Coursework Report

Steven Gibson
40270320@live.napier.ac.uk
Edinburgh Napier University - Module Title (SET09117)

Abstract

The objective of this coursework was to demonstrate my understanding of Algorithms and Data Structures and apply it to a project. My task was to implement a game of draughts.

1 Introduction

Background The Task of the this coursework was to design and implement a game of draughts that would allow the the user to play a game either player vs player or player vs computer. It also had include an Undo/Redo feature and have a game replay feature that would replay game autonomously.

2 Design

My design for the game I took a Divide and Conquer approach, I broke the game down into sub problems and solved them. I decided I would need 6 classes to begin with MainWindow.cs for the GUI, a Board class to build the board,Marker Class to make each marker, an AI Class, a History for storing games and a move class to check for valid moves.

2.1 Board

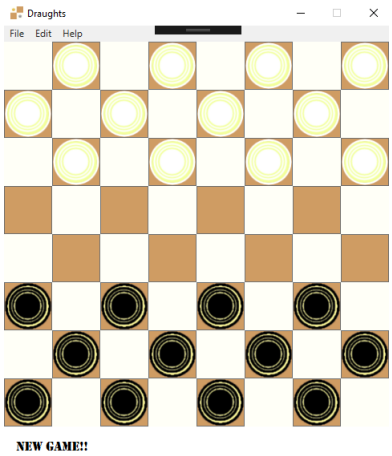

Figure 1: Board and Markers - GUI of game board with markers ready to play

I started with board which I made using a 2d array 8,8 so each square could be assigned to a number(-1 for invalid squares, 0 for empty squares, 1 White markers, 2 Black markers, 3 White kings, 4 Black kings) after the board was created I moved on to placing the markers, at this point I found a GUI was a better representation. I recreated the board using stack-panels still with the same numbering system.

2.2 Markers

I moved on placing markers on the board I used a case a statement to place them i felt this was a more effective than a series of if statements, The marker class contains 3 values the markers row and column and wither or not it has been promoted to a king. Then they get added to a stack panel.

2.3 Player vs Player

I decided the best approach was to build working game first for player vs player, then sort out an AI player.

2.4 Undo/Redo

To make my Undo/Redo Feature work I implemented four stacks. The undo feature uses Stacks after a player has preformed a move the move markers before and after position is added the a stack. if a marker was taken that markers co-ordinates re added to a taken marker stack so they can be replaced when a move is redone. The redo stack is added to after a move is undone and the retaken stack. After

2.5 AI

2.6 Game Replay

uses 1 queue and 2 stacks

3 Enhancements

Some common formatting you may need uses these commands for **Bold Text**, *Italics*, and underlined.

4 Critical Evaluation

Some common formatting you may need uses these commands for **Bold Text**, *Italics*, and underlined.

5 Persona Evaluation

Some common formatting you may need uses these commands for **Bold Text**, *Italics*, and underlined.

Here is a line followed by a double line break. This line is only one line break down from the above, Notice that latex can ignore this

We can force a break
with the break operator.