模型模块）：

包含Chess（棋子）、Computer（电脑对手）、Coord（坐标系统）、timerThread（计时器线程）等关键类。

1. View Module（视图模块）

负责渲染游戏界面，包括ChessBoard（棋盘）、MainBoard（主面板）、PCChessBoard（人机对决逻辑）、PCMainBoard（人机对决面板）、PPChessBoard（双人对决逻辑）、PPMainBoard（双人对决面板）、SelectMenu（模式选择面板）、WinDialog（获胜弹窗）等组件。

### 4.1.2 类、接口设计

JudgeWinner：用于实时决策获胜者。

Chess：代表棋子，拥有属性如类型（黑或白）等。

Computer：AI对手的实现，包含评分算法。

Coord：表示棋盘上的坐标，用于确定棋子位置。

timerThread：计时器线程，用于跟踪玩家的思考时间。

ChessBoard：棋盘类，负责显示棋盘和棋子，处理落子逻辑。

MainBoard：游戏的主界面，包含其他视图组件。

PCChessBoard：处理人机对战模式下的棋盘逻辑。

PCMainBoard：人机对战模式的棋盘界面。

PPChessBoard：处理双人对战模式下的棋盘逻辑。

PPMainBoard：双人对战模式的棋盘界面。

SelectMenu：启动界面，用来选择模式。

WinDialog：获胜后的弹窗页面，用户选择重新开始或者返回主界面。

### 4.1.3 美术资源

美术资源存放于images文件夹内，包含五子棋黑白棋子，透明棋盘，棋盘底色，主菜单页面这些资源的图片。程序读取并加载这些美术资源。

## 4.2 功能模块图

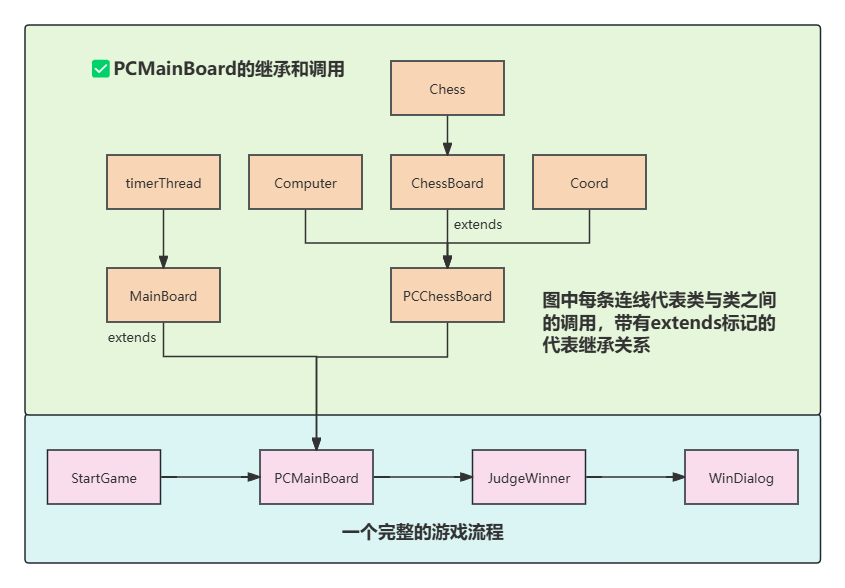


图 4 人机对战的功能模块图

## 4.3 重要功能描述

本部分我将挑选一些重要的功能进行详细的描述。包括类，方法，函数的调用以及实现逻辑，算法等。描述过程中仅提供关键代码和函数，完整的代码请参照附录。

### 4.3.1 Computer AI

五子棋AI的主要功能是通过一套评分机制来评估当前棋盘上每一个空位的得分，进而选择一个最优的位置进行落子。评分机制考虑了棋子连续的数量以及是否被对方棋子阻挡等因素，旨在选择一个对己方最有利的位置。这种方法可以快速计算出一个合理的落子位置，使AI能够在五子棋游戏中进行较为智能的对弈。具体来说，通过以下步骤实现这个功能：

**1. 遍历棋盘：**

遍历19x19的棋盘，对于每一个空位（即棋盘上没有棋子的地方），计算该位置的分数。

**2. 分数计算：**

基础分数：考虑位置的基本情况（例如边缘位置）进行初步评分。

方向性分数：在八个方向上（上下左右和四个对角线方向）分别计算连子数量。根据棋子的连续性和是否被对方棋子阻挡的情况，给出该位置的评分。

**3. 评分机制：**

基础评分：根据棋盘位置的基础得分，边缘位置得分较低。

单子：如果一个位置在某方向上有一个棋子，得分+10。

活二：如果一个位置在某方向上有两个连续的棋子且没有被对方棋子阻挡，得分较高，白棋+200，黑棋+400。

冲二：如果一个位置在某方向上有两个连续的棋子，但被对方棋子阻挡一端，得分+50。

双活二：如果一个位置在某方向上有两个连续的棋子且在多个方向上都有两个连续的棋子且没有被对方棋子阻挡，得分+84000。

活三：如果一个位置在某方向上有三个连续的棋子且没有被对方棋子阻挡，白棋得分+85000，黑棋得分+86000。

冲三：如果一个位置在某方向上有三个连续的棋子，但被对方棋子阻挡一端，白棋得分+300，黑棋得分+1000。

活四：如果一个位置在某方向上有四个连续的棋子且没有被对方棋子阻挡，白棋得分+90000，黑棋得分+91000。

冲四：如果一个位置在某方向上有四个连续的棋子，但被对方棋子阻挡一端，得分+87000。

五连：如果一个位置在某方向上有五个连续的棋子，得分+100000。

表1 评分函数的评分机制



**4. 选择最佳位置：**

对于每一个空位，分别计算白棋和黑棋的得分，存储在everyPlayerPointScore和everyComputerPointScore数组中。

使用findBestPos\_medium方法找到最高得分的位置，并选择该位置作为下一步的落子位置。

**5.输出结果：**

最终返回计算得出的最佳落子位置coord。

### 4.4.2 JudgeWinner

JudgeWinner类中的PPJudge(int x, int y, int[][] chess, int role)函数用于判断当前玩家是否在五子棋游戏中获胜。方法通过检查当前棋子所在位置的八个方向（上下左右和四个对角线方向）是否有连续的五个相同角色的棋子来判断胜负。如果找到连续五个相同角色的棋子，返回当前玩家的角色编号；否则返回0，表示没有玩家获胜。一共有以下五种判断方法：

① 竖直方向判断；

② 水平方向判断；

③ 右倾斜方向判断；

④ 左倾斜方向判断；

⑤ 无玩家获胜。

表2 PPJudge判断获胜者方法



### 4.4.3 PCChessBoard

PCChessBoard类继承自ChessBoard类，表示电脑对战模式下的五子棋棋盘面板。它包含了玩家和电脑的棋子操作、胜负判断、悔棋、以及游戏结束处理等功能。

表3 PCChessBoard类成员变量表



1. setResult(int result)：该方法用于设置游戏的当前结果。result 值为 1 时表示游戏进行中，0 表示游戏结束。
2. PCChessBoard(PCMainBoard mb)：构造函数初始化棋盘面板，设置玩家角色为白棋，并初始化电脑对象和获胜对话框。
3. saveStep(int posX, int posY)：该方法记录每一步棋的坐标，将当前步数加 1，并保存坐标到 step 数组中。
4. backStep()：该方法用于悔棋，回退上两步的棋子，并将棋盘上的相应位置重置为空。
5. WinEvent(int winner)：该方法处理游戏结束事件，根据获胜者不同，显示相应的获胜对话框并执行后续操作（如重新开始游戏或回到菜单）。
6. mousePressed(MouseEvent e)：该方法处理鼠标按下事件，负责玩家下棋及电脑下棋的操作流程：
7. 获取鼠标点击位置并转换为棋盘坐标。
8. 如果点击位置没有棋子，玩家下棋并保存位置。
9. 判断玩家是否获胜，若获胜，显示获胜信息并结束游戏。
10. 若玩家未获胜，电脑下棋并保存位置。
11. 判断电脑是否获胜，若获胜，显示获胜信息并结束游戏。
12. 若电脑未获胜，继续游戏，设置棋盘为可点击状态。

# 第五章 测试与分析

## 5.1 测试样例

本节中，我将以截图的形式分别测试人机对战和双人对战（简略介绍），并给出一定的操作步骤。

### 5.1.1 人机对战测试

1. 双击Setup\_Gobang\_java\_v1.1.1.exe安装或从源码SelectMenu.java启动五子棋。



图 6 主界面

2. 点击“棋室挑战”进入人机对决模式。

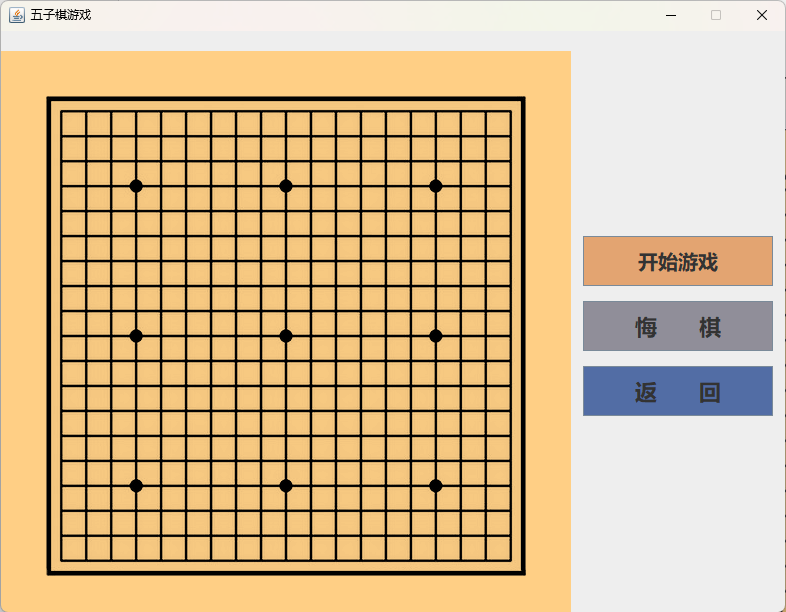


图 7 人机对决开始游戏界面

3. 点击开始游戏，开始落子，此时计时器启动并且“开始游戏”变成正在游戏。

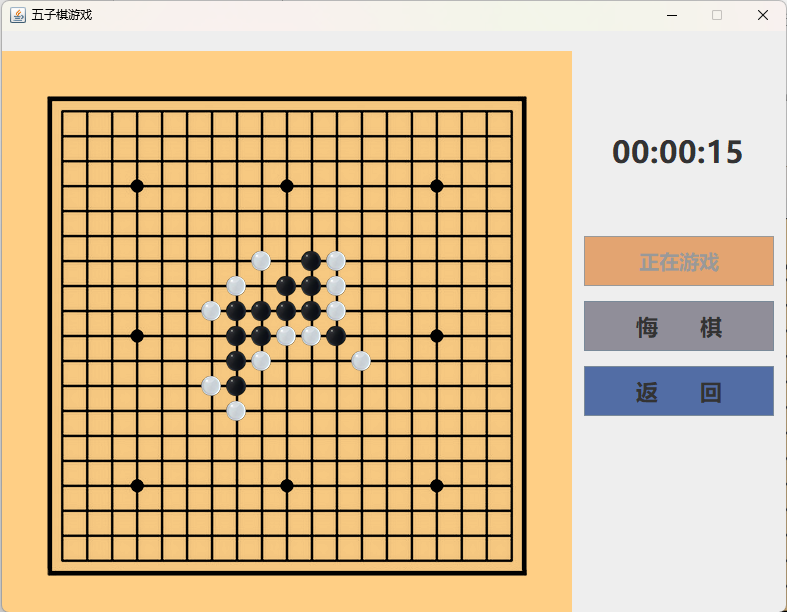


图 8 开始下棋

4. 点击悔棋进行撤销。

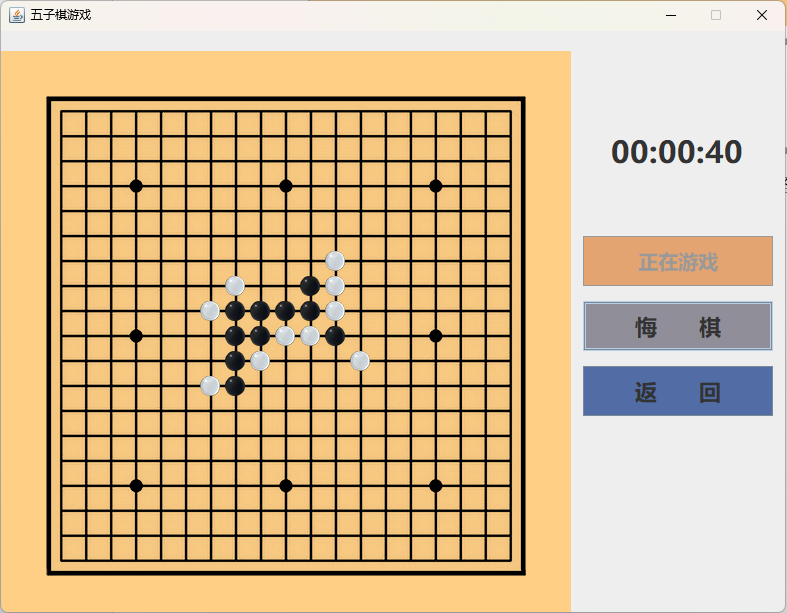


图 9 悔棋样例

5. 一方获胜后弹出获胜窗口。此时用户可以选择再来一把或退出到主页面。

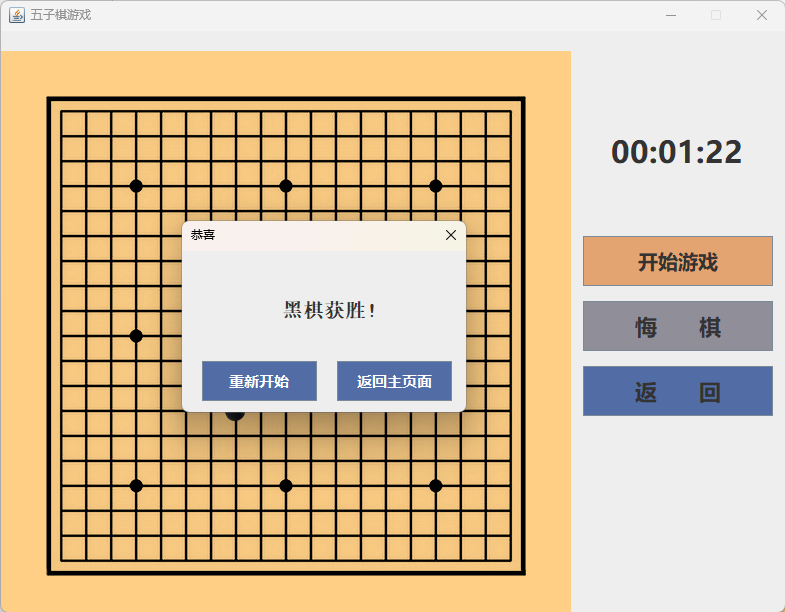


图 10 获胜窗口

### 5.2.2 双人对战测试

1. 在主页面中选择“人人对战”进入局域网双人对战模式。输入对方IP地址进行连接，等待双方都连接后，白棋先手下棋。

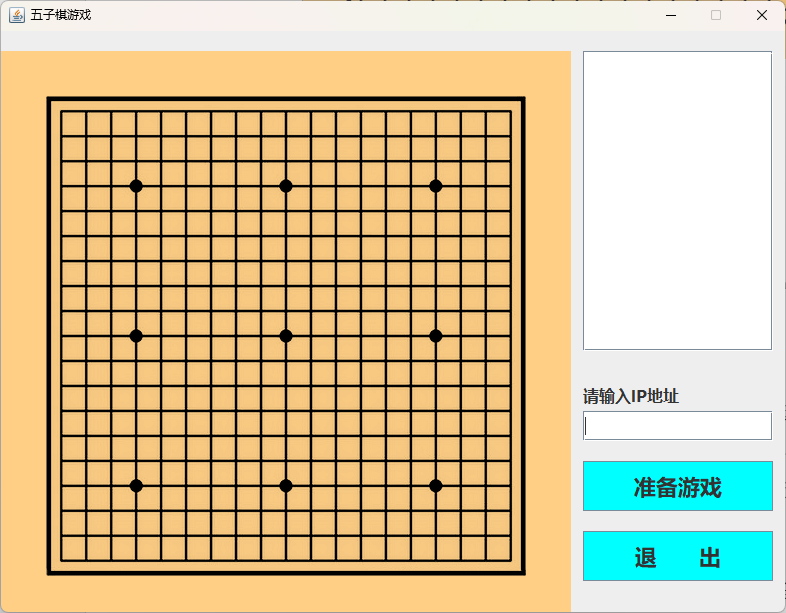


图 11 双人对战模式

## 5.2 缺陷与改进

在软件开发的生命周期中，识别缺陷并提出改进措施是至关重要的步骤，它有助于提升产品的质量和用户体验。针对五子棋游戏，我们可以通过测试、用户反馈和性能监控来识别潜在的缺陷，并根据这些信息来规划改进策略。

### 5.2.1 缺陷识别

游戏平衡性：AI难度可能过于困难，对于一些新手玩家不太友好。

网络延迟：局域网对战中可能存在网络延迟，影响游戏流畅度。

用户界面：用户界面可能不够直观，新用户可能难以理解某些功能。

系统兼容性：游戏可能在某些操作系统或设备上表现不佳。

### 5.2.2 改进措施

动态难度调整：引入更为复杂的算法，根据玩家的游戏表现动态调整AI的难度。

网络优化：改进网络代码，使用更有效的数据传输协议，减少延迟。

界面优化：进行用户测试，收集反馈，并根据这些信息优化用户界面，使其更加友好。

兼容性测试：在多种操作系统和硬件上进行广泛测试，确保游戏的兼容性和性能。

### 5.2.3 持续改进

软件开发是一个持续的过程。随着技术的发展和用户需求的变化，五子棋游戏也需要不断地进行更新和改进。可以设置定期的回顾会议，不断检视产品性能，根据用户反馈进行迭代开发。

用户反馈：定期收集和分析用户反馈，以便快速响应用户需求。

性能监控：实施性能监控机制，以便及时发现并解决性能瓶颈。

技术迭代：跟踪最新的技术趋势，评估并采用新技术以提升游戏性能和用户体验。

安全更新：保持对安全漏洞的警觉，定期进行安全更新和修补。

通过实施这些改进措施，我们能够确保五子棋游戏在未来能持续吸引和满足玩家，同时也保持其在竞争激烈的游戏市场中的地位。

# 结束语

随着这一学期Java课程的结束，我站在了一个重要的节点上，回顾所学知识和完成的大作业——五子棋游戏。这段旅程不仅加深了我们对Java编程语言的理解，而且通过实际项目的完成，我得以将理论知识运用于实践，体验从构思到实现的完整软件开发周期。

Java课程涵盖了基础语法、面向对象编程、异常处理、文件IO、网络编程以及多线程处理等多个方面。通过系统的学习，我掌握了Java的核心概念，包括但不限于类与对象、继承与多态、接口与抽象类。此外，通过探索Java的API和各种框架，我学会了如何高效地利用已有资源来解决复杂的编程问题。

在完成五子棋游戏的过程中，我获得了宝贵的实战经验。通过这个项目，我深入理解了MVC架构模式，这对于软件设计来说是一个非常宝贵的资产。实现局域网对战和人机对战功能时，我不仅提升了网络编程能力，还学会了如何将算法与游戏逻辑相结合，尤其是应用剪枝算法来优化AI的性能。

我也遇到了挑战，如性能优化、用户界面设计、跨平台兼容性等，这些问题迫使我们思考如何编写更加高效、更加用户友好的代码。通过不断的测试和迭代，我学会了如何分析和解决问题，以及如何收集用户反馈并将其转化为实际的改进措施。

这一课程和大作业不仅仅是关于编程技能的提升，更是关于解决问题、项目管理的学习。我学会了如何将一个复杂的问题分解为可管理的小部分，并且在过程中不断自我完善。这些体会将在我未来的学习和职业生涯中发挥重要作用。

Java课程和五子棋游戏的大作业是我们编程能力成长的一个缩影。它们不仅提高了我的技术技能，也锻炼了我的逻辑思维、创新思维和解决问题的能力。当我继续前进，在技术的道路上探索新的高度时，这些经验将作为我宝贵的资产。

# 附 录

## 1. 项目结构

Gobang/src

├─Control

│ JudgeWinner.java

├─images

│ black.png

│ gobang.png

│ lace.png

│ menu.png

│ white.png

├─Model

│ Chess.java

│ Computer.java

│ Coord.java

│ timerThread.java

├─Net

│ NetTool.java

└─View

ChessBoard.java

MainBoard.java

PCChessBoard.java

PCMainBoard.java

PPChessBoard.java

PPMainBoard.java

SelectMenu.java

WinDialog.java

## 2. 项目源码

为获得更好的代码阅读体验，项目源码强烈建议前往Github仓库阅读，地址如下：<https://github.com/SUC-DriverOld/gobang-java>

### JudgeWinner.java

package Control;

public class JudgeWinner {

/\*\*

\* 判断当前玩家是否获胜

\*

\* @param x 当前棋子所在的横坐标

\* @param y 当前棋子所在的纵坐标

\* @param chess 当前棋盘的状态，19x19的二维数组

\* @param role 当前玩家的角色（例如1代表黑子，2代表白子）

\* @return 如果当前玩家获胜，则返回角色编号；如果没有玩家获胜，则返回0

\*/

public static int PPJudge(int x, int y, int[][] chess, int role) {

int startX = x, startY = y, count = 1;

// 竖直方向上判断输赢

// 向上判断

while (true) {

y--;

// 检查棋子是否在棋盘内且与当前角色一致

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else {

break;

}

}

y = startY; // 恢复原始位置

// 向下判断

while (true) {

y++;

// 检查棋子是否在棋盘内且与当前角色一致

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else {

break;

}

}

// 判断是否形成5个连续的棋子

if (count >= 5) {

return role;

}

// 水平方向判断输赢

x = startX;

y = startY;

count = 1;

// 向左判断

while (true) {

x--;

// 检查棋子是否在棋盘内且与当前角色一致

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else {

break;

}

}

x = startX; // 恢复原始位置

// 向右判断

while (true) {

x++;

// 检查棋子是否在棋盘内且与当前角色一致

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else {

break;

}

}

// 判断是否形成5个连续的棋子

if (count >= 5) {

return role;

}

// 右倾斜方向判断输赢

x = startX;

y = startY;

count = 1;

// 向左上判断

while (true) {

y--;

x--;

// 检查棋子是否在棋盘内且与当前角色一致

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else {

break;

}

}

x = startX;

y = startY; // 恢复原始位置

// 向右下判断

while (true) {

x++;

y++;

// 检查棋子是否在棋盘内且与当前角色一致

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else {

break;

}

}

// 判断是否形成5个连续的棋子

if (count >= 5) {

return role;

}

// 左倾斜方向判断输赢

x = startX;

y = startY;

count = 1;

// 向左下判断

while (true) {

x--;

y++;

// 检查棋子是否在棋盘内且与当前角色一致

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else {

break;

}

}

x = startX;

y = startY; // 恢复原始位置

// 向右上判断

while (true) {

x++;

y--;

// 检查棋子是否在棋盘内且与当前角色一致

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else {

break;

}

}

// 判断是否形成5个连续的棋子

if (count >= 5) {

return role;

}

// 如果没有玩家获胜，返回0

return 0;

}

}

### Chess.java

package Model;

public class Chess {

// 定义黑白棋子

public static int WHITE = 1;

public static int BLACK = 2;

// 定义无棋子

public static int NO\_CHESS = 0;

}

### Computer.java

package Model;

public class Computer {

// 记录当前的分数

private int scores;

// 当前选定的坐标

private Coord coord = new Coord();

// 每个玩家点的评分

private final int[][] everyPlayerPointScore = new int[19][19];

// 每个电脑点的评分

private final int[][] everyComputerPointScore = new int[19][19];

/\*\*

\* 计算下棋的位置

\*

\* @param chess 棋盘的当前状态，19x19的二维数组

\* @return 计算出的最佳下棋位置

\*/

public Coord computePos(int[][] chess) {

// 根据当前棋盘状态计算最佳下棋位置

countMaxLines\_medium(chess, Chess.WHITE);

return coord;

}

/\*\*

\* 找出分数最大的坐标

\*

\* @param chess 棋盘数组

\* @param role 当前角色（白棋还是黑棋）

\*/

public void countMaxLines\_medium(int[][] chess, int role) {

Coord playerCoord = new Coord();

Coord computerCoord = new Coord();

int x, y;

// 遍历棋盘每一个点

for (x = 0; x < 19; x++) {

for (y = 0; y < 19; y++) {

// 如果该位置没有棋子

if (chess[x][y] == Chess.NO\_CHESS) {

// 计算该位置对玩家和电脑的评分

countEveryPos\_medium(x, y, chess, role);

everyPlayerPointScore[x][y] = scores;

countEveryPos\_medium(x, y, chess, Chess.BLACK);

everyComputerPointScore[x][y] = scores;

} else {

// 如果有棋子，分数为0

everyPlayerPointScore[x][y] = 0;

everyComputerPointScore[x][y] = 0;

}

}

}

// 找到评分最高的点

if (findBestPos\_medium(everyPlayerPointScore, playerCoord) >= findBestPos\_medium (everyComputerPointScore, computerCoord)) {

coord = playerCoord;

} else {

coord = computerCoord;

}

}

/\*\*

\* 找到最大分数点的坐标

\*

\* @param a 存储每个点的分数的数组

\* @param c 保存最大分数点的坐标

\* @return 最大的分数值

\*/

public int findBestPos\_medium(int[][] a, Coord c) {

int i, j, max = 0;

// 遍历找到最大分数的点

for (i = 0; i < 19; i++) {

for (j = 0; j < 19; j++) {

if (a[i][j] > max) {

max = a[i][j];

c.setX(i);

c.setY(j);

}

}

}

return max;

}

/\*\*

\* 根据连子数量和对方棋子的阻挡情况进行评分

\*

\* @param count 当前方向上连续相同颜色棋子的数量

\* @param i 阻挡当前方向的对方棋子数量

\* @param countTwo 两边有一个空位的两个连在一起的棋子数量

\* @param role 当前角色

\*/

public void mark\_medium(int count, int i, int countTwo, int role) {

if (count == 1) {

scores += 10;

} else if (count == 2 && i == 0 && role == Chess.WHITE && countTwo <= 1) {

scores += 200;

} else if (count == 2 && i == 0 && role == Chess.BLACK && countTwo <= 1) {

scores += 400;

} else if (count == 2 && i == 1) {

scores += 50;

} else if (count == 2 && i == 0 && countTwo > 1) {

scores += 84000;

System.out.println("双活二" + scores);

} else if (count == 3 && i == 0 && role == Chess.WHITE) {

scores += 85000;

} else if (count == 3 && i == 0 && role == Chess.BLACK) {

scores += 86000;

} else if (count == 3 && i == 1 && role == Chess.WHITE) {

scores += 300;

System.out.println("白子冲三" + scores);

} else if (count == 3 && i == 1 && role == Chess.BLACK) {

scores += 1000;

System.out.println("黑子冲三" + scores);

} else if (count == 4 && i == 0 && role == Chess.WHITE) {

scores += 90000;

} else if (count == 4 && i == 0 && role == Chess.BLACK) {

scores += 91000;

} else if (count == 4 && i == 1) {

scores += 87000;

} else if (count == 5) {

scores += 100000;

}

}

/\*\*

\* 根据坐标位置增加基础分数

\*

\* @param x 横坐标

\* @param y 纵坐标

\*/

public void basicScore(int x, int y) {

if (x == 0 || y == 0) {

scores += 3;

} else {

scores += 5;

}

}

/\*\*

\* 计算每个位置的分数

\*

\* @param x 当前坐标的横坐标

\* @param y 当前坐标的纵坐标

\* @param chess 当前棋盘的状态

\* @param role 当前角色

\*/

public void countEveryPos\_medium(int x, int y, int[][] chess, int role) {

scores = 0;

basicScore(x, y);

int matchRole = (role == Chess.BLACK) ? Chess.WHITE : Chess.BLACK;

int startX = x, startY = y;

int count = 0, up = 0, down = 0, left = 0, right = 0;

int leftUp = 0, leftDown = 0, rightUp = 0, rightDown = 0, countTwo = 0;

// 竖直方向上判断

while (true) {

y--;

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else if ((x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == matchRole) || y < 0) {

up++;

break;

} else {

break;

}

}

y = startY;

while (true) {

y++;

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else if ((x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == matchRole) || y > 19) {

down++;

break;

} else {

break;

}

}

if (count == 2 && (up + down == 0)) {

countTwo++;

}

mark\_medium(count, up + down, countTwo, role);

// 水平方向判断

x = startX;

y = startY;

count = 0;

while (true) {

x--;

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else if ((x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == matchRole) || x == 0) {

left++;

break;

} else {

break;

}

}

x = startX;

while (true) {

x++;

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else if ((x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == matchRole) || x > 19) {

right++;

break;

} else {

break;

}

}

if (count == 2 && (left + right == 0)) {

countTwo++;

}

mark\_medium(count, left + right, countTwo, role);

// 右倾斜方向判断

x = startX;

y = startY;

count = 0;

while (true) {

y--;

x--;

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else if ((x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == matchRole) || x < 0 || y < 0) {

leftUp++;

break;

} else {

break;

}

}

x = startX;

y = startY;

while (true) {

x++;

y++;

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else if ((x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == matchRole) || x > 19 || y > 19) {

rightDown++;

break;

} else {

break;

}

}

if (count == 2 && (leftUp + rightDown == 0)) {

countTwo++;

}

mark\_medium(count, leftUp + rightDown, countTwo, role);

// 左倾斜方向判断

x = startX;

y = startY;

count = 0;

while (true) {

x--;

y++;

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else if ((x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == matchRole) || x < 0 || y > 19) {

leftDown++;

break;

} else {

break;

}

}

x = startX;

y = startY;

while (true) {

x++;

y--;

if (x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == role) {

count++;

} else if ((x >= 0 && x < 19 && y >= 0 && y < 19 && chess[x][y] == matchRole) || x > 19 || y < 0) {

rightUp++;

break;

} else {

break;

}

}

if (count == 2 && (leftDown + rightUp == 0)) {

countTwo++;

}

mark\_medium(count, leftDown + rightUp, countTwo, role);

}

}

### Coord.java

package Model;

/\*\*

\* Coord 类用于表示棋盘上的一个坐标点。

\*/

public class Coord {

// 坐标的横坐标（列）

private int x;

// 坐标的纵坐标（行）

private int y;

/\*\*

\* 设置坐标的横坐标值。

\*

\* @param x 新的横坐标值

\*/

public void setX(int x) {

this.x = x;

}

/\*\*

\* 设置坐标的纵坐标值。

\*

\* @param y 新的纵坐标值

\*/

public void setY(int y) {

this.y = y;

}

/\*\*

\* 获取坐标的横坐标值。

\*

\* @return 当前的横坐标值

\*/

public int getX() {

return x;

}

/\*\*

\* 获取坐标的纵坐标值。

\*

\* @return 当前的纵坐标值

\*/

public int getY() {

return y;

}

}

### timerThread.java

package Model;

import javax.swing.\*;

/\*\*

\* timerThread 类用于在 JLabel 上显示计时器。该线程会定期更新 JLabel 显示的时间。

\*/

public class timerThread extends Thread {

// 用于显示时间的 JLabel 组件

private final JLabel label;

/\*\*

\* 构造函数，初始化 timerThread 实例。

\*

\* @param label 用于显示时间的 JLabel 组件

\*/

public timerThread(JLabel label) {

this.label = label;

}

/\*\*

\* 线程的运行方法，启动计时器并定期更新 JLabel 上的时间显示。

\*/

public void run() {

// 记录计时器开始的时间

long startTime = System.currentTimeMillis();

// 持续更新时间，直到线程被中断

do {

// 获取当前时间

long currentTime = System.currentTimeMillis();

// 计算经过的时间

long time = currentTime - startTime;

// 格式化经过的时间并更新 JLabel 的文本

label.setText(format(time));

} while (!this.isInterrupted());

}

/\*\*

\* 格式化经过的时间，将毫秒转换为 HH:mm:ss 格式的字符串。

\*

\* @param elapsed 经过的时间，单位为毫秒

\* @return 格式化后的时间字符串，格式为 HH:mm:ss

\*/

private String format(long elapsed) {

int hour, minute, second;

// 将毫秒转换为秒

elapsed = elapsed / 1000;

// 计算秒

second = (int) (elapsed % 60);

elapsed = elapsed / 60;

// 计算分钟

minute = (int) (elapsed % 60);

elapsed = elapsed / 60;

// 计算小时

hour = (int) (elapsed % 60);

// 返回格式化后的时间字符串

return String.format("%02d:%02d:%02d", hour, minute, second);

}

}

### NetTool.java

package Net;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.net.MulticastSocket;

/\*\*

\* NetTool 类提供网络工具方法，用于发送 UDP 广播消息。

\*/

public class NetTool {

/\*\*

\* 发送 UDP 广播消息。

\*

\* @param ip 目标 IP 地址

\* @param msg 要发送的消息

\*/

public static void sendUdpBroadCast(String ip, String msg) {

try {

// 创建一个多播套接字

MulticastSocket ms = new MulticastSocket();

// 获取目标 IP 地址

InetAddress ia = InetAddress.getByName(ip);

// 创建一个数据报包，包含消息的字节数组，指定目标地址和端口

DatagramPacket dp = new DatagramPacket(msg.getBytes(), 0, msg.length(), ia, 1111);

// 通过多播套接字发送数据报包

ms.send(dp);

// 关闭多播套接字

ms.close();

} catch (Exception e) {

// 捕获并打印异常

e.printStackTrace();

}

}

}

### ChessBoard.java

package View;

import Model.Chess;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

/\*\*

\* 自定义棋盘面板类，继承自 JPanel，并实现 MouseListener 接口，用于绘制五子棋棋盘和处理鼠标事件。

\*/

public class ChessBoard extends JPanel implements MouseListener {

// 游戏结束标志

public static final int GAME\_OVER = 0;

// 棋盘的列数

public static final int COLS = 19;

// 棋盘的行数

public static final int RAWS = 19;

// 游戏结果标志，初始为 1，表示游戏进行中

public int result = 1;

// 白棋图片

public Image whiteChess;

// 黑棋图片

public Image blackChess;

// 棋盘图片

public Image chessBoardImage;

// 边框装饰图片

public Image laceImage;

// 棋子的横坐标

public int chessX;

// 棋子的纵坐标

public int chessY;

// 棋盘上的隐形坐标数组，每个小区域代表一个数组元素

public int[][] chess = new int[COLS][RAWS];

// 是否可点击的标志

public int clickable;

/\*\*

\* 构造函数，初始化棋盘的图片，并初始化数组。

\*/

public ChessBoard() {

chessBoardImage = Toolkit.getDefaultToolkit().getImage("src/images/gobang.png");

whiteChess = Toolkit.getDefaultToolkit().getImage("src/images/white.png");

blackChess = Toolkit.getDefaultToolkit().getImage("src/images/black.png");

laceImage = Toolkit.getDefaultToolkit().getImage("src/images/lace.png");

initArray();

addMouseListener(this);

}

/\*\*

\* 设置棋盘是否可点击。

\*

\* @param clickable 是否可点击的标志

\*/

public void setClickable(int clickable) {

this.clickable = clickable;

}

/\*\*

\* 初始化棋盘数组，将数组中的值全部设为 Chess.NO\_CHESS。

\*/

public void initArray() {

for (int i = 0; i < COLS; i++) {

for (int j = 0; j < RAWS; j++) {

chess[i][j] = Chess.NO\_CHESS;

}

}

}

/\*\*

\* 重写 paintComponent 方法，绘制棋盘和棋子。

\*

\* @param g 该参数是绘制图形的句柄

\*/

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g);

g.drawImage(laceImage, 0, 0, null);

g.drawImage(chessBoardImage, 35, 35, null);

for (int i = 0; i < COLS; i++) {

for (int j = 0; j < RAWS; j++) {

if (chess[i][j] == Chess.BLACK) {

g.drawImage(blackChess, 60 + i \* 25 - 11, 62 + j \* 25 - 12, null);

} else if (chess[i][j] == Chess.WHITE) {

g.drawImage(whiteChess, 60 + i \* 25 - 11, 62 + j \* 25 - 12, null);

}

}

}

}

@Override

public void mouseClicked(MouseEvent e) {

}

@Override

public void mousePressed(MouseEvent e) {

}

@Override

public void mouseReleased(MouseEvent e) {

}

@Override

public void mouseEntered(MouseEvent e) {

}

@Override

public void mouseExited(MouseEvent e) {

}

}

### MainBoard.java

package View;

import Model.timerThread;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

/\*\*

\* 主游戏界面类，继承自 JFrame，实现了 MouseListener 和 ActionListener 接口。

\* 用于创建五子棋游戏的主窗口和相关组件。

\*/

public class MainBoard extends JFrame implements MouseListener, ActionListener {

// 定义常量，表示是否可以点击

public static final int CAN\_CLICK\_INFO = 1;

public static final int CAN\_NOT\_CLICK\_INFO = 0;

// 显示计时的标签

public JLabel label\_timeCount;

// 计时线程

public timerThread timer;

/\*\*

\* 获取计时线程

\*

\* @return 当前的计时线程对象

\*/

public timerThread getTimer() {

return timer;

}

/\*\*

\* 获取显示时间的标签

\*

\* @return 显示时间的标签对象

\*/

public JLabel getLabel() {

return label\_timeCount;

}

/\*\*

\* 构造函数，初始化主窗口

\*/

public MainBoard() {

// 设置布局为空布局

setLayout(null);

// 设置窗口位置和大小

setBounds(300, 70, 800, 620);

// 初始化组件

init1();

// 设置窗口可见

setVisible(true);

// 设置默认关闭操作

setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

// 设置窗口不可调整大小

setResizable(false);

// 设置窗口标题

setTitle("五子棋游戏");

}

/\*\*

\* 重写 paint 方法，调用父类的 paint 方法并重新绘制组件

\*

\* @param g 该参数是绘制图形的句柄

\*/

@Override

public void paint(Graphics g) {

super.paint(g);

repaint();

}

/\*\*

\* 初始化组件，添加计时标签

\*/

public void init1() {

// 初始化计时标签

label\_timeCount = new JLabel();

// 设置字体样式和大小

label\_timeCount.setFont(new Font("Microsoft Yahei", Font.BOLD, 30));

// 设置标签位置和大小

label\_timeCount.setBounds(610, 100, 230, 40);

// 添加标签到主窗口

add(label\_timeCount);

}

/\*\*

\* ActionListener 接口的实现方法

\*

\* @param e 触发的动作事件

\*/

@Override

public void actionPerformed(ActionEvent e) {

}

/\*\*

\* MouseListener 接口的实现方法，鼠标点击事件

\*

\* @param e 触发的鼠标事件

\*/

@Override

public void mouseClicked(MouseEvent e) {

}

/\*\*

\* MouseListener 接口的实现方法，鼠标按下事件

\*

\* @param e 触发的鼠标事件

\*/

@Override

public void mousePressed(MouseEvent e) {

}

/\*\*

\* MouseListener 接口的实现方法，鼠标释放事件

\*

\* @param e 触发的鼠标事件

\*/

@Override

public void mouseReleased(MouseEvent e) {

}

/\*\*

\* MouseListener 接口的实现方法，鼠标进入事件

\*

\* @param e 触发的鼠标事件

\*/

@Override

public void mouseEntered(MouseEvent e) {

}

/\*\*

\* MouseListener 接口的实现方法，鼠标退出事件

\*

\* @param e 触发的鼠标事件

\*/

@Override

public void mouseExited(MouseEvent e) {

}

}

### PCChessBoard.java

package View;

import Control.JudgeWinner;

import Model.Chess;

import Model.Computer;

import Model.Coord;

import javax.swing.border.LineBorder;

import java.awt.\*;

import java.awt.event.MouseEvent;

/\*\*

\* 电脑对战模式下的棋盘面板，继承自 ChessBoard 类。

\*/

public class PCChessBoard extends ChessBoard {

private final int role; // 角色，表示玩家执棋的颜色

private int result = 1; // 游戏结果

private final PCMainBoard mb; // 主游戏面板

private final WinDialog dialog; // 获胜对话框

private final Computer com; // 电脑对象，用于计算电脑下棋位置

private final int[][] step = new int[30 \* 30][2]; // 记录每一步棋的坐标

private int stepCount = 0; // 记录步数

/\*\*

\* 设置游戏结果。

\*

\* @param result 游戏结果，1 表示游戏未结束，0 表示游戏结束。

\*/

public void setResult(int result) {

this.result = result;

}

/\*\*

\* 构造函数，初始化 PCChessBoard 对象。

\*

\* @param mb 主游戏面板对象

\*/

public PCChessBoard(PCMainBoard mb) {

this.mb = mb;

role = Chess.WHITE; // 玩家执白棋

com = new Computer(); // 初始化电脑对象

dialog = new WinDialog(mb, "恭喜", true); // 初始化获胜对话框

setBorder(new LineBorder(new Color(0, 0, 0))); // 设置边界

}

/\*\*

\* 保存黑白棋子的坐标到二维数组中。

\*

\* @param posX 棋子的横坐标

\* @param posY 棋子的纵坐标

\*/

private void saveStep(int posX, int posY) {

stepCount++;

step[stepCount][0] = posX;

step[stepCount][1] = posY;

}

/\*\*

\* 悔棋操作，回退上一步的棋子。

\*/

public void backStep() {

if (stepCount >= 2) {

chess[step[stepCount][0]][step[stepCount][1]] = 0;

chess[step[stepCount - 1][0]][step[stepCount - 1][1]] = 0;

stepCount = stepCount - 2;

}

}

/\*\*

\* 处理游戏结束事件，包括玩家赢和电脑赢两种情况。

\*

\* @param winner 获胜者，Chess.WHITE 表示玩家赢，Chess.BLACK 表示电脑赢。

\*/

public void WinEvent(int winner) {

// 白棋赢

if (winner == Chess.WHITE) {

result = GAME\_OVER;

try {

mb.getTimer().interrupt(); // 中断计时器线程

} catch (Exception e1) {

e1.printStackTrace();

}

mb.getBtn\_startGame().setText("开始游戏"); // 设置开始游戏按钮的文本

mb.getBtn\_startGame().setEnabled(true); // 启用开始游戏按钮

dialog.setWinnerInfo("白棋获胜!"); // 设置获胜对话框的内容

setClickable(MainBoard.CAN\_NOT\_CLICK\_INFO); // 设置不能点击棋盘

dialog.setVisible(true); // 显示获胜对话框

// 根据对话框的选择来决定下一步操作

if (dialog.getMsg() == WinDialog.BACK) {

mb.dispose();

new SelectMenu();

} else {

initArray();

}

}

// 黑棋赢

else if (winner == Chess.BLACK) {

result = GAME\_OVER;

try {

mb.getTimer().interrupt(); // 中断计时器线程

} catch (Exception e1) {

e1.printStackTrace();

}

mb.getBtn\_startGame().setText("开始游戏"); // 设置开始游戏按钮的文本

mb.getBtn\_startGame().setEnabled(true); // 启用开始游戏按钮

setClickable(MainBoard.CAN\_NOT\_CLICK\_INFO); // 设置不能点击棋盘

dialog.setWinnerInfo("黑棋获胜!"); // 设置获胜对话框的内容

dialog.setVisible(true); // 显示获胜对话框

// 根据对话框的选择来决定下一步操作

if (dialog.getMsg() == WinDialog.BACK) {

mb.dispose();

new SelectMenu();

} else {

initArray();

}

}

}

/\*\*

\* 处理鼠标按下事件，即玩家下棋操作。

\*

\* @param e 鼠标事件对象

\*/

@Override

public void mousePressed(MouseEvent e) {

int winner;

if (clickable == MainBoard.CAN\_CLICK\_INFO) { // 如果可以点击棋盘

chessX = e.getX(); // 获取鼠标点击的横坐标

chessY = e.getY(); // 获取鼠标点击的纵坐标

if (chessX < 524 && chessX > 50 && chessY < 523 && chessY > 50) { // 如果点击的位置在棋盘范围内

float x = (float) (chessX - 49) / 25;

float y = (float) (chessY - 50) / 25;

int x1 = (int) x;

int y1 = (int) y;

// 如果这个地方没有棋子

if (chess[x1][y1] == Chess.NO\_CHESS) {

chess[x1][y1] = role; // 玩家下棋

saveStep(x1, y1); // 保存下棋的坐标

setClickable(MainBoard.CAN\_NOT\_CLICK\_INFO); // 设置不能点击棋盘

winner = JudgeWinner.PPJudge(x1, y1, chess, role); // 判断游戏是否结束

WinEvent(winner); // 处理游戏结束事件

if (result != GAME\_OVER) { // 如果游戏没有结束

Coord coord = com.computePos(chess); // 计算电脑下棋位置

chess[coord.getX()][coord.getY()] = Chess.BLACK; // 电脑下棋

saveStep(coord.getX(), coord.getY()); // 保存电脑下棋的坐标

winner = JudgeWinner.PPJudge(coord.getX(), coord.getY(), chess, Chess.BLACK); // 判断游戏是否结束

WinEvent(winner); // 处理游戏结束事件

if (result != GAME\_OVER) { // 如果游戏没有结束

setClickable(MainBoard.CAN\_CLICK\_INFO); // 设置可以点击棋盘

}

}

}

}

}

}

}

### PCMainBoard.java

package View;

import Model.timerThread;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

/\*\*

\* PCMainBoard 类是五子棋游戏的主界面，继承自 MainBoard 类。

\*/

public class PCMainBoard extends MainBoard {

private PCChessBoard cb; // PCChessBoard 对象，表示五子棋棋盘

private JButton btn\_startGame; // 开始游戏按钮

private JButton btn\_back; // 悔棋按钮

private JButton btn\_exit; // 返回按钮

/\*\*

\* 获取开始游戏按钮对象

\* @return 开始游戏按钮对象

\*/

public JButton getBtn\_startGame() {

return btn\_startGame;

}

/\*\*

\* PCMainBoard 类的构造方法，初始化界面

\*/

public PCMainBoard() {

super(); // 调用父类构造方法初始化主界面

init(); // 初始化界面组件

setDefaultCloseOperation(DISPOSE\_ON\_CLOSE); // 设置默认的关闭操作

}

/\*\*

\* 初始化界面组件

\*/

public void init() {

cb = new PCChessBoard(this); // 创建 PCChessBoard 对象

cb.setClickable(PPMainBoard.CAN\_NOT\_CLICK\_INFO); // 设置初始不可点击

cb.setBounds(0, 20, 570, 585); // 设置 PCChessBoard 组件在界面中的位置和大小

cb.setVisible(true); // 设置 PCChessBoard 可见

// 创建并设置"开始游戏"按钮

btn\_startGame = new JButton("开始游戏");

btn\_startGame.setBounds(582, 205, 190, 50);

btn\_startGame.setBackground(new Color(227, 164, 113));

btn\_startGame.setFocusable(false);

btn\_startGame.setFont(new Font(Font.DIALOG, Font.BOLD, 20));

btn\_startGame.addActionListener(this);

// 创建并设置"悔棋"按钮

btn\_back = new JButton("悔 棋");

btn\_back.setBounds(582, 270, 190, 50);

btn\_back.setBackground(new Color(144, 142, 153));

btn\_back.setFocusable(false);

btn\_back.setFont(new Font(Font.DIALOG, Font.BOLD, 22));

btn\_back.addActionListener(this);

// 创建并设置"返回"按钮

btn\_exit = new JButton("返 回");

btn\_exit.setBackground(new Color(82, 109, 165));

btn\_exit.setBounds(582, 335, 190, 50);

btn\_exit.setFocusable(false);

btn\_exit.setFont(new Font(Font.DIALOG, Font.BOLD, 22));

btn\_exit.addActionListener(this);

// 将组件添加到主界面中

add(cb);

add(btn\_back);

add(btn\_startGame);

add(btn\_exit);

}

/\*\*

\* 按钮点击事件处理方法

\* @param e 事件对象

\*/

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btn\_startGame) { // 开始游戏按钮点击事件

btn\_startGame.setEnabled(false); // 禁用开始游戏按钮

btn\_startGame.setText("正在游戏"); // 设置按钮文本为"正在游戏"

cb.setClickable(CAN\_CLICK\_INFO); // 设置棋盘可点击

timer = new timerThread(label\_timeCount); // 创建并启动计时器线程

timer.start();

cb.setResult(1); // 设置游戏状态为进行中

} else if (e.getSource() == btn\_back) { // 悔棋按钮点击事件

cb.backStep(); // 调用 PCChessBoard 的悔棋方法

} else if (e.getSource() == btn\_exit) { // 返回按钮点击事件

dispose(); // 关闭当前窗口

new SelectMenu(); // 打开选择菜单窗口

}

}

}

### PPChessBoard.java

package View;

import Control.JudgeWinner;

import Model.Chess;

import Net.NetTool;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.MouseEvent;

/\*\*

\* PPChessBoard 类表示玩家对战的五子棋棋盘，继承自 ChessBoard 类。

\*/

public class PPChessBoard extends ChessBoard {

private int role; // 角色，黑色或白色

private JTextArea ta\_pos\_info; // 位置信息文本区域

private final PPMainBoard mb; // 主界面对象

private final WinDialog dialog; // 胜利对话框

/\*\*

\* 构造函数，初始化棋盘和相关组件

\* @param mb 主界面对象

\* @param dialog 胜利对话框对象

\*/

public PPChessBoard(PPMainBoard mb, WinDialog dialog) {

super(); // 调用父类构造方法初始化棋盘

this.mb = mb;

this.dialog = dialog;

setRole(Chess.WHITE); // 设置默认角色为白色

}

/\*\*

\* 设置位置信息文本区域

\* @param ta 位置信息文本区域对象

\*/

public void setInfoBoard(JTextArea ta) {

ta\_pos\_info = ta;

}

/\*\*

\* 设置棋子横纵坐标和角色

\* @param x 横坐标

\* @param y 纵坐标

\* @param r 角色，黑/白

\*/

public void setCoord(int x, int y, int r) {

if (r == Chess.WHITE) {

role = Chess.BLACK;

} else {

role = Chess.WHITE;

}

chess[x][y] = r;

if (r == Chess.WHITE) {

ta\_pos\_info.append("白棋位置为:" + x + "," + y + "\n");

} else {

ta\_pos\_info.append("黑棋位置为:" + x + "," + y + "\n");

}

int winner = JudgeWinner.PPJudge(x, y, chess, r);

WinEvent(winner);

setClickable(MainBoard.CAN\_CLICK\_INFO);

repaint();

}

/\*\*

\* 设置角色

\* @param role 角色，黑/白

\*/

public void setRole(int role) {

this.role = role;

}

/\*\*

\* 绘制棋盘图形

\* @param g 绘图对象

\*/

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g); // 调用父类方法绘制基本图形

}

/\*\*

\* 处理胜利事件

\* @param winner 胜利者

\*/

public void WinEvent(int winner) {

if (winner == Chess.WHITE) { // 白棋获胜

try {

mb.getTimer().interrupt();

} catch (Exception e1) {

e1.printStackTrace();

}

mb.getBtn\_startGame().setText("开始游戏");

mb.getBtn\_startGame().setEnabled(true);

ta\_pos\_info.append("白棋获胜\n");

result = Chess.WHITE;

dialog.setWinnerInfo("白棋获胜!");

setClickable(PPMainBoard.CAN\_NOT\_CLICK\_INFO);

dialog.setVisible(true);

if (dialog.getMsg() == WinDialog.BACK) {

mb.dispose();

new SelectMenu();

} else {

initArray();

mb.getLabel().setText(null);

ta\_pos\_info.setText(null);

}

} else if (winner == Chess.BLACK) { // 黑棋获胜

try {

mb.getTimer().interrupt();

} catch (Exception e1) {

e1.printStackTrace();

}

mb.getBtn\_startGame().setText("开始游戏");

mb.getBtn\_startGame().setEnabled(true);

ta\_pos\_info.append("黑棋获胜\n");

result = Chess.BLACK;

setClickable(MainBoard.CAN\_NOT\_CLICK\_INFO);

dialog.setWinnerInfo("黑棋获胜!");

dialog.setVisible(true);

if (dialog.getMsg() == WinDialog.BACK) {

mb.dispose();

new SelectMenu();

} else {

initArray();

mb.getLabel().setText(null);

ta\_pos\_info.setText(null);

}

}

}

/\*\*

\* 处理鼠标按下事件

\* @param e 鼠标事件对象

\*/

@Override

public void mousePressed(MouseEvent e) {

if (clickable == PPMainBoard.CAN\_CLICK\_INFO) { // 如果可以点击

chessX = e.getX();

chessY = e.getY();

if (chessX < 524 && chessX > 50 && chessY < 523 && chessY > 50) {

float x = (float) (chessX - 49) / 25;

float y = (float) (chessY - 50) / 25;

int x1 = (int) x;

int y1 = (int) y;

if (chess[x1][y1] == 0) { // 如果该位置没有棋子

chess[x1][y1] = role; // 放置棋子

if (role == Chess.WHITE) {

ta\_pos\_info.append("白棋位置为:" + x1 + "," + y1 + "\n");

} else {

ta\_pos\_info.append("黑棋位置为:" + x1 + "," + y1 + "\n");

}

NetTool.sendUdpBroadCast(mb.getIp(), "POS" + "," + x1 + "," + y1 + "," + role); // 发送位置信息

int winner = JudgeWinner.PPJudge(x1, y1, chess, role); // 判断是否有玩家获胜

WinEvent(winner); // 处理获胜事件

setClickable(MainBoard.CAN\_NOT\_CLICK\_INFO); // 设置不可点击

}

}

}

}

}

### PPMainBoard.java

package View;

import Model.timerThread;

import Net.NetTool;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

public class PPMainBoard extends MainBoard {

private PPChessBoard cb; // 对战棋盘

private JButton btn\_startGame; // 开始游戏按钮

private JButton btn\_back; // 退出按钮

private JTextField tf\_ip; // 输入IP地址文本框

private String ip; // IP地址

private DatagramSocket socket; // 数据报套接字

private String gameState; // 游戏状态

private String enemyGameState; // 对手游戏状态

/\*\*

\* 获取开始游戏按钮

\*

\* @return 开始游戏按钮对象

\*/

public JButton getBtn\_startGame() {

return btn\_startGame;

}

/\*\*

\* 获取IP地址

\*

\* @return IP地址字符串

\*/

public String getIp() {

return ip;

}

/\*\*

\* 构造函数，初始化主面板

\*/

public PPMainBoard() {

super();

label\_timeCount.setBounds(602, 330, 230, 40);

init();

setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

}

/\*\*

\* 初始化界面组件

\*/

public void init() {

gameState = "NOT\_START";

enemyGameState = "NOT\_START";

// 提示信息

JLabel help = new JLabel("请输入IP地址");

help.setBounds(582, 350, 190, 30);

help.setFont(new Font(Font.DIALOG, Font.BOLD, 16));

tf\_ip = new JTextField();

tf\_ip.setBounds(582, 380, 190, 30);

tf\_ip.addMouseListener(this);

btn\_startGame = new JButton("准备游戏");

btn\_startGame.setFocusPainted(false);

btn\_startGame.setBackground(Color.CYAN);

btn\_startGame.setFont(new Font(Font.DIALOG, Font.BOLD, 22));

btn\_startGame.addActionListener(this);

btn\_startGame.setBounds(582, 430, 190, 50);

btn\_back = new JButton("退 出");

btn\_back.setFocusPainted(false);

btn\_back.setBackground(Color.CYAN);

btn\_back.setFont(new Font(Font.DIALOG, Font.BOLD, 22));

btn\_back.addActionListener(this);

btn\_back.setBounds(582, 500, 190, 50);

JTextArea ta\_chess\_info = new JTextArea();

ta\_chess\_info.setEnabled(false);

ta\_chess\_info.setBackground(Color.WHITE);

ta\_chess\_info.setForeground(Color.black);

JScrollPane p = new JScrollPane(ta\_chess\_info);

p.setBounds(582, 20, 190, 300);

WinDialog dialog = new WinDialog(this, "恭喜", true);

cb = new PPChessBoard(this, dialog);

cb.setClickable(PPMainBoard.CAN\_NOT\_CLICK\_INFO);

cb.setBounds(0, 20, 570, 585);

cb.setVisible(true);

cb.setInfoBoard(ta\_chess\_info);

add(help);

add(tf\_ip);

add(cb);

add(btn\_startGame);

add(p);

add(btn\_back);

ReicThread();

repaint();

}

/\*\*

\* 接收信息放在线程中

\*/

public void ReicThread() {

new Thread(new Runnable() {

public void run() {

try {

byte[] buf = new byte[1024];

socket = new DatagramSocket(2222);

DatagramPacket dp = new DatagramPacket(buf, buf.length);

while (true) {

socket.receive(dp);

String data = new String(dp.getData(), 0, dp.getLength());

if (data.isEmpty()) {

cb.setClickable(CAN\_NOT\_CLICK\_INFO);

} else {

String[] msg = data.split(",");

System.out.println(msg[0]);

// 接收到对面准备信息并且自己点击了准备

if (msg[0].equals("ready")) {

enemyGameState = "ready";

System.out.println("对方已准备");

if (gameState.equals("ready")) {

gameState = "FIGHTING";

cb.setClickable(CAN\_CLICK\_INFO);

btn\_startGame.setText("正在游戏");

timer = new timerThread(label\_timeCount);

timer.start();

}

} else if (msg[0].equals("POS")) {

System.out.println("发送坐标");

// 接受坐标以及角色

cb.setCoord(Integer.parseInt(msg[1]), Integer.parseInt(msg[2]),

Integer.parseInt(msg[3]));

}

}

}

} catch (Exception e) {

e.printStackTrace();

}

}

}).start();

}

/\*\*

\* 响应按钮点击事件

\*

\* @param e 事件对象

\*/

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btn\_startGame) {

if (!tf\_ip.getText().isEmpty() &&

!tf\_ip.getText().equals("不能为空") &&

!tf\_ip.getText().equals("请输入IP地址") &&

!tf\_ip.getText().equals("不能连接到此IP")) {

ip = tf\_ip.getText();

btn\_startGame.setEnabled(false);

btn\_startGame.setText("等待对方准备");

tf\_ip.setEditable(false);

NetTool.sendUdpBroadCast(ip, "ready");

gameState = "ready";

if (enemyGameState != null && enemyGameState.equals("ready")) {

gameState = "FIGHTING";

cb.setClickable(CAN\_CLICK\_INFO);

btn\_startGame.setText("正在游戏");

timer = new timerThread(label\_timeCount);

timer.start();

}

} else {

tf\_ip.setText("不能为空");

}

} else if (e.getSource() == btn\_back) {

System.exit(0);

}

}

}

### SelectMenu.java

package View;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

/\*\*

\* SelectMenu 类代表游戏的主菜单，用户可以在此选择不同的游戏模式或退出游戏。

\*/

public class SelectMenu extends JFrame implements MouseListener, ActionListener, MouseMotionListener {

public Point pos;

/\*\*

\* 构造一个 SelectMenu 对象。

\*/

public SelectMenu() {

setCursor(new Cursor(Cursor.CROSSHAIR\_CURSOR));

setUndecorated(false);

setVisible(true);

setLayout(null);

setBounds(580, 185, 290, 420);

setResizable(false);

paintBg();

setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

addMouseListener(this);

}

/\*\*

\* 绘制主菜单的背景图片。

\*/

public void paintBg() {

ImageIcon image = new ImageIcon("src/images/menu.png");

JLabel la3 = new JLabel(image);

la3.setBounds(0, 0, this.getWidth(), this.getHeight());// 添加图片，设置图片大小为窗口的大小。

this.getLayeredPane().add(la3, new Integer(Integer.MIN\_VALUE)); // 将JLabel加入到面板容器的最高层

JPanel jp = (JPanel) this.getContentPane();

jp.setOpaque(false); // 设置面板容器为透明

}

@Override

public void mouseClicked(MouseEvent e) {}

@Override

public void mousePressed(MouseEvent e) {

int x = e.getX();

int y = e.getY();

if (x >= 65 && x <= 231 && y >= 120 && y <= 150) {

// 人机

dispose();

new PCMainBoard();

} else if (x >= 65 && x <= 231 && y >= 210 && y <= 250) {

// 人人

dispose();

new PPMainBoard();

} else if (x >= 65 && x <= 231 && y >= 325 && y <= 355) {

// 退出

System.exit(0);

}

}

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseExited(MouseEvent e) {}

public static void main(String[] args) {

new SelectMenu();

}

@Override

public void actionPerformed(ActionEvent e) {}

@Override

public void mouseDragged(MouseEvent e) {

Point newPos = new Point(getLocation().x + e.getX() - pos.x, getLocation().y + e.getY() - pos.y);

setLocation(newPos.x, newPos.y);

}

@Override

public void mouseMoved(MouseEvent e) {}

}

### WinDialog.java

package View;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

/\*\*

\* WinDialog 类表示游戏胜利时弹出的对话框，允许用户重新开始游戏或返回主菜单。

\*/

public class WinDialog extends JDialog implements ActionListener {

public static final int RESTART = 1;

public static final int BACK = 0;

private JButton btn\_restart, btn\_back;

private JLabel label;

private int msg;

/\*\*

\* 构造一个 WinDialog 对象。

\*

\* @param f 父窗口

\* @param title 对话框标题

\* @param b 模态对话框的模式

\*/

public WinDialog(JFrame f, String title, boolean b) {

super(f, title, b);

setLayout(null);

setResizable(false);

setBounds(500, 300, 300, 200);

init();

}

/\*\*

\* 获取用户的选择信息。

\*

\* @return 选择信息，1 为重新开始游戏，0 为返回主页面

\*/

public int getMsg() {

return msg;

}

/\*\*

\* 设置胜利信息。

\*

\* @param winnerInfo 胜利信息

\*/

public void setWinnerInfo(String winnerInfo) {

label.setText(winnerInfo);

}

private void init() {

btn\_restart = new JButton("重新开始");

btn\_restart.setFocusPainted(false);

btn\_restart.setBackground(Color.CYAN);

btn\_restart.addActionListener(this);

btn\_restart.setBounds(20, 110, 115, 40);

btn\_back = new JButton("返回主页面");

btn\_back.setBounds(155, 110, 115, 40);

btn\_back.setBackground(Color.CYAN);

btn\_back.addActionListener(this);

label = new JLabel();

label.setFont(new Font(Font.DIALOG\_INPUT, Font.BOLD, 20));

label.setBounds(100, 10, 100, 100);

add(btn\_restart);

add(btn\_back);

add(label);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btn\_restart) {

msg = RESTART;

setVisible(false);

} else if (e.getSource() == btn\_back) {

msg = BACK;

setVisible(false);

}

}

}