经典五子棋游戏对战程序框架说明文档

陈柘宇，濮成风，陈树伦，孙若凡，吴之琛
江苏省常州高级中学 2019级
指导教师：曹文

2021.3

目录

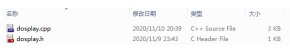
1 前言

对于程序设计研究而言，理论和实践都是必不可少的。虽然网上有现成的代码，但是绝大部分对于我们来说都过于混乱、不可使用。本着研究性学习的原则，我们自己动手实现了绝大部分程序。以下内容将讨论主要的代码构架和代码实现。所有代码均可在https://gitee.com/czyarl/gomoku获取，并在CC-BY-SA 4.0协议下使用。

编译方式：使用makefile进行编译。参见readme.md中的使用教程。

总文件一览（见[实现_总文件示意图]）

(a) basic/

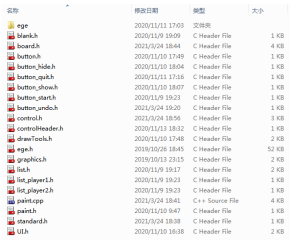


(b) dosplay/

(c) entity/

(d) entity/AIs

(e) libs/



(f) paint/

(g) 总目录

图 1: 实现_总文件示意图

2 概要

程序实现了如下部分：

1. 主程序
2. UI控制模块
3. dos控制模块
4. UI
5. 实体模块（规范了AI算法）

6. 数据模块

各部分联系见实现_控制结构图。

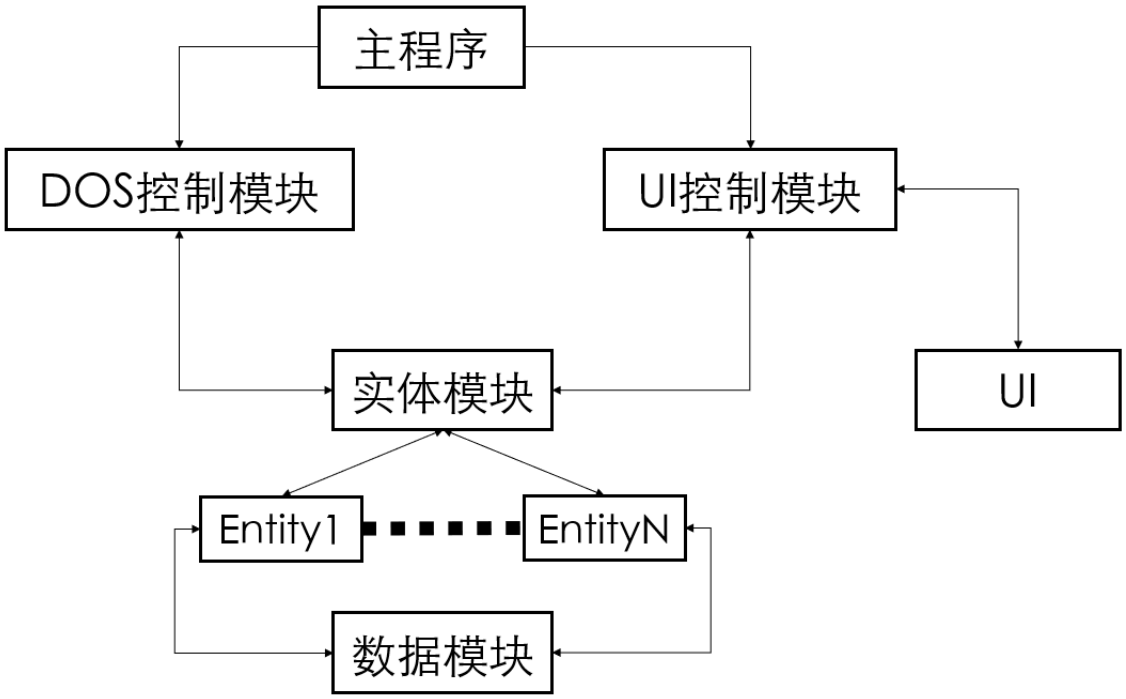


图 2: 实现_控制结构图

开发语言：c++
平台支持：64位Windows7/Windows10（支持可视化）,64位Linux（不支持可视化）

3 主程序

主程序的作用是让用户（仅对于Windows）选择可视化类型，或者进行批量的算法测试比较。（见[实现_主程序示意]）

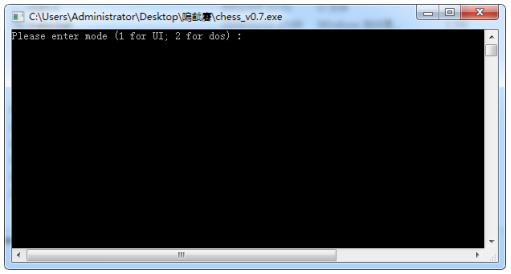


图 3: 实现_主程序示意

若用户输入1，将进入可视化界面（下文讲解）。若输入2，将进入DOS模式进行算法测试比较。

4 UI控制模块

由于本次程序设计代码量比较大（在一个较短的时间内），我们专门设置了控制模块来对各功能进行调控。控制模块主要进行UI与实体模块之间的信息交流。我们将fps设置为了60帧，每次帧结束时，控制模块都会检测是否从UI或实体模块中获得了信息。如果有，那么它将发出相应的指令。主要内容如下：

1. con_NewGame():进行新的一局游戏。
2. con_runGame():若正在进行游戏，那么检测是否落子，并进行相应的检测。
3. con_showNum(bool x):是否显示棋子的落子编号。
4. con_Undo():悔棋。
5. updateAll(status x):更新所有UI。其中x是一个状态变量，由用户决定。
6. init():进行UI的初始化。

5 DOS控制模块

选择DOS模式之后需要手动输入若干信息，使用示例见[实现_DOS模式示意]，详细说明如下：

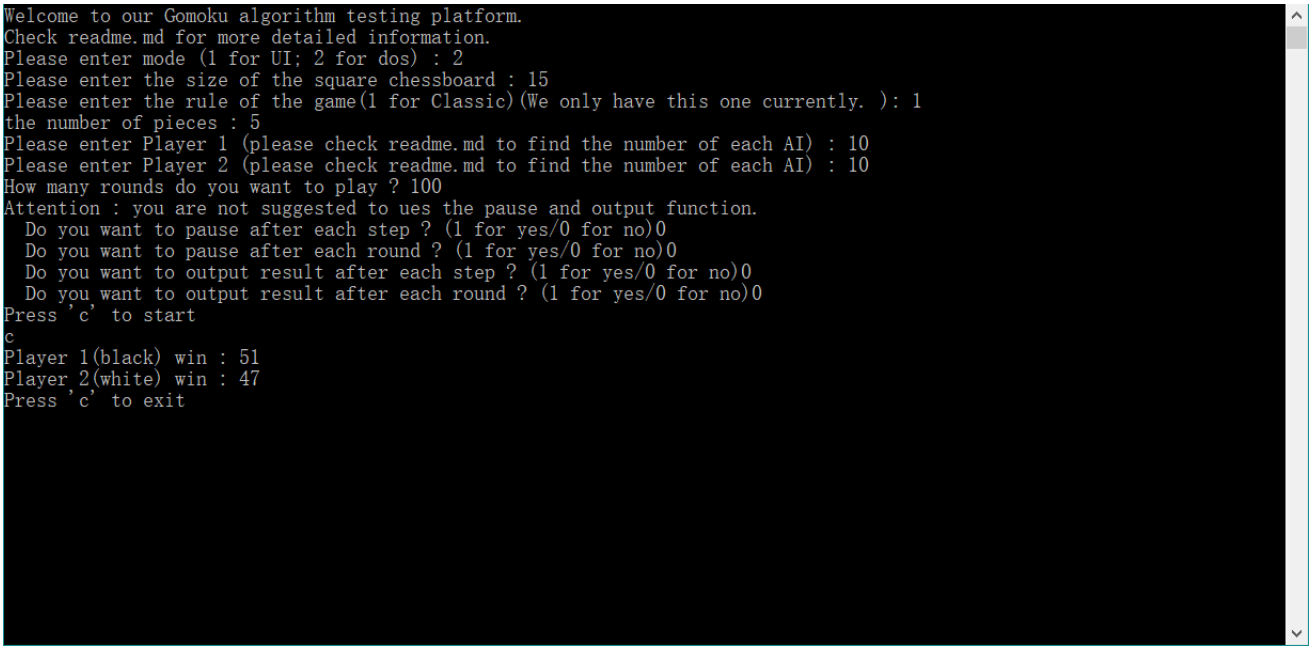


图 4: 实现_DOS模式示意

1. setting mode: 棋盘信息读入模式。1表示使用默认配置，0表示手动输入。若选择使用默认模式，将跳过读入棋盘大小、游戏规则、获胜连子数、随机开局并使用默认值。

2. size of the square chessboard: 棋盘的大小。默认为15
3. rule of the game: 游戏规则，目前只有1。
4. number of pieces: 能获胜的连子个数，为保证安全请输入正整数。可以不为5，不过有部分AI只支持五子棋。默认为5。
5. random gambit: 是否使用随机开局。0为不使用，1为使用如果使用，程序将在棋盘的最中间的3*3中随机生成两枚白子和两枚黑子。该功能用于测试确定性算法，请慎用。
6. Player 1: 黑方的编号。0为人类。如果该编号未出现，那么返回“Invalid Entity”，并尝试重新输入。
7. Player 2: 白方的编号。0为人类。
8. Output results: 在每局对局结束之后输出对局结果到records.txt中以防止程序意外中断导致数据丢失。
9. rounds: 测试的局数。
10. 调试信息，1为是，0为否。
    (a) 每一步棋下完后是否暂停。
    (b) 每一局棋下完后是否暂停。
    (c) 每一步棋下完后是否输出结果。
    (d) 每一局棋下完后是否输出结果。

等待一段时间，将得到两个算法的相关数据。该使用示例显示在100局测试中，算法10作为黑方获胜51局，算法10作为白方获胜47局，剩下的2局为平局。

6 UI

UI是绘图专用模块，它以ege.h为基础，进行了相应的绘制（见[实现_UI示意图]）。

1. 左侧棋盘：用于下棋的地方。其大小可以在代码中修改。
2. start：进行新的一局游戏。
3. show：显示棋子的标号。
4. quit：退出。
5. undo：悔棋。如果是人类和AI对局，那么撤销两步。
6. Player1/Player2:列表，可以自由选择已设计好的算法。如果要添加新算法，那么需要重新编译。

UI模块中用到的主要绘图函数：

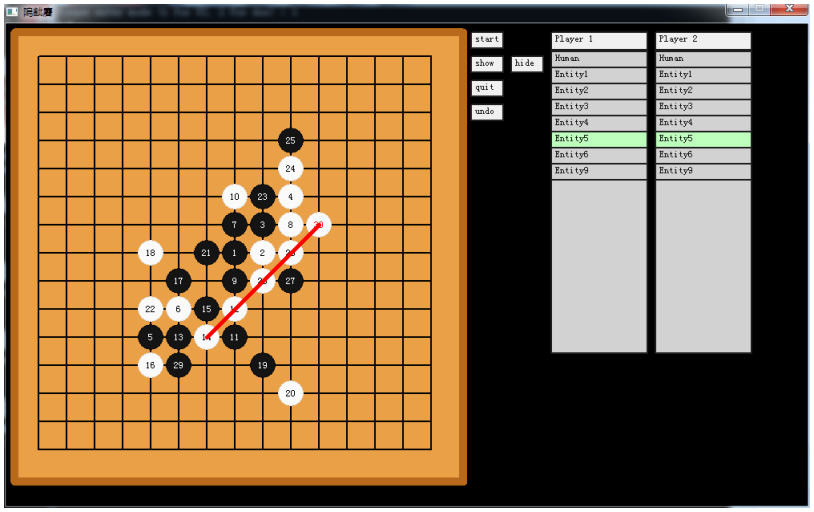


图 5: 实现_UI示意图

1. drawRect:绘制一个矩形。
2. drawLine:绘制一条线段。
3. drawCircle:绘制一个圆。
4. textOutMid:显示文字，并且居中。

UI模块中用到的主要结构体：

1. UI:是一个矩形元件，记录了它的左上角和右下角。
2. status:元件的状态，在信息交换时有用。
3. button:由UI派生而来，顾名思义，是按钮。
4. board:由UI派生而来，是棋盘。
5. list:由UI派生而来，是选择算法的列表。
6. blank:由UI派生而来，是列表的小方格，用于检测是否鼠标左键在了相应的区域内。

7 实体模块

实体模块由主控制和各个Entity组成，规范了AI算法。主控制调用若干Entity，进行相应的信息交换，而Entity就是实现的算法——只不过它们被封装起来了。

一个Entity由两部分组成，一个是定义的头文件*.h,它规范了需要实现的接口：

1. entity_startGame:表示进行了一局新的游戏，会传入棋盘的大小、游戏类型和先后手。
2. entity_playChess:传入当前棋盘、游戏类型和先后手，要求返回一个合法的落子位置。若无法落子，返回$(-1,-1)$。

3. entity_setPos:强制每个位置落子。
4. entity_resetPos:强制悔棋（传入从棋盘上取走的那个点的位置）
5. entity_endGame：结束了游戏。
6. entity_input(Player&):将该算法的前4个函数复制到Player上，即主控制读取了算法的函数接口。

在实现完若干个AI后，调用当前目录下的__build.exe即可生成相应的静态库，完成编译。

接着，在进行游戏时，主控制会收到若干信息：它们要么是进行新的游戏，要么是让某个算法（或者是人类）下一步棋，要么是进行撤销操作，要么是结束游戏：

1. en_startGame:传入棋盘的大小，游戏类型，黑方编号和白方编号。特别地，如果编号为0，说明是人类。
2. en_playChess:传入当前下棋的是哪一方、在哪下以及可能会用到的状态。如果当前算法得到了一个可行的位置，返回这个可行的位置。否则返回(-1,-1)。
3. en_PreviousStep:悔棋。
4. en_endGame:结束游戏。

8 数据模块

主要用于遗传算法的数据记录。