计算概论 A 期末大作业

指 导 教 师：王厚峰
小 组 成 员：张凯越 周子瑄

二〇二二年 十二 月

一、简介

本次大作业，我们用 C++实现玩家与 AI 交互完成“不围棋”游戏。

算法部分，基于避免自杀、困子的评价体系，我们采用贪心算法对棋局己方可以下的位置进行评估，使用贪心算法选择局部最优策略。按照考虑到贪心算法在前期具有高度随机性，我们手动编排了前期抢占边缘有利位置的算法，在十二步以后使用贪心算法，提高 AI 的下棋胜率。

界面设计部分，用 ege 实打印背景图像、获取鼠标信息、播放音乐等功能。绘制棋子棋盘时，没有采用 ege 绘图的方法，而是在其他软件中合成好图像，再直接调用插入（减少代码量，比较好看且直观）。我们没有采用键盘输入的方式，而是用“鼠标点击”的方式代替，这样避免了大量的非法输入，同时增强了游戏体验感。

二、功能设计

1. 基本功能

（1） 存盘：按“保存”键可保存当前棋局信息

（2） 读盘：按“读取”键可读取之前存储的信息

（3） 重新开始：一共有两处可以选择，游戏过程中可点击右下角“重新开始”开始新游戏，一局棋下完（胜负已分时）也可以点击“重新开始”。

（4） 退出游戏：刚开始程序时可点击“退出”，游戏过程中可点击右下角“退出”，一局棋下完（胜负已分时）若不点击“重新开始”也会在 10 秒后自动退出。

2. 背景音乐

在进入游戏界面后，背景音乐会自动开始播放，玩家可按“音乐暂停/播放”键暂停音乐，再次点击时音乐继续播放

3. 模式选择

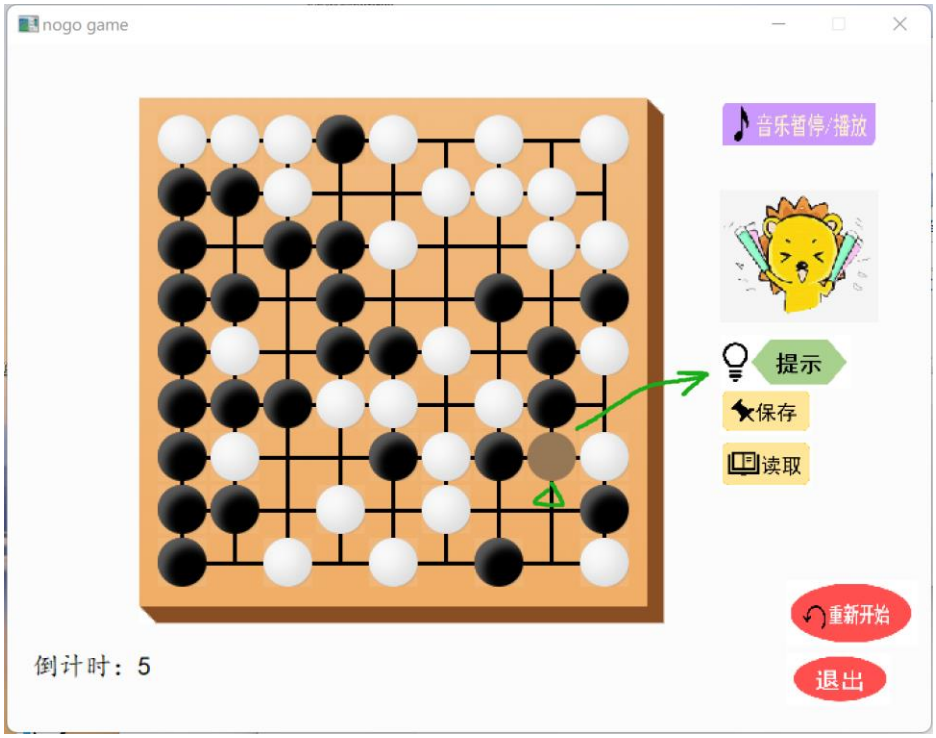
（1） 玩家可以选择“单人模式”和“双人模式”，单人模式即玩家与 AI 对弈，双人模式即两个玩家下棋或单人自娱自乐。双人模式中，AI 一般不参与，但玩家若点击“提示”，AI 也会根据棋局做出提示（见 4.提示）。

（2） 若玩家选择“单人模式”，界面会出现新的选择按键，玩家需选择“先手”或“后手”，若选择“双人模式”，则改界面不会出现。

（3） 选择结束后，倒计时 5 秒后自动进入游戏界面（棋盘）。

4. 提示

若点击“提示”键，AI 会站在玩家立场给出提示，提示的位置会用暗色棋子（与黑白做区分）显示出来，玩家可以采纳 AI 的建议，也可以不采纳。



5．规则提示

（1） 在第一个界面选择“新游戏”后，将跳转至第二界面，对不围棋的游戏规则和我们自己设计并增加的功能进行说明。

（2） 在“模式选择”过程中，若玩家选择“双人模式”或“单人模式 先手”，会出现进一步的游戏规则说明，“第一手请勿下在棋盘正中央”，如果选择“单人模式 后手”，则不会出现此提示。

6. 倒计时

在下棋过程中，玩家每一步有 10 秒的思考时间，界面左下角会显示倒计时（还剩几秒），若时间到了玩家还没有下棋，会提示“思考时间到，请尽快下棋！”，玩家下棋后该语句会消失。

7. 违规下棋提醒

游戏采用“鼠标点击棋盘格点”的方式传达指令，若玩家点击的位置已经有棋子，会出现提示“该位置已有棋子”，当玩家点击合法位置后，该语句也会自动消失。

8. 其他交互与信息反馈

（1） 在“模式选择”等界面中，系统会对人的选择做出回应，如“成功选择单人模式”。

（2） 在“游戏规则说明”界面，玩家阅读完游戏须知后，根据提示按空格键，进入下一个界面。

（3） 在胜负已分时，系统弹出“赢家说明”，若玩家想继续游戏，可点击“重新开始”，进入“游戏模式选择”界面，否则十秒倒计时（会显示）后，自动退出游戏。

三、代码架构

1. 函数主要分为六类：

（1）主函数

（2）打印界面的函数。共四个主界面，还有一个函数用于擦除原来打印的文字或图像

（3）与人交互的函数。获取鼠标信息，根据人的点击情况调用其他函数做出反应

（4）下棋的函数。调用相应的算法函数，判断棋子是否合法，打印棋子图像等

（5）菜单功能函数。实现存盘、读盘、重新开始、音乐暂停、提示等功能

（6）算法函数。判断是否有气，判断是否自杀，前 12 步采用抢位算法，后面棋子信息足够时采用贪心算法

2. 界面函数编写主要思路

（1）在主函数等几个涉及与人的交互的函数中，主体为一个 for 循环，不断监测鼠标信息，其中嵌套 if 等条件判断语句，根据鼠标位置判断选择的功能或落子的位置，再调用其他功能函数或下棋的函数。若点击“退出游戏”则直接 return 0 结束 main 函数，否则一直在 for 循环内部。

```
827     for (; is_run(); delay_fps(60))
828     {    //is_run()函数判断ege窗口是否还存在，delay_fps(60)指每1/60秒监听一次用户是否点击鼠标，
829         //间隔时间太长程序运行不灵敏，太短连续多次点击程序易崩溃
830         runTime = fclock();
831         remainTime = endTime - runTime;//当前时间就等于两个时间差
832         erase(13, 447, 130, 481);
833         xyprintf(19, 460, "倒计时: %d", (int)ceil(remainTime));
834         if (remainTime <= 0) { ... }
842
843         while (mousemsg())//getmouse 获取鼠标消息
844         {
845             msg = getmouse();
846             //xyprintf(80, 20, "鼠标位置:    x    = %4d     y = %4d", msg.x, msg.y);
847             if (msg.x > 540 && msg.x <= 660 && msg.y > 40 && msg.y < 80 && msg.is_down())
848                 music_judge();//暂停或播放音乐
849             if (msg.x <= 610 && msg.x > 540 && msg.y > 260 && msg.y < 300 && msg.is_down())
850                 save();//存盘
851             if (msg.x <= 610 && msg.x > 540 && msg.y > 300 && msg.y < 340 && msg.is_down())
852                 read();//读盘
853             if (msg.x > 540 && msg.x <= 640 && msg.y > 210 && msg.y < 260 && msg.is_down())
854                 prompt();//提示
```

（2）玩家点击某一格点后，根据公式算出对应的位置坐标，再在该位置打印棋子图像并调用算法函数回应。

```
308         x = (msg.x - 118) / 40 + 1;//对应格点x编号
309         y = (msg.y - 59) / 40 + 1;//对应格点y编号
333         else if (chess_color == 2)
334         {
335             putimage(114 + (x - 1) * 40, 53 + (y - 1) * 40, pimg_white);
336             board[x][y] = 2;
337         }
```

（3）在倒计时等功能中，需要不断在同一位置打印不同数字，为防止出现“叠加”效果，加入了 erase 函数，不断用相应大小的白色矩形进行覆盖。
（4）在 human_play 和 ai_play 两个函数中，首先判断是否出现自杀等可以分出胜负的情况，若有，进入显示赢家界面，同时根据玩家的进一步选择返回相应的值，否则打印棋子，并将棋子颜色变为相反的（chess_color = chess_color % 2 + 1;）。

四、AI 算法

介绍你使用的围棋算法
贪心算法

对于不围棋而言，自杀和困另一方的棋子均属失败。对于一个棋局，我们定义可下子为：避免落子导致自杀和困子的前提下，在棋局能下的子。可下子比对方多就意味着较对方处于有利地位(因为到最后己方仍有可下子而对方没有可下子，而对方又不得空手，所以只能下必败的子，导致己方获胜)。

基于这样的想法，我设计了判断自杀和困子的函数，并将其包装在判断**相对可下子 put_OK** 的整体函数中。己方先遍历棋局，找到所有可下子，并对每种落子导致的新棋局中的相对可下子：(己方可下子数目-对方可下子数目)进行评估。新棋局中的相对可下子数目越多，意味着这种下法越优。当局面出现多个同样优的下法时，随机选择一种决策。

在另一方面，贪心算法有着鲜明的局限性：决策的科学程度依赖于信息量的多寡，这意味着在前期双方落子数目较少时，其决策近似随机。为了尽量克服这一缺陷，我们手动了前 12 步的算法，旨在占领公认的优势边缘位置。等到双方有一定量的落子，信息量增加后，再转为贪心算法。最终取得更好的决策效果。

五、实验结果

1. 程序运行结果：
一切正常 面测效果较好，实现了所有预期基础功能。我方随机落子，AI 用算法落子，程序及时判定了 AI 获胜，页面显示了 “AI 获得胜利”。
2. botzone 比赛结果：
正常完成了测试赛

六、总结

本次大作业，我们前后总共花了约一个月的时间完成。期间经历了两人轮流发烧，中途决定改变算法（由 MCTS 改为贪心），界面突然无法显示（后来 debug 发现是宏定义的问题）等小插曲，最后在共同努力下顺利完成所有要求的内容，并在游戏设计中加入了自己的创意和想法。

其中，张凯越主要完成算法研究和实现；

周子瑄主要完成界面设计和基本功能实现。