

《程序设计实践》实验报告

五子棋

学号： 201611130126

姓名： 袁宇昊

学院： 信息科学学院

专业： 计算机科学

教师： 尹乾

日期： 2018.10.12

1. 设计目的

共有两个题目：第一个题目是五子棋游戏（难度有区别），第二个题目是数独。

五子棋游戏是在15\*15的棋盘上，黑白双方对弈下子，先完成五子连线即为胜者。

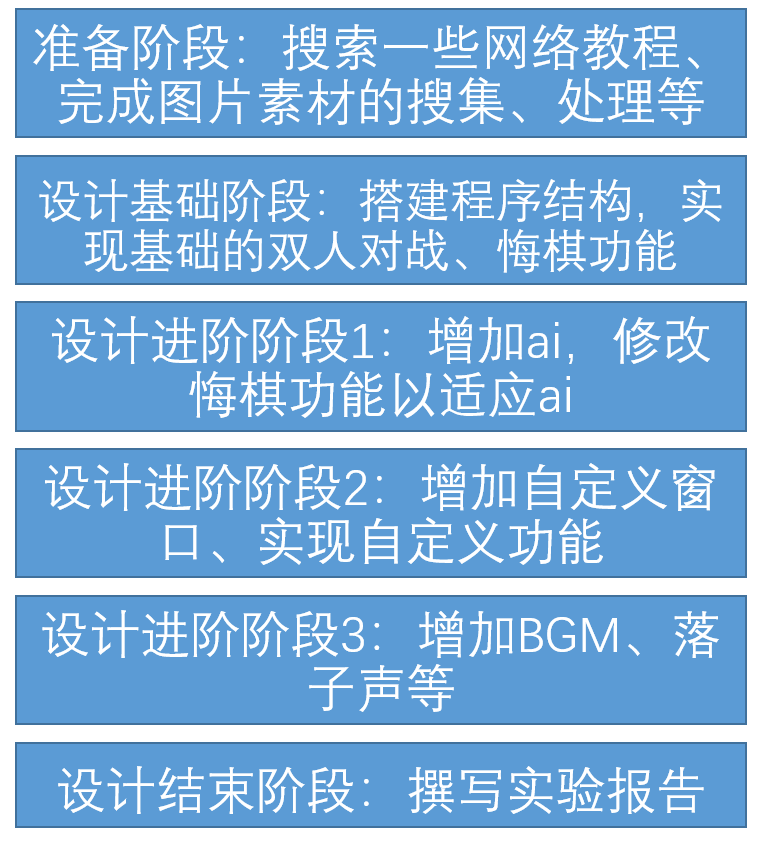
二、总体设计**（程序设计组成框图、流程图）**

整体架构是用python里的pygame库实现的。

版本是python的3.6版本和pygame的1.9.4版本。

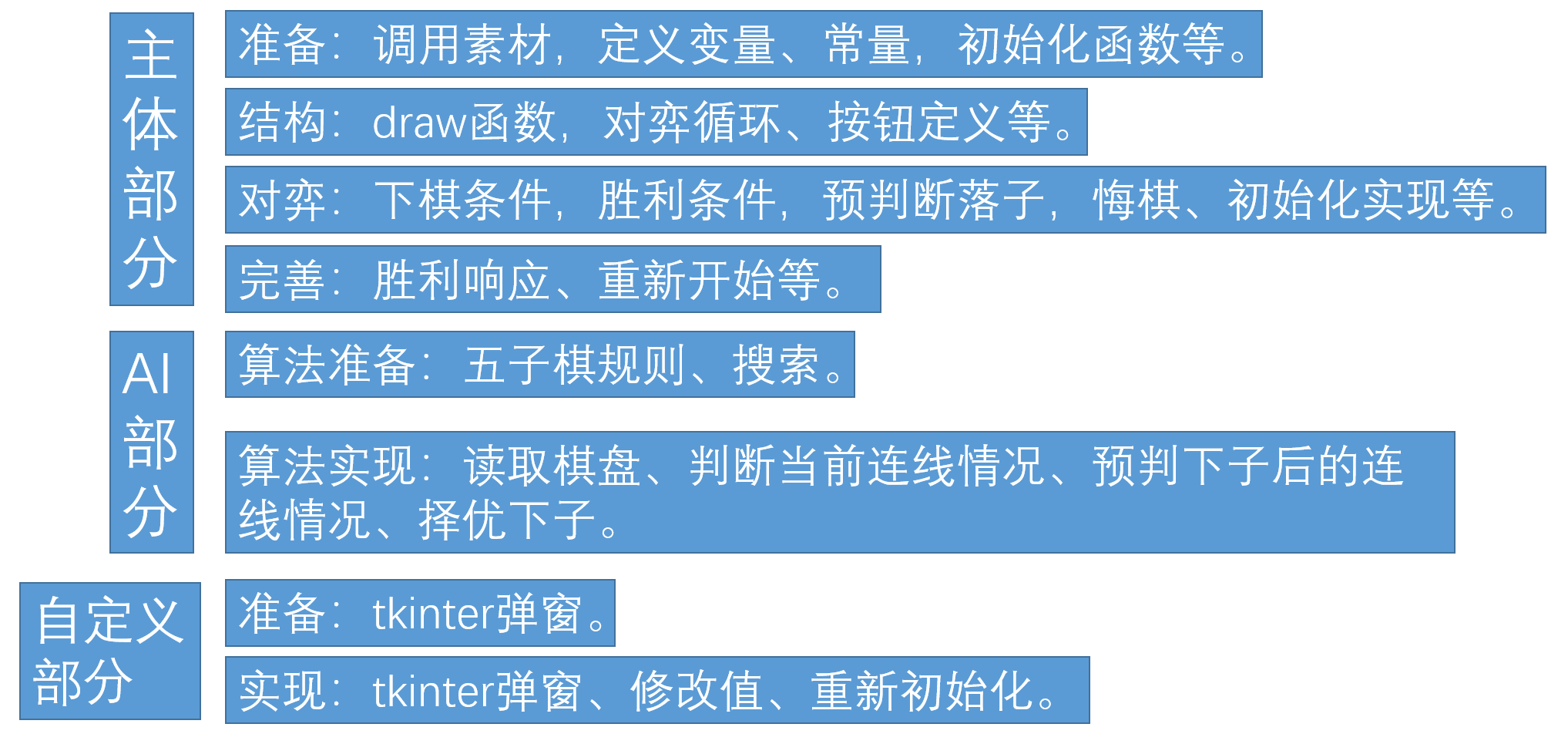
利用pygame库函数加载背景图片、棋子素材等图像材料。利用数组记录棋谱以实现悔棋功能。利用广度优先搜索算法实现ai功能。调用tkinter库实现一些简单的修改，如ai数量、棋盘大小、胜利条件等。

整体的程序设计流程图如下：



程序总共就分为了3大部分：主体部分、ai部分、自定义功能部分。其中主体部分是由2个循环构成，一个是游戏界面的循环（方便自定义函数），一个是下棋的循环（包括了悔棋功能）。

程序设计组成框图如下：



**三、详细设计（模块功能说明，如函数功能、入口及出口参数说明，函数调用关系描述等）**

**结构部分：**

**主要变量：棋子大小、格子大小等长度，棋盘大小、胜利条件等数量，图片句柄、音乐句柄等。**

Init ()函数：pygame库里的初始化函数。

mydraw（tx,ty,player）函数：主要界面的绘制、更新。tx,ty即当前下棋的位置，player为当前下棋的选手。函数逻辑大意为如果当前的格点（tx,ty）上没有棋子，那么绘制一个预落子，如果当前个点上有棋子，那么跳过。对于其他的格点就按照一般的处理。

**对弈部分：**

**主要变量：player当前的玩家，只能是1或2。mymatrix[][]棋盘矩阵，0为空格、1为黑子、2为白子。premymatrix[]棋谱数组，记录下棋记录。**

下棋功能：直接写在对弈循环里，大致实现为判断当前格点是否有棋子，没有棋子的话就更新棋盘矩阵为player，并且更新player和棋谱数组、调用check()函数。

悔棋功能：直接写在对弈循环里，大致实现为读取棋谱数组的尾步，棋盘矩阵直接置0，并且更新player和棋谱矩阵。

check（tx,ty,player）函数：判断当前棋盘是否胜利。tx,ty即当前下棋的位置，player为当前下棋的选手。函数逻辑大意为如果当前结点（tx,ty）下子（player的棋子）后,其八个方向上是否已经成功连线5子。返回值为胜利的玩家（player或0）。这个函数在后面的ai实现中有借鉴作用。

initail()函数：重置棋盘矩阵和棋谱数组等。主要在初始化按钮和重新开始的时候调用。

**AI部分：**

**主要变量：ai\_num ai的数量：0为PVE,1为PVE,2为EVE。**

ai\_check(x,y,player,tar)函数：check()函数的变体，只是把胜利条件的判断变成了tar变量，其他都是与check()函数是一样的。

ai\_find(sx,sy)函数：sx,sy是ai的对手下的上一步棋子位置。这个函数主要用到了搜索的办法。五子棋的规则是限制对伐子连得多同时尽量保证自己的子连得多。所以从最为危险的情况（tar=5）到最简单的情况(tar=2)下，先判断（sx,sy）周围的空格结点上先下对方的棋子后的棋盘情况，一旦上述情况下有连线到tar的情况，那么就直接返回这个位置即可。然后再判断下己方棋子后的棋盘情况，一旦上述情况下有连线到tar的情况，那么就直接返回这个位置即可。注意：如果有多个相同的情况，就随机选择一个位置返回。

这个函数的调用位置主要是在对方下棋后，所以也兼容了悔棋功能，不过要修改悔棋功能为一次悔棋两步。

**自定义部分：**

**主要变量：无。**

call\_set()函数：在reset按钮中调用的函数：主要是定义弹窗的界面，并且处理输入的数据并进行修改。现在支持对棋谱大小的修改、对胜利条件的修改和对ai数量的修改。有针对默认值和最大最小值的处理。

**音效部分：**使用了两个音效，一个是BGM在开始时播放，设置了空格键为停止播放和重新播放的信号。另一个是落子声，在下子后播放。

**四、调试与测试**

使用的是spider编辑器，使用了它自带的设置断点调试的功能，也用到了它的变量表查看变量，还用到了print()函数查看进行的程序步骤。

**主要遇到的问题：**

**1数组越界：**因为用到了mymatrix[][]矩阵数组和premymatrix[]数组来存放棋谱的情况，所以要防止数组的越界。调试的报错信息会给出出现错误的行数很有帮助。

具体遇到的情况如下：

实现draw函数时搞反了x、y轴导致越界，而且在后面的更新中没有一并更改导致了越界。

实现对弈功能时的搞反了x、y轴导致越界，并且在处理下棋的逻辑时多次越界，主要原因是在于没有想清楚下棋的逻辑就写了。

实现悔棋功能时的棋谱数组越界，主要是在没有棋子时不能进行悔棋。

Check函数中因为注意了数组的大小限制，所以出错了但是很快就改正了。

解决办法：一定要加上对数组下标的限制后再进行数组的访问。同时注意程序上下文的一致性。

**2 下棋的逻辑错误：**主要体现在对player变量的更新不足上，因为要实现悔棋功能，那么player变量必须时刻变化，有的地方就造成遗漏或者重复，体现出来有一下几点：

双ai转换玩家错误、单ai转换时的错误、无ai时悔棋后的玩家错误，单ai时悔棋后的玩家错误。

解决方法：主要是采用print()步骤跟踪的办法查出来出错的地方后再更改，如此往复。这种双人对战的游戏必须注意这种玩家转换中的逻辑。

**3 程序无响应：**比如点击无效、卡死、闪退等。一般是程序陷入了死循环。

解决办法：记录下发生无响应的操作，再设置断点进行调试和分析。

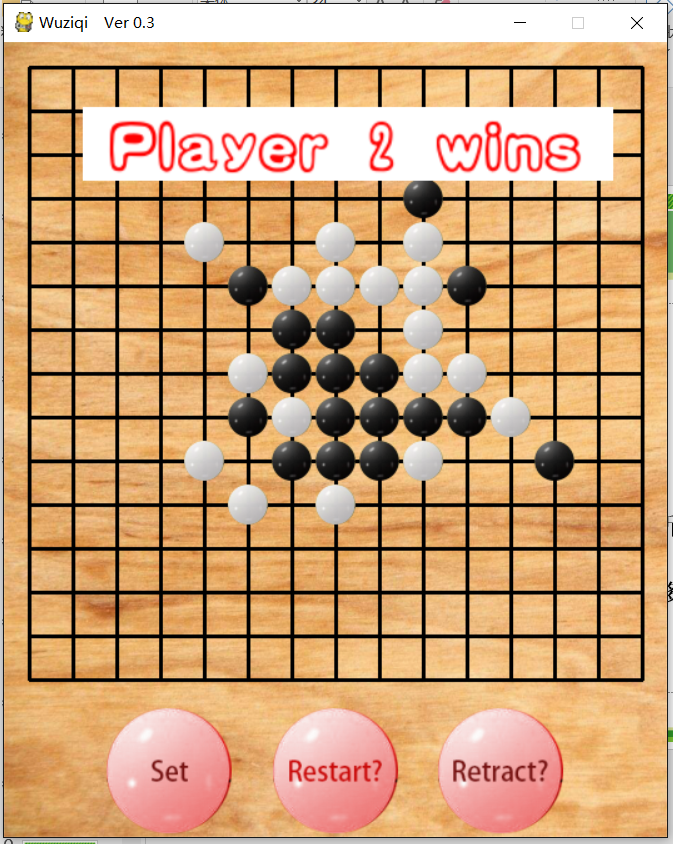
**4 封装后的程序太大：**采用PyInstaller去封装，第一次直接分开打包出来600+MB的文件。

解决办法：上网查询后得知主要是因为PyInstaller会把囊括的库都封装起来，于是优化了import部分的代码，然后采用" pyinstaller -F "直接封装成一个完整程序，最后得到一个225MB的文件。

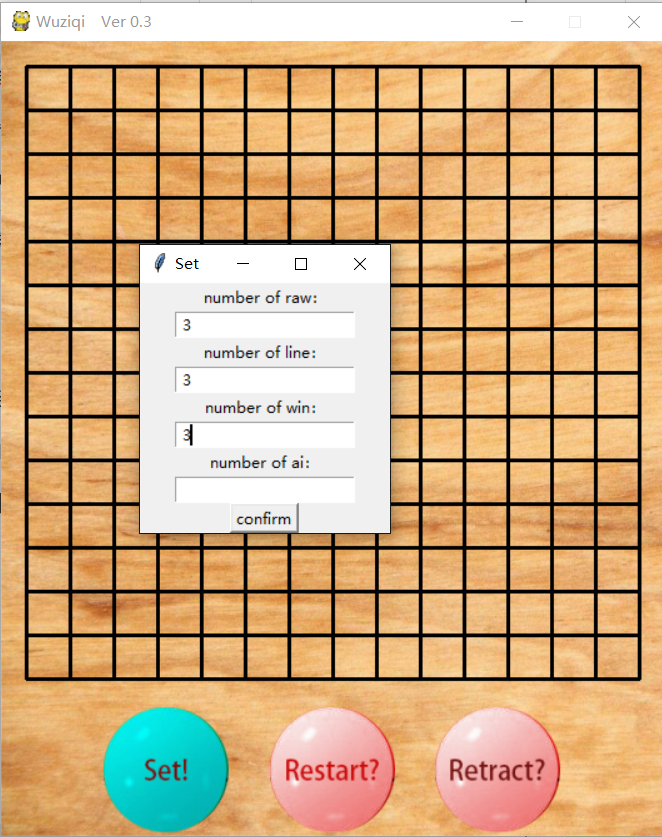
**五、源程序清单和执行结果**

因为要想把程序做整体的封装，所以只有一个主程序main.py，同时有九张图片的素材和两个音效素材。

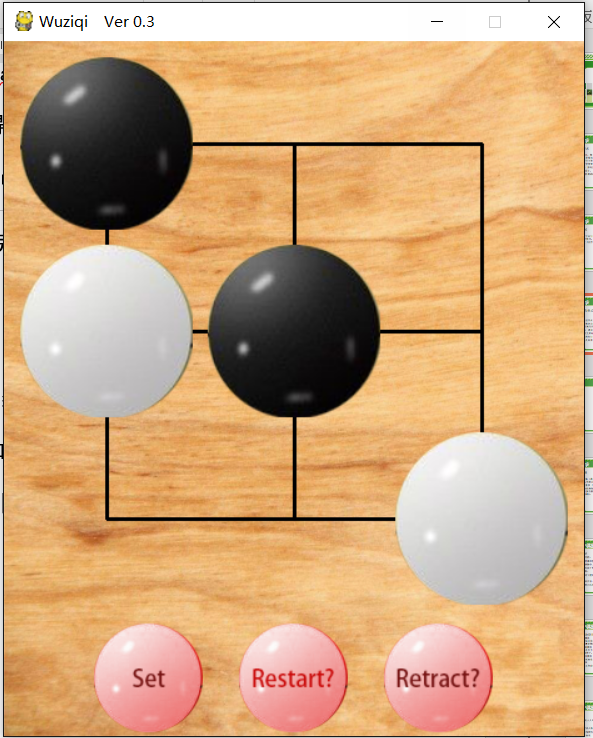
执行的结果可以参考下面的图片：



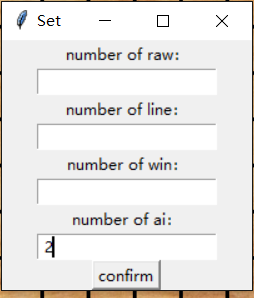
人机对战

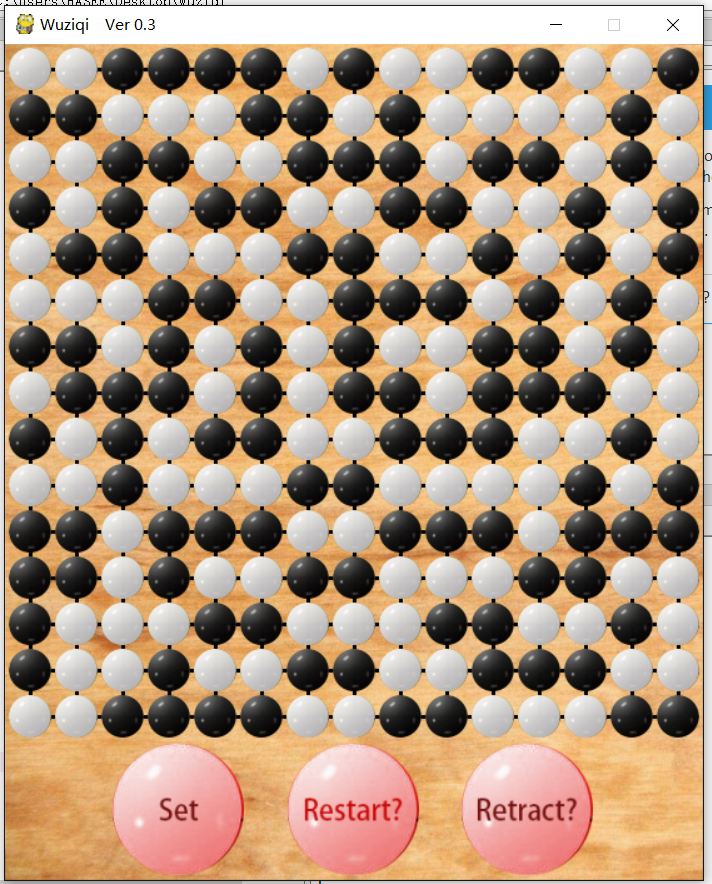


自定义窗口



通过自定义窗口自定义的3\*3的“三子棋”





两个ai下五子棋弈出的和棋