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

The design for the game a Divide and Conquer approach was taken, by braking the game down into sub problems and solving them, with this approach a class diagram was made

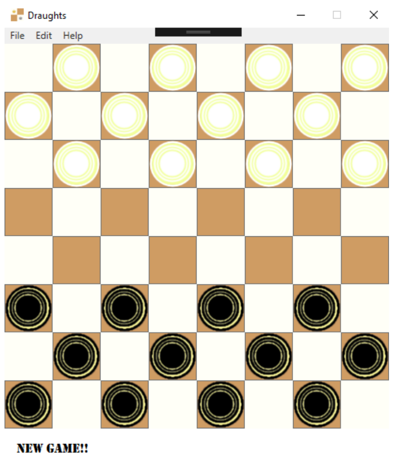

Figure 1: Class Diagram

six classes would be needed beginning 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

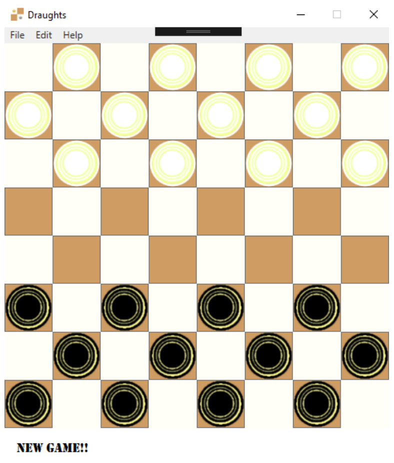

Figure 2: Board and Markers - GUI of game board with markers ready to play

To started with the board was made using a 2d int 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 for the game. I recreated the board using stack-panels still with the same numbering system.

2.2 Markers

The second stage moved on placing markers on the board a case a statement was used to determine the placement of the markers, this is a more effective way to carryout the placement than a series of if statements, The marker class contains three values: the markers row and column and wither or not it has been promoted to a king. They then got added to a stack panel. a button was created to allow users to select the marker and move it to an empty square. The buttons were named with their row and column as well as the marker it represents.

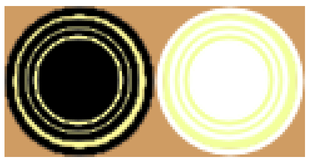

Figure 3: Markers - game markers, White and Black Markers

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. therefore

2.4 Undo/Redo

To make the 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 is added if piece that has been taken is replaced on the board.

2.5 AI

Lists and FisherâĂŞYates shuffle algorithm

2.6 Game Replay

uses 1 queue and 2 stacks

3 Enhancements

AI VS AI, Game replays saved to a file for persistence

4 Critical Evaluation

5 Persona Evaluation