电子科技大学

计算机专业类课程

实验报告

课程名称：程序设计基础

学院专业：计算机科学与工程学院

学生姓名：朱俊鑫

学　　号：2019080601011

指导教师：俸志刚

日　　期：2020 年 06 月 16 日

电 子 科 技 大 学

实 验 报 告

实验一

一、实验室名称：

家中卧室

二、实验项目名称：

五子棋人机对战程序的设计与实现

三、实验目的：

使用 C 语言实现五子棋的游戏环境，设计自动与人下棋对战的模块，且能够有较大概率胜过大部分未研究过五子棋的普通人。

四、实验主要内容：

使用C语言作为开发语言，基于Windows控制台API实现五子棋游戏的用户界面与人机交互。设计并实现游戏逻辑，以及基于博弈树、极大极小值搜索与Alpha-Beta剪枝的自动下棋模块。

五、实验器材（设备、元器件）：

硬件平台：

处理器 ： Intel(R) Core(TM) i5-3230M CPU @ 2.60GHz

内存大小： 4.00GB DDR3L

软件平台：

操作系统： Windows 10 Professional 1903

开发环境： MinGW-W64-builds-4.3.5

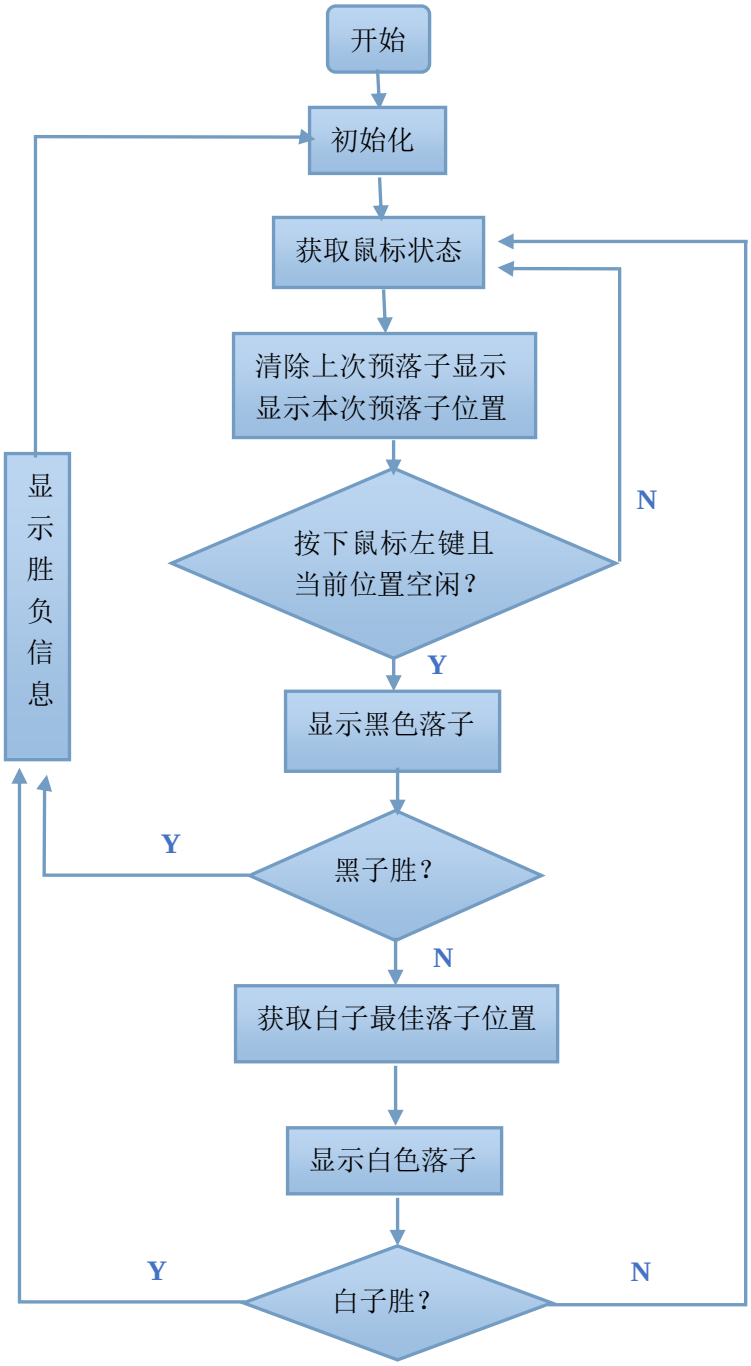
编辑器 ： Visual Studio Code 1.44.2

六、实验步骤：

1. 问题描述

人机双方轮流落子。轮到机器一方时，程序需要根据当前棋局局面，选择最利于本方的位置落子。每次落子后，判断游戏继续还是某方获胜游戏结束。

2. 算法分析与概要设计



程序的外部框架主要是两个循环的嵌套。外层循环是游戏回合的循环，程序不主动退出，以用户自己关闭窗口作为结束；内层循环是鼠标状态获取的循环，用以监听鼠标的实时状态，包括鼠标位置与鼠标事件。流程图中“显示胜负信息”中包括了对鼠标右键按下事件的等待，流程图中“获取白子最佳落子位置”的具体实现在下方展示。

3. 核心算法的详细设计与实现

```
Minimax(board, depth, score, alpha, beta):
    if(depth>MAX_DEPTH): //当大于最大搜索深度停止搜索
        return -1, -1, score;
    if(depth==1): //初次进入时生成待选点
        (points, n_points)=Generate(board);
    for(i=1→n_points):
        (x, y)=points[i];
        int delta=Score(board);
        PutChess(board, x, y, depth%2); //depth 奇数白子偶数黑子
        delta=Score(board)-delta;
        (_, _, gamma)= //向下搜索
            Minimax(board, depth+1, score+delta, alpha, beta);
        if(depth%2==1):
            if(gamma>alpha):
                (alpha, X, Y)=(gamma, x, y);
        else:
            if(gamma<beta):
                (beta, X, Y)=(gamma, x, y);
        RemoveChess(board, x, y, depth%2); //还原现场
        if(alpha>=beta):
            break; //当 alpha 大于等于 beta 时进行剪枝
    return X, Y, depth%2==1?alpha:beta; //返回选中坐标和更新值
```

以上即为极大极小值搜索与 Alpha-Beta 剪枝的伪代码实现，其中 Generate()函数生成棋盘中所有可选点，并对各点进行打分，而后进行排序，有利于电脑一方获胜的排在前面。这样生成待选点后，每个节点向下搜索时，都能尽早找到合适的点，Alpha-Beta 剪枝也能今早生效。

估值函数 Score()评估了当前点所在的行、列、右斜、左斜四条轴线上棋形

的总分数。具体评估方法是利用人工给出的已有基本棋形及其分数，在整个程序一开始生成一整行所有可能的棋形分数，并以哈希的方式储存（一行 15 个黑白棋子用三进制表示）。

对于双活三、双冲四、活三+冲四这三种必胜/必败情况进行特判。由于基本棋形的分数是有梯度的，所以落子后根据总分数的变化量可以得知某些棋形个数的变化。累加变化即可得知当前棋盘上某些棋形的个数，因此可以对这些棋形的组合增加额外的分数。并且这些必胜棋形组合越早出现，应得分值应该越多。

七、实验数据及结果分析：

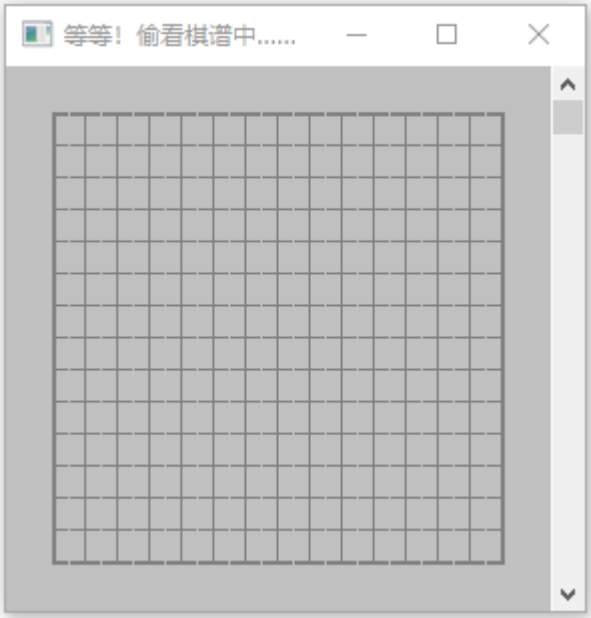


图 1 – 预处理等待界面

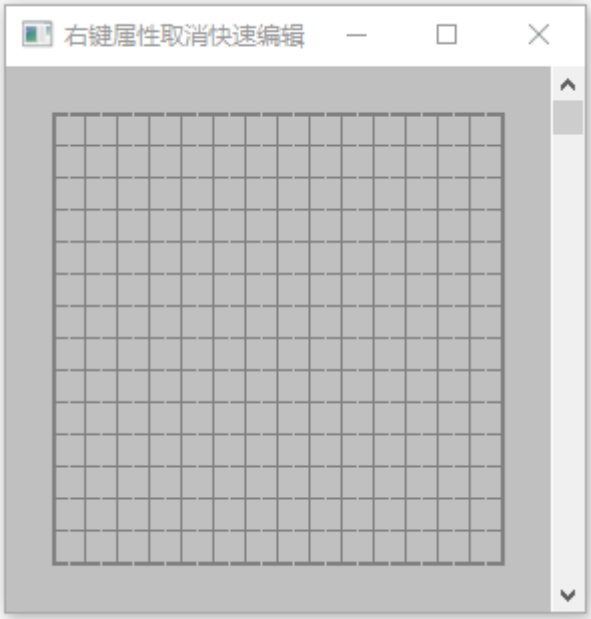


图 2 – 首次开始的提示

该程序使用了控制台字符界面作为伪图形界面，为了追求极致的简洁，把文字提示放置到标题栏中。每次打开程序时，都要等待一段时间以待程序对估值进行预处理，也可理解为“解压”过程（图 1）。**而后，若是首次打开该程序，且使用 Windows 10 系统，还需要右键单击标题栏，在右键菜单中选择【属性】，在属性面板的【选项】卡片中取消【快速编辑模式】的勾选，并确定，这样才能进行鼠标操作（图 2）。**

该程序实现了鼠标跟踪/预落子显示，如下图 3 所示（由于截图软件的特性，截屏时自动隐藏了鼠标在屏幕上的显示，所以图中灰色圆圈上本有鼠标停留）。

对于自动下棋的表现，程序能够敏锐捕捉到人的疏漏。个人能力有限，经测试，对于未研究过五子棋、未全力专注下五子棋的普通人，机器一方能有较大的胜率。下图 4 展示了机器获胜了的一次棋局。

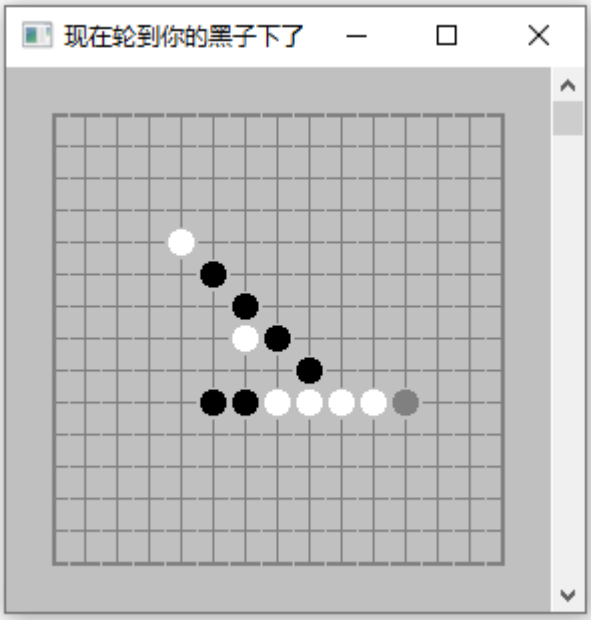


图 3 – 预落子功能展示

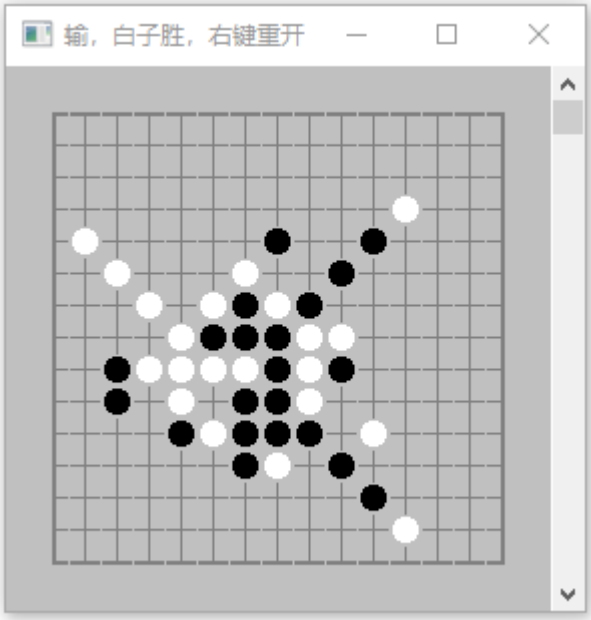


图 4 – 敏锐捕捉到人的疏漏

图 5 – AI 自我对战僵持许久

图 6 – AI 轻松败给经验人士

对于旗鼓相当的对手，即 AI 自身，双方能够僵持很久（图 5），可见双方棋力对等。同时，这样的结果也能间接地表明 AI 的棋力相对稳定。

但面对下五子棋有经验、套路的人来说，AI 在很少的步骤内就输了（图 6），因为各种经验实质上是对更远棋局走势的预测，而本 AI 只能搜索到敌我共 4 步。就算加上 2 步必胜和 4 步必胜的特判（在 brain.c 中的 bonus 函数），本 AI 最远只能看到某些情况下的 6 步或 8 步内某一方的获胜，但这只是某些情况（即第 4 步出现某一方达成 2 步必胜或 4 步必胜的棋形）。

面对同样类型的程序，此处找了一个网页版五子棋（http://html5.huceo.com/wzq/）与本程序进行对战，如下图 7 所示。网页版为先手黑子，本程序为后手白子。

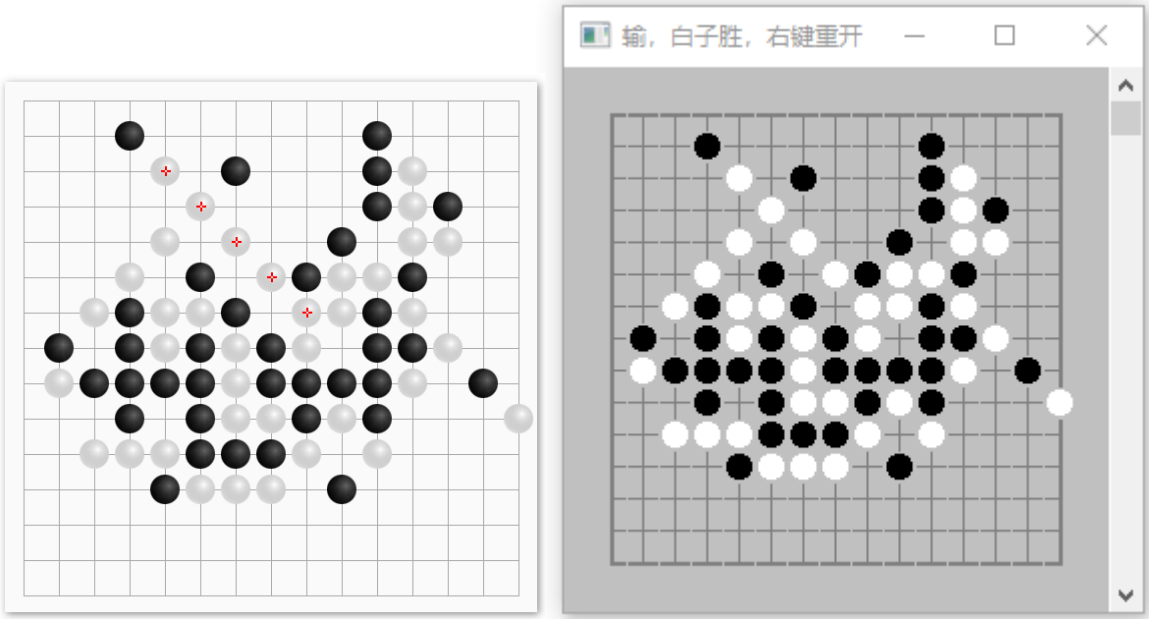


图 7 – 与同类 AI 对战获胜

在与网页版同类 AI 的一次对战中，本程序白子后手取胜。由于两 AI 下棋模式固定且网页版先手的第一个落子位置固定，所以两者对战棋局不变，只能进行一次测试，这一次结果不能确定两者的水平高下。但能确认本 AI 无大问题。

八、总结及心得体会：

该程序在运行之初预处理整行棋形的所有可能的分数，并使用哈希方法储存，大大降低了估值函数的时间开销，以及缩短了棋形判断的代码复杂度。但这种方式无可避免需要额外处理双活三等特殊的棋形组合。

另外，尽管对估值进行了优化，但程序运行时间主要消耗在了庞大的搜索空间上。虽然使用了 Alpha-Beta 剪枝，但剪枝效果受待选点的次序影响很大，该程序只使用了最简单的方式决定顺序，没有很好发挥剪枝的效果，所以只能局限于 4 层的搜索。

九、对本实验过程及方法、手段的改进建议及展望：

对于着法生成模块（Generate）可以引入启发式算法等方式来减小搜索空间，或者甚至可以采用部分不安全的剪枝以提高搜索效率，进而提高搜索层数，当然要兼顾准确率。激进一点还可从根本上替换 AI 的实现算法，采用深度强化学习的方式，通过棋谱、模拟训练获取数据得到模型。

报告评分：

指导教师签字：