计算概论 A 期末大作业
基于 C++ 的不围棋游戏程序的实现

immediate

1/4/2023

1 简介

作为 2012 年出现在大学生博弈大赛[1]中的一种新棋种，不围棋迅速在博弈比赛中流行起来。一般情况下，对围棋的基本理解是消灭敌人取得胜利，而不围棋则与其相反。不围棋的规则不允许有棋子死亡，无论是哪一方自杀、或是吃掉了对方的棋子都会判负。具体的说，不围棋使用 9x9 棋盘，黑棋先行，之后黑白交替落子，对弈中禁止自杀，如果一方吃掉另一方的子或者是选择 Pass 则被判负。吃子规则与围棋相同，就是指某种颜色的一个子或者一串棋子在棋盘上，与它直线紧邻的交叉点为它的气”，若所有的气都被另一种颜色占据，则被吃掉。这种规则要求玩家在和平中取胜，最后依然是比较双方占地盘的多少。从某种角度来说，不围棋更符合中华传统文化中“和为贵”的思想。

本实验使用 c++ 平台实现了一个不围棋的游戏程序，并在游戏中集成了基于 MCTS 的一个不围棋博弈程序，实现了人机对弈。

2 功能设计

作为游戏程序，实现的基本功能包括：1. 显示棋盘；2. 允许用户通过交互进行落子；3. 在落子后，程序将作为电脑方落子，并现实落子后的棋盘；4. 当胜负已分时显示胜负。此外需要实现的功能包括：5. 各类必要的

[1]中国人工智能学会机器博弈专业委员会. 国内计算机博弈网址 [EB/ OL]. [2014]. http:/ / computergames.caai.cn/ .

操作提示；6. 暂停、继续功能；7. SL（保存和加载存档）功能；8. 悔棋功能；9. 一些偏好设置，比如设置人或电脑先手。

出于可行性考虑，程序的交互采用命令行用户界面 CLI 设计。界面应包含：1. 一种包含多个平行选项的界面（下称：菜单界面），用于显示标题、暂停继续、SL 等功能；2. 一种包含一个棋盘的界面（下称：棋盘界面），用于显示游戏的对弈情况。

具体的说，每个界面应该包含：1. 一个标题（Title）；2. 一个副标题（SubTitle）；3. 一个主体（Body），对于菜单界面是多个平行选项，对于棋盘界面则是棋盘，用户可以与主体内的元素交互；4. 一个提示（Hint），用户进行必要的操作提示。此外，界面还包含一个“单次提示框”，灵感来源于 Android 系统的“toast”消息框，用于对用户的不正确操作进行即时的提示。单次提示框在用户进行下一个操作时将消失。为了反馈用户的键盘操作，程序采用高亮（反色）的方式提示目前正在选择的项目。

以“开始菜单（WelcomeMenu）”为例，程序界面如图：

```
不围棋 // 标题
==================
计算概论A大作业 // 副标题
------------------
新游戏 // 主体
继续游戏
打开存档
偏好设置
查看说明
退出
------------------
按上下方向键控制光标，按回车选择 // 提示
```

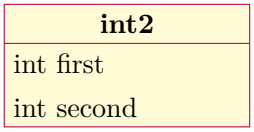
棋盘界面和菜单界面基本相似，只是主体部分更换为棋盘。

3 代码架构

程序主体采用结构化编程思想，程序的各个构件采用面向对象编程思想。

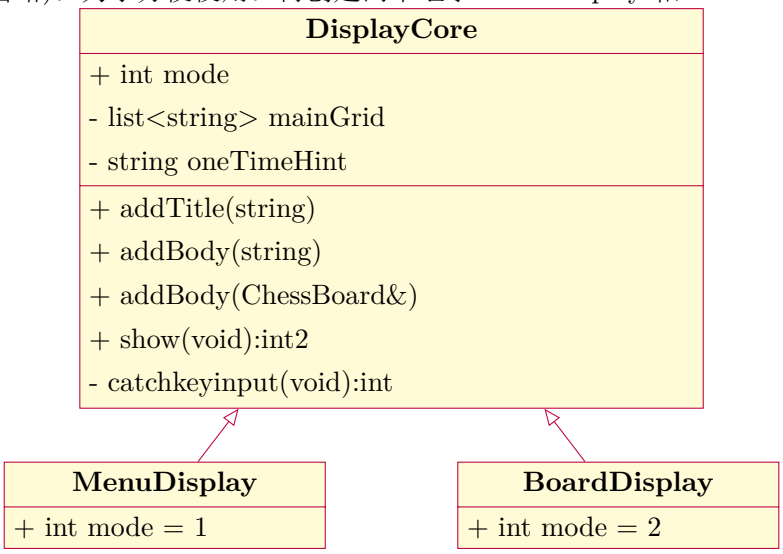
3.1 各个构件：

1. 数据结构：由于程序需要多次用到两个整数一组的“坐标”，因此创建一个坐标结构体，名为 int2，并用 define 的方式将它起别名 coordinate。



其中用 first 和 second 表示第一个和第二个坐标分量。

2. 界面显示构件：创建一个名为 DisplayCore 的类，其属性和方法如下(有省略)，为了方便使用，再创建两个继承 MenuDisplay 和 BoardDisplay：



其中 mode 表示显示模式，1 代表显示菜单界面，2 代表显示棋盘界面。DisplayCore 类内置了对键盘操作的捕捉。在运行调用 show() 方法后，DisplayCore 将持续运行直到用户进行选择。

调用 DisplayCore 显示一个界面（以开始界面为例）的代码如下：

```
int2 catchreturn = MenuDisplay().addTitle("不围棋")
    .setSubtitle("计算概论A大作业")
    .addBodyLine("新游戏")
    .addBodyLine("继续游戏")
    .addBodyLine("打开存档")
    .addBodyLine("偏好设置")
    .addBodyLine("查看说明")
    .addBodyLine("退出")
    .addHint("按上下方向键控制光标，按回车选择")
    .show();
//对catchreturn.first进行讨论，实现不同功能
```

3. 棋盘构件。创建一个名为 ChessBoard 的类，将下棋的各类方法集成在类中。

ChessBoard
+ int step - stack<int2> allStepStack - int grid[25][25]
+ judgeWin(void):int + checkChessAvailable(int2 cord,int color):bool + putChess(int2 cord):int + save(void) - isQiHave(int2 cord):bool - generateListString(void):list<string>

其中，allStepStack 记录了所有落子历史；generateListString 用于生成适合 DisplayCore 显示的棋盘图案。

4. 下棋 AI 构件，将下棋功能集中在类中，落子只需采用”ai.put()” 指令。

ChessAI
+ ChessBoard& board
+ put(void):int2

3.2 程序主体：

程序主体采用结构化设计思想，分为多个函数，主要用于显示界面，调用各个类的方法以实现程序各类功能，以及通过函数之间的互相调用实现界面的跳转。主要的函数包括：

```
int welcomeMenu();//显示标题画面
int pauseMenu();//显示暂停菜单
int opensaveMenu();//显示保存的存档列表
int openCheckMenu(string);//显示单个保存的存档，选择进行打开、删除等操作
int settingsMenu();//进行设置
int play(string);//进行游戏
int main();
```

程序流程图如图 1示：

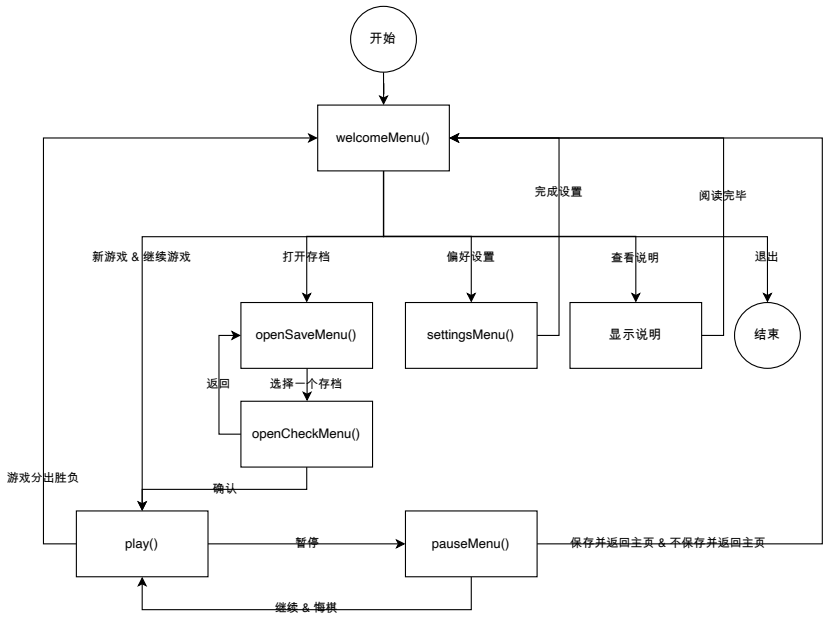


图 1: 程序流程图

4 AI 算法

程序采用一种简化的 MCTS 算法决定落子位置。MCTS（蒙特卡洛树搜索）是一种动态评估方法，更多的是利用数学统计中概率的思想。具体来说，就是对问题领域内的所有可选情况，通过不断反复地进行大量抽样，最终所得结果会在解空间上形成一个分布，而这个分布是接近真实的，进而就能够得到所需的最优解或近似的最优解。MCTS 方法主要包括以下 4 个方面的内容：

1. 搜索：从博弈树的根节点 (即终局状态) 向下搜索直至当前的叶子结点 (即当前局面)；
2. 扩展：对当前的博弈树叶子结点进行扩展；
3. 模拟：从当前的博弈树叶子结点开始进行蒙特卡洛概率模拟并给出一个蒙特卡洛概率统计数值；
4. 更新：把蒙特卡洛模拟的结果更新到搜索过程中博弈树的每一个节点上。

之后，重新从 1 开始，不断反复地进行迭代，使得添加的局面越来越多，则对于博弈树中所有的子节点的胜利率也越来越准. 最后，选择胜利率最高的选点。

在本程序中，对 MCTS 算法进行了简化。在“模拟”步，不通过蒙特卡洛模拟给出概率统计的 value 值，而通过对当前局面进行评估，直接将评估值作为当前节点的 value 值，利用此值进行反向传播（更新）。

具体的评估方式的伪代码为：

```
int 模拟(){
    r = 0;
    while(循环每一个坐标点){
        if(我方可在此处落子 且 对方不可在此处落子){
            r+=10;
        }
        if(对方可在此处落子 且 我方不可在此处落子){
            r-=10;
        }
    }
    return r;
}
```

由于这种“模拟”得到的 value 值有正有负，在实际落子中，发现最后选取“value 的绝对值最大”的点比选取“value 最大”的点效果更好。故在最终确定落子方案时，选择 value 绝对值最大的落点。

在本程序中，每次执行完 ChessAI 的 put 方法后，程序都将删除原来的博弈树，每次落子都是从头计算。经统计，一秒内大概能模拟 50000 以上个节点，可以给出足够好的落点。

5 实验结果

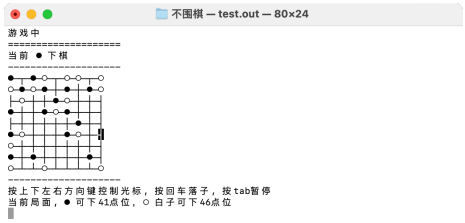
5.1 程序运行结果

图 2是一些程序关键部分的截图。

5.2 botzone 结果

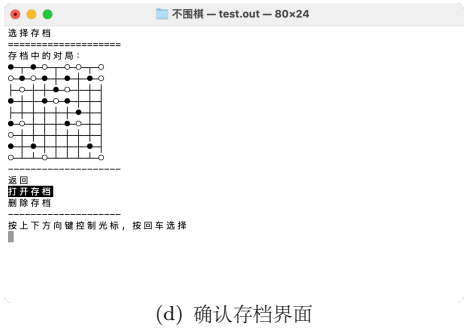
图 3是 botzone 上某次对局的结果，其中白方为本次程序涉及的 bot。

(a) 主菜单



(b) 游戏界面

(c) 选择存档界面



(d) 确认存档界面

图 2: 程序运行结果

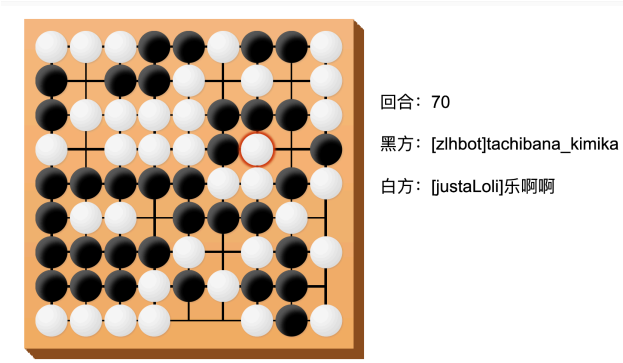


图 3: botzone 上的一次对局结果

6 结论

本次实验，使用 C++ 环境实现了一个包括落子 AI 的不围棋游戏程序。程序设计时才用了代码复用度优先、代码通用性优先的设计策略，一方面将重复的模块集成为类进行调用，有利于提升程序主体的简洁性；另一方面定义了大量常量，避免未经解释的字面量的出现，有利于提升程序代码的易懂程度，同时提升了程序的通用性。程序使用 CLI 用户界面，利用键盘操作，提供了相对美观、直觉的操作提示，并能对用户操作进行即时的反馈，对错误操作进行即时提醒。程序使用了一种简化的 MCTS 算法作为电脑的落子策略，实现了较好的效果。