

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	3
2	概要	3
3	主程序	4
4	UI控制模块	5
5	DOS控制模块	5
6	UI	6
7	实体模块	7
8	数据模块	8

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]）

名称	修改日期	类型	大小
BasicDef.h	2020/11/9 20:11	C Header File	1 KB
BasicType.h	2020/11/10 11:47	C Header File	3 KB

(a) basic/

名称	修改日期	类型	大小
dosplay.cpp	2020/11/10 20:39	C++ Source File	3 KB
dosplay.h	2020/11/9 23:43	C Header File	1 KB

(b) $\displaystyle/$

名称	修改日期	属性	大小
Als	2021/3/24 19:59	文件夹	
Entity.h	2020/11/11 17:41	C Header File	1 KB
EntityControl.cpp	2021/3/24 19:16	C++ Source File	4 KB
EntityControl.h	2021/3/24 19:22	C Header File	2 KB

(c) entity/

名称	修改日期	描述	大小
build.hpp	2020/11/11	C++ Source File	3 KB
lib_build.exe	2020/11/11	应用程序	1,319 KB
lib_hello.exe	2020/11/11	C Header File	1 KB
lib_hello2.exe	2020/11/12	787 字节	1 KB
decompile.cpp	2020/11/12	C Header File	1 KB
decompile.hpp	2020/11/12	C++ Source File	1 KB
EmptyLo	2020/11/11	C Header File	1 KB
EmptyLo.cpp	2020/11/12	0 字节	10 KB
EmptyLo2	2020/11/11	C++ Source File	1 KB
EmptyLo3	2020/11/11	C Header File	1 KB
EmptyLo4	2020/11/12	0 字节	10 KB
EmptyLo5	2020/11/11	C++ Source File	1 KB
EmptyLo6	2020/11/11	C Header File	1 KB
EmptyLo7	2020/11/11	C Header File	1 KB
EmptyLo8	2020/11/11	C++ Source File	1 KB
EmptyLo9	2020/11/11	C Header File	1 KB
EmptyLo10	2020/11/11	C Header File	1 KB
EmptyLo11	2020/11/11	C++ Source File	1 KB
EmptyLo12	2020/11/11	C Header File	1 KB
EmptyLo13	2020/11/12	0 字节	10 KB
EmptyLo14	2020/11/11	C++ Source File	1 KB
EmptyLo15	2020/11/11	C Header File	1 KB
EmptyLo16	2020/11/11	C Header File	1 KB
EmptyLo17	2020/11/11	0 字节	10 KB
EmptyLo18	2020/11/11	C++ Source File	1 KB
EmptyLo19	2020/11/11	C Header File	1 KB
EmptyLo20	2020/11/11	C Header File	1 KB
EmptyLo21	2020/11/11	C++ Source File	1 KB
EmptyLo22	2020/11/11	C Header File	1 KB
EmptyLo23	2020/11/11	C Header File	1 KB
EmptyLo24	2020/11/11	C++ Source File	1 KB
EmptyLo25	2020/11/11	C Header File	1 KB
EmptyLo26	2020/11/11	C Header File	1 KB
EmptyLo27	2020/11/11	C++ Source File	1 KB
EmptyLo28	2020/11/11	C Header File	1 KB
EmptyLo29	2020/11/11	C Header File	1 KB
EmptyLo30	2020/11/11	C++ Source File	1 KB
EmptyLo31	2020/11/11	C Header File	1 KB
EmptyLo32	2020/11/11	C Header File	1 KB
EmptyLo33	2020/11/11	C++ Source File	1 KB
EmptyLo34	2020/11/11	C Header File	1 KB
EmptyLo35	2020/11/11	C Header File	1 KB
EmptyLo36	2020/11/11	C++ Source File	1 KB
EmptyLo37	2020/11/11	C Header File	1 KB
EmptyLo38	2020/11/11	C Header File	1 KB
EmptyLo39	2020/11/11	C++ Source File	1 KB
EmptyLo40	2020/11/11	C Header File	1 KB
EmptyLo41	2020/11/11	C Header File	1 KB
EmptyLo42	2020/11/11	C++ Source File	1 KB
EmptyLo43	2020/11/11	C Header File	1 KB
EmptyLo44	2020/11/11	C Header File	1 KB
EmptyLo45	2020/11/11	C++ Source File	1 KB
EmptyLo46	2020/11/11	C Header File	1 KB
EmptyLo47	2020/11/11	C Header File	1 KB
EmptyLo48	2020/11/11	C++ Source File	1 KB
EmptyLo49	2020/11/11	C Header File	1 KB
EmptyLo50	2020/11/11	C Header File	1 KB
EmptyLo51	2020/11/11	C++ Source File	1 KB
EmptyLo52	2020/11/11	C Header File	1 KB
EmptyLo53	2020/11/11	C Header File	1 KB
EmptyLo54	2020/11/11	C++ Source File	1 KB
EmptyLo55	2020/11/11	C Header File	1 KB
EmptyLo56	2020/11/11	C Header File	1 KB
EmptyLo57	2020/11/11	C++ Source File	1 KB
EmptyLo58	2020/11/11	C Header File	1 KB
EmptyLo59	2020/11/11	C Header File	1 KB
EmptyLo60	2020/11/11	C++ Source File	1 KB
EmptyLo61	2020/11/11	C Header File	1 KB
EmptyLo62	2020/11/11	C Header File	1 KB
EmptyLo63	2020/11/11	C++ Source File	1 KB
EmptyLo64	2020/11/11	C Header File	1 KB
EmptyLo65	2020/11/11	C Header File	1 KB
EmptyLo66	2020/11/11	C++ Source File	1 KB
EmptyLo67	2020/11/11	C Header File	1 KB
EmptyLo68	2020/11/11	C Header File	1 KB
EmptyLo69	2020/11/11	C++ Source File	1 KB
EmptyLo70	2020/11/11	C Header File	1 KB
EmptyLo71	2020/11/11	C Header File	1 KB
EmptyLo72	2020/11/11	C++ Source File	1 KB
EmptyLo73	2020/11/11	C Header File	1 KB
EmptyLo74	2020/11/11	C Header File	1 KB
EmptyLo75	2020/11/11	C++ Source File	1 KB
EmptyLo76	2020/11/11	C Header File	1 KB
EmptyLo77	2020/11/11	C Header File	1 KB
EmptyLo78	2020/11/11	C++ Source File	1 KB
EmptyLo79	2020/11/11	C Header File	1 KB
EmptyLo80	2020/11/11	C Header File	1 KB
EmptyLo81	2020/11/11	C++ Source File	1 KB
EmptyLo82	2020/11/11	C Header File	1 KB
EmptyLo83	2020/11/11	C Header File	1 KB
EmptyLo84	2020/11/11	C++ Source File	1 KB
EmptyLo85	2020/11/11	C Header File	1 KB
EmptyLo86	2020/11/11	C Header File	1 KB
EmptyLo87	2020/11/11	C++ Source File	1 KB
EmptyLo88	2020/11/11	C Header File	1 KB
EmptyLo89	2020/11/11	C Header File	1 KB
EmptyLo90	2020/11/11	C++ Source File	1 KB
EmptyLo91	2020/11/11	C Header File	1 KB
EmptyLo92	2020/11/11	C Header File	1 KB
EmptyLo93	2020/11/11	C++ Source File	1 KB
EmptyLo94	2020/11/11	C Header File	1 KB
EmptyLo95	2020/11/11	C Header File	1 KB
EmptyLo96	2020/11/11	C++ Source File	1 KB
EmptyLo97	2020/11/11	C Header File	1 KB
EmptyLo98	2020/11/11	C Header File	1 KB
EmptyLo99	2020/11/11	C++ Source File	1 KB
EmptyLo100	2020/11/11	C Header File	1 KB

(d) entity/AIs

名称	修改日期	类型	大小
lib4ls.a	2020/11/11 19:15	A 文件	11
libgraphics64.a	2019/10/13 23:25	A 文件	1.17

(e) `libs/`

名称	创建日期	创建者	大小
egg	2020/11/17 17:03	张奕	1
blackbox	2020/11/9 15:09	C# Header File	1
	2021/2/24 18:44	C# Header File	4
	2020/11/17 17:49	C# Header File	1
button_hdujs	2020/11/10 18:04	C# Header File	1
button_guoh	2020/11/11 17:16	C# Header File	1
	2021/2/24 15:07	C# Header File	1
button_start	2020/11/19 19:20	C# Header File	1
button_end	2021/2/24 19:21	C# Header File	1
	2021/2/24 18:58	C# Header File	1
comunicatehdujs	2020/1/11 13:52	C# Header File	1
	2020/11/17 17:48	C# Header File	2
drawTriangle	2019/10/26 18:45	C# Header File	52
	2021/2/24 13:15	C# Header File	2
polyhich	2020/11/19 18:17	C# Header File	2
16h	2020/11/19 18:23	C# Header File	2
16h_playchess	2021/2/24 18:41	C++ Source Code	4
playchess	2020/11/16 19:47	C# Header File	1
	2021/2/24 18:38	C# Header File	1
	2021/2/24 18:18	C# Header File	2

(f) paint/

[illegible]

(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。

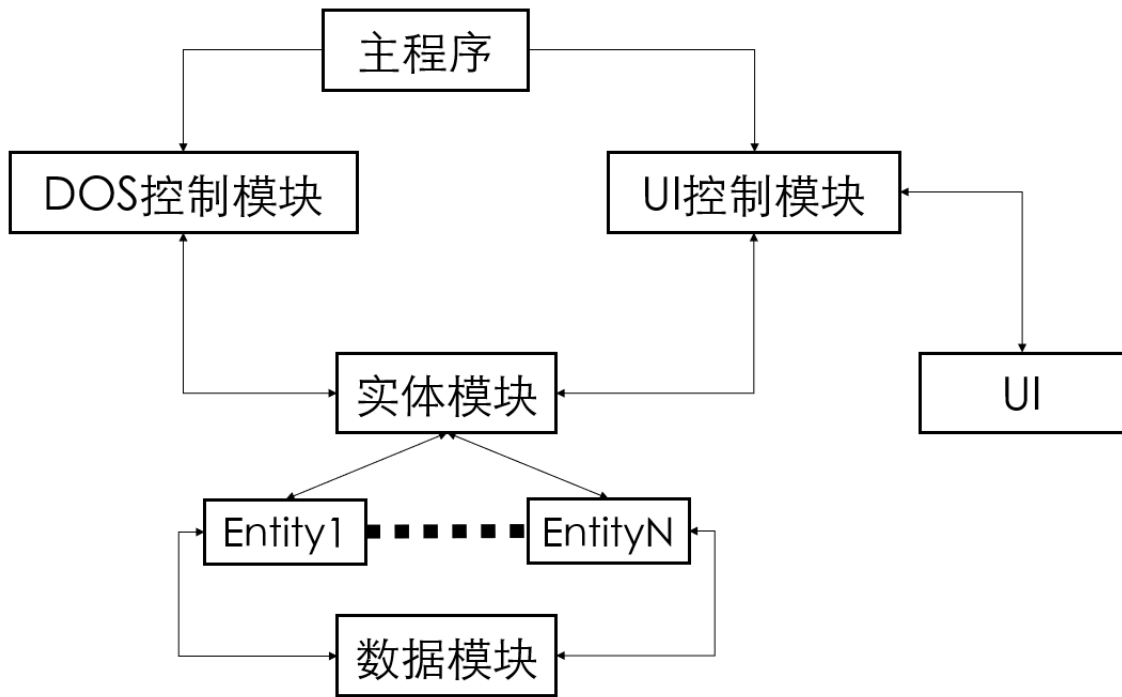


图 2: 实现_控制结构图

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]）

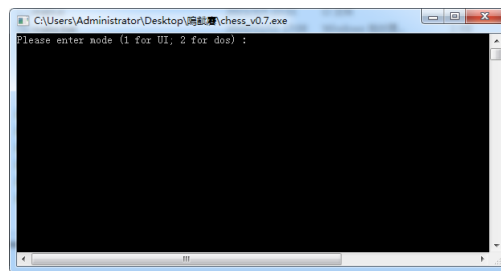


图 3: 实现_主程序示意

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：

```
Welcome to our Gomoku algorithm testing platform.
Check readme.md for more detailed information.
Please enter mode (1 for UI; 2 for dos) : 2
Please enter the size of the square chessboard : 15
Please enter the rule of the game(1 for Classic)(We only have this one currently. ): 1
the number of pieces : 5
Please enter Player 1 (please check readme.md to find the number of each AI) : 10
Please enter Player 2 (please check readme.md to find the number of each AI) : 10
How many rounds do you want to play ? 100
Attention : you are not suggested to ues the pause and output function.
  Do you want to pause after each step ? (1 for yes/0 for no)0
  Do you want to pause after each round ? (1 for yes/0 for no)0
  Do you want to output result after each step ? (1 for yes/0 for no)0
  Do you want to output result after each round ? (1 for yes/0 for no)0
Press 'c' to start
c
Player 1(black) win : 51
Player 2(white) win : 47
Press 'c' to exit
```

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：

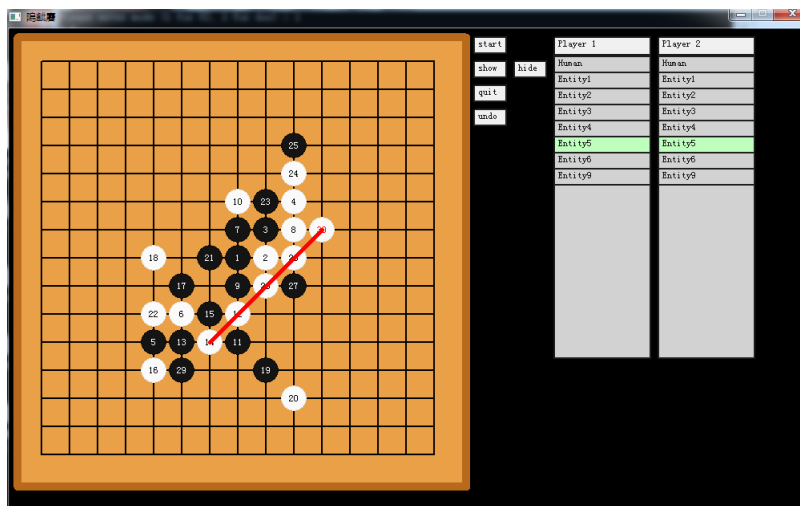


图 5: 实现_UI示意图

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。