基于 Java 的五子棋人机对战平台

姓名： 黄耀
班级： 精测 0
学号： 2010010626
指导老师： 郑莉、尹龑燊

目　录

摘要.....1
一、　背景介绍.....1
1.1 五子棋游戏发展历史.....1
1.2 五子棋规则介绍.....1
1.3 五子棋专业术语.....1
二、　需求分析及开发环境.....2
2.1 需求分析.....2
2.2 开发环境.....2
三、　设计方案.....2
3.1 功能分析图.....2
3.2 类图.....3
3.3 几种典型功能.....5
四、　核心算法实现.....6
4.1 极小极大算法.....6
4.2 剪枝算法.....7
4.3 估价函数.....8
4.4 五子棋算法.....8
五、　测试分析.....10
六、　使用说明.....12
6.1 快速入门.....12
6.2 功能介绍.....13

摘要

本文简要地介绍了五子棋人机对战平台实现的基本方法，简单介绍了其基本背景，重点介绍了设计方案，较为详细地介绍了几个重要类的属性和方法，具体实现见源码。然后详细介绍了极大极小算法以及基于 alpha-beta 剪枝的极大极小博弈树算法。最后为简易版的用户手册，介绍了其基本功能以及使用方法。

关键词：五子棋、算法、智能、事件监听

一、 背景介绍

1.1 五子棋游戏发展历史

五子棋是起源于中国古代的传统黑白棋种之一，发展于日本，流行于欧美。相传起源于四千多年前的尧帝时期，比围棋的历史还要悠久，可能早在“尧造围棋”之前，民间就已有五子棋游戏。据日本史料文献记载，中国古代的五子棋先由中国传到高丽（今朝鲜），然后于公元 1688 年至 1704 年日本的元禄时代再从高丽传到日本，最初在皇宫和贵族大家庭中流行，到元禄末期，开始在民间盛行。20 世纪初传统五子棋及连珠从日本传入欧洲。

五子棋，在日文中有连五子、五子连、串珠、五目、五目碰、五格、五石、五法、五联、京棋等多种称谓，英文则称之为 FIR、、Gobang、connect 5 等。而且许多国家的人对五子棋都有不同的爱称，例如，韩国人把五子棋称为“情侣棋”，欧洲人称其为“绅士棋”，说明了五子棋深受大众的喜爱。

1.2 五子棋规则介绍

职业规则：通过采取禁手方法来实现游戏的公平性。1、第一回合先手只能下一手，其余回合可以下连续两手；2、后手每回合均可以下两手；3、准备 19*19 棋盘两张；4,、每颗棋子所投的棋盘没有限制；5、只要任意一方在两个棋盘上且同一个回合上连为五子即为胜；6、若任意一方在两个棋盘上且不同一个回合上连为五子为负；7、若任意一方在不足两个棋盘上且同一个回合上连为五子为负。

1.3 五子棋专业术语

五子棋有较多的专业术语，为便于后续算法的描述，现在总结如下：

〖着〗：在对局过程中，行棋方把棋子落在棋盘无子的点上，不论落子的手是否脱离棋子，均被视为一着。

〖成五〗含有五枚同色棋子所形成的连。

〖四〗在一条阳线或阴线上连续相邻的 5 个点上只有四枚同色棋子的棋型。

〖活四〗有两个点可以成五的四。〖冲四〗只有一个点可以成五的四。

〖死四〗不能成五的四。

〖三〗在一条阳线或阴线上连续相邻的 5 个点上只有三枚同色棋子的棋型。

〖活三〗再走一着可以形成活四的三。

〖连活三〗即：连的活三（同色棋子在一条阳线或阴线上相邻成一排的活三）。

〖跳活三〗中间隔有一个空点的活三。简称“跳三”。

〖眠三〗再走一着可以形成冲四的三。

〖死三〗不能成五的三。

〖二〗在一条阳线或阴线上连续相邻的 5 个点上只有两枚同色棋子的棋型。

〖活二〗再走一着可以形成活三的二。

〖连活二〗即：连的活二（同色棋子在一条阳线或阴线上相邻成一排的活二）。

〖跳活二〗中间隔有一个空点的活二。简称“跳二”。

〖大跳活二〗中间隔有两个空点的活二。简称“大跳二”。

〖眠二〗再走一着可以形成眠三的二。

〖死二〗不能成五的二。

〖先手〗对方必须应答的着法，相对于先手而言，冲四称为“绝对先手”。

二、 需求分析及开发环境

2.1 需求分析

五子棋作为一个棋类竞技运动，十分流行，是一种休闲娱乐益智的好方法。但随着信息化网络的迅猛发展，越来越多的人大部分工作时间都离不开计算机，为了缓解疲劳，有一个网络版的五子棋十分必要。虽然现在网络上有许多版本的五子棋平台，但是都有一些缺点。有的界面太粗糙，有的功能过于单一，只是实现简单的五子棋对战功能不具备足够的智能。因此，尽管做过五子棋版本的人不少，但是其提升空间依然很大。我作出一个尽可能完善的五子棋多功能平台，可以实现多种功能，增添多媒体功能，并对界面粗糙、算法不智能等问题进行优化。而且五子棋平台综合了图形界面、网络通信、人工智能算法等知识，是一个很好的课程设计题目，可以帮助我们锻炼 JAVA 图形界面编程以及网络通信等知识。同时，这学期我选修了人工智能课程，我也想通过人工智能算法课程的学习，对五子棋人机对战部分算法做进一步的优化处理，这也是本次大作业的一个难点。

2.2 开发环境

操作系统：Windows7

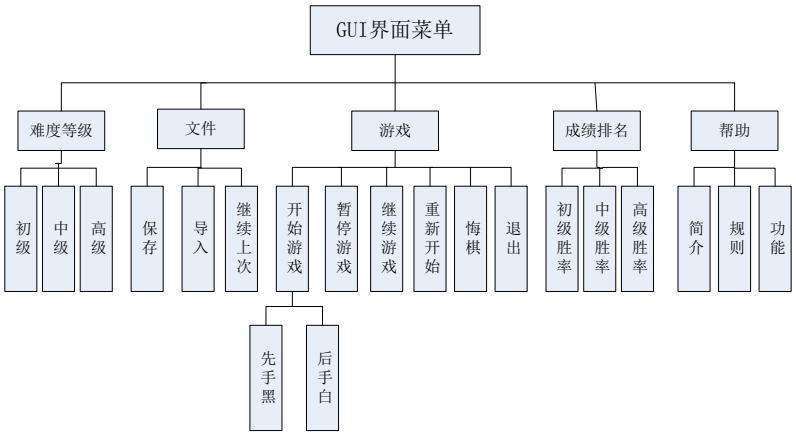
开发平台：Eclipse

开发语言：JAVA　　Java 开发包：JDK 1.7

三、 设计方案

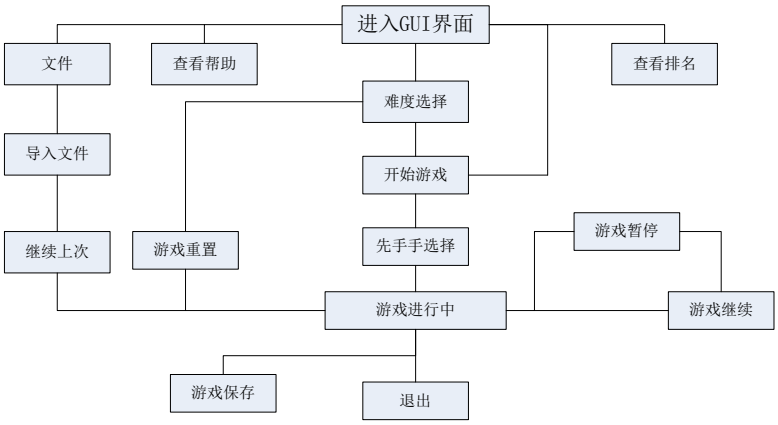
3.1 功能分析图

进入 GUI 界面时，我们会看到游戏、难度等级、文件、成绩排名、帮助五个菜单。其中每个菜单下又会有若干个选择，具体内容见下面 GUI 界面图框架。



GUI 界面框架图

该平台具备一些典型功能，其具体功能框架图如下：

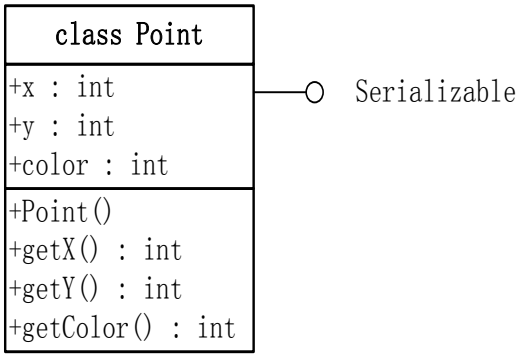


五子棋平台功能框架图

3.2 类图

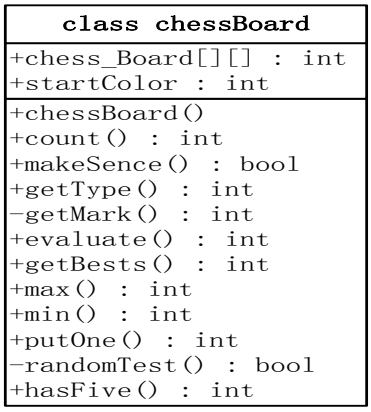
本次大作业程序主要由 myFrame 类、chessBoard 类、Point 类、ArrComparator 类组成。其中，Point 类是对于棋盘位置的定义，chessBoard 类是棋盘类的定义，myFrame 类实现 GUI 界面，ArrComparator 实现比较的接口。

1、 Point 类。



Point 类定义了下棋点的节点类型，其中（x，y）代表下棋位置，color 表示棋子颜色，若 color=0,代表黑色棋子，color=1,代表白色棋子。而且 Point 类实现了 Serializable 接口，方便数据的保存。

2、 chessBoard 类。



ChessBoard 类定义了棋盘的类型，封装了一些棋盘计算的算法等。其中，属性 chess_Board 为 15*15 的二维数组。StartColor 为先下棋一方的颜色。

函数 int[ ] count(int x,int y,int ex,int ey,int color)，用来统计特定方向上特定颜色棋子相连的数目。参数中，（x，y）代表棋子位置，（ex，ey）代表某一特定方向（方向向量），color 待统计棋子的颜色。

函数 boolean makeSence（int x，int y，int ex，int ey，int color）用来统计在某一位置下棋是否有意义。比如，若某个方向上五连的两端被对方下棋，那么中间位置就变得毫无意义。

函数 int getType(int x,int y,int color)，来得到特定落子形成的棋型。主要包括成 5、成活 4 或双死 4 或死活 3、成双活 3、成死 3 活 3、成死 4、单活 3、成双活 2、成死 2 活 2、成活 2、成死 2、其他等。

函数 Int getMark（int k），根据得到不同的棋型给予不同的分值。分值表如下：

棋型	成 5	成活 4、双死 4、死活 3	成双活 3	成死 3 活 3	成死 4	单活 3
分数	100000	10000	5000	1000	500	200
棋型	成双活 2	成死 3	成死 2 活 2	成活 2	成死 2	其他
分数	100	50	10	5	3	0

函数 int[ ][ ] getBests(int color)得到一些比较好的下棋位置。

函数 int max（int alpha，int beta，int step），int min（int alpha，int beta，int step），二者合并一起构成极大极小搜索算法，后文会具体介绍。

函数 int[ ] putOne(int color,int Level)中，Level 代表极大极小搜索算法的深度，该函数会依据极小极大算法计算出指定 color 颜色棋子的最佳下棋位置，当最佳下棋位置不止一个时，利用随机的策略随机选择一个下棋位置。而且随机下棋位置更倾向于中心，因为我们认为中心的棋子要比边缘上的棋子更有意义一些。

函数 Boolean randomTest（int tk）判断随机产生的随机数是否能够被 tk 整除。

函数 boolean hasFive（int color），用来判断颜色为 color 的棋子是否出现五连子。该函数沿着水平、竖直、左上—右下，左下—右上四个方向依次扫描，统计是否出现五连子。

3、 myFrame 类。

MyFrame 类主要包含 GUI 界面设计和事件监听器的实现两大部分组成。

其属性 int color 代表下棋的颜色，若 color=0 为黑色棋子，color=1 为白色棋子。Boolean computer，若 computer=0 为人机对战，computer=1 为人人对战。属性 int State 代表下棋状态，若 state=0,则选手可以下棋，state=1，电脑下棋，state=2 为暂停，都无法下棋。Int Level 代表下棋等级水平，Level=1，初级；Level=2，中级；Level=3，高级。

其函数方法中主要有 init( )、paint（）以及各种事件监听器的实现。下面简要介绍事件监听器的实现。

Class ack_menuItem_FIRST 实现先下棋时的棋盘初始化设置工作；

Class ack_menuItem_LAST 实现电脑先下棋时的初始化设置工作；

Class ack_menuItem_RESTART 实现重新开始的事件监听；

Class ack_menuItem_PAUSE 实现下棋过程中的暂停功能；

Class ack_menuItem_CONTINUE 实现暂停后的重新下棋；

Class ack_menuItem_EXIT 实现游戏退出功能；

Class ack_menuItem_LOW 设置游戏为初级水平；

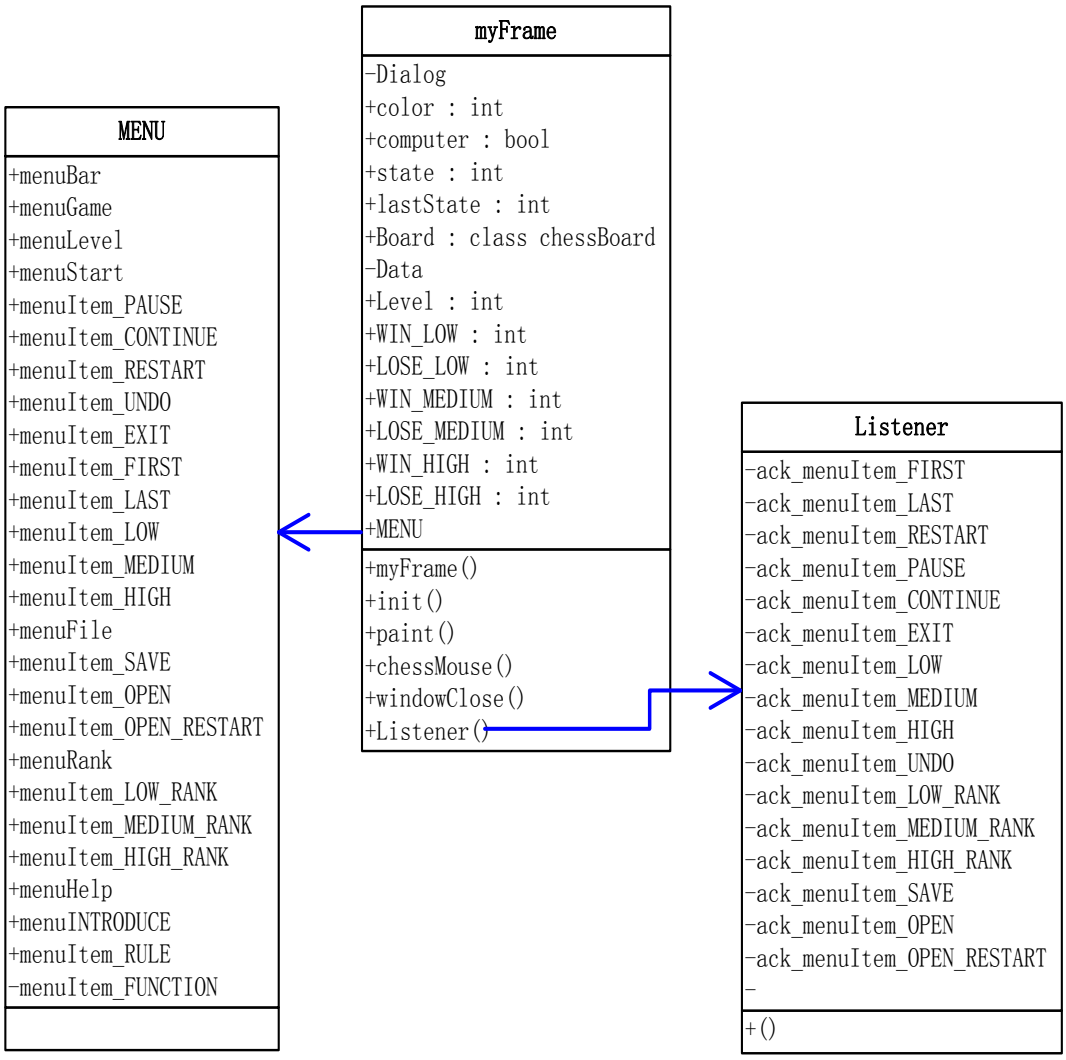
Class ack_menuItem_MEDIUM 设置游戏为中级水平；

Class ack_menuItem_HIGH 设置游戏为高级水平；

Class ack_menuItem_UNDO 实现悔棋功能；

Class ack_menuItem_LOW_RANK 实现低等级成绩的计算；

Class ack_menuItem_MEDIUM_RANK 实现中级等级成绩的计算；
Class ack_menuItem_HIGH_RANK 实现高级等级成绩的计算；
Class ack_menuItem_SAVE 实现保存的功能；
Class ack_menuItem_OPEN 实现打开保存数据文件的功能；
Class ack_menuItem_OPEN_RESTART 实现打开文件后继续上次游戏的功能；
Class chessMouse 实现对棋盘的监听；
Class windowClose 实现对窗口关闭的监听，若窗口关闭了则将一些重要数据保存。



3.3 几种典型功能

1、难度等级设置

通过设置极大极小算法的深度来实现不同的难度等级。由于受到个人计算机计算能力的限制，我分别设置了搜索深度为 depth=1,2,3 的不同等级，分别对应于难度等级的初级、中级和高级。

2、成绩排名实现

在 MyFrame 中定义了不同等级赢的次数和输的次数。WIN_LOW、WIN_MEDIUM、

WIN_HIGH 分别代表初级、中级和高级时玩家赢的次数，LOSE_LOW、LOSE_MEDIUM、LOSE_HIGH 分别代表初级、中级和高级时玩家输的次数。每次游戏结束时，将数据写入文件中，每次重新开始游戏时读入数据。当点击相关按钮时，则相应成绩排名计算的事件，计算出成绩排名。

3、 游戏暂停

利用 state 来设置游戏状态，因为 state=2 则棋盘进入锁定状态，人和计算机都无法下棋，游戏进入暂停状态。

4、 游戏暂停重新开始

原理上可以通过更改 state 的状态来实现，但需要注意一些细节，比如轮到哪方下棋以及棋子颜色问题等。

5、 游戏重置

通过游戏棋盘数据清空和重新初始化，开始游戏。

6、 悔棋

通过 Data 记录棋盘已下棋子（point 类型），然后删除最后两个棋子，重新开始。

7、 游戏保存、导入、继续

利用对象序列化的方法，Point 类实现了 Serializable 接口。具体实现见源码。

四、 核心算法实现

4.1 极小极大算法

极大极小算法的基本思想是急于博弈的双方，一方选择将其优势最大化，而另一方则选择令对手优势最小化。即极大极小算法始终站在博弈一方的立场给棋局估值，有利于这一方的棋局给予较高的评价，不利于这一方（其实也等价于利于对方）给予较低的评价值。当使用该算法时，一方选择评价值极大地子节点走，另一方选择评价值极小的子节点。

下棋时，不同的下法会产生不同的局面。假设黑方下了第一步棋，那么轮到白棋时就有 224 种落子方式，每种都会产生不同的棋局，即不同的子节点，这样一直进行下去，将会生成一颗博弈树。博弈树的最终叶节点有黑胜、白胜和平手三种类型。博弈树为 N 叉树结构，假设每个局面有不超过 10 种可用下法，即 N=10，且黑白双方能在 40 步内完成比赛，则博弈树深度为 D=40，可以计算博弈树最大节点数目为 N^(D+1)-(N-1),约等于 10^41 个局面，而实际中 N 和 D 都比较大，我们不可能去搜索整棵博弈树，因此往往搜索岛一定深度就会停止搜索，然后根据评估函数对局势进行判断。

其伪代码如下：

```
function minimax(node, depth)

    if ( node is a terminal node) or (depth = 0)

        return    evaluate(node)

    if    the adversary is to play at node

        α= +∞

        for each child of node

            α = min(α, minimax(child, depth-1))
```

```
else
    let α := -∞
    for each child of node
        α := max(α, minimax(child, depth-1))
return α
```

4.2 剪枝算法

我们可以发现即便使用深度有限的极大极小算法，棋盘的搜索工作量依然十分巨大。在搜索的过程中我们需要遍历整棵博弈树，每个节点都访问了一次，这样效率低下，搜索量十分巨大。假若将叶节点进行评估，计算倒退值和博弈树产生同时进行，这样就有可能大量减少所需搜索的节点数目，这就是基于 alpha-beta 剪枝的极大极小算法。

搜索过程中，极大节点的 alpha 值等于当前子节点中最大倒推值；极小节点的 beta 值等于当前子节点的最小倒推值。当任何极小节点的 beta 值小于等于它的极大地祖先节点的 alpha 值，则可以终止该极小节点以下的搜索；当任何极大节点的 alpha 值大于等于它的极小祖先节点的 beta 值，则可以终止该极大节点以下的搜索。

其伪代码如下：

```
function alphabeta(node, depth, α, β, maximizingPlayer)
    if ( depth = 0 ) or ( node is a terminal node )
        return the heuristic value of node
    if maximizingPlayer
        for each child of node
            α := max(α, alphabeta(child, depth - 1, α, β, FALSE))
            if β ≤ α
                break (* β cut-off *)
        return α
    else
        for each child of node
```

```
        β := min(β, alphabeta(child, depth - 1, α, β, TRUE))
        if β ≤ α
            break (* α cut-off *)
    return β
```

4.3 估价函数

一般博弈中估价形式可分为基于状态特征的线性组合形式和非线性组合的函数形式。事实上，简单的线性组合函数带来的计算速度上的优势可以抵消复杂非线性函数在精确性上的优势并且整体性能上超越。简单的线性组合估价函数一般形式为：

$$E(p) = g(\sum_{i=1}^{n} w_i f_i(p))$$

P 代表游戏中某个局面，f（p）代表反映该游戏局面某一特征好坏的特征函数，w（i）为权值系数。

五子棋中估价函数选取构造如下：

采取广泛应用的棋型分类。判断是否成 5，若是机器方给予 100000 分，人方给予-100000 分；判断是否成活 4、双死 4、死 4 活 3，若为机器方给予 10000 分，人方给予-10000 分；判断是否成双活 3，若为机器方给予 5000 分，人方给予-5000 分；判断是否成死 3 活 3，若机器方给予 1000 分，人方给予-1000 分；判断是否成死 4，若为机器方给予 500 分，人方给予-500 分；判断是否成单活 3，若为机器方给予 200 分，人方给予-200 分；判断是否成双活 2，若为机器方给予 100 分，人方给予-100 分；判断是否死 3，若机器方给予 50 分，人方给予-50 分；判断是否成双活 2，若机器方给予 10 分，人方给予-10 分；判断是否成活 2，若为机器方给予 5 分，人方给予-5 分；判断是否成死 2，若机器方给予 3 分，人方给予-3 分；其他情况为 0 分。

假设当前落子点为（x，y），棋型为为 f（x，y），对应估值为 g（f（x，y））。五子棋中常用估值方式有两种：

一种是对整个盘面估价，考虑下棋后整个盘面所有可落子点的总得分，即该落子点的估价为 E（x，y）= $\sum_{i,j=0..15且空} g(f(i,j))$ 。

另一种是对该落子点进行估价，直接把该落子点的得分作为估价，即 E（x，y）=g（f（x，y））。

很显然，第一种方法对落子的估价要更准确，但是代价也很大，每次要统计盘面上所有可落子点的得分；第二种准确性不是很好，但是快速简单。本次程序中采用第二种估价方法。

4.4 五子棋算法

本次五子棋决策算中我们通过 max 和 min 函数共同实现。

Max 函数如下：

```
public int max(int alpha,int beta,int step){
    int mx=alpha;
```

```
    if(step==0)
        return evaluate();
    for(int i=3;i<11;i++)
        for(int j=3;j<11;j++)
            if(chess_Board[i][j]==2){
                if(getType(i,j,1-startColor)==1)
                    return 100*getMark(1);
                chess_Board[i][j]=1-startColor;
                int t=min(mx,beta,step-1);
                chess_Board[i][j]=2;
                if(t>mx)
                    mx=t;
                if(mx>=beta)
                    return mx;
            }
    return mx;
}
```

Min 函数：

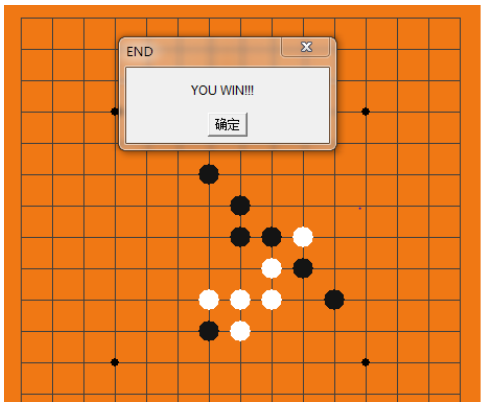
```
public int min(int alpha,int beta,int step){
    int mn=beta;
    if(step==0)
            return evaluate();
```

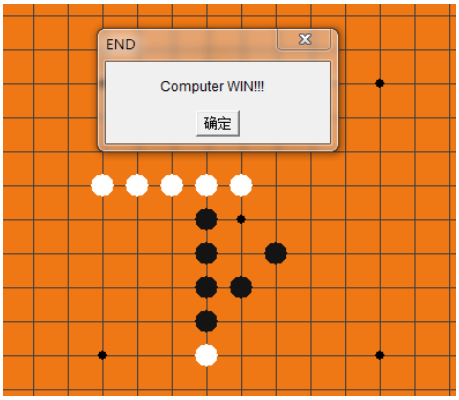
```
int[][] rt=getBests(startColor);
for(int i=0;i<rt.length;i++){
    int ii=rt[i][0];
    int jj=rt[i][1];
    if(getType(ii,jj,startColor)==1)
        return -100*getMark(1);
    chess_Board[ii][jj]=startColor;
    int t=max(alpha,mn,step-1);
    chess_Board[ii][jj]=2;
    if(t<mn)
        mn=t;
    if(mn<=alpha)
        return mn;    }
return mn;
}
```

五、 测试分析

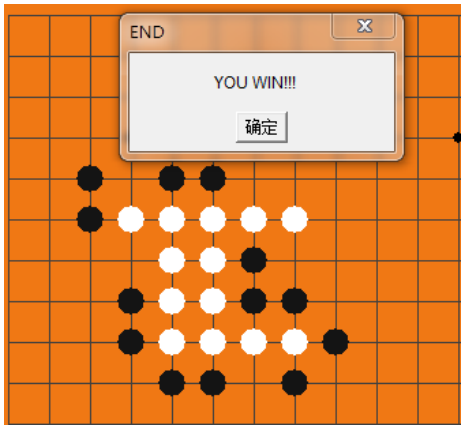
1、 测试游戏能否实现判断胜负以及先后手问题。

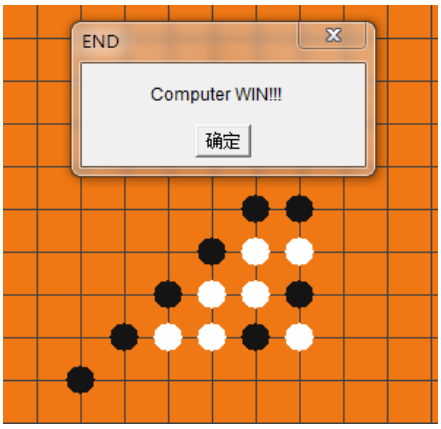
我选择先手（黑棋）：



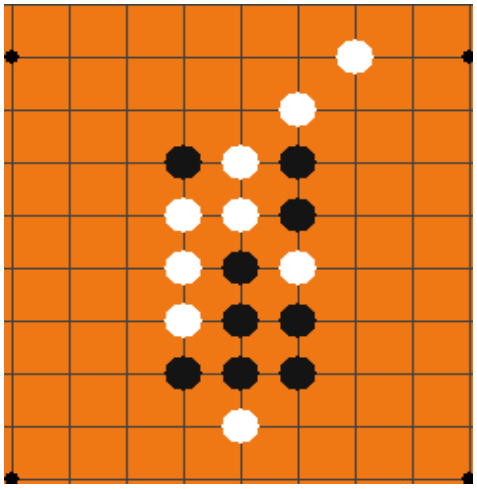


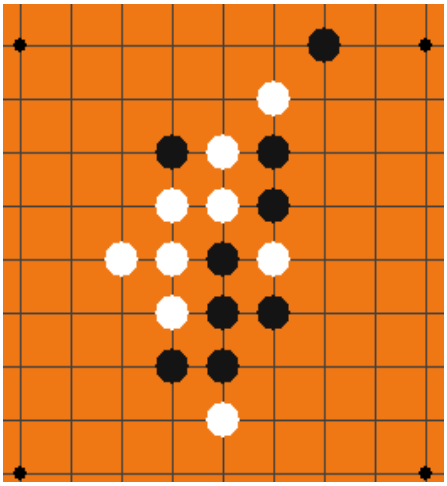
我选择后手（白棋时）：





2、 测试悔棋功能。



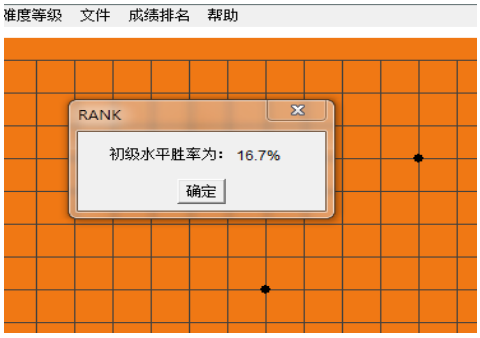


3、 测试不同难度等级程序运行状况。

这个界面显示并没有很大的差别，但运行时间有较大的差别，可以直观感受到初级水平因深度为 1，计算时间很短，而高级水平深度为 3，计算时间较长，计算机反应迟钝，若继续加大难度则由于计算机性能的限制无法完成要求。

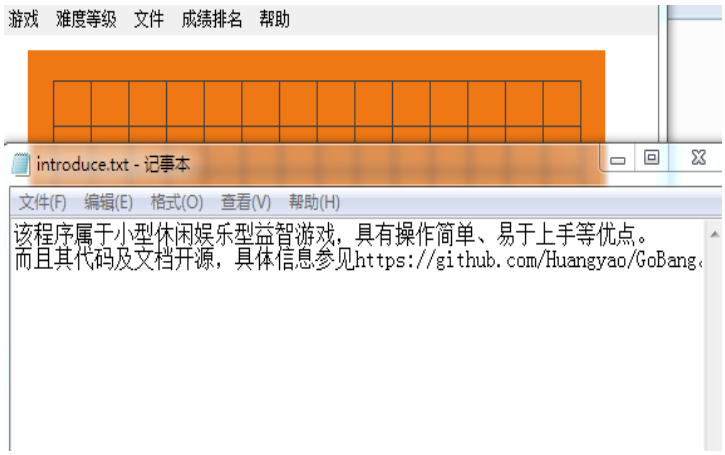
4、 测试不同难度胜率状况。

进入菜单查看成绩排名，即可测试不同等级的胜率。

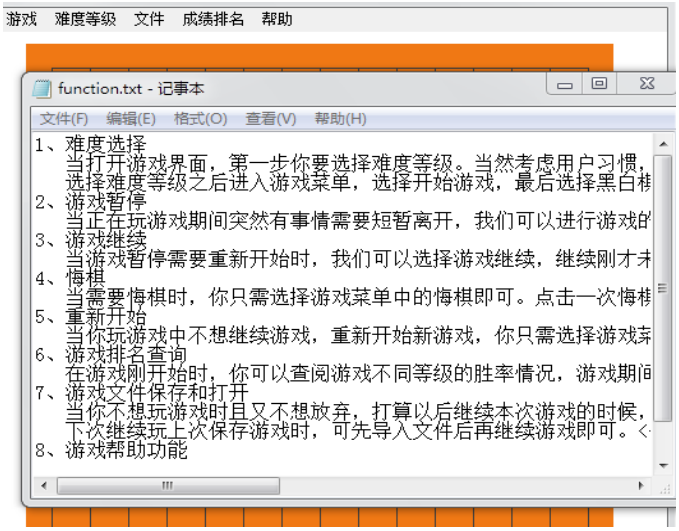


5、 测试帮助菜单

当进入帮助菜单中简介选项时，我们可以打开介绍文档。如下图：



当进入功能选项时，如下图：

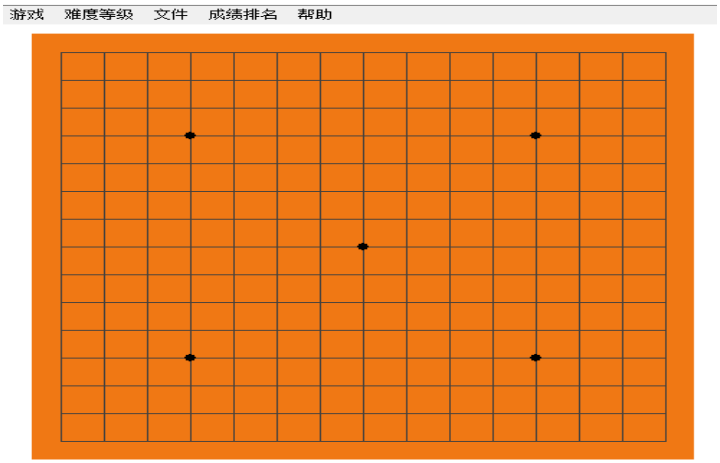


由于写测试报告不知道怎么入手，这次就简要介绍这些测试结果吧。

六、 使用说明

6.1 快速入门

运行主程序会出现如下界面：



然后，你进入游戏菜单，选择开始游戏，最后选择先后手。
比如游戏—>开始游戏—>先手（黑），就开始游戏了呢。
当然你也可以参考帮助菜单。

6.2 功能介绍

1、难度选择

当打开游戏界面，第一步你要选择难度等级。当然考虑用户习惯，如果你忘记选择难度等级，我们会采取默认的难度等级进行下棋。**难度等级—>初级（中级、高级）**

选择难度等级之后进入游戏菜单，选择开始游戏，最后选择黑白棋。**游戏—>开始游戏—>先手黑（后手白）。**

2、游戏暂停

当正在玩游戏期间突然有事情需要短暂离开，我们可以进行游戏的暂停。只有当游戏开始之后我们才能选择游戏暂停。**游戏—>暂停游戏。**

3、游戏继续

当游戏暂停需要重新开始时，我们可以选择游戏继续，继续刚才未结束的游戏。**游戏—>继续游戏。**

4、悔棋

当需要悔棋时，你只需选择游戏菜单中的悔棋即可。点击一次悔棋一步。**游戏—>悔棋。**

5、重新开始

当你玩游戏中不想继续游戏，重新开始新游戏，你只需选择游戏菜单中的重新开始即可，然后直接回到初始状态。**游戏—>重新开始。**

6、游戏排名查询

在游戏刚开始时，你可以查阅游戏不同等级的胜率情况，游戏期间则无法查阅胜率状况。**成绩排名—>初级胜率（中级胜率、高级胜率）。**

7、游戏文件保存和打开

当你不想玩游戏时且又不想放弃，打算以后继续本次游戏的时候，你可以选择游戏文件的保存、下次继续玩时导入即可。保存文件：**文件—>保存。**

下次继续玩上次保存游戏时，可先导入文件后再继续游戏即可。**文件—>导入—>继续。**

8、游戏帮助功能

如果想要快速了解该游戏规则及快速开始，可以进入帮助菜单中的简介、快速开始、功能介绍等。**帮助—>简介（规则、功能）。**

该项目托管在 GitHub，有可能该项目会进一步完善。详细内容参考 https://github.com/Huangyao/GoBang