

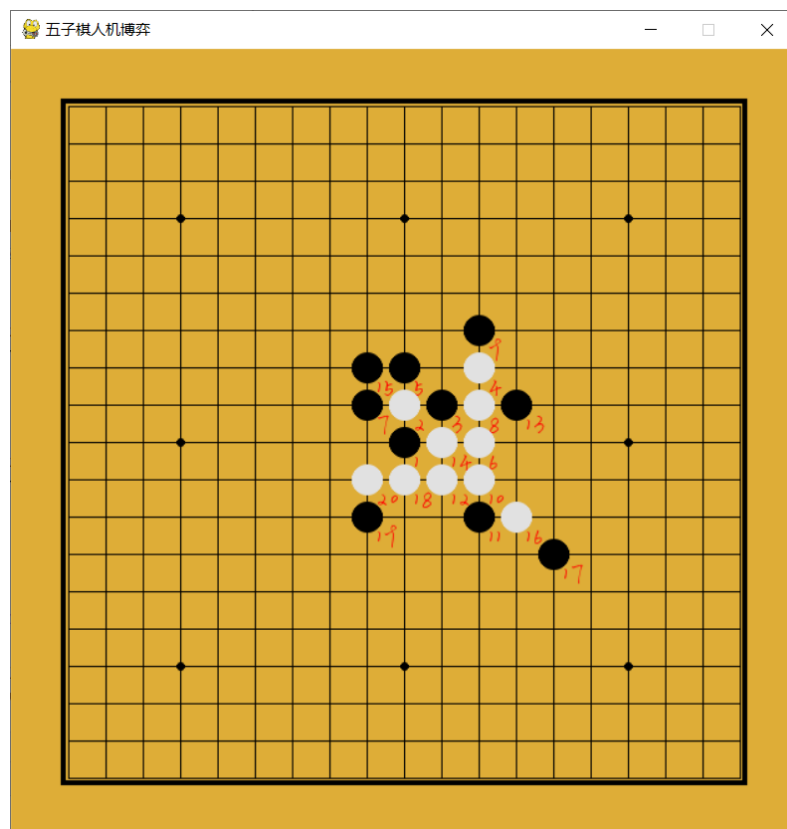
1. 程序设计功能介绍

- (1) 五子棋基础功能：
 - 1. 玩家落子：玩家自主选择落子位置；
 - 2. 回合交换：玩家与电脑交替落子；
 - 3. 胜负判定：同颜色 5 子一线即获胜
- (2) 友好的用户界面：鼠标点击落子；
- (3) 响应时间较快；
- (4) AI 搜索求解策略落子：设置评价函数，启发式搜索求解；
- (5) 落子顺序标注复盘：落子后标记是第几手落子。

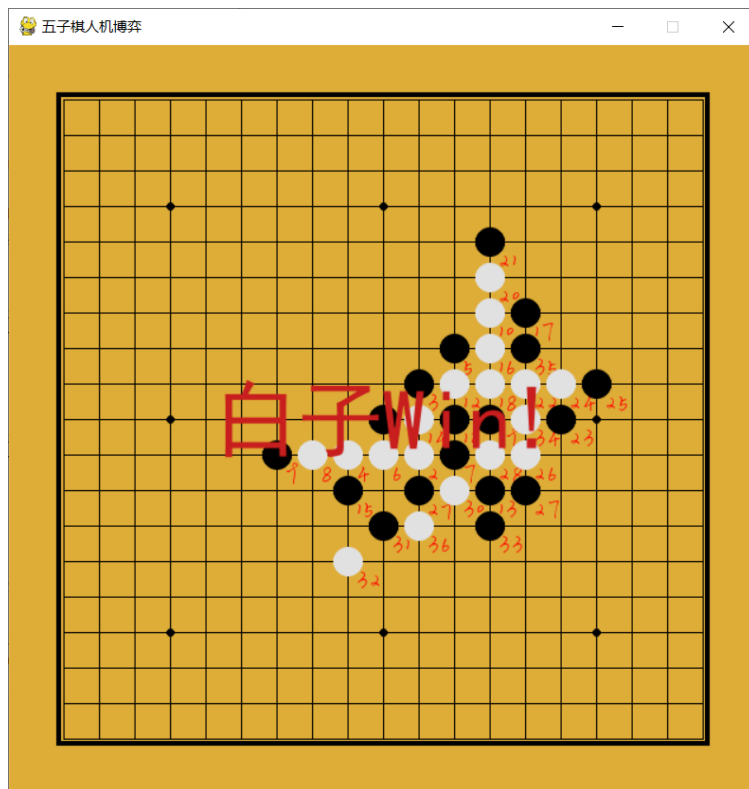
2. 实验平台及语言

- (1) 实验平台：Pycharm
- (2) 编程语言：Python
- (3) 库：pygame 及系统库

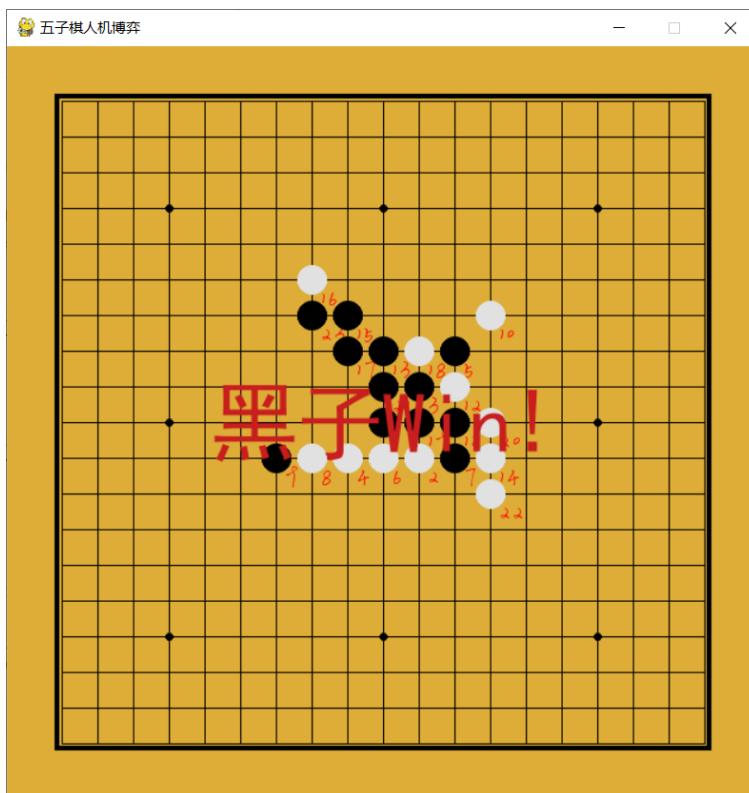
3. 程序运行界面展示



运行界面 1



运行界面 2



运行界面 3

4. 程序设计简述

4.1 五子棋基础功能

(1) 落子回合交换

```
def player_turn_change(current_player):
```

(2) 棋盘绘制

```
def draw_checkbox(screen):
```

(3) 棋子绘制

```
def draw_pieces(screen, point, stone_color):
```

(4) 标记落子顺序

```
def record_turn(screen, point):
```

(5) 局势情况

```
class Checkerboard:
```

① 判断是否可落子

```
def can_drop(self, point):
```

② 判断是否分出胜负

```
def _win(self, point):
```

4.2 外设交互:

(1) 捕捉鼠标点击, 决定落子位置

```
def get_mousecheck(click_pos):
```

(2) 输出文字信息

```
def outtext(screen, font, x, y, text, fcolor=(255, 255, 255)):
```

4.3 搜索策略

```
class AI:
```

① 求某方向的得分

```
def line_score(self, point, x_direction, y_direction):
```

② 求某落子位置得分

```
def get_score(self, point):
```

③ 在得分最大位置落子

```
def AI_drop(self):
```

4.4 数据结构定义:

```
Chessman = namedtuple('Chessman', 'Name Value Color')
```

```
Point = namedtuple('Point', 'X Y')
```

```
PLAYER_TURN_CNT = 0
```

```
BLACK_CHESSMAN = Chessman('黑子', 1, (0, 0, 0))
```

```
WHITE_CHESSMAN = Chessman('白子', 2, (225, 225, 225))
```

```
chess_box = [[0 for x in range(19)] for y in range(19)]  
direction = [(1, 0), (0, 1), (1, 1), (1, -1)]
```

4.5 参数设置:

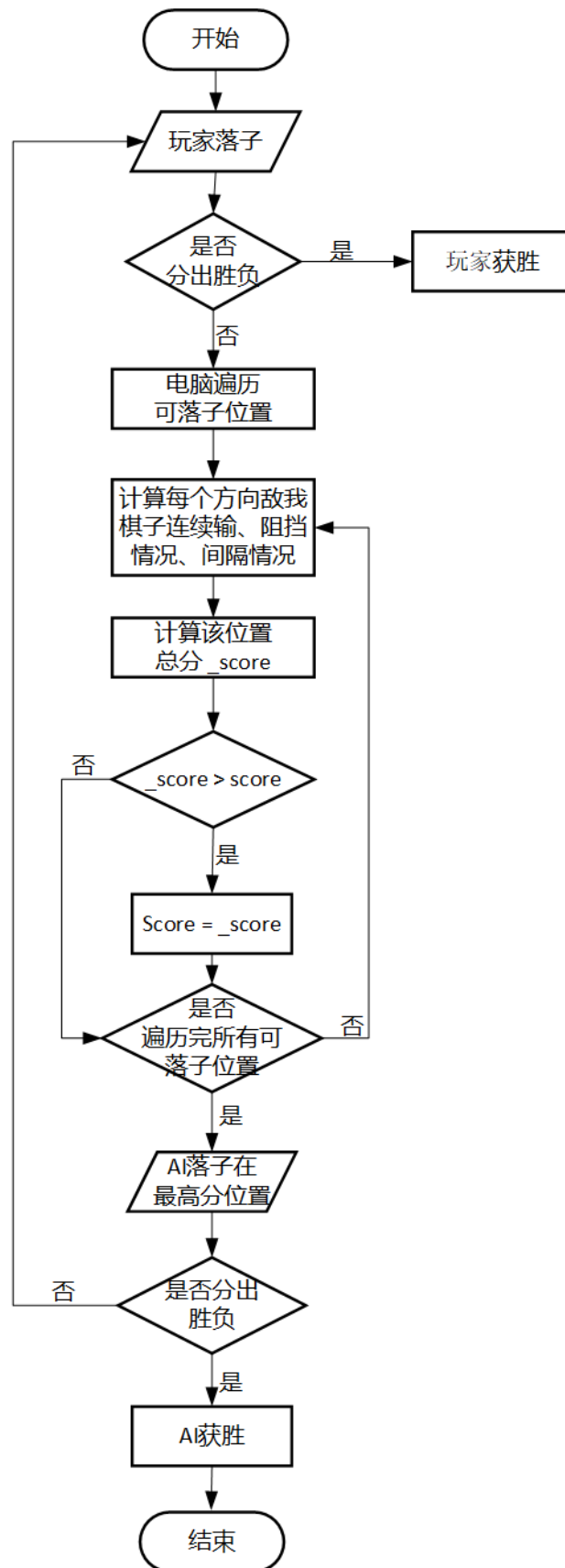
(1) 棋盘棋子大小规格参数:

```
SIZE = 30  
linecounts = 19  
Boxwid = 4  
Border_Length = SIZE * (linecounts - 1) + 4 * 2 + Boxwid  
Start_X = Start_Y = 40 + int(Boxwid / 2) + 4  
screenH = SIZE * (linecounts - 1) + 40 * 2 + Boxwid + 4 * 2  
screenwid = screen
```

(2) 颜色参数

```
Stone_Radius = SIZE // 2 - 3 # 棋子半径  
Stone_Radius2 = SIZE // 2 + 3  
Checkerboard_Color = (222, 173, 55) # 棋盘 RGB  
BLACK_COLOR = (0, 0, 0)  
WHITE_COLOR = (255, 255, 255)  
RED_COLOR = (200, 30, 30)  
BLUE_COLOR = (30, 30, 200)
```

5. 算法流程图

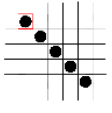
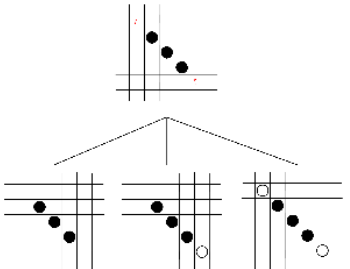
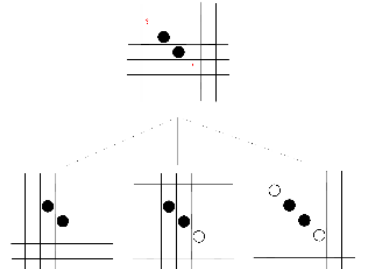
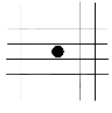
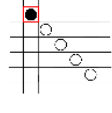
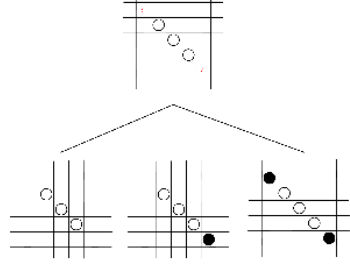
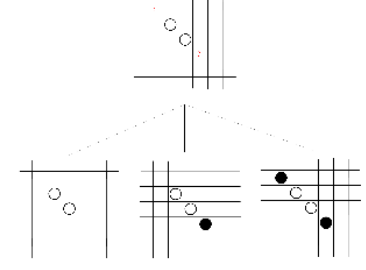
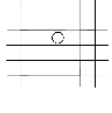


6. 评价函数

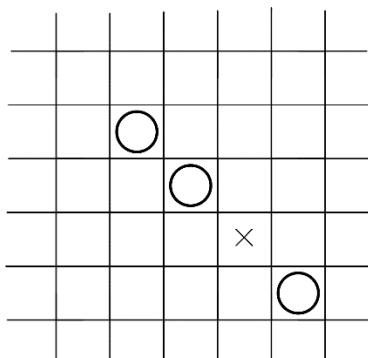
6.1 评价函数的设置

- 结合本人以往五子棋博弈的经验，设置不同情况下的评价函数值：
- (1) 若我方四子相连，落一字即取胜，设置为最高分；而敌方四子相连，必须堵，分数仅次之；
 - (2) 若我方四子在任意方向连成一线，则获胜率最高，因为此时敌方无论在连线的任意一端有落子，我方都会胜；
 - (3) 若我方三子连续，则敌方在连线的两端落子数越多，我方获胜率越小；
 - (4) 若我方二子连续，获胜率较三子连续相比更小，并且敌方在连线的两端落子数越多，我方获胜率越小；
 - (5) 若我方周围没有连续棋子，那么无论敌方在周边哪里有落子，我方获胜率都很小；
 - (6) 若敌方四子连续，则差一步就要胜利，此时我方落子在连线的两端对我方利益最大。由于自身取得优势大于让对方劣势，评价函数值略小于情况（1）；
 - (7) 若敌方三子连续，我方在连线的两端落子可减小敌方获胜率；
 - (8) 若敌方二子连续，此时情况不那么紧迫，在连线两端落子对我方提高获胜率的影响较小，但依然可减小敌方获胜率；
 - (9) 若敌方周围没有连续棋子，我方在周边落子可略微减小敌方获胜率，但作用较小。

6.2 评价函数值表：

	四子连续	三子连续	二子连续	单子
我方				
评价函数值	10000	1000 100 0	100 10 0	10
敌方				
评价函数值	9000	900 90 0	90 9 0	9

特别地，当我方连续子之间存在间隔，即如下图所示：



得分 score 要变为原 score 的 0.5 倍，同理，敌方连续子之间存在间隔 score 变为原 score 的 0.4 倍；当同时为双方连续子交叉的含间隔位置时，score 变为原 score 的 0.8 倍。

7. 优化点

考虑到项目主要是为了实现启发式搜索策略的应用，受时间限制，此项目还未充分考虑将游戏所有功能最优化，在此基础上，可优化的方向为：

- (1) 完善规则，设置禁手以及三手交换五手两打规则；
- (2) 增加悔棋功能，将每次落子入栈，玩家悔棋将两次落子出栈；
- (3) 玩家可选择先手或者后手；
- (4) 估价函数设置，将双活三、活三四等交叉落子设置较高的评价函数。