

# 快速使用手冊

May 10, 2023

# Outline

1 Download

2 Get Ready

- Open Room
- Enter Room

3 Sample Code

# Download

- 下載愛因斯坦棋遊戲平台
- 下載 Java
- 下載愛因斯坦棋 sample code
- 下載 MinGW (參考教學)

# Get Ready - Open Room

進入 open 資料夾。

名稱	狀態	修改日期	類型	大小
enter	✓	2023/4/20 下午 03:27	檔案資料夾	
open	✓	2023/4/20 下午 03:27	檔案資料夾	
5_4_3.md	✓	2022/12/2 下午 11:45	Markdown 來源...	6 KB
README.pdf	✓	2022/12/2 下午 11:45	Adobe Acrobat ...	1,841 KB
RUN_ALL.bat	✓	2022/12/2 下午 11:45	Windows 批次檔案	1 KB
RUN_ENTER.bat	✓	2022/12/2 下午 11:45	Windows 批次檔案	1 KB
RUN_OPEN.bat	✓	2022/12/2 下午 11:45	Windows 批次檔案	1 KB

Figure 1: open 資料夾

# Get Ready - Open Room

資料夾結構如下圖所示，請雙擊 Launcher.jar 開啟檔案。

名稱	狀態	修改日期	類型	大小
Library	✓	2023/4/20 下午 03:27	檔案資料夾	
LocalServer	✓	2023/4/20 下午 03:27	檔案資料夾	
Search	✓	2023/4/20 下午 03:27	檔案資料夾	
Setting	✓	2023/4/20 下午 03:27	檔案資料夾	
WebServer	✓	2023/4/20 下午 03:27	檔案資料夾	
Launcher.jar	✓	2022/12/2 下午 11:45	Executable Jar File	2,958 KB

Figure 2: 資料夾結構

# Get Ready - Open Room

開啟後，會出現如下圖視窗。

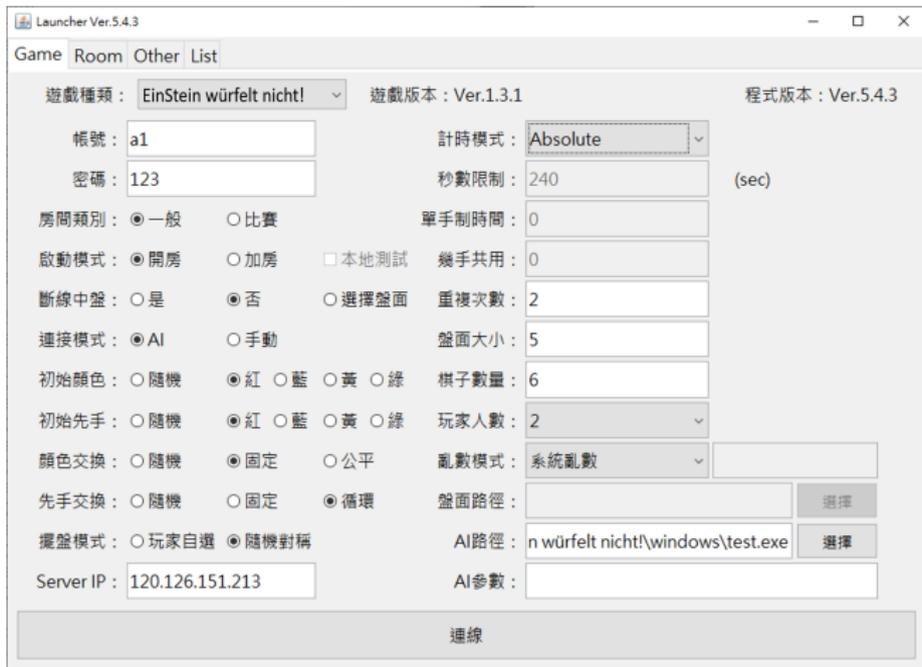


Figure 3: 程式初始畫面

# Get Ready - Open Room

- 帳號/密碼：
  - 訪客帳號為 a1 至 a20000
  - 密碼皆為 123
- 擺盤模式：
  - 玩家自選 (由玩家自行設定初始盘面)
  - 隨機對稱 (由 server 設定隨機且雙方玩家對稱的盘面)
- Server IP 不需更改
- 重覆次數：總共要進行幾局對戰
- AI 路徑：選擇寫好的 AI 程式 (.exe 檔)

# Get Ready - Open Room

按下連線按鈕，與伺服器連線。

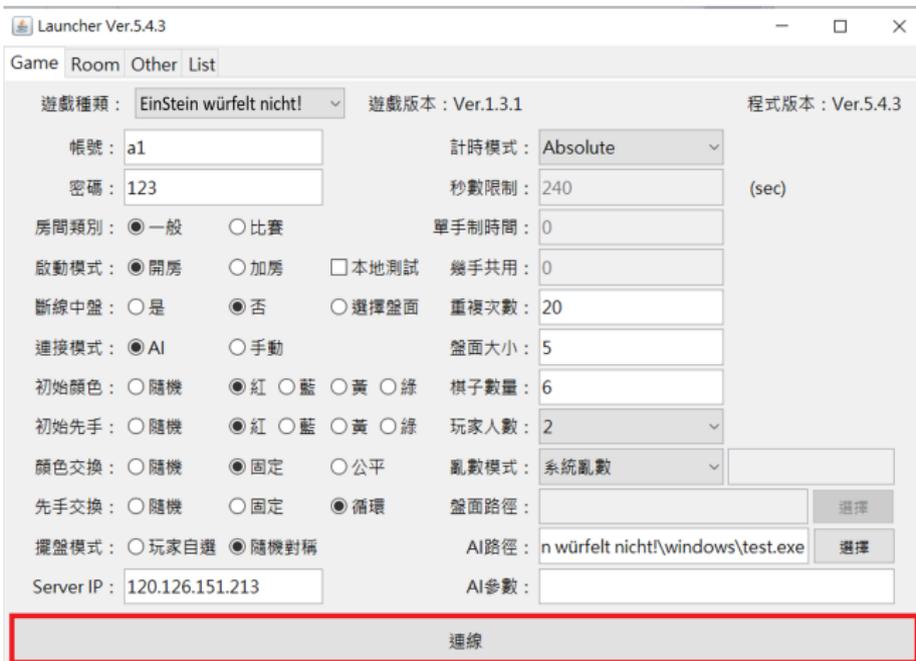


Figure 4: 連接

# Get Ready - Open Room

會看到網頁開啟如下圖。

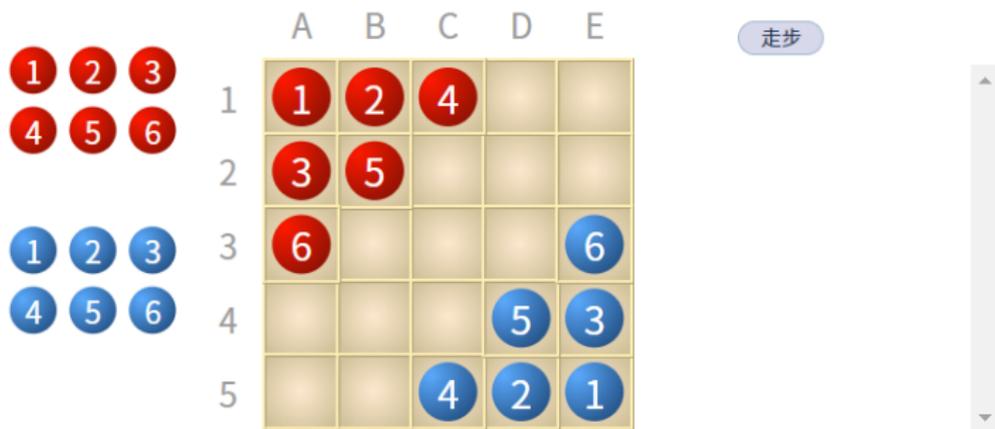


Figure 5: 盤面網頁

# Get Ready - Open Room

程式會開啟如下圖的頁面。

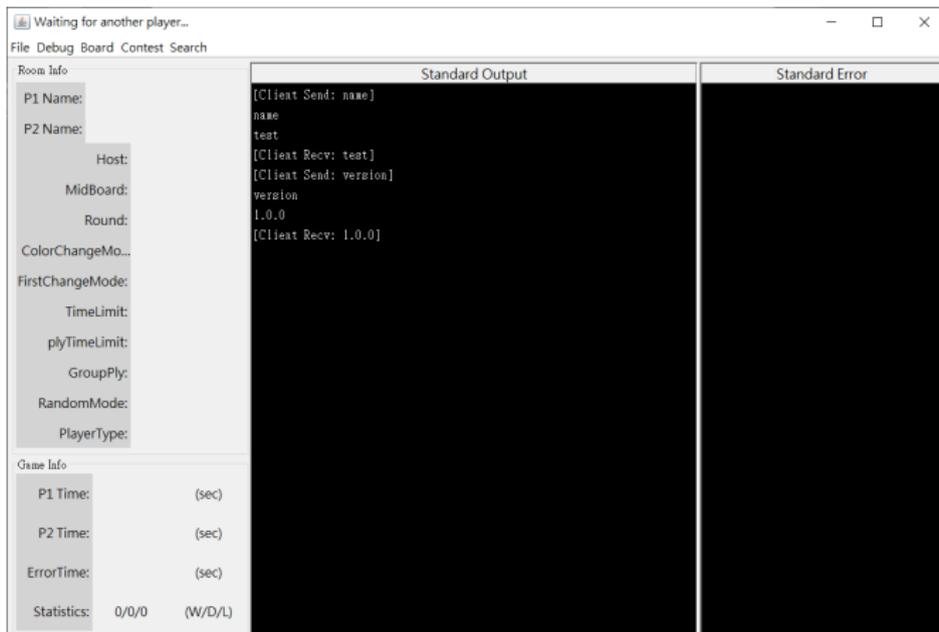


Figure 6: 房間頁面

# Get Ready - Enter Room

進入 enter 資料夾。

名稱	狀態	修改日期	類型	大小
enter	✓	2023/4/20 下午 03:27	檔案資料夾	
open	✓	2023/4/20 下午 03:27	檔案資料夾	
5_4_3.md	✓	2022/12/2 下午 11:45	Markdown 來源...	6 KB
README.pdf	✓	2022/12/2 下午 11:45	Adobe Acrobat ...	1,841 KB
RUN_ALL.bat	✓	2022/12/2 下午 11:45	Windows 批次檔案	1 KB
RUN_ENTER.bat	✓	2022/12/2 下午 11:45	Windows 批次檔案	1 KB
RUN_OPEN.bat	✓	2022/12/2 下午 11:45	Windows 批次檔案	1 KB

Figure 7: enter 資料夾

# Get Ready - Enter Room

資料夾結構如下圖所示，請雙擊 Launcher.jar 開啟檔案。

名稱	狀態	修改日期	類型	大小
Library	✓	2023/4/20 下午 03:27	檔案資料夾	
LocalServer	✓	2023/4/20 下午 03:27	檔案資料夾	
Search	✓	2023/4/20 下午 03:27	檔案資料夾	
Setting	✓	2023/4/20 下午 03:27	檔案資料夾	
WebServer	✓	2023/4/20 下午 03:27	檔案資料夾	
Launcher.jar	✓	2022/12/2 下午 11:45	Executable Jar File	2,958 KB

Figure 8: 資料夾結構

# Get Ready - Enter Room

開啟後，會出現如下圖視窗。

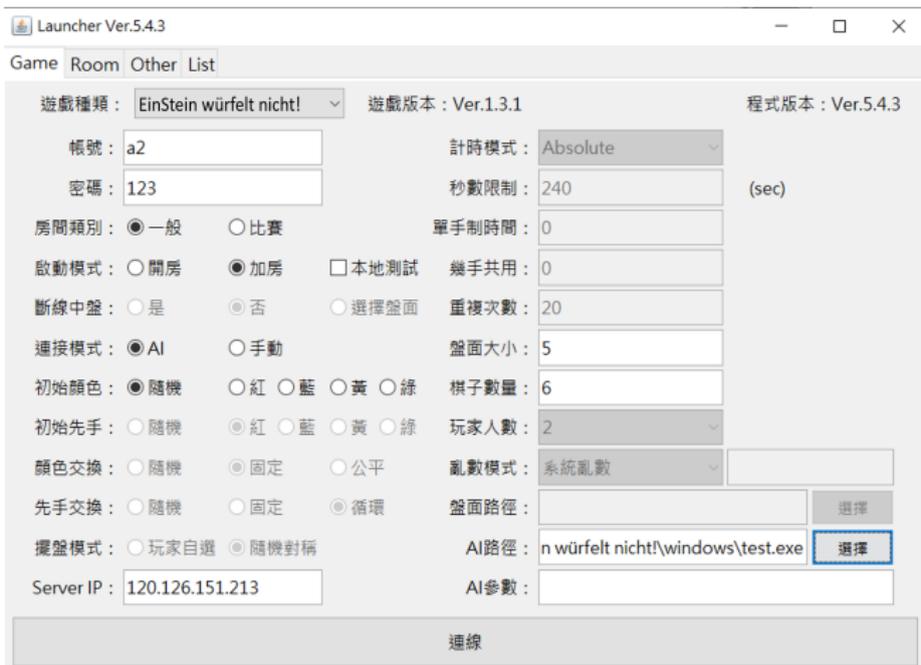


Figure 9: 程式初始畫面

# Get Ready - Enter Room

- 加房方不需設定房間相關設定
- 帳號/密碼：
  - 訪客帳號為 a1 至 a20000
  - 密碼皆為 123
- Server IP 不需更改
- AI 路徑：選擇寫好的 AI 程式 (.exe 檔)

# Get Ready - Enter Room

按連線按鈕，與伺服器連線。

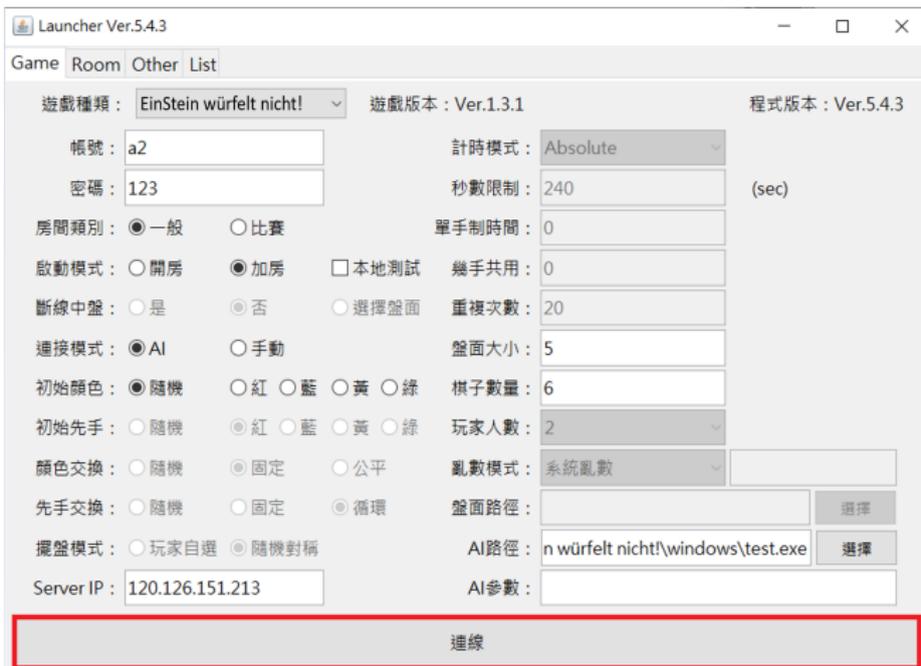


Figure 10: 連接

# Get Ready - Enter Room

會看到網頁開啟如下圖。

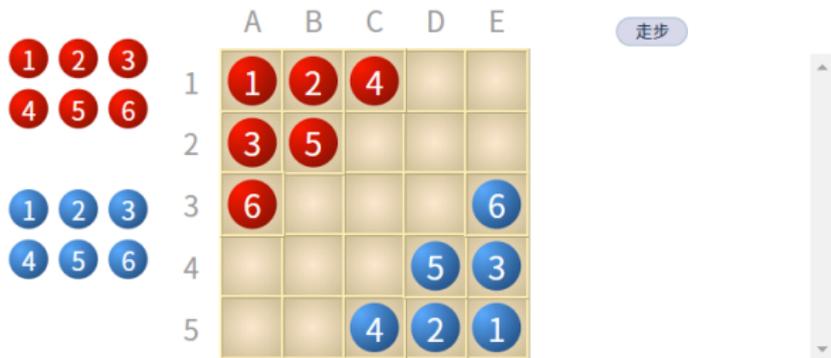


Figure 11: 盤面網頁

# Get Ready - Enter Room

程式會開啟如下圖的頁面，按下 1. 紅框處確認房間資訊，再按 2. OK 按鈕加房。

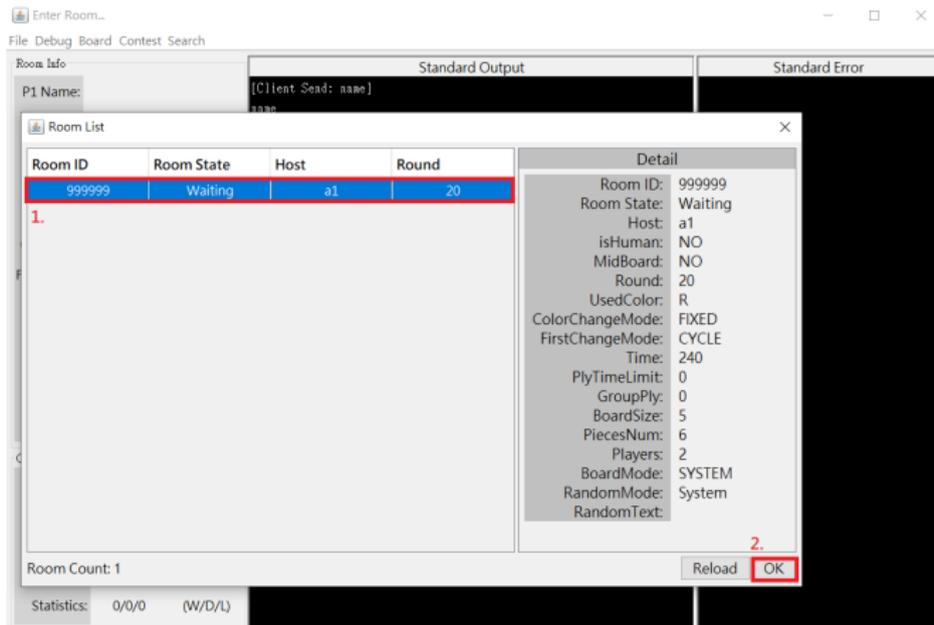


Figure 12: 房間列表

# Get Ready

如看到雙方畫面皆如下圖所示，即成功完成測試。

The screenshot displays a game test application window titled "C:\Users\zihyang\Desktop\game platform\windows\enter\Search\EinSteinChess\windows\test.exe". The interface is divided into several sections:

- Room Info:** P1 Name: a2 (blue square), P2 Name: a1 (red square), Host: <====, MidBoard: NO, Round: 20, ColorChangeMo...: FIXED, FirstChangeMode: CYCLE, TimeLimit: 240, plyTimeLimit: 0, GroupPly: 0, RandomMode: System, PlayerType: AI vs AI.
- Game Info:** P1 Time: 239.523 (sec), P2 Time: 239.474 (sec), ErrorTime: 0.000 (sec), Statistics: 1/0/0 (W/D/L).
- Standard Output:** A log of game events including board settings, ply requests, and board states. For example: "[Client Recv: 1]", "[Client Send: board\_getting 5 6 2]", "board\_getting 5 6 2", "[Client Recv: 1]", "[Client Send: get B 4 B4 D5 D4 E3 E5 C5 A2 B1 B2 A3 A1 C1]", "get B 4 B4 D5 D4 E3 E5 C5 A2 B1 B2 A3 A1 C1", "E3 D2", "[Client Recv: E3 D2]", "[Client Send: get B 6 B4 D5 D4 D2 E5 C5 B3 B1 B2 A3 A1 C1]", "get B 6 B4 D5 D4 D2 E5 C5 B3 B1 B2 A3 A1 C1", "C5 B4", "[Client Recv: C5 B4]", "[Client Send: get B 2 B4 D5 D4 D2 E5 B4 C4 B1 B2 A3 A1 C1]", "get B 2 B4 D5 D4 D2 E5 B4 C4 B1 B2 A3 A1 C1", "D5 C4", "[Client Recv: D5 C4]", "[Client Send: get B 5 B4 C4 D4 D2 E5 B4 C2 B2 A3 A1 C1]", "get B 5 B4 C4 D4 D2 E5 B4 C2 B2 A3 A1 C1", "B5 D4", "[Client Recv: B5 D4]", "[Client Send: get B 4 B4 C4 C0 D2 D4 C0 C2 B2 B4 A1 C1]", "get B 4 B4 C4 C0 D2 D4 C0 C2 B2 B4 A1 C1", "D2 C1", "[Client Recv: D2 C1]".
- Standard Error:** A log of board states and ply numbers, such as "R5R2B0[]", "[R3[]B4[]", "R4R1[][]", "[[]]B3B1", "[[]]B0B2B5", "R5R2B0[]", "[R3[]B4[]", "R4[][][]", "[B0R1B3B1", "[[]]B2B5", "R5[]R0[]", "[R3R2B4[]", "R4[][][]", "[B0B2B3B1", "[[]]B5", "R5[]R0[]", "[R3R2B4[]", "[[]][]", "[R4B2B5B1", "[[]][]".

Figure 13: 遊戲進行中

# Get Ready

- Room Info: 房間相關資訊
- Game Info: 玩家剩餘時間、勝負情形
- Standard Output: AI 程式與 Client 互相溝通的指令
- Standard Error: 用 stderr 印出的資訊 (debug)

# Get Ready

網頁上會顯示當前遊戲進行狀況。

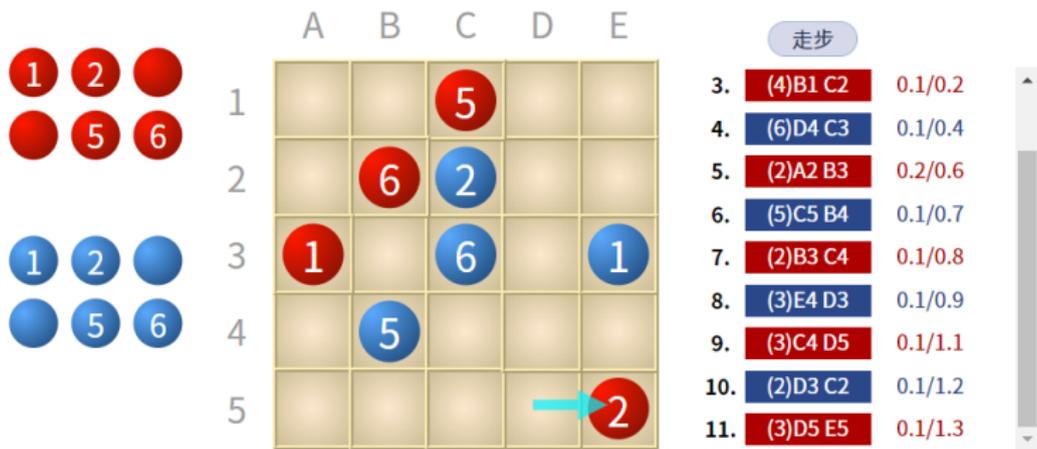


Figure 14: 遊戲進行中

## Sample Code - Board

1	A1	B1	C1	D1	E1
2	A2	B2	C2	D2	E2
3	A3	B3	C3	D3	E3
4	A4	B4	C4	D4	E4
5	A5	B5	C5	D5	E5

A B C D E

Figure 15: 棋盤位置代號

## Sample Code - Commands

command	meaning
name	回傳 AI 程式名稱
version	回傳 AI 程式版本
time_setting	設定時間
board_setting	設定盤面大小、棋子數量、玩家人數
ini	要求產生擺棋位置
get	要求產生走步
exit	要求程式終止

Figure 16: 指令列表

# Sample Code - Commands

- name
  - Client sends "name"
  - AI sends "MyAI"
- version
  - Client sends "version"
  - AI sends "1.0.0"
- time\_setting
  - Client sends "time\_setting 240 0 0"
  - AI sends "1"
- board\_setting
  - Client sends "board\_setting 5 6 2"
  - AI sends "1"

# Sample Code - Commands

- ini
  - Client sends "ini B"
  - AI sends "C5 D4 E3 E5 D5 E4"
- get
  - Client sends "get B 5 E3 E5 D4 D5 E4 C5 A3 B2 0 B1 A2 C1"
  - AI sends "E4 D3"
- exit
  - Client sends "exit"
  - Close the AI

# Sample Code

開啟 EWN sample\_code 資料夾。

 main	2022/11/22 上午 09:40	C++ Source File	2 KB
 MyAI	2022/11/22 上午 09:40	C++ Source File	11 KB
 MyAI	2022/11/22 上午 09:40	C Header File	2 KB

Figure 17: 資料夾結構

# Sample Code

- Sample code 中已處理好 Command 的接收與回覆
- 修改 MyAI.cpp 中的 Generate\_move 函式並設計搜尋演算法
- 若整個程式都想自己寫，請注意 Command 是否都能正確接收、回覆

# Sample Code

修改 MyAI.cpp 中的 Generate\_move 函式並設計搜尋演算法。

```
void MyAI::Generate_move(char* move)
{
    int result[100];
    // get legal moves
    int move_count = this->get_legal_move(result);
    // randomly choose a legal move
    int rand_move = rand() % move_count;
    int piece = result[rand_move * 3];
    int start_point = result[rand_move * 3 + 1];
    int end_point = result[rand_move * 3 + 2];
    sprintf(move, "%c%c%c%c", 'A' + start_point % BOARD_SIZE, '1' + start_point / BOARD_SIZE,
            piece, end_point);
    this->Make_move(piece, start_point, end_point);
    // print the result
    fprintf(stderr, "=====\nMy result:\n");
    if(piece <= PIECE_NUM) fprintf(stderr, "Blue piece %d: (%c%c) -> (%c%c)\n", piece, move[0], move[1]);
    else fprintf(stderr, "Red piece %d: (%c%c) -> (%c%c)\n", piece - PIECE_NUM, move[0], move[1]);
    this->Print_chessboard();
    fprintf(stderr, "=====\n");
}
```

將紅框區塊修改成搜尋演算法，並產生出走步

Figure 18: Generate\_move

## Sample Code - MyAI

result 陣列用來存走步，每三個為一走步。

0: piece 1: start\_point 2: end\_point

```
181 void MyAI::Generate_move(char* move)
182 {
183     int result[100];
184     // get legal moves
185     int move_count = this->get_legal_move(result);
186     // randomly choose a legal move
187     int rand_move = rand() % move_count;
188     int piece = result[rand_move * 3];
189     int start_point = result[rand_move * 3 + 1];
190     int end_point = result[rand_move * 3 + 2];
```

Figure 19: result

## Sample Code - MyAI

用來產生目前骰到的棋的合法走步，將走步存進 result 陣列，並回傳 move\_count 代表合法走步數量。

```
181 void MyAI::Generate_move(char* move)
182 {
183     int result[100];
184     // get legal moves
185     int move_count = this->get_legal_move(result);
186     // randomly choose a legal move
187     int rand_move = rand() % move_count;
188     int piece = result[rand_move * 3];
189     int start_point = result[rand_move * 3 + 1];
190     int end_point = result[rand_move * 3 + 2];
```

Figure 20: get\_legal\_move

## Sample Code - MyAI

piece: 要走的棋 (Red: 1 to 6 Blue: 7 to 12)

start\_point: 走步的起始位置 (參考 Figure22 的代號)

end\_point: 走步的目標位置 (參考 Figure22 的代號)

```
181 void MyAI::Generate_move(char* move)
182 {
183     int result[100];
184     // get legal moves
185     int move_count = this->get_legal_move(result);
186     // randomly choose a legal move
187     int rand_move = rand() % move_count;
188     int piece = result[rand_move * 3];
189     int start_point = result[rand_move * 3 + 1];
190     int end_point = result[rand_move * 3 + 2];
```

Figure 21: 三個變數

## Sample Code - MyAI

1	0	1	2	3	4
2	5	6	7	8	9
3	10	11	12	13	14
4	15	16	17	18	19
5	20	21	22	23	24
	A	B	C	D	E

Figure 22: AI 程式產生走步時的盤面位置代號

## Sample Code - MyAI

將 piece, start\_point, end\_point 傳進 Make\_move 函式移動走步。

```
181 void MyAI::Generate_move(char* move)
182 {
183     int result[100];
184     // get legal moves
185     int move_count = this->get_legal_move(result);
186     // randomly choose a legal move
187     int rand_move = rand() % move_count;
188     int piece = result[rand_move * 3];
189     int start_point = result[rand_move * 3 + 1];
190     int end_point = result[rand_move * 3 + 2];
191     sprintf(move, "%c%c%c", 'A' + start_point % BOARD_SIZE, '1' + s
192     this->Make_move(piece, start point, end point);
193     // print the result
194     fprintf(stderr, "=====\nMy result:\n");
195     if(piece <= PIECE_NUM) fprintf(stderr, "Blue piece %d: (%c%c) ->
196     else fprintf(stderr, "Red piece %d: (%c%c) -> (%c%c)\n", piece -
197     this->Print_chessboard();
198     fprintf(stderr, "=====\n");
```

Figure 23: Make\_move

## Sample Code - Compile

輸入 cmd，按 enter 可開啟 cmd。

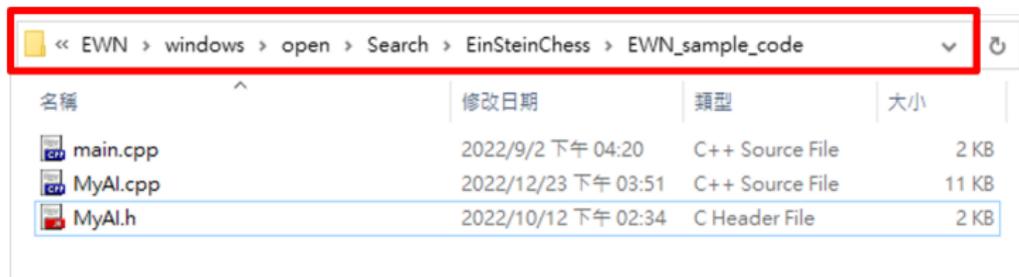
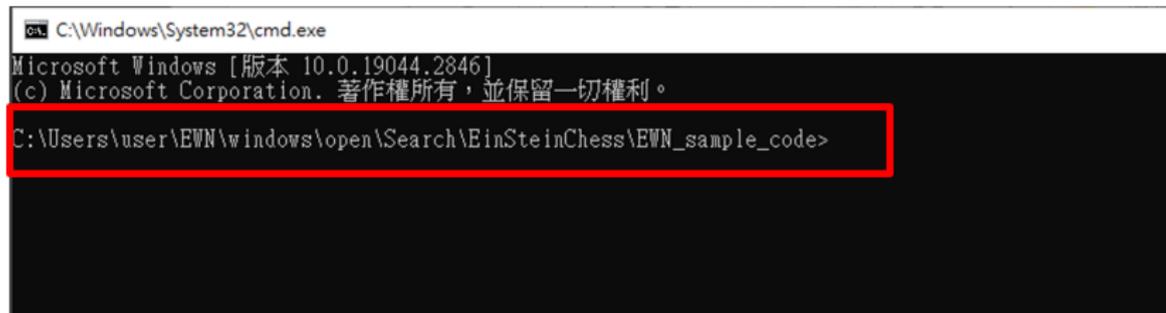


Figure 24: 開啟 cmd

## Sample Code - Compile

確認路徑 (需與 MyAI.cpp 同路徑)。



```
C:\Windows\System32\cmd.exe
Microsoft Windows [版本 10.0.19044.2846]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\user\EWN\windows\open\Search\EinSteinChess\EWN_sample_code>
```

Figure 25: 確認路徑

## Sample Code - Compile

指令格式: `g++ -o <EXE 名稱 >< 編譯檔案 >`

舉例: `g++ -o MyAI main.cpp MyAI.cpp`

若順利執行，資料夾下會產生一 exe 檔。

名稱 ^	狀態	修改日期	類型	大小
 main.cpp		2022/9/2 下午 04:20	C++ 來源檔案	2 KB
 MyAI.cpp		2023/4/19 下午 12:12	C++ 來源檔案	11 KB
 MyAI.exe		2023/4/18 下午 03:54	應用程式	61 KB
 MyAI.h		2022/10/12 下午 02:34	C Header 來源檔案	2 KB

Figure 26: 產生執行檔