Edinburgh Napier UNIVERSITY

Coursework Report

Ritvars Timermanis
40298561@napier.ac.uk
Edinburgh Napier University - Algorithms & Data Structures (SET09117)

Abstract

This report explains the inner workings and structure of the checkers game that has been developed.

Keywords – Fill, These, In, So, google, can, find, your, report

1 Introduction

The purpose of this report is to document and outline a checkers game written in Python. The game should follow all the basic rules of a normal checkers game and have several extra features which are explained below.

The first feature is undo/redo. This feature allows the player to undo the most recent move that the player has made and also to redo the undone move. For this feature to work, there is a mechanism that stores the moves made, in a sequential order.

The next feature is to play against an AI. Since this game mode is single player, an AI is implemented that can play against the human player. A unique algorithm was developed for the AI.

Finally, there also is a cancel feature which was not required, but is extremely useful to have. The cancel feature enables the player to cancel the move they are about to make if they choose to do so.

The report will also include recommendations on how the checkers game could be improved in the future.

2 Design

The program is written fully in Python, because it is an easy to use language and it is one of the most popular languages out there.

In this section of the report the following design areas will be described - architecture, algorithms and data structures.

2.1 Architecture

The game was developed with code readability and future updates in mind. Because of this a decision was made to separate the core functions of the game in separate Python files. In total there is 5 separate files.

checkers.py This is the main game module containing functions like "play()" and "main_menu()". To play the game this is the module that would be executed. From here, all other game files are executed.

board.py This module is responsible for printing the game board and current score. There is only one function in this module called "print_board()", but the reason why it is in a separate module and not in the main game module is to maintain the code readability.

piece.py This is a very important part of the program as it contains all the functions and logic related to moving the pieces around the board and performing the calculations necessary for legal piece movement.

ai.py This module contains the AI algorithm for it's decision making.

There are two functions in this file:

1. pick_piece.() - Loops through all the pieces on the board, lists all pieces which are movable and finally returns a random piece that is movable.
2. pick_destination() - Lists all the moves that can be made with the piece selected above and randomly picks a move.

g.py Module that contains all of the global variables, such as the board itself, which players turn it is and etc. The reason this module is called "g", is because it simplifies the variable calls made to the module.

2.2 User Interface

The user interface for the game is a CLI (Command Line Interface). There are several benefits to using a CLI instead of a GUI (Graphical User Interface), some of them are as follows:

- **Speed** - Can be run on lower end systems
- **System Resources** - Requires less storage space (no sprites need to be stored), RAM and processing power.
- **Development complexity** - It is significantly easier to develop than a graphical user interface (at least in this case)