Coursework Report

Cool Student
40202779@napier.ac.uk
Edinburgh Napier University - Module Title (SET09117)

1 Introduction

For this coursework I was required to make a game of checkers with certain features, these features were implementing all of the game's rules (obviously), an undo/redo function, a replay function, and a computer player for each colour. I successfully implemented all of these bar the undo/redo function.

2 Design

When designing this application I made sure to plan out what I wanted the program to be able to do, making notes and pseudocode where helpful. This was particularly helpful when creating the game rules as it allowed me to consolidate my thoughts, e.g. for a time I considered using an array to store the board state and I ended up "talking" myself out of it by realising that it would be difficult to differentiate between colours when kings got involved. I also thought about how I wanted the board state to be shown to the user(s) and came to the conclusion that using buttons to signify each tile on the board would be both practical and sleek looking.

To go into greater depth - I ended up using several lists to store the location of each player's pieces, removing and adding items as the game progressed (adding only to initialise and for moving pieces, i.e. remove A1 add B2). When a player clicks one of their pieces then if it is their turn, they are shown each valid move for that piece. Every time this happens, if another piece they control could capture an enemy piece, it is pointed out to them by turning the relevant pieces' backgrounds blue and sending a message saying that they must take a piece if they are able to. Once a piece is taken, the piece that took it is checked to ensure that if it can take another piece, it will prompt the user to do so (not ending the turn until they take all pieces available with that one).

Next, the replay function was implemented with the use of a queue, adding strings consisting of the Original Location of the piece being moved, a character to visually split the two values, and the Target Location of what the selected move was. Then, once the game is over, the user(s) are given the option to save the replay to a file whose name consists of two random numbers between zero and one thousand concatenated then replay.rp, this means an example file name could be "849689replay.rp". From here the user can then enter the file name in a text box in the top left and click the Load Replay button to open the replay window, this uses an algorithm separate to to one for the actual game which simply queues each action from the replay file and then executes them on a blank slate, allowing the user to step through each move made.

The computer player was fairly challenging to implement into the game rules that I had created. The "AI" first randomly chooses a piece that it controls - each legal move that piece could make is then stored into a list and then one is chosen at random to be executed. This is only valid, however assuming that there are no pieces that may be taken by the AI's pieces, if there are one or more pieces that can be taken: the AI chooses one of the pieces that it controls that can take an opposing piece at random, after which the piece is moved and the captured piece removed. As with human players, once the AI captures a piece, another check is made to detect whether the piece that just captured an opposing piece can capture another, if it is found to have another capture target then the piece is taken and the cycle continues until no more pieces can be taken by that piece.

3 Enhancements

By no means do I think that my implementation of the checkers game is perfect! I feel that some improvements that I could make to the game system itself could be to allow each player to enter their names and include those in the replay file's name as to make them more logically named. Another could be to allow the spectator of an AI vs AI game to choose between a game being played out continuously by the computer or stepping through each computer player's turn (i.e. the way in which it currently works is that when one computer player makes a move, the other instantly does theirs and then the first makes a move after that etcetera). I think that my game system's code is not very readable and could be improved upon by splitting up the main method (which spans over a thousand lines) into several separate functions, I believe that this would go a long way to improve the readability of the code.

For the replay function, the user could be allowed to choose a file path to store their replays and then could also allow the user to select a replay file from any directory, given that with the current system the file names are not intuitive. An undo and redo function would also be useful to move backward and forwards in the game, this would likely require changing the queues to simple lists instead as it would be unproductive to dequeue an action and then try to undo that action since it would not be in the queue.

There are several improvements that could be made to the computer player, for example instead of making it choose pieces and moves at random, it could have some sort of minimax algorithm to ensure that it takes the "best" move for each situation. This would of course require a lot of research on how it could be implemented on my part, but I feel that it would be worth the effort to make the game more challenging and enjoyable since as it stands, the current computer player does not make "correct" plays (even if they are legal moves).

4 Critical Evaluation

A feature that I found to be very effective is my computer player, this is because although it may not be a good player it is adept at following the game's rules and makes moves relatively quickly (quasi-instantly most of the time). Another part of it that was done well were the safeguards to prevent it from infinitely trying to select a piece that had no valid moves, this was done in the form of a list to store each previously attempted location so that the algorithm would not try it again.
What I believe was poorly executed about the computer player was the inefficient code associated with finding whether a piece could be taken, it seemed like the only option at the time but I feel that there was a better way to do so (note that this is the same for human players also).

For the replay system, it functions well and and the only real issues that I can find with it are the file names, and not having an undo and redo system (as previously mentioned).

For the game system I think that the major flaws come from within, as previously stated I had considered using an array to store the game state at all times, upon further reflection after implementing the grid of buttons I think that it would have been significantly easier to implement that or, at least, it would have meant that my code would be a lot more readable and it would have required less specific snippets of code (a major problem that the code has is that it tended to have very similar but slightly different sections of code, this could almost assuredly been resolved by making some more general functions and calling those when required but at the time I had not considered that).

5 Personal Evaluation

I think that in comparison to previous projects, where I had not written any pseudocode and had not done any forward thinking near the start of the project, this one went far more smoothly as I had set out a rough plan of what it was that I wanted to do.
An issue I faced midway through was that I rushed making the computer player before fully finishing the game rules (read before I had implemented taking pieces). Once I did fully implement the rules it was necessary to fully redo the computer player from the ground up as the functions I had in place relied on only running them a single time for each player, whereas because of the system for taking several pieces it would often require running the functions several times, making the original computer player algorithms not work as intended.

Overall I feel that I performed well, albeit I did miss a feature or two. I correctly managed my time and navigated around or through the various problems that I came across as previously stated. I learned more about C and the various preexisting objects in Visual Studio, like buttons, which enabled me to solve the initial problem in the way that I did.

6 Conclusion

Even though I wrote some fairly horrendous code, I learned a great deal from doing this project. The main point of success was with my computer player, which I worked on throughout the project, even though the original code was scrapped, near the beginning. There were several points upon which I could expand the project and given enough time and research I could eventually make a functional computer player that could pose a real "threat" to a human adversary.