Checkers Coursework

Gary Murie
40210109@live.napier.ac.uk
Edinburgh Napier University - Algorithms and Data Structures (SET09117)

Keywords – Checkers, Draughts, Console, App, Application, Game, AI, Player

1 Introduction

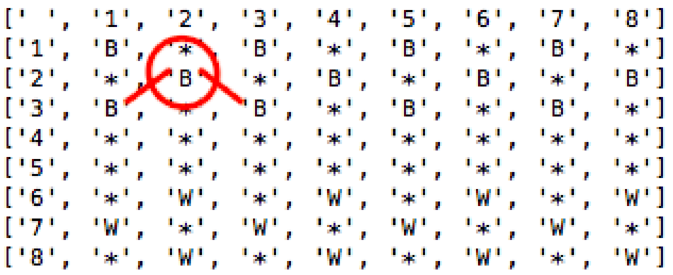
This task consisted of making a draughts application in which a user can play another user, a user can play an AI (Artificial Intelligence) and an AI can play an AI. The features to be included in this game consist of an undo feature where the user can undo moves, a redo feature where the user can redo the moves that have been undone and a replay feature so that the game that has just been played can be watched back move by move automatically. The use of Algorithms and data structures are to be implemented in practice. The program was to be written in any programming language of the developers choice, in this case Python was used.

2 Design

For the board a 2D list was used, this was to allow there to be coordinates for every piece on the board, it was an easy implementation of this, as coordinates are already in place with this type of data structure. This allows for easy selection of a counter, without much more code than the user input of the coordinates they want to use in order to play the game. The way the application was written, in terms of code layout, such that the developer had to make sure that blocks of code were in the right order, to make sure that the program would check if the move was legal before is actually made a move. The win and lose function. This part of the task was complete by looking for the counters in the board and if it found one or more then increasing an int variable, which is set to zero, by one, if it does so then the game would continue, if the int variable was still zero, this meant there was none of that counter on the board, thus the opposing team wins. This can be found in the Code Listing Section, number 2.

3 Enhancements

If there was a little more time, the GUI (graphical user interface) within the console could be updated so itâĂŹs easily readable to the user. As it is now the user has to squint their eye and get in close to the monitor and follow the counter to make sure they get the correct coordinates in order to move it. An AI was part of the specification for this task, so an implementation of even a simple AI where it picks areas at random as long as its in the parameters of the game. The Undo and Redo features were also in the specification for this task, to have had these working even is the user could undo one move, in case they move to a wrong place, or the counter they picked up was not the one they wanted to, would have made this game more user friendly. In the case of this version of the game, once the user has picked up a counter, it can no longer be dropped and have to move that counter, so if they pick up a counter, lets say at the beginning of the game and they pick it up in the second row that cannot move anywhere as there are counters in front of it and it cannot move backwards, there fore the game has to be needed and restarted in order for the game to continue. This can be seen below.



```
[' ', '1', '2', '3', '4', '5', '6', '7', '8']
['1', 'B', '*', 'B', '*', 'B', '*', 'B', '*']
['2', '*', 'B', '*', 'B', '*', 'B', '*', 'B']
['3', 'B', '*', 'B', '*', 'B', '*', 'B', '*']
['4', '*', '*', '*', '*', '*', '*', '*', '*']
['5', '*', '*', '*', '*', '*', '*', '*', '*']
['6', '*', 'W', '*', 'W', '*', 'W', '*', 'W']
['7', 'W', '*', 'W', '*', 'W', '*', 'W', '*']
['8', '*', 'W', '*', 'W', '*', 'W', '*', 'W']
```

Figure 1: **Moving Issue** - If you select this counter, you're stuck.

If time was on the developers side during this task, then the application could have been up to standards with at least the undo and redo working with a basic, random, AI.

There is an error where once you become a king, you can move diagonally but only three out of the four ways, if the user tries to move up and to the right, i.e. going from coordinates, 4,4 to 3,5 the game will crash. The counter can move all other diagonal moves and the conditions are there so it cant move vertically or horizontally, but it cannot move up and to the right, and this is the case for both teams kings. This can be seen in the Code Listing Part 1. If time was on the developers side during this task, then the application could have been up to standards with at least the undo and redo working with a basic, random, AI.

4 Critical Evaluation

When the users counter reaches the end of the board it automatically becomes a king, there is no need for input from the user, and then can move back up the board as a king.

The jumps also work well as the user input the coordinates for the counter they want to jump and without fail, the counter jumps and removes the enemy counter. Within the app also, there are conditions so the user cannot freely move a counter anywhere they please. For example, the regular counter cannot move backwards, but when it changes to a king it can them move up and down the board. However in this version of the game you input the coordinates of where the counter you want to jump is, it then checks that space for a white regular or king counter, and if there is one then jump it and take the counter, this is fine but if there is another counter on the other side of the counter you just jumped it will just be replaced with the counter you just used to jump an enemy counter. Also that the counters can only move diagonally. Meaning that the game can be played in a controlled manner where no user can go out of the game rules. They cannot move horizontally or vertically, and can only jump counters that are next to them diagonally, adding to the rules of the game. The how to play part was a good idea, as this is a console application, the average user may not be familiar with the workings/layout of a console application, the how to play part of the game showing the user how to play was a good addition to the game.

4.1 Code Listing

Examples of code used within the Application.

Listing 1: Condition

```
elif counterPlaceBYWhile == newCounterBY:
        print ("You Can Only Move Diagonally")
        continue
elif counterPlaceBXWhile == newCounterBX:
        print ("You Can Only Move Diagonally")
        continue
```

Listing 2: Win/Lose Condition

```
    for i in range (9):
        for j in range(9):
            if board[i][j] == 'W':
                whiteWin = whiteWin + 1
    for i in range (9):
        for j in range(9):
            if board[i][j] == 'O':
                whiteWin = whiteWin + 1
    if whiteWin == 0:
    print("Black Wins")
```

5 Personal Evaluation

The developer has used Python for the creation of this application, only the second time he has used it. However the other time has been using it for web development, so its been a learning curve to learn a new language as well as building an application along side. The language was fairly easy to get a grasp of but there were some speed bumps along the way. One of those was putting code in the right place so that when the program ran through it, it would check that the counter was moving to a legal move area before it actually moved. Another would be that the win and lose function did not seem to read through the board and look for the counters required, after many trial and errors, the bug was finally sourced to be that the for loop was in the wrong place and that two were needed to go through the regular counters and the king counters.

6 Conclusion

The game works on a player versus player basis, with the exception of the bug of the kings. Therefore it is a playable game as long as the user does not go up and to the right, in the near future with updates, this problem will be fixed so its a fully player versus player game.