五子棋程序开发日志

作者：刘继轩

2400017722

最后修改日期：2024 年 11 月 27 日

目录

1 2024 年 11 月 13 日

1.1 今日进展

今天了解了作业的具体要求，了解了使用 C++ 实现五子棋对弈程序所需要的算法基础，即 Min-Max 算法和 Alpha-Bata 剪枝优化。然后在 GitHub 上创建了仓库，方便后续的版本控制和更新。并使用 GPT 生成了 latex 模板方便后续开发日志的记录。同时今天找到了一些宝贵的学习参考资源，比如 GitHub 上基于 Javascript 语言的五子棋 AI 教程https://github.com/lihongxun945/gobang?tab=readme-ov-file 和 bilibili 上的算法教程视频https://www.bilibili.com/video/BV1v94y1r7F8/?spm_id_from=333.880.my_history.page.click&vd_source=2f0075ad419feeef529bb2dce0adc975。

1.2 本次作业的基本要求

1.2.1 五子棋详细规则

黑白双方轮流落子，黑方为先手。

在横、竖、斜方向上连成五子（连续五个棋子皆为己方）者为胜。

黑棋在行棋过程中，如果违反以下“禁手规则”会被判负。

三三禁手：黑棋在一个位置下子后，形成两个或两个以上的活三。活三是指在棋盘上有三个连续的黑子，并且两端都有空位可以继续下子形成五连珠。

四四禁手：黑棋在一个位置下子后，形成两个或两个以上的活四。活四是指在棋盘上有四个连续的黑子，并且至少有一个空位可以继续下子形成五连珠。

长连禁手：黑棋在一个位置下子后，形成六个或更多连续的黑子。

四三禁手：黑棋在一个位置下子后，同时形成一个活四和一个活三。这种情况也被视为禁手。

注意到这里的禁手规则，后续需要特定的函数实现。

1.2.2 其他要求

棋盘大小可以自定义，如果要参加 Botzone https://botzone.org.cn/ 比赛，则棋盘大小为 15*15。

1.3 算法基础介绍

Min-Max 算法：

五子棋看起来有各种各样的走法，而实际上把每一步的走法展开，就是一颗巨大的博弈树。在这个树中，从根节点为 0 开始，奇数层表示电脑可能的走法，偶数层表示玩家可能的走法。

那么我们如何才能知道哪一个分支的走法是最优的，我们就需要一个评估函数能对当前整个局势作出评估，返回一个分数。我们规定对电脑越有利，分数越大，对玩家越有利，分数越小，分数的起点是 0。

我们遍历这颗博弈树的时候就很明显知道该如何选择分支了：

电脑走棋的层我们称为 MAX 层，这一层电脑要保证自己利益最大化，那么就需要选分最高的节点。

玩家走棋的层我们称为 MIN 层，这一层玩家要保证自己的利益最大化，那么就会选分最低的节点。

而每一个节点的分数，都是由子节点决定的，因此我们对博弈树只能进行深度优先搜索而无法进行广度优先搜索。深度优先搜索用递归非常容易实现，然后主要工作其实是完成一个评估函数，这个函数需要对当前局势给出一个比较准确的评分。

alpha-beta 剪枝：即每次更新节点的数值时，查看其是否被父节点所兼容：如果父节点已经得到了合理的结果，就可以通过 break 语句进行剪枝。

1.4 明日计划

编写一些基本的函数，实现输入与输出的读取。

1.5 后续待实现的内容

胜负判断函数；禁手规则判断函数；局势评估函数；

2 2024 年 11 月 27 日

2.1 今日进展

1. 完成了棋盘初始化、显示、落子验证的实现。

2. 实现了胜负判断逻辑，通过检查横向、纵向、两种斜向的五子连珠状态，判断玩家是否获胜。

3. 实现了黑白棋的交替落子逻辑，并在有效落子后实时更新棋盘。

4. 优化了终端显示，解决了中文输出乱码问题。

2.2 成果展示

2.2.1 程序界面截图

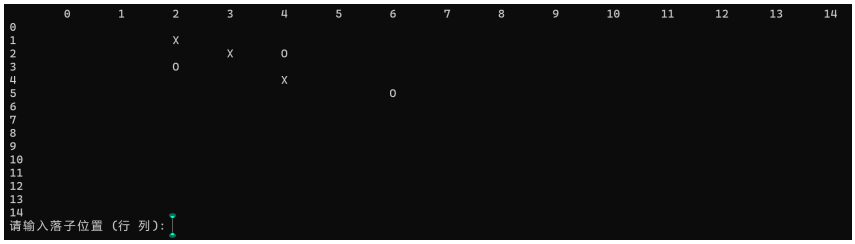

图 1: 程序运行截图

2.2.2 运行结果

程序成功运行，能够初始化指定大小的棋盘，支持黑白棋交替落子，并正确判断胜负。

2.3 问题与解决方案

2.3.1 遇到的问题

1. 在终端中显示中文提示时出现乱码。

2. 无法确定落子是否形成五子连珠，导致游戏胜负判断逻辑缺失。

2.3.2 解决方法

1. 使用 ‘SetConsoleOutputCP(65001)‘ 将控制台编码设置为 UTF-8，解决了中文乱码问题。

2. 编写了 ‘checkwin‘ 函数，通过遍历四个方向检查当前玩家是否形成五子连珠。

2.4 代码分析

2.4.1 棋盘初始化与显示

```
vector<vector<char>> initializeBoard(int size)
{
    return vector<vector<char>>(size, vector<char>(size, '␣'));
}

void displayBoard(const vector<vector<char>>& board)
{
    cout << "\t";
    for (int i = 0; i < board.size(); ++i) {
        cout << i << "\t";
    }
    cout << endl;

    for (int i = 0; i < board.size(); ++i) {
        cout << i << "\t";
        for (int j = 0; j < board[i].size(); ++j) {
            cout << board[i][j] << "\t";
        }
        cout << endl;
    }
}
```

Listing 1: 棋盘初始化与显示代码

2.4.2 胜负判断核心代码

```
bool checkwin(const vector<vector<char>>& board, int x, int y, int
   currentPlayer) {
    int directions[4][2] = {{1, 0}, {0, 1}, {1, 1}, {1, -1}};
    for (auto direction : directions) {
        int count = 1;

        for (int i = 1; i < 5; i++) {
            int nx = x + i * direction[0];
            int ny = y + i * direction[1];
            if (nx >= 0 && nx < board.size() && ny >= 0 && ny < board
               .size() &&
```

```
                board[nx][ny] == currentPlayerchar[currentPlayer]) {
                count++;
            } else {
                break;
            }
        }
        for (int i = 1; i < 5; i++) {
            int nx = x - i * direction[0];
            int ny = y - i * direction[1];
            if (nx >= 0 && nx < board.size() && ny >= 0 && ny < board
                .size() &&
                board[nx][ny] == currentPlayerchar[currentPlayer]) {
                count++;
            } else {
                break;
            }
        }
        if (count == 5) return true;
    }
    return false;
}
```

Listing 2: 五子连珠胜负判断代码

2.5 明日计划

1. 增加平局检测功能。
2. 实现黑棋禁手规则（如三三禁手、四四禁手）。
3. 为游戏添加菜单功能，支持重新开始或退出。

参考文献

[1] 作者, *书名*, 出版社, 出版年份.

[2] 在线资源标题, https://example.com