Final Project Description

December 22, 2016
Update January 05, 2017

Announcement and Important Data

- Competition date: 2017/01/19 09:30-16:30 at CSIE 204.
- Rules:
  http://www.iis.sinica.edu.tw/~tshsu/tcg/2016/hwks/rules.pdf
- Submit page: http://w.csie.org/~tcg/2016/final.php
  - Submission due data: 2017/01/20 09:00
- Download link for final project files:
  - http://w.csie.org/~tcg/tcg_2016_final_project.zip
  - Check your school e-mail address for password.

Basic Requriement

- Your program **must**
  - cotains the following two features:
    - NegaScout
    - Hash table
    - TA will trace your code.
  - has the ability to complete a game normally.
- You **must** provide a **makefile** for linux or
  - handing a Windows version execution file in demo day.
  - specify how to compile your program in the report
- Your zip file should be formated as:
  - A folder named by your ID
    - A folder named by code (Contains all the code and makefile)
    - report.pdf

Grading Policy

- Final score = Coding score + Document score + Bonus
  - Coding score + Document score is at most 40.
  - Bonus score depends on the tournament performance.
    - Bonus does not include in the 40% final project score
- Warning:
  - In the following situation, your score will be very low:
    1. You claim something you have done but you didn't.
    2. Your program can not be compiled.
    3. Your program can not be executed.
    4. Your program can not finish a game normally.
    5. You didn't do anything and just upload the template code.

Brief Rules for Final Project

- 瑞士制
  - 抽籤決定第一輪對手。（請提前到場抽籤）
  - 預估六至八輪。
  - 一個對戰組合互先一次。
    - 勝一局得2分
    - 平一局得1分
    - 輸一局得0分
    - e.g. 6W3D3L $= 6 \times 2 + 3 \times 1 + 3 \times 0 = 15$分
- 單場限時 900秒，超時算敗。
- 三循環算和
- 30步無吃無翻算和。
- 單局犯規(crash)兩次算輸
- 犯規時由裁判決定棋局繼續進行方式。
- 裁判擁有所有棋局最終判決權。

瑞士制範例: Round 1

ID\Round	1		2		3		對手分	Rank
1	1 v.s. 2 2W	4						
2	2 v.s. 1 2L	0						
3	3 v.s. 4 1W1L	2						
4	4 v.s. 3 1W1L	2						
5	5 v.s. 6 2D	2						
6	6 v.s. 5 2D	2						
7	7 v.s. 8 2W	4						
8	8 v.s. 7 2L	0						

瑞士制範例: Round 2

ID\Round	1		2		3		對手分	Rank
1	1 v.s. 2 2W	4	1 v.s. 7 1W1L	6				
2	2 v.s. 1 2L	0	2 v.s. 8 2D	2				
3	3 v.s. 4 1W1L	2	3 v.s. 5 2D	4				
4	4 v.s. 3 1W1L	2	4 v.s. 6 1W1D	5				
5	5 v.s. 6 2D	2	5 v.s. 3 2D	4				
6	6 v.s. 5 2D	2	6 v.s. 4 1D1L	3				
7	7 v.s. 8 2W	4	7 v.s. 1 1W1L	6				
8	8 v.s. 7 2L	0	8 v.s. 2 2D	2				

瑞士制範例: Round 3

ID\Round	1		2		3		對手分	Rank
1	1 v.s. 2 2W	4	1 v.s. 7 1W1L	6	1 v.s. 4 2D	8		
2	2 v.s. 1 2L	0	2 v.s. 8 2D	2	2 v.s. 6 1W1D	5		
3	3 v.s. 4 1W1L	2	3 v.s. 5 2D	4	3 v.s. 7 2L	4		
4	4 v.s. 3 1W1L	2	4 v.s. 6 1W1D	5	4 v.s. 1 2D	7		
5	5 v.s. 6 2D	2	5 v.s. 3 2D	4	5 v.s. 8 2W	8		
6	6 v.s. 5 2D	2	6 v.s. 4 1D1L	3	6 v.s. 2 1D1L	4		
7	7 v.s. 8 2W	4	7 v.s. 1 1W1L	6	7 v.s. 3 2W	10		
8	8 v.s. 7 2L	0	8 v.s. 2 2D	2	8 v.s. 5 2L	2		

瑞士制範例: Final Ranking

ID\Round	1		2		3		對手分	Rank
1	1 v.s. 2 2W	4	1 v.s. 7 1W1L	6	1 v.s. 4 2D	8	22	2
2	2 v.s. 1 2L	0	2 v.s. 8 2D	2	2 v.s. 6 1W1D	5	14	5
3	3 v.s. 4 1W1L	2	3 v.s. 5 2D	4	3 v.s. 7 2L	4	25	6
4	4 v.s. 3 1W1L	2	4 v.s. 6 1W1D	5	4 v.s. 1 2D	7	16	4
5	5 v.s. 6 2D	2	5 v.s. 3 2D	4	5 v.s. 8 2W	8	10	3
6	6 v.s. 5 2D	2	6 v.s. 4 1D1L	3	6 v.s. 2 1D1L	4	20	7
7	7 v.s. 8 2W	4	7 v.s. 1 1W1L	6	7 v.s. 3 2W	10	14	1
8	8 v.s. 7 2L	0	8 v.s. 2 2D	2	8 v.s. 5 2L	2	23	8

Files and Description

- Template code folder
  - final_project_template (Provide by TA, used in class only)
- GUI interface folder:
  - CDC_package_win7_3.0
    - 暗棋對弈平台_使用手册_win7.pdf
  - CDC_package_linux_3.0
    - 暗棋對弈平台_使用手册_ubuntu.pdf

GUI Mode and Protocal Type

- 連線方法:
  - 單機模式: 透過 Board.txt, Move.txt
  - 連線模式: 透過 Socket
- Play mode:
  - Human v.s. Human
  - Human v.s. Computer (Computer play as second player)
  - Computer v.s. Human (Computer play as first player)
  - Computer v.s. Computer
- 帳號/密碼: 大寫學號

Template Code

- final_project_template
  - main.cc
  - anqi.cc
  - anqi.h
  - Protocol.h
  - Protocol.cpp
  - ClientSocket.h
  - ClientSocket.cpp
- You only need to modify codes in main.cc, anqi.cc and anqi.h
- Need -static -s when compiling in all system
- Need -lwsock32 when compiling in windows system
  - Can use -D _WINDOWS to enable extra WINDOWS only feature.

High Level Flow Chart of Protocol: 單機模式

- Initial the game state according to board.txt
- Generate a move
- Write the move to move.txt

Main Function 讀檔模式 main.cc:159–164

```
159:if (argc!=3) {
160:    BOARD B;
161:    TimeOut=(B.LoadGame("board.txt")-3)*1000;
162:    if(!B.ChkLose()) Output(Play(B))
163:    return 0;
164:}
```

- BOARD B
  - struct BOARD, see anqi.hh:58–75
- TimeOUT
  - number of thinking time in ms.
- Play(B)
  - Generate the best move accroding to B, see main.cc:88–110
- Output(MOV)
  - Output MOV to "move.txt", see anqi.cc:67–82

High Level Flow Chart of Protocol: 連線模式

- Initial the socket connection
- Initial the game state through the socket
- While(true)
  - If it's your turn
    - Generate a move and send it through the socket
    - Update the local game state
    - Receive the message from the server
    - Change the turn
  - If it's opponent's turn
    - Wait for opponents' move
    - Receiving Opponent's move and update the game state.
    - Change the turn

Main function: 連線模式: main.cc:165–183

- main.cc:

```
165:Protocol *protocol;
166:protocol = new Protocol();
167:protocol->init_protocol(argv[1],atoi(argv[2]));
168:int iPieceCount[14];
169:char iCurrentPosition[32];
170:int type, remain_time;
171:bool turn;
172:PROTO_CLR color;
174:char src[3], dst[3], mov[6];
175:History moveRecord;
176:protocol->init_board(iPieceCount, iCurrentPosition, moveRecord, remain_time);
177:protocol->get_turn(turn,color);
179:TimeOut = (DEFAULTTIME-3)*1000;
181:B.Init(iCurrentPosition, iPieceCount, (color==2)?(-1):(int)color);
183:MOV m;
```

Main function: 連線模式: main.cc:184–215

```
184:if(turn)
185:{
186:    m = Play(B);
187:    sprintf(src, "%c%c",(m.st%4)+'a', m.st/4+'1');
188:    sprintf(dst, "%c%c",(m.ed%4)+'a', m.ed/4+'1');
189:    protocol->send(src, dst);
190:    protocol->recv(mov, remain_time);
191:    if( color == 2)
192:    color = protocol->get_color(mov);
193:    B.who = color;
194:    B.DoMove(m, chess2fin(mov[3]));
195:    protocol->recv(mov, remain_time);
196:    m.st = mov[0] - 'a' + (mov[1] - '1')*4;
197:    m.ed = (mov[2]=='(')?m.st:(mov[3] - 'a' + (mov[4] - '1')*4);
198:    B.DoMove(m, chess2fin(mov[3]));
199:}
200:else
201:{
202:    protocol->recv(mov, remain_time);
203:    if( color == 2)
204:    {
205:        color = protocol->get_color(mov);
206:        B.who = color;
207:    }
208:    else {
209:        B.who = color;
210:        B.who^=1;
211:    }
212:    m.st = mov[0] - 'a' + (mov[1] - '1')*4;
213:    m.ed = (mov[2]=='(')?m.st:(mov[3] - 'a' + (mov[4] - '1')*4);
214:    B.DoMove(m, chess2fin(mov[3]));
215:}
```

Main function: 連線模式: main.cc:217–234

```
217:while(1)
218:{
219:    m = Play(B);
220:    sprintf(src, "%c%c",(m.st%4)+'a', m.st/4+'1');
221:    sprintf(dst, "%c%c",(m.ed%4)+'a', m.ed/4+'1');
222:    protocol->send(src, dst);
223:    protocol->recv(mov, remain_time);
224:    m.st = mov[0] - 'a' + (mov[1] - '1')*4;
225:    m.ed = (mov[2]=='(')?m.st    (mov[3] - 'a' + (mov[4] - '1')*4);
226:    B.DoMove(m, chess2fin(mov[3]));
227:    B.Display();
228:
229:    protocol->recv(mov, remain_time);
230:    m.st = mov[0] - 'a' + (mov[1] - '1')*4;
231:    m.ed = (mov[2]=='(')?m.st    (mov[3] - 'a' + (mov[4] - '1')*4);
232:    B.DoMove(m, chess2fin(mov[3]));
233:    B.Display();
234:}
```

Function Play main.cc:88–110

```
MOV Play(const BOARD &B) {
POS p; int c=0;
// 新遊戲？隨機翻子
if(B.who==-1){p=rand()%32;return MOV(p,p);}
// 若搜出來的結果會比現在好就用搜出來的走法
if(SearchMax(B,0,2)>Eval(B))return BestMove;
// 否則隨便翻一個地方 但小心可能已經沒地方翻了
for(p=0;p<32;p++)if(B.fin[p]==FIN_X)c++;
if(c==0)return BestMove;
c=rand()%c;
for(p=0;p<32;p++)if(B.fin[p]==FIN_X&&--c<0)break;
return MOV(p,p);
}
```

Function SearchMax main.cc:56–70

```
56:SCORE SearchMax(const BOARD &B,int dep,int cut) {
57:     if(B.ChkLose())return -WIN;
58:
59:     MOVLST lst;
60:     if(cut==0||TimesUp()||B.MoveGen(lst)==0)return +Eval(B);
61:
62:         SCORE ret=-INF;
63:     for(int i=0;i<lst.num;i++){
64:         BOARD N(B);
65:         N.Move(lst.mov[i]);
66:         const SCORE tmp=SearchMin(N,dep+1,cut-1);
67:         if(tmp>ret){ret=tmp;if(dep==0)BestMove=lst.mov[i];}
68:     }
69:     return ret;
70:}
```

Board Index

Template-Code

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
24	25	26	27
28	29	30	31

Protocal

28	29	30	31
24	25	26	27
20	21	22	23
16	17	18	19
12	13	14	15
8	9	10	11
4	5	6	7
0	1	2	3

Protocal-(x,y)

(0,7)	(1,7)	(2,7)	(3,7)
(0,6)	(1,6)	(2,6)	(3,6)
(0,5)	(1,5)	(2,5)	(3,5)
(0,4)	(1,4)	(2,4)	(3,4)
(0,3)	(1,3)	(2,3)	(3,3)
(0,2)	(1,2)	(2,2)	(3,2)
(0,1)	(1,1)	(2,1)	(3,1)
(0,0)	(1,1)	(2,0)	(3,0)

Frequently Asked Questions

- For MAC:
  - Q: Is there a GUI version for MAC?
  - A: Currently there is no MAC version.
- For Linux
  - Q: What does permission denied means?
  - A: Your "search" file needs to be executable.
  - Q: Why my GUI interface does not work?
  - A: Make sure to add the LD_LIBRARY_PATH=. to include the GameDLL.so
- For Windows
  - Q: When using "背景" mod, the GUI just hanging there?
  - A: In some combinations of Windows OS and compiler, the number arguments passing by the commnd line is not implemented as the stander. Check the value of argc, if it's not 3, then in Line 167 of main.cc
    protocol→init_protocol(argv[1],atoi(argv[2]));
    should be changed to
    protocol→init_protocol(argv[0],atoi(argv[1]));

Infrequently Asked Qustions

- A: What's the difference bettwen "背景" and "讀檔"
- Q: In "讀檔" mode, GUI called search engine for every single move. That is, the search program is terminated once it returned the move. In "背景" mode, the search engine is called in the beginning of the game, and it will be terminated only when the game is over.
- A: What if I want to compare the search engine for two different version?
- Q; You need to create another folder contains the GUI interface. Use one GUI interface to create the game room, and use the other to join the game room.