大作业报告: 同化棋

姓名: 张龄心 学号: 2100013018

一、 简介

本作业实现了一个同化棋的小游戏,可以进行不同难度的人机对战和双人对战。

Al 算法方面,采用 alpha-beta 剪枝,搜索时间卡 0.85s,搜索深度随棋局进展而变化,变化的方式按 botzone 上表现最佳的算法。

人机交互方面,使用 EasyX 实现了可以鼠标交互的图形界面,包括难度选择、AI 提示、悔棋、存档和读档、音乐、实时棋子数目显示等功能。

创意在于采用了卡通形象的棋子, 并据此调整了整个游戏外观的风格。

(*需要用 Visual Studio 运行 sln 文件,如果直接打开 exe 文件无法显示图片素材)



二、 AI 算法简介

三个难度中,低难度采用的算法是一步贪心;中难度和高难度采用的是三层 alphabeta 剪枝搜索,估值算法返回场面双方棋子的数目差值,如果检测到游戏结束(棋盘摆满或一方无路可走或一方棋子数目为 0)就判断胜负,并返回 INF 或-INF。其中,中难度固定搜索深度为 3,高难度为动态搜索层数。

高难度的搜索层数 depth 随棋子数 n 的变化如下:

- ① n≤36, depth=5;
- ② $37 \le n \le 41$, depth=6;
- 3 42 \leq n \leq 44, depth=7;
- ④ n≥45, depth=8;

经过在 botzone 上的试验,这样的层数递进既能保证在规定时间内尽量完成搜索(不必提前退出),又能在中后期有比较优秀的表现。

```
OU /
    //判断玩家输入是否合法
608

★ int legal(int player, int sx, int sy, int nx, int ny)

609
619
    //落子并同化
620
   621
642
    //ai部分(alphabeta剪枝+动态层数)
643
   #void walk() { ... }
644
671
    //alphabeta递归函数
672

★ int albt(int depth, int alpha, int beta) { ...
673
697
    //寻找合法走步
698
   699
735
    //模拟落子并同化
736
   737
765
    //估值函数(贪心)
766
   767
```

Alphabeta 函数的代码如下:

```
//alphabeta递归函数
if (depth < 0)
           return evaluate();
       int val;
      //交换走的一方
      sidetomove = 3 - sidetomove;
int thisside = sidetomove;
      //如果棋盘满了则判断胜负,返回一个很大或
if (fakepiecenum[depth + 1][1] + fakepiecenum[depth + 1][2] == 49) {
  if (fakepiecenum[depth + 1][sidetomove] > fakepiecenum[depth + 1][3 - sidetomove])
                return 500;
           else return -500;
      //找出所有合法棋步
       {\tt memcpy} (fakeboard[depth], \ fakeboard[depth + 1], \ sizeof(fakeboard[depth + 1]));\\
      findlegalmove(depth);
       //如果无路可走
      if (fakepiecenum[depth + 1][sidetomove] > fakepiecenum[depth + 1][3 - sidetomove])
                     return 500;
                else if (fakepiecenum[depth + 1][sidetomove] < fakepiecenum[depth + 1][3 - sidetomove])
                     return -500:
                //如果双方棋子数目相同, 白方胜利
                if (sidetomove == 2) return 500;
else return -500;
       //按顺序搜索所有合法棋步, alphabeta剪枝
      for (int i = 1; i <= movenum[depth]; i++) {
    memcpy(fakeboard[depth], fakeboard[depth + 1], sizeof(fakeboard[depth + 1]));
    memcpy(fakepiecenum[depth], fakepiecenum[depth + 1], sizeof(fakepiecenum[depth + 1]));
           fakeputpiece(depth, i);
val = -albt(depth - 1, -beta, -alpha);
           sidetomove = thisside;
if (val >= beta) return beta;//alpha-beta剪枝
if (val > alpha) alpha = val;
      return alpha:
```

三、 人机交互

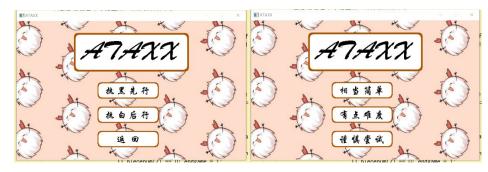


(1) 主界面: 对应函数 homeitf()

有"人机对战""双人对战""读取存档"三个选项。

a) 人机对战

玩家点击后出现"执黑先行""执白后行""返回"三个选项。 选择先后手后,出现三种难度选择。选择难度后游戏开始。



b) 双人对战

玩家点击后直接开始游戏。

c) 读取存档

玩家点击后出现"人机存档""双人存档""退出"三个选项,选择即可继续之前保存的游戏进度。

*本游戏为人机对战和双人对战各保存一个存档,新存档会覆盖旧存档。



(2) 游戏界面

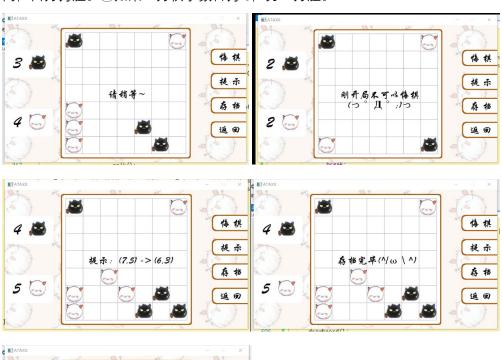
a) 人机对战,对应函数 boarditf_human_ai()

界面中央是棋盘,右侧有"悔棋""提示""存档""返回"四个按钮,左侧是双方当前棋子数目。

- ① 移动棋子: 玩家点击要移动的棋子后再点击要落子的位置即可, 点击后界面上会短时间显示"请稍等", 然后显示出 ai 走步。(图 1)
- ② 悔棋:玩家点击悔棋即可回退到上一轮状态。 *如果开局状态下悔棋.会提示"开局不能悔棋".点击任意位置后消失。(图 2)
- ③ 提示: 玩家点击"提示"后, 棋盘中央会出现 ai 计算的最佳走步, 点击任意位置后消失。(图 3)
- ④ 存档: 玩家点击"存档"后,自动进行存档,并显示"存档完毕"。(图 4) *下次在主界面点击"读取存档"即可继续游戏。
- ⑤ 返回:点击后返回主界面,当前游戏进度不保存。

当游戏结束(棋盘填满/一方棋子数目为 0/一方无路可走),会显示游戏胜负,点 击任意位置返回主界面。(图 5)

胜负规则:①棋盘填满/一方无路可走,棋子数目多的为胜;如果双方棋子数量相同,白方为胜。②如果一方棋子数目为 0,另一方胜。





b) 双人对战,对应函数 boarditf_human_human() 界面和功能同上,区别在于结束时显示的胜利提示不同。



四、音乐和外观

音乐使用 mciSendString 函数进行播放; 棋子的绘制使用图片素材,配色和背景尽量与之相符。 字体设置为华文行楷,使用 EasyX 库的 settextstyle 函数。