

简易的五子棋对战程序

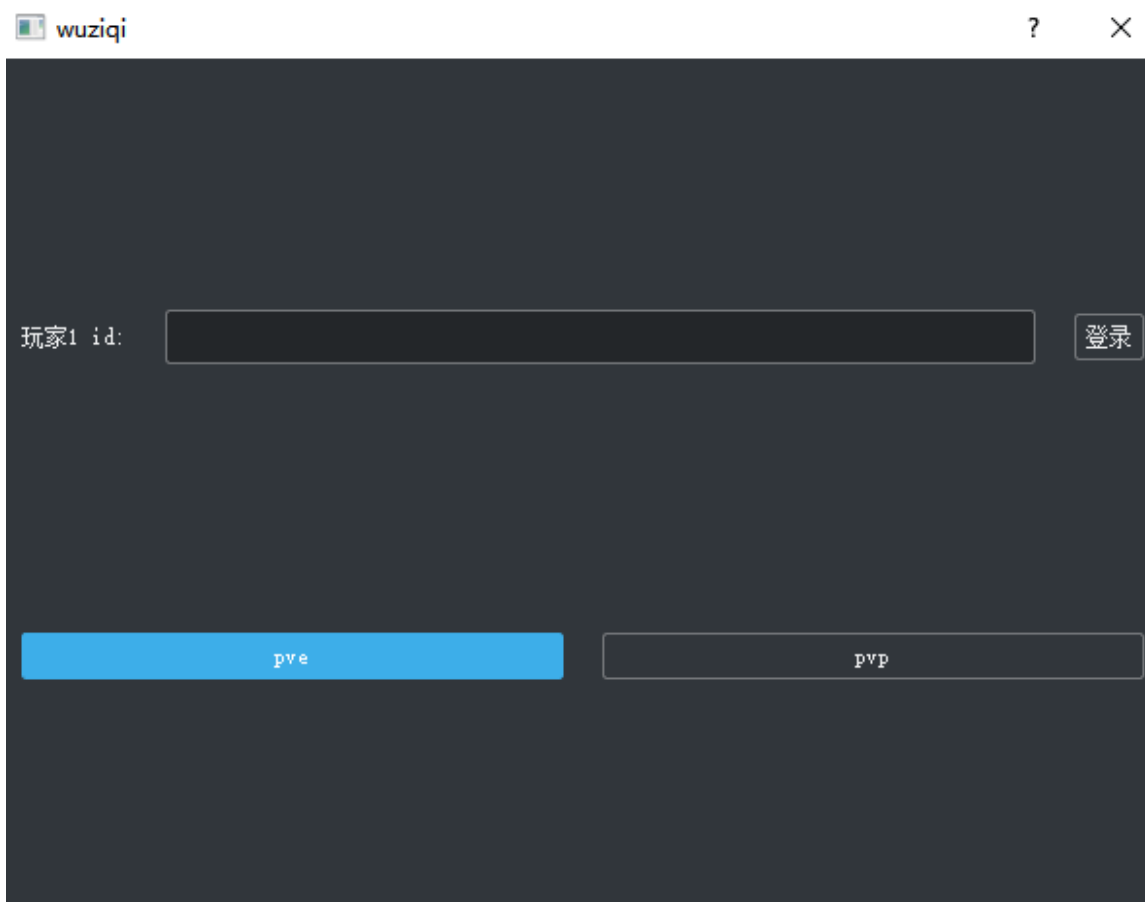
-----OOP 课程设计

一、基本介绍

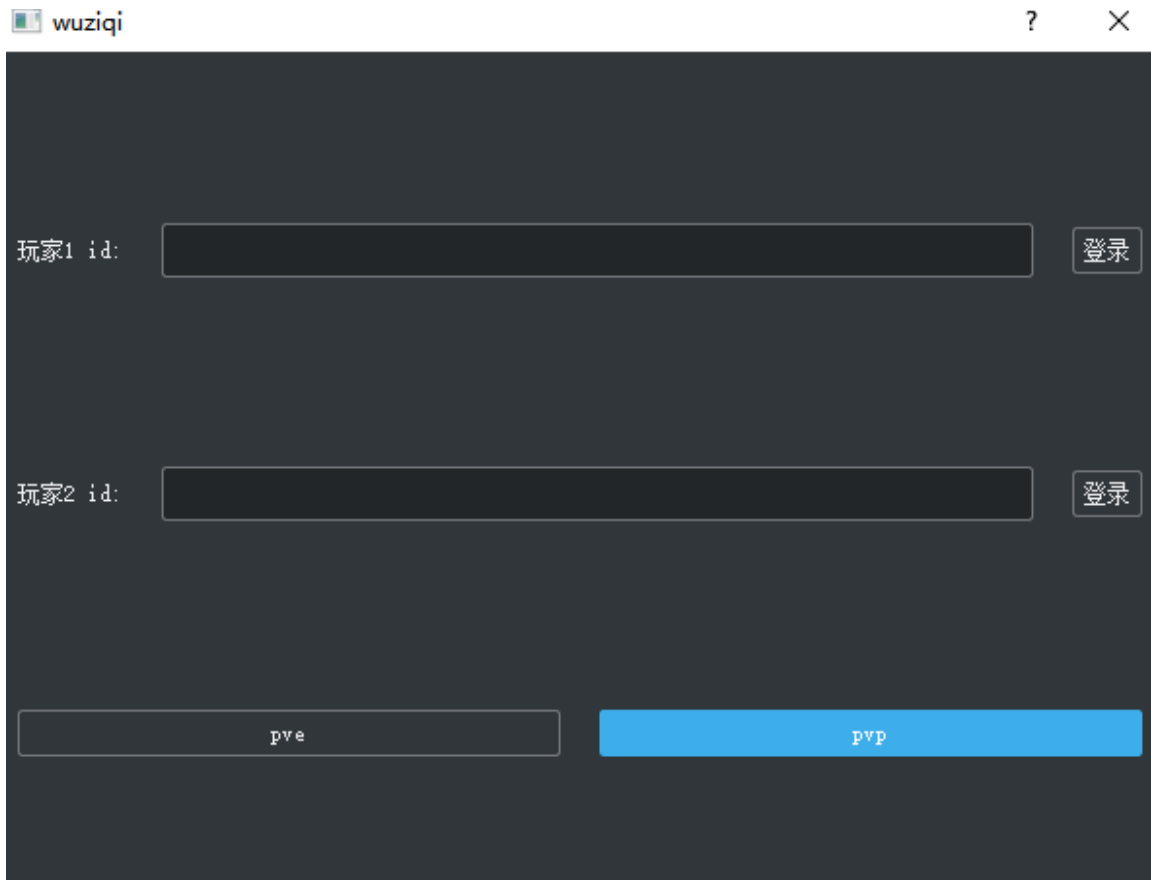
这是一个基于 Qt Creator 平台开发的简易五子棋对战游戏，能选择人机模式和玩家对战模式，系统能自动判断胜负并记录。

二、程序界面与规则

在登录界面中，可以选择 Pvp 模式或 Pve 模式，选择后，可以在输入玩家的用户名，Pvp 模式需要两个玩家都按下登录按钮后才能进入主界面。默认为玩家 1 黑棋，玩家 2 白棋。Pve 模式时玩家为先手。当登录完成时，登录界面关闭，同时出现主游戏界面。



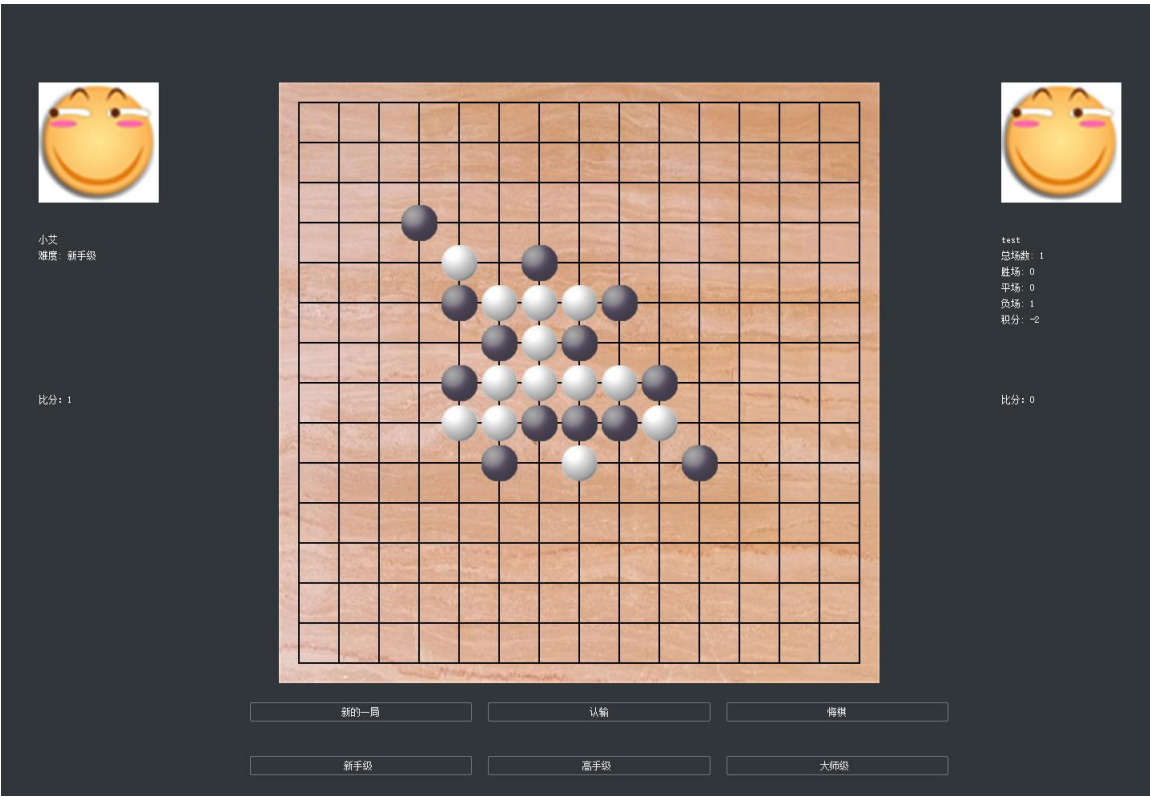
(pve 模式下的登录界面，只需要一位玩家 ID)



The screenshot shows a dark-themed login window titled 'wuziqi'. It features two input fields for player IDs. The first field is labeled '玩家1 id:' and has a '登录' (Login) button to its right. The second field is labeled '玩家2 id:' and also has a '登录' button to its right. At the bottom, there are two buttons: 'pve' and 'pvp'. The 'pvp' button is highlighted in blue, while the 'pve' button is dark grey.

(pvp 模式下的登录界面，需要两位玩家的 ID)

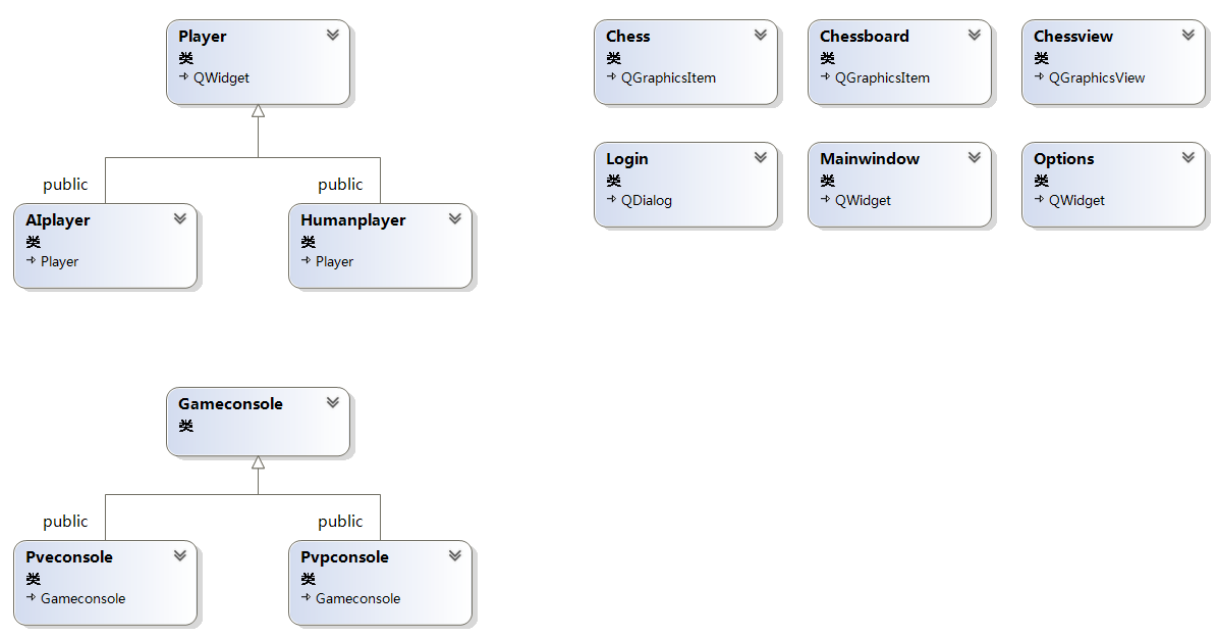
在主界面中，可以选择重新开始，悔棋，和认输。PvP 模式中，无法在游戏中选择再来一局，可以选择认输结束当前局。当游戏结束时，无法选择悔棋和认输，可以选择再来一局开始新的一局。玩家信息在游戏结束时更新。主界面包含两个玩家的信息，其中 ID 在登录时确定。Pve 模式中，可以随时选择再来一局。当游戏结束时，无法选择悔棋和认输，可以选择再来一局开始新的一局。另外，在 Pve 模式中，可以选择三个难度的电脑。主界面同样包含了两个玩家的信息，如果为 Pve 模式，AI 玩家的信息由系统设定。



（对弈时的界面）

三、程序界面的代码框架

整个程序界面基于 Qt5 开发，完全采用面向对象的方法进行程序设计。整个类的设计如下：



Login 类继承 QDialog 类，用于实现登录界面。类中包含了相应的控件。通过四个槽函数来完成模式选择和玩家信息的确定，一旦确定，主界面的构造发生在槽函数中。Mainwindow 类继承 QWidget 类，是主游戏窗口的实现，Gameconsole 类是游戏控制的抽象类，用于判棋，下棋的处理。Pveconsole 类和 Pvpconsole 类继承了 Gameconsole 类，前者处理 Pve 模式的游戏控制，后者处理 Pvp 模式的游戏控制。

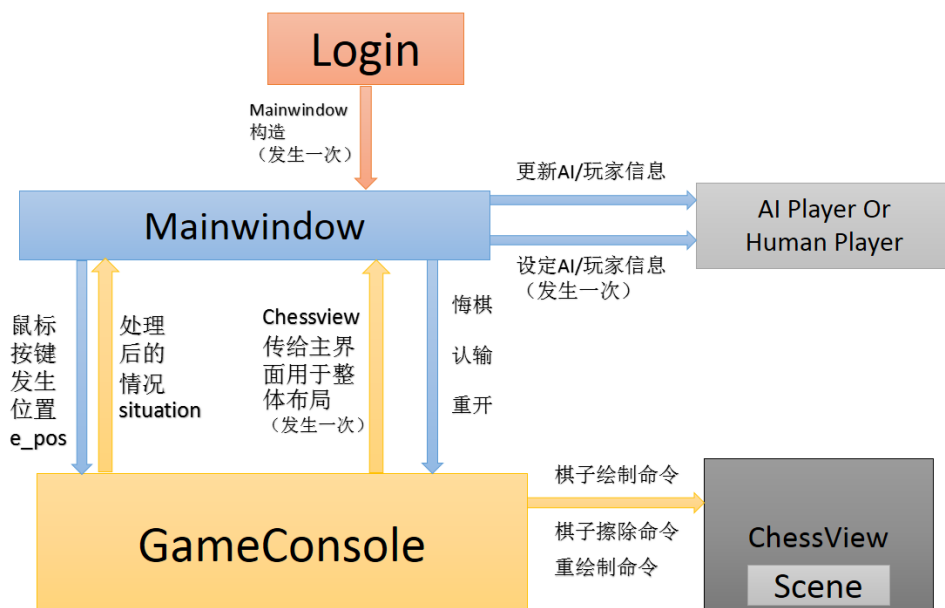
Player 类继承 QWidget 类，是用于记录和显示玩家信息的抽象类，作为主界面的子窗口，被 AIplayer 和 Humanplayer 类所继承，分别显示 AI 信息和玩家信息。Options 类继承 QWidget 类，同样作为主界面的子窗口，包含了所有的选项按钮，所有选项的布局和信息都由 Options 完成。

Mainwindow 类的私有成员中有两个 Player 对象，一个 Options 对象，构成主窗口的子窗口关系，而且选择 AI 难度需要点击 Options 中的按钮，而更新 AI 信息需要调用 Player 类的更新函数，二者的通信可以通过在 Mainwindow 写槽函数来完成，分别为 easy_clicked(), mid_clicked(), hard_clicked(); 因为鼠标按键 mousePressEvent 是在 Mainwindow 接受的，鼠标按键的位置需要发给控制部分，Options 中的悔棋，认输，重开局按钮也需要发送给控制部分处理，所以采取让一个 Gameconsole 对象作为 Mainwindow 的私有成员的方式。两个 Player, Options 和 Gameconsole 在 mainwindow 构造的时候同时构造，并确定是哪种模式。

Chess 类的棋子图元类，用于绘制棋子，Chessboard 是棋盘图元类，用来绘制棋盘。二者都继承 QGraphicsItem 类。

Chessview 继承 QGraphicsView 类，即用来下棋的界面采用视图和场景，同时因为场景和棋盘只需要在视图建立是建立一次就好，而棋子需要动态操作（绘制和擦除），所以 Chessview 的私有成员包含一个 scene 和 chessboard，他们在 Chessview 的构造时只构造一次。Chessview 提供了 canDraw 接口，用于判断进行鼠标操作时是否在视图的棋盘内部。DrawChess 接口用于下棋请求时的绘制棋子，eraseChess 接口用于悔棋请求时擦除一个棋子，reInit 用于重开局请求时的清光棋子操作。三个函数均只由 Gameconsole 类的对象来调用，所以一个 Chessview 对象作为 Gameconsole 的私有成员，但 Chessview 需要布局在主窗口中，所以 Gameconsole 类中使用 getChessview 函数传给 Mainwindow 类用来布局，并且仅在此情况下需要传出 Chessview 对象。

整个流程如下：



四、游戏逻辑

- 1. 点击落子：**鼠标在棋盘界面单击后，会有函数响应记录此时的鼠标位置，并将实际位置转化为相对应的棋盘格点位置，若该格点为空，则设置该点有一个对应颜色的棋子，并继续游戏。
- 2. 判断输赢：**因为棋盘信息在程序中是以二维数组的形式存储，因而每一次判断棋盘局面都遍历整个二维数组，查找是否存在连续的五个同色棋子。如果存在，则游戏结束，若不存在，更改下棋方，游戏继续。
- 3. 悔棋：**程序有一个栈用来记录每一次下棋的点的位置，当点击悔棋并确认后，将弹出栈顶的点的位置，并将该点上的棋子抹去，实现悔棋功能
- 4. AI 计算落子点的简单算法（基础级）：**

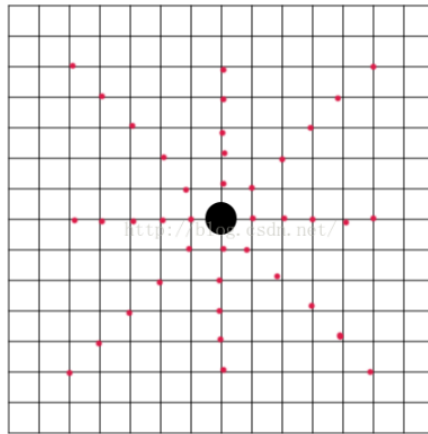
函数原型 `AiGoBasic(int& t, int &h)` 通过引用两坐标，确定下棋位置。

BasicAI 算法实现：定义棋型数组 `GameSituation[2][15][15][8][2]`，每一个五元组就是一种局势，[2]区分白棋子和黑棋子[15][15]空格位置 Y 轴坐标 / 空格位置 X 轴坐标[8]区分八个方向[2]用于判断边界问题

算法思路：

- 先利用 `int GameSituation[2][15][15][8][2]` 来记录所有空格沿八个方向，白棋和黑棋所连续相连的个数，并记录边界点的情况。

B. 根据个数多少和边界情况进行评分。即下完棋后，以该点为中心统计周围 5 个点的分数（八个方向）。即红点，统计每个点的分数。



C. 先后找出白棋和黑棋评分最高的点, 对于己方来说, 若发现对方有所下点比己方评分最高点高时, 即下对方评分高的那点; 反之, 下己方评分最高达点。

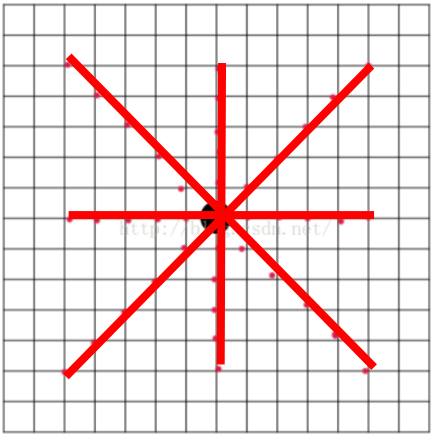
5. AI 计算落子的另一种方法 (大师级):

设计五子棋 AI 首先需要为各种棋型进行匹配。以白方为例，首先根据黑子的位置将棋盘上的一条直线分割成若干段，然后对分割出来的片段进行匹配。由于只有白子和空位，棋型匹配只需要观察直线上 6 个连续的位置的棋子状态，如果长度超过 6 个，则以 6 个位置为单位依次检查。如果长度为 5，则做一些特殊处理，而长度小于 5 的不予以考虑。

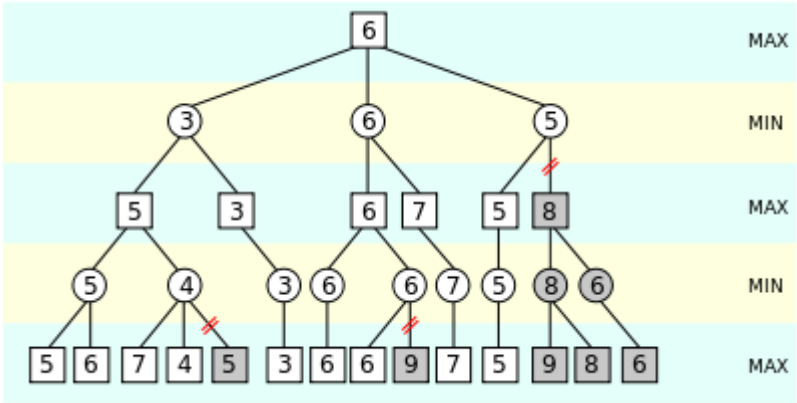
001110010011 \rightarrow 001110010011

由于限制了位数，为了加快匹配速度，这里使用了打表的方法。即生成一个长度为 64 的数组，同时将棋型看作一个二进制串白棋为 ‘1’ 然后直接查表就可以得到棋型。这一个 AI 的主要逻辑是利用 alpha beta 剪枝写的，为此需要估值函数。为了下棋的严谨性，在估值时会有一定范围（范围是这个 AIValue 类的一个参数）进行多方位的扫描，找到最具价值的棋型，这是递归最底层的估值函数。但是由于递归涉及到的位置非常多，递归数量呈指数增长，这样操作会非常消耗时间，因此需要快速地找到可能会考虑的位置，因此需要对某一个位置进行估值。具体方法是：对该位置，首先在这里下黑棋或者白棋（两方面都要考虑），无论下黑棋还是白棋的估值高，这就是一个候选点。估值方法就是对经过这个

位置的四个方向进行上述模型匹配，并找出最有价值的模型，算出分数。这样找出候选点之后，选出分数最高的几个进行递归，这样就可以大大减小递归的数量。



而剪枝部分，则需要每一层都将目前的最值传递到下一层，比如在 min 层中，上一个 max 层传递来的值已经大于当前 min 层的某一个值，就不需要再递归求出剩下的值了。排除一些特殊情况，通过这样剪枝求最值就能算出应该下的位置。



五、任务分工：

唐作鑫 3150101226：AI 算法，实验报告

李小艾 3150101291：AI 算法 实验报告

吴桐 3150104174：程序界面设计 实验报告

孙健 3150103723：程序界面设计 实验报告

注：界面的背景主题来源于网络上的 QSS 并做了修改