

2020 计算概论大作业实验报告

一、程序概述

在这款游戏中，玩家会和电脑进行不围棋的博弈。程序中含有菜单选择（开始新游戏、继续游戏、保存并退出游戏）等人机交互方式，完全实现了作业的基本要求，总代码量达到一千一百余行，并在基本要求之上加入了更有利于玩家体验的特色功能。本程序历时近两个月完成，在此期间进行过六次大规模改动和二十余次小规模修改，目前的版本号为 5.2.2。特别鸣谢：北京大学信息科学技术学院、Microsoft Visual Studio、CSDN、botzone、知乎、中国知网、easyX 插件等对程序编写提供的帮助。

二、特色功能简介

本不围棋程序在实现基础功能的前提之上，根据笔者对于棋类博弈程序的认知，加入了八项有利于增加玩家体验的特色功能，简介如下：

（1）**图形界面、鼠标交互。**整个程序都是以图形界面的形式呈现，并且完整的进行了鼠标交互的支持，极大地提高玩家体验感。

（2）**背景音乐和音效。**本程序采用了一首中国古风音乐作为循环播放的背景音乐，以体现不围棋游戏背后所蕴含的以和为贵等中华传统精神。此外，当玩家进行鼠标点击以及电脑决策落子时，都会播放“落子”音效，以模拟对弈过程；当玩家胜利或者失败时，也会播放不同的音效。

（3）**规则介绍和致谢。**当玩家点击“开始新游戏”后，程序会显示“不围棋”游戏的基本规则，以帮助新玩家更快了解游戏，提高用户体验。此外，程序还会显示致谢内容。

（4）**先后手选择。**在不围棋对弈中，先手执黑棋，后手执白棋，玩家在开始游戏之前可以选择先手或者后手。

（5）**难度选择。**本不围棋程序提供了三种难度的选择，分别为初级、中级、高级，从而能更好的适用于不同层次的玩家。

（6）**游戏结束自动存盘。**在游戏结束（无论输赢）后，系统都会自动存盘，再次打开后可以通过点击“继续游戏”来复盘，便于玩家分析胜利或失败原因。

（7）**记事本存档。**使用记事本进行对局存档，保存成为 .txt 文件，可以由玩家对其进行自由检查。

（8）**自由存档。**玩家可以在游戏的任意回合对局面进行存档。

三、函数介绍

在本不围棋程序中，笔者编写了若干个实现程序所必须的函数。下面对这些函数做出简要说明和介绍。

（1）`int startMenu();`

函数功能：游戏开始界面以及菜单界面，当中加入了鼠标交互模式，玩家可用鼠标点击三个选项。实现结果如图 3.1 所示。



图 3.1

(2) `void chessboard();`

函数功能：利用 easyX 绘制棋盘。用 easyX 函数库绘制水平直线、垂直直线、加粗格点等。实现结果如图 3.2 所示。



图 3.2

(3) `void updateWithInput();`

函数功能：实现鼠标交互。本函数中，程序会根据玩家鼠标左键的点击位置来判断玩家所决定的行棋位置，并在该位置绘制相应颜色的棋子。

(4) `int findLocation(int x, int mod);`

函数功能：找到最近的格点 ($\text{mod}==0$ 表示找横坐标 x ， $\text{mod}==1$ 表示找纵坐标 y)。本函数是配合 `updateWithInput` 函数使用的，辅助判断玩家决定的行棋位置。

(5) `int startUp();`

函数功能：数据的初始化。对窗口标题文字等进行修改，并对相应的数据进行初始化。

(6) `void rules();`

函数功能：介绍规则。本界面显示不围棋游戏的简要规则，并且显示特别鸣谢。实现结果如图 3.3 所示。



图 3.3

(7) `void show();`

函数功能：显示选择先后手的画面，请玩家选择。实现结果如图 3.4 所示。

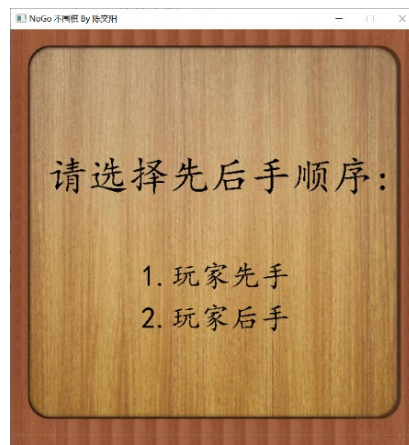


图 3.4

(8) `void chooseDifficulty();`

函数功能：选择游戏难度。玩家可以通过鼠标点击选择初阶、中阶、高阶三种难度。实现结果如图 3.5 所示。



图 3.5

(9) `bool inBoard(int x, int y);`

函数功能：判断点是否在棋盘内部，以便后面判断此位置是否合法。

(10) `bool haveAir(int fx, int fy);`

函数功能：判断是否有“气”。若返回 true，则表示有“气”，以便后面判断此位置是否合法。

(11) `void decideByComputer();`

函数功能：电脑决策并落子（低阶随机版）。电脑生成二维 1-9 的随机数，根据随机坐标来落子，这是低阶难度下的电脑决策策略。

(12) `int winOrLose();`

函数功能：判断是否胜利，即遍历整个棋盘，判断是否存在一点已经没有“气”。若存在，则必有一方已经胜利、另一方已经失败。如果这一点是黑子，则返回 1，如果这一点是白子，则返回-1。

(13) `void endGame(int mod);`

函数功能：游戏的结束界面。该函数根据 winOrLose 函数的返回值以及当前的回合数来判断是哪一方胜利、哪一方失败。失败又包含“吃子”和“自杀”两种情况，此函数也可以根据最近一步行棋位置是否有“气”来加以区分这两种情况。判断后，此函数会显示不同画面。胜利、因“吃子”失败、因“自杀”失败的情况分别如图 3.6、3.7、3.8 所示。



图 3.6



图 3.7



图 3.8

(14) `void readRecordFile(int(*board)[10]);`

函数功能：读取游戏数据文件存档，恢复棋盘状态、回合数、当前落子情况、游戏模式以及游戏难度。

(15) `void writeRecordFile(int n, int(*board)[10]);`

函数功能：存储游戏数据文件存档，保存回合数、棋盘状态、当前落子情况、游戏模式以及游戏难度。

(16) `bool checkQi(int x, int y);`

函数功能：检查落子后是否吃子，即检查上下左右不同颜色棋的气数，进行递归，若没有气了就被提子，这种情况下返回 false，其余情况返回 true。

(17) `int dfs(int x, int y, int colour);`

函数功能：返回点 (x, y) 有多少口“气”，其中进行递归，若 (x, y) 周围某点与该点颜色相同，则“气”应一并计算在内。

(18) `int getQi(int x, int y);`

函数功能：首先对辅助数组进行初始化，然后返回 dfs 的结果。

(19) `bool checkStep(int x, int y);`

函数功能：检查点 (x, y) 是否可以落子的基本条件。这里需要检查三个条件：一是 inBoard (x, y) 是否为 true，二是 (x, y) 这一点是否为空，三是是否是在第一回合且选择天元点。

(20) `bool goStep(int x, int y, int c);`

函数功能：模拟落子，参数 `c` 为颜色（1 或 -1）。此函数中，或在 (x, y) 处落下颜色为 `c` 的棋子之后，若 `getQi(x, y)` 为 `false`，则此函数返回 `false`。否则就返回 `true`。

(21) `bool testgoStep(int x, int y, int c);`

函数功能：检查点 (x, y) 是否可以落子的进阶条件。在满足基础条件的基础上，还需要 `goStep(x, y)` 和 `checkQi(x, y)` 均返回 `true`，这样的点就是可以落子的点。

(22) `int evaluatel(int colour)`

函数功能：对局面进行评估。若该局面下，我方可以落子的点越多，对方可以落子的点越少，则此局面越优。

(23) `void color(int x, int y); int calc(int c);`

函数功能：统计棋局中同种颜色但不相邻色块个数。这是为了在游戏中期执行“打散规则”，拉开局面。

(24) `int canItWin(int c, int dep)`

函数功能：在残局下进行搜索。选取合适的点进行落子。

以上是本不围棋程序所用到的关键的 24 个函数，并未包含所有函数。

四、Bot 思路简介

有关本程序 Bot 的编写过程，持续了近一个半月之久。限于笔者知识水平和能力有限，笔者尝试使用 UCT 算法但并未成功。因此笔者使用的 Bot 是以静态估值为主、残局时使用搜索算法来实现的。有关 Bot 的实现思路如下。

首先，当计算机面临决策问题时，首先计算机会利用 `testgoStep` 函数去寻找并记录所有可以落子的点。如果已经没有可以落子的点，此时计算机会直接返回 `false`，进而现实玩家胜利的画面。

之后，如果当前回合数在 20 步及以内，计算机会通过 `getQi` 函数优先选择所含“气”较少的点，这样做是防止对手在开局“造眼”而形成优势。如果当前回合数大于 20 步且小于等于 55 步，认为处于中局阶段，计算机会优先选择能使棋面较“散”（即：棋局中同种颜色但不相邻色块个数较多，函数 `calc` 返回值较大）的位置。上述选取很有可能会存在最优状况不止一个位置的结果。于是，计算机再对这些最优状况进行下一步估值，即评估落子之后己方可下位置与对方可下位置之差，此差值越大越好。通过两步估值，最终选择一个最优的点进行落子。如果当前回合数大于 55 步，即认为处在残局阶段，计算机通过 `canItWin` 函数进行搜索。先统计计算当前未试下局面的优势手数，再循环遍历所有可下点，如果这个点下后优势手数增加那么就计数。之后继续向下搜，以对手的视角来下下一步。如此递归下去，只有找到既满足优势步骤并且已经找到满足优势步骤的点的总数 `num < 7`（防止搜索宽度太大）并且在超时前向下的深度已经到终局（此时的终局我方必然在原来的基础上更优），才会停止，并在第一次选取的那一点落子。如果既超时，又没能在所有的拓展中达到终局，那么就会在所有可下点中随机选取一个落子（事实上，这种情况在实际博弈中极少出现）。

以上就是本程序 Bot 的主要思路。

本程序在 `botzone` 平台上的对战结果是共比赛 7 局、胜利 4 局，胜率约为 57.1%，已经是笔者尽全力的结果，笔者感到满意。

五、一点感想

计算概论大作业——不围棋 (NoGo) 终于完工了！在接到这项作业之前，我也从来没听说过还有“不围棋”这种棋类。游戏本身还是很有趣的，规则很新颖。历时近两个月，一千多行代码，好几次濒临崩溃，想过放弃，无数次 debug……最终还是基本实现了这个游戏。其实笔者应该是第一次用这么长时间去完成一项作业。虽然写出来的 AI 不是很强，但是游戏的基本功能都实现了，也还算令人满意。非常欢迎大家来跟我的程序来博弈一下～希望我的 AI 能不要被虐的太惨～现在想来，编写程序的过程还是很奇妙的～