北京大学

亚马逊棋实验报告

计算概论 A 大作业

北京大学 信息科学技术学院
2019 级 本科生
姓名：王颖
学号：1900013016
指导老师：李戈
助教：王泽钧

一、撰写日志呈现

2019/11/25 画棋盘、完成一个随机版本、一个一层搜索、一个二层搜索的 bot
2019/11/28 实现一个新的稍精确一些的估价函数
发现：实际上估价函数的重要性极大
2019/12/02 实现人机交互界面
2019/12/03 实现读档、存档功能
不幸的是读档后发生故障 需要调整
游戏结束画面缺失 可能只是缺 pause
2019/12/08 修正读档后 AI 替人走一步的故障
加入 pause
完善读档时档案不存在的情况,避免出故障
加入悔步操作(仅供输入失误的情况使用)
实现 让 AI 玩选项(这个在视觉化版本中没有)
2020/01/03 更新评估函数 增加 KingMove 评估
这里需要修改很多接口(评估函数 double 化)
加入领域特征值评估、灵活度评估
2020/01/05 利用初步估价更改枚举方案的顺序，使 alpha-beta 剪枝的优化效果更好
加入卡时(利用 clock()), 避免决策超时
2020/01/06 解决"自杀"导致输棋的问题
优先考虑没有确定领地的部分
2020/01/07 调整先手优势系数 Ec(0.2->0.08)
希望缓解后手胜利先手反而失败的情况
了解 qt，制作棋盘棋子素材
2020/01/08 开始用 qt 写界面
画棋盘
完成人机交互界面
2020/01/09 增加"对方落子"键
完成存档读档功能
进行鲁棒性测试和修正
增加"重新落子"键
增加游戏难度选项 但不幸出现先后手键不可按的情况
2020/01/10 修正先后手键不可按的 bug
增加 AI 辅助落子功能
加入时间戳, 防止瞬间多次点击造成棋局快速推进, 改善游戏体验
根据 AI 执子颜色确定的先手优势系数(仅在网站版本中用到)

二、程序设计思路描述

该作业主要分为博弈和界面设计两个部分，这里将分别描述两个部分的设计思路。

（一）博弈部分

博弈部分采用搜索的方法求一个较优决策，包括搜索(枚举可行决策)和评估(评估决策的优劣)两个子部分。

搜索部分：

关键词：MinMax 搜索、α－β剪枝、决策排序、卡时机制

最初实现的是一个单层搜索，后改为二层搜索。二层搜索不仅枚举己方的决策，也枚举对手的决策。注意到己方决策时总是希望自己的优势尽量大，而对手决策时总是希望己方的优势尽量小，两层的决策者对决策的选择标准是不同的（己方层总是取最优决策， 对手层总是取最劣决策（对己方而言）），采用 MinMax 搜索模拟这个过程。同时，二层搜索中，穷举的方案增多，尤其是在开局阶段，枚举己方和对手的所有决策所花费的时间代价是不可接受的，需要考虑剪枝。注意到在对手足够聪明的前提下，对手一定会选择对己方最劣的决策，因此，对一个己方的决策，如果存在一个对手决策使得对手决策后的评估值比历史上其他己方、对手决策后的最优值小，那么这个己方决策一定不是当前评估函数下的一个最优决策。由此我们可以进行剪枝（即α－β剪枝）。在这里， α－β剪枝剪枝的效果取决于 Max 层决策枚举的顺序，当较优解靠前出现时，剪枝效果很好，相反，当决策从劣到优依次出现时，剪枝完全起不到作用。基于这种原理，我们提前将所有的 Max 层可行决策按照单层的评估值排序（我们认为当前层的评估和下一层的评估有一定的正相关性），从优到劣地枚举决策进行搜索。值得注意的是，即使在这种情况下，在开局时还有可能有决策超时的情形，于是我加入了卡时机制，防止决策超时。

评估部分：

关键词：territory 特征值、position 特征值、mobility 灵活度特征值、KingMove、QueenMove、stability

参考了论文《亚马逊棋机器博弈系统中评估函数的研究》，利用 territory 特征值，position 特征值，mobility 灵活度特征值，根据 KingMove，QueenMove 两种走法，评估棋局。其中，territory 特征值的依据是到某一特定格子的步数大小反映双方棋子对该格子的控制能力，先手在控制该格子上有一定优势。Position 特征值依靠双方到某一特定格子步数的差值反映双方对格子控制能力的强弱。Mobility 灵活度可以防止棋子过早地被堵死。采用 KingMove 和 QueenMove 两种走法，可以对棋子到空格的步数、距离都有一定把握。除此之外，认为完全属于自己的领地（对方棋子不可进入）可以稍后填充，双方都能控制的领地应该先抢夺。于是，我定义了 stability 反映一个棋子的领地是否稳定（领地完全属于自己），决策时，认为走不稳定棋子是一个更好的决策（优先和对手抢夺领地）。

（二）界面设计部分

界面设计部分使用了 Qt（一个 1991 年由 Qt Company 开发的跨平台 C++图形用户界面应用程序开发框架）。利用 MainWindow 里的 paintEvent()函数，在空棋盘图片上的对应位置绘制透明画布的棋子、障碍等的图片，达到绘制棋盘的效果。决策输入方面，利用 MainWindow 的 mousePressEvent(QMouseEvent *event)函数，利用 event 的坐标进行换算，求出鼠标点击位置所对应的的棋盘上格子的坐标，实现鼠标输入操作。一个决策过程仿照 botzone（https://www.botzone.org.cn/）上 Amazon 游戏桌的设定，当前决策方的棋子是可点击的，下一步的可行位置将会用标记标识出来，再下一步的位置也会被标识出来。依次点击的三个坐标如果构成一个合法的决策，则输入结束。

在人机交互过程中，我使用文件记录当前局面的信息，在人或机决策前重新载入棋盘数据，保证相应数组内数据的可靠性。利用文件也可以实现存档、读档、输入错误时的单步悔棋操作。

对新游戏、退出、存档、读档、单步悔棋操作的申请，用 Qt 中的 QPushButton 类型设

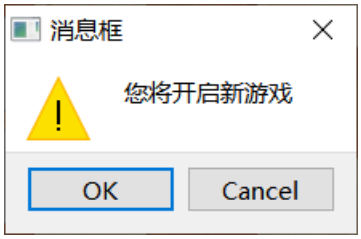
置一个按键实现。游戏过程中字幕提示出现由 QLabel 类型实现；用 QMessageBox 实现简单的消息框。利用 MainWindow 类型中的 close()函数实现关闭游戏的功能。

三、功能说明

该作业实现了图形化界面，主要功能有新游戏、退出、读档、存档，以下作简要说明。

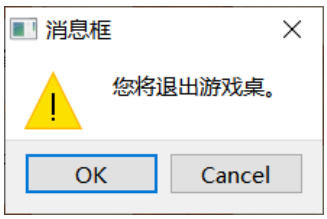
（一）新游戏功能

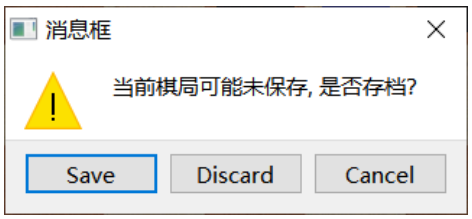
点击“新游戏”键，将有消息窗弹出向玩家确认或取消开启新游戏的操作。玩家确认以后，将开启一个新游戏，由玩家选择游戏难度以及先后手。下图为开启新游戏的消息框、玩家选择先后手和游戏难度界面。



（二）退出功能

游戏中途或游戏结束时，点击“退出”键，将有消息窗弹出向玩家确认或取消退出游戏的操作。确认以后，玩家将退出游戏；取消时，玩家将回到游戏桌。特别的，在游戏途中，若玩家或 AI 在上一次读档或上一次开启新游戏后有决策，消息框会提醒玩家当前棋局可能未保存，此时有保存、放弃、取消三个选项。下图为两个消息框。

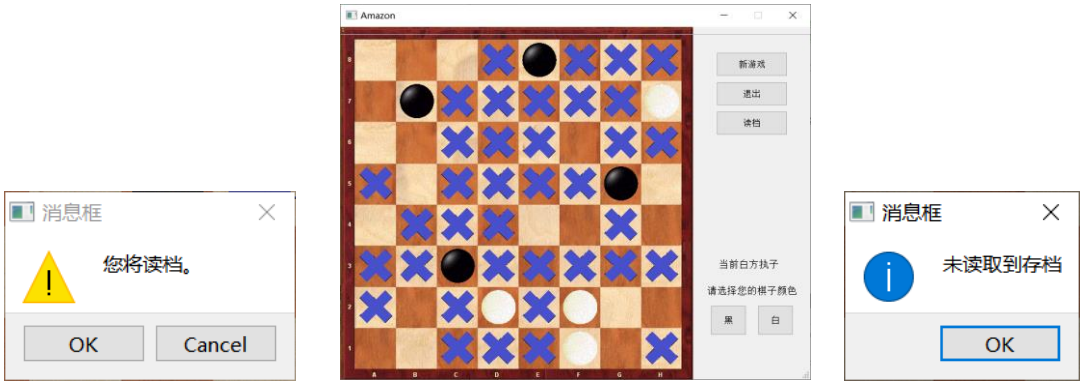




（三）读档功能

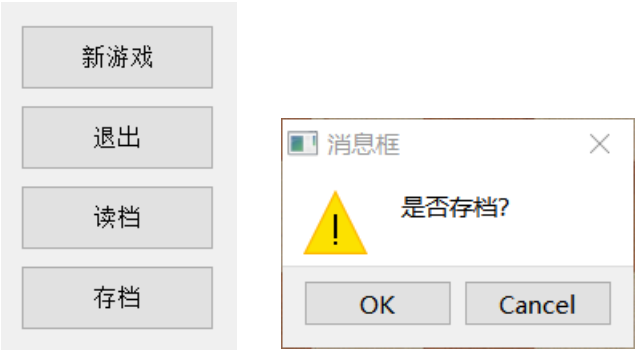
玩家点击“读档”键时，将弹出消息框提示玩家将要读档，玩家可以点击“OK”键确认读档，点击“Cancel”键取消读档。确认读档后，界面会弹出上一次存档的棋局，在右下角显示出当前决策方的颜色，让玩家选择棋子颜色（利用读档的这一点玩家可以在将输时和 AI 换颜色，实现作弊）。特别的，当游戏之前没有存档时，将会弹出消息框显示“未读取到存

档”，读档事件将不会发生。下图为读档示例图。



（四）存档功能

游戏中途（胜负未定时），若玩家或 AI 在上一次读档或上一次开启新游戏后有决策，游戏界面右方将出现“存档”键。点击“存档”键，将弹出消息框向玩家询问是否存档，点击“OK”，则存档成功，点击“Cancel”则取消存档。以下为示意图。



四、特色介绍

（一）视觉化

精心设计、制作图像界面，鼠标点触无需键盘操作，改善游戏体验。

（二）难度设计

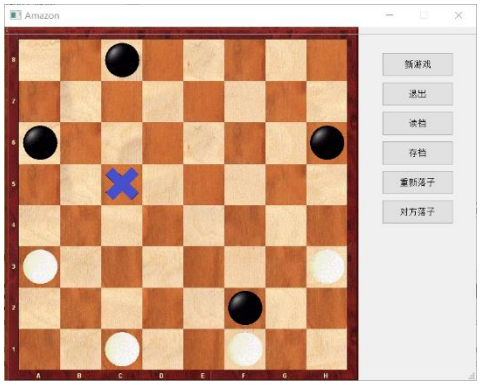
游戏分为简单、中等、困难三个难度等级，使游戏更具挑战性和趣味性。

（三）设置“对方落子”键

己方决策后界面更新，显示己方决策结果。点击“对方落子”键后，对方可落子，明确己方、对方落子过程，便于了解棋局，控制游戏节奏。

（四）设置“重新落子”键

决策失误怎么办？点击“重新落子”键实现单步回退功能，即可重新决策。避免决策失误错过胜机带来的差体验（玩家决策后如果胜利，将不设置“重新落子”键）。



（五）AI 辅助落子

增加“AI 辅助落子”键，采用困难版本的 AI 辅助落子，在没有好的想法时由 AI 帮助，游戏过程更轻松愉快。

（六）防误触机制

新游戏、存档、读档、退出时弹出消息框让玩家确认，防止玩家误触，保证良好的游戏体验。同时注意到游戏过程中，“AI 辅助落子”键与“对方落子”键在同一位置交替出现，快速连续点击时可能出现游戏局面迅速更新或瞬时更新的情况，为此，我设立了时间戳，有意识地识别连击现象，并对连击现象中非第一次鼠标操作不予处理。确保游戏画面不突变，有序、合理。

五、总结

该大作业是用 Qt 实现的，代码部分包括两个头文件（amazon.h、mainwindow.h）和三个源文件（amazon.cpp、main.cpp、mainwindow.cpp）。其中 amazon.cpp 主要负责博弈部分，mainwindow.cpp 主要负责界面设计部分，main.cpp 实现一些界面设置以及 MainWindow 窗口的调用。本次大作业利用图片素材、paintEvent() 绘制棋盘，利用 mousePressEvent(QMouseEvent *event)实现鼠标输入、通过各种键的设置申请各种功能，完成视觉化。博弈部分采用 MinMax 搜索、α－β 剪枝、决策排序、卡时机制、各种估价的分段、综合，达到一个还算好的效果。在界面设计和游戏需求考量上，我也费了一些心思，棋盘、棋子的素材由网络素材改造而来，部分素材由我本人绘制，应用 Qt 的 QPushButton、QLabel、QMessageBox 等类型，尽量给玩家更好的视觉效果和游戏体验。综上述，我认为该作业可以视为一份较优质的作业，有一定的特色。

备注：源代码无法直接编译运行，需要安装 QT。