Software Engineering Group Project

Maintenance Manual

|  |  |
| --- | --- |
| Authors: | Jack Book [jab153], Will Abbott [wia14], Bilal Yousufzai [biy1] |
| Config Ref: | SE.G02.MaintenceManual |
| Date: | 4th May 2023 |
| Version: | 0.4 |
| Status: | Draft |

Department of Computer Science

Aberystwyth University

Aberystwyth

Ceredigion

SY23 3DB

Copyright © Aberystwyth University 2023

CONTENTS

CONTENTS 2

1. Introduction 3

1.1 Purpose of this Document 3

1.2 Scope 3

1.3 Objectives 3

2. Program description 3

3. program structure 3

4. Algorithms 3

5. Main data areas 3

6. files 3

7. interfaces 3

8. suggestions for improvements 3

9. things to watch for while making changes 3

10. physical limitations of the proram 3

11. rebuilding and testing 3

REFERENCES 3

DOCUMENT HISTORY 4

# Introduction

## Purpose of this Document

This document is the maintenance manual for the program, it is designed to allow a program maintainer to be guided to the specific part of the program that might need changing due to updates or bug fixing.

## Scope

This document should be understood by all principal programmers and any future program maintainers.

## Objectives

# Program description

This program allows a user to learn how to play chess, whilst taking turns against an opponent who can play and understands the rules. It allows the user to understand how pieces move, as well as the specific scenarios that can lead to special moves being played. The User Interface is also intuitive enough to help the user to understand how and where each piece can move.

# program structure

Methods listed in Section 4.4 of Design Specification Document [1].

# Algorithms

Significant algorithms include, move a piece, castling, enPassant, Update GUI, Test game, Check Checker, and Checkmate Checker. Details of these algorithms can be seen in section 5.3 of Design Specification Document [1].

# Main data areas

Details of main data areas explained in section 5.4 of design specification document [1].

# files

When saving a game, the game will create a save folder containing a file per game move, in a location specified by the user.

If the user wishes to load a saved game, they select their desired game folder, and the game’s previous state will be displayed.

If the user wishes to replay a previous game, the game must have been finished, in order to be able to load it. The user selects the game folder they wish to replay, and the game will display the board at the beginning, with arrows at the top, allowing the user to scroll forward and backwards through all the .xml file game states that were made within that game.

Each singular game state is stored as a singular .xml file, holding information regarding whether or not each cell is occupied, and if so, by what.

# interfaces

Use of digital interface (GUI), buttons are used to initiate games, start games, and show menus. Text boxes are used for user(s) to input names into the program. Radio-buttons for user/s to input choice of player color, also ensures only one color can be chosen at player/s menu. Mouse event handlers are used to being able to interact with the chess board, allowing pieces to be clicked on the board and moved to a new square.

-Buttons Rules: Buttons simply work by clicking them, however, when certain buttons are clicked e.g., Player clicks “Resign” and then “Yes” all other buttons need to be disabled besides the resignation menu in the middle of the screen. This also applies for “Draw”

-Text boxes Rules: Text boxes allow all types of names, no matter the format. However, it cannot be left empty as an alias or name must be present to start a game. If either player 1 and/or player 2 the name box is left empty. A warning should be given to the player(s) regarding this issue.

-Radio buttons Rules: Radio buttons allow user(s) to decide color for player 1. Selection of color has been restricted for player 1 only as two inputs for choice would leave one selection void. One radio-button must be selected as game will not start without a color being assigned to each player. If no radio-button is selected, a warning will prompt user(s) to ensure a color has been selected.

-Mouse event handers (Click event) Rules: Event handlers are added to each square on the chess board as well as the promotion pieces (in instance of pawn promotion). However, no action is taken when the wrong color piece has been clicked when it's not their turn. This will prompt users with a warning telling them to select the correct color piece. If a square on the chess board with no piece is clicked, no action should be taken. If a square with a piece is clicked and it is according to the correct player's turn, the possible moves for clicked piece should be shown on the board. The only places a piece can be moved are on the board, on squares that are valid moves. Any clicks that are not valid moves will be dis-regarded and if an invalid click of an opponent's piece is clicked at this point, a warning will be given reminding user of whose turn it is.

# suggestions for improvements

# things to watch for while making changes

# physical limitations of the proGram

# rebuilding and testing

REFERENCES

[1] Software Engineering GP02 Project Design Specification Document

DOCUMENT HISTORY

| *Version* | *Issue No.* | *Date* | *Changes made to document* | *Changed by* |
| --- | --- | --- | --- | --- |
| 0.1 | N/A | 24-04-2023 | N/A – original version | JAB153 |
| 0.2 | N/A | 24-04-2023 | Added program description and simple file description | WIA14 |
| 0.3 | N/A | 04-05-2023 | Added introductory sections. | JAB153 |
| 0.4 | N/A | 04-05-2023 | Updated file description | WIA14 |
| 0.5 | N/A | 04-05-2023 | Added interface information | BIY1 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |