

JAVA课程设计文档

**作者：陈飞良**

**2017年6月18日**

**目录**

[一.小组说明 3](#_Toc13597)

[二.分工说明 3](#_Toc5763)

[①算法思想上 3](#_Toc1246)

[②游戏界面上 3](#_Toc152)

[③代码实现上 3](#_Toc32189)

[三.课程设计思路说明 4](#_Toc10050)

[I.各个类 4](#_Toc12805)

[II.各个操作的实现 5](#_Toc32056)

[III.电脑的下子 5](#_Toc5339)

[四.程序的成长过程，特点，以及需要改进的地方 7](#_Toc28455)

[1.成长过程 7](#_Toc5589)

[2.程序的特点 11](#_Toc30085)

[3.需要改进的地方 14](#_Toc15868)

## 一. 小组说明：

组名：CST

组长：陈飞良（C）：2015221119200024

组员：

沈珂 （S）：2015221119200141

谭明航 （T）：2015221119200254

## 二.分工说明:

### ①算法思想上：

**本程序的代码实现思想由三人共同讨论得出，其中组员沈珂的“假设下子”思想尤为精妙，让代码实现更为简单，在这基础上，组员谭明航 ，心思缜密，考虑到各种特殊情况，让整个更加智能。组长陈飞良则负责在他们的基础上设计算法进行完善。**

### ②游戏界面上：

**游戏的整体界面由三人共同讨论，组长陈飞良使用PS制作而成。**

### ③代码实现上：

**Ⅰ.游戏的总体布局与架构由组长陈飞良完成。**

**Ⅱ.本游戏的核心是电脑的下子方法，共有七步**

①假设该子为电脑子，判断能连成四子**（由谭明航实现）**

②假设该子为玩家子，判断看是否存在，玩家4子相连，即不算此位置玩家将四子的情况，有则拦截**（由谭明航实现）**

③排除垫脚石的状态, 如果此位置上一位置是玩家子，玩家子可成四子，则该位置优先级为-9**（由沈珂实现）**

④排除连子最多，但是到顶端也无法组成4个连子的**（由谭明航实现）**

⑤如果电脑放一子下一步 电脑能同时形成2种方法成四子 ，则设置max\_x[x] = 4,即优先级为 4。**（由陈飞良实现）**

⑥如果玩家放一子下一步玩家能同时形成2种方法成四子则拦截 ，则设置max\_x[x] = 10,即优先级为 10**（由陈飞良实现）**

⑦普通攻击，假设电脑按照优先级先后(在max\_x中有各列优先级)，同级则随机落子。**（由沈珂实现）**

## 课程设计思路说明

### I.各个类

本程序共设计了一个6个类，主要说下3个类

#### 1.

其中有位置类，即Position类。并且Position类中有一个整型标志成员Label\_Status.

Label\_Status = 0 表示空 .

Label\_Status = 1 表示玩家子.

Label\_Status = 2 表示电脑子.

#### 2.

Game类为主类，放了布局的各个控件，以及，玩家和电脑的移动方法等。

#### 3.

MainPanel类为主面板

### II.各个操作的实现

#### 1.初始化

在设计主界面初始化时先将每个格子放下代表玩家子和电脑子的Label成员.同时设置为不可见,并且Label\_Status = 0 . 即该位置为空

#### 2.下子

实现方法：使相应位置的玩家Label或者电脑Label可见，同时更改标志。

玩家的下子：通过监听键盘和鼠标进行相应的移动，获得玩家下子的位置

### III.电脑的下子

电脑的下子是整个程序的核心，基本是遵循人的思维

设置一个\_y[]数组每个元素初始化为-1，表示该列无可下位置

#### 1.找到每列可以下的位置

遍历某一列的每一行( x : 0-- 6)，找到能下子的那一行,即该子的y坐标

找到后位置赋值给\_y[],即 \_y[x] = y’;

当然此处的y坐标对应的位置应该为115 + 100 \* y

同时x坐标对应的位置为375 + 100 \* x

（这个位置由棋盘大小，和布局时埋子的方式决定）

#### 2. 找出各列放电脑子后可成最大连子数

定义了一个max\_x存储每一列（x ： 0 --- 7）所下位置各方向可成最大连子，每个元素初始化为-10(只是取一个小于0的数可以随意取，但是要方便后期的优先级设置，可以为-7 -8 等)

使用三次Math.max可求出

max\_x[r] = Math.max(Math.max(you, you\_xia), Math.max(xia, you\_shang));

#### 3.考虑各种特殊情况

①假设该子为电脑子，判断能连成四子

this.pos[x][y].setLabel\_Status(2);

遍历查找，成立，则落子，直接宣布比赛结果。

如果不可以，标志恢复

this.pos[x][y].setLabel\_Status(0);

以下各种情况同理

②假设该子为玩家子，判断看是否存在，玩家4子相连，即不算此位置玩家将四子的情况，有则拦截

③排除垫脚石的状态, 如果此位置上一位置是玩家子，玩家子可成四子，则该位置优先级为-9

把该位置的max\_x[x] = -9; （前期max\_x元素初始化为-10）即除非没地方

下，才下此处

④排除连子最多，但是到顶端也无法组成4个连子的

max\_x[x] = 0; 优先级比一般的小，但比垫脚石高

⑤如果电脑放一子下一步 电脑能同时形成2种方法成四子 ，则设置max\_x[x] = 10,即优先级为 4

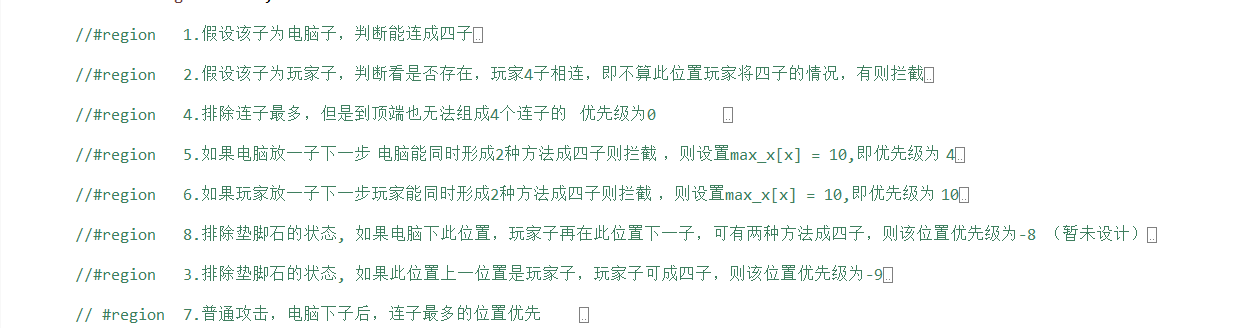
⑥如果玩家放一子下一步玩家能同时形成2种方法成四子则拦截 ，则设置max\_x[x] = 10,即优先级为 10

⑦普通攻击，假设电脑下子后，连子最多的位置(在max\_x中有各列最大)随机落子。（前面的垫脚石类的情况，可以改变max\_x中的值来改变优先级）

如果是2--4列 max\_x += 0.5 ; 同等优先级，中间再优先一点，且不会跨级。

可创建一个数组a,把max\_x数据赋值过去，然后利用Arrays.Sort方法排序最后去a最后一个元素，即为a的最大值，即max\_x最大值

具体代码编写顺序如下：

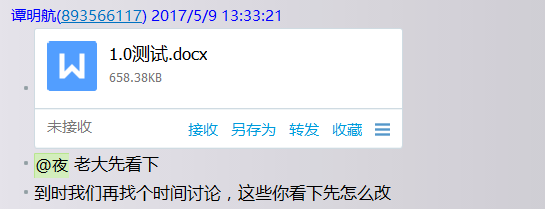


## 四.程序的成长过程，特点，以及需要改进的地方

### 1.成长过程

#### ①.自我完善

由于之前照着网上的例子敲过人与人对战的五子棋，所以完成的效率非常的高，在老师说公布题目之后，我们组便开了的个会，把基本的算法定下了。组员沈珂的“假设下子”思想，对我们的代码实现作用非常之大，在这第一次会议中便确定了7个主要的方法，同时在一周后，便写出了1.0版。然后一路的查漏补缺。







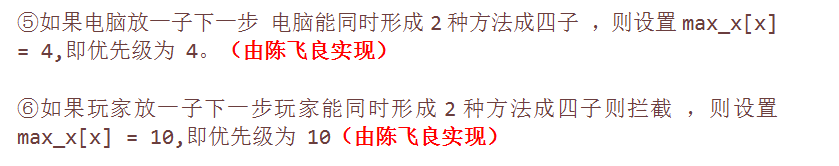




#### ②.取长补短

本程序曾与两个网站的游戏进行对垒

1. 一个是 <http://www.4399.com/flash/48470_1.htm> ，这个网站算法有个优点就是懂得占据中间位置，比过之后，便决定在我们2--4列的优先级+0.5，使其在能形成同样多连子数的时候，比其他位置优先，尽量占据中间位置，但是其他位置比下此位置连子多时，仍然会下那个连子更多的位置。由于这个网站没有写垫脚石的算法，所以我们改完之后的胜率是80%左右。
2. 第二个是 <http://www.7k7k.com/swf/50447.htm> ，这个网站的优点是能主动形成双三，比过之后，我们便在我们程序中加了两个方法：



但是，这个网站的方法还有一个方法便是，在下一个子之后成三子，我们被迫去拦截，同时还给他自己垫脚，此时我们必输。

同时，我们程序没有写防止自己给对方垫脚，使其成双三的算法:



所以，在完善后，我们的胜率只能在50--60%之间徘徊。

### 2.程序的特点

本程序的”假设下子”思想是采用枚举的方法优点在于不需要考虑具体情况，如：



在方法6中，可以拦截的情况有：

#### 1.水平将有两种方法成4子



#### 2.斜方向上将有两种方法成4子



#### 3.双方向成3子



...........等等。即只要符合下一子，能有两种方法成四子就拦，并不需要考虑具体情况，由电脑“假设下子”去枚举，代码效率非常的高，与其他组的程序对比，基本以可以20行代码发挥200行代码的作用。

### 3.需要改进的地方

#### （1）完善方法8



#### （2）写出，提前拦截 玩家在下一个子之后成三子，导致我们被迫去拦截，同时还给他自己垫脚 的方法