CS4330
Yi Zhu

Checkers Data Model

CheckerBoard

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

Obejct: **Player**

int **type**: 0 (computer) / 1(player 1) / 2(player 2)
string **name**
boolean **color_black**: true (pieces are black and object moves first) / false (pieces are white object moves later)
int **piece_num**: decrease when piece eaten by the other player
boolean **choose_first**: true (player can decide **color_black**) / false
time **resttime**

Method:

Choose_Color(): if **Game.pc** is false, then the player can choose color, and **Player.color_black** changes.

Choose_Piece(): click by player, test **Piece[i].can_bechose** is true, change **Piece[i].chose** to true

Choose_Destination(): click by player, change **Piece.next_location**
Get_Warning()
Get_Result()

Object: **Piece**

boolean **can_bechose**: true (the piece can be chose) / false
boolean **chose**: true (the piece is chose) / false
boolean **king:** true (the piece is king) / false
boolean **eaten**: true (the piece is eaten) / false
boolean **color**: true (black) / false
(int, int) **location**

(int, int) **next_location** : When player choose destination, it changes. Set for test part.

Method:

Move_LeftUp(): location(-1,-1)

Move_RightUp(): location(-1,+1)

Move_LeftDown(): location(+1,-1)

Move_RightDown(): location(+1,+1)

Change_Location(): In a for loop or a while loop, test every case, then combine **Move_LeftUp(), Move_RightUp(), Move_LeftDown() and Move_RightDown()**

Become_King(): king turns to be true. It can **Move_LeftDown()** and **Move_RightDown()**

Test_Destination(): test next_location. If next location is not legal, then warning shows up.

Be_Eaten(): Change eaten to true. And move this piece from the board.

Object: **Game**

boolean **pc** : true (player and computer) / false (player and player)

int **max_num**: number of lines and number of columns

int **max_piecenum**: At first, the number of pieces one player has.

time **maxtime**

int **destination_1**: number of next destinations for player1 can be chose

int **destination_2**: number of next destinations for player2 can be chose

Method:

Initialization()

Refresh()

UpdateAllDestination_1()

UpdateAllDestination_2()

ChangePlayer()

TestWin(Boolean): When (Player[i].**piece_num == 0**) or **(destination_1 ==0** or **destination_2 == 0),** then return true.

WhoWin(String): if (Player[i].piece_num == 0) then return Player[i]

if (**destination_1 ==0** or **destination_2 == 0)** then return tie

ShowWarning()

ShowResult()