快速使用手冊

May 10, 2023

Outline

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。

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。

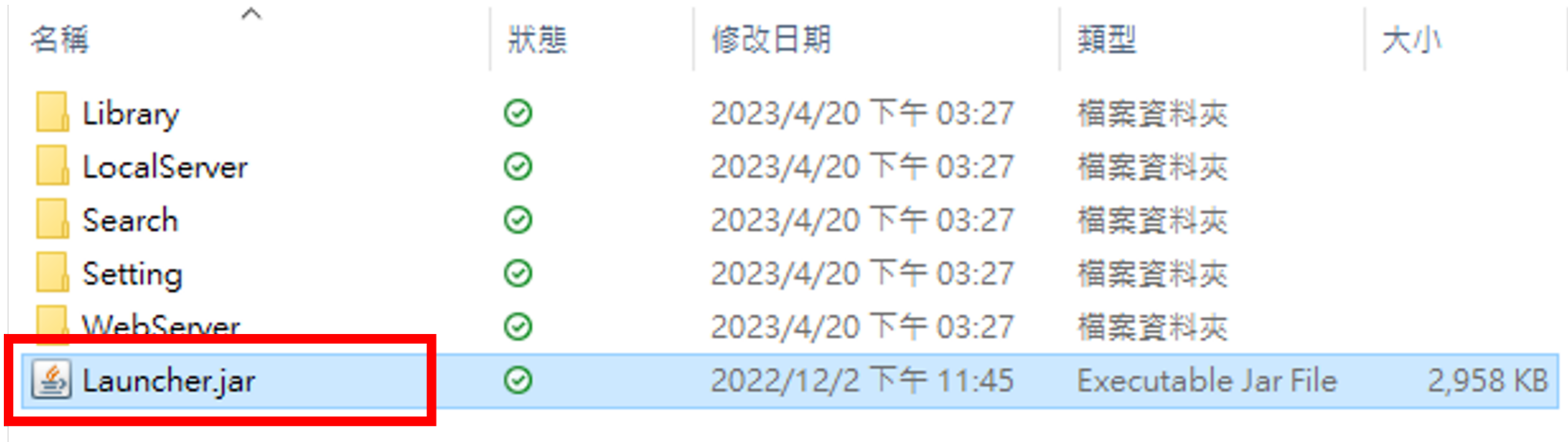


Figure 2: **資料夾結構**

Get Ready - Open Room

開啟後，會出現如下圖視窗。

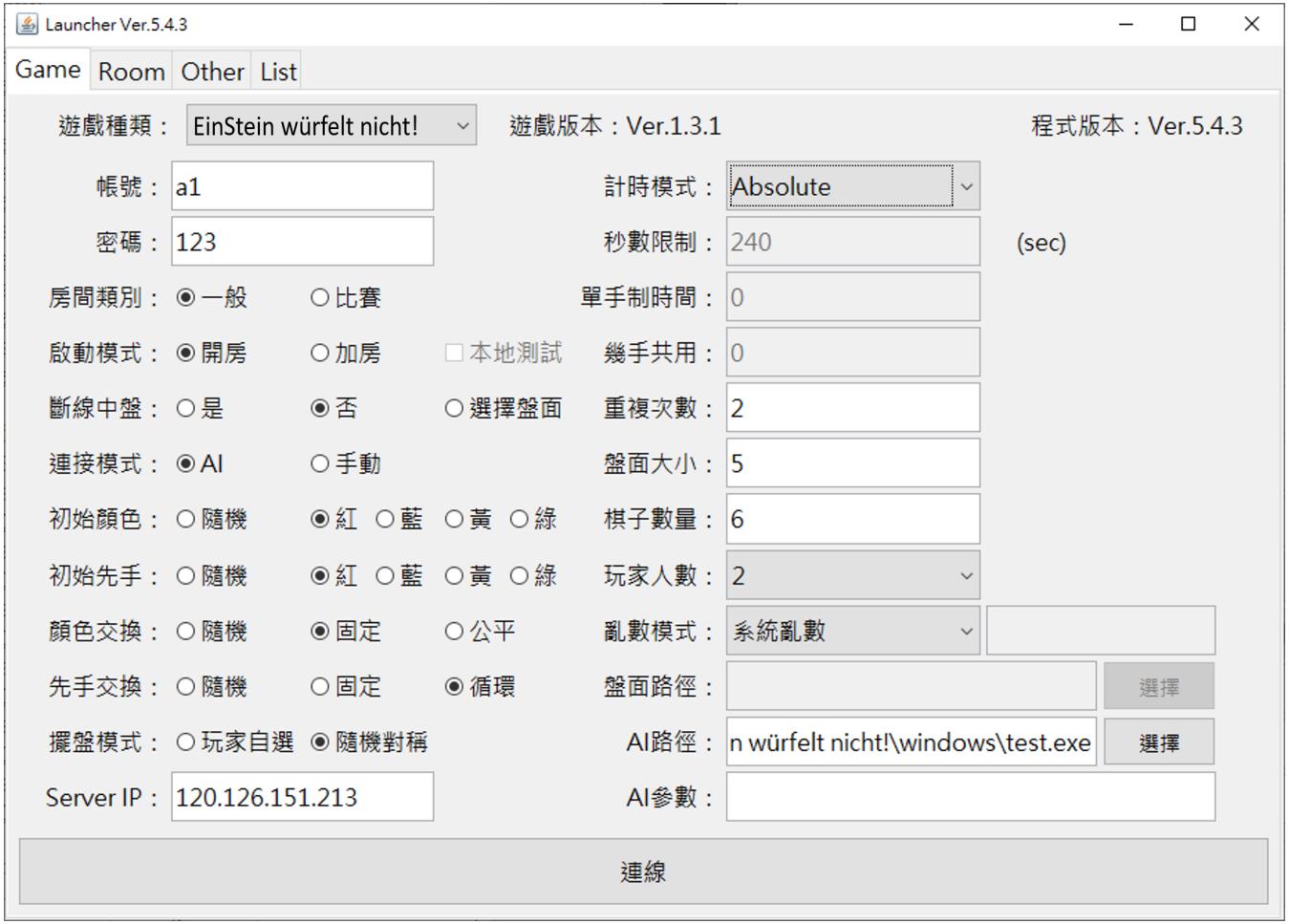


Figure 3: **程式初始畫面**

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盤面)
  - 隨機對稱 (由 server 設定隨機且雙方玩家對稱的盤面)
- 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。

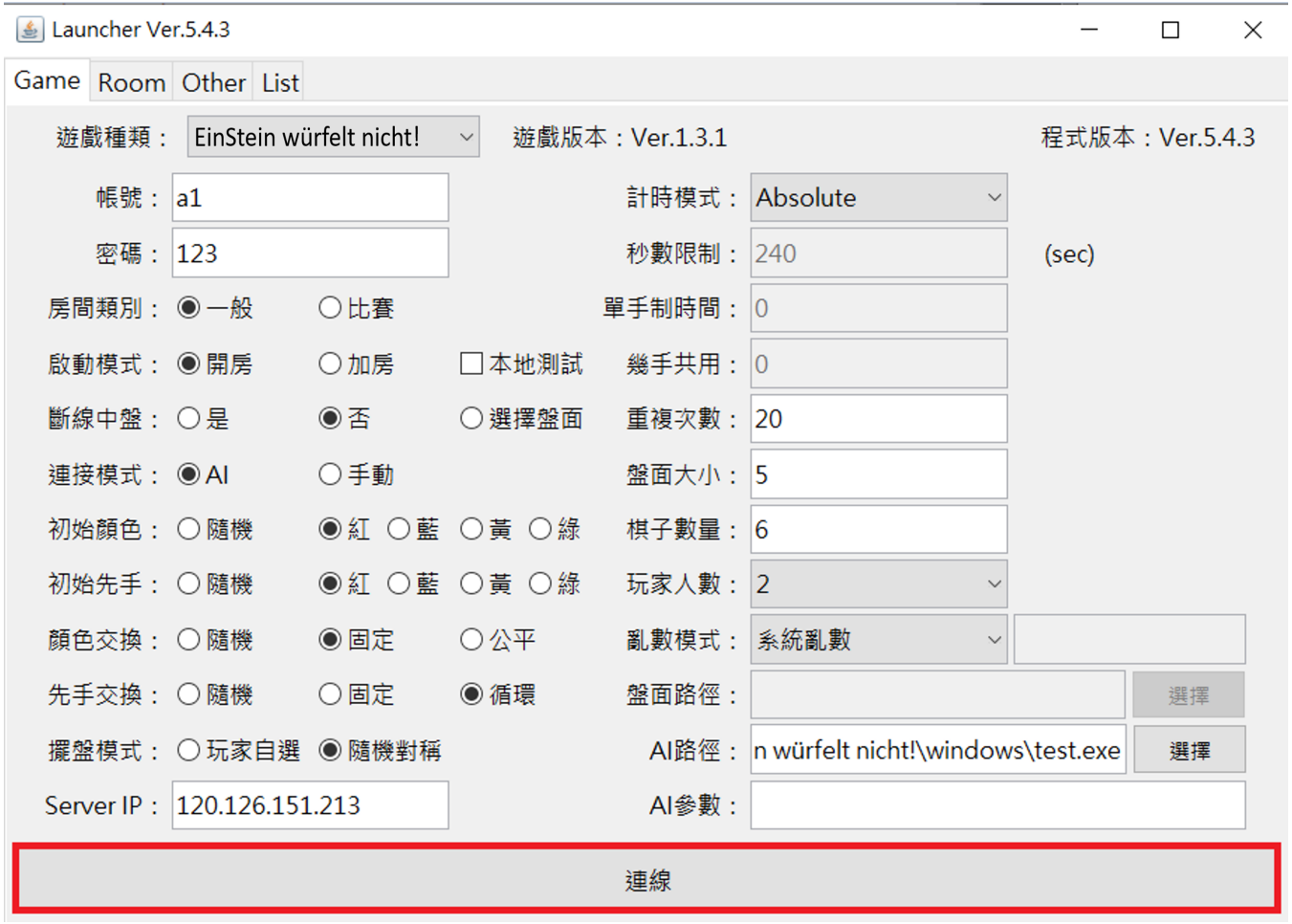


Figure 4: **連接**

Get Ready - Open Room

會看到網頁開啟如下圖。

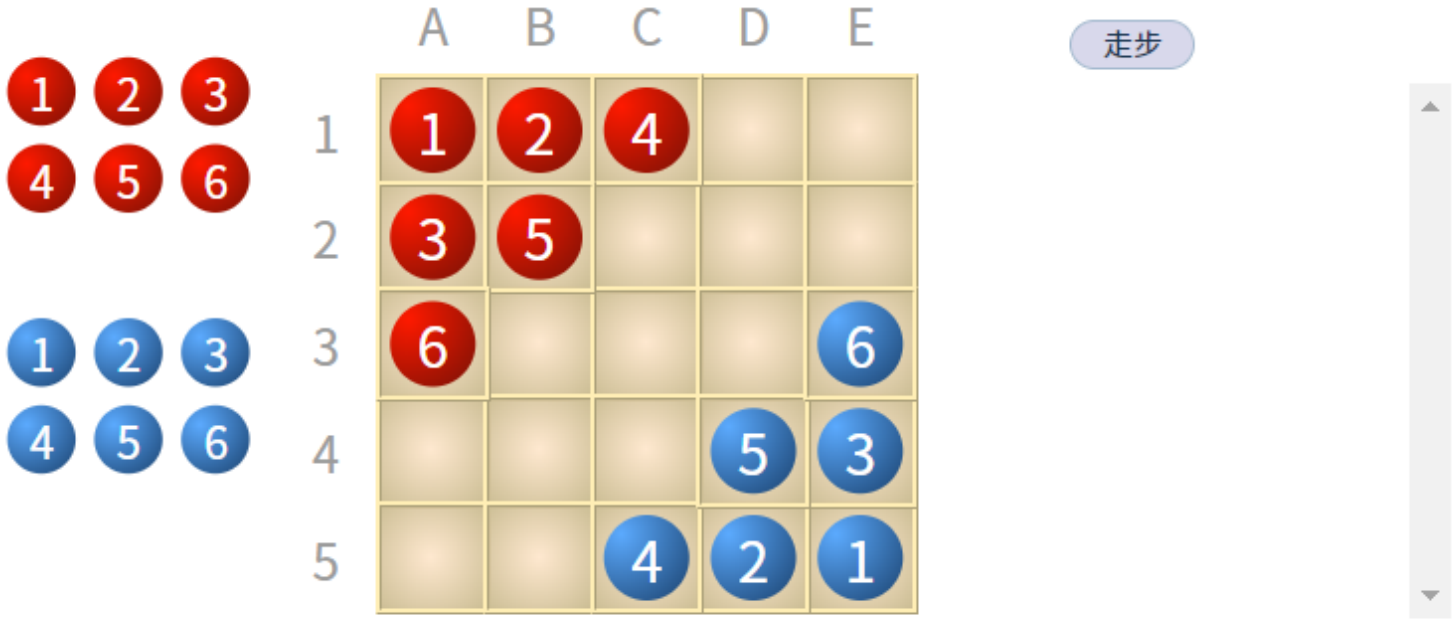


Figure 5: **盤面網頁**

Get Ready - Open Room

程式會開啟如下圖的頁面。

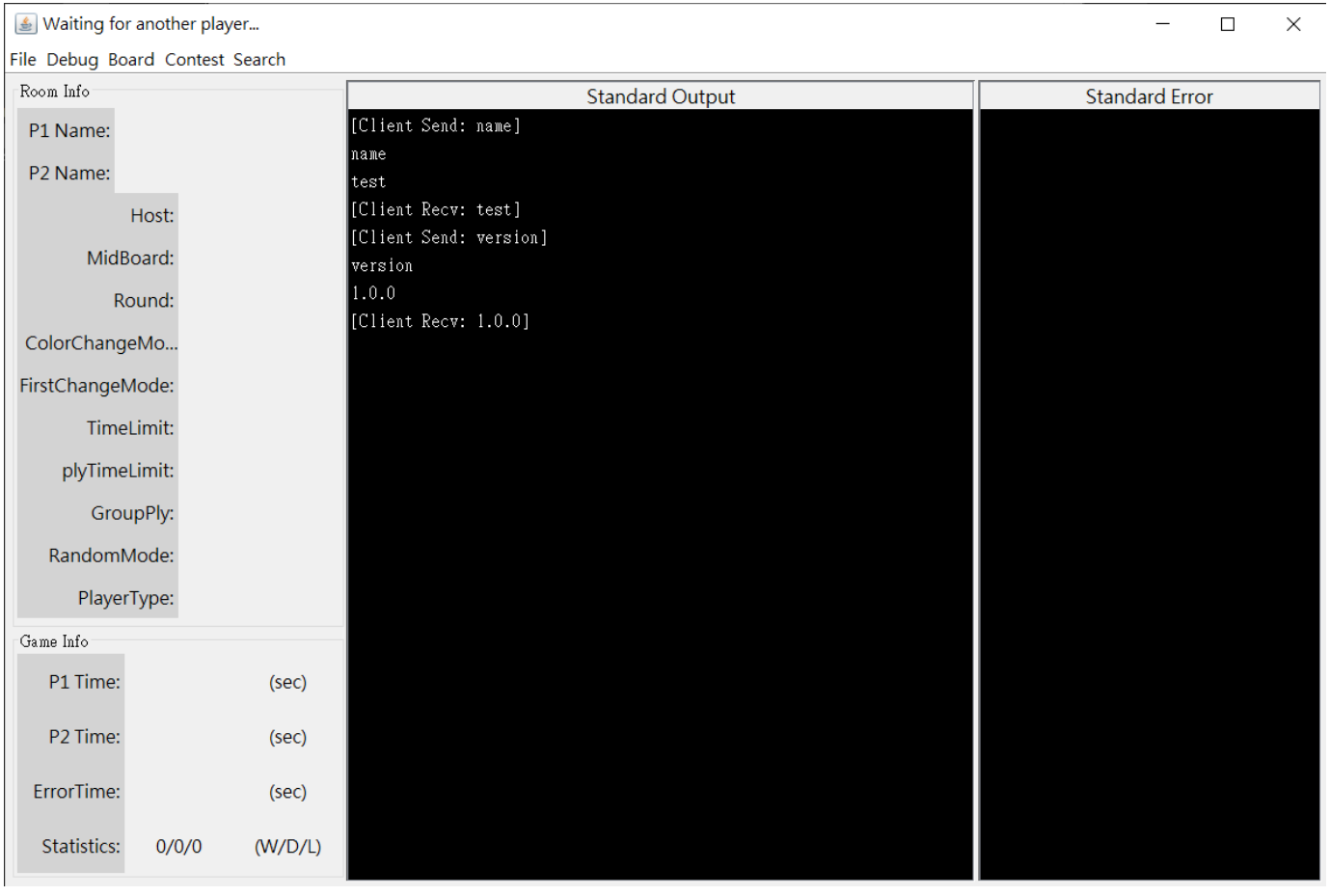


Figure 6: 房間頁面

Get Ready - Enter Room

進入 enter 資料夾。

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。

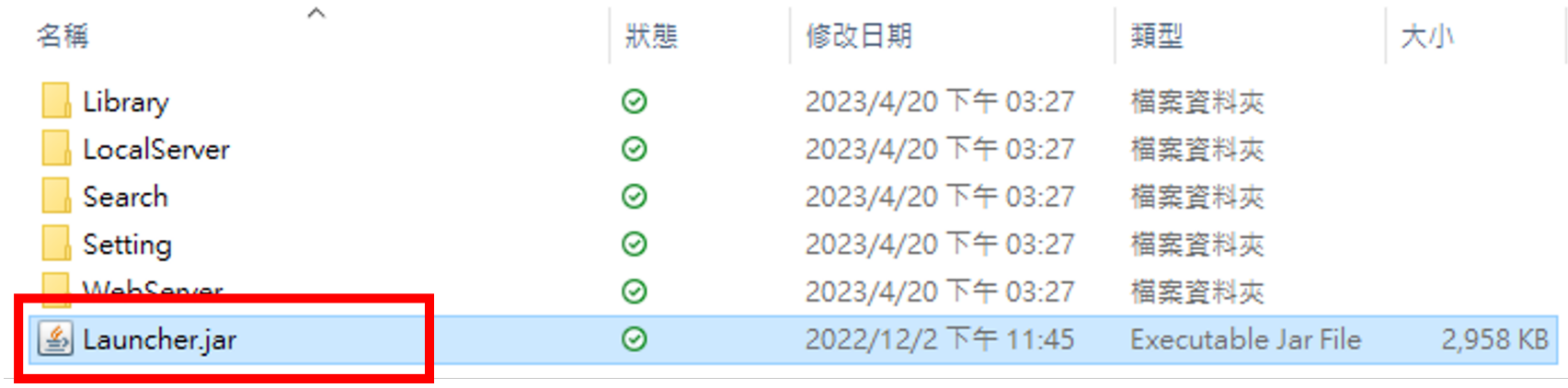


Figure 8: 資料夾結構

Get Ready - Enter Room

開啟後，會出現如下圖視窗。

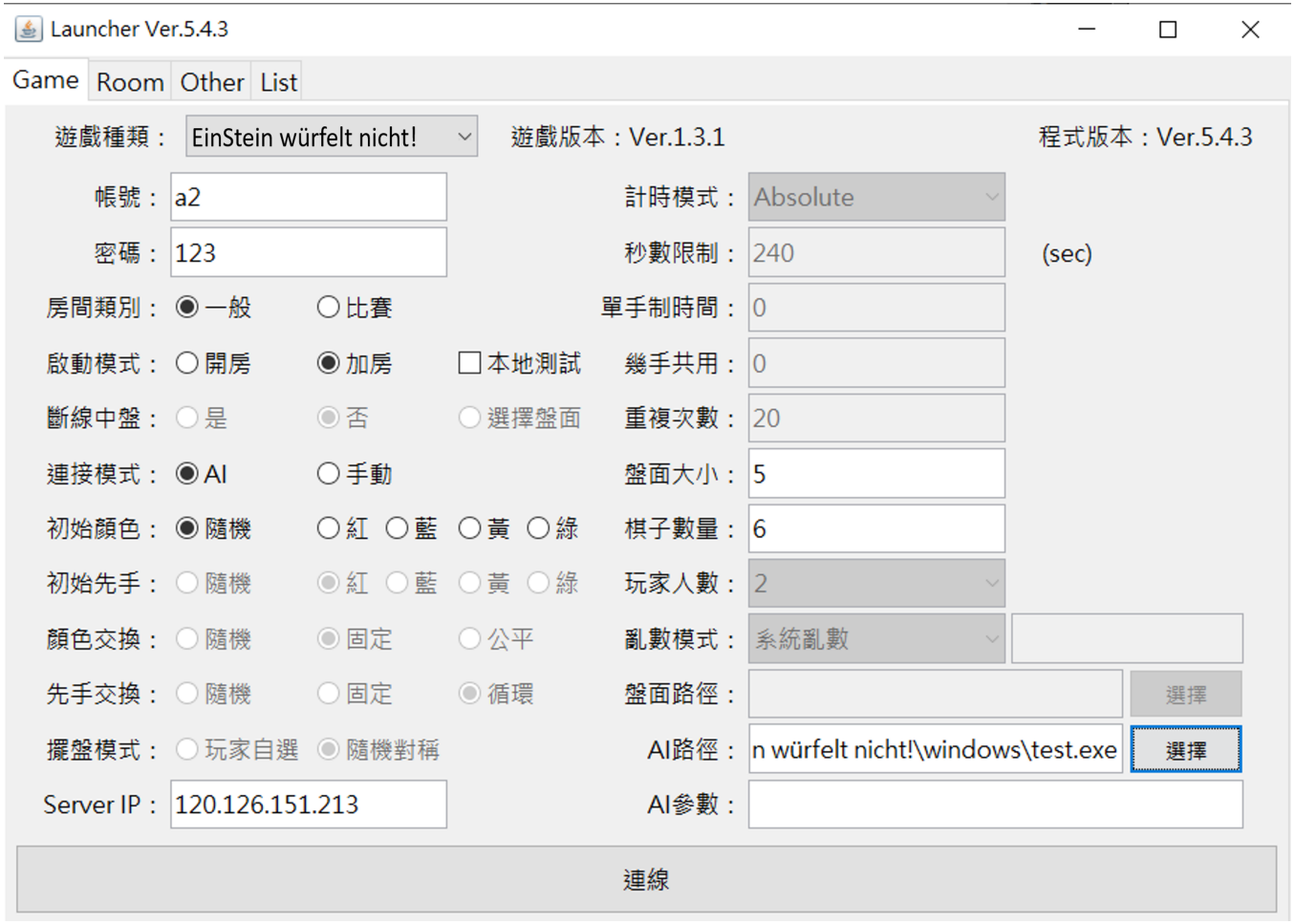


Figure 9: **程式初始畫面**

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。

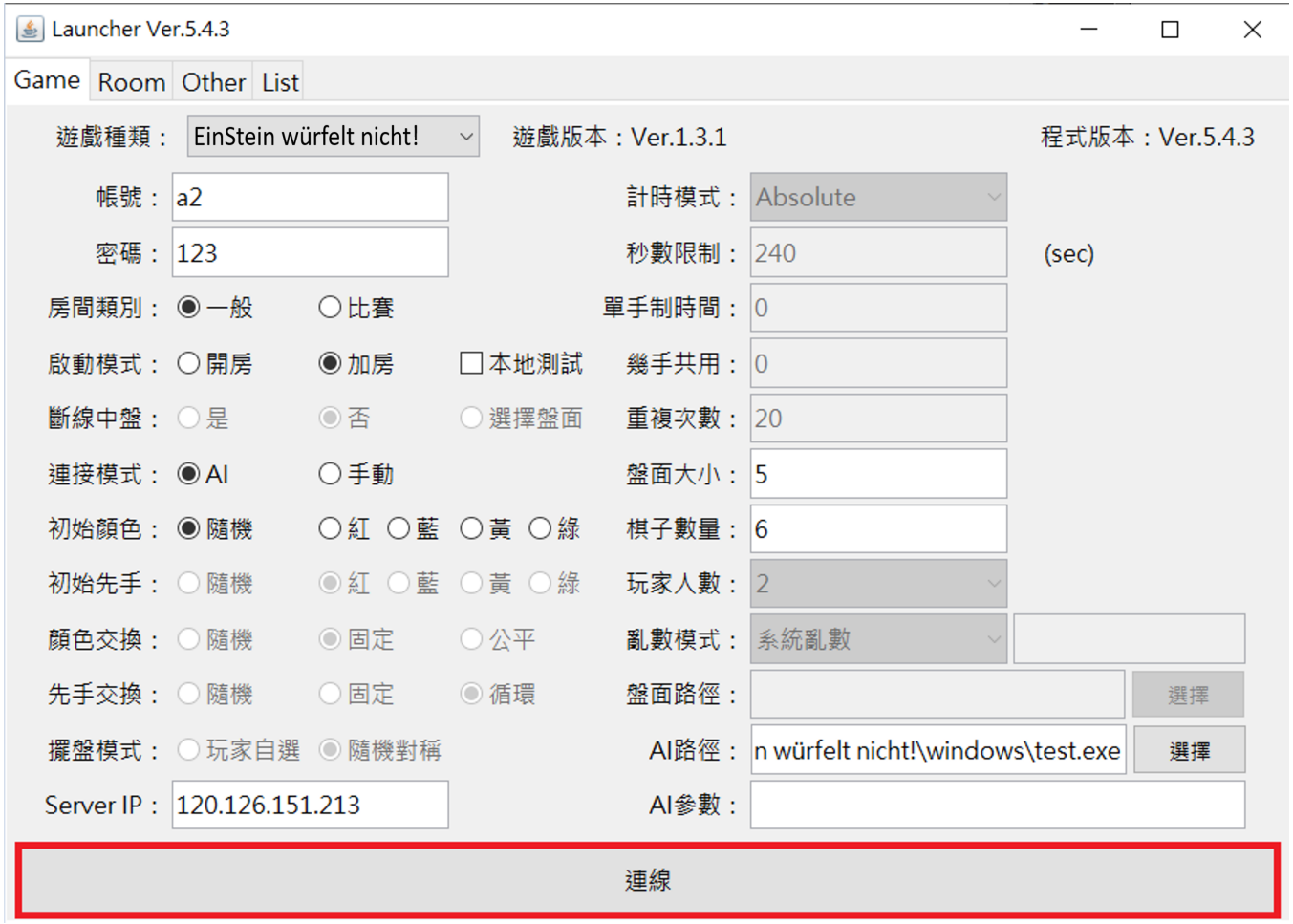


Figure 10: **連接**

Get Ready - Enter Room

會看到網頁開啟如下圖。

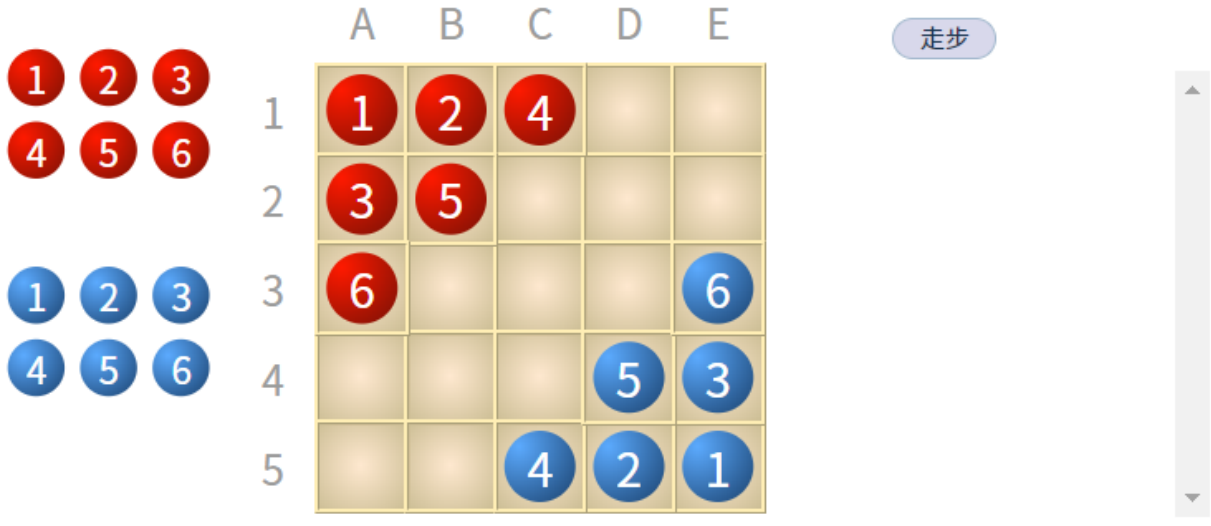


Figure 11: **盤面網頁**

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。

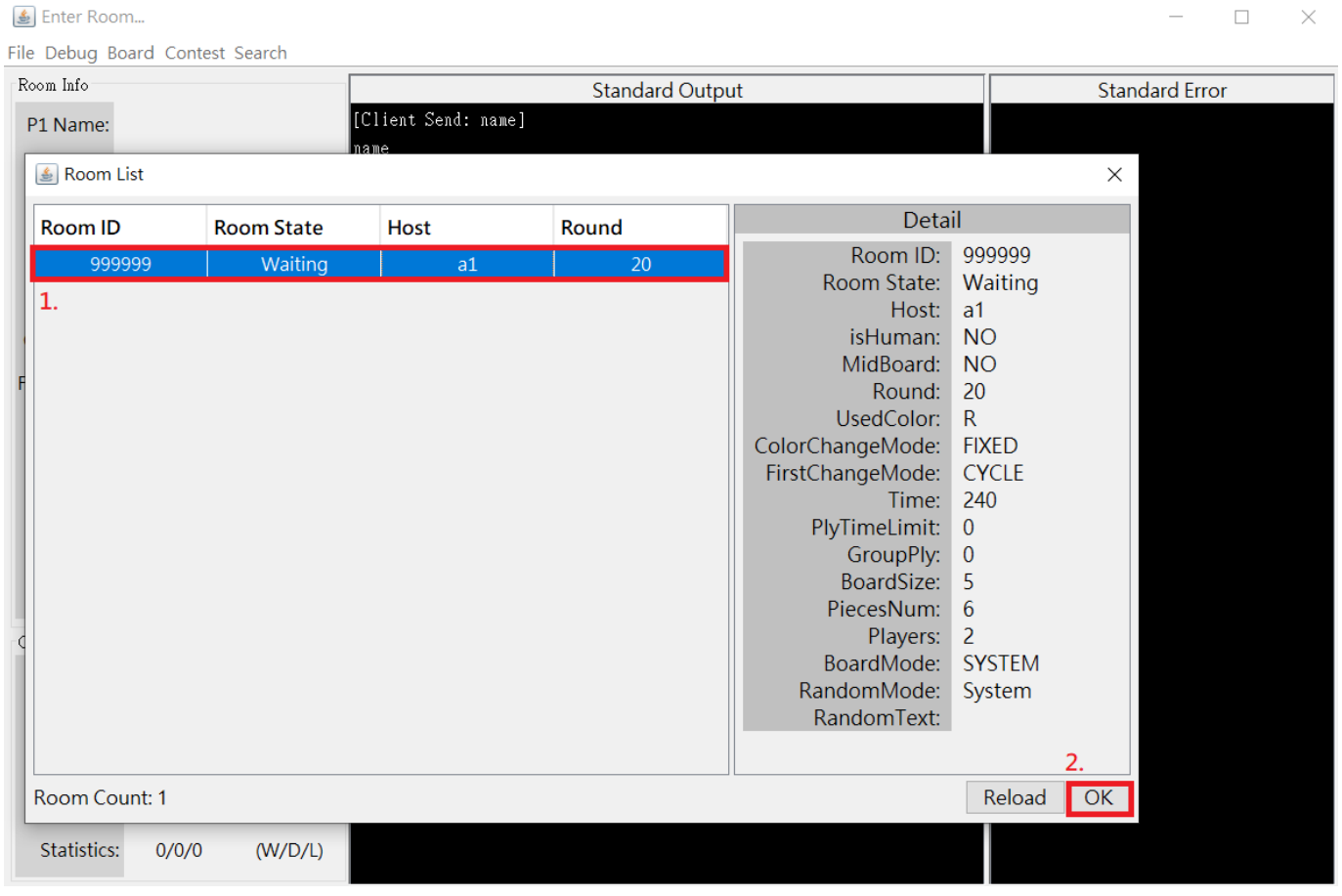


Figure 12: 房間列表

Get Ready

如看到雙方畫面皆如下圖所示，即成功完成測試。

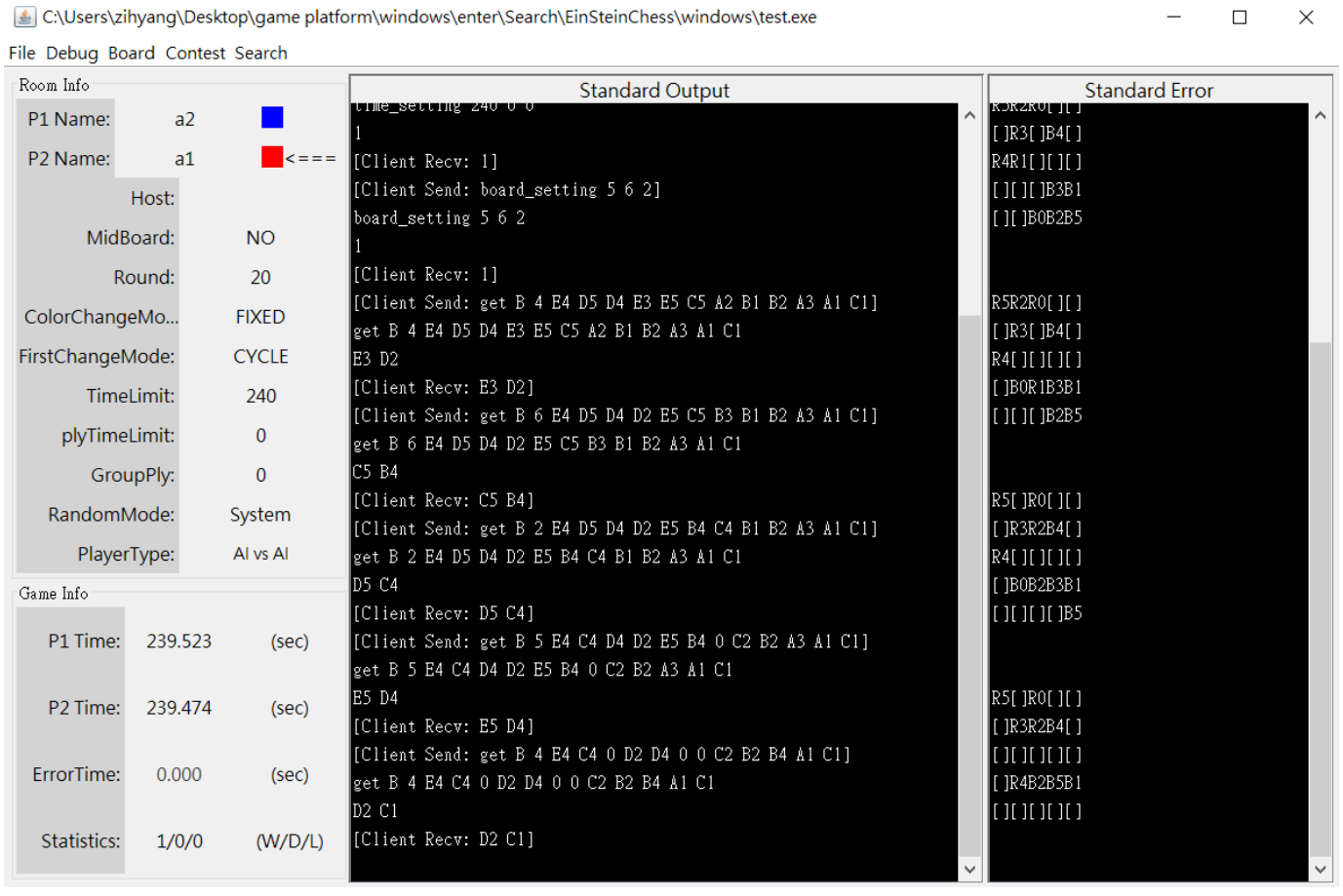


Figure 13: 遊戲進行中

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。

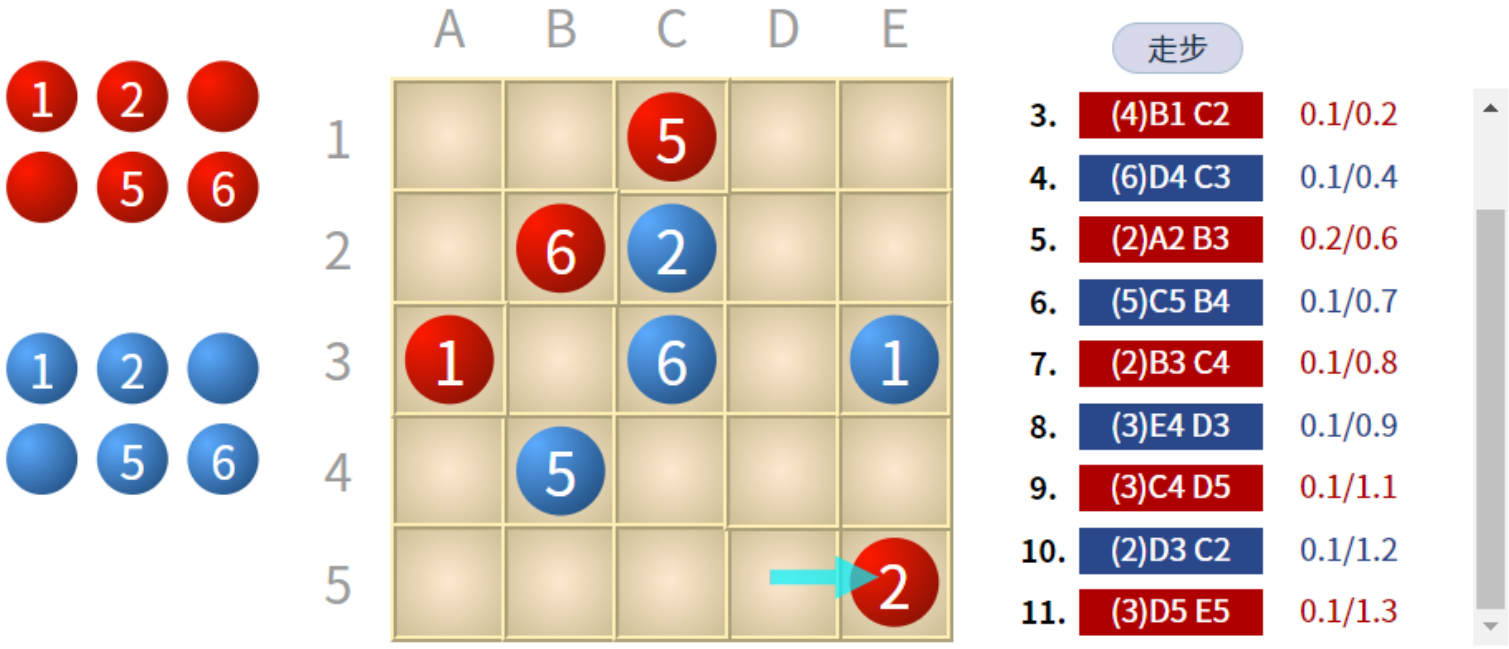


Figure 14: 遊戲進行中

Sample Code - Board

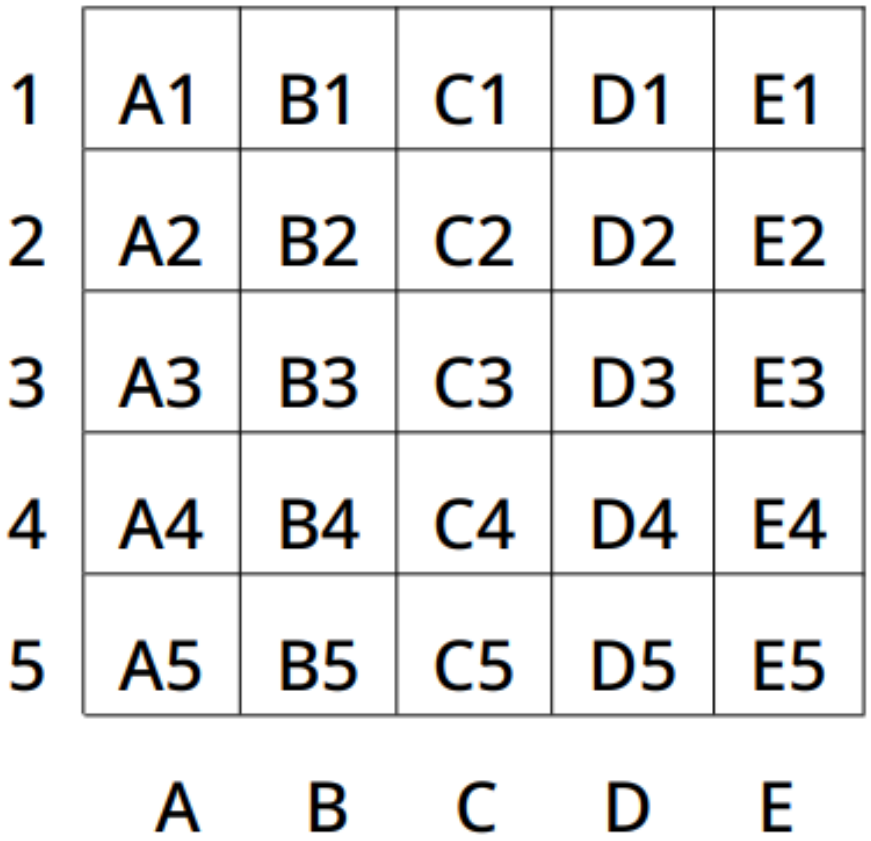


Figure 15: 棋盤位置代號

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。

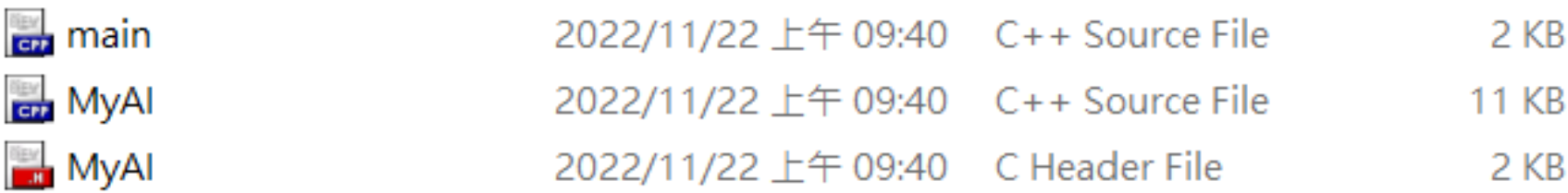


Figure 17: 資料夾結構

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
    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[
    else fprintf(stderr, "Red piece %d: (%c%c) -> (%c%c)\n", piece - PIECE_NUM, move[0], mov
    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

用來產生目前殷到的棋的合法走步，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

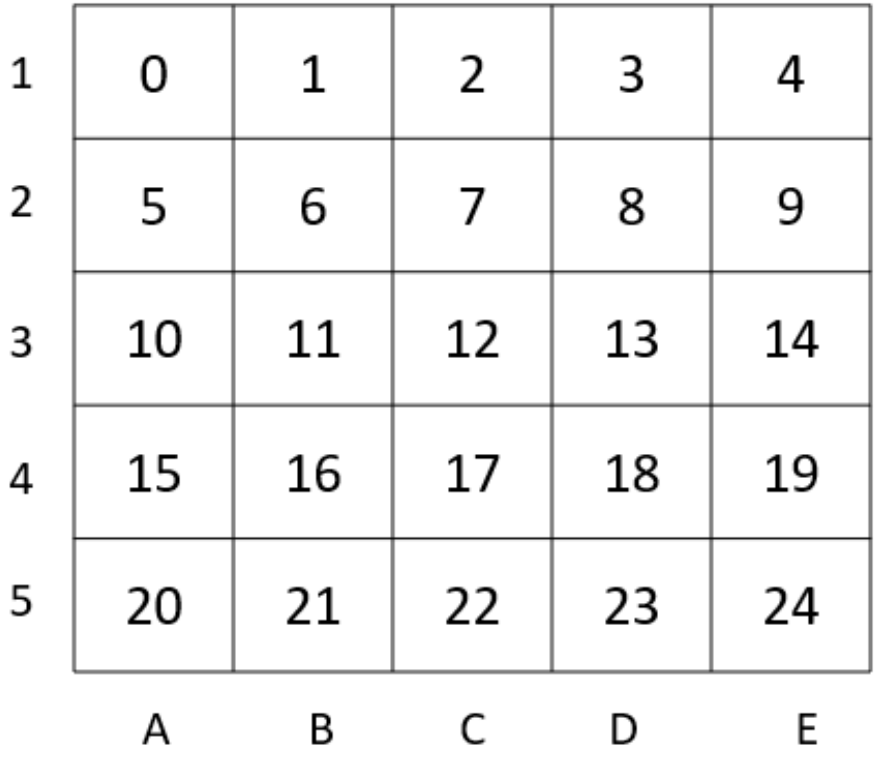


Figure 22: AI 程式產生走步時的盤面位置代號

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%c", 'A' + start_point % BOARD_SIZE, '1' + st
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 - P
197      this->Print_chessboard();
198      fprintf(stderr, "============================\n");
```

Figure 23: Make_move

Sample Code - Compile

輸入 cmd，按 enter 可開啟 cmd。

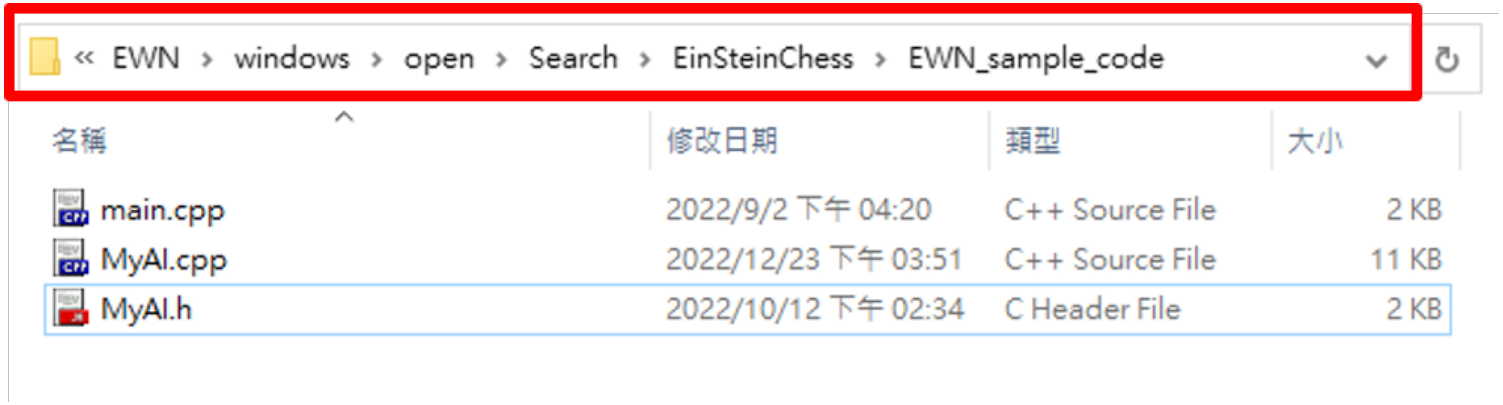


Figure 24: 開啟 cmd

Sample Code - Compile

確認路徑 (需與 MyAI.cpp 同路徑)。

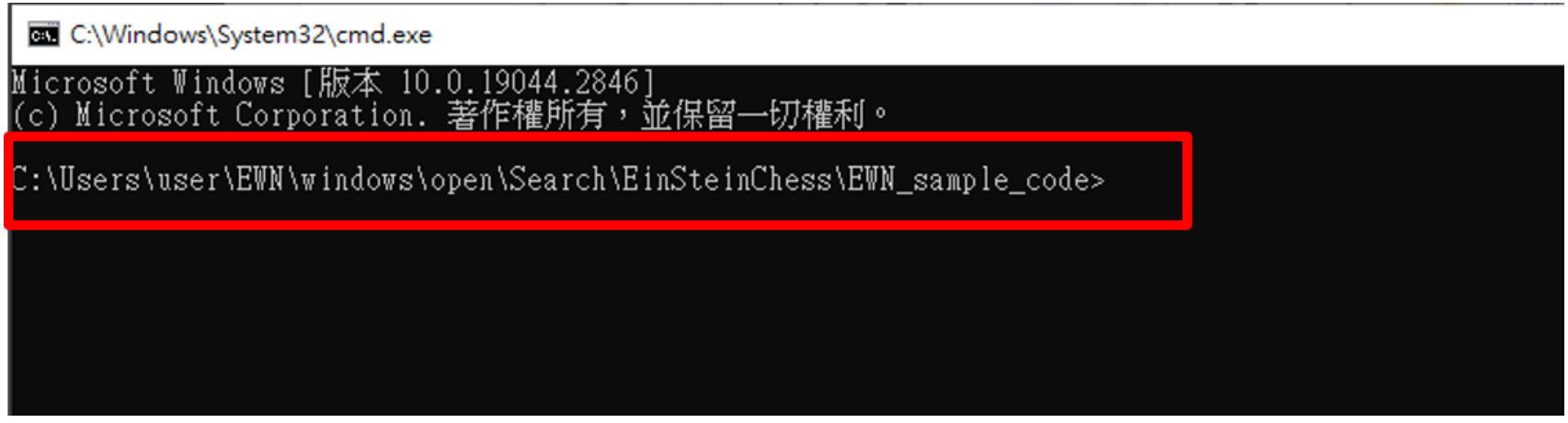
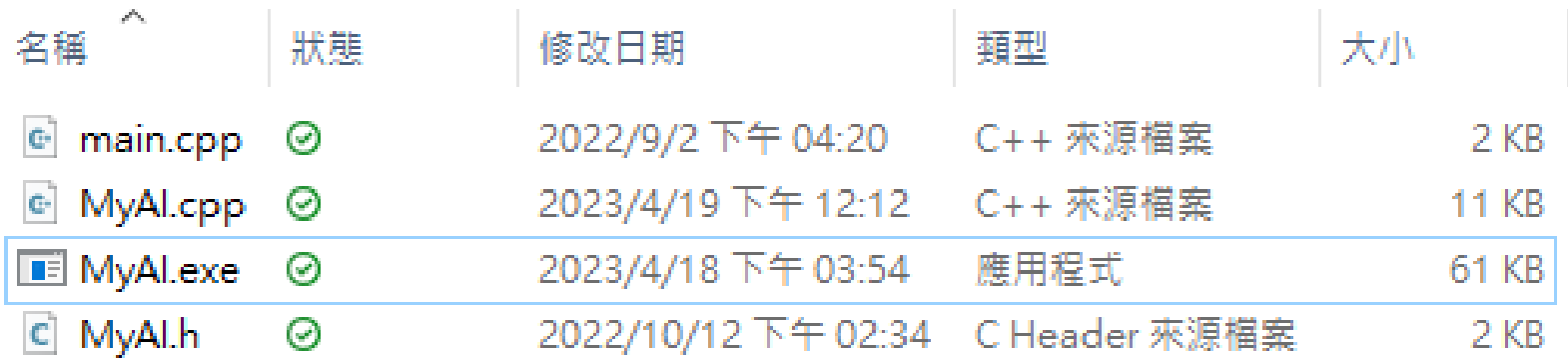


Figure 25: 確認路徑

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**