# 四子棋项目实验报告

[四子棋项目实验报告 1](#_Toc198382503)

[一、 项目概述 2](#_Toc198382504)

[二、 功能索引 2](#_Toc198382505)

[2.1 欢迎对话框（WelcomeDialog，main.py 20行） 2](#_Toc198382506)

[用户创建（main.py 29行） 2](#_Toc198382507)

[自动登录（main.py 93行） 3](#_Toc198382508)

[选择用户登录（main.py 20~54行） 3](#_Toc198382509)

[防止意外退出（main.py 67行） 3](#_Toc198382510)

[2.2主界面（MainMenu.py） 3](#_Toc198382511)

[用户操作（307行，351~388行） 3](#_Toc198382512)

[用户等级评定（224行） 3](#_Toc198382513)

[人机对战（415行） 3](#_Toc198382514)

[MiniMax算法（ai\_player.py，236行） 4](#_Toc198382515)

[双人对弈（联机）（67~92行，426~530行） 4](#_Toc198382516)

[双人对弈（不联机）（426行） 5](#_Toc198382517)

[历史复盘（451行） 5](#_Toc198382518)

[2.3对弈界面（Game\_Window.py） 5](#_Toc198382519)

[对弈双方数据显示 5](#_Toc198382520)

[判断落子（181，255，264行） 5](#_Toc198382521)

[状态提示及计时（383，391行） 5](#_Toc198382522)

[AI落子（216，249行） 5](#_Toc198382523)

[联机落子（171行） 6](#_Toc198382524)

[绘制功能（271~301行） 6](#_Toc198382525)

[胜率显示（Winrate\_bar.py） 6](#_Toc198382526)

[胜负判断（447行） 7](#_Toc198382527)

[棋盘功能区（505~529行） 7](#_Toc198382528)

[彩蛋功能 7](#_Toc198382529)

[2.3历史复盘（game\_history.py，replay.py） 7](#_Toc198382530)

[历史记录选择界面（game\_history.py） 7](#_Toc198382531)

[回放界面设计（replay.py） 7](#_Toc198382532)

[模拟落子系列功能（202行等） 8](#_Toc198382533)

## 项目概述

本项目为基于python的四子棋游戏，并基于PyQt5使用GUI代替了图形文字界面。除了本地双人对战以外，本项目还提供了3种难度的人机对战，双人局域网内联机对战，和复盘功能。同时，在对局过程中，本项目还提供胜率显示、序数标注、三联高亮等特色功能；在复盘时，可以根据“最佳列”提示使用模拟落子功能，起到教学作用。为了实现以上功能，本项目建立起了简易的用户体系，将用户数据保存在userdata文件夹内。

本项目的入口是main.py，不过整个项目已经使用pyinstaller打包为main.exe，可双击直接运行。main.py负责用户的登录，随后交给Main\_Menu.py负责用户主界面，Game\_Window.py负责游戏界面，game\_history负责历史记录界面，Winrate\_bar.py负责胜率条显示，replay.py负责复盘的对局界面，ai\_player.py负责AI以及胜率计算。

如助教想运行本项目源代码，所需要的库列在了requirements.txt中。

祝助教游玩愉快！

## 功能索引

### 2.1 欢迎对话框（WelcomeDialog，main.py 20行）

#### 用户创建（main.py 29行）

为方便起见，程序若没有检测到userdata里的users.json文件，直接创建一个用户，用户名为“默认用户”，用户直接选择登录即可，后续可以更改用户名。由于四子棋不涉及资产相关，不设置密码。

#### 自动登录（main.py 93行）

为方便起见，程序若检测到users.json文件有default\_account信息，将会直接使用该用户的信息登录。如果没有，则会生成欢迎对话框，此时用户还可以选择是否将本账号作为默认登录账号。

#### 选择用户登录（main.py 20~54行）

如果没有默认登录账号，欢迎对话框会读取users.json文件，提取其中的users信息并以combo box的形式给用户选择登录账户。

#### 防止意外退出（main.py 67行）

由于欢迎对话框会读取users数据并将其清空，如果在程序还未将新数据写入之前就被关闭，会导致users.json数据无法恢复。因此不可以关闭欢迎对话框。

### 2.2主界面（MainMenu.py）

#### 用户操作（307行，351~388行）

由于用户操作并非本游戏的主要功能，故在此简单介绍。这些功能基于QMainWindow类提供的菜单栏（menu bar）来实现，并使用状态提示栏（status tip）提示。

切换用户：在本地已有用户列表选择切换，并可以选择是否以后默认登录新账号。

新建用户：可以使用新的用户名新建账号，注意不要输入字母和数字以外的字符。

注销用户：删除该用户的所有信息，并回到欢迎对话框重新登录。

更改用户名：允许用户更改不重名的用户名。

#### 用户等级评定（224行）

通过已下局数和胜率来评定用户等级，鼓励玩家在数量和质量上进步。其中，由“青铜”到“钻石”仅需通过增加局数来达成，分别需要达到10，30，50，100。超过100局后，胜率超过60%可达到“黑钻”等级，否则保持“钻石”等级。

#### 人机对战（415行）

选择相应难度，点击“人机对战”按钮进入人机对战。程序将发起FirstMoveDialog对话框，询问玩家谁作为先手，随后转换到GameWindow界面，模式为1。

简单模式的AI仅会伪随机落子，一般而言玩家一定可以获胜；普通模式和困难模式的AI共用一套逻辑，即minimax算法。区别在于搜索深度，普通搜索2层，困难搜索5层（最多5层，否则程序会未响应）。此算法下的AI极难战胜。

#### MiniMax算法（ai\_player.py，236行）

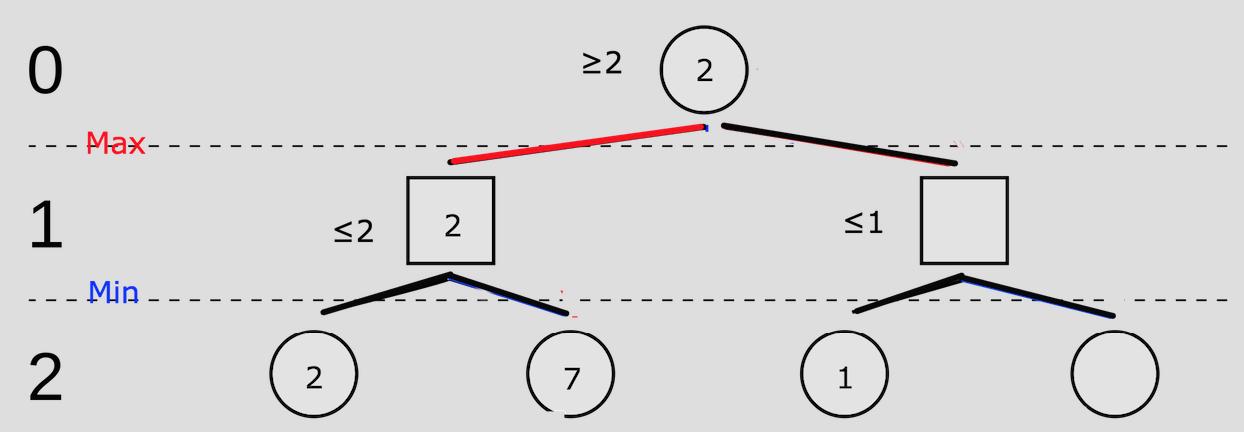
这里只介绍AI算法的核心，之后对于此算法的使用在各自的位置有介绍。

首先，算法会更改当前棋盘数据，故需要先进行深拷贝，并简要处理棋盘数据。随后进入minimax算法部分。这一算法适用于所有玩家和电脑轮流对下的全知游戏，即先搜索一定层数，通过得分进行量化估计。算法假设两个玩家都执行最优博弈，这也是为什么有时胜率计算会出现跳动的原因，玩家可能走出了算法所不考虑的落子。

假设当前轮到AI落子，搜索5层，需要最大化AI的分数。第一层为玩家落子，以此类推，第五层也为玩家落子。玩家落子会让AI的分数最小化，AI落子会让AI的分数最大化，反复对当前层的所有情况交替取min和max，就可以倒推出当前AI的分数。

于是，我们枚举AI在每一可能的列落子，随后计算对应的minimax得分，取最大值作为AI真正的落子列，取这一列的得分来进行胜率计算。

我们注意到，这样总共需要模拟7^6≈11万的情况，对于python而言会花费超过1秒。Minimax还有一个剪枝策略，我们假设这一层需要对分数取最小值。如果发现这一层的某一分支已搜索到的最小值大于另一分支任意节点的值，那么另一分支的最小值一定小于本分支的最小值，由于这一层的前一层需要对分数取最大值，那么本分支的最小值一定作为上一层的最大值，另一分支就不需要继续搜索了，如下图所示。



具体而言，α：AI节点能取得的最小值（AI层更新），β：玩家节点能取得的最大值（玩家层更新）。此时α和β作为递归函数参数不断被继承和使用，每一层会更新其中一个变量，同时继承上一层的另一个变量，并在每次更新时做β剪枝。

#### 双人对弈（联机）（67~92行，426~530行）

选择联机按钮点击双人对弈按钮，程序提示玩家选择作为主机还是客户端（506行），若选择主机，则会显示此电脑的ip地址（496行），并提示用户输入端口号（默认12345）。确认完毕后，启动server类监听所有连接。由于accept是阻塞函数，在监听线程中保持监听（45行）。若选择客户端，则会提示用户输入主机ip地址和端口号。确认完毕后，启动client类尝试connect，最多给予60秒尝试（因为需要等待主机玩家选择先手，而若在此过程由于主机没有回复客户端可能会认为连接失败，所以给予主机用户60秒时间选择先手，也给两方线上交流的可能）。客户端一旦连接成功，立刻发送hello信息，内含自身用户名及等级；主机收到后立刻回复hello信息，同样包括自身用户名及等级；客户端收到后回复ready信息；主机收到ready信息后，发起FirstMoveDialog对话框，询问主机玩家谁作为先手，确认后发送ready信息，内含先手信息，同时转换到GameWindow界面，模式为3；客户端收到后，也转换到GameWindow界面，模式为3。此时交换游戏开始所需的基本信息的过程结束（455行），游戏正式开始。

#### 双人对弈（不联机）（426行）

选择不联机按钮点击双人对弈按钮，程序默认当前登录玩家为发起人，可以选择本地的另一位玩家作为对手，发起SelectSecondPlayerDialog对话框。随后发起FirstMoveDialog对话框，此时对战双方理应在现实中会面，由二人协商谁作为先手，随后转换到GameWindow界面，模式为2。

#### 历史复盘（451行）

由于复盘记录可以删除以节约存储空间，因此可以复盘的局数会小于玩家已下局数，故在右侧显示。点击历史复盘按钮进入GameHistory界面。

### 2.3对弈界面（Game\_Window.py）

#### 对弈双方数据显示

机器人使用难度作为其等级，默认当前登录用户为红方，机器人或客户端为蓝方。红蓝方与先手无关。若先手不是红方，后面涉及该谁下的计算是需要加上1（self.turn）。

#### 判断落子（181，255，264行）

游戏提供两种落子方式，按A~G对应从左到右的1~7列落子；鼠标点击落子。鼠标点击时获取点击的横坐标，除以棋盘宽度计算落子列。程序会判断该列是否已满以及点击位置是否正确，成功后更改棋盘数据，并开始落子动画，播放落子提示音。注意，如果正在动画中或者AI还未落子时也不可以落子。

#### 状态提示及计时（383，391行）

一旦落子完毕，会根据当前棋盘数据提示该谁下。同时开始落子倒计时（仅双人对战开启）。

#### AI落子（216，249行）

如果是模式1，那么玩家落子后需要让AI落子。但AI计算很快，如果玩家落子动画没有播放完毕就让AI落子，会导致二者动画出现重叠，影响观感，所以使用计时器等待1秒后执行AI落子。AI落子需要将此时棋盘数据和刚刚谁下的信息告知AI类即可。

#### 联机落子（171行）

联机落子较为不同，落子数据需要远程传输。在启动游戏后，主机和客户端仍会通过send发送move信息，内含落子列。在我方落子完毕，对方落子之前不允许我方落子。

#### 绘制功能（271~301行）

每次棋盘状态更新都需要重新绘制棋盘方格，重新绘制已有棋子，按帧数绘制动画棋子，对三联棋子高亮显示，并按需绘制棋子序号，如确定胜负还需要绘制四联棋子的连接线。

棋子动画通过每15ms下落15px来完成，此时棋盘数据虽已更新，但处于锁定状态防止在动画结束前就在对应位置绘制出棋子来。当计算出已下落到指定行，动画结束并正常绘制棋子。

动画一旦结束，需要更新三联棋子提示以及更新胜率。若有三联棋子，会将内部填充绘制成低透明度。注意，此处三联棋子仅包括有发展成四联棋子可能的，若两头都被堵上了不计。

如果需要标注序数，则在棋子中央按照红色标黑，蓝色标白来标记序数。

#### 胜率显示（Winrate\_bar.py）

计算胜率需要用到困难AI（为使胜率计算最准确）来模拟后续棋局。AI不仅提供最佳列，还会提供最佳分数。由于分数评定系统被设置为胜利1000，失败-1000，和局0，程序采取距离反比来计算胜率，即若分数为500，距离胜利500分，距离失败1500分，距离平局500分，呈1：3：1，由于距离越近越有可能，概率比为3：1：3，胜率为3/7，失败为1/7，平局为3/7。这一胜率计算方法较为实用，在实战中效果较好，例如开局时，由于四子棋还是较为公平的游戏，平局概率接近100%；随着游戏进行，双方落子会不断提高或削弱自己的胜率，平局概率不断减小，局势逐渐明朗；游戏将要结束时，AI已经算出结果，此时某一方胜率会接近100%。

由于Qt没有很好的显示胜率的控件，游戏自己定义了一个widget，计算总宽度的比例来绘制矩形，填充红色或蓝色，平局则没有颜色。同时在下方标注详细胜率数据。注意，这个控件也可以用来显示胜负信息（只需要把数据替换掉就可以），一石二鸟。

小提示：双人对战看到自己胜率100%时不要高兴太早，你不一定能下到最佳列！

#### 胜负判断（447行）

判断胜负理应在落子后进行，但事实上，判断三联棋子的同时也可以判断出四联，因此在board\_hint部分可以直接判断。游戏出结果后，对话框提示胜者，随后储存棋盘数据，更新用户数据。双人对战数据仅会在发起人这里储存一份棋盘数据，但双方的用户数据都会更新。双人联机的话由于是两份游戏，各自都会存储。

同时，认输与求和也可以直接影响胜负，此时需要修改胜率（不能根据当前棋盘数据计算）。

#### 棋盘功能区（505~529行）

所有对局都会记录已下步数和对局时间，人机对战不会倒计时，但双人对战会。人机对战可以悔棋，悔棋一悔悔两步。双人对战可以求和，如果是不联机状态，游戏默认两位玩家在现实中已经商量好了，于是直接结束游戏；如果是联机状态，求和方会发送peace信息，被求和方会收到对话框提示，可以选择接受与否，将结果含在peace信息里回复，求和方收到后提示用户求和是否成功。

#### 彩蛋功能

为了调试方便，该你落子时，按L可以获得落子提示。同样，这一提示也是基于困难AI的计算，让你获得对于人类对手和普通AI的优势。但是，为了防止该特性被滥用，也是由于程序内部对棋盘列从0开始编号，事实上AI的提示需要+1。

### 2.3历史复盘（game\_history.py，replay.py）

#### 历史记录选择界面（game\_history.py）

这一界面的设计非常简单，故简要介绍。历史记录采用tablewidget类呈现，只需要加载玩家的所有游戏数据即可。上方调用WinrateBar类显示胜负信息，下方有按钮给用户操作。考虑到空间存储的原因，游戏允许用户删除不想要的历史记录。

#### 回放界面设计（replay.py）

这一部分与GameWindow类基本一致，但使用了overlay让“最佳列”提示覆盖于棋子之上。具体来说（31行），overlay只覆盖label，通过label的大小来确定“最佳列”提示的显示位置。按钮区域提供上一步、下一步、（暂停）自动播放、撤销模拟落子、回到真实棋局这些功能按钮。逻辑是，按下自动播放后，自动播放按钮自动更名为暂停自动播放。除去按钮控制之外，用户可以在焦点位于棋盘（点击棋盘边缘）时使用右箭头、左箭头、空格来执行下一步、上一步、开始/暂停自动播放的功能。

#### 模拟落子系列功能（202行等）

模拟落子可以直接鼠标点击，因此只提供撤销模拟落子的按钮。同时，模拟落子可以在任意自动播放未开启的时刻进行，也就是说可以从任意步开始模拟。撤销模拟时，也只能撤销开始模拟后的棋子，不能撤销原历史记录的落子。模拟落子的序数标注会使用斜体，且替换为黄色、绿色，建议用户使用模拟落子功能时打开序数提示，否则容易分不清。开始模拟时，状态栏也会提示“模拟中”，但已下步数不会更新，仍保持模拟前原样。