

暗棋對弈平台使用手冊

陳志昌、范綱宇、楊曜榮

一、設定

二、開房設定

三、加房設定

四、讀檔模式(Read Mode)

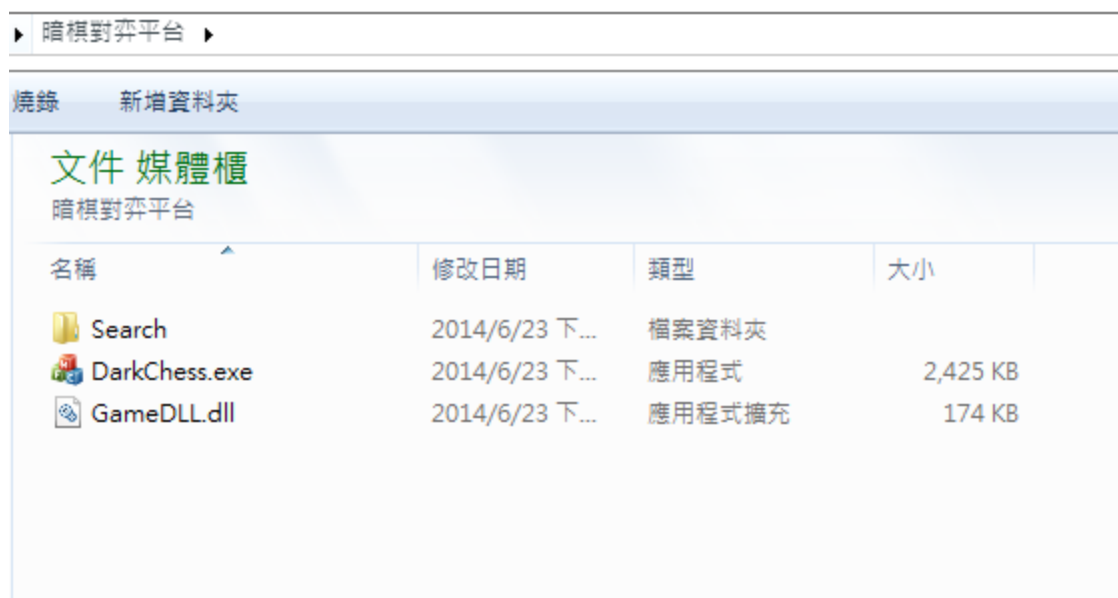
五、背景思考模式(Protocol Mode)

六、顯示介面

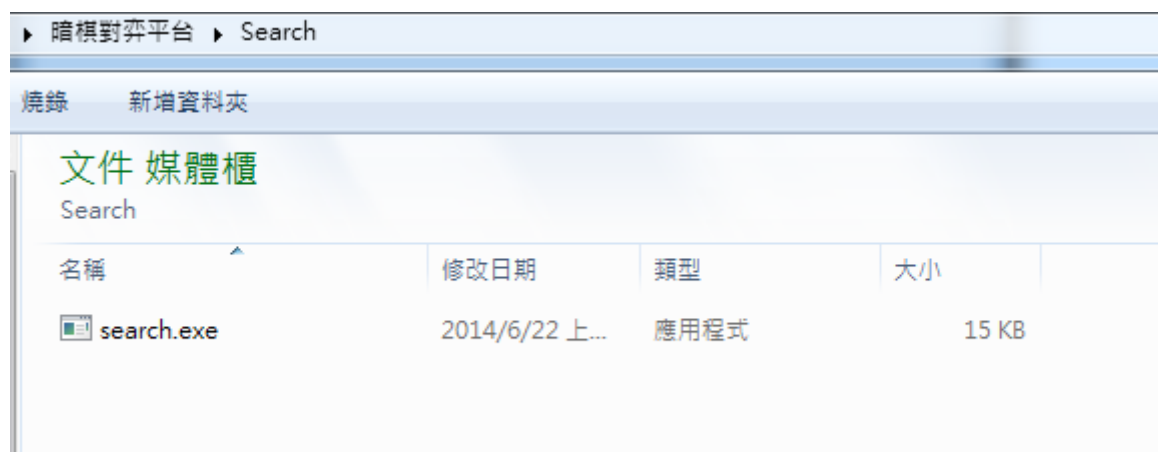
七、連絡資訊

一、 設定

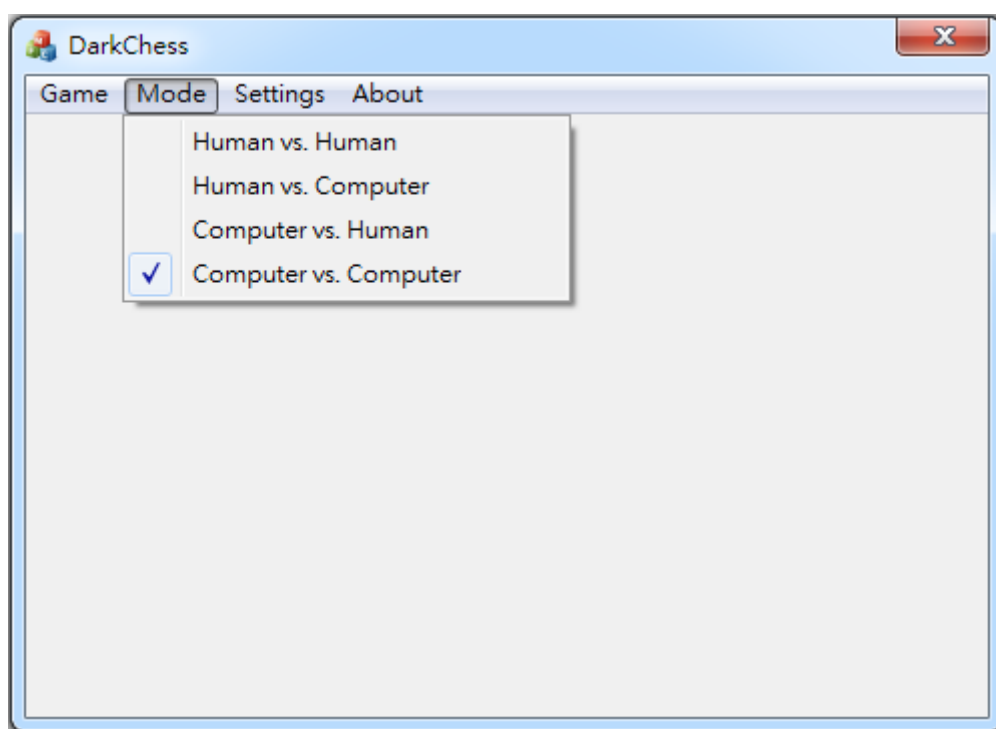
1.1 暗棋界面中，包含 DarkChess.exe 與 GameDLL.dll 及 Search 資料夾



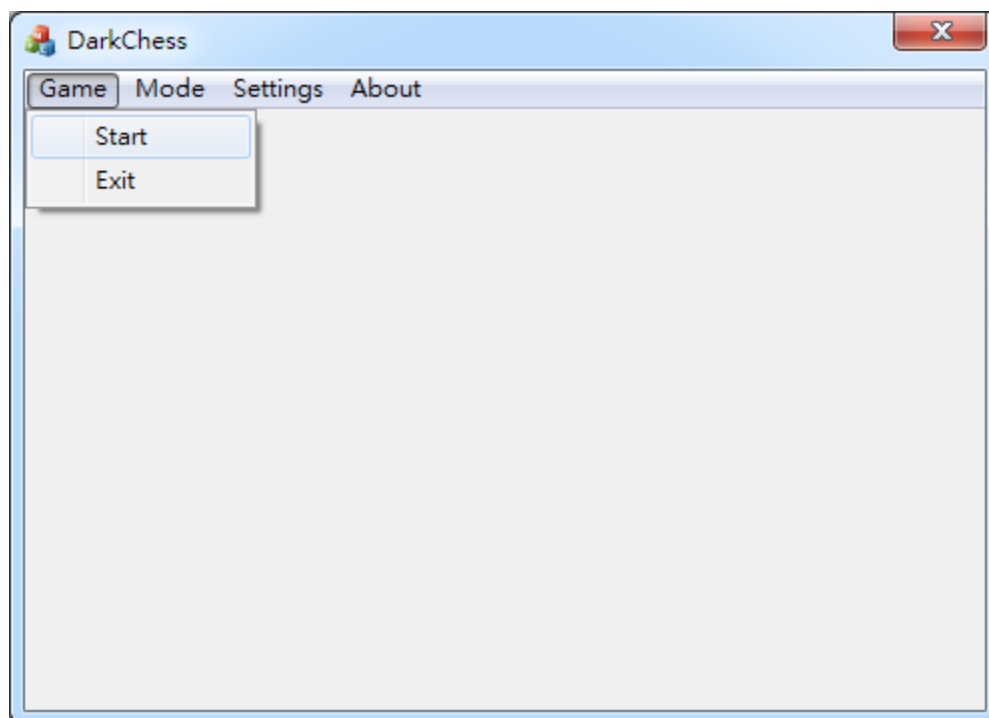
1.2 在 Search 資料夾中，置放 search.exe 的搜尋檔



1.3 開啟 DarkChess.exe 主程式，將 Mode 模式改為 Computer vs. Computer。



1.4 點選 Game 選單的 Start 開始遊戲



二、 開房設定

- 設定玩家資訊 (預設帳號：a0~a10000 密碼：123)



- 帳號：預設帳號 "a0"、"a1"至"a10000"
- 密碼：預設密碼 "123"
- 啟動模式：選擇"開房"
- 斷線中盤："是"，當發生斷線時，繼續上一局的斷線盤面
- 思考模式：search 與介面連接的模式
 - 背景：使用提供背景思考功能的 protocol，使得 search 程式透過 protocol 取得遊戲參數（背景思考模式(Protocol Mode)）
 - 讀檔：由介面開啟 search，讀取 board.txt 並輸出 move.txt （讀檔模式(Read Mode)）
- Debug 模式：顯示 command 視窗，方便輸出 log
- 自動重下：自動繼續新的遊戲
 - 重複次數：自動繼續遊戲的回合數
- 先後手："先"，player1 玩家，先走第一步

- 先後手互換：每節數一場，與對手互換先、後手
- 計時賽制：玩家的思考時間限制，超時做負
 - 回合制：一場遊戲中，玩家有固定的思考總時間，不限制每個 ply 的思考時間
 - 單手制：一場遊戲中，限制每個 ply 的思考時間
- 秒數限制：回合或單手制的時間限制
- 長捉次數：發生循環盤面的次數限制，若達循環次數的下一手為循環的第一步，雙方判和
ex: 循環盤面(長捉)3次 n連續循環步，判和。(n：任意循環步)

(a) 追棋：假設盤面只有兩顆棋子，b2 為仕，a1 為象，a2, a3, b1, b3 為空

[4 步循環] * 1. b2-b1 a1-a2
 * 2. b1-b2 a2-a1
 b2-b1, a1-a2, b1-b2, a2-a1 ，此四步算一次長捉。

[8 步循環] * 1. b2-b1 a1-a2
 * 2. b1-b2 a2-a3
 * 3. b2-b3 a3-a2
 * 4. b3-b2 a2-a1
 b2-b1, a1-a2, b1-b2, a2-a3, b2-b3, a3-a2, b3-b2, a2-a1 ，此八步算一次長捉。

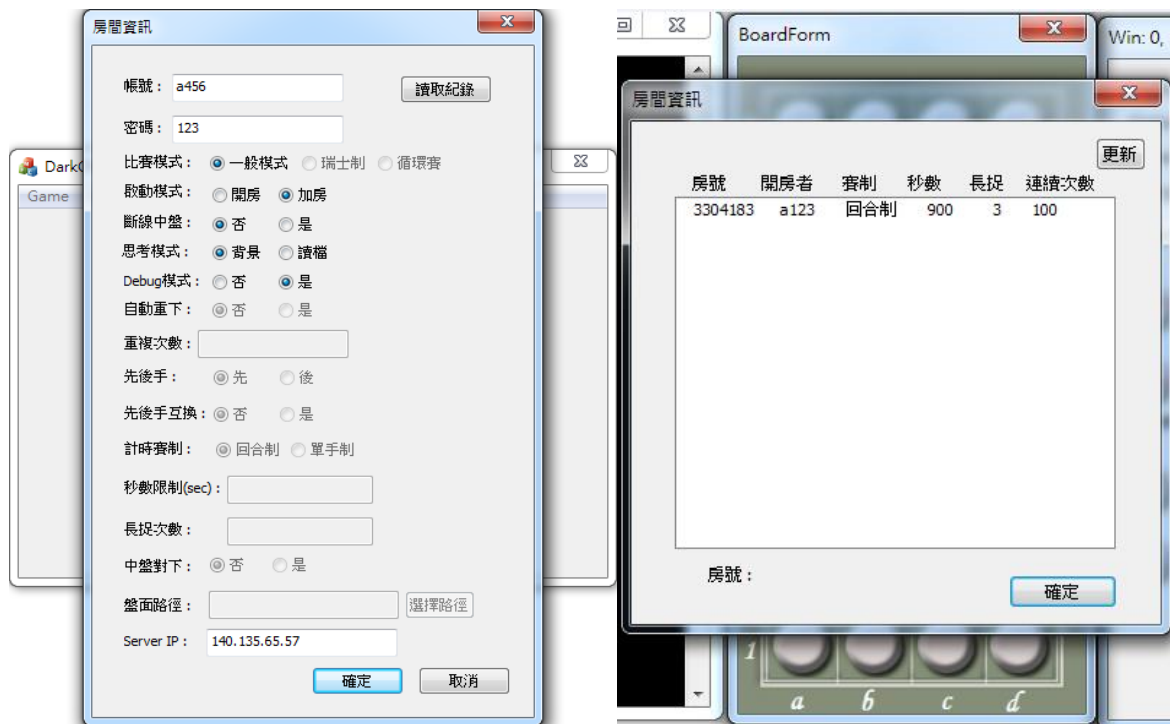
(b) 走閒步：假設盤面只有兩顆棋子，a1 為將，d7 為炮，a2 為空，d8 為空

* 1. a1-a2 d7-d8
 * 2. a2-a1 d8-d7
 a1-a2, d7-d8, a2-a1, d8-d7 ，此四步算一次長捉。

以上(a)、(b)，連續三次出現相同的四步，下一手為循環盤面的第一步，不論有沒有捉棋皆判和(其中若有吃子，則不算該步)。如：**a1-a2**, d7-d8, a2-a1, d8-d7, a1-a2, d7-d8, a2-a1 d8-d7, a1-a2 d7-d8, a2-a1, d8-d7, **a1-a2**

- 中盤對下：“是”，由盤面路徑選擇起始盤面檔案
- Server IP：預設 IP 為 140.135.65.57，比賽時，由裁判提供的 IP 為主

三、 加房設定



- 啟動模式：選擇”加房”，確定
- 房間資訊：選擇欲對奕的房號，開始遊戲

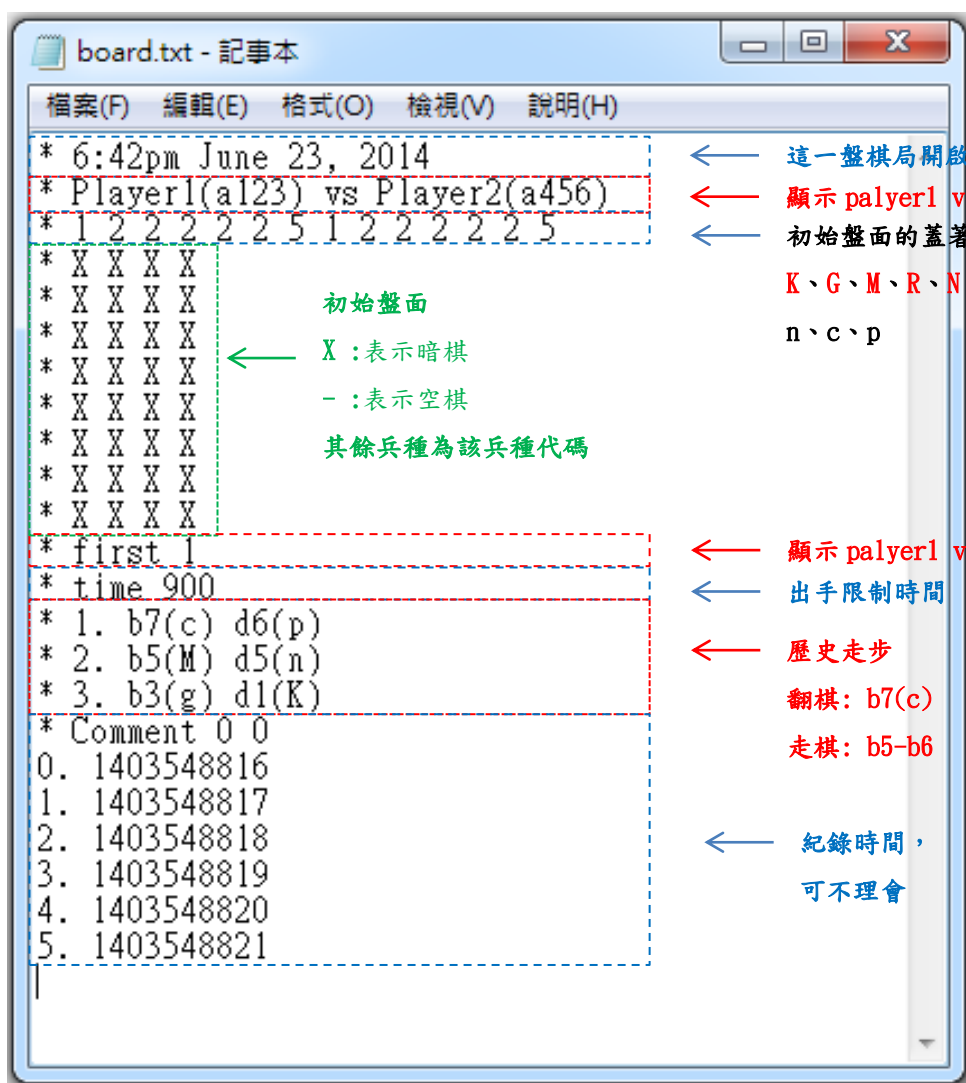
四、 讀檔模式(Read Mode)

4.1 棋子代碼：

紅：帥(K)、仕(G)、像(M)、碑(R)、碼(N)、炮(C)、兵(P)。

黑：將(k)、士(g)、象(m)、車(r)、馬(n)、包(c)、卒(p)。

4.2 board 檔：每一個 ply 的開始，將目前的棋局紀錄存於 Search\board.txt，使得 search.exe 讀取目前盤面，搜尋最佳步



The screenshot shows a Notepad window titled "board.txt - 記事本" with a menu bar (檔案(F), 編輯(E), 格式(O), 檢視(V), 說明(H)). The text content is as follows:

```
* 6:42pm June 23, 2014
* Player1(a123) vs Player2(a456)
* 1 2 2 2 2 2 5 1 2 2 2 2 2 5
* X X X X
* X X X X
* X X X X
* X X X X
* X X X X
* X X X X
* X X X X
* X X X X
* first 1
* time 900
* 1. b7(c) d6(p)
* 2. b5(M) d5(n)
* 3. b3(g) d1(K)
* Comment 0 0
0. 1403548816
1. 1403548817
2. 1403548818
3. 1403548819
4. 1403548820
5. 1403548821
```

Annotations on the right side of the image:

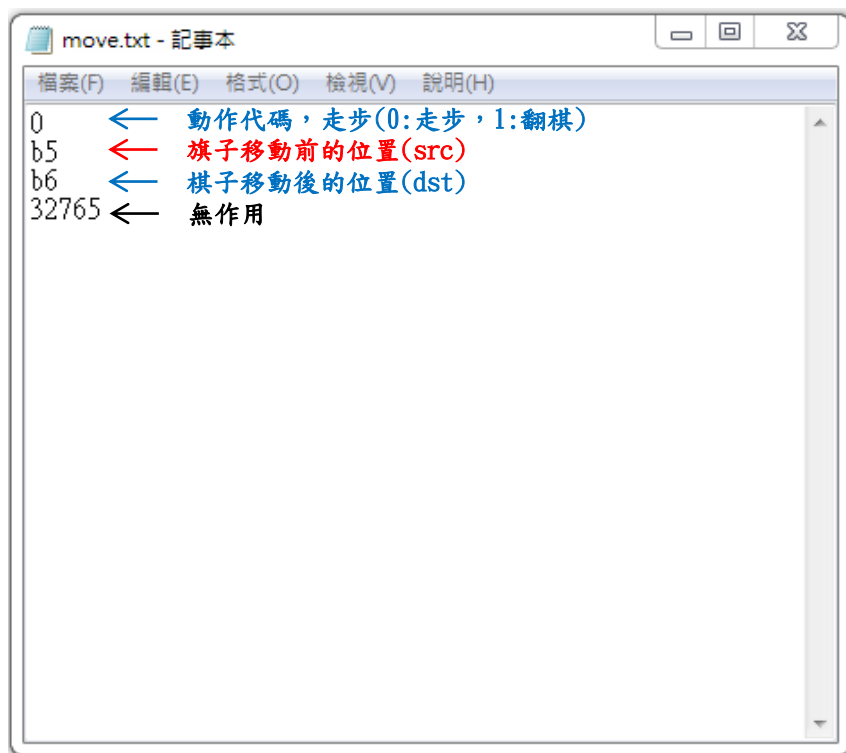
- ← 這一盤棋局開盤的時間
- ← 顯示 palyer1 vs player 2
- ← 初始盤面的蓋著的棋子個數，分別為 K、G、M、R、N、C、P、k、g、m、r、n、c、p
- ← 顯示 palyer1 vs player 2
- ← 出手限制時間
- ← 歷史走步
- 翻棋: b7(c) 為在 b7 的位置翻出包
- 走棋: b5-b6 為 b5 的棋子移動至 b6
- ← 紀錄時間，可不理會

Annotations on the left side of the image:

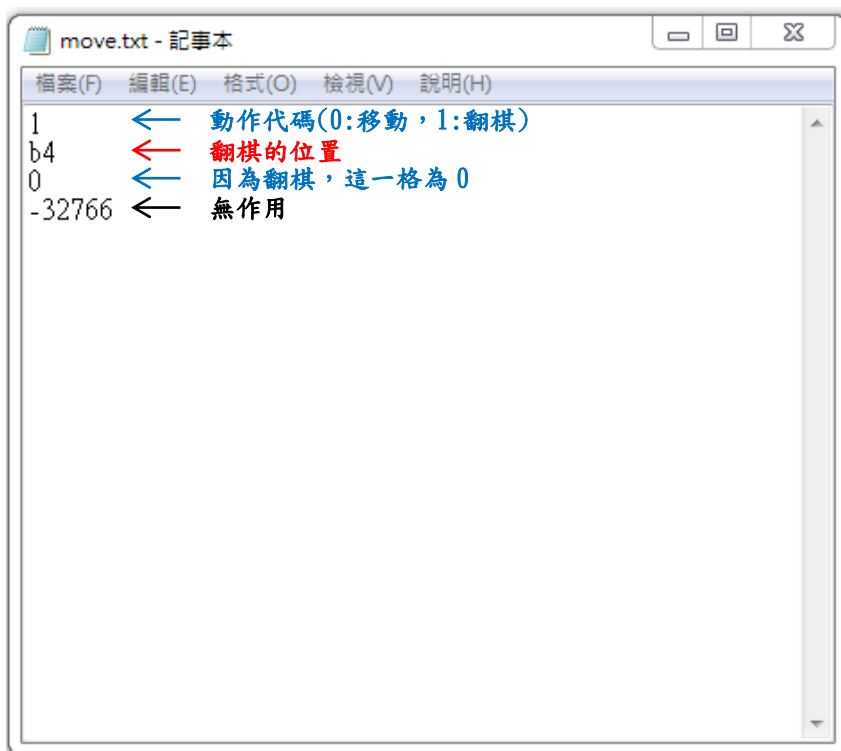
- ← 初始盤面
- ← X :表示暗棋
- ← - :表示空棋
- ← 其餘兵種為該兵種代碼

4.3 move 檔：每一個 ply 的最佳走步，當 search.exe 搜尋完畢，將最佳走步輸出至 Search\move.txt

■ 走步



■ 翻棋



五、 背景思考模式(Protocol Mode)

此模式提供對局程式在連線時背景思考的功能，而第四章讀檔模式則不具備背景思考的功能。CDC_client.zip 壓縮檔中，包含

- ClientSocket.h, ClientSocket.cpp, Protocol.h and Protocol.cpp (**the client protocol**)
- myai.h, myai.cpp (**the random demo program**)
- main.cpp, main_clear.cpp (**to connect myai and the client**)

當你的程式要連接 Protocol Mode 時，必須使用 ClientSocket.h、ClientSocket.cpp、Protocol.h 與 Protocol.cpp，而流程部分僅需修改 main_clear.cpp 中，出現“// todo”的程式碼片段。

Protocol 範例程式包含 main.cpp、myai.h、myai.cpp。假設使用 g++，則編譯指令為：

```
g++ -o search.exe main.cpp Protocol.cpp ClientSocket.cpp myai.cpp
```

假若你的程式為 **YourAI.cpp**，編譯指令為：

```
g++ -o search.exe main.cpp Protocol.cpp ClientSocket.cpp YourAI.cpp
```

編譯完成後，將 search.exe 取代 Search 目錄中的 search 檔，也就是 Search\\search.exe 檔。遊戲設定時，將**思考模式**選為“**背景**”（詳細說明在第二章-開房設定）

以下為連接Protocol的class定義與function使用說明

```
enum PROTO_CLR {PCLR_RED, PCLR_BLACK, PCLR_UNKNOWN};  
class Protocol  
{  
public:  
    Protocol();  
    ~Protocol();  
    void init_protocol(const char *ip, const int port);  
    void init_board(int piece_count[14], char current_position[32], struct History &history,int &time);  
    void get_turn(bool &turn, PROTO_CLR &color);  
    void send(const char src[3], const char dst[3]);  
    void send(const char move[6])  
    void recv(char move[6], int &time);
```

```
PROTO_CLR get_color(const char move[6]);  
};
```

5.1 初始化 Protocol

```
void init_protocol(const char *ip, const int port);
```

GUI 介面透過參數傳遞 ip 和 port 到此來連接到 server 取的初始資訊
在一場比賽開始時 Init_protocol 必須被呼叫。

```
#include "protocol.h"  
int main(int argc, char **argv)  
{  
    Protocol protocol;  
    switch (argc) {  
    case 3:  
        if (!protocol.init_protocol(argv[1], atoi(argv[2]))) return 0;  
        break;  
    }  
    ...  
    return 0;  
}
```

5.2 格式定義

兵種：

- 字元 'K', 'G', 'M', 'R', 'N', 'C', 'P' 分別表示為紅方的帥、仕、相、俥、傴、炮、兵。
- 字元 'k', 'g', 'm', 'r', 'n', 'c', 'p' 分別表示為黑方的、將、士、象、車、馬、包、卒。
- 字元 'X' 表示為蓋著的棋子（暗子）。
- 字元 '-' 表示為空的位置。

棋盤 (Fig1):

- 橫軸字母從左到右為 a 到 c
- 縱軸數字從下到上為 1 到 8

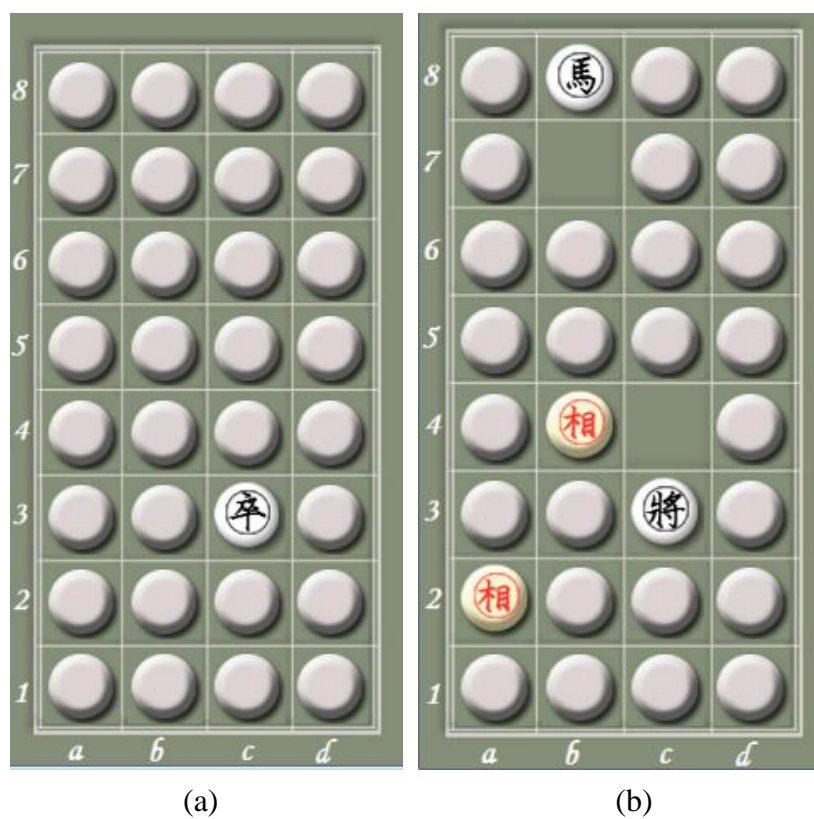


Fig1：兵種和棋盤

Ex：

在 Fig1(a), 卒(表示為 p)位於 c3, 並且有一暗子在 b4(表示為 X)。

在 Fig1(b), 帥(表示為 K)位於 d3, 並且有一空子位於 a4(表示為 -)。

5.3 struct History

```
struct History{  
char** move;  
int number_of_moves;  
};
```

定義 move 和 number_of_move

- **move:**

走子或吃子

Ex：在第二手時將棋子從a3移動到a4, 則move[2] = "a3-a4"

翻子

Ex：在第二手時翻開位置c2為帥(K), 則move[2] = "c2(K)"

- **number_of_moves:**

總共下了幾手

Ex：若 number_of_moves = 3, 則我們可以使用 move[0]、move[1]和move[2]

如果你需要讀取歷史走步,你可以在範例程式碼中如下程式區塊內實作(在TODO的區塊)

```
struct History history;  
protocol->init_board(piece_count, current_position, &history);  
for (int i = 0; i < history.number_of_moves; i++) {  
    // TODO: restore the history to your program.  
}
```

5.4 init board

```
void init_board(int piece_count[14], char current_position[32], struct History history, int &time);
```

當你呼叫**init_board**時你會取以下這些棋局資訊：

- **piece_count[14]**：14個兵種存活的個數。
- **current_position[32]**：棋盤上各個位置的兵種(格式定義為在第5.2章)。
- **history**：歷史走步。
- **time**：我方剩餘思考時間（單位：millisecond）。

Ex：在 Fig2, 在棋盤中存在**相**、**相**、**馬**、**將**, 而有一隻**兵**與一隻**砲**已陣亡。

piec_count[14]和**current_position[32]**的值表示如下：

```
piece_count[14] = {1, 2, 2, 2, 2, 2, 4, 1, 2, 2, 2, 2, 1, 5}
```

```
current_position[32] = "XnXXX-XXXXXXXXXXXXM-XXXkXMXXXXXXXX"
```

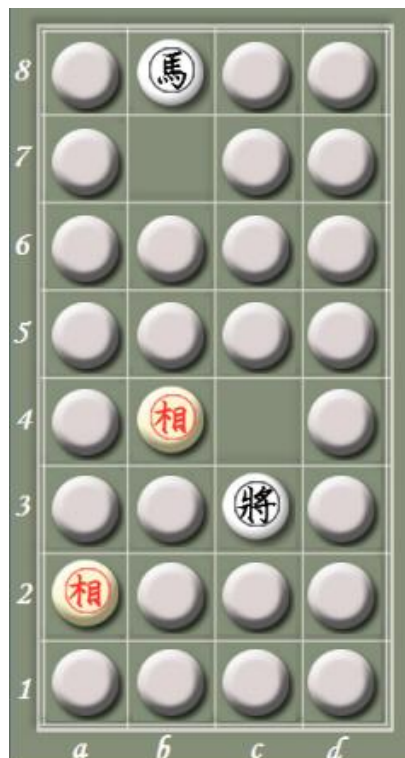


Fig2：current_position[32].

5.5 get turn

```
enum PROTO_CLR {PCLR_RED, PCLR_BLACK, PCLR_UNKNOWN};  
void get_turn(bool &turn, PROTO_CLR &color);
```

turn

true 表示為先手
false 表示為後手。

color

PCLR_RED 表示為紅子方。
PCLR_BLACK 表示為黑子方。
PCLR_UNKNOWN 表示為未知。

當你從初始盤面開始比賽時,你取得的顏色是 **PCLR_UNKNOWN**(需要透過第一手翻子來決定你的顏色)。若是從中盤對下,則你取得的 color 為 **PCLR_RED** 或 **PCLR_BLACK**。

5.6 send

從以下兩個函式中,選擇一個來傳送你所計算的最佳著手。

```
void send(const char src[3], const char dst[3]);
```

走子或吃子

Ex: 從 d5 移動到 c5, 則 src = "d5" and dst = "c5".

翻子

Ex: 翻開 d5 的棋子,則 src = "d5" and dst = "c5". (src = dst)

```
void send(const char move[6]);
```

走子或吃子

Ex: 從 d5 移動到 c5,則 move = "d5-c5"

翻子

Ex：翻開 d5 的棋子,則 move = "d5-d5"

5.7 recv

```
void recv(char move[6], int &time);
```

move是經由Server傳送過來的對手走步。

走子或吃子

Ex：從 a3 移動到 a4,則 move = "a3-a4"

翻子

Ex：翻開帥位於 c2,則 move = "c2(**K**) "

time是目前我方的剩餘思考時間（單位：millisecond）。

5.8 get_color

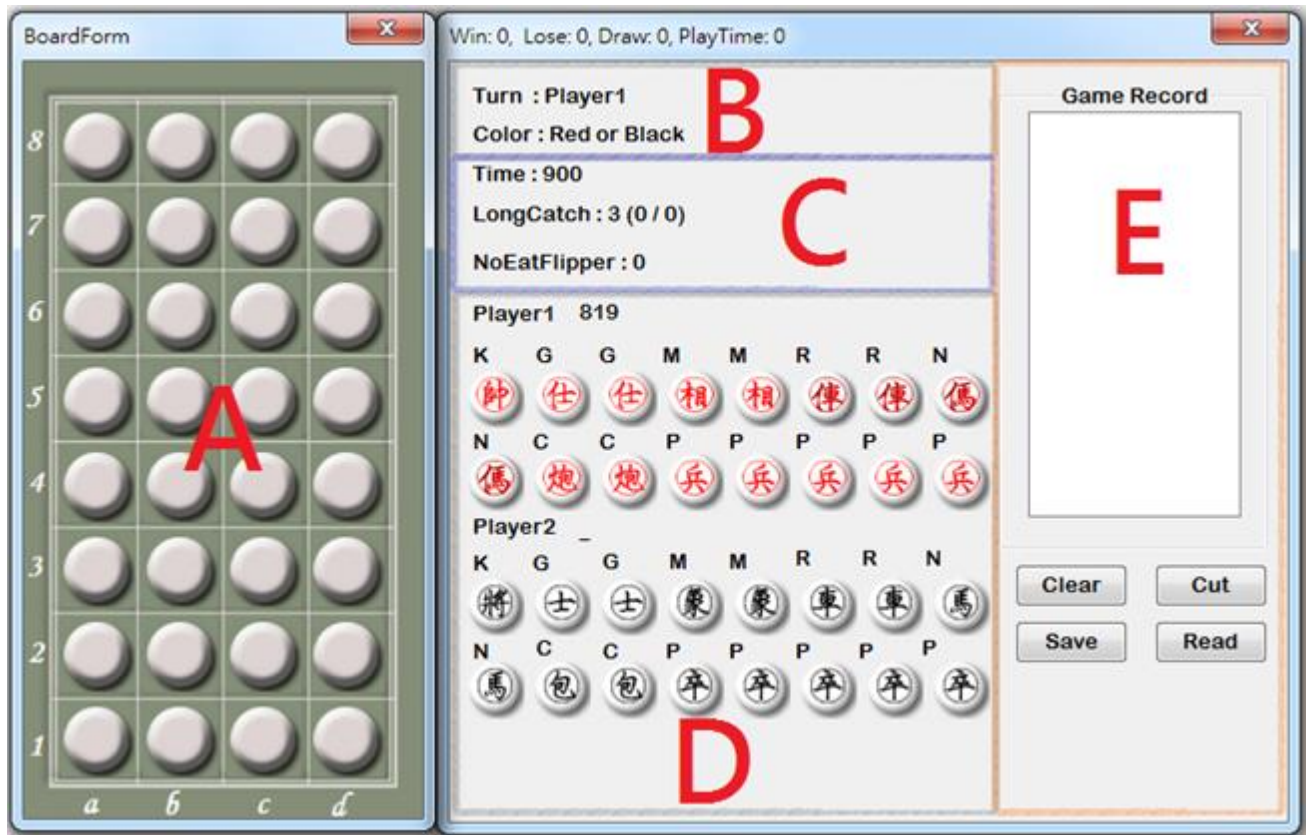
```
enum PROTO_CLR {PCLR_RED, PCLR_BLACK, PCLR_UNKNOWN};  
PROTO_CLR get_color(const char move[6]);
```

此函式回傳棋子的顏色。

Ex：

```
PROTO_CLR color;  
char move[6] = "a8(G)";  
color = get_color(move);    /* color == PCLR_RED */  
move[6] = "d6(p)";  
color = get_color(move);    /* color == PCLR_BLACK */
```

六、顯示介面



A. 棋盤

顯示目前盤面狀態的功能。

B. 輪手、顏色資訊

Turn：顯示目前輪到誰思考、出步。

Color：目前 Turn 的顏色，初始是 Red or Black，第一手過後，即可知道雙方顏色。

C. 棋規

Time：對弈時，可使用的固定時間，為 900 秒。

LongCatch：長捉次數，當發生長捉、循環盤面 3 次則和棋。

(A / B)：A 為長捉、循環盤面次數

B 為幾步循環盤面

NoEatFilpper：無吃無翻步數。

D. 剩餘秒數與盤面狀態

剩餘秒數：Player1 與 Player2 右方的數字，表示該方的剩餘思考秒數。

盤面狀態：Player1 與 Player2 下方的兵種狀態，顯示的兵種表示於目前盤面上存活著，若是已經陣亡的兵種，則顯示暗棋。

E. 棋步資訊

Game Record：記錄目前盤面以前的走步資訊。

紀錄勝負：右方頁面的標題記錄著目前對弈的勝（Win）、負（lose）、和（draw）。

Clear 按鈕：清除盤面，重新開始。

Cut 按鈕：可以任意裁減 Game Record 的走步，回溯到裁減的步數，方便 Debug。

Save 按鈕：儲存目前的盤面到檔案中，預設檔案相對路徑是 Search\Saveboard.txt。

Read 按鈕：讀取檔案相對路徑 Search\Saveboard.txt 的紀錄檔。

七、 連絡資訊

若使用手冊有任何未能詳盡的敘述，請與我們連絡：

- 陳志昌 jcchen@cycu.edu.tw
- 范綱宇 imloed10000@gmail.com
- 楊曜榮 kevin12345621@gmail.com

暗棋棋規與代號，參考自以下論文：

- Chen, B.N., Shen, B.J., and Hsu, T.s., "Chinese Dark Chess," *ICGA Journal*, vol. 33, no. 2, pp. 93-106, 2010.
- Chen, J.C., Lin, T.Y., Hsu, T.s., "Equivalence Classes in Dark Chess Endgames," accepted by *IEEE Transactions on Computational Intelligence and AI in Games (IEEE TCIAIG)* (DOI: 10.1109/TCIAIG.2014.2317832).
- Yen, S.J, Chou, C.W., Chen, J.C., Wu, I.C., Kao, K.Y., "Design and Implementation of Chinese Dark Chess Programs," accepted by *IEEE TCIAIG* (DOI: 10.1109/TCIAIG.2014.2329034).