

CHECKERS

CS101 AUTUMN SEMESTER (2014) COURSE PROJECT

Professor: Dr. Supratik Chakraborty

Slot 11 Group 14

Team Members:

Yogesh Kumar (Group Leader)- 140050004

Shalin Mayank- 140020099

Ravi Lakshay- 140020109

INDEX:

1. Introduction
2. Rules of the game
3. User interface requirements
4. Instructions for the players
5. Specifications of the functions used

INTRODUCTION:

This project is a game of checkers which has functionality for single (using AI) and multiplayer (2 players) modes for the players.

It has a scope of identifying the moves of the user and choosing the most aggressive counter move in turn.

User must know the rules of the game beforehand which is illustrated in this report in the later part.

RULES OF THE GAME:

There are many rules in the standard game of checkers. We have used some of them.

Each player begins the game with 12 coloured discs on their side. The checkers is same as that of a typical chess board. We have used the disc colours to be red and blue.

The board is positioned in such a way that each player has a light coloured box in the right hand corner closest to him or her.

Discs are placed on both sides of the board on alternate black squares.

Black moves first. Players are allowed to move only on black squares.

Only diagonal moves are allowed.

Soldiers (initially all disc are soldiers until they reach the other side of the board) are allowed to move in forward direction. After reaching the other side

of the board a soldier becomes a king. A king is allowed to move in both forward and backward directions.

Each player is allowed to move only one square in each move (except for the capturing move).

In a capturing move, the player needs to jump diagonally off the opponent's disc diagonally. When a piece is captured it is removed from the board. If there is chance of capture the player is should take that move. If more than one capture can be made then the player must decide to take the appropriate move keeping in mind that the player whose all discs are out of the board loses or he loses if he is not able to move any further moves.

USER INTERFACE REQUIREMENTS

The sample code is written in Code::Blocks 13.12 . The Compiler used for debugging and testing the functions is GCC Compiler.

Simple CPP is used for all the graphic requirements of the game.

The game starts with an opening screen containing multiple options to chose from. The player gets an option to chose from a single player or a multiplayer game. The game opens with a black and white board (similar to a chess board).

A 2D array has been used to represent the board. For instance, 0s are used to represent the empty positions, 1s are used to represent the discs of one player and 2s are used to represent the discs of his opponent. For instance, the following figure illustrates it:

0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
2	0	2	0	2	0	2	0
0	2	0	2	0	2	0	2
2	0	2	0	2	0	2	0

INSTRUCTIONS:

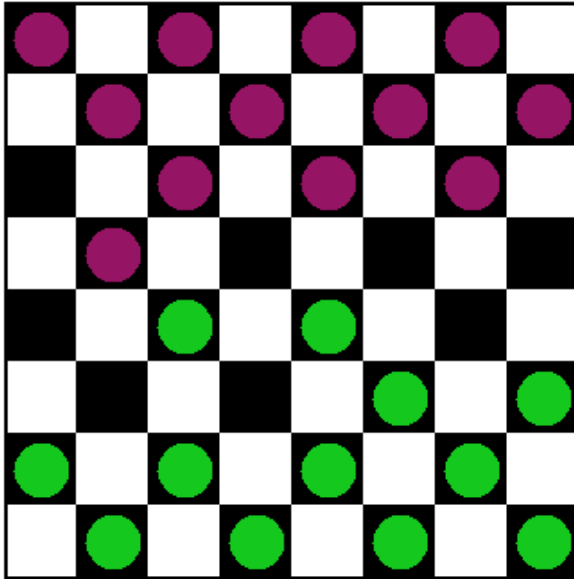
On opening the game the user encounters an opening screen showing various options for the user to chose from.

For the sake of his/her understanding, the user must click on the “help” first to know the basic rules for the game to be played.

He/she can chose to play a single player or a double player game.

Figure of the opening screen of the game is shown below:

CHECKERS



Player Vs Player

Player Vs Computer

How To Play

Credits

Exit

SPECIFICATIONS OF THE FUNCTONS USED:

1. **Playmoves1** : checks whether the coin selected by player1 is valid or not.
2. **Playmoves1f** : checks whether the position on which player1 keeps his coin is a valid position or not.
3. **Playmoves2** : checks whether the coin selected by player2 is valid or not.
4. **Playmoves2f** : checks whether the position on which player2 keeps his coin is a valid position or not.
5. **Kingmoves1** : insures that a coin of player1 after becoming king moves

in all possible way (forward as well as backward).

6. **Kingmoves2** : insures that a coin of player2 after becoming king moves in all possible way (forward as well as backward).
7. **Checkingplay1** : checks whether the next move can be a king move or not for player1 or not.
8. **Checkingplay2** : checks whether the next move can be a king move or not for player2 or not.
9. **Yesking1** : if a coin f player1 becomes the king then it increases the radius of that coin.

10. **Yesking2** : if a coin of player2 becomes the king then it increases the radius of that coin.
11. **Checkwin1** : checks if all the coins of player2 are over, then declares player1 as the winner of the game.
12. **Checkwin2** : checks if all the coins of player1 are over, then declares player2 as the winner of the game.
13. **Mainmenu** : gives the front screen of the game containing all the options for a player to choose from.
14. **Intrfcgm** : checks whether any mouse button click event has happened or not in main menu.
15. **Playervsplayer** : does all functionality for a 2 player game.

16. **Compmove** : makes all the moves possible for the comp to move in a single player game and moves the most aggressive move.
17. **Noofkills** : decides the number of kills made by any side.
18. **Noofkillsking** : checks the number of kings killed.
19. **Tieup** : if the number of coins on both sides are equal and no moves are possible then it declares a tie.
20. **Onemovecoin** : if there is no chance for a player to move an aggressive move to kill an opponent's coin then it moves a single move.
21. **Checkfunc** : it checks the validity of the computer moves.

